BEATRIZ TRINCHÃO ANDRADE

UTILIZANDO FOTOGRAFIAS DIGITAIS DE ALTA QUALIDADE NA GERAÇÃO DE TEXTURA PARA MODELOS 3D: UMA ABORDAGEM PRÁTICA NA PRESERVAÇÃO DIGITAL DE ACERVOS CULTURAIS E NATURAIS

CURITIBA

2009

BEATRIZ TRINCHÃO ANDRADE

UTILIZANDO FOTOGRAFIAS DIGITAIS DE ALTA QUALIDADE NA GERAÇÃO DE TEXTURA PARA MODELOS 3D: UMA ABORDAGEM PRÁTICA NA PRESERVAÇÃO DIGITAL DE ACERVOS CULTURAIS E NATURAIS

Proposta de Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientadora: Profa. Dra. Olga Regina Pereira Bellon
Orientador: Prof. Dr. Luciano Silva

CURITIBA

2009

AGRADECIMENTOS

Inicialmente quero agradecer aos professores Olga Bellon e Luciano Silva, pela confiança durante o desenvolvimento deste trabalho, pela orientação e pela amizade. Agradeço também a chance que me proporcionaram durante este mestrado, de trabalhar em uma área que gosto, em um ambiente excelente.

À CAPES, pelo financiamento deste mestrado.

Ao amigo Alexandre Vrubel, pelas idéias e colaboração essenciais para o desenvolvimento deste trabalho.

Ao Museu de Arqueologia e Etnologia da UFPR e ao Museu de Ciências Naturais da UFPR, pelo acesso às peças dos seus acervos.

Aos alunos do Grupo Imago, pela amizade, receptividade, e disposição em ajudar sempre que me foi necessário.

Ao meu noivo André, pelo amor, compreensão e companheirismo que tem mostrado não só durante este mestrado, mas durante todo o nosso relacionamento.

Aos meus pais, por tudo que fizeram pela minha formação, e pelo apoio e exemplo que têm me dado durante toda a minha vida.

CONTEÚDO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A Preservação Digital 3D é uma área da Computação Gráfica que visa gerar modelos tridimensionais virtuais de objetos que possuem valor cultural ou científico. A preservação digital possibilita a visualização realística do objeto através de museus virtuais ou aplicações científicas; e a restauração do objeto preservado, em caso de desgaste natural ou acidentes. Nesta área, a representação detalhada das características do objeto é essencial, visto que armazena informações importantes sobre o objeto preservado.

Neste contexto, este trabalho apresenta um estudo sobre a geração de textura para modelos tridimensionais. Nele, é feita uma revisão sobre a modelagem da geometria e da fotometria, e é desenvolvido um algoritmo para preservar a aparência do objeto original através do uso de fotografias de alta resolução na geração de textura para o modelo 3D.

Os modelos 3D renderizados com as texturas obtidas através do processo desenvolvido neste trabalho são exibidos em um museu virtual. Entre os patrimônios digitalizados estão artefatos indígenas pertencentes ao acervo do Museu de Arqueologia e Etnologia da UFPR, e conchas e fósseis pertencentes ao Museu de Ciências Naturais da UFPR.

O algoritmo desenvolvido calcula a textura de um objeto a partir do seu modelo 3D e um conjunto de imagens obtidas por um *scanner* a *laser* e uma câmera fotográfica de alta resolução. O método desenvolvido gera texturas de alta qualidade, aumentando substancialmente o realismo do modelo 3D em comparação com texturas geradas apenas por imagens do scanner. Ele também não requer nenhum aparato especial ou um grande número de fotografias coloridas, simplificando seu uso por outros pesquisadores.

ABSTRACT

3D Digital Preservation aims to generate tridimensional virtual models of objects that have cultural or scientific value. The digital preservation provides realistic visualization of objects through virtual museums or scientific applications; and the preserved object's restoration, in case of natural wear or accidents. In this area, the detailed representation of the object characteristics is essential since they store important information about the preserved object.

In this context, this work presents a study about texture generation for 3D models of objects. It makes a review about the geometric and photometric modeling, and develops an algorithm to preserve the appearance of the original object, through the use of high resolution photographs during the 3D models texture generation.

The 3D models using the generated texture are exhibited in a 3D virtual museum. Amongst the preserved assets, there are Brazilian Indigenous artworks, from UFPR's Archaeology and Ethnology Museum, and shells and fossils from UFPR's Natural Science Museum.

The developed algorithm calculates the texture of an object from its 3D model and a set of images obtained from a laser scanner and a high resolution photographic camera. It generates high quality textures, improving substantially the realism of the 3D model as compared to textures generated only from scanner images. Also, it does not require any special apparatus nor a large number of color images, simplifying its use by other researchers.

CAPÍTULO 1

INTRODUÇÃO

A construção e visualização de modelos tridimensionais de objetos é um ramo bastante pesquisado na Computação Gráfica devido à grande quantidade de aplicações existentes. Uma aplicação de grande importância nesta área é a Preservação Digital Tridimensional (3D), que visa gerar modelos tridimensionais virtuais de objetos que possuem valor cultural ou científico.

Patrimônios culturais e naturais tendem a sofrer desgastes com o tempo ou ser destruídos, devido à deterioração, desastres e acidentes. Com a digitalização, são gerados modelos tridimensionais dos objetos que permitem a sua preservação para gerações futuras. Estes modelos também podem ser disponibilizados em museus virtuais, utilizados em pesquisa, ou modificados, para o planejamento da restauração dos objetos reais. Como exemplos de trabalhos relevantes nesta área destacam-se o Projeto Michelangelo Digital [18] e o Projeto Grande Buda [14].

O Michelangelo Digital foi um projeto pioneiro na preservação digital 3D de obras de grande porte. Nele foi apresentado o processo de digitalização de algumas das obras do artista Michelangelo, como a escultura Davi. A proposta deste projeto é a disponibilização das obras, possibilitando que usuários visualizem as esculturas do artista sem a necessidade de estar no local, e manipulá-las virtualmente, obtendo assim um grau de liberdade que não é possível nas obras originais.

No Projeto Grande Buda foram digitalizadas obras importantes para a cultura japonesa, onde se destacam a estátua do Buda de Kamakura, no Japão, e as faces do Templo Bayon, no Camboja. É importante ressaltar que neste projeto os modelos 3D gerados foram utilizados na obtenção de informações inéditas sobre as obras, como a quantidade

de ouro utilizada em uma estátua, e para a simulação da aparência original de obras que foram modificadas ao longo dos séculos.

No que concerne ao processo de digitalização e reconstrução, dois problemas se destacam: a modelagem geométrica e a modelagem fotométrica de objetos [30]. A primeira visa à reprodução da forma do objeto e seus detalhes. A modelagem fotométrica, por sua vez, tem como objetivo representar a aparência da superfície do objeto.

Ambos os problemas são importantes, e essenciais para a construção e visualização realística do modelo digital. No entanto, ao passo em que há uma grande variedade de estudos sobre a modelagem geométrica [3, 14, 25], a modelagem fotométrica [24, 30] é ainda uma área pouco explorada no que se refere a abordagens práticas.

Neste contexto, esta dissertação faz um estudo sobre as técnicas de modelagem fotométrica de maior destaque na literatura. A partir deste estudo, é desenvolvido um método de geração de texturas para modelos 3D. Esse método se baseia na adição de uma câmera fotográfica profissional ao sistema de aquisição de imagens, e gera textura para o modelo 3D utilizando as fotografias capturadas. Estas fotografias possuem alta resolução, e permitem a geração de texturas que preservam a aparência do objeto com fidelidade de cor e maior nível de detalhes.

O trabalho desenvolvido é uma continuação do projeto iniciado em Vrubel [33], onde é desenvolvido um pipeline para geração de modelos 3D com ênfase na modelagem geométrica dos objetos. O método de geração de texturas desenvolvido nesta dissertação é adicionado ao pipeline existente, possibilitando a geração de modelos com textura de alta qualidade. As réplicas digitais obtidas são disponibilizadas em um museu virtual, o Museu Virtual 3D [20].

Entre os objetos preservados nesta dissertação estão peças que possuem valor cultural e científico, onde se destacam artefatos indígenas pertencentes ao acervo do Museu de Arqueologia e Etnologia da UFPR, e conchas e fósseis de animais primitivos, pertencentes ao Museu de Ciências Naturais da UFPR. Os modelos tridimensionais obtidos a partir

destas peças, gerados com texturas de alta fidelidade, estão disponíveis no Museu Virtual 3D do IMAGO[1].

Esta dissertação está organizada conforme descrito a seguir. O Capítulo 2 introduz o processo de modelagem geométrica, descrevendo os algoritmos mais utilizados na geração de modelos tridimensionais. No Capítulo 3 é apresentado o processo de modelagem fotométrica e seu estado da arte. O método para geração de texturas desenvolvido é introduzido no Capítulo 4, e os resultados são descritos no Capítulo 5. Por fim, no Capítulo 6 são apresentadas as conclusões e os próximos passos deste trabalho.

[1]Museu Virtual 3D do IMAGO: http://www.inf.ufpr.br/imago/pt_museu3d.html.

CAPÍTULO 2

MODELAGEM GEOMÉTRICA

A modelagem geométrica é o processo que visa reproduzir a forma do objeto e seus detalhes. Na Preservação Digital 3D, este processo é iniciado com a aquisição de dados sobre o objeto a ser preservado, e termina com a geração do seu modelo tridimensional.

Existem diferentes formas de realizar esta operação. Neste trabalho, as etapas escolhidas são as utilizadas no processo desenvolvido em [33], aplicado na geração dos modelos 3D apresentados nesta dissertação. Neste processo, a geração dos modelos é dividida em cinco etapas: aquisição, alinhamento, integração, preenchimento de buracos, e geração do modelo.

Neste capítulo serão apresentados métodos clássicos utilizados em cada uma destas etapas, e é descrito como o processo desenvolvido em [33] implementa cada uma delas. As etapas de aquisição, alinhamento, integração, preenchimento de buracos, e geração do modelo são detalhadas nas Seções 2.1, 2.2, 2.3, 2.4 e 2.5, respectivamente. A metodologia definida em [33] para cada uma das etapas é descrita na Seção 2.6.

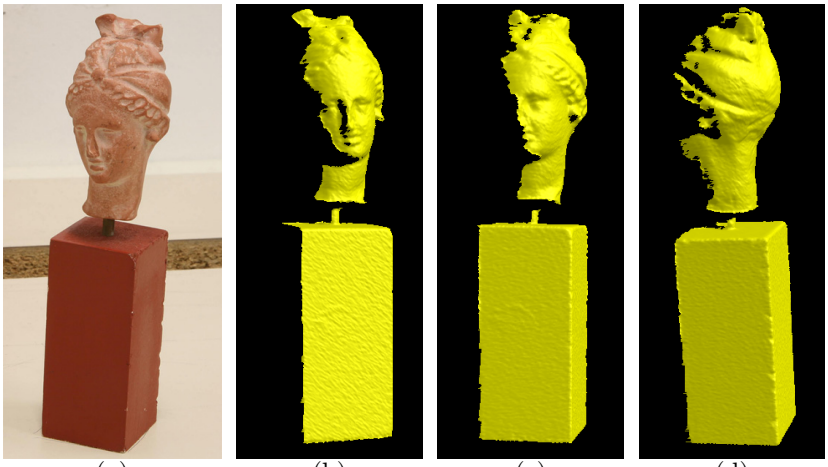
2.1 Aquisição

A etapa de aquisição consiste na amostragem de medidas de profundidade de um objeto ou superfície. Durante a aquisição de dados, são capturadas informações sobre a geometria do objeto através de diferentes tipos de técnicas, como ressonância magnética, escaneamento a *laser*, imageamento estéreo, e iluminação estruturada.

Neste trabalho é dado foco ao escaneamento a laser. Nesta técnica são capturadas imagens de profundidade, que são medidas de profundidade dispostas em uma grade

de amostragem regular. Essas imagens são obtidas a partir da varredura de um sensor sobre a superfície do objeto e do ambiente onde ele está inserido. Na Figura 2.1, são exibidas renderizações de imagens de profundidade correspondentes a regiões do objeto da Figura 2.1(a).

Como pode ser observado, uma única imagem de profundidade não contém informação suficiente para reconstruir integralmente o objeto que será digitalizado [8]. Assim, dado um conjunto de imagens de profundidade que registram diferentes regiões de um objeto, é possível integrá-lo, de forma a gerar um modelo tridimensional.



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

Figura 2.1: Exemplo de imagens de profundidade de um objeto: (a) fotografia do objeto; (b), (c) e (d) imagens de profundidade do objeto, capturadas através de diferentes perspectivas.

2.2 Alinhamento

Em dispositivos de aquisição que capturam imagens de profundidade, é necessário que sejam obtidas imagens de profundidade do objeto sob diferentes perspectivas (vistas), de forma a obter dados sobre a sua superfície. Quando duas vistas digitalizam regiões em comum do objeto, e estas podem ser total ou parcialmente sobrepostas dadas uma posição e uma orientação apropriadas, diz-se que elas podem ser registradas. Na etapa de alinhamento é feito o cálculo das transformações que efetuam o registro entre diferentes vistas, de forma a criar uma representação tridimensional do objeto digitalizado.

Para evitar mínimos locais, os métodos de alinhamento de um modo geral supõem que as imagens de profundidade estão inicialmente alinhadas de forma aproximada. Na prática, pré-alinhamento pode ser feito de forma manual [33], através do uso de heurísticas [22], ou através de instrumentação, para calibrar o movimento do objeto perante o *scanner*, ou do *scanner* em relação ao objeto [18].

Na etapa de alinhamento, destacam-se soluções que utilizam algoritmos iterativos de pontos mais próximos (*Iterative Closest Point - ICP*), por serem as mais utilizadas. A idéia central do ICP é, a partir de duas malhas de pontos tridimensionais e uma estimativa inicial para a transformação relativa entre elas, refinar iterativamente a transformação através da geração de pares de pontos correspondentes nas malhas, minimizando uma métrica de erro.

Besl e McKay [4] introduziram um algoritmo ICP que se tornou uma das principais técnicas utilizadas para o registro de conjuntos de dados 3D. Ele consiste em uma técnica genérica de registro que trabalha sobre conjuntos de pontos correspondentes amostrados, minimizando a distância entre cada ponto da primeira superfície e o ponto mais próximo da segunda. Este processo é ilustrado na Figura 2.2, onde uma superfície P é alinhada iterativamente a uma superfície Q até obter a configuração P_n que efetua o alinhamento entre as duas.

Em Chen e Medioni é apresentada uma outra abordagem para efetuar o registro de imagens [7]. Nela, ao invés de utilizarem a distância entre pontos correspondentes em duas superfícies para efetuar o alinhamento, é usada a distância entre um ponto em uma superfície, e a reta tangente ao ponto correspondente na outra.

Devido ao fato de a função de distância ser calculada através de uma distância ponto-a-reta, e não ponto-a-ponto, a convergência se torna mais rápida. Isso ocorre pois o cálculo da distância ponto-a-ponto trabalha com conjuntos de pontos correspondentes, que impõem limitações durante o registro da imagem. Como estes conjuntos nem sempre são corretos, o algoritmo demora a convergir. No caso da distância ponto-a-reta, somente é limitada a direção na qual a distância pode ser reduzida. O ponto correspondente pode

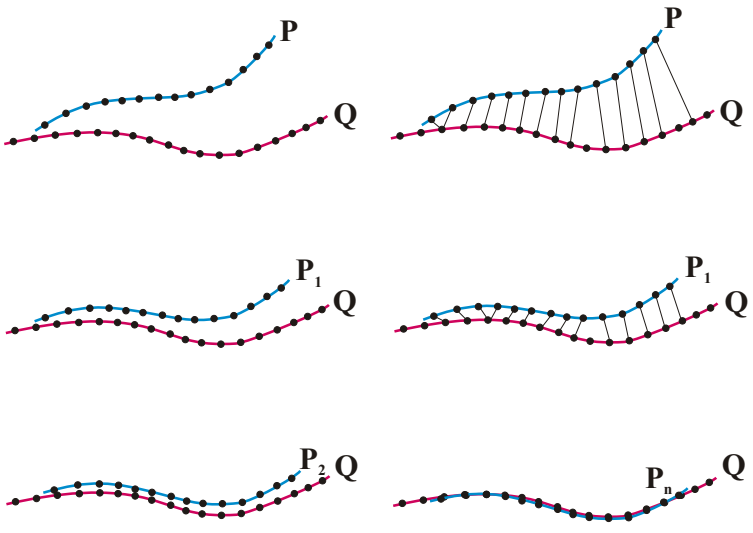


Figura 2.2: Processo iterativo de alinhamento entre duas superfícies P e Q.

então ser escolhido com maior liberdade, podendo se mover ao longo da reta tangente, de acordo com as limitações que possam ocorrer durante o registro, acelerando assim a convergência do algoritmo.

Posteriormente, em Pulli [23], é apresentado um método que possibilita o registro simultâneo de várias imagens de profundidade. A motivação para esta implementação é que, através do registro simultâneo, é possível evitar problemas que ocorrem nos processos iterativos de alinhamento, como a integração incorreta de novas vistas à representação 3D do objeto.

No entanto, este tipo de registro tem um custo computacional elevado, pois entre outros problemas, exige que todas as imagens de profundidade estejam carregadas na memória para que seja efetuado. Quando esta operação é aplicada em conjuntos muito grandes de dados, o problema se agrava. Assim, Pulli apresenta neste mesmo trabalho uma solução que faz o registro simultâneo baseado em restrições obtidas a partir de uma etapa de alinhamento entre pares efetuada previamente.

2.3 Integração

O objetivo desta etapa é reconstruir superfícies a partir de grupos de imagens de profundidade alinhadas, de forma que o processamento de um conjunto de imagens de profundidade resulte em um único objeto volumétrico. Com o surgimento do algoritmo *Marching-Cubes* [19], o problema da geração da representação volumétrica foi extremamente simplificado. Isso se deve ao fato de que, com este algoritmo, o problema principal durante a integração passou a ser a forma como é calculada a distância de um ponto a uma superfície do objeto. Assim, os métodos para integração de imagens de profundidade passaram a focar nesta questão.

A aproximação inicial, que seria o cálculo da distância entre um ponto e um triângulo da superfície mais próxima não é efetiva, por ser extremamente susceptível a erros na presença de ruídos. Assim, existem soluções alternativas para o cálculo da função de distância sinalizada que visam à escolha de uma superfície ideal, que possui maior probabilidade de pertencer ao objeto.

Em Curless e Levoy [8], é realizado o cálculo da função sinalizada de distância através da integração de distâncias estimadas em cada voxel. Cada imagem de profundidade é analisada separadamente, e é percorrida a linha de visão para cada ponto da imagem de profundidade, integrando a função de distância sinalizada para cada voxel que a linha atravessa. A distância final é estimada como uma média ponderada destas estimativas.

Posteriormente, em Wheeler *et al.* [34], foi desenvolvido um algoritmo para calcular a função de distância entre um ponto arbitrário e N superfícies trianguladas de visões do objeto através da estimação local da superfície, fazendo uma média entre observações da mesma superfície.

Para isso, foi desenvolvido um método que identifica e coleta todas as observações similares. Neste método, observações próximas são comparadas através da localização e da normal de suas superfícies. Se a normal e a localização estiverem dentro de uma tolerância, as observações são consideradas como de uma mesma superfície. Caso um número insufi-

ciente de observações de uma mesma superfície seja encontrado, essa superfície pode ser descartada como isolada ou não confiável. Assim, para que uma superfície seja considerada, ela deve possuir um dado quorum de observações.

2.4 Preenchimento de Buracos e Espaços

Para capturar toda a superfície de um modelo, este deve ser digitalizado pelo sensor a partir de vários pontos de vista. Se o objeto possuir reentrâncias muito fundas ou regiões inacessíveis, não será possível fazer a leitura da área, o que resultará em varreduras com oclusões que conseqüentemente produzirão modelos tridimensionais com buracos.

Em alguns casos, como na pesquisa por informações sobre o objeto preservado, por exemplo, um modelo com buracos pode ser utilizado, devido ao fato de conter somente informações extraídas originalmente da sua superfície. Porém, a maioria das aplicações requer a construção de modelos sem buracos, que limitem um volume no espaço. Exemplos de tais aplicações incluem a computação de propriedades físicas, a fabricação de réplicas, e a apresentação de modelos em escolas ou museus, onde buracos são esteticamente desagradáveis e confusos [9]. Assim, muitas vezes é necessário que sejam aplicados algoritmos que preencham os buracos e espaços existentes.

Em [8], é apresentada uma técnica denominada *space carving* (esculpindo no espaço), que é aplicada dentro do processo de geração de modelos descrito no mesmo trabalho. A técnica é aplicada sobre o volume, que, de acordo com os autores, contém mais informações do que a malha reconstruída.

A idéia central do algoritmo é classificar todos os pontos no volume como: não vistos; vazios; ou próximos da superfície. Voxels próximos da superfície são os voxels que ficam dentro de um limiar interno ou externo em relação à superfície; voxels vazios são voxels que estão nas linhas de visão que vão da superfície observada até a origem; e voxels não vistos são os que não foram classificados no espaço volumétrico. Para que possam ser identificados mais voxels vazios, o artigo sugere o uso de planos de fundo atrás das

superfícies digitalizadas. Após a classificação, buracos na superfície seriam então indicados através de fronteiras entre regiões de pontos não vistos e vazios. Assim, o algoritmo insere superfícies nestas regiões, tapando os buracos.

Posteriormente, Davis *et al.* [9] utilizam o processo de difusão para preencher buracos em representações volumétricas de uma superfície. O processo de difusão consiste em passos alternados de borramento e composição. O objetivo do algoritmo é estender o domínio da função de distância d_s para uma função d que seja definida sobre todo o volume. O cálculo de d é conseguido através da difusão do valor de d_s, partindo de superfícies observadas em direção a áreas indefinidas. Ao passo que o domínio da função aumenta, também cresce o conjunto zero de d_s (superfície). O processo de difusão é aplicado então sobre os buracos, passando sobre eles. O processo se repete até que os buracos estejam fechados, e que as mudanças na superfície causadas pela difusão sejam menores que o ruído inerente ao *scanner*. Uma vez que o processo de difusão se completa, o conjunto zero de d_s é a superfície sem buracos.

Sagawa e Ikeuchi [27] propõem um processo que também gera a função de distância sinalizada utilizando múltiplas imagens de profundidade. É requerido, porém, que os valores da função contenham as distâncias Euclidianas sinalizadas de um voxel ao ponto mais próximo no modelo da malha. Como os sinais das funções de distância tornam-se instáveis quando próximos a buracos ou espaços, é desenvolvido no artigo um método para interpolar buracos em imagens de profundidade através da tomada de um consenso entre o sinal da função de distância e os sinais de voxels próximos. É assumido no artigo que, uma vez que as funções de distância sinalizadas são consistentes, os buracos ou espaços estão preenchidos com eficiência.

2.5 Geração do Modelo

Esta etapa abrange a geração do modelo tridimensional a partir do objeto volumétrico. Nela se destaca o algoritmo *Marching-Cubes* [19], que gera superfícies para modelos 3D

a partir de uma função de distância de um objeto volumétrico. Este algoritmo foi criado originalmente para a construção de modelos 3D a partir de dados médicos bidimensionais (obtidos em exames de Tomografia Computadorizada, Ressonância Magnética, etc.), mas devido à sua portabilidade, pode ser adaptado para qualquer aplicação que forneça um objeto volumétrico representado por uma função de distância.

Ao contrário dos algoritmos de geração de modelos existentes até então, que calculavam um valor binário em cada voxel para indicar se o voxel está vazio ou não, o algoritmo *Marching-Cubes* requer que os dados na grade de volume representem uma superfície implícita. Assim, em cada voxel é guardada a distância $f(x)$ que vai do centro do voxel até o ponto na superfície mais próxima. O sinal de $f(x)$ indica se o ponto está fora $(f(x) > 0)$, dentro $(f(x) < 0)$, ou sobre a superfície $(f(x) = 0)$. O *Marching-Cubes* [34] então constrói a superfície através de uma "marcha"em volta dos cubos enquanto segue os cruzamentos de zero na superfície implícita $f(x) = 0$.

O algoritmo *Marching-Cubes* e a representação de superfície implícita provêem uma alternativa atraente em relação aos outros esquemas de geração de modelos. Isso se deve ao fato deste algoritmo eliminar o problema de topologia sobre como várias superfícies são conectadas para que possam ser integradas, e apresentar uma representação que possibilita a modelagem de objetos de topologia arbitrária enquanto a amostragem da grade for suficiente para capturar a topologia. Por fim, o motivo mais importante é a redução do problema de criar uma representação volumétrica para a questão do que é a distância sinalizada entre um dado ponto e uma superfície [34].

2.6 Método de Geração de Modelos Geométricos Utilizado

Para gerar o modelo geométrico de objetos, Vrubel [33] inicialmente captura imagens de profundidade do objeto utilizando um scanner a laser. As vistas são pré-alinhadas manualmente, e para cada par com sobreposição suficiente, é feito o registro utilizando uma versão modificada do algoritmo ICP. A seguir, é executada uma etapa de registro

global, onde é utilizado o algoritmo de Pulli [23].

A versão modificada do ICP possui duas fases. A primeira utiliza uma variante do ICP que possui a métrica de erro baseada na distância ponto-a-plano, com características que levam a uma boa convergência, porém com precisão limitada. Quando esta fase converge, é iniciada a segunda fase. Nela, são utilizados todos os pontos de ambas as vistas, adicionando uma distância máxima no teste de compatibilidade dos pares. Essa distância é relacionada ao erro do scanner, e geralmente é muito pequena (*e.g.* em torno de 0.7mm). Durante a minimização do erro, é utilizada a métrica de erro de ponto-a-plano. Essa versão do ICP tem convergência limitada, mas apresenta boa precisão. Como a primeira fase alcança um alinhamento quase ótimo, esta fase apenas melhora a precisão do resultado.

Após o alinhamento, as malhas são integradas com o objetivo de construir uma única malha de triângulos para o objeto. Em [33] foi desenvolvida uma abordagem volumétrica, que combina elementos dos métodos apresentados por Curless e Levoy [8] e Wheeler *et al.* [34]. O algoritmo desenvolvido possui duas fases. Na primeira, é utilizada uma versão modificada do método de Curless e Levoy em conjunto com o método de *space carving* para gerar a representação volumétrica inicial. A modificação do método de Curless e Levoy consiste em uma nova curva de pesos, que dá valores mais altos para voxels que estão fora do objeto. O método de *space carving* desenvolvido leva em consideração apenas dados sobre o objeto, e opcionalmente sobre os planos de suporte detectados no estágio de aquisição, tendo como objetivo principal a eliminação de *outliers*.

O resultado volumétrico desta fase funciona como uma base consensual para a segunda fase do algoritmo, que constrói a representação volumétrica definitiva integrando apenas as medidas em consenso com o resultado obtido na primeira fase. O consenso é testado em cada voxel candidato, entre a normal no ponto mais próximo da superfície de cada vista e o gradiente do resultado volumétrico da primeira fase. O *space carving* efetuado na primeira fase é também utilizado na remoção dos dados incorretos na região externa ao objeto.

Por fim, foi escolhido o algoritmo de difusão criado por Davis *et al.* [9] para preencher os buracos da superfície, e o algoritmo *Marching Cubes* [19] para gerar a malha de triângulos a partir da representação volumétrica obtida nos estágios anteriores.

CAPÍTULO 3

MODELAGEM FOTOMÉTRICA

Para representar com fidelidade as características originais de um objeto, um modelo 3D para aplicações em Preservação Digital deve conter informações sobre a forma e aparência do objeto preservado. Quando são consideradas as propriedades fotométricas, o objetivo é determinar os parâmetros que indicarão a aparência do objeto sob diferentes tipos de iluminação.

Neste trabalho, o processo de modelagem fotométrica será direcionado para modelos 3D obtidos a partir de imagens de profundidade capturadas por *scanners* a *laser*. Além das imagens de profundidade, é assumido que o *scanner* também obtém imagens coloridas a partir do mesmo dispositivo óptico. Assim, é possível mapear diretamente pontos 2D das imagens coloridas a pontos 3D das imagens de profundidade correspondentes.

O scanner a laser é atualmente o dispositivo que possibilita a captura de pontos tridimensionais com maior resolução [21]. A resolução dos scanners atuais é suficiente para capturar a geometria de objetos com um bom nível de detalhes (*e.g.* 640×480). No entanto, ela não é suficiente para a captura de informação adequada para o cálculo de texturas realísticas. Além disso, as imagens coloridas do scanner apresentam baixa fidelidade de cor, o que prejudica a reprodução das propriedades da superfície dos objetos.

Nesta situação, a exemplo de trabalhos como [3, 14, 16], é adicionada uma câmera fotográfica ao sistema de captura, obtendo assim imagens coloridas de alta resolução do objeto. Nesta dissertação, estas imagens são chamadas `imagens da câmera`, ao passo em que as imagens de profundidade e coloridas capturadas pelo scanner são denominadas `imagens de profundidade` e `imagens do scanner`, respectivamente.

O problema decorrente do uso da câmera profissional é que as `imagens da câmera`

estão em sistemas de coordenadas diferentes das `imagens de profundidade`. Assim, não é possível descobrir diretamente a localização de pontos 3D das `imagens de profundidade` nas `imagens da câmera`.

Como as `imagens do scanner` são mapeadas às `imagens de profundidade`, pode-se resolver este problema a partir da associação de características presentes nas `imagens do scanner` e nas `imagens da câmera`. A partir desta associação, é possível mapear pontos 3D nas `imagens de profundidade` a pontos 2D nas `imagens da câmera`, e com isso calcular a equação que efetua a transformação entre os sistemas de coordenadas da câmera e do scanner.

Assim, foram estudadas técnicas de registro que visam à obtenção da equação que faz a transformação entre os sistemas de coordenadas da câmera e do scanner. A calibração e o alinhamento por características, duas formas de efetuar o registro, são discutidas na Seção 3.1.

Após o registro é possível saber, para cada vértice do modelo, o subconjunto de `imagens da câmera` onde ele aparece, e sua cor em cada uma delas. Com o conjunto de cores observadas para cada vértice e informações sobre a iluminação do ambiente, é possível a formulação de parâmetros de refletância. Nesta fase são utilizadas informações sobre a iluminação, geometria, e cores de cada ponto do objeto em diferentes imagens. Algumas soluções existentes na literatura para este problema são apresentadas na Seção 3.2.

3.1 Registro de Imagens

3.1.1 Calibração

O problema da calibração consiste em calcular a matriz que efetua a transformação entre sistemas de coordenadas da imagem e de uma cena. Como cada pixel é registrado na imagem de acordo com a projeção da perspectiva, ele corresponde a uma linha de pontos na cena. Assim, a calibração consiste em determinar a equação para esta linha no sistema

de coordenadas absolutas da cena [15].

No contexto deste trabalho, a cena considerada no problema da calibração será o conjunto de pontos tridimensionais obtidos através do *scanner*. Assim, a calibração é utilizada para efetuar a correspondência entre pontos bidimensionais, capturados pela câmera, e pontos tridimensionais obtidos pelo *scanner*.

O processo de calibração deve ser refeito a cada alteração no posicionamento relativo entre a câmera e o *scanner*. Ele consiste na captura de imagens de um objeto com geometria conhecida a partir dos dois dispositivos. São então obtidas as localizações de pontos conhecidos do objeto em cada uma das imagens, gerando assim um conjunto de pares de pontos correspondentes. Os pontos obtidos pela câmera são bidimensionais, ao passo que no *scanner* são utilizados os pontos tridimensionais, armazenados nas `imagens de profundidade`.

Com este conjunto de pontos é então calculada a matriz de transformação que efetuará a calibração (matriz de calibração). A primeira aproximação para este problema é considerar uma matriz de calibração de tamanho 3×4, e resolver um sistema linear que utiliza no mínimo cinco pares e meio de pontos correspondentes [1].

Para isso, sejam as coordenadas homogêneas de um dado ponto da imagem $(U, V) = (u, v, t)$. Desta maneira:

$$U = \frac{u}{t} \quad e$$
$$V = \frac{v}{t}. \tag{3.1}$$

Seja também a matriz de calibração C, com elementos C_{ij} e colunas C_j. Assim, para todo ponto (x, y, z) da cena, é necessário uma matriz C tal que:

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} C = \begin{bmatrix} u & v & t \end{bmatrix}.$$

Assim:

$$u = \begin{bmatrix} x & y & z & 1 \end{bmatrix} C_1,$$
$$v = \begin{bmatrix} x & y & z & 1 \end{bmatrix} C_2,$$
$$t = \begin{bmatrix} x & y & z & 1 \end{bmatrix} C_3. \tag{3.2}$$

Calculando os produtos internos e reescrevendo com $u - Ut = 0$ e $v - Vt = 0$, temos:

$$xC_{11} + yC_{21} + zC_{31} + C_{41} - UxC_{13} - UyC_{23} - UzC_{33} - UC_{43} = 0 \quad e$$
$$xC_{12} + yC_{22} + zC_{32} + C_{42} - VxC_{13} - VyC_{23} - VzC_{33} - VC_{43} = 0. \tag{3.3}$$

Devido à formulação homogênea, a escala de C é irrelevante. Assim, de acordo com [1], C_{43} pode ter seu valor considerado como 1. Com isso, as Equações (3.3) podem ser escritas na forma matricial da seguinte maneira:

$$\begin{bmatrix} x^1 & y^1 & z^1 & 1 & 0 & 0 & 0 & 0 & -U^1x^1 & -U^1y^1 & -U^1z^1 \\ 0 & 0 & 0 & 0 & x^1 & y^1 & z^1 & 1 & -V^1x^1 & -V^1y^1 & -V^1z^1 \\ x^2 & y^2 & z^2 & 1 & . & . & . & & & & . \\ . & & & & & & & & & & . \\ . & & & & & & & & & & . \\ . & & & & & & & & & & . \\ 0 & 0 & 0 & 0 & x^n & y^n & z^n & 1 & -V^nx^n & -V^ny^n & -V^nz^n \end{bmatrix} \begin{bmatrix} C_{11} \\ C_{21} \\ . \\ . \\ . \\ C_{33} \\ 1 \end{bmatrix} = \begin{bmatrix} U^1 \\ V^1 \\ . \\ . \\ . \\ U^n \\ V^n \end{bmatrix}. \tag{3.4}$$

Para calcular o valor de C, são necessárias onze equações. Assim, devem ser obtidos cinco pares e meio de pontos correspondentes. Para levar em consideração um número maior de pontos, é usada a resolução do sistema linear através de mínimos quadrados, através da utilização da pseudo-inversa da matriz.

O problema nesta aproximação é o fato de que o conjunto de pares de pontos nem sempre é obtido corretamente, havendo casos de pares falsos, que afetam de forma considerável a precisão da matriz de calibração. Uma maneira de solucionar este problema é fazer uma seleção dos melhores pontos, de forma que a matriz de calibração obtida através destes pontos possua o melhor limiar de erro quando aplicada a pontos com correspondência conhecida.

Para isso, deve-se obter subconjuntos de pontos, que podem ser selecionados de forma aleatória, ou podem ser consideradas todas as combinações possíveis. Para cada subconjunto de pontos é então obtida uma matriz de calibração. Esta matriz é aplicada em todos os pontos do conjunto, cujas correspondências são conhecidas, e é verificado o erro entre o ponto obtido através da aplicação da matriz de calibração e o ponto correspondente. A matriz que obtiver o menor erro será então a matriz utilizada no processo de calibração.

Um dos métodos mais conhecidos que seguem esta linha é o RANSAC (*Random Sample Consensus*) [12]. Nele, para um conjunto de N correspondências, são escolhidos N_S subconjuntos contendo N_C correspondências cada. Para cada subconjunto, estima-se a matriz de calibração e calcula-se o erro desta para cada uma das N correspondências. Cada erro é comparado com um limiar, e se o erro estiver abaixo dele, a correspondência é considerada boa. Este cálculo é feito em cada subconjunto, e no final é eleita a matriz que obteve o menor erro. Esta matriz é então aplicada novamente em todos os pontos de correspondência, e todos que possuírem erro abaixo do limiar são utilizados para calcular a matriz final, através da técnica de menores quadrados.

No RANSAC, o número N_S de subconjuntos escolhidos é calculado com base em N_C, ϵ, e p, onde ϵ é a probabilidade de que um dado ponto selecionado possua erro abaixo do limiar e p é a probabilidade de que ao menos um subconjunto seja formado apenas por boas correspondências. Assim, N_S é calculado da seguinte maneira:

$$N_S = log(1-p)/log(1-\epsilon^{N_C}). \tag{3.5}$$

Neste método, caso o erro de uma correspondência esteja abaixo de um limiar, o valor retornado será zero; caso contrário, será um. O somatório dos erros das correspondências é o critério para eleger a melhor matriz. Isso implica que, se o limiar for elevado, todas as correspondências serão consideradas igualmente boas.

Em [31], é proposta uma métrica de estimação de erro, a MSAC (*M-Estimator Sample Consensus*). Nela, caso o erro esteja abaixo de um limiar, o valor retornado será o próprio erro. Caso seja maior ou igual, o valor retornado será o limiar. Desta maneira, boas correspondências contribuem de forma proporcional ao seu ajuste na matriz utilizada. Com isso, um mesmo número de boas correspondências acarreta valores diferentes em matrizes diferentes.

Apesar da otimização proposta pelo MSAC, em problemas que exigem maior precisão os resultados ainda não são satisfatórios. O motivo para este fato é que o modelo de calibração com base em sistemas homogêneos não é suficiente para lidar com distorções produzidas pela lente da câmera.

Faugeras e Toscani [11] propõem um modelo para calibração de câmeras que leva em consideração: (1) a mudança de um ponto ${}^{W}P_{w} = ({}^{W}X_{w}, {}^{W}Y_{w}, {}^{W}Z_{w})$ em coordenadas do mundo para o sistema de coordenadas da câmera; (2) a projeção do ponto 3D no plano da imagem; e (3) a mudança do sistema de coordenadas da imagem da câmera para imagem do computador ${}^{I}P_{d} = ({}^{I}X_{d}, {}^{I}Y_{d})$. Com base nestes critérios, é gerado um conjunto de equações que pode ser expresso da seguinte maneira, na forma matricial:

$$\begin{bmatrix} s^{I}X_{d} \\ s^{I}Y_{d} \\ s \end{bmatrix} = \begin{bmatrix} \alpha_{u} & 0 & u_{0} & 0 \\ 0 & \alpha_{v} & v_{0} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_{x} \\ r_{21} & r_{22} & r_{23} & t_{y} \\ r_{31} & r_{32} & r_{33} & t_{z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{W}X_{w} \\ {}^{W}Y_{w} \\ {}^{W}Z_{w} \\ 1 \end{bmatrix}. \tag{3.6}$$

Nesta equação, que segue a notação proposta em [28], (u_0, v_0) são os componentes do ponto principal em pixels; $\alpha_u = -fk_u$ e $\alpha_v = -fk_v$, onde f é a distância focal, e (k_u, k_v) são os parâmetros que transformam medidas métricas no sistema de coordenadas da câmera para pixels no sistema de coordenadas da imagem. Os valores (t_x, t_y, t_z) expressam a posição da origem do sistema de coordenadas do mundo com respeito ao plano da câmera, e r_{ij} são os coeficientes da matriz de rotação que representa a orientação do sistema de coordenadas do mundo com respeito ao sistema de coordenadas da câmera.

Calculando o produto entre as matrizes da Equação (3.6), a seguinte matriz de calibração é obtida:

$$\begin{bmatrix} s^I X_d \\ s^I Y_d \\ s \end{bmatrix} = \begin{bmatrix} \alpha_u r_{11} + u_0 r_{31} & \alpha_u r_{12} + u_0 r_{32} & \alpha_u r_{13} + u_0 r_{33} & \alpha_u t_x + u_0 t_z \\ \alpha_u r_{21} + u_0 r_{31} & \alpha_u r_{22} + u_0 r_{32} & \alpha_u r_{23} + u_0 r_{33} & \alpha_v t_y + v_0 t_z \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} {}^W X_w \\ {}^W Y_w \\ {}^W Z_w \\ 1 \end{bmatrix}, \quad (3.7)$$

$$A = \begin{bmatrix} \alpha_u r_{11} + u_0 r_{31} & \alpha_u r_{12} + u_0 r_{32} & \alpha_u r_{13} + u_0 r_{33} & \alpha_u t_x + u_0 t_z \\ \alpha_u r_{21} + u_0 r_{31} & \alpha_u r_{22} + u_0 r_{32} & \alpha_u r_{23} + u_0 r_{33} & \alpha_v t_y + v_0 t_z \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}. \quad (3.8)$$

Os parâmetros da câmera podem então ser obtidos ao igualar a matriz A com a matriz obtida através da técnica de calibração simples, apresentada em [1], da seguinte maneira:

$$\begin{aligned} & u_0 = A_1 {A_3}^T, \quad v_0 = A_2 {A_3}^T, \\ & \alpha_u = -(A_1 {A_1}^T - u_0^2)^{1/2}, \quad \alpha_v = -(A_2 {A_2}^T - v_0^2)^{1/2}, \\ & r_1 = \frac{1}{\alpha_u}(A_1 - u_0 A_3), \quad r_2 = \frac{1}{\alpha_v}(A_2 - v_0 A_3), \quad r_3 = A_3, \\ & t_x = \frac{1}{\alpha_u}(A_{14} - u_0 A_{34}), \quad t_y = \frac{1}{\alpha_v}(A_{24} - v_0 A_{34}) \quad e \quad t_z = A_{34}. \end{aligned} \quad (3.9)$$

Onde:

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \end{bmatrix} = \begin{bmatrix} A_1 & A_{14} \\ A_2 & A_{24} \\ A_3 & A_{34} \end{bmatrix}. \tag{3.10}$$

Faugeras e Toscani propõem um método alternativo à calibração simples para calcular a matriz de calibração. Nela, o modelo de calibração é rearranjado da seguinte maneira:

$$^{I}X_d = T_1 {}^{W}P_W + C_1 - T_2 {}^{W}P_W {}^{I}X_d \quad e \quad {}^{I}Y_d = T_3 {}^{W}P_W + C_2 - T_2 {}^{W}P_W {}^{I}Y_d. \tag{3.11}$$

Onde:

$$T_1 = A_1/A_{34}, \quad T_2 = A_3/A_{34}, \quad T_3 = A_2/A_{34}, \quad C_1 = A_{14}/A_{34} \quad e \quad C_2 = A_{24}/A_{34}. \tag{3.12}$$

Seja então o vetor de incógnitas $X = (T_1, T_2, T_3, C_1, C_2)^T$, obtido através do uso da técnica de mínimos quadrados ($B = QX$), de forma que:

$$Q = \begin{bmatrix} {}^{W}X_{w1} & {}^{W}Y_{w1} & {}^{W}Z_{w1} & -{}^{I}X_{d1}^{W}X_{w1} & -{}^{I}X_{d1}^{W}Y_{w1} & -{}^{I}X_{d1}^{W}Z_{w1} & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -{}^{I}Y_{d1}^{W}X_{w1} & -{}^{I}Y_{d1}^{W}Y_{w1} & -{}^{I}Y_{d1}^{W}Z_{w1} & {}^{W}X_{w1} & {}^{W}Y_{w1} & {}^{W}Z_{w1} & 0 & 1 \\ {}^{W}X_{w2} & {}^{W}Y_{w2} & {}^{W}Z_{w2} & . & . & . & & & & & . \\ . & & & & & & & & & & . \\ . & & & & & & & & & & . \\ . & & & & & & & & & & . \\ 0 & 0 & 0 & -{}^{I}Y_{dn}^{W}X_{wn} & -{}^{I}Y_{dn}^{W}Y_{wn} & -{}^{I}Y_{dn}^{W}Z_{wn} & {}^{W}X_{wn} & {}^{W}Y_{wn} & {}^{W}Z_{wn} & 0 & 1 \end{bmatrix},$$

$$B^T = \begin{bmatrix} {}^{I}X_{d1} & {}^{I}Y_{d1} & {}^{I}X_{d2} & . & . & . & {}^{I}Y_{dn} \end{bmatrix}. \tag{3.13}$$

Neste método, o vetor X é obtido através do cálculo da pseudo-inversa. Os parâmetros de calibração são então extraídos do vetor X, com base na Equação (3.7):

$$\begin{aligned} &T_1 = \frac{r_3}{t_z}u_0 + \frac{r_1}{t_z}\alpha_u, \quad T_2 = \frac{r_3}{t_z}, \quad T_3 = \frac{r_3}{t_z}v_0 + \frac{r_2}{t_z}\alpha_v, \\ &C_1 = u_0 + \frac{t_x}{t_z}\alpha_u \quad e \quad C_2 = v_0 + \frac{t_y}{t_z}\alpha_v. \end{aligned} \tag{3.14}$$

Os parâmetros da câmera são obtidos a partir das Equações 3.15:

$$\begin{aligned} &u_0 = \frac{T_1 {T_2}^T}{||T_2||^2}, \quad v_0 = \frac{T_2 {T_3}^T}{||T_2||^2}, \\ &\alpha_u = -\frac{||{T_1}^T \times {T_2}^T||}{||T_2||^2}, \quad \alpha_v = -\frac{||{T_2}^T \times {T_3}^T||}{||T_2||^2}, \\ &r_1 = -\frac{||T_2||}{||{T_1}^T \times {T_2}^T||}\left(T_1 - \frac{T_1 T_2^T}{||T_2||^2}T_2\right), \quad r_2 = -\frac{||T_2||}{||{T_2}^T \times {T_3}^T||}\left(T_3 - \frac{T_2 T_3^T}{||T_2||^2}T_2\right), \quad r_3 = \frac{T_2}{||T_2||} \\ &t_x = -\frac{||T_2||}{||{T_1}^T \times {T_2}^T||}\left(C_1 - \frac{T_1 T_2^T}{||T_2||^2}\right), \quad t_y = -\frac{||T_2||}{||{T_2}^T \times {T_3}^T||}\left(C_2 - \frac{T_2 T_3^T}{||T_2||^2}\right) \quad e \\ &t_z = \frac{1}{||T_2||}. \end{aligned} \tag{3.15}$$

Foi provado em Tsai [32] que somente o primeiro termo das séries de distorção radial é suficiente para modelar a distorção na maioria dos casos. Assim, de acordo com [28], a distorção é modelada através das Equações 3.16.

$$\delta_{xr} = k_1^C X_d({}^C X_d^2 + {}^C Y_d^2), \quad \delta_{yr} = k_1^C Y_d({}^C X_d^2 + {}^C Y_d^2). \tag{3.16}$$

Nas Equações (3.16), k_1 é o primeiro termo das séries de distorção radial, e $({}^C X_d, {}^C Y_d)$ são os pontos bidimensionais distorcidos, no sistema de coordenadas da câmera. Ao acrescentar estas equações ao modelo de Faugeras-Toscani, tem-se o seguinte modelo de calibração:

$$\begin{aligned}
{}^C X_d + {}^C X_d k_1 r^2 &= f\frac{r_{11}\,{}^W X_w + r_{12}\,{}^W Y_w + r_{13}\,{}^W Z_w + t_x}{r_{31}\,{}^W X_w + r_{32}\,{}^W Y_w + r_{33}\,{}^W Z_w + t_z}, \\
{}^C Y_d + {}^C Y_d k_1 r^2 &= f\frac{r_{21}\,{}^W X_w + r_{22}\,{}^W Y_w + r_{23}\,{}^W Z_w + t_x}{r_{31}\,{}^W X_w + r_{32}\,{}^W Y_w + r_{33}\,{}^W Z_w + t_z}, \\
r &= \sqrt{{}^C X_d^2 + {}^C Y_d^2}.
\end{aligned} \tag{3.17}$$

Por fim, devem ser usadas as Equações (3.18) para efetuar a transformação de coordenadas métricas $({}^C X_d, {}^C Y_d)$ em pixels $({}^I X_d, {}^I Y_d)$.

$${}^I X_d = -k_u\,{}^C X_d + u_0, \quad {}^I Y_d = -k_v\,{}^C Y_d + v_0. \tag{3.18}$$

Ao acrescentar a distorção ao modelo de Faugeras-Toscani, o sistema se torna não-linear, e para que as equações sejam resolvidas, é necessária a aplicação de um método iterativo. Em [28], é sugerido o método de Newton-Raphson, com uma estimativa inicial obtida através do método linear de Faugeras-Toscani, e assumindo inicialmente o primeiro termo das séries de distorção radial como zero.

O método apresentado em [32] também inclui a distorção radial das lentes em seu modelo de calibração. Inicialmente, o modelo de Tsai é equivalente ao Faugeras-Toscani com distorção (Equações (3.17)), porém a transformação de coordenadas métricas para pixels é dada através das Equações 3.19.

$$^{I}X_d = -s_x d'_x \ ^{-1C}X_d + u_0, \quad ^{I}Y_d = -d_y \ ^{-1C}Y_d + v_0. \tag{3.19}$$

Nas Equações (3.19), (u_0, v_0) são os componentes do ponto principal em pixels, s_x é o fator de escala da imagem, $d'_x = d_x N_{cx}/N_{fx}$, onde dx é a distância entre os centros de sensores adjacentes na direção de X e d_y é esta distância na direção de Y. N_{cx} é o número de sensores na direção X, e N_{fx} é o número de pixels amostrados pelo computador em uma linha da imagem.

Uma vez que as coordenadas $(^{C}X_d, ^{C}Y_d)$ são obtidas em coordenadas métricas, elas são representadas em pixels através das Equações (3.19). Através das coordenadas em pixels obtidas, é encontrada uma relação entre um ponto P_d da imagem (em coordenadas métricas) e um ponto P_w, do objeto. Através da Figura 3.1, que mostra como a distorção radial afeta o modelo da câmera, nota-se que o segmento $\overline{O_R P_d}$ é paralelo ao segmento $\overline{P_{oz} P_w}$. Através desta restrição, é estabelecido um relacionamento que é utilizado para obter um sistema com n equações e sete incógnitas.

Essas incógnitas são calculadas através da técnica de mínimos quadrados, e a partir dos valores obtidos são calculados os parâmetros de rotação, translação em X e Y, e escala do modelo. Para computar a distância focal, o coeficiente de distorção radial e a translação em Z, é usada uma aproximação linear que não considera o parâmetro k_1.

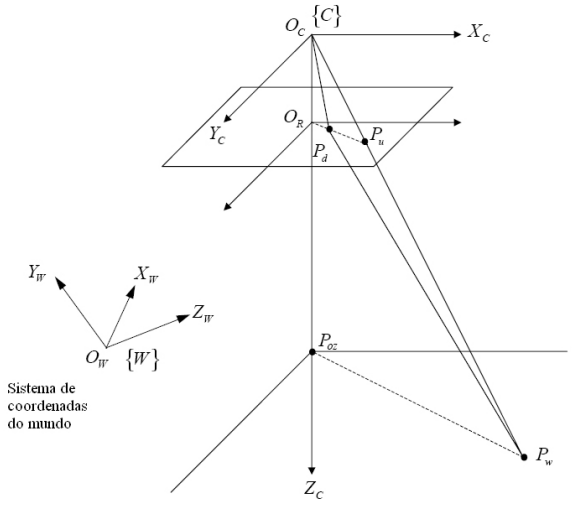


Figura 3.1: Ilustração da restrição de distorção radial [32].

Essa aproximação é então aplicada ao conjunto inteiro de pontos de teste, e é obtido um sistema de n equações e duas incógnitas. A aproximação linear das duas incógnitas (f e t_z) pode ser calculada com o uso de uma matriz pseudo-inversa. No entanto, para calcular uma aproximação mais exata, e o parâmetro k_1, é necessário iterar uma aproximação linear obtida a partir das Equações (3.17), considerando $k_1 = 0$ como solução inicial. Depois de obter todos os parâmetros, estes são otimizados iterativamente com o objetivo de alcançar uma solução mais precisa.

3.1.2 Alinhamento por Características

Quando o objeto a ser digitalizado não pode ser movido, a técnica de calibração apresenta um problema prático, pois é necessário recalibrar o sistema a cada captura de uma nova perspectiva. Neste tipo de situação, é interessante a utilização de uma técnica que dê maior liberdade durante a captura de dados, eliminando a necessidade do posicionamento fixo entre a câmera e o scanner.

Neste contexto estão situados métodos que efetuam o alinhamento entre `imagens da câmera` capturadas a partir de posições aleatórias e um modelo tridimensional. Esse alinhamento é feito com base em características reconhecidas nas `imagens da câmera` e no modelo 3D.

O modelo 3D é gerado previamente através das `imagens de profundidade` (ver Capítulo 2), e pode conter informações adicionais, como cor ou refletância, caso estas sejam obtidas pelo *scanner* durante a etapa de aquisição de imagens. Considerando que tais informações são capturadas pelo mesmo dispositivo óptico, pode-se afirmar que elas estão alinhadas às `imagens de profundidade`.

Kurazume *et al.* [17] apresentam uma técnica de alinhamento que usa as imagens de refletância obtidas pelo *scanner* como referência. Tais imagens são obtidas como produto colateral em *scanners* que utilizam tempo de retorno do *laser* como técnica para medir distâncias.

Ao passo em que o tempo de retorno do *laser* fornece informações sobre a profundidade, a intensidade com a qual o *laser* retorna provê uma medida da refletância de cada ponto percorrido pelo *scanner*. Com isso, é possível a obtenção de imagens de refletância, que consistem em uma coleção de intensidades da energia retornada para cada pixel.

Estas imagens são empregadas no alinhamento de `imagens de profundidade` com `imagens da câmera`. As imagens de refletância possuem características similares às `imagens da câmera`, visto que as duas estão relacionadas à aspereza da superfície [14]. Desta maneira, arestas de refletância observadas ao longo das imagens obtidas refletem diferenças do material da superfície do objeto.

Partindo do pressuposto que materiais diferentes acarretam cores diferentes, tais arestas de refletância também podem ser observadas nas `imagens da câmera` (Figura 3.2). Assim, neste método de alinhamento, as arestas de refletância são copiadas no modelo geométrico 3D. Como bordas de oclusão (arestas pertencentes à silhueta do objeto) variam de acordo com a direção do observador, arestas ao longo destas bordas são inicialmente removidas das imagens de refletância.

As arestas ao longo das bordas de oclusão são estimadas a partir do modelo tridimensional e da direção de visualização corrente. As arestas de refletância projetadas são então alinhadas com as arestas das `imagens da câmera` de forma que o erro da posição 3D destas arestas seja minimizado através de um cálculo iterativo.

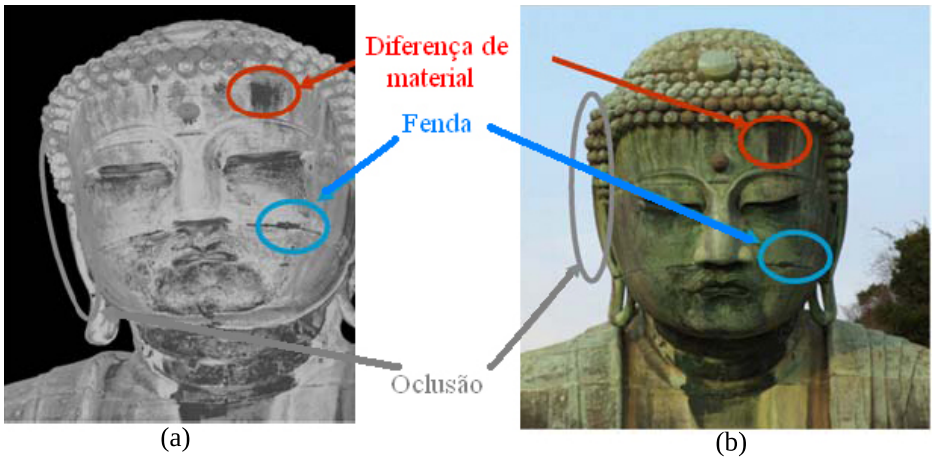


Figura 3.2: Arestas: (a) na imagem de refletância; e (b) na imagem colorida [14].

Para estabelecer a correspondência, o sistema encontra os pontos da imagem colorida que estão mais próximos dos pontos de refletância projetados. Essa operação é similar ao algoritmo ICP (*Iterative Closest Point*) [7] e [4]. Para determinar a posição relativa que coincide com a posição das arestas coloridas 2D e os pontos de refletância projetados, é usado um *M-estimator*.

Nele, é utilizado um algoritmo para minimizar a distância z_i entre um ponto da aresta 2D de refletância projetada no sistema de coordenadas do modelo 3D (H), e um ponto P da aresta 3D de refletância (ver Figura 3.3). Minimizando esta distância entre todos os pontos das arestas, o algoritmo encontra uma configuração que corresponde ao posicionamento entre a câmera e o sensor de profundidade.

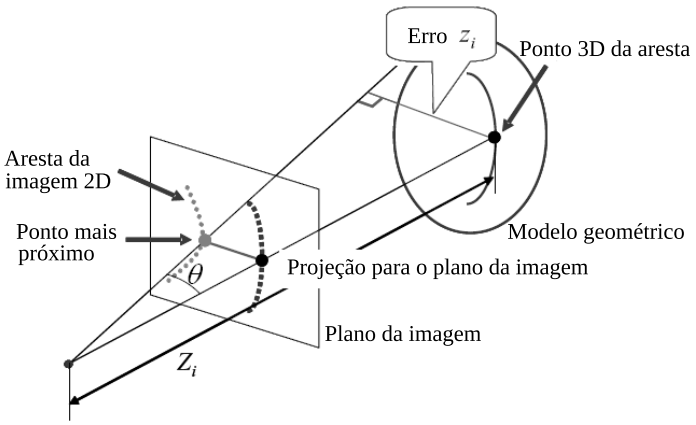


Figura 3.3: Distâncias tridimensional e bidimensional [14].

3.2 Parametrização do Modelo Fotométrico

Quando uma imagem e um modelo 3D estão registrados é possível descobrir, para cada vértice do modelo, sua posição nas imagens coloridas. Assim, a partir de um conjunto de imagens onde o ponto do objeto correspondente a um dado vértice aparece, podemos obter o conjunto de cores observadas para este vértice, mapeadas nos pixels destas imagens. Quando este conjunto de cores apresenta uma variação, geralmente causada por mudanças na iluminação incidente sobre o objeto, pode-se analisar este conjunto, e calcular informações sobre a cor e refletância do vértice.

Nesta seção serão apresentados métodos que têm como objetivo modelar as propriedades fotométricas de um objeto através de sua observação. Tais métodos possuem como base a idéia de que, a partir de imagens de uma mesma superfície com iluminações ou ângulos diferentes, é possível modelar sua refletância.

Em [30], são utilizados um modelo tridimensional de um objeto e uma seqüência de imagens coloridas deste objeto para estimar os parâmetros de um modelo de reflexão. Este método se baseia na separação dos componentes de reflexão difusa e especular da seqüência de imagens coloridas. Depois de obtidos, os parâmetros de refletância para cada componente de reflexão são estimados separadamente.

Neste trabalho, uma lâmpada incandescente é usada como uma fonte de luz pontual e o objeto está localizado sobre um braço robótico. Na Figura 3.4, extraída do artigo [30], o sistema de aquisição de imagens é ilustrado. A cada movimentação do objeto são capturadas as imagens de profundidade e de cores. Como o ponto de luz é fixo, a cada mudança de posição a iluminação muda sobre a superfície. Nos experimentos feitos no trabalho, a direção e a cor da luz foram calibrados.

A geometria do objeto é reconstruída através de uma seqüência de imagens de profundidade do objeto. Após a construção, para melhorar o armazenamento e a renderização, o número de triângulos é reduzido através da simplificação da forma do objeto.

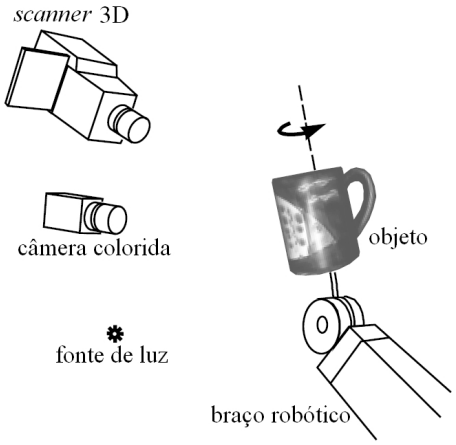


Figura 3.4: Sistema de aquisição de imagens utilizado em [30].

Segundo os autores, para estimar parâmetros de refletância em um ponto, é necessário que se tenha conhecimento sobre três valores no ponto analisado: a direção do observador, da fonte de luz, e da normal da superfície. A normal da superfície é computada em pontos dispostos em uma grade regular, contida dentro de cada triângulo da malha tridimensional. A resolução da grade de pontos pode ser alterada de acordo com o tamanho de cada triângulo, para que a densidade de pontos se mantenha uniforme ao longo da superfície do objeto.

As propriedades de refletância são calculadas com o uso do modelo da superfície do objeto e das imagens coloridas. Inicialmente, os dois principais componentes de reflexão (difusa e especular) são estimados a partir das imagens de entrada. De acordo com os autores, a separação dos componentes de reflexão permite a obtenção de aproximações confiáveis para os parâmetros de reflexão especulares. Além disso, o componente de reflexão especular não afeta a estimativa dos parâmetros de reflexão difusa do objeto.

No trabalho é apresentado um modelo de reflexão genérico, que é descrito em termos de três componentes de reflexão: o componente de reflexão difusa, o componente de reflexão especular e os "furos especulares". Os furos especulares existem apenas em superfícies espelhadas onde os raios de luz do componente especular foram refletidos em uma direção. Assim, é difícil observar esse componente em conjuntos de direções de observação amostrados de maneira ruim.

Na análise feita é utilizado o modelo de Torrance-Sparrow, que representa os componentes de reflexão difusa e especular da seguinte maneira:

$$I_m = K_{D,m} cos\theta_i + K_{S,m} \frac{1}{cos\theta_r} e^{-\alpha^2/2\sigma^2}, \quad para \quad m = R, G, B. \tag{3.20}$$

Neste modelo, θ_i é o ângulo entre a normal da superfície e a direção da fonte de luz, θ_r é o ângulo entre a normal da superfície e a direção do observador, e α é o ângulo entre a normal da superfície e a bissetriz da direção da fonte de luz e da direção do observador (ver Figura 3.5). $K_{D,m}$ e $K_{S,m}$ (onde $m = R, G, B$) são constantes para os componentes

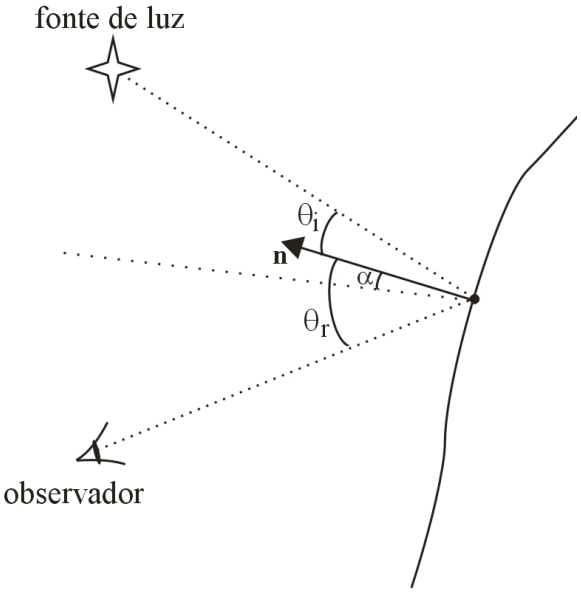


Figura 3.5: Ilustração de variáveis utilizadas no modelo de Torrance-Sparrow.

de reflexão difusa e especular para cada faixa de cor da imagem, e σ é o desvio padrão de uma observação da curva do modelo de Torrance-Sparrow.

Esse modelo de reflexão é válido apenas para objetos convexos, e não pode ser usado para representar inter-reflexões em superfícies côncavas. Desta maneira, foi assumido empiricamente que inter-reflexões não afetam a análise de forma significante.

Para efetuar a separação dos componentes de reflexão difusa e especular, e para estimar os parâmetros para cada componente de reflexão, é necessária a seqüência das cores observadas em cada ponto da superfície do objeto enquanto este foi girado sobre o braço robótico. Assim, a partir das correlações existentes entre os sistemas de coordenadas, são geradas matrizes de transformação e projeção que são utilizadas para mapear um dado ponto da superfície em diferentes imagens de profundidade do objeto.

Assim, a cor observada para um dado ponto tridimensional P em uma imagem f será a cor do pixel na localização (x, y), encontrada através da aplicação das matrizes de transformação e projeção sobre o ponto P. Caso o ponto P não seja visível na imagem, a cor observada para este ponto será $(R, G, B) = (0, 0, 0)$. Repetindo este procedimento para todas as imagens, é então obtida uma seqüência de cores observadas para o ponto P.

Para separar os componentes de reflexão, é usado um algoritmo introduzido em outro trabalho dos autores [29]. Utilizando três canais de cores (vermelho, verde e azul), os coeficientes $K_{D,m}$ e $K_{S,m}$ da Equação (3.20) são generalizados como dois vetores linearmente independentes:

$$\underset{\sim}{\mathrm{K}}_D = \left[\begin{array}{ccc} K_{D,R} & K_{D,G} & K_{D,B} \end{array} \right]^T \quad e \quad \underset{\sim}{\mathrm{K}}_S = \left[\begin{array}{ccc} K_{S,R} & K_{S,G} & K_{S,B} \end{array} \right]^T . \tag{3.21}$$

As seqüências de cores observadas para cada ponto do objeto são guardadas nas linhas de uma matriz M, de dimensão $n \times 3$. Levando em consideração o modelo de refletância (Equação (3.20)) e os dois vetores de cores da Equação (3.21), os valores de intensidade nos canais de R, G e B são representados como:

$$\begin{aligned} M &= \left[\begin{array}{ccc} \underset{\sim}{\mathrm{M}}_R & \underset{\sim}{\mathrm{M}}_G & \underset{\sim}{\mathrm{M}}_B \end{array} \right] \\ &= \left[\begin{array}{cc} cos\theta_{i1} & E(\theta_{r1}, \alpha_1) \\ cos\theta_{i2} & E(\theta_{r2}, \alpha_2) \\ \vdots & \vdots \\ cos\theta_{in} & E(\theta_{rn}, \alpha_n) \end{array} \right] \left[\begin{array}{ccc} K_{D,R} & K_{D,G} & K_{D,B} \\ K_{S,R} & K_{S,G} & K_{S,B} \end{array} \right] \\ &= \left[\begin{array}{cc} \underset{\sim}{\mathrm{G}}_D & \underset{\sim}{\mathrm{G}}_S \end{array} \right] \left[\begin{array}{c} \underset{\sim}{\mathrm{K}}_D^T \\ \underset{\sim}{\mathrm{K}}_S^T \end{array} \right] \\ &\equiv GK. \end{aligned} \tag{3.22}$$

Na Equação (3.22), $E(\theta_r, \alpha) = exp(-\alpha^2/2\sigma^2)/cos\theta_r$, e os vetores $\underset{\sim}{\mathrm{G}}_D$ e $\underset{\sim}{\mathrm{G}}_S$ representam os valores de intensidade dos componentes de reflexão difusa e especular com respeito às direções de iluminação e observação θ_i, θ_r e α. Os vetores $\underset{\sim}{\mathrm{K}}_D$ e $\underset{\sim}{\mathrm{K}}_S$ representam as cores das reflexões difusas e especulares, respectivamente.

É suposto em [30] que existe uma estimativa da matriz K. Assim, dois componentes de reflexão representados pela matriz G são obtidos através da projeção de M (matriz

com as reflexões observadas) sobre os vetores de cores $\underset{\sim}{K}_D$ e $\underset{\sim}{K}_S$:

$$G = MK^{+}, \tag{3.23}$$

onde K^{+} é a matriz pseudo-inversa da matriz K.

Esta derivação se baseia no fato de que a matriz K é conhecida. Nos experimentos realizados, um procedimento de calibração mede o vetor de cores $\underset{\sim}{K}_S$ (reflexão especular) como a cor da fonte de luz. Assim, apenas o vetor $\underset{\sim}{K}_D$ (reflexão difusa) é desconhecido e precisa ser estimado.

Como a distribuição do componente de reflexão especular é limitada a um ângulo fixo que depende de σ, é observado em [30] que se o α é suficientemente grande em um ponto da superfície do objeto, a cor observada no ponto representa a cor do componente difuso. Os ângulos α, θ_i e θ_r são computados com o uso de dados obtidos previamente, *i.e.* localização da fonte de luz, a matriz de projeção da câmera e as matrizes de transformação do objeto. Sabendo-se os valores da matriz K, os componentes de reflexão e difusão podem ser obtidos através da decomposição em valores singulares da matriz M [29]:

$$M_D = \underset{\sim}{G}_D\underset{\sim}{K}_D^T \quad e \quad M_S = \underset{\sim}{G}_S\underset{\sim}{K}_S^T. \tag{3.24}$$

Com o componente de reflexão difusa segmentado da seqüência de cores, pode-se estimar então os parâmetros de reflexão difusa ($K_{D,R}$, $K_{D,G}$ e $K_{D,B}$) sem a interferência dos efeitos indesejáveis do componente de reflexão especular.

Utilizando o ângulo θ_i, os parâmetros de reflexão difusa são calculados através da aplicação do modelo de reflexão no componente de reflexão difusa. Os parâmetros de reflexão difusa são estimados em pontos contidos em grades regulares dentro de cada triângulo da malha, assim como as normais da superfície. A resolução deve ser alta, para capturar detalhes da reflexão difusa sobre a textura da superfície. Para o cálculo dos parâmetros de reflexão, a resolução é determinada com base no número médio de pixels

das imagens coloridas que se encontram dentro de cada triângulo do modelo do objeto.

Os parâmetros da reflexão especular ($K_{S,R}$, $K_{S,G}$ e $K_{S,B}$) são computados com base nos ângulos θ_r e α. Ao passo que para a reflexão difusa os parâmetros podem ser estimados desde que a superfície do objeto esteja iluminada e no campo de visão da câmera, na obtenção de parâmetros de reflexão especular o intervalo de direções de visualização é menor, pois somente são considerados os pontos que apresentam especulares.

Em um conjunto limitado de imagens coloridas, o componente de reflexão especular é observado apenas em uma pequena parte da superfície do objeto, e quando é observado, deve obedecer a alguns critérios definidos no artigo para que seja considerado confiável. Devido a este problema, os parâmetros de reflexão especular são medidos de forma esparsa sobre a superfície do objeto. Assim, para compensar a quantidade de informações obtidas, é usada interpolação para inferir as reflexões especulares sobre a superfície.

Posteriormente, Rushmeier e Bernardini desenvolveram um método que também calcula normais e cores de um objeto a partir de seu modelo tridimensional e de dados fotométricos [24]. Neste trabalho é utilizado um sensor que captura as imagens de profundidade e coloridas simultaneamente, e são utilizadas cinco lâmpadas halógenas que, ao serem acesas uma de cada vez, permitem o imageamento de uma mesma cena com cinco fontes de luz diferentes.

A partir dessas imagens com diferentes iluminações, é computado um conjunto de normais e cores. No entanto, esse conjunto é susceptível a erros, causados por variações na iluminação e no posicionamento do sensor. Esses erros acarretam diferenças entre as normais obtidas e as existentes nas malhas geométricas, e inconsistências na coloração calculada. Rushmeier e Bernardini [24] apresentam um método que tem como objetivo a correção destas variações.

Para a correção das normais, é feito um ajuste local da intensidade da luz utilizada nos cálculos, utilizando informações sobre a geometria da região analisada. Como a geometria do objeto é obtida através de várias malhas, estas são alinhadas e unidas antes de

serem usadas como referência. Um procedimento semelhante de registro global é utilizado para corrigir as variações cromáticas, computando assim tons de vermelho, verde e azul corrigidos a partir dos dados fotométricos.

O método para o cálculo das normais e cores utilizado neste projeto é baseado na técnica da fotometria estéreo. Em [24] são obtidas várias imagens de um mesmo objeto, a partir de um mesmo ângulo da câmera, com iluminações diferentes e conhecidas. No trabalho, é assumida uma função ideal, que aproxima a função lambertiana de distribuição de refletância bidirecional (*Bidirectional Reflectance Distribution Function* - BRDF) e fontes de luz idênticas. Com estes valores, é assumido que a normal e refletância relativas para cada pixel podem ser calculadas a partir de três imagens, de acordo com o seguinte conjunto de equações:

$$\frac{\rho_p}{\pi} L_o \Delta_w \begin{bmatrix} l_{1,x} & l_{2,x} & l_{3,x} \\ l_{1,y} & l_{2,y} & l_{3,y} \\ l_{1,z} & l_{2,z} & l_{3,z} \end{bmatrix} \begin{bmatrix} n_{p,x} \\ n_{p,y} \\ n_{p,z} \end{bmatrix} = \begin{bmatrix} \alpha L_{r,1,p} \\ \alpha L_{r,2,p} \\ \alpha L_{r,3,p} \end{bmatrix}. \tag{3.25}$$

Nesta equação, ρ_p/π é a BRDF no pixel p, L_o é a radiância da fonte de luz, $L_{r,i,p}$ é a radiância refletida no ponto visível pelo pixel p na imagem i, e Δ_w é o ângulo formado pela fonte de luz. A variável n_p representa a normal da superfície em p e l_i é o vetor de direção para a (infinitamente distante) i-ésima fonte de luz (ver Figura 3.6).

A constante α representa uma escala da câmera para a radiância refletida em um valor entre 0 e 255, feita depois que as imagens estão ajustadas. As equações podem ser resolvidas diretamente para $(\rho_p L_o \Delta_w / \alpha\pi) n_p$. Como a magnitude de n_p é 1, a partir deste resultado pode-se obter o vetor normal $\mathbf{n}_p$ e o valor $\rho_{rel,p} = \rho_p L_o \Delta_w / \alpha\pi$, que é a refletância no pixel p em relação aos outros pixels na imagem.

Seguindo este processo, surgem dois problemas: a presença de componentes altamente especulares na BRDF e de sombras. Para contornar estes problemas, são obtidas fotos de uma cena com fontes de luz adicionais. Com isso, os pixels que apresentam especularidade

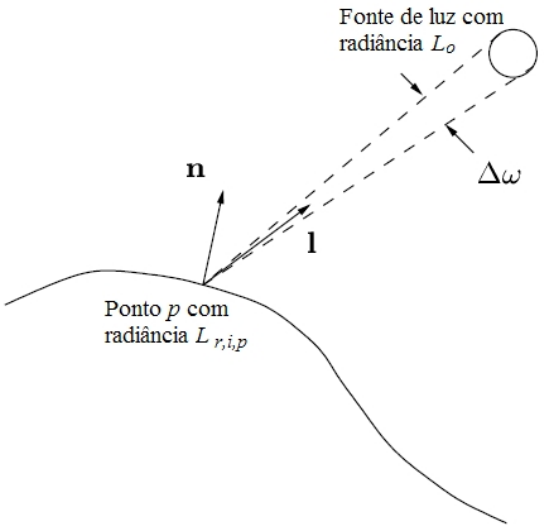


Figura 3.6: Representação da refletância de luz utilizada na Equação 3.25 [24].

alta ou sombras não são utilizados. Estes pixels são identificados através de valores relativamente muito altos ou baixos em comparação às outras fotos, respectivamente.

No sistema apresentado em [24] são utilizadas cinco fontes de luz, possibilitando duas medidas redundantes para cada localização de um dado pixel. Somente são consideradas as localizações de pixels caso seus valores estejam dentro do limiar em no mínimo três imagens. Neste trabalho, são apontados alguns problemas que surgem na aplicação do modelo de iluminação descrito. Apesar de ser utilizado o mesmo tipo de fonte de luz, a iluminação e o espectro originados podem variar de lâmpada para lâmpada. Entre os fatores que causam este problema estão o tempo de vida da lâmpada, o nível de energia elétrica disponível, e o fato de lâmpadas de baixo custo serem fabricadas com pequenas diferenças na radiância emitida, o que afeta o processo de captura das cores.

Para compensar estas limitações, é utilizada a malha de baixa resolução obtida durante a digitalização do objeto, e o registro global dos resultados. Assim, a Equação (3.25) é reformulada, de forma a ser expressa com radiâncias variáveis e ângulos sólidos. É assumido então que o raio d de uma fonte de luz é o mesmo para todas as fontes de luz, e que d é menos que 10 por cento menor do que a distância $r_{i,p}$ entre um ponto p na superfície para cada fonte de luz i. Desta maneira, o ângulo sólido Δw_i para cada fonte de luz pode ser aproximado como $\pi d^2/r_{i,p}^2$ (Equação 3.26).

$$\frac{{\rho_p}^2}{d}\begin{bmatrix} l_{1,p,x} & l_{2,p,x} & l_{3,p,x} \\ l_{1,p,y} & l_{2,p,y} & l_{3,p,y} \\ l_{1,p,z} & l_{2,p,z} & l_{3,p,z} \end{bmatrix}\begin{bmatrix} n_{p,x} \\ n_{p,y} \\ n_{p,z} \end{bmatrix} = \begin{bmatrix} \alpha L_{r,1,p} {r_{1,p}}^2 / L_{o,1,p} \\ \alpha L_{r,2,p} {r_{2,p}}^2 / L_{o,2,p} \\ \alpha L_{r,3,p} {r_{3,p}}^2 / L_{o,3,p} \end{bmatrix}. \tag{3.26}$$

Nesta equação, $L_{o,i,p}$ representa a radiância da fonte de luz i na localização do pixel p. Todos os valores utilizados, com exceção de α e d variam de acordo com os pixels da imagem.

Cada uma das malhas capturadas pelo *scanner* de profundidade utilizado está registrada a cinco imagens com diferentes iluminações. Assim, as malhas obtidas são alinhadas e integradas, e é gerado o modelo geométrico do objeto. Durante a geração da malha é também obtida a transformação que mapeia um pixel nas imagens coloridas a pontos da malha.

Com estas informações, são então calculados valores para cada pixel das imagens coloridas que são utilizados para obter alguns dos parâmetros da Equação (3.26). Desta maneira, podem ser calculados valores que precisam da localização de um pixel p, visível na superfície, para serem obtidos. Assim, são calculados os valores para a distância $r_{i,p}$, que vai do ponto p até a fonte de luz i; a direção $l_{i,p}$ entre p e a fonte de luz, e uma aproximação na normal da superfície n'_p em p.

Para calcular a radiância da fonte de luz no pixel ($L_{o,i,p}$) que leve em consideração a variação das fontes de luz, são equacionadas a refletância relativa da malha iluminada por uma fonte de luz ideal ($\widetilde{L}_{r,i,p}$), e as radiâncias relativas da imagem na vizinhança do pixel p, $\alpha\bar{L}_{r,i,p}$. A partir das formulações definidas em [24], a seguinte relação é definida:

$$\widetilde{L}_{r,i,p} / \alpha\bar{L}_{r,i,p} \simeq L_u / \rho_p \alpha L_{o,i,p}. \tag{3.27}$$

Isolando o termo $L_{o,i,p}$, tem-se:

$$L_{o,i,p} \simeq \frac{L_u \alpha \bar{L}_{r,i,p}}{\rho_p \alpha \widetilde{L}_{r,i,p}}. \tag{3.28}$$

Utilizando o resultado da Equação (3.28) na Equação (3.25), e simplificando os termos do resultado, tem-se:

$$\frac{L_u}{\alpha} d^2 \begin{bmatrix} l_{1,p,x} & l_{2,p,x} & l_{3,p,x} \\ l_{1,p,y} & l_{2,p,y} & l_{3,p,y} \\ l_{1,p,z} & l_{2,p,z} & l_{3,p,z} \end{bmatrix} \begin{bmatrix} n_{p,x} \\ n_{p,y} \\ n_{p,z} \end{bmatrix} = \begin{bmatrix} \alpha L_{r,1,p} {r_{1,p}}^2 \widetilde{L}_{r,1,p} / \alpha \bar{L}_{r,1,p} \\ \alpha L_{r,2,p} {r_{2,p}}^2 \widetilde{L}_{r,2,p} / \alpha \bar{L}_{r,2,p} \\ \alpha L_{r,3,p} {r_{3,p}}^2 \widetilde{L}_{r,3,p} / \alpha \bar{L}_{r,3,p} \end{bmatrix}. \tag{3.29}$$

Esta equação pode ser resolvida para um vetor na direção da normal da superfície, e a normalização dará a normal n_p resultante. O efeito da refletância da superfície ρ_p atua tanto nos valores individuais de pixels, como na média de valores de pixels. Como resultado, o termo da refletância é cancelado da equação, e o comprimento do vetor na direção da normal à superfície deixa de ter uma interpretação física utilizável.

Dada a aproximação para a normal de cada pixel, podem ser computadas versões das imagens coloridas capturadas sem os efeitos de sombreamento causados pela iluminação. Com a média ponderada destas imagens corrigidas, podem ser estimadas as refletâncias nos canais vermelho, verde e azul. Através do uso de medições adicionais da refletância espectral nos pontos, as imagens corrigidas são ajustadas para aproximar o real valor da refletância em intervalos espectrais definidos.

Seja o comprimento de onda da luz λ e o subscrito C, que representa o canal de cor (vermelho, verde ou azul), o valor da cor armazenado $\alpha L_{r,i,p,C}$ para cada pixel p em cada imagem colorida i é:

$$\alpha L_{r,i,p,C} = \int_0^\infty \rho(\lambda) L_{o,i,p}(\lambda) (d^2 / {r_{i,p}}^2) \mathbf{l}_{i,p} \cdot \mathbf{n}_p S_C(\lambda)\, d\lambda. \tag{3.30}$$

Na Equação (3.30), $S_C(\lambda)$ é a sensibilidade espectral da câmera no canal C. Utilizando as normais obtidas através do método descrito previamente, é computada uma refletância relativa para cada canal de cor através da fórmula:

$$L_{C_r}\rho_{rel,i,p,C} = \alpha L_{r,i,p,C} {r_{i,p}}^2 / \mathbf{l}_{i,p} \cdot \mathbf{n}_p. \tag{3.31}$$

Esta refletância possui valor relativo aos outros pixels no mesmo canal, da mesma imagem. Assim como no cálculo das normais, os pixels com valores mais altos e baixos não são utilizados para evitar o uso de dados em áreas com sombras ou especulares. Para evitar a inclusão de efeitos de variação da intensidade de luz em uma dada direção, são apenas utilizados os pixels que estão situados dentro de um cone pequeno, em torno da direção para onde a luz é projetada.

Para compensar variações na intensidade da fonte de luz, as cinco imagens de refletância relativas computadas através da Equação (3.31) são ajustadas em relação uma à outra. Os pixels nos quais todas as cinco imagens têm valor diferente de zero são encontrados, e as médias destes pixels sobrepostos são computadas. O nível das imagens é ajustado fazendo-se todos os valores médios iguais.

Após o ajuste dos níveis das imagens, as cinco imagens são combinadas de forma a produzir uma imagem corrigida para a posição da câmera. A combinação é feita através de uma média ponderada, onde o peso atribuído a cada pixel em cada imagem aumenta com a distância à aresta de cor preta mais próxima na imagem.

As refletâncias relativas computadas apresentam alguns problemas que as impedem de serem usadas da forma como são calculadas. Elas não estão escaladas para a porcentagem de luz real refletida e contém os efeitos do espectro da fonte de luz e da sensibilidade espectral da câmera. Além disso, os canais vermelho, verde e azul não estão definidos em termo de comprimento de onda.

Para lidar com estes problemas, foram feitas medições ponto-a-ponto da refletância espectral através de um programa de administração de cores (*Colortron*). De acordo com

os autores, as refletâncias espectrais se apresentam como funções simples, e através delas podem ser definidos os limites para o vermelho, verde e azul de forma que as refletâncias espectrais variem pouco a partir da média para o intervalo definido. Assim são definidos os comprimentos de onda máximos e mínimos para cada canal ($\lambda_{C,min}$ e $\lambda_{C,max}$), e é computada a refletância média:

$$\rho_{ave,C} = \int_{\lambda_{C,min}}^{\lambda_{C,max}} \rho(\lambda)/(\lambda_{C,max} - \lambda_{C,min})\, d\lambda. \tag{3.32}$$

Através da proporção entre $\rho_{ave,C}$ e $\rho_{rel,C}$ no local da imagem onde a medição ponto-a-ponto foi feita, pode ser então estimada a escala para os efeitos espectrais da fonte de luz e da câmera na imagem. Com o uso da escala para cada canal de cores da imagem são obtidos valores aproximados para os valores absolutos de refletância em intervalos de espectro claramente definidos.

Uma vez que apenas algumas imagens foram ajustadas para aproximar as refletâncias nos três canais de cores, todas as imagens para as quais $\rho_{rel,C}$ foi computado são ajustadas. Para isso, é exigido que as cores nos pontos de correspondência situados em imagens que se sobrepõem sejam as mesmas. Assim como no registro geométrico, os ajustes entre pares geram acúmulo de erros. Assim, no trabalho é feito um registro global simultâneo.

Esse registro utiliza dados obtidos no alinhamento geométrico inicial para formular equações para o registro da cor. Um dos dados utilizados é uma lista dos pontos em imagens que se sobrepõem, que representam uma mesma localização no objeto. É computada uma cor para cada um destes pontos através da média de seus vizinhos na imagem corrigida. Seja $\beta_{C,k}$ o ajuste do nível para a k-ésima imagem corrigida no canal C. Cada par de pontos correspondentes gera uma equação no seguinte formato, para duas imagens m e n:

$$\rho_{rel,C,m}\beta_{C,m} - \rho_{rel,C,n}\beta_{C,n} = 0. \tag{3.33}$$

Como existem muito mais pontos do que imagens, é utilizada a técnica de mínimos quadrados para computar os valores de β. Para distribuir os erros resultantes de maneira homogênea sobre o modelo, a seguinte equação é adicionada ao conjunto:

$$\sum_{k}^{N} \beta_{C,k} = N. \tag{3.34}$$

Na Equação (3.34), N é o número total de imagens. O uso de N como resultado do somatório produz valores de β que variam em torno de 1. Depois que todas as imagens foram ajustadas em relação uma à outra, os valores de correção global para o vermelho, verde e azul são adaptados às medidas obtidas através de um dispositivo de calibração e correção de cores.

CAPÍTULO 4

MÉTODO PARA GERAÇÃO DE TEXTURA

Como pode ser observado no Capítulo 3, os trabalhos existentes na literatura que visam a geração do modelo fotométrico de objetos geralmente dependem de aparatos de suporte mecânico [30], ou dispositivos e aplicativos especiais [10, 24]. Essa dependência dificulta a implementação e uso destes trabalhos por outros pesquisadores.

Outro problema encontrado é que, apesar de existirem soluções para os principais problemas da modelagem fotométrica (*e.g.* registro, parametrização do modelo fotométrico), faltam trabalhos que detalhem problemas específicos que surgem durante as etapas do processo de modelagem fotométrica, e que apresentem soluções que possam ser incorporadas de forma simples.

Neste contexto, este capítulo apresenta uma técnica prática para aplicar imagens de alta qualidade em modelos 3D, passo essencial no processo de geração de modelos fotométricos com alta resolução e fidelidade. Essa técnica dispensa o uso de aparatos especiais, obtendo uma textura com influência reduzida de sombras e especulares. Também são apresentadas soluções para problemas encontrados durante o processo de calibração e durante a geração da textura para o modelo 3D.

Além da geração de texturas, este método obtém o conjunto de cores observadas para cada vértice do modelo, ou seja, para um dado vértice do modelo 3D, é possível encontrar suas coordenadas em cada fotografia da câmera onde ele apareça, e obter sua cor em cada uma delas. Isso faz com que este método possa ser utilizado em conjunto com as técnicas de modelagem fotométrica descritas na Seção 3.2.

O método desenvolvido possui três etapas principais: aquisição de dados, calibração do sistema e geração da textura. Na etapa de aquisição de dados, conforme definido

no Capítulo 3, são utilizados uma câmera digital profissional e um *scanner* a laser para capturar imagens coloridas e de profundidade do objeto. Na etapa de calibração, pontos de correspondência entre o *scanner* e a câmera são encontrados, e utilizados para calcular a matriz de calibração entre os dois dispositivos. Nesta etapa, é proposta uma modificação da matriz de calibração para lidar com o problema da oclusão, que é apresentado neste capítulo.

O modelo 3D do objeto é gerado a partir das imagens obtidas pelo *scanner*, através do processo apresentado em [33]. Esse modelo, juntamente com as imagens coloridas e de profundidade do objeto, é utilizado como dado de entrada na etapa de geração de texturas, que visa gerar a textura do modelo 3D com base nas `fotografias da câmera`.

Neste capítulo serão apresentadas cada uma das etapas do método desenvolvido. As etapas de aquisição dos dados e calibração do sistema são descritas na Seção 4.1. A etapa de geração de textura é detalhada na Seção 4.2. Os resultados obtidos com o método desenvolvido, bem como detalhes sobre a implementação de cada etapa, são apresentados no Capítulo 5.

4.1 Aquisição dos Dados e Calibração

O sistema de aquisição dos dados é composto de um *scanner* a laser baseado em triangulação [1] e uma câmera fotográfica profissional [2]. Esta câmera permite a captura de imagens de alta qualidade e resolução(4367×2911), ao passo em que as `imagens do scanner` possuem resolução de (640×480). Na Figura 4.1 são exibidas `imagens da câmera` e do `scanner` de um objeto.

É importante ressaltar que o *scanner* captura imagens de profundidade e coloridas através do mesmo dispositivo óptico, permitindo o mapeamento entre pontos 2D das imagens coloridas e pontos 3D das imagens de profundidade correspondentes. Por ser composto de dispositivos diferentes, o sistema de aquisição precisa de um procedimento

[1] *Scanner* Vivid 910: Konica Minolta Sensing Inc (http://www.konicaminolta.com/).
[2] Câmera Canon EOS 5D: Canon Inc (http://www.canon-europe.com/).

(a) (b)

Figura 4.1: Imagens de um vaso de metal, capturadas através: (a) do *scanner*; e (b) da câmera.

de registro entre os sistemas de coordenadas da câmera e do *scanner*. Nesta dissertação o registro é feito através da calibração.

A calibração foi escolhida em relação ao alinhamento por características pois ela não requer imagens de refletância, podendo ser utilizada com *scanners* a *laser* que, como o utilizado neste trabalho, não retornam informações sobre refletância. Além disso, a calibração, ao contrário do alinhamento, pode ser utilizada em objetos com diferentes tipos de textura, retornando sempre uma distribuição confiável de pontos de correspondência. No entanto, para utilizar as matrizes de transformação obtidas através da calibração no contexto deste trabalho foi necessário superar algumas limitações, que serão apresentadas nesta seção.

Para melhor entendimento de como é feita a aquisição dos dados e calibração, é apresentado o fluxograma do processo na Figura 4.2. A calibração dos dispositivos é feita a cada alteração do posicionamento relativo entre a câmera e o *scanner*. Para efetuar a calibração, é necessário capturar uma ou mais `triplas` de imagens de um objeto de calibração. Nesta dissertação, uma `tripla` é composta por uma `imagem do scanner`,

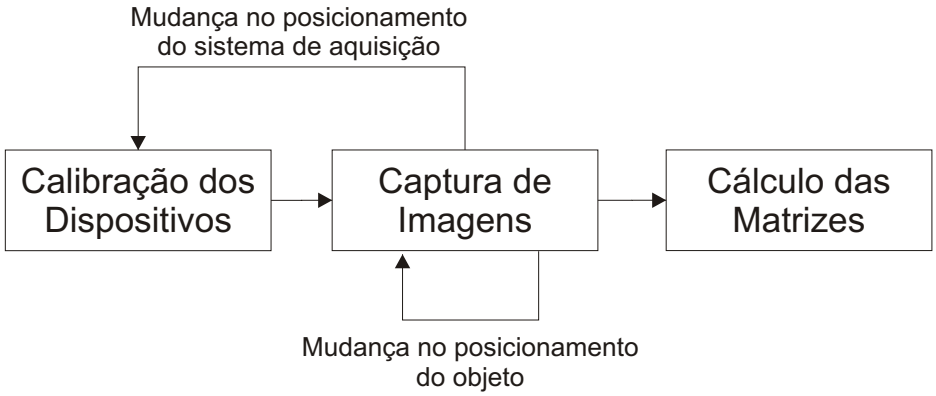


Figura 4.2: Fluxograma dos processos de aquisição de dados e calibração.

uma `imagem de profundidade` e uma `imagem da câmera`.

Um objeto de calibração é um objeto com geometria ou textura conhecida, utilizado para definir os pontos de correspondência que serão utilizados para calcular a matriz de transformação (ver Subseção 3.1.1). Neste trabalho, o objeto de calibração é um cubo com textura de tabuleiro em cada uma de suas faces, e os pontos conhecidos são as quinas dos tabuleiros (Figura 4.3).

Assim como na calibração, durante a captura de imagens do patrimônio a ser preservado é obtido um conjunto com três imagens do patrimônio (uma `imagem do scanner`, uma `imagem de profundidade` e uma `imagem da câmera`). Esta etapa é repetida com o objetivo de digitalizar toda a sua superfície. Para isso, pode-se movimentá-lo (não interfere na calibração do sistema), ou mudar o posicionamento da câmera ou do *scanner* (exige recalibração do sistema). A cada mudança é capturado um conjunto de três imagens do patrimônio, e cada conjunto é associado à calibração dos dispositivos corrente.

O cálculo das matrizes é feito depois da captura das imagens, e não depende dos dispositivos nem dos objetos digitalizados. Nela, uma matriz de transformação é obtida para cada calibração dos dispositivos realizada. Essas matrizes são associadas aos conjuntos de imagens que utilizaram a calibração dos dispositivos que originou a matriz.

A matriz de transformação é obtida através do processo de calibração, apresentado na Subseção 3.1.1. Inicialmente devem ser obtidos os pontos de correspondência entre

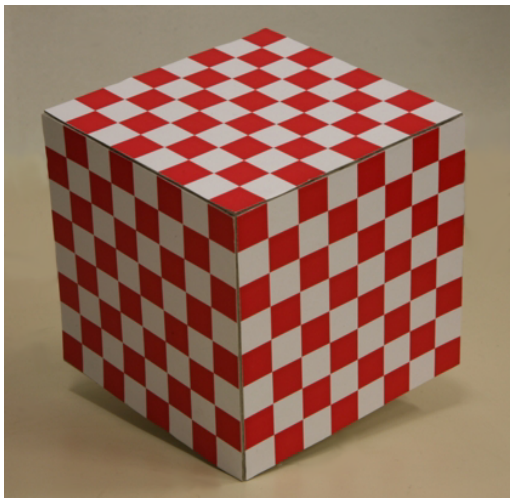
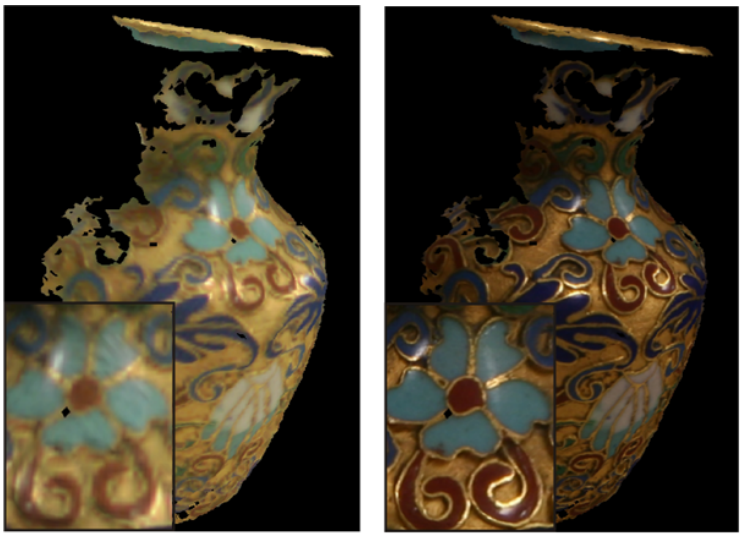


Figura 4.3: Objeto de calibração.

`imagens da câmera` e `imagens de profundidade`. O processo de busca destes pontos é descrito na Subseção 4.1.1. Os pontos de correspondência são utilizados para formular a matriz de transformação entre os sistemas de coordenadas da câmera e do *scanner*. Neste trabalho foram analisadas algumas das técnicas de calibração apresentadas na Subseção 3.1.1. Os resultados obtidos são descritos no Capítulo 5.

A aproximação inicial no uso das matrizes de transformação consiste na geração de uma malha triangular para cada `imagem de profundidade` (vista), e na projeção dos triângulos da malha na `imagem da câmera` correspondente. As áreas triangulares nas `imagens da câmera` formadas pelas projeções são então renderizadas como a textura da malha, através da associação de cada vértice dos triângulos às coordenadas dos pixels obtidos através da projeção do vértice (coordenada de textura).

A Figura 4.4(a) mostra a renderização de uma `imagem do scanner` em uma vista, e a Figura 4.4(b) mostra a renderização de uma `imagem da câmera` na mesma vista. Como esperado, o uso das `imagens da câmera` apresentam uma melhora substancial em relação ao uso das `imagens do` ***`scanner`***. No entanto, essa abordagem deve ser modificada para considerar texturas de um modelo 3D completo, formado pela integração de um conjunto de vistas. Além disso, essa aproximação inicial renderiza de cores errôneas, devido à oclusão. Isso acontece porque vértices da malha que são oclusos por outros quando vistos a partir da perspectiva da câmera recebem as mesmas coordenadas de textura que o



(a) (b)

Figura 4.4: Imagens da renderização de uma vista, utilizando como textura: (a) uma `imagem do scanner`; e (b) uma `imagem da câmera`.

vértices que ficam na frente dele.

Na Subseção 4.1.2 é apresentada a solução desenvolvida neste trabalho para o problema da oclusão de vértices. Na Seção 4.2 é descrito o processo desenvolvido que visa gerar mapas de textura para o modelo 3D a partir das `imagens da câmera` registradas.

4.1.1 Detecção dos Pontos de Correspondência

Os pontos de correspondência são pontos cujas coordenadas 3D e 2D são conhecidas nas `imagens de profundidade` e nas `imagens da câmera`, respectivamente. Para obter estes pontos, inicialmente são localizadas as quinas do cubo em cada par de `imagens da câmera` e `imagens do scanner`, gerando assim um conjunto de pares de pontos 2D. Neste trabalho são utilizadas 147 quinas (três faces do cubo, com 49 quinas em cada).

É utilizado o método de detecção de quinas desenvolvido por Sadlo *et al.* [26]. Esse método possui duas fases. A primeira encontra estimativas iniciais sobre a posição das quinas, e depois aplica uma função da biblioteca OpenCV (*Open Source Computer Vision Library*) [3] sobre as estimativas, localizando assim a posição da quina com precisão

[3]OpenCV: Intel Corporation (http://www.intel.com/technology/computing/opencv/index.htm/).

fracionária (sub-pixel).

A biblioteca OpenCV possui uma função para calcular estimativas, porém de acordo com [26], este método não funciona bem em diferentes condições de visualização do objeto e iluminação. Nesta dissertação foi verificado o método, e este se mostrou falho na estimativa de quinas em imagens com alta resolução. Isso acontece pois em imagens de alta resolução uma quina ocupa uma região de vários pixels na imagem, de forma que a posição exata da quina deve ser obtida a partir da variação de gradientes na região. O método do OpenCV, por não receber como parâmetro uma indicação do tamanho da região a partir da qual uma quina pode ser inferida, não consegue encontrar sua posição, funcionando bem apenas em imagens com baixa resolução (*e.g.* 640×480).

O método de estimativa sugerido por [26] se baseia em pontos de fuga (*vanishing points*). Para cada tabuleiro, estes pontos são obtidos a partir das quatro quinas externas m_i, onde $i = 1, ..., 4$ (ver Figura 4.5). Os pontos de desaparecimento serão então s e t: respectivamente a interseção das retas que passam por $\overline{m_1 m_2}$ e $\overline{m_3 m_4}$, e das retas que passam por $\overline{m_1 m_3}$ e $\overline{m_2 m_4}$. Na aplicação desenvolvida durante esta dissertação, os pontos m_i são selecionados pelo usuário, com a ajuda do *mouse*. Além destes pontos, o método necessita de um conhecimento prévio sobre a dimensão da grade formada pelas quinas do tabuleiro. Na aplicação desenvolvida, o tamanho em milímetros da casa do tabuleiro é inserido pelo usuário e este valor é calculado.

É considerada então uma grade regular com as dimensões definidas. As coordenadas dos pontos dispostos na grade são normalizadas para valores entre 0 e 1. Assim, tem-se um sistema de coordenadas onde cada quina corresponde a uma posição (u, v), onde u, $v \in [0, 1]$. A Figura 4.6(a) ilustra uma grade regular para um tabuleiro com dimensões 7×7.

A partir das Equações 4.1, são obtidos os pontos $p_1(s_0, 0)$, $p_2(s_1, 1)$, $p_3(0, t_0)$ e $p_4(1, t_1)$. A interseção entre os segmentos de reta $\overleftrightarrow{p_1 p_2}$ e $\overleftrightarrow{p_3 p_4}$ estima a posição em perspectiva de uma quina com coordenadas (u, v) na grade regular. Na Figura 4.6(b), a posição estimada de uma quina está ilustrada como um ponto verde, ao passo em que a posição dos pontos

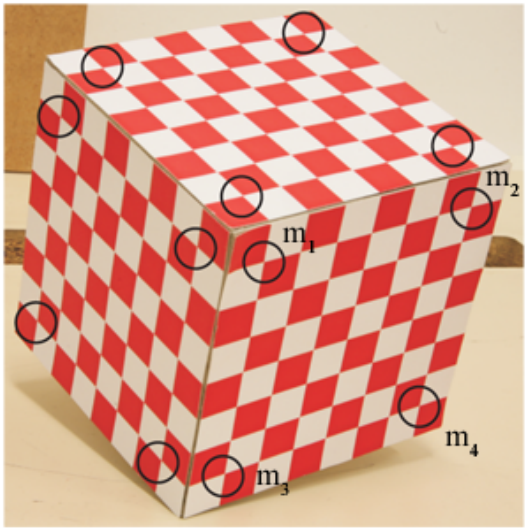


Figura 4.5: Imagem do objeto de calibração, capturada pela câmera fotográfica. As regiões destacadas correspondem às quatro quinas externas de cada tabuleiro.

obtidos em [26] através das Equações 4.1 estão em cinza. É importante observar que estes valores estão normalizados entre 0 e 1.

$$s_i = \frac{1}{1 + \frac{1-u}{u} \cdot \frac{\|m_{2i+2} - s\|}{\|m_{2i+1} - s\|}}, \quad t_i = \frac{1}{1 + \frac{1-v}{v} \cdot \frac{\|m_{i+1} - t\|}{\|m_{i+3} - t\|}}. \tag{4.1}$$

Para converter os pontos p_i para coordenadas da imagem, deve-se multiplicar cada um deles por uma semi-reta correspondente. Esta semi-reta é formada por duas quinas m_j e m_k $(j, k = 1, ..., 4)$, de forma que $\overrightarrow{m_j m_k}$ possui a mesma direção de p_i em relação ao tabuleiro. Por exemplo, o ponto p_3 será multiplicado pela semi-reta $\overrightarrow{m_1 m_3}$.

Encontradas as estimativas iniciais para cada quina, o passo seguinte é a utilização da biblioteca OpenCV para refinar a localização das quinas. É utilizada a função `cvFindCornerSubPix`, que efetua o refinamento com base no gradiente da região do ponto.

Tendo as correspondências entre pontos bidimensionais nas `imagens da câmera` e do `scanner`, o passo seguinte é obter os pontos tridimensionais correspondentes aos pontos 2D capturados pelo *scanner*. O *scanner* a laser captura imagens de profundidade e coloridas através do mesmo dispositivo óptico. Assim, estas imagens estão alinhadas, de forma que é possível mapear diretamente um ponto bidimensional da imagem colorida a

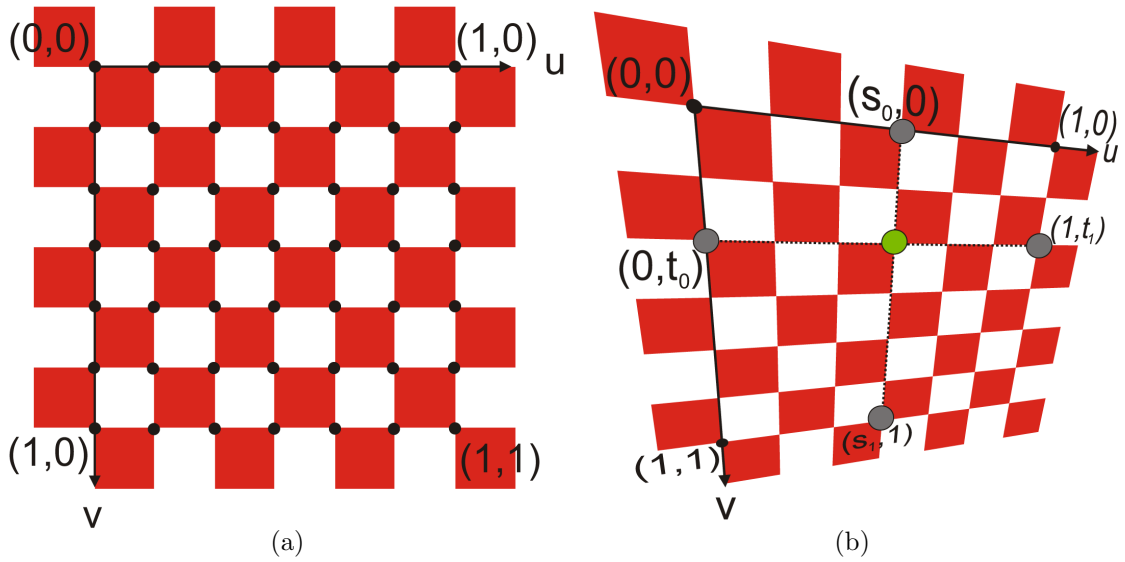


Figura 4.6: Tabuleiros normalizados: (a) quinas do tabuleiro (grade regular com dimensão 7×7); (b) tabuleiro em perspectiva. A posição da quina em verde é calculada a partir dos pontos de fuga e da posição da quina na grade regular da figura (a).

seu correspondente na imagem de profundidade. Devido à obtenção de pontos de correspondência com precisão sub-pixel, este mapeamento deixa de existir, pois as imagens coloridas capturadas pelo *scanner* são amostradas como valores inteiros.

Para efetuar o mapeamento, é feita uma interpolação nos pontos tridimensionais do *scanner*. Assim, com base nos quatro pixels vizinhos de cada ponto 2D fracionário, são mapeados os seus pontos tridimensionais correspondentes. Estes pixels vizinhos são obtidos a partir dos valores de teto e solo das coordenadas x e y do ponto 2D fracionário. O ponto 3D final será obtido a partir da interpolação destes pontos, onde os pesos aplicados são calculados com base no valor do ponto 2D fracionário.

Assim, a partir de uma imagem obtida pelo *scanner* e outra pela câmera, podem ser calculados os pontos de correspondência (quinas do tabuleiro) com precisão sub-pixel. Com base na localização dos pontos na imagem colorida do *scanner*, são interpolados os valores tridimensionais do ponto.

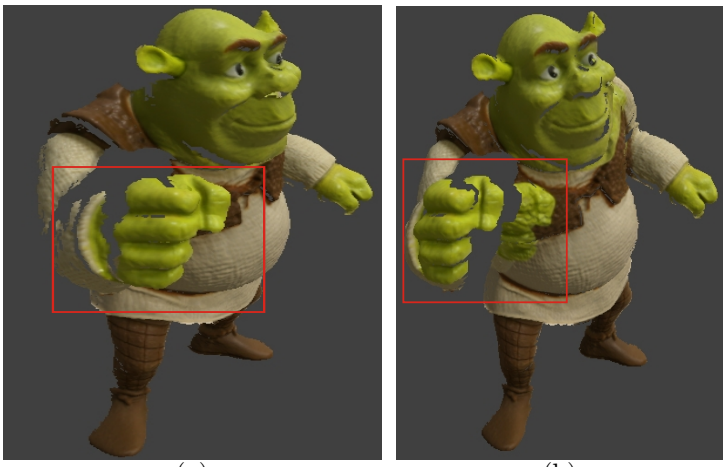
4.1.2 Tratamento da Oclusão

É possível perceber o problema da oclusão em objetos que apresentam saliências. As Figuras 4.7(a) e 4.7(b) mostram uma malha obtida a partir de uma `imagem de profundidade` de um brinquedo, renderizada utilizando uma `imagem da câmera` correspondente como textura. A Figura 4.7(a) mostra a malha sob a perspectiva da câmera, e é possível observar a mão do brinquedo ocultando uma parte do seu corpo. Girando a malha (Figura 4.7(b)), percebe-se que a região oclusa recebe a mesma textura da mão do brinquedo. Esse resultado pode ser comparado com a Figura 4.7(c), que mostra a mesma malha renderizada com uma textura obtida a partir de uma `imagem do scanner`, capturada através da mesma perspectiva que a `imagem de profundidade` que gerou a malha.

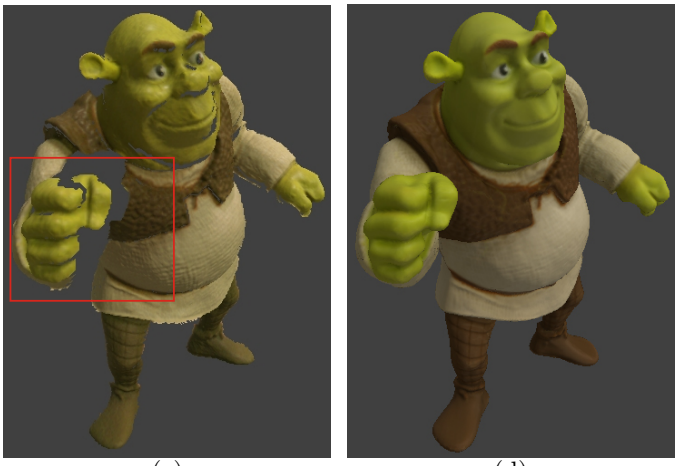
Esse problema ocorre durante a renderização das imagens na malha. Vértices que estão situados na mesma linha de perspectiva da câmera são mapeados pela matriz de transformação para a mesma posição no espaço 2D. Assim, durante a renderização, estes vértices são associados à mesma coordenada 2D da `imagem da câmera`, recebendo a mesma textura. Esse problema não acontece com as `imagens do scanner` pois elas são capturadas através da mesma perspectiva que as imagens de profundidade, utilizadas para gerar a malha.

Para superar esta limitação, foi desenvolvida neste trabalho uma forma de estimar a profundidade dos pontos 2D através da matriz de calibração. Desta forma, podem ser identificados os vértices oclusos a partir da perspectiva da câmera, não sendo atribuídas coordenadas de textura errôneas a eles.

Matrizes de calibração mapeiam pontos 3D para pontos 2D em uma transformação do tipo muitos para um. Assim, a operação inversa não determina uma posição no espaço 3D, e sim uma reta neste espaço. Isso faz com que a matriz de calibração seja insuficiente para descobrir oclusões em imagens 2D, pois através dela não é possível transformar pontos 2D em pontos 3D e analisar seu posicionamento. Neste trabalho, é proposta uma modificação da matriz de calibração, de forma que seja possível estimar a profundidade do ponto 2D



(a) (b)



(c) (d)

Figura 4.7: Exemplo do problema da oclusão: (a) e (b) diferentes perspectivas de uma vista, renderizadas utilizando uma `imagem da câmera`; (c) a vista renderizada com uma `imagem do scanner`; e (d) o modelo final com textura obtida a partir de `imagens da câmera`, utilizando o tratamento de oclusão.

em relação à perspectiva da câmera.

Inicialmente, deve-se considerar a representação 3×4 de uma matriz de calibração apresentada na Equação 4.2.

$$\begin{bmatrix} x_w \\ y_w \\ w \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{4.2}$$

A Equação 4.2 representa uma matriz C que projeta um ponto 3D $P(X, Y, Z)$ em

um ponto 2D da imagem $p^w(x_w, y_w, w)$ em coordenadas homogêneas. Para retornar as coordenadas em pixels, deve-se dividir x_w e y_w por w, obtendo o ponto 2D correspondente $p(x, y)$.

Interpretando os componentes da matriz de calibração de forma geométrica, pode-se definir os vetores 3D $C_X = [c_{11}\ c_{12}\ c_{13}]$ e $C_Y = [c_{21}\ c_{22}\ c_{23}]$ que possuirão, respectivamente, a mesma direção dos eixos X e Y da imagem. Assim, quando a matriz de calibração C é multiplicada pelo ponto P, é feita uma projeção de P no vetor C_X, seguida da adição de um fator de translação c_{14}, obtendo a coordenada X de p^w. De maneira similar, a coordenada Y de p^w é obtida através da projeção de P no vetor C_Y, seguido da adição de c_{24}. Desta forma, as seguintes expressões são obtidas:

$$\begin{aligned} x^w &= P \cdot C_X + c_{14}, \\ y^w &= P \cdot C_Y + c_{24}. \end{aligned} \tag{4.3}$$

Neste método serão utilizadas apenas as direções dos vetores; no entanto, deve-se ressaltar que C_X e C_Y estão associados a fatores de escala de *zoom*, transformação de perspectiva e de *viewport*.

Com base no conceito exposto, é proposta uma modificação da matriz de calibração, de forma a estimar a profundidade de um ponto 2D da imagem. A solução desenvolvida faz o produto vetorial entre os vetores C_X e C_Y, obtendo uma nova linha para a matriz de calibração que, devido às propriedades de C_X e C_Y, projeta P em um eixo que é ortogonal aos eixos X e Y da imagem, *i.e.* uma simulação do eixo Z.

Considerando a representação da matriz na Equação 4.2, a matriz de calibração modificada C' possui a formulação apresentada na Equação 4.4.

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ w \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c'_1 & c'_2 & c'_3 & 0 \\ c_{31} & c_{32} & c_{33} & c_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{4.4}$$

onde:

$c'_1 = c_{12} * c_{23} - c_{13} * c_{22}$,

$c'_2 = c_{13} * c_{21} - c_{11} * c_{23}$,

$c'_3 = c_{11} * c_{22} - c_{12} * c_{21}$.

O elemento na quarta coluna da terceira linha recebe o valor 0, pois representa o fator de translação em Z da transformação. De forma similar à matriz C, deve-se dividir x_w, y_w e z_w por w para obter as coordenadas em pixels $p'(x, y, z)$.

Assim, para dois pontos 3D que são mapeados no mesmo ponto 2D através da matriz de calibração, pode-se descobrir qual é o ocluso através do valor z obtido para cada ponto 2D. É importante enfatizar que, devido aos fatores de escala de C_X e C_Y, z não representa o valor real da profundidade do pixel, possuindo escala diferente. No entanto, ele mantém a monotonicidade da função, o que é suficiente para descobrir quando um pixel está na frente de outro.

Considerando neste trabalho que a origem do sistema de coordenadas da imagem é localizado no canto superior esquerdo da imagem, pode-se afirmar que os valores obtidos para z são menores quando P é mais próximo da câmera, e maiores quando P está mais longe. A Figura 4.7(d) mostra o modelo 3D final do brinquedo, renderizado utilizando `imagens da câmera`. O problema da oclusão foi corrigido através da utilização de matrizes de calibração modificadas, onde foram atribuídas coordenadas de textura somente aos vértices que não são oclusos na perspectiva da câmera.

4.2 Geração da Textura

O sistema de aquisição utilizado na geração de modelos 3D captura `triplas` de imagens do objeto a ser preservado (Seção 4.1). Cada `tripla` é composta de uma `imagem de profundidade`, uma `imagem do scanner` e uma `imagem da câmera`. A `imagem do scanner` e a `imagem de profundidade` estão no sistema de coordenadas do scanner, e uma matriz de calibração modificada efetua a transformação entre os sistemas de coordenadas do scanner e o sistema de coordenadas da câmera. Desta forma, uma `imagem de profundidade` pode ser mapeada para uma `imagem da câmera`, retornando também a profundidade dos pixels na `imagem da câmera` (Subseção 4.1.2).

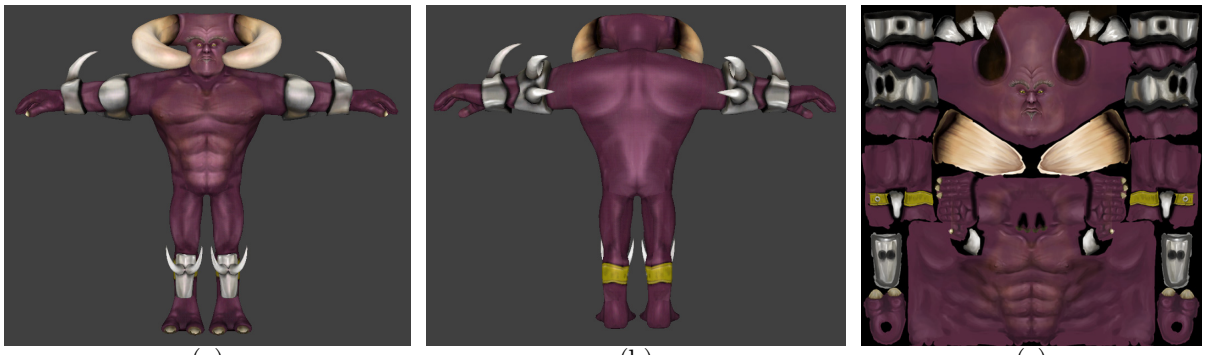
O objetivo na aquisição de dados é digitalizar a superfície do objeto, assim o ideal é que estas `triplas` consigam capturar toda a geometria de sua superfície, através das `imagens de profundidade`, e sua cor, através das `imagens do scanner e da câmera`. A geometria da superfície é calculada durante o processo de modelagem geométrica, descrito no Capítulo 2. Nesta dissertação foi desenvolvida uma técnica para de geração de texturas para o modelo 3D gerado, que pode posteriormente ser utilizada na parametrização do seu modelo fotométrico (Seção 3.2).

A etapa de geração de textura tem como objetivo calcular um mapa de textura que contém informações sobre a aparência do modelo. Para isso, são utilizados como parâmetros de entrada as `triplas` de imagens capturadas pelo sistema de aquisição e o modelo 3D do objeto, gerado através do processo apresentado na Subseção 2.6 [33]. A partir destes dados, são gerados mapas de textura parciais dos objetos, um para cada tripla de imagens. O mapa de textura final é obtido a partir da união dos mapas parciais, feita de acordo com um peso que reflete a confiabilidade de cada pixel nos mapas de textura parciais. A Subseção 4.2.1 detalha como estes mapas e seus pesos são gerados, ao passo que a Subseção 4.2.2 mostra como é feita a integração destes mapas e é gerada a textura final.

4.2.1 Mapas de Textura Parciais

Mapas de textura são estruturas de dados que contêm a descrição da aparência visual do objeto. Esses mapas são obtidos através da planificação do modelo 3D, feita através do mapeamento de cada vértice do modelo 3D para uma coordenada (coordenada de textura) em uma imagem 2D. Este mapeamento é feito através de uma função, onde cada triângulo do modelo é mapeado em uma região triangular no mapa de textura, que conterá a textura deste triângulo.

Para visualizar este conceito, as Figuras 4.8(a) e 4.8(b) mostram um modelo 3D de um monstro [4], e a Figura 4.8(c) mostra o seu mapa de textura, criado para o modelo 3D. É importante observar que a textura consiste em várias regiões do modelo, renderizadas separadamente e armazenadas no mapa de textura. Há diferentes formas de planificar um objeto 3D. A pesquisa apresentada em [13] revisa algumas das principais técnicas existentes na literatura.



(a) (b) (c)

Figura 4.8: Exemplo de modelo 3D e sua textura. (a) Vista frontal do modelo 3D de um monstro; (b) Vista traseira do modelo; (c) Mapa de textura do modelo. Tanto o modelo quanto a textura foram criados manualmente.

Neste trabalho, dada uma tripla de imagens de um objeto, o mapa de textura de uma `imagem de profundidade` (vista) é a `imagem da câmera` correspondente. A função que encontra as coordenadas de textura para os vértices da vista é a transformação efetuada pela matriz de calibração. Estas vistas renderizadas com `imagens da câmera` contêm informação de textura sobre uma parte do objeto, que corresponde à região vista durante

[4] Monstro criado no 3D Studio MAX para o jogo Inferno, desenvolvido pela empresa Continuum Entertainment (http://www.continuum.com.br).

a captura das imagens da tripla. Assim, a partir de um conjunto de trios, deve ser gerado um mapa de textura para o modelo 3D completo.

Neste trabalho é utilizado o método de planificação desenvolvido em [33]. Este método divide o objeto em regiões que possuem geometria quase planar, através da análise da normal dos vértices do modelo. Desta forma, um vértice pertence a uma região se o ângulo entre sua normal e a normal média da região estiver abaixo de um limiar. Uma região começa com um vértice e vai agregando outros até que todos os vértices vizinhos da região excedam o limiar. Definida uma região, uma nova região é iniciada com um vértice vizinho da região anterior. Após a classificação de todos os vértices, as regiões obtidas são mapeadas para regiões do mapa de textura.

Através do mapa de textura é possível mapear vértices do modelo a coordenadas de textura. No entanto, resta o problema de como obter a informação de textura que preencherá o mapa de textura. A solução encontrada neste trabalho consiste na geração de mapas de texturas parciais para cada `tripla` de imagens do objeto obtida pelo sistema de aquisição. O mapa de textura parcial corresponde a um mapa de textura do modelo 3D completo, porém que somente contém a textura dos triângulos vistos através da perspectiva da câmera fotográfica no momento da aquisição da `tripla`.

Um mapa de textura parcial é associado a uma máscara do mesmo tamanho que armazena o peso de cada pixel na textura final. O cálculo destes pesos é detalhado na Subseção 4.2.1.1. Se não há informação de cor para um dado pixel no mapa de textura parcial, o peso do pixel recebe valor 0.

São utilizadas imagens do tipo RGBA para representar os mapas de textura parciais. Os canais RGB armazenam a textura das faces vistas e o canal alfa, a máscara. A Figura 4.9 mostra um exemplo dos canais RGB e alfa de um mapa de textura parcial.

De acordo com o que foi apresentado neste capítulo, dada uma `tripla` i de imagens do objeto, pode-se mapear uma imagem de profundidade R_i a uma `imagem da câmera` P_i através de uma matriz de calibração modificada M_i. Para gerar um mapa de textura

(a) (b)

Figura 4.9: Mapa de textura parcial gerado a partir das faces vistas no mapa de faces da Figura 4.10(b): (a) canais RGB; e (b) o canal alfa, que corresponde ao peso de cada pixel em (a).

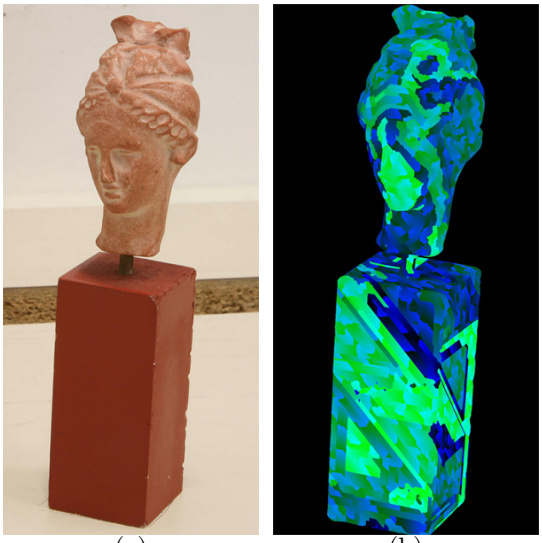
parcial para i, primeiro é desenhada uma imagem que mostra que faces do modelo final são vistas pela perspectiva da câmera em P_i. Neste trabalho, estas imagens são chamadas de `mapas de faces`.

O modelo 3D final é gerado através do processo descrito na Seção 2.6. Durante a geração do modelo geométrico, cada vista é projetada em um sistema de coordenadas único, para que sejam alinhadas. Desta forma, o modelo está em um sistema de coordenadas diferente do sistema de coordenadas das vistas, havendo uma matriz de transformação T_i que projeta a imagem de profundidade de i no sistema de coordenadas do modelo.

Um `mapa de faces` é gerado para i através da projeção de todas as faces do modelo 3D em uma textura associada a um *Z-buffer*. Essa projeção é feita através da aplicação da matriz inversa de T_i (projeta os vértices do modelo no sistema de coordenadas da vista i) seguida da aplicação da matriz M_i (transforma o sistema de coordenadas da vista no sistema de coordenadas da câmera). Desta forma, para cada vértice v do modelo, podemos obter suas coordenadas de textura (u, v, z) em P_i através da seguinte sequência de transformações:

$$(u, v, z) = M_i T_i^{-1} v. \tag{4.5}$$

Cada face cujos vértices foram transformados com a Equação 4.5 é desenhada no mapa de faces com uma cor que a identifica unicamente no modelo 3D. Paralelamente, o Z-buffer tratará a profundidade das faces desenhadas através do valor de z calculado para cada vértice da face. A Figura 4.10(a) mostra uma `imagem da câmera` de uma estátua, e a Figura 4.10(b) mostra as faces do modelo 3D que foram observadas através da perspectiva da câmera.



(a) (b)

Figura 4.10: Imagens da estátua: (a) `imagem da câmera`; e (b) o mapa de faces gerado para a `imagem da câmera`.

Percorrendo o `mapa de faces`, são identificadas as faces vistas a partir das cores encontradas. Essas faces são então projetadas em P_i com a Equação 4.5, e as regiões triangulares obtidas são desenhadas no canal RGB da textura parcial, na posição especificada pela função paramétrica.

Este mapa de textura, no entanto, contém muitas regiões com informação errônea de cor, afetadas por sombras ou especulares. Assim, são associados pesos aos mapas de textura parciais que indicam quão confiáveis são os pixels de cada face. O cálculo destes pesos é descrito em 4.2.1.1.

O método implementado utiliza as funções do *Z-buffer* do hardware gráfico para renderizar oclusões entre faces projetadas. Para utilizar o hardware gráfico para processar tex-

turas de alta resolução, associadas ou não a *Z-buffers*, é usada uma extensão do OpenGL[5] que suporta desenho e renderização *off-screen*.

4.2.1.1 Pesos de Confiabilidade

O peso de confiabilidade é calculado para cada vértice pertencente às faces observadas em uma vista, e é interpolado dentro da área da face na máscara (canal alfa) dos mapas de textura parciais. Este peso representa a confiabilidade da cor observada para um vértice, com base na observação de que vértices do modelo cuja normal aponta na direção da câmera possuem cores mais confiáveis.

O peso é calculado através do produto escalar entre o vetor que vai do vértice do modelo para a posição da câmera e a normal do vértice (Fig. 4.11). O resultado deste produto escalar varia entre -1 e 1. Valores abaixo de 0 representam vértices cuja normal aponta na direção oposta à posição da câmera. Esses vértices recebem peso 0, pois devido à direção de sua normal, pode-se assumir que eles estão situados na parte de trás do objeto, a partir da perspectiva da câmera. Para balancear a contribuição de vértices cuja normal aponta na direção da câmera, mas possuem cores errôneas (influenciadas por especulares ou sombras), o valor do peso é limitado. Assim, foi definido empiricamente que se o produto escalar possui um valor maior que 0.7, seu valor é limitado a 0.7.

Para calcular este peso, é necessário descobrir a posição do foco da câmera. Este foco é calculado a partir da matriz de calibração associada à `tripla` de imagens que originará o mapa de textura parcial, com base no método desenvolvido em [2]. A matriz contém informações sobre o posicionamento do foco da câmera, que pode ser obtido a partir do seguinte conceito: uma reta da imagem forma um plano quando projetada no espaço 3D; assim, se projetarmos duas retas que se interceptam, termos dois planos cuja interseção formará uma reta que passa pelo foco da câmera. Assim, se for calculado um conjunto destas retas, a posição do foco pode ser estimada através do ponto onde estas

[5]OpenGL Framebuffer Object Extension. Mais informações na página: http://www.opengl.org/registry/specs/EXT/framebuffer_object.

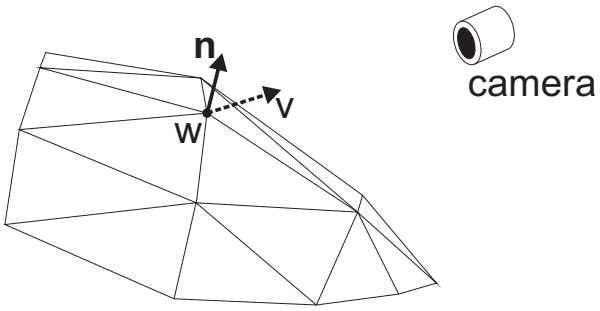


Figura 4.11: Perspectiva da câmera em um vértice w com normal **n**.

retas convergem.

Os parâmetros destas retas são obtidos a partir da matriz de calibração e de um conjunto de pontos 2D/3D correspondentes. O conjunto de pontos utilizado é o conjunto de pares de correspondência, obtido durante a calibração (Subseção 4.1.1).

Assim, considerando a matriz de calibração e um ponto 2D de correspondência $i(x, y)$, é preciso obter a direção da reta que passa por i e pelo foco da câmera. Para isso, primeiro devem ser obtidos os parâmetros dos dois planos. O primeiro plano (P_x) representa os vértices 3D que são mapeados na coluna x da imagem, e o segundo (P_y) representa os vértices 3D mapeados na coluna y. A interseção entre estes dois planos define a direção da reta que passa pelo ponto 3D correspondente a i e vai em direção ao foco da câmera. Assim, considerando a matriz de calibração da Equação 4.2, pode-se inferir que:

$$\begin{aligned} X * c_{11} + Y * c_{12} + Z * c_{13} + c_{14} = x_w \quad e \\ X * c_{31} + Y * c_{32} + Z * c_{33} + c_{34} = w, \end{aligned} \tag{4.6}$$

então:

$$\frac{X * c_{11} + Y * c_{12} + Z * c_{13} + c_{14}}{X * c_{31} + Y * c_{32} + Z * c_{33} + c_{34}} = x. \tag{4.7}$$

Isolando cada componente do ponto 3D em (4.7):

$$X * (c_{11} - x * c_{31}) + Y * (c_{12} - x * c_{32}) +$$
$$Z * (c_{13} + x * c_{33}) + (c_{14} - x * c_{34}) = 0. \tag{4.8}$$

A Equação 4.8 define a equação do plano P_x. Utilizando cálculo similar, é obtida a equação de P_y. A partir destas equações, são obtidos os parâmetros dos planos $P_x(a_1, b_1, c_1, d_1)$ e $P_y(a_2, b_2, c_2, d_2)$ da seguinte maneira:

$$\begin{aligned} a_1 &= c_{11} - c_{31} * x, \quad & a_2 &= c_{21} - c_{31} * y, \\ b_1 &= c_{12} - c_{32} * x, \quad & b_2 &= c_{22} - c_{32} * y, \\ c_1 &= c_{13} - c_{33} * x, \quad & c_2 &= c_{23} - c_{33} * y, \\ d_1 &= c_{14} - c_{34} * x, \quad & d_2 &= c_{24} - c_{34} * y. \end{aligned} \tag{4.9}$$

Finalmente, a interseção entre estes dois planos é obtida através do produto vetorial entre as normais dos planos. Isso resultará no vetor $\vec{v}(v_x, v_y, v_z)$:

$$\begin{aligned} v_x &= b_1 * c_2 - b_2 * c_1, \\ v_y &= c_1 * a_2 - c_2 * a_1, \\ v_z &= a_1 * b_2 - a_2 * b_1. \end{aligned} \tag{4.10}$$

O vetor de direção $\vec{v}$ e o ponto 3D correspondente de i são utilizados para definir a reta que passa através de i em direção ao ponto do foco. Idealmente, quaisquer duas destas retas se interceptam no ponto de foco, mas devido a erros de calibração, elas geralmente não se interceptam umas com as outras.

Para lidar com este problema, é calculada a distância entre as duas retas. O ponto médio do segmento de reta resultante é utilizado como aproximação da interseção entre as duas retas (ver ponto p na Figura 4.12). Como este resultado é por natureza uma

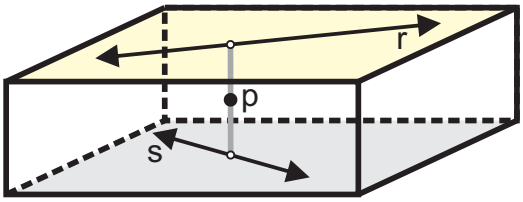


Figura 4.12: Ponto médio do segmento de reta formado pela menor distância entre r e s.

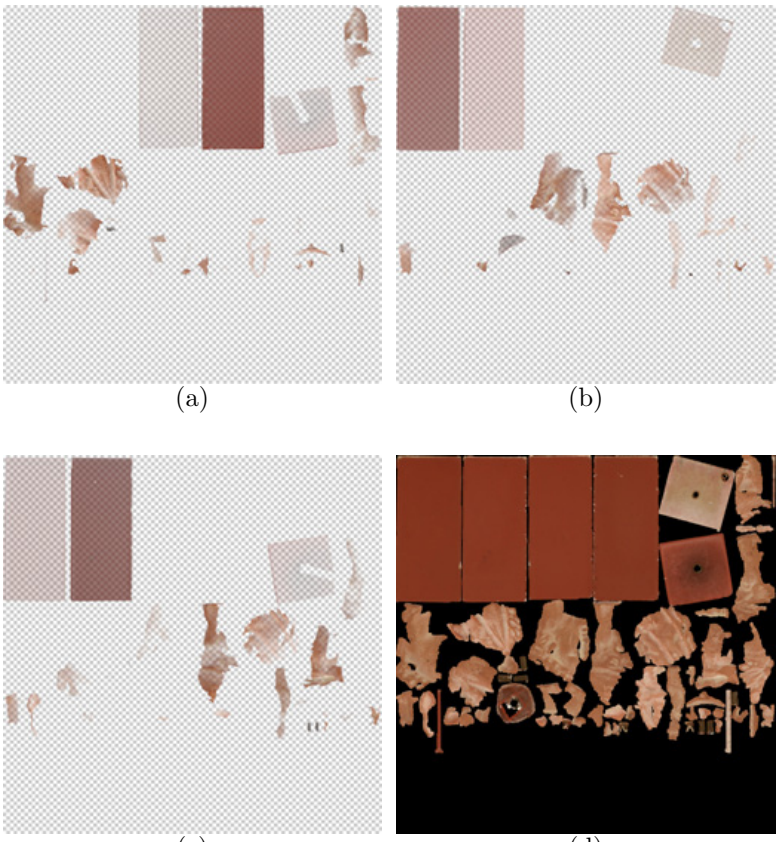
aproximação, para cada reta calculada é obtida sua interseção aproximada com todas as outras. Isso gerará uma nuvem de pontos, onde o ponto médio desta nuvem é a posição aproximada do foco da câmera.

4.2.2 Integração das Texturas Parciais

Os mapas de textura parciais utilizam a mesma função de parametrização do modelo 3D (ver Seção 4.2.1). Assim, cada pixel da textura final é calculado a partir da soma ponderada dos pixels na mesma coordenada nas texturas parciais. O peso utilizado na soma representa a confiabilidade do pixel (Subseção 4.2.1.1).

Os mapas de textura parciais ocupam um grande espaço em disco porque contêm quatro canais, têm alta resolução, e são gerados para cada vista do objeto. Para que todos eles sejam carregados na memória ao mesmo tempo, as texturas parciais são divididas em blocos. Durante a união dos mapas, é carregado o mesmo bloco em todas as texturas parciais. É feita a soma ponderada dos blocos, e o bloco resultante é escrito diretamente no arquivo do mapa de textura final. A Figura 4.13 apresenta 3 de 27 vistas parciais da estátua exibida na Figura 4.10(a), e o mapa de textura final.

Durante a análise dos resultados, foi notada a existência de erros de projeção nas transformações efetuadas pelas matrizes utilizadas na geração das texturas parciais. Esses erros prejudicam a textura dos modelos, e são gerados durante o cálculo das matrizes que transformam o sistema de coordenadas de cada vista i no sistema de coordenadas do modelo (matrizes T_i, calculadas durante a fase de alinhamento), e durante o cálculo das matrizes de calibração M_i. Uma análise numérica dos erros gerados pelas matrizes de



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

Figura 4.13: Mapas de textura da estátua: (a), (b) e (c) 3 de 27 mapas de textura parciais; (d) mapa de textura final.

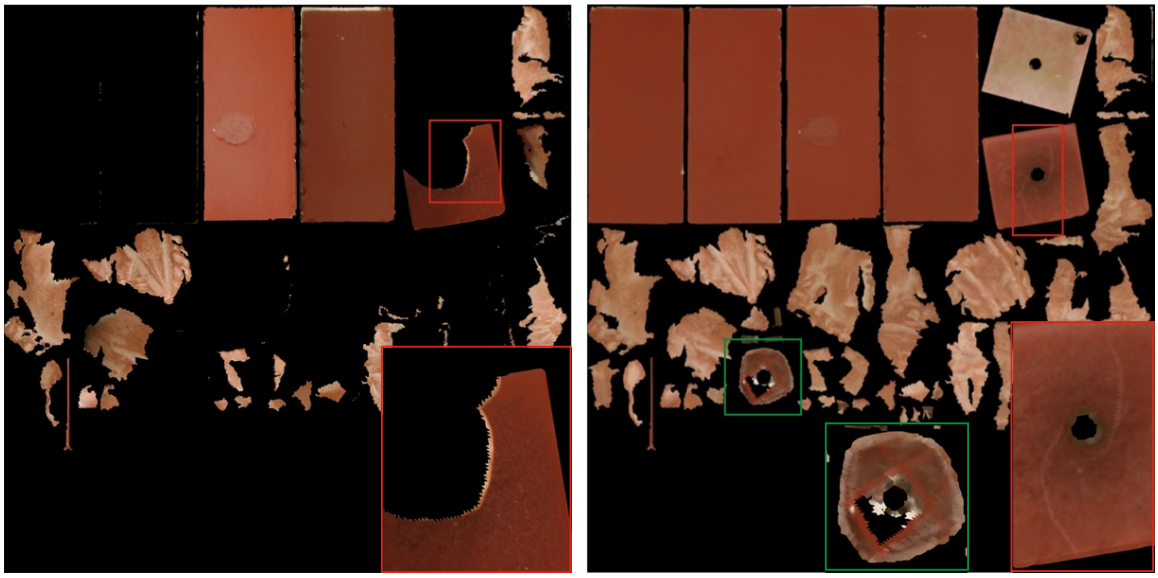
calibração é feita na Subseção 5.2.1.

Esses erros na projeção são em sua maioria aceitáveis, porém em vistas que apresentam regiões oclusas o deslocamento dos pixels causa uma projeção de pixels da região que está na frente para a região oclusa. Esse defeito aparece nos mapas de textura parciais, e gera cores incorretas que serão atenuadas após a integração, porém afetarão a qualidade da textura do modelo final.

Como exemplo, seja o modelo ilustrado na Figura 4.14, onde pode-se perceber a textura incorreta na região destacada. Esse erro tem origem no mapa de textura parcial (região destacada da Figura 4.15(a)), e é atenuado após a integração, no mapa de textura final (região destacada em vermelho na Figura 4.15(b)), porém não desaparece. Inclusive, pode-se perceber mais erros gerados por outros mapas de textura parciais incorretos na região destacada em verde.

(a) (b)

Figura 4.14: Exemplo do erro gerado em modelos 3D com texturas obtidas a partir de `imagens da câmera`. (a) modelo com textura errônea; e (b) detalhe da Figura (a). O modelo corrigido é exibido na Figura 4.17.



(a) (b)

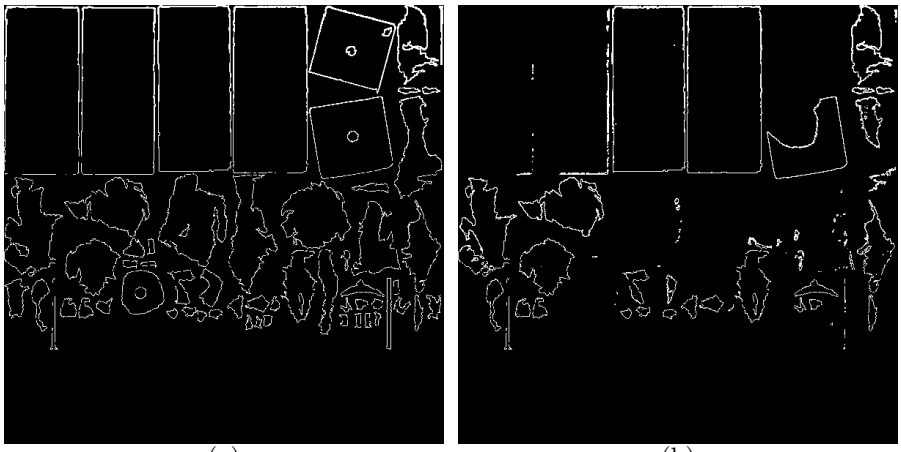
Figura 4.15: Mapas de textura da estátua, com erros de projeção destacados: (a) mapa de textura parcial; e (b) mapa de textura final.

Assim, foi implementado um método de correção destes erros nas vistas parciais que faz uso da função de parametrização utilizada. O método é aplicado nas vistas parciais, e se baseia na comparação entre as regiões do mapa de textura final, e as regiões preenchidas nos mapas de textura parciais. Considerando que as regiões dos mapas de textura correspondem a superfícies quase planares do modelo 3D, pode-se assumir que quando

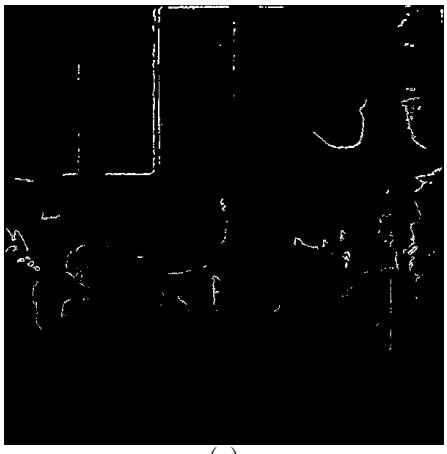
uma região é parcialmente preenchida nos mapas de textura parciais, ela está sendo oclusa por outra.

No método implementado, inicialmente é utilizado o filtro Canny [5] para a detecção das bordas das regiões preenchidas nos mapas de textura parcial (Figura 4.16(b)) e final (Figura 4.16(a)). Essas bordas são então subtraídas, de forma a encontrar apenas as bordas que correspondem às oclusões (Figura 4.16(c)).

Encontradas as bordas de oclusão, o passo seguinte é alterar o mapa de textura parcial, de forma a diminuir o peso dos pixels nessas regiões. Os pixels mais próximos destas bordas têm seus pesos reduzidos. Essa redução de peso vai diminuindo conforme a distância da borda aumentar, de forma que após um limiar de distância, os pixels não terão seus pesos alterados.



(a) (b)



(c)

Figura 4.16: Bordas obtidas a partir dos mapas de textura: (a) bordas do mapa de textura completo; (b) bordas do mapa de textura parcial; (c) subtração entre bordas de (a) e (b).

Essa abordagem apresentou bons resultados, como pode ser visto na Figura 4.17, que apresenta o mesmo modelo da Figura 4.14, porém com a correção das bordas de oclusão.

É importante observar que, no momento da integração, tem-se um conjunto de vistas parciais para o modelo 3D. Dada uma coordenada de pixel, temos a coloração deste pixel em cada textura parcial, e podemos observar a cor que ele recebe em cada vista do objeto. Conforme foi apresentado neste capítulo, esse pixel corresponde a um ponto do modelo, que pode estar situado em um de seus vértices, ou dentro de uma face.

As cores observadas para cada ponto do modelo são utilizadas na parametrização do modelo fotométrico do objeto. Assim, juntamente com informações sobre a iluminação do ambiente, os resultados obtidos a partir do algoritmo desenvolvido neste trabalho podem ser utilizados para parametrizar o modelo fotométrico do objeto, através de métodos como os apresentados na Seção 3.2.

Figura 4.17: Detalhe do modelo 3D com erro de textura corrigido.

CAPÍTULO 5

RESULTADOS

Uma das metas deste trabalho é, a partir do estudo e avaliação de técnicas existentes na Computação Gráfica, criar um conjunto de ferramentas que viabilize desde a captura da imagem da câmera digital até a geração da textura de um objeto.

Neste capítulo serão apresentados os aplicativos desenvolvidos ou adaptados para implementar cada etapa do algoritmo de geração de textura detalhado no Capítulo 4. O processo desenvolvido é integrado sempre que possível ao processo desenvolvido por [33], de forma a melhorar e aumentar os recursos disponíveis no processo de geração de modelos 3D.

As implementações deste trabalho foram desenvolvidas utilizando a linguagem de programação C++. São utilizados os conjuntos de bibliotecas OpenCV e OpenGL [1] para efetuar operações básicas de Computação Gráfica e Processamento de Imagens. A codificação foi feita nos ambientes Microsoft Visual Studio C++ 6.0 [2] e Borland C++ Builder 6.0 [3].

Além das implementações, serão validados e exibidos os resultados de cada etapa. Desta forma, a Seção 5.1 apresenta os resultados e o aplicativo utilizado na etapa de aquisição de dados; a Seção 5.2, a etapa de calibração; e a Seção 5.3, a etapa de geração de textura.

[1] OpenGL: Silicon Graphics International (http://www.opengl.org).
[2] Microsoft Visual Studio C++ 6.0: Microsoft Corporation (msdn.microsoft.com/vstudio/).
[3] Borland C++ Builder 6.0: Borland Software Corporation (http://www.borland.com/).

5.1 Aquisição de Dados

Nesta etapa é utilizado um aplicativo implementado em [33], o VividControl, para capturar as `imagens de profundidade` e as `imagens do scanner`. Com a inserção da câmera fotográfica no sistema de aquisição, esse aplicativo foi adaptado de forma a também capturar imagens coloridas da câmera fotográfica.

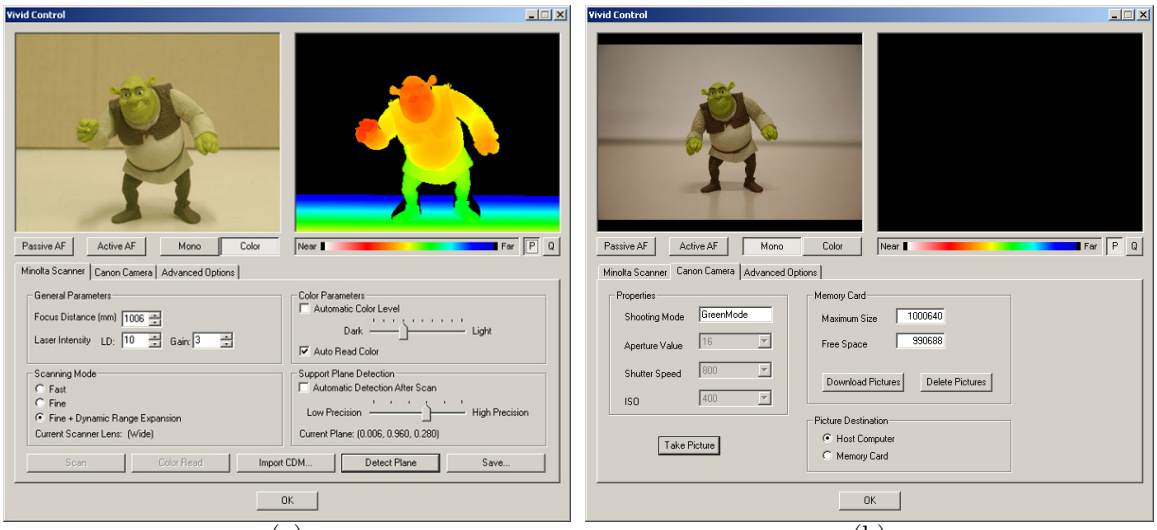
O aplicativo original envia comandos a partir de um computador para o *scanner* a *laser* via uma placa SCSI II. Ele permite a configuração da intensidade do laser, o modo de captura, entre outras funcionalidades. O aplicativo desenvolvido também permite a eliminação do fundo nas imagens de profundidade, eliminando assim pontos 3D que não correspondem ao objeto.

Neste trabalho, o VividControl foi adaptado para também se comunicar com a câmera fotográfica via um cabo USB. Assim, a partir do aplicativo, são enviados comandos de ajuste da abertura, velocidade e ISO da câmera, bem como para configuração do meio onde as fotos serão gravadas (no computador ou no cartão de memória da câmera). Para executar estes comandos através da linguagem C++, foi utilizado o SDK desenvolvido pela Canon [6].

A utilização de um mesmo aplicativo para capturar as imagens da câmera e do *scanner* simplificam o processo de aquisição, já que não é necessário manipular os dispositivos diretamente para utilizá-los. Na Figura 5.1 são exibidas imagens do aplicativo desenvolvido, onde a Figura 5.1(a) apresenta a aba de recursos para o *scanner* e a Figura 5.1(b), a aba da câmera. Nesta fase, o produto resultante é um conjunto de `triplas` do objeto de calibração e do patrimônio, que será utilizado nas fases seguintes.

5.2 Calibração

Neste trabalho foi desenvolvido um aplicativo para efetuar a etapa de calibração entre a câmera e o *scanner*. Ele é dividido em duas fases: a detecção dos pontos, e o cálculo da



(a) (b)

Figura 5.1: Aplicativo de aquisição de dados: (a) aba de captura de dados do *scanner*; e (b) aba de captura dos dados da câmera.

matriz de calibração.

Na detecção de pontos são utilizadas as `triplas` de imagens do objeto de calibração. Inicialmente, as imagens devem ser carregadas. A seguir, é dada a opção de detecção manual ou automática de quinas. Essa opção é definida em uma janela de configuração, onde também são definidos o número de tabuleiros existentes na imagem e a dimensão deles.

Na detecção manual, as quinas são selecionadas através de cliques do *mouse*. Após a seleção, a posição das quinas são refinadas através do método do `cvFindCornerSubPix`, da biblioteca OpenCV. Nesta forma de seleção é necessário que também sejam selecionadas as quinas externas de cada tabuleiro, pois estas serão utilizadas para calcular um parâmetro de entrada da função `cvFindCornerSubPix`.

Na detecção automática, são selecionadas com o *mouse* apenas as quinas externas dos tabuleiros. São calculadas as coordenadas de todas as quinas através do método descrito em [26], e estas são refinadas através da função `cvFindCornerSubPix`, da biblioteca OpenCV.

Para selecionar um dado ponto de uma imagem, é necessário que o usuário clique sobre ele na imagem desejada. Se ele clicar com o botão esquerdo, o ponto selecionado é classi-

ficado como ponto de correspondência. Caso clique com o direito, ele é classificado como quina externa. As coordenadas nos pontos são exibidas em uma tabela. Na Figura 5.2 é exibida uma imagem do aplicativo após a detecção dos pontos.

Os pontos detectados são exibidos nas imagens. Após a seleção dos pontos e seu refinamento, o usuário deve salvar os dados obtidos. Neste processo os pontos 3D do *scanner* são obtidos por interpolação e salvos em um arquivo juntamente com os pontos 2D correspondentes da câmera. Este arquivo contém também o caminho e nome dos arquivos que contêm a imagem da câmera e a imagem de profundidade do *scanner*, e é utilizado como parâmetro de entrada no cálculo da matriz de calibração.

O cálculo das matrizes é feito em outra aba do aplicativo (Figura 5.3). Nela, o arquivo que contém os pontos de correspondência que gerarão a matriz deve ser carregado. Após o carregamento dos pontos, deve ser escolhido o método de calibração desejado. As opções que geram matrizes de calibração modificadas (ver Subseção 4.1.2) são os métodos de calibração simples (utiliza seis pontos de correspondência), mínimos quadrados e mínimos quadrados com MSAC.

Os métodos de Faugeras e Faugeras Interativo foram implementados, porém apresentam resultados incorretos com os conjuntos de pontos de correspondência utilizados. Para validar a implementação, foi feita uma comparação com os resultados obtidos através do programa desenvolvido por Salvi durante o trabalho [28], que calcula as matrizes de transformação e os parâmetros adicionais a partir do arquivo com pontos correspondentes. Neste programa, as técnicas de Faugeras e Faugeras Interativo também obtiveram resultados inconsistentes, de forma que tais técnicas não serão avaliadas. Na Subseção 5.2.1 é apresentada uma análise dos métodos de calibração utilizados neste trabalho.

Após a seleção do método de calibração desejado, é calculada a matriz de calibração modificada, e são exibidos os erros obtidos com o método escolhido. Com isso, é possível testar os métodos disponíveis até obter o que exibiu melhores resultados.

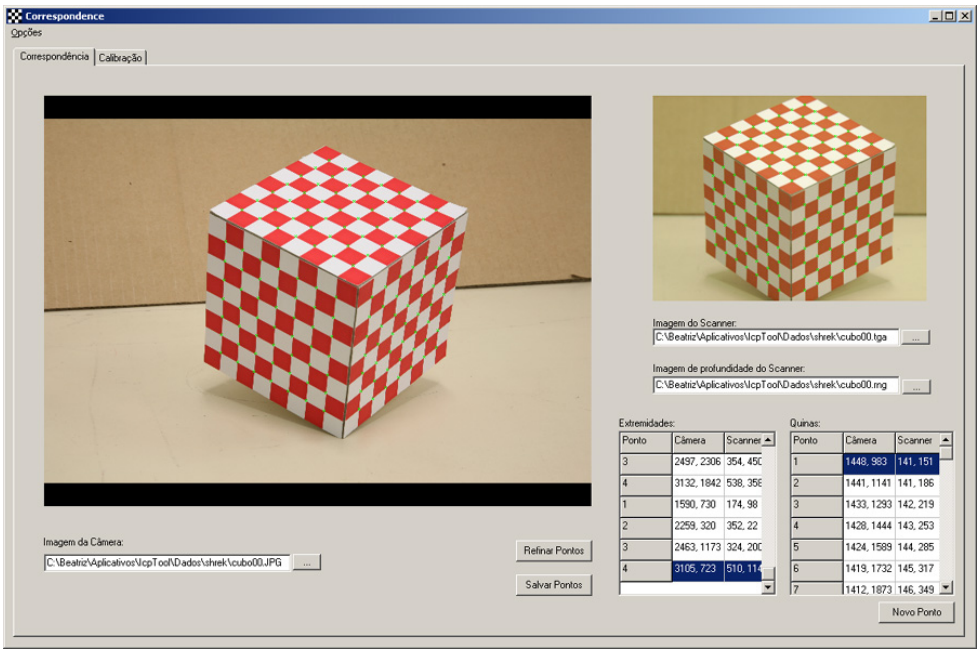


Figura 5.2: Janela de aquisição de pontos do aplicativo de calibração.

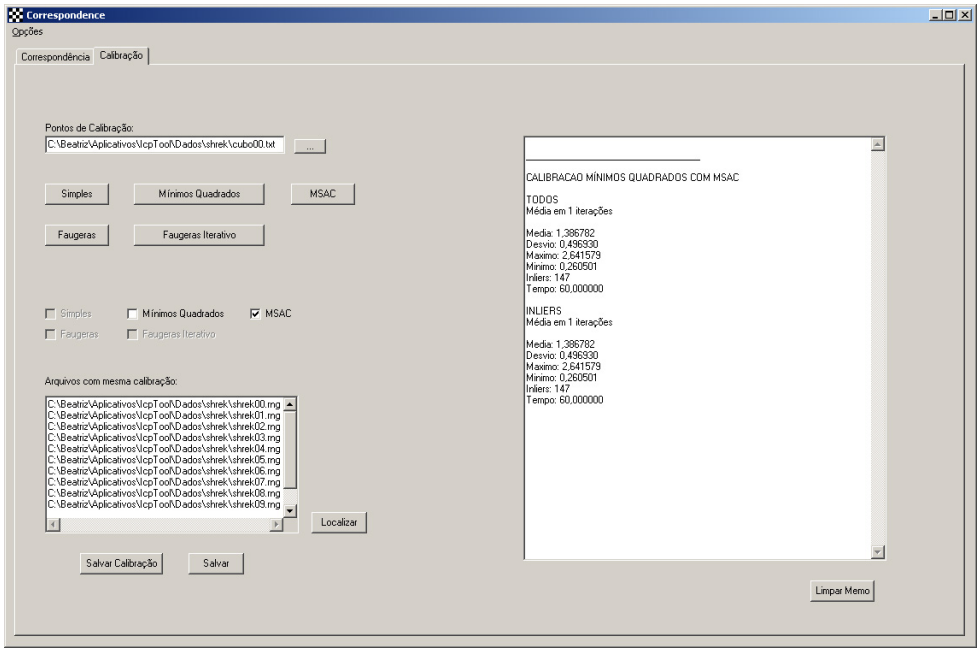


Figura 5.3: Aplicativo de calibração: janela de cálculo das matrizes de calibração.

Escolhido o método de calibração, deve-se associar no programa a matriz de calibração obtida com as `imagens de profundidade` pertencentes às `triplas` de imagens associados à calibração (ver Seção 4.1). Desta forma, para uma dada `tripla`, tal matriz irá efetuar o mapeamento entre os pontos da `imagem de profundidade` e da `imagem da câmera` desta

tripla. As associações são salvas em um arquivo que contém os valores da matriz, a posição calculada para o foco da câmera na vista (ver Subseção 4.2.1.1), e o nome dos arquivos que contêm as imagens de profundidade. Este arquivo será posteriormente utilizado na geração do modelo 3D.

5.2.1 Análise dos Métodos de Calibração

Foram utilizadas duas formas de análise dos resultados dos métodos de calibração. A primeira forma é numérica, onde inicialmente é obtida a matriz de calibração a partir de um conjunto de pontos correspondentes C_P. Esta matriz é então aplicada em todos os pontos tridimensionais, obtendo-se assim valores bidimensionais estimados. Os erros são então avaliados com base na distância euclidiana entre o valor calculado e o valor real do ponto bidimensional correspondente. A segunda forma é gráfica, onde uma matriz de calibração é utilizada para projetar uma imagem da câmera em uma imagem de profundidade do *scanner*.

Para obter o número de subconjuntos N_S avaliados no MSAC (Equação 3.5), foram feitos testes do algoritmo em três conjuntos de pontos onde todas as possíveis combinações são avaliadas. Estes conjuntos foram obtidos através de experimentos realizados com a câmera e com o *scanner*. O tamanho de cada subconjunto (N_C) é assumido como 6, visto que este é o menor número inteiro de pontos que pode ser utilizado para efetuar a calibração de uma matriz.

A partir destes testes, foi estimado um valor para a variável ϵ, que corresponde à porcentagem de falsas correspondências. Para validar os resultados, nesta análise foram selecionados conjuntos com 49 pontos com média de 38.666 pontos verdadeiros (*inliers*). Com isso, temos que ϵ possui valor 0.2109. A probabilidade p de que ao menos uma das soluções aleatórias não contenha erros foi assumida empiricamente como 0.9999999999. Com estes valores, é então calculado o número de subconjuntos adequado, através da Equação 3.5. Como resultado, o valor obtido foi $N_S = 83,33009, \lceil N_S \rceil = 84$.

Tabela 5.1: Resultados do algoritmo MSAC ao ser aplicado sobre $C_{49,6}$ combinações de pontos. Os erros foram calculados sobre os pontos de cada conjunto.

	Conjunto 1	**Conjunto 2**	**Conjunto 3**	**Média**
Número de *Inliers*	38	37	41	38.666667
Média das Distâncias (pixels)	1,545141	1,613533	2,111014	1,756563
Desvio Padrão (pixels)	0,934188	0,994205	1,579260	1,169218
Distância Máxima (pixels)	4,380228	4,451014	6,565209	5,132150
Distância Mínima (pixels)	0,189010	0,277137	0,203911	0,223353
Tempo	37'48"	33'57"	33'45"	35'10"

Tabela 5.2: Resultados da calibração com o algoritmo MSAC, com N_S = 84. Os erros foram calculados sobre todos os pontos de cada conjunto.

	Conjunto 1	**Conjunto 2**	**Conjunto 3**	**Média**
Número de *Inliers*	33	32	32	32,333333
Média das Distâncias (pixels)	1,713663	1,689528	2,269944	1,891045
Desvio Padrão (pixels)	1,168138	1,100931	1,695049	1,321373
Valor Máximo (pixels)	4,923235	4,799325	7,706308	5,809623
Valor Mínimo (pixels)	0,171000	0,261647	0,242807	0,225151
Tempo	0,012"	0,012"	0,015"	0,013673"

Nas Tabelas 5.1 e 5.2 são exibidos os resultados obtidos com a avaliação de todos os subconjuntos de pontos possíveis ($C_{49,6}$ pontos), e com N_S = 84 subconjuntos, respectivamente. Como a escolha dos subconjuntos é aleatória no segundo caso, os resultados exibidos na Tabela 5.2 são uma média dos resultados de 200 execuções do algoritmo MSAC com N_S subconjuntos. Em ambas as tabelas os erros foram calculados sobre o conjunto total de pontos, incluindo os *outliers*.

Como pode ser observado, a aplicação do algoritmo MSAC em todos os subconjuntos de pontos apresenta melhores resultados, porém é mais custosa, levando em média 35 minutos para ser executada em cada conjunto de pontos, utilizando um computador com 512 MB de memória RAM, e um processador Intel Pentium 4 com 2.67 GHz. Por outro lado, apesar de obter resultados um pouco inferiores, a aplicação do MSAC em um subconjunto com N_S pontos leva em média 0.013 segundos. Considerando que a diferença entre os resultados não compensa o tempo de processamento dispensado, neste trabalho o MSAC é utilizado com um subconjunto N_S de pontos.

A calibração com MSAC obteve uma média de resultados melhor do que calibração com mínimos quadrados. Isto pode ser observado durante a comparação das Tabelas 5.2 e 5.3. Na Tabela 5.3 são apresentados os resultados da calibração mínimos quadrados com os três conjuntos de pontos de teste. O tempo de processamento da calibração com

Tabela 5.3: Resultados da calibração com mínimos quadrados.

	Conjunto 1	**Conjunto 2**	**Conjunto 3**	**Média**
Média das Distâncias (pixels)	1.609387	1.839630	3.101207	2,183408
Desvio Padrão (pixels)	0.801000	0.982458	1.759128	1,180862
Distância Máxima (pixels)	3.086098	4.445644	8.244014	5,258585
Distância Mínima (pixels)	0.186626	0.425664	0.487898	0,366729

Tabela 5.4: Resultados da calibração com o algoritmo MSAC, com N_S = 84. Os erros foram calculados somente sobre os *inliers*.

	Conjunto 1	**Conjunto 2**	**Conjunto 3**	**Média**
Número de *Inliers*	33	32	32	32,333333
Média das Distâncias (pixels)	1,068749	1,025260	1,371214	1,155074
Desvio Padrão (pixels)	0,516881	0,408909	0,691858	0,539216
Valor Máximo (pixels)	2,186787	1,991386	3,019900	2,399358
Valor Mínimo (pixels)	0,171000	0,262082	0,243522	0,225535
Tempo	0,0127	0,0125	0,0159	0,013673

mínimos quadrados não foi detalhado na tabela pois tende a zero para todos os conjuntos de pontos. Pelo fato de desconsiderar *outliers*, a matriz de calibração obtida através do MSAC representa melhor o comportamento dos pontos, ao passo que a calibração por mínimos quadrados tenta se ajustar a todos, fazendo assim que *outliers* sejam considerados no cálculo e afetem a matriz de calibração.

É importante observar que nas Tabelas 5.1 e 5.2 todos os pontos, incluindo os *outliers*, são considerados no cálculo dos erros. Para uma avaliação dos resultados apenas com os *inliers*, ver Tabela 5.4. Nesta tabela é exibida a média dos resultados de 200 iterações do MSAC para cada conjunto de pontos. A diferença de resultados entre as tabelas 5.2 e 5.4 se deve ao fato de que, na primeira, *outliers* que não foram utilizados para gerar a matriz são considerados no cálculo de erro, afetando assim o resultado final. É interessante, porém, observar que mesmo calculando o erro com pontos considerados incorretos, o MSAC apresenta, de maneira geral, resultados melhores do que a calibração com mínimos quadrados.

Na análise gráfica, podem ser visualizados os resultados da aplicação de matrizes de calibração em vistas de alguns objetos. Nas duas primeiras linhas da Figura 5.4 são exibidas imagens de objetos renderizados com `imagens da câmera`. Nas Figuras 5.4(a) e 5.4(b), foi utilizado o método mínimos quadrados para gerar a matriz que projeta a `imagem da câmera` na vista 3D. Nas Figuras 5.4(c) e 5.4(d), o método utilizado foi o

MSAC. Em ambos os casos, foram utilizados conjuntos com 147 pontos para gerar as matrizes de calibração. Como pode ser notado, não há uma diferença significativa entre estas imagens, de forma que a análise numérica dos métodos se mostra mais precisa na validação dos resultados.

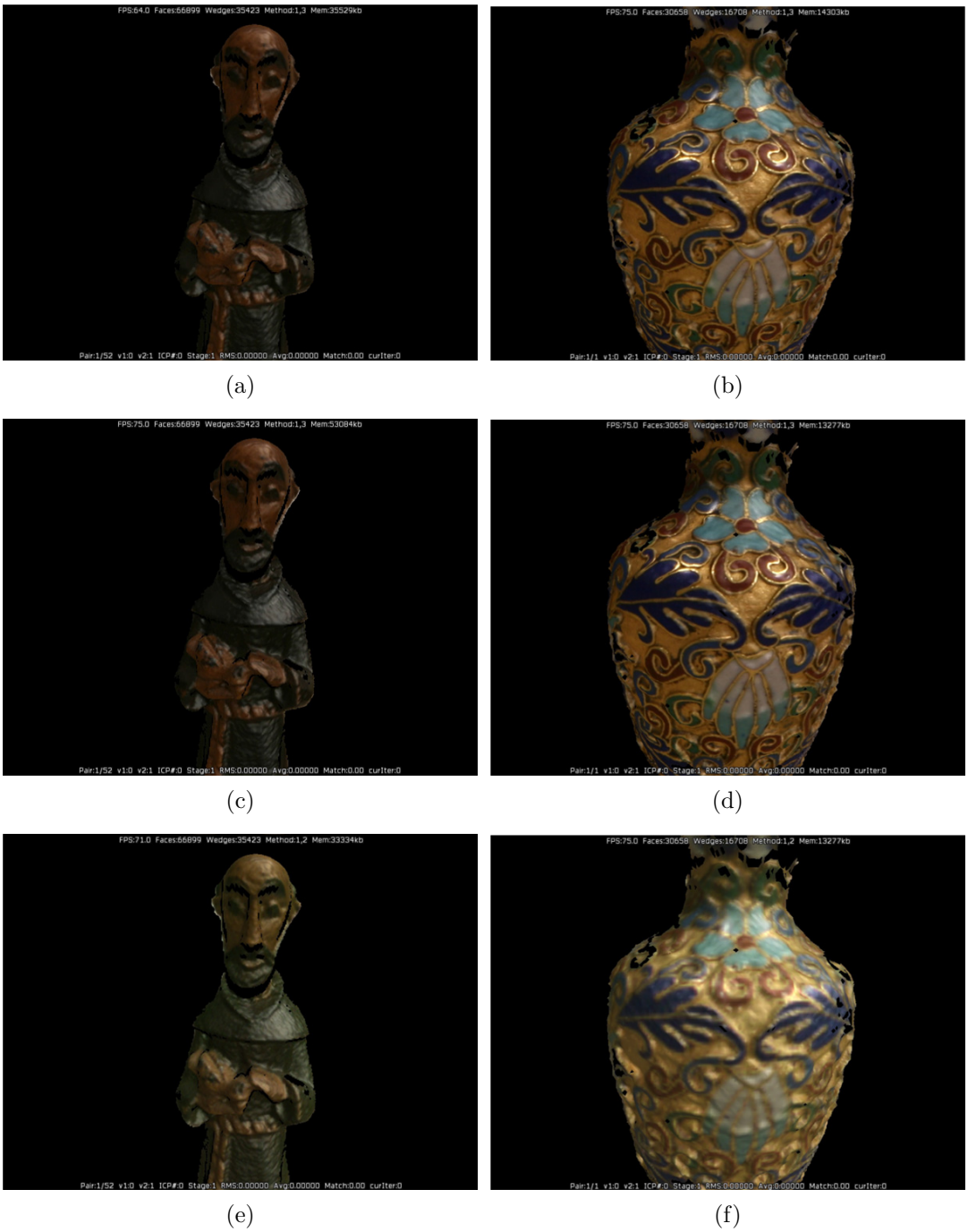


Figura 5.4: Resultados obtidos com a aplicação de diferentes imagens e transformações: (a) e (b) objetos renderizados com `imagens da câmera` e transformação mínimos quadrados; (c) e (d) objetos renderizados com `imagens da câmera` e MSAC; e (e) e (f) objetos renderizados com `imagens do scanner`.

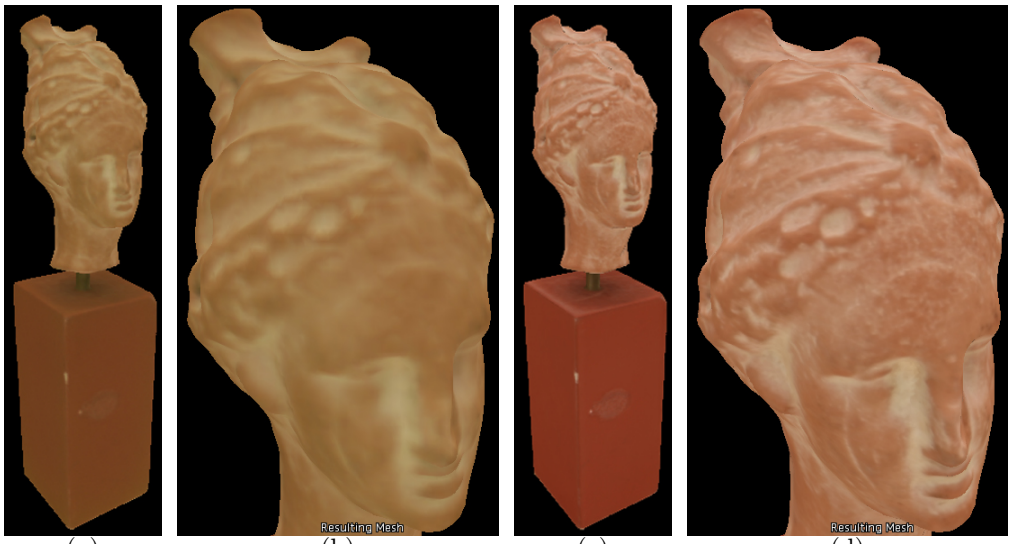
Como era esperado, em comparação com as `imagens do scanner`, o uso das `imagens da câmera` apresentaram um melhora substancial. Nas Figuras 5.4(e) e 5.4(f) são exibidas vistas dos objetos renderizadas com `imagens do scanner`. É notado um aumento visível nos detalhes e uma maior definição de cor, que preserva com maior fidelidade as características originais do objeto.

5.3 Geração de Textura

Nesta fase é utilizado outro aplicativo desenvolvido em [33]. Este, o IcpTool, tem como objetivo a geração do modelo 3D do objeto. Este aplicativo recebe como entrada um arquivo que especifica as imagens obtidas pelo scanner (`imagem de profundidade` e `imagens do scanner`), os pares de imagens de profundidade que são sobrepostas, e um conjunto de parâmetros de configuração para os algoritmos utilizados durante a modelagem geométrica. O IcpTool implementa o processo descrito na Seção 2.6, gerando um modelo 3D do objeto a partir da entrada descrita. Além do modelo 3D, o aplicativo também gera uma textura para este modelo a partir das `imagens do scanner`.

Neste trabalho, este aplicativo foi modificado de forma a aceitar também como parâmetro dados necessários para a geração de textura de alta qualidade (arquivo de saída da etapa de calibração e `imagens da câmera`). Assim, utilizando estes dados, as `imagens da câmera` são aplicadas no modelo 3D através do algoritmo apresentado no Capítulo 4.

O aplicativo modificado foi utilizado na preservação de diferentes objetos, com o objetivo de validar a abordagem desenvolvida. Assim, nesta seção são exibidas comparações entre texturas geradas com `imagens da câmera` e `do scanner`. Na Figura 5.5 são exibidas imagens renderizadas do modelo 3D de uma estátua, e a Figura 5.6 apresenta as imagens de uma ave de cerâmica.



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

Figura 5.5: Imagens renderizadas do modelo 3D da estátua: (a) e (b) utilizando `imagens do scanner` como textura; e (c) e (d) utilizando `imagens da câmera` como textura.

(a)

(b)

Figura 5.6: Imagens renderizadas do modelo 3D da ave de cerâmica: (a) utilizando `imagens do scanner` como textura; e (b) utilizando `imagens da câmera` como textura.

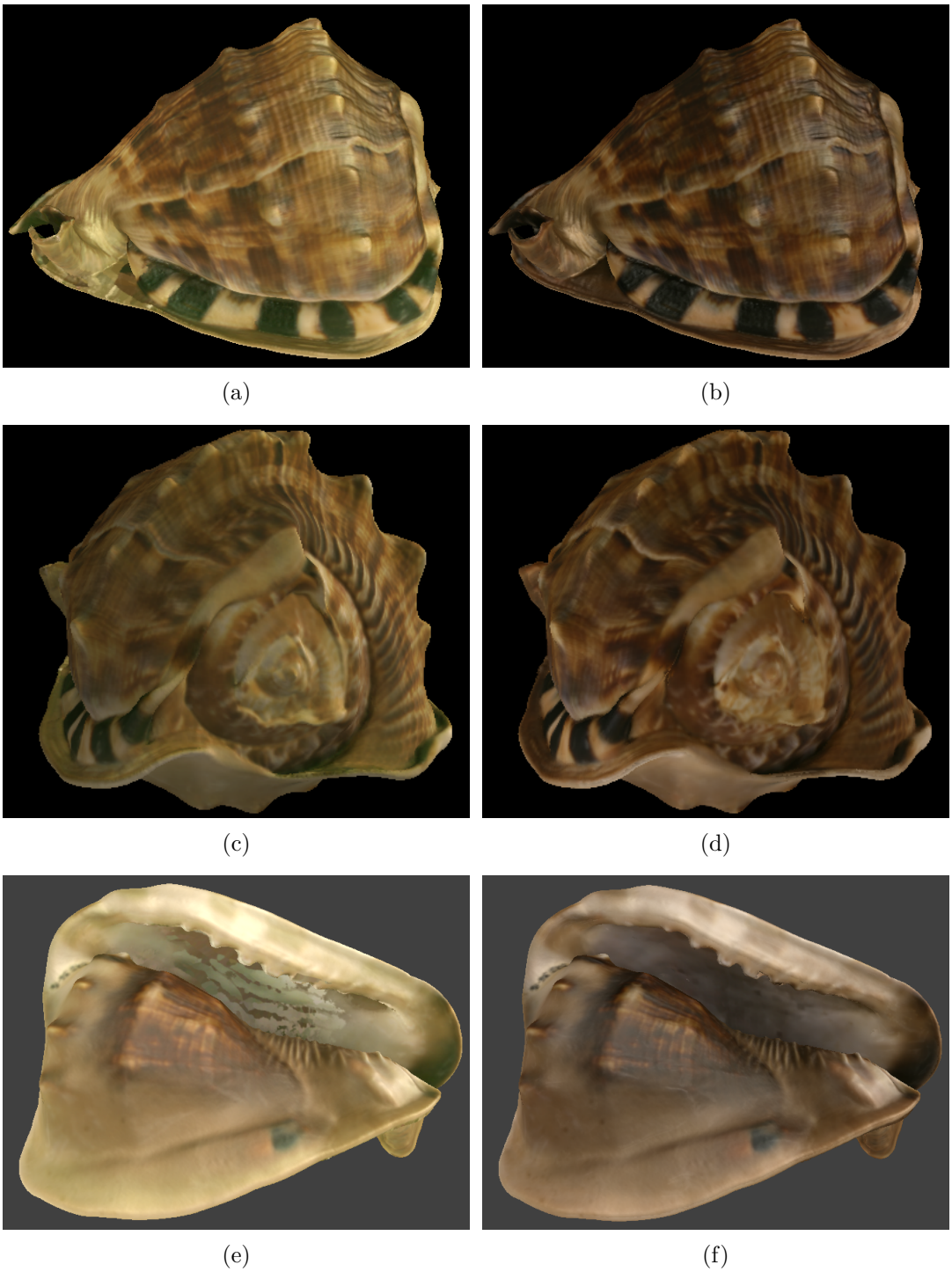
Foi notado que a geometria do objeto interfere na qualidade da textura gerada. Quando o objeto possui uma superfície com geometria homogênea, o alinhamento entre as vistas se torna menos preciso. Isso acontece porque nesta situação há poucas características a serem utilizadas como referência na etapa de alinhamento (Seção 3.1.2). Isso afeta as matrizes de transformação resultantes desta fase, que efetuam a transformação entre os sistemas de coordenadas das vistas e do modelo 3D final.

Essa precisão menor no alinhamento resulta em um efeito de borramento na textura, pois o algoritmo de geração de textura utiliza as matrizes de transformação das vistas para gerar os mapas de textura parciais. Uma possível solução para este problema é considerar informações de cor durante o alinhamento entre vistas. As Figuras 5.7(a) e 5.7(b) mostram um detalhe do objeto nas Figuras 5.6(a) e 5.6(b), respectivamente. O efeito do borramento pode ser observado durante a comparação da Figura 5.7(b) com a Figura 5.7(c), que mostra o detalhe em uma `imagem da câmera`.

Na Figura 5.8 são apresentadas renderizações de uma concha do gênero *Cypraecassis*, pertencente ao acervo do Museu de Ciências Naturais da UFPR. Nela, pode ser observado na Figura 5.8(e) que a textura do *scanner* está incorreta na área interna da concha. Isso acontece pois a região, por ser de difícil observação, apresenta buracos no modelo 3D. Isso pode ser observado na Figura 5.9, que apresenta uma renderização do modelo 3D da concha sem o preenchimento de buracos, utilizando cores referentes ao vetor normal de cada vértice como textura.

(a) (b) (c)

Figura 5.7: Influência da geometria do objeto na geração de textura: (a) detalhe da ave da Figura 5.6(a); (b) o mesmo detalhe na Figura 5.6(b); e (c) detalhe na textura obtida a partir de uma `imagem da câmera`.



(a) (b)

(c) (d)

(e) (f)

Figura 5.8: Imagens renderizadas do modelo 3D de uma concha do gênero *Cypraecassis*: (a) (c) e (e) usando `imagens do scanner` como textura; (b), (d) e (e) usando `imagens da câmera` como textura.

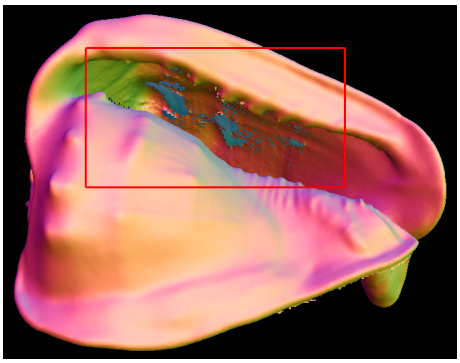


Figura 5.9: Renderização do modelo 3D da concha sem o preenchimento de buracos. No destaque, estão alguns dos buracos observados. A textura deste modelo foi calculada utilizando cores referentes ao vetor normal de cada vértice.

O método desenvolvido em Vrubel [33] preenche os buracos através do método de difusão volumétrica (ver Seção 2.4), e como são criados novos vértices para o modelo, ele difunde a informação de cor de vértices vizinhos aos novos vértices. Como nesta situação o tamanho dos buracos é grande em relação ao modelo, a difusão da cor nos vértices gerou regiões com cores errôneas. Por utilizar o modelo 3D final, já sem buracos, a textura da câmera não apresentou este problema, gerando resultados fidedignos (Figura 5.8(f)).

Entre os objetos preservados neste trabalho, estão artefatos indígenas pertencentes ao acervo do Museu de Arqueologia e Etnologia da UFPR. Na Figura 5.10 é apresentada uma comparação dos resultados obtidos com um pássaro feito com sambaqui. A Figura 5.11 mostra uma anta de cerâmica, manufaturada pela Comunidade Wauja, do Alto Xingu. Na Figura 5.12 são exibidas imagens de um pato de cerâmica pintada, feito pela Comunidade Karaja nos anos 50. Nestas figuras, pode ser observada a melhoria na fidelidade de cor e na qualidade das texturas dos modelos que utilizam `imagens da câmera`.

Os modelos 3D apresentados neste trabalho podem ser visualizados no Museu 3D Virtual do IMAGO, onde é possível um alto grau de interação com as réplicas digitais. Na Tabela 5.5 são exibidas informações sobre os objetos preservados e sobre os modelos 3D apresentados nesta seção. Os dados foram obtidos a partir de testes feitos em um computador com 1.5 GB de memória RAM, e um processador Intel Pentium 4 com 2.67 GHz.

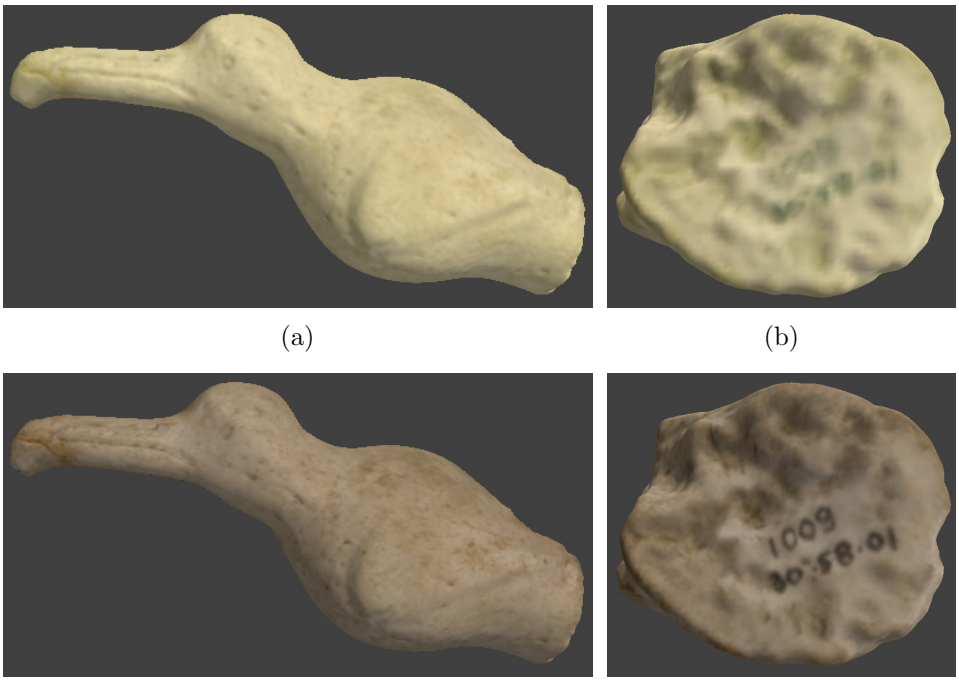


Figura 5.10: Imagens renderizadas do modelo 3D do pássaro: (a) e (b) usando `imagens do scanner` como textura; (c) e (d) usando `imagens da câmera` como textura.

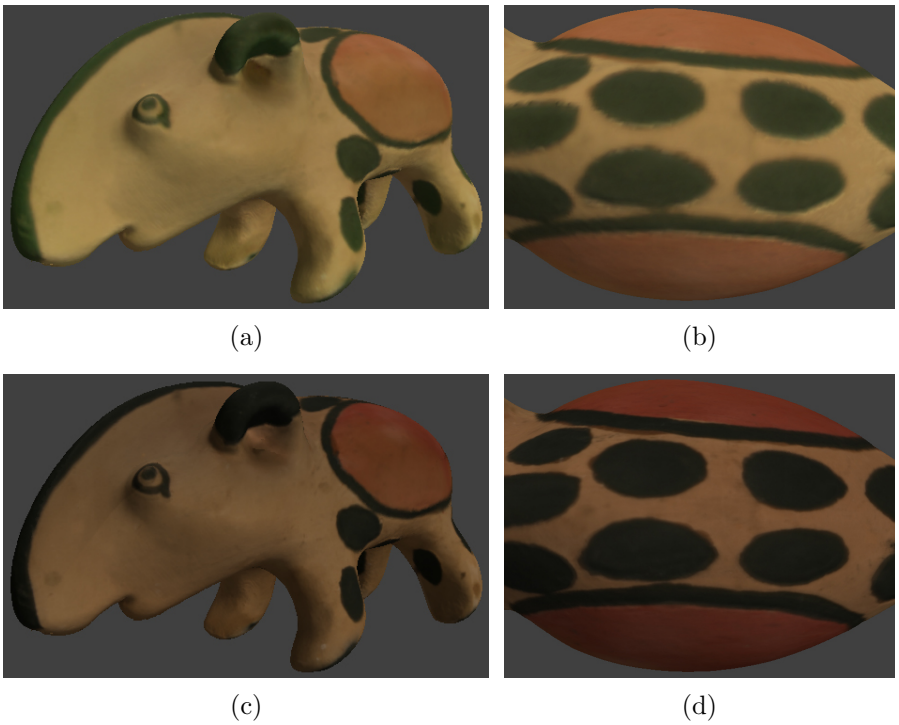


Figura 5.11: Imagens renderizadas do modelo 3D de uma anta de cerâmica: (a) e (b) usando `imagens do scanner` como textura; (c) e (d) usando `imagens da câmera` como textura.

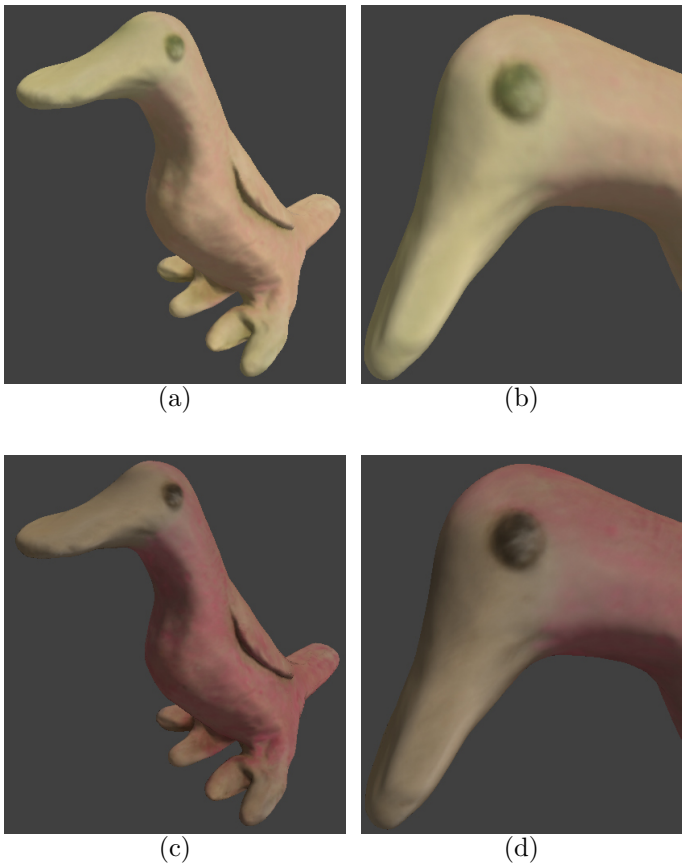


Figura 5.12: Imagens renderizadas do modelo 3D de um pato de cerâmica: (a) e (b) usando `imagens do scanner` como textura; (c) e (d) usando `imagens da câmera` como textura.

O valor da resolução apresentado na Tabela 5.5 é obtido a partir da divisão da altura em pixels do objeto nas imagens da câmera pela altura do objeto real, em milímetros. Este valor é utilizado como base no cálculo do tamanho da textura, para que esta mantenha a resolução das fotografias. No entanto, o OpenGL permite a renderização de texturas com tamanho máximo de 4096×4096 pixels. Com isso, algumas vezes é necessário limitar o valor da resolução de alguns objetos para que o tamanho da textura não ultrapasse o limite.

O tamanho da textura também é influenciado pelo método de planificação do modelo 3D utilizado (Subseção 4.2.1). Assim, objetos com dimensões e resoluções semelhantes podem apresentar texturas com tamanhos diferentes devido ao número de regiões geradas para o mapa de textura, e à alocação destas regiões no mapa. Como exemplo, tem-se o pato e o pássaro. Na Tabela 5.5, eles apresentam dimensões e resoluções semelhantes, porém possuem tamanhos de textura diferentes. Ao observar seus mapas de textura

Tabela 5.5: Informações sobre os objetos e seus modelos 3D.

	Estátua	**Ave**	**Pássaro**	**Pato**	**Anta**	***Cypraecassis***
Dimensões (cm)	$4 \times 3.5 \times 14.5$	$11 \times 22.5 \times 11$	$14 \times 5 \times 4$	$13 \times 6 \times 8$	$26 \times 12 \times 14$	$13 \times 20 \times 15.5$
Número de `triplas` utilizadas	28	31	47	52	33	53
Tempo de geração da textura	00:04:30	00:06:05	00:16:22	1:27:19	1:04:25	1:39:11
Tamanho da textura	2048×2048	2048×2048	2048×2048	4096×4096	4096×4096	4096×4096
Resolução da textura (ppm)	10	4.5	12	10	8	5
Número de vértices do modelo	74247	366523	81929	94079	223372	671459
Número de faces do modelo	146353	704922	154312	173700	418602	1270042

(Figura 5.13), pode-se notar que o pássaro foi divido em regiões maiores e melhor aproveitadas que o pato, demandando assim menor espaço na textura para manter a resolução desejada.

Uma observação a ser feita é que o tamanho da textura a ser calculada afeta o tempo de execução do algoritmo. Como pode ser observado na Tabela 5.5, os objetos com textura de tamanho 4096×4096 exigiram tempo de processamento muito maior que os objetos com tamanho 2048×2048. Isso acontece devido ao aumento na quantidade de informação que deve ser processada para cada um dos mapas de textura parciais e durante a integração destes para gerar o mapa de textura final.

(a) (b)

Figura 5.13: Mapas de textura calculados: (a) para o pato; e (b) para o pássaro. As texturas possuem tamanhos diferentes, apesar de serem obtidas a partir de objetos com resoluções e dimensões semelhantes.

CAPÍTULO 6

CONCLUSÕES

Este trabalho apresenta um estudo sobre a preservação digital de objetos com foco na textura. Nele, é feita uma revisão sobre a modelagem da geometria e da fotometria de objetos, e é desenvolvido um algoritmo para melhorar as texturas dos modelos 3D utilizando fotografias de alta resolução e qualidade. O uso destas fotografias aumenta substancialmente o realismo do modelo 3D, e consiste num passo essencial para a obtenção de modelos fotométricos com alta qualidade.

O algoritmo desenvolvido é utilizado na preservação digital de patrimônios, em conjunto com o método para geração de modelos 3D desenvolvido em [33]. Os modelos são exibidos no Museu Virtual 3D do Grupo Imago.

Este trabalho apresenta relevância cultural pois quanto mais detalhado for o modelo tridimensional, mais informações sobre o patrimônio preservado ele armazenará. As informações sobre a textura registram detalhes essenciais do objeto, possibilitando sua preservação para gerações futuras. Entre as aplicações para os modelos, pode-se destacar sua utilização em museus virtuais ou aplicações científicas; ou como um meio de simulação da restauração do patrimônio, em caso de desgaste natural ou acidentes.

No aspecto computacional, o algoritmo desenvolvido possui uma contribuição importante, pois poucos trabalhos detalham fases específicas do processo de modelagem fotométrica de objetos. As soluções apresentadas para os problemas encontrados na calibração e na geração de texturas também podem ser utilizadas em outros contextos de aplicação. Como exemplos de soluções, são ressaltados o cálculo do foco da câmera (Subseção 4.2.1.1), e a modificação da matriz de calibração (Subseção 4.1.2) com o objetivo de estimar a profundidade de pontos bidimensionais.

O algoritmo gera mapas de textura de alta qualidade sem o uso de um número muito grande de imagens de profundidades e coloridas, nem de aparatos especiais, se mostrando útil na validação e implementação por parte de outros pesquisadores. Ele também pode ser utilizado em conjunto com *pipelines* de geração de modelos fotométricos existentes, devido ao fato de ser facilmente adaptável a outros contextos.

O uso dos pesos de confiabilidade para integrar mapas de textura parciais apresenta bons resultados. Especulares e sombras que aparecem em um subconjunto de imagens de regiões do objeto são bastante reduzidas. Por sua vez, o uso dos mapas de textura parciais provê uma grande variedade de opções de integração. Podem ser consideradas a correção da iluminação e a detecção de especulares e sombras na integração, por exemplo.

Entre os trabalhos futuros, tem-se a inclusão de informações sobre a iluminação ambiente durante a integração de mapas de textura parciais, a fim de reduzir a influência da iluminação. Como neste trabalho o objetivo é desenvolver um *pipeline* independente de aparatos, onde nestes se incluem aparatos de iluminação, uma solução neste aspecto é a calibração da iluminação. Nela, um objeto que permita a obtenção de informações sobre a iluminação seria utilizado (como uma bola de bilhar, por exemplo), e as posições dos focos de luz seriam estimadas. A partir do modelo 3D e destas posições poderia então ser feita uma simulação da iluminação, e com isso descobrir regiões susceptíveis a sombras ou especulares.

Outro trabalho futuro é parametrizar o modelo fotométrico do objeto através de métodos como os apresentados na Seção 3.2, possibilitando uma renderização ainda mais realística.

BIBLIOGRAFIA

[1] Dana H. Ballard e Christopher M. Brown. *Computer Vision*. Prentice Hall, 1982.

[2] Olga R. P. Bellon. Visão computacional: um sistema para localização de objetos no espaço 3D, 1991. Tese de Mestrado.

[3] Fausto Bernardini e Holly Rushmeier. The 3D model acquisition pipeline. *Computer Graphics Forum*, 21(2):149–172, 2002.

[4] Paul J. Besl e Neil D. McKay. A method for registration of 3D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[5] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[6] Canon Inc. *EDSDK API Programming Reference*.

[7] Yang Chen e Gérard Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155, 1992.

[8] Brian Curless e Marc Levoy. A volumetric method for building complex models from range images. *Proc. International Conference on Computer Graphics and Interactive Techniques*, páginas 303–312, 1996.

[9] James Davis, Stephen R. Marschner, Matt Garr, e Marc Levoy. Filling holes in complex surfaces using volumetric diffusion. *Proc. 3D Data Processing, Visualization and Transmission*, páginas 428–861, 2002.

[10] Mohamed Farouk, Ibrahim El Rifai, Shady El-Tayar, Hisham El-Shishiny, Mohamed Hosny, Mohamed El Rayes, Jose Gomes, Frank Giordano, Holly Rushmeier, Fausto Bernardini, e Karen Magerlein. Scanning and processing 3d objects for web display. *Proc. International Conference on 3-D Digital Imaging and Modeling*, páginas 310–317, 2003.

[11] Olivier D. Faugeras e Giorgio Toscani. The calibration problem for stereo. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, páginas 15–20, 1986.

[12] Martin A. Fischler e Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[13] Kai Hormann, Bruno Levy, e Alla Sheffer. Mesh parameterization: Theory and practice. *International Conference on Computer Graphics and Interactive Techniques Course Notes*, 2007.

[14] Katsushi Ikeuchi, Takeshi Oishi, Jun Takamatsu, Ryusuke Sagawa, Atsushi Nakazawa, Ryo Kurazume, Ko Nishino, Mawo Kamakura, e Yasuhide Okamoto. The great buddha project: Digitally archiving, restoring, and analyzing cultural heritage objects. *International Journal of Computer Vision*, 75(1):189–208, 2007.

[15] Ramesh Jain, Rangachar Kasturi, e Brian G. Schunck. *Machine Vision*. 1995.

[16] Ryo Kurazume, Ko Nishino, Mark D. Wheeler, e Katsushi Ikeuchi. Mapping textures on 3d geometric model using reflectance image. *Systems and Computers in Japan*, 36(13):92–101, 2005.

[17] Ryo Kurazume, Ko Nishino, Zhengyou Zhang, e Katsushi Ikeuchi. Simultaneous 2d images and 3D geometric model registration for texture mapping utilizing reflectance attribute. *Proc. Asian Conference on Computer Vision*, volume 1, páginas 99–106, 2002.

[18] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, e Duane Fulk. The Digital Michelangelo Project: 3D Scanning of Large Statues. *Proc. International Conference on Computer Graphics and Interactive Techniques*, páginas 131–144, 2000.

[19] William E. Lorensen e Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Proc. Annual Conference on Computer Graphics and Interactive Techniques*, páginas 163–169, 1987.

[20] Caroline M. Mendes, Dyego R. Drees, Olga R. P. Bellon, e Luciano Silva. Sistema para visualização de museu virtual 3D. *V Workshop of Undergraduated Work, SIBGRAPI*, 2007.

[21] George Pavlidis, Anestis Koutsoudis, Fotis Arnaoutoglou, Vassilios Tsioukas, e Cristodoulos Chamzas. Methods for 3D digitization of cultural heritage. *Journal of Cultural Heritage*, 8(1):93–98, 2007.

[22] Michael Potmesil. Generating models for solid objects by matching 3D surface segments. *Proc. International Conference on Neural Networks*, volume 2, páginas 281–286, 1989.

[23] Kari Pulli. Multiview registration for large data sets. *Proc. International Conference on 3D Digital Imaging and Modeling*, páginas 160–168, 1999.

[24] Holly Rushmeier e Fausto Bernardini. Computing consistent normals and colors from photometric data. *Proc. International Conference on 3D Digital Imaging and Modeling*, páginas 99–108, 1999.

[25] Szymon Rusinkiewicz e Mark Levoy. Efficient variants of the ICP algorithm. *Proc. International Conference on 3D Digital Imaging and Modeling*, páginas 145–152, 2001.

[26] Filip Sadlo, Tim Weyrich, Ronald Peikert, e Markus Gross. A practical structured light acquisition system for point-based geometry and texture. *Proc. Eurographics Symposium on Point-Based Graphics*, páginas 89–98, 2005.

[27] Ryusuke Sagawa e Katsushi Ikeuchi. Taking consensus of signed distance field for complementing unobservable surface. *Proc. International Conference on 3-D Digital Imaging and Modeling*, páginas 410–417, 2003.

[28] Joaquim Salvi, Xavier Armangué, e Joan Batlle. A comparative review of camera calibrating methods with accuracy evaluation. *Pattern Recognition Letters*, 35(7):1617–1635, 2002.

[29] Yoichi Sato e Katsushi Ikeuchi. Temporal-color space analysis of reflection. *Journal of Optical Society of America*, 11(11):2990–3002, 1994.

[30] Yoichi Sato, Mark Wheeler, e Katsushi Ikeuchi. Object shape and reflectance modeling from observation. *Proc. International Conference on Computer Graphics and Interactive Techniques*, páginas 379–387, 1997.

[31] Phil Torr e Andrew Zisserman. Robust computation and parametrization of multiple view relations. *Proc. International Conference on Computer Vision*, páginas 727–732, 1998.

[32] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *International Journal of Robotics and Automation*, RA-3:323–344, 1987.

[33] Alexandre Vrubel. Reconstrução digital de objetos com scanners 3D e triangulação laser, 2008. Dissertação de Mestrado.

[34] Mark Wheeler, Yoichi Sato, e Katsushi Ikeuchi. Consensus surfaces for modeling 3D objects from multiple range images. *Proc. International Conference on Computer Vision*, páginas 917–924, 1998.