Renderização interativa de câmeras virtuais a partir da integração de múltiplas câmeras esparsas por meio de homografias e decomposições planares da cena

Jeferson Rodrigues da Silva

Dissertação apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Mestre em Ciências

Programa: Ciência da Computação
Orientador: Prof. Dr. Carlos Hitoshi Morimoto

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro do CNPq

São Paulo, Abril de 2010

Renderização interativa de câmeras virtuais a partir da integração de múltiplas câmeras esparsas por meio de homografias e decomposições planares da cena

Este exemplar corresponde à redação
final da dissertação devidamente corrigida
e defendida por Jeferson Rodrigues da Silva
e aprovada pela Comissão Julgadora.

Banca Examinadora:

- Prof. Dr. Carlos Hitoshi Morimoto (orientador) - IME-USP.
- Prof. Dr. Ronaldo Fumio Hashimoto - IME-USP.
- Prof. Dr. Marcelo Walter - UFRGS.

à minha família.

Agradecimentos

Ao meu orientador, aos meus amigos e a minha família.

Resumo

As técnicas de renderização baseadas em imagens permitem que novas visualizações de uma cena sejam geradas a partir de um conjunto de imagens, obtidas a partir de pontos de vista distintos. Pela extensão dessas técnicas para o tratamento de vídeos, podemos permitir a navegação no tempo e no espaço de uma cena obtida a partir de múltiplas câmeras.

Nesse trabalho, abordamos o problema de gerar novas visualizações fotorealistas de cenas dinâmicas, com objetos móveis independentes, a partir de vídeos obtidos de múltiplas câmeras com pontos de vista distintos. Os desafios para a solução do problema envolvem a fusão das imagens das múltiplas câmeras minimizando as diferenças de brilho e cor entre elas, a detecção e extração dos objetos móveis da cena e a renderização de novas visualizações combinando um modelo estático da cena com os modelos aproximados dos objetos móveis. Além disso, é importante que novas visualizações possam ser geradas em taxas de quadro interativas de maneira a permitir que um usuário navegue com naturalidade pela cena renderizada.

As aplicações dessas técnicas são diversas e incluem aplicações na área de entretenimento, como nas televisões digitais interativas que permitem que o usuário escolha o ponto de vista de filmes ou eventos esportivos, e em simulações para treinamento usando realidade virtual, onde é importante que se haja cenas realistas e reconstruídas a partir de cenas reais.

Apresentamos um algoritmo para a calibração das cores capaz de minimizar a diferença de cor e brilho entre as imagens obtidas a partir de câmeras que não tiveram as cores calibradas. Além disso, descrevemos um método para a renderização interativa de novas visualizações de cenas dinâmicas capaz de gerar visualizações com qualidade semelhante à dos vídeos da cena.

Abstract

Image-based rendering techniques allow the synthesis of novel scene views from a set of images of the scene, acquired from different viewpoints. By extending these techniques to make use of videos, we can allow the navigation in time and space of a scene acquired by multiple cameras.

In this work, we tackle the problem of generating novel photorealistic views of dynamic scenes, containing independent moving objects, from videos acquired by multiple cameras with different viewpoints. The challenges presented by the problem include the fusion of images from multiple cameras while minimizing the brightness and color differences between them, the detection and extraction of the moving objects and the rendering of novel views combining a static scene model with approximate models for the moving objects. It is also important to be able to generate novel views in interactive frame rates allowing a user to navigate and interact with the rendered scene.

The applications of these techniques are diverse and include applications in the entertainment field, with interactive digital televisions that allow the user to choose the viewpoint while watching movies or sports events, and in virtual-reality training simulations, where it is important to have realistic scenes reconstructed from real scenes.

We present a color calibration algorithm for minimizing the color and brightness differences between images acquired from cameras that didn't have their colors calibrated. We also describe a method for interactive novel view rendering of dynamic scenes that provides novel views with similar quality to the scene videos.

Sumário

Lista de Figuras

Lista de Algoritmos

Capítulo 1

Introdução

Considere o problema de gerar novas visualizações de uma cena a partir de uma série de imagens da mesma cena, mas de diferentes pontos de vista. Esse problema pode ser resolvido com a utilização de técnicas de renderização baseadas em imagens. Através da interpolação das várias imagens base é possível gerar novas visualizações de cenas complexas utilizando ou não modelos geométricos explícitos. [SK00]

Atualmente, com os muitos avanços atingidos com as técnicas de renderização baseadas em imagens e com os constantes avanços tecnológicos e diminuições de custo na área de hardware, é possível estender essa técnica para tratarmos cenas dinâmicas. A técnica de renderização baseada em vídeos trata o problema de utilizar múltiplos vídeos de uma mesma cena, filmados a partir de pontos de vista diferentes, para gerar novas visualizações da cena em um instante qualquer. Isso pode ser feito em tempo real permitindo que um expectador possa interagir alterando a posição da câmera enquanto assiste ao vídeo. [Mag05]

A área de pesquisa de renderização baseada em vídeos é uma área relativamente recente, bastante interdisciplinar, que tanto faz uso como contribui com avanços principalmente nas áreas de computação gráfica e visão computacional. Do lado da visão computacional está o estudo da análise das

imagens de múltiplos vídeos em cada instante para permitir, por exemplo, a reconstrução geométrica da cena ou a calibração das câmeras. Do lado da computação gráfica está o estudo de técnicas de renderização que permitam a síntese, em tempo real, de novas visualizações com base em múltiplos vídeos.

As áreas de aplicação da renderização baseada em vídeos são diversas. Na área de entretenimento podemos citar filmes ou eventos esportivos que podem ser assistidos a partir de qualquer ponto de vista ou as televisões tridimensionais que permitem que o usuário veja imagens tridimensionais sem a necessidade de acessórios adicionais. Também podemos citar as técnicas de modelagem baseada em vídeos e captura de movimento sem marcações que podem reduzir bastante o tempo que hoje é gasto em animações produzidas por computador e também na indústria de jogos. Outra aplicação possível é o uso em aplicações de monitoramento de ambientes em sistemas de segurança permitindo a visualização integrada de todas as câmeras em uma representação 3D dos ambientes monitorados. Por fim, citamos as simulações de treinamento e aplicações de realidade virtual que podem se beneficiar de cenas realistas e reconstruídas a partir de cenas reais produzidas pelas técnicas de renderização baseada em vídeos.

O problema abordado neste trabalho pode ser definido da seguinte forma: Dado um conjunto de múltiplos vídeos de uma mesma cena com pontos de vista distintos queremos gerar novas visualizações da cena, em tempo real, a partir do ponto de vista de uma câmera virtual escolhida pelo usuário. Estamos interessados particularmente na solução desse problema para cenas dinâmicas contendo objetos móveis independentes como aquelas encontradas em sistemas de segurança para monitoramento de ambientes.

O primeiro desafio para a solução desse problema consiste da fusão das imagens das múltiplas câmeras. Geralmente, essas imagens apresentam diferenças visíveis de brilho e cor devido ao uso de câmeras com configurações ou propriedades distintas. Como as novas visualizações da cena são geradas a partir da combinação das imagens das múltiplas câmeras, é necessário minimizar as diferenças

entre elas para que as novas visualizações sejam mais realistas.

Um outro desafio envolve a renderização das novas visualizações da cena combinando as imagens dos vídeos com um modelo 3D simplificado da cena. Para que seja possível gerar novas visualizações de uma cena dinâmica é necessário primeiro identificar e extrair os objetos móveis presentes em cada vídeo da cena. A extração desses objetos dos vídeos gera buracos que posteriormente precisam ser preenchidos para que as novas visualizações não apresentem artefatos nessas regiões. Finalmente, os objetos extraídos precisam ser renderizados junto ao modelo 3D da cena em suas posições correspondentes e com a perspectiva correta relativa à câmera virtual controlada pelo usuário em tempo real.

Neste trabalho utilizamos uma combinação de algumas das técnicas de renderização baseada em imagens e vídeos existentes e as estendemos com alguns toques originais para o tratamento de vídeos de cenas dinâmicas. Apresentamos um algoritmo baseado na técnica de recozimento simulado (*simulated annealing*) [KGV83] para a calibração das cores das imagens provenientes de múltiplas câmeras. O algoritmo pode ser aplicado a imagens de câmeras que não passaram por uma etapa de calibração de cores desde que as imagens apresentem regiões de sobreposição entre si. Com base nos histogramas das regiões de sobreposição de uma imagem de referência e uma imagem a ser corrigida, nosso algoritmo encontra transformações de histogramas que quando aplicadas à imagem a ser corrigida minimiza a diferença entre os histogramas das áreas de sobreposição. A imagem resultante apresenta cor e brilho bastante próximos da imagem usada como referência permitindo assim realizar suavamente a fusão das imagens das múltiplas câmeras.

Também apresentamos um método para a renderização interativa de novas visualizações de uma cena dinâmica a partir de múltiplos vídeos. Nosso método utiliza uma abordagem híbrida de imagem e geometria para a geração de novas visualizações. Por esse motivo utilizamos um modelo 3D simplificado da cena texturizado a partir de imagens da cena real. Além disso, utilizamos técnicas de segmentação e rastreamento para possibilitar a extração e posterior renderização dos objetos móveis

presentes na cena. Para a renderização das novas visualizações, utilizamos uma variação da técnica de texturização dependente da vista (*view-dependent texture mapping*) [DYB98] onde as regiões de oclusão são preenchidas pelo modelo texturizado da cena e os objetos móveis são renderizados como painéis (*billboards*) texturizados também pela técnica de texturização dependente da vista. Resultados experimentais mostram que as novas visualizações podem ser geradas em tempo real e possuem qualidade semelhante à das imagens originais. Além disso, as imagens resultantes fornecem resultados melhores nas regiões de oclusão se comparadas à tecnica original de texturização dependente da vista já que utilizamos detalhes da própria cena no preenchimento enquanto a técnica original preenche essas regiões pela interpolação das cores das áreas vizinhas.

1.1 Organização do texto

No Capítulo 2, apresentamos algumas das técnicas estudadas para o desenvolvimento do método. Apresentamos diferentes abordagens para o problema de renderização de novas visualizações a partir de imagens de múltiplas câmeras. Também mostramos algumas técnicas para a reconstrução da forma 3D de objetos a partir de múltiplas imagens. No Capitulo 3, apresentamos todas as etapas do método para a geração de novas visualizações. Na primeira etapa, apresentamos os problemas de construção dos modelos 3D da cena, a segmentação e o rastreamento dos objetos móveis e o algoritmo para a correção de cores. Na segunda etapa, descrevemos o método para a geração das novas visualizações utilizando os dados obtidos na primeira etapa. Começamos pela descrição do processo de geração das texturas de fundo da cena e da textura de objetos, descrevemos então a renderizaçàо do modelo da cena pela variação da técnica de texturização dependente da vista e finalmente descrevemos o processo de renderização dos objetos móveis usando painéis texturizados. O Capítulo 4 mostra os resultados dos testes realizados usando o método descrito neste trabalho. Apresentamos os dados utilizados nos testes, os resultados dos testes para o algoritmo de correção de

cores e também as novas visualizações resultantes da aplicação do método. Também apresentamos alguns dos problemas encontrados e possíveis soluções. Finalmente, o Capítulo 5 conclui o texto e apresenta possíveis trabalhos futuros.

Capítulo 2

Renderização baseada em vídeos

Neste capítulo apresentamos algumas das técnicas de renderização baseadas em imagens e vídeos. Essas técnicas são descritas nos levantamentos publicados por Shum e Kang em [SK00] e mais recentemente por Zhang e Chen em [ZC04] e aquelas mais relevantes ao desenvolvimento deste trabalho são descritas em maiores detalhes neste capítulo.

Muitas das técnicas de renderização baseadas em vídeos são derivações diretas das técnicas de renderização baseadas em imagens. Para cenas onde a iluminação não se altera e as câmeras são estáticas, a extensão de muitas dessas técnicas é trivial devido ao fato de que podemos aplicar as técnicas de renderização baseadas em imagens para cada instante dos vídeos das múltiplas câmeras considerando as imagens de cada câmera para aquele instante. Novas abordagens são necessárias para tratar cenas onde as câmeras se movem pois a calibração das câmeras deve ser conhecida para cada instante dos vídeos, ou também para tratar cenas dinâmicas que envolvem objetos em movimento.

2.1 Renderização baseada em imagens

As técnicas de renderização baseadas em imagens têm como objetivo a renderização interativa de novas visualizações de cenas ou objetos a partir de imagens da cena ou dos objetos obtidas a partir de pontos de vista distintos sem que haja a necessidade de obter as informações da geometria da cena.

Formalmente, definimos a renderização baseada em imagens como o processo de amostragem e renderização da função plenóptica, proposta por Adelson e Bergen em [AB91]. A função plenóptica é uma função com 7 dimensões que modela a aparência 3D de uma cena pela distribuição dos raios de luz em qualquer posição do espaço (x, y, z), para todas as direções (α, β), sobre todos os comprimentos de onda (λ) e em qualquer instante (t). Uma vez que a função plenóptica $l(x, y, z, \alpha, \beta, \lambda, t)$ de uma cena 3D é conhecida, podemos renderizar novas visualizações da cena a partir de qualquer ponto de vista. Na prática não é possível fazer a aquisição da função plenóptica de uma cena 3D pois isto exigiria a obtenção de uma quantidade impraticável de imagens da cena de modo a recuperar todos os dados necessários.

Por esse motivo, utilizamos na prática o campo de luz (*light field* ou *lumigraph*) da cena que é obtido pela amostragem e quantização da função plenóptica. Os trabalhos desenvolvidos por Levoy e Hanrahan [LH96] e por Gortler *et al.*[GGSC96] demonstram como obter o campo de luz, reduzindo a dimensionalidade da função plenóptica, e como renderizar novas visualizações a partir dele.

2.2 Abordagem híbrida baseada em imagem e geometria

O trabalho realizado por Debevec *et al.*[DTM96] compreende um conjunto de técnicas para a construção de modelos geométricos e renderização fotorealista de cenas arquitetônicas a partir de fotografias. A técnica de modelagem fotogramétrica (*photogrammetric modelling*) permite que um usuário crie um modelo simples com base em imagens através de correspondências de arestas entre o modelo

e as imagens, e restrições adicionais de posicionamento dos elementos que compõem o modelo. A técnica de texturização dependente da vista (*view-dependent texture mapping*) é utilizada na renderização de novas cenas para selecionar como as imagens serão compostas para formar a textura que será aplicada no modelo. Finalmente, a técnica de estéreo baseado em modelo (*model-based stereo*) é utilizada para recuperar detalhes adicionais da cena que não puderam ser recuperados pelo uso da primeira técnica.

Para facilitar o processo de reconstrução, é desejável que as câmeras utilizadas na obtenção das imagens estejam calibradas [HZ00], isto é, as coordenadas e direções dos pontos das imagens em relação aos centros das câmeras devem ser conhecidas. Para a calibração das câmeras, o método de Zhang [Zha00] utiliza objetos especiais com marcações em forma de linhas retas e paralelas que permitem determinar os parâmetros das câmeras. Para o caso de imagens obtidas de câmeras não calibradas, é possível recuperar os parâmetros das câmeras a partir das próprias imagens fazendo uso das linhas retas e dos pontos de fuga de linhas paralelas como no trabalho desenvolvido por Wang e Tsai [WT91].

A seguir, descrevemos em mais detalhes as técnicas citadas acima.

2.2.1 Modelagem fotogramétrica

O processo de modelagem fotogramétrica é um processo que envolve a criação incremental de um modelo geométrico, o registro das correspondências entre as arestas do modelo e das imagens, e finalmente a reconstrução do modelo com base nas correspondências e nas restrições do modelo.

O modelo geométrico é composto por um conjunto de primitivas geométricas parametrizadas como caixas, prismas e superfícies de revolução. Estas primitivas são relacionadas através de um modelo hierárquico permitindo que a posição dos elementos descendentes seja definida em relação a um elemento pai. Por exemplo, um prisma representando o telhado de uma casa pode ter a sua coordenada y mínima restrita à coordenada y máxima de uma caixa representando a estrutura da

casa. Cada uma das coordenadas x, y e z pode ou não ser restrita individualmente assim como as rotações ao redor dos eixos x, y e z. A escolha dessa forma de representação se deve ao número reduzido de parâmetros utilizados para a representação do modelo o que torna mais viável a aplicação do algoritmo de reconstrução utilizado posteriormente.

O registro das correspondências também deve ser feito manualmente pelo usuário. O processo consiste em marcar algumas arestas nas imagens da cena e correspondê-las a arestas das primitivas geométricas do modelo. Não é preciso marcar todas as arestas presentes na cena já que um número pequeno de arestas pode ser suficiente para restringir as posições possíveis dos elementos do modelo. De posse das correspondências é possível aplicar o algoritmo de reconstrução automática que ajusta as arestas do modelo às arestas das imagens de forma a minimizar a diferença entre elas satisfazendo as restrições de posicionamento impostas na criação do modelo. Além disso, as informações fornecidas pelo usuário na forma do modelo e das correspondências também permitem a localização da posição da câmera para cada imagem usada.

O algoritmo de reconstrução é composto de três etapas. As duas primeiras etapas são usadas para estimar a rotação e a translação da câmera em cada imagem e são necessárias para evitar situações de mínimos locais na terceira etapa, permitindo uma convergência rápida para a solução ótima. A terceira etapa consiste na otimização não-linear dos parâmetros do modelo sobre todo o espaço de parâmetros para obter a melhor reconstrução possível da cena.

2.2.2 Texturização dependente da vista

Para que seja possível renderizar novas visualizações da cena de maneira fotorealista, é desejável que as texturas utilizadas no modelo sejam obtidas a partir das próprias imagens da cena. Para a renderização de uma cena a partir de um novo ponto de vista, é possível recortar partes de cada imagem e utilizá-las como texturas para polígonos individuais do modelo mas essa abordagem geraria uma cena bastante artificial já que algumas propriedades da cena como sombras, reflexos ou pequenas

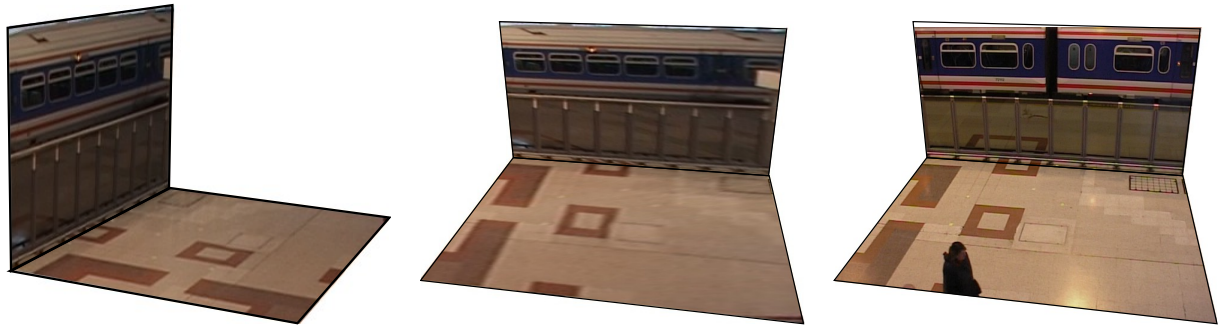
variações de relevo variam de acordo com o ponto de vista. Ao escolher uma das imagens para utilizar como textura, as propriedades da cena como as citadas acima serão exibidas de forma incorreta para pontos de vistas diferentes do original. A técnica de texturização dependente da vista [DYB98] tem como objetivo reduzir esse problema utilizando uma combinação das várias imagens como textura para o modelo.

O processo de texturização do modelo utilizando as imagens da cena é feito usando-se a texturização projetada. Conhecendo-se a posição da câmera em cada imagem é possível texturizar o modelo projetando as próprias imagens sobre o modelo a partir dessas posições conhecidas.

Com essa abordagem, cada pixel da imagem da cena pode ter sido texturizado pela projeção de 0, 1 ou mais imagens. Para os pixels que foram texturizados pela projeção de mais de 1 imagem é necessário decidir qual a melhor combinação de imagens a ser utilizada para determinar a cor final do pixel. A técnica de texturização dependente da vista determina que a imagem a ser utilizada deve ser aquela cujo ângulo de visão seja o mais próximo do ângulo de visão da nova visualização da cena. Para melhorar a qualidade da renderização é possível misturar as cores provenientes das múltiplas imagens que são projetadas em um mesmo pixel atribuindo como pesos a proximidade ao ângulo de visão da nova visualização.

Para os pixels que não foram texturizados pela projeção de nenhuma imagem, isto é, para aqueles pixels em que não há informação disponível de textura devido a obstrução, é necessário que estes sejam preenchidos de forma a melhorar a qualidade final da cena renderizada. O preenchimento pode ser feito utilizando a média das cores dos pixels vizinhos dos pixels que não foram texturizados.

Do ponto de vista de desempenho, esta técnica pode ser bastante custosa computacionalmente já que é necessário determinar as melhores imagens a serem utilizadas como textura para cada pixel da nova imagem. Apesar disso, é possível calcular esses resultados de forma aproximada e hoje já é possível realizar esses cálculos diretamente em hardware utilizando as placas aceleradoras de vídeo



(a) (b) (c)

Figura 2.1: Texturização dependente do ponto de vista. **(a)** Uma renderização do modelo 3D de uma cena. **(b)** Renderização da mesma cena a partir de um ponto de vista diferente. Podemos notar que as texturas usadas em **(a)** já não são mais adequadas para o novo ponto de vista e geram erros de perspectiva na nova visualização. **(c)** Renderização do novo ponto de vista usando texturas de uma câmera com ponto de vista mais próximo usando texturização dependente da vista.

mais recentes [Mag05].

2.2.3 Estéreo baseado em modelo

O modelo obtido através da modelagem fotogramétrica é apenas um modelo simplificado da geometria real da cena pois alguns detalhes mais sutis são representados pelo uso de texturas como é o caso quando existem pequenas variações de relevo nas superfícies da cena. A técnica de estéreo baseado em modelo [DTM96] é utilizada para gerar renderizações que representem melhor esses detalhes.

Os métodos de estéreo tradicionais procuram determinar a profundidade dos pixels da cena com base na triangulação dos pontos entre as várias imagens disponíveis. A técnica de estéreo baseado em modelo recupera as informações da cena comparando as visualizações originais das imagens com as visualizações aproximadas obtidas utilizando-se as técnicas de modelagem fotogramétrica e a texturização dependente da vista. As diferenças entre as visualizações são utilizadas para calcular as informações de profundidade para cada pixel das imagens.

As informações de profundidade podem então ser utilizadas para gerar renderizações através de

outras técnicas de renderização baseadas em imagens como a interpolação de imagens. Além disso, é possível combinar essas técnicas com a técnica de texturização dependente da vista para gerar visualizações mais realistas. Com o hardware disponível atualmente em computadores pessoais, já é possível aplicar essa combinação de técnicas de forma a gerar renderizações em tempo real.

2.3 Realidade virtualizada

O trabalho desenvolvido por Kanade e Rander [KRN97] descreve um método para visualização de vídeos a partir de pontos de vista arbitrários utilizando vários vídeos sincronizados e gravados a partir de pontos de vista distintos. Com a recuperação do mapa de profundidade dos pixels dos vídeos, é possível gerar visualizações sintéticas com qualidade aceitável e em tempo real.

2.3.1 Aquisição das imagens

A aquisição das imagens é realizada utilizando-se diversas câmeras não muito espaçadas entre si mas distribuídas de maneira a cobrir a cena a partir de vários ângulos de visão. O processo de sincronização dos vídeos pode ser feito em tempo real com o hardware adequado mas também pode ser feito posteriormente desde que o vídeo contenha informações suficientes para a realização da sincronização *offline*.

Nesta etapa também é necessário obter os parâmetros de calibração para as câmeras utilizadas. Existem vários algoritmos [Zha00] [Tsa86] que podem ser utilizados para isso e normalmente é necessário usar marcações na própria cena ou objetos específicos que permitam determinar os parâmetros da câmera.

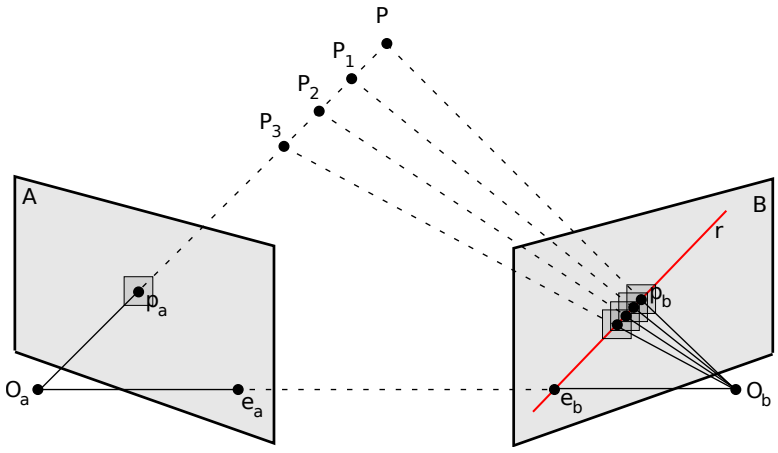


Figura 2.2: **Restrição epipolar** - O_a e O_b representam os pontos focais das câmeras, e_a e e_b representam os pontos epipolares e os pontos P, P_1, P_2 e P_3 representam possíveis posições do ponto p_a na cena. A restrição epipolar determina que o ponto p_b correspondente ao ponto p_a deve estar posicionado na reta r. [HZ00]

2.3.2 Recuperação do mapa de profundidade das imagens

A recuperação do mapa de profundidade das imagens é feito através da correspondência dos pontos entre as várias imagens. Para isso podemos comparar uma pequena janela de pixels de uma imagem A com as janelas correspondentes em uma imagem B para determinar a correspondência entre os pixels calculando, por exemplo, a soma dos quadrados das diferenças dos pixels contidos em cada janela. As possibilidades de janelas correspondentes em B podem ser reduzidas pela geometria epipolar que diz que um ponto p da imagem A está restrito a uma reta na imagem B [HZ00]. Com isso, basta comparar a janela da imagem A com todas as janelas possíveis na reta correspondente na imagem B e escolher aquele pixel para o qual o resultado da aplicação da janela mais se assemelha com o resultado da janela aplicada na imagem A [KRN97] como ilustrado na Figura 2.2.

2.3.3 Renderização por malha de polígonos

Utilizando-se os parâmetros das câmeras e as informações de profundidade dos pixels, é possível calcular as coordenadas (x, y e z) de cada ponto das imagens. As coordenadas de cada ponto permitem renderizar novas visualizações da cena usando-se uma malha de polígonos que pode ser renderizada em tempo real pelo hardware disponível atualmente.

Para a criação da malha, primeiro é necessário determinar qual imagem possui o ponto de vista mais próximo do ponto de vista desejado para a nova visualização. A malha de polígonos é composta por triângulos encaixados de modo que cada vértice da malha corresponde a um ponto da imagem selecionada como referência. Como a coordenada de cada ponto é conhecida, a posição de cada vértice da malha pode ser ajustado de acordo com a posição da câmera.

Além de utilizar uma imagem de referência, também é necessário utilizar uma imagem de apoio com ponto de vista diferente para obter informações de possíveis pixels que não puderam ser observados na imagem de referência. Mesmo assim, possivelmente alguns pixels podem não terem sido observados em ambas as imagens gerando regiões de oclusão que não podem ser preenchidas na nova visualização.

2.4 Escultura do espaço

No trabalho desenvolvido por Seitz e Kutulakos [SK99], os autores descrevem um método para a reconstrução de cenas com níveis de complexidade arbitrários a partir de um número qualquer de imagens da cena obtidas de pontos de vista diferentes. Ao contrário dos outros métodos já citados, este método procura ser o mais geral possível sem fazer uso de nenhuma informação a priori sobre a geometria da cena. O trabalho também apresenta um algoritmo de escultura do espaço (*space carving*) capaz de recuperar a forma 3D da cena a partir de múltiplas imagens desde que as posições e orientações das câmeras sejam conhecidas.

2.4.1 Restrições na figura e foto-consistência

O método de escultura do espaço define o problema de recuperação de forma 3D da cena como um problema de satisfação de restrições. Cada imagem da cena dá origem a restrições na forma da cena que são satisfeitas por todas as cenas cuja projeção resulta nessa mesma imagem. A união das cenas que satisfazem essas restrições é denominada forma foto-consistente.

As restrições são representadas na forma de uma função de radiância que determina, para cada ponto p da superfície da cena $\mathcal{V}$, a cor da luz refletida ao longo de todos os raios que passam por p. Cada imagem impõe restrições na função de radiância para um conjunto de pontos visíveis da superfície da cena de modo que somente as formas que são projetadas de forma exata sobre as imagens de referência satisfazem essas restrições como ilustrado na Figura 2.3. A união das cenas que satisfazem todas essas restrições constituem a forma maximal foto-consistente.

Uma cena $\mathcal{V}$ juntamente com a funcão de radiância para cada ponto da superfície de $\mathcal{V}$ compõem a descrição da cena e esta descrição é suficiente para reconstruir a cena a partir de um ponto de vista arbitrário.

2.4.2 Algoritmo de escultura do espaço

O algoritmo de escultura do espaço [SK99] calcula a forma 3D da cena "esculpindo" uma forma 3D inicial que contém a forma 3D da cena. Para cada ponto da superfície, o algoritmo verifica se o ponto satisfaz as restrições impostas pelas imagens da cena. Se o ponto não satisfaz alguma restrição, então ele não é foto-consistente e portanto ele pode ser removido da forma em progresso. O algoritmo é repetido até que todos os pontos sejam foto-consistentes.

A forma final obtida pode ser então renderizada utilizando-se técnicas de renderização volumétricas usando pequenos elementos volumétricos denominados voxels que são coloridos pela função de radiância para cada ponto.

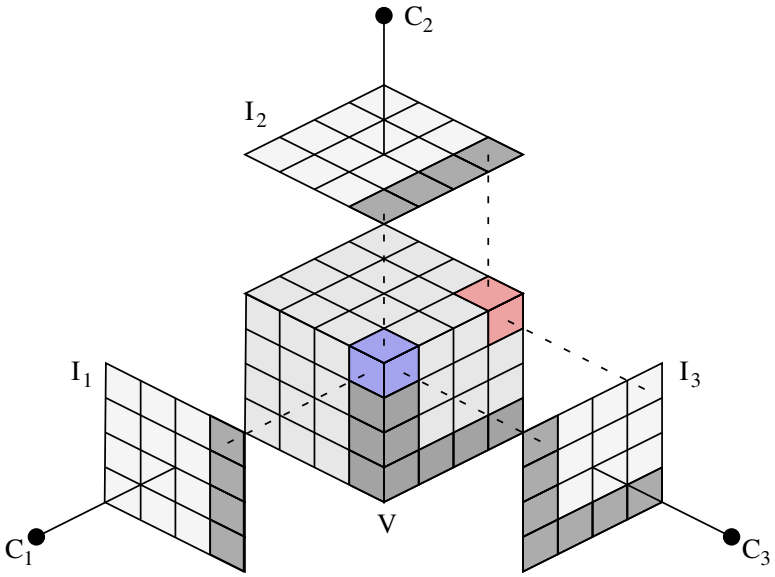


Figura 2.3: V representa o volume inicial a ser esculpido, I_k representa a imagem correspondente à câmera C_k para $k = 1, 2, 3$. O volume marcado em cinza escuro determina a forma real do objeto a ser esculpido. O volume marcado em azul é foto-consistente com as imagens I_1, I_2 e I_3 e deve permanecer no volume V. O volume marcado em vermelho não é foto-consistente já que ele não satisfaz a restrição imposta pela imagem I_3 e portanto deve ser removido de V.

2.5 Fecho visual

O fecho visual (*visual hull*) é uma entidade geométrica descrita por Laurentini [Lau94] que pode ser obtida a partir das silhuetas de objetos e que pode ser utilizada como uma aproximação das formas 3D reais dos objetos em questão.

Para um dado objeto $\mathcal{O}$, o fecho visual pode ser definido como a forma geométrica maximal pela qual o objeto $\mathcal{O}$ pode ser substituído sem que nenhuma de suas silhuetas seja alterada. Isso também equivale a dizer que o fecho visual é a melhor aproximação do objeto $\mathcal{O}$ que pode ser obtida pela abordagem de intersecção dos volumes determinados pelas posições das câmeras e as silhuetas correspondentes como ilustrado na Figura 2.4.

O fecho visual é frequentemente usado como uma aproximação para a geometria real de objetos reconstruídos pois é possível obter resultados satisfatórios com poucas câmeras bastante espaçadas entre si. O mesmo não é possível com os métodos de profundidade a partir de estéreo já que câmeras bastante espaçadas prejudicam a identificação das correspondências entre os pontos de imagens distintas.

Atualmente existem diversos algoritmos para o cálculo do fecho visual utilizando abordagens diferentes e que já podem ser utilizados em tempo real utilizando-se o hardware adequado. O trabalho desenvolvido por Theobalt *et al.*[TLMS03] apresenta um algoritmo de cálculo do fecho visual volumétrico através da discretização do espaço em voxels que são testados em relação à consistência com as silhuetas das imagens fornecidas. Voxels que não são consistentes são removidos e os voxels restantes no final do algoritmo representam uma aproximação do fecho visual da cena. Em outra abordagem, Li *et al.*[LMS03] apresenta um algoritmo de cálculo do fecho visual poliédrico utilizando a abordagem de intersecção dos cones determinados pelas posições das câmeras e as silhuetas correspondentes. Ambas abordagens podem ser implementadas diretamente em hardware e fornecem resultados em taxas de quadros interativas.

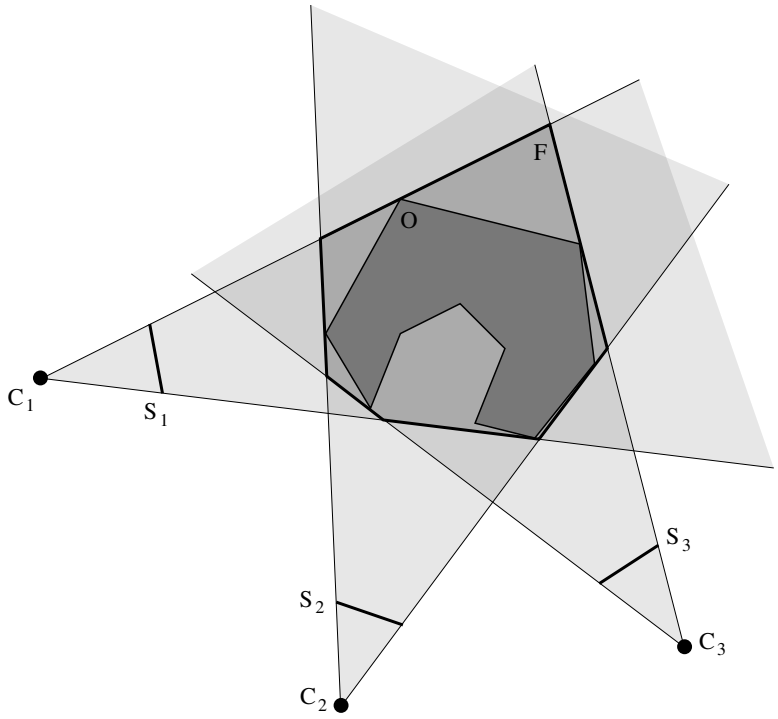


Figura 2.4: Para um objeto O observado pelas câmeras C_1, C_2 e C_3, o fecho visual F é dado pela intersecção dos cones determinados pela posição de cada câmera C_k e a silhueta S_k do objeto O correspondente.

2.6 Trabalhos correlatos

O trabalho desenvolvido por Saito *et al.*[SBK+99] é capaz de renderizar novas visualizações de cenas dinâmicas utilizando um grande número de câmeras posicionadas próximas umas das outras. O processo inicia-se pela estimação da profundidade dos pixels da cena nas sequências de imagens obtidas de cada câmera pela técnica de estéreo com múltiplas linhas de base (multiple-baseline stereo) proposta por Okutomi e Kanade em [OK93]. Um modelo volumétrico da cena é construído em seguida pela fusão de todos os dados de profundidade das sequências de imagens. Este modelo é usado para calcular a correspondência entre os pontos de câmeras distintas permitindo que até mesmo as regiões de oclusão possam ser correspondidas e renderizadas em novas visualizações. A geração de novas visualizações é feita pela interpolação das imagens de duas câmeras adjacentes o que restringe o posicionamento da câmera virtual.

Outra abordagem para a renderização de novas visualizações de cenas dinâmicas é apresentada por Moezzi *et al.*[MKKJ96]. Nesta abordagem um modelo 3D estático da cena é combinado com modelos 3D dos objetos móveis para compor as novas visualizações de uma cena dinâmica em tempo real a partir de um número pequeno de câmeras. O sistema é dividido em 3 etapas que são realizadas em tempo real. A primeira etapa consiste da detecção e rastreamento dos objetos móveis da cena usando subtração de fundo da cena e os dados de calibração das câmeras. A segunda etapa consiste da construção dos modelos dos objetos móveis pela discretização do volume 3D da cena como na técnica de escultura de espaço já apresentada. Os modelos volumétricos finais dos objetos são aproximados por malhas de polígonos que são renderizadas junto ao modelo 3D estático da cena usando texturização projetiva como na técnica de texturização dependente da vista. Uma possível desvantagem desse método se deve ao método utilizado para o rastreamento dos objetos móveis que pode não ser robusto para cenas envolvendo uma grande quantidade de objetos móveis.

O trabalho desenvolvido por Ohta *et al.*[OKK+07] apresenta um sistema capaz de produzir e

transmitir cenas dinâmicas em 3D similar ao método descrito neste texto mas que pode ser aplicado a eventos com transmissões ao vivo já que todas as suas etapas são realizadas em tempo real. Essa abordagem combina um modelo 3D estático da cena com modelos em 3D para a representação dos objetos móveis. A detecção e rastreamento dos objetos móveis é realizada utilizando-se conhecimento a priori do evento em questão, por exemplo, o número de objetos móveis esperados em um jogo de futebol, e as imagens de uma câmera posicionada especificamente para capturar toda a área de interesse onde se encontram os objetos móveis. Essas imagens são segmentadas ajustando um limiar de modo a obter um número de regiões equivalente ao número de objetos móveis esperados na cena. Cada região deve representar um objeto móvel e sua posição é dada por uma transformação homográfica que converte as coordenadas de cada região para as coordenadas reais de cada objeto na cena. Em seguida, são construídas texturas para cada objeto que são obtidas dos próprios vídeos da cena pelo recorte das regiões retangulares que representam cada objeto. Finalmente, os objetos são renderizados na cena utilizando-se painéis texturizados de acordo com o ponto de vista escolhido. Apesar de produzir novas visualizações em tempo real para eventos ao vivo, a aplicação desse sistema se limita a ambientes controlados além de exigir uma grande quantidade de câmeras já que a transição entre as texturas das múltiplas câmeras é feita diretamente sem nenhuma suavização como na técnica de texturização dependente da vista. Além disso, a geração das texturas dos objetos também exige imagens de várias câmeras para que seja possível resolver conflitos de oclusão entre os objetos da cena de modo a gerar as texturas corretamente.

Na área de interatividade, o trabalho apresentado por Jain e Wakimoto em [JW95] demonstra algumas das possibilidades dos vídeos interativos com múltiplas perspectivas (multiple perspective interactive video). O protótipo apresentado neste trabalho permite que o usuário busque eventos específicos e obtenha cenas dos vídeos que melhor representam os critérios especificados pelo usuário. No contexto de um evento esportivo, podemos por exemplo, buscar todos os lances nos quais um jogador específico teve participação ou então as câmeras que possuem as melhores imagens de um jogador

em um determinado instante. Uma outra funcionalidade pode permitir que o usuário identifique e obtenha informações sobre um jogador apenas posicionando um cursor sobre o jogador escolhido. O desenvolvimento de um sistema completo para a realização de vídeos com múltiplas perspectivas envolve a captura dos vídeos da cena, a extração e o processamento dos objetos da cena e dos eventos de interação entre os objetos, a geração de um banco de dados que permita a realização das buscas por estes objetos e eventos, e finalmente a renderização dos resultados das buscas.

Em nosso trabalho, estamos interessados no tratamento de cenas dinâmicas capturadas com um número pequeno de câmeras bastante espaçadas entre si. Por esse motivo não foi possível utilizar o método de realidade virtualizada apresentado por Kanade [KRN97]. Escolhemos então estender a abordagem híbrida de imagem e geometria de Debevec *et al.*[DTM96]. Um modelo 3D texturizado da cena é construído e a técnica de texturização dependente da vista é utilizada para projetar os vídeos da cena sobre o modelo.

Para a renderização dos objetos móveis, utilizamos uma abordagem similar à apresentada no trabalho de Ohta [OKK+07], isto é, representamos cada objeto por um painel texturizado 3D que é posicionado no modelo 3D da cena. Em nossa abordagem, cada um desses painéis é texturizado com os vídeos da própria cena utilizando a técnica de texturização dependente da vista, que foi modificada para que fosse possível calcular os pesos das câmeras em tempo real. Uma alternativa para a representação dos objetos móveis seria o uso da técnica do fecho visual já apresentada. Para isso, os modelos 3D seriam construídos em uma etapa de pré-processamento pois a construção dos fechos visuais é bastante custosa computacionalmente.

Adicionalmente, desenvolvemos um algoritmo para a calibração das cores das câmeras para que fosse possível aplicar o método a vídeos de cenas cujas câmeras não foram calibradas quanto às cores durante a aquisição.

Capítulo 3

Renderização interativa de câmeras virtuais

As técnicas de renderização baseadas em imagens geralmente exigem menos cuidados que as técnicas de renderização baseadas em vídeos. As imagens de uma cena estática podem ser adquiridas utilizando-se uma única câmera que pode ser posicionada várias vezes de modo a adquirir imagens de múltiplos pontos de vista. Para tratarmos cenas dinâmicas, essa abordagem já não é mais possível pois é necessário que haja imagens de pontos de vista distintos para cada instante da cena em questão. Por esse motivo, é necessário utilizar múltiplas câmeras para adquirir a cena.

O uso de múltiplas câmeras origina uma série de problemas que devem ser tratados para que seja possível aplicar com sucesso as técnicas de renderização baseadas em vídeo. Dentre esses problemas estão a sincronização e calibração das câmeras, a fusão de imagens obtidas por câmeras diferentes que podem apresentar brilho e cor bastante diferentes entre si e o problema de detecção e extração dos objetos móveis para que seja possível reconstruí-los em novas visualizações da cena. Além disso, para aplicarmos as técnicas de renderização baseadas em vídeos escolhidas para o problema, precisamos

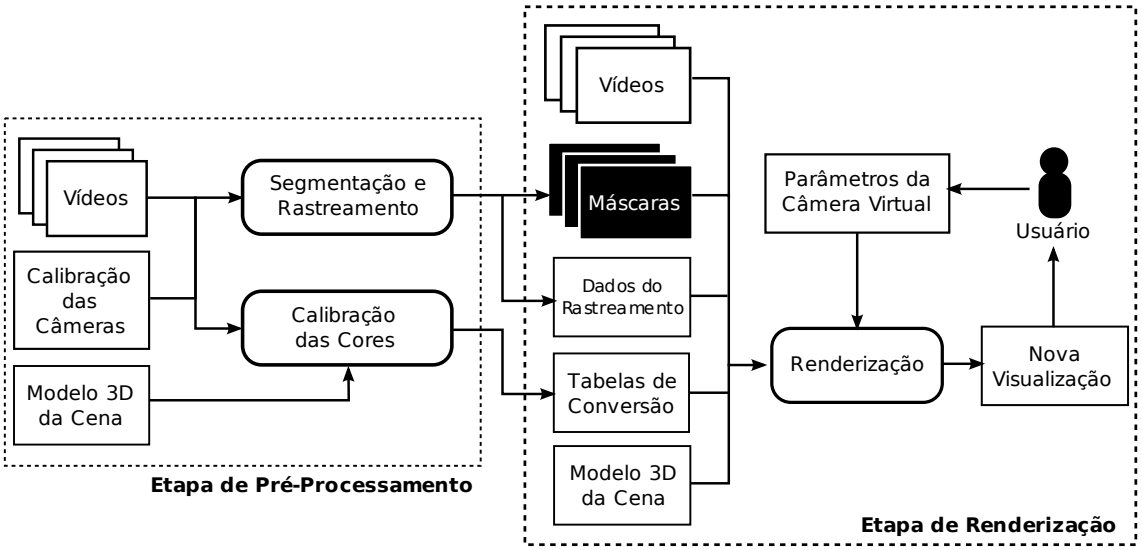


Figura 3.1: Diagrama de blocos do método

também construir um modelo 3D simplificado da cena.

3.1 Visão geral do método

Nosso método de renderização de novas visualizações de uma cena a partir de múltiplos vídeos é constituído de duas etapas. O diagrama da Figura 3.1 mostra a relação entre as duas etapas e seus componentes.

A primeira etapa é denominada etapa de pré-processamento. Nessa etapa estamos interessados em gerar todos os dados necessários para que seja possível renderizar novas visualizações da cena em tempo real na etapa seguinte.

Começamos a etapa de pré-processamento construindo um modelo 3D simplificado da cena para que possamos utilizar a técnica escolhida para a etapa de renderização que se baseia na abordagem híbrida baseada em imagem e geometria. O modelo simplificado deve ser texturizado preferencial-

mente utilizando-se imagens obtidas da cena real para que as novas visualizações sejam mais realistas e para que possamos combiná-lo melhor com os vídeos da cena.

O próximo passo consiste da calibração das câmeras e da sincronização dos vídeos. Através da calibração das câmeras obtemos os dados necessários para a utilização do método como a posição das câmeras e as direções para quais elas apontam. Já a sincronização dos vídeos é necessária para que possamos relacionar um mesmo instante em cada um dos vídeos. Dessa forma, podemos combinar as imagens dos múltiplos vídeos em um determinado instante para gerar novas visualizações da cena e dos objetos móveis para um novo ponto de vista escolhido pelo usuário.

Em seguida, realizamos a segmentação e o rastreamento dos objetos móveis em cada um dos vídeos. A segmentação irá nos fornecer uma máscara que permite separar o fundo da cena dos objetos que se movem. Isso é necessário pois renderizamos o fundo da cena e os objetos móveis em momentos diferentes usando técnicas diferentes na etapa de renderização. O rastreamento dos objetos nos permite identificar cada objeto móvel a partir de um rótulo e também conhecer suas posições reais na cena. A identificação dos objetos permite que o usuário possa interagir com os objetos individualmente destacando aqueles julgados mais interessantes e ocultando aqueles menos importantes, por exemplo. A posição de cada objeto na cena real possibilita o posicionamento correto dos objetos no modelo 3D da cena durante a etapa de renderização.

O último passo envolve a calibração das cores dos vídeos. Nesse passo queremos encontrar transformações do espaço de cores de cada um dos vídeos que minimizem a diferença de brilho e cores presentes entre eles que existem devido ao uso de câmeras com propriedades distintas. A calibração de cores é essencial para a etapa de renderização pois nela precisamos mesclar as imagens das múltiplas câmeras com o modelo da cena para gerar novas visualizações.

Na segunda etapa, denominada etapa de renderização, utilizamos todos os dados obtidos na etapa anterior em conjunto com os dados fornecidos em tempo real pelo usuário para gerar novas visualizações da cena a partir de qualquer ponto de vista.

Começamos renderizando o modelo 3D simplificado a partir do ponto de vista escolhido pelo usuário. Em seguida, aplicamos as transformações de cores em cada um dos vídeos para o instante atual e separamos as imagens do fundo da cena das imagens dos objetos móveis usando as máscaras obtidas na segmentação. Projetamos então sobre o modelo 3D uma combinação das imagens do fundo da cena com pontos de vista mais próximos ao ponto de vista escolhido pelo usuário. Depois desses passos, temos uma nova visualização da cena sem a presença dos objetos móveis.

No passo seguinte, os objetos móveis são renderizados sobre o modelo na forma de painéis texturizados com uma combinação das imagens dos vídeos com pontos de vista mais próximos àquele escolhido pelo usuário. Esses painéis são posicionados de acordo com as posições reais obtidas durante o rastreamento.

Nas seções seguintes abordaremos cada um dos problemas envolvidos nas duas etapas do método detalhando e mostrando as soluções adotadas para o desenvolvimento do método.

3.2 Pré-processamento

3.2.1 Sincronização dos vídeos e calibração das câmeras

Muitas das técnicas de renderização baseadas em imagens necessitam que as câmeras estejam calibradas, isto é, que seus parâmetros extrínsecos e intrínsecos sejam conhecidos. Os métodos mais conhecidos para calibração de câmeras o fazem relacionando pontos da cena real, que possuem coordenadas conhecidas, com pontos correspondentes na imagem da cena. A escolha dos pontos a serem relacionados pode ser feita manualmente ou automaticamente através do uso de padrões de calibração que devem ser posicionados na cena de modo que eles possam ser observados por todas as câmeras. Neste trabalho utilizamos o método de calibração de Zhang [Zha00].

A sincronização dos vídeos normalmente é feita utilizando-se um sistema distribuído de PCs que são responsáveis por iniciar a gravação simultaneamente em todas as câmeras e também por

processar e armazenar em disco os vídeos de cada câmera. Uma outra possível abordagem utilizada para sincronizar as sequências de vídeo deste trabalho é a sincronização baseada em eventos, onde algum evento observado por todas as câmeras é utilizado como ponto de referência para sincronizar vídeos que foram armazenados sem informação de sincronização.

3.2.2 Construção do modelo

No capítulo 2, citamos algumas das técnicas utilizadas para se obter informações da geometria da cena a partir de múltiplas imagens. A recuperação da geometria de cena depende da capacidade de se obter a profundidade dos pixels das imagens mas esse é um problema difícil devido à presença de ambiguidades principalmente quando o número de imagens é pequeno. A técnica de renderização utilizada neste trabalho baseia-se na técnica de Debevec *et al.*[DTM96] que faz uso de um modelo 3D da cena combinado com as imagens das múltiplas câmeras para gerar novas visualizações da cena.

Neste trabalho partimos da suposição de que um modelo 3D da cena já está disponível e foi construído usando algum dos métodos existentes já que a construção do modelo não é nosso objetivo. Ainda assim, para que fosse possível realizar os testes, construímos os modelos 3D simplificados para 2 cenas distintas utilizando apenas um conjunto de planos texturizados a partir de imagens das próprias cenas.

3.2.3 Segmentação e rastreamento

A segmentação e o rastreamento permite separar os objetos móveis do fundo da cena e também permite associar a cada um deles um rótulo identificador e suas posições reais na cena. Precisamos separar os objetos do fundo pois utilizamos técnicas distintas para renderizar em momentos diferentes o fundo da cena e os objetos móveis. Além disso, as posições fornecidas pelo rastreamento são essenciais para o posicionamento dos objetos no modelo 3D da cena. Finalmente, os rótulos permitem que o usuário interaja de maneira individual com cada objeto.

Realizamos a segmentação e o rastreamento dos objetos móveis usando o método descrito por Santos em [San09]. Neste método, a segmentação é feita usando um algoritmo de subtração de background onde são usadas misturas de gaussianas para modelar os pixels do background.

Depois da segmentação, as posições dos objetos móveis são calculadas usando restrição homográfica em múltiplas câmeras com integração de suporte. Seja Π o plano de referência no qual os objetos se encontram e H_i as homografias que levam um ponto de Π ao ponto correspondente na imagem da câmera c_i. Para cada pixel de foreground x_i na imagem segmentada da câmera c_i, calculamos o suporte $s_(x_i)$ que é dado pelo número de pixels de foreground acima de x_i. As Figuras 3.2a, 3.2b e 3.2c mostram os suportes para cada câmera representados por pixels com alta intensidade. Acumulamos os resultados de $s_(x_i)$ no ponto $H_i^{-1}x_i$ do plano Π como mostrado na Figura 3.2d.

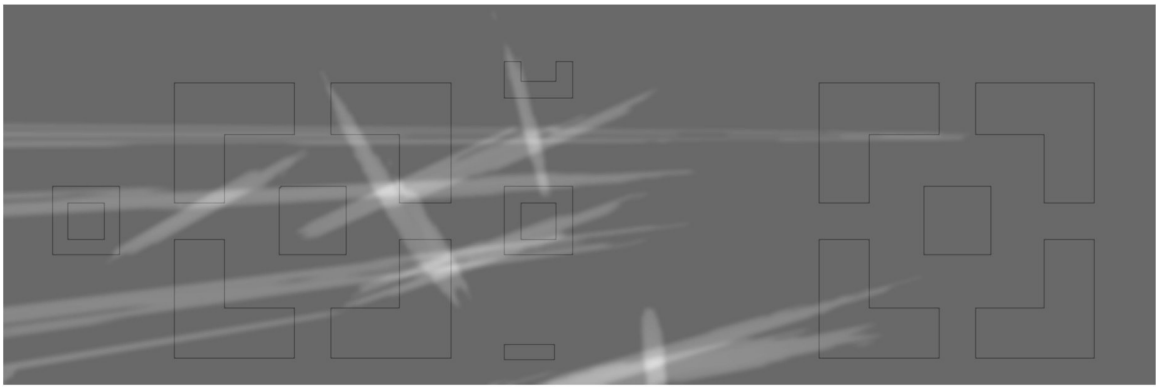
Supondo que um objeto é representado por uma linha vertical l sobre o plano Π, podemos obter as projeções l_i de l sobre Π com relação ao ponto de vista de cada uma das câmeras c_i como ilustrado na Figura 3.3. Os pontos de intersecção das projeções l_i juntamente com os máximos locais dos valores acumulados dos suportes fornecem as posições dos objetos móveis no plano Π. Na Figura 3.2d podemos ver as regiões de máximos locais indicadas pelas regiões de alta intensidade dos pixels e na Figura 3.2e temos as posições finais dos objetos calculadas com base nos máximos locais dos suportes e na restrição homográfica.

Para o rastreamento são usados filtros de Kalman e modelos de aparência para associar cada objeto da cena ao seu respectivo rastreador em todas as sequências de vídeo.

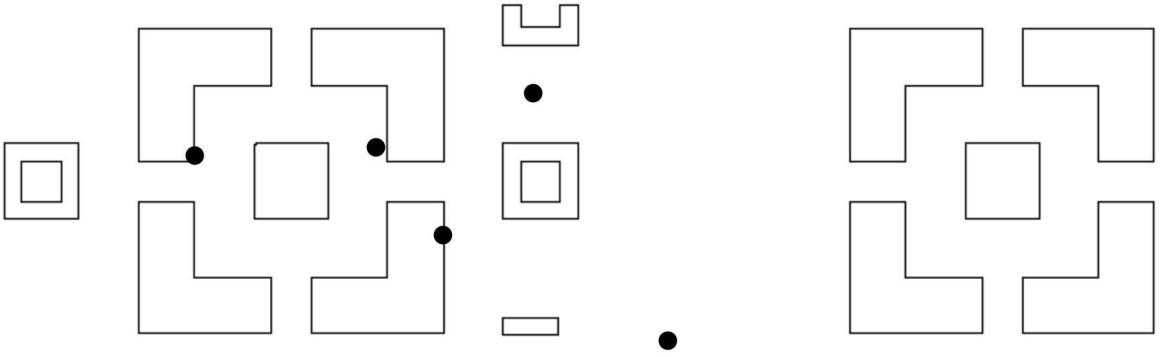
3.2.4 Fusão das imagens das múltiplas câmeras

A fusão das imagens das múltiplas câmeras diz respeito à minimização da diferença de brilho e cor existente entre as imagens de câmeras distintas. Podemos ver claramente essas diferenças na figura 3.4. Para que seja possível gerar novas visualizações realistas da cena combinando o modelo 3D com as imagens das câmeras, precisamos minimizar essas diferenças de modo que o usuário não perceba

(a) Suporte da câmera 1 (b) Suporte da câmera 2 (c) Suporte da câmera 3



(d) Resultado acumulado do suporte das três câmeras



(e) Posições dos objetos móveis da cena

Figura 3.2: Integração de suporte para obtenção das posições dos objetos móveis, da referência [San09]

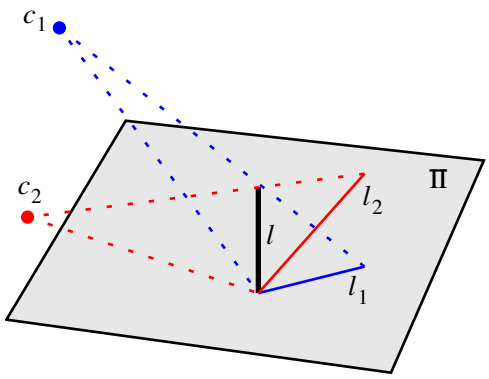


Figura 3.3: Restrição homográfica

(a) (b)

Figura 3.4: Diferenças de brilho e cor em imagens da mesma cena obtidas no mesmo instante com câmeras distintas. Podemos notar que **(a)** apresenta maior brilho e contraste em relação a **(b)**. Além disso, as cores de elementos correspondentes diferem bastante nas duas imagens.

as transições entre imagens de câmeras diferentes ou as transições das imagens das câmeras para as texturas do modelo 3D.

Definições básicas

As transformações de correção apresentadas nesse texto são definidas sobre imagens em escala de cinza, isto é, imagens com um único canal. Para aplicar essas transformações em imagens coloridas, podemos decompor a imagem original em três imagens em escala de cinza representando cada um dos

canais R, G e B. Podemos aplicar então as transformações em cada uma dessas imagens e reconstruir a imagem colorida pela composição das imagens resultantes de cada canal. Por esse motivo e também para simplificar a notação, todas as definições a seguir serão feitas sobre imagens em escala de cinza bastando seguir o procedimento descrito para lidar com imagens coloridas.

Seja $I = (S, C)$ uma imagem em escala de cinza com 8 bits, onde S é um conjunto de pontos representando a forma da imagem I e C uma função que atribui uma intensidade de cor a cada ponto em S. Formalmente, definimos S e C da seguinte forma:

$$S = \{(x, y) \in \mathbb{N}^2 : 0 \leq x < largura, 0 \leq y < altura\}$$

$$C : \mathbb{N} \times \mathbb{N} \rightarrow [0, 255]$$

Para um determinado $p = (x, y) \in S$, sua cor é dada por $C(p) = v$.

Um histograma de uma imagem em escala de cinza é uma função discreta que representa a distribuição de todos os possíveis valores de cinza presentes na imagem. Formalmente, o histograma h da imagem $I = (S, C)$ é dado por:

$$h_I(k) = |\{p \in S : C(p) = k\}|, \text{ para } k \in \mathbb{N} \text{ e } 0 \leq k \leq 255$$

onde $|A|$ indica a cardinalidade de A.

Podemos definir também o histograma normalizado da imagem por

$$H_I(k) = \frac{h_I(k)}{|S|}$$

Definimos a diferença entre dois histogramas h_1 e h_2 por:

$$D(h_1, h_2) = \sum_{0 \leq k \leq 255} (h_1(k) - h_2(k))^2$$

Correção por transformação de histograma baseada nas regiões de sobreposição das imagens

Nossa abordagem para o problema de fusão de imagens consiste em aplicar transformações de histograma baseadas em tabelas de conversão para minimizar a diferença entre os histogramas das imagens nas suas áreas de sobreposição. Uma vez que as tabelas de conversão são calculadas para as regiões de sobreposição, podemos aplicá-las sobre as imagens originais para obter o resultado desejado. A Figura 3.5 mostra um exemplo de aplicação dessa abordagem.

As áreas de sobreposição devem ser escolhidas de modo a conter a maior quantidade de amostras de pares de cores que apresentem alguma diferença de cor e brilho entre as duas imagens. Testes empíricos mostraram que na maioria dos casos um mínimo de 10% da área de cada imagem é suficiente para obter bons resultados na calibração das cores desde que as áreas escolhidas contenham amostras para a maior parte das cores das imagens que apresentam alguma diferença de cor e brilho. Quando possível, o aumento das áreas de sobreposição fornece melhores resultados na calibração conforme as distribuições das cores das áreas de sobreposição se aproximam da distribuição real das cores de cada imagem. O desempenho do algoritmo não é significativamente afetado pelo uso de áreas maiores já que o nosso algoritmo de calibração das cores trabalha apenas com os histogramas das áres de sobreposição e o aumento das áreas afeta apenas o cálculo inicial do histograma de cada área de sobreposição.

Sejam $I_a = (S_a, C_a)$ e $I_b = (S_b, C_b)$ duas imagens distintas de uma mesma cena que apresentam alguma região de sobreposição. Sejam R_a e R_b as regiões que delimitam um plano comum entre as imagens I_a e I_b e seja G a homografia que leva um ponto $p_a \in R_a$ ao ponto correspondente

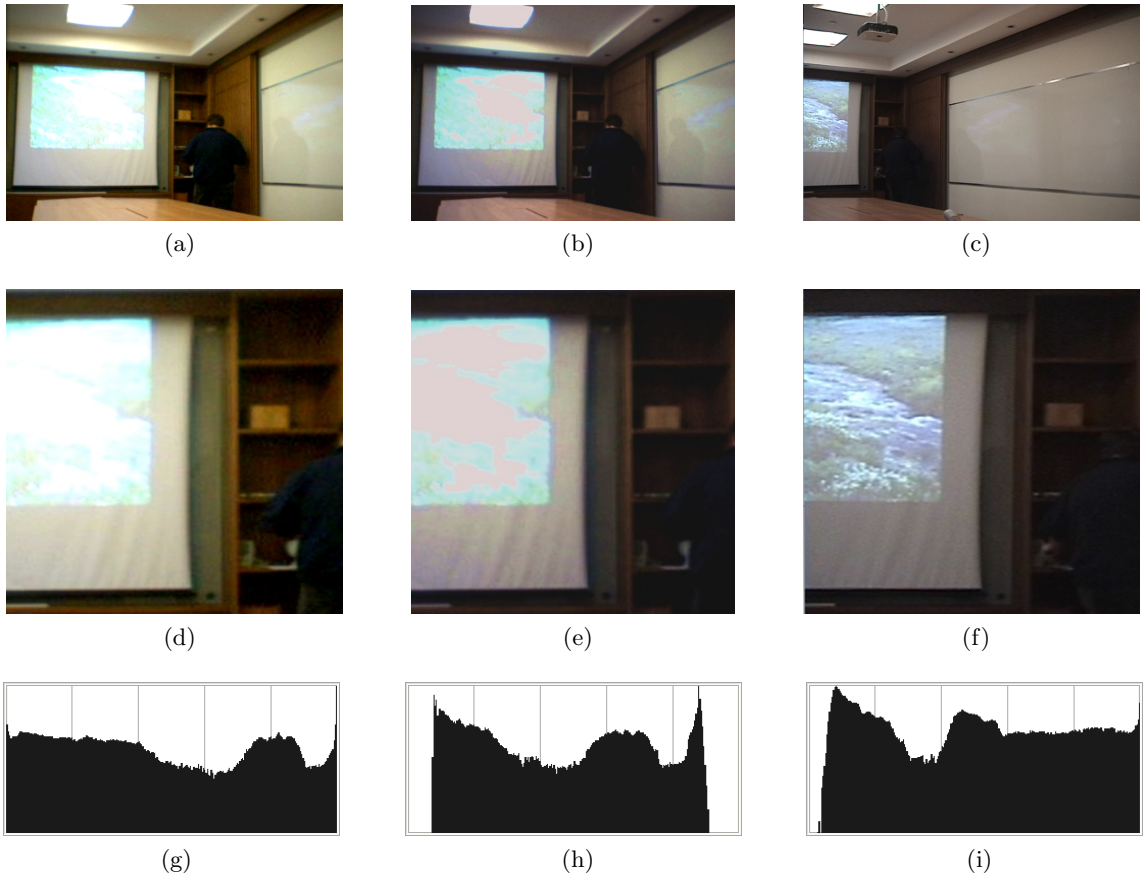


Figura 3.5: Exemplo de correção de cor por transformação de histogramas. **(a)** e **(c)** representam as imagens de um mesmo instante obtidas por duas câmeras distintas. **(d)** e **(f)** representam as regiões de sobreposição das duas câmeras e **(g)** e **(i)** representam os histogramas das intensidades dos pontos das imagens de sobreposição correspondentes. O processo de correção de cor determina uma transformação por tabelas de conversão que minimiza a diferença entre os histogramas **(g)** e **(i)**. O histograma transformado pode ser visto em **(h)** e imagem original corrigida pode ser vista em **(b)**.

$p_b \in R_b$. Definimos então as regiões de sobreposição de I_a e I_b pelas sub-imagens $I_{ab} = (S_{ab}, C_a)$ e $I_{ba} = (S_{ba}, C_b)$ onde

$$S_{ab} = \{p_a \in R_a : Gp_a \in S_b\}$$

$$S_{ba} = \{p_b \in R_b : G^{-1}p_b \in S_a\}$$

Uma tabela de conversão pode ser definida como uma função que atribui um novo valor para cada nível de cinza de uma imagem. Formalmente, uma tabela de conversão é definida por $L : [0, 255] \rightarrow [0, 255]$. Seja $l(x) = c_0 + c_1x + c_2x^2 + \cdots + c_nx^n$ um polinômio de grau n então podemos obter uma tabela de conversão a partir de l calculando $L(x) = l(x)$ com $x \in \mathbb{N}$ e $0 \leq x \leq 255$.

Ao aplicarmos uma tabela de conversão L sobre uma imagem $I = (S, C)$ obtemos uma nova imagem $L(I) = L(S, C) = (S, L(C))$.

Com isso podemos descrever o problema como um problema de otimização onde queremos encontrar o polinômio l para o qual $D(H_{L(I_{ab})}, H_{I_{ba}})$ ou $D(H_{L(I_{ba})}, H_{I_{ab}})$ é mínima, onde L é a tabela de conversão obtida a partir de l.

Implementação

Para resolver o problema, utilizamos a técnica de recozimento simulado (*simulated annealing*) [KGV83] que é uma técnica de busca probabilística. A busca começa em um estado inicial s_0 geralmente escolhido aleatoriamente. Em cada passo o algoritmo faz uma transição para um estado vizinho s_n aleatório com probabilidade p que depende de uma função temperatura T e também da diferença de energia entre o estado atual e o vizinho $\delta E = E(s_n) - E(s)$. Transições com $\delta E < 0$ são realizadas com $p = 1.0$ e transições com $\delta E \geq 0$ são realizadas com $p = f(\delta E, T)$ onde f é inversamente proporcional a δE e diretamente proporcional a T. A função temperatura T é uma função decrescente que reduz gradualmente a probabilidade de que o algoritmo aceite transições para estados vizinhos com energia maior que o estado atual. Dessa forma, inicialmente o algoritmo aceita transições para

estados vizinhos com energia maior ou menor que o estado atual com probabilidade elevada reduzindo assim a possibilidade de escolha de mínimos locais. Conforme a temperatura T se aproxima de 0, o algoritmo tende a rejeitar as transições para estados com energia maior aceitando apenas transições para estados de menor energia.

Para utilizar esse método na resolução do problema de correção precisamos remodelar o problema nos termos dos parâmetros usados no recozimento simulado.

Começamos definindo um estado por um conjunto de $n+1$ pontos posicionados sobre uma grade $g \times g$ definida no espaço $[0, 255] \times [0, 255]$ que representará um polinômio de grau n que passa por cada um dos $n + 1$ pontos. Formalmente, definimos um estado por $s = (p_0, p_1, p_2, \ldots, p_n)$ com $p_k = (x_k, y_k)$. Os valores x_k e y_k devem se restringir à grade $g \times g$ e por isso $x_k = gu$, $y_k = gv$ com $g, u, v \in \mathbb{N}$, $0 \leq x_k \leq 255$ e $0 \leq y_k \leq 255$.

Além disso definimos algumas restrições para o posicionamento dos pontos de modo a evitar a geração de polinômios que não sejam monotonicamente crescentes. Dados dois pontos $p_i = (x_i, y_i)$ e $p_{i+1} = (x_{i+1}, y_{i+1})$ com $0 \leq i < n$ e $i \in \mathbb{N}$, devemos ter $x_{i+1} > x_i$ e $y_{i+1} > y_i$.

Sejam $s = (p_0, p_1, p_2, \ldots, p_n)$ e $t = (q_0, q_1, q_2, \ldots, q_n)$ dois estados arbitrários. Dizemos que s e t são vizinhos se $\exists i \,|\, p_i \neq q_i$, $p_k = q_k \forall k \neq i$ e $|x_{p_i} - x_{q_i}| + |y_{p_i} - y_{q_i}| = g$. A Figura 3.6 ilustra um possível estado e todos os seus possíveis estados vizinhos.

A energia de um estado $s = (p_0, p_1, \ldots, p_n)$ é obtida calculando-se $D(H_{L_s(I_{ab})}, H_{I_{ba}})$, onde L_s é a tabela de conversão obtida a partir de s. Para construir a tabela de conversão L_s, precisamos obter o polinômio $l(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$ representado por s. Esse polinômio pode ser obtido resolvendo-se o seguinte sistema:

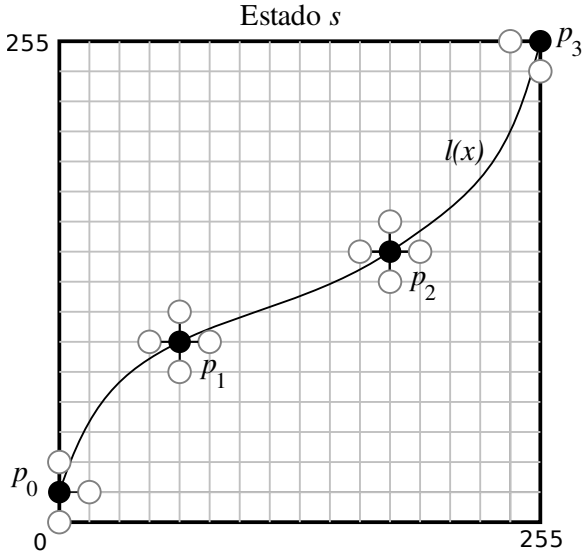


Figura 3.6: Estado e vizinhos para o recozimento simulado. O estado $s = (p_0, p_1, p_2, p_3)$ representa um polinômio de terceiro grau indicado na figura por $l(x)$. Cada vizinho do estado s é obtido movendo um p_k qualquer para algum dos seus vizinhos indicados pelos círculos preenchidos de branco na grade.

$$\begin{bmatrix} x_0^n & x_0^{n-1} & x_0^{n-2} & \dots & x_0 & 1 \\ x_1^n & x_1^{n-1} & x_1^{n-2} & \dots & x_1 & 1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ x_n^n & x_n^{n-1} & x_n^{n-2} & \dots & x_n & 1 \end{bmatrix} \begin{bmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_0 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

Finalmente, a função temperatura é definida por uma temperatura inicial T_0, escolhida de acordo com o problema a ser resolvido, e atualizada a cada K iterações por $T_{i+1} = \alpha T_i$ com $0 \leq \alpha < 1$.

No Algoritmo 1 mostramos a descrição completa do algoritmo para o cálculo da tabela de correção de cores da imagem I_a para a imagem I_b.

Algoritmo 1 Cálculo da tabela de correção de cores da imagem I_a para a imagem I_b

Entrada: $s_0, T_0, \epsilon, K, H_{I_{ab}}, H_{I_{ba}}$
Ensure: L

1. $i \leftarrow 0$
2. $s \leftarrow s_0$
3. $T \leftarrow T_0$
4. **enquanto** $T > \epsilon$ **faça**
5. **se** $i = K$ **então**
6. $T \leftarrow$ AtualizaTemperatura(T)
7. $i \leftarrow 0$
8. **fim se**
9. $s_n \leftarrow$ VizinhoAleatório(s)
10. $L_s \leftarrow$ CalculaTabelaCorreção(s)
11. $L_{s_n} \leftarrow$ CalculaTabelaCorreção(s_n)
12. $E_s \leftarrow D(H_{L_s(I_{ab})}, H_{I_{ba}})$
13. $E_{s_n} \leftarrow D(H_{L_{s_n}(I_{ab})}, H_{I_{ba}})$
14. $\delta E \leftarrow E_{s_n} - E_s$
15. **se** $\delta E < 0$ **então**
16. $p \leftarrow 1.0$
17. **senão**
18. $p \leftarrow e^{\frac{-\delta E}{T}}$
19. **fim se**
20. $q \leftarrow$ GeraProbabilidade()
21. **se** $q < p$ **então**
22. $s \leftarrow s_n$
23. **fim se**
24. $i \leftarrow i + 1$
25. **fim enquanto**
26. $L \leftarrow$ CalculaTabelaCorreção(s)
27. **devolva** L

3.3 Renderização

Para a renderização das novas visualizações, utilizamos uma variação da técnica de texturização dependente da vista descrita em [DYB98] incorporando a fusão das imagens das múltiplas câmeras, a renderização dos objetos móveis e a texturização de dependente da vista em tempo real para o modelo 3D da cena e para os modelos representantes dos objetos móveis.

3.3.1 Visão geral

Na etapa de renderização utilizamos os dados obtidos na etapa anterior para gerar novas visualizações da cena, em tempo real, permitindo a interação do usuário através da escolha da posição e da direção de visualização da câmera, do modo de reprodução do vídeo (play, pause, fast-forward, etc.) ou através da escolha de quais objetos móveis devem ser mostrados ou destacados na cena, por exemplo.

A renderização começa com o tratamento do quadro atual de cada vídeo aplicando-se as tabelas de correção de cores obtidas anteriormente com o objetivo de minimizar a diferença de cores entre as imagens do quadro atual. Em seguida, usamos as máscaras obtidas na segmentação para separar as imagens de cada vídeo em duas imagens: a primeira contendo apenas o fundo da cena e a segunda contendo os objetos móveis. Então, renderizamos o modelo 3D da cena a partir do ponto de vista escolhido pelo usuário e projetamos sobre esse modelo uma combinação das imagens de fundo que melhor se aproximam do ponto de vista escolhido. Finalmente, renderizamos os objetos móveis sobre a cena na forma de painéis texturizados com as imagens dos objetos móveis.

Nas seções seguintes, apresentaremos detalhadamente cada um dos passos que compõem a etapa de renderização.

3.3.2 Geração das texturas

Durante a etapa de renderização trabalhamos com texturas de 32-bits no formato RGBA, isto é, imagens com três canais de cores R, G, e B, cada um com 8-bits, e um canal alfa de 8-bits que representa a opacidade de cada pixel na imagem. Podemos então definir uma textura com largura L e altura A pelo par ordenado $T = (S, C)$ onde:

$$S = \{(x, y) \in \mathbb{N}^2 : 0 \leq x < L, 0 \leq y < A\}$$

$$C : \mathbb{N} \times \mathbb{N} \to [0, 255]^4$$

Para um determinado $p = (x, y) \in S$, sua cor $c = (r, g, b, a)$ é dada por $C(p)$. Para simplificar a notação adotamos que $T(p) = C(p)$ com $p \in S$.

Sejam $V_1, V_2, \ldots, V_n$ os n vídeos da cena em questão onde cada vídeo V_i é uma função que associa a cada instante t uma textura $T_{i,t} = (S_{i,t}, C_{i,t})$.

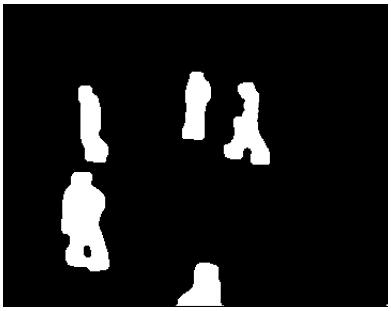
O primeiro passo da renderização consiste da aplicação das tabelas de conversão de cores em cada um dos vídeos da cena. Da etapa de pré-processamento, temos as tabelas de conversão de cor $L_{i_R}, L_{i_G}, L_{i_B}$ correspondentes a cada um dos canais R, G e B do vídeo V_i. Seja $T_{i,t} = (S_{i,t}, C_{i,t}) = V_i(t)$ e $C_{i,t}(p) = (r, g, b, a)$, realizamos a correção de cor de $T_{i,t}$ pela aplicação de L_i da seguinte forma:

$$L_i T_{i,t}(p) = L_i C_{i,t}(p) = (L_{i_R}(r), L_{i_G}(g), L_{i_B}(b), a)$$

Com isso, para um determinado instante t, obtemos as texturas corrigidas fazendo $T'_{i,t} = L_i T_{i,t}$ para $1 \leq i \leq n$.

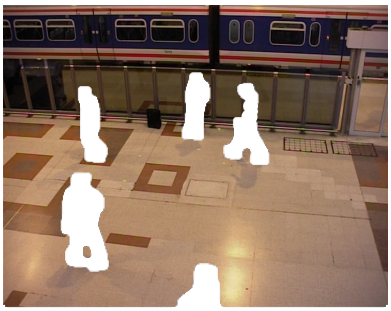
O próximo passo envolve a separação do fundo da cena dos objetos móveis em cada vídeo. Começamos definindo a máscara da segmentação obtida na etapa de pré-processamento. Definimos a máscara $M_{i,t} = (S_{i,t}, A_{i,t})$ como uma imagem, contendo um único canal $A_{i,t}$ de 8-bits, associada

(a) Imagem de uma câmera



(b) Máscara da segmentação

(c) Textura de objetos



(d) Textura de fundo

Figura 3.7: Geração das texturas.

ao instante t do vídeo V_i.

Podemos então definir duas novas texturas $F_{i,t}$ e $B_{i,t}$ derivadas de $T'_{i,t}$. A textura $B_{i,t}$, denominada textura de fundo, deve conter apenas o fundo da cena com regiões transparentes no lugar dos objetos móveis. A textura $F_{i,t}$, denominada textura de objetos, deve ser o complemento de $B_{i,t}$, isto é, deve conter os objetos móveis e regiões transparentes no lugar do fundo da cena. As regiões de transparência são representadas por *pixels* com opacidade(canal alfa) próxima ou igual a zero. Um exemplo de geração dessas texturas pode visto na Figura 3.7.

Para construir $F_{i,t}$ e $B_{i,t}$ devemos combinar o canal alfa de $M_{i,t}$ com o canal de alfa de $T'_{i,t}$ preservando os canais de cores R, G e B de $T'_{i,t}$. Seja $T'_{i,t}(p) = (r, g, b, a)$ então:

$$F_{i,t}(p) = (r, g, b, \frac{aA(p)}{255})$$

$$B_{i,t}(p) = (r, g, b, \frac{a(255 - A(p))}{255})$$

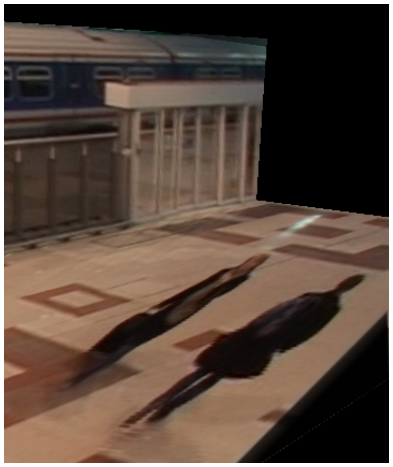
3.3.3 Renderização do modelo da cena

A renderização da cena se dá pela uso de uma variação da técnica de texturização dependente da vista. Na técnica original, um modelo 3D da cena é renderizado e cada de suas faces é texturizada com uma combinação das imagens das várias câmeras usadas na aquisição da cena. Nem todas as faces do modelo podem ser texturizadas pelas imagens das câmeras pois existe a possibilidade de que essas faces não tenham sido observadas por alguma das câmeras. Nesse caso, essas faces têm suas texturas aproximadas pela interpolação das cores das faces vizinhas. Esse processo é denominado preenchimento das regiões de oclusão.

Em nossa variante da técnica, resolvemos o problema de preenchimento das regiões de oclusão utilizando um modelo 3D da cena onde cada uma de suas faces já foi texturizada durante sua construção. Dessa forma, evitamos o problema de existir alguma face do modelo não observada pelas câmeras utilizadas. Essa abordagem também gera renderizações mais realistas já que usamos texturas obtidas da própria cena ao invés de aproximarmos os detalhes das regiões não observadas pela interpolação das cores das faces vizinhas.

Para que seja possível renderizar os objetos móveis precisamos extraí-los primeiro das imagens das câmeras. Como as imagens de cada câmera são projetadas sobre o modelo 3D da cena, se os objetos não forem extraídos dessas imagens eles serão renderizados com a perspectiva incorreta para pontos de vista diferentes dos pontos de vista das câmeras utilizadas como mostra a Figura 3.8.

A extração dos objetos móveis da imagens das câmeras origina texturas com regiões de oclusão



(a) Perspectiva incorreta

(b) Perspectiva correta

Figura 3.8: Erro de perspectiva na renderização dos objetos móveis. **(a)** mostra a renderização dos objetos móveis pela projeção do vídeo da cena sobre um modelo simplificado. **(b)** mostra a renderização dos objetos móveis pelo uso de painéis(*billboards*) posicionados de acordo com o novo ponto de vista.

nos locais antes ocupados por esses objetos. Essas regiões são preenchidas de forma simples através de múltiplas aplicações de textura para cada face do modelo. Inicialmente, renderizamos o modelo da cena e aplicamos as texturas definidas durante a sua construção. Em seguida, sobre cada face, aplicamos uma combinação das texturas das câmeras considerando os valores do canal alfa dessa nova textura. Com isso, a nova textura irá sobrepor as texturas do modelo da cena em todas as regiões exceto nas regiões de oclusão, que são transparentes, onde as texturas resultantes serão iguais àquelas do modelo da cena.

Começamos então a descrição de nossa variante da técnica de renderização dependente da vista com algumas definições.

Definimos uma câmera como um par ordenado formado por um vetor $\mathbf{p} \in \mathbb{R}^3$ que descreve sua posição e um vetor $\mathbf{v} \in \mathbb{R}^3$ que descreve sua orientação. Seja $c = (\mathbf{p}, \mathbf{d})$ a câmera escolhida pelo usuário e sejam $c_i = (\mathbf{p}_i, \mathbf{d}_i)$ para $1 \leq i \leq n$ as n câmeras utilizadas na aquisição da cena.

Iniciamos a renderização de uma nova visualização renderizando o modelo 3D da cena a partir do ponto de vista determinado por c. Em seguida, precisamos aplicar sobre cada face do modelo uma combinação das texturas $B_{i,t}$ que melhor represente a textura daquela face. Para isso, precisamos considerar o ponto de vista escolhido pelo usuário c e também os pontos de vista de cada uma das câmeras da cena para atribuir pesos a cada uma das câmeras conforme a sua proximidade com o novo ponto de vista.

Para descrever como calculamos os pesos correspondentes a cada câmera vamos considerar uma face triangular qualquer do modelo determinada pelos vértices $\mathbf{a} = (x_a, y_a, z_a)$, $\mathbf{b} = (x_b, y_b, z_b)$ e $\mathbf{c} = (x_c, y_c, z_c)$. Utilizaremos o ponto $\mathbf{f} = (\frac{x_a+x_b+x_c}{3}, \frac{y_a+y_b+y_c}{3}, \frac{z_a+z_b+z_c}{3})$, que é o baricentro do triângulo formado pelos vértices **a**, **b** e **c**, para representar a face escolhida, como ilustrado na figura 3.9.

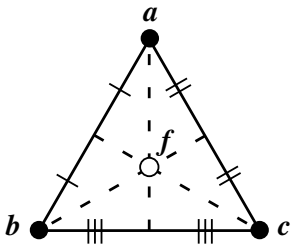


Figura 3.9: Baricentro de um triângulo calculado pela intersecção das medianas.

Queremos agora discretizar o espaço de direções originadas em **f** como pontos na superfície de uma esfera de raio 1 com centro em **f**. Seja $l \in \mathbb{N}$ uma constante tal que $l \geq 4$ e sejam $s \in \mathbb{Z}, t \in \mathbb{N}$ onde $\frac{-l}{2} \leq s \leq \frac{l}{2}$, $0 \leq t < l$. Usamos o sistema de coordenadas esféricas com origem em **f** para definir o conjunto de direções ao redor de **f** pelos pontos $v_{s,t} = (r_{s,t}, \theta_{s,t}, \varphi_{s,t})$, onde $r_{s,t}$ representa a distância de $p_{s,t}$ em relação a **f**, $\varphi_{s,t}$ representa o azimute, ou seja, a rotação de $p_{s,t}$ em relação ao eixo Y e $\theta_{s,t}$ a elevação de $p_{s,t}$ em relação ao plano XZ. As coordenadas de $v_{s,t}$ são calculadas como a seguir:

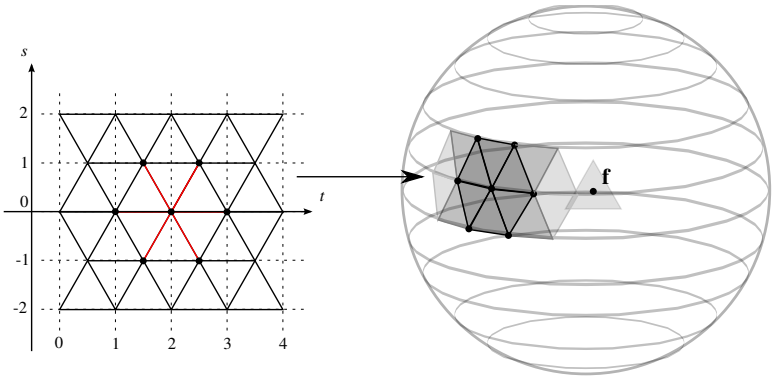


Figura 3.10: Discretização do espaço de direções ao redor do ponto $\mathbf{f}$.

$$r_{s,t} = 1$$

$$\theta_{s,t} = \frac{\pi s}{l}$$

$$\varphi_{s,t} = \begin{cases} \dfrac{2\pi t}{l} & \text{se } s \text{ é par} \\ \dfrac{\pi}{l} + \dfrac{2\pi t}{l} & \text{se } s \text{ é ímpar} \end{cases}$$

Nosso próximo passo consiste em escolher os três pontos mais próximos do segmento de reta com origem em $\mathbf{f}$ e término em $\mathbf{p}$, a posição da câmera $c = (\mathbf{p}, \mathbf{d})$ escolhida pelo usuário. Uma forma simples de encontrar esses pontos consiste em ligar cada um dos pontos definido sobre a esfera aos seus 6 vizinhos mais próximos. Dessa forma, definimos uma malha de triângulos sobre a esfera de raio 1 com centro em $\mathbf{f}$ como demonstrado na Figura 3.10. Os pontos procurados ficam então determinados pelos vértices do triângulo interceptado pelo segmento de reta $\overline{\mathbf{fp}}$ como na Figura 3.11.

Devido à forma como os pontos sobre a esfera foram definidos, podemos encontrar os pontos mais próximos de $\overline{\mathbf{fp}}$ diretamente sem que seja preciso testar sua intersecção com cada triângulo da malha.

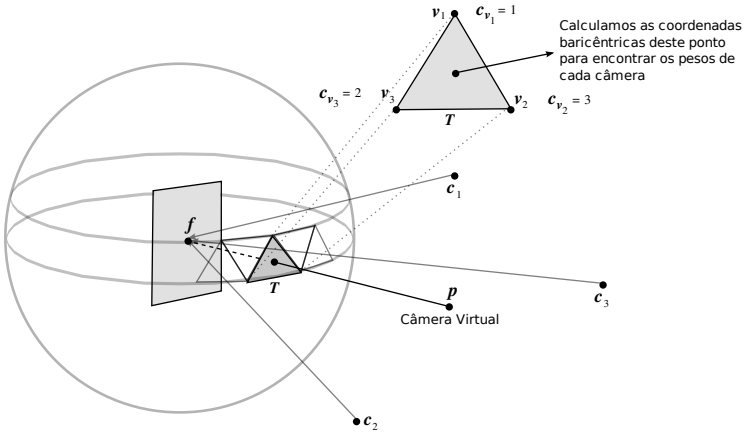


Figura 3.11: Cálculo dos pesos das câmeras para a texturização dependente da vista. Primeiro, determinamos o triângulo interceptado pelo segmento $\overline{\mathbf{fp}}$ e associamos a cada vértice desse triângulo a câmera com orientação mais próxima da orientação representada pelo vértice. Depois, calculamos as coordenadas baricêntricas do ponto dado pela intersecção do segmento $\overline{\mathbf{fp}}$ com o triângulo para encontrar o peso da câmera associada a cada vértice.

Começamos definindo um vetor unitário representando a direção de $\overline{\mathbf{fp}}$ por $\mathbf{r} = \frac{\mathbf{p}-\mathbf{f}}{||\mathbf{p}-\mathbf{f}||}$. Também calculamos as coordenadas esféricas de $\mathbf{r}$ com o objetivo de restringir a região da esfera na qual devemos testar a intersecção do segmento $\overline{\mathbf{fp}}$. Seja $\mathbf{r} = (x, y, z)$ temos que a projeção de $\mathbf{r}$ sobre o plano XZ é dada por $\mathbf{r}_{XZ} = (x, 0, z)$. Obtemos o azimute de $\mathbf{r}$ calculando $\theta_{\mathbf{r}} = \arccos(< \frac{\mathbf{r}_{XZ}}{||\mathbf{r}_{XZ}||}, \mathbf{x} >)$ onde $\mathbf{x} = (1, 0, 0)$. A elevação de $\mathbf{r}$ em relação ao plano XZ é dada por $\varphi_{\mathbf{r}} = \arcsin(z)$. Sejam $s' \in \mathbb{Z}$ e $t' \in \mathbb{N}$ definidos como:

$$s' = \begin{cases} \dfrac{\theta_{\mathbf{r}}}{l} & \text{se } \theta_{\mathbf{r}} \geq 0 \\ \dfrac{\theta_{\mathbf{r}}}{l} - 1 & \text{se } \theta_{\mathbf{r}} < 0 \end{cases}$$

$$t' = \frac{\varphi_{\mathbf{r}}}{l}$$

Os valores de s' e t' nos indicam a região da esfera que contém o ponto de intersecção do segmento

$\overline{\mathbf{fp}}$ com a esfera. Dessa forma, podemos definir um conjunto de 5 pontos que são os únicos candidatos a pontos mais próximos do segmento $\overline{\mathbf{fp}}$. Vamos denominar estes pontos de p_a, p_b, p_c, p_d e p_e. Suas coordenadas são determinadas de acordo com o valor de s'. Para s' par, temos:

$$p_a = (r_a, \theta_a, \varphi_a) = (1, \frac{\pi(s'+1)}{l}, \frac{\pi}{l} + \frac{2\pi(t'-1)}{l})$$

$$p_b = (r_b, \theta_b, \varphi_b) = (1, \frac{\pi(s'+1)}{l}, \frac{\pi}{l} + \frac{2\pi t'}{l})$$

$$p_c = (r_c, \theta_c, \varphi_c) = (1, \frac{\pi(s'+1)}{l}, \frac{\pi}{l} + \frac{2\pi(t'+1)}{l})$$

$$p_d = (r_d, \theta_d, \varphi_d) = (1, \frac{\pi s'}{l}, \frac{2\pi t'}{l})$$

$$p_e = (r_e, \theta_e, \varphi_e) = (1, \frac{\pi s'}{l}, \frac{2\pi(t'+1)}{l})$$

Caso s' seja ímpar, temos:

$$p_a = (r_a, \theta_a, \varphi_a) = (1, \frac{\pi(s'+1)}{l}, \frac{2\pi t'}{l})$$

$$p_b = (r_b, \theta_b, \varphi_b) = (1, \frac{\pi(s'+1)}{l}, \frac{2\pi(t'+1)}{l})$$

$$p_c = (r_c, \theta_c, \varphi_c) = (1, \frac{\pi s'}{l}, \frac{\pi}{l} + \frac{2\pi(t'-1)}{l})$$

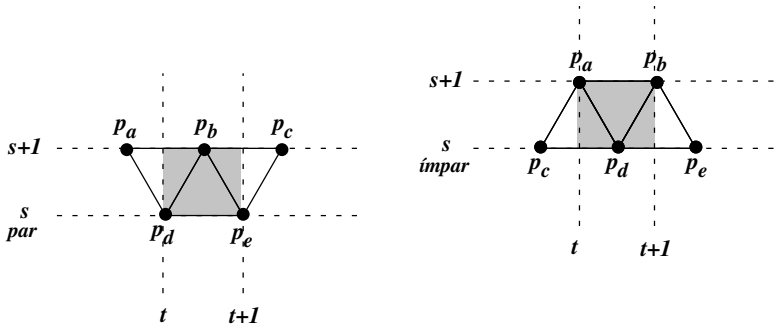


Figura 3.12: Definição dos pontos p_a, p_b, p_c, p_d e p_e de acordo com os valores de s' e t'.

$$p_d = (r_d, \theta_d, \varphi_d) = (1, \frac{\pi s'}{l}, \frac{\pi}{l} + \frac{2\pi t'}{l})$$

$$p_e = (r_e, \theta_e, \varphi_e) = (1, \frac{\pi s'}{l}, \frac{\pi}{l} + \frac{2\pi (t'+1)}{l})$$

A figura 3.12 mostra as duas possíveis configurações para esses pontos.

Em cada um dos casos mostrados acima, os pontos p_a, p_b, p_c, p_d e p_e formam três triângulos distintos que podem ser testados individualmente para determinar qual deles é interceptado pelo segmento $\overline{\mathbf{fp}}$. Para isso precisamos converter cada um desses pontos para o sistema cartesiano. A conversão é simples e pode ser realizada com o uso de duas matrizes de rotação $R_Z(\theta)$, correspondente a uma rotação de θ radianos em torno do eixo Z, e $R_Y(\varphi)$ correspondente a uma rotação de φ radianos em torno do eixo Y. Para converter um ponto $p = (r, \theta, \varphi)$ para o sistema de coordenadas cartesiano basta calcularmos $\mathbf{v} = R_Y(\varphi)R_Z(\theta)\mathbf{x} + \mathbf{f}$ onde $\mathbf{x} = (1, 0, 0)$.

Então, sejam $\mathbf{v}_a, \mathbf{v}_b, \mathbf{v}_c, \mathbf{v}_d$ e $\mathbf{v}_e$ os vetores no sistema cartesiano correspondentes aos pontos p_a, p_b, p_c, p_d e p_e. Queremos calcular a intersecção do segmento $\overline{\mathbf{fp}}$ com os triângulos $\triangle \mathbf{v}_a\mathbf{v}_b\mathbf{v}_d$, $\triangle \mathbf{v}_d\mathbf{v}_b\mathbf{v}_e$ e $\triangle \mathbf{v}_b\mathbf{v}_c\mathbf{v}_e$ para o caso de t' par ou com os triângulos $\triangle \mathbf{v}_c\mathbf{v}_a\mathbf{v}_d$, $\triangle \mathbf{v}_a\mathbf{v}_b\mathbf{v}_d$ e $\triangle \mathbf{v}_d\mathbf{v}_b\mathbf{v}_e$ para

o caso de t' ímpar.

Sejam v_1, v_2 e v_3 as coordenadas dos três vértices de um triângulo qualquer. Podemos calcular o ponto de intersecção do segmento de reta $\overline{\mathbf{fp}}$ com o triângulo $\triangle \mathbf{v}_1\mathbf{v}_2\mathbf{v}_3$ usando a equação de intersecção da reta com o plano. A equação fornece os valores de λ_1, λ_2 e λ_3 que representam as coordenadas baricêntricas do ponto de intersecção de $\overline{\mathbf{fp}}$ com o triângulo $\triangle \mathbf{v}_1\mathbf{v}_2\mathbf{v}_3$. Para que $\overline{\mathbf{fp}}$ intercepte o triângulo, devemos ter $\lambda_2 + \lambda_3 \leq 1$, $\lambda_2 \geq 0$ e $\lambda_3 \geq 0$. Descrevemos abaixo a equação de intersecção da reta com o plano:

$$\begin{bmatrix} t \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} \mathbf{p} - \mathbf{f} & \mathbf{v}_2 - \mathbf{v}_1 & \mathbf{v}_3 - \mathbf{v}_1 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{p} - \mathbf{v}_1 \end{bmatrix}$$

$$\lambda_1 = 1 - \lambda_2 - \lambda_3$$

Seguindo o processo descrito até agora, encontramos os três pontos mais próximos do segmento de reta $\overline{\mathbf{fp}}$. Suponha que os pontos encontrados sejam os pontos $\mathbf{v}_1, \mathbf{v}_2$ e $\mathbf{v}_3$ e que as coordenadas baricêntricas encontradas sejam λ_1, λ_2 e λ_3. O passo final consiste em associar a cada dos vértices v_i o índice j da câmera $c_j = (\mathbf{p}_j, \mathbf{d}_j)$ cuja direção de visualização $\mathbf{d}_j$ é a mais próxima da direção representada por v_i. Calculamos então c_{v_i}, o índice da câmera associada ao vértice i, conforme abaixo:

$$c_{\mathbf{v}_i} = \max_j < -\mathbf{d}_j, \mathbf{v}_i >, \forall j \in 1, 2, \ldots, n \text{ e } \forall i \in 1, 2, 3$$

Finalmente, concluímos a renderização do modelo da cena projetando a textura $B_{c_{v_i},t}$ sobre o modelo a partir do ponto de vista da câmera c_{v_i} com peso λ_i para $i \in 1, 2, 3$. O peso λ_i é aplicado de modo multiplicativo sobre o canal alfa da textura $B_{c_{v_i},t}$.

3.3.4 Renderização dos objetos móveis

O processo de renderização dos objetos móveis é bastante semelhante ao processo de renderização do modelo da cena. Basicamente, usamos a técnica de renderização por painéis(*billboarding*) onde renderizamos cada um dos objetos móveis como um plano texturizado cuja normal aponta na direção do plano de visualização da câmera escolhida pelo usuário. Cada um desses planos é então texturizado com uma combinação das texturas $F_{i,t}$ cujos pesos são calculados da mesma forma como são calculados os pesos na etapa de renderização do modelo. As principais diferenças estão na seleção das regiões das texturas $F_{i,t}$ a serem usadas para cada objeto e no uso da técnica de renderização por painéis com texturização dependente da vista.

Inicialmente, definimos um plano de referência Π na qual todos os objetos móveis se encontram. Seja $\mathbf{o}_{j,t}$ a posição central de contato do objeto j com o plano Π no instante t, onde $0 \leq j < m$, e seja H_i a homografia que leva um ponto $\mathbf{q}$ de Π ao ponto correspondente na imagem da câmera c_i, com $0 \leq i < n$. Para calcular o retângulo envolvente (*bounding box*) do objeto j na textura $F_{i,t}$, primeiro calculamos a posição $\mathbf{b}_{i,j,t} = H_i\mathbf{o}_{j,t}$ que corresponde à posição do objeto na textura de objetos da câmera i. Seja $h_{i,j,t}$ a altura aproximada do objeto j na textura $F_{i,t}$ no instante t. Temos então que o retângulo envolvente do objeto j em $F_{i,t}$ fica determinada pelo par de pontos $(\mathbf{z}, \mathbf{w})$, correspondentes aos cantos superior esquerdo e inferior direito do retângulo envolvente como na Figura 3.13. O par $(\mathbf{z}, \mathbf{w})$ é calculado da seguinte forma:

$$\mathbf{z} = \mathbf{b}_{i,j,t} - \frac{\gamma \mathbf{x} h_{i,j,t}}{2} + \mathbf{y} h_{i,j,t}$$

$$\mathbf{w} = \mathbf{b}_{i,j,t} + \frac{\gamma \mathbf{x} h_{i,j,t}}{2}$$

onde $\mathbf{x} = (1, 0), \mathbf{y} = (0, 1)$ e $\gamma \in \mathbb{R}$ é uma constante que determina a largura do retângulo

envolvente em relação à sua altura. Para usar o método com objetos alongados na vertical como pessoas, devemos ter $\gamma < 1$. Caso os objetos a serem tratados sejam alongados na horizontal, devemos escolher um $\gamma > 1$ adequado.

Após determinarmos os retângulos envolventes para o objeto j em cada uma das texturas $F_{i,t}$, devemos posicionar corretamente o painel que irá representar o objeto no modelo 3D da cena. Como o modelo da cena é construído utilizando o mesmo sistema de coordenadas e a mesma escala da cena real, a posição de cada painel no modelo é dada diretamente por $\mathbf{o}_{j,t}$. Inicialmente definimos que todos os painéis possuem normal $\mathbf{n}_{j,t} = (0, 1, 0)$.

O passo seguinte consiste em rotacionar cada painel de modo que suas normais apontem na direção do plano de visualização da câmera virtual $c = (\mathbf{p}, \mathbf{d})$. A normal do painel ajustado será dada por $\mathbf{n'_{j,t}} = R_Y(\varphi)R_Z(\theta)\mathbf{n}_{j,t}$, onde R_Y é uma matriz de rotação em torno do eixo Y e R_Z é uma matriz de rotação em torno do eixo Z. Seja $\mathbf{d} = (x, y, z)$ e $\mathbf{d}_{XZ}$ a projeção de $\mathbf{d}$ no plano XZ, então os parâmetros φ e θ ficam determinados por:

$$\varphi = \arccos(< \mathbf{n}_{j,t}, \frac{-\mathbf{d}_{XZ}}{||\mathbf{d}_{XZ}||} >)$$

$$\theta = \arcsin(\frac{z}{||\mathbf{d}||})$$

O cálculo dos pesos das texturas $F_{i,t}$ é realizado da mesma forma apresentada na renderização do modelo da cena. Neste caso, a única diferença é que discretizamos o espaço de direções ao redor de cada objeto para obter os índices e os pesos das três câmeras com pontos de vista mais próximos do ponto de vista da câmera virtual. Finalmente, aplicamos as texturas no painel usando os pesos obtidos para combiná-las usando *alpha blending*. Como os recortes de cada textura possuem tamanhos diferentes, eles são normalizados e redimensionados para que eles cubram toda a área do painel quando usados como texturas.

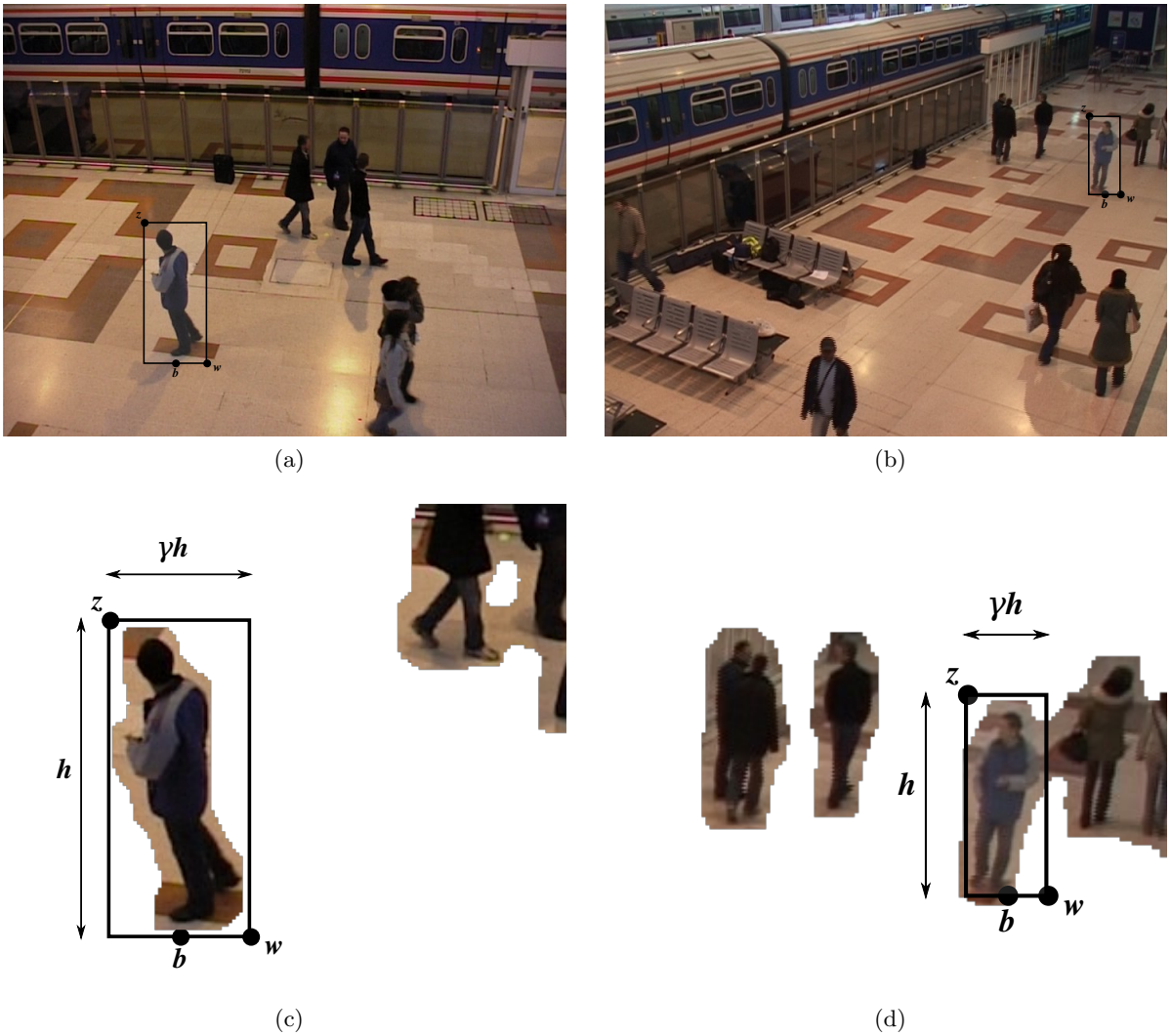


Figura 3.13: Recorte das texturas para os painéis. **(a)** e **(b)** mostram imagens de duas câmeras distintas em um mesmo instante. O objeto considerado para o recorte das texturas está delimitado por uma retângulo nas duas imagens. Em **(c)** e **(d)**, vemos as texturas de objetos correspondentes às imagens **(a)** e **(b)** e temos uma visão mais detalhada das regiões que serão utilizadas como textura do objeto em questão.

3.3.5 Diferenças em relação à técnica de texturização dependente da vista

A aplicação do método descrito neste trabalho para renderização de novas visualizações em tempo real requer que alguns dados sejam obtidos em uma etapa inicial de pré-processamento. Além da construção do modelo 3D simplificado e da calibração e sincronizaçào dos vídeos, calculamos as tabelas de conversão de cores para cada vídeo e realizamos a segmentação e o rastreamento dos objetos móveis nesta primeira etapa de pré-processamento.

Na etapa de renderização, nosso método para a geração de novas visualizações de cena dinâmicas difere em relação a técnica original primeiramente pelo uso de um modelo 3D da cena já texturizado. Esse modelo permite que as regiões de oclusão da cena, presentes em áreas que não foram observadas por nenhuma câmera ou em áreas ocupadas pelos objetos móveis, sejam preenchidos com mais detalhes. Essa alteração gera renderizações mais realistas se comparadas ao método de preenchimento das regiões de oclusão utilizado na técnica original pois preenche essas regiões com detalhes da própria cena enquanto a técnica original preenche as regiões pela interpolação das cores das regiões vizinhas.

Uma outra diferença é a fusão das imagens das múltiplas câmeras que permite uma melhor mesclagem entre as imagens dos vídeos e o modelo 3D da cena. Como as imagens dos múltiplos vídeos são combinadas entre si e com o modelo para gerar novas visualizações, as imagens com brilho e cor corrigidas permitem que o resultado da mistura seja bastante imperceptível contribuindo para a geração de visualizações mais realistas.

Finalmente, uma última diferença é a inclusão da renderização dos objetos móveis independentes utilizando-se a técnica de renderização por painéis com texturização dependente da vista. Na técnica original, mapas de visualização para cada face da cena são pré-calculados antes da renderização para que seja possível escolher a melhor combinação de câmeras para a texturização de cada face do modelo 3D. Para o caso de cenas dinâmicas com objetos móveis, seria necessário pré-calcular mapas de visualização para cada instante dos vídeos da cena. Na abordagem apresentada neste trabalho,

calculamos essa combinação de câmeras antes de renderizar cada quadro usando um algoritmo linear no número de câmeras utilizadas na aquisição da cena e descrito em detalhes nas seções anteriores.

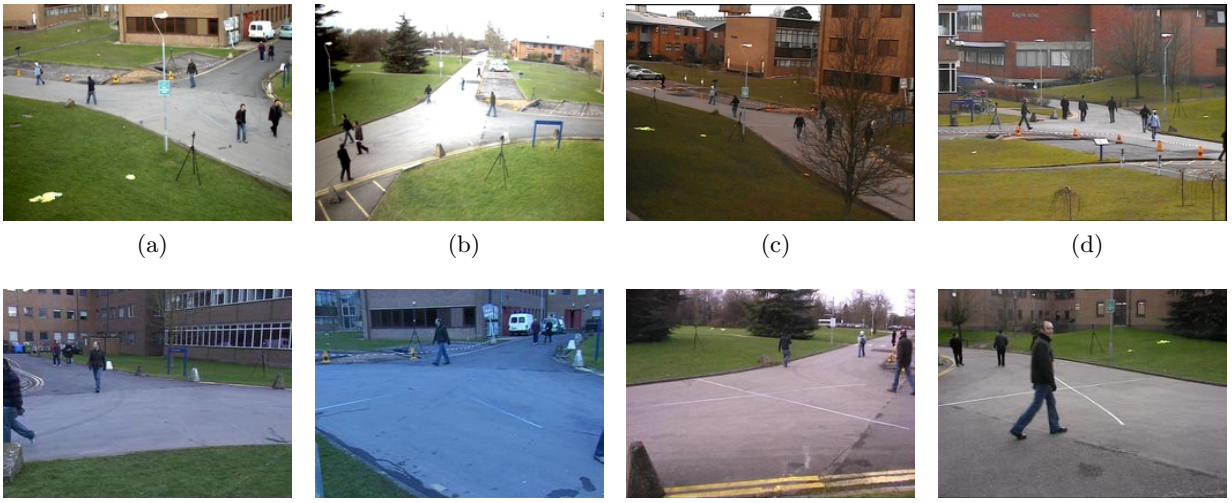
Capítulo 4

Resultados experimentais

Para verificar que o método apresentado neste trabalho produz os resultados desejados, realizamos alguns testes utilizando dados cujas condições se aproximam das condições reais esperadas para a aplicação do método.

Inicialmente discutimos brevemente a construção dos modelos 3D das cenas, a segmentação e o rastreamento dos objetos móveis em cada cena. Em seguida, descrevemos os dados utilizados e os testes realizados para as etapas de correção de cores e geração de novas visualizações. No primeiro teste, utilizamos imagens de múltiplas câmeras com diferenças visíveis de cor e brilho e aplicamos nosso algoritmo de correção de cores com o objetivo de obter imagens corrigidas com diferença mínima de cor e brilho em relação às imagens de referência. No segundo teste, utilizamos duas cenas diferentes para testar a geração de novas visualizações do ponto de vista de performance do método e da qualidade das novas visualizações produzidas. Finalmente, discutimos os resultados, os problemas e algumas possíveis soluções.

Um vídeo demonstrativo da aplicação do método pode ser encontrado no seguinte endereço: `http://www.ime.usp.br/~jefsilva/fvv.html`



(a) (b) (c) (d)

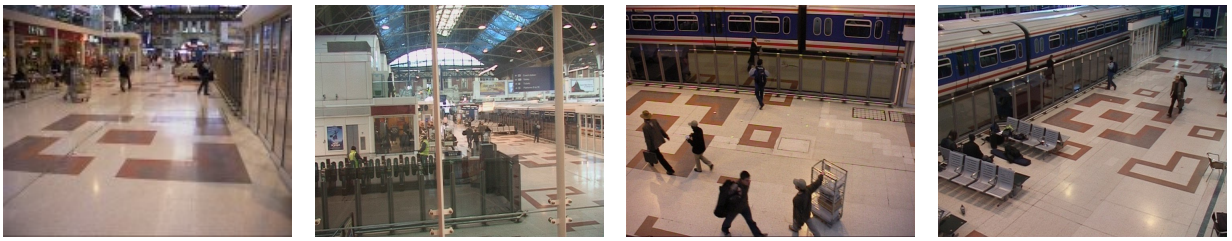
(e) (f) (g) (h)

Figura 4.1: Imagens dos dados do PETS 2009

4.1 Descrição dos dados

Para a realização dos testes de correção de cor e brilho, utilizamos imagens dos dados de benchmark do Workshop PETS 2009 [200b]. O objetivo do workshop era a utilização dos dados de benchmark para a estimação do número e da densidade de pessoas na cena em diversos cenários. Os dados foram obtidos utilizando-se 4 pares de câmeras onde em cada par o mesmo tipo de câmera foi utilizado. Nos dois primeiros pares foram utilizadas câmeras Axis 223M e PTZ Axis 233D com resolução de 768×576 e taxa de captura de aproximadamente 7 quadros por segundo. Nos dois pares restantes foram utilizadas câmeras Sony DCR-PC1000E 3xCMOS e Canon MV-1 1xCCD w com resolução de 720×576 também com taxa de captura de aproximadamente 7 quadros por segundo. A Figura 4.1 mostra as imagens de um mesmo instante do ponto de vista das 8 câmeras utilizadas.

Podemos notar que as imagens mostradas nas Figuras 4.1a, 4.1c, 4.1e e 4.1f apresentam diferenças perceptíveis de cor e brilho quando comparadas entre si e portanto caracterizam imagens adequadas para o teste do algoritmo de correção de cor e brilho.



(a) (b) (c) (d)

Figura 4.2: Imagens dos dados do PETS 2006

Para a realização dos testes de geração de novas visualizações, utilizamos duas cenas dinâmicas com a presença de objetos móveis e compostas por vários vídeos obtidos a partir de múltiplas câmeras estáticas dispostas esparsamente nas cenas.

A primeira cena consiste dos vídeos do Workshop PETS 2006 [200a]. Este workshop tinha como objetivo a detecção de bagagens abandonadas em um estação de metrô. Os dados são compostos por vídeos sincronizados de 2 pares de câmeras com pontos de vista distintos. As câmeras utilizadas no primeiro par de câmeras eram 2 Canon MV-1 1xCCD e no segundo par eram 2 Sony DCR-PC1000E 3xCMOS todas com resolução de 768×576 com taxa de captura de 25 quadros por segundo. A Figura 4.2 mostra as imagens da cena em um mesmo instante do ponto de vista de cada câmera. Para nosso testes, descartamos a câmera número 2 mostrada na Figura 4.2b porque ela apresenta pouca sobreposição do campo de visão com as outras 3 câmeras.

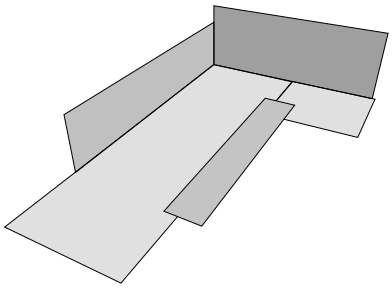
Para a segunda cena, utilizamos os dados coletados pelo Centro de Pesquisa de Automação(CfAR) da Universidade de Maryland utilizados em [SC08] com o objetivo de realizar a detecção e rastreamento de pessoas com o auxílio de múltiplas câmeras. Os dados consistem de vídeos sincronizados de 4 câmeras PTZ NTSC com resolução de 640×480 e com pontos de vista distintos do estacionamento em frente ao prédio A.V.Williams na Universidade de Maryland. A Figura 4.3 mostra as imagens da cena do ponto de vista de cada câmera.

Figura 4.3: Imagens da cena do A.V.Williams

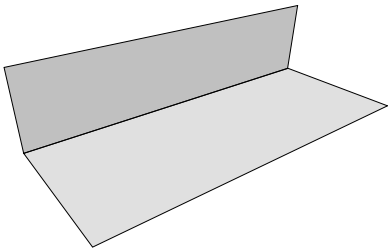
4.2 Construção do modelo, segmentação e rastreamento

Para gerar novas visualizações da cena, utilizamos uma abordagem híbrida de imagem e geometria o que exige que um modelo 3D simplificado de cada cena seja construído anteriormente. Construímos então os modelos 3D simplificados para as cenas do PETS 2006 e do estacionamento do A.V.Williams. Esses modelos foram construídos utilizando-se um conjunto de planos texturizados a partir de imagens obtidas anteriormente de cada cena. Cada um dos planos foi posicionado e dimensionado de acordo com medidas disponibilizadas pelos dados, como no caso do PETS 2006, ou obtidas a partir de marcações nas cenas no caso da cena do estacionamento do A.V.Williams. A Figura 4.4 mostra os modelos das duas cenas em versões com e sem texturas.

Além dos modelos 3D da cena também é necessário gerar as máscaras de segmentação e os dados do rastreamento dos objetos móveis de cada cena. Os dados de segmentação e rastreamento utilizados nos testes foram fornecidos por Thiago T. Santos e consistem de máscaras de segmentação para cada quadro dos vídeos das duas cenas. Além disso, para cada quadro os dados também forneciam a posição de cada objeto no plano do piso, uma estimativa da altura do objeto e um rótulo utilizado para identificar cada objeto. A Figura 4.5 mostra alguns exemplos das máscaras de segmentação resultantes para a cena do PETS 2006.

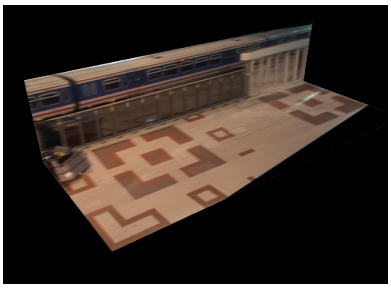


(a) Modelo do AVW sem texturas



(b) Modelo do PETS 2006 sem texturas

(c) Modelo do AVW texturizado



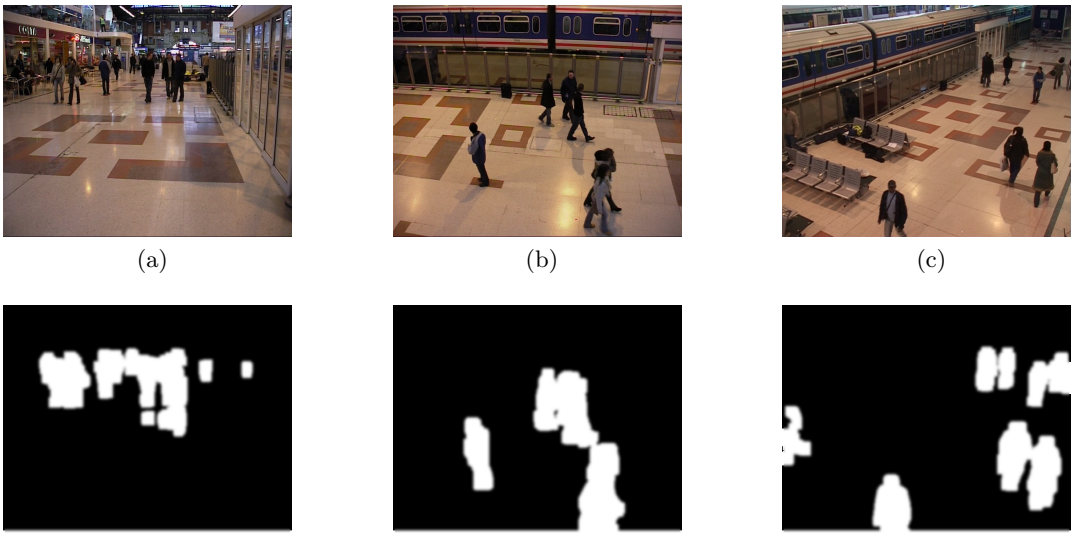
(d) Modelo do PETS 2006 texturizado

Figura 4.4: Modelos 3D das cenas.

4.3 Correção de cor e brilho

Para os testes de correção de cor e brilho, escolhemos algumas das imagens do PETS 2009 pois elas apresentam diferenças bastante aparentes de brilho e cor entre as imagens das câmeras. Como imagem de referência utilizamos a imagem da Figura 4.1a pois ela apresenta brilho e cores mais naturais. Escolhemos também as Figuras 4.1f, 4.1e e 4.1c para serem corrigidas em relação à imagem de referência. Aplicamos então o algoritmo de correção nas imagens a serem corrigidas fornecendo a região de sobreposição dessas imagens com a imagem de referência. Os resultados da aplicação do algoritmo podem ser vistos na Figura 4.6.

Podemos notar que as imagens corrigidas apresentam cor e brilho muito semelhante à imagem de referência e também entre si. A imagem da Figura 4.6a apresentava um deslocamento das cores



(a) (b) (c)

(d) (e) (f)

Figura 4.5: Máscaras de segmentação para a cena do PETS 2006

tendendo para o azul e após ser corrigida o deslocamento é neutralizado. A imagem da Figura 4.6d é um pouco escura e também apresenta um deslocamento para o azul. Após a correção a imagem se torna mais clara e as cores também se tornar mais corretas mas a cor do prédio ao fundo acaba se deslocando um pouco para o verde. A imagem da Figura 4.6g apresentava predominantemente uma diferença de brilho em relação à imagem de referência. Essa diferença também é praticamente neutralizada após a aplicação do algoritmo.

Também fizemos testes com as imagens da sequência do A.V.Williams onde as imagens dos vídeos diferem bastante das imagens utilizadas na texturização do modelo. O resultado aplicado na geração de novas visualizações pode ser visto comparando as Figuras 4.7b e 4.7d.

Para a sequência do PETS 2006, duas das câmeras já apresentam cor e brilho bastante semelhantes enquanto a câmera restante apresenta uma pequena diferença. Nesse caso específico a aplicação do algoritmo de correção não fornece resultados significativos devido a presença de especularidades e

também pelo tamanho da área de sobreposição com as outras câmeras. Cenas com grandes regiões de especularidades nas áreas de sobreposição prejudicam o algoritmo de correção pois as regiões podem apresentar distribuições de cores diferentes para pontos de vista distintos. Por esse motivo, o algoritmo não funciona corretamente nesses casos pois ele se baseia na suposição de que as distribuições das cores das regiões de sobreposição seja semelhante. Uma possível solução para esse problema seria a aplicação de técnicas de visão computacional para a detecção da cor da fonte de luz na cena para a aplicação de algoritmos de redução de especularidades antes da aplicação do algoritmo de correção.

Para testar o algoritmo de correção de cores do ponto de vista do desempenho, realizamos testes utilizando três métodos de correção. O primeiro método utiliza o mesmo espaço de estados do recozimento simulado já apresentado e consiste da utilização de busca exaustiva para encontrar um ótimo global. O segundo método consiste da utilização de busca exaustiva aplicando transformações de normalização de histograma e correção gama para minimizar a diferença entre os histogramas das áreas de sobreposição. Finalmente, o terceiro método consiste do método de recozimento simulado detalhado nesse trabalho. Devemos lembrar que os resultados dos testes independem do tamanho das imagens utilizadas já que todos os métodos trabalham apenas sobre histogramas de 8-bits para a geração das tabelas de correção.

Para o método de busca exaustiva, as tabelas de correção para uma única imagem são obtidas após um tempo de execução de 1414,2 minutos ou 23,57 horas. Para o método de correção por normalização de histogramas e correção gama, as tabelas são obtidas após 76,2 minutos ou 1,27 horas de execução. Finalmente, o método de recozimento simulado fornece os resultados para uma única imagem em 0,61 minutos ou 37 segundos.

Como podemos observar nos resultados, não é possível utilizar o algoritmo para calcular as tabelas de correção em tempo real. Isso seria necessário para a utilização do método para o tratamento de vídeos com muitas mudanças de iluminação.

(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figura 4.6: Resultados da correção de cor e brilho para imagens de teste com os parâmetros $T_0 = 256$, $K = 100$, $\epsilon = 10^{-5}$. A coluna da esquerda mostra as imagens a serem corrigidas, a coluna da direita mostra a imagem usada como referência para a correção e a coluna do meio mostra as imagens corrigidas.

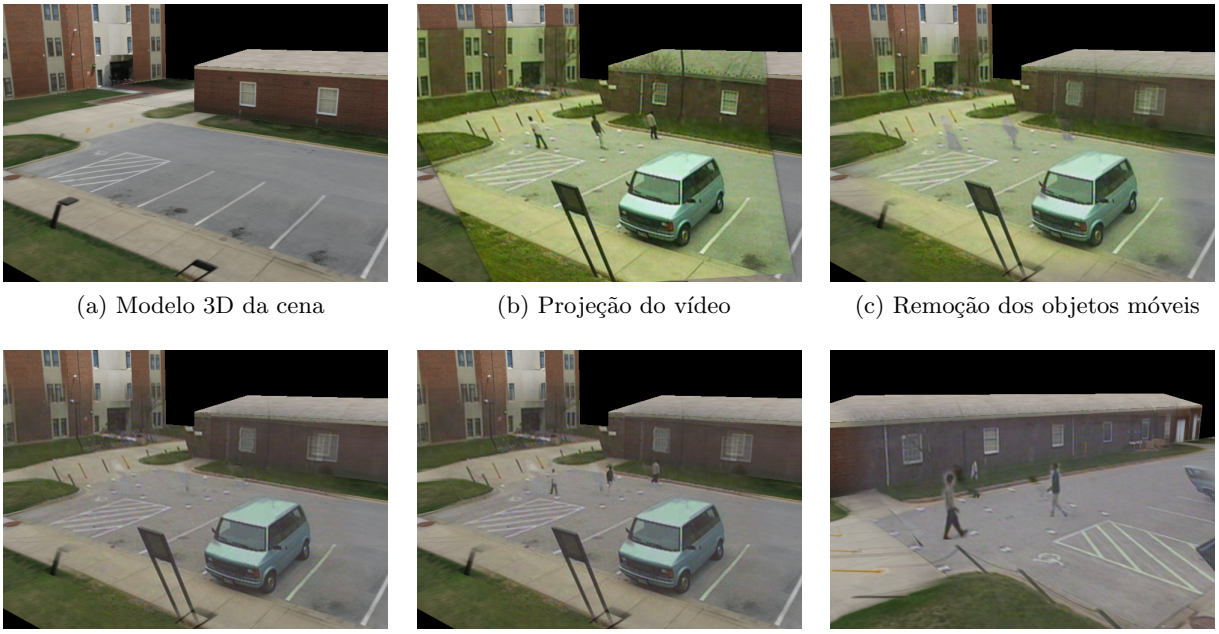
4.4 Geração de novas visualizações

Para testar o nosso método de renderização de novas visualizações, um sistema interativo foi implementado em C++ usando as bibliotecas OpenGL para a renderização e OpenCV para tratamento das sequências de vídeos e para o processamento de imagens. O computador utilizado para os testes possui um processador Intel Core 2 Duo 2.0Ghz, 3GB de RAM e uma placa de vídeo NVidia 8600M GT com 512MB de memória.

O sistema implementado é capaz de renderizar novas visualizações a uma taxa de quadros de aproximadamente 60 quadros por segundo. Os vídeos utilizados para os testes possuem ambos taxa de aproximadamente 24 quadros por segundo e portanto nosso sistema é capaz de renderizar novas visualizações em tempo real para as duas cenas.

Uma descrição passo-a-passo em figuras do processo de renderização pode ser visto na Figura 4.7. A Figura 4.7a mostra o começo do processo onde o modelo 3D da cena é renderizado a partir de ponto de vista escolhido pelo usuário. A Figura 4.7b mostra o segundo passo de renderização onde o vídeo, que melhor aproxima o novo ponto de vista, é projetado sobre o modelo. Podemos observar que a transição do vídeo projetado para o modelo é bastante aparente assim como a diferença de cores entre as texturas do modelo e o vídeo. Na Figura 4.7c, as transições do vídeo para o modelo foram suavizadas e os objetos móveis foram removidos. Os buracos deixados pelos objetos móveis são claramente visíveis na imagem. No próximo passo, aplicamos a transformação de calibração de cores e o resultado pode ser visto na Figura 4.7d. Agora a fusão da projeção do vídeo e dos buracos com o modelo é quase imperceptível. No passo final, os objetos móveis são renderizados como painéis texturizados e os resultados podem ser vistos na Figura 4.7e e na Figura 4.7f.

Os resultados da renderização para a sequência do PETS 2006 podem ser vistos na Figura 4.8. Na Figura 4.8a e na Figura 4.8b, podemos observar alguns artefatos de renderização devido a erros de segmentação. Na Figura 4.8c, podemos ver um outro tipo de artefato que fica aparente no grupo de



(a) Modelo 3D da cena (b) Projeção do vídeo (c) Remoção dos objetos móveis

(d) Calibração das cores (e) Painéis texturizados (f) Nova visualização

Figura 4.7: Passos da renderização da sequência do A.V.Williams

pessoas próximo à borda esquerda da imagem. Este artefato aparece durante a geração das texturas de cada painel quando o retângulo envolvente de um objeto é ocludido pelo retângulo envolvente de outro objeto. A textura resultante dos painéis para esses objetos irão compartilhar a região de sobreposição de seus retângulos envolventes resultando em partes desses objetos sendo replicadas na textura de ambos os painéis. Parte desse problema pode ser minimizada calculando os retângulos envolventes mínimos para cada objeto usando crescimento de regiões juntamente com a posição e a altura de cada objeto como referência. Também é possível detectar quando as oclusões entre os painéis ocorrem de modo a utilizar uma textura sem oclusão salva anteriormente ou um modelo alternativo para os objetos em oclusão.

Durante a reprodução do vídeo, o usuário pode alterar a posição da câmera e sua orientação e também a reprodução do vídeo escolhendo pausar, voltar e reproduzir a qualquer momento.

(a) Renderização 1 (b) Renderização 2 (c) Renderização 3

Figura 4.8: Novas visualizações da sequência do PETS 2006

A câmera $c = (\mathbf{p}, \mathbf{d})$, definida como na Figura 4.9, é controlada usando o mouse e o teclado em conjunto. As setas do teclado correspondem à translação de $\mathbf{p}$ em um plano Π perpendicular ao vetor $\mathbf{up}$ e que contém $\mathbf{p}$. A seta para frente sempre move a câmera na direção dada pelo vetor $\mathbf{d}$ e portanto a seta para trás move a câmera na direção contrária. As setas para os lados movem a câmera nas direções perpendiculares a $\mathbf{d}$ contidas em Π. O movimento no eixo X do mouse é equivalente a uma rotação de $\mathbf{d}$ em torno do vetor $\mathbf{up}$ e o movimento no eixo Y do mouse corresponde a uma rotação em torno do vetor dado pelo produto vetorial $\mathbf{d} \times \mathbf{up}$.

Outras possíveis comandos podem ser adicionados ao sistema, por exemplo, a seleção do ponto de vista de um objeto da cena ou a seleção de objetos a serem ocultados ou marcados em destaque. Também há possibilidade de definir objetos na cena com os quais o usuário pode interagir da mesma forma que um hipervídeo, sendo redirecionado para outras cenas ou para outros tipos de mídias.

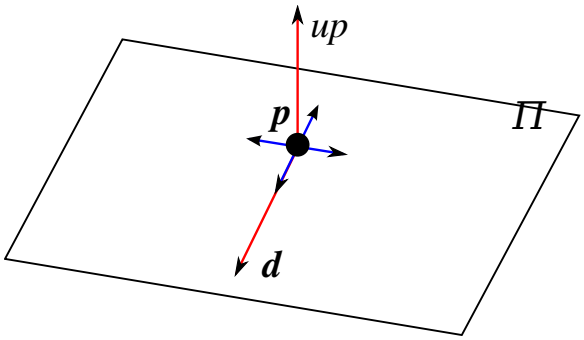


Figura 4.9: Controle da câmera virtual

Capítulo 5

Conclusão

As técnicas de renderização baseadas em imagens e vídeos permitem que novas visualizações de uma cena sejam geradas a partir de imagens e vídeos de múltiplas câmeras. Com o desenvolvimento das técnicas mais recentes e com os constantes avanços tecnológicos na área de hardware, essas técnicas já podem ser aplicadas para a renderização interativa de novas visualizações que podem ser aplicadas nas áreas de entretenimento com as televisões 3D digitais, na área de segurança para o monitoramento de ambientes e na área de realidade virtual em simulações de treinamento.

Neste trabalho apresentamos um método de renderização baseado em vídeos capaz de gerar novas visualizações de cenas dinâmicas, em tempo real, a partir de um ponto de vista arbitrário.

Para o desenvolvimento do método, combinamos resultados de várias áreas de pesquisa. Utilizamos resultados recentes e robustos para a realização da segmentação e rastreamento dos objetos móveis e desenvolvemos um algoritmo para a calibração de cores dos vídeos. Além disso, estendemos algumas técnicas de renderização baseada em imagens descritas no Capítulo 2 para que elas pudessem ser aplicadas em sequências de vídeos.

Nosso algoritmo para a calibração de cores é capaz de corrigir imagens de câmeras que não passaram por uma etapa de calibração de cores antes da aquisição das imagens. O algoritmo utiliza

as regiões de sobreposição das câmeras para encontrar transformações de histograma que podem ser aplicadas às imagens originais para minimizar a diferença entre os histogramas das áreas de sobreposição corrigindo consequentemente a cor e a iluminação das imagens.

Os resultados obtidos pela aplicação do nosso método para a geração das novas visualizações são satisfatórios do ponto de vista da performance e do foto-realismo.

Para obter esses resultados utilizamos resultados na área de segmentação e rastreamento para extrair os objetos móveis dos vídeos da cena e obter o posicionamento dos objetos móveis mesmo em situações de oclusão severa. Após a extração dos objetos móveis dos vídeos, preenchemos as regiões antes ocupadas por eles com detalhes da própria cena por meio do modelo 3D texturizado da cena, produzindo resultados melhores em relação à técnica original de texturização dependente da vista que preenche essas regiões pela interpolação das cores das áreas vizinhas.

O modelo 3D texturizado da cena é renderizado e retexturizado com a projeção dos vídeos que melhor aproximam o ponto de vista escolhido pelo usuário. Nessa etapa aplicamos as tabelas de conversão geradas pelo algoritmo de correção de cor para obter melhores resultados visualmente melhores na fusão dos vídeos. Finalmente, renderizamos os objetos móveis na cena utilizando painéis texturizados usando a técnica de texturização dependente da vista. Estes objetos são posicionados no modelo 3D da cena utilizando os dados obtidos pelo rastreamento.

Apesar de alguns artefatos aparentes devido à segmentação e a oclusão dos objetos nas texturas, o método fornece resultados com qualidade semelhante à dos vídeos da cena e permite que o usuário controle a velocidade de reprodução dos vídeos e também a câmera virtual.

Para trabalhos futuros, existe a possibilidade de melhorar a qualidade das renderizações obtendo melhores resultados na segmentação dos objetos móveis, reduzindo assim alguns dos artefatos presentes nas renderizações geradas pelo método. Uma outra possibilidade de melhoria dos resultados se encontra no tratamento das oclusões das texturas dos painéis. Algumas soluções possíveis incluem a geração de modelos para cada objeto, que seriam utilizados para renderizá-los quando estes sofres-

sem algum tipo de oclusão, ou em uma abordagem mais simples, a utilização de texturas de outros instantes onde o objeto não se encontra em oclusão. Outras melhorias incluem o desenvolvimento de uma interface gráfica para o usuário e o estudo de melhores maneiras de permitir que o usuário navegue pela cena. Também existe o interesse em se desenvolver uma interface que forneça funções para que o usuário interaja com os objetos da cena(ocultar, destacar, mover) e também a detecção de eventos de interação dos objetos entre si.

Referências Bibliográficas

[200a] PETS 2006. Pets 2006 benchmark data. http://pets2006.net/. 57

[200b] PETS 2009. Pets 2009 benchmark data. http://www.cvg.rdg.ac.uk/PETS2009/a.html. 56

[AB91] E.H. Adelson and J.R. Bergen. The plenoptic function and the elements of early vision. In M. Landy and J.A. Movshon, editors, *Computational models of visual processing*, page Chapter 1, The MIT Press, 1991. 8

[DTM96] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *Computer Graphics*, 30(Annual Conference Series):11–20, 1996. 8, 12, 22, 27

[DYB98] Paul E. Debevec, Yizhou Yu, and George Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. In George Drettakis and Nelson L. Max, editors, *Rendering Techniques*, pages 105–116. Springer, 1998. 4, 11, 38

[GGSC96] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54, New York, NY, USA, 1996. ACM. 8

[HZ00] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000. 9, 14

[JW95] R. Jain and K. Wakimoto. Multiple perspective interactive video. *Multimedia Computing and Systems, International Conference on*, 0:0202, 1995. 21

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983. 3, 34

[KRN97] Takeo Kanade, Peter Rander, and P. J. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE MultiMedia*, 4(1):34–47, – 1997. 13, 14, 22

[Lau94] Aldo Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Trans. Pattern Anal. Mach. Intell*, 16(2):150–162, 1994. 18

[LH96] Marc Levoy and Pat Hanrahan. Light field rendering. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42, New York, NY, USA, 1996. ACM. 8

[LMS03] Ming Li, Marcus Magnor, and Hans-Peter Seidel. Online accelerated rendering of visual hulls in real scenes. *Journal of WSCG*, 11(2):290–297, 2003. 18

[Mag05] Marcus Magnor. *Video-based Rendering*. A K Peters, Wellesley, USA, 2005. 1, 12

[MKKJ96] Saied Moezzi, Arun Katkere, Don Y. Kuramura, and Ramesh Jain. Immersive video. In *VRAIS '96: Proceedings of the 1996 Virtual Reality Annual International Symposium (VRAIS 96)*, page 17, Washington, DC, USA, 1996. IEEE Computer Society. 20

[OK93] M. Okutomi and Takeo Kanade. *A Multiple-Baseline Stereo*, 15(1):353–363, April 1993. 20

[OKK+07] Yuichi Ohta, Itaru Kitahara, Yoshinari Kameda, Hiroyuki Ishikawa, and Takayoshi Koyama. Live 3d video in soccer stadium. *Int. J. Comput. Vision*, 75(1):173–187, 2007. 20, 22

[San09] Thiago T. Santos. *Detecção e rastreamento de múltiplos objetos em condição de oclusão severa por meio de integração de suporte sob restrição homográfica.* PhD thesis, Universidade de São Paulo, 2009. 28, 29

[SBK+99] Hideo Saito, Shigeyuki Baba, Makoto Kimura, Sundar Vedula, and Takeo Kanade. Appearance-based virtual view generation of temporally-varying events from multi-camera images in the 3d room. In *Proceedings of Second International Conference on 3-D Digital Imaging and Modeling*, pages 516 – 525, October 1999. 20

[SC08] Aswin C. Sankaranarayanan and Rama Chellappa. Optimal multi-view fusion of object locations. In *WMVC '08: Proceedings of the 2008 IEEE Workshop on Motion and video Computing*, pages 1–8, Washington, DC, USA, 2008. IEEE Computer Society. 57

[SK99] S. M. Seitz and K. N. Kutulakos. A theory of shape by space carving. In *ICCV*, pages 307–314, 1999. 15, 16

[SK00] H.-Y. Shum and S. B. Kang. Review of image-based rendering techniques. In K. N. Ngan, T. Sikora, and M.-T. Sun, editors, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 4067 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 2–13, May 2000. 1, 7

[TLMS03] C. Theobalt, M. Li, M. Magnor, and H.-P. Seidel. A flexible and versatile studio for synchronized multi-view video recording. *Proc. IMA Vision, Video, and Graphics 2003 (VVG'03),* Bath, UK, July 2003. 18

[Tsa86] R. Y. Tsai. A efficient and accurate camera calibration technique for 3d machine vision. *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR 86)*, pages 364–374, 1986. 13

[WT91] Ling-Ling Wang and Wen-Hsiang Tsai. Camera calibration by vanishing lines for 3-D computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-13(4):370–376, April 1991. 9

[ZC04] C. Zhang and T. Chen. A survey on image-based rendering – representation, sampling, and compression. *Signal Processing: Image Communication*, 19(1):1–28, Jan 2004. 7

[Zha00] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell*, 22(11):1330–1334, 2000. 9, 13, 26

Índice Remissivo