Dissertação para obtenção do grau de mestre em matemática pelo

INSTITUTO NACIONAL DE MATEMÁTICA PURA E APLICADA

Calibração Robusta de Vídeo Para Realidade Aumentada

por

BRUNO EDUARDO MADEIRA

Orientador: LUIZ VELHO

Co-Orientador: PAULO CEZAR PINTO CARVALHO

18 de Dezembro de 2006

Abstract

One of the problems that must be solved for the development of an augmented reality system is the calibration problem. This problem consists in estimating camera parameters used to capture video frames that we need to combine with synthetic images. In this thesis we present an algorithm that solves this problem combining different computer vision techniques. The solution relies on correspondences from 3D scene points through frame-to-frame associations between 2D image points over the video sequence. Because even short videos are made by hundreds of frames, the correspondence must be done automatically. Kanade-Lucas-Tomasi (KLT) algorithm is used for tracking characteristic points. The algorithm developed is robust to outliers and assumes that the scene is rigid which makes the camera parameters estimation possible.

Resumo

Um dos problemas que precisa ser resolvido para o desenvolvimento de um sistema de realidade aumentada é o problema de calibração. Este problema consiste na determinação dos parâmetros da câmera utilizados na captação dos quadros do vídeo que se deseja combinar com imagens sintéticas. Nesta dissertação apresentamos um algoritmo que resolve esse problema combinando diversos procedimentos baseados em visão computacional. A solução é obtida utilizam-se correspondências entre projeções de diversos pontos da cena sobre os diversos quadros do vídeo. Tendo em vista que, mesmo vídeos de curta duração são formados por centenas de quadros, é necessário que a correspondência entre as projeções seja feita de forma automática. O algoritmo Kanade-Lucas-Tomasi (KLT) é utilizado no acompanhamento de pontos característicos. O algoritmo desenvolvido é robusto e assume que a cena é rígida, o que torna possível a solução do problema de estimação dos parâmetros da câmera.

Agradecimentos

Aos professores Luiz Velho e Paulo Cezar, pelos ensinamentos, pela atenção e paciência dispensadas a mim, desde meu ingresso no IMPA, como aluno de iniciação científica em 1998. A importância desses dois mestres em minha formação acadêmica extrapola o trabalho apresentado neste texto.

Aos professores Jonas Gomes, Ralph Teixeira, e Luiz Henrique, que junto com os professores Luiz Velho e Paulo Cezar, despertaram em mim o gosto pela pesquisa em Computação Gráfica e Visão Computacional.

Aos demais professores do IMPA, por terem mudado minha maneira de ver a Matemática, transformando-a em uma aliada poderosa na busca de soluções para problemas.

Aos meus pais, pelo apoio durante todos os anos de minha vida.

Sumário

Lista de Figuras

Capítulo 1

Introdução

1.1 Caracterização do problema

Um dos principais problemas que precisa ser resolvido para o desenvolvimento de um sistema de realidade aumentada é a determinação dos parâmetros da câmera utilizados na captação dos quadros do vídeo que se deseja combinar com imagens sintéticas. Nesta dissertação apresentamos um algoritmo, composto por diversos procedimentos heurísticos baseados em visão computacional, que resolve esse problema. Para isso utilizam-se correspondências entre projeções de diversos pontos da cena sobre os diversos quadros do vídeo.

A cena precisa ser predominantemente rígida, ou seja, a maioria dos pontos da cena não podem ter sua posição modificada, pois as restrições impostas pela rigidez sobre suas projeções é que tornam possível a determinação dos parâmetros da câmera.

Tendo em vista que, mesmo vídeos de curta duração são formados por centenas de quadros, é necessário que a correspondência entre as projeções seja feita de forma automática. Para isso é utilizado o algoritmo Kanade-Lucas-Tomasi (KLT), descrito detalhadamente em [16]. O preço pago pela automatização é a possibilidade de falha nas medições das projeções dos pontos, que torna necessário o uso de técnicas robustas.

Uma vez que tenham sido estabelecidas as correspondências entre as projeções de pontos da cena nos diversos quadros do vídeo, aplicam-se técnicas de calibração, que permitem determinar as câmeras associadas a cada quadro. Com o conhecimento dessas

Figura 1.1: Quadros de um vídeo em que foi aplicado o algoritmo apresentado nessa dissertação. Os pontos marcados nas imagens foram escolhidos e acompanhados automaticamente, sendo utilizados por um processo de calibração, que estimou as câmeras empregadas na síntese das imagens do cubo.

câmeras, pode-se inserir um objeto virtual na cena, como ilustrado na Figura 1.1.

Muitas das idéias utilizadas aqui são baseadas em [6]. Existem, entretanto, grandes diferenças no que diz respeito à estratégia de robustecimento empregada. Além disso, no nosso caso, os parâmetros da câmera são determinados em dois estágios. No primeiro estágio, utiliza-se um objeto com marcações feitas em posições conhecidas para estimar os parâmetros intrínsecos da câmera, e, em um segundo estágio, faz-se a determinação dos parâmetros extrínsecos utilizando-se os parâmetros intrínsecos estimados anteriormente. Essa estratégia elimina a necessidade de se utilizar um algoritmo de auto-calibração.

1.2 Estrutura da dissertação

O assunto tratado na dissertação foi dividido em capítulos da seguinte maneira:

Capítulo 2: São descritos modelos de câmera utilizados na solução de problemas de calibração e síntese de imagens. São apresentadas as relações existentes entre os dois tipos de modelos através da identificação dos parâmetros comuns ao modelos definidos em [7] e [8], sendo o primeiro usado em síntese de imagens, e o segundo usado em calibração. No final, explica-se detalhadamente o processo de compati-

bilização dos parâmetros da API do OpenGL com os parâmetros de uma câmera estimada por um processo de calibração.

Capítulo 3: É descrito um método para estimar os parâmetros intrínsecos de uma câmera a partir da imagem de um objeto com marcações em posições conhecidas. Como parte da solução do problema, é descrito um algoritmo que resolve o problema de calibração de uma câmera a partir de um conjunto de correspondências 3D-2D.

Capítulo 4: É descrito um algoritmo capaz de calibrar um par de câmeras, sendo conhecidas as projeções de um conjunto de pontos da cena sobre um par de imagens captado por elas. Descreve-se também um algoritmo capaz de realizar a reconstrução tridimensional de um conjunto de pontos a partir de um conjunto de projeções obtidas por um par de câmeras calibradas.

Capítulo 5: É descrito o algoritmo Kanade-Lucas-Tomasi (KLT), que é utilizado na automatização da correspondência entre pontos homólogos nos quadros de um vídeo. Assim como os dois capítulos anteriores, esse capítulo ajuda a preparar o terreno para o Capítulo 6.

Capítulo 6: Esse é o principal capítulo da dissertação. Faz-se inicialmente um conjunto de definições, visando caracterizar o problema de calibração de famílias de câmeras. Em seguida, apresenta-se um algoritmo para resolver esse problema de calibração, combinando os algoritmos descritos nos capítulos 3, 4 e 5 com o algoritmo RANSAC, que é explicado no capítulo.

Capítulo 7: É descrita a arquitetura de um sistema implementado, que tem a capacidade de inserir, de forma geometricamente consistente, objetos virtuais sobre os quadros de um vídeo capturado por uma câmera. Para fazer isso, ele combina os resultados sobre especificação de parâmetros do OpenGL, apresentados no Capítulo 2, com o algoritmo de calibração de famílias de câmeras apresentado no Capítulo 6. No final, são apresentados exemplos de resultados produzidos pelo sistema.

1.3 Notação e convenções matemáticas

1.3.1 Símbolos

A maioria dos símbolos matemáticos empregados no texto são de uso consagrado na literatura. Adotamos os mesmos significados para os símbolos feito em [10], e assumimos também o seguinte:

$a, b \in U$ significa que $a \in U$ e $b \in U$;

$f : W \subset U \to V$ significa $f : W \to V$, onde $W \subset U$;

$(X)_n$ é uma n-upla de elementos indexados $(X_1, \cdots, X_n)$;

$a \approx b$ significa que a é aproximadamente igual a b;

Se M é uma matriz então M_{ij} é o elemento da i-ésima linha e j-ésima coluna;

$\nabla f(x)$ é o vetor gradiente no ponto x associado a uma aplicação $f : U \subset \mathbb{R}^n \to \mathbb{R}$ diferenciável em $x \in U$;

$J_f(x)$ é a matriz jacobiana no ponto x associada a uma aplicação $f : U \subset \mathbb{R}^n \to \mathbb{R}^m$ diferenciável em $x \in U$;

$H_f(x)$ é a matriz hessiana no ponto x associada a uma aplicação $f : U \subset \mathbb{R}^n \to \mathbb{R}$ duas vezes diferenciável em $x \in U$;

$diag(\lambda_1, \cdots, \lambda_n)$ é a matriz diagonal definida de forma que se $M = diag(\lambda_1, \cdots, \lambda_n)$ então $M_{ii} = \lambda_i$;

$d(x, y)$ é a distância euclidiana entre os pontos x e y.

1.3.2 Coordenadas homogêneas

Em muitas partes da dissertação faz-se uso de geometria projetiva, mais especificamente, utilizam-se os espaços projetivos $\mathbb{R}P^2$ e $\mathbb{R}P^3$.

Temos que coordenadas de pontos projetados nas imagens são especificados tanto como pares ordenados $\mathbb{R}^2$ como em coordenadas homogêneas de $\mathbb{R}P^2$. Essa identificação é dada pela transformação

$$(x, y, z)^T \mapsto \left(\frac{x}{z}, \frac{y}{z}\right)^T,$$

definida quando $z \neq 0$.

O mesmo ocorre com coordenadas de pontos da cena: elas são especificadas tanto em $\mathbb{R}^3$ como em coordenadas homogêneas de $\mathbb{R}P^3$, seguindo um processo análogo. Para evitar confusões, procurou-se indicar explicitamente a que conjunto os pontos pertencem. Por exemplo, para indicar um ponto da cena dizemos "um ponto $X \in \mathbb{R}^3$ da cena" ou "um ponto $X \in \mathbb{R}P^3$ da cena".

Existe no texto avaliações de distâncias entre pontos cujas coordenadas são especificadas de forma homogêneas, ou seja, $d(x, y)$ definida com $x, y \in \mathbb{R}P^2$. Neste caso, assumiremos que, antes de ser avaliada a função distância, faz-se implicitamente a conversão das coordenadas de x e de y para $\mathbb{R}^2$, como descrito anteriormente.

Os conhecimentos de geometria projetiva necessários para a compreensão da dissertação são bastante elementares. Uma boa referência é [7]. Uma apresentação um pouco mais detalhada sobre o mesmo assunto pode ser encontrada em [8].

Capítulo 2

Câmera virtual

Uma câmera virtual é um objeto matemático que descreve o funcionamento de uma câmera óptica, ou seja, estabelece a correspondência existente entre elementos do mundo tridimensional e suas projeções em uma imagem.

No contexto de Realidade Aumentada [1] são necessários modelos de câmera que permitam resolver dois tipos de problemas:

1. Problemas de Síntese de Imagens.

2. Problemas de Calibração.

Neste capítulo abordaremos modelos de câmeras apropriados para a resolução destes problemas. Inicialmente apresentaremos um modelo de câmera básico, que será definido sem fazer uso de geometria projetiva. Este servirá de base para a definição de dois modelos criados com geometria projetiva, sendo um utilizado na resolução de problemas de síntese de imagens, e outro utilizado na resolução de problemas de calibração.

Serão apresentadas parametrizações para esses modelos. De forma semelhante a outros textos, agruparemos os parâmetros em duas categorias: os parâmetros extrínsecos e os parâmetros intrínsecos.

[1] Realidade Aumentada é o processo de composição de imagens captadas por uma câmera com imagens de objetos geradas por computador. Este processo pode ser feito em tempo real, ou não. Estamos tratando na dissertação de um processo de realidade aumentada que não é feito em tempo real.

Os parâmetros extrínsecos descrevem o posicionamento e a orientação da câmera. Já os parâmetros intrínsecos, estes descrevem o efeito da câmera sobre os raios luminsosos, e a ação dos sensores da câmera na formação da imagem. As propriedades da câmera controladas pelos parâmetros intrínsecos incluem: a distância focal, a resolução da imagem, as dimensões dos pixels, a distorção radial causada pela lente, ... etc.

O mapeamento de modelos de câmeras usados em calibração sobre os modelos usados em síntese de imagens será feito implicitamente, pela adoção de uma mesma nomeclatura na parametrização de ambos. Por exemplo, a letra d será utilizada para especificar a distância focal tanto nos modelos usados em síntese de imagens como nos modelos usados em calibração.

No final do capítulo será definido o mapeamento dos parâmetros dos modelos usados em calibração sobre os parâmetros utilizados na especifição de câmeras pela biblioteca OpenGL. A importância deste mapeamento deve-se ao fato do OpenGL ser um dos padrões de síntese de imagens mais utilizados na atualidade [19], e de ter sido utilizado no desenvolvimento do sistema descrito no Capítulo 7.

2.1 Modelo básico

Será apresentado agora um modelo de câmera básico, que será posteriormente especializado na resolução de problemas de síntese de imagens e calibração.

Uma hipótese adotada em todo o texto é que o efeito sobre os raios luminosos produzido por uma câmera que possui lentes, pode ser aproximado pelo efeito produzido por uma câmera de furo [4], que é o tipo de câmera considerado nos modelos que serão apresentados. Um tratamento mais geral, que leva em consideração a distorção radial causada pelas lentes, pode ser encontrado em [1].

Uma câmera de furo realiza uma projeção perspectiva dos pontos de uma cena sobre um anteparo. Como o centro óptico da câmera encontra-se entre o anteparo e os objetos projetados, ocorre uma inversão da imagem captada. Embora isso não gere grandes problemas do ponto de vista matemático, é comum descrever o efeito de uma câmera de furo por uma projeção perspectiva, em que o plano de projeção encontra-se entre o centro de projeção e os objetos projetados, obtendo assim um resultado equivalente,

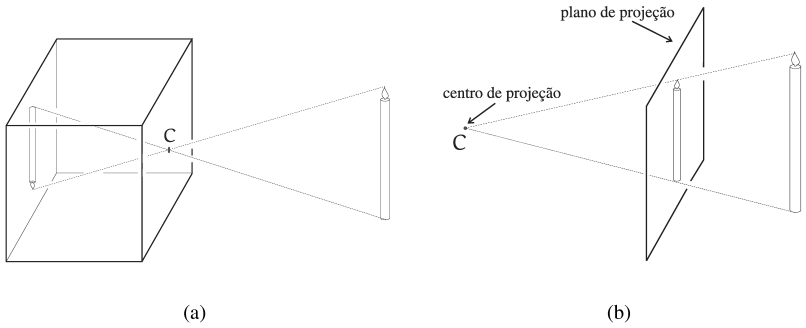

Figura 2.1: Câmera de furo (a); Modelo de câmera (b)

porém, sem a inversão, como ilustrado na Figura 2.1.

A seguir serão definidas três transformações, chamadas de T_1, T_2 e T_3, que serão combinadas para formar o modelo de câmera básico.

2.1.1 Câmera na origem

Para uma projeção perspectiva cujo centro de projeção está posicionado em $(0,0,0)^T$, e cujo plano de projeção é perpendicular ao eixo-z, temos que a transformação associada é $T_1 : S \subset \mathbb{R}^3 \to \mathbb{R}^2$, definida por

$$T_1\{(x,y,z)^T\} = \left(d\frac{x}{z}, d\frac{y}{z}\right)^T, \tag{2.1}$$

onde S é o conjunto formado pelos pontos de $\mathbb{R}^3$ que não possuem a coordenada $z = 0$, e d corresponde à distância entre o centro e o plano de projeção. Essa distância é denominada distância focal.

2.1.2 Câmera em posição genérica

A transformação correspondente a uma câmera posicionada de maneira arbitrária é dada pela composição $T_1 \circ T_2 : T_2^{-1}(S) \to \mathbb{R}^2$, onde $T_2 : \mathbb{R}^3 \to \mathbb{R}^3$ é um movimento rígido definido por

$$T_2(x) = R(x - c), \tag{2.2}$$

em que c é a posição do centro de projeção, e R é uma matriz de rotação, que determina a orientação da câmera.

A matriz de rotação R e o vetor c podem ser parametrizados por 6 números reais, que correspondem aos parâmetros extrínsecos da câmera.

2.1.3 Câmera digital

No caso de câmeras digitais, temos que a imagem é projetada sobre uma matriz de sensores, que realizam uma amostragem da mesma. Essa amostragem define um novo sistema de coordenadas para a imagem projetada. A mudança de coordenadas da imagem é definida por uma transformação afim do plano $T_3 : \mathbb{R}^2 \to \mathbb{R}^2$, da forma,

$$T_3(x) = diag(m_x, m_y) + (x_0, y_0)^T, \tag{2.3}$$

onde m_x e m_y correspondem ao número de sensores por unidade de comprimento na direção x e y respectivamente, e o par $(x_0, y_0)^T$ corresponde ao ponto principal, que define as coordenadas em escala de pixels, da projeção ortogonal do centro de projeção sobre o plano de projeção.

2.1.4 Parâmetros intrínsecos

Vamos analisar agora a composição $T_3 \circ T_1 : S \to \mathbb{R}^2$. Essa transformação é definida por

$$T_3 \circ T_1 \left\{(x, y, z)^T\right\} = \left(dm_x \frac{x}{z} + x_0, dm_y \frac{y}{z} + y_0\right)^T. \tag{2.4}$$

É imediato verificar, pela expressão acima, que câmeras digitais com distâncias focais diferentes podem produzir o mesmo resultado, bastando para isso escolher uma resolução espacial apropriada. Isso ocorre pois esses valores aparecem combinadas na forma dos produtos dm_x e dm_y.

Os valores x_0, y_0, dm_x e dm_y definem os parâmetros intrínsecos do modelo de câmera básico.

2.1.5 Dimensão do espaço de câmeras virtuais

Temos que as transformações $T_3 \circ T_1 \circ T_2 : T_2^{-1}(S) \to \mathbb{R}^2$ definem um espaço de câmeras virtuais que possui 10 graus de liberdade, sendo 3 graus de liberdades associados à rotação R, 3 graus de liberdade associados à posição do centro de projeção c, e os demais 4 graus de liberdades definidos pelos parâmetros intrínsecos.

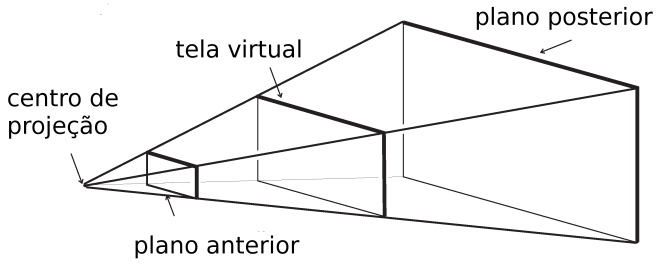

Figura 2.2: Pirâmide de visão.

Observamos que ao considerarmos um modelo com 10 graus de liberdade, estamos desconsiderando que as dimensões do anteparo da câmera de furo são parâmetros intrínsecos. Do ponto de vista de calibração, isso não gera nenhum problema pois as limitações do anteparo estão sendo aplicadas fisicamente pela câmera, por outro lado, do ponto de vista de síntese de imagens, essas dimensões são importantes.

2.2 Câmera para síntese de imagens

O problema de síntese de imagens pode ser definido como o de criar imagens a partir de descrições de cenas tridimensional. Esta seção e a próxima tratam de um modelo de câmera apropriado para síntese de imagens a partir de cenas cuja descrição da geometria dos objetos é feita por uma representação poliedral.

2.2.1 Terminologias

Serão apresentados agora os principais termos usados na especificação de câmeras em computação gráfica. A Figura 2.2 ilustra cada um deles.

Tela virtual é o retângulo do plano de projeção que contém a imagem projetada. Essa limitação definida sobre o plano de projeção corresponde fisicamente às limitações nas dimensões do anteparo onde os raios luminosos são projetados.

Pirâmide de visão é a pirâmide definida pelo centro de projeção e pela tela virtual.

Plano anterior é um plano posicionado a frente do centro de projeção. Apenas pontos que estão a frente do plano anterior são projetados na imagem.

Plano posterior é um plano posicionado a frente do plano anterior. Apenas pontos que estão atrás do plano posterior são projetados na imagem.

Volume de visão é o tronco de pirâmide definido pela porção da pirâmide de visão delimitada pelo plano anterior e pelo plano posterior.

2.2.2 Recorte e visibilidade

O modelo de câmera básico definido pela transformação $T_3 \circ T_1 \circ T_2 : T_2^{-1}(S) \rightarrow \mathbb{R}^2$ é capaz de descrever a posição na imagem de todos os pontos da cena que são projetados. Por outro lado, ele define projeções para pontos da cena que não seriam projetados pela câmera de furo correspondente. Mais precisamente, para que um ponto da cena $X \in \mathbb{R}^3$ seja projetado por uma câmera ele precisa satisfazer as seguintes propriedades:

1. X deve estar à frente da câmera;
2. A projeção de X deve estar contida no anteparo da câmera;
3. X não deve sofrer oclusão de outro ponto da cena.

A pirâmide de visão é o lugar geométrico dos pontos que satisfazem as propriedades 1 e 2. A determinação dos pontos da cena que pertencem à pirâmide de visão é chamada de recorte em relação à pirâmide de visão. Já o problema de determinar os pontos que satisfazem a propriedade 3 é conhecido com o nome de problema de visibilidade.

Em computação gráfica, exige-se, além dessas três propriedades, que X pertença a região do espaço delimitada pelos planos anterior e posterior, substituindo a operação de recorte em relação a pirâmide de visão pelo recorte em relação ao volume de visão.

O objetivo da restrição dada pelo plano anterior é evitar problemas numéricos ao se realizar divisões por números muito pequenos. Esse tipo de erro pode ocorrer, por exemplo, se aplicarmos a transformação T_1, definida pela equação (2.1), a um ponto muito próximo do centro de projeção. Já o objetivo da restrição dada pelo plano posterior

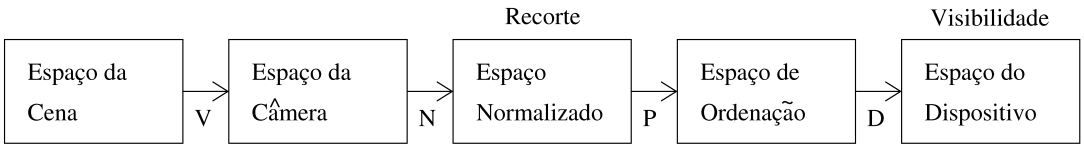

Figura 2.3: Transformações que compõem o modelo de câmera usado em síntese de imagens.

é limitar a profundidade da região da cena que será projetada, permitindo que se possa empregar o algoritmo *Z-buffer* na resolução de problemas de visibilidade.

Uma análise de algoritmos que resolvem problemas de visibilidade e recorte estão fora do escopo desta dissertação. Tal assunto é abordado detalhadamente em [7].

2.3 Transformação de visualização

Normalmente, utiliza-se em síntese de imagens um modelo de câmera formado por uma seqüência de transformações projetivas em $\mathbb{R}P^3$ que são aplicadas de forma sucessiva, intercaladas com algoritmos que resolvem os problemas de recorte e visibilidade. Trataremos de uma seqüência em particular que é apresentada na Figura 2.3.

As transformações apresentadas a seguir são uma adaptação do modelo definido em [7] à notação estabelecida na seção 2.1.

2.3.1 Posicionamento da câmera

A transformação $V : \mathbb{R}P^3 \to \mathbb{R}P^3$ faz a mudança do sistema de coordenadas da cena para o sistema de coordenadas da câmera, ou seja, é uma versão projetiva para o movimento rígido definido por T_2 na seção 2.1.2. Sua representação matricial é

$$V = \begin{pmatrix} R & -Rc \\ 0^T & 1 \end{pmatrix}. \tag{2.5}$$

2.3.2 Transformação de normalização

A transformação $N : \mathbb{R}P^3 \to \mathbb{R}P^3$ faz a mudança do sistema de coordenadas da câmera para um sistema de coordenadas normalizado, onde o problema de recorte fica

simplificado. Sua representação matricial é

$$N = \begin{pmatrix} \frac{d}{fs_x} & 0 & \frac{x_0 - m_x s_x}{f m_x s_x} & 0 \\ 0 & \frac{d}{fs_y} & \frac{y_0 - m_y s_y}{f m_y s_y} & 0 \\ 0 & 0 & \frac{1}{f} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{2.6}$$

onde temos respectivamente que n e f são as distâncias do plano anterior e posterior ao centro de projeção, e $2s_x$ e $2s_y$ são as dimensões horizontal e vertical da tela virtual.

O problema de recorte em relação à pirâmide de visão fica simplificado, pois no sistema de coordenadas normalizado, a pirâmide de visão é mapeada na pirâmide definida como

$$\left\{ (x, y, z)^T \in \mathbb{R}^3 : -z < x < z, -z < y < z, 0 < z \right\}.$$

2.3.3 Projeção perspectiva

A transformação $P : \mathbb{R}P^3 \to \mathbb{R}P^3$ faz a mudança do sistema de coordenadas normalizado para o sistema de coordenadas de ordenação. Sua representação matricial é

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{f}{f-n} & \frac{-n}{f-n} \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{2.7}$$

Ao descrevermos a cena no sistema de coordenadas de ordenação, obtemos duas propriedades interessantes, que são:

1. Nesse referencial, ao aplicarmos uma transformação $\Pi : \mathbb{R}P^3 \to \mathbb{R}^2$, definida por $(a_x, a_y, a_z, 1)^T \mapsto (a_x, a_y)^T$, obtemos a projeção perspectiva feita pela câmera virtual correspondente;

2. Nesse referencial, um ponto $A = (a_x, a_y, a_z, 1)^T$ exerce uma oclusão sobre um ponto $B = (b_x, b_y, b_z, 1)^T$, se e somente se, $\Pi(A) = \Pi(B)$ e $a_z < b_z$

Essas duas propriedades mostram que tanto o cálculo de perspectiva, como a solução para o problema de visibilidade podem ser realizados de maneira trivial no sistema de coordenadas de ordenação.

2.3.4 Coordenadas do dispositivo

A transformação $D : \mathbb{R}P^3 \rightarrow \mathbb{R}P^3$ faz a mudança do sistema de coordenadas de ordenação para o sistema de coordenadas do dispositivo. Esse sistema de coordenadas possui algumas propriedades interessantes:

1. As duas propriedades do referencial de ordenação continuam válidas;
2. As coordenadas dos eixos x e y são dadas em escala de pixels;
3. O volume de visão, na direção do eixo-z, corresponde exatamente ao intervalo de representação do *Z-buffer*.

A representação matricial da transformação D é dada por:

$$D = \begin{pmatrix} s_x m_x & 0 & 0 & s_x m_x \\ 0 & s_y m_y & 0 & s_y m_y \\ 0 & 0 & Z_{max} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{2.8}$$

onde, $[0, Z_{max}]$ é o intervalo de representação do *Z-buffer*.

2.4 Comparação com o modelo básico

2.4.1 Parâmetros intrínsecos

A matriz associada à composição $DPN : \mathbb{R}P^3 \rightarrow \mathbb{R}P^3$ é dada por

$$DPN = \begin{pmatrix} dm_x & 0 & x_0 & 0 \\ 0 & dm_y & y_0 & 0 \\ 0 & 0 & \frac{fZ_{max}}{f-n} & \frac{-nfZ_{max}}{f-n} \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{2.9}$$

A restrição da imagem de DPN ao plano-xy é igual ao efeito da transformação $T_3 \circ T_1$ definida em pela equação (2.4). Mais precisamente, $\Pi \circ DPN = T_3 \circ T_1$, onde $\Pi : \mathbb{R}P^3 \to \mathbb{R}^2$ é definida como na seção 2.3.3. Já o efeito de DPN na direção do eixo-z é um ajuste afim do volume de visão sobre o intervalo $[0, Z_{max}]$.

Embora essa matriz não apareça explicitamente em um sistema gráfico, pois não se pode compor DP com N, pois é necessário realizar a operação de recorte após a aplicação de N, temos que ela é interessante pois deixa evidente que o efeito dos parâmetros intrínsecos x_0, y_0, dm_x e dm_y na projeção dos pontos da cena é o mesmo do modelo básico. Além disso, essa matriz exibe dois parâmetros intrínsecos extras n e f, que não correspondem a parâmetros de câmeras do mundo real, e cujo efeito na imagem gerada é a eliminação de superfícies projetadas.

O valor Z_{max} não é determinado pelo estado da câmera, mas pelo dispositivo utilizado pelo algoritmo *Z-buffer*, logo, não é um parâmetro da câmera, e não causa nenhuma influência na imagem gerada pelo modelo.

Outra observação importante é que DPN é livre em s_x e em s_y, que é um fato esperado, visto que esses valores não aparecem em $T_3 \circ T_1$. No entanto s_x e s_y são valores relevantes pois definem as dimensões da imagem, logo são parâmetros intrínsecos da câmera.

2.4.2 Dimensão

Conclui-se que o modelo de câmera usado em síntese de imagens possui 14 graus de liberdade. Além dos 10 graus de liberdade do modelo básico, existem outros quatro parâmetros intrínsecos, sendo dois correspondentes às dimensões da tela virtual e os outros dois correspondentes às distâncias do centro de projeção aos planos anterior e posterior.

2.4.3 Vantagens sobre o modelo básico

Os motivos que tornam vantajoso o uso de transformações projetivas no $\mathbb{R}P^3$, na construção de sistemas gráficos, no lugar da formulação feita no modelo básico são os seguintes:

1. Transformações projetivas em $\mathbb{R}P^3$ permitem representar tanto movimentos rígidos no espaço como operações de projeção.

2. O problema de visibilidade fica simplificado escolhendo-se um sistema de coordenadas apropriado de $\mathbb{R}P^3$, como nos casos dos sistemas de coordenadas de ordenação e do dispositivo.

2.5 Câmeras para calibração

O problema de calibração consiste em determinar os parâmetros extrínsecos e intrínsecos de um conjunto de câmeras. Esses problema genérico pode ser especializado em diferentes modalidades. No nosso caso estamos interessados na seguinte formulação em particular:

Dado um conjunto de n imagens, determinar os parâmetros extrínsecos e intrínsecos das n câmeras que captaram essas imagens.

Nesse caso, diremos que as n câmeras são consistentes com as n imagens, e que as n câmeras fornecem uma explicação para as n imagens.

Na prática, o problema de calibração é formulado sob um ponto de vista de otimização, tendo em vista que erros de medições nas imagens geralmente fazem com que não exista um conjunto de câmeras consistente. Dessa forma, o problema de calibração passa a ser reformulado como:

Dado um conjunto de n imagens, determinar os parâmetros extrínsecos e intrínsecos das n câmeras que melhor explicam as n imagens.

A formalização matemática desse problema de otimização, e um algoritmo que o resolve, são apresentados no Capítulo 6.

2.5.1 Modelo projetivo

O modelo de câmera empregado em calibração pode ser obtido reescrevendo-se as transformações T_1, T_2 e T_3 definida na seção 2.1 como transformações projetivas $T_1 : \mathbb{R}P^3 \to \mathbb{R}P^2$, $T_2 : \mathbb{R}P^3 \to \mathbb{R}P^3$ e $T_3 : \mathbb{R}P^2 \to \mathbb{R}P^2$, obtendo as seguintes representações matriciais:

$$T_1 = \begin{pmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, T_2 = \begin{pmatrix} R & -Rc \\ 0^T & 1 \end{pmatrix} \text{ e } T_3 = \begin{pmatrix} m_x & 0 & x_0 \\ 0 & m_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}.$$

2.5.2 Notação $K\,[R|t]$

É imediata a verificação que as transformações projetivas $T_3 \circ T_1 \circ T_2 : \mathbb{R}P^3 \to \mathbb{R}P^2$ podem ser representadas pelo produto de uma matriz 3×3 por uma matriz 3×4, como mostrado abaixo:

$$T_3 \circ T_1 \circ T_2 = \begin{pmatrix} dm_x & 0 & x_0 \\ 0 & dm_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} R & -Rc \end{pmatrix}. \tag{2.10}$$

Nesse caso, é comum utilizar a notação compacta $K\,[R| - Rc]$ para expressar esse produto. Nessa notação, K corresponde à matriz 3×3 que especifica os parâmetros intrínsecos da câmera, e $[R| - Rc]$ corresponde à matriz 4×3 que especifica os parâmetros extrínsecos. É comum também o uso da notação $K\,[R|t]$ cuja única diferença para a notação anterior é que a posição do centro de projeção não é explicitada, tendo em vista que o produto $-Rc$ é substituído por um vetor $t \in \mathbb{R}^3$, que representa a translação da câmera.

2.5.3 Câmera projetiva genérica

Temos que as transformações $T_3 \circ T_1 \circ T_2 : \mathbb{R}P^3 \to \mathbb{R}P^2$ definem um conjunto de matrizes 4×3 que possui 10 graus de liberdade. Considerando que o conjunto formado por todas as transformações projetivas definidas em $\mathbb{R}P^3 \to \mathbb{R}P^2$ possui 11 graus de liberdade, conclui-se que certamente existem transformações projetivas desse conjunto que não correspondem a nenhuma câmera.

Será mostrado na seção 3.2 que esse grau de liberdade extra pode ser obtido considerando-se um modelo para câmeras definido por transformações projetivas da forma

$$\begin{pmatrix} f_1 & s & x_0 \\ 0 & f_2 & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} R & -Rc \end{pmatrix}.$$

Esse modelo caracteriza uma câmera projetiva genérica [8], que possui 5 parâmetros intrínsecos: f_1, f_2, s, x_0 e y_0. O grau de liberdade extra permite que o ângulo θ definido pelos eixos x e y, que especificam o sistema de coordenadas da imagem, possa ser modificado. Fisicamente isso pode ser interpretado como um cisalhamento na matriz de sensores de uma câmera digital.

Os parâmetros f_1, f_2 e s relacionam-se com os parâmetros do modelo de 10 graus de liberdade [4]:

$$f_1 = dm_x, \tag{2.11}$$

$$f_2 = \frac{dm_y}{sen\theta}, \tag{2.12}$$

$$s = -f_1 cotg\theta. \tag{2.13}$$

O par $(x_0, y_0)^T$ possui a mesma interpretação do modelo de 10 graus de liberdade, especificando as coordenadas, em escala de pixels, do ponto principal.

2.6 Câmera no OpenGL

Mostramos nas seções anteriores como os parâmetros intrínsecos e extrínsecos são inseridos nas transformações projetivas que compõem modelos de câmeras utilizados em calibração de câmeras e em síntese de imagens. Apresentaremos agora como esses parâmetros podem ser utilizados na especificação de uma câmera da biblioteca OpenGL. Mais precisamente, mostraremos as chamadas de funções da biblioteca OpenGL necessárias para definir os parâmetros de uma câmera $K\,[R|t]$, possivelmente estimados por um processo de calibração. Detalhes sobre as funções dessa biblioteca podem ser encontrados em [20].

2.6.1 Especificação dos parâmetros extrínsecos

Os parâmetros extrínsecos de uma câmera $K\,[R|t]$ podem ser especificados no OpenGL realizando-se as seguintes chamadas de funções

1. `gluLookAt(0, 0, 0, 0, 0, 1, 0, 1, 0)`, para definir um sistema de coordenadas canônico;

2. `glLoadMatrixd(m)`, onde o argumento m é um vetor que representa a matriz

$$\begin{pmatrix} R & t \\ 0^T & 1 \end{pmatrix}.$$

2.6.2 Especificação dos parâmetros intrínsecos

A especificação dos parâmetros intrínsecos é menos imediata. Observamos inicialmente que as câmeras definidas pelo OpenGL não apresentam cisalhamento na matriz de sensores, ou seja, se desejarmos especificar os parâmetros intrínsecos de uma câmera $K[R|t]$ é necessário que a matriz K seja da forma

$$K = \begin{pmatrix} f_1 & 0 & u_0 \\ 0 & f_2 & v_0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Nesse caso, pode-se utilizar a função `glFrustum`, cujo protótipo é definido por

```
void glFrustum( GLdouble left, GLdouble right, GLdouble bottom,
        GLdouble top, GLdouble near, GLdouble far );
```

Os argumentos da função `glFrustum` definem no referencial da câmera as coordenadas em $\mathbb{R}^3$ dos vértices esquerdo inferior e direito superior da tela virtual, como sendo $(\texttt{left}, \texttt{bottom}, \texttt{near})^T$ e $(\texttt{right}, \texttt{top}, \texttt{near})^T$ respectivamente. Os parâmetros `near` e `far` definem a distância do centro de projeção aos plano anterior e posterior. Além disso, o plano anterior é coincidente como o plano de projeção, ou seja, a distância focal é `near`.

Precisamos determinar os argumentos que devem ser passados para `glFrustum` de forma que o volume de visão seja compatível com os parâmetros intrínsecos da matriz K.

Observando as equações (2.11) e (2.12) temos que o número de sensores por unidade de comprimento na horizontal e vertical, medidos sobre a tela virtual, são definidos respectivamente por

$$m_x = \frac{f_1}{\texttt{near}}, \tag{2.14}$$

$$m_y = \frac{f_2}{\texttt{near}}. \tag{2.15}$$

As coordenadas do ponto principal, medidas no sistema de coordenadas da imagem, são $(u_0, v_0)^T$. Como as coordenadas do ponto principal sobre a tela virtual são $(0, 0, \texttt{near})^T$, conclui-se que deve-se chamar a função `Frustum` passando os seguintes argumentos

$$\texttt{left} = \frac{-u_0}{m_x}, \texttt{right} = \frac{w - u_0}{m_x}, \texttt{bottom} = \frac{-v_0}{m_y} \text{ e } \texttt{top} = \frac{h - v_0}{m_y},$$

onde w e h correspondem respectivamente à resolução horizontal e vertical da imagem captada pela câmera.

Capítulo 3

Parâmetros intrínsecos

O objetivo deste capítulo é descrever um método para encontrar os parâmetros intrínsecos de uma câmera. O método descrito pode ser encontrado em [17]. Ele utiliza um objeto, chamado de objeto calibrador. Tal objeto possui um conjunto de marcações cujas posições, definidas em relação a um referencial associado a ele, são conhecidas. Nos experimentos foi utilizado o objeto da Figura 3.1.

O método é composto por duas etapas:

1. Calibração em relação ao objeto calibrador.

2. Isolamento dos parâmetros da câmera.

A calibração em relação ao objeto calibrador corresponde à determinação da transformação projetiva $P : \mathbb{R}P^3 \rightarrow \mathbb{R}P^2$ que define a câmera em relação ao referencial associado ao objeto calibrador.

O isolamento dos parâmetros da câmera corresponde à determinação das matrizes, 3×3, K e R, e do vetor $t \in \mathbb{R}^3$, tais que $P = K\,[R|t]$. Ficando então determinada a matriz K, que é a resposta ao problema.

Essa estratégia é interessante pois não exige nenhuma restrição sobre o posicionamento da câmera em relação ao referencial associado ao objeto calibrador. Analogamente tem-se que escolhas diferentes de referenciais sobre o objeto calibrador não alteram a matriz K.

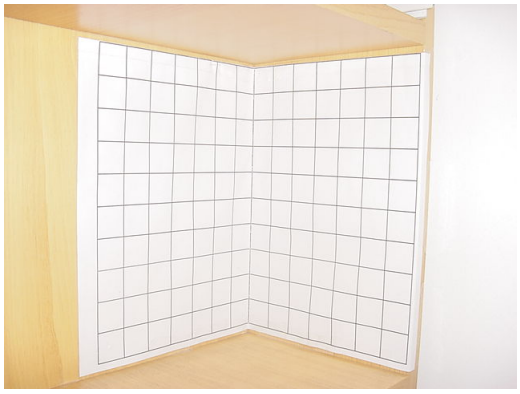

Figura 3.1: Objeto com marcações em posições conhecidas, usado para calibração

No final do capítulo, é apresentado um método para determinar os parâmetros intrínsecos impondo a restrição de não cisalhamento da matriz de sensores. Saber impor essa restrição é importante, pois ela é exigida pela maioria dos sistemas gráficos comerciais, como ilustrado na seção 2.6.2, no caso do OpenGL.

3.1 Calibração em relação ao objeto calibrador

O problema de calibração em relação ao objeto calibrador pode ser definido por

Problema 3.1. *Sendo conhecidas as projeções* $x_1, ..., x_n$*, com* $x_i \in \mathbb{R}P^2$*, correspondentes aos pontos* $X_1, ..., X_n$*, com* $X_i \in \mathbb{R}P^3$*, definidas no referencial do objeto calibrador. Determinar a transformação* $P : \mathbb{R}P^3 \to \mathbb{R}P^2$ *tal que* $PX_i = x_i$*,* $i \in \{1, 2, ..., n\}$*.*

3.1.1 Calibração usando seis correspondências

Considerando os elementos da matriz associada a P como variáveis, temos que cada sentença da forma $PX_i = x_i$ define duas equações lineares com 12 variáveis. Conseqüentemente, se forem estabelecidas 6 correspondências entre pontos e projeções, tem-se que o sistema possui solução caso não existam linhas linearmente dependentes.

Como as coordenadas de cada x_i são normalmente corrompidas por ruído, por serem obtidas por uma câmera, é introduzido erro na solução do sistema, tornando interessante o uso de um número maior de correspondências em uma formulação super-

determinada. Além disso, o sistema obtido com 6 correspondências embora possa parecer bem determinado, não o é. Trata-se de um sistema super-determinado, pois P é definido a menos de uma multiplicação por um escalar. Mostraremos a seguir como esse problema pode ser resolvido utilizando-se uma quantidade arbitrária de correspondências.

3.1.2 Encontrar $x \in S^n$ que minimiza $\|Ax\|$

Seja A uma matriz $m \times n$. Uma maneira de reformular um sistemas da forma $Ax = 0$, com $x \in \mathbb{R}P^n$, no caso em que o número de equações é maior do que $n-1$, é considerar como solução a resposta ao problema de encontrar $x \in S^n$ que minimiza $\|Ax\|$. Denotaremos esse problema por $\min_{\|x\|=1} \|Ax\|$. Sua solução pode ser facilmente determinada pela proposição abaixo.

Proposição 3.1. *Seja $Udiag(\lambda_1, \lambda_2, ..., \lambda_n)V^T$, com $\lambda_1 \geqslant \lambda_2 \geqslant ... \geqslant \lambda_n \geqslant 0$, a decomposição SVD de uma matriz A, $m \times n$, em que $m \geqslant n$. Se $v \in \mathbb{R}^n$ é o vetor correspondente a n-ésima coluna de V, tem-se que v é o vetor que minimiza a função $x \mapsto \|Ax\|$, definida sobre os pontos de $\mathbb{R}^n$ que satisfazem $\|x\| = 1$.*

Demonstração

$$\min_{\|x\|=1} \|Ax\| = \min_{\|x\|=1} \|USV^T x\|, \tag{3.1}$$

onde $S = diag(\lambda_1, \lambda_2, ..., \lambda_n)$.

Como U é uma isometria temos que

$$\min_{\|x\|=1} \|USV^T x\| = \min_{\|x\|=1} \|SV^T x\|. \tag{3.2}$$

Definindo $y = V^T x$ obtemos

$$\min_{\|x\|=1} \|SV^T x\| = \min_{\|y\|=1} \|Sy\|. \tag{3.3}$$

Usando a definição de S temos que

$$\min_{\|y\|=1} \|Sy\| = \min_{y_1^2+...+y_n^2=1} \sqrt{\lambda_1^2 y_1^2 + \ldots \lambda_n^2 y_n^2}. \tag{3.4}$$

Como $\lambda_1 \geqslant \lambda_2 \geqslant ... \geqslant \lambda_n \geqslant 0$ concluímos que a solução para esse problema de otimização é o vetor $y = (0, \ldots, 0, 1)^T$. Logo o vetor v que resolve $\underset{\|x\|=1}{min} \|Ax\|$ é dado por $V\,(0, \ldots, 0, 1)^T$, que corresponde a n-ésima coluna de V.

3.1.3 Calibração usando mais de seis correspondências

Para adequar o resultado anterior ao nosso problema, basta verificar que encontrar P que satisfaz

$$\forall i \in \{1, \ldots, n\}\,, PX_i = (u_i, v_i, 1)^T \tag{3.5}$$

é equivalente a resolver o sistema $A\mathcal{P} = 0$, onde

$$A = \begin{pmatrix} X_1^T & 0^T & -u_1 X_1^T \\ 0^T & X_1^T & -v_1 X_1^T \\ X_2^T & 0^T & -u_2 X_2^T \\ 0^T & X_2^T & -v_2 X_2^T \\ \vdots & \vdots & \vdots \\ X_n^T & 0^T & -u_n X_n^T \\ 0^T & X_n^T & -v_n X_n^T \end{pmatrix}, \tag{3.6}$$

e $\mathcal{P} = (P_{11}, P_{12}, ..., P_{33}, P_{34})^T$ é um vetor cujos elementos são os 12 elementos da matriz P, a serem determinados.

Podemos utilizar a proposição 3.1 para resolver o problema $\underset{\|\mathcal{P}\|=1}{min} \|A\mathcal{P}\|$, que fornece uma estimativa para os elementos da matriz P.

3.2 Isolamento dos parâmetros da câmera

Consideremos que estamos de posse de uma matriz P, 3×4, que representa uma transformação projetiva. Mostraremos agora um processo para fatorar P na forma $K\,[R|t]$. Esse processo é importante por dois motivos. Por um lado funciona como uma demonstração, por construção, que transformações projetivas definidas em $\mathbb{R}P^3 \rightarrow \mathbb{R}P^2$ são sempre modelos para câmeras projetivas genéricas. Por outro lado, serve como um algoritmo para determinar os parâmetros intrínsecos e extrínsecos de uma câmera.

Seja $P = \lambda K\,[R|t]$, onde λ é uma constante que pode assumir qualquer valor em $\mathbb{R} - \{0\}$. Assumindo as seguintes definições:

$$P = \begin{pmatrix} a_1^T & b_1 \\ a_2^T & b_2 \\ a_3^T & b_3 \end{pmatrix},\ K = \begin{pmatrix} f_1 & s & u_0 \\ 0 & f_2 & v_0 \\ 0 & 0 & 1 \end{pmatrix} \text{ e } [R|t] = \begin{pmatrix} R_1^T & t_1 \\ R_2^T & t_2 \\ R_3^T & t_3 \end{pmatrix}.$$

temos que

$$\begin{pmatrix} a_1^T & b_1 \\ a_2^T & b_2 \\ a_3^T & b_3 \end{pmatrix} = \lambda \begin{pmatrix} f_1 R_1^T + s R_2^T + u_0 R_3^T & f_1 t_1 + s t_2 + u_0 t_3 \\ f_2 R_2^T + v_0 R_3^T & f_2 t_2 + v_0 t_3 \\ R_3^T & t_3 \end{pmatrix}.$$

Mostraremos agora como determinar todos os parâmetros intrínsecos e extrínsecos associados a P.

Determinando λ

Podemos determinar $|\lambda|$ usando que $|\lambda| = |\lambda|\|R_3^T\| = \|a_3^T\|$. Assumiremos por enquanto que $\lambda > 0$, no final iremos concluir se essa escolha foi ou não apropriada. Ou seja, vamos assumir que

$$\lambda = \|a_3^T\|. \tag{3.7}$$

Determinando R_3 e t_3

Definindo $P' = \frac{1}{\lambda} P$ obtemos

$$P' = K\,[R|t] = \begin{pmatrix} a_1'^T & b_1' \\ a_2'^T & b_2' \\ a_3'^T & b_3' \end{pmatrix} = \begin{pmatrix} f_1 R_1^T + s R_2^T + u_0 R_3^T & f_1 t_1 + s t_2 + u_0 t_3 \\ f_2 R_2^T + v_0 R_3^T & f_2 t_2 + v_0 t_3 \\ R_3^T & t_3 \end{pmatrix}.$$

Como conseqüência temos que

$$R_3 = a'_3, \tag{3.8}$$

e

$$t_3 = b'_3. \tag{3.9}$$

Determinando v_0

Para determinar v_0 basta observar que

$$f_2 R_2^T + v_0 R_3^T = a_2'^T \Rightarrow f_2 R_2^T R_3 + v_0 R_3^T R_3 = a_2'^T R_3$$

Como $R_2 \perp R_3$ e $R_3^T R_3 = 1$ temos que

$$v_0 = a_2'^T R_3. \tag{3.10}$$

Determinando u_0

Para determinar u_0 basta observar que

$$f_1 R_1^T + s R_2^T + u_0 R_3^T = a_1'^T \Rightarrow f_1 R_1^T R_3 + s R_2^T R_3 + u_0 R_3^T R_3 = a_1'^T R_3.$$

Como $R_1 \perp R_3$, $R_2 \perp R_3$ e $R_3^T R_3 = 1$ temos que

$$u_0 = a_1'^T R_3. \tag{3.11}$$

Determinando f_2, R_2 e t_2

Temos que

$$f_2 R_2^T + v_0 R_3^T = a_2'^T \Rightarrow f_2 R_2^T = a_2'^T - v_0 R_3^T \| f_2 \| = \| a_2'^T - v_0 R_3^T \|$$

Podemos escolher o sinal de f_2. Optamos por escolher f_2 positivo, ou seja

$$f_2 = \| a_2'^T - v_0 R_3^T \|. \tag{3.12}$$

Para determinarmos R_2 e t_2 utilizamos que

$$R_2^T = \frac{1}{f_2} \left(a_2'^T - v_0 R_3^T \right) \tag{3.13}$$

e

$$t_2 = \frac{1}{f_2}\left(b'_2 - v_0 t_3\right). \tag{3.14}$$

Determinando R_1

R_1 pode ser obtido diretamente a partir de R_2 e R_3 considerando-se que R é uma rotação. Temos então que

$$R_1 = R_2 \times R_3. \tag{3.15}$$

Determinando f_1, s e t_1

Temos que

$$f_1 R_1^T + s R_2^T + u_0 R_3^T = a_1'^T \Rightarrow f_1 R_1^T R_2 + s R_2^T R_2 + u_0 R_3^T R_2 = a_1'^T R_2 \tag{3.16}$$

Como $R_1 \perp R_2$, $R_2 \perp R_3$ e $R_2^T R_2 = 1$ temos que

$$s = a_1'^T R_2. \tag{3.17}$$

Com um raciocínio análogo podemos concluir que

$$f_1 = a_1'^T R_1. \tag{3.18}$$

O valor de t_1 pode ser obtido observando-se que

$$t_1 = \frac{1}{f_1}\left(b'_1 - u_0 t_3 - s t_2\right). \tag{3.19}$$

Corrigindo o sinal de λ

Estamos interessados em definir um sistema de coordenadas associado à câmera que satisfaça $\hat{\imath} \times \hat{\jmath} = \hat{k}$, onde $\hat{k}$ especifica a direção e o sentido de visada da câmera. Por outro lado, queremos que o sistema de coordenadas da imagem seja definido com origem no canto esquerdo inferior, como ilustrado na figura 3.2. Para que essas definições sejam consistentes é preciso que tenhamos $f_2 > 0$ e $f_1 < 0$ [1]

[1] Aqui estamos considerando que a orientação dos sistemas de coordenadas da câmera e da cena são opostas. Essa consideração não foi feita no capítulo anterior, isso significa que, se f_1 for estimado como descrito nesta seção, deve-se trocar o seu sinal na equação (2.14) para que o OpenGL funcione apropriadamente.

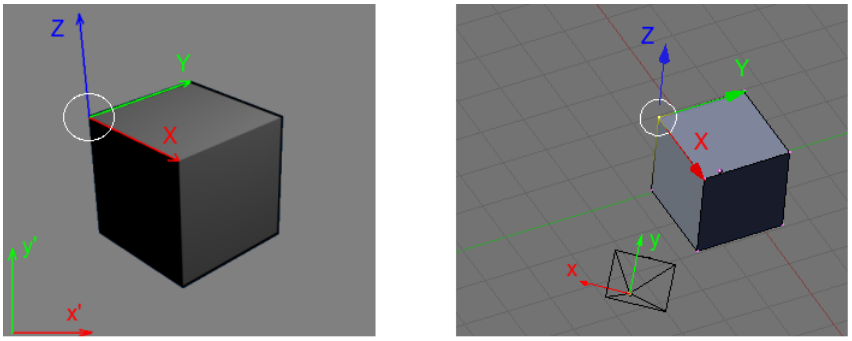

Figura 3.2: (a) exibe a imagem de um cubo correspondente à descrição da cena apresentada em (b). O sistema de coordenadas da imagem (a) é definido com uma orientação diferente do sistema da câmera apresentado em (b). Com essa definição o sinal de f_1 precisa ser negativo.

O procedimento de cálculo de parâmetros intrínsecos descrito anteriormente encontra uma solução que satisfaz $f_2 > 0$, já o sinal de f_1 pode ser tanto positivo como negativo. Se f_1 for negativo podemos interpretar esse fato como uma escolha inapropriada para o sinal de λ em (3.7). Essa mudança do sinal de λ corresponde à seguinte transformação sobre a resposta encontrada:

$$\begin{pmatrix} f_1 & s & u_0 \\ 0 & f_2 & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} R_1^T & t_1 \\ R_2^T & t_2 \\ R_3^T & t_3 \end{pmatrix} \mapsto \begin{pmatrix} -f_1 & s & u_0 \\ 0 & f_2 & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} R_1^T & t_1 \\ -R_2^T & -t_2 \\ -R_3^T & -t_3 \end{pmatrix}.$$

3.3 Câmera para síntese de imagens

Os parâmetros intrínsecos obtidos pela fatoração de uma câmera projetiva genérica apresentam um grau de liberdade que não existe nos modelos de câmera empregados em síntese de imagens. Se a matriz dos parâmetros intrínsecos de uma câmera for

$$K = \begin{pmatrix} f_1 & s & u_0 \\ 0 & f_2 & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.20}$$

temos que ela só pode ser enquadradas nos modelos tradicionais de síntese de imagens no caso em que $s = 0$. Como estimamos a câmera projetiva genérica a partir de medições feitas por uma câmera que foi manufaturada com o objetivo de apresentar a propriedade $s = 0$, temos que a matriz K obtida será definida com um valor pequeno para $|s|$. Sendo assim, o efeito da transformação K' obtida pela substituição do valor de s por zero em K, é semelhante ao da transformação K, sendo que a primeira pode ser adaptada ao propósito de síntese de imagens. Não utilizaremos a matriz K' diretamente, mostraremos como utiliza-la como ponto de partida para um algoritmo de otimização que encontrará a solução que procuramos.

3.4 Calibração por otimização restrita

Consideremos o seguinte problema

Problema 3.2. *Seja Ω o espaço das câmeras projetivas tais que suas matrizes de parâmetros intrínsecos satisfazem a restrição $s = 0$, seguindo a notação da equação (3.20). Conhecidas as projeções $x_1, ..., x_n$, com $x_i \in \mathbb{R}P^2$, correspondentes aos pontos $X_1, ..., X_n$, com $X_i \in \mathbb{R}P^3$, definidas no referencial do objeto calibrador. Determinar a transformação $P = K[R|t] \in \Omega$, tal que $\sum_{i=0}^{n} d(PX_i, x_i)^2$ é mínimo.*

Essa formulação para o problema de calibração de uma câmera apresenta dois aspectos importantes

1. A função objetivo possui um significado geométrico baseado no erro de reprojeção dos pontos $X_1, ..., X_n$, que é mais natural do que o erro algébrico definido na seção 3.1.3.

2. A solução encontrada é ótima no espaço Ω, ou seja, a matriz K é a melhor escolha de parâmetros intrínsecos que pode ser feita para explicar as projeções de $X_1, ..., X_n$, mantendo a compatibilidade com o modelo empregado em síntese de imagens.

Descreveremos a seguir os métodos de otimização Gauss-Newton e Levenberg-Marquardt, que podem ser utilizados para resolver esse problema. Na prática o algoritmo empregado é o Levenberg-Marquardt por apresentar uma melhor condição de

convergência. Para utiliza-los é necessário que seja conhecido um elemento de Ω próximo da solução ótima. Uma câmera com essa propriedade pode ser encontrada da seguinte maneira:

1. Estima-se uma câmera projetiva genérica P, como descrito na seção 3.1.3;
2. Fatora-se P na forma $K\,[R|t]$, como descrito na seção 3.2;
3. Faz-se a substituição do parâmetro intrínseco s da matriz K por zero, como descrito na seção 3.3, obtendo-se assim a câmera $K'\,[R|t] \in \Omega$.

Descreveremos primeiro o método Gauss-Newton pois Levenberg-Marquardt é uma modificação do mesmo.

3.4.1 Método Gauss-Newton

O método Gauss-Newton tem por objetivo encontrar um mínimo $\hat{x} \in \mathbb{R}^n$ para uma função $g : \mathbb{R}^n \to \mathbb{R}$ definida por $g(x) = \frac{1}{2}\|f(x) - x_0\|^2$, onde $x_0 \in \mathbb{R}^m$, e $f : \mathbb{R}^n \to \mathbb{R}^m$ é uma função definida de forma que próximo de $\hat{x}$ ela é de classe C^2. Tem-se como hipótese que é conhecido um ponto $\kappa_1 \in \mathbb{R}^n$, que é uma estimativa para o mínimo, ou seja, $\|\kappa_1 - \hat{x}\|$ é pequeno.

Podemos definir um polinômio de Taylor associado a g no ponto κ por

$$g\,(\kappa_1 + h) \approx g\,(\kappa_1) + g'\,(\kappa_1) \cdot h + \frac{1}{2} g''\,(\kappa_1) \cdot h \cdot h \tag{3.21}$$

Como g é diferenciável, temos que g assume um mínimo em $\kappa_1 + h$ se e somente se $g'\,(\kappa_1 + h) = 0$. Utilizando uma aproximação de primeira ordem para g obtemos $g'\,(\kappa_1 + h) = g''\,(\kappa_1) \cdot h + g'\,(\kappa_1)$. Logo para encontrar o vetor h que minimiza $g\,(\kappa_1 + h)$ basta resolver o sistema

$$H_g\,(\kappa_1)\,h = -\nabla g\,(\kappa_1)\,. \tag{3.22}$$

Usando o fato que $g(x) = \frac{1}{2}\|f(x) - x_0\|^2$, temos que $g'(x) \cdot u = \langle f'(x) \cdot u, f(x)\rangle$, e conseqüentemente

$$g''(x) \cdot u \cdot v = \langle f''(x) \cdot u \cdot v, f(x)\rangle + \langle f'(x) \cdot u, f'(x) \cdot v\rangle. \tag{3.23}$$

Utilizando uma aproximação de primeira ordem para f, obtemos

$$g''(x) \cdot u \cdot v = \langle f'(x) \cdot u, f'(x) \cdot v \rangle. \tag{3.24}$$

Podemos reescrever matricialmente as relações para a primeira e segunda derivadas obtendo $\nabla g(\kappa_1) = J_f(\kappa_1) f(\kappa_1)$, e $H_g(\kappa_1) = J_f^T(\kappa_1) J_f(\kappa_1)$. Substituindo em (3.22) temos que h pode ser estimado pela resolução do sistema

$$J_f^T(\kappa_1) J_f(\kappa_1) h = -J_f(\kappa_1) f(\kappa_1). \tag{3.25}$$

Devido as aproximações que estão sendo feitas, tem-se em geral que $\kappa_1 + h$ não é um mínimo de g. O que se faz é definir $\kappa_2 = \kappa_1 + h$ como uma nova estimativa para o mínimo, e repete-se o processo até que se obtenha um κ_i tal que $\|\nabla g(\kappa_i)\|$ seja considerado suficientemente pequeno.

A convergência, ou não, da seqüência (κ_n) para $\hat{x}$ vai depender da qualidade da estimativa inicial κ_1. Entretanto, quando essa convergência ocorre, pode-se mostrar, como pode ser visto em [5], que ela é de ordem dois, ou seja, $\exists c \in \mathbb{R}$ tal que $\|\kappa_{i+1} - \hat{x}\| \leqslant c\|\kappa_i - \hat{x}\|^2$.

3.4.2 Algoritmo Levenberg-Marquardt

O algoritmo Levenberg-Marquardt é uma adaptação do método Gauss-Newton utilizada quando a estimativa inicial para o mínimo não é suficientemente boa para garantir sua convergência. A idéia do algoritmo é fazer uma transição gradativa de uma otimização por descida pelo gradiente para o método Gauss-Newton, conforme a estimativa do ponto ótimo se torna cada vez melhor. No algoritmo Levenberg-Marquadt tem-se que $\kappa_{i+1} = \kappa_i + h$, onde h é solução do sistema

$$\left(J_f^T(\kappa_i) J_f(\kappa_i) + \lambda I\right) h = -J_f(\kappa_i) f(\kappa_i). \tag{3.26}$$

Tem-se que $\lambda \in \mathbb{R}$ é um valor que pode ser modificado a cada iteração. λ é inicializado com um certo valor, e a cada iteração λ pode ser multiplicado ou dividido por um certo fator, com o objetivo de garantir que o vetor h obtido produza uma redução no valor função objetivo.

O aumento no valor de λ, faz com que o termo λI aumente sua importância, quando comparado com $J_f^T(\kappa_i) J_f(\kappa_i)$. Isso faz com que a solução do sistema se aproxime de $-\lambda^{-1} J_f(\kappa_i) f(\kappa_i) = -\lambda^{-1} \nabla g(\kappa_i)$, fazendo com que o algoritmo passe a ter um comportamento semelhante a de um algoritmo de descida pelo gradiente.

Quando κ_i se torna mais próximo da solução ótima, o valor de λ vai se reduzindo, fazendo com que o algoritmo passe a ter um comportamento semelhante ao método Gauss-Newton, o que acelera a convergência.

3.4.3 Adaptação dos algoritmos ao problema

Podemos empregar o algoritmo Levenberg-Marquardt na solução do problema 3.2. Para isso, usando a notação estabelecida nesse problema, vamos definir uma aplicação $\psi : U \subset \mathbb{R}^{10} \to \left(\mathbb{R}^2\right)^m$ como

$$\psi_i(z) = P(z) X_i, \tag{3.27}$$

para i $\in \{1, \cdots, m\}$, onde U e $P : U \to \Omega$ são definidos de forma que P seja uma aplicação sobrejetora no subconjunto de Ω das câmeras que aplicadas a $X_1, \cdots, X_m$ geram projeções que são pontos afins de $\mathbb{R}P^2$.

Para resolver o problema 3.2 basta utilizar o algoritmo Levenberg-Marquardt para encontrar o mínimo da função $g : U \to \mathbb{R}$ definida por

$$g(z) = \frac{1}{2} \|\psi(z) - (x_1, \cdots, x_m)^T\|^2. \tag{3.28}$$

Destacamos que quando consideramos a imagem de cada ψ_i como um vetor de $\mathbb{R}^2$, estamos colocando embutida a transformação definida por $(x, y, z)^T \mapsto \left(\frac{x}{z}, \frac{y}{z}\right)^T$, que faz a conversão de coordenadas de pontos afins de $\mathbb{R}P^2$ para coordenadas do $\mathbb{R}^2$, e estamos fazendo o mesmo com as coordenadas homogêneas de $x_1, \cdots, x_m$.

Apresentaremos na seção 3.4.5 uma definição de P que faz com que a aplicação ψ seja de classe C^2 para quase todos os pontos de U. Possibilitando o emprego do algoritmo Levenberg-Marquardt. Antes mostraremos como obter uma parametrização para uma rotação via especificação de um eixo e de um ângulo de rotação.

3.4.4 Parametrização de rotações

A rotação de ângulo $\theta \in \mathbb{R}$, ao redor de um eixo especificado pelo vetor $\omega = (\omega_1, \omega_2, \omega_3)^T \in \mathbb{R}^3$, com $\|\omega\| = 1$, é dada pela transformação linear $R : \mathbb{R}^3 \to \mathbb{R}^3$ cuja representação matricial é [2]:

$$R = \begin{pmatrix} \omega_1^2 + C\left(1-\omega_1^2\right) & \omega_1\omega_2\left(1-C\right) - \omega_3 S & \omega_1\omega_3\left(1-C\right) + \omega_2 S \\ \omega_1\omega_2\left(1-C\right) + \omega_3 S & \omega_2^2 + C\left(1-\omega_2^2\right) & \omega_2\omega_3\left(1-C\right) - \omega_1 S \\ \omega_1\omega_3\left(1-C\right) - \omega_2 S & \omega_2\omega_3\left(1-C\right) + \omega_1 S & \omega_3^3 + C\left(1-\omega_3^2\right) \end{pmatrix}, \quad (3.29)$$

onde $C = cos\theta$, e $S = sen\theta$.

Reciprocamente, pode-se obter o eixo de rotação ω e o ângulo θ a partir da transformação R. Para isso, basta observar que o sub-espaço gerado por esse eixo é invariante por R, ou seja, ω é um auto-vetor de R. Além disso, a restrição de R ao subespaço gerado por ω é a transformação identidade, logo o auto-valor associado a ω é unitário. Ou seja, para determinar ω basta encontrar uma solução não trivial para a equação $(R - I)\,\omega = 0$, que pode ser obtida pela proposição 3.1.

O ângulo θ, pode ser determinado utilizando-se que

$$cos\theta = \frac{1}{2}\left(tr(R) - 1\right), \quad (3.30)$$

$$sen\theta = \frac{1}{2}\langle \omega, \tau \rangle, \quad (3.31)$$

onde $\tau = (R_{32} - R_{23}, R_{13} - R_{31}, R_{21} - R_{12})^T$.

Para se obter uma representação explícita para os três graus de liberdade associados a uma rotação, basta utilizar o vetor $\omega' \in \mathbb{R}^3$ definido por $\omega' = \theta\omega$.

A obtenção de ω e θ a partir de ω' é feita da seguinte maneira

1. Caso $\|\omega'\| \neq 0$, então $\omega = \frac{\omega'}{\|\omega'\|}$ e $\theta = \|\omega'\|$;
2. Caso $\|\omega'\| = 0$, então $\theta = 0$ e ω pode ser qualquer vertor unitário.

3.4.5 Parametrização do espaço de câmeras

Utilizando a parametrização das matrizes de rotação apresentada acima podemos definir a aplicação $P : U \subset \mathbb{R}^{10} \to \Omega$, utilizada para definir ψ na equação (3.27), satisfazendo as propriedades exigidas na seção 3.4.3.

$$P\left(f_1, f_2, u_0, v_0, t_1, t_2, t_3, \omega_1, \omega_2, \omega_3\right) = \begin{pmatrix} f_1 R_1^T\left(\omega\right) + u_0 R_3^T\left(\omega\right) & f_1 t_1 + u_0 t_3 \\ f_2 R_2^T\left(\omega\right) + v_0 R_3^T\left(\omega\right) & f_2 t_2 + v_0 t_3 \\ R_3^T\left(\omega\right) & t_3 \end{pmatrix}, \quad (3.32)$$

onde $\begin{pmatrix} R_1^T\left(\omega\right) \\ R_2^T\left(\omega\right) \\ R_3^T\left(\omega\right) \end{pmatrix}$ é a matriz que, para $\|\omega\| \neq 0$, representa uma rotação no sentido anti-horário de $\|\omega\|$ radianos ao redor do eixo $\omega = \left(\omega_1, \omega_2, \omega_3\right)^T$. E para $\|\omega\| = 0$ é a matriz identidade.

Como $\forall z \in U$, $P\left(z\right) X_i$ é um ponto afim de $\mathbb{R}P^2$, temos que $\psi_i\left(z\right)$ fica sendo definida por

$$\left(\frac{f_1 R_1^T\left(\omega\right) X_i + u_0 R_3^T\left(\omega\right) X_i + f_1 t_1 + u_0 t_3}{R_3^T\left(\omega\right) X_i + t_3}, \frac{f_2 R_2^T\left(\omega\right) X_i + v_0 R_3^T\left(\omega\right) X_i + f_2 t_2 + v_0 t_3}{R_3^T\left(\omega\right) X_i + t_3}\right)^T$$

Temos então que, ψ é de classe C^∞ para os pontos de U que satisfazem $\|\omega\| \neq 0$, pois as coordenadas de cada ψ_i são frações em que o numerador e o denominador são formados por polinômios somados a produtos de polinômios por fatores da forma $\frac{1}{\|\omega\|^2}, cos\|\omega\|$, $\frac{sen\|\omega\|}{\|\omega\|}$ ou $(1 - cos\|\omega\|)$.

3.4.6 Pontos problemáticos da parametrização

O algoritmo Levenberg-Marquardt avalia P em uma seqüência de elementos do espaço euclidiano $\mathbb{R}^{10}$. Em princípio, não podemos garantir que ele não tentará avaliar P fora de seu domínio. Além disso, para que ele possa ser aplicado no nosso caso, precisamos garantir que ψ seja de classe C^2 nas proximidades da região de convergência da seqüência.

Na seção anterior mostramos que ψ é de classe C^∞ para os pontos de $U - W$, onde W são os pontos de U correspondentes a câmeras cuja orientação é descrita por um vetor $\omega = (0, 0, 0)^T$. Veremos agora porque não precisamos nos preocupar com o fato de ψ não ser definida fora de U, e de não ser de classe C^2 em W.

Pontos fora de U

As câmeras que não são parametrizadas por pontos de U são aquelas que satisfazem $R_3^T(\omega) X_i + t_3 = 0$, para algum $i \in \{1, \cdots, m\}$. Essas são as configurações que fazem com que algum dos X_i não possuam projeção, que ocorre quando a coordenada z de X_i é nula no referencial da câmera. Com efeito, basta lembrar que $t_3 = -R_3^T c$, onde c é a posição do centro de projeção, e observar que

$$R_3^T(\omega) X_i + t_3 = 0 \Leftrightarrow R_3^T(\omega)(X_i - c) = 0 \Leftrightarrow [R(\omega)(X_i - c)]_3 = 0.$$

Essa região de $\mathbb{R}^{10}$ em que ψ não é definida, e nem suas derivadas, não gera problemas durante a execução do algoritmo Levenberg-Marquardt, pois a função objetivo assume valores muito elevados nos pontos de U que pertencem a pequenas vizinhanças dessas configurações, pois o erro de reprojeção associado a algum dos X_i é muito grande. Conseqüentemente as seqüências de configurações geradas pelas iterações do algoritmo não devem se aproximar dessa região.

Pontos pertencentes a W

Em relação ao conjunto W temos o seguinte:

1. W tem medida nula em $\mathbb{R}^{10}$, logo é improvável que a seqüência gerada pelo algoritmo Levenberg-Marquardt contenha algum elemento desse conjunto;

2. Pode-se escolher o sistema de coordenadas do objeto calibrador de forma que a câmera esteja afastado de uma configuração da forma $K[I|t]$, como nos exemplos da Figura 7.2, do Capítulo 7. Dessa forma, a estimativa inicial fornecida ao Levenberg-Marquardt deve estar muito afastada de W, e a seqüência provavelmente deve convergir sem se aproximar do conjunto W;

3. Para representar câmeras da forma $K[I|t]$, não é necessário utilizar um elemento de W. Pode-se escolher outro elemento de U cuja rotação seja da forma $2k\pi$, com $k \in \mathbb{N}$.

Capítulo 4

Calibração de pares de câmeras

O objetivo desse capítulo é descrever um algoritmo que determina o posicionamento relativo entre as câmeras que foram utilizadas na captação de duas imagens. Mais precisamente, estamos interessados em resolver o seguinte problema

Problema 4.1. *Dado um conjunto* $\{(x_1, \hat{x}_1), (x_2, \hat{x}_2), ..., (x_n, \hat{x}_n)\}$*, com* $(x_i, \hat{x}_i) \in \mathbb{R}P^2 \times \mathbb{R}P^2$*, que correspondem às projeções em um par de imagens* I_1 *e* I_2*, de um conjunto de pontos da cena* $\{X_1, X_2, ..., X_n\}$*, com* $X_i \in \mathbb{R}P^3$*. Determinar o posicionamento relativo entre as câmeras que captaram* I_1 *e* I_2*, supondo-se que os parâmetros intrínsecos de ambas são conhecidos.*

Os elementos do par $(x_i, \hat{x}_i)$ são chamados de pontos homólogos associados a X_i.

4.1 Representação do posicionamento relativo

Para representar o posicionamento relativo entre duas câmeras assumiremos, sem perda de generalidade, que uma das câmeras é $K_1 [I|0]$, que corresponde a uma câmera posicionada na origem com direção de visada na direção do eixo-z. Dessa maneira os parâmetros extrínsecos da outra câmera, $K_2 [R|t]$, especificam o posicionamento relativo entre elas.

Um fato importante é que as projeções de um conjunto de pontos da cena $\{X_1, X_2, \cdots, X_n\}$, com $X_i \in \mathbb{R}^3$, relativas às câmeras $K_1[I|0]$ e $K_2[R|t]$ são iguais às projeções do conjunto $\{\lambda X_1, \lambda X_2, \cdots, \lambda X_n\}$ relativas às câmeras $K_1[I|0]$ e $K_2[R|\lambda t]$, com $\lambda \in \mathbb{R}^+$. Com efeito, basta observar as seguintes igualdades:

$$\begin{aligned} K_1[I|0]\left(\lambda X_i^T, 1\right)^T &= K_1(\lambda X_i) = K_1(X_i) = K_1[I|0]\left(X_i^T, 1\right)^T, \\ K_2[R|\lambda t]\left(\lambda X_i^T, 1\right)^T &= K_2(R(\lambda X_i) + \lambda t) = K_2\lambda(RX_i + t) = \\ &= K_2(RX_i + t) = K_2[R|t]\left(X_i^T, 1\right)^T. \end{aligned}$$

Isso mostra que o problema 4.1 é definido com uma ambigüidade de escala, pois o valor de $\|t\|$ não pode ser determinado.

4.2 Movimento rígido

A proposição abaixo, apresentada em [14], estabelece uma restrição para as coordenadas definidas em dois referenciais do $\mathbb{R}^3$, que estão relacionados por um movimento rígido.

Proposição 4.1. *Seja $X, \hat{X} \in \mathbb{R}^3$ definidos de forma que $\hat{X} = RX + t$, onde R é uma matriz de rotação, e $t \in \mathbb{R}^3$. Se $[t]_\times : \mathbb{R}^3 \to \mathbb{R}^3$ é o operador linear definido por $[t]_\times(x) = t \times x$, então vale a relação $\hat{X}^T\left([t]_\times R\right)X = 0$.*

Demonstração

Usando o fato do vetor $\hat{X} \times t$ ser perpendicular tanto a $\hat{X}$ quanto a t, temos que

$$\left(\hat{X} \times t\right) \cdot \hat{X} = 0 \text{ e } \left(\hat{X} \times t\right) \cdot t = 0.$$

Como conseqüência vale

$$\hat{X}^T\left([t]_\times R\right)X = \hat{X} \cdot (t \times RX) = \left(\hat{X} \times t\right) \cdot RX = \left(\hat{X} \times t\right) \cdot (RX + t) = \left(\hat{X} \times t\right) \cdot \hat{X} = 0.$$

4.3 Outro modelo de projeção

O efeito obtido pela câmera projetiva, definida pela transformação $[I|0]$, é equivalente ao efeito da transformação $T : \mathbb{R}^3 \to \mathbb{R}P^2$, que aplica cada ponto $x \in \mathbb{R}^3$ em um ponto de $\mathbb{R}P^2$, cujas coordenadas homogêneas são λx, onde $\lambda \in \mathbb{R} - \{0\}$.

Em ambos os casos o efeito é o mesmo da projeção perspectiva $T : \mathbb{R}^3 \to \mathbb{R}^2$ definida por $(x, y, z) \mapsto \left(\frac{x}{z}, \frac{y}{z}\right)$.

4.4 Geometria Epipolar

Geometria Epipolar é o estudo das relações geométricas existentes entre as projeções de um conjunto de pontos sobre duas imagens obtidas por câmeras projetivas.

Será feito a seguir um desenvolvimento algébrico da Geometria Epipolar. Inicialmente será considerado o caso em que as câmeras são da forma $[I|0]$ e $[R|t]$, ou seja, a matriz dos parâmetros intrínsecos de ambas as câmeras é a matriz identidade. Nesse caso as relações de epipolaridade serão caracterizadas pela Matriz Essencial.

Posteriormente será tratado o caso geral, em que as câmeras são da forma $K_1\,[I|0]$ e $K_2\,[R|t]$. Nesse caso as relações de epipolaridade serão caracterizadas pela Matriz Fundamental.

4.4.1 Matriz essencial

Definindo $E = [t]_\times R$, temos pela proposição 4.1 que vale a expressão $\hat{X}^T E X = 0$, que relaciona as coordenadas, em $\mathbb{R}^3$, de um ponto da cena nos referenciais associados as câmeras $[I|0]$ e $[R|t]$. Para se obter uma relação entre as coordenadas das projeções desse ponto nas imagens captadas por essas câmeras basta observar que para todo $\lambda_1, \lambda_2 \in \mathbb{R} - \{0\}$ vale

$$\hat{X}^T E X = 0 \iff \left(\lambda_1 \hat{X}^T\right) E\,(\lambda_2 X) = 0. \tag{4.1}$$

Temos então, pelo que foi apresentado na seção 4.3, que se $x \in \mathbb{R}P^2$ e $\hat{x} \in \mathbb{R}P^2$ são as coordenadas homogêneas das projeções de um ponto da cena obtidas pelas câmeras $[I|0]$ e $[R|t]$ respectivamente, então vale a relação $\hat{x}^T E x = 0$, onde nesse caso tem-se que a matriz E, chamada de matriz essencial, fica definida a menos de um produto por um escalar.

4.4.2 Matriz fundamental

Consideremos agora que $x \in \mathbb{R}P^2$ é a projeção de um ponto $X \in \mathbb{R}P^3$ obtida pela câmera $K\,[R \mid t]$. A projeção do mesmo ponto X obtida pela câmera $[R \mid t]$ é dada

por $K^{-1}x$. Com esse resultado podemos generalizar a relação estabelecida pela matriz essencial para o caso em que as câmeras não possuem a matriz dos parâmetros intrínsecos iguais a I. Mais precisamente, dadas duas câmeras $K_1 [I \mid 0]$ e $K_2 [R \mid t]$, temos que se as projeções de um ponto X relativas a essas câmeras forem respectivamente x e $\hat{x}$, então vale a relação

$$\left(K_2^{-1}\hat{x}\right)^T \left([t]_\times R\right) \left(K_1^{-1}x\right) = 0. \tag{4.2}$$

Essa relação pode ser reescrita como

$$\hat{x}^T F x = 0, \tag{4.3}$$

onde

$$F = K_2^{-T} [t]_\times R K_1^{-1} \tag{4.4}$$

é uma matriz 3×3, denominada matriz fundamental.

4.5 Algoritmo de 8 pontos

O algoritmo de 8 pontos foi apresentado inicialmente em [11]. Sua entrada é um conjunto de pares de pontos homólogos $\{(x_1, \hat{x}_1), (x_2, \hat{x}_2), ..., (x_n, \hat{x}_n)\}$ definidos sobre duas imagens, e sua resposta é a matriz fundamental associada ao par de imagens. Seu nome deve-se ao fato de serem necessários, no mínimo, 8 pares de pontos homólogos para que o algoritmo possa ser executado. Ele é composto por duas etapas:

1. Etapa 1: Determinação da matriz F, que melhor satisfaz $\hat{x}_i^T F x_i = 0$, para todo $i \in \{1, 2, \cdots, n\}$.

2. Etapa 2: Determinação da matriz $\tilde{F}$ que é mais próxima de F, e que satisfaz $det\left(\tilde{F}\right) = 0$. A matriz $\tilde{F}$ é a saída do algoritmo.

Os detalhes de execução das duas etapas, bem como o significado preciso das expressões “melhor satisfaz”e “mais próxima”, serão apresentados a seguir.

4.5.1 Cálculo de F

Considerando que cada um dos elementos de F é uma variável, e que os valores de x_i e $\hat{x}_i$ são conhecidos para cada $i \in \{1, 2, 3, \cdots, n\}$, tem-se que a expressão $\hat{x}_i^T F x_i = 0$ define uma equação linear sobre 9 variáveis.

Se F_0 é uma solução para a equação anterior, então λF_0 também é solução para todo $\lambda \in \mathbb{R} - \{0\}$. Isso mostra que é suficiente utilizar um conjunto de 8 pares de pontos homólogos para formar um sistema linear que permita determinar o valor de F. Por motivos análogos aos apresentados na estimação de câmeras, na seção 3.1.1, tem-se que a solução obtida utilizando-se apenas 8 pares de pontos homólogos não é boa, sendo interessante utilizar-se de um conjunto maior de pontos, convertendo o problema em um problema de otimização.

4.5.2 Usando mais de 8 pontos

Para resolver o sistema linear definido pelas equações $\hat{x}_i^T F x_i = 0$, utilizando mais de 8 pares de pontos homólogos, pode-se reformular o problema como sendo o de encontrar a matriz F_0 que minimiza a função objetivo

$$F \mapsto \sum_{i=1}^{n} \left(\hat{x}_i^T F x_i\right)^2,$$

que pode ser resolvido pela proposição 3.1, bastando para isso ser reescrito na forma $\underset{\|\mathcal{F}\|=1}{min} \|A\mathcal{F}\|$, com $\mathcal{F} = (F_{11}, F_{12}, F_{13}, F_{21}, F_{22}, F_{23}, F_{31}, F_{32}, F_{33})^T$ e A definida por

$$\begin{pmatrix} u'_1u_1 & v'_1u_1 & u_1 & u'_1v_1 & v'_1v_1 & v_1 & u'_1 & v'_1 & 1 \\ u'_2u_2 & v'_2u_2 & u_2 & u'_2v_2 & v'_2v_2 & v_2 & u'_2 & v'_2 & 1 \\ u'_3u_3 & v'_3u_3 & u_3 & u'_3v_3 & v'_3v_3 & v_3 & u'_3 & v'_3 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u'_mu_m & v'_mu_m & u_m & u'_mv_m & v'_mv_m & v_m & u'_m & v'_m & 1 \end{pmatrix},$$

onde $\hat{x}_i = (u_i, v_i, 1)^T$ e $x_i = (u'_i, v'_i, 1)^T$.

A restrição $\|\mathcal{F}\| = 1$ faz sentido pois as matrizes fundamentais são definidas a menos de uma multiplicação por um escalar.

4.5.3 Cálculo de $\tilde{F}$

O objetivo do cálculo de $\tilde{F}$ é obrigar que a resposta do algoritmo de 8 pontos satisfaça uma propriedade importante das matrizes fundamentais, que é o fato delas serem matrizes singulares [8]. Essa restrição não é imposta durante o cálculo de F.

Pode-se definir $\tilde{F}$ como sendo a matriz singular tal que $\|\tilde{F} - F\|$ assume o valor mínimo. Considerando a norma utilizada como sendo a norma de Frobenius, existe uma maneira simples de calcular $\tilde{F}$, que consiste em utilizar diretamente a proposição abaixo, cuja demostração pode ser encontrada em [18].

Proposição 4.2. *Se $Udiag(r, s, t)V^T$ é a decomposição SVD de F, com $r \geqslant s \geqslant t$, então a matriz singular $\tilde{F}$, tal que $\|\tilde{F} - F\|$ é mínima, é dada por $\tilde{F} = Udiag(r, s, 0)V^T$.*

4.6 Algoritmo de 8 pontos normalizado

O algoritmo de 8 pontos é mal condicionado. Uma modificação simples que o torna melhor condicionado é descrita em [9]. A modificação consiste em aplicar duas transformações $H_1 : \mathbb{R}P^2 \to \mathbb{R}P^2$ e $H_2 : \mathbb{R}P^2 \to \mathbb{R}P^2$ aos pontos homólogos do conjunto de entrada $A = \{(x_1, \hat{x}_1), (x_2, \hat{x}_2), ..., (x_n, \hat{x}_n)\}$, transformando-o no conjunto $B = \{(H_1x_1, H_2\hat{x}_1), (H_1x_2, H_2\hat{x}_2), ..., (H_1x_n, H_2\hat{x}_n)\}$, onde H_1 e H_2 são definidas de forma a satisfazerem as seguintes propriedades:

1. H_1 e H_2 são transformações afins que realizam uma translação e um escalamento em $\mathbb{R}^2$;

2. Ambos os conjuntos, $\{H_1x_1, H_1x_2, \cdots, H_1x_n\}$ e $\{H_2\hat{x}_1, H_2\hat{x}_2, \cdots, H_2\hat{x}_n\}$, têm o ponto $(0, 0)^T \in \mathbb{R}^2$ como sendo o centróide;

3. O valor de *RMS* das distâncias dos pontos de ambos os conjuntos, $\{H_1x_1, H_1x_2, \cdots, H_1x_n\}$ e $\{H_2\hat{x}_1, H_2\hat{x}_2, \cdots, H_2\hat{x}_n\}$, ao ponto $(0, 0)^T$ é $\sqrt{2}$.

O algoritmo de 8 pontos estima uma matriz fundamental F' de forma bem condicionada ao utilizar B como entrada. Temos então que para todo par de pontos homólogos $(x, \hat{x}) \in A$ vale a expressão

$$\hat{x}^T \left(H_2^T F' H_1\right) x = \left(H_2 \hat{x}\right)^T F' \left(H_1 x\right) = 0.$$

Isso mostra que a matriz fundamental que estabelece a epipolaridade dos pontos de A é definida por $F = H_2^T F' H_1$.

4.7 Determinando os parâmetros extrínsecos

Se os parâmetros intrínsecos de uma câmera são conhecidos, é possível, a partir de uma matriz fundamental F, determinar as possíveis posições relativas entre duas câmeras que explicam essa matriz fundamental.

Dada a matriz fundamental $F = K_2^{-T} [t]_\times R K_1^{-1}$ que estabelece a relação de epipolaridade das projeções obtidas pelas câmeras $K_1 [I \mid 0]$ e $K_2 [R \mid t]$, podemos definir uma matriz essencial

$$E = K_2^T F K_1, \tag{4.5}$$

que relaciona as projeções obtidas pelas câmeras $[I \mid 0]$ e $[R \mid t]$.

Sendo assim, a matriz $E = [t]_\times R$ é o produto da matriz anti-simétrica $[t]_\times$, pela matriz de rotação R. A determinação dos possíveis valores de t e R fica resolvida pela proposição abaixo, cuja demonstração pode ser encontrada em [8].

Proposição 4.3. *Supondo que a decomposição SVD de uma matriz essencial E é igual a $U diag(1,1,0) V^T$, existem duas maneiras de fatorar E, de forma que $E = SR$, onde S é uma matriz ainti-simétrica e R é uma matriz de rotação. Tem-se que $S = UZU^T$ e $R = UWV^T$ ou $R = UW^T V^T$, onde*

$$W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ e } Z = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

A proposição 4.3 mostra que existem duas possíveis escolhas para a matriz de rotação R. Para determinarmos quais são os possíveis vetores t, basta levar em conta os seguintes fatos:

1. $[t]_\times t = t \times t = 0$;

2. O vetor t é definido a menos de uma multiplicação por um escalar.

Usando a notação da proposição, temos pelo primeiro fato que, todo vetor t deve pertencer ao núcleo de $[t]_\times$. Tendo em vista que $[t]_\times = S = UZU^T$, conclui-se que todo vetor t deve ser da forma

$$t = \lambda U\,(0,0,1)^T\,, \tag{4.6}$$

onde $\lambda \in \mathbb{R}$.

O segundo fato, demonstrado na seção 4.1, implica que t pode ser qualquer elemento da forma $\lambda U\,(0,0,1)^T$, com $\lambda \in \mathbb{R}$. Se nos restringirmos aos casos em que $\|t\| = 1$, temos que t pode ser $U\,(0,0,1)^T$ ou $-U\,(0,0,1)^T$.

4.7.1 Adicionando recorte ao modelo

Podemos concluir que, sendo conhecida uma matriz fundamental F, que relaciona projeções obtidas por um par de câmeras P_1 e P_2, cujos parametros intrínsecos são definidos por matrizes K_1 e K_2. Se $P_1 = K_1\,[I \mid 0]$ então P_2 pode ser definida de quatro maneiras:

$$K_2\left[UWV^T \mid U\,(0,0,1)^T\right],$$
$$K_2\left[UW^TV^T \mid U\,(0,0,1)^T\right],$$
$$K_2\left[UWV^T \mid -U\,(0,0,1)^T\right],$$
$$K_2\left[UW^TV^T \mid -U\,(0,0,1)^T\right],$$

onde U e W podem ser calculados a partir de F utilizando-se a equação (4.5) e a proposição 4.3. Ao afirmarmos que existem apenas essas quatro soluções, estamos considerando que está implícita a indeterminação da distância entre os centros de projeção de P_1 e P_2.

O modelo de câmera que estamos utilizando não descreve a operação de recorte em relação à pirâmide de visão. O resultado disso é que, apenas uma dessas quatro configurações de câmeras é fisicamente realizável, como exemplificado pela Figura 4.1.

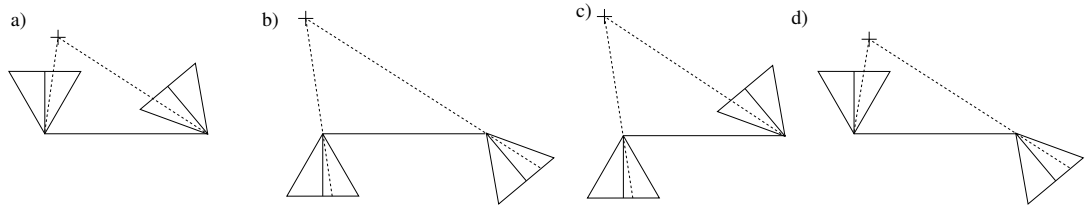

Figura 4.1: Embora existam quatro configurações que explicam projetivamente o par de pontos homólogos, apenas em (a) o ponto projetado está posicionado à frente de ambas as câmeras.

A solução para esse problema consiste em descartar as soluções que fazem com que a reconstrução tridimensional de pontos homólogos possua a coordenada z negativa para algum dos referenciais definidos pelas câmeras [8]. Mostraremos como obter a reconstrução tridimensional de um ponto a partir de suas projeções na próxima seção.

4.7.2 Reconstrução tridimensional

Sejam $x_1 \in \mathbb{R}P^2$ e $x_2 \in \mathbb{R}P^2$ as projeções de um ponto $X \in \mathbb{R}P^3$ relativas as câmeras P_1 e P_2, ou seja, $x_1 = P_1X$ e $x_2 = P_2X$. Mostraremos agora como determinar X quando x_1, x_2, P_1 e P_2 são conhecidos.

Interpretando $x_1 = (u, v, 1)^T$ e PX como vetores do $\mathbb{R}^3$, temos que $x_1 \times (P_1X) = 0$. Chamando de P_i^n a n-ésima linha de P_i, pode-se reescrever essa expressão como o seguinte conjunto de três equações lineares em X, onde duas são linearmente independentes

$$u\left(P_1^3X\right) - \left(P_1^1X\right) = 0, \tag{4.7}$$

$$v\left(P_1^3X\right) - \left(P_1^2X\right) = 0, \tag{4.8}$$

$$u\left(P_1^2X\right) - v\left(P_1^1X\right) = 0. \tag{4.9}$$

Analogamente temos que $x_2 = (u\prime, v\prime, 1)^T$ pode ser utilizado para obtermos mais outras duas equações lineares em X, e linearmente independentes, bastando observar que $x_2 \times (P_2X) = 0$. Agrupando quatro dessas equações obtemos um sistema linear

homogêneo da forma $AX = 0$ onde

$$A = \begin{pmatrix} uP_1^3 - P_1^1 \\ vP_1^3 - P_1^2 \\ u\prime P_2^3 - P_2^1 \\ v\prime P_2^3 - P_2^2 \end{pmatrix}. \tag{4.10}$$

Esse é um sistema linear de quatro equações sobre as quatro coordenadas homogêneas de X, logo é um sistema linear super-determinado, que pode ser convertido para o problema de otimização $\underset{\|X\|=1}{min} \|AX\|$, cuja solução é dada pela proposição 3.1.

Capítulo 5

Acompanhamento de pontos

O próximo capítulo apresentará um processo de calibração para famílias de câmeras. Tal processo precisa que seja realizada a correspondência entre projeções de diversos pontos da cena sobre diversos quadros de um vídeo. Tendo em vista que mesmo vídeos de curta duração são formados por centenas de quadros, é necessário que essa correspondência seja feita de forma automática. Descreveremos neste capítulo o algoritmo Kanade-Lucas-Tomasi, que será utilizado para resolver esse problema. Uma descrição mais detalhada pode ser encontrada em [16] e [12].

5.1 Definições

Adotaremos as seguintes definições:

1. **Imagem**
   Uma imagem é uma função $I : [a, b] \times [c, d] \to \mathbb{R}$. Nesse caso, estamos considerando um modelo para imagens de tons de cinza, em que para cada ponto do suporte $[a, b] \times [c, d]$ associa-se um valor de brilho.

2. **Vídeo**
   Um vídeo é uma família finita de imagens $(I)_n = (I_1, ..., I_n)$, onde cada imagem I_k corresponde a um quadro captado por uma câmera. Tem-se ainda que a ordem definida pela indexação dos quadros corresponde a ordem em que os quadros foram captados pela câmera.

3. **Janela**

   Uma janela de uma imagem $I : [a, b] \times [c, d] \to \mathbb{R}$ é uma imagem $I\,|_w$ obtida pela restrição do domínio de I a um pequeno retângulo $w = [a', b'] \times [c', d'] \subset [a, b] \times [c, d]$.

5.2 Algoritmo Kanade-Lucas-Tomasi

Kanade-Lucas-Tomasi (KLT) é um algoritmo capaz de acompanhar janelas em um vídeo. Dado um vídeo $(I)_n$, ele procura localizar janelas em um quadro I_{j+1} que estejam correlacionadas por uma translação com janelas de I_j. Mais precisamente, o algoritmo é capaz de determinar um vetor $h \in \mathbb{R}^2$, chamado de disparidade, tal que

$$\forall x \in w, I_{j+1}\,|_{w'}\,(x + h) = I_j\,|_w\,(x) + \eta\,(x)\,, \tag{5.1}$$

onde w' é o retângulo obtido adicionando h aos vértices de w, e $\eta : w \to \mathbb{R}^+$ é uma função que define o erro pontual de correlação entre as janelas. O algoritmo busca então determinar a disparidade h que minimiza esse erro sobre toda a janela.

A utilidade do correlacionamento de janelas para os nossos objetivos decorre do fato de que janelas que sejam semelhantes, e estejam próximas em quadros consecutivos, possuem uma grande chance de corresponderem à projeção de um mesmo conjunto de pontos da cena tridimensional. Isso significa que, sendo $x_0 \in w$, é razoável utilizar $x_0 + h$ como estimativa para seu homólogo em I_{j+1}. No processo de calibração apresentado no próximo capítulo são utilizados os centros de janelas correlacionadas pelo KLT como pontos homólogos. Dessa forma, o problema de acompanhamento de pontos é convertido em um problema de acompanhamento de janelas.

5.3 Acompanhamento de janelas

Usando a notação estabelecida na equação (5.1), e tendo sido fixado um vetor disparidade h, pode-se definir uma medida para o erro de correlacionamento

$$E = \int_w \eta\,(x)^2\,dx. \tag{5.2}$$

Dessa forma, o problema de determinação da disparidade pode ser formalizado através do seguinte problema de otimização:

Problema 5.1. *Encontrar um vetor* $h \in \mathbb{R}^2$ *que minimiza* $\int_w \left[I_{j+1}\left(x+h\right) - I_j\left(x\right)\right]^2 dx$, *onde* w *é o retângulo que define a janela em* I_j.

Realizando a mudança de variáveis $\nu = x + h$, temos que esse problema é equivalente ao de encontrar o vetor $h \in \mathbb{R}^2$ que minimiza

$$\int_w \left[I_{j+1}\left(\nu\right) - I_j\left(\nu - h\right)\right]^2 d\nu. \tag{5.3}$$

Assumindo que I_{j+1} é diferenciável, e que a disparidade entre quadros consecutivos é pequena, podemos fazer a seguinte aproximação

$$I_j\left(\nu - h\right) \approx I_j\left(\nu\right) - I_j'\left(\nu\right) \cdot h. \tag{5.4}$$

Com isso temos que a função objetivo pode ser reescrita como

$$h \mapsto \int_w \left[\Phi\left(\nu\right) - I_j'\left(\nu\right) \cdot h\right]^2 d\nu, \text{ onde } \Phi\left(\nu\right) = I_{j+1}\left(\nu\right) - I_j\left(\nu\right).$$

Essa aplicação possui um mínimo em um ponto crítico $h = \left(h_1, h_2\right)^T$ que satisfaz

$$\forall u \in \mathbb{R}^2, \int_w \left[\Phi\left(\nu\right) - I_j'\left(\nu\right) \cdot h\right] \left(I_j' \cdot u\right) d\nu = 0. \tag{5.5}$$

Em particular, essa propriedade vale quando u são os vetores da base canônica $\left(1,0\right)^T$ e $\left(0,1\right)^T$, permitindo que reescrevamos a expressão acima em termos de derivadas parciais, obtendo o seguinte sistema linear, que nos permite determinar h:

$$\left[\int_w \left(\frac{\partial I_j}{\partial x_1}\left(\nu\right)\right)^2 d\nu\right] h_1 + \left[\int_w \frac{\partial I_j}{\partial x_1}\left(\nu\right) \frac{\partial I_j}{\partial x_2}\left(\nu\right) d\nu\right] h_2 = \int_w \Phi\left(\nu\right) \frac{\partial I_j}{\partial x_1}\left(\nu\right) d\nu \tag{5.6}$$

$$\left[\int_w \left(\frac{\partial I_j}{\partial x_2}\left(\nu\right)\right)^2 d\nu\right] h_2 + \left[\int_w \frac{\partial I_j}{\partial x_1}\left(\nu\right) \frac{\partial I_j}{\partial x_2}\left(\nu\right) d\nu\right] h_1 = \int_w \Phi\left(\nu\right) \frac{\partial I_j}{\partial x_2}\left(\nu\right) d\nu \tag{5.7}$$

5.4 Escolha das janelas

Além de definir um algoritmo de acompanhamento de janelas, temos que o algoritmo KLT define um processo automático de seleção de janelas a serem acompanhadas. Esse processo de seleção é baseado em um critério definido de forma que a solução do sistema linear formado pelas equações (5.6) e (5.7) possa ser obtida com precisão.

Consideremos o sistema escrito na forma $Ah = b$. Para que sua solução possa ser obtida com precisão é necessário que ele seja bem condicionado, e que os coeficientes da matriz A estejam definidos acima do nível de ruído da imagem.

Para que o sistema seja bem condicionado é necessário que os dois auto-valores de A, λ_1 e λ_2, sejam da mesma ordem de grandeza. Na prática, isso sempre ocorre, tendo em vista que o valor do brilho em cada ponto da imagem é limitado.

Para que os coeficientes de A estejam definidos acima do nível de ruído da imagem é necessário que λ_1 e λ_2 não sejam pequenos. Sendo assim, o algoritmo KLT utiliza o valor $min\{\lambda_1, \lambda_2\}$ como medida de qualidade para o acompanhamento de uma janela.

A escolha das m janelas do quadro $I_k : U \rightarrow \mathbb{R}$ que são mais bem acompanháveis faz-se através da comparação dos valores de qualidade de acompanhamento considerando-se todas as possíveis escolhas de janelas $w \subset U$. São escolhidas as janelas de melhor qualidade, e que são delimitadas por retângulos $w_1, ..., w_m$ que não se sobrepõem, ou seja, $w_i \cap w_j = \emptyset$, para $i, j \in \{1, ..., n\}$.

5.5 Descarte de janelas

Após determinar o vetor disparidade, o algoritmo avalia o erro de correlacionamento, definido na equação (5.2). Se esse valor for superior a um certo limiar, ele para de acompanhar a janela a partir desse quadro, pois a disparidade obtida relaciona janelas que são muito diferentes. Essa janela pode ser substituída por uma nova, que deve ser escolhida como sendo a mais bem acompanhável no quadro, e que não se sobreponha às outras janelas que ainda estão sendo acompanhadas.

Figura 5.1: Exemplos de pontos que não são projeções de pontos fixos da cena. No caso do ponto 1, o KLT está acompanhando uma região de brilho de uma superfície. O problema é que essa região se move com a movimentação da câmera. No caso do ponto 2, o KLT está acompanhado a superposição da projeção dos bordos de duas superfícies distintas da cena.

5.6 Problemas no uso do KLT

O nosso interesse pelo algoritmo KLT é utilizá-lo para determinar as projeções de um conjunto de pontos de uma cena tridimensional em um vídeo. Infelizmente, não existem garantias de que as projeções encontradas por ele satisfazem essa propriedade.

Um dos problemas é que a estratégia de descarte do algoritmo KLT evita apenas erros grosseiros cometidos em quadros consecutivos. Ela não impede o acúmulo de pequenos erros ao longo de um acompanhamento sobre uma seqüência de quadros. Isso significa que os resultados podem ser imprecisos, principalmente no caso de acompanhamentos feitos sobre seqüências longas.

Outro problema pode ser compreendido analisando-se a Figura 5.1, que exibe três quadros de um vídeo no qual se aplicou o algoritmo KLT para acompanhar a projeção de 20 pontos da cena. Vê-se claramente que os dois pontos indicados nas imagens são problemáticos, pois não correspondem às projeções de pontos fixos da cena.

Uma modificação do algoritmo KLT que procura resolver o primeiro problema pode ser encontrada em [15]. Nessa modificação, além de ser feito o acompanhamento de pontos em quadros consecutivos, faz-se a comparação da vizinhaça de cada ponto com a vizinhança de seu correspondente no quadro em que ele foi selecionado para ser acompanhado. Se as vizinhanças se tornam muito diferentes o ponto deixa de ser

acompanhado. Essa versão de KLT não foi utilizada neste trabalho, pois a hipótese de rigidez da cena nos permite resolver tanto o primeiro como o segundo problema simultaneamente. Uma maneira de fazer isso será apresentada no próximo capítulo.

Capítulo 6

Calibração de famílias de câmeras

Neste capítulo descrevemos um algoritmo robusto, capaz de determinar os parâmetros extrínsecos assumidos por uma câmera na captação dos quadros de um vídeo, dado que os parametros intrínsecos foram previamente estimados, usando o que foi visto no Capítulo 3. A cena que é apresentada no vídeo precisa ser predominantemente rígida, ou seja, a maioria dos pontos da cena não podem ter sua posição modificada, pois as restrições impostas pela rigidez sobre suas projeções é que tornam possível a determinação dos parâmetros da câmera.

6.1 Definições

Adotaremos as seguintes definições:

1. **Família de pontos homólogos**
   Dado um vídeo $(I)_n = (I_1, ..., I_n)$, dizemos que a família $(x)_n = (x_1, ..., x_n)$, onde $x_i \in \mathbb{R}P^2$, é uma família de pontos homólogos associada ao vídeo $(I)_n$ se existe um ponto $X \in \mathbb{R}P^3$, da cena, tal que a projeção de X em I_j é x_j, para todo $j \in \{1, ..., n\}$.

2. **Matriz de pontos homólogos**
   Uma matriz M, $m \times n$, formada por elementos de $\mathbb{R}P^2$, é uma matriz de pontos homólogos associada a um vídeo $(I)_n$ se cada uma de suas linhas define uma família

de pontos homólogos associada a $(I)_n$. Com essa definição temos também que a j-ésima coluna de M corresponde aos pontos homólogos do quadro I_j.

3. **Configuração**
   Uma configuração é um par $((P)_n, \Omega)$, onde $(P)_n = (P_1, \ldots, P_n)$ é uma família de câmeras e $\Omega = \{X_1, \ldots, X_m\}$, com $X_i \in \mathbb{R}P^3$, é um conjunto de pontos da cena.

4. **Explicação para famílias de pontos homólogos**
   Estabelecida uma tolerância $\varepsilon \in \mathbb{R}^+$, definimos que uma explicação projetiva para uma família de pontos homólogos $(x)_n = (x_1, ..., x_n)$ é uma configuração $((P)_n, \Omega)$ tal que $\forall i \in \{1, ..., n\}$, $\exists X_j \in \Omega$ que satisfaz $d(P_i X_j, x_i) < \varepsilon$. Nesse caso, dizemos também que a configuração $((P)_n, \Omega)$ explica projetivamente a família de pontos homólogos $(x)_n$.

5. **Explicação para matrizes de pontos homólogos**
   Uma explicação projetiva para uma matriz de pontos homólogos M é uma configuração que explica todas as famílias de pontos homólogos das linhas de M. Nesse caso, dizemos também que a configuração explica projetivamente a matriz de pontos homólogos M.

6.2 Calibrando aos pares

Não é possível estender, de maneira imediata, o processo de calibração de pares de câmeras, apresentado no Capítulo 4, para uma calibração de diversas câmeras, via calibração par a par. O motivo é a indeterminação da escala existente em cada calibração par a par, como foi apresentado na seção 4.1.

Por exemplo, se considerarmos que estamos de posse de um vídeo $(I)_n$, e aplicarmos a técnica de calibração do Capítulo 4, usando os pontos homólogos dos pares $(I_1, I_2), (I_1, I_3), ..., (I_1, I_n)$, obteremos como resposta pares de câmeras $(K[I|0], K[R_1|t_1])$, $(K[I|0], K[R_2|t_2]), ..., (K[I|0], K[R_{n-1}|t_{n-1}])$, onde as direções e os sentidos dos vetores $t_1, t_2, ..., t_{n-1}$, podem ser determinados, mas os valores de $\|t_1\|$, $\|t_2\|$, ..., $\|t_{n-1}\|$ não podem.

A possibilidade de determinar apenas as direções e os sentidos dos vetores $t_1, t_2, ...,$

t_{n-1}, é uma limitação do processo de calibração realizado par a par. A verdadeira indeterminação de escala, que é inerente ao problema de calibração de várias câmeras, é mais fraca. Embora os valores de $\|t_1\|$, $\|t_2\|$, ..., $\|t_{n-1}\|$ não possam ser determinados, as relações $\frac{\|t_i\|}{\|t_j\|}$ podem, ou seja, é possível encontrar como resposta, uma família de n câmeras da forma $(K\left[I|0\right], K\left[R_1|\lambda t_1\right], K\left[R_2|\lambda t_2\right], ..., K\left[R_{n-1}|\lambda t_{n-1}\right])$, onde $\lambda \in \mathbb{R}^+$ é um fator que não pode ser determinado.

6.3 Calibração em três passos

Apresentaremos agora um algoritmo que encontra uma explicação projetiva $((P)_n, \{X_1, ..., X_m\})$ para uma matriz de pontos homólogos M associada a um vídeo $(I)_n$. Embora não tenha sido foco de destaque, este algoritmo aparece como parte do processo de calibração descrito em [6].

O algoritmo é formado pelos seguintes passos:

1. Passo 1: Utilizar as colunas de M correspondentes aos pontos homólogos de um par de quadros I_i e I_j para determinar P_i e P_j.
2. Passo 2: Utilizar o par P_i e P_j e a matriz M para determinar o conjunto $\{X_1, ..., X_m\}$.
3. Passo 3: Utilizar o conjunto $\{X_1, ..., X_m\}$ e a matriz M para determinar a família de câmeras $(P)_n$.

Como apresentado nos capítulos anteriores, tem-se que cada um dos três passos pode ser resolvido utilizando-se a proposição 3.1.

Este processo de calibração em três passos é interessante, pois evita o uso de uma modelagem matemática sofisticada baseada em tensores trifocais. Um estudo sobre calibração feita com tensores trifocais pode ser encontrado em [8] e [4].

6.4 Problemas da calibração em três passos

Uma implementação ingênua da calibração em três passos apresenta resultados ruins devido aos seguintes problemas:

1. Problema do passo 1: Podem ocorrer erros grosseiros durante a execução do passo 1, pois a matriz fundamental é estimada utilizando-se um conjunto de pontos homólogos que pode apresentar erros grosseiros, já que estamos considerando que esses são determinados automaticamente pelo algoritmo KLT, que não oferece garantias sobre sua precisão ou correção.

2. Problema do passo 2: Podem ocorrer erros grosseiros durante a execução do passo 2 devido a problemas de condicionamento do processo de reconstrução, pois é possível que algum ponto da cena, reconstruído, seja tal que uma grande perturbação de sua posição em uma direção cause uma pequena modificação nas coordenadas das projeções obtidas pelas câmeras.

3. Problema do passo 3: O passo 3 não impõe a restrição dada pelo fato dos parâmetros intrínsecos serem conhecidos. Tais parâmetros são utilizados no passo 1 quando se obtém a matriz essencial a partir da equação (4.5).

Mostraremos como resolver esses problemas de maneira a tornar robusta a calibração feita em três passos. Para tal, faremos uso do algoritmo RANSAC.

6.4.1 Algoritmo RANSAC

O algoritmo RANSAC (Random Sample Consensus), foi proposto por Fischler e Bolles em [3], onde foi apresentado nos seguintes termos

"Dados um modelo que precisa de um mínimo de n pontos para ter seus parâmetros livres instanciados, e um conjunto de pontos P, tal que o número de pontos de P é maior do que n, isto é $\sharp(P) \geqslant n$. Selecione aleatoriamente um subconjunto S_1, de n pontos de P e instancie o modelo. Utilize o modelo instanciado M_1 para determinar um subconjunto S_1^* de pontos de P, que satisfazem um critério de tolerância de erro em relação a M_1. O conjunto S_1^* é chamado de conjunto de consenso de S_1.

Se $\sharp(S_1^*)$ for maior que um certo limiar t, que é função de uma estimativa do número de erros grosseiros em P. Use S_1^* para computar (possivelmente usando mínimos quadrados) um novo modelo M_1^*.

Se $\sharp(S_1^*)$ for menor que t, selecione aleatoriamente um novo subconjunto S_2 e repita o processo acima. Caso depois de um número pré-determinado de iterações, nenhum conjunto de consenso com t ou mais elementos tiver sido encontrado, encontre o modelo correspondente ao maior conjunto de consenso, ou termine acusando um erro."

Apresentaremos a seguir como é possível utilizar o RANSAC para resolver os problemas dos passos 1 e 2. Utilizaremos a notação definida acima para tornar simples a identificação dos princípios do paradigma RANSAC. As duas colunas de M, correspondentes aos pontos homólogos utilizados na reconstrução tridimensional feita no passo 2, serão chamadas de colunas base.

6.4.2 Solução para o problema do passo 1

Podemos, nesse caso, considerar que o algoritmo de oito pontos fornece uma maneira de se obter uma matriz fundamental, que corresponde ao modelo M_1, a partir de um conjunto formado por oito pares de pontos homólogos correspondentes a S_1, obtidos nas colunas base de M.

Pode-se utilizar um critério de tolerância para definir o conjunto de consenso S_1^* baseado na função objetivo do algoritmo de oito pontos, mais precisamente, dado um limiar $\eta_1 \in \mathbb{R}^+$ estabelecido empiricamente, incluimos em S_1^* os pares de pontos homólogos (x_i, x_j) das colunas base de M, se $|x_i'^T F x_j| < \eta_1$, onde F é a matriz fundamental estimada usando o conjunto S_1. O modelo M_1^* é uma matriz fundamental que pode ser obtida aplicando-se o próprio algoritmo de oito pontos sobre os pontos homólogos de S_1^*.

6.4.3 Solução para o problema do passo 2

Seja Q o conjunto formado pelas reconstruções tridimensionais dos pares de pontos homólogos das colunas base de M que fazem parte do conjunto de consenso encontrado durante a aplicação do RANSAC na estimação da matriz fundamental.

Para resolvermos o problema de condicionamento do passo 2 vamos utilizar o RANSAC durante a execução do passo 3. Para isso temos que o conjunto Γ, formado por seis pares (X, m), faz o papel do modelo S_1, onde X é um elemento de Q, e m é a linha de M correspondente à família de pontos homólogos associada a X. O modelo M_1

corresponde a uma família de câmeras $(P)_n$ obtida pela aplicação do passo 3 utilizando-se apenas os elementos de Γ. O critério de tolerância usado para definir S_1^* é baseado na medida do erro de reprojeção. Mais precisamente, dado um limiar $\eta_2 \in \mathbb{R}^+$ escolhido empiricamente, inserimos em S_1^* os pares $(X', m\prime)$, com $X' \in Q$, que satisfazem, $\forall j \in \{1, ..., n\}$, $d\left(P_j X', m\prime_j\right) < \eta_2$. O modelo M_1^* corresponde a uma família de câmeras $(P^*)_n$, estimada a partir do conjunto S_1^*.

Dessa forma, temos que o conjunto formado pelos pontos X' inseridos em S_1^*, e a família de câmeras $(P^*)_m$, definem uma explicação projetiva, de tolerância η_2, para uma matriz de pontos homólogos M', formada por linhas de M.

6.4.4 Solução para o problema do passo 3

Considerando que a matriz de pontos homólogos M possui n colunas, temos que existem $\left(n^2 - n\right)/2$ possíveis escolhas para o par de colunas base. Sendo assim, pode-se tentar resolver o problema do passo 3, descartando-se a solução, caso os parâmetros intrínsecos de alguma das câmeras encontradas seja muito diferente dos parâmetros que estamos assumindo como conhecidos. Os três passos são repetidos considerando escolhas diferentes de colunas bases até que uma solução satisfatória seja encontrada. Mais precisamente, dado um limiar $\eta_3 \in \mathbb{R}^+$ escolhido empiricamente, recusamos a família $(P^*)_n$ caso $\|K_j - K\| > \eta_3$[1], para algum $j \in \{1, \ldots, n\}$, onde K_j é matriz dos parâmetros intrínsecos obtida pela fatoração de P_j na forma $K_j\,[R_j|t_j]$, e K é a matriz dos parâmetros intrínsecos que estamos assumindo como conhecida.

6.5 Escolha das colunas base

Como temos a possibilidade de escolher $\left(n^2 - n\right)/2$ pares de colunas bases para usarmos nos passos 1 e 2, faz sentido escolhermos aquele que forneça o melhor resultado.

[1]Foram realizados experimentos bem sucedidos utilizando tanto a norma de Frobenius, como a norma definida por $\|A\| = \max|A_{nm}|$. Uma estratégia de descarte melhor, porém computacionalmente mais cara, seria avaliar o erro de reprojeção introduzido ao se substituir K_j por K, e depois comparar esse valor com um limiar.

Podemos então definir que, o melhor resultado é a configuração que não foi descartada por problemas de parâmetros intrínsecos no passo 3 e que explica o maior número de linhas da matriz de pontos homólogos M. Uma maneira bastante eficiente para determinar esse par foi obtida utilizando-se a seguinte estratégia:

1. Não se deve tentar utilizar colunas bases cuja distância média dos pontos homólogos não supere um certo limiar.

2. Se o número de pares de pontos homólogos obtido pelo RANSAC aplicado ao passo 1 for menor que o número de linhas de M explicadas por uma configuração C, já calculada utilizando-se uma outra escolha de colunas base, deve-se abortar a execução, pois é impossível que a configuração C seja melhorada. Com isso evitamos a realização do RANSAC no passo 2, que é o de maior custo computacional.

3. Devemos utilizar primeiro colunas afastadas de M como colunas base, pois normalmente essas fornecem um resultado melhor que as colunas próximas. Isso faz com que os bons resultados sejam determinados antes dos ruins, e com isso aumentamos o efeito do item anterior.

6.6 Calibração via Levenberg-Marquardt

Seja $((P)_n, \{X_1, \ldots, X_m\})$ uma explicação projetiva para uma matriz de pontos homólogos M. Podemos definir o erro de reprojeção associado a essa explicação como

$$\sum_{k=1}^{n}\sum_{i=1}^{m} d\left(P_k X_i, M_{ik}\right)^2 .$$

Temos que, quanto menor o erro de reprojeção, melhor é a explicação. Com isso, faz sentido definirmos o problema de encontrar uma explicação projetiva ótima para uma matriz de pontos homólogos M. Esse problema pode ser atacado utilizando-se o algoritmo Levenberg-Marquardt. Nesse caso, a função objetivo é dada por

$$g(x) = \frac{1}{2}\|f(x) - x_0\|^2, \tag{6.1}$$

onde $x_0 \in \mathbb{R}^{2mn}$ é um vetor cujas componentes são as coordenadas das projeções dos n pontos, nas m imagens obtidas pelas câmeras, e a função $f : E^n \times \mathbb{R}^{3m} \to \mathbb{R}^{2mn}$ é

definida por

$$(P_1, \cdots, P_n, X_1, \cdots, X_m) \mapsto (P_1X_1, \cdots, P_1X_m, \cdots, P_nX_1, \cdots, P_nX_m)\,,$$

onde $E \subset \mathbb{R}^{12}$ é um espaço de representação de câmeras virtuais.

O conjunto $E^n \times \mathbb{R}^{3m}$ é formado por representações de configurações de n câmeras e m pontos.

6.7 Representação de uma configuração

Pode-se representar uma configuração de m pontos e n câmeras por um vetor de $\mathbb{R}^{12n+3m}$, onde $12n$ coordenadas correspondem aos elementos de n matrizes 3×4 associadas às n câmeras, e $3m$ coordenadas correspondem às coordenadas de m pontos da cena tridimensional. O problema dessa representação, no nosso contexto, é que ela não impõe a restrição caracterizada pelo fato dos parâmetros intrínsecos das câmeras serem conhecidos. Uma maneira de impor essa restrição é utilizar um vetor de $\mathbb{R}^{6n+3m}$ como representação para uma configuração. Nessa representação, as câmeras possuem apenas seis graus de liberdade, que correspondem aos parâmetros extrínsecos. Desses seis graus de liberdade, três especificam a rotação, que define a orientação do referencial da câmera, e três especificam o posicionamento do centro de projeção. Essa forma de parametrização é semelhante àquela feita na seção 3.4.5, com a diferença que os parâmetros intrínsecos são fixados.

6.8 Ciclos de refinamento

Um dos problemas existentes na calibração em três passos é a possibilidade de alguma família de pontos homólogos ser descartada por apresentar um erro de reprojeção muito elevado em algum quadro, devido ao fato da reconstrução tridimensional realizada pelo passo 2 só levar em consideração um único par de quadros do vídeo. A solução adotada para esse problema foi combinar a calibração em três passos com uma calibração feita com Levenberg-Marquardt.

Inicialmente é determinada uma explicação projetiva $((P)_n, \Omega_1)$ obtida pela execução da calibração em três passos utilizando-se um limiar η_2, definido na seção 6.4.3, relativamente alto, escolhido de maneira que uma grande quantidade de famílias de pontos homólogos seja aceita mesmo que alguns pontos com erros grosseiros possam contaminar a solução. Essa solução é então refinada por um algoritmo formado por ciclos de quatro passos que são apresentados a seguir, com o objetivo de selecionar de maneira mais criteriosa as famílias de pontos homólogos que devem ser consideradas na estimação da explicação projetiva.

1. Executam-se algumas iterações do algoritmo Levenbeg-Marquardt, utilizando como estimativa inicial a explicação projetiva $((P)_n, \Omega_1)$, determinando-se uma outra explicação projetiva $((P')_n, \Omega_2)$ de menor erro de reprojeção associado.

2. Utilizam-se pares de câmeras de $(P')_n$ para determinar uma nova reconstrução Ω_3 para todos os pontos homólogos de M. Esse processo pode ser realizado escolhendo-se pares de câmeras diferentes para reconstruir cada ponto de Ω_3, de forma que, cada par utilizado seja aquele que minimiza o erro de reprojeção associado a cada ponto.

3. Descartam-se os pontos de Ω_3 cujo erro de reprojeção em relação às câmeras de $(P')_n$ são maiores que um limiar η_2', escolhido de forma mais rigorosa que que η_2, ou seja, $\eta_2' < \eta_2$. Obtém-se assim um novo conjunto de pontos Ω_4.

4. Estima-se uma nova família de câmeras $(P'')_n$ a partir do conjunto de pontos Ω_4 e das respectivas linhas da matriz de pontos homólogos M. Com isso, obtemos uma explicação projetiva $((P'')_n, \Omega_4)$ que pode ser utilizada para alimentar um novo ciclo de refinamento.

A cada ciclo pode-se utilizar um limiar de tolerância para o erro de reprojeção cada vez menor, tendo em vista, que como a solução fica cada vez mais correta, podemos ser cada vez mais rigorosos.

Após executarmos um determinado número de ciclos de refinamentos podemos aplicar o algoritmo Levenberg-Marquardt até sua convergência, obtendo uma explicação

projetiva, cujo erro de reprojeção associado às famílias de pontos homólogos selecionadas é um mínimo local.

6.9 Decomposição do vídeo em fragmentos

Em um vídeo $(I)_n$ é possível que existam quadros I_a e I_b que não admitam nenhum par de pontos homólogos, no caso de nenhum ponto da cena ser projetado em ambas as imagens. Além disso, o algoritmos KLT pode não conseguir acompanhar com precisão pontos em longas seqüências de imagens. Como conseqüência, tem-se que não é possível, em geral, definir uma matriz de pontos homólogos para um vídeo completo.

Valendo-se do fato do movimento da câmera ser contínuo, pode-se realizar uma decomposição do vídeo $(I)_n$ em fragmentos, de forma que todos os fragmentos admitam uma matriz de pontos homólogos. Sendo mais preciso, estamos definindo como um fragmento, de $k+1$ quadros, de um vídeo $(I_1, ..., I_n)$, como sendo um vídeo da forma $(I_j, ..., I_{j+k})$, onde $\{j, j+1, ..., j+k\} \subset \{1, 2, ..., n\}$.

Nos experimentos realizados, os fragmentos foram determinados por uma heurística. A solução adotada foi que cada fragmento seria obtido comparando-se um quadro I_j com seus sucessores até que fosse encontrado um quadro I_{j+k}, em que os pontos homólogos de I_j e I_{j+k}, apresentassem uma distância média acima de um limiar $\varepsilon \in \mathbb{R}^+$, escolhido experimentalmente, obtendo-se assim um fragmento de $k+1$ quadros $(I_j, I_{j+1}, ..., I_{j+k})$.

Para que os fragmentos possam ser unidos posteriormente, tem-se que a decomposição é feita de forma que exista a superposição de um quadro entre cada par de fragmentos adjacentes. Ou seja, o vídeo $(I)_k$ é decomposto em fragmentos da forma $(I_1, ..., I_{k_1})$, $(I_{k_1}, ..., I_{k_2})$, ..., $\left(I_{k_{n-2}}, ..., I_{k_{n-1}}\right)$, $\left(I_{k_{n-1}}, ..., I_{k_n}\right)$, onde cada fragmento é obtido como explicado acima.

É possível que, ao tentar determinar o último fragmento, não seja possível satisfazer a restrição do limiar ε, devido ao encontro do final do vídeo. Nesse caso, descartam-se esse últimos quadros, para evitar problemas de calibração causados pela pequena modificação das coordenadas dos pontos das famílias de pontos homólogos associadas ao fragmento.

6.10 Junção de fragmentos

Consideremos que foram determinadas explicações projetivas para as matrizes de pontos homólogos dos fragmentos de um vídeo $(I)_n$. Mostraremos agora como utilizar essas explicações para determinar uma família de câmeras $(P)_n$ correspondente às câmeras que foram utilizadas para captar $(I)_n$. É preciso levar em consideração que cada explicação projetiva foi definida em um referencial próprio, e em uma escala própria. Sendo assim, vamos dividir o problema em dois:

1. Alinhamento de fragmentos.

2. Compatibilização de escalas.

6.10.1 Alinhamento de fragmentos

Dadas duas configurações $E_1 = ((G)_r, \Omega)$ e $E_2 = ((Q)_s, \Psi)$, que explicam projetivamente as matrizes de pontos homólogos M_1 e M_2, associadas respectivamente aos fragmentos consecutivos $F_1 = (I_k, I_{k+1}, ..., I_{k+r})$, e $F_2 = (I_{k+r}, I_{k+r+1}, ..., I_{k+r+s})$ de um vídeo $(I)_n$, queremos determinar um movimento rígido que transforma $(Q)_s$ em uma família de câmeras $(Q')_s$ tal que $G_r = Q'_1$. Diremos nesse caso que $(G)_r$ e $(Q')_s$ estão alinhadas.

Sejam $Q_1 = K[R_1|t_1]$ e $G_r = K[R_2|t_2]$. Podemos determinar a família $(Q')_s$ aplicando a seguinte transformação aos elementos de $(Q)_s$

$$K[R|t] \mapsto K\left[\left(RR_1^T R_2\right)|RR_1^T(t_2 - t_1) + t\right].$$

Podemos usar repetidas vezes essa transformação para alinharmos todas as famílias de câmeras associadas a cada um dos fragmentos de $(I)_n$.

6.10.2 Compatibilização de escalas

O fato de duas famílias de câmeras $(G)_r$ e $(Q)_s$, associadas a fragmentos consecutivos, estarem alinhadas, não significa que elas estejam prontas para serem concatenadas de forma a gerar a família de câmeras utilizada na captação dos dois fragmentos. Isso ocorre pois, geralmente $(G)_r$ e $(Q)_s$ estão calibradas em escalas diferentes.

Podemos resolver o problema de compatibilização de escalas explorando o fato que dadas duas explicações projetivas $E_1 = ((G)_r, \Omega)$ e $E_2 = ((Q)_s, \Psi)$ associadas a fragmentos consecutivos, normalmente existe um conjunto não vazio $S \subset \Omega$ cujos elementos são pontos da cena que também aparecem em Ψ. O fator de escala λ pode ser obtido como resposta do seguinte problema de otimização

Problema 6.1. *Determinar $\lambda \in \mathbb{R}^+$ tal que aplicando-se a transformação $K[R|t] \mapsto K[R|\lambda t]$ sobre todas as câmeras em $(Q)_s$, obtém-se uma nova família de câmeras que ao ser alinhada com a família $(G)_r$ define uma família de câmeras $(Q')_s$ que faz com que o erro de reprojeção associado à explicação projetiva $((Q')_s, S)$ seja mínimo.*

6.10.3 Compatibilização robusta de escalas

Resolver o problema 6.1 não é simples, pois como as coordenadas dos elementos de S são estimadas através de um processo de minimização do erro de reprojeção associado a $((G)_r, \Omega)$, é possível que algum dos pontos de S apresente erros grosseiros de reprojeção quando feitas por câmeras de $(Q')_s$. Isso pode ocorrer caso grandes modificações das coordenadas de pontos de S, em alguma direção, não produzam alterações significativas sobre as projeções obtidas pelas câmeras de $(G)_r$.

Com o objetivo de resolver o problema 6.1 de forma robusta, aplicamos idéias presentes no algoritmo RANSAC, obtendo uma solução em dois passos:

1. Passo 1: Encontra-se o conjunto $\Lambda \subset \mathbb{R}^+$ formado pelos valores de λ que, ao serem utilizados na compatibilização de escalas maximizam o número de pontos de S cujo erro de reprojeção por câmeras de $(Q')_s$ são inferiores a um limiar $\xi \in \mathbb{R}^+$. Esses pontos de S definem o conjunto Θ;

2. Passo 2: Resolve-se o problema 6.1 modificado pela substituição do conjunto S pelo seu subconjunto Θ.

Capítulo 7

Experimentos computacionais

Esse capítulo descreve experimentos realizados com um sistema desenvolvido a fim de testar o algoritmo de calibração de famílias de câmeras apresentado no capítulo anterior. O sistema é capaz de inserir objetos virtuais sobre um vídeo digital de forma geometricamente consistente, ou seja, é um sistema capaz de fazer realidade aumentada sobre um vídeo. Para fazer isso, os parâmetros estimados na calibração são utilizados na especificação de uma câmera do OpenGL equivalente, como apresentado na seção 2.6, que é empregada na criação do objeto virtual.

7.1 Bibliotecas utilizadas

Apresentamos aqui a lista de bibliotecas e programas que foram empregados no desenvolvimento do sistema, junto com uma descrição resumida das respectivas funcionalidades utilizadas.

1. **GNU Scientific Library**
   Foi a principal biblioteca utilizada, foram explorados seus recursos de álgebra linear numérica, sua implementação do algoritmo Levenberg-Marquardt, e seu algoritmo de otimização de funções de uma variável real. Serviu de base para a implementação de todos os algoritmos de calibração de câmeras apresentados nos Capítulos 3, 4 e 6.

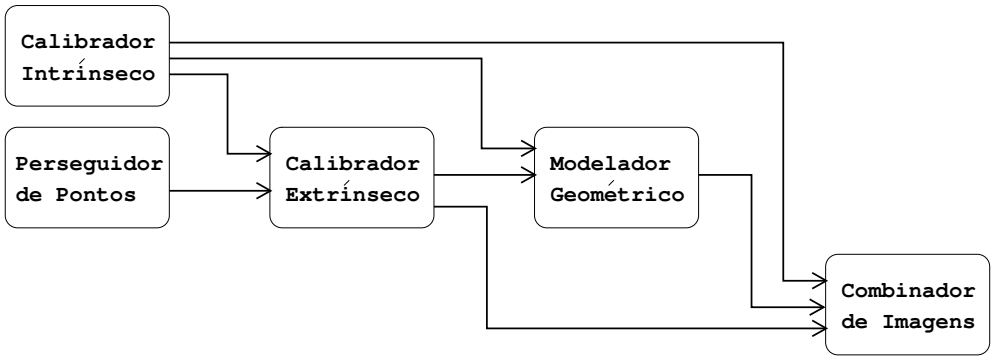

Figura 7.1: Arquitetura do sistema

2. **KLT**

   Essa biblioteca forneceu a implementação do algoritmo Kanade-Lucas-Tomasi, que foi utilizado para obter famílias de pontos homólogos sobre os quadros de um vídeo.

3. **MPEG Library**

   Essa biblioteca foi utilizada na decodificação de vídeos codificados no formato MPEG.

4. **MPEG2 Encoder**

   Esse programa foi utilizado na codificação do vídeo de saída no formato MPEG.

5. **OpenGL**

   Essa biblioteca foi utilizada na implementação dos processos de síntese e composição de imagens.

6. **S3D**

   Foram utilizadas estruturas definidas nessa biblioteca na representação de imagens e objetos poliedrais.

7.2 Arquitetura do sistema

O sistema é composto por um conjunto de módulos combinados em uma arquitetura de filtros e canais como ilustrada na Figura 7.1.

O processamento realizado por cada módulo é o seguinte

1. **Calibrador Intrínseco**
   Recebe como entrada um conjunto de correspondências de pontos 3D-2D e fornece como saída uma matriz de parâmetros intrínsecos obtida pela aplicação do algoritmo apresentado no Capítulo 3.

2. **Perseguidor de Pontos**
   Recebe como entrada um vídeo digital e fornece como saída um conjunto de famílias de pontos homólogos estimados pelo algoritmo KLT, como explicado no Capítulo 5.

3. **Calibrador Extrínseco**
   Recebe como entrada uma matriz de parâmetros intrínsecos e um conjunto de pontos homólogos associados aos quadros de um vídeo, e fornece como saída os parametros extrínsecos associados a cada quadro, como explicado no Capítulo 6.

4. **Modelador Geométrico**
   Recebe como entrada um vídeo digital, os parâmetros intrínsecos da câmera que o captou, os parametros extrínsecos associados a cada quadro do vídeo, e um objeto poliedral P. Esse módulo apresenta uma interface gráfica que permite que um usuário modifique a posição e as dimensões de P observando interativamente o efeito correspondente sobre um conjunto de quadros do vídeo. A saída do módulo é o objeto P modificado.

5. **Combinador de Imagens**
   Recebe como entrada um vídeo digital, os parâmetros intrínsecos da câmera que o captou, os parametros extrínsecos associados a cada quadro do vídeo, e um objeto poliedral. A saída é o vídeo formado pela composição dos quadros do vídeo de entrada com o objeto virtual.

7.3 Estimação de parâmetros intrínsecos

Foram estimados os parâmetros intrínsecos de uma câmera fotográfica modelo SONY MVC-FD85, capaz de capturar vídeos de 15 segundos com resolução espacial

Figura 7.2: Imagens do objeto calibrador obtidas por uma mesma câmera posicionada de formas diferentes.

320×240. Tendo em vista que tal câmera não fornece essa baixa resolução para a captura de fotografias, utilizou-se uma resolução 640×480 na captura das imagens do objeto calibrador, e posteriormente fez-se os ajustes necessários aos resultados.

As Tabelas 1, 2 e 3 exibem os parâmetros intrínsecos e extrínsecos estimados pelo módulo Calibrador Intrínseco, utilizando as correspondências 3D-2D obtidas com as imagens (a) e (b) da Figura 7.2. O sistema de coordenadas 3D adotado é o indicado nas imagens, assumindo-se que os lados das quadrículas são unitários.

As coordenadas das projeções dos vértices de cada quadrícula não são determinadas de forma automatizada, ou seja, o usuário é responsável por fornecer as correspondências 3D-2D ao sistema.

A Tabela 1 mostra os resultados obtidos aplicando-se diretamente a proposição 3.1, como descrito na seção 3.1.3. A Tabela 2 mostra os resultados obtidos por uma pequena modificação desse mesmo algoritmo com a adição de um processo de normalização de coordenadas semelhante ao feito com os pontos homólogos na seção 4.6. A descrição dessa modificação pode ser encontrada em [8] e seu objetivo é a melhoria do condicionamento do algoritmo. Essa foi a versão utilizada na implementação da calibração em três passos.

A Tabela 3 mostra os resultados obtidos aplicando-se o algoritmo Levenberg-

Marquardt impondo a restrição de não cisalhamento da matriz de sensores da câmera, como explicado na seção 3.4. Neste caso, os erros de reprojeção encontrados para os pontos marcados em (a) e (b) foram 1,1 e 1,0 pixels respectivamente, que na resolução 320×240 corresponde a um erro de aproximadamente 0,5 pixel.

Tabela 1 **Calibração sem restrição**

Imagem	K	$[R\|t]$
(a)	$\begin{pmatrix} -799.316 & 1.406 & 322.985 \\ 0.000 & 796.551 & 223.889 \\ 0.000 & 0.000 & 1.000 \end{pmatrix}$	$\begin{pmatrix} 0.658 & -0.752 & -0.024 & 0.583 \\ -0.050 & -0.076 & 0.995 & -4.532 \\ -0.751 & -0.654 & -0.088 & 25.351 \end{pmatrix}$
(b)	$\begin{pmatrix} -790.628 & -1.348 & 325.534 \\ 0.000 & 792.078 & 235.334 \\ 0.000 & 0.000 & 1.000 \end{pmatrix}$	$\begin{pmatrix} -0.073 & -0.071 & 0.994 & -4.529 \\ -0.677 & 0.735 & 0.002 & 0.204 \\ -0.732 & -0.673 & -0.102 & 28.535 \end{pmatrix}$

Tabela 2 **Calibração sem restrição (normalizada)**

Imagem	K	$[R\|t]$
(a)	$\begin{pmatrix} -801.825 & 0.303 & 323.708 \\ 0.000 & 798.297 & 227.076 \\ 0.000 & 0.000 & 1.000 \end{pmatrix}$	$\begin{pmatrix} 0.657 & -0.752 & -0.024 & 0.611 \\ -0.047 & -0.074 & 0.996 & -4.637 \\ -0.751 & -0.654 & -0.085 & 25.419 \end{pmatrix}$
(b)	$\begin{pmatrix} -787.428 & -0.430 & 321.565 \\ 0.000 & 788.703 & 232.649 \\ 0.000 & 0.000 & 1.000 \end{pmatrix}$	$\begin{pmatrix} -0.072 & -0.069 & 0.994 & -4.670 \\ -0.678 & 0.734 & 0.001 & 0.302 \\ -0.731 & -0.674 & -0.100 & 28.385 \end{pmatrix}$

Tabela 3 **Calibração restrita feita com Levenberg-Marquardt**

Imagem	K	$[R\|t]$
(a)	$\begin{pmatrix} -801.744 & 0.000 & 323.703 \\ 0.000 & 798.296 & 227.075 \\ 0.000 & 0.000 & 1.000 \end{pmatrix}$	$\begin{pmatrix} 0.657 & -0.753 & -0.024 & 0.613 \\ -0.047 & -0.074 & 0.996 & -4.637 \\ -0.752 & -0.652 & -0.085 & 25.418 \end{pmatrix}$
(b)	$\begin{pmatrix} -787.594 & 0.000 & 321.570 \\ 0.000 & 788.709 & 232.663 \\ 0.000 & 0.000 & 1.000 \end{pmatrix}$	$\begin{pmatrix} -0.072 & -0.069 & 0.994 & -4.670 \\ -0.677 & 0.735 & 0.001 & 0.301 \\ -0.731 & -0.674 & -0.100 & 28.388 \end{pmatrix}$

Os resultados ilustram a influência do posicionamento da câmera em (a) e (b) sobre a estimação dos parâmetros extrínsecos e intrínsecos. Enquanto os parâmetros extrínsecos são modificados drasticamente, os parâmetros intrínsecos sofrem uma modificação pequena.

Em todas as tabelas, os parâmetros intrínsecos f_1, f_2, x_0 e y_0 associados às imagens (a) e (b) sofreram modificações inferiores a 2%. Já o parâmetro s se comportou como uma pequena variação no ângulo de cisalhamento da matriz de sensores. No caso da tabela 1, em que não se aplicou a normalização de coordenadas, a variação foi de aproximadamente 0,2 graus. Já no caso da tabela 2, em que as coordenadas foram normalizadas, o ângulo variou aproximadamente 0,05 graus.

As modificações existentes nos parâmetros intrínsecos são justificadas pela não adequação exata do modelo de câmera de furo ao caso de câmeras com lente, e pelas imprecisões inseridas na confecção do objeto calibrador e na avaliação das coordenadas dos pontos projetados.

7.4 Calibração de fragmentos

A Figura 7.3 exibe quadros de vídeos sobrepostos por pontos acompanhados pelo módulo Perseguidor de Pontos. Como explicado no Capítulo 6, nem todos os pontos acompanhados são aproveitados em todas as etapas da calibração de um fragmento. Eles são submetidos a testes que podem descarta-los ou readimiti-los. De forma resumida,

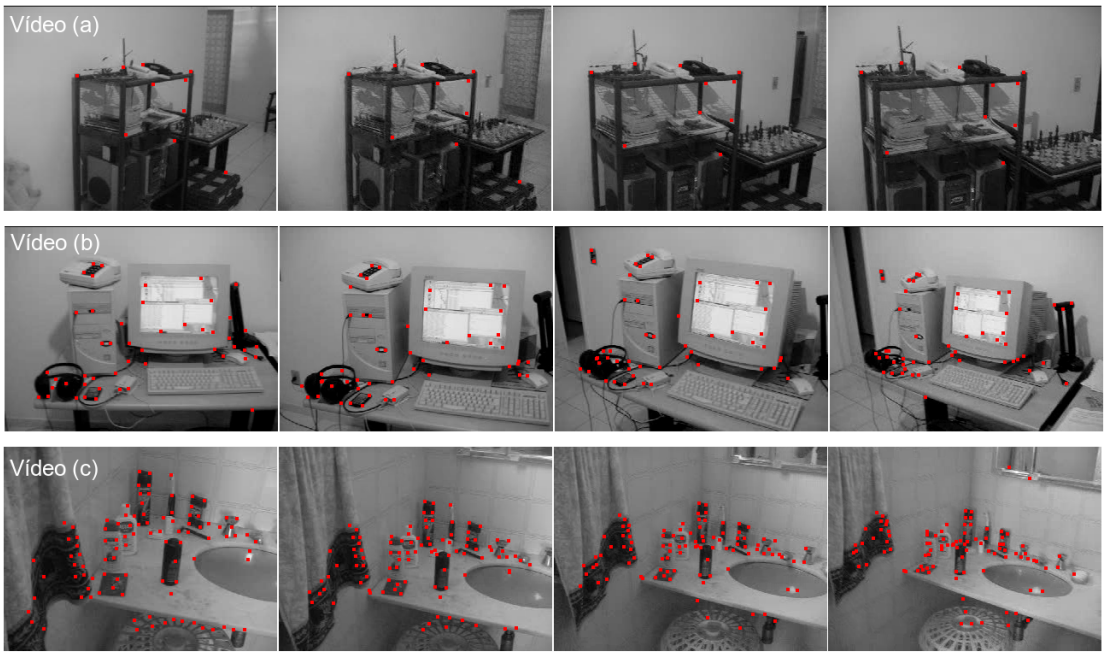

Figura 7.3: Quadros de vídeos ilustrando o acompanhamento realizado pelo módulo Perseguidor de Pontos. Temos respectivamente em (a), (b) e (c) um acompanhamento de 10, 50 e 100 pontos.

essa variação na quantidadede de pontos pode ocorrer nos seguintes momentos:

1. Durante a execução do KLT, quando pontos podem ser eliminados, caso não sejam acompanhados com sucesso, devido a uma grande modificação de suas vizinhanças em quadros consecutivos;

2. Durante a execução do algoritmo de calibração em três passos, quando pontos podem ser eliminados, por não fazerem parte do conjunto de consenso definido pelo algoritmo RANSAC;

3. Durante os ciclos de refinamento, quando pontos podem ser descartados ou readmitidos, de acordo com seus erros de reprojeção nos quadros do fragmento.

A Figura 7.4 apresenta dois gráficos que indicam a quantidade de pontos utilizada nas diversas etapas da calibração de dois fragmentos, extraídos dos vídeos (a) e (c), da

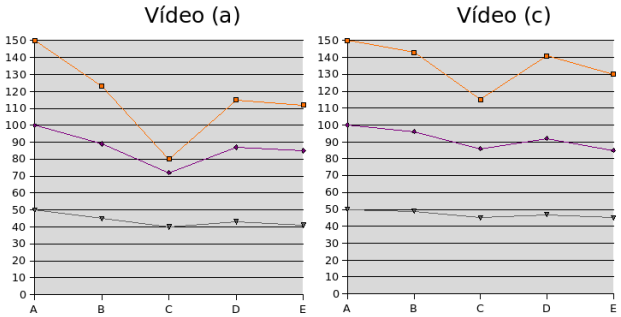

Figura 7.4: Quantidade de pontos selecionados nas diversas etapas da calibração de fragmentos dos vídeos (a) e (c) da Figura 7.3. Cada curva representa um experimento feito com uma quantidade diferente de pontos iniciais. No eixo horizontal temos: A - Pontos selecionados no início do fragmento; B - Pontos acompanhados pelo KLT por todo o fragmento; C - Pontos pertencentes ao conjunto de consenso do RANSAC utilizado pelo algoritmo de calibração em três passos; D - Pontos reconstruídos pelo primeiro ciclo de refinamento; E - Pontos reconstruídos pelo segundo ciclo de refinamento.

Figura 7.3. Cada gráfico exibe três curvas, que correspondem aos resultados associados a seleções de 50, 100 e 150 pontos, no primeiro quadro do fragmento.

Os fragmentos foram obtidos como descrito na seção 6.9, escolhendo-se um deslocamento médio de 10 pixels por ponto. Com essa escolha, foram obtidos fragmentos de aproximadamente 2 segundos em todos os casos apresentados nos gráficos.

O limiar de aceitação para o erro de reprojeção estabelecido para o RANSAC, durante a execução do algoritmo de calibração em três passos, foi de 5 pixels. Após o término deste algoritmo foram executados dois ciclos de refinamento, o primeiro aceitando um erro de reprojeção de 3 pixels, e um segundo aceitando um erro de 2 pixels.

Esses gráficos evidenciam o efeito dos ciclos de refinamento, que permitiram um melhor aproveitamento dos pontos acompanhados pelo KLT. Basta observar que, normalmente, no final de ambos os ciclos de refinamento, a quantidade de pontos satisfazendo um erro de reprojeção de 2 pixels foi maior do que a dos pontos que satisfizeram o limiar de 5 pixels aplicado pelo RANSAC na calibração em três passos.

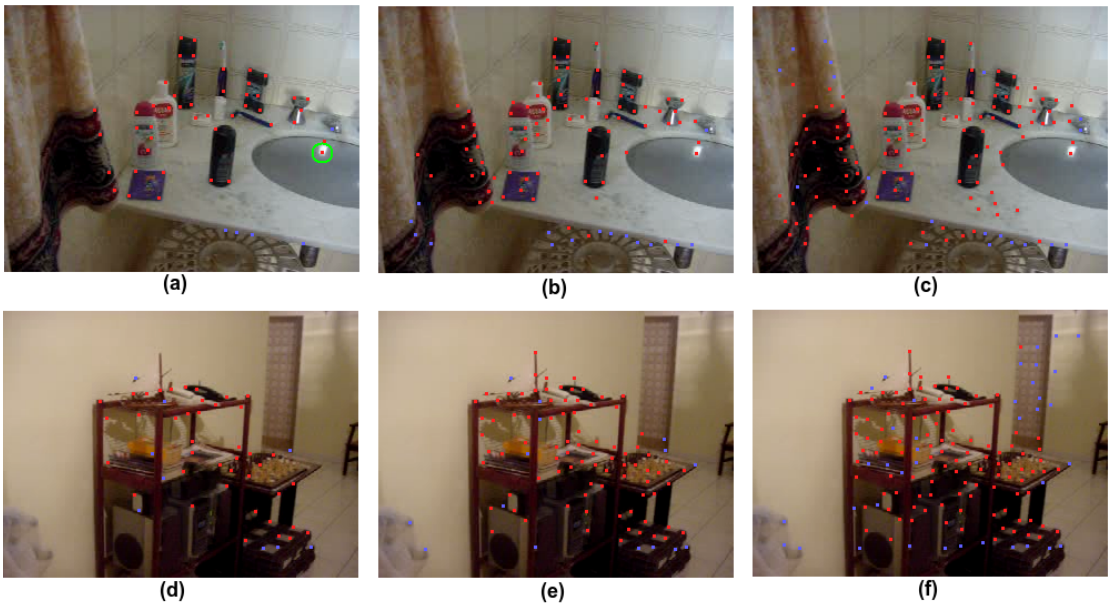

Figura 7.5: Essas imagens localizam espacialmente os pontos associados às letras A e E dos gráficos da Figura 7.4. Os pontos vermelhos são aqueles que foram aceitos no último ciclo de refinamento, e os azuis são aqueles que foram descartados. (a), (b) e (c) exibem os resultados utilizando-se respectivamente uma seleção inicial de 50, 100 e 150 pontos. (d), (e) e (f) fazem o mesmo para o outro vídeo. Vê-se que, o ponto destacado em (a), embora seja móvel, não foi descartado.

Fica evidente também que, quanto maior é o número de pontos escolhidos pelo KLT no primeiro quadro, maior é a importância do uso de ciclos de refinamento. Isso pode ser explicado pelo fato do KLT escolher os pontos seguindo uma ordem de expectativa de precisão do processo de acompanhamento. Conseqüentemente, conjuntos com muitos pontos selecionados pelo KLT devem ter muitos pontos acompanhados de forma pouco precisa. Essa imprecisão prejudica a reconstrução tridimensional feita durante a calibração em três passos, aumentando o descarte indevido de pontos, explicado na seção 6.8.

Os resultados da calibração de fragmentos foram bons. Normalmente a grande maioria dos pontos consegue satisfazer o limiar de 2 pixels após os ciclos de refinamento, como ilustrado nas figuras 7.4 e 7.5. Além disso, quando se aplica posteriormente o

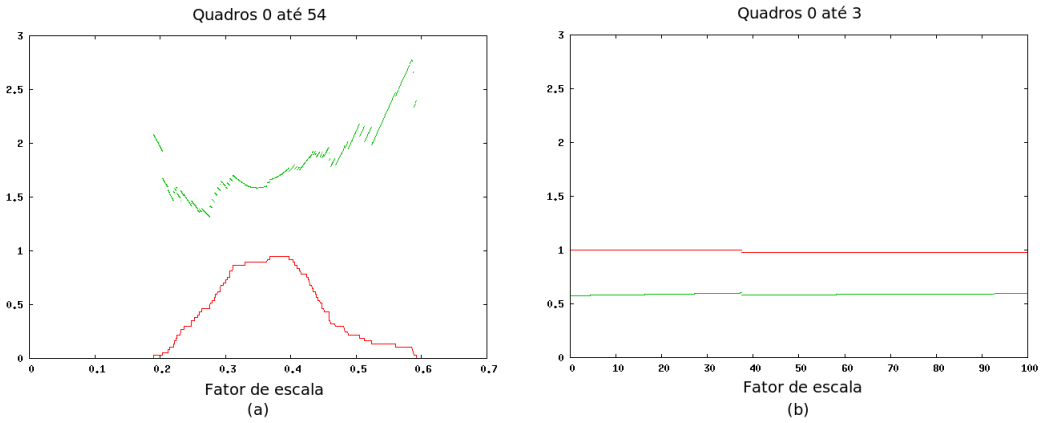

Figura 7.6: A curva vermelha indica a fração do número de pontos reconstruídos no fragmento indicado, cujos erros de reprojeção nos quadros do fragmento consecutivo são inferiores à 5 pixels. A curva verde indica o erro médio cometido nessa reprojeção. As informações são parametrizadas pelas escolhas de escalas na solução do problema 6.1. O resultado obtido aplicando-se o algoritmo definido em 6.10.3 sobre (a) é de 0,368. O resultado da letra (b) está mal determinado.

algoritmo Levenberg-Marquardt até sua convergência, obtém-se erros de reprojeção inferiores a um pixel por ponto. Por outro lado, a Figura 7.5 mostra que, o processo de descarte pode não eliminar todos os pontos móveis da cena. Um exemplo disso é o ponto selecionado sobre a reflexão especular ocorrida na pia; mesmo sendo móvel, ele admite uma reconstrução tridimensional com erro de reprojeção inferior a 2 pixels sobre todos os quadros do fragmento. Problemas desse tipo ocorrem com freqüência em fragmentos muito pequenos. Um caso extremo é apresentado na próxima seção.

7.5 Junção de fragmentos

Com o objetivo de simplificar notação, no que se segue, chamaremos de $[a, b]$ o fragmento cujos quadros vão de um índice a até um índice b.

A Figura 7.6 exibe dois gráficos que apresentam informações sobre fragmentos do vídeo (c), da Figura 7.3. Eles ilustram o processo de resolução definido em 6.10.3

para o problema de compatibilização de escalas entre fragmentos.

Na letra (a), temos que a curva vermelha indica a fração do número de pontos reconstruídos durante a calibração de $[0, 54]$ e acompanhados pelo KLT em $[54, 95]$, cujos erros de reprojeção nos quadros de $[54, 95]$ são inferiores a 5 pixels. A curva verde indica o erro médio de reprojeção, medido em escala de pixels, apresentado pelos pontos indicados pela linha vermelha. Temos uma interpretação análoga na letra (b), sendo que os fragmentos considerados são $[0, 3]$ e $[3, 6]$. Os valores exibidos nos gráficos são parametrizados pelas escolhas de escalas utilizadas na solução do problema 6.1.

Analisando esses gráficos fica evidente que o algoritmo de compatibilização robusta de escalas, definido na seção 6.10.3, funciona de forma apropriada em (a), mas funciona muito mal em (b). Esse resultado indica que não se pode realizar uma decomposição do vídeo em fragmentos muito curtos, como no exemplo (b).

Nos experimentos que produziram (a) e (b) foram acompanhados 50 pontos pelo algoritmo KLT, dos quais 35 foram selecionados pelos ciclos de refinamento executados em (a), e 49 foram selecionados pelos ciclos executados em (b). A pouca eliminação de pontos ocorrida em (b) indica que muitos pontos móveis deixaram de ser descartados durante a calibração do fragmento, sendo este outro problema dos fragmentos curtos.

Por outro lado, verificou-se que também existem motivos para evitar os fragmentos longos. Os experimentos com vídeos de realidade aumentada mostraram que, embora o aumento no comprimento dos fragmentos reduza o número de junções destes em um vídeo, os erros inseridos pelas junções se tornam cada vez mais perceptíveis. Ao utilizar fragmentos mais curtos, o erro passa a ser melhor distribuído ao longo da trajetória da câmera, gerando resultados como os da Figura 7.8.

7.6 Modelagem geométrica

Para que fosse possível posicionar objetos virtuais na cena real, foi desenvolvido o módulo Modelador Geométrico, que permite que um usuário modifique o posicionamento de um objeto poliedral codificado no formato PLY, de maneira interativa.

O módulo fornece uma interface gráfica que permite ao usuário visualizar simultaneamente o objeto virtual sobre um conjunto de quadros do vídeo, e modificar sua

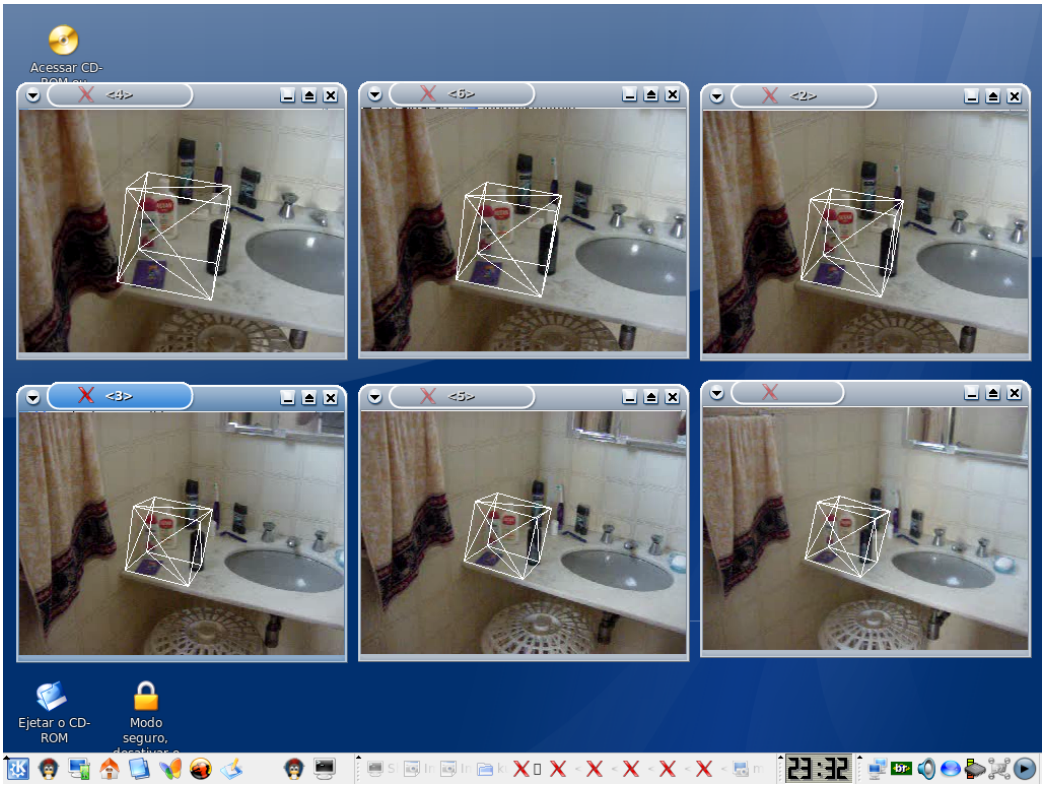

Figura 7.7: Interface gráfica do Modelador Geométrico.

posição, bastando para isso utilizar teclas para redimensionar, transladar ou rotacionar o objeto. A Figura 7.7 exibe a interface gráfica do Modelador Geométrico.

7.7 Resultados finais

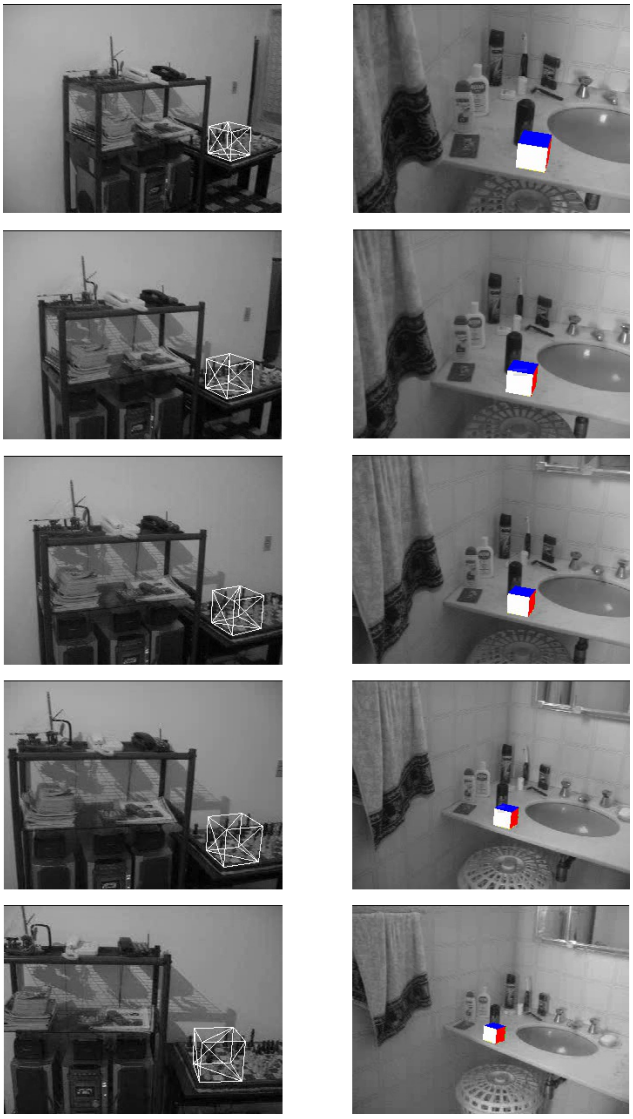

Figura 7.8: Quadros de vídeos gerados pelo módulo Combinador de Imagens.

7.8 Considerações sobre desempenho

O trabalho experimental teve por objetivo ilustrar o processo de seleção de pontos do algoritmo apresentado na dissertação, e mostrar sua aplicabilidade em realidade aumentada. Não foi feita uma análise detalhada do desempenho deste algoritmo. Os motivos desta omissão foram os seguintes:

1. O algoritmo apresentado utiliza muitos parâmetros especificados pelo usuário de forma empírica. Como a escolha destes parâmetros influencia significativamente no desempenho do algoritmo, acreditamos que seja necessário reduzir o número de parâmetros antes de relaciona-los com o tempo gasto para calibrar vídeos. Deixamos esse problema para um trabalho futuro.

2. Foi utilizado o algoritmo Levenberg-Marquardt da biblioteca GNU Scientific Library. Esta biblioteca não explora particularidades do problema de calibração, que podem ser utilizadas para reduzir a complexidade deste algoritmo [8]. Por esse motivo, o tempo de execução do protótipo tem seu tempo aumentado, não refletindo o tempo que seria alcançado com o uso de uma implementação de Levenberg-Marquardt otimizada.

A grosso modo, obtivemos uma relação da ordem de dezenas de minutos para calibrar cada segundo de vídeo. Nestes testes foi utilizando um computador com processador Pentium IV de 3GHz.

Capítulo 8

Conclusões e trabalhos futuros

Apresentamos nesta dissertação um algoritmo capaz de determinar os parâmetros extrínsecos das câmeras utilizadas na captação de um vídeo, sem a necessidade de realizar nenhum tipo de marcação sobre os objetos da cena. Os resultados obtidos foram suficientemente bons para fazer realidade aumentada em vídeos de curta duração.

Foi descrito de forma mais detalhada que em [6] a resolução do problema de estimação de uma explicação projetiva por uma solução em três passos, sem o uso de tensores trifocais. Foram explicitados os possíveis problemas durante a execução desses três passos, tendo sido apresentadas soluções, que foram testadas no protótipo implementado. Apresentou-se também um método de refinamento para a solução obtida em três passos, que foi chamado de ciclo de refinamento, e cujo efeito positivo foi avaliado no capítulo anterior.

8.1 Problemas pendentes na calibração

O método de calibração apresentado ainda possui as seguintes deficiências, que esperamos que sejam resolvidas em trabalhos futuros:

1. Existem muitos limiares independentes que precisam ser ajustados para que o algoritmo funcione apropriadamente;

2. Não existem garantias de que em todos os passos do algoritmo existirá um conjunto suficiente de famílias de pontos homólogos para que se possa aplicar a proposição 3.1;

3. O resultado final não é uma otimização global sobre o erro de reprojeção em todos os quadros do vídeo. O que o algoritmo faz é uma otimização em cada fragmento, seguida de uma junção ótima das famílias de câmeras estimadas.

8.2 Propostas para trabalhos futuros

Apresentamos agora algumas propostas de possíveis trabalhos que podem ser desenvolvidos como continuação deste. Foram feitos alguns experimentos iniciais de algumas dessas continuações, como será mostrado.

8.2.1 Problema de visibilidade

O processo de calibração de câmeras não produz informação suficiente para que se possa em geral combinar objetos virtuais de forma geometricamente consistente com um vídeo. Isso se deve ao fato do objeto virtual poder ser parcialmente ocludido pela cena. Nos experimentos esse problema foi solucionado posicionando o objeto virtual de forma a ficar entre a câmera e a cena em todos os quadros. Neste caso é necessário apenas sobrepor a imagem do objeto virtual sobre os quadros do vídeo.

Uma possível continuação para o trabalho seria estimar a geometria da cena a partir dos quadros do vídeo, e da família de câmeras obtida pela calibração. Com isso seria possível atacar o problema de visibilidade levando em consideração tanto as superfícies dos objetos virtuais como as superfícies da cena.

8.2.2 Ferramenta de modelagem para realidade aumentada

Os experimentos realizados com o módulo Modelador Geométrico mostraram que é difícil posicionar objetos virtuais apenas observando suas projeções em um conjunto de quadros. Essa tarefa ficaria muito mais fácil se fosse possível para o usuário estabelecer algum tipo de relação entre o objeto virtual e a cena, como apoiar ou alinhar o objeto virtual com objetos reais.

Em sistemas de realidade aumentada que utilizam marcações na cena pode-se normalmente definir um sistema de coordenadas onde é fácil encostar o objeto virtual na cena. Isso ocorre, por exemplo, em sistemas baseados na biblioteca ARToolKit, onde objetos virtuais são desenhados sobre quadrados desenhados em superfícies planas. Infelizmente isso não ocorre no nosso caso.

Seria interessante que fosse desenvolvida um ferramenta de modelagem geométrica capaz de posicionar objetos virtuais, de forma que o usuário conseguisse estabelecer relações com a cena, mesmo sem esta ter sido marcada.

8.2.3 Fotorrealismo

Além dos aspectos geométricos, tem-se que para que um objeto virtual seja integrado de maneira realista em uma imagem é necessário que exista uma compatibilização entre a iluminação da cena e a iluminação usada para gerar o objeto virtual. Uma abordagem possível seria estimar o posicionamento das fontes de luz da cena e utilizar essa informação na síntese da imagem do objeto virtual. Essa abordagem apresenta alguns problemas, como por exemplo, a inexistência de sombras entre objetos virtuais e objetos presentes no vídeo.

Uma abordagem muito mais ambiciosa seria buscar a compatibilização de iluminação via construção de um modelo global de iluminação que integrasse tanto o objeto virtual como um modelo da cena estimado a partir do vídeo. Esse modelo precisaria conter informações geométricas e radiométricas sobre as superfícies da cena, incluindo informações sobre superfícies que não aparecem no vídeo, mas que interferem na iluminação.

Foram feitos alguns experimentos com o objetivo de produzir realidade aumentada com melhor qualidade visual. Para isso substituiu-se a biblioteca OpenGL, no módulo Combinador de Imagens, pelo programa YafRay (*Yet Another Free Ray Tracer*), que utiliza um modelo global de iluminação para gerar imagens a partir de uma descrição de cena codificada no formato XML. Esse trabalho encontra-se em desenvolvimento, e ainda não foi feito nenhum tipo de compatibilização de iluminação. Um resultado inicial pode ser visto na Figura 8.1.

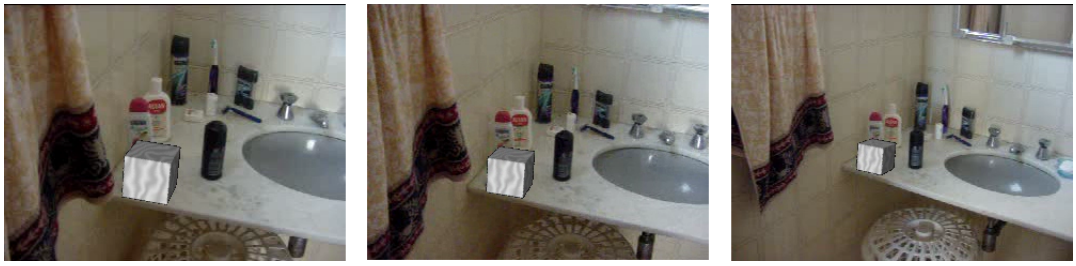

Figura 8.1: Composição da imagem de um cubo gerado pelo YafRay com alguns quadros de um vídeo.

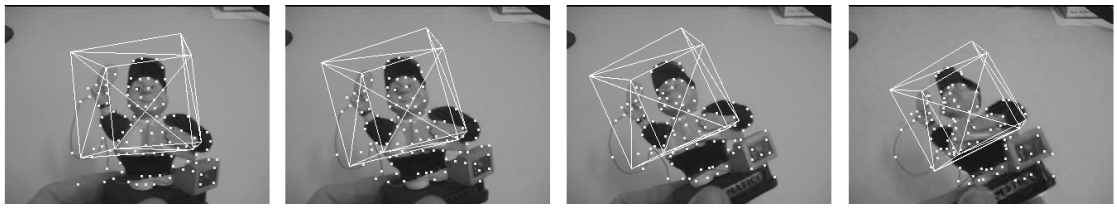

Figura 8.2: O cubo ao redor do boneco ilustra o uso da calibração na estimação do movimento realizado por um corpo rígido.

8.2.4 Acompanhamento espacial de corpos rígidos em vídeo

Podemos interpretar o resultado da calibração de uma câmera em relação a uma cena como sendo o movimento da cena em relação a câmera. Com isso, o sistema apresentado nessa dissertação poderia ser usado para estimar o movimento de rotação e translação de um corpo rígido em um vídeo, como ilustrado na Figura 8.2. Para que isso funcione é necessário que o KLT selecione uma quantidade maior de pontos no objeto que no fundo. Essa limitação pode ser facilmente contornada, pois uma versão do sistema capaz de acompanhar vários corpos rígidos pode pode ser criada modificando o algoritmo RANSAC, de forma que ele encontre diversos conjuntos de consenso, no lugar de encontrar o conjunto de consenso maximal.

Referências Bibliográficas

[1] F. Devernay and O. Faugeras. Automatic calibration and removal of distortion from scenes of structured environments. In *SPIE*, volume 2567, San Diego, CA, July 1995.

[2] Gerald Farin and Dianne Hansford. *The Geometry Toolbox for Graphics and Modeling*, chapter 12, page 181. AK Peters, LTD, 1998.

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

[4] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2003.

[5] Helmut Fritzsche. *Programação não-linear*. Edgar Blücher, 1978.

[6] Simon Gibson, Jon Cook, Toby Howard, Roger Hubbold, and Dan Oram. Accurate camera calibration for off-line, video-based augmented reality. In *International Symposium on Mixed and Augmented Reality (ISMAR'02)*, page 37, 2002.

[7] Jonas Gomes and Luiz Velho. *Fundamentos da Computacao Grafica*. IMPA, 2003.

[8] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in computer vision, second edition*. Cambrige University Press, Cambridge, United Kingdom, 2003.

[9] Richard I. Hartley. In defence of the 8-point algorithm. In *ICCV*, pages 1064–1070, 1995.

[10] Elon Lages Lima. *Curso de Análise Volume 2 - Sexta Edição*. IMPA, 2000.

[11] H. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[12] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI81*, pages 674–679, 1981.

[13] Ton Roosendaal and Stefano Selleri. *The Official Blender 2.3 Guide: Free 3D Creation Suite for Modeling, Animation, and Rendering*. No Starch Press, June 2004.

[14] Chaman L. Sabharwal. Stereoscopic projections and 3d scene reconstruction. In *SAC '92: Proceedings of the 1992 ACM/SIGAPP symposium on Applied computing*, pages 1248–1257, New York, NY, USA, 1992. ACM Press.

[15] Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, June 1994.

[16] C. Tomasi and T. Kanade. Detection and tracking of point features. *Technical Report CMU-CS-91-132*, 24(6), April 1991.

[17] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall PTR, Upper Saddler River, NJ, USA, 1998.

[18] R. Y. Tsai and T. S. Huang. Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:13–27, 1984.

[19] Luiz Velho and Jonas Gomes. *Sistemas Gráficos 3D*. Série Computação e Matemática. SBM / IMPA, 2001.

[20] Mason Woo, Jackie Neider, and Tom David. *OpenGL 1.2 Programming Guide, 3rd Edition: The Official Guide to learning OpenGL, Version 1.2*. Addison Wesley, 1999. WOO m 99:1 1.Ex.