Fotografia 3D

Paulo Cezar Carvalho
Luiz Velho
Asla Sá
Esdras Medeiros
Anselmo Antunes Montengro
Adelailson Peixoto
Luis Antonio Rivera Escriba

editores: Luiz Velho e Paulo Cezar Carvalho

Prefácio

Este livro teve origem numa série de atividades realizadas no Laboratório VISGRAF do IMPA na área de modelagem baseada em imagens desde 2001. Essas atividades incluem diversos cursos, palestras, projetos de pesquisa, teses de mestrado e doutorado. Enfim, elas constituem uma linha completa de investigação que se tornou um dos principais focos de interesse dos organizadores do curso, Luiz Velho e Paulo Cezar Carvalho. Por esse motivo, resolvemos convidar para escrever as notas do curso nossos colaboradores nessa pesquisa: Asla Sá, Esdras Soares, Anselmo Antunes Montenegro, Luiz Rivera e Adelailson Peixoto. Todos eles abordaram assuntos relacionados com o tema em suas teses de mestrado e doutorado.

O livro apresenta uma área de pesquisa recente em Computação Gráfica, conhecida como *Fotografia 3D*. Essa área integra diversos métodos para a reconstrução de objetos tridimensionais a partir de imagens. A estrutura do livro contém uma visão geral do processo de fotografia 3D, incluindo a discussão do problema de calibração de câmera, aquisição de informações geométricas e fotométricas, além de processamento dos modelos.

Os pré-requisitos do livro são poucos: cálculo de uma e várias variáveis, álgebra linear, noções básicas de algoritmos e estruturas de dados.

Agradecemos à Comissão Organizadora do 25º Colóquio Brasileiro de Matemática a oportunidade de ministrar esse curso.

Rio de Janeiro, 3 de abril de 2005.

Luiz Velho e Paulo Cezar Carvalho

Sumário

Lista de Figuras

Capítulo 1

Fotografia 3D: Uma Visão Geral

Avanços recentes, tanto na indústria de equipamentos gráficos, quanto no estado da arte em modelagem geométrica, tornaram possível a digitalizaçãoo tridimensional de objetos reais com alto grau de fidelidade. Essa forma de aquisição de dados gráficos permite recriar no computador o modelo digital de um objeto 3D real. O conjunto de técnicas utilizadas para esse fim vem sendo chamada de *fotografia 3D*.

A fotografia 3D é uma das áreas de pesquisa recentes que tem se mostrado ser mais promissoras em computação gráfica. Isso se deve a varios motivos. Em primeiro lugar, a área unifica técnicas de visão computacional, processamento de imagem, modelagem geometrica e visualização. Nesse sentido, ela pode ser considerada como uma sub-area da modelagem e visualização baseada em imagens. Em segundo lugar, o rápido desenvolvimento de equipamentos digitais de fotografia e video possibilita a construção de sistemas de aquisição 3D muito efetivos e de baixo custo. Em terceiro lugar, a reconstrução de objetos tridimensionals a partir de imagens tem inúmeras aplicações em diversos campos, tais como Arqueologia, Patrimônio Histórico e

Cultural, Arte, Educação, Comércio Eletrônico e Desenho Industrial.

Esse livro tem como objetivo dar uma visão abrangente da fotografia 3D. Ele é direcionado para alunos e pesquisadores de Matemática Aplicada que tem interesse na area, tanto para conhecer o seu potencial, quanto para aprofundar o conhecimento de suas técnicas.

A estrutura do livro cobre as tecnicas básicas da área, incluindo noções de geometria projetiva e calibração de camera, triangulação e estéreoscopia, metodos locais e globais de reconstrução de superficies, bem como a integração de dados e geração de malhas. Esses topicos organizados da seguinte forma: o Capitulo 1 dá uma visão geral da área, categorizando seus problemas e métodos; o Capitulo 2 cobre os conceitos fundamentais de visão computacional necessários para a fotografia 3D; os Capítulos 3 e 4 discutem as duas abordagens principais para reconstrução de objetos a partir de imagens; o Capitulo 5 apresenta as técnicas de construção dos modelos. Finalmente, o Apêndice descreve um sistema de fotografia 3D desenvolvido no Laboratório VISGRAF do IMPA.

1.1 Modelagem Tridimensional

A modelagem digital de objetos tridimensionais é um dos problemas básicos da Computação Gráfica, com implicações em práticamente todas as áreas de atuação dessa disciplina.

O problema engloba os vários aspectos relacionados com a descrição e manipulação de objetos 3D no computador. Assim, sua solução deve ser equacionada em níveis de abstração distintos, que incluem: a escolha de um modelo matemático apropriado para o estudo dos objetos de interesse; o estabelecimento de um esquema de representação geométrica baseado nesse modelo; e a formulação de estruturas de dados e rotinas para implementar essa representação no computador.

Além disso, um sistema de modelagem deve suportar uma gama de recursos computacionais para a manipulação de objetos 3D. Dentre estes, os mais importantes dizem respeito à criação, modificação, visualização, simulação e armazenamento de dados.

O processo de modelagem de objetos 3D involve duas fases principais:

1. *Criação de modelos geometricos*: que pode ser feita à partir de um objeto real usando um método de aquisição, ou à partir de especificações geométicas usando técnicas de modelagem.

2. *Processamento de modelos geométricos*: que inclui a visualização e análise dos objetos representados pelo modelo.

Um sistema de modelagem geométrica é constituido por componentes de hardware e software específicos para essa aplicação. Esses dois componentes estão fortemente interrelacionados e, em geral, dependem do tipo do sistema de modelagem.

Num ambiente de fotografia 3D é de fundamental importancia dispor de uma plataforma genérica e flexível para se obter bons resultados. Essa plataforma deve facilitar a investigação de novos problemas e, ao mesmo tempo, permitir validar as soluções de maneira consistente com as aplicações reais.

1.2 Tecnologia de Digitalização 3D

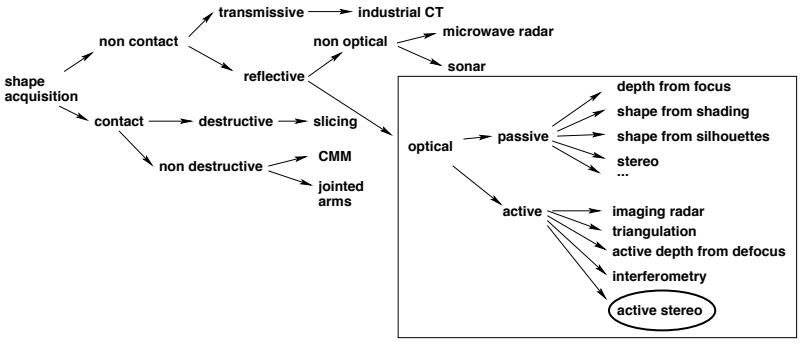
Um sistema de modelagem para fotografia 3D emprega equipamentos para digitalização de objetos tridimensionais juntamente com técnicas para o processamento desses dados.

Para a captura dos dados são utilizados equipamentos que realizam a amostragem da geometria e da cor em um conjunto de pontos na superfície do objeto digitalizado.

As técnicas de captura de dados geométricos para fotografia 3D dividem-se em dois grandes grupos: 1) sensoriamento direto e 2) baseado em imagens.

Uma classificação das técnicas de aquisição de geometria de objetos é sumarizada na Figura 1.1 a região da figura marcada com o retângulo, corresponde aos métodos de aquisição baseados em imagens, isto é, o sensor que servirá para 'medir' os objetos da cena são sensores óticos (câmeras).

As técnicas de sensoriamento direto fazem medições diretas da forma do objeto 3D. Várias tecnologias podem ser empregadas para



(a)

Figura 1.1: Tecnologias de digitalização 3D

esse fim. Por exemplo, equipamentos como o time-of-flight laser operam de forma semalhate a um radar para medir distancias.

As técnicas de captura baseada em imagens se subdividem em: estéreo passivo e estéreo ativo. As técnicas de estéreo passivo empregam duas ou mais câmeras calibradas e calculam correspondências entre as projeções de pontos em imagens distintas para medir a profundidade dos pontos na cena, de forma semelhante ao olho humano. As técnicas de estéreo ativo são baseadas nos mesmos princípios da estéreoscopia passiva, mas utilizam um par câmera / projetor, que permite iluminar a cena com um padrão de luz estruturada. O sistema ótico do projetor tem um papel semelhante ao da segunda câmera nas medições estereoscópicas e o padrão projetado facilita correspondências entre pontos.

Dentre todas as técnicas ilustradas, a escolha de uma delas envolve uma análise das qualidades e limitações de cada técnica, tais como resolução e precisão do dado adquirido. É de fundamental importância conhecer a aplicação para decidir o *software* e o *hardware* mais adequados. Por exemplo, adquirir os dados de um prédio ou construção é uma tarefa muito diferente da tarefa de adquirir os dados de um vaso com propriedades fotométricas variáveis.

1.3 Etapas do Processo

O processo de fotografia 3D envolve a construção da representação de um modelo do objeto tridimensional a partir dos dados digitalizados.

Em geral, os dados digitalizados são coletados por partes. Além disso, dois conjuntos de dados de natureza diferente são obtidos: um mapa de distancias e um mapa de atributos.

- *Mapa de Distâncias*: define um retalho da superfície através de uma parametrização local. Os pontos do objeto, usualmente, são estruturados segundo malhas retangulares, que induz uma parametrização linear por partes para cada retalho da superfície. Outra opção é uma parametrização cilindrica.
- *Mapa de Atributos*: especifica o material da superfície por meio de uma função de textura, que pode incluir propriedades fotométricas tais como: cor, reflectância, etc.

Esses mapas podem ser digitalizados separadamente ou de forma integrada, dependendo do equipamento de captura.

Para a construção do modelo os elementos que compõem o objeto são colados para formar uma superfície completa. Assim é necessário fazer o alinhamento dos retalhos; a sua integração e finalmente a conversão para um modelo apropriado.

As etapas do processo de fotografia 3D são:

1. **Captura:** Na etapa de captura de dados utiliza-se sensores para coletar informações geométricas e fotométricas do objeto físico em retalhos.
2. **Alinhamento:** Os retalhos capturados são definidos em seus próprios sistemas de coordenadas locais. O registro dos retalhos corresponde ao alinhamento de cada retalho com os seus vizinhos em relação a um sistema de coordenadas global.
3. **Integração:** O alinhamento dos retalhos resulta em um conjunto de amostras da superfície definidas num sistema de coordenadas comum. A integração transforma essas amostras esparsas numa superfície contínua. Isso possibilita reconstruir a geometria e a topologia do objeto. Normalmente, emprega-se uma estrutura de malha simplicial para este fim.

4. **Conversão:** O modelo digital da superfície é produzido convertendo-se a malha simplicial numa descrição adaptada. A conversão elimina as redundancias existentes nos dados e identifica as características marcantes da forma do objeto. Uma alternativa é empregar um modelo em multi-escala para descrever o objeto.

Essas etapas serão discutidas em detalhe nos Capítulos 3 e 4, que apresentam duas abordagens distintas para a implementação do processo de fotografia 3D.

Além dessas etapas, é necessario que o equipamento de captura esteja calibrado e devidamente referenciado em relação ao sistema de coordenadas global da cena 3D. Para isso, temos que realizar uma etapa preliminar de **Calibração** do equipamento de captura. Essa etapa, incluí a calibração intrínseca e extrínseca de cameras e outros dispositivos utilizados. A calibração intrínseca determina os parametros dos dispositivos de captura e a calibração extrínseca determina a relação entre os sistemas de coordenadas locais dos dispositivos e o sistema de coordenadas global da cena 3D. O Capitulo 2 estuda os princípios basicos de geometria projetiva e suas aplicações para a calibração dos dispositivos de captura.

O resultado final do processo de fotografia 3D é um modelo geométrico do objeto que define a geometria, topologia, e outras propriedades do objeto 3D real. Esse modelo permite extrair diversas informações para operações com o objeto, tais como: medições, simulação, visualização e outras. A geração de malhas para o modelo geométrico será abordada no Capítulo 5.

1.4 Abordagens para Reconstrução 3D

Existem duas abordagens fundamentais para a implementação do processo de fotografia 3D:

- Reconstrução no espaço da imagem;
- Reconstrução no espaço da cena 3D;

A primeira é baseada na reconstrução do objeto no espaço local do dispositivo de captura e a segunda realiza a reconstrução do objeto diretamente no espaço global da cena.

A principal diferença entre essas duas abordagens está na maneira como os dados capturados são integrados para construir a representação do objeto 3D, e em que sistemas de coordenadas as operações são feitas.

Usualmente, os dados capturados estão referenciados no sistema de coordenadas local do dispositivo de captura (sem perda de generalidade, podemos supor que o dispositivo de captura funciona de forma semelhante a uma camera – no sentido que os dados são coletados ao longo de um raio de visada a partir de um ponto de vista). Por outro lado, podemos referenciar os objetos capturados em um sistema de coordenadas global associado à cena 3D. Além disso, a representação final do objeto 3D é definida em um sistema de coordenadas próprio, que pode ser ou não o mesmo da cena.

Na reconstrução baseada no espaço da imagem, a maior parte dos cálculos é feita no sistema de coordenadas local do dispositivo de captura. Dessa forma, os dados geométricos e fotometricos são estruturados em partes e a integração global dessas partes é postergada para a última etapa do processo de reconstrução.

Na reconstrução baseada no espaço da cena, a maioria dos cálculos é feita no sistema de coordenadas global associado ao objeto. Assim, os dados geométricos e fotométricos, logo após serem capturados, são referenciados no sistema de coordenadas global para serem imediatamente integrados na representação do objeto.

Um aspecto importante do processo de reconstrução está ligado ao tipo de representação geometrica utilizada no processo. Métodos locais empregam descrições paramétricas que exploram a estrutura dos dispositivos de captura. Métodos globais empregam descrições implicitas basedas em uma subdivisão do espaço da cena. Muitas vezes, se faz necessária a conversão entre esses dois tipos de representação geométrica. Em geral, tais conversões implicam numa mudança de sistemas de coordenadas – de um referencial local para um referencial global.

Os Capítulos 3 e 4 discutem as técnicas adotadas nessas duas abordagens, bem como a arquitetura dos sistemas de fotografia 3D que as utilizam. O Capítulo 3 apresenta a reconstrução no espaço da imagem, com uma ênfase em sistemas de aquisição por luz estruturada. O Capítulo 4 apresenta a reconstução no espaço da cena, com ênfase nos sistemas de aquisição por consistencia fotometrica.

É digno de nota que os métodos de visualização também podem ser classificados segundo esse mesmo critério dos sistemas de coordenadas [30]. A visualização por traçado de raios, por exemplo, opera no espaço da cena, enquanto que o algoritmo de Z-buffer trabalha no espaço da imagem. De fato, existe uma forte relação entre a solução do problema de visualização e de reconstrução de cenas 3D. Isso é mais uma evidência da dualidade entre a Visão Computacional e a Computação Gráfica.

1.5 Exemplo de Aplicação: Digitalização de Esculturas

Uma aplicação especialmente interessante da fotografia 3D é a digitalização de esculturas e outros artefatos históricos-culturais. Esses objetos apresentam grandes desafios técnicos e um alto potencial de utilização. Vale ressaltar também que esculturas digitais têm um valor intrínseco tanto do ponto de vista cultural quanto comercial.

A digitalização de esculturas tem sido motivo de intensas pesquisas. Os dois projetos pioneiros nessa área foram: o Digital Michelangelo e o Projeto Pietà. O projeto Digital Michelangelo desenvolvido pela Stanford University digitalizou várias estátuas desse artista, dentre as quais a famosa escultura do Davi. O projeto Pietà conduzido pelo IBM T.J. Watson research center construiu um modelo da Pietà florentina de Michelangelo para estudos pelo historiador Jack Wasserman. Esses projetos foram seguidos por outras inciativas. O grupo do ISTI-CNR em Pisa vem realizando diversos trabalhos de digitalização de esculturas, dentre os quais o projeto de restauração da estátua de Minerva de Arezzo.

Mais recentemente, vários grupos de pesquisa começaram a se associar para o desenvolvimento de projetos conjuntos. Em 2001 foi formado o consórcio VIHAP-3D (Virtual Heritage: High-Quality Acquisition and Presentation), liderado pelo Max-Planck Institute, com o ISTI-CNR, o centro Gedas de realidade Virtual, a Universidade da Caltalonia e diversos museus na Europa.

Outro projeto interessante é o Digital Parthenon, realizado por Paul Debevec do UCS-ICT. Esse projeto teve apoio do consórcio VIHAP-3D e construiu um modelo digital do Parthenon com suas

esculturas. Tais dados foram usados para produzir um filme sobre o monumento histórico.

O grupo do IBM T.J. Watson research center foi comissionado pelo Ministério da Cultura do Egito para montar um museu virtual, chamado de "Ethernal Egypt", para o qual foram digitalizadas as coleções de diversos museus.

Capítulo 2

Geometria Projetiva e Calibração

2.1 Introdução

Os métodos de aquisição de forma tridimensional estudados neste livro utilizam imagens (capturadas por uma ou mais câmeras) para reconstruir pontos no espaço tridimensional. O princípio básico é o mesmo que permite a um sistema de visão binocular (como é o caso de nosso sistema visual) avaliar a distância a que está situado um determinado ponto. Quando uma única imagem está disponível, a posição do ponto na imagem não determina sua posição no espaço: o ponto, no espaço, pode estar em qualquer lugar da reta que une o ponto de vista ao ponto projetado, como ilustrado na Figura 2.1. Em um sistema binocular, no entanto, são capturadas duas imagens em que o ponto de interesse aparece. Neste caso, são conhecidas duas retas que contêm o ponto correspondente do espaço, que pode, então, ser determinado em um processo chamado de triangulação, que vai ser discutido em mais detalhes no Capítulo 3.

A reprodução deste processo em um ambiente computacional tem um passo crucial, que é a determinação dos pontos correspondentes nas duas imagens. Embora este problema tenha sido bastante estudado, as soluções produzidas, mesmo pelos melhores algoritmos, são

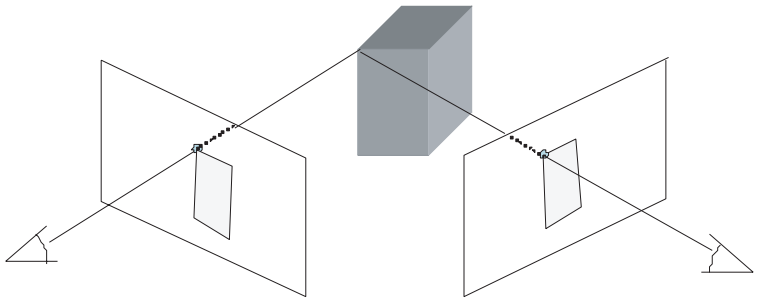


Figura 2.1: Visão binocular.

sujeitas a imprecisões, especialmente nos casos em que os objetos sob observação apresentam pouca variação de cor ou textura. Para evitar este problema, os sistemas práticos de aquisição de forma recorrem ao chamado *estéreo ativo*, em que o objeto de interesse é iluminado por um padrão conhecido, facilitando a resolução do problema de correspondência. Este padrão pode ser produzido, por exemplo, através de um projetor de vídeo. Na verdade, ao invés de se utilizar o padrão projetado como um simples meio de facilitar a correspondência entre imagens obtidas por duas câmeras, normalmente utiliza-se o projetor como se fosse uma segunda câmera; neste caso, os dois raios a serem intersectados são o raio visual correspondente ao ponto na imagem e o raio projetado pelo projetor.

Em qualquer caso, é necessário determinar a relação existente entre as coordenadas de um ponto no espaço e as coordenadas deste ponto na imagem adquirida por uma câmera ou projetada por um projetor. O problema de determinar esta relação é o problema de calibração, tratado nas seções a seguir. Inicialmente, discutiremos o problema de calibrar a câmera (ou câmeras) envolvidas. Depois, veremos as (pequenas) adaptações necessárias para projetores.

2.2 Modelos de Câmera

O modelo mais simples de câmera é o chamado modelo *pinhole*. Tal tipo de câmera pode ser realizado por uma caixa fechada, em que se faça um pequeno orifício (Figura 2.2). Os raios luminosos passam por este orifício (o centro óptico) e atingem o plano do fundo da caixa, onde uma imagem se forma. Se este plano for revestido por um material sensível à luz (filme fotográfico ou sensores digitais, por exemplo), é possível registrar a imagem formada.

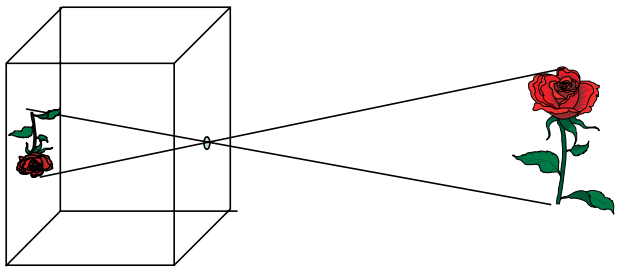


Figura 2.2: Câmera *pinhole*.

A imagem formada em uma câmera *pinhole* é determinada pela interseção, com o plano do fundo da câmera, dos raios luminosos emitidos ou refletidos pelos objetos da cena e que passam pelo orifício. Como esta imagem é invertida, é mais conveniente considerar a imagem formada pela interseção destes mesmos raios com o plano situado à mesma distância do orifício, mas colocado à sua frente. Isto equivale a considerar o centro óptico O como sendo o olho de um observador, que observa o mundo através de uma janela (Figura 2.3).

Em ambos os casos, o funcionamento geométrico da câmera é definido através de uma correspondência que associa cada ponto P do espaço tridimensional ao ponto correspondente p no plano de formação imagem (isto é, ao ponto obtido pela interseção da reta OP com aquele plano). Esta correspondência é chamada de *projeção perspectiva*, cujo estudo, iniciado na época do Renascimento, levou ao desenvolvimento de uma teoria matemática chamada *Geometria Projetiva*, cujos fundamentos abordamos na próxima seção.

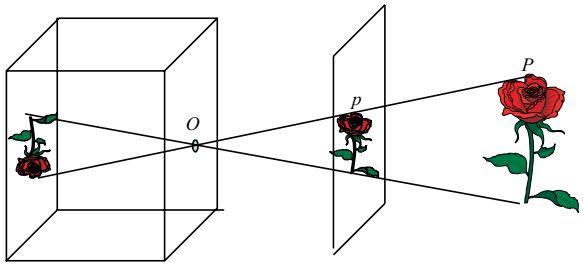


Figura 2.3: Câmera *pinhole* e projeção perspectiva.

O modelo *pinhole* fornece uma boa aproximação em muitas situações, mas não corresponde exatamente ao que ocorre em câmeras fotográficas reais. O modelo pinhole pressupõe que um único raio luminoso proveniente de cada ponto da cena atinja o plano sensível. Assim, parece desejável construir uma tal câmera fazendo um orifício de dimensões extremamente reduzidas. Infelizmente, isto não é possível, por duas razões. Primeiro, a luz a atingir o plano sensível teria intensidade extremamente baixa, já que poucos raios luminosos (ou seja, poucos fótons) provenientes de cada ponto atravessam o furo. Segundo, para dimensões do furo da ordem do comprimento de onda da luz, ocorre o fenômeno de difração, que interfere na direção de propagação, causando franjas na imagem. Aumentar o tamanho do furo também não é uma alternativa, pois neste caso os raios luminosos provenientes de um ponto deixam de atingir o plano sensível em um único ponto, causando borramento na imagem.

A solução consiste em usar um sistema de lentes, que concentra os raios provenientes de um ponto da cena, permitindo usar uma abertura maior, que faz com que a intensidade luminosa na superfície sensível seja maior. Por outro lado, o uso de lentes causa algumas dificuldades. Há exatamente um plano da cena que está exatamente em foco; em torno deste plano, há uma região que fica razoavelmente focada, cuja largura determina a chamada profundidade de foco, que diminui à medida que se aumenta a abertura da lente. Além disso, os raios luminosos, ao incidir na lente, sofrem desvios em sua trajetória, que leva a distorções na imagem. O tipo de distorção mais importante

para a nossa aplicação é a chamada *distorção radial*, que provoca a curvatura de retas situadas na borda da imagem, como ilustrado na Figura 2.4. Veremos como tratar com tais distorções na Seção 2.4.6.

Figura 2.4: Distorção radial: linhas retas aparecem curvas na imagem.

2.3 Noções de Geometria Projetiva

2.3.1 Projeção Perspectiva

Consideremos a função que associa a cada ponto do espaço a sua projeção perspectiva, definida por um centro óptico O e um plano de projeção π, situado a uma distância f de O. Para expressar esta função, é conveniente escolher um sistema de coordenadas para o espaço cuja origem esteja em O e que tenha um dos eixos perpendicular a π, conforme mostra a Figura 2.5.

Neste referencial, a reta que passa pelo centro óptico e por $P = (X, Y, Z)$ é formada por todos os pontos da forma $\alpha(X, Y, Z)$. Destes, o que está sobre o plano π é o que tem a última coordenada igual a f. Assim, $\alpha = f/Z$ e as coordenadas do ponto na imagem são dadas por:

$$x = fX/Z, \quad \text{e} \quad y = fY/Z \tag{2.1}$$

É interessante observar alguns fatos sobre esta transformação. Para simplificar a discussão, vamos considerar que $f = 1$. Inicialmente, observamos que ela não está definida em todo o espaço: o raio de projeção correspondente a pontos tais que $Z = 0$ (ou seja,

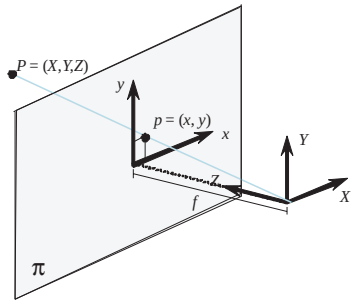


Figura 2.5: Determinação da projeção perspectiva.

que estejam no plano paralelo a π passando por O) não cortam π. Por outro lado, ela preserva retas: de fato, os raios de projeção correspondentes a três pontos colineares estão contidos em um único plano, que intersecta o plano de projeção segundo uma reta.

Vejamos o que acontece com a projeção de um feixe de retas paralelas do espaço. Tomemos, por exemplo, as retas paralelas ao eixo Z. Cada uma delas é da forma $\{(X_0, Y_0, Z) | Z \in R\}$. A projeção de um ponto genérico de uma destas retas é dada por $\left(\frac{X_0}{Z}, \frac{Y_0}{Z}\right)$, onde $Z \neq 0$, já que a projeção de $(X_0, Y_0, 0)$ não está definida. Para cada par (X_0, Y_0), o conjunto de todos estes pontos formam uma reta que passa por $(0, 0)$, com este ponto excluído. Assim, a imagem de todas as retas paralelas paralelas ao eixo Z, determinam, na projeção perspectiva, um feixe de retas concorrentes em $(0, 0)$. Ou seja, retas que são paralelas no espaço são vistas como concorrentes na projeção perspectiva. Intuitivamente, a origem $(0, 0)$ pode ser vista como a imagem de um "ponto no infinito", comum a todas estas retas.

2.3.2 Espaço Projetivo

A Geometria Projetiva tem por objetivo tornar precisas as idéias intuitivas acima. A noção fundamental da Geometria Projetiva é a de *espaço projetivo*. O espaço projetivo n-dimensional é construído de modo a fornecer um modelo matemático que amplia o espaço Euclidiano n-dimensional, incluindo novos pontos (os *pontos no infinito*

ou *ideais*), que correspondem, como veremos, as direções do plano Euclidiano. Isto pode ser feito identificando os elementos do espaço projetivo n-dimensional, denotado por RP^n, com as retas de R^{n+1} que passam pela origem.

Para fixar as idéias, consideremos o caso do plano projetivo (ou seja, o caso $n = 2$). Uma reta passando pela origem fica definida por qualquer um de seus pontos não nulos. Assim, a reta associada ao ponto (u, v, w) está igualmente associada a todos os pontos da forma $(\lambda u, \lambda v, \lambda w)$, onde $\lambda \neq 0$. Cada uma destas triplas fornecem uma representação (nas chamadas *coordenadas homogêneas* ou *projetivas*) para o mesmo ponto do plano projetivo, que será denotado por $[u\ v\ w]$. Naturalmente, $[\lambda u\ \lambda v\ \lambda w]$, para qualquer $\lambda \neq 0$, representa o mesmo ponto projetivo (escrevemos $[u\ v\ w] \simeq [\lambda u\ \lambda v\ \lambda w]$). Este ponto projetivo pode ser interpretado, em relação ao plano Euclidiano, por meio da identificação deste último com o plano $w = 1$. Quando $w \neq 0$, existe uma única representação em coordenadas homogêneas com $w = 1$, dada por $[u/w \quad v/w \quad 1]$, que é o ponto de interseção da reta associada ao ponto projetivo com o plano $w = 1$ (Figura 2.6). Assim, o ponto do espaço projetivo de coordenadas $[u\ v\ w]$, com $w \neq 0$, é identificado com o ponto $(u/w, v/w)$ do plano Euclidiano. Ou seja, há uma correspondência biunívoca entre o subconjunto dos pontos projetivos com última coordenada não nula e o plano Euclidiano.

Quando $w = 0$, a reta $(\alpha u, \alpha v, \alpha w)$, é paralela ao plano $w = 1$. Neste caso, associamos o ponto projetivo $[u\ v\ 0]$ ao vetor (u, v) do plano Euclidiano, ou seja, a uma *direção* do plano.

De modo geral, o espaço projetivo n-dimensional estende o espaço Euclidiano de mesma dimensão. Cada ponto $[x_1\ x_2 \ldots x_n\ x_{n+1}]$ do espaço projetivo corresponde a um ponto (dado por $(x_1/x_{n+1},\ x_2/x_{n+1}, \ldots,\ x_n/x_{n+1})$, no caso em que $x_{n+1} \neq 0$) ou a uma direção (dada por $(x_1, x_2, \ldots, x_n)$, no caso em que $x_{n+1} = 0$) do espaço Euclidiano. O mais importante, porém, é que esta construção permite operar, indiferentemente, com pontos ou direções, o que é extremamente conveniente para o estudo das projeções perspectivas, como veremos a seguir.

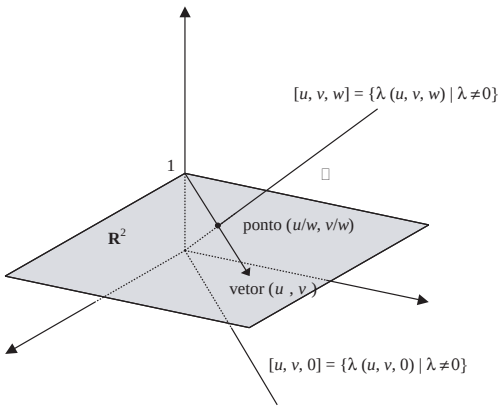


Figura 2.6: O plano projetivo.

2.3.3 Colineações e Transformações Projetivas

Dados dois espaços projetivos, de dimensões m e n, as transformações lineares entre os espaços R^{m+1} e R^{n+1} definem transformações entre os espaços projetivos, chamadas de *colineações*. As restrições destas colineações aos espaços euclidianos R^m e R^n são chamadas de *transformações projetivas*.

O termo colineação é decorrente da seguinte propriedade fundamental:

Teorema 1. *Uma transformação projetiva preserva retas.*

No caso de uma transformação projetiva do plano, podemos usar um argumento geométrico, baseado na idenficação do plano Euclidiano com o plano $w = 1$ (Figura 2.7). Uma transformação linear do R^3 leva a reta que contém os pontos A, B e C pertencentes ao plano $w = 1$ em uma outra reta de R^3. As imagens A', B' e C' por esta transformação determinam um plano passando pela origem. Logo,

as imagens A'', B'' e C'' pela transformação projetiva também estão alinhadas. Para uma demonstração mais geral, veja [30].

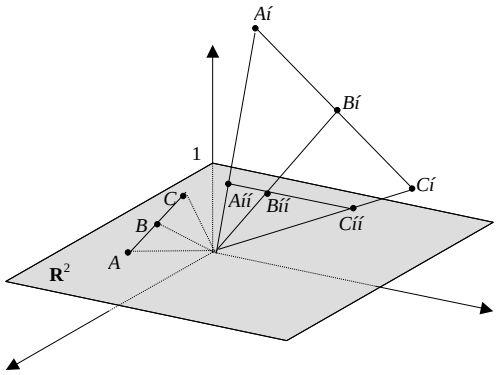


Figura 2.7: Transformações projetivas preservam retas.

A construção acima sugere que, embora transformações projetivas preservem colinearidade, elas não preservam razões entre os segmentos determinados pelos pontos. Na Figura 2.7, o ponto B é médio de AC; esta relação é preservada na transformação linear, mas não na projeção sobre o plano $w = 1$. No entanto, transformações projetivas preservam as chamadas *razões cruzadas*, como estabelecido no teorema a seguir, cuja demonstração pode ser encontrada em [30].

Teorema 2. *Uma transformação projetiva preserva a razão cruzada* $\frac{CA/CB}{DA/DB}$ *para quaisquer pontos colineares A, B, C e D.*

A projeção perspectiva vista no início desta seção é um exemplo de colineação de RP^3 em RP^2. Em coordenadas homogêneas, a transformação perspectiva dada pela equação (2.1) pode ser expressa como:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \simeq \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$$

De fato, exprimindo-se um ponto (X, Y, Z) do espaço em coordenadas homogêneas como $[X\ Y\ Z\ 1]$ e efetuando a multiplicação indicada obtém-se $[fX\ fY\ Z]$, que é uma das representações, em coordenadas homogêneas, do ponto $(fX/Z,\ fY/Z)$ do plano da imagem. Note que as coordenadas cartesianas da projeção perspectiva são obtidas divindo as duas primeiras coordenadas homogêneas pela terceira.

Com esta nova formulação, a imagem pela projeção perspectiva passa a ser definida para todo ponto do espaço (com exceção da origem), podendo ocorrer os seguintes casos:

1. ponto próprio projetado sobre ponto próprio.

   Por exemplo, a imagem de $P = [0\ 0\ 1\ 1]$ é $p = [0\ 0\ 1]$; ou seja, a origem do espaço é levada na origem da imagem.

2. ponto no infinito projetado sobre ponto próprio.

   Por exemplo, a imagem de $P = [0\ 0\ 1\ 0]$ também é $p = [0\ 0\ 1]$. Ou seja, a direção $(0, 0, 1)$ do espaço é transformada no ponto $(0, 0)$ da imagem. Isto corresponde ao fato que observamos acima: feixes de retas do espaço paralelas à direção $(0, 0, 1)$ são projetados em um feixe de retas que concorrem na origem. O ponto próprio correspondente à projeção de um ponto no infinito é chamado de *ponto de fuga* da direção correspondente. Assim, a origem é o ponto de fuga das retas do espaço paralelas ao eixo Z.

3. ponto próprio projetado sobre ponto no infinito.

   Esta é a relação dual a anterior; por exemplo, a imagem de $P = [1\ 0\ 0\ 1]$ é $p = [1\ 0\ 0]$. Isto pode ser interpretado do seguinte modo: o feixe de retas do espaço que são concorrentes no ponto $(1, 0, 0)$ projeta-se como um feixe de retas da imagem paralelas à direção $(0, 1)$.

4. ponto no infinito projetado sobre ponto no infinito.

   Por exemplo, a imagem de $P = [1\ 0\ 0\ 0]$ é $p = [1\ 0\ 0]$, que significa que as retas do espaço paralelas ao eixo x permanecem paralelas em projeção. Na verdade, o paralelismo é preservado na projeção perspectiva para retas paralelas ao plano de projeção.

Há, portanto, uma grande vantagem em se utilizar coordenadas homogêneas (ou seja, as idéias da Geometria Projetiva) para expressar projeções perspectivas: elas passam a ser definidas não só para todos os pontos do espaço (exceto a origem), mas também para direções, propiciando um tratamento unificado.

Além disto, é importante observar que a classe de transformações projetivas engloba as transformações lineares e afins. Tais transformações podem ser vistas como transformações projetivas que levam pontos próprios em pontos próprios e pontos no infinito em pontos no infinito. De fato, a transformação linear de R^m em R^n cuja matriz é A corresponde à colineação de RP^m em RP^n dada pela matriz $\begin{bmatrix} A & 0 \\ 0 & 1 \end{bmatrix}$. A transformação afim $x \mapsto Ax + t$, por sua vez, é representada pela colineação de matriz $\begin{bmatrix} A & t \\ 0 & 1 \end{bmatrix}$.

Portanto, transformações projetivas fornecem um tratamento unificado para os diversos tipos de transformação do espaço que devem ser consideradas na modelagem matemática do processo de obter a projeção de um ponto do espaço. Além disso, como podem ser expressas através de matrizes, podem utilizar todo o aparato existente em sistemas computacionais para dar suporte a operações com matrizes.

2.4 Transformações de Câmera

Nesta seção, obteremos a expressão matemática da função que associa, a cada ponto do espaço, expresso em um sistema de referência ortogonal, a posição correspondente na imagem capturada por uma câmera. Naturalmente, para escrever tal função necessitamos ter informação sobre a câmera; ou seja, precisamos especificar certos parâmetros, que definem seu comportamento. Tais parâmetros formam dois grupos. Os parâmetros *extrínsecos* descrevem a posição e orientação da câmera. Os parâmetros *intrísecos* descrevem seu funcionamento interno.

2.4.1 Sistema de coordenadas

Para expressar a correspondência entre pontos do espaço e pontos da imagem, é conveniente considerar quatro sistemas de coordenadas, de modo que a transformação de câmera possa ser expressa como a composta de transformações simples realizadas entre estes sistemas.

Os sistemas de coordenadas considerados são:

Sistema de Coordenadas do Mundo (SCM), também chamado de Sistema de Coordenadas Absoluto (SCA). É um sistema tridimensional, escolhido de modo a ser conveniente para a descrição da cena (ou para os objetos utilizados para calibrar as câmeras). Na Figura 2.8, o SCM tem origem no ponto O e as coordenadas com relação a este referencial são denotadas por (X, Y, Z).

Sistema de Coordenadas da Câmera (SCC), um sistema tridimensional com origem no centro óptico da câmera (ou seja, o orifício, no caso de uma câmera *pinhole*). Os eixos X e Y são escolhidos de modo a serem paralelos às bordas da imagem a ser formada sobre o plano de projeção, enquanto o terceiro eixo é perpendicular a aquele plano. Na Figura 2.8 o sistema de coordenadas da câmera tem origem no ponto $\tilde{O}$ e as coordenadas neste referencial são denotadas por $(\tilde{X}, \tilde{Y}, \tilde{Z})$. Denotamos por f a distância do plano de projeção π ao centro óptico, chamada de *distância focal* da câmera. Portanto, no SCC o plano de projeção (ou seja, de formação da imagem) é o plano de equação $\tilde{Z} = f$.

Sistema de Coordenadas de Imagem (SCI), um sistema de referência bidimensional, situado no plano de projeção. Sua origem é o ponto C, obtido por meio da projeção ortogonal do centro óptico $\tilde{O}$ sobre o plano de projeção. Denotaremos as coordenadas de um ponto neste referencial por (x, y).

Sistema de Coordenadas em Pixel (SCP), um sistema bidimensional, com coordenadas expressas em pixels, que define a posição de um ponto da imagem com relação à grade de pixels. Em geral, a origem deste sistema é o canto superior (ou infe-

rior) esquerdo da imagem. Denotaremos as coordenadas neste sistema por (u, v).

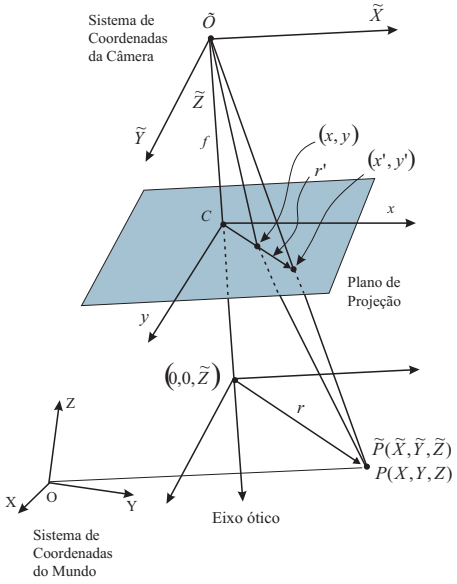


Figura 2.8: Geometria da câmera com projeção perspectiva e distorção da lente.

O processo de imageamento de um ponto do espaço através de uma câmera digital consiste em uma sequência de transformações entre estes espaços, descritos em detalhe nas seções a seguir. Inicialmente, consideraremos uma câmera *pinhole*; a seguir, trataremos da modelagem de uma câmera com lentes.

2.4.2 Do SCM ao SCC: mudança de referencial 3D

O primeiro passo na obtenção da posição na imagem correspondente a um ponto do espaço (X, Y, Z) é expressar estas coordenadas tridimen-

sionais no SCC. Ou seja, devemos realizar a mudança de coordenadas entre dois referenciais ortogonais do espaço tridimensional. Seja T o vetor que fornece a origem O do mundo no referencial SCC, e seja R a matriz cujas colunas r_1, r_2 e r_3 são as coordenadas dos vetores unitários i, j e k dos eixos do SCM com relação à base formada pelos vetores $\tilde{i}$, $\tilde{j}$ e $\tilde{k}$ dos eixos do SCC. Note que, como os dois referenciais são ortogonais, a matriz R é ortogonal (isto é, $RR^t = I$).

Dado um ponto $P = (X, Y, Z)$, suas coordenadas no SCC são as do vetor $\tilde{O}P = \tilde{O}O + OP$, portanto dadas por

$$(\tilde{X}, \tilde{Y}, \tilde{Z}) = T + Xr_1 + Yr_2 + Zr_3.$$

Assim, denotando o vetor de coordenadas do ponto no SCC por $\tilde{P}$, temos:

$$\tilde{P} = RP + T; \tag{2.2}$$

ou, ainda, em coordenadas homogêneas:

$$\begin{bmatrix} \tilde{X} \\ \tilde{Y} \\ \tilde{Z} \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{2.3}$$

A transformação inversa é obtida multiplicando os dois lados da equação 2.2 por R^t, obtendo-se, assim,

$$P = R^t\tilde{P} - R^tT. \tag{2.4}$$

A matriz R e o vetor T determinam a orientação e posição da câmera, e representam seus parâmetros extrínsecos. É importante observar que, embora a matriz R tenha 9 elementos, eles não são independentes, já que R é ortogonal. Na verdade, R pode ser codificada através de apenas 3 números reais, que caracterizam a rotação do espaço que leva os eixos de um sistema nos eixos do outro (caso os sistemas sejam escolhidos com a mesma orientação). Isto pode ser feito de vários modos:

Por meio dos ângulos de Euler, ou seja dos ângulos sucessivos de rotação em torno dos eixos.

Por meio da forma de Rodriguez, em que a rotação é codificada através de um vetor $\omega = (\omega_1, \omega_2, \omega_3)$, que representa a direção do eixo de rotação e o ângulo de rotação em torno deste eixo (através de sua norma). A matriz de rotação associada a ω é dada por

$$R = \cos\theta I + [\omega\times]\,\frac{\sin\theta}{\theta} + ww^t\,\frac{1-\cos\theta}{\theta^2},$$

onde $\theta = ||\omega||$ e $[\omega\times] = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z 0 & -\omega_x & \\ \omega_y & \omega_x & 0 \end{bmatrix}$.

Por meio de quaternions (ver [30]).

2.4.3 Do SCC ao SCI: projeção perspectiva

A transformação do SCC para o SCI, em câmeras *pinhole*, consiste em uma projeção perspectiva (o caso em que há lentes é tratado na Subseção 2.4.6).

Como visto na Seção 2.3.3, a projeção perspectiva efetuada por uma câmera *pinhole* com distância focal f pode ser escrita como:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \simeq \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \tilde{X} \\ \tilde{Y} \\ \tilde{Z} \\ 1 \end{bmatrix}. \tag{2.5}$$

A transformação projetiva descrita pela equação (2.5) é não inversível. Cada ponto do espaço corresponde a um único ponto da imagem, mas um ponto da imagem corresponde a uma infinidade de pontos do espaço. Mais precisamente, o ponto (x, y) da imagem é a projeção de todos os pontos do espaço SCC da forma $\lambda(x, y, f)$, onde $\lambda \neq 0$.

2.4.4 Do SCI ao SCP: registro no sensor

Em uma câmera digital, há ainda outra transformação importante a ser levada em conta. Quando um raio luminoso atinge o plano de formação da imagem, ele é registrado por um sensor. Os sensores

(que tipicamente usam tecnologia CCD ou CMOS) usualmente estão dispostos segundo uma matriz retangular. Pode ocorrer, porém, do espaçamento entre linhas ser diferente do espaçamento entre colunas. Além disso, pode ser conveniente considerar a possibilidade de que as colunas não sejam dispostas perpendicularmente às linhas (devido a imperfeições geométricas de construção ou devido ao processo de aquisição dos valores dos sensores). Este fato é modelado admitindo que o eixo horizontal do SCI corresponde exatamente ao eixo horizontal do SCP, mas que os eixos verticais possam não estar perfeitamente alinhados. Finalmente, enquanto a origem do SCI é a projeção do centro óptico (usualmente localizada no centro da imagem), a origem do SCP é normalmente um de seus cantos.

A discussão acima nos leva a modelar a transformação do SCI para o SCP como uma transformação afim da seguinte forma:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & \tau & u_c \\ 0 & s_y & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{2.6}$$

Os coeficientes da matriz acima formam, juntamente, com a distância focal f os parâmetros intrínsecos da câmera. Os significados destes parâmetros são os seguintes:

- s_x and s_y representam o número de pixels por unidade de comprimento, nas direções horizontal e vertical, respectivamente; na maior parte das câmeras, idealmente s_x e s_y são iguais, ou seja, os pixels são quadrados;

- u_c and v_c fornecem a posição, em pixels, da projeção ortogonal C da origem sobre o plano de projeção; na maior parte das câmeras, C está no centro da imagem e os valores de u_c e v_c são idealmente iguais à metade das dimensões da imagem;

- τ é a tangente do ângulo que as colunas de pixels formam com a perpendicular às linhas; na maior parte das câmeras, idealmente as colunas são perpendiculares às linhas, ou seja, $\tau = 0$.

2.4.5 Compondo as transformações

A transformação que leva um ponto do SCM à sua projeção em SCP pode ser obtida compondo (isto é, multiplicando) as transformações das equações (2.3), (2.5) e (2.6). Representando as coordenadas homogêneas do ponto no SCM como $[P]$ e da imagem no SCP como $[p]$, temos:

$$[p] \simeq \begin{bmatrix} s_x & \tau & u_c \\ 0 & s_y & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} [P];$$

ou ainda,

$$[p] \simeq \begin{bmatrix} fs_x & f\tau & u_c \\ 0 & fs_y & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R & T \end{bmatrix} [P]. \tag{2.7}$$

A matriz $K = \begin{bmatrix} fs_x & f\tau & u_c \\ 0 & fs_y & v_c \\ 0 & 0 & 1 \end{bmatrix}$ é chamada de *matriz de calibração* e reúne os parâmetros intrínsecos da câmera, enquanto $[R \quad T]$ representa os extrínsecos.

A expressão de K mostra que as ações dos parâmetros discutidos anteriormente sobre a equação de imageamento não são independentes. Isto tem uma consequência importante para o procedimento de calibração, que consiste em estimar os parâmetros a partir de pares de pontos do espaço e suas respectivas posições na imagem. Como os parâmetros f, s_x e s_y aparecem na transformação de câmera através dos seus produtos fs_x e fs_y, não é possível estimar os valores individuais de f, s_x e s_y (a menos que se tenha informações a respeito de s_x e de s_y provenientes do fabricante da câmera).

Assim, na maior parte dos casos é conveniente expressar a matriz de calibração como $K = \begin{bmatrix} f_x & c & u_c \\ 0 & f_y & v_c \\ 0 & 0 & 1 \end{bmatrix}$.

A transformação dada pela equação (2.7) não é inversível. Um ponto p da imagem corresponde a uma reta do espaço, que pode ser

obtida em dois passos. Inicialmente, calculamos $\bar{p} = K^{-1}p$, que corresponde à imagem do ponto P que seria produzida por uma câmera em que a matriz de calibração fosse a identidade (isto é, tal que $f = s_x = s_y = 1$ e $\tau = u_c = v_c = 0$). Em outras palavras, $\bar{p}$ é obtido a partir de p removendo o efeito dos parâmetros intrínsecos; as coordenadas $\bar{x}$ e $\bar{y}$ de $\bar{p}$ são chamadas de *coordenadas normalizadas* do ponto p da imagem. Os pontos cuja projeção é p são que, no SCC, têm coordenadas da forma $\lambda(\bar{x}, \bar{y}, 1)$, onde $\lambda \in R$; para expressar estes pontos no SCM basta usar a equação (2.4). Este fato será empregado, no Capítulo 3, para reconstruir um ponto tridimensional a partir de suas imagens.

2.4.6 Modelando as lentes

Até este momento, consideramos apenas câmeras do tipo *pinhole*. Embora câmeras reais possuam lentes, a aproximação fornecida pelo modelo *pinhole* é suficiente para muitas aplicações. No caso de Fotografia 3D, no entanto, não levar em conta o desvio dos raios luminosos causados pelas lentes pode levar a imprecisões inaceitáveis na reconstrução tridimensional. Por outro lado, a modelagem exata do comportamento das lentes não é prática, devido a sua complexidade.

A alternativa consiste em utilizar um modelo empírico para a distorção. Este modelo considera apenas a distorção radial, que afeta igualmente todos os pontos da imagem situados a uma mesma distância do centro (isto faz SENTIDO DEvido à simetria rotacional dos sistemas ópticos em torno de seu eixo).

A distorção radial afeta a transformação do SCC para o SCI. Ao invés da imagem se formar, no plano de projeção, no ponto (x, y) de acordo com a projeção perspectiva dada pela equação (2.5), ela se forma em um ponto (x', y'). Os pares (x, y) e (x', y') estão relacionados por

$$(x, y) = (1 + d)\,(x', y')\,, \tag{2.8}$$

onde o fator de distorção d depende unicamente de $r = \sqrt{(x')^2 + (y')^2}$.

A dependência entre d e r é usualmente modelada por uma função polinomial; por exemplo, é comum considerar que $d = k_1 r^2 + k_2 r^4$; os coeficientes k_1 e k_2 são chamados de *coeficientes de distorção radial*.

A menos que o ângulo de visão da lente seja muito grande, basta, em geral, levar em conta o coeficiente k_1 na modelagem da distorção radial.

2.5 Calibração de Câmeras

O problema de calibrar uma câmera é o de estimar seus parâmetros intrínsecos e extrínsecos. A forma mais natural de calibração consiste em obter, na imagem, um conjunto de pontos $p_1, p_2, ..., p_n$ que correspondam a pontos conhecidos $P_1, P_2, ..., P_n$ do espaço tridimensional e obter os valores dos parâmetros intrísecos tais que as imagens $\hat{p}_1, \hat{p}_2, ..., \hat{p}_n$ obtidas com uma câmera, com estes parâmetros, estejam o mais próximo possível das imagens observadas, de acordo com uma certa medida de erro. Por exemplo, pode-se formular o problema de calibração como o de minimizar $\sum_{i=0}^{n} ||\hat{p}_i - p_i||^2$ para todos os possíveis valores dos parâmetros R, T e K. Versões diferentes do problema são obtidas de acordo com as restrições adotadas para os parâmetros intrínsecos. Algumas possibilidades para formular o problema através de um modelo simplificado são:

- ignorar a distorção radial, modelando a câmera como *pinhole*;
- considerar que a projeção do centro óptico é conhecida e coincide com o centro da imagem;
- considerar que os pixels são quadrados (isto é, que $s_x = s_y$ e, assim, $f_x = f_y$);
- considerar que as colunas de pixels são ortogonais às linhas (isto é, $\tau = c = 0$).

Em qualquer caso, calibrar uma câmera consiste em resolver um problema de otimização não-linear do tipo mostrado em (2.7), o que pode ser feito, por exemplo, usando o método iterativo de Levenberg-Marquardt ([63]). O maior problema, neste tipo de método, é encontrar uma boa solução inicial. Nas seções a seguir, descreveremos dois métodos que podem ser empregados para encontrar uma câmera que tanto pode ser usada como o resultado da calibração como ser empregada como a solução inicial de um método iterativo.

Ambos os algoritmos funcionam através da captura de uma ou mais imagens de um *padrão de calibração*. Tais padrões são construídos de modo que seja fácil identificar (de preferência, automaticamente) na imagem pontos cujas coordenadas sejam conhecidas. A Figura 2.9 mostra padrões de calibração propostos em diversos trabalhos.

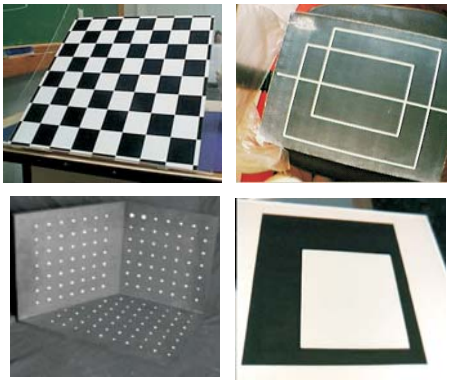


Figura 2.9: Padrões de calibração.

2.5.1 Método de Tsai

Em um artigo clássico ([83]), Roger Tsai propôs um método simples de calibração de câmeras. Na verdade, Tsai propôs dois métodos distintos. Abaixo, descrevemos em detalhes a versão coplanar, que se aplica à situação em que todos os pontos utilizados para calibração estão situados em um único plano.

O método de Tsai para o caso coplanar baseia-se nas seguinte hipóteses fundamentais:

1. Os referenciais do SCM e do SCC têm a mesma orientação. Isto significa que R é efetivamente uma matriz de rotação (ou seja, seu determinante é igual a 1). Uma consequência deste fato é a de que, caso desejemos utilizar um referencial de orientação positiva para o SCM e, além disso, que o eixo $\tilde{Z}$ do SCC aponte

para a cena, os eixos do SCC devem ser escolhidos conforme indica a Figura 2.10.

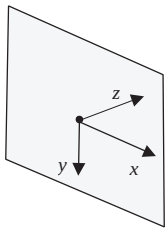


Figura 2.10: Escolha dos eixos para o método de Tsai.

2. Todos os pontos de calibração estão no plano $Z = 0$ do SCM.

3. Os parâmetros intrínsecos da câmera, com exceção da distância focal f e do coeficiente de distorção angular k_1, são conhecidos, de modo que a partir de cada ponto na imagem (u, v) se possa determinar o ponto correspondente (x', y') no plano de projeção. Na verdade, no entanto, é possível utilizar o método mesmo que esses parâmetros não sejam conhecidos, desde que conheçamos o ponto (u_c, v_c) da imagem que corresponde à projeção do centro óptico (em geral, o centro da imagem) e que possamos supor que $\tau = 0$ e que $s_x = s_y$, ou seja, que os pixels da câmera são quadrados [1]. Nestas condições, tomamos $s_x = s_y = 1$ e, a partir de cada ponto (u, v) na imagem, obtemos o ponto correspondente no plano de projeção através das relações:

$$x' = u - u_c, \quad \text{e} \quad y' = v - v_c. \tag{2.9}$$

É importante observar que, no caso de arbitrarmos os valores de s_x e s_y como acima, o valor da distância focal que resulta do processo de calibração é expresso em pixels. Para obter seu valor real é necessário conhecer as dimensões de cada pixel (ou, equivalentemente, da imagem capturada).

[1] Também podemos efetuar a calibração se os pixels forem retangulares desde que a razão s_y/s_x seja conhecida.

4. A origem do sistema do mundo é escolhida de modo a não se projetar próximo ao centro da imagem ou ao eixo x da imagem (esta hipótese se destina a assegurar que a componente T_y do vetor de translação entre o SCC e o SCM não assuma um valor próximo de zero; isto é importante para o bom comportamento numérico do método). Note que é sempre possível escolher a origem do SCM de modo que esta condição seja cumprida.

Suponhamos que tenhamos uma imagem capturada pela câmera que mostre n pontos coplanares $P_1, P_2, ..., P_n$ do espaço, de coordenadas conhecidas. Sejam $(X_i, Y_i, 0)$ as coordenadas de P_i no SCM e (u_i, v_i) as coordenadas (em pixels) do ponto correspondente na imagem. As coordenadas (x'_i, y'_i) no plano de projeção podem ser obtidas pela equação (2.8). O ponto (x'_i, y'_i), por sua vez, se relaciona com a projeção perspectiva (x_i, y_i) através da expressão

$$(x_i, y_i) = (1 + k_1 r^2)(x'_i, y'_i),$$

onde $r = \sqrt{(x')^2 + (y')^2}$.

Como vimos na Seção 2.3.3, a projeção perspectiva (x_i, y_i) de um ponto de coordenadas (X_i, Y_i, Z_i) é dada por:

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \simeq \begin{bmatrix} fr_{xx} & fr_{xy} & fr_{xz} & fT_x \\ fr_{yx} & fr_{yy} & fr_{yz} & fT_y \\ r_{zx} & r_{zy} & r_{zz} & T_z \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix},$$

onde f é a distância focal da câmera e $R = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{bmatrix}$ e $T = [T_x\ T_y\ T_z]^t$ são, respectivamente, a matriz de rotação e o vetor de translação que relacionam o SCM e o SCC.

Levando em conta que $Z_i = 0$, estas relações podem ser reescritas na forma

$$x_i = f\frac{r_{xx}X_i + r_{xy}Y_i + T_x}{r_{zx}X_i + r_{zy}Y_i + T_z},$$

$$y_i = f\frac{r_{yx}X_i + r_{yy}Y_i + T_y}{r_{zx}X_i + r_{zy}Y_i + T_z}.$$

Dividindo as duas equações, temos:

$$\frac{x_i}{y_i} = \frac{r_{xx}X_i + r_{xy}Y_i + T_x}{r_{yx}X_i + r_{yy}Y_i + T_y}. \tag{2.10}$$

No entanto, como $(x_i, y_i) = (1 + k_1 r^2)(x'_i, y'_i)$, as razões x_i/y_i e x'_i/y'_i são iguais. Substituindo x_i/y_i por x'_i/y'_i em (2.10) e eliminando os denominadores, obtemos:

$$y'_i X_i r_{xx} + y'_i Y_i r_{xy} + y'_i T_x - x'_i X_i r_{yx} - x'_i Y_i r_{yy} = x'_i T_y. \tag{2.11}$$

Finalmente, dividindo por T_y e fazendo $U_1 = r_{xx}/T_y$, $U_2 = r_{xy}/T_y$, $U_3 = T_x/T_y$, $U_4 = r_{yx}/T_y$ e $U_5 = r_{yy}/T_y$, encontramos:

$$y'_i X_i U_1 + y'_i Y_i U_2 + y'_i U_3 - x'_i X_i U_4 - x'_i Y_i U_5 = x'_i.$$

Utilizando os n pontos de calibração, obtém-se um sistema de equações lineares da forma

$$AU = b,$$

onde cada linha de A é dada por $a_i = [y'_i X_i \;\; y'_i Y_i \;\; y'_i \;\; -x'_i X_i \;\; x'_i Y_i]$ e $b = [x'_1, \; x'_2, ..., x'_n]^t$.

Para que o sistema seja determinado são necessários pelo menos 5 pontos de calibração. Tipicamente, é utilizada uma quantidade bem maior de pontos de calibração. Neste caso, o sistema linear possui mais equações do que incógnitas e deve ser resolvido como um problema de mínimos quadrados. Ou seja, devemos encontrar $U = [U_1 \; U_2 \; U_3 \; U_4 \; U_5]$ tal que $||AU - b||$ seja mínima. Por exemplo, pode-se usar a decomposição em valores singulares de A (ver [63]).

Uma vez encontrados U_1, U_2, U_3, U_4 e U_5, os parâmetros da câmera são encontrados utilizando a sequência de etapas descrita a seguir.

Primeira etapa: *Cálculo da magnitude de T_y.*

Os elementos das duas primeiras linhas e colunas de R podem ser expressos em termos dos valores U_i e de T_y, através das relações $r_{xx} = T_y U_1$, $r_{xy} = T_y U_2$, $r_{yx} = T_y U_4$ e $r_{yy} = T_y U_5$.

O fato de as linhas de R serem unitárias e ortogonais entre si permite escrever o seguinte sistema de equações envolvendo T_y, r_{xz} e r_{yz}.

$$\begin{cases} (U_1^2+U_2^2)T_y^2+r_{xz}^2 &= 1, \\ (U_4^2+U_5^2)T_y^2+r_{yz}^2 &= 1, \\ (U_1U_4+U_2U_5)T_y^2+r_{xz}r_{yz} &= 0. \end{cases} \tag{2.12}$$

Eliminando r_{xz} e r_{yz} do sistema, obtemos a seguinte equação em T_y^2:

$$DT_y^4 - ST_y^2 + 1 = 0, \tag{2.13}$$

onde $\mathrm{S} = U_1^2+U_2^2+U_4^2+U_5^2$ e $D = (U_1U_5-U_2U_4)^2$.

Se $D = 0$, a solução de (2.13) é

$$T_y = \frac{1}{S} = \frac{1}{U_1^2+U_2^2+U_4^2+U_5^2};$$

senão, (2.13) tem como solução:

$$T_y^2 = \frac{S \pm \sqrt{S^2-4D}}{2D} = \frac{2}{S \mp \sqrt{S^2-4D}}. \tag{2.14}$$

No entanto, somando as duas primeiras equações em (2.12), verifica-se facilmente que $ST_y \leq 2$, o que mostra que, das soluções fornecidas por (2.14), a correta é a dada por

$$T_y^2 = \frac{2}{S+\sqrt{S^2-4D}}. \tag{2.15}$$

Note que esta expressão é válida mesmo no caso em que $D = 0$, analisado acima.

Segunda etapa: *Determinação do sinal de T_y.*

O sinal de T_y vai determinar os sinais de $r_{xx}, r_{xy}, r_{yx}, r_{yy}$ e T_x, já que estes parâmetros podem ser calculados através das expressões:

$$\begin{aligned} r_{xx} &= U_1T_y, \\ r_{xy} &= U_2T_y, \\ T_x &= U_3T_y, \\ r_{yx} &= U_4T_y, \\ r_{yy} &= U_5T_y. \end{aligned} \tag{2.16}$$

Desejamos escolher o sinal de T_y de modo que o sistema de coordenadas da câmera tenha o seu eixo z orientado na direção da cena. Isto é, de modo que o denominador das equações de projeção seja positivo para todos os pontos da cena. Ou, ainda, de modo que os numeradores destas mesmas equações tenham o mesmo sinal das coordenadas na imagem.

Assim, podemos adotar o seguinte procedimento para verificar o sinal de T_y. Provisoriamente, adotamos o sinal positivo para T_y e calculamos os demais parâmetros pelas equações (2.16). A seguir, escolhemos um ponto $P_0 = (X_0, Y_0, 0)$ cuja projeção no plano da imagem (x'_0, y'_0) esteja suficientemente afastada de um dos eixos da imagem (digamos, do eixo y) e verificamos se o sinal de $r_{xx}X_0 + r_{xy}Y_0 + T_x$ é igual ao de x'_0. Em caso afirmativo, mantemos o sinal de T_y; senão, trocamos o sinal de T_y (e dos demais parâmetros calculados pelas equações (2.16)).

Terceira etapa: *Cálculo dos demais elementos da matriz de rotação R.*

Cada linha da matriz R é um vetor unitário. Logo:

$$\begin{aligned} r_{xz} &= \pm\sqrt{1 - r_{xx}^2 - r_{xy}^2}, \\ r_{yz} &= \pm\sqrt{1 - r_{yx}^2 - r_{yy}^2}. \end{aligned} \tag{2.17}$$

Provisoriamente, tomamos r_{xz} positivo; o sinal de r_{yz} é escolhido usando o fato de que as duas primeiras linhas de R são ortogonais. Assim, se $r_{xx}r_{yx} + r_{xy}r_{yy} > 0$, então r_{yz} é tomado com sinal negativo; caso contrário, r_{yz} é positivo.

Uma vez conhecidas as duas primeiras linhas de R, a terceira linha pode ser obtida através do produto vetorial das duas primeiras. Assim:

$$\begin{aligned} r_{zx} &= r_{xy}r_{yz} - r_{xz}r_{yy}, \\ r_{zy} &= r_{xz}r_{yx} - r_{xx}r_{yz}, \\ r_{zz} &= r_{xx}r_{yy} - r_{xy}r_{yx}. \end{aligned} \tag{2.18}$$

Note que, a exemplo de r_{yz}, os sinais de r_{zx} e r_{zy} dependem do sinal de r_{xz}, que escolhemos como positivo de modo arbitrário.

Na Quarta etapa o acerto desta escolha vai ser testada e, se for o caso, o sinal de r_{xz} (e, em consequência, de r_{yz}, r_{zx} e r_{zy}) será trocado.

Quarta etapa: *Cálculo aproximado de f e T_z ignorando a distorção da lente.*

Neste momento, já obtivemos estimativas para quase todos os parâmetros de calibração (embora alguns sinais tenham sido estabelecidos de modo apenas provisório). Falta determinar somente T_z e a distância focal f, além do parâmetro de distorção radial k_1.

Para obter uma primeira estimativa de f e T_z, consideraremos $k_1 = 0$. Portanto, provisoriamente tomamos $(x_i, y_i) = (x'_i, y'_i)$. As equações em (2.10) permitem obter, para cada ponto de calibração $P_i = (X_i, Y_i, Z_i)$, com imagem (x'_i, y'_i), as seguintes equações envolvendo f e T_z:

$$\begin{array}{rcl} (r_{xx}X_i + r_{xy}Y_i + T_x)f - x'_iT_z & = & x'_i(r_{zx}X_i + r_{zy}Y_i), \\ (r_{yx}X_i + r_{yy}Y_i + T_y)f - y'_iT_z & = & y'_i(r_{zx}X_i + r_{zy}Y_i). \end{array} \tag{2.19}$$

Obtemos, assim, um sistema com $2n$ equações nas incógnitas f e T_z, que deve ser resolvido como um problema de mínimos quadrados.

Se o sinal positivo atribuído a r_{xz} na etapa anterior estiver correto, devemos encontrar $f > 0$. Se isto não ocorrer, o sinal de r_{xz} deve ser negativo e, em consequência, os sinais de r_{yz}, r_{zx}, r_{zy} e dos valores calculados nesta etapa para f e T_z devem ser trocados.

Quinta etapa: *Cálculo exato de f, T_z e k_1.*

Esta etapa só é necessária se desejarmos incluir a distorção radial no modelo de câmera. Neste caso, as equações lineares em (2.19) se transformam nas seguintes equações não lineares,

com incógnitas f, T_z e k_1:

$$(r_{xx}X_i + r_{xy}Y_i + T_x)f - \frac{x'_i(r_{zx}X_i + r_{zy}Y_i + T_z)}{1 + k_1((x'_i)^2 + (y'_i)^2)} = 0,$$
$$(r_{yx}X_i + r_{yy}Y_i + T_y)f - \frac{y'_i(r_{zx}X_i + r_{zy}Y_i + T_z)}{1 + k_1((x'_i)^2 + (y'_i)^2)} = 0.$$

Temos, portanto, um problema de regressão não linear, cujo objetivo é miniminizar a soma dos quadrados dos erros nas n equações assim obtidas. Este problema pode ser resolvido por métodos iterativos, como o de Levenberg-Marquardt, usando como solução inicial os valores de f e T_z obtidos na Quarta etapa e $k_1 = 0$.

2.5.2 Um exemplo completo

Exemplificaremos os diversos passos do algoritmo de Tsai para a situação dada na Figura 2.11, que mostra uma imagem, de 640 × 480 pixels, de um jogo de futebol.

A câmera pode ser calibrada usando o método de Tsai para pontos coplanares devido ao fato de que, na fotografia, aparece um número suficiente de pontos de coordenadas conhecidas no espaço – as marcações do campo, indicadas na figura. Fixaremos o SCM com origem na bandeira de escanteio, com eixo X na direção da linha lateral, eixo Y na direção da linha de fundo e eixo Z para cima. O SCI tem origem no centro da imagem, eixo horizontal orientado para a direita e eixo vertical orientado para baixo (para que, com o terceiro eixo orientado para dentro da imagem, determinem um sistema de orientação positiva).

Como não temos nenhum conhecimento da câmera que obteve a imagem, consideramos apenas que os pixels são quadrados e tomamos $s_x = s_y = 1$. Em consequência, a distância focal f obtida na calibração será expressa em pixels da imagem. Para converter f em uma medida nas unidades usuais de comprimento necessitaríamos conhecer o tamanho real s_x (ou, equivalentemente, as dimensões reais da imagem capturada pela câmera), expressas, por exemplo, em mm.

A Tabela 2.5.2 mostra os pontos de calibração $(X_i, Y_i, 0)$ e suas respectivas imagens, expressas em pixels (u_i, v_i) e em coordenadas no

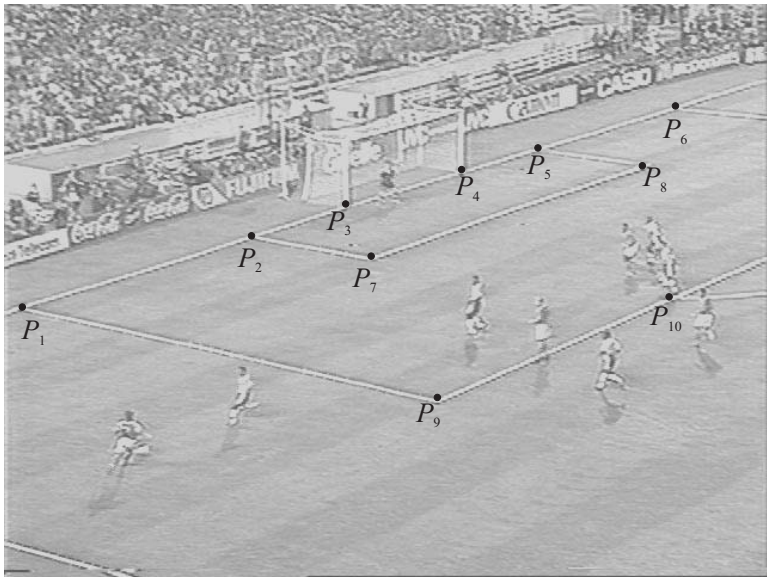


Figura 2.11: Cena para calibração.

plano da imagem (x'_i, y'_i) (obtidas através das expressões $x'_i = u_i - u_c$, $y'_i = v_i - v_c$, onde $u_c = 320$ e $v_c = 240$ são as coordenadas do centro da imagem.

Com os dados da Tabela 2.5.2 obtemos um sistema com 10 equações, nas incógnitas U_1, U_2, U_3, U_4 e U_5 que, resolvido por mínimos quadrados, fornece $U_1 = 0.2169$, $U_2 = 0.1712$, $U_3 = 0.0419$, $U_4 = -0.0617$ e $U_5 = -5.5907$.

Daí, obtemos, pela equação (2.15), $T_y^2 = 13.0932$, ou seja, $T_y = \pm 3.6184$ e, daí, pela equação (2.16), os seguintes valores: $r_{xx} = \pm 0.7847$, $r_{xy} = \pm 0.6195$, $r_{yx} = \pm 0.1515$, $r_{yy} = \mp 0.2231$, $T_x = \mp 20.2297$.

Com a primeira escolha de sinal, a equação que fornece a abscissa da projeção do ponto P_1 tem numerador igual a $0.7847 \times 0 + 0.6195 \times 13.84 - 20.2297 = -11.6556$. Como este sinal é o mesmo de $x'_1 = -305$, concluímos que os sinais estão corretos.

i	X_i	Y_i	u_i	v_i	x'_i	y'_i
1	0.00	13.84	15	254	−305	14
2	0.00	24.84	209	195	−111	−45
3	0.00	30.34	287	169	−33	−71
4	0.00	37.66	383	141	63	−99
5	0.00	43.16	449	121	129	−119
6	0.00	54.16	562	87	242	−153
7	5.50	24.84	307	213	−13	−27
8	5.50	43.16	536	135	216	−105
9	16.50	13.84	362	333	42	93
10	16.50	26.69	563	245	243	5

Tabela 2.1: Pontos para calibração.

Os demais parâmetros de rotação podem, então, ser calculados pelas equações (2.17) e (2.18), encontrando-se: $r_{xz} = \pm 0.0201$, $r_{yz} = \pm 0.9629$, $r_{zx} = \pm 0.6010$, $r_{zy} = \mp 0.7526$, $r_{zz} = \mp 0.2690$.

Finalmente, escolhendo a primeira alternativa de sinal, formando o sistema com 20 equações e 2 incógnitas em f e T_z e resolvendo-o, encontramos $f = -1755.14$ e $T_z = -56.7600$. Como o sinal de f é negativo, os sinais de r_{xz}, r_{yz}, r_{zx} e r_{zy} devem ser trocados, o mesmo ocorrendo com os de f e T_z.

Os valores finais dos parâmetros da câmera são $f = 1755.14$, $T = [-20.2297 \quad 3.6184 \quad 56.7600]^t$ e $R = \begin{bmatrix} 0.7847 & 0.6195 & -0.0201 \\ 0.1515 & -0.2231 & -0.9629 \\ -0.6010 & 0.7526 & -0.2690 \end{bmatrix}$.

Com estas informações, podemos, por exemplo, obter a posição da câmera no SCM. Para tal, basta converter a origem do SCC para o SCM, através da equação (2.4). O resultado é $-R^tT = [49.4412 \ -29.3773 \ 18.3459]$, o que posiciona a câmera a aproximadamente 18 m de altura, afastada aproximadamente 29 metros da linha lateral e situada a 49 m da linha de fundo (isto é, posicionada aproximadamente na linha do meio de campo). O valor da distância focal ($f = 1755.14$), expresso em pixels, permite obter o ângulo de visão da

câmera, ilustrado na Figura 2.12. Temos $\tan(\frac{\alpha}{2}) = \frac{240}{1755.14} = 0.1367$ e, daí, $\alpha = 15.58^o$.

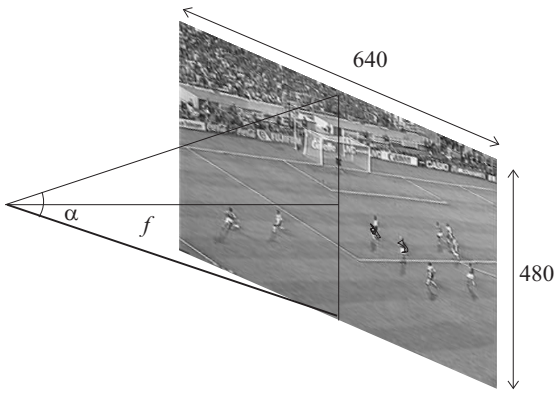


Figura 2.12: Determinação do ângulo de visão.

2.5.3 Método de Zhang

Padrões de calibração coplanares são mais simples de construir do que padrões tridimensionais, o que faz com que métodos baseados neles (como a versão coplanar do método de Tsai vista na Seção 2.5.1) sejam atraentes. Eles apresentam, no entanto, a desvantagem de que a calibração somente é precisa para pontos do espaço que estejam próximos ao plano do padrão. Isto faz com que eles não sejam ideais para aplicações de reconstrução tridimensional.

Uma alternativa ao emprego de padrões tridimensionais é o método de Zhang [94], que usa um padrão que é bidimensional mas que é posicionado em diversas posições do espaço, dando origem a m imagens, conforme ilustrado na Figura 2.13. Em cada uma destas imagens, admitimos que haja n pontos cujas coordenadas no sistema de referência do padrão sejam conhecidas. Admitimos, ainda, que os parâmetros intrínsecos da câmera são os mesmos em todas as imagens; somente os parâmetros extrínsecos mudam quanto o padrão é reposicionado.

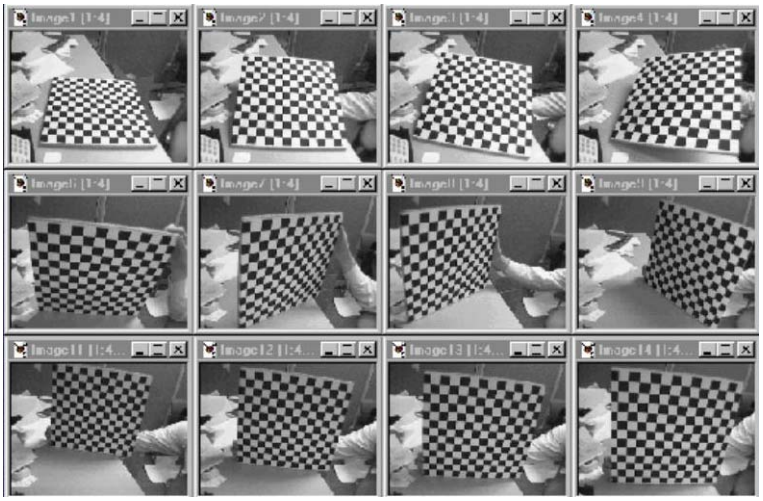


Figura 2.13: Padrão visto em diversas posições.

Como no método de Tsai, inicialmente consideramos uma câmera *pinhole* e, a seguir, consideramos a distorção radial. No entanto, o fato de termos mais de uma imagem do padrão permite que, no método de Zhang, possamos ajustar um modelo mais geral de câmera, definido por uma transformação, em coordenadas homogêneas, da forma

$$[u\ v\ 1]^t \simeq K[R\ \ T][X\ Y\ Z\ 1]^t, \tag{2.20}$$

onde $K = \begin{bmatrix} \alpha & c & u_c \\ 0 & \beta & v_c \\ 0 & 0 & 1 \end{bmatrix}$.

A restrição desta transformação ao plano $Z = 0$ define uma transformação projetiva (ou uma *homografia*) H entre este plano e o plano da imagem, cuja expressão em coordenadas homogêneas é $[u\ v\ 1]^t \simeq K[r_1\ r_2\ T][X\ Y\ 1]^t$, onde r_1 e r_2 são as duas primeiras colunas de R.

A seguir, descrevemos as diversas etapas do método de Zhang:

estimação da homografia H para cada imagem, estimação dos parâmetros intrínsecos e extrínsecos, estimação dos coeficientes de distorção, e refinamento dos parâmetros determinados.

Primeira etapa: *Estimação das homografias.*

As correspondências entre os pontos do padrão p_i $(i = 1 \ldots n)$ e suas posições q_i na imagem permitem formular um problema de otimização não-linear para a determinação da homografia que os associa: $\min_{||H||=1} \sum_i^n ||q_i - H(p_i)||^2$. Este problema pode ser resolvido por um método iterativo, como o de Levenberg-Marquardt.

Para obter uma solução inicial pode-se proceder do seguinte modo. Cada par de pontos correspondentes $p_i = [x_i\ y_i\ 1]^t$ e $q_i = [u_i\ v_i\ 1]^t$ determina uma equação envolvendo as linhas da matriz que representa a homografia H que os associa. Denotando estas linhas por $\bar{h}_1^t$, $\bar{h}_2^t$ e $\bar{h}_3^t$, temos:

$$\begin{aligned} \bar{h}_1^t p_i - u_i \bar{h}_3^t = 0, \\ \bar{h}_2^t p_i - v_i \bar{h}_3^t = 0. \end{aligned}$$

Escrevendo estas equações para todos os pontos p_i, obtemos um sistema da forma $Ah = 0$, onde h é o vetor 9×1 formado pelos elementos de H. Uma boa solução inicial para o algoritmo iterativo de minimização pode ser encontrada obtendo h, com $||h|| = 1$, tal que $||Ah||$ seja mínima. Tal h é o vetor singular correspondente ao menor valor singular de A (ou, equivalentemente, o autovetor correspondente ao menor autovalor de A^tA) (ver [32] para detalhes).

Segunda etapa: *Estimação dos parâmetros intrínsecos e extrínsecos.*

Como $H = \lambda K[r_1\ r_2\ T]$ e os vetores r_1 e r_2 são unitários e ortogonais um ao outro, temos:

$$\lambda^2 H^t K^{-T} K^{-1} H = [r_1\ r_2\ T]^t [r_1\ r_2\ T] = \begin{bmatrix} r_1^t r_1^t & r_1^t r_2 & r_1^t T \\ r_2^t r_1^t & r_2^t r_2 & r_2^t T \\ T^t r_1 & T^t r_2 & T^t T \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & r_1^t T \\ 0 & 1 & r_2^t T \\ T^t r_1 & T^t r_2 & T^t T \end{bmatrix}.$$

Em consequência, temos:

$$\begin{aligned} &h_1^t K^{-T} K^{-1} h_2 = 0, \\ &h_1^t K^{-T} K^{-1} h_1 = h_2^t K^{-T} K^{-1} h_2. \end{aligned} \tag{2.21}$$

Assim, para cada imagem, obtemos duas equações lineares envolvendo os elementos da matriz simétrica

$$K^{-T} K^{-1} = B = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix}$$
$$= \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{c}{\alpha^2\beta} & \frac{cv_c - u_c\beta}{\alpha^2\beta} \\ -\frac{c}{\alpha^2\beta} & \frac{c^2}{\alpha^2\beta^2} + \frac{1}{\beta^2} & -\frac{c(cv_c - u_c\beta)}{\alpha^2\beta^2} - \frac{v_c}{\beta^2} \\ \frac{cv_c - u_c\beta}{\alpha^2\beta} & -\frac{c(cv_c - u_c\beta)}{\alpha^2\beta^2} - \frac{v_c}{\beta^2} & \frac{(cv_c - u_c\beta)^2}{\alpha^2\beta^2} + \frac{v_c^2}{\beta^2} + 1 \end{bmatrix}.$$

Tomando $b = [B_{11}\ B_{12}\ B_{22}\ B_{13}\ B_{23}\ B_{33}]^t$ e definindo $v_{ij} = [h_{1i}h_{1j},\ h_{1i}h_{2j} + h_{2i}h_{1j},\ h_{2i}h_{2j},\ h_{3i}h_{1j} + h_{1i}h_{3j},\ h_{3i}h_{2j} + h_{2i}h_{3j},\ h_{3i}h_{3j}]$, temos $h_i^t B h_j = v_{ij} b$.

Assim, as equações em (2.21) podem ser escritas na forma:

$$\begin{bmatrix} v_{12} \\ (v_{11} - v_{22}) \end{bmatrix} b = 0. \tag{2.22}$$

Para m imagens, obtemos um sistema de $2m$ equações representadas por

$$Vb = 0, \tag{2.23}$$

onde V é uma matriz $2m \times 6$.

Em geral, o sistema em (2.23) não tem solução exata. Uma solução adequada é escolher b como o vetor com $||b|| = 1$ tal que $||Vb||$ é mínimo. Como no caso da estimação da homografia, isto corresponde ao vetor singular de V associado ao menor valor singular ou, equivalentemente, ao autovetor associado ao menor autovalor de V^tV.

A partir daí, podemos obter os elementos de K, além do valor de λ, através das equações:

$$\begin{aligned} v_c &= (B_{12}B_{13} - B_{11}B_{23})/(B_{11}B_{22} - B_{12}^2), \\ \lambda &= B_{33} - [B_{13}^2 + v_c(B_{12}B_{13} - B_{11}B_{23})]/B_{11}, \\ \alpha &= \sqrt{\lambda/B_{11}}, \\ \beta &= \sqrt{\lambda B_{11}/(B_{11}B_{22} - B_{12}^2)}, \\ c &= -B_{12}\alpha^2\beta/\lambda, \\ u_c &= cv_c/\alpha - B_{13}\alpha^2/\lambda. \end{aligned}$$

Conhecido K, calculam-se os parâmetros extrînsecos (R_i e T_i), para cada imagem i, por meio das equações:

$$\begin{aligned} r_1 &= \lambda K^{-1}h_1, \\ r_2 &= \lambda K^{-1}h_2, \\ r_3 &= r_1 \times r_2, \\ T &= \lambda A^{-1}h_3. \end{aligned}$$

Terceira atapa: *Estimação dos coeficientes de distorção.*

Zhang considera os coeficentes k_1 e k_2 de distorção radial, definidos pela seguinte relação entre pontos com distorção (x, y) e sem distorção (x', y') da imagem:

$$\begin{aligned} x &= x' + x'(k_1r^2 + k_2r^4), \\ y &= y' + y'(k_1r^2 + k_2r^4). \end{aligned}$$

onde $r = \sqrt{x'^2 + y'^2}$.

As coordenadas (x, y) se relacionam com os pontos observados (u, v) (em pixels) da imagem através das equações:

$$\begin{aligned} u &= \alpha x + cy + u_c, \\ v &= \beta y + v_c. \end{aligned}$$

Assim, as relações entre os pontos observados e os pontos ideais (sem distorção) são dadas por:

$$\begin{array}{rcl} u & = & u' + (u' - u_c)(k_1 r^2 + k_2 r^4), \\ v & = & v' + (v' - v_c)(k_1 r^2 + k_2 r^4); \end{array} \tag{2.24}$$

que pode ser re-escrita matricialmente como:

$$\left[\begin{array}{cc} (u' - u_c)r^2 & (u' - u_c)r^4 \\ (v' - v_c)r^2 & (v' - v_c)r^4 \end{array} \right] \left[\begin{array}{c} k_1 \\ k_2 \end{array} \right] = \left[\begin{array}{c} u - u' \\ v - v' \end{array} \right].$$

Para n pontos de controle em cada uma das m imagens, tem-se um sistema de $2mn$ equações, da forma

$$DK = d, \quad \text{com} \quad K = [k_1,\ k_2]^t. \tag{2.25}$$

Essa equação deve ser resolvida através de mínimos quadrados, e sua solução é dada por:

$$K = (D^t D)^{-1} D^t d. \tag{2.26}$$

Quarta etapa: *Refinamento dos parâmetros.*

Uma vez tendo-se uma estimativa de todos os parâmetros, pode-se utilizá-la como solução inicial de um método iterativo (como o de Levenberg-Marquardt) para minimizar uma medida não-linear de erro. Por exemplo, pode-se minimizar:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} \| p_{ij} - \hat{p}(K, k_1, k_2, R_i, T_i, P_j) \|^2,$$

onde p_{ij} é a projeção observada do ponto P_j na imagem i e $\hat{p}(K, k_1, k_2, R_i, T_i, P_j)$ é a projeção do mesmo ponto obtida de acordo com a expressão (2.20) e pela aplicação da distorção radial de acordo com a equação (2.24).

2.6 Calibração Conjunta

Como vimos na Seção 2.1, para que se possa recuperar a posição tridimensional de um ponto em uma cena, é necessário observá-lo em

duas imagens capturadas por câmeras posicionadas em pontos distintos ou, alternativamente, iluminar a cena com um padrão produzido por um projetor e observar a cena assim iluminada por uma câmera. As duas câmeras ou o par câmera-projetor devem ser calibrados em relação ao mesmo sistema de referência do mundo, para que se possa correlacionar as posições de pontos correspondentes na imagem com sua localização no espaço.

2.6.1 Rotação e translação relativas

Na maior parte dos casos, não há particular interesse em usar o sistema de referência utilizado na calibração. Na verdade, é mais conveniente utilizar o sistema de referência de uma das câmeras. Isto equivale a representar as coordenadas de um ponto reconstruído como (x, y, z), onde x e y fornecem sua posição na imagem e z é a sua profundidade em relação à câmera. O resultado é o que chamamos uma *imagem com profundidade* (ou *range image*).

Suponhamos, assim, que as duas câmeras, cujos sistemas de coordenadas denotaremos por SCC1 e SCC2 tenham parâmetros extrínsecos dados por $[R_1\ T_1]$ e $[R_2\ T_2]$, respectivamente, em relação a um dado SCM utilizado para a calibração. Em relação ao sistema de coordenadas da primeira câmera, os parâmetros extrínsecos da primeira câmera são $[I\ 0]$, já que neste caso o SCM e o SCC1 coincidem. Para obter os da segunda câmera, precisamos obter as coordenadas, no SCC2, de um ponto expresso no SCC1 (ou seja, precisamos determinar a matriz de rotação e translação de um sistema relativamente ao outro). Para tal, basta efetuar a mudança do SCC1 para o SCM e, a seguir, do SCM para o SCC2. A primeira mudança, em coordenadas homogêneas, é feita através da multiplicação por $\begin{bmatrix} R_1^t & -R_1^t T_1 \\ 0 & 1 \end{bmatrix}$, e a segunda através da multiplicação por $\begin{bmatrix} R_2 & T_2 \\ 0 & 1 \end{bmatrix}$.

A transformação procurada é dada pelo produto das transformações acima e é igual a $\begin{bmatrix} R_2 R_1^t & -R_2 R_1^t T_1 + T_2 \\ 0 & 1 \end{bmatrix}$. Portanto, os parâmetros intrínsecos da segunda câmera com respeito ao SCC1 são dados por $R = R_2 R_1^t$ e $T = -R_2 R_1^t T_1 + T_2$. Ou seja, as co-

ordenadas P_1 e P_2 de um ponto do espaço, com relação ao SCC1 e SCC2, respectivamente, satisfazem as relações $P_2 = RP_1 + T$ e $P_1 = R^tP_2 - R^tT$.

2.6.2 Calibração de projetores

A calibração de qualquer número de câmeras em relação a um mesmo sistema de coordenadas é simples. Basta capturar imagens, em cada câmera, de um padrão de calibração que é mantido estacionário. No caso de um par câmera-projetor, algumas adaptações são necessárias, já que o projetor projeta uma imagem no mundo tridimensional, ao invés de capturar uma imagem do mundo tridimensional. Assim, a idéia básica é projetar uma imagem conhecida e identificar as coordenadas dos pontos da cena em que determinados pontos desta imagem se projetam.

Uma forma conveniente de obter essas coordenadas consiste em usar a câmera que já foi calibrada. Por exemplo, pode-se projetar a imagem sobre o mesmo plano do padrão bidimensional usado para a calibração da câmera e capturar-se uma fotografia da cena assim iluminada.

A Figura 2.14 mostra duas imagens capturadas pela mesma câmera. A primeira é uma imagem de um padrão de calibração quadriculado, formada por quadrados medindo 6 cm por 6 cm. A segunda é uma imagem obtida projetando-se um padrão conhecido (uma imagem formada por quadrados de 25 por 25 pixels) sobre o verso do padrão de calibração (portanto, projetando-se no mesmo plano do espaço).

Para obter as coordenadas no SCM de um ponto (u, v) da segunda imagem, que esteja sobre o plano do basta usar a transformação inversa de câmera, com a informação adicional de que o ponto está no plano $Z = 0$. É mais conveniente, no entanto, expressar estas coordenadas diretamente no sistema da câmera (deste modo, os parâmetros extrínsecos do projetor já estarão dados neste sistema, sem a necessidade de executar o passo descrito na Subseção 2.6.1). No sistema da câmera, os pontos que se projetam sobre (u, v) estão sobre a reta $\{\lambda(u, v, f)|\lambda \in R\}$. As coordenadas destes pontos no SCM são dadas por $\lambda R^t[u\ v\ f]^t - R^tT$. O ponto desejado é tal que sua coordenada Z no SCM é igual a zero. Portanto, o valor de λ é dado por

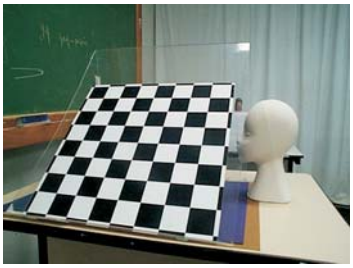



Figura 2.14: Calibrando câmera e projetor.

$\lambda = \frac{<r_3,T>}{<r_3,(u,v,f)>}$, onde r_3 é a terceira coluna da matriz R de rotação da câmera e T é seu vetor de translação.

Uma vez obtidas as correspondências entre pontos da imagem projetadas (expressos em pixels) com pontos do espaço (expressos no sistema da câmera) o projetor pode ser calibrado pelos métodos vistos anteriormente. É importante observar, no entanto, que diferentemente do que ocorre com câmeras, normalmente a projeção do centro óptico de um projetor não está no centro da imagem. Para diminuir a distorção quando o projetor é colocado sobre uma mesa ou no teto, o centro de projeção está próximo a uma das bordas. Para uma calibração adequada é necessário obter esta informação do fabricante.

Capítulo 3

Reconstrução no Espaço da Imagem

Nesse capítulo vamos abordar os métodos de fotografia 3D baseados em reconstrução no espaço da imagem. Como dissemos no Capítulo 1, esses métodos realizam a maior parte das operações no sistema de coordenadas local do dispositivo de captura e sómente no final do processo é que os dados são levados para o sistema de coordenadas global.

O processo consiste básicamente de três etapas:

1. Aquisição de Retalhos da Superfície do Objeto

2. Alinhamento dos Retalhos

3. Construção da Superfície

A etapa de aquisição dos retalhos resulta em imagens com informações geometricas e fotometricas do objeto ("range images"). Nessa etapa é feito o calculo da distancia de pontos visiveis do objeto em relação ao dispositivo de captura. A etapa de alinhamento correlaciona os vários retalhos capturados de pontos de vista diferentes que cobrem todo o objeto. Finalmente, os retalhos alinhados são integrados para formar uma única superficie.

A etapa 1 é realizada no sistema de coordenadas local de cada retalho. A etapa 2 faz a passagem dos sistemas locais dos retalhos para o sistema global da cena. A etapa 3 é realizada no sistema de coordenadas global. Essas etapas serão discutidas nas seções seguintes.

3.1 Estereoscopia e Triangulação

O princípio básico que permite a recuperação de informação de profundidade a partir de imagens é o princípio de visão estéreo. A visão estéreo é baseada no fato de que se o mesmo ponto X de uma cena for observado por dois pontos de vista, é possível reprojetar dois raios a partir da imagem na cena que se interseptam no ponto aonde a luz foi refletida pela cena. Esse processo é chamado *triangulação*.

Os métodos de estéreo passivo não interagem com os objetos da cena, o sensor utilizado afere medições do objeto em suas condições naturais. Os métodos ativos enviam algum tipo de sinal a ser refletido pelo objeto e então medido pelo sensor, o tipo de sinal emitido está completamente relacionado ao tipo de sensor a ser utilizado para medição.

Algumas das principais limitações dos métodos de aquisição baseados em sensores óticos é o fato de medirem apenas as porções visíveis, do ponto de vista da câmera, do objeto e serem sensíveis à propriedades fotométricas como brilho, transparência, regiões de reflectância muito baixa, interreflexões, etc.

No nosso caso, estamos interessados em objetos com dimensões comparáveis às dimensões de um vaso e que podem ter propriedades fotométricas bastante variáveis, pretendemos adquirir os dados de um objeto por vez, em um ambiente controlado, o que significa que podemos controlar a luz incidente ao objeto e o pano de fundo do cenário.

Um problema difícil de ser resolvido no caso da utilização de estéreo passivo é o problema de corresponder automaticamente os pontos da cena para calcular a triangulação. Para evitar esse problema podemos substituir uma das câmeras do estéreo passivo por um emissor de sinal, no caso uma fonte de luz conhecida e calibrada, que marca a cena com algum padrão conhecido.

Focaremos nossa atenção no estudo dos métodos de estéreo ativo,

nos quais um tipo especial de iluminação é direcionado para o objeto, substituindo o papel da segunda câmera em métodos de estéreo passivo, juntamente com uma câmera que faz o papel de sensor do sinal refletido. A aborbagem com métodos ativos, em geral, facilita a solução do problema de correspondência entre pontos da imagem na abordagem estéreo.

Na seção 3.2, vamos revisar os métodos de luz estruturada codificada, que consistem em iluminar o objeto a ser adquirido com slides subsequêntes que produzem um padrão codificado. Nessa seção vamos supor que um método de codificação foi utilizado e que esse método fornece a correspondencia entre pontos nos sistemas de coordenadas da camera e do projetor.

A idéia que dá origem ao conceito de luz estruturada é o de marcar a cena com algum padrão conhecido. O padrão mais simples possível é um ponto, ponteiros de laser foram utilizados para essa finalidade: um ponto de laser é projetado na cena e a imagem adquirida é analisada para detectar o ponto de laser e então calcular a triangulação. Uma evolução simples desse conceito é o de, ao invés de projetar um ponto, projetar uma linha na imagem, a triangulação então é feita intersectando pontos da imagem com o plano projetado, esse esquema é ilustrado na Figura 3.1. Esse princípio é o mais utilizado nas técnicas de estéreo ativo e, por isso, daremos mais ênfase a ele.

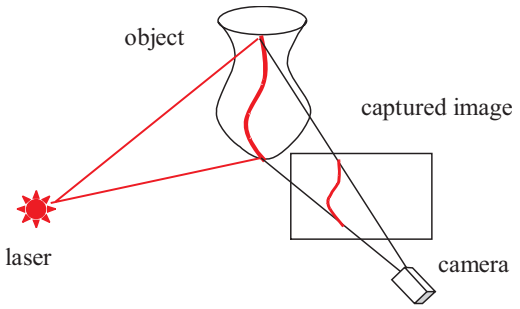


Figura 3.1: Projeção de plano de luz no objeto para triangulação

O pricípio da triangulação para a reconstrução da posição de um

ponto no espaço tridimensional a partir de sua projeção em dois planos de imagem conhecidos é bastante simples. Considere um ponto P no espaço com coordenadas $P_a = (X_a, Y_a, Z_a)$ e $P_b = (X_b, Y_b, Z_b)$, respectivamente em relação aos sistemas de referência de duas cameras A e B (ou camera e projetor). Supondo que as cameras estão calibradas, conhecemos os seus parâmetros intrinsecos, extrisecos e também a transformação relativa entre elas. Sejam, R a matrix de rotação e T o vetor de translação, que mapeiam o sistema de coordenadas da camera B na camera A. Então:

$$P_a = RP_b + T \tag{3.1}$$

Temos ainda que p_a e p_b são as projeções de P expressas em coordenadas homogêneas, nos planos de imagem das cameras A e B, respectivamente, tal que $p_a = P_a/Z_a = (\bar{x}_a, \bar{y}_a, 1)^t$ e $p_b = P_b/Z_b = (\bar{x}_b, \bar{y}_b, 1)^t$ (ver Figura 3.2-dir).

Com esses dados, podemos calcular a profundidade relativa de P nos sistemas de coordenadas das cameras A e B, resp. Z_a e Z_b.

Reescrevendo eq. 3.1 temos:

$$Z_a p_a = Z_b R p_b + T$$

ou

$$\begin{bmatrix} -Rp_b & p_a \end{bmatrix} \begin{bmatrix} Z_b \\ Z_a \end{bmatrix} = T$$

Usando minimos quadrados obtemos:

$$\begin{bmatrix} Z_b \\ Z_a \end{bmatrix} = (M^t M)^{-1} M^t T$$

com $M = \begin{bmatrix} -Rp_b & p_a \end{bmatrix}$, (uma matrix 2×3).

A expressão explicita para z_b fica então:

$$Z_b = \frac{||p_a||^2 < w, T > - < w, p_a >< p_a, T >}{||w||^2 ||p_a||^2 - < w, p_a >^2} \tag{3.2}$$

onde $w = -Rp_b$.

Essa solução é equivalente a encontrar o ponto P que minimiza simultaneamente a distancia aos dois raios de visada originados nos centros de projeção de cada camera, O_a, O_b e passando pelas projeções p_a e p_b.

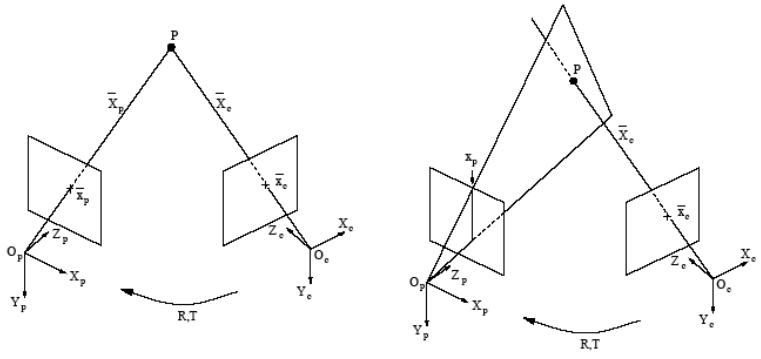


Figura 3.2: Triangulação

No caso de interesse, para triangulação por luz estruturada, uma das cameras é substituida por um projetor e o padrão básico de correspondencia é constituido por faixas verticais. Nesse caso, o problema de triangulação deve ser resolvido calculando-se a interseção de um raio partindo da camera com o plano gerado pela faixa de interesse (ver Figura 3.2-esq). A solução do problema estéreo para dois pontos pode ser utilizada nesse caso, bastando para isso considerar a coordenada $\bar{x}$ constante para cada faixa do projetor.

Note que na discussão acima, estamos supondo que os pontos no sistema da camera e do projetor estão em coordenadas normalizadas. Isso significa, em particular, que coordenadas em pixel nas imagens foram devidamente transformadas usando os parametros intrinsecos da camera, e que foi aplicada a correção da distorção radial.

3.2 Métodos de Codificação por Luz Estruturada

Existem várias maneiras de codificar os padrões de luz estruturada. Os primeiros métodos de codificação foram propostos na década de 80 [36], na década de 90 os padrões de codificação propostos foram aprimorados e análises de precisão e qualidade foram feitos.

Para amostrar toda a superfície de um objeto e não apenas uma linha, temos que adquirir várias imagens projetando o laser de modo a varrer a superfície. Um passo a mais nesse conceito seria o de projetar várias linhas ao mesmo tempo, para que o sistema de correspondência continue funcionando temos que saber diferenciar as linhas projetadas, daí surge o conceito de codificação de luz estruturada. Na figura 3.3 um padrão contendo várias faixas é projetado na cena, os planos aonde podemos recuperar a geometria com maior precisão é na transição dessas faixas.

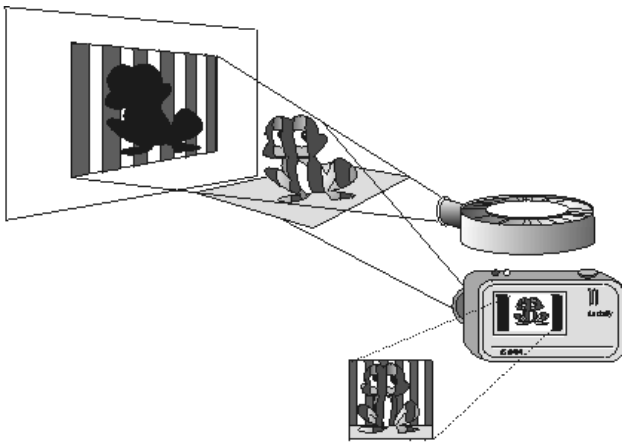


Figura 3.3: Padrão de faixas projetado na superfície.

Em resumo, as etapas básicas para construção de um sistema de aquisição de objetos baseado na técnica de luz estruturada é:

- Calibração de câmera e projetor.
- Estabelecimento de correspondências entre pontos da imagem adquirida e do padrão projetado.
- Reconstrução das coordenadas tridimensionais através de triangulação.

A seguir vamos nos concentrar no problema de correspondência câmera projetor.

3.2.1 Codificação de luz

Nos anos 80 testes pioneiros em codificação de luz estruturada foram feitos e os principais conceitos foram concebidos. Vamos fazer uma revisão dos principais conceitos propostos, uma descrição mais detalhada de cada código pode ser encontrada em [36].

Codificação no tempo

Uma idéia para codificar a posição de um padrão no projetor é a de modular a informação no tempo. A codificação em base binária é bastante usada nos sitemas de luz estruturada. Na Figura 3.4 ilustramos um código binário sendo representado por faixas pretas e brancas. A projeção de uma sequência de n slides produz 2^n faixas codificadas, e a resolução (número de faixas) de aquisição aumenta com o aumento do número de slides projetados.

Para decodificar a posição de um pixel do projetor, temos que identificar a intensidade da luz projetada em cada canal de cor. Áreas de sombra devem ser detectadas para serem excluídas. Na prática, um sinal binário é transmitido através da intensidade da luz (analogicamente), e sofre interferência da superfície do objeto, para então ser capturado pela câmera e transformado em sinal digital novamente.

Em 1982 [37] propos uma codificação temporal binária, e em 1984 [59] propos a substituição da binária por uma codificação mais robusta baseada no código binário de Gray ilustrado na Figura 3.4. Esse código foi revisitado, reimplementado, refinado e analisado muitas vezes, e ainda é usado como uma técnica de aquisiçào de geometria robusta para cenas estáticas. O principal problema da codificação

binária temporal é o grande número de slides que deve ser projetado/adquirido para atingir a resolução desejada, além disso, seu uso é restrito a cenas estáticas.

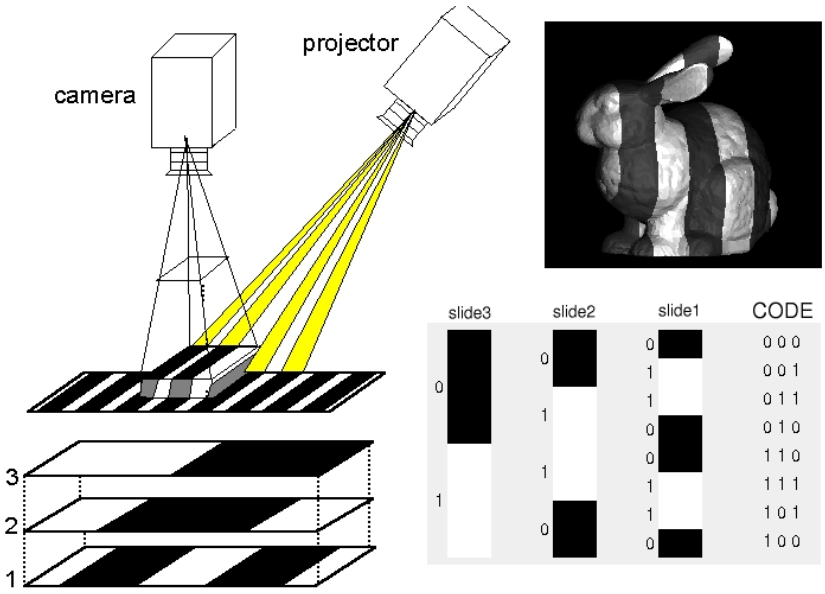
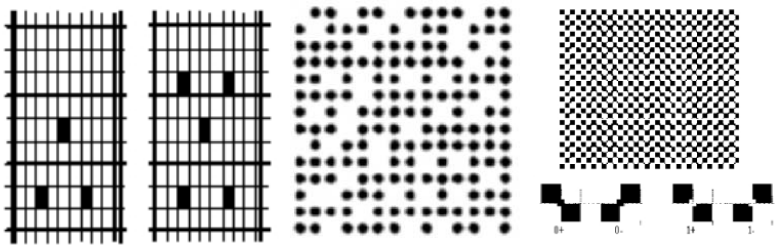


Figura 3.4: Codificação temporal (Gray Code)

Codificação espacial

O desejo de adquirir cenas não estáticas e de reduzir o numero de imagens a ser adquirida motivou a busca por codificações em um único slide. A única maneira de codificar a posição em um único slide é aumentando a base da codificação, isto é, aumentando a variedade de símbolos que são transmitidos de modo que hajam símbolos suficientes para atingir a resolução desejada. Uma maneira simples de fazer isso é considerar a vizinhança de um determinado pixel para executar a codificação, chamada de codificação espacial, ou modular a intensidade da luz projetada como função da posição do projetor (a ser discutido mais adiante).



(a) grid pattern (b) M-array code (c) 2×3 pixels code

Figura 3.5: Exemplos de codificação espacial (fonte: [36]).

Na Figura 3.5 mostramos alguns esquemas de codificação espacial. O maior desafio nesse tipo de estratégia é o de reconhecer o padrão projetado que foi deformado pela superfície do objeto. Na Figura 3.5 são mostradas três codificações representativas desse tipo de abordagem: na Figura 3.5(a)um reticulado retangular serve de base para uma codificação parcial da posição do reticulado [54]; na Figura 3.5(b) a codificação é feita por por linhas em vetores de tamanho constante [33] e na Figura 3.5(c) a região de codificação é uma janela de 2×3 pixels é usada com um alfabeto de 4 padrões diferentes proposta em [68], uma dificuldade em usar esse tipo de codificação está no fato de que o objeto frequentemente deforma o padrão projetado, e o seu reconhecimento pode ser prejudicado.

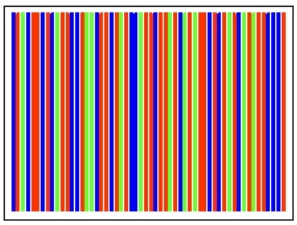
Codificação espacial impõe restrições de suavidade em uma vizinhança da região projetada, em alguns casos a restrição pode até ser global, ou seja, só são permitidos objetos suaves.

Introduzindo o uso de cores

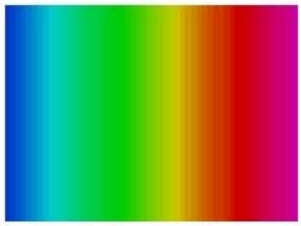
Para codificar a posição de um ponto da imagem projetada em um único slide sem recorrer à informação de vizinhança (espacial ou temporal) temos que modular a intensidade da luz e sermos capazes de recuperar as nuances da iluminação projetada, essa abordagem é bastante sensível a ruídos e para ser usada deve supor que as propriedades

refletivas dos objetos sejam bastante bem comportadas - idealmente objetos monocromáticos e lambertianos.

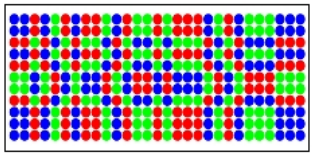
O uso de cores em codificação foi possibilitado no final dos anos 80 graças ao avanço tecnológico na captura de imagens coloridas. O principal ganho é o de poder usar os 3 canais para codificação ao invés de um só. No entanto, luz colorida tem um comportamento singular em cenas com objetos coloridos, desse modo, a robustez da decodificação só é conseguida ao restringir o uso de luzes coloridas em objetos de cor neutra.



(a) horizontal codeword



(b) wavelength modulation



(c) cardeal neighbors

Figura 3.6: Examples of color based codes (from [36]).

Codificações baseadas em cor são mostradas na Figura 3.6. Faixas verticais codificadas considerando a cor da faixa vizinha proposta em [40] é mostrada na Figura3.6(a); na Figura3.6(b)mostramos um

padrão arco-íris em que as cores são moduladas suavemente no espaço [38]; na Figura 3.6(c) mostramos um acodificação espacial que utiliza cores.

Nos anos 90 os grandes avanços em software e hardware permitiram que esses padrões fossem mais explorados e que suas limitações fossem sendo melhor estudadas, fundamentações teóricas e comparações mais precisas foram feitas. Novas codificações foram propostas e codificações já propostas foram re-implementações, alguns trabalhos representativos são [36, 55, 61, 89].

Observando aspectos mais teóricos, em [64] é feito um estudo para determinar o maior tamanho possível de uma matriz codificada com padrão de pontos, uma matriz grande é desejável para aumentar a resolução fornecida pelo padrão. Em [92, 91] o autor atenta para o fato de que apesar de vários códigos terem sido propostos o problema de decodificação não tinha sido adequadamente explorado, e propõe um algoritmo de decodificação eficiente para ser aplicado em códigos gerados a partir de grafos. Em 1999, o artigo [24] propõe um critério para buscar o design ótimo de padrões de luz estruturada explorando a semelhança entre a projeção de padrões e a transmissão de sinais em sistemas de comunicação.

Taxonomia

Como foi visto nas seções anteriores existe uma grande variedade de métodos para codificação de luz estruturada, uma classificação sistemática desses métodos é de grande utilidade para entendê-los. A seguir apresentamos uma tebela com a classificação usual dos métodos considerando as diferenças entre os padrões de codificação e as restrições impostas sobre as cenas a serem capturadas. A modulação da intensidade projetada impõe restrições sobre as propriedades fotométricas do objeto, codificação que explora vizinhança espacial impõe restrições de suavidade do objeto enquanto codificação temporal restringe o movimento da cena.

Assumptions	Methods	Restrictions
coerência espacial	métodos baseados em vizinhança	discontinuidades na superfície dos objetos não são permitidas
coerência temporal	vários slides vs. um slide	restrito a cenas estáticas vs. permite cenas dinâmicas.
reflectividade	binario/tons de cinza vs. codificação com cor	não restringe a cor dos objetos vs. objetos com cores neutras.

Como já foi observado anteriormente, existe uma analogia natural entre a projeção de padrões codificados em uma cena e sistemas digitais de comunicação. A cena se comporta como o canal de transmissão do sinal, e em cada pixel da câmera recebemos uma informação emitida pelo projetor que sofreu uma degradação de transmissão causado principalmente no meomento de reflexão na superfície de um objeto da cena. Considerando essa analogia podemos estudar dois principais problemas:

1. Limitações do canal de transmissão, relacionado às propriedades de reflectância das superfícies.

2. Como codificar os pixels do projetor, que vão restringir a classe de objetos a ser scaneada.

A seguir vamos apresentar uma tabela que classifica os métodos através de conceitos usados no contexto de codificação e decodificação de sinais para transmissão em canais sujeitos a ruído. Considera-se o tamanho do alfabeto que é o conjunto de símbolos básicos a serem utilizado na comunicação e o tamanho e a estrutura das palavras, isto é, como é a estrutura de concatenação dos símbolos básicos.

método	num (code size)	intensidade modulação/canal (no. of characters)	vizinhança (character's region)	resolução (alphabet size)
código Gray (Fig.3.4)	n	bin'rio(2) monocromático	um pixel	2^n por linha
Gray colorido	n	binary(2)/ RGB	single pixel	$2^{3\times n}$ por linha
padrão arco-íris (Fig.3.6(b))	2	2^8/ RGB	um pixel	$(2^8)^3$ por linha
dot matrix (Fig.3.6(c))	1	3 (R, G or B)		3^5
janela de codificação Fig.3.5(c)	1	binário	4 pixels 2×3 janela	4^6

Um avanço recente em termos de codificação foi proposto em [80] aonde se propõe a codificação da transição das faixas projetadas ao invés de codificar as faixas como é feito tradicionalmente, isso permite a redução da quantidade de slides a serem projetados por aumentar a base de codificação.

Em [48] uma técnica de programação dinâmica é utilizada para otimizar o processo de aquisição. Sistemas completos de aquisição são apresentados em [80] nesses trabalhos são encarados diferentes desafios como o de adquirir objetos grandes, cenas em movimento, aquisição em tempo-real e aquisição de textura.

3.3 Alinhamento de Retalhos

Os dispositivos de digitalização 3D fornecem a geometria dos objetos na forma de retalhos da superfície, capturados a partir de um único ponto de vista.

Em geral, para objetos complexos, é necessario realizar a captura de retalhos a partir de vários pontos de vista de forma a cobrir toda a superfície do objeto.

Nesses retalhos a geometria está descrita em relação ao sistema de coordenadas local do dispositivo. Temos então o problema de alinhar os retalhos em relação uns aos outros dentro de um sistema de coordenadas global.

Para realizar o alinhamento podem ser utilizadas informações do sistema de aquisição, correspondencia entre pontos caracteristicos do objeto, ou mesmo uma combinação desses dois elementos. Por exemplo, se o sistema de aquisição tem uma base giratória para posicionar o objeto em relação à camera, o ângulo de rotação pode ser usado o alinhamento. Tambem, a determinação de um conjunto de pontos marcantes do objeto, como quinas ou protuberancias, em dois retalhos distintos pode ser usada para calcular o movimento rígido que faz o casamento desses pontos (note que para ser possivel esse casamento, é necessário que os retalhos tenham uma área de inteseção na superfície do objeto).

Normalmente, esse problema é equacionado em duas etapas: na primeira etapa os retalhos são alinhados entre si, aos pares, e na segunda etapa o alinhamento de pares é melhorado considerando todos os retalhos.

3.3.1 Alinhamento aos Pares

Independentemente do método utilizado, seja ele baseado em dados do sistema ou casamento de pontos marcantes, o alinhamento inicial em geral não têm uma precisão satisfatória e precisa ser refinado usando uma outra técnica.

O método mais conhecido para alinhar pares de retalhos é baseado no algoritmo ICP ("Iterative Closest Point"), [8, 17], que iterativamente transforma dois conjuntos de pontos minimizando a distância entre eles.

O algoritmo ICP consiste de dois passos que são executados iterativamente até a convergencia. Primeiro, pares de pontos correspondentes nos dois retalhos são identificados. Depois um método de otimização computa o movimento rígido que reduz a distancia entre os dois conjuntos, no sentido de mínimos quadrados.

Em [8], Besl e McKay, provaram que esse algoritmo converge, não necessariamente para o minimo global, mas pelo menos para um mínimo global. Na prática, se os dois conjuntos estão proximos na

inicialização do algoritmo, a solução ótima é obtida.

O programa 1 descreve a estrutura desse algoritmo.

Programa 1 : icp(a, b)

repeat
find closest points in a, b
compute $T(a, b)$ such that dist(a, b) is minimal
until convergence

3.3.2 Alinhamento Global

Quando o alinhamento aos pares é utilizado sequencialmente para alinhar um conjunto de retalhos, os erros tendem a se acumular e o alinhamento global fica insatisfatório. Isso acontece, porque o alinhamento aos pares resolve separadamente o casamento de somente dois retalhos de cada vez.

O problema do alinhamento global é intratável do ponto de vista de otimização e por isso diferentes herísticas são empregadas para obter uma solução aceitável, mas não ótima em geral, [67, 75, 6].

Os algoritmos propostos na literatura, são baseados em três estratégias:

- Alinhamento incremental em relação a um retalho central;
- Casamento incremental de pontos em multiplos retalhos; e
- Relaxação pseudo-física por sistema mola-massa.

3.4 Integração da Superfície

O alinhamento dos retalhos forneçe uma cobertura do objeto na qual os dados estão referenciados em um sistema de coordenadas global. Entretanto, esses dados representam pedaços desconectados da superfície do objeto.

A integração dos retalhos tem a finalidade de reconstruir a geometria e topologia do objeto em uma representação única e coerente. Esse é um problema difícil por várias razões: em geral os dados

são ruidosos e podem conter erros; algumas partes do objeto podem não ter sido capturadas por problemas de visibilidade; e densidade de amostragem pode não ser suficiente para uma reconstrução fiel.

Os métodos de integração normalmente procuram explorar a estrutura dos retalhos para melhor estimar a superfície do objeto. Eles se dividem em duas categorias principais: métodos baseados em pontos; e métodos volumetricos.

Os métodos baseados em pontos reconstroem a superfície diretamente fazendo uma triangulação dos pontos para construir uma malha poligonal. Um algoritmo de triangulação bastante utilizado é o "ball-pivoting", que será descrito no Capitulo 5. Observamos que esse algoritmo assume uma densidade de amostragem uniforme, que está relacionada com a resolução do dispositivo de digitalização.

Os métodos volumétricos constroem uma descrição implicita do objeto, calculando a função de distancia com sinal da superfície do objeto. Essa descrição implicita, é dada de forma volumétrica na qual os valores da função de distancia são armazenados numa matriz tridimensional representando uma decomposição uniforme do espaço em torno do objeto. O cálculo da função de distância pode ser realizado fazendo a rastrização 3D dos retalhos na grade de voxels ou traçando raios a partir da posição do sensor aos retalhos, passando pelos voxels [22]. Uma vez obtida a representação volumetrica, técnicas de poligonização de superfícies implicitas, podem ser utilizados para construir a malha poligonal, como por exemplo o algoritmo de "marching cubes", que será descrito no Capitulo 5.

Capítulo 4

Reconstrução no Espaço da Cena

4.1 Introdução

Os métodos que serão apresentados neste capítulo se caracterizam basicamente por descrever a cena a ser reconstruída através de uma representação volumétrica. Normalmente, em Computação Gráfica, associamos o termo volumétrico a algum tipo de amostragem. Nesta exposição, adotaremos a interpretação de Slabaugh et al. [77], o qual considera como sendo volumétrica uma representação da ocupação espacial da cena sem considerar a existência de uma amostragem subjacente. Por este motivo podemos incluir nesta categoria os primeiros métodos de reconstrução de forma que descreviam a cena obtida através da interseção de volumes geométricos.

Agruparemos os métodos de reconstrução volumétrica de cenas em duas grandes categorias que se impuseram naturalmente à medida em que as técnicas foram sendo desenvolvidas:

- métodos baseados em silhuetas.
- métodos baseados em critérios de foto-consistência.

Uma excelente compilação das técnicas de reconstrução

volumétrica pode ser encontrado no trabalho de Slabaugh, Culbertson e Malzbender [77].

4.2 Reconstrução Baseada em Silhuetas

Os primeiros métodos de reconstrução volumétrica buscavam aproximar uma estrutura denominada Envoltória Visual(*Visual Hull*), a qual é definida como sendo a forma maximal que reproduz as silhuetas do objeto original quando renderizada a partir de todos os pontos de vista posicionados fora do seu fecho convexo. O conceito de Envoltória Visual foi estabelecido por Laurentini e é a base para a maioria das técnicas de reconstrução de cenas baseadas em silhuetas [45].

Na prática não é possível utilizar um conjunto contendo um número infinito de imagens, o que nos obriga a buscar uma aproximação da *Envoltória Visual* através da determinação do volume de ocupação da cena com base em um conjunto de restrições estabelecidas pela segmentação do objeto de estudo nas imagens. Em geral, isto é feito através de um processo de interseção de volumes capaz de determinar o que chamamos de *Envoltória Visual Inferida* (Figura 4.1). As imagens são segmentadas binariamente em regiões de fundo e regiões do objeto. As regiões contendo as projeções do objeto, juntamente com os centros de projeção associados, determinam volumes cônicos cuja interseção no espaço 3D determina uma aproximação para a envoltória visual.

As principais propriedades que relacionam o Fecho Visual Aproximado à forma original do objeto de estudo são as seguintes:

- é uma aproximação que engloba a forma real do objeto.
- o seu tamanho decresce à medida que mais imagens são consideradas.
- fornece uma aproximação para o fecho convexo da cena.

Os métodos apresentados a seguir se baseiam no processo de interseção volumétrica, variando-se apenas o tipo da estrutura de dados utilizada na representação dos volumes cônicos e os métodos utilizados no cálculo de interseções.

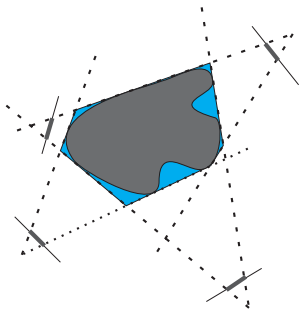


Figura 4.1: Envoltória visual inferida

4.2.1 Representação por Segmentos de Volume

Martin e Aggarwal [49] foram os primeiros a propor um método para reconstrução volumétrica de cenas a partir de silhuetas. Inicialmente, obtém-se as segmentações dos objetos de interesse nas imagens, sobre as quais é realizada uma análise de componentes conexas de forma a determinar as silhuetas.

Após a determinação das silhuetas constrói-se uma *estrutura de paralelogramos* no espaço 3D com base na combinação das projeções ortográficas das silhuetas de duas imagens arbitrariamente escolhidas.

Em um passo seguinte, constrói-se uma representação por *volumes segmentados* o qual é composto por um conjunto de segmentos de reta paralelos a um dos eixos de um sistema de coordenadas local escolhido adequadamente. A estrutura de dados utilizada para representar os volumes segmentados é organizada de forma hierárquica em relação a cada uma das componentes do sistema de coordenadas local adotado, de forma a facilitar o processo de interseção de segmentos.

Após esta etapa, a representação por volumes segmentados é refinada através de silhuetas provenientes de outros pontos de vista, permitindo assim que uma representação poliédrica adequada da superfície possa ser obtida.

O método de Martin e Aggarwal é essencialmente um método geométrico, cuja descrição se torna um pouco complexa sem o auxílio de figuras. Para maiores detalhes, encorajamos o leitor a verificar o

trabalho original.

4.2.2 Representação por Octrees

Chien [18, 19] foi o primeiro a sugerir a utilização de octrees para a representação do volume de ocupação de cenas reconstruídas por métodos baseados em silhuetas. O método por ele proposto consiste na utilização de três imagens binárias obtidas a partir de três câmeras paralelas ortogonais, as quais são convertidas em estruturas quadtree, e posteriormente recombinadas de forma a obter uma representação da envoltória visual através de uma octree. As principais limitações do método proposto por Chien são a restrição do número de imagens utilizadas, a restrição de ortogonalidade dos eixos óticos e a utilização exclusiva de projeção paralela.

Apesar de conter severas restrições, o método proposto por Chien foi de suma importância para o aprimoramento das técnicas de reconstrução volumétrica ao introduzir a utilização de estruturas espaciais adaptativas capazes de permitir que o processo de reconstrução fosse realizado com maior eficiência e simplicidade. Um aprimoramento das técnicas propostas por Chien pode ser encontrado no trabalho de Veenstra [85] que é capaz de gerar uma reconstrução do fecho visual de uma cena a partir de 13 pontos de vista ortográficos pré-definidos.

Potmesil [65] introduziu uma nova técnica capaz de reconstruir cenas através de imagens geradas por projeções perspectivas a partir de pontos de vista posicionados arbitrariamente no espaço. Em um primeiro momento, calcula-se uma representação através de uma estrutura *octree* de cada volume cônico determinado pelas silhuetas e seus respectivos centros de projeção. Em uma segunda etapa, calcula-se a interseção entre todos os cones definidos pelas *octrees*, gerando assim um modelo global sobre o qual é efetuado uma análise de componentes conexas 3D com o objetivo de rotular objetos individuais. Uma vez obtido o volume de ocupação da cena, associa-se um conjunto de vetores normais ao objeto, calculados com base na topologia da *octree*. Finalmente, efetua-se um processo de texturização, no qual um mecanismo de combinação é utilizado nas regiões da superfície do objeto que sofrem influência de mais de uma textura. Srivastava [79] propõe um trabalho similar no qual os contornos são aproximados por polígonos que, por sua vez, são decompostos em componentes con-

vexas de forma a facilitar os testes de interseção entre as projeções das células da octree e as regiões delimitadas pelas silhuetas.

Um grande passo no desenvolvimento das técnicas de reconstrução a partir de silhuetas foi dado por Richard Szeliski [82]. Szeliski aproveitou de maneira eficaz as propriedades das estruturas octree de forma a explorar intensamente a coerência existente entre formas obtidas em diversos níveis de resolução. Outras contribuições importantes deste trabalho foram a proposta de um método automático para segmentação do objeto através de subtração de imagens, além de um mecanismo bastante prático para determinação da orientação do sistema de captura, o qual era composto por um prato rotativo e uma câmera fixa.

Quando propôs este novo método, Szeliski tinha dois objetivos principiais a serem atingidos: processar cada imagem a partir do momento em que estivesse disponível e produzir rapidamente um modelo em baixa reso–
lução, o qual poderia ser refinado incrementalmente à medida em que novas imagens fossem capturadas pelo sistema. Um outro objetivo de Szeliski era investigar métodos altamente paralelizáveis.

Basicamente, o algoritmo constrói uma representação do volume de ocupação do objeto de interesse através de um procedimento de refinamento efetuado sobre uma *octree*, cujas células são rotuladas da seguinte forma:

- negra - célula que está totalmente contida no objeto
- branca - célula que está completamente fora do objeto
- cinza - célula ambígua.

Primeiramente, assume-se que a cena seja representada por um volume inicial o qual é definido por uma célula negra ou por um pequeno conjunto de células negras, determinando, por exemplo, um volume de 8x8x8 elementos. Para cada nova imagem adquirida, projeta-se todos os cubos associados às células negras(ativas) da *octree* na nova imagem e verifica-se se eles recaem completamente dentro ou completamente fora da silhueta. O novo rótulo da célula é determinado de acordo com a tabela 4.1.

cor antiga ⇒ resultado ⇓	negra	cinza	branca
dentro	negra	cinza	branca
ambíguo	cinza	cinza	branca
fora	branca	branca	branca

Tabela 4.1: Esquema de atualização dos rótulos da octree

Após um determinado nível de resolução ter sido processado, o que corresponde a uma revolução completa do objeto em relação à câmera fixa, refina-se as células cinzas subdividindo-as em oito novas células as quais passam a ser rotuladas como negras. As novas células negras assim obtidas são processadas exatamente como as do nível anterior.

4.2.3 Representação por Voxels

Uma alternativa à representação do espaço da cena através de estruturas espaciais pode ser obtida através de representações volumétricas regulares.

Massone [50] foi um dos primeiros a propor um método que determinava a envoltória visual a partir de um volume descrito por voxels. Uma das características mais notáveis deste método era sua capacidade de reconstruir cenas reais(não sintéticas) cujas imagens eram obtidas através de câmeras *Vidcon*. Este foi, na realidade, o primeiro método a lidar tanto com projeções paralelas quanto com projeções perspectivas.

Um pouco mais tarde, Fromherz [27] retoma a idéia de Massone e descreve um método, restrito a projeções ortográficas, que trabalha sobre um volume de voxels cujo tamanho das projeções são da escala dos pixels das imagens. Cada uma das silhuetas é obtida através de um processo de segmentação automática, assim como no método de Szeliski. Posteriormente tal método foi aprimorado através da introdução de um passo de refinamento do modelo com base em informações de luminância [28]. Em cada iteração, os voxels pertencentes à superfície do modelo inicial obtido são projetados em

pares de imagens seqüenciais em uma seqüência de rotação. Se as luminâncias dos pixels em cada uma das regiões de projeção de um voxel diferem por uma quantidade maior que um determinado valor constante, então o voxel é removido da cena. Pela primeira vez é proposto um mecanismo para descarte de voxels com base na verificação de algum tipo de consistência fotométrica.

A reconstrução volumétrica de cenas a partir de seqüências de vídeo foi introduzida por Moezzi [52], que desenvolveu um sistema de 17 câmeras centralizadas em uma cena dinâmica contida em uma região volumétrica de $1m \times 1m \times 2m$. Em uma etapa de pré-processamento, cada quadro é segmentado em objeto e fundo gerando-se, desta forma, uma seqüência de imagens binárias. Em seguida é efetuada a etapa de interseção dos volumes com o objetivo de reconstruir o envoltória visual e por último, aplica-se um passo adicional para extração da superfície poligonal do volume obtido, sobre a qual são aplicadas texturas provenientes das imagens de entrada. Em [53] foi proposta uma melhoria no procedimento de colorização.

4.3 Reconstrução através de Foto-Consistência

O surgimento dos métodos de reconstrução volumétrica baseados em foto-consistência foi uma conseqüência natural do desenvolvimento das técnicas de reconstrução baseadas em silhuetas. Podemos observar claramente que os trabalhos de Szeliski e principalmente Fromherz, com sua função de consistência baseada em luminância, tiveram importância fundamental no surgimento dos métodos baseados em escultura do espaço que hoje conhecemos.

O grande diferencial entre os métodos baseados em silhuetas e os métodos baseados em foto-consistência é a capacidade destes últimos em lidar verdadeiramente com o problema de estereoscopia. Os métodos baseados em silhuetas, na verdade, só são capazes de reconstruir objetos da cena cuja segmentação fundo/objeto for explícita, permitindo assim a determinação do espaço de ocupação da cena através da interseção de volumes. É óbvio que estes métodos não são capazes de reconstruir corretamente objetos descritos por superfícies com concavidades, já que estas informações não podem ser capturadas

por silhuetas.

Os métodos de reconstrução volumétrica baseados em foto-consistência não só são capazes de reconstruir cenas com geometrias arbitrárias como também são capazes de gerar um modelo completamente colorido sem necessidade de uma etapa de texturização adicional. Todo o processo é realizado através de consultas que verificam se um determinado elemento é capaz de explicar as cores nas regiões do conjunto de imagens em que se encontra visível.

Por outro lado, o problema de estereoscopia é um problema muito mais difícil, sendo bastante sensível a erros sistemáticos como erros de calibração, ruído, aliasing e etc. Contudo, a vantagem em relação aos métodos baseados somente em silhuetas é bastante grande, e o que é mais animador, nada impede que as duas técnicas sejam combinadas de forma a auxiliar o processo de reconstrução.

Na seção seguinte iremos descrever os métodos de *Coloração de Voxels* e *Escultura do espaço*, assim como suas diversas variações encontradas na literatura afim. Os conceitos de reconstrução com base em foto-consistência serão introduzidos de forma intuitiva através do método de coloração de voxels, enquanto que uma abordagem mais formal da teoria será descrita na seção que trata do método de Escultura do Espaço.

4.4 Coloração de Voxels

O método proposto por S. Seitz e C. Dyer [71, 72] trata o problema de reconstrução de uma cena a partir de fotos como um problema de coloração de voxels. Primeiramente, descreveremos o problema, introduzindo as notações necessárias. Em seguida avaliaremos cada um dos aspectos importantes associados ao problema e finalmente especificaremos o algoritmo capaz de solucioná-lo.

4.4.1 Definição do Problema

Primeiramente, considera-se que a cena tridimensional S, a ser reconstruída, esteja contida em um subconjunto fechado $U \subset \mathbb{R}^3$, o qual é representado através de um conjunto de voxels $\mathcal{V}$, onde cada voxel $v \in \mathcal{V}$ ocupa um volume homogêneo do espaço e possui uma única

cor. Identificamos S com sua representação discreta induzida por $\mathcal{V}$, com a exigência de que todo voxel $v \in S$ seja completamente opaco. Além disso, utilizamos a notação $cor(v, S)$ para representar a cor de um voxel v em uma cena S.

O conjunto de imagens é representado por $I = \{I_1, \cdots, I_n\}$. A cor de um pixel p em uma imagem I_i é representada pela notação $cor(p, I_i)$. De maneira análoga, $C = \{C_i, \cdots, C_n\}$ denota o conjunto de todas as câmeras a partir das quais o conjunto de imagens I foi obtido. Dado um pixel p de uma imagem I_i qualquer e uma cena S, chamamos de $S(p)$ um voxel de S que se projeta em p.

Definição 4.4.1 (Cena completa). *Uma cena S é completa em relação a um conjunto de imagens I quando, para toda imagem I_i e para todo pixel $p \in I_i$, existe um voxel v tal que $v = S(p)$[71].*

Definição 4.4.2 (Cena consistente). *Uma cena completa S é considerada consistente em relação a um conjunto de imagens I quando, para toda imagem I_i e para todo pixel $p \in I_i$, $cor(p, I_i) = cor(S(p), S)$[71].*

Definição 4.4.3 (Problema). *Seja $\mathcal{V}$ o conjunto de voxels que corresponde à discretização do espaço $U \subset \mathbb{R}^3$ no qual S está contida. Além disso, seja dada uma coleção de imagens I obtidas a partir de um conjunto de câmeras calibradas C cujos centros de projeção se encontram em pontos cp_i tais que $cp_i \in \mathbb{R}^3 - U$. O problema de coloração do conjunto de voxels $\mathcal{V}$ consiste em obter uma atribuição de cores a cada um dos elementos $v \in \mathcal{V}$ de tal forma que o conjunto $\mathcal{V}$, quando renderizado a partir de cada uma das câmeras C_i, reproduza as images I_i.*

Uma vez que o problema de coloração de voxels tenha sido especificado, devemos considerar as seguintes questões:

- Existe uma solução?
- A solução obtida é única?
- Como encontrá-la?

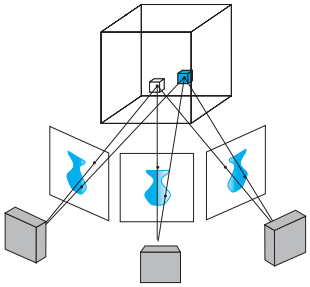


Figura 4.2: Coloração de voxels

4.4.2 Restrições

Iremos primeiramente analisar em que condições o problema definido acima admite uma solução. O fato de considerarmos que cada voxel possui uma única cor associada nos leva a restrições importantes sobre as propriedades físicas da cena. De fato, só conseguiremos apresentar soluções para cenas cujas projeções de seus pontos possuam uma mesma cor em todas as imagens em que forem visíveis. Esta igualdade é definida a menos de erros introduzidos pelo sistema de aquisição como, por exemplo, ruído, *aliasing* e erros de calibração, os quais não levaremos em consideração neste momento. Logo, a reconstrução fica limitada a superfícies denominadas *lambertianas*, nas quais cada ponto emite a mesma radiação luminosa em todas as direções.

Em segundo lugar, para que a forma obtida possa reproduzir as imagens de entrada quando visualizada, é necessário que a discretização do volume seja feita com uma resolução compatível com as mesmas, isto é, as regiões associadas à reprojeção de um voxel devem ser aproximadamente iguais a um pixel. Isto pode ser resolvido, de forma aparentemente simples, através de uma aumento arbitrário da resolução(mais tarde veremos que o problema não é tão simples assim). Mais uma vez, com o intuito de simplificar a análise, iremos considerar, assim como no trabalho clássico de Seitz[71], que a discretização é realizada com uma resolução grande o suficiente para que um voxel possa ser aproximado satisfatoriamente através de seu

centróide.

Supondo-se que a superfície da cena se comporte aproximadamente como uma superfície lambertiana, podemos perceber, de forma intuitiva, que sempre é possível apresentar uma solução, bastando para isso indicar o subconjunto de voxels de $\mathcal{V}$ cujas projeções, nas imagens em que são visíveis tenham uma mesma cor.

4.4.3 Determinação da Visibilidade

Uma questão importantíssima para a análise do método de coloração de voxels é a de como determinar a visibilidade dos voxels durante o de processo de reconstrução. No caso geral, a visibilidade de um voxel v em relação a um ponto de vista associado a uma câmera específica C_i depende diretamente da ausência de voxels opacos entre o segmento de reta que liga V ao centro ótico de C_i. Logo, a determinação da visibilidade envolve, em um caso mais genérico, algum procedimento capaz de determinar a interseção entre o raio de projeção e os elementos que compõem a cena parcial em uma dado instante do processo de coloração.

Uma forma de se evitar o tratamento da determinação da visibilidade por meios mais sofisticados é através do conhecimento de uma relação de ordem existente entre os voxels e o conjunto de câmeras consideradas.

Através de uma ordenação dos elementos de $\mathcal{V}$ em relação ao sistema de câmeras, podemos percorrer os voxels no sentido dos mais próximos às câmeras para os mais distantes. Assim, a visibilidade dos voxels mais próximos as câmeras sempre fica determinada antes da visibilidade dos voxels mais distantes. Com base neste artifício não corremos o risco de que a visibilidade de um voxel específico venha a ser modificada posteriormente por uma operação de remoção de voxels, o que invalidaria as decisões já tomadas.

Com o objetivo de facilitar o processo de determinação da visibilidade dos voxels e para permitir que o algoritmo possa ser efetuado em um único passo, Seitz [71] propõe que o processo de reconstrução de cenas através de coloração de voxels seja restrito a configurações de câmera que satisfaçam uma restrição de ordenação da visibilidade.

Definição 4.4.4 (Restrição de ordenação da visibilidade). *Existe uma norma $\| \bullet \|$ tal que, para todos os pares de pontos Q e Q' pertencentes à cena e para todas as imagens de entrada I, Q oculta Q', em relação ao conjunto I, somente se $\|Q\| \leq \|Q'\|$[71].*

Infelizmente, esta norma nem sempre existe, o que significa que uma ordenação global nem sempre pode ser determinada. No entanto, existem casos práticos em que a restrição de ordenação de visibilidade pode ser satisfeita. Seitz mostra em seu artigo que sempre é possível determinar uma ordenação global dos voxels em relação ao sistema de câmeras se o volume ocupado pela cena se encontra totalmente fora do fecho convexo determinado pelos centros de projeção das câmeras.

4.4.4 Invariantes à Coloração e Unicidade

Considerando a questão da unicidade, podemos verificar que, infelizmente, existem várias soluções possíveis para um determinado conjunto de entrada para o problema. Podemos construir exemplos em que vários conjuntos diferentes de voxels, contidos no espaço que envolve a cena, podem ser capazes de reproduzir as imagens de entrada através de uma atribuição adequada de cores. Logo, precisamos adicionar alguma restrição ao problema de forma que a solução a ser obtida seja única.

Uma forma de garantirmos a unicidade da solução do problema é impor a condição de que esta venha a ser composta por elementos que satisfaçam alguma propriedade que seja *invariante* em todas as cenas consistentes.

Um exemplo de invariante é o chamado *invariante forte* o qual é formado por elementos que estejam presentes em todas as possíveis cenas consistentes. A grande dificuldade associada aos invariantes fortes é que estes são raros e nem sempre existem em número suficiente para obtermos uma reconstrução completa com relação ao conjunto de imagens de entrada.

Seitz propõe que, ao invés de trabalharmos com invariante rígidos, os quais não nos garantem a existência de uma solução viável, devemos procurar *invariantes à coloração* os quais são definidos da seguinte forma:

Definição 4.4.5 (Invariante à coloração). *Um voxel v é invariante à coloração em relação ao conjunto de imagens de entrada I, se para todo par de cenas consistentes S e S', se $v \in S \cap S'$ então $cor(v, S) = cor(v, S')$[71].*

Em outras palavras, um voxel invariante à coloração não precisa estar presente em todas as cenas capazes de reproduzir os dados de entrada, contudo, sua cor deve ser sempre a mesma em cada uma delas em que estiver presente.

Identificaremos agora que elementos de $\mathcal{V}$ satisfazem as condições necessárias para serem considerados invariantes à coloração.

Seja p um pixel pertencente a uma imagem I_i. Definimos o voxel v_p como o voxel mais próximo ao conjunto de câmeras dentre os voxels pertencentes ao conjunto $\{S(p) \mid S \, \textit{é uma cena consistente}\}$. Afirmamos que v_p é um invariante a coloração.

Prova 4.4.6. *Suponha que $v_p \in S$, onde S é alguma cena consistente; então $v_p = S(p)$. Isto é verdade já que se $v_p \neq S(p)$ então $S(p)$ seria mais próximo ao sistema de câmeras que v_p, o que é impossível pela definição de v_p. Logo, v_p tem sempre a mesma cor em todas as cenas foto-consistentes S em que estiver presente, pois $cor(v_p, S) = cor(S(p), S) = cor(p, I_i)$.*

Logo, uma coloração de voxels a partir de um conjunto de imagens $I_1, \cdots, I_m$ de uma cena S é definida como:

$$\overline{S} = \{v_p \mid p \in I_i, 1 \leq i \leq m\} \tag{4.4.1}$$

Em seu artigo[71], Seitz prova que uma coloração de voxels definida acima satisfaz as seguintes propriedade:

- $\overline{S}$ é uma cena consistente.
- Todo $v \in \overline{S}$ é invariante à coloração.
- $\overline{S}$ pode ser obtido a partir de qualquer conjunto C de câmeras que satisfaça a restrição de ordenação da visibilidade.

As provas de cada uma destas afirmações podem ser encontradas no trabalho original.

4.4.5 Cálculo da Coloração de Voxels

Quando a configuração das câmeras satisfaz a restrição de ordenação da visibilidade, podemos afirmar que a norma que mede a distância dos pontos no espaço às câmeras determina uma partição do conjunto de voxels $\mathcal{V}$ em um conjunto de camadas determinada pelas seguintes expressões:

$$\mathcal{V}_C^d = \{v | \, \|v\|_C = d\} \tag{4.4.2}$$

$$\mathcal{V} = \bigcup_{i=1}^{r} \mathcal{V}_C^{d_i} \tag{4.4.3}$$

onde $d_1, \cdots, d_r$ é uma seqüência crescente de números reais.

O algoritmo consiste, então, em percorrer os voxels em cada uma das camadas, em uma ordem crescente de distância, projetando-os nas imagens em que estão visíveis e avaliando-os segundo sua consistência. Se um voxel projetado é considerado consistente então ele é colorido com base em uma função das cores obtidas das imagens em que se encontra visível. Caso contrário, atribui-se uma cor transparente, o que corresponde a removê-lo do modelo.

Na prática, a visibilidade pode ser calculada através da associação de um mapa de visibilidade Mv_i para cada imagem I_i (Figura 4.3). No início do algoritmo, atribui-se o valor 0 para cada uma das posições dos mapas de visibilidade. Quando um voxel v é avaliado, determina-se para cada I_i a projeção $proj_{I_i}(v)$ do centróide de v em I_i. Se o valor na posição dada por $proj_{I_i}(v)$ em Mv_i for igual a zero então o voxel é visível, caso contrário, ele está ocluso. Os mapas de visibilidade são atualizados cada vez que um voxel é considerado consistente, bastando para isso atribuir o valor 1 à posição associada à projeção do centróide de v em cada uma das imagens em que estiver visível.

Devido ao fato de que as imagens da cena não refletem uma cena completamente lambertiana, além da presença de artifícios causados por ruído e quantização, é necessário introduzir medidas estatísticas que meçam a verossimilhança da consistência de um voxel. Seitz propõe que a consistência do voxel seja decidida pelo teste da razão da verossimilhança baseada na estatística:

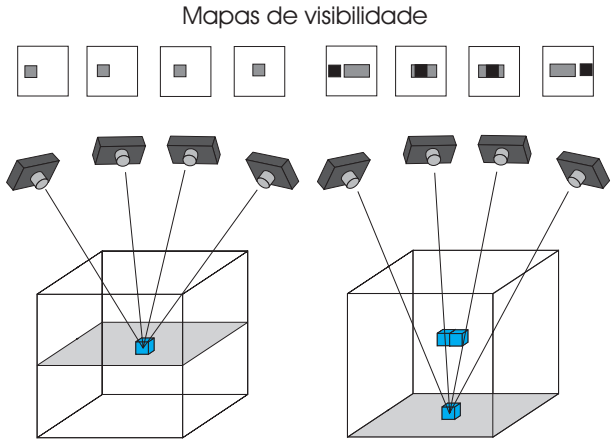


Figura 4.3: Determinação da visibilidade

$$\lambda_k(V) = \frac{(n-1)s^2}{\sigma_0^2} \tag{4.4.4}$$

onde k é o número de imagens em que o voxel está visível, s o desvio padrão calculado sobre o conjunto de cores em $proj_{I_i}(v)$, n a cardinalidade deste conjunto de cores e σ_0 o desvio padrão de uma distribuição normal representando os erros introduzidos pelo sensor. Sob hipótese de consistência, $\lambda_k(V)$ tem distribuição χ^2_{n-1}. Se $\lambda_k \geq T$, onde T é um limite global calculado na distribuição χ^2 para um nível de significância escolhido, então o voxel é considerado consistente, caso contrário é inconsistente e deve ser removido do modelo. Abaixo descrevemos o algoritmo de coloração de voxels (Algoritmo 2).

Antes de encerrarmos a descrição do método proposto por Seitz, seria importante lembrar que uma outra hipótese foi assumida para que pudéssemos aplicar as idéias aqui descritas. Esta hipótese é a de que as colorações dos diferentes voxels sejam totalmente independentes, hipótese esta que permite que o problema possa ser resolvido através de simples consultas locais sobre a foto-consistência de cada voxel sem nos obrigar a utilizar métodos mais sofisticados para deter-

Programa 2 Coloração de Voxels

$S \leftarrow \emptyset$
for $i = 1, \cdots, r$ **do**
  for all $V \in \mathcal{V}_C^{d_i}$ **do**
    projete V em $I_1, \cdots, I_n$ e calcule $\lambda_k(V)$
    if $\lambda_k(V) \leq T$ **then**
      $S \leftarrow S \cup \{V\}$

minar uma coloração que levasse em consideração a coerência espacial existente entre elementos vizinhos. Iremos em seguida rever a maior parte destas idéias dentro de um esquema formal, o qual define uma teoria sobre forma com base em foto-consistência.

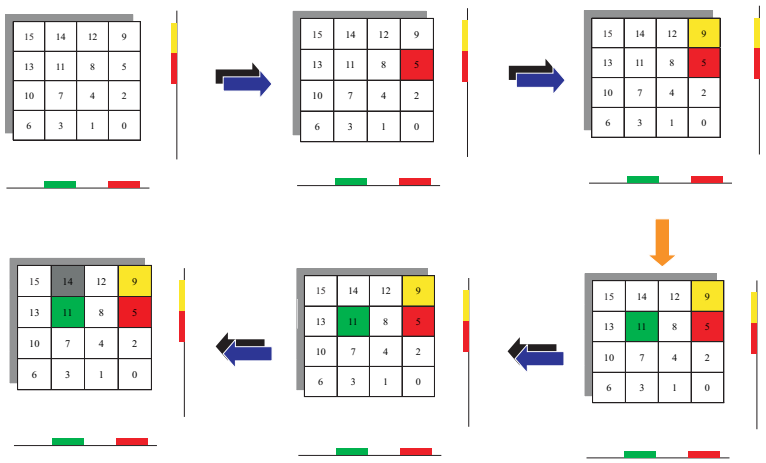


Figura 4.4: Exemplo de aplicação do método de coloração de voxels

4.5 Escultura do Espaço (*Space Carving*)

A teoria de reconstrução de forma através de escultura do espaço, estabelecida por Seitz e Kutulakos [44], define precisamente o que

podemos inferir sobre a forma de uma cena a partir de uma coleção de imagens obtidas em condições gerais e não controladas.

Apesar de boa parte das idéias apresentadas nesse trabalho já estarem presentes no trabalho sobre coloração de voxels de Seitz e Dyer, podemos destacar algumas diferenças fundamentais, como a apresentação de uma análise sobre reconstrução de cenas totalmente independente de algoritmos e a proposta de um método baseado na teoria subjacente, capaz de reconstruir cenas a partir de configurações de câmeras posicionadas arbitrariamente no espaço que contém a cena. Este novo algoritmo, denominado *Escultura do Espaço*, pode ser visto como uma generalização do algoritmo de Coloração de Voxels para configurações de câmeras arbitrárias.

Uma outra característica importante é a ênfase na reconstrução de cenas com base em um mínimo possível de restrições ou de conhecimentos sobre a cena. Algumas restrições, utilizadas normalmente em métodos convencionais de reconstrução, não são levadas em consideração como, por exemplo, restrições sobre a geometria e topologia da cena, restrições sobre o posicionamento das câmeras, existência de feições específicas nas imagens de entrada (como pontos, arestas, cantos, linhas, contornos, textura) e conhecimento à priori de correspondências.

Nos trabalhos anteriores, jamais se cogitou a possibilidade de se resolver o problema de reconstrução de cenas sem supor ao menos uma das condições descritas acima. Porém, Kutulakos e Seitz demonstraram que isto pode ser feito nos casos em que a radiância dos pontos da cena a ser reconstruída pertence à classe de funções de radiância *localmente computáveis*, que definiremos nesta seção.

Toda a teoria de Kutulakos e Seitz é baseada na formulação do problema de reconstrução de formas como um problema de satisfação de um conjunto de restrições. É mostrado que um conjunto de fotos de uma cena rígida determina um conjunto de restrições que devem ser satisfeitas por qualquer cena que se projete nestas fotos. Kutulakos e Seitz, ao investigar a classe de equivalência das formas que reproduzem as imagens de entrada, provaram a existência de um membro especial desta classe, chamado *Photo Hull*, que pode ser calculado através do algoritmo denominado Escultura do Espaço e que engloba todas as possíveis formas foto-consistentes. A introdução do conceito de *Photo Hull* tem importância similar à introdução do

conceito de *Fecho Visual* por Laurentini [45].

Veremos como as restrições impostas pelas imagens obtidas de uma cena determinam a classe de equivalência das formas que as reproduzem e como a relação existente entre elementos desta classe, e suas propriedades, nos permite especificar um algoritmo capaz de determinar o *Photo Hull*.

4.5.1 Restrições Fotométricas

Seguindo a notação utilizada em [44], uma forma $\mathcal{V}$ é definida como um conjunto fechado de pontos do $\mathbb{R}^3$ visto por um conjunto de n câmeras $C_i, \cdots, C_n$ em $\mathbb{R}^3 - \mathcal{V}$. Definimos a *radiância de um ponto* p na superfície $Surf(\mathcal{V})$ da forma com sendo a função $rad_p(\xi)$ que associa a cada raio orientado ξ a cor da luz refletida por p ao longo de ξ. Uma *descrição forma-radiância* nada mais é que uma forma $\mathcal{V}$ a qual associamos uma função de radiância para cada ponto p em $Surf(\mathcal{V})$. Esta descrição é suficiente para reproduzir uma foto da cena a partir de qualquer ponto de vista.

Kutulakos afirma que uma imagem de uma cena 3D particiona o espaço de todas as possíveis descrições *forma-radiância* em duas famílias: aquelas que reproduzem a imagem e aquelas que não a reproduzem. Esta restrição é caracterizada pelos conceitos de foto-consistência hierarquicamente definidos abaixo.

Definição 4.5.1 (Foto-consistência de um ponto). *Seja U um subconjunto arbitrário do $\mathbb{R}^3$. Um ponto $p \in U$ visível a partir de C_i é foto-consistente com a foto em C_i quando (i) p não se projeta em um pixel de fundo, e (ii) a cor na projeção de p é igual a $rad_p(\vec{pc})$. Se p não é visível a partir de C_i, então ele é trivialmente foto-consistente em relação à foto C_i.[44]*

Definição 4.5.2 (Foto-consistência de uma descrição forma-radiância). *Uma descrição forma-radiância de uma cena é foto-consistente com uma foto em C_i, quando todos os pontos visíveis a partir de C_i são foto-consistentes e todo pixel que não pertence ao fundo é a projeção de um ponto de $\mathcal{V}$.[44]*

Definição 4.5.3 (Foto-consistência de uma forma). *Uma forma $\mathcal{V}$ é foto-consistente com um conjunto de fotos quando existe uma*

atribuição de funções de radiância aos pontos visíveis de $\mathcal{V}$ que torna a descrição forma-radiância resultante consistente com todas as fotos.[44]

Basicamente existem dois tipos de restrições determinadas pelas imagens de entrada sobre a forma de uma cena foto-consistente:

- restrições impostas pela segmentação fundo/objeto das imagens da cena.
- restrições impostas pela coerência entre as cores dos pontos da cena vistos pelas imagens.

A primeira restrição impede que um ponto de uma cena $\mathcal{V}$ se projete em um pixel pertencente ao fundo de uma imagem. Logo, uma imagem I_i tomada de uma câmera com centro de projeção C_i restringe o espaço que contém a cena foto-consistente a um cone determinado pelos raios que passam pelos pontos na região delimitada pela silhueta do objeto na imagem. A interseção dos cones determinados por cada uma das imagens segmentadas define a envoltória visual, como já vimos anteriormente.

Infelizmente, a envoltória visual se degenera no $\mathbb{R}^3$ na ausência de informações provenientes de uma segmentação fundo/objeto, o que nos obriga a considerar o emprego de um outro tipo de restrição.

A restrição imposta pelas cores da cena vista pelas imagens só é válida quando a cena em questão emite cor de forma coerente. Isto é o caso quando a função de radiância definida sobre a superfície da cena pertence à classe das funções de radiância *localmente computáveis*.

Definição 4.5.4 (Função de radiância localmente computável). *Uma função de radiância é denominada* localmente computável *se o seu valor em um ponto específico independe dos valores nos demais pontos na superfície.*

Isto significa que efeitos de iluminação global como inter-reflexões, sombras e transparência devem poder ser considerados irrelevantes.

Para cenas cuja função de radiância é localmente computável podemos supor a existência de um critério simples capaz de medir a de foto-consistência de um ponto da cena com base nas cores emitidas em cada imagem e suas respectivas direções.

Definição 4.5.5. *Um algoritmo* $consiste_K()$ *toma como entrada um conjunto de pelo menos* $K \leq N$ *cores,* $col_1, \ldots, col_k$, K *vetores* $\xi_1, \ldots, \xi_k$ *e as posições das fontes de luz(caso não lambertiano), e decide se é possível para um único ponto da superfície emitir luz na cor* col_i *na direção* ξ_i *simultaneamente para todos os* $i = 1 \cdots K$. *Assume-se que* $consiste_K$ *seja monotônico, o que significa que* $consiste_K$ $(col_1, col_2, \ldots, col_j, \xi_1, \xi_2, \ldots, \xi_j)$ *implica em* $consiste_K(col_1, col_2, \ldots,$ $col_{j-1}, \xi_1, \xi_2, \ldots, \xi_{j-1})$ *para toda permutação de* $1, \ldots, j$.*[44]*

Através do critério de foto-consistência estabelecido anteriormente, podemos determinar quando uma cena descrita através de uma representação forma-radiância é consistente com o conjunto de imagens de entrada. Precisamos agora especificar um método capaz de construir uma possível forma foto-consistente em relação a uma coleção de imagens de uma cena. Isto será feito com base em uma análise de propriedades que relacionam uma forma com subconjuntos de si mesma e com as imagens de entrada.

Lema 4.5.6 (Lema da visibilidade). *Seja* p *um ponto sobre a superfície de* $\mathcal{V}$, *e seja* $Vis_{\mathcal{V}}(p)$ *a coleção de todas as fotos de entrada nas quais* p *não se encontra oculto devido à* $\mathcal{V}$. *Se* $\mathcal{V}' \subset \mathcal{V}$ *é uma forma que também contém* p *em sua superfície então* $Vis_{\mathcal{V}}(p) \subseteq Vis_{\mathcal{V}'}(p)$.*[44]*

Lema 4.5.7 (Lema da não-foto-consistência). *Se* $p \in Surf(\mathcal{V})$ *não é foto-consistente com um subconjunto de* $Vis_{\mathcal{V}}(p)$, *então ele não é foto-consistente com* $Vis_{\mathcal{V}}(p)$.*[44]*

Em poucas palavras, o *lema da visibilidade* e o *lema da não-foto-consistência* demonstram que a visibilidade e a não foto-consistência de uma cena expressam uma certa forma de monotonicidade. O lema da visibilidade indica que a visibilidade de um ponto na superfície de uma descrição de forma $\mathcal{V}$ sempre aumenta à medida em que $\mathcal{V}$ se torna menor. Analogamente, o lema da não foto-consistência afirma que uma parte da cena que não é foto-consistente com um subconjunto das imagens de entrada não pode ser foto-consistente com o conjunto completo. Estes dois lemas nos levam ao teorema do sub-

conjunto o qual nos fornece a chave para um algoritmo incremental capaz de produzir uma forma foto-consistente.

Teorema 4.5.8 (Teorema do subconjunto). *Se $p \in Surf(\mathcal{V})$ não é foto-consistente, então nenhum subconjunto foto-consistente de $\mathcal{V}$ pode conter p.[44]*

Prova 4.5.9. *Seja $\mathcal{V}' \subset \mathcal{V}$ uma forma que contém um ponto p. Se p está na superfície de $\mathcal{V}$ então p tem que estar na superfície de $\mathcal{V}'$. Pelo lema da visibilidade segue-se que $Vis_{\mathcal{V}}(p) \subset Vis_{\mathcal{V}'}(p)$. Além disso, pelo lema da não-foto-consistência e pelo fato de que a radiância em p independe da radiância nos demais pontos, se p não é foto-consistente com em relação a $Vis_{\mathcal{V}}(p)$ então ele não pode ser foto-consistente com $Vis_{\mathcal{V}'}(p)$. Desta forma nenhum subconjunto de $\mathcal{V}$ foto-consistente pode conter p.*

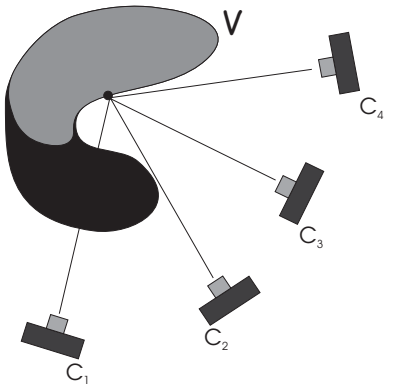


Figura 4.5: Teorema do subconjunto

Em resumo, o teorema do subconjunto indica que a não foto-consistência em um ponto de uma forma determina a não foto-consistência de uma família completa de formas.

4.5.2 O *Photo Hull*

A partir de uma estimativa inicial $\mathcal{V}$ que contenha a cena a ser reconstruída, é possível produzir uma forma foto-consistente através da aplicação sucessiva de operações de remoção de voxels não foto-consistentes na superfície de $\mathcal{V}$. Iremos agora mostrar que uma forma foto-consistente assim obtida possui propriedades especiais. Na verdade, esta forma converge para o que chamamos de *Photo Hull*.

Teorema 4.5.10 (Teorema do Photo Hull). *Seja $\mathcal{V}$ um subconjunto arbitrário do $\mathbb{R}^3$. Se $\mathcal{V}^*$ é a união de todas as formas foto-consistentes em $\mathcal{V}$ então todo ponto na superfície de $\mathcal{V}^*$ é foto-consistente. A forma $\mathcal{V}^*$ é denominada* Photo Hull.*[44]*

Prova 4.5.11 (Por contradição). *Suponha que p é um ponto sobre a superfície de $\mathcal{V}^*$ que não é foto-consistente. Se $p \in \mathcal{V}^*$ então existe uma forma foto-consistente, $\mathcal{V}' \subset \mathcal{V}^*$, que também contem p em sua superfície. Pelo teorema do subconjunto segue-se que $\mathcal{V}'$ não é foto-consistente.[44]*

O teorema acima descrito estabelece a relação existente entre o *Photo Hull* e a classe de formas foto-consistentes com um conjunto de imagens de uma cena. Podemos afirmar que o *Photo Hull* determina um limite superior justo para todas as soluções foto-consistentes com um conjunto de imagens e pode ser vista como a maior forma foto-consistente que pode ser obtida.

Podemos observar agora que os invariantes à coloração utilizados no algoritmo de Coloração de Voxels são um caso particular do *Photo-Hull* para o problema em que as câmeras satisfazem a restrição da ordenação de visibilidade.

Apesar de ter sido provado aqui que o Photo Hull satisfaz os critérios de foto-consistência, não foi mostrado que o mesmo define um conjunto fechado satisfazendo assim a definição de forma especificada no início da seção. A prova de que o Photo Hull consiste em um conjunto fechado é apresentada no trabalho original. Para questões práticas basta sabermos que, no caso de representações discretas, o Photo Hull satisfaz as condições necessárias para ser considerado uma forma.

4.5.3 Algoritmo

O algoritmo de escultura do espaço é baseado no *Teorema do Subconjunto* o qual nos indica que, dado um volume inicial que contenha a cena, é possível determinar uma forma foto-consistente através da remoção incremental de elementos não foto-consistentes em sua superfície. Sabemos pelo *Teorema do Photo Hull* que este processo converge para a forma maximal desejada.

O algoritmo para a determinação do *Photo Hull*, descrito abaixo (Algoritmo 3), é similar ao algoritmo de coloração de voxels, excetuando-se o fato de que a questão da visibilidade não é mencionada explicitamente.

Programa 3 Escultura do Espaço

```
Estabeleça um volume inicial V que contenha a cena.
fotoconsistência ← 1
repeat
  L ← Surf(V)
  repeat
    Escolha um voxel v ∈ L ainda não visitado
    Projete v em todas as imagens pertencentes a Vis_L(v).
    Determine a foto-consistência de v usando
    consiste_K(col_1, col_2, ..., col_j, ξ_1, ξ_2, ..., ξ_j)
    if v não for foto-consistente then
      V ← V − {v}
      fotoconsistência ← 0
    else
      fotoconsistência ← 1
  until fotoconsistência = 0 ∧ L = ∅
until fotoconsistência = 1
V* ← V
retorne V*
```

Implementação

O tratamento da visibilidade é uma questão importante no algoritmo de escultura do espaço, já que não admitimos restrição alguma sobre

a configuração das câmeras.

Kutulakos propõe uma solução elegante para a determinação da visibilidade no caso de configurações genéricas a qual se baseia em várias iterações de um algoritmo denominado Varredura por planos, similar ao algoritmo de Coloração de Voxels. Tal algoritmo é aplicado a cada uma das seis direções principais até que nenhuma remoção tenha sido efetuada, garantindo dessa forma a convergência do processo.

Primeiramente determina-se um volume inicial $\mathcal{V}$ que envolve a cena a ser reconstruída, juntamente com um plano de varredura Π posicionado totalmente à frente de $\mathcal{V}$. Move-se Π em direção a $\mathcal{V}$ até que Π corte um subconjunto de voxels de $\mathcal{V}$.

Em seguida, determina-se o conjunto de voxels na interseção de Π e $\mathcal{V}$. Para cada um dos voxels na interseção, calcula-se sua foto-consistência em relação ao conjunto de imagens em que se encontra visível.

O mesmo esquema de mapas de visibilidade adotado no algoritmo de coloração de voxels pode ser utilizado para manter a informação sobre a visibilidade dos voxels.

Os voxels que não satisfazem os critérios de foto-consistência são removidos; caso contrário são coloridos de acordo com as imagens em que estão visíveis. Após este passo, os mapas de visibilidade são atualizados convenientemente de forma que representem coerentemente a nova superfície visível.

Finalmente, efetua-se uma movimentação do plano em direção ao volume por uma distância de comprimento igual a uma das dimensões de um voxel, iniciando-se, logo em seguida, uma nova iteração. O processo termina quando o plano Π está posicionado atrás do volume inicial. Os passos que descrevemos encontram-se codificados em pseudo-linguagem no Algoritmo 4.

Através do algoritmo *Varredura por Planos* podemos calcular uma reconstrução parcial em relação a um subconjunto de câmeras que se posiciona atrás de um plano perpendicular a uma das direções principais. Contudo, isto não é suficiente para termos uma reconstrução global correta da forma da cena desejada.

Kutulakos propõe então que efetuemos uma seqüência de aplicações do algoritmo *varredura por planos* em cada uma das seis direções principais utilizando-se um conjunto de câmeras distinto, em

cada aplicação, de forma a evitar que uma mesma câmera seja considerada repetidamente em varreduras realizadas em diferentes direções (Figura 4.6).

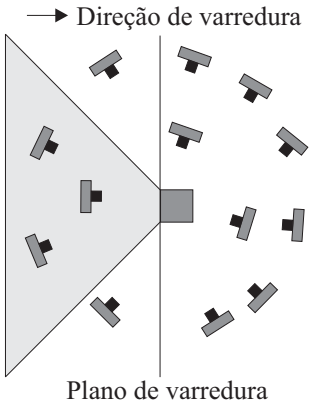


Figura 4.6: Câmeras consideradas em uma varredura para um determinado voxel.

Ao final das varreduras em cada uma das seis direções principais determina-se o conjunto de voxels que foram avaliados em mais de uma varredura e efetua-se uma checagem de consistência considerando-se todo o conjunto de câmeras em que estão visíveis. O processo termina quando nenhuma remoção de voxels é efetuada em uma determinada iteração (ver Algoritmo 5).

A implementação da checagem de consistência pode ser efetuada de forma similar ao algoritmo de Coloração de Voxels. Uma análise interessante sobre estatísticas utilizadas para a checagem de consistência em algoritmo de escultura do espaço pode ser encontrada na tese de Broadhurst [13].

Programa 4 Varredura por planos

Estabeleça um volume inicial $\mathcal{V}$ que contenha a cena.
Posicione um plano de varredura Π à frente do volume inicial $\mathcal{V}$.
Mova Π em direção a $\mathcal{V}$ até que eles se interceptem
repeat
  Calcule $\Pi \cap \mathcal{V}$
  for all voxel $v \in \Pi \cap \mathcal{V}$ **do**
    Sejam $c_i, \ldots, c_j$ as câmeras à frente de Π para as quais v se projeta em pixels não marcados.
    Determine a foto-consistência de v usando $consiste_K(col_1, col_2, \ldots, col_j, \xi_1, \xi_2, \ldots, \xi_j)$
    if v for inconsistente **then**
      $\mathcal{V} \leftarrow \mathcal{V} - \{v\}$
    else
      marque os pixels em que v se projeta.
  Mova o plano um voxel em direção ao $\mathcal{V}$
until $\mathcal{V}$ esteja a frente de Π

Programa 5 Escultura do Espaço por Múltiplas Varreduras

Determine um volume inicial $\mathcal{V}$ que contenha a cena.
repeat
  Aplique o algoritmo *Varredura por Planos* em cada uma das seis direções principais e atualize $\mathcal{V}$ adequadamente.
  for all voxel em ($\mathcal{V}$) cuja consistência foi avaliada em mais de uma varredura por planos **do**
    Sejam $c_i, \ldots, c_j$ as câmeras que foram utilizadas na checagem de consistência de v em alguma varredura de planos.
    Determine a foto-consistência de v usando $consiste_K(col_1, col_2, \ldots, col_j, \xi_1, \xi_2, \ldots, \xi_j)$
until nenhum voxel tenha sido removido em nenhuma das etapas
$\mathcal{V}^* \leftarrow \mathcal{V}$

4.6 Variações

Diversas variações sobre os métodos de Coloração de Voxels e Escultura do Espaço podem ser encontradas na literatura. Normalmente estas variações consistem em modificações e especializações com base em algum dos seguintes aspectos:

- tipo de representação utilizada (octree, voxels).
- tipo de função de consistência.
- tratamento da visibilidade.
- abordagem determinística ou probabilística.
- utilização de métodos de refinamento através de otimização.
- uso de espaços volumétricos alternativos.

Iremos apresentar aqui algumas das principais variações encontradas na literatura sobre Escultura do Espaço e Coloração de Voxels. Com exceção de Broadhurst e Cipolla [15], os quais introduziram mudanças significativas na modelagem do problema ao proporem uma abordagem probabilística, a maioria dos métodos mantém a estrutura básica proposta por Seitz, Kutulakos e Dyer.

4.6.1 Coloração de Voxels baseada em Múltiplas Hipóteses

O método de Coloração de Voxels baseada em múltiplas hipóteses foi proposto por Eisert et al. [25]. Uma hipótese nada mais é que uma possível atribuição de cor a um voxel específico. Diferentemente do método de coloração de voxels, esta técnica é capaz de lidar com quaisquer configurações de câmeras possibilitando reconstruções de cenas arbitrárias. Uma outra característica importante a ser mencionada é a ênfase de Eisert na combinação das técnicas baseadas em critérios de foto-consistência e na segmentação dos objetos de interesse nas imagens de entrada para obtenção de reconstruções de boa qualidade.

O método se subdivide em três etapas principais:

- Determinação do volume inicial que contém a cena.
- Atribuição de hipóteses a cada voxel do volume inicial.
- Verificação da consistência das hipóteses em relação à cada ponto de vista seguido de sua eventual remoção ou validação.

O primeiro passo consiste em determinar um volume que contenha a cena a ser reconstruída. Como no método de Seitz, o volume é determinado por uma caixa envolvente discretizada, gerando um conjunto $\mathcal{V}$ de voxels onde cada um deles é referenciado através da notação v_{lmn}, onde l, m e n são índices em uma matriz tridimensional. Ao contrário da coloração de voxels, atribui-se inicialmente a cada voxel uma coloração transparente.

Na etapa seguinte, atribui-se a cada voxel um conjunto de hipóteses onde a k-ésima hipótese atribuída ao voxel V_{lmn} é dada por

$$H^k_{lmn} = (R(X_i, Y_i), G(X_i, Y_i), B(X_i, Y_i)) \tag{4.6.1}$$

onde (X_i, Y_i) são as coordenadas da projeção perspectiva do centro do voxel na i-ésima imagem e R,G e B especificam as componentes de cor associadas à hipótese. Uma hipótese H^k_{lmn} é atribuída a um voxel v_{lmn}, se a sua cor associada for consistente em relação a pelo menos duas imagens I_i e I_j. Nesta etapa não é levada em consideração a visibilidade dos voxels, já que a geometria ainda não é conhecida.

Finalmente, efetua-se um passo de remoção das hipóteses, levando-se em consideração a visibilidade de cada um dos voxels na cena em construção. Para cada ponto de vista, associamos um mapa de visibilidade que guarda o índice do voxel visível para cada pixel.

Os voxels não transparentes (aos quais foram atribuídas hipóteses) na superfície do volume envolvente são tomados como uma estimativa inicial da geometria da cena. Então, para cada vista separadamente, é feita uma checagem de consistência das hipóteses de cada voxel na superfície visível, isto é, a cor de cada hipótese é comparada com a cor da projeção do voxel visível pela câmera associada ao ponto de vista corrente. Se a diferença entre a cor associada à uma hipótese e a cor do pixel correspondente à projeção do voxel for maior que um certo valor arbitrário, então a hipótese é descartada. Se não restar

hipótese alguma para um determinado voxel então associamos a ele uma cor transparente e atualizamos a superfície visível em relação àquela vista. Efetua-se o mesmo processo várias vezes para cada vista até que não haja nenhuma remoção de hipóteses. O conjunto de voxels opacos (não transparentes) resultante determina a coloração de voxels desejada.

Uma diferença fundamental entre o método de Seitz e o método de Eisert é que no método de Seitz a checagem de consistência é feita considerando-se todas as imagens simultaneamente, enquanto que no método de múltiplas hipóteses checa-se a consistência em relação a uma única imagem por vez. A estratégia do método de múltiplas hipóteses simplifica a determinação das superfícies de visibilidade já que elas são calculadas em relação a um único ponto de vista através de uma simples varredura no sentido da menor distância para a maior distância em relação às câmeras. Por outro lado é necessário um processamento adicional já que todos os voxels devem ser testados, inclusive aqueles pertencentes ao interior do volume.

4.6.2 GVC(Generalized Voxel Coloring)

O método GVC [21] consiste em uma implementação alternativa para o algoritmo de escultura do espaço no qual a visibilidade é calculada de forma exata, isto é considerando-se todos os pontos de vista em que um voxel esteja visível durante a verificação de sua foto-consistência. É necessário esclarecer que a primeira versão do algoritmo de escultura do espaço [43], contemporânea ao surgimento do método GVC, não era capaz de lidar com a visibilidade de forma exata, utilizando somente um subconjunto das imagens em que um voxel está visível durante a checagem da foto-consistência. Somente quando Kutulakos e Seitz publicaram o trabalho intitulado *A Theory of Shape by Space Carving* [44] é que a questão do tratamento exato da visibilidade foi definitivamente incorporada ao algoritmo de Escultura do Espaço.

Existem duas variantes do algoritmo GVC, cada qual utilizando uma diferente estrutura de dados para lidar com o problema de determinação da visibilidade (Figura 4.7). A primeira delas, chamada GVC-IB, utiliza uma estrutura chamada buffer de items (*item buffers*), a qual armazena, para cada pixel p de cada imagem $I_i \in I$, o voxel que esta visível a partir de p. A segunda versão, denominada

GVC-LDI, é mais sofisticada e armazena para cada pixel p de cada imagem $I_i \in I$ uma lista ordenada por profundidade de todos os voxels que se projetam em p. Esta estrutura na verdade é conhecida em computação gráfica por *Layered Depth Images*[74].

GVC-IB

Primeiramente, associa-se um identificador único a cada voxel. Em seguida, um buffer de itens é calculado para cada imagem através da rasterização de cada voxel por meio de uma algoritmo similar ao *z-buffer*. Cada elemento do buffer de itens armazena um identificador e um valor de distância em relação à câmera associada. No momento em que um voxel é rasterizado é possível determinar quais são os pixels que estão contidos em sua projeção.

A determinação do conjunto de pixels $Vis(v)$ para os quais um voxel v está visível é simples. Basta projetar v na imagem e comparar o identificador no buffer de itens associado a cada pixel na projeção de v com o identificador de v. Os pixels para os quais for verificada a igualdade entre os identificadores são inseridos no conjunto $Vis(v)$ sobre o qual é calculado a foto-consistência.

A operação de remoção de voxels torna os buffers de itens desatualizados, o que leva a necessidade de recalculá-los freqüentemente. Como isso é um procedimento custoso, deixamos os buffers de itens desatualizados na maior parte do tempo, o que não causa dano algum ao processo, já que a informação armazenada sobre a visibilidade indicará que alguns voxels estarão visíveis em apenas um subconjunto do conjunto total de câmeras onde a visibilidade é verificada. Como o algoritmo de coloração de voxels é conservativo, jamais irá remover voxels que façam parte da cena. No final do processo, nenhuma remoção ocorrerá e a consistência será calculada usando as informações atualizadas e exatas sobre a visibilidade dos voxels.

GVC-LDI

O GVC-LDI funciona de maneira similar ao anterior porém, como armazenamos uma lista completa de todos os voxels que se projetam em um determinado pixel de uma imagem, então a atualização dos voxels visíveis, após operações de remoção, é imediata, bastando para

isso retirar o elemento da cabeça da lista associada aos pixels afetados.

A vantagem do GVC-LDI é a de que as informações sobre a visibilidade dos voxels está sempre atualizada, o que nos leva a remover voxels mais rapidamente que o GVC-IB. Por outro lado, a estrutura de dados utilizada no GVC-LDI é extremamente grande e requer um espaço de armazenamento muito maior que a estrutura utilizada pela versão GVC-IB.

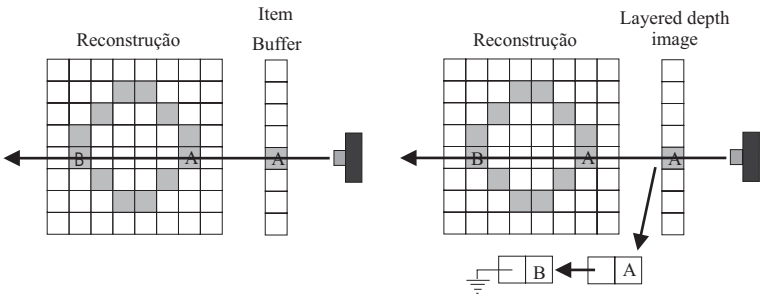


Figura 4.7: GVC-IB e GVC-LDI

4.6.3 Multiresolução de Prock

A tarefa de reconstruir cenas através de imagens requer um grande esforço computacional mesmo quando realizada através de técnicas simples e diretas como a coloração de voxels. Apesar de possuírem complexidade $O(nm)$, onde n é o número de voxels e m o número de imagens, os algoritmos de coloração de voxels envolvem um processamento árduo devido ao fato de trabalharem sobre dados volumétricos, os quais são sempre bastante grandes.

A. Prock e C. Dyer foram os primeiros a enfrentar o desafio de reduzir o tempo de processamento envolvido nos métodos de coloração de voxels, tendo como objetivo aproximar o tempo total de processamento ao necessário para reconstruções em tempo real [66].

Apesar de não alcançarem totalmente seu objetivo, já que o método proposto leva em torno de $0.5s$ a $1s$ para efetuar uma reconstrução, podemos afirmar sem dúvida que foi um grande avanço

em relação ao algoritmo clássico proposto por Seitz em direção a reconstruções em tempo real.

A idéia de Prock para reduzir o tempo total de processamento é baseada em três idéias chaves:

- uso de texturização em hardware.
- aproveitamento da coerência espacial.
- aproveitamento da coerência temporal.

Texturização em Hardware

A cada iteração do algoritmo de coloração de voxels necessitamos projetar centenas de milhares ou às vezes milhões de voxels pertencentes a uma camada do volume. Uma abordagem mais eficiente consiste em aproximar uma camada de voxels por um plano, o qual é projetado como um todo sobre as imagens de entrada. Isto pode ser feito de forma mais eficiente via texturização, através da qual projetamos as imagens de entrada sobre o plano correspondente a uma camada de voxels como se fossem verdadeiros *slides*. O procedimento de texturização não introduz nenhum custo adicional, já que pode ser efetuado eficientemente através das facilidades existentes nas placas gráficas aceleradas por hardware.

Refinamento Adaptativo

A principal motivação para a utilização de reconstrução por refinamento adaptativo se baseia no fato de que, apesar de representarmos o espaço que contém a cena através de uma representação volumétrica enumerativa, estamos, na realidade, interessados em determinar a superfície que envolve o volume de ocupação da cena. A superfície que determina a forma da cena, mesmo que representada através de voxels, contém um número de elementos pequeno quando comparado ao número de elementos presente no espaço de busca inicial.

Por este motivo, uma maior eficiência pode ser obtida se pudermos concentrar esforços especificamente nas regiões do espaço em que a superfície da cena está contida, ao mesmo tempo em que minimizamos o número de operações nas regiões consideradas vazias.

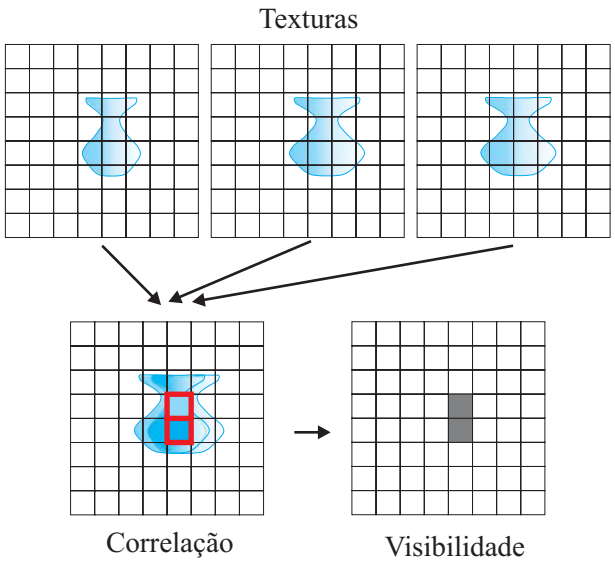


Figura 4.8: Projeção das texturas no espaço da cena

Obviamente, por não conhecermos a cena, não podemos determinar, logo de início, quais são estes espaços vazios, de tal forma que possamos efetuar de imediato uma subdivisão espacial adaptativa.

Prock propõe uma estratégia em que o algoritmo de coloração é aplicado sobre um conjunto inicial de voxels em baixa resolução sendo o resultado assim obtido fornecido como entrada para um processo de refinamento. O processo de refinamento consiste em subdividir cada voxel consistente em oito novos voxels, os quais são armazenados em uma lista. Percorre-se cada elemento da lista, na ordem determinada pela subdivisão em camadas do volume em função de sua distância ao conjunto de câmeras, verificando-se se cada um deles deve ser removido ou colorido. Em seguida, aplica-se o algoritmo de refinamento recursivamente sobre os elementos da lista que foram considerados consistentes.

Através desta representação em diversos níveis de resolução, pode-

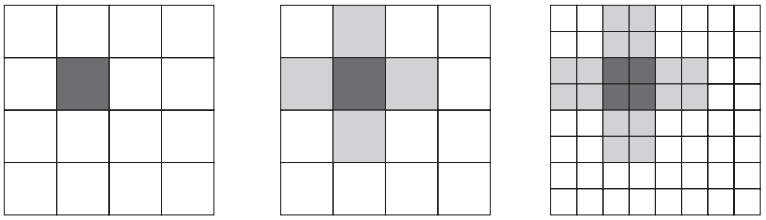


Figura 4.9: Refinamento

mos tornar o algoritmo muito mais eficiente, ainda que preservando todas as facilidades de manipulação fornecidas por uma representação volumétrica enumerativa.

Uma questão importante a se considerar é de que ao diminuirmos a resolução do espaço de representação corremos risco de perder partes da cena. Isto ocorre já que, quando diminuímos a resolução, estamos aumentando a região delimitada por um voxel. Isto aumenta a probabilidade de que as projeções de cada um destes elementos contenham mais pixels não consistentes, o que aumenta a possibilidade de que um voxel venha a ser descartado, apesar de conter fragmentos da superfície da cena em seu interior. No trabalho original podemos encontrar uma estimativa do quanto da cena é perdido ao se reduzir a resolução do espaço que contém a cena.

Prock, baseando-se na existência de uma coerência espacial entre partes vizinhas da cena, propõe uma heurística para evitar este problema. A idéia consiste em subdividir não só cada voxel v considerado consistente mas também todos aqueles que estiverem em uma vizinhança de v. De forma a não introduzir, no processo, um custo adicional muito grande, utiliza-se uma vizinhança de norma unitária.

Ao adotar uma estratégia de refinamento em que os voxels podem ter dimensões arbitrárias, torna-se claro que a aproximação de um voxel por seu centróide não mais é adequada. Prock adota uma aproximação determinada por uma região planar paralela à direção de varredura passando pelo centro do voxel. Apesar de ser uma aproximação melhor que a anterior, já que leva em consideração que a

projeção de um voxel nas imagens é dada por uma região poligonal e não por um ponto, ainda temos problema quando o voxel tem todas as três dimensões não desprezíveis.

Resolução	Original	Multiresolução	*Speedup*
32	0.9291s	0.7509s	1.23
64	5.777s	1.512s	3.82
128	48.33s	4.093s	11.8
256	335.8s	15.70s	21.4
512	2671s	64.98s	41.1

Tabela 4.2: Coloração de Voxels em Multiresolução x Coloração de Voxels Original. Os experimentos de Prock foram realizados em uma SGI O2 R5000 de 200 Mhz.

Uma limitação importante do método de Prock é a necessidade de se iniciar o algoritmo com uma boa estimativa da cena. De certa forma, a estimativa inicial deve conter uma boa "semente", capaz de gerar a cena completa com todos os detalhes significativos à medida em que é refinada nos estágios posteriores. Caso alguma parte da cena não esteja presente na estimativa inicial, então este erro será propagado para os demais estágios de refinamento, causando um comprometimento do restante do processo.

Tratamento de Cenas Dinâmicas

Prock foi além da reconstrução de cenas estáticas e ousou propor uma abordagem para a reconstrução de cenas dinâmicas. Utilizando as técnicas descritas acima, Prock explora a existência de coerência temporal entre quadros consecutivos para tornar ainda mais eficiente o processo de reconstrução.

Sua estratégia consiste em utilizar uma reconstrução em baixa resolução obtida a partir de um conjunto de quadros em um instante de tempo t_k como ponto de partida para reconstrução da cena em um instante t_{k+1}.

Assim como há problemas de perda de elementos em cenas estáticas devido à redução da resolução, também há problemas semelhantes e de certa maneira ainda mais graves quando se trata de cenas

dinâmicas. Infelizmente, se os objetos se moverem muito rapidamente através da cena, então a reconstrução em baixa resolução fornecida como volume inicial para a reconstrução em um instante posterior pode não conter partes importantes. Isto pode ser resolvido, com um pouco mais de dificuldade através de um processo de acompanhamento (*tracking*) do movimento dos objetos de interesse de forma a adicionar regiões ao volume inicial para que este leve em consideração o movimento dos objetos.

Ao invés de utilizar um procedimento de *tracking* sofisticado, Prock sugere a utilização da mesma heurística de busca local em uma vizinhança adotada para o caso de cenas estáticas. No entanto, a utilização desta estratégia fica limitada a cenas cujos objetos não tenham velocidade superior a um certo limiar. Alguns resultados da aplicação desta técnica sobre cenas dinâmicas são apresentados no trabalho original.

4.6.4 Visão Estéreo Aproximada

Neste trabalho, Kutulakos [42] propõe uma solução para o problema de visão estéreo aproximada através de vários pontos de vista (*approximated n-view stereo*) o qual consiste em recuperar uma seqüência de reconstruções volumétricas que aproximam incrementalmente uma cena 3D com forma arbitrária.

A solução apresentada para este problema é baseada em uma nova função de consistência baseada em uma transformação denominada *shuffle transform*.

Definição 4.6.1 (Shuffle Transform). *Uma transformação* $2D$ $T : I_1 \rightarrow I_2$ *que leva os pixels de uma imagem* I_1 *nos pixels de uma imagem* I_2 *é uma r-Shuffle transform, se e somente se, podemos encontrar para cada pixel* $j \in I_2$ *um pixel* $i \in I_1$ *com a mesma cor que* j *em um raio de vizinhança* r *em torno das coordenadas de* j*. A constante* r *é denominada raio de dispersão da transformação* T*.*

Em outras palavras uma *Shuffle Transform* nada mais é que uma reorganização dos pixels de uma imagem dentro de uma raio de vizinhança limitado. Além de representar transformações paramétricas, uma shuffle transform também é capaz de representar transformações

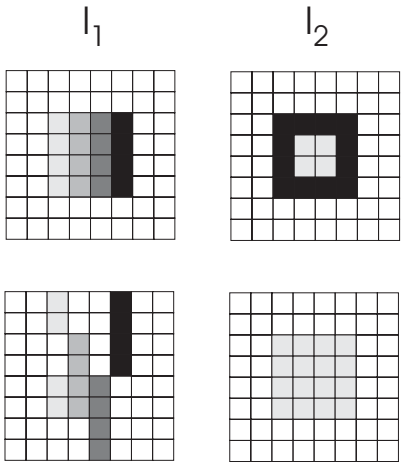


Figura 4.10: Exemplos de shuffle transforms

envolvendo descarte e substituição de elementos, como podemos ver na coluna à direita na figura 4.10.

Através de *shuffle transforms* podemos definir um novo tipo de consistência que nos permite caracterizar uma família de formas consistentes com as imagens, a menos de uma transformação.

Definição 4.6.2 (r-Consistência). *Um volume $\mathcal{V}$ é* r-consistente *se para toda imagem de entrada I_i existe uma r-shuffle $T : I_i \rightarrow I'_i$ que torna $\mathcal{V}$ consistente com as fotos $I'_i, \cdots, I'_m$.*

O restante do método é similar ao método de escultura do espaço. A única diferença está na utilização da função de consistência baseada em *shuffle transforms*. Na prática, a checagem de consistência é implementada através de um algoritmo capaz de determinar a existência de pelo menos um pixel com a mesma cor em cada um dos discos de raio r em torno das projeções de um voxel específico v. Quando esta existência é confirmada então dizemos que as cores associadas à projeção de um voxel v são consistentes a menos de uma *r-shuffle*.

Kutulakos afirma que seu método é aplicável à reconstrução na presença de erros de calibração, trata adequadamente a questão da

utilização de imagens discretas e permite reconstruções incrementais, chegando mesmo a definir um *Espaço de Escala Foto Consistente*.

Apesar da enorme versatilidade do método proposto algumas questões importantes foram deixadas em aberto:

1. como determinar o raio de dispersão mínimo que leva a uma reconstrução válida.

2. como aplicar as técnicas ao problema de reconstrução e auto-calibração a partir de n imagens.

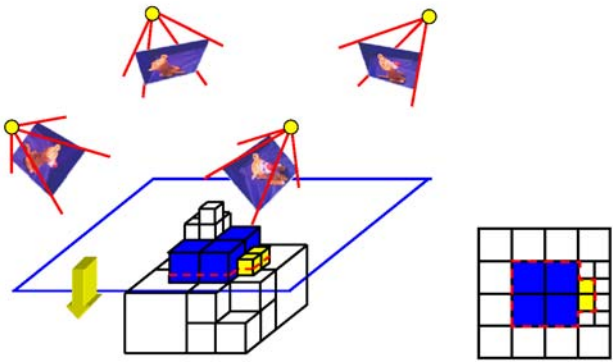
3. como desenvolver um algoritmo adaptativo em multiresolução através do qual diferentes partes de uma mesma cena possam ser aproximadas por elementos de diferentes escalas.

4.6.5 Refinamento Adaptativo através de Octrees

Em [57] foi proposta uma abordagem adaptativa baseada em refinamento adaptativo no qual o espaço de reconstrução é representado através de uma octree.

O método proposto inicia com uma representação grosseira do espaço de reconstrução definida por uma caixa que envolve a cena, a qual é representada pela célula na raiz da octree. Procura-se então, classificar a célula inicial de acordo com sua foto-consistência em relação as imagens de entrada. Se for possível classificar a célula como foto-consistente ou não-foto-consistente então o processo termina, e uma cor é atribuída à célula em função das cores nas regiões em que se projeta nas imagens de entrada. Caso isto não seja possível, a célula é subdivida em oito novas células que devem ser classificadas da mesma forma. O processo é repetido até que todas as células sejam classificadas quer seja como foto-consistente ou não-foto-consistente ou um nível de resolução máximo seja alcançado.

De acordo com o método, uma célula pode se classificada como foto-consistente, não-foto-consistente e indefinida. Diferentemente dos métodos anteriores, o autor propõe que uma célula seja considerada não-foto-consistente somente quando o teste de consistência falha para células no nível máximo de resolução, ou quando a célula se projeta completamente no vazio em relação ao conjunto de imagens de entrada. Células que não são consideradas foto-consistentes, mas que



(a) Escultura do Espaço Adaptativo (b) Seleção

Figura 4.11: (a) - Escultura do Espaço em uma octree. As células são processadas na ordem indicada pela seta. (b) - seleção das células em uma camada em um dado nível de refinamento (as células amarelas são as que devem ser processadas)

também não satisfazem as condições mencionadas anteriormente são consideradas indefinidas. A condição que envolve o teste de projeção no vazio naturalmente implica que sejam conhecidas as silhuetas do objeto a ser reconstruído, isto é a segmentação da cena em uma região de fundo e uma região do objeto.

De modo similar ao método de Prock e Sainz, o processo de determinação da foto-consitência de uma célula é realizado diretamente no espaço da cena através do registro das informações fotométricas sobre planos que aproximam conjuntos de células na octree em um determinado nível de resolução. Isto é feito através de mapeamento de textura projetivo, entretanto, diferentemente de Prock e Sainz, as informacões são projetadas com níveis de resolução compatíveis com o nível de refinamento corrente da octree.

Como é necessário checar se uma determinada célula se projeta completamente no vazio em relação ao conjunto de imagens de en-

trada também é necessário registrar informações de segmentação em cada uma das imagens de entrada. Além disso o teste de foto-consistência utilizado é robusto o suficiente para trabalhar com imagens que foram obtidas através de câmeras cujo nível de ruído varia de sensor para sensor. Logo, estas informações são também levadas para o espaço da cena através do registro de mapas de desvio que correspondem aos níveis de ruído dos sensores de cada uma das câmeras utilizadas para a tomada das imagens.

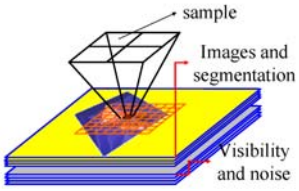


Figura 4.12: Registro

Programa 6 Escultura do Espaço Adaptativa

```
no ← octree
Faça no.celula receber o volume incial.
repeat
  nivel ← 0
  Limpe os mapas de visibilidade.
  for all Varrer os planos π_k, k = 2^level..0 do
    Determinar os mapas projetados(Registro) .
    for all os nós nd tais que (nd.celula ∩ π_k ≠ ∅) and
    (nd.celula.classe =NOT_EVALUATED) do
      pr ← nd.celula ∩ π_k
      Avaliar a foto-consistência de nd.celula.classe
      if nd.celula.classe = CONSISTENT then
        Atribuir as cores a nd.cell.
      else if nd.celula.classe = UNDEFINED then
        Criar 8 novos nós filhos de nd.
        Subdividir nd.celula em oito novas células.
        Rotular cada nova célula como NOT_EVALUATED.
        Atribuir as novas células aos filhos de nd.
    Atualizar os mapas de visibilidade.
  nivel ← nivel + 1
until que nenhuma célula tenha sido subdividida ou nivel < MAX_LEVEL
```

4.6.6 Escultura do Espaço através de uma única Câmera

Um dos grandes desafios na reconstrução por escultura do espaço é o de como realizá-la utilizando uma única câmera de mão. Em [] foi proposto um método no qual os parâmetros intrínsecos e extrínsecos de uma câmera de mão são determinados no momento da captura da imagem, ao contrário dos métodos anteriores que trabalhavam com esquemas de câmeras fixas pré-calibradas.

Os autores mostram que isto pode ser realizado através da inserção de um padrão de calibração no espaço da cena sem alterar a visibilidade dos objetos de interesse. O padrão, por sua vez, pode ser parcialmente encoberto pelos objetos da cena. De forma a lidar com essa situação foi adotado um método de calibração baseado em reconhecimento de modelos.

Uma das grandes dificuldades desta abordagem é a de como utilizar informações sobre a segmentação como por exemplo é feito em []. O problema reside no fato de que a princípio não é possível obter um modelo estatístico do fundo da cena, que permita a segmentação do objeto, para imagens capturadas de diferentes pontos de vista por uma única câmera não fixa. Para superar esta dificuldade os autores do método proporam uma estratégia no qual um modelo de fundo aproximado para cada imagem de entrada é calculado com base em um conjunto de imagens da cena sem os objeto de interesse. O conjunto de imagens da cena sem os objetos e registrado para cada imagem de entrada através de warping permitindo assim a segmentação do objeto desejado.

4.6.7 Outras Variantes

Uma variação bastante importante dos métodos de escultura do espaço / coloração de voxels se baseia em procedimentos de otimização para o refinamento de uma solução inicial através da minimização de *erros de reprojeção*. O erro de reprojeção é compreendido como o erro determinado pelas diferenças entre as imagens de entrada e as imagens geradas pela renderização da reconstrução obtida a partir dos mesmos pontos de vistas originais.

O objetivo destes métodos é lidar com o problema de que, em al-

guns casos, um único limiar (*threshold*) não é suficiente para se obter boas reconstruções, por exemplo, quando a cena não se comporta perfeitamente como uma superfície lambertiana. O que ocorre é que com o aumento do limiar pode-se obter soluções de baixa qualidade, por outro lado, com a redução deste mesmo valor, partes importantes da cena acabam sendo removidas.

O processo de refinamento por otimização utiliza *thresholds* especialmente adequados para cada uma das regiões da cena, gerando assim modelos que se ajustam melhor à forma desejada.

Diferentemente do que ocorre nos métodos mais convencionais, o processo de otimização pode tanto remover quanto adicionar voxels com o objetivo de minimizar o erro de projeção. Um exemplo deste método é o método de Slabaugh [76]. Um outro trabalho dentro desta classe de métodos é o trabalho de De Bonet e Paul Viola que propuseram um método chamado *Roxels Algorithm*, o qual tenta determinar reconstruções de cenas com valores de opacidade arbitrários [12].

Outras variações importantes encontradas na literatura são as que utilizam espaços de voxels alternativos, como os trabalhos de Saito [70], Kimura [39] e Slabaugh [78]. Vale a pena mencionar também o trabalho de Vedula [84] para cenas dinâmicas que utiliza um espaço de voxels 6D conectando dois espaços de voxels 3D consecutivos no tempo. Um pouco mais distantes em filosofia, mas também relacionados, são os trabalhos baseados em interfaces de propagação e Level Sets como [60, 26, 73].

Uma nova variação muito importante é a classe dos métodos probabilísticos de escultura do espaço [9, 14, 15, 13], os quais não serão descritos neste trabalho. Abaixo apresentamos uma pequena tabela que compara alguns dos métodos apresentados.

	câmeras em posições arbitrárias	multires.	cenas dinâmicas	prob.	visibil. exata
Coloração de Voxels	não	não	não	não	ok
Escultura do Espaço	ok	não	não	não	ok
GVC-IB	ok	não	não	não	não
GVC-LDI	ok	não	não	não	ok
Múlt. hipóteses	ok	não	não	não	ok
Coloração de Voxels em Multiresolução	não	ok	ok	não	ok
Shuffle Transforms	ok	possível	não	não	ok
Refinamento Adaptativo	ok	ok	não	ok	ok
Escultura do espaço	ok	não	não	ok	ok
Probabilística					

Tabela 4.3: Classificação dos algoritmos

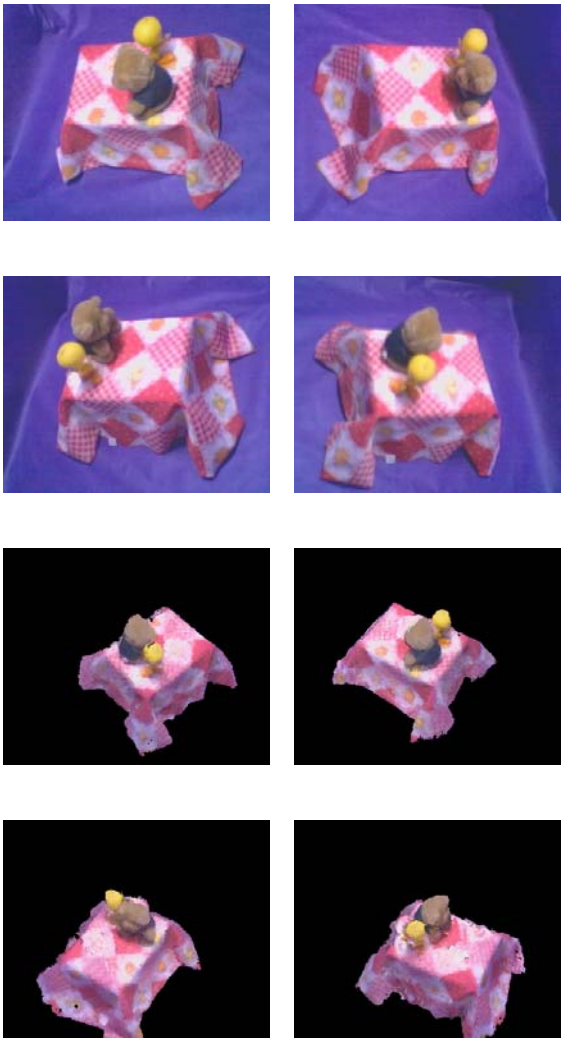
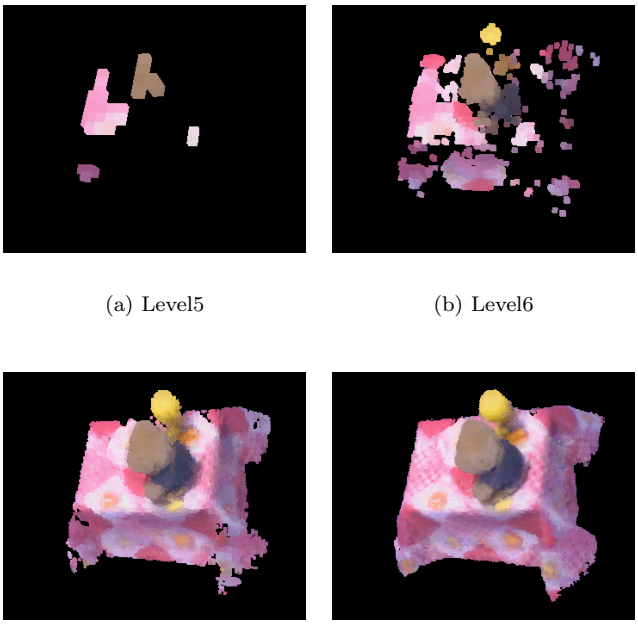


Figura 4.13: Imagens de entrada e a cena reconstruída (as quatro figuras inferiores)



(a) Level5 (b) Level6

(c) Level7 (d) Level8

Figura 4.14: Imagens de diferentes níveis de refinamento

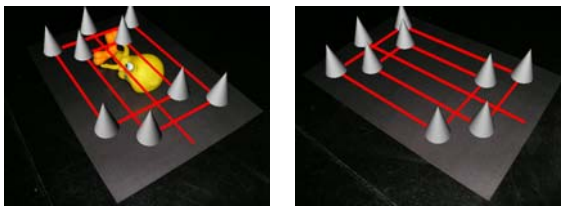
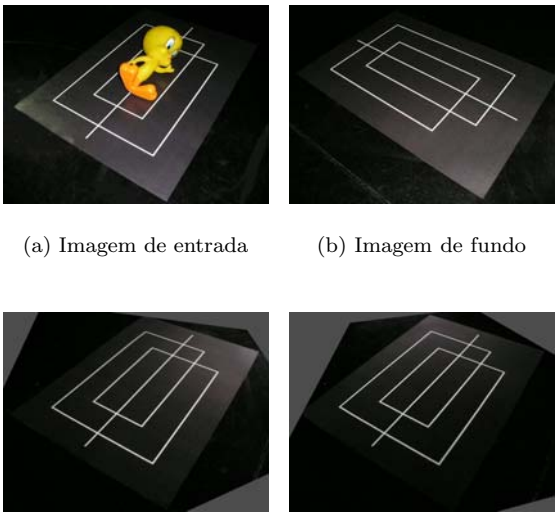


Figura 4.15: Image de entrada e de fundo com marcadores verificando a calibração



(a) Imagem de entrada

(b) Imagem de fundo

(c) Imagem de fundo registrada

(d) Outra imagem de fundo registrada

Figura 4.16: Imagens registradas

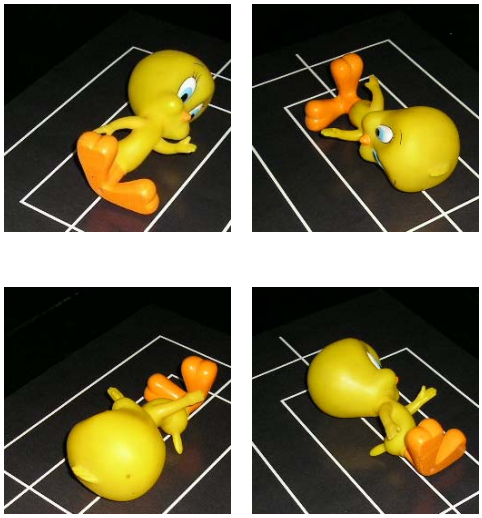


Figura 4.17: Imagens de entradas

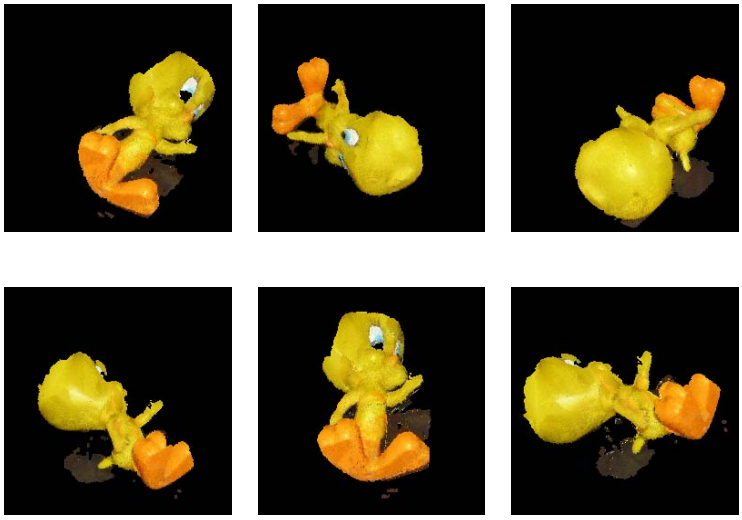


Figura 4.18: Imagens de entradas(quatro superiores) e reconstruções; as duas últimas correspondem a novos pontos de vista

Capítulo 5

Geração de Malhas

5.1 Introdução

O problema que vamos ressaltar neste capítulo é, de modo geral, enunciado da seguinte maneira: dado um conjunto finito de pontos $P \subset \mathbb{R}^3$ gerar uma superfície S tal que $P \subset S$ ou $max \{ ||p - S|| : p \in P \}$ é suficientemente pequeno.

Esta formulação acima não é muito precisa porque dá espaço a diversas interpretações e é mais classificado como um problema de construção de superfícies. Mas se conhecermos algo sobre a origem da nuvem de pontos P (e.g. o seu modelo real) e nosso objetivo é obter uma descrição exata desta superfície no computador, deparamo-nos com o problema de recontrução de superfícies (fig. 5.1). O conhecimento sobre a superfície é importante na escolha do melhor algoritmo de reconstrução.

A maioria dos métodos que propõem uma solução requerem pontos "suficientemente densos" para gerar a superfície desejada. Tal suficiência, conhecida como condição de amostragem, deve garantir uma boa aproximação em relação à superfície original, isto é, a superfície reconstruída é homeomorfa e suficiente próxima. Por exemplo, em Amenta et. al
refamenta é proposto o *power crust*, método baseado numa aproximação do MAT (Medial Axis Transform). Neste trabalho, é definida a característica local (LFS) de um ponto $p \in S$ como a distância

Figura 5.1: Superfície amostrada e reconstruída. Cortesia de A. Peixoto.

ao ponto mais próximo do eixo medial de S. Verifica-se então que se $P \subseteq S$ é uma r-amostra com $(r \leq 0.1)$, ou seja, a distância de qualquer ponto $p \in S$ ao ponto mais próximo da amostra P é no máximo $r \cdot LFS(p)$ então o método nos dá uma boa reconstrução. Intuitivamente, para curvas planares, isso exige que, em locais onde a curvatura é grande, deve haver um maior coeficiente de amostragem.

Um dos desafios do problema de reconstrução de superfícies está na busca de um método que cubra, de acordo com a necessidade da aplicação a ser considerada, o maior número de formas possível além de ter que resolver dados ruidosos.

Mencl & Muller [51] descrevem a existência de quatro classes de algoritmos de reconstrução. São elas: algoritmos baseados em subdivisão espacial (cf. Boissonant [11]), deformação (cf. Osher et al. [95]), funcões distância ou poligonização implícita (cf. Hoppe [34]), e crescimento incremental de superfícies ou de avanço de frente (cf. Bernardini et al. [7]. Daremos ênfase aos dois últimos enquadrando-os em aspectos matemáticos topológicos e geométricos. Para algoritmos de avanço de frente, faremos um estudo de caso detalhado sobre o Algoritmo Ball-Pivoting, que está incorporado ao software do curso.

Uma observação importante é que os métodos de geração de malhas são usados tanto na reconstrução no espaço da imagem, quanto na reconstrução no espaço da cena. No primeiro caso, malhas são geradas para cada uma das visadas de camera e depois essas malhas, depois de registradas num sistema de coordenadas comum, são integradas por costura, como no algoritmo de Zipper. Um alternativa para esse procedimento, é utilizar apenas os pontos e empregar um algoritmo de avanco de frente, como o Ball Pivot, ou mesmo fazer

uma triangulação de Delaunay diretamente (ver Sessão 5.2). No segundo caso, em geral a poligonização é feita a partir de uma função implicita associada aos dados volumetricos no espaço da cena (ver Sessão 5.3.1).

5.2 Algoritmos de Avanço de Frente

Em poligonização de superfícies usando métodos de avanço de frente [46, 29], a malha é construída progressivamente associando triângulos ao seu bordo a partir de algum critério geométrico (veja figura 5.2). O bordo da malha é composto de ciclos fechados de curvas lineares por partes. Este conjunto de curvas fechadas formam uma frente de avanço que é a fronteira entre regiões poligonizadas e não-poligonizadas. A iteração de um passo básico que incorpora triângulos ao bordo da malha resulta em uma propagação da frente que só é finalizada quando toda superfície for coberta pela malha.

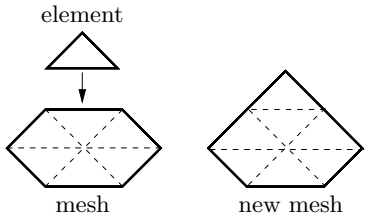
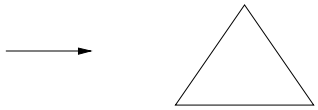


Figura 5.2: Idéia básica do método de avanço de frente. Linhas tracejadas representam arestas interiores e linhas sólidas representam arestas de bordo.

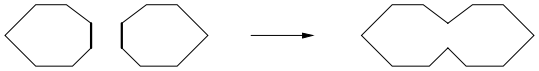
Embora a idéia de avanço de frente seja simples, os detalhes algoritmicos do método são complexos. A principal dificuldade com este método está no controle da junção ou separação dos diferentes ciclos que compõem a frente de avanço. Freqüentemente cada aresta de um novo triângulo criado na iteração é colada com outra aresta da frente, mudando a sua topologia. Para efeito de melhor entendimento, mudar topologia significa dizer que houve um acréscimo ou decréscimo do número de ciclos de bordo. Existem quatro tipos de mudanças

topológicas que podem ocorrer:

1. Criação de uma curva:

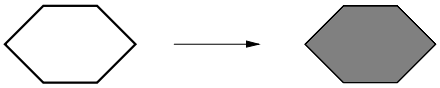


2. Duas curvas se juntam formando uma só:



3. Uma curva divide-se em duas:

4. Uma curva é fechada:



Essas mudanças topológicas são objetos de estudo da teoria de *handlebody*. É o que veremos logo adiante.

5.2.1 Teoria de Handlebody

A Teoria de Handlebody [69] é uma ferramenta matemática que irá nos ajudar a entender melhor as mudanças topológicas na construção da malha de uma superfície. Ela também nos fornece condições necessárias e suficientes para lidarmos com tais mudanças. Primeiramente vamos introduzir o conceito de *handle*(alça).

Definição 1. *$H_\lambda = (A_\lambda, B_\lambda)$ é um handle de índice $\lambda = 0, 1, 2$ tal que $B_\lambda \subseteq \partial A_\lambda$ onde $A_\lambda = D^\lambda \times D^{2-\lambda}$ e $B_\lambda = (\partial D^\lambda) \times D^{2-\lambda}$.*

De acordo com a definição acima, só existem três tipos de *handle*:

Tipo-0, $\lambda = 0$:

$$A_0 = D^0 \times D^2 = \quad \text{  }$$

$$B_0 = (\partial D^0) \times D^2 = \emptyset$$

Tipo-1, $\lambda = 1$:

$$A_1 = D^1 \times D^1 = \quad \text{  }$$

$$B_1 = (\partial D^1) \times D^1 = \quad \text{  }$$

Tipo-2, $\lambda = 2$;

$$A_2 = D^2 \times D^0 = \quad \text{  }$$

$$B_2 = (\partial D^2) \times D^0 = \quad \text{  }$$

Anexar um *handle* $H_\lambda = (A_\lambda, B_\lambda)$ ao bordo de uma superfície S consiste em associar por um homeomorfismo o conjunto $B_\lambda \subseteq \partial A_\lambda$ com um subconjunto de ∂S. Temos então o seguinte Teorema:

Teorema 3. *(Decomposição de Superfícies por Handlebodies) Para toda superfície S, existe uma sequência finita de superfícies $S_{i=1\ldots N}$ tal que $S_0 = \emptyset$ e a superfície S_i é obtida anexando um handle $H_\lambda = (A_\lambda, B_\lambda)$ ao bordo de S_{i-1}.*

Um exemplo clássico é a decomposição do toro com a sequência $S_4 = (((S_0 + H_0) + H_1) + H_1) + H_2$ (fig. 5.3)

Para cada *handle* existe um tipo de mudança topológica na superfície:

- O *handle* do tipo-0 cria sempre uma nova componente conexa homeomorfa a um disco e uma curva de bordo.

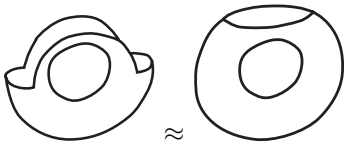
- Quando um *handle* do tipo-1 é anexado a uma superfície, podem ocorrer três situações:

$S_0 = \emptyset$

$S_1 = S_0 + \mathbf{H}_0$

$S_2 = S_1 + \mathbf{H}_1$

$S_3 = S_2 + \mathbf{H}_1$



$S_4 = S_3 + \mathbf{H}_2$

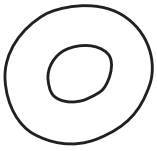


Figura 5.3: Decomposição do toro por *Handlebodies*. Cortesia de Hélio Lopes. $S_4 = (((S_0 + H_0) + H_1) + H_1) + H_2$.

  - ser anexado a dois intervalos disjuntos na mesma curva de bordo. Nesse caso, uma curva se divide em duas.
  - ser associado a intervalos de diferente curvas de bordo em uma superfície. A mudança topológica é caracterizada pelo aumento do gênero da superfície.
  - ser associado a diferentes componentes conexas.

- No caso do *handle* de tipo-2, uma curva de bordo é fechada.

A fim de aplicar essa teoria na construção de superfícies, precisamos de uma representação discreta bem como operadores discretos que cuidarão das as mudanças topológicas discutidas acima. Esse *framework* computacional será introduzido no próximo tópico.

5.2.2 Representação de Malha e Operadores de Handle

Cada malha é definida como $M = (V, E, F, B)$ onde V, E, F, B são respectivamente o conjunto de vértices, arestas, faces e curvas de bordo (ciclos fecados de poligonais). E e V são arestas e vértices que possuem, de modo geral, informações de vizinhança baseada na estrutura *half winged edge*[7].

Um aspecto que merece atenção é a diferença entre *pontos* e *vértices*. Seus papéis são, respectivamente, representar a malha geométrico e topologicamente. Podemos dizer também que o *ponto* é a realização geométrica do *vértice*. Na figura 5.4 mostramos um exemplo em que distinção entre geometria e topologia resolve ambigüidades: uma curva que é geometricamente não-variedade mas topologicamente pode representar uma curva ou duas.

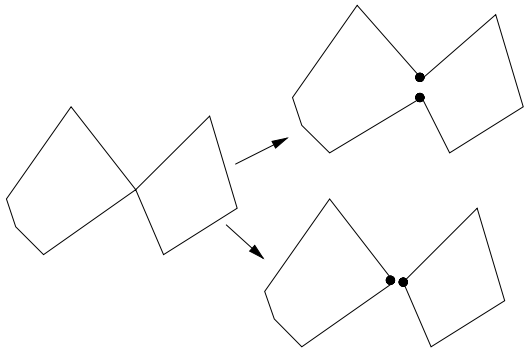


Figura 5.4: (à esquerda)- Representação geométrica de um bordo; (à direita)- Duas representacões topológicas do mesmo bordo.

Observe que um *ponto* pode ser associado a mais de um vértice e

por fins de otimização em consultas, que veremos mais adiante, cada ponto mantém uma referência para uma lista de vértices que aponta para ele mesmo(veja figura 5.5).

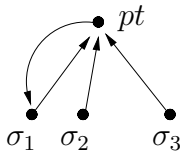


Figura 5.5: Relação ponto–vértice

A fim de construirmos a malha iterativamente em métodos de avanço de frente podemos agora usar a teoria de *handlebody* e nossa representação de malha introduzindo assim, os operadores de *handle*. Nosso objetivo é imitar o processo de construção descrito no Teorema 3.

Sejam e_{ij} uma aresta, p_i, p_j seus respectivos pontos e σ_i, σ_j seus respectivos vértices.

Definição 2. *Duas arestas e_{ij} e e_{kl} são geometricamente (resp. topologicamente) coincidentes se elas têm a mesma geometria (resp. topologia), isto é, $\{p_i, p_j\} = \{p_k, p_l\}$ (resp. $\{\sigma_i, \sigma_j\} = \{\sigma_k, \sigma_l\}$).*

Definição 3. *Duas arestas e_{ij} e e_{kl} são topologicamente semi-coincidentes se $\#(\{\sigma_i, \sigma_j\} \cap \{\sigma_k, \sigma_l\}) = 1$.*

Definição 4. *Duas arestas e_{ij} e e_{kl} são topologicamente não-coincidentes se $\#(\{\sigma_i, \sigma_j\} \cap \{\sigma_k, \sigma_l\}) = 0$*

Note que se duas arestas são topologicamente coincidentes então elas são também geometricamente coincidentes.

Podemos então definir quatro tipos de operadores discretos de malha:

1. O *operador de handle* do tipo-0 cria um novo triângulo. Ele sempre gera uma nova componente conexa;

2. O *operator de handle* do tipo-1 identifica duas arestas geometricamente coincidentes no bordo mas que são topologicamente

não-coincidentes. As arestas podem estar na mesma curva de bordo ou em diferentes. No primeiro caso a curva é dividida em duas. No segundo as curvas são unidas em uma só.

3. O *operados de handle* do tipo-2 identifica duas arestas geometrica e topologicamente coincidentes. Este operador fecha uma curva.

4. O *homeomorfismo* identifica duas arestas geometricamente semi-coincidentes. Ele faz um "zip", i.e., o tamanho da curva de bordo é decrementada em duas arestas. Não há mudança topológica na superfície.

Agora vamos definir as API's topológicas *criar* e *colar*. Elas irão implementar os operadores de *handle* descritos acima e serão usados para construir a malha em um algoritmo de avanço de frente.

A primeira rotina *criar*(p_0, p_1, p_2), recebe três pontos como entrada e cria uma face triangular, três arestas, três vértices e uma curva de bordo. Essa API é equivalente ao operador de *handle* do tipo-0.

A segunda rotina, *colar*(e_{ij}, e_{kl}), recebe duas arestas geometricamente coincidentes e trata internamente os três últimos casos de operadores de *handle* descritos acima. Ele atualiza a estrutura de dados da malha juntando vértices e arestas apropriadamente. Também mantém a lista de curvas de bordo.

Como parte do processo de Fotografia 3D no *software* deste curso, vamos agora para o estudo de caso, que é o algoritmo Ball-Pivoting.

5.2.3 *Ball-Pivoting*

No processo de Fotografia 3D o Algoritmo Ball-Pivoting (BPA)[7] tem-se destacado durante a etapa de construção de malhas principalmente por sua elegância e simplicidade. Dentre as mais importantes contribuições para reconstrução de superfícies, podemos dizer que o BPA:

- Gera uma malha que é um subconjunto do *Alpha Shapes* [23], herdando suas propriedades;

- É apropriado para dados em grande escala porque é bastante eficiente em termos de tempo de execução e em armazenamento;

- É robusto quando se depara com ruídos.

O BPA, como já mencionado, é um algoritmo de avanço de frente. Descrevendo-o de forma suscinta inicialmente sua frente é um ciclo composto apenas pelas arestas de um triângulo que pertence à malha e que foi gerado por uma rotina que denominaremos de *primeira-escolha*. Novos triângulos são associadados à frente por um passo chamado de *passo-principal*, e consiste em nada mais do que o critério geométrico para os algoritmos de avanço de frente. Neste passo giramos uma bolinha de raio fixo (circunscrita a um triângulo da malha) em torno de uma das arestas da frente até tocar um novo ponto da amostragem (cf. fig 5.6).

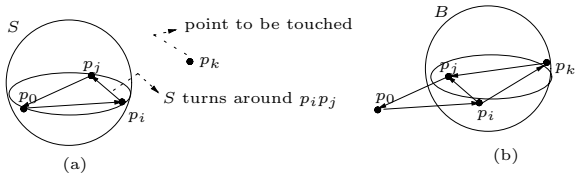


Figura 5.6: *Ball-Pivoting* intuitivo. Inicialmente a frente é formada pela poligonal $p_1p_2p_3$ (a). Após o *passo-principal* a frente muda para a poligonal $p_1p_2p_3p_4$ (b).

Geometria

Vamos primeiramente detalhar a parte geométrica do BPA. Discutiremos as rotinas *primeira-escolha* e *passo-principal* e detalharemos a estrutura de dados de aceleração. Mas antes, façamos algumas definições importantes a fim caracterizar os triângulos da malha gerada pelo BPA e definamos uma entrada de dados mais apropriada.

Definição 5. *Um triângulo Δ_T α-exposto é aquele que possui uma α-bola circunscrita e vazia.*

Sabemos que todo triângulo possui no máximo duas α-bolas circunscritas. Mas como o triângulo está orientado então podemos escolher a α-bola, que iremos denominar por $B_\alpha(\Delta_T)$, cujo centro está no semi-espaço positivo determinado pelo plano suporte do triângulo (fig. 5.7).

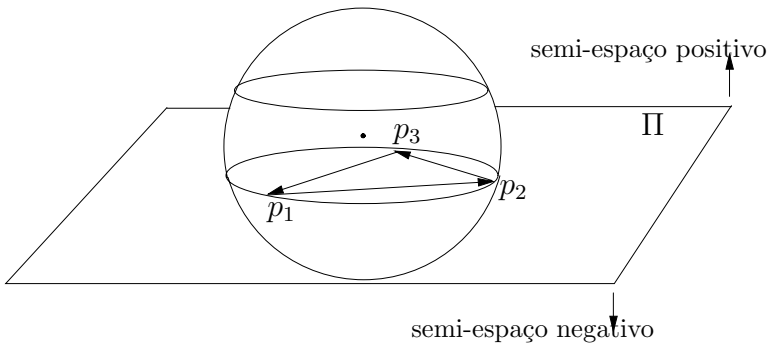


Figura 5.7: Semi-espaços positivo e negativo

Vamos acrescentar um atributo ao conjunto $P \subset \mathbb{R}^3$. Como estes pontos pertencem a uma superfície S consideraremos como dados de entrada $\tilde{P} = \{(p, n) \in \mathbb{R}^3 \times \mathbb{S}^2 \quad | \quad p \in P \text{ e } n \text{ é normal a } S \text{ em } p\}$. No entanto para efeitos de implementação, um vetor n que aponte "fora" da superfície S já é suficiente.

Seja $T = \{p_1, p_2, p_3\}, p_i \in \tilde{P}$. O 2-simplexo σ_T tem duas orientações, que são $p_1p_2p_3$ ou $p_3p_2p_1$ e que são associadas às duas normais opostas $(p_2 - p_1) \wedge (p_3 - p_2)$ e $(p_2 - p_3) \wedge (p_1 - p_2)$ respectivamente. Vamos definir triângulo orientado Δ_T como o 2-simplexo σ_T com uma orientação definida.

Definição 6. *Dizemos que Δ_T é consistente quando $\langle n_i, N\rangle > 0, \forall i$, onde N é a normal de Δ_T.*

Feitas as definições, vamos agora detalhar as duas principais rotinas geométricas.

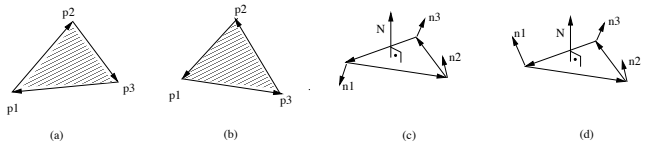


Figura 5.8: (a) e (b) são orientações possíveis, (c) e (d) triângulo não-consistente e consistente respectivamente.

Primeira-escolha

Essa rotina procura por um triângulo Δ_T α-exposto afim de iniciar o processo de avanço de frente descrito já descrito. O problema que queremos resolver é: dado $\tilde{P}$, achar $p_1, p_2, p_3 \in \tilde{P}$ tais que $\Delta_{p_1p_2p_3}$ é α-exposto e consistente. Para resolvê-lo adotamos uma heurística que descrevemos em seguida.

Primeiramente fazemos uma partição uniforme do espaço células M_i, ou seja, $\tilde{P} \subset \cup M_i$, com cada célula contendo uma lista de pontos (confira a implementação no tópico de *aceleração*). Escolhemos um ponto $q \in M_i$ e criamos uma lista de segmentos $\overrightarrow{qp_l}$, $p_l \in M_i$. Daí, para cada $\overrightarrow{qp_l}, \overrightarrow{qp_m}$ testamos a existência de $B_\alpha(qp_lp_m)$, a consistência de qp_lp_m e verificamos se $B_\alpha(qp_lp_m)$ é α-exposto. Veja abaixo como fica o pseudo-algoritmo refprimeira-escolha.

Programa 7 primeira-escolha

```
for all M_i do
    q ← primeiro ponto em M_i
    L ← gerar lista de segmentos qp_l, p_l ∈ M_i
    for all qp_l, qp_m ∈ L; l ≠ m do
        if (existe B_α(qp_ip_j) e qp_ip_j é consistente e B_α(qp_ip_j) é α-
        exposto) then
            return qp_ip_j
return NULO;
```

Obviamente, o algoritmo pára assim que encontra o primeiro triângulo que satisfaz as condições.

Critério Geométrico: O Passo-principal

O BPA constrói a malha criando triângulos que são circunscritos a uma esfera vazia (i.e. são α-expostos). De fato, isto é conseqüência imediata do *passo-principal*, critério geométrico adotado no caso do BPA como algoritmo de avanço de frente. É o que vamos descrever agora.

Sejam $\Delta_T = p_1p_2p_3$ e C_{123} o centro de $B_\alpha(\Delta_T)$. O *passo-principal* consiste em girar a esfera $B_\alpha(\Delta_T)$ no sentido do vetor normal de Δ_T em torno de um de seus lados, digamos p_1p_3, até tocar algum outro ponto de $\tilde{P}$.

O primeiro ponto a ser tocado tem o menor ângulo de rotação. Logo, precisamos calcular para todos os pontos de $V(m)$ [1], onde m é o ponto médio de p_3p_1, o menor ângulo formado entre os segmentos mC_{123} e mC_{i13}, sendo C_{i13} o centro de $B_\alpha(\Delta_{p_ip_1p_3})$.

Seja $p_i \in \tilde{P}$. A trajetória descrita por C_{123} é um círculo de centro $m = \frac{p_3+p_1}{2}$ e raio $|C_{123} - m|$. Usando a Proposição 1 do próximo tópico que detalha do cálculo do centro da esfera $B_\alpha(\Delta_T)$, determinamos o centro C_{i13} do triângulo $p_ip_2p_3$ e através de produto vetorial encontramos o ângulo orientado $C_{123}\hat{m}C_{i13}$ tendo o vetor $\overrightarrow{p_3p_1}$ como referência. Veja então como fica assim descrito no pseudo-algoritmo 8.

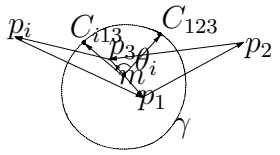


Figura 5.9: Trajetória γ de $B_\alpha(\Delta_T)$ no passo principal.

A seguir datalharemos a cálculo do centro de $B_\alpha(\Delta_T)$. Se o leitor desejar, poderá omitir este tópico sem nenhum prejuízo a compreensão do restante do capítulo.

[1] $V(m)$ é uma certa vizinhaça de m que depende de α. Confira o tópico sobre aceleração para maiores detalhes

Programa 8 passo-principal

Entrada(M,$B_\alpha(p_1p_2p_3), m, C_{123}$)
Saída ($p_0 \in V(m)$ tal que $C_{123}\hat{m}C_{031}$ é mínimo)

$p_0 \leftarrow NULO$
$minang \leftarrow 2\pi$
for all $p_i \in V(m)$ **do**
  calcule o centro de C_{i13} de $B_\alpha(p_ip_1p_3)$
  calcule o angulo θ_i entre $\overrightarrow{mC_{123}}$ e $\overrightarrow{mC_{i23}}$
  if $\theta_i < minang$ **then**
    $p_0 \leftarrow p_i$
return p_0

Determinando o centro de $B_\alpha(\Delta_T)$

Usaremos como ferramenta somente geometria euclidiana. Mas antes enunciemos o seguinte lema:

Lema 1. *Sejam $\Delta_T = p_1p_2p_3$ um triângulo orientado e Π seu plano suporte, então $\forall p \in \Pi$ existem coeficientes λ_1, λ_2 e $\lambda_3 \in \mathbb{R}$ tais que $\lambda_1p_1 + \lambda_2p_2 + \lambda_3p_3 = p$ com $\lambda_1 + \lambda_2 + \lambda_3 = 1$. Além disso, esses coeficientes são invariantes por transformaçoes afins e $\lambda_1 = \frac{S_{pp_2p_3}}{S_{p_1p_2p_3}}, \lambda_2 = \frac{S_{pp_3p_1}}{S_{p_1p_2p_3}}$ e $\lambda_3 = \frac{S_{pp_1p_2}}{S_{p_1p_2p_3}}$, onde S_{abc} é a área do triângulo orientado abc.*

Demonstração: Π é parametrizado como $p = p_1 + \lambda_2(p_2 - p_1) + \lambda_3(p_3 - p_1) = (1 - \lambda_2 - \lambda_3)p_1 + \lambda_2(p_2) + \lambda_3(p_3)$. Fazendo $\lambda_1 = 1 - \lambda_2 - \lambda_3$, temos que $\lambda_1 + \lambda_2 + \lambda_3 = 1$ e provamos a primeira parte.

Seja $T(p) = A(p) + t_0$ uma tranformção afim, com A linear e t_0 um vetor fixo. Logo:

$$\begin{aligned} T(\lambda_1p_1 + \lambda_2p_2 + \lambda_3p_3) = \\ & A(\lambda_1p_1 + \lambda_2p_2 + \lambda_3p_3) + t_0 = \\ & \lambda_1A(p_1) + \lambda_2A(p_2) + \lambda_3A(p_3) + t_0 = \\ & \lambda_1(A(p_1) + t_0) + \lambda_2(A(p_2) + t_0) + \lambda_3(A(p_3) + t_0) = \\ & \lambda_1T(p_1) + \lambda_2T(p_2) + \lambda_3T(p_3). \end{aligned}$$

provando a segunda parte. Considerando uma isometria $I(p) = p'$ (rotação + translação) que leva Π em $z = 1$ e utilizando o resultado da primeira parte, encontramos λ_i solucionando o seguinte sistema linear,

$$\begin{pmatrix} p'_1 & p'_2 & p'_3 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix} = p'$$

e cuja solução é $\lambda_1 = \frac{|p'p'_2p'_3|}{|p'_1p'_2p'_3|} = \frac{S_{pp_2p_3}}{S_{p_1p_2p_3}}$. Obtemos analogamente λ_2 e λ_3 conforme o enunciado ■

Seja C o centro de $B_\alpha(\Delta_T)$ e sejam d, O e $\vec{N}$ a distância de C ao plano suporte de Δ_T, o seu circuncentro e sua normal unitária de Δ_T, respectivamente (fig 5.10). Então:

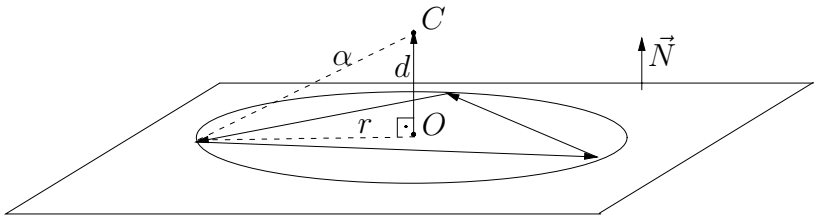


Figura 5.10: Teorema de Pitágoras

- Pelo Teorema de Pitágoras $d = \sqrt{r^2 - \alpha^2}$, onde $r = \frac{|a||b||c|}{4S_{p_1p_2p_3}}$ é o raio do círculo circunscrito de Δ_T e $a = (p_2 - p_1), b = (p_3 - p_2), c = (p_1 - p_3)$ são os seus lados;
- $\vec{N} = \frac{a \times b}{|a \times b|}$;
- Pela Proposição 1, $O = \lambda_1 p_1 + \lambda_2 p_2 + \lambda_3 p_3$ e $\lambda_1 = \frac{\|a \times (O - p_3)\|}{\|a \times b\|}$, $\lambda_2 = \frac{\|b \times (O - p_1)\|}{\|a \times b\|}$, $\lambda_3 = \frac{\|c \times (O - p_2)\|}{\|a \times b\|}$;
- $C = O + d\vec{N}$;

Observe que esses quatro passos oferecem uma maneira algorítmica de encontrar o centro de $B_\alpha(\Delta_T)$ e, de forma suscinta, completa a prova da seguinte proposição:

Proposição 1. *As coordenadas do centro de $B_\alpha(\Delta_T)$ são completamente determinadas pelas coordenadas dos pontos de Δ_T e sua orientação.*

Aceleração

Uma das características do algoritmo é o fato de ser local e para tanto, fazemos uma subdivisão uniforme do espaço em voxels de lado 2α. Em termos de implementação estes voxels são armazenados numa matriz tridimensional porque é bastante cômodo identificar os índices da matriz de um ponto $p \in \tilde{P}$ fazendo uma conta simples. A seguir faremos em detalhes este cálculo para o caso 2D que é análogo ao 3D.

Sejam $(x_0, y_0), (x_1, y_1)$ o canto inferior esquerdo e o canto superior direito da caixa envoltória B dos pontos de $\tilde{P}$, respectivamente. O espaço que queremos dividir de maneira uniforme é um retângulo R que tem as seguintes propriedades:

- O canto inferior esquerdo é (x_0, y_0)
- $B \subset R$
- O comprimento é $n(2\alpha)$ e a altura é $m(2\alpha)$

onde $m = \lfloor \frac{y_1 - y_0}{2\alpha} \rfloor + 1$ e $n = \lfloor \frac{x_1 - x_0}{2\alpha} \rfloor + 1$. R é interpretado como uma matriz de m linhas e n colunas e seus elementos são identificados por R_{ij}.

Seja $p = (x, y) \in R$. Então $p \in R_{ij}$ onde $i = \lfloor \frac{y - y_0}{2\alpha} \rfloor + 1$ e $j = \lfloor \frac{x - x_0}{2\alpha} \rfloor + 1$. Este cálculo que identifica a célula R_{ij} a que p pertence é a principal operação que usaremos na matriz R.

Esta é a estrutura em que os dados de entrada, $\tilde{P}$, estão configurados. Note que é bastante simples de implementar e o tempo gasto para sua montagem é $O(|\tilde{P}|)$.

No caso tridimensional, teríamos uma estrutura de matriz M de voxels que armazena os pontos de $\tilde{P}$ e que tem a capacidade de identificar em tempo constante o elemento M_{ijk} a que pertence um ponto do espaço.

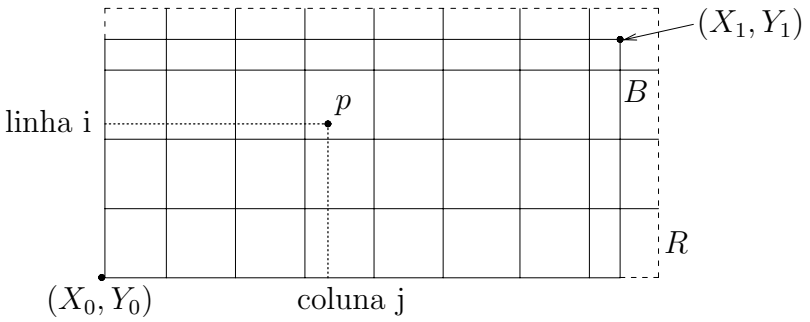


Figura 5.11: Representação da matriz R em que os pontos estarão organizados.

Seja $p \in |M|$, onde $|M| \subset \mathbb{R}^3$ é o espaço coberto pelos elementos de M. Então $V(p)$ é uma vizinhança de p definida como a união de todos os seus 27 voxels vizinhos, inclusive o voxel em que p está contido.

Note que, observando a rotina *passo-principal* a vizinhança é grande o suficiente para conter todas os possíveis pontos, isto é, se $p_0 \notin V(m)$ então $|m - p_0| > 2\alpha$ e não existe $B_\alpha(p_0p_1p_2)$. Assim, o espaçamento 2α do grid M está bem definido.

O uso dessa estrutura de aceleração é um dos motivos do BPA ter comportamento assintótico linear em pontos uniformemente amostrados na superfície. Isto ressalta ainda mais sua importância dentre os algoritmos de reconstrução de superfícies.

O Algoritmo

Reproduzimos logo adiante no pseudo-algoritmo 9 o BPA como desctrito em Bernadini et al. [7] com algumas alterações a fim de encaixá-lo dentro do nosso *framework* topológico de algoritmos de avanço de frente por meio das API's *criar* e *colar*:

Mais precisamente a construção da malha ocorre da seguinte forma: de uma nuvem de pontos $\tilde{P}$, inicialmente a malha M é apenas um triângulo que foi gerado a partir do algoritmo *primeira-escolha* cuja fronteira são seus 3 lados. Este passo corresponde ao operador

Programa 9 Ball-Pivoting

```
while (verdadeiro) do
  while ((e_ij ← primeiro elemento de Q ≠ NULO) do
    p_k ← passo-principal(e_ij)
    if (p_k ≠ ∅) e (nao_usado(p_k) ou indice(p_k)) then
      criar(p_i, p_j, p_k)
      colar(e_ij, e_ji)
      if e_ki ∈ B then
        colar(e_ik, e_ki)
      if e_jk ∈ B then
        colar(e_kj, e_jk)
    else
      marque-como-bordo(e_ij)
  if (p_i, p_j, p_k)=primeira-escolha() then
    criar(p_i, p_j, p_k)
  else
    break;
```

do tipo-0. Criamos então uma fila de arestas, que chamaremos de frente Q, e conterá todas as arestas de fronteira da malha num determinado instante. Retiramos o primeiro elemento de Q e executamos o *passo-principal*. Caso o *passo-principal* retorne um ponto inexistente então a aresta é de bordo, caso contrário criamos mais três arestas, que passam por operações topológicas de colagem.

Há duas consultas que são fundamentais para o algoritmo. A primeira é a rotina *indice* cujo objetivo é saber se um ponto é de fronteira ou do interior da malha. Para isso, atribuímos um valor inteiro a cada ponto que indicará o número de arestas vizinhas sem um par oposto. Vejamos, então, as propriedades do índice que respondem a nossa consulta:

- Um ponto p do interior da malha tem índice zero e é único, isto é, não existe outro vértice de bordo que aponte para p.
- Um ponto p com índice $n \geq 2$ tem $\frac{n}{2}$ vértices de bordo distintos apontando para p.

Na segunda consulta, dado e_{ij}, queremos verificar se $e_{ji} \in B$.

Uma das características da estrutura *half winged edge* é a de que permite resolvermos consultas simples como a de encontrar todos os vértices vizinhos de um dado vértice. Logo, a solução natural para a segunda consulta é procurar na vizinhança de $\sigma_j \rightarrow pt$ por um vértice σ_k tal que $\sigma_k \rightarrow pt = \sigma_i \rightarrow pt$. Outra maneira seria varrer todas as curvas de bordo. Ambas possuem vantagens e desvantagens: na primeira há maior consumo de memória já que precisaríamos, como visto antes, armazenar em cada *ponto* uma lista de vértices, e menor tempo de busca enquanto Na segunda maneira ocorre justamente o contrário. Note que a busca é sempre de natureza geométrica.

O *loop* mais interno trata dos outros tipos de operadores (tipo-1 e tipo-2). O caso mais interessante é quando todas as rotinas topológicas são chamadas nesse bloco(i.e. uma chamada à rotina **criar** e três chamadas à rotina **colar**). Neste caso, uma curva de bordo está sendo fechada. A chamada **criar** gera uma nova componente conexa e uma nova curva de bordo (handle tipo-0). A primeira chamada em **colar** junta este novo bordo com a frente da malha em e_{ij} (handle tipo-1). Note que o propósito desse passo é juntar duas arestas que são geometricamente coincidentes em uma única aresta (essas aresta são sempre coincidentes por construção como consequência do *passo-principal*).

A segunda chamada e a terceira para **colar** ocorrem somente quando as outras duas aresta do triângulo associado encontram seus respectivos pares, geometricmente coincidentes, que já se encontram no bordo. A segunda chamada corresponde a um homeomorfismo (zip) e, depois, a terceira chamada fecha a curva de bordo (handle tipo-2). De fato, É possível mostar que o handle do tipo-2 acontece somente quando todas as três operações de colagem são chamadas na mesma iteração do *loop while*. A figura 5.12 ilustra co detalhes estes passos do algoritmo.

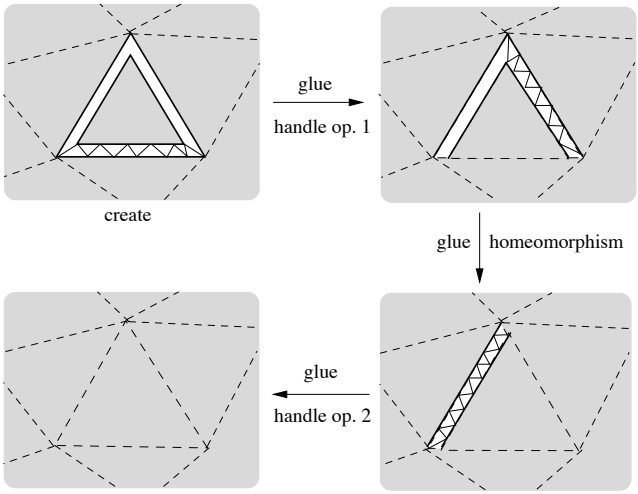


Figura 5.12: Sequência do operador de handle fechando uma curva de bordo num algoritmo de avanço de frente.

5.3 Poligonização de Superfícies Implícitas

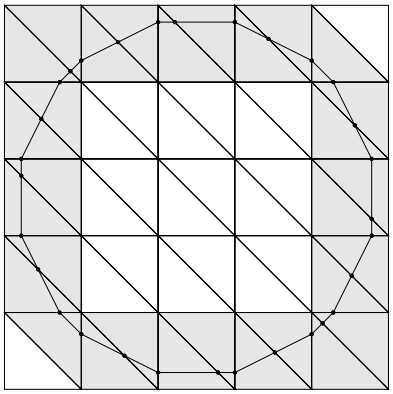
Métodos de poligonização para superfícies implícitas geralmente empregam uma decomposição do espaço ambiente. Por esta razão, eles são chamados de métodos extrínsecos de decomposição. O trabalho seminal de Allgower, [4], é um exemplo clássico dessa categoria.

Uma visão intuitiva do algoritmo de poligonização é descrito na próxima seção.

5.3.1 Algoritmo de Poligonização Extrínseco

Para gerar a variedade combinatória $\widetilde{M}$ que aproxime uma superfície implícita M vamos proceder da seguinte maneira: primeiro determinamos todas as células que intersectam M. Estas células constituem uma enumeração espacial de M.

Dado que esta enumeração espacial, digamos C, associada a M já



(a)

Figura 5.13: Ilustração do algoritmo de poligonização

tenha sido computada, $\widetilde{M}$ será gerada utilizando seguinte método:

Varrer e classificar as células c_i de C.

Para todas as células c_j intersectando M

Calcule a interseção de M e c_j.

Criar e adicionar os polígonos correspondentes a $\widetilde{M}$.

O leitor deve observar cada um dos passos a acima na Figura 5.13. A seguir vamos descrever os três principais passos do algoritmo.

Classificação de simplexos

A classificação de células depende da propiedade do ponto associada à função implícita F. É possível determinar se a célula está dentro ou fora, ou se intersecta o bordo do objeto de acordo com os valores de F nos vértices da célula. Assumindo que a subdivisão C é suficientemente fina a classificação é como segue: Se o sinal de F é negativo nos vérices de c então a célula está no interior do objeto. Se o sinal de F

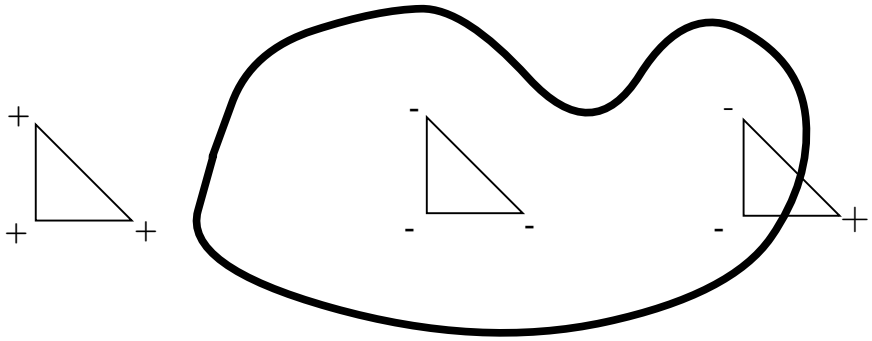


Figura 5.14: Classificação de célula

é positivo nos vértices de c então a célula está no exterior do objeto. Quando o sinal de F não é o mesmo em todos os vértices da célula então c deve intersectar o bordo do objeto. Uma célula intersectada é também chamada de *célula transversa*. Estes casos estão ilustrados na Figura 5.14.

Cálculo da interseção

Nosso problema agora é: para cada face unidimensional σ_1 da célula, verificar se M intersecta σ_1 e se for o caso, calcular $M \cap \sigma_1$. A interseção com cada aresta consiste em somente um ponto. Para verificar a interseção temos que verificar apenas o sinal de F em cada um dos dois vértices v_0 e v_1 de σ_1. Se $F(v_0)$ e $F(v_1)$ têm sinais opostos, $M = F^{-1}(0)$ intersecta a aresta σ_1. O cálculo do ponto de interseção reduz-se ao ao problema de encontrar raiz. Se $\gamma(t) = tv_1 + (1-t)v_0$ é a equação paramétrica da linha suporte da aresta, devemos encontrar as raízes da equação $F(\gamma(t)) = 0$ no intervalo $(0, 1)$.

Geração do Polígono

Polígonos são criados em dois passos: primeiro, os vértices do polígono que já têm sido calculados pela interseção de faces 1-dimensional de cada célula c com a superfície implícita M. Agora

os polígonos são formados pela ordenação dos vértices,induzidos de forma consistente pela orientação da triangulação original do espaço.

Note que este método toma vantagem da enumeração da subdivisão do espaço associado ao objeto implícito a fim de executar tanto a amostragem como a estruturação.

5.4 Métodos de Poligonização Extrínsecos

Em decorrência dos os três critérios discutidos na seção anterior, os métodos de poligonização que usam decomposição extrínseca do espaço são subdivididos de acordo com:

(i) A classe do complexo celular empregado no espaço de decomposição,

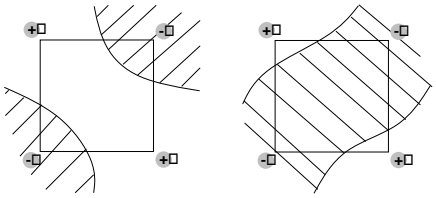
(ii) a estratégia de rastreamento adotada para varrer as as células intersectadas, e

(iii) o tipo de subdivisão usado.

Com relação ao domínio de decomposição, os métodos de poligonização podem ser: simpliciais ou não-simpliciais. *Métodos simpliciais* triangulam o domínio de F formando um complexo simplicial. *Métodos não-simpliciais* subdividem o dominio de F construindo um complexo celular geral.

Com relação à estratégia de varredura, os métodos de poligonização podem ser: de continuação ou de varredura total. *Métodos de continuação* iniciam com uma célula seminal que intercecta a superfície e busca células vizinhas para determinar quais também são intersectadas. Este processo é repetido até que todas as células transversas sejam encontradas. *Métodos de varredura total* visitam e classificam todos os elementos do complexo celular para encontrar quais células intersectam a superfície.

Com relação ao tipo de subdivisão espacial os métodos de poligonização podem ser: uniforme ou adaptativo. *Métodos uniformes* subdividem o espaço em intervalos regulares, gerando uma decomposição



(a)

Figura 5.15: Poligonização Ambígua.

de resolução fixa. *Métodos adaptativos* particionam o espaço irregularmente de tal forma que o resultado da decomposição é mais fino onde mais detalhe for preciso.

Vamos discutir brevemente cada um dos métodos acima.

5.4.1 Métodos não-simpliciais

O método não-simplicial mais popular de poligonização é o "marching cubes algorithm" [47]. Ele emprega uma subdivisão cúbica uniforme do espaço. O método consiste essencialmente em três passos: o complexo celular é varrido para identificar os cubos transversos; a topologia dos polígonos em cada cubo é determinada através de uma "look-up table procedure"; e por fim, a localização dos vértices são calculadas por interpolação.

Um método similar foi desenvolvido independentemente por [90] e [62]. A diferença entre eles está no modo de derivação da topologia dos polígonos.

Métodos não-simpliciais são rápidos e muito simples de implementar. A principal desvantagem é que eles não são robustos. Isto acontece porque a interseção de $\widetilde{F}^{-1}(0)$ e uma célula retangular não pode ser unicamente determinada. Como consequência, decisões inconsistentes podem produzir buracos na poligonização. A figura 5.15 mostra um exemplo bidimendional onde situações ambíguas podem ocorrer.

Este problema pode ser corrigido usando diferentes artifícios. Um

artifício sem ambigüidades ad-hoc, proposto por [5], consiste em verificar o centro das faces do cubo. Isto alivia o problema usando somente amostras extras. A solução mais correta consiste em escolher um método globalmente consistente para subdivisão cada célula intersectada [56, 58, 20].

Um método de poligonização que segue a mesma estrutura algorítmica do método de *Marching Cubes* clássico e que também faz amostragem sensível a detalhes está descrito em [41]. Tal método emprega funções de distância relativas às direções coordenadas principais para calcular a interseção exata de uma representação volumétrica amostrada.

5.4.2 Simplicial Methods

O algoritmo de poligonização descrito na seção 5.3.1 está nesta categoria.

Métodos simpliciais de poligonização são teoricamente melhores porque dependem da estrutura linear por partes induzida pela triangulação do domínio. A formulação teórica desta teoria tem sido estudada por anos na área de topologia algébrica. Este fato é refletido nas discussões teóricas que antecedem a descrição do algoritmo de poligonização, incluindo o teorema da triangulação para superfícies diferenciáveis.

O primeiro trabalho em métodos de poligonização simplicial foi escrito por Allgower [4, 3]. Seu método é baseado na triangulação de Coxeter-Freudenthal e ele usa um esquema de pivoteamento para mover-se de um simplexo para outro [2]. A aproximação simplicial de uma superfície implícita em cada célula é obtida calculando-se o núcleo de $\widetilde{F}$ sobre o simplexo. Isto acarreta numa solução linear de sistemas de equações.

Uma variação do método anterior foi sugerido por [16]. Seu principal objetivo foi melhorar a performance.

5.4.3 Métodos de Continuação

Métodos de continuação exploram a coerência da variedade PL $\widetilde{M}$ para reduzir o espaço de busca das células transversas [1], [93].

Dada uma superfície compacta M e uma célula c que intersecta M, é possível gerar uma variedade compbinatória fechada $\widetilde{M}$ que aproxima M visitando somente as células transversas em C. O método consiste em processar de forma sistemática os vizinhos de c. Ele mantém uma lista W de células ativas. Vizinhos das células em W são inseridos na lista, e a célula c é mantida em W enquanto existem vizinhos de c a serem inseridos. Quando a lista W está vazia todas as células transversas têm sido processadas.

Uma característica dos métodos de continuação é a inicialização. No caso da poligonização precisamos de um ponto em cada componente conexa do objeto. Isto não é usualmente um problema porque ele acontece naturalmente na maioria das aplicações.

5.4.4 Malhas Adaptativas

Como não temos, em geral, a *priori*, estimativas da vizinhança tubular da variedade implícita, precisamos iniciar com uma decomposição suficientemente fina do espaço. Isto aumenta o número de células e conseqüentemente o número de polígonos no modelo final. Isto pode ser evitado em parte refinando-se as células decompostas somente onde for necessário. Isto é a base de métodos adaptativos.

Métodos adaptativos criam decomposições espaciais que são sensíveis para alguma característica do objeto. Eles começam com uma subdivisão inicial do domínio do objeto, e refinam-no recursivamente até um critério de adaptação ser alcançado. Isto depende muito da aplicação. No caso da poligonização implícita de superfícies pode-se medir a fidelidade de uma aproximação PL. Para análise de elementos finitos está freqüentemente relacionado com propriedades do material do objeto.

Existem dois principais métodos para adaptatividade: A pricipal diferença está na forma que eles restringem a subdivisão. Uma restringe as células da decomposição, enquanto que a outra restringe as arestas da poligonização.

O primeiro método subdivide células independemente restringindo arestas do polígono baseado num critério de refinamento. Isto garante que na poligonização resultante não haverá discrepâncias entre níveis diferentes de resolução. Uma implementação simplicial deste método é dada em [86]. Outra similar em [35]. Uma variante

desse método que refina a poligonização como pós-processamento foi sugerida por [5]. Uma implementação desse algoritmo é encotrada em [88]. Uma desvantagem dessa solução é que ele pode falhar em detectar alguns detalhes da superfície, já que ele atua somente na poligonização.

O segundo método gera uma árvore restrita. Afim de manter a estrutura da subdivisão balanceada, este método executa repetidos passos de refinamento modificando em cada passo uma célula somente. Sempre que a célula for dividida, seus vizinhos são restringidos a manterem o nível imediatamente acima ou abaixo. Uma implementação desse algoritmo baseado em *octrees* foi introduzido por [10]. [31] propôs uma uma versão desse algoritmo para complexos simpliciais. Um método geral de **tesselation** adaptativa para superfícies implícitas e paramétricas que usa uma árvore binária restrita para construção está em [87].

Como temos visto na Seção 5.3.1 métodos de poligonização extrínseca precisam iniciar com uma decomposição do espaço euclidiano que captura a topologia da forma implícita. Uma forma de fazê-lo é por meio de uma função de Morse definida no domínio de F. Uma poligonização adaptativa para superfícies dinâmicas baseada no mesmo princípio está em [81].

Apêndice A

Um Sistema de Fotografia 3D

Nesse apêndice apresentamos um sistema de Fotografia 3D desenvolvido no Laboratório VISGRAF nos últimos anos. A arquitetura do sistema foi concebida para facilitar a expereimentação e comparação entre os diversos métodos da área. Esse sistema usa a abordagem de reconstrução baseada no espaço da imagem, descrita no Capítulo 3.

Além desse sistema, implementamos tambem programas de Fotografia 3D usando a abordagem de reconstrução no espaço da cena [57].

Esse software será oportunamente disponibilizado para a comunidade acadêmica.

A.1 Estrutura de Arquivos

Apresentamos aqui a estrutura de diretórios utilizada no projeto de Fotografia 3D desenvolvido no Visgraf. Por convenção, os diretórios escritos em sublinhado são definidos pelo usuário durante a execução dos programas.

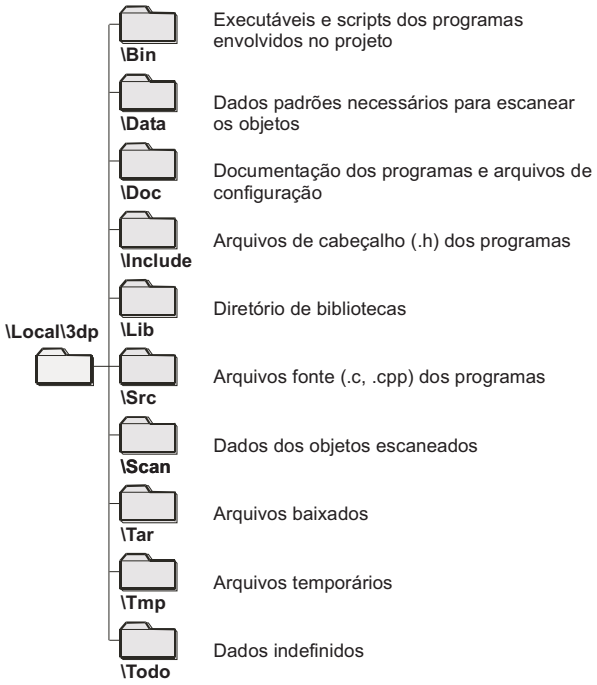


Figura A.1: Estrutura geral dos diretórios.

A.1.1 Estrutura Geral

O processo de aquisição, processamento e reconstrução de dados requer uma grande quantidade de informações que estarão estruturadas de acordo com a árvore de diretórios da figura A.1.

A.1.2 Dados Padrões (\Local\3dp\Data)

O diretório \Local\3dp\Data contém as informações padrões utilizadas para escanear qualquer objeto. Estas informações estão contidas nos diretórios \Pattern, \Code e \Calib conforme mostra a figura A.2:

Os diretórios \Pattern, \Code e \Calib contêm respectivamente

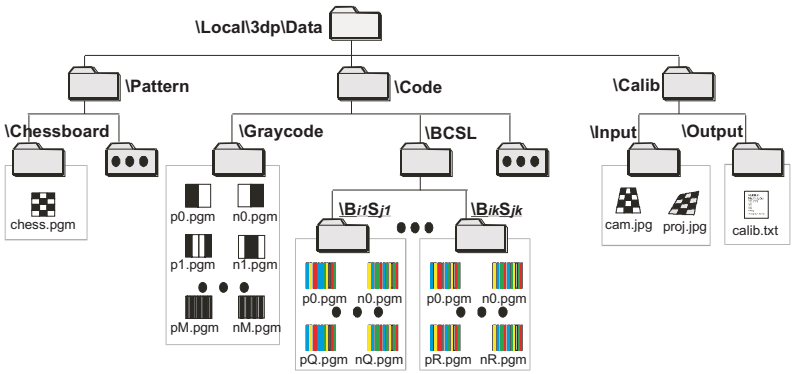


Figura A.2: Diretórios dos Dados Padrões.

as seguintes informações:

- O tipo de padrão utilizado para calibrar a câmera e o projetor (\Pattern) É possível realizar a calibração a partir de diferentes padrões. Foi implementado até o momento o padrão xadrez, cuja imagem chess.pgm encontra-se em \Pattern\Chess. As imagens dos padrões foram geradas sinteticamente para serem projetadas posteriormente.

- A codificação utilizada para relacionar as imagens da câmera e do projetor (\Code)O código que faz a correspondência entre imagens do par câmera/projetor é calculado a partir de algum padrão (como listras) projetado nas imagens. Os padrões implementados, até o momento, foram o Graycode e o (b,s)BCSL. O diretório \Code\Graycode contém as imagens dos padrões positivos e negativos nos diversos níveis de resoluções (p0.pgm, p1.pgm, ..., pM.pgm e n0.pgm, n1.pgm, ..., nM.pgm). O diretório \Code\BCSL contém um diretório para cada dupla (b,s) no qual encontram-se os padrões coloridos negativos e positivos. Todos os padrões foram gerados sinteticamente para serem projetados sobre os objetos durante sua reconstrução.

- Os dados de entrada e de saída no processo de calibração (\Calib) O diretório \Calib contém todos os dados envolvidos

no processo de calibração. Os dados de entrada são as imagens cam.jpg e proj.jpg, usadas para calibrar a câmera e o projetor respectivamente, e devem ficar no diretório \Calib\Input (também são aceitos os formatos jpeg e bmp). Estas imagens são fotografias adquiridas com uma câmera digital. O dado de saída é um arquivo texto, chamado calib.txt, que contém as matrizes de calibração, e fica armazenado em \Calib\Output. O arquivo calib.txt contém as informações de calibração padrão utilizadas para escanear qualquer objeto e é gerado pelo programa de calibração.

A.1.3 Dados Escaneados (\Local\3dp\Scan)

O diretório \Local\3dp\Scan, é formado pelos subdiretórios de cada objeto escaneado: \Objeto1, \Objeto2, ... , \ObjetoN. O nome dos diretórios dos objetos é informado pelo usuário no momento em que o objeto é escaneado. A figura A.3 mostra este esquema:

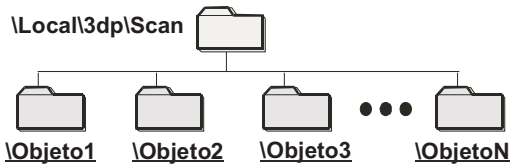


Figura A.3: Estrutura geral dos diretórios.

A.1.4 Objeto textitObj (\Local\3dp\Scan\Obj)

O diretório \Local\3dp\Scan\Obj é criado no momento que o usuário vai escanear o objeto *obj*. É composto pelos subdiretórios \Calib, \Img, \Points e \Volume, conforme a figura A.4:

O diretório \Calib contém os subdiretórios \Input e \Output que armazenarão respectivamente os dados de entrada (cam.jpg e proj.jpg) e saída (calib.txt) da calibração utilizada para o objeto obj. Se a calibração utilizada for a padrão, então os dados de entrada e saída foram copiados do diretório padrão \Local\3dp\Data\Calib.

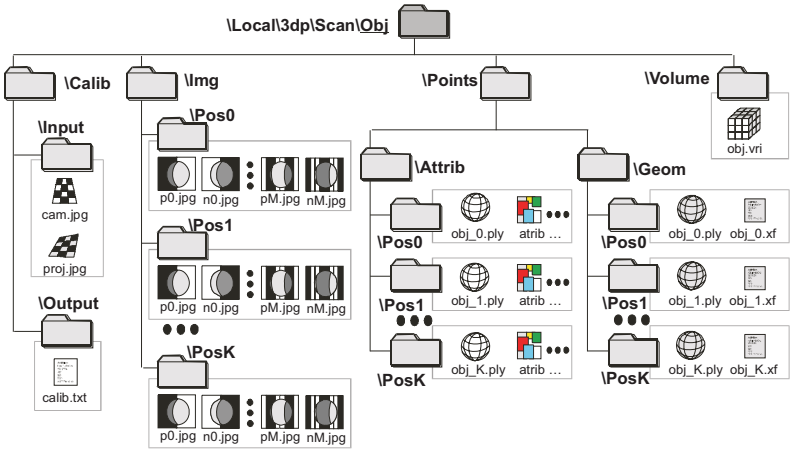


Figura A.4: Estrutura de diretórios de cada objeto.

Se a calibração for especificamente para o objeto *obj*, os dados foram gerados pelo programa de calibração.

O diretório \Img armazena todas as imagens resultantes da projeção dos padrões (armazenados em \Local\3dp\Scan\Code\Graycode ou em \Local\3dp\Scan\Code\BCSL) sobre o objeto. Estas imagens estão organizadas de acordo com a posição na qual o objeto foi fotografado. Assim, para cada parte do objeto, correspondente à posição i, é criado um diretório \Img\Pos$_i$ que armazena as imagens específicas daquela posição. Os arquivos correspondentes às imagens de cada diretório \Img\Pos$_i$ são p0.jpj, ..., pM.jpg, (projeções dos padrões positivos) e n0.jpj, ..., nM.jpg (projeções dos padrões negativos) como mostrado na figura A.4.

O diretório \Points armazena os atributos e a geometria do objeto nos subdiretórios \Points\Attrib e \Points\Geom, respectivamente. Estas informações também são armazenadas separadamente, de acordo com a posição i na qual o objeto foi fotografado, nos subdiretórios \Points\Attrib\Pos$_i$ e \Points\Geom\Pos$_i$. Cada diretório \Points\Attrib\Pos$_i$ possui um arquivo points.ply que contém a nuvem de pontos do objeto na posição i, e arquivos de atributos, como

texturas, que devem ser aplicados a esta nuvem de pontos. Cada diretório \Points\Geom\Posi tambeém possui um arquivo points.ply, que contém a nuvem de pontos do objeto na posição i (este formato difere um pouco do points.ply gravado no subdiretório de atributos), e um arquivo points.xf, que será usado para formar um dado volumétrico composto por todas as nuvens de pontos.

O subdiretório \Volume armazena o arquivo points.vri que é um dado volumétrico formado a partir de todas as nuvens de pontos.

A.2 Programa de Calibração

Este programa é utilizado para calibrar a câmera e o projetor. O processo de calibração consiste em encontrar um conjunto de parâmetros, representados por matrizes, que determinam o posicionamento, orientação, distância focal e centro de projeção da câmera e/ou do projetor. Este processo é realizado em duas etapas no programa: definição (*calibration settings*) e especificação (*calibration specification*) da calibração.

A.2.1 Definição da Calibração

Figura A.5: Settings: definição da calibração.

Na definição da calibração o usuário deve selecionar (figura A.5):

- Tipo de Calibração: é o método utilizado para calibrar a câmera e o projetor. Por enquanto o único método implementado é o de Tsai.
- Tipo de Padrão: a partir do padrão é selecionado um conjunto pré-definido de pontos, que são utilizados para calcular as matrizes de calibração da câmera e do projetor. Utilizamos padrões de xadrez. Foram implementadas duas formas de calibração: a primeira utiliza duas imagens com padrões xadrez,

uma para calibrar a câmera e outra para calibrar o projetor. A segunda forma utiliza apenas uma imagem para calibrar tanto a câmera quanto o projetor.

- Opção de Otimização: esta opção indica se será aplicada alguma otimização no processo de calibração da câmera e do projetor.

A.2.2 Especificação da Calibração

Trata-se da interface utilizada para calibrar o par câmera/projetor. Depende da definições anteriores, *Calibration Settings*. O método de Tsai requer que o usuário selecione imagens com os padrões para a calibração. É possível selecionar duas imagens, uma para calibrar a câmera e outra para calibrar o projetor, ou apenas uma imagem, utilizada tanto para a calibrar a câmera quanto para o projetor. A seleção de uma ou duas imagens depende da opção *Calibration Settings*.

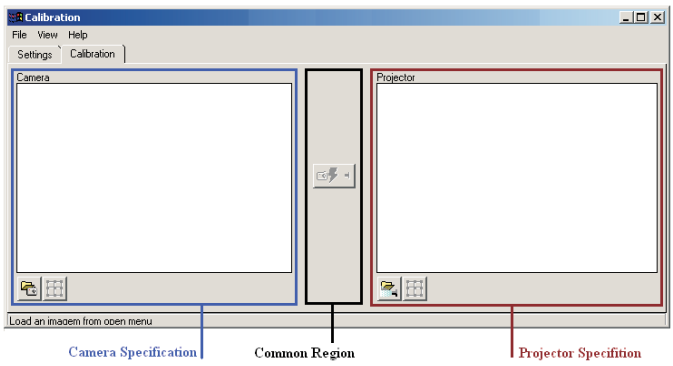


Figura A.6: Especificão da calibração com duas imagens.

Existem 3 regiões principais na área de especificação (ver figura A.6):

- Região da Câmera: formada pelo canvas da imagem da câmera (que mostra a imagem do padrão utilizada para calibrar a câmera), pelo botão utilizado para carregar a imagem da

câmera, pelo botão utilizado para selecionar os pontos ou *grade de pontos* sobre a imagem da câmera (ativado apenas após a imagem da câmera ser carregada).

- Região do Projetor: formada pelo canvas da imagem da projetor (que mostra a imagem do padrão utilizada para calibrar a projetor), pelo botão utilizado para carregar a imagem da projetor, pelo botão utilizado para selecionar os pontos ou *grade de pontos* sobre a imagem da projetor (ativado apenas após a imagem do projetor ser carregada).

- Região Comum: formada pelo botão de calibração. Este botão só é ativado depois da seleção das grades de pontos sobre as imagens de calibração da câmera e do projetor.

Se a calibração for feita com apenas uma imagem, a interface passará a exibir apenas um botão na área comum (figura A.7).

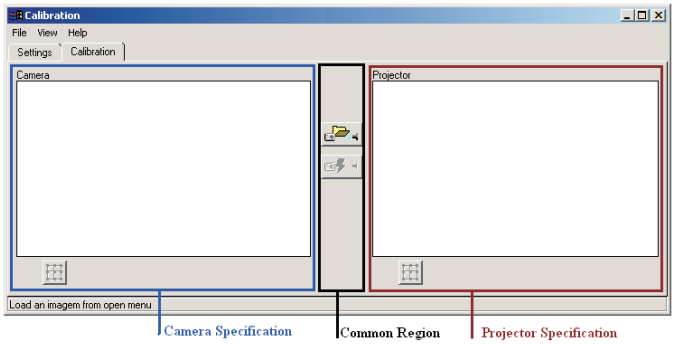


Figura A.7: Especificação da calibração com uma única imagem.

A.2.3 Calibração

O primeiro passo para calibrar câmera e projetor é abrir as imagens do padrão. Se forem utilizadas duas imagens, elas são diretamente utilizadas para calibrar a câmera e o projetor (figura A.8).

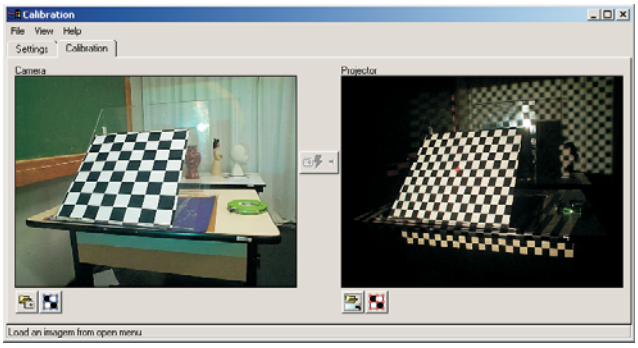


Figura A.8: Duas imagens utilizadas na calibração.

Se apenas uma imagem for utilizada, o programa aplica operações de cores a esta imagem, gerando automaticamente duas outras imagens, uma para calibrar a câmera outra para calibrar o projetor (figuras A.9 e A.10)

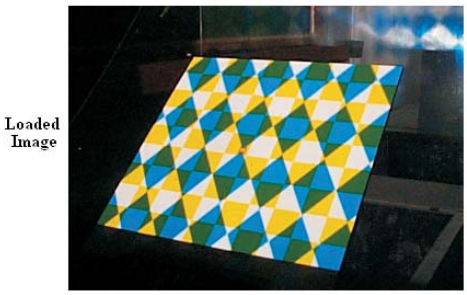


Figura A.9: Imagem única utilizada para calibrar câmera e projetor.

A.2.4 Especificando a Calibração

Após carregar as imagens, os botões das grades de pontos são ativados. Ao clicar no botão da grade de pontos da câmera, será exibida uma nova janela na qual devem ser especificados os pontos para cal-

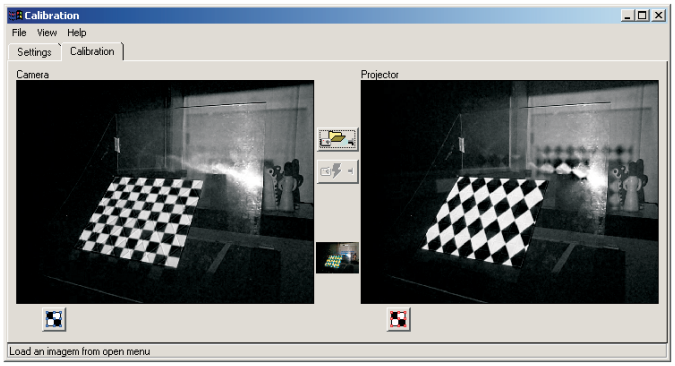


Figura A.10: Imagens calculadas a partir da única imagem.

ibrar a câmera (figura A.11).

A janela mostra a imagem com o padrão de calibração da câmera. O usuário deve selecionar 4 pontos sobre o padrão. Estes 4 pontos definirão uma região convexa R com N_x x N_y retângulos. Em seguida o usuário deve especificar os parâmetros do padrão: número de retângulos na direção X (N_x), número de parâmetros na direção Y (N_y), comprimento, em cm, de cada retângulo na direção X (D_x), comprimento, em cm, de cada retângulo na direção Y (D_y). Estes comprimentos correspondem às dimensões reais do padrão fotografado. Após especificar os 4 pontos e os parâmetros do padrão, o usuário deve selecionar o botão *Grid* para a extração de todos os pontos que serão utilizados para calcular a calibração da câmera. Agora, para confirmar a especificação, o botão *Ok* deve ser selecionado. Já para cancelar a especificação, o botão *Cancel* deve ser selecionado.

Agora o usuário deve especificar os parâmetros utilizados para calibrar o projetor. Primeiro deve selecionar o botão da grade do projetor. Uma nova janela, semelhante àquela utilizada na especificação da Câmera, será exibida e mostrará a imagem com o padrão do projetor (figura A.12). O usuário deve selecionar 4 pontos do padrão, além de mais um ponto central indicado na própria imagem do padrão. Em seguida deve especificar os seguintes parâmetros: número de retângulos na direção X (N_x), número de retângulos na direção Y (N_y), comprimento, em pixels, de cada retângulo na direção

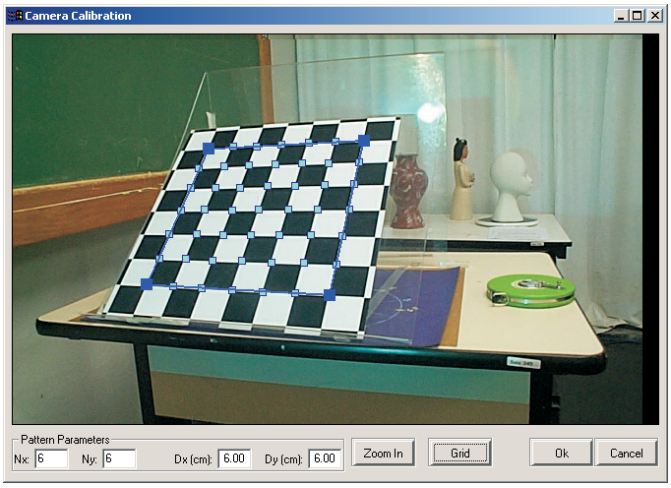


Figura A.11: Janela de especificação dos parâmetros da câmera.

X (D_x), comprimento, em pixels, de cada retângulo na direção Y (D_y). Estes comprimentos correspondem, em pixels, às dimensões do padrão projetado, a partir do qual a imagem foi gerada. Além disto, o usuário deve especificar a largura e altura , em pixels, do padrão projetado. Após estas especificações o usuário deve selecionar o botão *Grid*para extrair os pontos utilizados na calibração do projetor. Para confirmar a especificação, o botão *Ok* deve ser selecionado. Já para cancelar a especificação, o botão *Cancel* deve ser selecionado.

A.2.5 Calibrando

Uma vez que especificados os parâmetros da câmera e o projetor, o botão de calibração (localizado à área comum) é ativado (figura A.13).

O usuário deve selecioná-lo para executar a calibração. Quando este botão é selecionado, as matrizes de calibração da câmera e do projetor são calculadas. Elas são automicamente salvas no arquivo local/3dp/data/calib/output/calib.txt. O usuário pode salvá-la em outro arquivo selecionando a opção *Save Calibration...* no menu *File*.

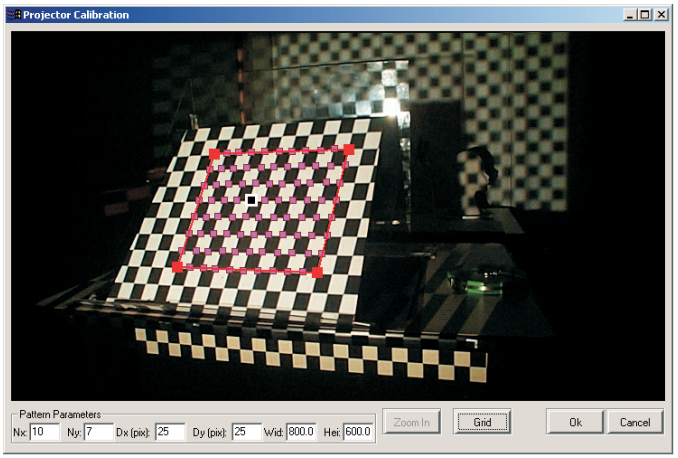


Figura A.12: Janela de especificação dos parâmetros do projetor.

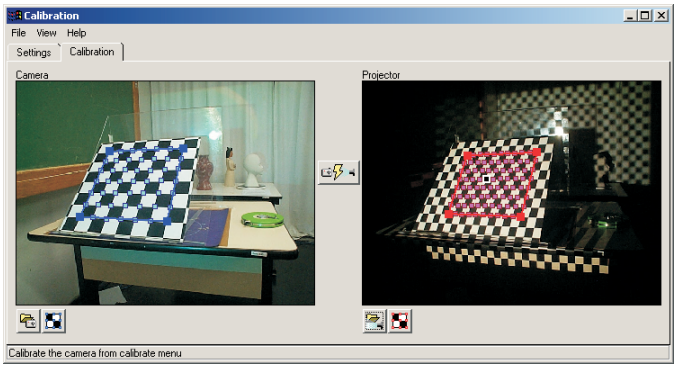


Figura A.13: Execução da calibração.

É possível visualizar a câmera e o projetor reconstruídos no espaço 3D. Para isto, basta o usuário selecionar a opção *3D Scene...* no menu *View*. A figura A.14 mostra a câmera e o projetor reconstruídos.

O usuário pode também ver os pontos 3D do padrão original projetados de volta nas imagens. Para isto basta selecionar a opção *Show*

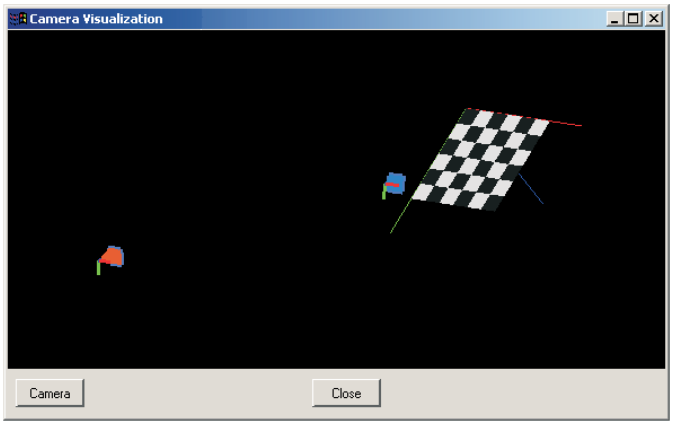


Figura A.14: Reconstrução da câmera e do projetor.

Projected Grid a partir do menu *View* (figura A.15).

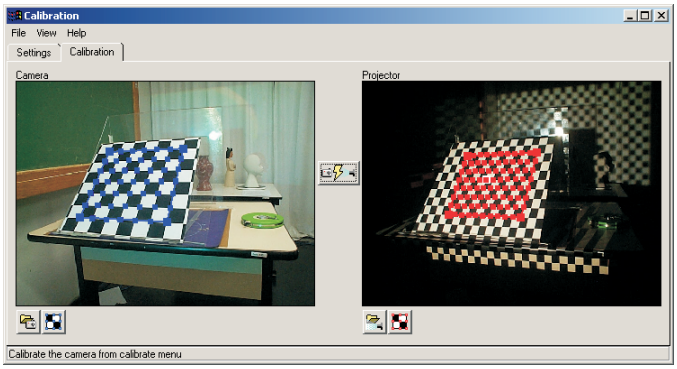


Figura A.15: Pontos 3D projetados de volta.

A.3 Programa de Triangulação

Triangular um objeto significa calcular nuvens de pontos 3D a partir de várias fotografias do objeto do mundo real e das matrizes de calibracão da câmera e do projetor. Cada nuvem de pontos corresponde a uma parte do objeto. Estas nuvens serão utilizadas posteriormente em outro programa para reconstruir completamente o objeto. Portanto, o processo de triangular um objeto requer que a calibração da câmera e do projetor tenha sido realizada, bem como a etapa de aquisição das fotografias do objeto.

Após fotografar um objeto *obj*, as imagens ficam armazenadas no diretório \local\3dp\obj\img. Para cada posição $i$do objeto fotografado é criado um diretório \local\3dp\obj\img\pos$_i$, dentro do qual são armazenadas todas as imagens relativas àquela posição. Já os dados relacionados à calibração da câmera e do projetor ficam em \local\3d\obj\calib.

Estes diretórios são acessados automaticamante pelo programa de triangulação para gerar as nuvens de pontos 3D relacionadas a cada posição do objeto.

O programa de triangulação é composto das seguintes partes (descritas em detalhes na seção A.4 e mostradas na figura A.16):

- Botões de seleção do objeto *Obj* a ser triangulado.
- Canvas principal, para visualização do objeto *Obj*.
- Árvore com todas as partes do objeto *Obj*, correspondentes às posiçôes fotografadas.
- Região de manipulação da posição corrente *Obj*
- Janela independente de visualização das imagens correspondentes à posição corrente.

A.3.1 Selecionando um Objeto para Triangulação

Para triangular um objeto *Obj*, o usuário deve selecionar o diretório com o nome do objeto. Assim, o usuário deve selecinar o botão *Open Scan...*, situado na parte superior esquerda. Será aberta uma janela

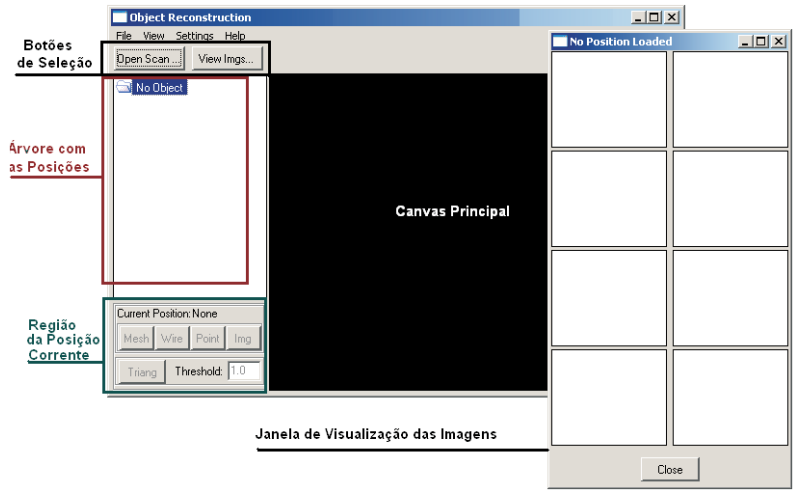


Figura A.16: Programa de Triangulação.

de seleção de diretórios, na qual o usuário deve selecionar o diretório do objeto *Obj*, ou seja, o diretório \local\3dp\obj (figura A.17).

Neste momento o programa abre este diretório e verifica a existência do arquivo **info.txt**. Este arquivo contém as informações mostradas no exemplo abaixo:

8 (número de faixas ou níveis utilizados para o gray code)
640 480 (resolução das imagens do objeto capturadas pela câmera)
800 600 (resolução das imagens dos padrões de faixas projetadas)
3 (número de posições fotografadas do objeto)

O programa entra em cada diretório \local\3dp\obj\img\pos$_i$ e lê as imagens negativas e positivas correspondentes à posição i do objeto *obj*. É necessário que o diretório de cada posição contenha um arquivo **mask.jpg** (ou .jpeg, ou .bmp), que é uma máscara indicando quais pixels da imagem correspondem ao objeto e quais correspondem ao fundo. Este arquivo de máscara pode ser gerado manualmente no photshop, paint, corel ou outro programa.

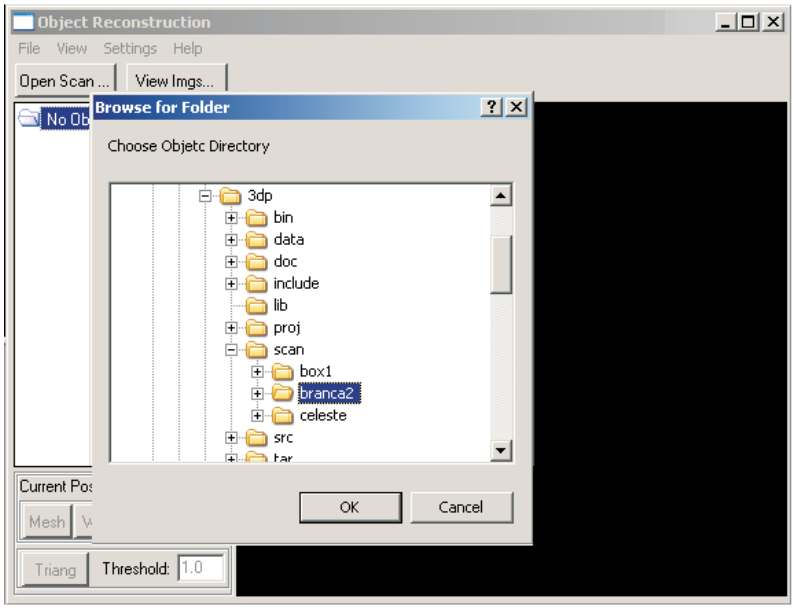


Figura A.17: Abertura de um diretório de objeto.

Se não existir o aquivo info.txt ou algum arquivo de máscara para cada posição, o programa envia uma mensagem ao usuário e a triangulação não prossegue.

A.3.2 Estabelecendo a Correspondência

Esta etapa estabelece a correspondência entre os pixels das imagens fotografadas e os pixels das faixas projetadas. É feita completamente de modo automático, sem interferência do usuário. Para tal o programa gera, dentro de cada diretório de posição i (\local\3dp\obj\img\pos$_i$)um arquivo chamado **corresp.txt**. Como a geração deste arquivo é cara, ela só é acionada se o arquivo **corresp.txt** não existir na posição i.

A.3.3 Cálculo da Triangulação

Esta etapa também é realizada de modo completamente automático. Aqui é gerado, para cada posição i do objeto, um conjunto de pontos 3D. Este conjunto representa a superfície correspondente à parte do objeto visível pela câmera a partir da posição i. Para gerar cada conjunto i, chamado de nuvem de pontos i, é necessário o arquivo **corresp.txt** da posição i. Também é necessário que exista o aquivo \local\3dp\obj\calib\output**calib.txt**, que contém as informações da calibração da câmera e do projetor. Se este arquivo não existir, o programa tenta procurá-lo em \local\3dp\data\calib\output, que é o diretório de calibração global. Se não existir, o programa envia uma mensagem ao usuário e a triangulação não prossegue.

A triangulação gera cada nuvem de pontos i, a qual é armazenada no diretório \local\3dp\obj\points\geom\pos$_i$ em um arquivo chamado **points.ply**.

A.4 Elementos da Interface

A.4.1 Botões de Seleção do Objeto

Situados na região superior esquerda da janela principal, contém dois botões: o botão *Open Scan...* (figura A.18), que permite selecionar um diretório de objeto para gerar a triangulação (figura A.17), e o botão *View Imgs...*, que permite visualizar, em uma nova janela, as imagens do objeto capturadas em uma dada posição. Se a janela de visualização destas imagens estiver aberta, o label deste botão muda para *Hide Imgs*, o qual, se for selecionado , faz com que a janela desapareça.

A.4.2 Canvas Principal

Localizado ao centro da janela principal, exibe a malha do objeto correspondente à posição corrente. As figuras A.18 e A.20 mostram malhas em *wireframe* de um objeto e a figura A.19 mostra uma malha renderizada.

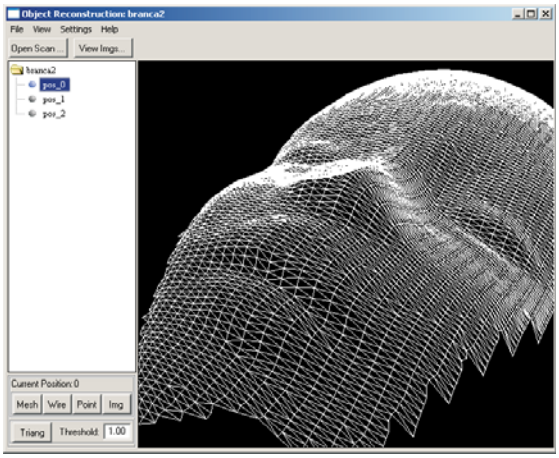


Figura A.18: Malha Triangulada.

A.4.3 Janela de Visualização de Imagens

Janela independente da principal que mostra várias imagens do objeto fotografado em uma dada posição (posição corrente).

Cada imagem mostra um nível de resolução das faixas projetadas no objeto. A janela contém vários canvas com miniaturas das imagens (figuras A.19 e A.20). Se o usuário quiser ver a imagem ampliada, deve dar um clique duplo na miniatura e será exibida uma nova janela com a imagem em tamanho natural (figura A.21).

A.4.4 A Árvore de Posições

Esta árvore, situada à esquerda da janela principal, possui o nome do objeto como raiz e seus nós (folhas) representam as posições nas quais o objeto foi fotografado. Sempre que o usuário selecionar o nó correspondente a uma posição i, toda a interface do programa exibe informações daquela posição(figuras A.19 e A.20): o canvas exibe a nuvem de pontos i, a janela de visualização de imagens exibe as projeções das faixas verticais sobre o objeto posicionado na posição i. Além disso, a região de manipulação da posição corrente (situada

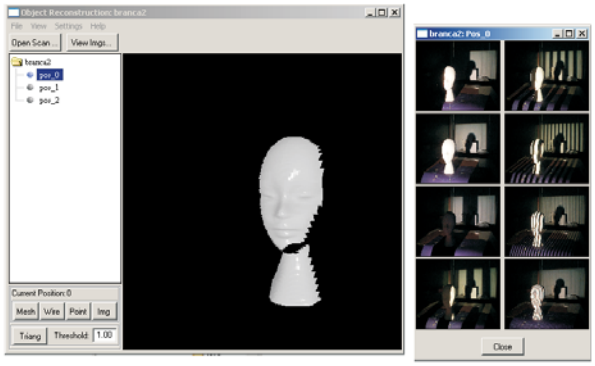


Figura A.19: Janela principal e janela de visualização das imagens.

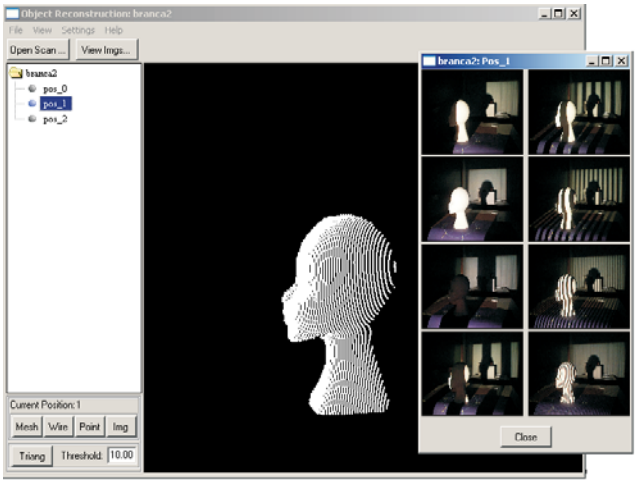


Figura A.20: Objeto visualizado em outra posição.

na parte esquerda inferior) é setada para o posição corrente i. No primeiro momento em que é gerada a triangulação de um objeto, a posição 0 (zero) é tomada como default.

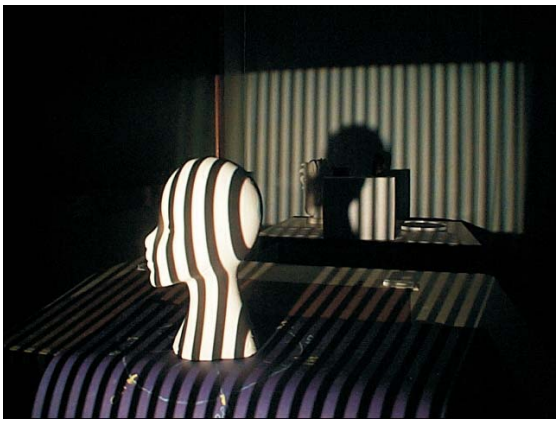


Figura A.21: Imagem ampliada.

A.4.5 Região de Manipulação da Posição Corrente

Esta região exibe vários botões para indicar como a nuvem de pontos correspondende à posição i é visualizada no canvas principal. A nuvem pode ser vista de três formas: como pontos (botão *points*), como uma malha renderizada (botão *mesh*), ou em wireframe (botão *wire*). Além disto, é possível exibir a imagem original capturada pela câmera. Esta imagem serve pra comparar se a visualização dos pontos, gerada ao fazer a câmera do opengl igual aos parâmetros de calibração, equivale à imagem capturada pela câmera.

Há ainda um outro botão que permite atualizar a triangulação utilizando um novo threshold. O valor do threshold indica a distância mínima em que dois pontos vizinhos devem formar uma aresta na triangulação da malha. O threshold default é 1.

A.5 Programa de Geração do Modelo

O programa de estruturação do modelo, realiza a filtragem dos dados e a construção do modelo com uma superfície de multiresolução.

Devido à complexidade dos objetos 3D a ser scaneados as nuvens de pontos obtidas como resultado do processo de fotografia 3D apresentam regiões com detalhes finos e de alta variação de curvatura. Devido a que vários range imagens precisam ser combinadas para obter uma representação completa do objeto scaneado, as nuvens de pontos resultantes contêm ruído devido a erros no processo de alinhamento destas range imagens. Também o ruído pode aparecer como resultado do espelhamento da luz ao fazer contato com o material do objeto. Devido a que algumas regiões quedam ocultas no processo de scaneado as nuvens de pontos resultantes podem apresentar uma amostragem bastante irregular.

Este programa concentra-se no desenvolvimento de um software de reconstrução de superfícies capaz de lidar satisfatoriamente com dados com ruído, amostragem irregular e grande tamanho. Produzindo como saída uma superfície linear por parte aproximando a nuvem de pontos original.

O principal resultado deste projeto foi a criação de um programa para a reconstrução de superfícies a partir de dados ruidosos mantendo os detalhes e as características salientes presentes nestes objetos gráficos.

O programa consta de quatro módulos principais. Eliminação de Ruído, Redução, Reconstrução e Refinamento:

O programa tem uma interface gráfica onde se podem executar cada um dos passos (Eliminação de Ruído, Redução, Reconstrução e Refinamento) ajustando os parâmetros correspondentes de cada modulo, de maneira que o usuário pode ver os resultados intermediários de cada modulo. Também o programa pode ser executado em modo bach onde o usuário especifica, mediante parâmetros, quais são os passos que deseja aplicar a nuvens de pontos, por exemplo: Eliminação de Ruído e Reconstrução ou Eliminação de Ruído, Redução e Reconstrução. O usuário deve passar os parâmetros de cada passo, caso contrário, o programa usará os parâmetros definidos por default.

A Figura A.22 mostra a interface do programa, com os parametros de controle.

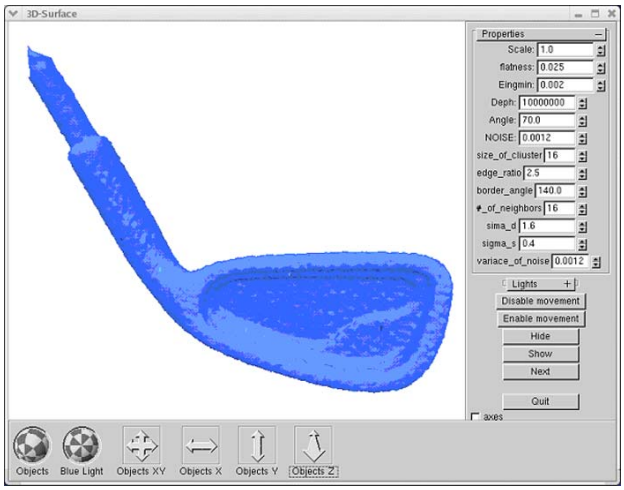


Figura A.22: Interface do Programa

A.5.1 Eliminação de Ruído

A entrada do módulo é uma nuvem de pontos possivelmente ruidosa e produz como saída uma nuvem de pontos sem ruido que mantém as características salientes e detalhes finos presentes nos dados iniciais.

O algoritmo empregado baseia-se na técnica de "Moving Least Square"para a aproximação de superfícies a partir de dados esparsos. Como mudança fundamental a este respeito, estendeu-se "Moving Least Square"para ser mais robusto ao ruído presente nos dados, de forma tal que a nuvem de pontos resultante é suave e mantém as características salientes presentes no objeto 3D original. Nesse sentido, uma nova função de energia baseada num "M-estimator"foi empregada com o objetivo de que o processo de filtragem separe o ruído dos detalhes e as características salientes dos modelos 3D. (Ver figura A.23).

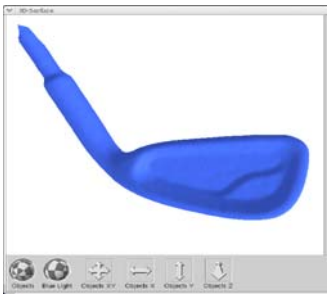
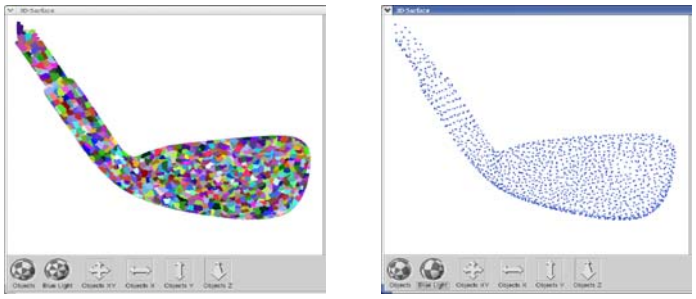


Figura A.23: Superficie filtrada com eliminação de ruido

A.5.2 Redução

A saída do módulo anterior é dividida num conjunto de clusters usando um método hierárquico de divisão do espaço (numa árvore BSP). De cada cluster extrai-se um elemento representante correspondente ao centróide do cluster. A saída do módulo é um conjunto de pontos representativos de cada cluster. (Ver figura A.24).



(a) (b)

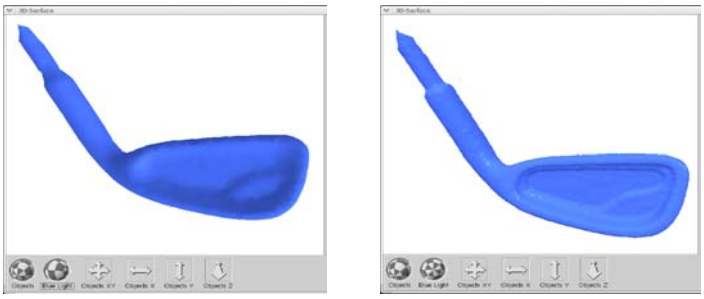
Figura A.24: (a) Conjunto de clusters dividindo a nuvem de pontos; (b) Conjunto de pontos representativos extraídos dos clusters.

A.5.3 Reconstrução

Este módulo encarrega-se da construção de uma malha inicial que interpola o conjunto de pontos representativos produzidos no módulo anterior. O algoritmo de reconstrução de superfícies empregado é do tipo "Advancing Front"baseado nos k-vizinhos mais próximos. À saída do módulo é uma superfície linear por partes interpolando o conjunto de pontos representativos, sendo a malha inicial armazenada numa estrutura de dados "Half Edge". O algoritmo tem um bom comportamento frente a uma amostragem não uniforme e aos buracos presentes nos dados. (Ver figura A.25-a).

A.5.4 Refinamento

O objetivo deste módulo é refinar a malha inicial obtida no módulo de Reconstrução adicionando detalhes nas regiões da malha correspondentes a regiões com detalhes finos na nuvem de pontos suavizada. Desta forma obtemos como resultado uma malha final adaptada à geometria da superfície original. O método usa um operador de projeção baseado na técnica de "Moving Least Square". (Figura A.25-b).



(a) (b)

Figura A.25: (a) Triangulação do conjunto de pontos representativos; (b) Refinamento da malha inicial mediante duas iterações do algoritmo.

A implementação do programa foi desenvolvida usando a biblioteca CGAL www.cgal.org, isto é, as estruturas de dados usadas para armazenamento das malhas e a implementação do algoritmo foram feitas baseadas na classe Polyhedrom e Half-Edge da CGAL. Para a implementação dos métodos numéricos usados no algoritmo de suavização foi usada a biblioteca gsl www.gsl.org. Também, para a determinação dos vizinhos mais próximos foi usado a biblioteca ANN www.ANN.org.

Referências Bibliográficas

[1] E. L. Allgower and K. Georg. *Introduction to Numerical Continuation Methods*. Springer-Verlag, Berlin, Heildelberg, 1990.

[2] E. L. Allgower and K. Georg. Simplicial pivotiong for mesh generation of implicitly defined surfaces. *Comp. Aid. Geom. Des.*, 1991.

[3] E. L. Allgower and S. Gnutzmann. An algorithm for piecewise linear approximation of implicitly defined two-dimensional surfaces. *SIAM Journal of Numerical Analysis*, 24(2):2452–2469, April 1987.

[4] E. L. Allgower and P. H. Schmidt. An algorithm for piecewise linear approximation of an implicitly defined manifold. *SIAM Journal of Numerical Analysis*, 22:322–346, 1985.

[5] T. Beier. Practical uses for implicit surfaces in animation. ACM Siggraph Course Notes, 1990. Modeling and Animating with Implicit Surfaces.

[6] R. Bergevin, M. Soucy, H. Gagnon, and D. Laurendeau. Towards a general multiview registration technique. *IEEE Trans. Pat. Anal. and Mach. Intel.*, 18(5):540–547, 1996.

[7] F. Bernardini, J. Mittlelman, H. Rushmeir, and C. Silva. The ball-pivoting algorithm for surface reconstruction. *IEEE Trans. Visual. Comput. Graphics*, 5(4):349–359, 1999.

[8] P. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pat. Anal. and Mach. Intel.*, 14(2):239–256, 1992.

[9] Rahul Bhotika, David J. Fleet, and Kyriakos N. Kutulakos. A probabilistic theory of occupancy and emptiness. Technical Report 753, University of Rochester, Department of Computer Science, 2001.

[10] J. Bloomenthal. Polygonization of implicit surfaces. *Comp. Aid. Geom. Des.*, 5(4):341–355, 1988.

[11] J. D. Boissonant. Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics*, 1999.

[12] J. De Bonet and P. Viola. Roxels: Responsibility weighted 3d volume reconstruction. In *IEEE International Conference on Computer Vision, Vol1*, pages 415 – 425. IEEE, 1999.

[13] A. Broadhurst. *A Probabilistic Framework for Space Carving*. PhD thesis, University of Cambridge, Trinity College, September 2001. Dissertation submitted to the University of Cambridge for the degree of Doctor of Philosophy.

[14] A. Broadhurst and R. Cipolla. A statistical consistency check for the space carving algorithm. In M. Mirmehdi and B.Thomas, editors, *11th British Machine Vision Conference, Volume1*, pages 282–291, Bristol, September 2000.

[15] A. Broadhurst, T. Drummond, and R. Cipolla. A probabilistic framework for the space carving algorithm. In *8th International Conference of Computer Vision*, pages 388–393, Vancouver, Canada, July 2001. IEEE Computer Society Press.

[16] A. Castelo, S. R. Freitas, and G. Tavares. Simplicial approximation of implicitly defined manifolds, 1988. (unpublished manuscript).

[17] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. *Image and Vis. Computing*, 10(3):145–155, 1992.

[18] C. H. Chien and J. K. Aggarwal. A Volume/Surface representation. In *International Conference on Pattern Recognition*, pages 817–820, Montreal, Canada, July 30 Aug. 2 1984.

[19] C. H. Chien and J. K. Aggarwal. Volume/Surface octrees for the representation of three-dimensional objects. In *Computer Vision, Graphics, and Image Processing, Vol. 36, No.*, pages 100–113, October 1986.

[20] P. Cignoni, F. Ganovelli, C. Montani, and R. Scopigno. Reconstruction of topologically correct and adaptive trilinear isosurfaces, 2000.

[21] W. B. Culbertson, T. Malzbender, and G. Slabaugh. Generalized voxel coloring. In *ICCV Workshop, Vision Algorithms Theory and Practice*, pages 100–115. Springer-Verlag Lecture Notes in Computer Science 1883, September 1999.

[22] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH 96 Conference Proceedings*, pages 303–312. ACM SIGGRAPH, Addison Wesley, August 1996.

[23] H. Edelsbrunner and E. P. Mucke. Three-dimensional alpha shapes. *ACM Trans. Graph. 10(1):43–72*, 1994.

[24] E.Horn and N.Kiryati. Toward optimal structured light patterns. *Image and Vision Computing*, 17(2):87–97, 1999.

[25] P. Eisert, E. Steinbach, and B. Girod. Multi-hypothesis, volumetric reconstruction of 3-d objects from multiple calibrated camera views. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 3509–3512., 1999.

[26] O. Faugeras and R. Keriven. Variational principles, surface evolution, PDE's, level set methods, and the stereo problem. In *IEEE Transactions on Image Processing, Vol. 7, No.*, pages 336–344., March 1998.

[27] T. Fromherz and M. Bichsel. Shape from contours as initial step in shape from multiple cues. In *ISPRS Commission III*

Symposium on Spatial Information from Digital Photogrammetry and Computer Vision, pages 240–256, Munich, Germany, 1994.

[28] T. Fromherz and M. Bichsel. Shape from multiple cues: Integrating local brightness information. In *Fourth International Conference for Young Computer Scientist, ICYCS 95*, pages 855–862, Beijing, P. R. China, 1995.

[29] P.L. George and E. Seveno. The advancing front mesh generation method revisited, 1994. International Journal for Numerical Methods in Engineering, vol. 37, pp. 3605-3619.

[30] Jonas Gomes and Luiz Velho. *Fundamentos da Computacao Grafica*. IMPA, 2004.

[31] M. Hall and J. Warren. Adaptive polygonization of implicitly defined surfaces. *IEEE Computer Graphics and Applications*, 10(6):33–43, 1990.

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

[33] H.Morita, K.Yajima, and S.Sakata. Reconstruction of surfaces of 3d objects by m-array pattern projection method. In *Proc. ICCV*, pages 468–473, 1988.

[34] H. Hopp, T. DeRose, T. Duchanp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points, 1992. ACM Computer Graphics.

[35] K. C. Hui and Z. H. Jiang. Tetrahedra based adaptive polygonization of implicit surface patches. *Computer Graphics Forum*, 18(1):57–68, March 1999. ISSN 1067-7055.

[36] E. Mouaddib J. Batlle and J. Salvi. Recent progress in coded structured light as a technique to solve the correspondence problem: A survey. *Pattern Recognition*, 31(7):963–982, 1998.

[37] J.L.Posadamer and M.D.Altschuler. Surface measurement by space-encoded projected beam systems. *Comput. Graphics Image Process*, 18:1–17, 1982.

[38] J.Tajima and M.Iwakawa. 3-d data acquisition by rainbow range finder. In *Proc. Int. Conf. on Pattern Recognition*, pages 309–313, 1990.

[39] M. Kimura, H. Saito, and T. Kanade. 3d voxel construction based on epipolar geometry. In *International Conference on Image Processing*, pages 135–139, 1999.

[40] K.L.Boyer and A.C.Kak. Color-encoded structured light for rapid active ranging. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(1):14–28, 1987.

[41] Leif P. Kobbelt, Mario Botsch, Ulrich Schwanecke, and Hans-Peter Seidel. Feature-sensitive surface extraction from volume data. *Proceedings of SIGGRAPH 2001*, pages 57–66, August 2001. ISBN 1-58113-292-1.

[42] K. N. Kutulakos. Approximate n-view stereo. In *European Conference on Computer Vision*, pages 67–83. Springer Lecture Notes in Computer Science 1842, June/July 2000. Volume1.

[43] K. N. Kutulakos and S. M. Seitz. What do n photographs tell us about 3d shape? Technical Report 680, Computer Science Dept. U. Rochester, January 1998.

[44] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218, July 2000.

[45] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2), February 1994.

[46] S.H. Lo. A new mesh generation scheme for arbitrary planar domains, 1985. International Journal for Numerical Methods in Engineering, vol. 21, pp. 1403-1426.

[47] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[48] B.Curless L.Zhang and S.M.Seitz. Rapid shape acquisition using color structured light and multi-pass dynamic programming. In *Proc. Symp. on 3D Data Processing Visualization and Transmission (3DPVT)*, 2002.

[49] W. Martin and J. K. Aggarwal. Volumetric descriptions of objects from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):150–158, March 1983.

[50] L. Massone, P. Morasso, and R. Zaccaria. Shape from occluding contours. In *SPIE Conference on Intelligent Robots and Computer Vision, SPIE Vol. 521*, pages 114–120, November 1985.

[51] E. Mencl and H. Muller. Interpolation and approximation of surfaces from three-dimensional scattered data points, 1998. State of the Art Reports, ACM Eurographics 98, pages 63–69.

[52] S. Moezzi, A. Katkere, D. Kuramura, and R. Jain. Reality modeling and visualization from multiple video sequences. *IEEE Computer Graphics and Applications*, 16(6):58–63, 1996.

[53] S. Moezzi, L. Tai, and P. Gerard. Virtual view generation for 3d digital video. *IEEE Multimedia*, 4(1):18–26, January - March 1997.

[54] J. Le Moigne and A. M. Waxman. Structured light pattern for robot mobility. *IEEE J. Robotics and Automation*, 4(5):541–548, 1988.

[55] T. Monks. *Measuring the shape of time-varying objects*. PhD thesis, Department of Eletronics and Computer Science, University of Southampton, 1994.

[56] C. Montani, R. Scateni, and R. Scopigno. A modified look-up table for implicit disambiguation of marching cubes, 1994.

[57] Anselmo Montenegro, Paulo Cezar Carvalho, Marcelo Gattass, and Luiz Velho. Space carving with a hand-held camera. In *Proceedings of SIBGRAPI / SIACG*. IEEE Press, 2004.

[58] B. Natarajan. On generating topologically consistent isosurfaces from uniform samples, 1994.

[59] O.Hall-Holt and S.Rusinkiewicz. Stripe boundary codes for real-time structured-light range scanning of moving objects. In *Proc. ICCV*, pages 13–19, 2001.

[60] S. Osher and J. Sethian. Fronts propagating with curvature dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79:12– 49, 1988.

[61] T. Pajdla. Bcrf - binary illumination coded range finder: Reimplementation. Technical Report KUL/ESAT/MI2/9502, Katholieke Universiteit Leuven, Belgium, 1995.

[62] A. Pasko and V. Pilyugin. Geometric modeling in the analysis of trivariate functions. *Computer and Graphics*, 12(3-4):457–465, 1988.

[63] Luiz Velho Paulo C. P. Carvalho, Jonas Gomes and Luiz H. De Figueiredo. *Métodos de Otimização em Computação Gráfica*. IMPA, 1999.

[64] L.S.Narasimhan P.M.Griffin and S.R.Yee. Generation of uniquely encoded light patterns for range data acquisition. *Pattern Recognition*, 25(6):609–616, 1992.

[65] M. Potmesil. Generating octree models of 3d objects from their silhouettes in a sequence of images. *, Computer Vision, Graphics and Image Processing*, 40(1):1–29, October 1987.

[66] A. Prock and C. Dyer. Towards real-time voxel coloring. In *DARPA Image Understanding Workshop*, pages 315–321, 1998.

[67] Kari Pulli and Linda G. Shapiro. Surface reconstruction and display from range and color data. *Graphical Models*, 62(3):165–201, May 2000.

[68] P.Vuylsteke and A.Oosterlinck. Range image acquisition with a single binary-encoded light pattern. *IEEE Trans. PAMI*, 12(2):148–164, 1990.

[69] C. P. Rourke and B. J. Sanderson. Introduction to piecewise linear topology, 1982. Springer Study Edition.

[70] H. Saito and T. Kanade. Shape reconstruction in projective grid space from large number of images. In *IEEE Conference on Computer Vision and Pattern Recognition - Volume2*, pages 49–54, June 23-25 1999.

[71] S. Seitz and C. Dyer. Photorealistic scene reconstruction by voxel coloring. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1067 – 1073, June 1997.

[72] S. Seitz and C. Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):151–173, 1999.

[73] J. Sethian. *, Level Set Methods and Fast Marching Methods*. Cambridge University Press, second edition, 1999.

[74] J. Shade, S. Gortler, L. He, and R. Szeliski. Layered depth images. In *Proceedings of A.C.M. SIGGRAPH*, pages 231– 242, 1998.

[75] G. C. Sharp, S. W. Lee, and D. K. Wehe. Multiview registration of 3d scenes by minimizing error between coordinate frames. In *7th European Conference on Computer Vision (ECCV 2002) (Part II)*, pages 587–597, May 2002.

[76] G. Slabaugh, W. B. Culbertson, T. Malzbender, and R. Schafer. Improved voxel coloring via volumetric optimization. Technical Report 3, Center for Signal and Image Processing, Georgia Institute of Technology, 2000.

[77] G. Slabaugh, W. B. Culbertson, T. Malzbender, and R. Schafer. A survey of methods for volumetric scene reconstruction from photographs. In *International Workshop on Volume Graphics*, 2001.

[78] G. Slabaugh, T. Malzbender, and W. B. Culbertson. , volumetric warping for voxel coloring on an infinite domain. In *Proceedings of the Workshop on 3D Structure from Multiple Images for Large-Scale Environments (SMILE)*, pages 41–50, July 2000.

[79] S. Srivastava and N. Ahuja. Octree generation from object silhouettes in perspective views. *Computer Vision, Graphics and Image Processing*, 49(1):68 – 84, January 1990.

[80] O.Hall-Holt S.Rusinkiewicz and M.Levoy. Real-time 3d model acquisition. In *Proc. SIGGRAPH 2002*, 2002.

[81] Barton T. Stander and John C. Hart. Guaranteeing the topology of an implicit surface polygonization for interactive modeling. *Proceedings of SIGGRAPH 97*, pages 279–286, August 1997. ISBN 0-89791-896-7. Held in Los Angeles, California.

[82] R. Szeliski. Rapid octree construction from image sequences. *Computer Vision, Graphics and Image Processing: Image Understanding*, 58(1):23–32, July 1993.

[83] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, 3(4):323–344, 1987.

[84] S. Vedula, S. Baker, S. Seitz, and T. Kanade. Shape and motion carving in 6d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 592– 598, 2000. Volume2.

[85] J. Veenstra and N. Ahuja. Efficient octree generation from silhouettes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 537–542, Miami Beach, Florida, June 1986.

[86] L. Velho. Adaptive polygonization of implicit surfaces using simplicial decomposition and boundary constraints. In *Proceedings of Eurographics 90*. Elsevier Science Publisher, 1990.

[87] Luis Velho, Luiz Henrique de Figueiredo, and Jonas Gomes. A unified approach for hierarchical adaptive tesselation of surfaces. *ACM Transactions on Graphics*, 18(4):329–360, October 1999. ISSN 0730-0301.

[88] Luiz Velho. Simple and efficient polygonization of implicit surfaces. *Journal of Graphics Tools*, 1(2):5–24, 1996. ISSN 1086-7651.

[89] V.Smutny and T. Pajdla. Rainbow range finder and its implementation at the CVL. Technical Report K335-96-130, Czech Technical University, Prague, 1996.

[90] G. Wyvill, C. McPheeters, and B. Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4):227–234, 1986.

[91] Y.C.Hsieh. A note on the structured light of three-dimensional imaging systems. *Pattern Recognition Letters*, 19:315–318, 1998.

[92] Y.C.Hsieh. Decoding structured light patterns for three-dimensional imaging systems. *Pattern Recognition*, 34(2):343–349, 2001.

[93] C. Zahlten and H. Jurgens. Continuation methods for approximating isovalued complex surfaces. In *Proceedings of Eurographics 91*, pages 5–19, 1991.

[94] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. *International Conference on Computer Vision (ICCV)*, pages 666–673, 1999.

[95] H.-K Zhao, S. Osher, B. Merriman, and M.Kang. Implicit, non-parametric shape recontruction from unorganized points using variational levelset method, 2002. Computer Vision and Image Processing.