



Universidade Federal de Alagoas

Programa de Pós-Graduação em Matemática

DISSERTAÇÃO DE MESTRADO

Um Sistema de Calibração de Câmera

Clarissa Codá dos Santos Cavalcanti Marques

Rio São Francisco

Um Sistema de Calibração de Câmera

Clarissa Codá dos Santos Cavalcanti Marques

Dissertação de Mestrado na área de concentração de Computação Gráfica submetida em 5 de Fevereiro de 2007 à Banca Examinadora, designada pelo Colegiado do Programa de Pós-Graduação em Matemática da Universidade Federal de Alagoas, como parte dos requisitos necessários à obtenção do grau de mestre em Matemática.

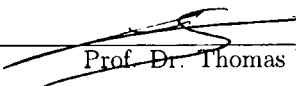
Banca Examinadora:



Prof. Dr. Adailson Peixoto (orientador)



Prof. Dr. Luiz Velho



Prof. Dr. Thomas Lewiner

Aos meus pais, Severino e Dilze.
Aos meus irmãos Gustavo e Fernando.

Agradecimentos

Primeiramente a Deus, pelo amor incondicional e eterno, e por não me deixar desamparada em nenhum momento de minha vida.

Aos meus queridos pais Severino e Dilze, e meus irmãos Fernando e Gustavo por todo o carinho, apoio, compreensão e força em todos os momentos de minha vida. A vocês só tenho a agradecer por tudo.

À minha família pelo carinho e força principalmente neste momento em que meus pais estão viajando.

Aos professores Luiz Velho e Thomas Lewiner pela disponibilidade, sugestões e contribuições de extrema importância para o desenvolvimento deste trabalho.

Ao professor Hilário Alencar pela significativa contribuição na minha vida acadêmica, em especial à orientação na Iniciação Científica.

Aos professores Adán Corcho, Krerley Oliveira, Ediel Azevedo e Marcos Petrúcio pelos ensinamentos e cursos ministrados no mestrado.

Ao professor Vínicius Mello pelas conversas e colaboração com este trabalho.

Aos meus colegas de turma Júlio Cesar, Thales Miranda e Thiago Fontes pela amizade e companherismo durante o mestrado.

Aos amigos André Pizzaia, Daniel Nicolau, Erikson Alexandre, Fábio Boia, José Arnaldo, Leandro Favacho, Márcio Henrique, Sofia Carolina, e Március Petrúcio pela amizade, descontrações e carinho que me proporcionaram neste tempo.

Aos amigos Allyson Cabral, Denisson Padilha, Douglas Cedrim, Kledson Marques, e Pedro Henrique (Ppeu/POG), pela maravilhosa e sincera amizade, companherismo, colaboração com o desenvolvimento do trabalho e alegrias proporcionadas. Com vocês todos os momentos de estresse tornaram-se momentos de muitas risadas.

Um agradecimento em especial aos amigos Claudemir Leandro e Maria de Andrade, por toda a amizade, aprendizado, carinho, compreensão, paciência e constante presença, mesmo distantes, em minha vida. Muito obrigada por tudo.

Um agradecimento muito especial ao professor Adelailson Peixoto, pela orientação neste trabalho, pelo seu apoio tanto no meio acadêmico quanto pessoal. Pelos seus conselhos e pela sua preciosa amizade que contribuíram com a minha formação. E principalmente por acreditar em mim e dar forças em todos os momentos que precisei.

A todos do Instituto de Matemática e de Computação que contribuíram com este trabalho e que estavam sempre presentes em minha vida.

À FAPESP, pelo apoio financeiro.

Resumo

Um processo de calibração de câmera consiste no problema de determinar as características geométricas digitais e ópticas da câmera a partir de um conjunto de dados iniciais. Este problema pode ser dividido em três etapas: aquisição de dados iniciais, o processo de calibração em si e otimização. Este trabalho propõe o desenvolvimento de uma ferramenta de calibração baseada em uma arquitetura genérica para qualquer processo de calibração. Para este propósito, o sistema apresentado neste trabalho permite a personalização de cada etapa da calibração. A inclusão de novos métodos de calibração é realizada de forma dinâmica, permitindo assim maior integração e flexibilidade entre os módulos do sistema.

Palavras-chave: Calibração de Câmera, Geometria Projetiva, Fotografia 3D.

Abstract

A camera calibration procedure corresponds to determine the digital geometric and optical characteristics of the camera from a known initial data set. This problem can be divided into three steps: a) acquisition of initial data; b) calibration process itself; and c) optimization. This work presents the development of a calibration tool based on a generic architecture for any calibration approach. For this aim, the presented system allows the personalization of each calibration step. In the proposed tool new calibration procedures are introduced dynamically, allowing a better integration between the modules of the system.

Keyword: Camera Calibration, Geometry Projective, 3D Photography.

Sumário

Introdução	1
Objetivos	1
Motivação	1
Estrutura da Dissertação	2
1 Calibração de Câmera e Geometria Projetiva	3
1.1 Introdução	3
1.2 Geometria Projetiva	5
1.2.1 Transformações Perspectivas no Espaço Euclidiano	5
1.2.2 Espaço Projetivo	6
1.2.3 Transformações Projetivas	7
1.3 Transformações de Câmera	9
1.3.1 Mudança do SCM para SCC	10
1.3.2 Mudança do SCC para SCI	11
1.3.3 Distorções das Lentes	12
1.3.4 Mudança do SCI para SCP	12
1.3.5 Composição das Transformações	13
2 Arquitetura Proposta para um Sistema de Calibração	15
2.1 Introdução	15
2.2 Etapas Envolvidas	15
2.2.1 Etapa de Aquisição de Dados Iniciais	16
2.2.2 Etapa de Calibração	17
2.2.3 Etapa de Otimização	17
2.3 Arquitetura Proposta	18
3 Aquisição de Dados e Correspondência	19
3.1 Introdução	19
3.2 Padrões	19
3.3 Imagens	22
3.4 Correspondência	23
3.4.1 Correspondência para padrão Xadrez	24
3.4.2 Correspondência para padrão com Círculos	26
4 Métodos de Calibração	28
4.1 Introdução	28
4.2 Método Tsai Coplanar	30
4.3 Método Zhang	35

4.4	Método Direct Linear Transformation (DLT)	40
4.5	Método Círculos Coplanares com Invariância Quase-Afim	44
5	Otimização	49
5.1	Introdução	49
5.2	Levenberg-Marquardt	50
5.3	Gloptipoly	51
6	Implementação do Sistema de Calibração	53
6.1	Introdução	53
6.2	Módulo EditCalib	53
6.2.1	Padrão	55
6.2.2	Equipamento	56
6.2.3	Imagem	57
6.2.4	Correspondência	58
6.2.5	Calibração	59
6.2.6	Otimização	61
6.2.7	Gerenciador de Pipelines	62
6.3	Módulo ExecCalib	63
7	Testes e Resultados	65
7.1	Introdução	65
7.2	Resultados	65
7.2.1	Arquitetura	65
7.2.2	Módulo EditCalib	66
7.2.3	Módulo ExecCalib	68
7.3	Testes de Calibração	69
8	Conclusão e Trabalhos Futuros	76
8.1	Conclusão	76
8.2	Trabalhos futuros	77

Lista de Figuras

1.1	Calibração de Câmera.	3
1.2	Câmera Pinhole.	4
1.3	Projeção de uma imagem por uma câmera pinhole.	4
1.4	Conseqüências da distorção radial.	4
1.5	As dimensões não são preservadas na projeção perspectiva.	5
1.6	Projeção perspectiva de um ponto.	6
1.7	Interseção de retas paralelas no infinito.	7
1.8	As retas são preservadas por uma projeção perspectiva.	7
1.9	Os sistemas de coordenadas do processo de calibração de câmera	10
1.10	Transformação do SCM para SCC.	10
1.11	Transformação do SCC para SCI.	11
1.12	Transformação do SCI para SCP.	13
2.1	Etapas da Calibração de Câmera.	16
2.2	Padrão Xadrez.	16
2.3	Retângulos Encaixados.	16
2.4	Etapas de aquisição de dados iniciais.	17
2.5	Processo de Calibração de Câmera.	17
2.6	Modelagem do Sistema.	18
3.1	Exemplos de padrão não-coplanar.	19
3.2	Dados do padrão Xadrez.	21
3.3	Padrão de círculos.	22
3.4	Dados do padrão de círculos.	22
3.5	Resultado da precisão subpixel.	24
3.6	Determinação dos centros.	25
3.7	Determinação dos pontos médios.	25
3.8	Determinação dos pontos.	26
3.9	Determinação da grade de pontos.	26
3.10	Padrão após threshold.	27
3.11	Borda das componentes conexas.	27
3.12	Determinação dos centros e recuperação dos círculos.	27
4.1	Subclasses para a classe de Calibração.	29
4.2	Primeiro Estágio do Método Tsai.	34
4.3	Segundo Estágio do Método Tsai.	35
4.4	Etapas de Zhang.	40
4.5	Primeiro Estágio do DLT.	43
4.6	Triângulo.	44

4.7	Ortcentro	44
4.8	Segundo Estágio do DLT.	44
4.9	Caso (a).	46
4.10	Caso (b).	46
4.11	Caso (c).	46
4.12	Caso (d).	46
4.13	Caso (e).	46
4.14	Caso (f).	46
4.15	Etapas do Método dos Pontos Circulares.	48
5.1	Subclasses para a classe de Otimização.	49
6.1	Diagrama de Classes.	54
6.2	Módulo EditCalib.	55
6.3	Módulo ExecCalib.	64
7.1	Calibração não-convencional.	66
7.2	Combinação entre pipelines diferentes.	66
7.3	Novos pipelines.	66
7.4	Inclusão de um novo método de calibração.	67
7.5	Deletar um elemento.	67
7.6	Após clicar Deletar.	67
7.7	Alterar um elemento.	68
7.8	Selecione os dois elementos.	68
7.9	Após clicar Ligar.	68
7.10	Correspondência manual ou semi-automática.	69
7.11	Seleção dos pontos pelo usuário.	70
7.12	Extração dos demais pontos.	70
7.13	Seleção dos pontos pelo usuário.	71
7.14	Extração dos demais pontos.	71
7.15	Calibração de uma câmera.	72
7.16	Calibração de um projetor.	73
7.17	Execução de vários pipelines.	74
7.18	Execução de pipelines não-convencionais.	75

Introdução

Objetivos

Os métodos de calibração de câmera consistem em determinar as características geométricas e ópticas internas da câmera assim como sua orientação e posição em relação a um certo sistema de coordenadas do mundo. Este processo geralmente é dado pela resolução de um problema de otimização não-linear supondo ser conhecido um conjunto de pares de pontos P_i no sistema de coordenadas da câmera e sua projeção p_i no sistema de coordenadas em pixels.

Esta dissertação tem como principal objetivo o desenvolvimento de um sistema de calibração baseado em uma arquitetura geral para qualquer processo de calibração. Isto é, uma ferramenta na qual seja possível criar processos de calibração adaptados com a possibilidade de incluir diferentes métodos à estrutura proposta neste trabalho.

Inicialmente foram estudados vários métodos de calibração, entre os quais podemos destacar o método de Tsai (Tsai 1987), (Horn 2000), (Tapper et al. 2002), método de Zhang (Zhang 1998), método usando círculos coplanares e invariância quase-afim (Wu et al. 2006). O estudo destes métodos serviu de base para a proposta de uma arquitetura geral, em cima da qual fosse possível a implementação destes e de outros métodos de calibração.

O sistema desenvolvido consiste em dois módulos. O primeiro é denominado Editor de Calibração (EditCalib) e o segundo é o Sistema de Calibração (ExecCalib).

O módulo EditCalib permite ao usuário criar e editar *pipelines* de calibração de câmeras e/ou projetores. Um pipeline de calibração consiste em uma sequência de objetos que se comunicam de forma que o objeto final possua como saída o resultado da calibração, conforme veremos no capítulo 2.

O módulo ExecCalib tem como funcionalidade executar os pipelines gerados no módulo anterior, com a possibilidade de alterar os dados de cada elemento do pipeline.

Motivação

Existem diversos métodos de calibração de câmera (Alvarez 2001), (Kushal et al. 2002), e alguns deles funcionam através da captura de imagens de um padrão de calibração (Tsai 1987), (Zhang 1998), (Wu et al. 2006), (Szenberg 2001). O padrão é construído de forma que seja relativamente fácil determinar pontos cujas coordenadas sejam conhecidas.

São vários os trabalhos e sistemas de calibração disponíveis que geram bons resultados para calibrações comuns (Strobl et al. n.d.), (*Photo 3D* 1999), ou seja, com elementos básicos e métodos padrões. Porém, nota-se a ausência de um sistema de calibração que permita calibrar vários equipamentos, com a possibilidade de utilizar métodos diferentes para cada um, e realizar uma análise comparativa entre os mesmos com o resultado de cada calibração.

A motivação deste trabalho portanto surgiu durante a revisão bibliográfica sobre os métodos de calibração. A não existência destes sistemas de calibração que possam se adaptar a qualquer método, afim de posteriormente comparar seus resultados deu o impulso inicial para a proposta de uma estrutura generalizada e elaboração desta dissertação.

Estrutura da Dissertação

A dissertação consiste em oito capítulos assim organizados:

No Capítulo 1 estudamos o problema de calibração de câmera e as transformações existentes em cada etapa. Damos uma introdução à Geometria Projetiva.

No Capítulo 2 enumeramos as etapas necessárias para uma calibração, e com base nessas etapas, propomos uma arquitetura para um sistema genérico de calibração.

No Capítulo 3 descrevemos a etapa correspondente ao processo de aquisição de dados, e como estes dados estão interrelacionados.

No Capítulo 4 falamos a etapa de calibração em si e discutimos sobre vários métodos de calibração estudados.

No Capítulo 5 discutimos como otimizar o cálculo de calibração e destacamos dois tipos de otimização.

No Capítulo 6 mostramos a implementação do sistema de calibração, exibindo os dois módulos em que ele consiste. Mostramos como as etapas descritas anteriormente podem ser incorporadas ao sistema proposto.

No Capítulo 7 exibimos os testes realizados com o sistema e seus resultados, utilizando os métodos de Tsai coplanar com o padrão xadrez e o Tsai não coplanar com o uso de pontos adquiridos em arquivos.

No Capítulo 8 finalmente apresentamos as conclusões e recomendações para trabalhos futuros.

Capítulo 1

Calibração de Câmera e Geometria Projetiva

1.1 Introdução

A calibração de câmera consiste no processo de determinar dados geométricos digitais e ópticos da câmera, admitindo que sejam conhecidos um conjunto de pares de pontos bidimensionais em uma imagem e seus respectivos pontos tridimensionais. Com este intuito, fica evidente a necessidade de definir relações entre as coordenadas de pontos 3D com as coordenadas 2D de imagens dos mesmos. Maiores detalhes pode ser visto em (Carvalho et al. 2005).

Existe uma transformação que relaciona estes pontos tridimensionais com os respectivos bidimensionais, exceto por distorções e erros mínimos. Equacionando estas relações com o uso de equações lineares na variável de posição de um objeto, os coeficientes destas equações serão exatamente funções dos dados que a calibração determina. O processo de calibração de câmera pode se resumir a encontrar tais valores a partir de um conjunto de pontos 3D e 2D correspondentes.



Figura 1.1: Calibração de Câmera.

Os dados resultantes da calibração podem ser classificados em dois grupos: extrínsecos e intrínsecos. Os parâmetros extrínsecos fornecem informações da posição e orientação da câmera em relação a um certo sistema de coordenadas global (ou do mundo). Os parâmetros intrínsecos nos dão características ópticas e geométricas internas da câmera, que correspondem a distância focal, fatores de escala, posição em pixels da projeção ortogonal do centro óptico no plano de projeção e distorções.

Em uma câmera sem lente contendo apenas um orifício por onde a luz é captada para o interior da câmera, usualmente denominada câmera pinhole, a imagem formada é determinada pela interseção dos raios luminosos com o fundo da câmera, como ilustra acima. Entretanto, é geralmente considerada a imagem formada pela interseção dos raios luminosos com o plano situado à mesma distância do fundo da câmera ao orifício, porém na frente.

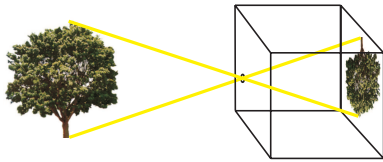


Figura 1.2: Câmera Pinhole.

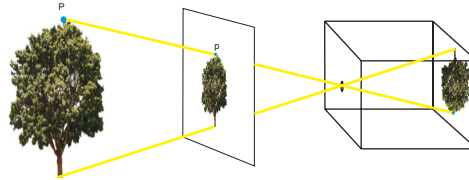


Figura 1.3: Projeção de uma imagem por uma câmera pinhole.

A câmera pinhole produz um processo de calibração ideal, onde fazemos a correspondência que associa a cada ponto P do espaço tridimensional ao ponto correspondente p do espaço bidimensional no plano de formação da imagem. Apesar disso, para produzir uma imagem nítida, a abertura de seu orifício deve ser pequeno suficiente, o que provocaria intensidade de luz baixa, já que poucos raios seriam captados pela câmera.

Este problema pode ser solucionado através do uso de lentes, onde o orifício possui uma abertura maior. Infelizmente, a calibração de câmera com lentes usuais deve apresentar distorções devido às irregularidades das lentes, como exemplo podemos citar a distorção radial que provocaria uma curvatura nas retas situadas nas bordas das imagens.

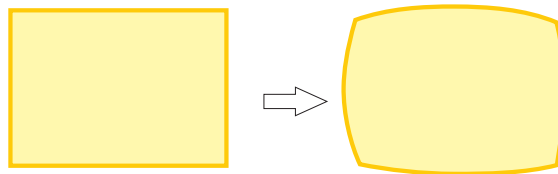


Figura 1.4: Conseqüências da distorção radial.

Com a introdução das distorções das lentes, como veremos na subseção 1.2.3, o processo de calibração de câmera torna-se um sistema de equações não-lineares que pode ser resolvido através de uma otimização não-linear, podendo-se tomar o resultado das equações lineares como solução inicial.

De um modo geral o processo de calibração pode ser descrito da seguinte forma:

1. Determinar um conjunto de pontos no sistema de coordenadas do mundo e suas respectivas projeções nas imagens;
2. Resolver as equações lineares;

- Otimizar os dados obtidos, utilizando o resultado da etapa anterior como solução inicial.

1.2 Geometria Projetiva

Esta seção traz uma breve introdução sobre geometria projetiva como base para as transformações necessárias para a obtenção das correspondências entre os pares de pontos (2D, 3D). Maiores informações sobre geometria projetiva (Velho & Gomes 2003), (Carvalho et al. 2005).

1.2.1 Transformações Perspectivas no Espaço Euclidiano

A transformação perspectiva é a mais comum e popular transformação usada na computação gráfica devido a sua capacidade de simular a profundidade em uma cena. Em uma projeção perspectiva as relativas dimensões não são preservadas, como pode ser visto na imagem abaixo.

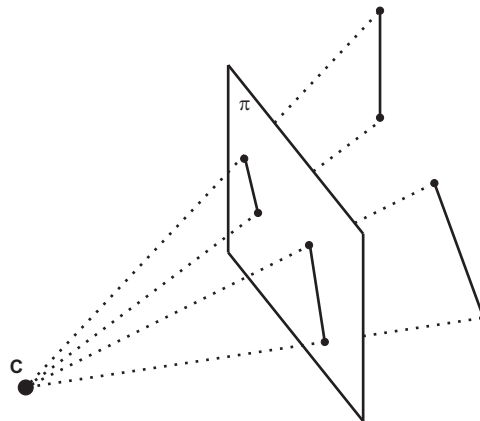


Figura 1.5: As dimensões não são preservadas na projeção perspectiva.

Considere o ponto C , conhecido como *centro de projeção* e um plano π situado a uma distância f do centro de projeção. A projeção perspectiva de um ponto no espaço euclidiano é definida como a interseção da reta que liga o mesmo ao centro de projeção com o plano π . A figura 1.6 mostra um ponto $P = (x_w, y_w, z_w)$ que está sendo projetado perspectivamente sobre o plano π em um ponto $P' = (x'_w, y'_w)$. Todos os pontos da reta \overline{CP} são da forma $\lambda(x_w, y_w, z_w)$. Ora, a equação do plano π é dada por $z'_w = f$, então temos que

$$z'_w = \lambda z_w \Rightarrow \lambda = \frac{f}{z_w}.$$

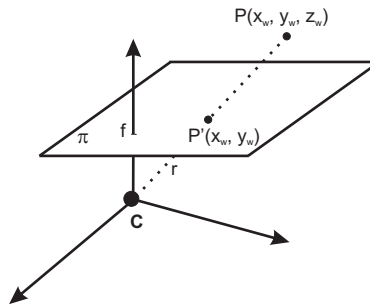


Figura 1.6: Projeção perspectiva de um ponto.

Daí, temos que a projeção de um ponto $P = (x_w, y_w, z_w)$ qualquer no sistema euclidiano com $z_w \neq 0$ é dada por

$$P' = \left(\frac{x'_w}{z'_w}, \frac{y'_w}{z'_w} \right),$$

considerando $f = 1$, para simplificar.

1.2.2 Espaço Projetivo

O espaço projetivo n -dimensional visa estender o espaço euclidiano incluindo novos pontos que serão denominados *pontos no infinito*. Para conhecermos melhor este ambiente, observemos três fatos sobre a projeção perspectiva. O primeiro é que a projeção perspectiva dos pontos P em que $z_w = 0$ não é definida. Estes pontos são os chamados pontos no infinito.

A segunda observação a ser feita é que retas paralelas se encontram no infinito. Para demonstrar este fato primeiro considere o ponto C do espaço euclidiano \mathbb{R}^{n+1} e o hiperplano $\Pi \subset \mathbb{R}^{n+1}$ tal que $C \notin \Pi$. Observe que todos os pontos da reta r que projeta um ponto P no plano Π possuem como projeção o ponto P' . Podemos dessa forma definir, em relação à projeção cônica, que uma reta r é um ponto projetivo “ P' ”. De forma análoga, podemos generalizar o fato anterior para planos, e assim concluíremos que as retas projetivas são planos que passam pela origem. E assim, através da figura 1.7 verificamos a segunda observação citada logo acima.

Podemos ainda generalizar as considerações anteriores e afirmar que os subespaços projetivos de dimensão m em relação à projeção cônica, são subespaços de dimensão $m + 1$ em \mathbb{R}^{n+1} , onde $m < n$.

O terceiro fato a ser observado é que a projeção perspectiva preserva retas. Para provar este argumento basta observar que três pontos colineares formam um plano que corta o plano de projeção π em uma reta, como ilustra a figura 1.8.

Assim, definiremos o espaço projetivo como

$$\mathbb{RP}^n = \{(x, 1) \cup (x, 0)\}, \quad x \in \mathbb{R}^n, \quad x \neq 0,$$

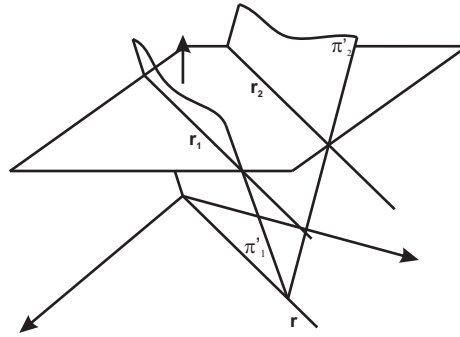


Figura 1.7: Interseção de retas paralelas no infinito.

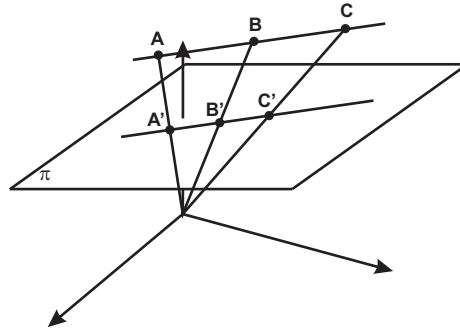


Figura 1.8: As retas são preservadas por uma projeção perspectiva.

onde os pontos da forma $(x, 1)$ são os pontos do plano euclidiano $z = 1$, também chamados *pontos afins*, e $(x, 0)$ são os pontos no infinito.

1.2.3 Transformações Projetivas

Dados dois espaços projetivos de dimensões m e n , as transformações $T : \mathbb{RP}^m \rightarrow \mathbb{RP}^n$ são dadas por transformações $T : \mathbb{R}^{m+1} \rightarrow \mathbb{R}^{n+1}$, chamadas transformações projetivas.

Para simplificar tomemos $m = n = 2$. Então, a transformação $T : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ é dada por uma matriz de ordem 3, que pode ser dividida em 4 blocos:

$$M = \begin{pmatrix} a & c & t_1 \\ b & d & t_2 \\ p_1 & p_2 & s \end{pmatrix} = \begin{pmatrix} A & T \\ P & S \end{pmatrix}.$$

A seguir mostraremos algumas propriedades destas transformações. Suponha,

$$P = \begin{pmatrix} 0 & 0 \end{pmatrix}, \quad T = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad S = (1).$$

Ou seja,

$$M = \begin{pmatrix} a & c & 0 \\ b & d & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Agora, aplicando a transformação acima em um ponto do infinito temos que:

$$M = \begin{pmatrix} a & c & 0 \\ b & d & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} = \begin{pmatrix} ax + cy \\ bx + dy \\ 0 \end{pmatrix}.$$

Portanto, a transformação sobre um ponto do infinito resultou em um ponto do infinito, o que mostra que as retas no infinito permanecem invariantes sobre tal transformação. Agora, se tomarmos um ponto afim,

$$M = \begin{pmatrix} a & c & 0 \\ b & d & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} ax + cy \\ bx + dy \\ 1 \end{pmatrix}.$$

Vemos que o ponto resultante também é um ponto afim. Ou seja, o plano afim do plano projetivo também é invariante pela transformação. Vale notar que,

$$\begin{pmatrix} a & c \\ b & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} ax + cy \\ bx + dy \end{pmatrix}$$

é apenas uma transformação linear do plano euclidiano. Isto mostra que as transformações projetivas do plano contém as transformações lineares do plano euclidiano. Agora, tomemos

$$\begin{pmatrix} 1 & c & 0 \\ 0 & 1 & 0 \\ p_1 & p_2 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ p_1x + p_2y + 1 \end{pmatrix}.$$

Ora, se $p_1 \neq 0$ ou $p_2 \neq 0$, temos que

$$p_1x + p_2y + 1 = 0$$

possui uma infinidade de soluções, e assim teremos um ponto afim sendo projetado em um ponto do infinito. E por outro lado,

$$\begin{pmatrix} 1 & c & 0 \\ 0 & 1 & 0 \\ p_1 & p_2 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} = \begin{pmatrix} x \\ y \\ p_1x + p_2y \end{pmatrix}.$$

Tomando pontos (x, y) de forma que $p_1x + p_2y \neq 0$, concluímos que pontos do infinito podem ser levados em pontos afins pelas transformações projetivas. Assim, nota-se que as transformações projetivas possuem um tratamento unificado para os diversos tipos de transformações do espaço, e devido a esses fatos sua importância na computação gráfica.

1.3 Transformações de Câmera

Como vimos anteriormente, o processo de calibração inicialmente busca corresponder os pontos tridimensionais com suas respectivas projeções bidimensionais no plano da imagem. Desta forma, podemos identificar 4 sistemas de coordenadas, de modo que a correspondência entre esses pontos possa ser expressa como uma composição de transformações que associe os pontos 3D aos 2D correspondentes.

Os sistemas de coordenadas considerados são:

Sistema de Coordenadas do Mundo (SCM) trata-se de um sistema de coordenadas tridimensionais escolhidos de forma conveniente a definir as coordenadas de cada ponto da cena. As coordenadas de um ponto P_w neste sistema serão denotadas como (x_w, y_w, z_w) , como ilustra a figura 1.9.

Sistema de Coordenadas da Câmera (SCC) consiste em um sistema de coordenadas com origem no centro óptico da câmera. Os eixos x e y são escolhidos de forma que sejam paralelos ao plano da imagem, e o eixo z sendo o mesmo do eixo óptico. Denotaremos como f a distância entre a origem deste sistema, e o plano de projeção, usualmente denominada distância focal, e (x, y, z) como as coordenadas do ponto P neste sistema. Assim definindo, vemos que a equação do plano de projeção é dada por $z = f$.

Sistema de Coordenadas da Imagem (SCI) é um sistema de coordenadas bidimensional situado no plano de projeção. A origem deste sistema corresponde à projeção ortogonal do centro óptico no plano de projeção. Iremos denotar as coordenadas de um ponto neste referencial por (X, Y) . Denotaremos também (X_u, Y_u) como as coordenadas da imagem, quando usamos uma câmera do tipo pinhole, isto é, sem as distorções da lente, e (X_d, Y_d) como as coordenadas da imagem com as distorções existentes.

Sistema de Coordenadas em Pixels (SCP) também consiste de um sistema de coordenadas bidimensional. As coordenadas de um ponto da imagem neste sistema são expressas em pixel. Usualmente, toma-se como centro deste sistema o canto superior esquerdo da imagem. As coordenadas de um ponto neste referencial será denotada por (X_f, Y_f) .

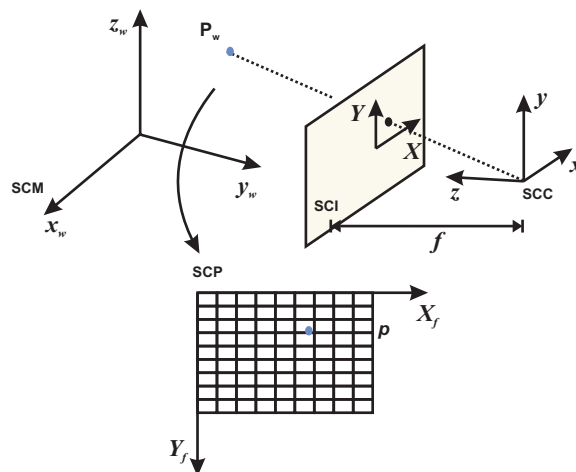


Figura 1.9: Os sistemas de coordenadas do processo de calibração de câmera

Então, as transformações necessárias para fazer a correspondência entre os pontos devem ser efetuadas entre cada sistema de coordenadas de forma a associar o ponto no SCM ao ponto correspondente no SCP. Assim, definiremos três transformações relacionadas em quatro etapas consecutivas que serão detalhadas nas subseções a seguir. Antes de iniciar, deve-se notar que inicialmente iremos tratar uma câmera do tipo pinhole, e em seguida consideraremos as distorções da câmera.

1.3.1 Mudança do SCM para SCC

A primeira etapa no processo de correspondência entre os pontos do sistema de coordenadas do mundo e da imagem, consiste em expressar as coordenadas de um ponto P do SCM no SCC. Para isto, é suficiente realizar uma mudança de coordenadas entre esses dois sistemas ortogonais, através uma transformação de corpo rígido.

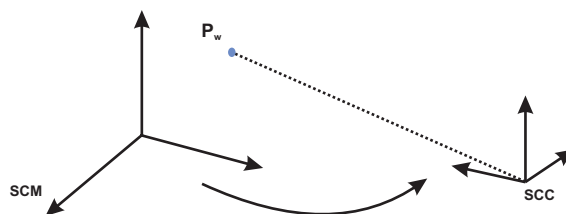


Figura 1.10: Transformação do SCM para SCC.

Sejam T o vetor com coordenadas da origem do SCM e R a matriz cujas colunas r_1, r_2, r_3 sejam as coordenadas dos vetores $\vec{i}, \vec{j}, \vec{k}$ no sistema de coordenadas da câmera. Então, dado um ponto P no SCM, suas coordenadas no SCC podem ser definidas da seguinte forma:

$$(x, y, z) = T + x_w \cdot r_1 + y_w \cdot r_2 + z_w \cdot r_3. \quad (1.1)$$

Reescrevendo a equação em forma matricial temos,

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + T.$$

ou,

$$P = RP_w + T, \quad (1.2)$$

onde

$$T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \text{ e } R = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix}.$$

Os parâmetros a serem calibrados são R e T , que definem a orientação e posição da câmera, ou seja correspondem aos parâmetros extrínsecos definidos na seção anterior.

Vale salientar que uma transformação de corpo rígido de um sistema de coordenadas cartesianas para outro é única se a transformação é definida como uma rotação 3D em torno de sua origem seguida de uma translação 3D.

1.3.2 Mudança do SCC para SCI

A segunda etapa consiste de uma projeção perspectiva que descreve o ponto com coordenadas no SCC em coordenadas da SCI, desconsiderando as distorções da câmera.

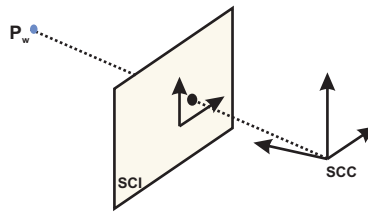


Figura 1.11: Transformação do SCC para SCI.

As coordenadas do resultado da projeção podem ser definidas como abaixo:

$$X_u = f \frac{x}{z} \quad Y_u = f \frac{y}{z}. \quad (1.3)$$

Podemos escrever a equação dada acima na forma matricial para obtermos:

$$\begin{bmatrix} X_u \\ Y_u \\ 1 \end{bmatrix} \simeq \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} fx & fy & z \end{bmatrix}.$$

Nesta etapa, o parâmetro a ser calibrado é a distância focal f . Esta projeção definida acima é não-inversível. Cada ponto do SCM corresponde a um único ponto da imagem, porém o inverso não é verdade, um ponto no SCI corresponde a uma infinidade de pontos do SCM, como vimos na seção de Transformação Projetiva.

1.3.3 Distorções das Lentes

Até o momento estávamos considerando lentes do tipo pinhole. Estes modelos de câmeras possuem a vantagem de obter os resultados de uma forma mais fácil e simples.

Uma solução era o uso de lentes que possuem a abertura do orifício maior que as da câmera pinhole, permitindo uma maior intensidade de luz na formação da imagem. Em contrapartida, a modelagem exata do comportamento das lentes se torna impraticável devido a sua complexidade. Usualmente é utilizado um modelo onde consideram-se apenas as distorções radiais.

A distorção radial, como o nome mesmo já indica, ela afeta todos os pontos da imagem situados a uma mesma distância do centro óptico no plano da imagem. Devido a esta distorção, na etapa anterior, ao invés do ponto $p = (x, y, z)$ ser projetado no ponto $P_u = (X_u, Y_u, 1)$, que corresponde ao ponto sem a distorção, ele será projetado no ponto $P_d = (X_d, Y_d, 1)$.

Este ponto P_d relaciona-se ao ponto P_u da seguinte forma:

$$P_u = P_d + D, \quad (1.4)$$

onde $D = (D_x, D_y)$, e

$$D_x = X_d(k_1 r^2 + k_2 r^4)$$

$$D_y = Y_d(k_1 r^2 + k_2 r^4),$$

$$r = \sqrt{X_d^2 + Y_d^2}.$$

Em geral, consideramos apenas a distorção k_1 , a menos que o ângulo de visão da lente seja muito grande.

1.3.4 Mudança do SCI para SCP

Em uma câmera digital quando o raio luminoso atinge o plano de formação da imagem, ele é registrado por sensores, os quais geralmente estão dispostos segundo uma matriz retangular.

Nem sempre as linhas e colunas desta matriz estão uniformemente alinhadas, devido às imperfeições na construção da mesma. Pode ocorrer de as linhas terem espaçamentos

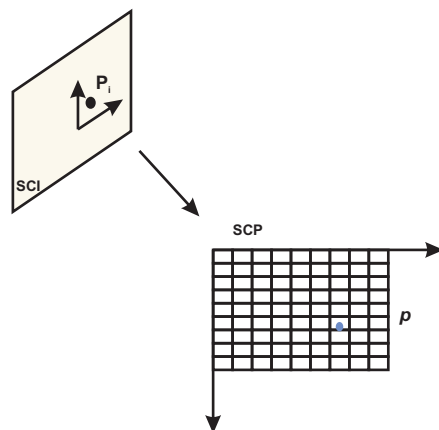
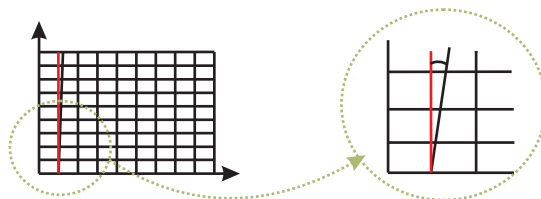


Figura 1.12: Transformação do SCI para SCP.

diferentes, assim como as linhas verticais não serem exatamente perpendiculares as linhas horizontais.



Levando em consideração estas pequenas distorções geométricas, a transformação que descreve um ponto no SCI para o SCP, pode ser descrita como abaixo:

$$\begin{bmatrix} X_f \\ Y_f \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & \tau & C_x \\ 0 & s_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_u \\ Y_u \\ 1 \end{bmatrix}, \quad (1.5)$$

onde

s_x e s_y correspondem ao número de pixels por unidade de comprimento em ambas direções, horizontal e vertical, respectivamente;

τ é dada pela tangente do ângulo que as colunas da matriz de pixels formam com a perpendicular com as linhas;

C_x e C_y representam as coordenadas da projeção ortogonal do centro óptico no plano de projeção.

1.3.5 Composição das Transformações

Desta forma, a transformação que associa um ponto do sistema de coordenadas do mundo ao sistema de coordenadas em pixels pode ser representado fazendo uma composição

das transformações descritas nas seções anteriores. O resultado final obtido encontra-se abaixo:

$$\begin{aligned} \begin{bmatrix} X_f \\ Y_f \\ 1 \end{bmatrix} &\simeq \begin{bmatrix} s_x & \tau & C_x \\ 0 & s_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} \\ [p] &\simeq \begin{bmatrix} fs_x & f\tau & C_x \\ 0 & fs_y & C_y \\ 0 & 0 & 1 \end{bmatrix} [R \ T][P]. \end{aligned} \quad (1.6)$$

A matriz $K = \begin{bmatrix} fs_x & f\tau & C_x \\ 0 & fs_y & C_y \\ 0 & 0 & 1 \end{bmatrix}$ é chamada de *matriz intrínseca de calibração* e agrupa todos os parâmetros intrínsecos da câmera. A equação 1.6 não é inversível. Ora,

$$\bar{p} = K^{-1}[p] \simeq [R \ T][P].$$

O ponto $\bar{p} = (\bar{x}, \bar{y}, 1)$ corresponde à imagem do ponto P tomando a matriz de calibração como a identidade. Os pontos que possuem p como projeção seriam da forma $\lambda(\bar{x}, \bar{y}, 1)$, onde $\lambda \in \mathbb{R}$. As coordenadas de \bar{p} são conhecidas como *coordenadas normalizadas do ponto p* da imagem.

Capítulo 2

Arquitetura Proposta para um Sistema de Calibração

2.1 Introdução

Uma das dificuldades de se projetar um sistema genérico de calibração de câmera está na variedade dos dados de entrada que cada método requer, principalmente na especificação e na correspondência entre estes dados. Outro obstáculo reside na forma como estes dados são transformados durante o processo de calibração. Este capítulo apresenta a modelagem de um sistema genérico de calibração. Para tal é necessário analisar o problema de uma forma global e identificar quais etapas estão envolvidas num processo de calibração, identificando os elementos necessários em cada etapa.

2.2 Etapas Envolvidas

Um processo padrão de calibração de câmera, como vimos no capítulo anterior, pode ser estruturado da seguinte forma:

1. Aquisição dos dados (especificação e correspondência);
2. Calibração (resolução de equações lineares);
3. Otimização dos dados obtidos, utilizando o resultado da etapa anterior como solução inicial.

Pelas etapas acima, podemos dizer que a calibração de uma câmera se resume no processo de obter os seus parâmetros, a partir de um conjunto de dados iniciais, conforme mostra a figura 2.1. É importante salientar que existem métodos que não necessitam dos elementos que iremos citar aqui, e nem ao menos requerem todas as etapas definidas acima. Portanto, estamos tentando definir elementos que possam vir a ser necessários em um processo de calibração generalizado.

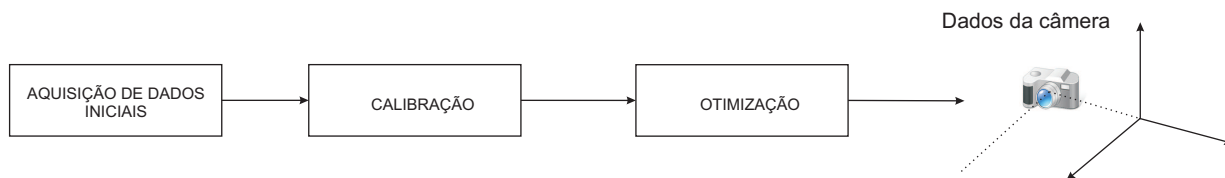


Figura 2.1: Etapas da Calibração de Câmera.

2.2.1 Etapa de Aquisição de Dados Iniciais

Esta etapa está altamente relacionada com a especificação do usuário, a partir da qual o conjunto de dados de entrada será definido. Esses dados, em geral, são composto por um conjunto de pares de pontos 2D e 3D. A partir desta especificação é realizada uma correspondência entre estes dois conjuntos.

Usualmente o conjunto de pontos 3D correspondem a pontos de um padrão de calibração. Diversos métodos utilizam padrões cujas coordenadas de seus pontos sejam relativamente fáceis de se calcular. São vários os padrões utilizados para calibração de câmera. Podemos citar como exemplo o padrão xadrez (2.2), o mais popularmente conhecido e utilizado, devido à simplicidade de obter as coordenadas de seus pontos. Outro exemplo clássico é o de retângulos encaixados (Szenberg 2001), (Szenberg et al. 2001) que se interceptam como mostra a figura 2.3.

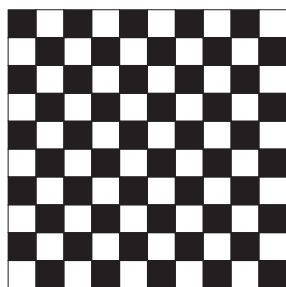


Figura 2.2: Padrão Xadrez.

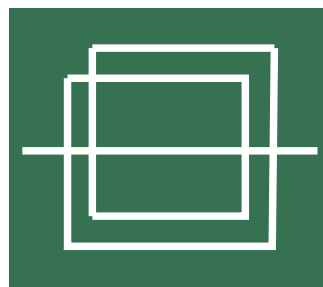


Figura 2.3: Retângulos Encaixados.

O conjunto de pontos 2D é extraído de uma imagem do padrão adquirida pelo equipamento a ser calibrado. As imagens são utilizadas para fazer a correspondência entre os pares de pontos 2D e 3D, e dependendo do método podem ser utilizadas várias imagens tiradas de ângulos diferentes ou mesmo apenas uma. A partir do padrão e das imagens precisamos de algum método para obter os pares de pontos correspondentes que serão usados na calibração. Nesta etapa de aquisição de dados iniciais definimos os seguintes elementos (figura 2.4):

- equipamento: corresponde ao equipamento que será calibrado (câmera ou projetor);
- padrão: corresponde ao padrão a partir do qual os pontos 3D serão obtidos.

- imagem: corresponde às imagens do padrão a partir das quais os pontos 2D serão obtidos;
- métodos de correspondência: indicam como os pontos 2D e 3D serão especificados e como eles se relacionam.

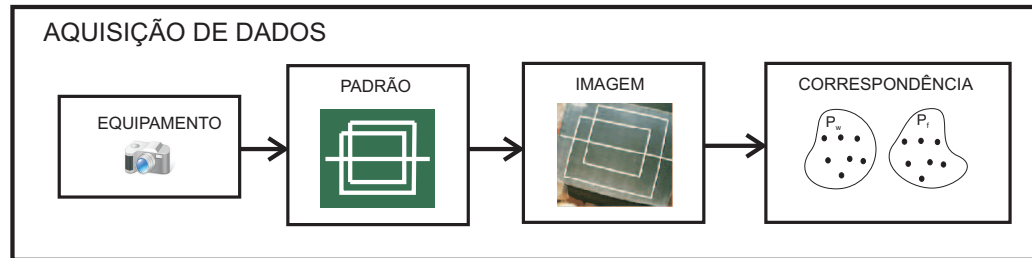


Figura 2.4: Etapa de aquisição de dados iniciais.

2.2.2 Etapa de Calibração

Dados um conjunto de pontos 2D e o correspondente conjunto de pontos 3D esta etapa tem como finalidade resolver um sistema de equações lineares que apontam uma solução inicial para os resultados intrínsecos e extrínsecos do equipamento a ser calibrado.

Existem várias heurísticas para resolver esta etapa, que define diferentes métodos como o Tsai, Zhang, etc. Assim o elemento a ser definido é o método de calibração. O capítulo 4 detalha alguns destes métodos.

2.2.3 Etapa de Otimização

Todos os elementos citados anteriormente possuem alta influência na precisão do resultado obtido pelo método de calibração. Isto implica que nem sempre os resultados são satisfatórios. Uma solução para este problema consiste em utilizar uma otimização para melhorar o resultado obtido. O método de otimização mais utilizado é o algoritmo de Levenberg-Marquadt (Ranganathan 2004), (Lourakis 2005), que provê uma solução numérica para o problema de minimização de uma função, esta geralmente não linear. Com isto, definimos o último elemento que pode fazer parte de um sistema de calibração.

A figura 2.5 mostra as etapas de calibração detalhando os elementos envolvidos.

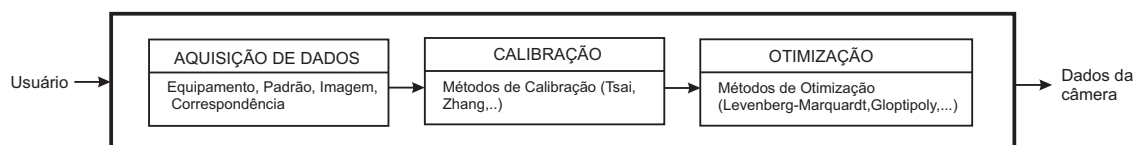


Figura 2.5: Processo de Calibração de Câmera.

2.3 Arquitetura Proposta

Para desenvolver um sistema adaptável a diferentes métodos de calibração, foi proposta uma arquitetura com todos os elementos possivelmente necessários a um pipeline padrão de calibração. Incluímos a possibilidade de adicionar novos padrões, equipamentos, métodos para calibração, correspondência e otimização permitindo assim uma maior variedade de pipelines.

A modelagem desta arquitetura encontra-se na figura 2.6.

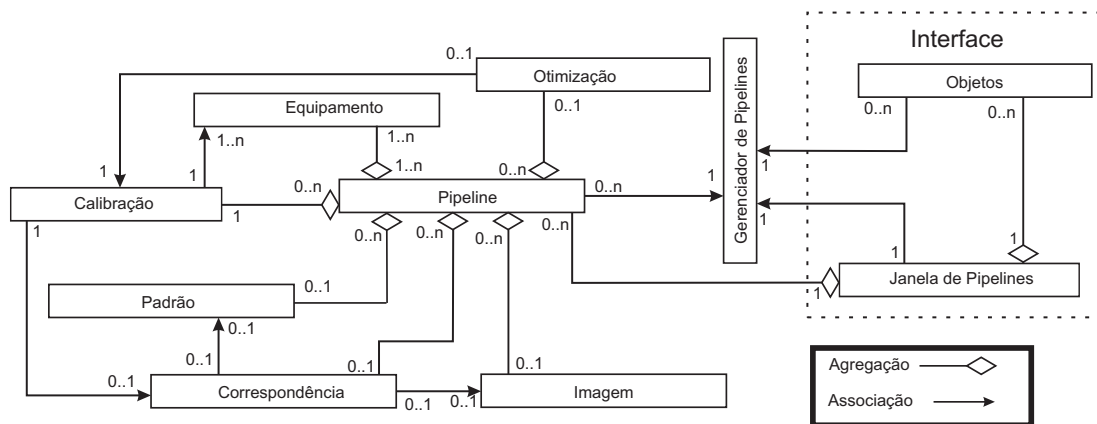


Figura 2.6: Modelagem do Sistema.

Cada elemento das etapas envolvidas corresponde a um módulo do sistema. Os relacionamentos envolvidos são basicamente de *associação*, que mostra o fluxo de dados entre os elementos, e a *agregação*, que mostra a relação de pertinência entre os elementos. Por exemplo, um objeto da classe pipeline contém um objeto da classe de calibração.

O elemento *padrão* e o elemento *imagem*, por exemplo, estão associados ao elemento *correspondência*, uma vez que os métodos de correspondência utiliza os dados do padrão e da imagem para obter os pontos 2D e 3D. O elemento *correspondência*, por sua vez, fornece os pontos 2D e 3D ao elemento *calibração*. Todos estes elementos estão agregados ao elemento *pipeline*.

Portanto a relação entre o elemento pipeline e os elementos que determinam um pipeline, conforme figura 2.6, pode ser definido como uma relação de agregação, ou seja todos estes elementos fazem parte do elemento pipeline.

O Gerenciador de Pipelines têm como principal finalidade adicionar (ou editar) a um pipeline os elementos que serão agregados a ele. A interface gráfica do sistema pode ser definida a partir dos elementos *objeto* e *janela de pipelines*. O elemento *objeto* são retângulos que serão exibidos na janela de pipelines e representam quaisquer elementos (equipamento, padrão, imagem, etc.). O elemento *janela de pipelines* é a janela principal onde os pipelines serão exibidos.

Capítulo 3

Aquisição de Dados e Correspondência

3.1 Introdução

Este capítulo trata dos elementos envolvidos na primeira etapa de calibração, que é a aquisição dos dados. Discute alguns tipos de padrão e alguns métodos de correspondência.

Vimos que a correspondência é realizada a partir de um padrão e de imagens deste padrão, que fornecem respectivamente um conjunto de pontos 3D e um conjunto de pontos 2D. A forma como estes pontos são especificados pelo usuário e como os pontos são relacionados determina diferentes métodos de correspondência.

Nas seções seguintes, classificamos e descrevemos os elementos (padrão, imagem e correspondência) utilizados em sistemas de calibração.

3.2 Padrões

Os padrões são referências utilizadas na calibração para obter os pontos tridimensionais. Geralmente são escolhidos padrões simples, de medidas e propriedades absolutamente conhecidas. Eles podem ser coplanares ou não-coplanares. Os padrões planares são mais atrativos devido a sua simplicidade de construção e determinação de suas coordenadas. Exemplos de padrões não-coplanares podem ser vistos nas imagens 3.1.

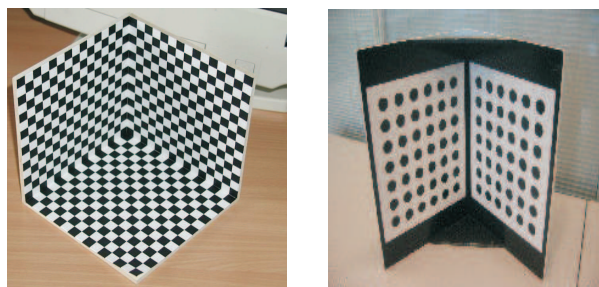


Figura 3.1: Exemplos de padrão não-coplanar.

Eles podem variar desde objetos comuns utilizados no dia a dia, como cenas, imagens, e até elementos geométricos como retas e círculos.

De acordo com a arquitetura proposta no capítulo 2, este trabalho implementa uma classe para representar padrões de forma que a mesma contenha as especificações necessárias para qualquer modelo de padrão a ser criado.

Para especificar um padrão é necessário definir as seguintes informações:

- Nome do padrão;
- Quantidade de parâmetros;
- Nome dos parâmetros;
- Valores dos parâmetros.

Os parâmetros do padrão correspondem às propriedades específicas que serão necessárias para determinar os pontos tridimensionais sobre o mesmo. Os valores destes parâmetros são os valores que eles assumem para que sejam calculados os pontos. O nome e os parâmetros do padrão são requeridos ao criar um novo padrão, enquanto que os valores do padrão deverão ser usados para a edição de um modelo já criado.

As operações envolvidas na definição de um padrão são:

- Atribuir/Ler o nome do padrão: possuem como finalidade atribuir/obter o nome do padrão;
- Definir/Ler os nomes dos parâmetros: especifica e obtém, respectivamente, o nome das propriedades específica do padrão.
- Obter a quantidade de parâmetros: informa a quantidade de parâmetros existentes no padrão.
- Atribuir/Ler valores aos parâmetros: atribui e recupera, respectivamente, os valores dos parâmetros de um objeto do padrão.

Vejamos alguns exemplos de padrão.

Exemplo 1: Padrão Xadrez

Os parâmetros definidos para este padrão são:

Tipo cujo valor está fixado como coplanar;

Dx que corresponde ao comprimento do retângulo;

D_y que corresponde à altura do retângulo;

N_x que corresponde ao número de pontos escolhidos na direção x ;

N_y que corresponde ao número de pontos escolhidos na direção y ;

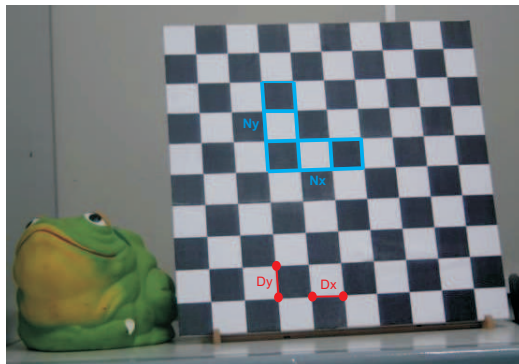


Figura 3.2: Dados do padrão Xadrez.

As coordenadas 3D de cada ponto (x_{ij}, y_{ij}, z_{ij}) de cada retângulo ij são dadas por:

$$x_{ij} = i \cdot D_x, \quad y_{ij} = j \cdot D_y \quad \text{e} \quad z_{ij} = 0. \quad (3.1)$$

Exemplo 2: Padrão com Círculos

Trata-se de outro padrão coplanar simples e fácil de ser implementado. Ele é constituído por vários círculos distribuídos de forma que em cada linha exista apenas um círculo. Esta disposição facilita a forma de identificar cada círculo no método de correspondência.

No sistema os parâmetros podem ser definidos como:

Tipo cujo valor será fixado como coplanar;

D_y que corresponde a distância vertical entre os pontos;

N número de círculos do padrão;

V_x que corresponde a um vetor de números correspondentes a primeira coordenada dos pontos.

A figura 3.3 ilustra este padrão, e a figura 3.4 mostra os parâmetros especificados acima.

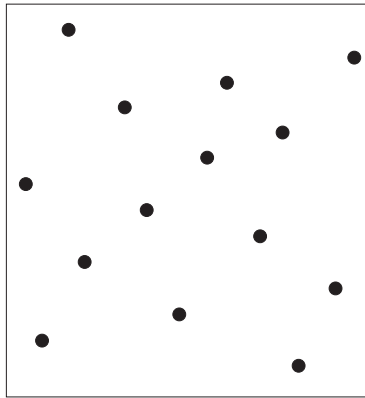


Figura 3.3: Padrão de círculos.

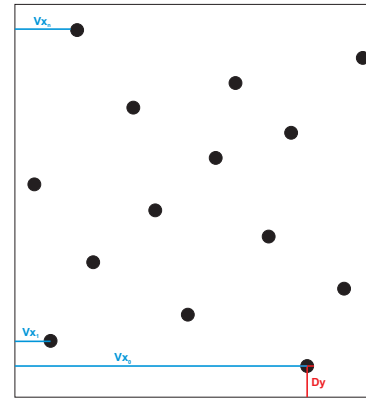


Figura 3.4: Dados do padrão de círculos.

3.3 Imagens

Alguns métodos de calibração de câmera utilizam uma ou múltiplas imagens de um padrão escolhido, tiradas de uma ou várias câmeras. A partir destas imagens são efetuados cálculos para então obter as informações desejadas para o processo de calibração. Na calibração de câmera, estas informações correspondem às projeções de pontos tridimensionais do padrão.

A quantidade de imagens utilizadas para a calibração dependerá do próprio método a ser executado. No método de Tsai (Tsai 1987), por exemplo, apenas uma imagem é necessária, e a correspondência entre os pontos pode ser realizada utilizando-se alguns pontos selecionados na imagem e seus respectivos pontos tridimensionais situados no padrão. Já o método de Zhang (Zhang 1998) utiliza várias imagens do padrão em posições diferentes. Neste último caso, faz-se a correspondência em todas as imagens utilizadas, e consideram-se os mesmos parâmetros intrínsecos para cada imagem.

Podemos ainda utilizar uma mesma imagem para calibrar uma câmera e um projetor. Definimos uma classe para o elemento imagem que visa representar uma ou várias imagens a serem usadas no processo de calibração.

Os dados necessários para especificá-la são:

- Lista de imagens;
- Quantidade de imagens.

As operações válidas para esta classe são:

- Atribuir/Recuperar as imagens: consiste em definir e obter quais as imagens representadas.
- Obter a quantidade de imagens: informa a quantidade de imagens armazenadas.

3.4 Correspondência

Os métodos de correspondência determinam a relação existente entre os pontos do padrão e suas respectivas projeções nas imagens e por isso eles são completamente dependentes do padrão. A partir desta relação é possível determinar pares de pontos correspondentes entre o padrão e a imagem, os quais serão usados no processo de calibração.

A especificação dos pontos pode ser:

- Manual: o usuário deverá escolher os pontos que serão usados pela calibração;
- Automática: método de correspondência deverá identificar automaticamente os pontos, por exemplo, usando técnicas de processamento de imagens;
- Semi-Automática: o usuário seleciona apenas alguns pontos e o método faz uma interpolação para obter os demais pontos.

A incorporação de novos métodos de correspondência ao sistema é realizada de forma dinâmica, ou seja, o método é implementado como uma biblioteca dinâmica e pode ser incorporado ao sistema através do módulo EditCalib e executado no módulo ExecCalib.

Esta solução foi adotada para a proteção do sistema e por ser uma forma mais simples para o usuário adicionar seu algoritmo no sistema. Os dados necessários para definir a correspondência são:

- Nome do método;
- Tipo do método, se é manual, automático ou semi-automático;
- Diretório da dll, lugar aonde está disponível a biblioteca dinâmica contendo a implementação do método;
- Quantidade de pontos selecionados na imagem (e conseqüentemente sobre o padrão);
- Lista de pontos 2D selecionados sobre a imagem;
- Lista de pontos 3D selecionados sobre o padrão;
- Lista de Imagens envolvidas;

As operações definidas para a correspondência são:

- Atribuir/Ler o nome do método: define/recupera o nome do método;
- Definir/Recuperar o tipo do método: especifica/obtem o tipo do método (manual, automático ou semi-automático);

- Definir/Obter a imagem: informa/obtem a lista de imagens usadas na correspondência;
- Definir/Ler o diretório da dll;
- Definir/Recuperar a quantidade de pontos selecionados e obtidos;
- Atribuir/Obter os pontos selecionados: armazena/recupera os pontos selecionados sobre a imagem;
- Atribuir/Ler os pontos 2D;
- Extrair/Ler os pontos 3D;
- Executar o método.

A seguir descreveremos alguns métodos de correspondência utilizados para os padrões descritos na seção 3.2.

3.4.1 Correspondência para padrão Xadrez

Um método de correspondência semi-automático para o padrão xadrez foi implementado para o sistema. Como dados de entrada, o método recebe quatro pontos selecionados, correspondentes aos vértices de um retângulo sobre a imagem do padrão.

Para uma melhor precisão dos resultados, é realizado um refinamento sobre os pontos selecionados para obter as coordenadas mais aproximadas das reais dos vértices do retângulo. A figura 3.5 mostra os pontos selecionados.

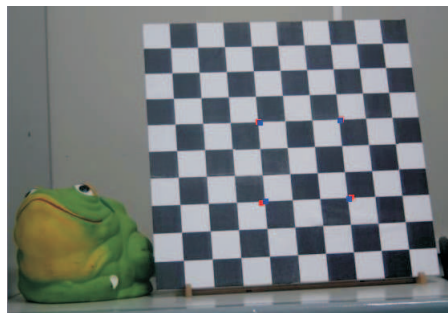


Figura 3.5: Resultado da precisão subpixel.

Com os pontos fornecidos podemos encontrar a grade de pontos que serão utilizados na calibração, encontrando as interseções das retas como segue abaixo:

1. Interseções da reta c_1 e c_2 , para obter o centro O ;
2. Interseções da reta r_1 e r_2 , para obter o ponto O_1 ;
3. Interseções da reta r_3 e r_4 , para obter o ponto O_2 ;
4. Interseções da reta $\overline{O_1O}$ e a reta r_4 , para obter o centro a ;
5. Interseções da reta $\overline{O_1O}$ e a reta r_3 , para obter o centro c ;
6. Interseções da reta $\overline{O_2O}$ e a reta r_1 , para obter o centro b ;
7. Interseções da reta $\overline{O_2O}$ e a reta r_2 , para obter o centro d ;

As figuras 3.6 e 3.7 ilustram os passos acima.

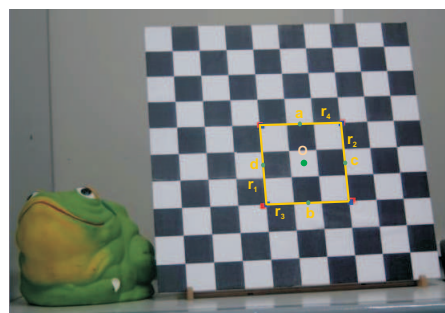
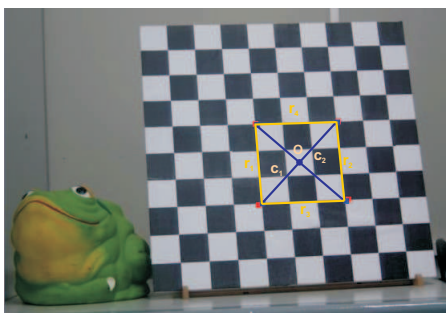


Figura 3.6: Determinação dos centros. Figura 3.7: Determinação dos pontos médios.

A partir destes novos pontos, podemos encontrar os pontos presentes em cada lado do retângulo, através das seguintes etapas:

1. Determinação dos pontos do xadrez contidos na reta r_4 , utilizando os pontos extremos do segmento de reta r_4 e a ;
2. Determinação dos pontos do xadrez contidos na reta r_2 , utilizando os pontos extremos do segmento de reta r_2 e c ;
3. Determinação dos pontos do xadrez contidos na reta r_3 , utilizando os pontos extremos do segmento de reta r_3 e b ;
4. Determinação dos pontos do xadrez contidos na reta r_1 , utilizando os pontos extremos do segmento de reta r_1 e d .

A figura 3.8 mostra os pontos de cada lado do retângulo obtidos pelas etapas dada acima. E por último, podemos calcular novas interseções com as retas formadas pelos últimos pontos obtidos afim de encontrar todos os pontos da grade que ainda restam. A figura 3.9 mostra a grade obtida pelo algoritmo dado.

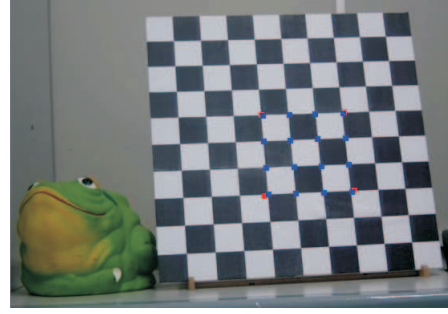
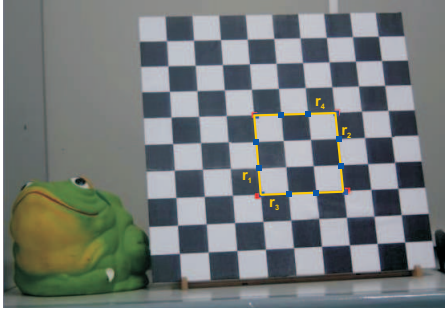


Figura 3.8: Determinação dos pontos. Figura 3.9: Determinação da grade de pontos.

3.4.2 Correspondência para padrão com Círculos

Para obtermos os pontos deste padrão precisamos primeiro identificar os círculos na imagem. Ou seja, devemos distinguir os círculos do fundo da imagem. Para resolver este problema utiliza-se um filtro “threshold” definido por:

$$value(x, y) = \begin{cases} 0, & \text{se } p(x, y) \geq t; \\ 1, & \text{caso contrário,} \end{cases}$$

onde $value(x, y)$ será o novo valor do pixel após o filtro, $p(x, y)$ é o valor original do pixel da linha x e coluna y , e t corresponde ao valor do threshold escolhido. Após isso, devemos identificar as componentes conexas da imagem agora binária. Uma componente conexa para este caso representa o conjunto de pixels conectados um ao outro, respeitando uma conectividade escolhida em relação a cor e adjacência do pixel.

No que diz respeito à cor, os vértices das componentes conexas devem possuir a mesma intensidade de cor, ou seja, os pixels vizinhos devem assumir o mesmo valor. Uma 4-vizinhança de um pixel $p(x, y)$ é dada pelo conjunto V_4 abaixo:

$$V_4 = \{(x + 1, y), (x - 1, y), (x, y + 1), (x, y - 1)\},$$

e uma 8-vizinhança definida por:

$$V_8 = V_4 \cup \{(x + 1, y + 1), (x - 1, y + 1), (x - 1, y - 1), (x + 1, y - 1)\}.$$

Portanto, um ponto q é 8-vizinho de p se $q \in V_8$. Desta forma, obtemos as componentes conexas em relação à adjacência, identificando as conectividades com 8-vizinhos. A figura 3.10 ilustra a imagem do padrão após o threshold e a figura 3.11 mostra as componentes conexas como definimos acima.

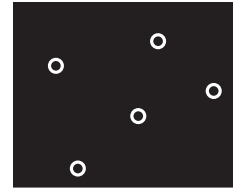
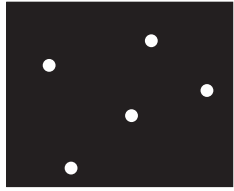


Figura 3.10: Padrão após threshold. Figura 3.11: Borda das componentes conexas.

Encontrados as componentes conexas, o último passo do método se resume a encontrar os centros dos mesmos, que podem ser obtidos determinando a posição média dos pontos da borda de cada componente conexa encontrada, e assim teremos os círculos novamente. A figura 3.12 ilustra estes passos.

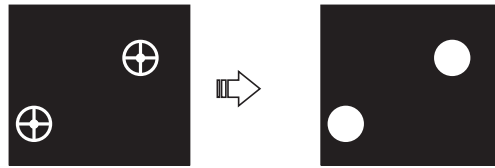


Figura 3.12: Determinação dos centros e recuperação dos círculos.

Capítulo 4

Métodos de Calibração

4.1 Introdução

Este capítulo apresenta a estrutura geral da etapa de calibração e como ela pode ser incorporada à arquitetura proposta na dissertação. Discutiremos também alguns métodos já existentes na literatura. O estudo e análise destes métodos serviram de base para as especificações de um método geral dentro da arquitetura proposta.

A calibração de câmera consiste em estimar os parâmetros intrínsecos e extrínsecos da câmera, através de transformações entre o espaço da câmera e o espaço da imagem. Como já foi visto estas transformações não podem ser perfeitamente modeladas por transformações perspectivas, devido às distorções das lentes.

Para simplificar o problema de calibração, várias considerações podem ser feitas, como por exemplo:

- Ignorar a distorção das lentes;
- Considerar que os pixels são quadrados, ou seja $s_x = s_y$ e $f_x = f_y$;
- Assumir apenas pontos coplanares;
- Considerar que a projeção do centro óptico coincide com o centro da imagem.

Desta forma, vários métodos de calibração podem ser propostos. Em geral, eles consistem em resolver um problema de otimização não linear, quando se consideram as distorções das lentes, e utilizam pontos tridimensionais de um padrão e suas respectivas projeções em imagens.

Para modelar a calibração na arquitetura proposta foi definida uma classe abstrata que contenha as informações genéricas de qualquer método. Cada novo método é modelado como uma subclasse que contenha suas informações específicas. A figura 4.1 mostra uma hierarquia de classes com alguns métodos de calibração.

As informações gerais consideradas em um método de calibração foram:

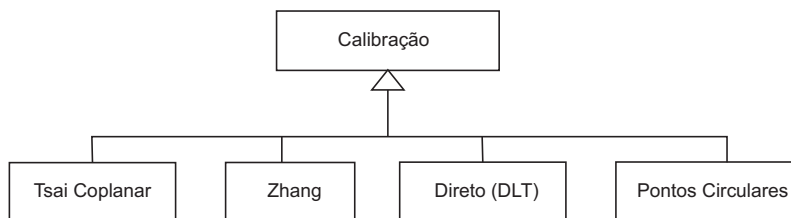


Figura 4.1: Subclasses para a classe de Calibração.

- Nome do método;
- Diretório da dll, onde se encontra a biblioteca dinâmica com a implementação do método;
- Lista de pontos 2D originados da correspondência;
- Lista de pontos 3D originados da correspondência;
- Quantidade de pontos;
- Matriz de rotação R;
- Matriz intrínseca K;
- Vetor de Translação;

As principais operações envolvidas na calibração são:

- Atribuir/Ler nome do método;
- Definir/Ler o diretório da dll;
- Atribuir/Recuperar a quantidade de pontos;
- Atribuir/Ler os pontos 3D;
- Atribuir/Ler os pontos 2D;
- Atribuir/Ler os parâmetros intrínsecos: atribui/obtem os parâmetros intrínsecos(matriz intrínseca);
- Atribuir/Ler os parâmetros extrínsecos: atribui/obtem os parâmetros extrínsecos(matriz de rotação, vetor de translação);
- Executar o método;

As próximas seções discutem alguns métodos de calibração de câmera. Suas diferenças foram fundamentais para a elaboração da arquitetura proposta por este trabalho.

4.2 Método Tsai Coplanar

O algoritmo proposto por Tsai (Tsai 1987), (Horn 2000), (Tapper et al. 2002) consiste em duas etapas. A primeira etapa calcula a matriz de rotação e o vetor translação, com exceção da sua coordenada Z . A segunda etapa é responsável pelo cálculo dos parâmetros intrínsecos e da coordenada Z do vetor de translação.

Algumas hipóteses são necessárias para a sua formulação coplanar:

1. Requer um padrão de calibração;
2. SCM e SCC possuem a mesma orientação;
3. Os pontos de calibração estão no plano $Z = 0$, na versão coplanar;
4. São conhecidos os valores intrínsecos da câmera, exceto a distância focal e a distorção angular κ_1 ;
5. Caso não sejam conhecidos os valores intrínsecos, basta considerar conhecido a projeção do centro óptico (C_x, C_y) e supor que a tangente do ângulo que as colunas da matriz de pixels formam com a perpendicular com as linhas, $\tau = 0$, e que o fator de escala $s_x = s_y = 1$;
6. A origem do sistema do mundo é escolhida de forma a não se projetar próximo ao centro da imagem ou ao eixo X da imagem.

Suponha que tenhamos um padrão coplanar, a sua imagem capturada pela câmera e n pontos coplanares $P_1, P_2 \dots P_n$ sobre o padrão, cujas coordenadas (x_{wi}, y_{wi}, z_{wi}) sejam conhecidas.

Sejam (X_{fi}, Y_{fi}) as coordenadas dos pontos na imagem em pixels. Os pontos $P_{di} = (X_{di}, Y_{di})$, que correspondem aos pontos considerando a distorção, podem ser facilmente obtidos através das relações:

$$X_{di} = X_{fi} - C_x \quad Y_{di} = Y_{fi} - C_y,$$

onde (C_x, C_y) corresponde a projeção do centro óptico. Agora, as coordenadas dos pontos $P_u = (X_u, Y_u)$ (coordenadas dos pontos desconsiderando a distorção) podem ser definidas através da expressão abaixo:

$$X_{ui} = D_{xi} + X_{di} \quad Y_{ui} = D_{yi} + Y_{di}, \quad (4.1)$$

onde,

$$D_{xi} = X_{di} (\kappa_1 r_i^2) \quad D_{yi} = Y_{di} (\kappa_1 r_i^2),$$

com $r_i = \sqrt{X_{di}^2 + Y_{di}^2}$. A partir da projeção perspectiva definida na seção 1.2.2, temos que:

$$\begin{bmatrix} X_{ui} \\ Y_{ui} \\ 1 \end{bmatrix} \simeq \begin{bmatrix} fr_1 & fr_2 & fr_3 & fT_x \\ fr_4 & fr_5 & fr_6 & fT_x \\ r_7 & r_8 & r_9 & T_z \end{bmatrix} \begin{bmatrix} x_{wi} \\ y_{wi} \\ z_{wi} \\ 1 \end{bmatrix}, \quad (4.2)$$

onde f é a distância focal. E assim temos as equações:

$$X_{ui} = f \frac{r_1 x_{wi} + r_2 y_{wi} + r_3 z_{wi} + T_x}{r_7 x_{wi} + r_8 y_{wi} + r_9 z_{wi} + T_z}, \quad (4.3)$$

$$Y_{ui} = f \frac{r_4 x_{wi} + r_5 y_{wi} + r_6 z_{wi} + T_y}{r_7 x_{wi} + r_8 y_{wi} + r_9 z_{wi} + T_z}. \quad (4.4)$$

Porém, como estamos tomando pontos coplanares, ou seja $z_{wi} = 0$, podemos simplificar tais equações para obtermos:

$$X_{ui} = f \frac{r_1 x_{wi} + r_2 y_{wi} + T_x}{r_7 x_{wi} + r_8 y_{wi} + T_z}, \quad (4.5)$$

$$Y_{ui} = f \frac{r_4 x_{wi} + r_5 y_{wi} + T_y}{r_7 x_{wi} + r_8 y_{wi} + T_z}. \quad (4.6)$$

Ora,

$$\frac{(1 + \kappa_1 r_i^2)}{(1 + \kappa_1 r_i^2)} = 1 = \frac{X_{ui}}{Y_{ui}} = \frac{X_{di}}{Y_{di}} = \frac{r_1 x_{wi} + r_2 y_{wi} + T_x}{r_4 x_{wi} + r_5 y_{wi} + T_y},$$

e daí,

$$X_{di} T_y = Y_{di} r_1 x_{wi} + Y_{di} r_2 y_{wi} + Y_{di} T_x - X_{di} r_4 x_{wi} - X_{di} r_5 y_{wi} \Rightarrow$$

$$X_{di} = Y_{di} x_{wi} \frac{r_1}{T_y} + Y_{di} y_{wi} \frac{r_2}{T_y} + Y_{di} \frac{T_x}{T_y} - X_{di} x_{wi} \frac{r_4}{T_y} - X_{di} y_{wi} \frac{r_5}{T_y}. \quad (4.7)$$

Faça $U_1 = \frac{r_1}{T_y}$, $U_2 = \frac{r_2}{T_y}$, $U_3 = \frac{T_x}{T_y}$, $U_4 = \frac{r_4}{T_y}$, $U_5 = \frac{r_5}{T_y}$, então a equação acima pode ser reescrita como:

$$X_{di} = Y_{di} x_{wi} U_1 + Y_{di} y_{wi} U_2 + Y_{di} U_3 - X_{di} x_{wi} U_4 - X_{di} y_{wi} U_5, \quad (4.8)$$

ou

$$\begin{bmatrix} Y_{di}x_{wi} & Y_{di}y_{wi} & Y_{di} & -X_{di}x_{wi} & -X_{di}y_{wi} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \end{bmatrix} = X_{di}. \quad (4.9)$$

Com $n > 5$, o sistema pode ser resolvido como um problema de mínimos quadrados, isto é, devemos encontrar $\begin{bmatrix} U_1 & U_2 & U_3 & U_4 & U_5 \end{bmatrix}$ tal que $\|AU - b\|$ seja mínimo. Após determinarmos os elementos de U , podemos prosseguir a calibração a partir das seguintes etapas:

Primeiro Estágio Cálculo da matriz de rotação R e do vetor de translação $T = (T_x, T_y)$.

1- Cálculo de $|T_y|$.

Considere a matriz de rotação R .

$$R = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix}.$$

Supomos nas hipóteses que suas linhas são unitárias e ortogonais, então temos que:

$$r_1^2 + r_2^2 + r_3^2 = 1,$$

$$r_4^2 + r_5^2 + r_6^2 = 1,$$

$$r_1r_4 + r_2r_5 + r_3r_6 = 0.$$

Mas, $U_1 = \frac{r_1}{T_y}$, $U_2 = \frac{r_2}{T_y}$, $U_3 = \frac{r_3}{T_y}$, $U_4 = \frac{r_4}{T_y}$, $U_5 = \frac{r_5}{T_y}$ então:

$$(U_1^2 + U_2^2) T_y^2 + r_3^2 = 1,$$

$$(U_4^2 + U_5^2) T_y^2 + r_6^2 = 1,$$

$$(U_1U_4 + U_2U_5) T_y^2 + r_3r_6 = 0.$$

E daí, eliminando r_3 e r_6 , obtemos:

$$r_3^2r_6^2 = 1 - (U_1^2 + U_2^2 + U_3^2 + U_4^2 + U_5^2) T_y^2 + (U_1^1U_4^2 + U_1^2U_5^2 + U_2^2U_4^2 + U_2^2U_5^2) T_y^4$$

$$1 - (U_1^2 + U_2^2 + U_3^2 + U_4^2 + U_5^2) T_y^2 + (U_1U_5 - U_2U_4)^2 T_y^4 = 0.$$

Fazendo $S_r = (U_1^2 + U_2^2 + U_3^2 + U_4^2 + U_5^2)$ e $D = (U_1U_5 - U_2U_4)$, temos:

$$1 - S_r T_y^2 + DT_y^4 = 0.$$

Se $D \neq 0$, então

$$T_y^2 = \frac{S_r \pm \sqrt{S_r^2 - 4D}}{2D}. \quad (4.10)$$

Se uma das linhas ou colunas da matriz

$$M = \begin{bmatrix} U_1 & U_2 \\ U_4 & U_5 \end{bmatrix} \quad (4.11)$$

se anula, então

$$T_y^2 = \frac{1}{U_i^2 + U_j^2}, \quad (4.12)$$

onde U_i^2 e U_j^2 são os elementos da linha ou coluna de M que não zeram.

2- *Determinação do sinal de T_y .*

Suponha inicialmente que $T_y > 0$. Então, podemos determinar os valores de r_1 , r_2 , T_x , r_4 e r_5 , com as expressões abaixo:

$$r_1 = U_1 T_y,$$

$$r_2 = U_2 T_y,$$

$$r_3 = U_3 T_y$$

$$r_4 = U_4 T_y$$

$$r_5 = U_5 T_y.$$

Escolhendo um ponto $P = (X_{fi}, Y_{fi})$ suficientemente distante do centro da imagem, com coordenadas no espaço do objeto (x_{wi}, y_{wi}, z_{wi}) , verificamos se o sinal de $r_1 x_{wi} + r_2 y_{wi} + T_x$ é igual ao de X_{fi} . Em caso contrário, trocamos o sinal de T_y e de todos os elementos calculados que dependem de seu valor.

3- *Cálculo dos demais elementos de rotação.*

Como cada linha da matriz de rotação R é unitária, temos:

$$r_3 = \pm \sqrt{1 - r_1^2 - r_2^2},$$

$$r_6 = \pm \sqrt{1 - r_4^2 - r_5^2}.$$

Suponha novamente que $r_3 > 0$. Se $r_1r_4 + r_2r_5 > 0$, então $r_6 < 0$, pois por hipótese a matriz de rotação é ortogonal. Caso contrário, r_6 é positivo. Agora, para encontrarmos a última linha, utilizamos novamente a condição de ortogonalidade da matriz.

$$r_7 = r_2r_6 - r_3r_5, \quad (4.13)$$

$$r_8 = r_3r_4 - r_1r_6, \quad (4.14)$$

$$r_9 = r_1r_5 - r_2r_4. \quad (4.15)$$

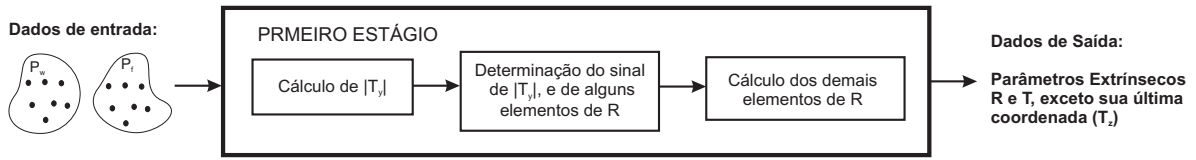


Figura 4.2: Primeiro Estágio do Método Tsai.

Segundo Estágio Cálculo de f , κ_1 e de T_z .

1- *Cálculo aproximado de f e T_z ignorando a distorção radial.*

Considere $\kappa_1 = 0$. Então, pela expressão 4.1 temos que $P_{ui} = P_{di}$, e assim utilizando as equações 4.5 e 4.6 obtemos para cada ponto de calibração $P_{wi} = (x_{wi}, y_{wi}, z_{wi})$ com suas respectivas projeções (X_{di}, Y_{di}) , as seguintes expressões:

$$(r_1x_{wi} + r_1y_{wi} + T_x)f - X_{di}T_z = X_{di}(r_7x_{wi} + r_8y_{wi}) \quad (4.16)$$

$$(r_4x_{wi} + r_5y_{wi} + T_y)f - Y_{di}T_z = Y_{di}(r_7x_{wi} + r_8y_{wi}). \quad (4.17)$$

Com n pontos de calibração, teremos $2n$ equações para as incógnitas f e T_z o que pode ser solucionado por mínimos quadrados. Caso r_3 seja realmente positivo, a distância focal f terá valor positivo, caso contrário, os sinais dos elementos r_3 , r_6 , r_7 e r_8 deverão ser trocados.

2- *Cálculo exato de f , T_z e κ_1 .*

Com a inclusão da distorção radial, as equações em 4.17 se tornam equações não lineares, que podem ser resolvidas por algum método de otimização, por exemplo o de

Levenberg-Marquardt, usando os dados f , T_z já calculados e $\kappa_1 = 0$ como soluções iniciais.

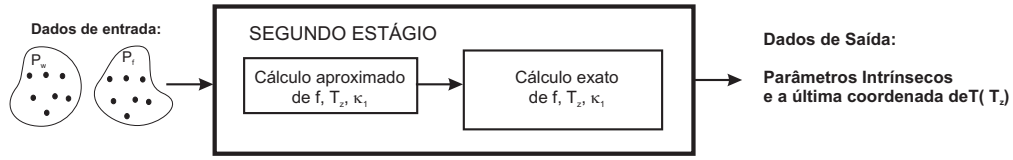


Figura 4.3: Segundo Estágio do Método Tsai.

4.3 Método Zhang

Este método (Zhang 1998) requer que a câmera visualize um padrão coplanar em várias posições diferentes, obtendo assim m imagens do padrão. Entretanto, admitiremos que os parâmetros intrínsecos são os mesmos para cada imagem. Assumiremos que os pontos são coplanares e daí a relação entre os pontos 3D e suas respectivas projeções em coordenadas homogêneas pode ser definida por:

$$s \begin{bmatrix} X_{fi} \\ Y_{fi} \\ 1 \end{bmatrix} \simeq K \begin{bmatrix} r_1 & r_2 & r_3 & T \end{bmatrix} \begin{bmatrix} x_{wi} \\ y_{wi} \\ 0 \\ 1 \end{bmatrix} = K \begin{bmatrix} r_1 & r_2 & T \end{bmatrix} \begin{bmatrix} x_{wi} \\ y_{wi} \\ 1 \end{bmatrix}, \quad (4.18)$$

onde s é um fator de escala e K é a matriz intrínseca da câmera dada por:

$$K = \begin{bmatrix} \alpha & c & C_x \\ 0 & \beta & C_y \\ 0 & 0 & 1 \end{bmatrix},$$

onde

α e β correspondem ao número de pixels por unidade de comprimento em ambas direções, horizontal e vertical, respectivamente;

c é dada pela tangente do ângulo que as colunas da matriz de pixels formam com a perpendicular com as linhas;

C_x e C_y representam as coordenadas da projeção ortogonal do centro óptico no plano de projeção.

Chamamos a essa transformação linear que associa as coordenadas do ponto do mundo às suas projeções na imagem de *homografia*, denotada por H na expressão abaixo:

$$s \begin{bmatrix} X_{fi} \\ Y_{fi} \\ 1 \end{bmatrix} \simeq H \begin{bmatrix} x_{wi} \\ y_{wi} \\ 1 \end{bmatrix}, \quad (4.19)$$

ou seja, $H = \lambda K \begin{bmatrix} r_1 & r_2 & T \end{bmatrix}$.

Podemos então descrever o método de Zhang em quatro etapas: estimação das homografias para cada imagem, estimação dos parâmetros intrínsecos e extrínsecos, estimação dos coeficientes de distorção e refinamentos destes parâmetros.

Primeira etapa: Estimação da Homografia para cada Imagem.

Suponha que em cada imagem tomemos n pontos de calibração. Então, teremos :

$$H(P_{wi}) - P_{fi} = 0. \quad (4.20)$$

Este problema pode ser resolvido por um método de otimização não-linear, como Levenberg-Marquardt, determinando a homografia H , com $\|H\| = 1$ que minimiza

$$\sum_i^n \|P_{fi} - H(P_{wi})\|^2.$$

Este método requer uma solução inicial, a qual pode ser obtida como segue. Denote as linhas da homografia que associa os pontos do padrão às suas respectivas projeções, por \bar{h}_1^T , \bar{h}_2^T , e \bar{h}_3^T , então

$$\bar{h}_1^T P_{wi} - X_{fi} \bar{h}_3^T = 0, \quad (4.21)$$

$$\bar{h}_2^T P_{wi} - Y_{fi} \bar{h}_3^T = 0. \quad (4.22)$$

As equações acima podem ser reescritas da forma:

$$Ah = 0, \quad (4.23)$$

onde h é o vetor 9×1 composto pelos elementos da homografia h . Minimizando 4.23, tal que $\|h\| = 1$, obtemos uma boa solução inicial para o método iterativo de Levenberg-Marquardt.

Segunda etapa: Estimação dos parâmetros da câmera.

Usando a hipótese de que as linhas da matriz de rotação são ortogonais e unitárias temos que:

$$\begin{bmatrix} r_1 \\ r_2 \\ T \end{bmatrix} \begin{bmatrix} r_1 & r_2 & T \end{bmatrix} = \begin{bmatrix} r_1^T r_1 & r_1^T r_2 & r_1^T T \\ r_2^T r_1 & r_2^T r_2 & r_2^T T \\ T^T r_1 & T^T r_2 & T^T T \end{bmatrix} = \begin{bmatrix} 1 & 0 & r_1^T T \\ 0 & 1 & r_2^T T \\ T^T r_1 & T^T r_2 & T^T T \end{bmatrix}. \quad (4.24)$$

Como,

$$H = \lambda K \begin{bmatrix} r_1 & r_2 & T \end{bmatrix}$$

Temos que,

$$\begin{bmatrix} r_1 & r_2 & T \end{bmatrix} = s K^{-1} H \Rightarrow \begin{bmatrix} r_1 & r_2 & T \end{bmatrix}^T = s H^T K^{-T}.$$

Portanto,

$$\begin{bmatrix} r_1 \\ r_2 \\ T \end{bmatrix} \begin{bmatrix} r_1 & r_2 & T \end{bmatrix} = s^2 H^T K^{-T} K^{-1} H, \quad (4.25)$$

e daí

$$h_1^T K^{-T} K^{-1} h_2 = 0, \quad (4.26)$$

$$h_1^T K^{-T} K^{-1} h_1 = h_2^T K^{-T} K^{-1} h_2. \quad (4.27)$$

Denote por B, a matriz

$$B = K^{-T} K^{-1} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix}$$

$$B = \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{c}{\alpha^2 \beta} & \frac{c C_y - C_x \beta}{\alpha^2 \beta} \\ -\frac{c}{\alpha^2 \beta} & \frac{c^2}{\alpha^2 \beta^2} + \frac{1}{\beta^2} & -\frac{c(c C_y - C_x \beta)}{\alpha^2 \beta^2} - \frac{C_y}{\beta^2} \\ \frac{c C_y - C_x \beta}{\alpha^2 \beta} & -\frac{c(c C_y - C_x \beta)}{\alpha^2 \beta^2} - \frac{C_y}{\beta^2} & \frac{(c C_y - C_x \beta)^2}{\alpha^2 \beta^2} + \frac{C_y^2}{\beta^2} + 1 \end{bmatrix}. \quad (4.28)$$

Note que B é simétrica e definida pelo vetor $b = \begin{bmatrix} B_{11} & B_{12} & B_{22} & B_{13} & B_{23} & B_{33} \end{bmatrix}$. Denotando $v_{ij} = \begin{bmatrix} h_{1i} h_{1j} & h_{1i} h_{2j} + h_{2i} h_{1j} & h_{2i} h_{2j} & h_{3i} h_{1j} + h_{1i} h_{3j} & h_{3i} h_{2j} + h_{2i} h_{3j} & h_{3i} h_{3j} \end{bmatrix}$, temos $h_i^T B h_j = v_{ij} b$. Portanto, as equações 4.27 podem ser reescritas por:

$$\begin{bmatrix} v_{12} \\ (v_{11} - v_{22}) \end{bmatrix} b = 0. \quad (4.29)$$

Para m imagens teremos $2m$ equações do tipo $Vb = 0$, onde V é uma matriz $2m \times 6$. Este sistema pode ser resolvido de forma análoga a da estimação da homografia. Ou seja, escolher $\|b\| = 1$ tal que $\|Vb\|$ seja mínimo. Assim, podemos obter os parâmetros da câmera através das seguintes expressões:

Parâmetros Intrínsecos

$$C_y = \frac{(B_{12}B_{13} - B_{11}B_{23})}{(B_{11}B_{22} - B_{12}^2)}$$

$$s = B_{33} - \frac{[B_{13}^2 + C_y(B_{12}B_{13} - B_{11}B_{23})]}{B_{11}}$$

$$\alpha = \sqrt{\frac{s}{B_{11}}}$$

$$\beta = \sqrt{\frac{sB_{11}}{(B_{11}B_{22} - B_{12}^2)}}$$

$$c = -\frac{B_{12}\alpha^2\beta}{s}$$

$$C_x = \frac{cC_y}{\alpha} - \frac{B_{13}\alpha^2}{s}$$

Parâmetros Extrínsecos

$$r_1 = sK^1h_1$$

$$r_2 = sK^1h_2$$

$$r_3 = r_1 \times r_2$$

$$T = sA^{-1}h_3.$$

Terceira etapa: Estimação dos coeficientes de distorção.

Diferente do Tsai, Zhang considera os coeficientes k_1 e k_2 de distorção radial. Assim, a relação entre o ponto cujas coordenadas não consideram as distorções da lente (P_u) e o ponto cujas coordenadas consideram as distorções (P_d) é dada por

$$X_u = X_d + X_d(\kappa_1 r^2 + \kappa_2 r^4), \quad (4.30)$$

$$Y_u = Y_d + Y_d(\kappa_1 r^2 + \kappa_2 r^4). \quad (4.31)$$

onde $r = \sqrt{X_d^2 + Y_d^2}$. As coordenadas de P_u por sua vez se relacionam com o ponto correspondente no sistema de coordenadas em pixels (P_f), pelas equações:

$$X_f = \alpha X_u + c Y_u + C_x,$$

$$Y_f = \beta Y_u + C_y,$$

e assim podemos relacionar os pontos observados e os pontos sem distorção através das equações:

$$X_f = X'_f + (X'_f - C_x) (\kappa_1 r^2 + \kappa_2 r^4), \quad (4.32)$$

$$Y_f = Y'_f + (Y'_f - C_y) (\kappa_1 r^2 + \kappa_2 r^4). \quad (4.33)$$

ou reescrevendo a mesma em forma matricial

$$\begin{bmatrix} (X'_f - C_x) r^2 & (X'_f - C_x) r^4 \\ (Y'_f - C_y) r^2 & (Y'_f - C_y) r^4 \end{bmatrix} \begin{bmatrix} \kappa_1 \\ \kappa_2 \end{bmatrix} = \begin{bmatrix} X_f - X'_f \\ Y_f - Y'_f \end{bmatrix}. \quad (4.34)$$

Dados n pontos em m imagens, teremos $2mn$ equações da forma

$$DK = d,$$

com $K = \begin{bmatrix} k_1 & k_2 \end{bmatrix}^T$, e assim a solução obtida por mínimos quadrados é dada por:

$$k = (D^T D)^{-1} D^T d.$$

Quarta etapa: Refinamento dos parâmetros.

Podemos agora calcular os valores exatos dos parâmetros através de uma otimização, por exemplo Levenberg-Marquardt, utilizando os parâmetros estimados como solução inicial. Então, podemos minimizar

$$\sum_{i=1}^n \sum_{j=1}^m \|P_{f_{ij}} - \hat{p}(K, \kappa_1, \kappa_2, R_i, T_i, P_{w_j})\|^2,$$

onde $P_{f_{ij}}$ é a projeção do ponto P_{w_j} na imagem i e $(K, \kappa_1, \kappa_2, R_i, T_i, P_{w_j})$ é a projeção deste ponto pela equação 4.18 e com a distorção de acordo com 4.33.

A figura 4.4 ilustra os passos do método de Zhang.

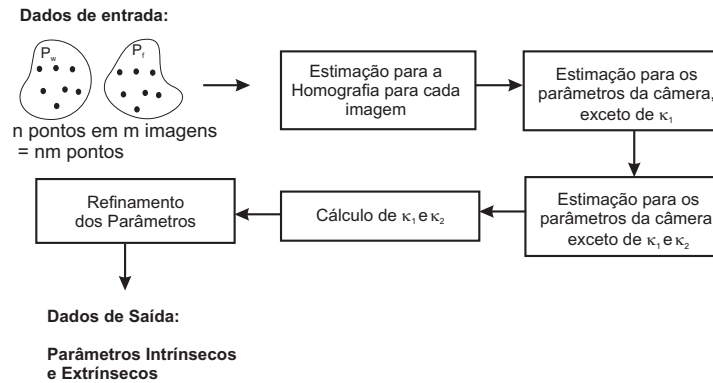


Figura 4.4: Etapas de Zhang.

4.4 Método Direct Linear Transformation (DLT)

No método direto (Abdel-Aziz & Karara n.d.) supomos que o sistema de coordenadas do mundo é conhecido, e ignoramos a distorção radial da lente. Assim, tomemos um ponto P_w no padrão e sua respectiva projeção P_f na imagem. Então,

$$X_f = -\frac{X_u}{s_x} + C_x,$$

$$Y_f = -\frac{Y_u}{s_y} + C_y,$$

e daí

$$X_f - C_x = -\frac{f}{s_x} \frac{r_1 X_w + r_2 Y_w + r_3 Z_w + T_x}{r_7 X_w + r_8 Y_w + r_9 Z_w + T_z},$$

$$Y_f - C_y = -\frac{f}{s_y} \frac{r_4 X_w + r_5 Y_w + r_6 Z_w + T_y}{r_7 X_w + r_8 Y_w + r_9 Z_w + T_z}.$$

Denote $f_x = \frac{f}{s_x}$, $f_y = \frac{f}{s_y}$, $\alpha = \frac{s_y}{s_x}$, $x = X_f - C_x$, e $y = Y_f - C_y$, e assim teremos:

$$x = -f_x \frac{r_1 X_w + r_2 Y_w + r_3 Z_w + T_x}{r_7 X_w + r_8 Y_w + r_9 Z_w + T_z},$$

$$y = -f_y \frac{r_4 X_w + r_5 Y_w + r_6 Z_w + T_y}{r_7 X_w + r_8 Y_w + r_9 Z_w + T_z},$$

$$\frac{x}{y} = \frac{f_x}{f_y} \frac{r_1 X_w + r_2 Y_w + r_3 Z_w + T_x}{r_4 X_w + r_5 Y_w + r_6 Z_w + T_y},$$

$$x f_y (r_4 X_w + r_5 Y_w + r_6 Z_w + T_y) = y f_x (r_1 X_w + r_2 Y_w + r_3 Z_w + T_x). \quad (4.35)$$

Agora podemos dividir o método em duas etapas:

1. Assumindo, C_x e C_y conhecidos, obter os parâmetros,

2. Determinar C_x e C_y .

Etapa 1. Assumindo, C_x e C_y conhecidos, obter os parâmetros.

Faça $v = \begin{bmatrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 \end{bmatrix}$, onde

$$\begin{aligned} v_1 &= r_4, & v_2 &= r_5, \\ v_3 &= r_6, & v_4 &= T_x, \\ v_5 &= r_1\alpha, & v_6 &= r_2\alpha, \\ v_7 &= r_3\alpha, & v_8 &= T_x, \end{aligned}$$

e daí a expressão 4.35 pode ser escrita na seguinte forma matricial para os n pontos de calibração:

$$Av = 0,$$

onde

$$A = \begin{bmatrix} x_1X_{w_1} & x_1Y_{w_1} & x_1Z_{w_1} & x_1 & -y_1X_{w_1} & -y_1Y_{w_1} & -y_1Z_{w_1} & -y_1 \\ x_2X_{w_2} & x_2Y_{w_2} & x_2Z_{w_2} & x_2 & -y_2X_{w_2} & -y_2Y_{w_2} & -y_2Z_{w_2} & -y_2 \\ \vdots & & & & & & & \vdots \\ x_nX_{w_n} & x_nY_{w_n} & x_nZ_{w_n} & x_n & -y_nX_{w_n} & -y_nY_{w_n} & -y_nZ_{w_n} & -y_n \end{bmatrix}.$$

Se $n \geq 7$ e os pontos são não-coplanares o sistema admite uma solução não-trivial que é proporcional a coluna de v correspondente ao menor valor singular de A , então:

$$\bar{v} = \begin{bmatrix} \bar{v}_1 & \bar{v}_2 & \bar{v}_3 & \bar{v}_4 & \bar{v}_5 & \bar{v}_6 & \bar{v}_7 & \bar{v}_8 \end{bmatrix} = \gamma \begin{bmatrix} r_4 & r_5 & r_6 & T_y & \alpha r_1 & \alpha r_2 & \alpha r_3 & \alpha T_x \end{bmatrix}.$$

Assim, podemos calcular os parâmetros usando as etapas a seguir:

a) *Determinar α e $\|\gamma\|$*

Usando o fato de que a matriz R é ortogonal, temos que

$$\begin{aligned} \|\gamma\| &= \sqrt{\bar{v}_1^2 + \bar{v}_2^2 + \bar{v}_3^2} \\ \alpha \|\gamma\| &= \sqrt{\bar{v}_5^2 + \bar{v}_6^2 + \bar{v}_7^2}. \end{aligned}$$

b) *Determinar os elementos da matriz de rotação*

$$\begin{aligned}
r_4 &= \frac{\bar{v}_1^2}{\|\gamma\|}, & r_1 &= \frac{\bar{v}_1^2}{\alpha \|\gamma\|}, \\
r_5 &= \frac{\bar{v}_2^2}{\|\gamma\|}, & r_2 &= \frac{\bar{v}_2^2}{\alpha \|\gamma\|}, \\
r_6 &= \frac{\bar{v}_3^2}{\|\gamma\|}, & r_3 &= \frac{\bar{v}_3^2}{\alpha \|\gamma\|}, \\
T_y &= \frac{\bar{v}_4^2}{\|\gamma\|}, & T_x &= \frac{\bar{v}_4^2}{\alpha \|\gamma\|}.
\end{aligned}$$

c) *Determinar o sinal de γ*

Temos que:

$$\begin{aligned}
x &= -f_x \frac{r_1 X_w + r_2 Y_w + r_3 Z_w + T_x}{r_7 X_w + r_8 Y_w + r_9 Z_w + T_z} = -\frac{f}{s_x} \frac{X_c}{Z_c} \\
y &= -f_y \frac{r_4 X_w + r_5 Y_w + r_6 Z_w + T_y}{r_7 X_w + r_8 Y_w + r_9 Z_w + T_z} = -\frac{f}{s_y} \frac{Y_c}{Z_c}.
\end{aligned}$$

Como f e Z_c são positivos, então se x e X_c possuem sinais opostos, deve-se trocar os sinais da primeira linha da matriz de rotação, e se y e Y_c possuem sinais opostos, deve-se trocar os sinais da segunda linha da matriz de rotação.

d) *Determinar f e T_z*

Podemos reescrever a equação

$$x(r_7 X_w + r_8 Y_w + r_9 Z_w + T_z) = -\frac{f}{s_x} (r_1 X_w + r_2 Y_w + r_3 Z_w + T_x)$$

para os n pontos de calibração, como a expressão matricial abaixo:

$$A \begin{bmatrix} T_z \\ f_x \end{bmatrix} = b,$$

onde

$$A = \begin{bmatrix} x_1 & r_1 X_{w_1} + r_2 Y_{w_1} + r_3 Z_{w_1} + T_x \\ x_2 & r_1 X_{w_2} + r_2 Y_{w_2} + r_3 Z_{w_2} + T_x \\ \vdots & \vdots \\ x_n & r_1 X_{w_n} + r_2 Y_{w_n} + r_3 Z_{w_n} + T_x \end{bmatrix},$$

e

$$b = \begin{bmatrix} -x_1 (r_7 X_{w_1} + r_8 Y_{w_1} + r_9 Z_{w_1} + T_z) \\ -x_2 (r_7 X_{w_2} + r_8 Y_{w_2} + r_9 Z_{w_2} + T_z) \\ \vdots \\ -x_n (r_7 X_{w_n} + r_8 Y_{w_n} + r_9 Z_{w_n} + T_z) \end{bmatrix}.$$

c) Determinar f_y

Para encontrar f_y basta usar a expressão abaixo:

$$f_y = \frac{f_x}{\alpha}.$$

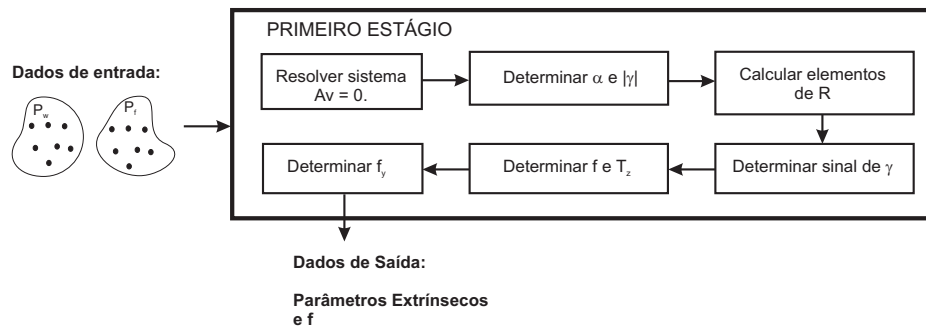


Figura 4.5: Primeiro Estágio do DLT.

Etapa 2 Determinar C_x e C_y .

Para encontrarmos a projeção do centro óptico basta calcular o ortocentro de um triângulo T cujos vértices são *pontos nulos*. Ou seja, toma-se três pares de linhas paralelas na imagem, e a interseção de cada par de reta no infinito consistenos vértices do triângulo e a partir de tal triângulo calcula-se seu ortocentro, que corresponderá ao parâmetro desejado, como vemos na figura 4.6.

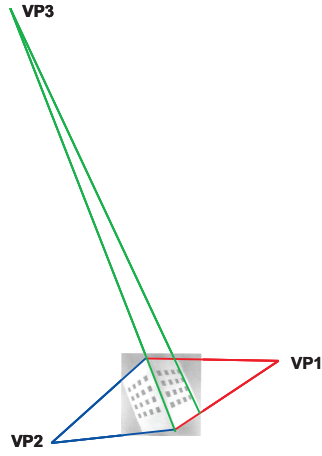


Figura 4.6: Triângulo.

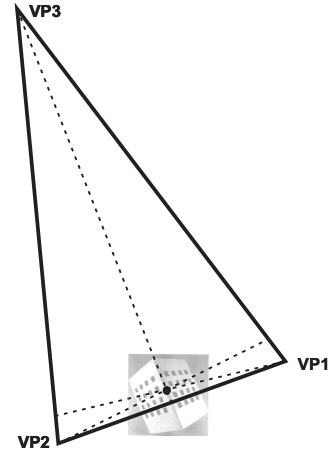


Figura 4.7: Ortocentro

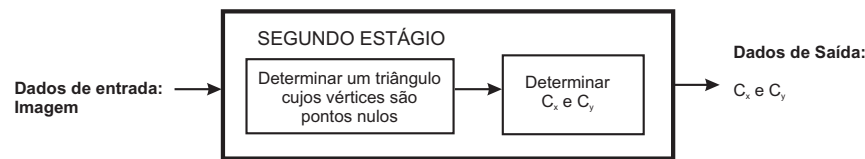


Figura 4.8: Segundo Estágio do DLT.

4.5 Método Círculos Coplanares com Invariância Quase-Afim

Este método, (Wu et al. 2006), se baseia na invariância quase-afim associada a dois círculos coplanares, cujos resultados serão aplicados para calibrar uma câmera. Assim, inicialmente daremos algumas preliminares sobre invariância quase-afim, pontos circulares e sua relação com a calibração de uma câmera.

Definição 4.5.1. *O fecho convexo de um conjunto $B \subset \mathbb{R}^n$, denotado por \bar{B} , é o menor conjunto convexo que contém B .*

Definição 4.5.2. *Sejam $B \subset \mathbb{R}^n$ e h uma transformação projetiva do sub-espço projetivo n -dimensional P^n . Então a transformação projetiva h é dita quase-afim em relação a B se $h^{-1}(L_\infty)$ não intersecta \bar{B} , onde L_∞ representa o subespaço infinito de P^n .*

Teorema 4.5.1. *Se B é um conjunto de pontos no plano do objeto em \mathbb{R}^3 , e B está inteiramente na frente de uma câmera projetiva, então a aplicação que associa o plano do objeto ao plano da imagem definida pela câmera é quase-afim em relação a B .*

Proposição 4.5.1. *Sejam x_0 e x_1 dois pontos no espaço, π um plano que não passa em nenhum destes pontos, e h uma transformação quase-afim em respeito aos dois pontos*

associando o ponto x_i ao ponto x'_i e levando o ponto π ao plano π' . Então x_0 e x_1 se situam do mesmo lado de π se e somente se x'_0 se situa do mesmo lado de x'_1 em π' .

Definição 4.5.3. Chamamos a imagem de uma reta no infinito de reta nula. E a imagem de um ponto no infinito é denominado ponto nulo.

Definição 4.5.4. A cônica absoluta é um ponto cônico virtual em P^3 o qual é invariante por transformações Euclidianas. Ela consiste de pontos $x = (x_1, x_2, x_3, 0)^T$ tal que

$$x_1^2 + x_2^2 + x_3^2 = 0, \quad x_4 = 0,$$

ou

$$x^T x = 0.$$

Definição 4.5.5. Seja l a interseção do plano $\pi \subset \mathbb{R}^3$ no plano no infinito. Então l , no plano do infinito, intersecta a cônica absoluta em um par de pontos complexos conjugados, chamados pontos circulares. Todo círculo no plano π passa através de dois pontos circulares.

Definição 4.5.6. A equação da imagem da cônica absoluta é dada por:

$$x^T K^{-T} K^{-1} x = 0,$$

onde K é a matriz intrínseca da câmera definida no capítulo 1.

$$K = \begin{bmatrix} f_x & s & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix}.$$

Então se y é imagem de um ponto circular, a equação abaixo deve ser satisfeita:

$$y^T K^{-T} K^{-1} y = 0.$$

Definição 4.5.7. Dois círculos reais C_1 e C_2 no espaço Euclidiano se intersectam em quatro pontos com multiplicidade sobre o campo dos complexos. Dois deles são os pontos circulares que estão na reta no infinito, denotada por L_∞ , e os outros dois ou são complexos conjugados ou são reais. A reta que passa por estes são sempre reais, e será denotada por L . Chamaremos as duas retas reais L e L_∞ de retas associadas a C_1 e C_2 .

A partir destas definições e resultados, podem ser feitas análises sobre as distribuições possíveis dos círculos coplanares e suas retas associadas:

- a) Caso concêntrico. C_1 e C_2 são virtualmente tangentes em dois pontos circulares, e as duas retas associadas coincidem com a reta passando através dos dois pontos circulares.

- b) Caso de tangência interna. C_1 e C_2 se intersectam em um único ponto real com multiplicidade dois e os dois pontos circulares. As retas associadas correspondem a única reta tangente aos dois círculos, e a reta no infinito que é separada dos dois círculos.
- c) Caso de tangência externa. C_1 e C_2 se intersectam em um único ponto real com multiplicidade dois e os dois pontos circulares. As retas associadas correspondem a única reta tangente aos dois círculos, e a reta no infinito que é separada dos dois círculos.
- d) Caso de interseção. C_1 e C_2 se intersectam em dois pontos reais e dois pontos circulares. As duas retas associadas são a reta entre as duas reais interseções e a reta no infinito que é separada dos dois círculos.
- e) Caso separado. C_1 e C_2 se intersectam em dois pares diferentes de complexos conjugados. As duas retas associadas são uma reta tal que os círculos se situem em lados diferentes da mesma, e a reta no infinito tal que os círculos estejam do mesmo lado dela.
- f) Caso de inclusão mas não concêntrico. C_1 e C_2 se intersectam em dois pares diferentes de complexos conjugados. As retas associadas, uma das quais é a reta do infinito, são tais que os círculos se situam no mesmo lado de ambas.

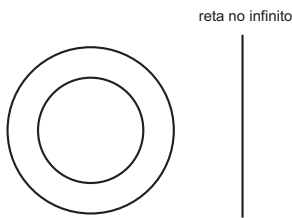


Figura 4.9: Caso (a).

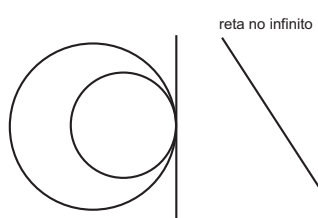


Figura 4.10: Caso (b).

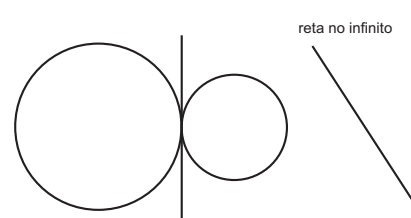


Figura 4.11: Caso (c).

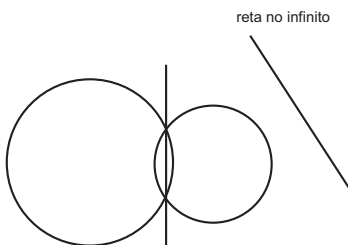


Figura 4.12: Caso (d).

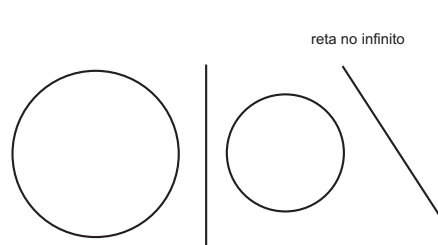


Figura 4.13: Caso (e).

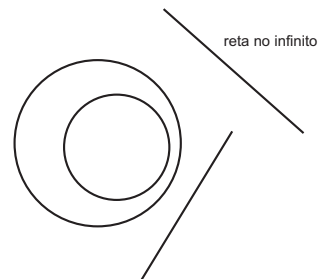


Figura 4.14: Caso (f).

Ao contrário dos casos **a** a **d**, os casos **e** e **f** não são invariantes quanto a uma transformação projetiva. Porém, eles permanecem invariantes quanto a transformação quase-afim.

Teorema 4.5.2. *Sejam c_1 e c_2 as imagens de dois círculos coplanares C_1 e C_2 sobre uma câmera pinhole, e chame l e l_∞ as imagens das duas retas associadas de C_1 e C_2 , as quais são determinadas pela interseção de c_1 com c_2 , onde l_∞ é a reta nula, então:*

- (i) *se C_1 e C_2 são separados, então c_1 e c_2 se situam em lados diferentes de l e do mesmo lado de l_∞ .*
- (ii) *se C_1 e C_2 estão um contido no outro, então ambas c_1 e c_2 se situam no mesmo lado de l e l_∞ .*

O fato de todos os círculos passarem por pontos circulares, torna o círculo de grande importância e utilidade para a calibração de câmera. O passo principal deste método é determinar as imagens dos pontos circulares das imagens dos círculos, que serão usados para calcular os parâmetros intrínsecos.

Suponha que existam dois círculos coplanares na cena e chame de c_1 e c_2 suas imagens, com respectivas equações cônicas algébricas homogêneas e_1 e e_2 . Note que para o caso **f** não podemos identificar a reta nula das duas retas que passam através dos dois pares de complexos conjugados, assim não podemos determinar as imagens dos pontos circulares.

Isto pode ser solucionado se conhecermos a imagem do centro de um dos círculos, ou um ponto nulo do plano contendo os círculos. Pois o centro de um círculo está em polaridade com os pontos circulares em relação ao círculo, e um ponto nulo é colinear com as imagens de pontos circulares.

Assim, podemos identificar as retas nulas dos casos **a** a **e** citados usando apenas as equações algébricas e_1 e e_2 através das invariância quase-afim ou projetivas, e dessa forma podemos obter também as imagens dos pontos circulares, desde que elas estão justo nas retas nulas.

Com isto, vemos que se existem dois círculos coplanares na cena, então podemos encontrar dois pontos, as imagens dos pontos circulares, na imagem da cônica absoluta, o que segue que teremos duas equações lineares nos parâmetros intrínsecos. Supondo que os parâmetros intrínsecos não variam, a câmera pode ser calibrada usando três imagens de dois círculos coplanares.

Então, o método proposto pode ser descrito como segue:

1. Tome as imagens c_1 e c_2 de dois círculos coplanares C_1 e C_2 para obter as equações algébricas homogêneas e_1 e e_2 .
2. Resolva o sistema de equações e_1 e e_2 e de acordo com a distribuição de C_1 e C_2 , determine as imagens dos pontos circulares, denotando-as por y_i , $i = 1, \dots, n$, $n \geq 6$.
3. Faça $N = K^{-T}K^{-1}$, e resolva por mínimos quadrados as equações $y_i^T N y_i = 0$, $i = 0, \dots, n$. N é uma matriz simétrica com cinco parâmetros independentes;

4. Faça a decomposição de N por Cholesky para n , invertendo seu resultado obtemos uma estimativa inicial para os parâmetros intrínsecos, denotados por K_0 .
5. Por último pode ser feita uma otimização não-linear com solução inicial dada por K_0 .

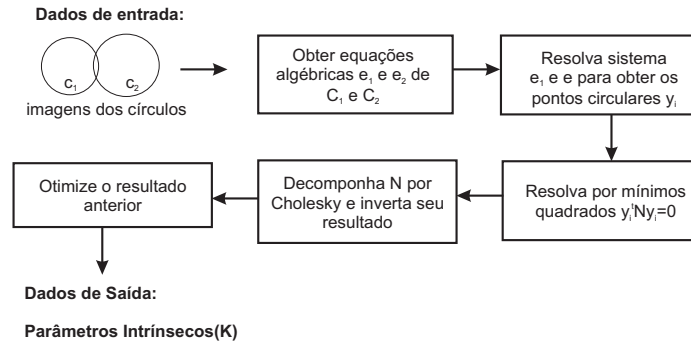


Figura 4.15: Etapas do Método dos Pontos Circulares.

Capítulo 5

Otimização

5.1 Introdução

Devido à modelagem do comportamento das lentes, as equações para a calibração tornam-se não lineares e assim, podem ser resolvidas por métodos de otimização não linear, como Levenberg-Marquardt (Ranganathan 2004), (Lourakis 2005).

O método de Levenberg-Marquardt é um dos mais utilizados para a minimização de funções representadas por uma soma de quadrados de funções reais de mais de uma variável. Apesar disto, ele determina apenas o mínimo local da função. Uma proposta para trabalhos futuros consiste em utilizar o método de Gloptipoly (Henrion & Lasserre 2002), (Schweighofer 2005), (Lasserre 2001), que determina o mínimo global, ou indica o intervalo ao qual ele pertence.

Cada novo método criado é implementado como uma subclasse da classe abstrata de otimização. A figura 5.1 ilustra uma hierarquia de classes com dois métodos de otimização.

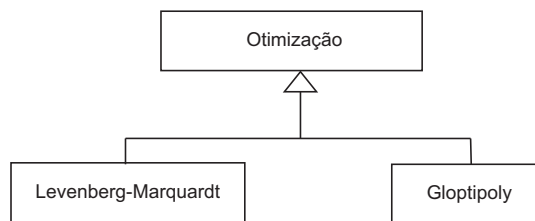


Figura 5.1: Subclasses para a classe de Otimização.

A etapa da otimização envolve as seguintes informações:

- Nome do método;
- Diretório da dll, onde se encontra a biblioteca dinâmica com a implementação do método;
- Matriz intrínseca K , calculada na etapa de calibração;

- Matriz de rotação R, calculada na etapa de calibração;
- Vetor de Translação T, calculado na etapa de calibração;

As operações principais envolvidas na otimização são:

- Atribuir/Ler nome do método;
- Definir/Ler o diretório da dll;
- Atribuir/Ler os parâmetros intrínsecos;
- Atribuir/Ler os parâmetros extrínsecos;
- Executar o método;

A seguir descreveremos os métodos de otimização Levenberg-Marquardt e Gloptipoly.

5.2 Levenberg-Marquardt

O método de Levenberg-Marquardt (Ranganathan 2004), (Lourakis 2005) é uma técnica iterativa de calcular o mínimo de uma função de várias variáveis expressa por uma soma de quadrados de funções reais não-lineares. Ele pode ser visto como uma combinação do método *Steepest-Descent* e do *Gauss-Newton*. Quando a solução atual está longe da correta, ele se comporta como “Steepest-Descent”, lento mas garantindo a convergência, e quando a solução está próxima da correta ele se torna o método de Gauss-Newton. Para maiores detalhes consultar (Lourakis 2005).

O problema a ser solucionado então consiste em encontrar x que minimize a função

$$f(x) = \frac{1}{2} \sum_{j=1}^m r_j^2(x),$$

onde $x = (x_1, x_2, \dots, x_n)$ é um vetor, e cada r_j é uma função de \mathbb{R}^n em \mathbb{R} , com $m \geq n$ denominadas *resíduos*. Tomando $r : \mathbb{R}^n \rightarrow \mathbb{R}^m$, $r(x) = (r_1(x), r_2(x), \dots, r_m(x))$, podemos reescrever f como

$$f(x) = \frac{1}{2} \sum_{j=1}^m \|r(x)\|^2.$$

Para um pequeno $\|\delta_x\|$, a expansão em série de Taylor fornece

$$r(x + \delta_x) \simeq r(x) + J_x \delta_x,$$

onde J_x é a matriz jacobiana $\frac{\partial r(x)}{\partial x}$. Como qualquer método iterativo, este método começa com um ponto inicial x_0 e irá produzir uma série de vetores x_1, x_2, \dots , que irá

convergir para um mínimo local p' de r . Então para cada passo, é necessário determinar um δ_x que minimize

$$\begin{aligned} \|\hat{r} - r(x + \delta_x)\| &\simeq \|\hat{r} - r(x) - J_x \delta_x\| \simeq \\ &\|\epsilon - J_x \delta_x\|, \end{aligned}$$

denotando $\epsilon = \hat{r} - r(x)$. O valor de δ_x é a solução para o problema de mínimos quadrados linear. O mínimo é obtido quando $J_x^T (J_x \delta_x - \epsilon) = 0$, o qual fornece δ_x como solução para as chamadas *equações normais*

$$J_x^T J_x \delta_x = J_x^T \epsilon. \quad (5.1)$$

A matriz $J_x^T J_x \delta_x$ é uma aproximação para a matriz Hessiana em x . O método Levenberg-Marquardt resolve uma pequena variação da equação 5.1, conhecidas como as *equações normais aumentadas*,

$$N \delta_x = J_x^T \epsilon, \quad (5.2)$$

onde os elementos que não pertencem a diagonal de N são idênticos aos elementos de $J_x^T J_x$ e os elementos da diagonal são dados por $N_{ii} = \mu + [J_x^T J_x]_{ii}$, para algum $\mu > 0$. A estratégia de alterar os valores da diagonal de uma matriz é chamada de *redução*, e ϵ é denominado *pivot*.

Se o vetor $x + \delta_x$ com δ_x calculado por 5.2 nos dá uma redução para ϵ , o valor é aceito e o processo se repete com o pivot decrementado. Caso contrário, o pivot é incrementado, as equações normais aumentadas são resolvidas e as iterações permanecem até encontrar δ_x que reduza o erro. O processo de resolver a equação 5.2 para diferentes valores do pivot corresponde a uma iteração do método LM. O algoritmo termina com no mínimo uma das seguintes condições:

- A magnitude de $J_x^T \epsilon$ fica inferior a um threshold ϵ_1 ,
- Uma mudança na magnitude de δ_x fica inferior a um threshold ϵ_2 ,
- O erro $\epsilon^T \epsilon$ fica inferior a um threshold ϵ_3 ;
- O máximo de iterações é alcançado.

5.3 Gloptipoly

O método Levenberg-Marquardt obtém apenas o mínimo local das funções. Para obter uma aproximação do mínimo global é necessário buscar outras soluções. Isto pode ser

obtido através de relaxações convexas, por uma função polinomial restrita a um conjunto de desigualdades polinomiais (Schweighofer 2005).

Definição 5.3.1. *Seja $f : S \rightarrow \mathbb{R}$, onde $S \subset \mathbb{R}^n$ é um conjunto convexo não vazio. Então a função convexa $u : S \rightarrow \mathbb{R}$ é uma relaxação convexa se*

$$u(x) \leq f(x), \forall x \in S.$$

Gloptipoly é um “add-on” do Matlab que resolve relaxações convexas de problemas de otimização de polinômio globais não-convexos com funções polinomiais de várias variáveis. O método utilizado no Gloptipoly se baseia no método usado por Lasserre (Lasserre 2001), (Schweighofer 2005), o qual daremos uma breve introdução do que se trata a seguir.

Denote $\overline{X} = (X_1, \dots, X_n)$ e $\mathbb{R}[\overline{X}]$ o anel dos polinômios reais em n variáveis. Seja então o conjunto semi-algébrico fechado:

$$S = \{g_1(x) \geq 0, \dots, g_m(x) \geq 0\},$$

definidos pelos polinômios $g_1, \dots, g_m \in \mathbb{R}[\overline{X}]$. O método de Lasserre se propõe a determinar

$$f^* := \inf \{f(x) \mid x \in S\} \in \mathbb{R} \cup \{\infty\},$$

e, se possível um ponto ótimo, ou seja, um elemento do conjunto

$$S^* := \{x^* \in S \mid \forall x \in S : f(x^*) \leq f(x)\}.$$

No Gloptipoly o polinômio pode conter até 19 variáveis. Este método pode ser adaptável ao problema de calibração para otimizar os parâmetros intrínsecos da câmera, uma vez que teremos funções polinomiais cujas variáveis são geralmente apenas os três parâmetros: translação no eixo Z (T_z), a distorção radial (κ_1), e a distância focal (f).

Capítulo 6

Implementação do Sistema de Calibração

6.1 Introdução

Este capítulo mostra a implementação do sistema de calibração modelado de acordo com a arquitetura proposta. Trata-se de uma ferramenta de calibração que possibilita a edição de processos de calibração e cujas etapas podem ser totalmente personalizadas pelo usuário. Além disso, a execução de vários processos ao mesmo tempo pode tornar o sistema mais atrativo e útil para aplicações que necessitem de uma análise comparativa entre resultados de diferentes métodos de calibração.

O sistema desenvolvido consiste em dois módulos: o *EditCalib* e o *ExecCalib*. O módulo *EditCalib* permite ao usuário a criação e edição dos elementos possíveis em uma calibração, através da composição dos pipelines.

O módulo *ExecCalib* fica responsável pela leitura e interpretação dos pipelines gerados no módulo anterior e pela execução da calibração. Este módulo permite a troca de dados antes da execução, o que proporciona mais agilidade e flexibilidade em etapas de testes e comparações de resultados.

6.2 Módulo EditCalib

Os programas de calibração disponíveis na literatura, em geral, são ferramentas robustas de calibração que fornecem ao usuário bons resultados. Porém, às vezes se torna útil a combinação de diferentes elementos ou etapas para determinadas aplicações. Essas ferramentas nem sempre fornece tais facilidades.

O módulo *EditCalib* permite ao usuário a criação e edição de cada elemento que possa constar em um pipeline de calibração. Assim como é possível combinar quaisquer elementos que possam ser adaptados a novos processos de calibração. Apesar de a sintaxe

permitir tais combinações, é importante realizar uma análise semântica do problema ao se propor um novo método de calibração, visto que o resultado obtido dependerá da coerência entre as etapas. Este módulo é responsável apenas pela sintaxe do processo de calibração, e apenas analisa possíveis restrições entre alguns elementos que serão definidas pelo próprio usuário.

Cada elemento da calibração foi implementado como classes dentro do sistema. O diagrama de classes deste módulo encontra-se ilustrado na figura 6.1.

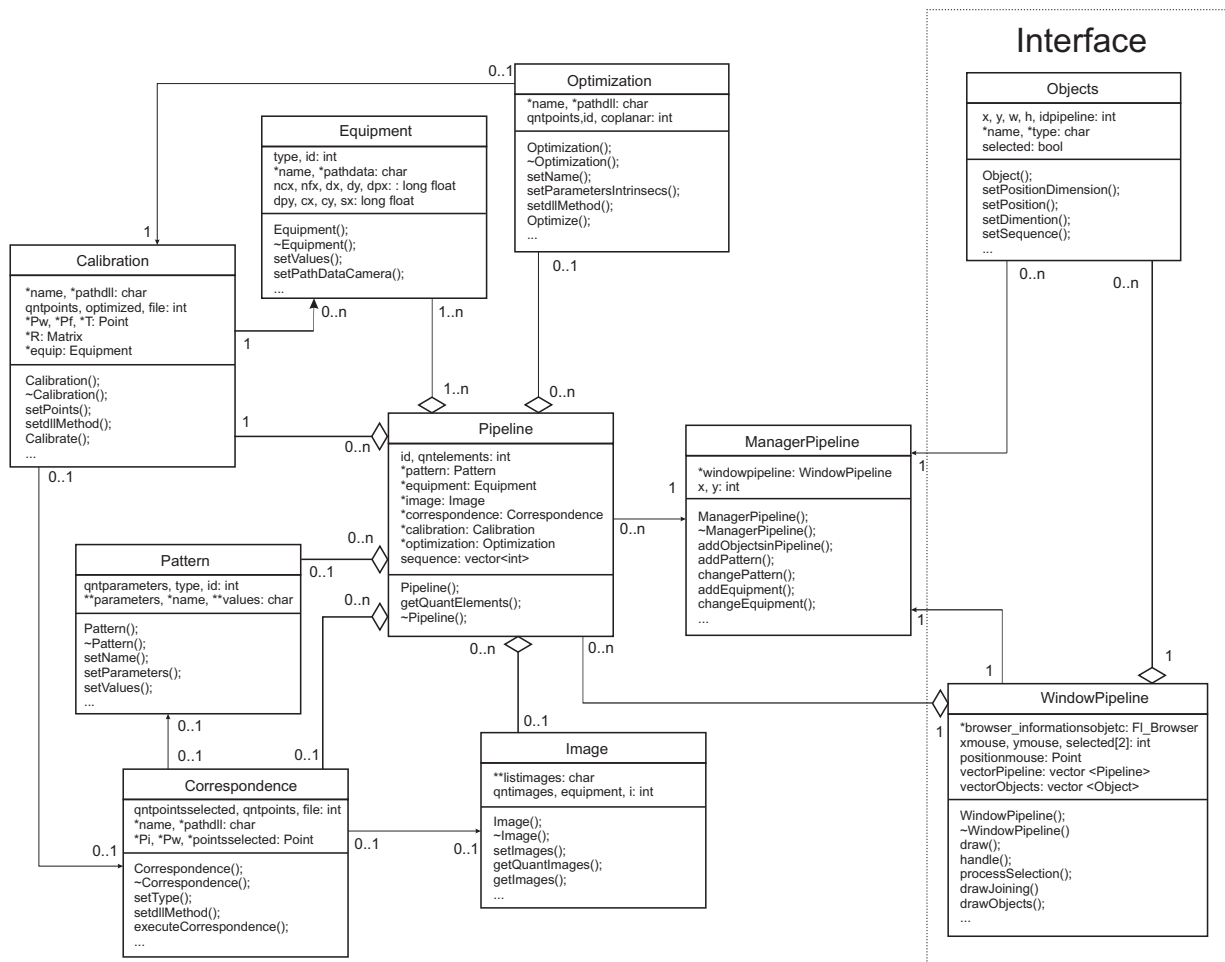


Figura 6.1: Diagrama de Classes.

A interface da ferramenta foi elaborada de forma simples, para facilitar a interação entre o usuário e o sistema, como pode se na figura 6.2.

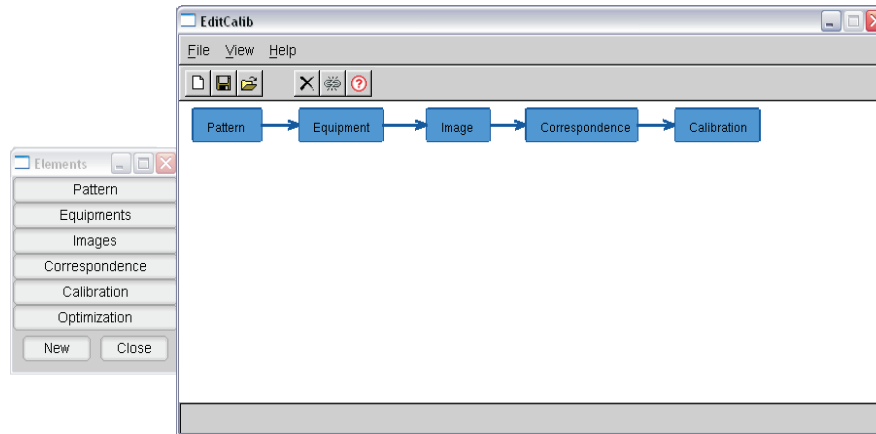


Figura 6.2: Módulo EditCalib.

Iremos detalhar agora como cada elemento pode ser criado ou personalizado dentro do sistema.

6.2.1 Padrão

Os padrões foram implementados em uma classe denominada Pattern. Para adicionar novos modelos de padrões, o usuário deverá especificar o nome, o tipo (se coplanar ou não), e cada parâmetro que o padrão precisa para determinar seus pontos. Escolhido um novo modelo de padrão para ser adicionado ao pipeline de calibração, o usuário deve informar o valor de cada parâmetro. Estes podem ser salvos em arquivos para posterior uso em novos pipelines. Restringimos o sistema de forma que cada pipeline só possa ter um padrão por pipeline. A classe implementada para o padrão encontra-se descrita abaixo:

```

1 class Pattern {
2
3 private:
4
5     int  qntparameters , type , id ;
6     char **parameters , *name , ** values ;
7
8 public:
9
10    Pattern () ;
11    Pattern (char *namepattern , int  qntparams , char **params ) ;
12    void setName (char *namepattern ) ;
13    void setParameters (int  qntparams , char **params ) ;

```

```
14 void setValues(int qntvalues , char **valuespattern );
15 void setId(int Id);
16 void getName(char **namepattern);
17 void getParameters(char ***params);
18 void getValues(char ***valuespattern);
19 void getType(int *typepattern);
20 void getQuantParameters(int *qntparams);
21 void getId(int *Id);
22 virtual ~Pattern();
23 };
```

6.2.2 Equipamento

Existem dois tipos de equipamentos disponíveis no sistema: câmera e projetor. Os atributos das câmeras são praticamente os mesmos para os projetores. A única diferença é que a calibração de projetores pode requerer os dados de calibração da câmera que fotografou as imagens utilizadas na calibração do projetor. Desta forma, para calibrar um projetor, é necessária a leitura de um arquivo que contenha as informações obtidas na calibração de uma câmera específica.

É possível a adição de novos modelos de equipamentos no sistema, apenas informando seu tipo, o nome do modelo e as informações específicas do equipamento dadas pelo fornecedor. Só é possível a adição de um equipamento por pipeline. A classe implementada para o equipamento encontra-se abaixo:

```
1 class Equipment {
2
3 private:
4
5     int type , id;
6     char *name , *pathdata;
7
8 public:
9
10    long double ncx , nfx , dx , dy , dpx , dpy , cx , cy , sx;
11
12    Equipment(int typeequipment);
13    void setName(char *nameequipment);
14    void setValues(char *nameequipment , long float Ncx ,
15    long float Nfx , long float Dx , long float Dy,
```

```

16     long float Dpx, long float Dpy,
17     long float Cx, long float Cy, long float Sx);
18     void setValuesOnly(long float Ncx, long float Nfx,
19     long float Dx, long float Dy, long float Dpx,
20     long float Dpy, long float Cx, long float Cy, long float Sx);
21     void setPathDataCamera(char *pathdatacamera);
22     void setId(int Id);
23     void getName(char **nameequipment);
24     void getPathDataCamera(char **pathdatacamera);
25     void getId(int *Id);
26     virtual ~Equipment();
27
28 };

```

6.2.3 Imagem

Foi implementado uma classe Image para especificar as imagens que serão usadas na calibração. Um objeto desta classe pode representar várias ou apenas uma imagem. A lista de imagens pode ser salva em arquivo ou alterada para um posterior uso em outros pipelines. A classe implementada para a imagem encontra-se abaixo:

```

1 class Image {
2
3 private:
4     char **listimages;
5     int qntimages, id;
6
7 public:
8
9     Image();
10    void setImages(int qnt, char **pathsimages);
11    void setId(int Id);
12    void getImages(char ***pathsimages);
13    void getImage(int id, char **pathimage);
14    void getEquipment(int *typeequipment);
15    void getQuantImages(int *qnt);
16    void getId(int *Id);
17    virtual ~Image();
18

```

```
19 };
```

6.2.4 Correspondência

Cada novo método de correspondência a ser incluído no sistema corresponde a um algoritmo que deverá ser incorporado dinamicamente ao sistema. Assim para adicionar o novo método ao sistema, o usuário deverá informar o nome, o tipo de correspondência e o diretório da dll.

A implementação de um método de correspondência deverá ser realizada através do método abstrato *executeCorrespondence*. Apenas um método de correspondência pode ser adicionado a um pipeline. A classe implementada para a correspondência encontra-se abaixo:

```
1 class Correspondence {
2
3 protected:
4
5     int type, id, coplanar, qntpointsselected, qntpoints,
6         qntparams;
7     char *name, *pathdll;
8     Point *Pi, *Pw, *pointsselected;
9
10 public:
11
12     Image *img;
13
14     Correspondence();
15     Correspondence(int typecorrespondence, char *namecorresp);
16     virtual void refreshParameters();
17     void setName(char *namemethod);
18     void setType(int typecorrespondence);
19     void setId(int Id);
20     void setdllMethod(char *path);
21     void setMethod(char *namecorrespondence, char *path);
22
23     void setRestrictionCoplanar(int typepattern);
24     void setPointsSelected(Point *points);
25     void setQuantPoints(int qnt);
26     void setPointSCM(Point *pw);
```

```

27     void setPointSCI(Point *pi);
28     void setPoints(Point *pi , Point *pw);
29     void getPointSCM(Point **pw);
30     void getPointSCI(Point **pi);
31     void getPoints(Point **pi , Point **pw);
32     void getPointsSelected(Point **points);
33     virtual int executeCorrespondence();
34     void getName(char **namemethod);
35     void getType(int *typecorrespondence);
36     void getId(int *Id);
37     void getdllMethod(char **path);
38     void getRestrictionCoplanar(int *typepattern);
39     void getQuantPoints(int *qnt);
40     virtual ~Correspondence();
41 };

```

6.2.5 Calibração

A classe de calibração também foi implementada utilizando biblioteca dinâmica. Um método de calibração usual tem como dados de entrada um conjunto de pares de pontos bidimensionais e tridimensionais. Estes pontos podem ser adquiridos a partir de um método de correspondência executado previamente, ou mesmo de um arquivo. Ainda assim, o uso destes dados não são obrigatórios, o que permite o usuário implementar algoritmos de calibração não-usuais.

Assim como para a correspondência, a execução da calibração é feita através de métodos abstratos. A classe implementada para a calibração encontra-se abaixo:

```

1
2 protected:
3
4     char *name, *pathdll;
5     int qntpoints, id, coplanar, optimized;
6     Point *Pw, *Pf;
7     Point *T;
8     Matrix *R, *K;
9
10 public:
11
12     Equipment *equip;

```

```
13
14     Calibration ();
15     void setName(char *method);
16     void setQntPoints(int numpoints);
17     void setPointsSCM(Point *pw);
18     void setPointsSCI(Point *pf);
19     void setPoints(Point *pw, Point *pf);
20     void setId(int Id);
21     void setdllMethod(char *path);
22     void setMethod(char *namecalibration , char *path);
23     void setRestrictionCoplanar(int typepattern);
24     void setRestrictionImages(int qntimages);
25     void setRestrictionOptimization(int option);
26     void setRestrictions(int typepattern , int option);
27     void setParametersExtrinsecs(Matrix *r, Point *t);
28     void setVectorTranslation(Point *t);
29     void setMatrixRotation(Matrix *r);
30     void setMatrixIntrinsec(Matrix *k);
31     void setDatasCalibration(Matrix *r, Point *t, Matrix *k);
32     virtual int Calibrate();
33     void getParametersExtrinsecs(Matrix **r, Point **t);
34     void getVectorTranslation(Point **t);
35     void getMatrixRotation(Matrix **r);
36     void getMatrixIntrinsec(Matrix **k);
37     void getDatasCalibration(Matrix **r, Point **t, Matrix **k);
38     void getQntPoints(int *numpoints);
39     void getPointsSCM(Point **pw);
40     void getPointsSCI(Point **pf);
41     void getPoints(Point **pw, Point **pf);
42     void getId(int *Id);
43     void getdllMethod(char **path);
44     void getMethod(char **namecorrespondence , char **path);
45     void getRestrictionCoplanar(int *typepattern);
46     void getRestrictionOptimization(int *option);
47     void getMethod(char **method);
48     virtual ~Calibration();
49
50 };
```

6.2.6 Otimização

Assim como na calibração e na correspondência, novos métodos de otimização podem ser adicionados dinamicamente ao sistema. Restringimos também o uso de uma otimização por pipeline. A classe de otimização também foi implementada como uma classe abstrata:

```
1 class Optimization {
2
3 private:
4
5     char *name, *pathdll;
6     int id;
7     Matrix *K, *R;
8     Point *T;
9
10 public:
11
12     int qntpoints;
13     Optimization();
14
15     Optimization(char *method);
16     void setName(char *method);
17     void setParametersIntrinsecs(Matrix *K);
18     void setId(int Id);
19     void setdllMethod(char *path);
20     void setMethod(char *nameOptimization, char *path);
21     virtual int Optimize();
22     void getParametersIntrinsecs(Matrix *K);
23     void getParametersOptimized(Matrix **K);
24     void getId(int *Id);
25     void getdllMethod(char **path);
26     void getMethod(char **method);
27     virtual ~Optimization();
28
29 };
```


6.2.7 Gerenciador de Pipelines

Para representar um pipeline, foi implementada uma classe Pipeline que conterá todos os elementos necessários para o processo de calibração. Para gerenciar a inclusão e alterações dos objetos em um determinado pipeline, foi criada uma classe ManagerPipeline que faz a comunicação entre o usuário e o pipeline.

O ManagerPipeline permite a criação no máximo de cinco pipelines. Durante a execução do programa, apenas uma instância dessa classe é necessária.

Também foi implementada uma classe Objects que deverá representar cada elemento do pipeline na Interface do EditCalib. Esta representação é feita através de “caixinhas” cujas posições na janela de pipelines, WindowPipeline, serão pré-determinadas.

Ao adicionar um novo elemento no pipeline, o Gerenciador de Pipelines criará um objeto da classe Objects representando o elemento adicionado, que será exibido na janela de pipelines. A classe implementada para o gerenciador de pipelines encontra-se abaixo:

```
1 class ManagerPipeline {
2
3 private:
4
5     WindowPipeline *windowpipeline;
6     int x, y;
7
8 public:
9
10    ManagerPipeline(WindowPipeline *window);
11    ManagerPipeline();
12    void addObjectsinPipeline(int id, int X, int Y, char *label,
13                               char *nameobject);
14    void addPattern(int id, Pattern *pattern);
15    void changePattern(int id, Pattern *pattern);
16    void addEquipment(int id, Equipment *equipment);
17    void changeEquipment(int id, Equipment *equipment);
18    void addImages(int id, Image *image);
19    void changeImages(int id, Image *image);
20    void addCorrespondence(int id, Correspondence *corresp);
21    void changeCorrespondence(int id, Correspondence *corresp);
22    void addCalibration(int id, Calibration *calibration);
23    void changeCalibration(int id, Calibration *calibration);
24    void addOptimization(int id, Optimization *optim);
```

```
25     void changeOptimization(int id, Optimization *optim);
26     void createnewPipeline(int *id);
27     bool checkRestrictionsPattern(int id, int typepattern);
28     bool checkTypePattern(int id, int typepattern);
29     bool checkOptimization(int id, int optimized);
30     bool checkRestrictionOptimization(int id);
31     void setWindow(WindowPipeline *window);
32     virtual ~ManagerPipeline();
33
34 };
```

6.3 Módulo ExecCalib

Este módulo têm por finalidade a leitura, interpretação e execução dos pipelines criados pelo módulo EditCalib. Estes pipelines foram gravados em arquivos no EditCalib. Os arquivos podem conter um ou mais pipelines que serão lidos e interpretados pelo módulo ExecCalib e em seguida executados.

Os pipelines serão interpretados e executados sequencialmente para então exibir seus resultados. A execução de cada pipeline, deverá ser realizada também de forma sequencial, e seus resultados serão armazenados em variáveis que serão passadas para etapas posteriores.

Para exemplificar, considere a execução da Correspondência. O resultado da execução de um método de Correspondência são pontos que deverão ser usados no processo de Calibração.

Durante a execução dos pipelines será possível acompanhar as etapas de execução através de uma janela que detalhará os passos que estarão sendo realizados, assim como os resultados de cada etapa principal, como correspondência, calibração e otimização serão exibidos em novas janelas, com a possibilidade de salvar os mesmos em arquivos.

É possível executar todos os pipelines sequencialmente, ou apenas um em uma lista de pipelines abertos. Ao selecionar um pipeline todas as informações de seus elementos serão exibidas no canvas ao lado. Existe a possibilidade de, ao executar a etapa de correspondência, tomarmos pontos já armazenados em um arquivo de correspondências anteriores, em vez de o usuário especificar na imagem. Isto permite maiores possibilidades de testes e análises.

A figura 6.3 mostra o módulo ExecCalib.

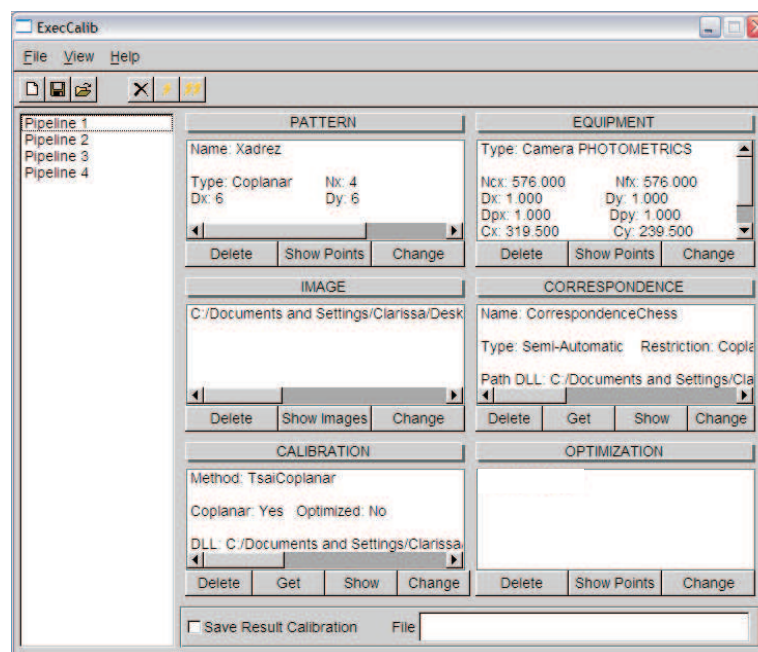


Figura 6.3: Módulo ExecCalib.

Capítulo 7

Testes e Resultados

7.1 Introdução

Iremos agora exibir alguns resultados obtidos neste trabalho baseados na arquitetura proposta e discutir alguns testes. Ilustraremos a técnica utilizada para a inclusão de novos métodos de calibração, correspondência e otimização.

7.2 Resultados

7.2.1 Arquitetura

A arquitetura proposta permite que quaisquer elementos sejam combinados em um pipeline e seu resultado seja os parâmetros extrínsecos e intrínsecos do equipamento calibrado. A implementação realizada neste trabalho permite restrições a esta proposta. A coerência entre os elementos combinados deve ser mantida pelo usuário de forma que o resultado seja o esperado. Como exemplo podemos citar que um pipeline apenas composto por uma otimização e uma imagem não terá como resultado os parâmetros do equipamento calibrado.

Já o pipeline construído na figura 7.1 pode obter os resultados desejados, desde que ou o elemento de calibração ou o de correspondência tome seus dados de entrada em um arquivo.

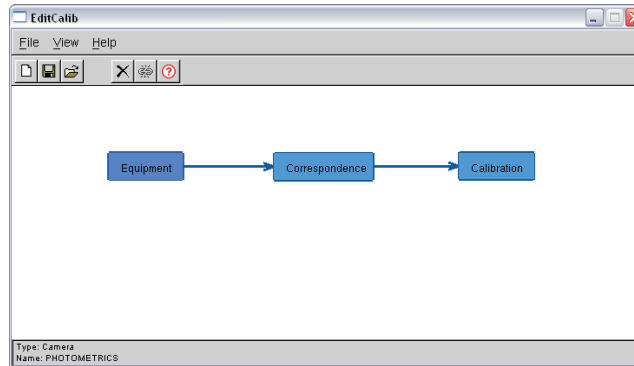


Figura 7.1: Calibração não-convencional.

O método de calibração definido no pipeline acima é o Tsai Coplanar, que apenas precisa do conjunto de pares de pontos 2D e 3D, e algumas especificações do equipamento. Assim, a semântica do pipeline da figura 7.1 está correta.

É possível ainda a combinação entre elementos de pipelines diferentes. Neste caso, os pipelines antigos deverão ser reformulados de forma que a coerência nos mesmos permaneça. Para exemplificar considere os pipelines da figura 7.2. Este resultado da figura 7.2 é interpretado como os dois pipelines em destaque na figura 7.3.

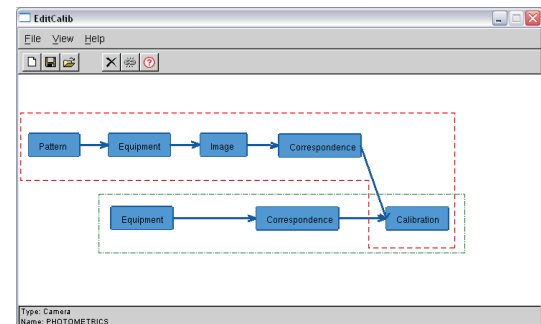
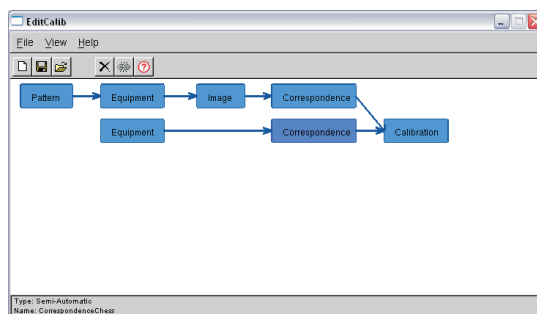


Figura 7.2: Combinação entre pipelines diferentes.

Figura 7.3: Novos pipelines.

É importante observar que, caso um elemento seja selecionado, suas informações serão exibidas na barra da janela como mostra a figura 7.3.

7.2.2 Módulo EditCalib

O foco deste módulo consiste na personalização de cada elemento possivelmente presente em um pipeline.

Inclusão de Elementos

A inclusão de novos padrões, equipamentos e imagens é feita naturalmente apenas informando dados necessários para a identificação dos mesmos.

Porém, a integração de novos métodos de correspondência, calibração e otimização não ocorre de forma tão natural, visto que seria necessária a inclusão do código do método no sistema. Vimos que este problema foi solucionado com o uso de bibliotecas dinâmicas (dll's), visando proteger o sistema e torná-lo mais fácil e flexível para o usuário. A figura 7.4 ilustra a inclusão de um novo método de calibração.

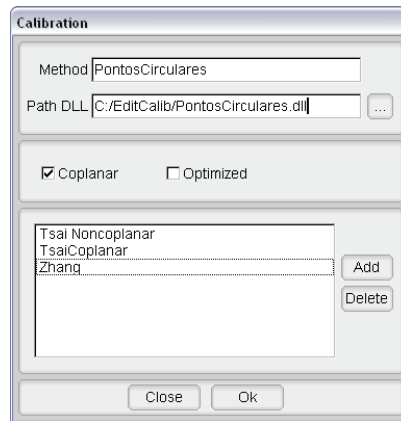


Figura 7.4: Inclusão de um novo método de calibração.

Exclusão de Elementos

Os elementos que compõem um pipeline também podem ser deletados. Para isto, basta selecionar e apertar a tecla 'D' ou usar o botão responsável por isto, destacado na figura 7.6.

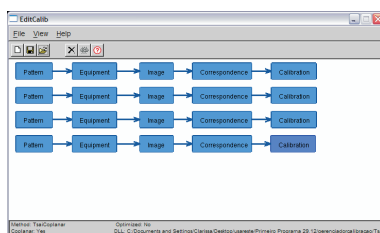


Figura 7.5: Deletar um elemento.

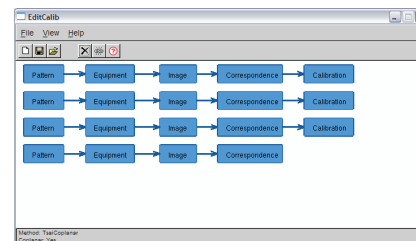


Figura 7.6: Após clicar Deletar.

Edição de Elementos e Relações

Outra funcionalidade disponível é a alteração de elementos contidos em um pipeline, e das relações entre os mesmos. Para alterar as informações de um elemento, basta dar um clique duplo sobre o elemento para que seja exibida a janela com as informações

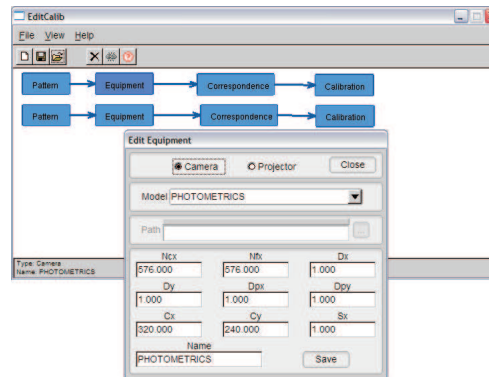


Figura 7.7: Alterar um elemento.

do mesmo. Após a alteração dos dados clica-se em salvar para confirmar as alterações realizadas. Um exemplo desta operação encontra-se na figura 7.7.

Para alterar a relação existente entre os elementos basta selecionar os dois elementos que deverão ser combinados apertando a tecla ‘SHIFT’ continuamente, e clicar no botão de combinar elementos, como mostra as figuras 7.8 e 7.9.

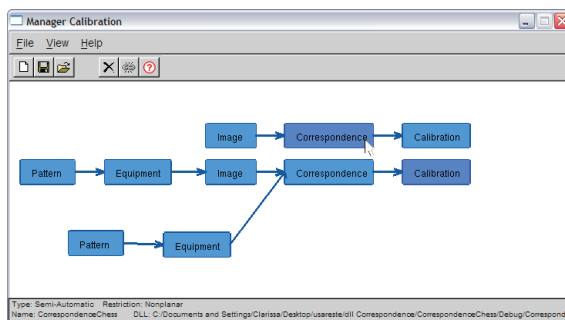


Figura 7.8: Selecione os dois elementos.

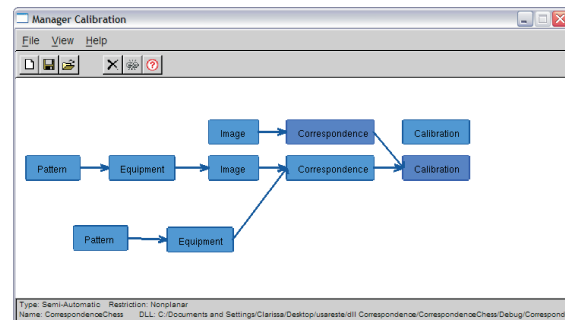


Figura 7.9: Após clicar Ligar.

7.2.3 Módulo ExecCalib

Este módulo é responsável pela execução dos pipelines. Os pipelines são executados sequencialmente, quando existe mais de um a ser executado. Cada elemento de um pipeline será interpretado e processado caso seja preciso. Os resultados processados serão passados adiantes em etapas seguintes para chegar ao resultado desejado.

Se um método de correspondência manual ou semi-automático foi definido em um pipeline, a execução sequencial dos pipelines será interrompida para a intervenção do usuário na correspondência. Após efetuada a operação do usuário, o processo de execução dos pipelines continuará até uma nova interrupção ou término da execução. A figura 7.10 ilustra este processo.

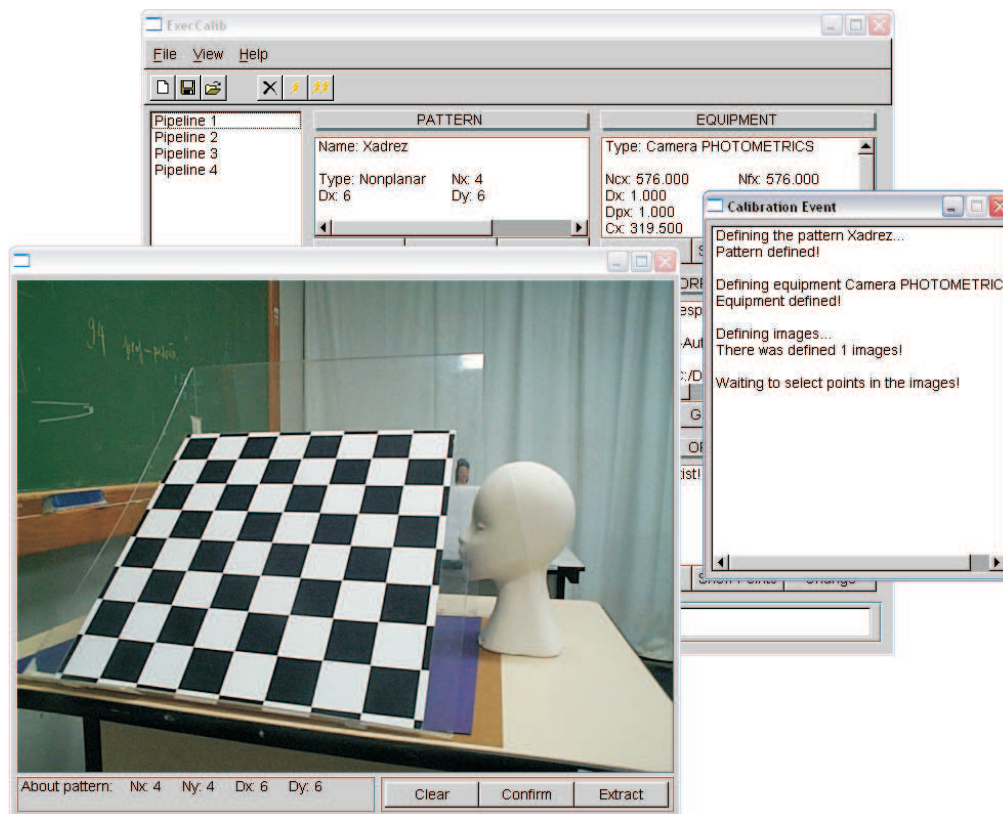


Figura 7.10: Correspondência manual ou semi-automática.

7.3 Testes de Calibração

O sistema desenvolvido foi testado com os métodos Tsai Coplanar e Tsai Não-Coplanar. O método de Tsai Coplanar utilizou um padrão xadrez construído no laboratório de Pós-Graduação da Matemática e utilizando arquivos com informações da correspondência. O método do Tsai Não-Coplanar utilizou pontos obtidos através da leitura de arquivos.

Também foram feitos testes com vários pipelines e diferentes combinações entre elementos. A implementação de novos métodos de calibração, correspondência e otimização foram propostos como trabalhos futuros.

Foi implementado e integrado ao sistema o método de correspondência para padrões Xadrez. Ele foi testado em calibrações de câmeras e projetores com os padrões construídos no laboratório de Pós-graduação.

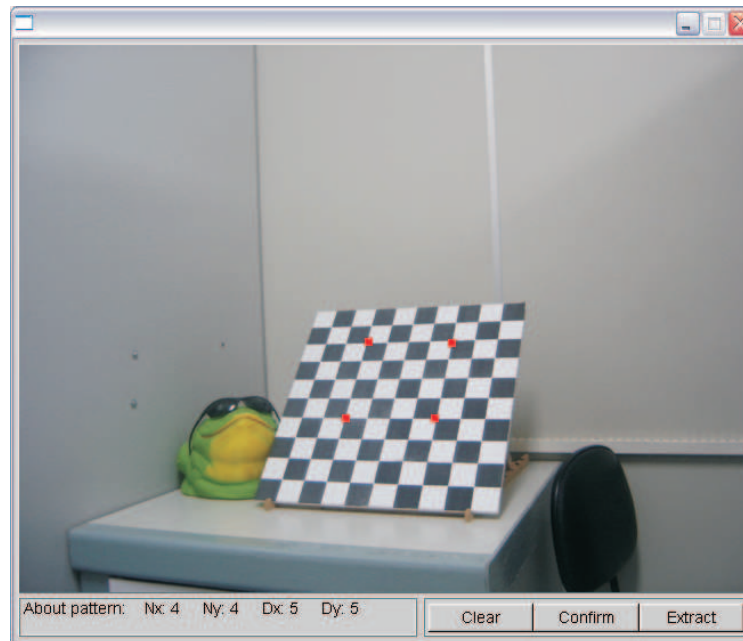


Figura 7.11: Seleção dos pontos pelo usuário.

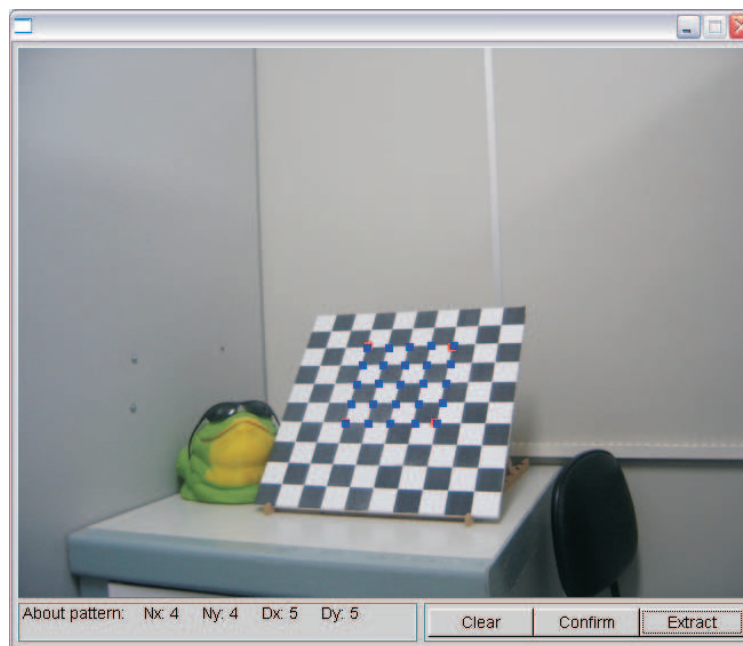


Figura 7.12: Extração dos demais pontos.

As figuras 7.11 e 7.12 mostram os testes realizados pelo método de correspondência para a calibração de uma câmera. A figura 7.11 corresponde a seleção dos pontos pelo usuário, e a figura 7.12 ilustra os pontos determinados pelo método de correspondência a partir dos quatros pontos da figura 7.11.

Da mesma forma, as figuras 7.13 e 7.14 mostram os testes da correspondência realizados na calibração de um projetor. O padrão e o método de correspondência utilizado são os mesmos das figuras 7.11 e 7.12.

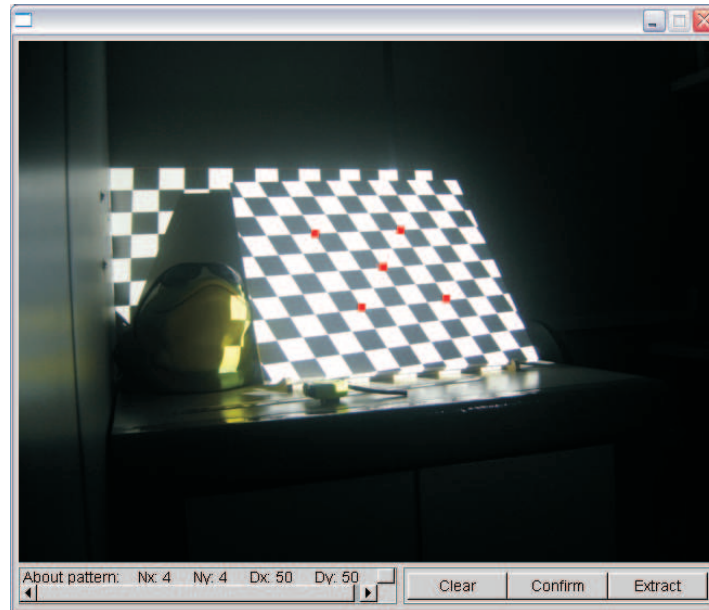


Figura 7.13: Seleção dos pontos pelo usuário.

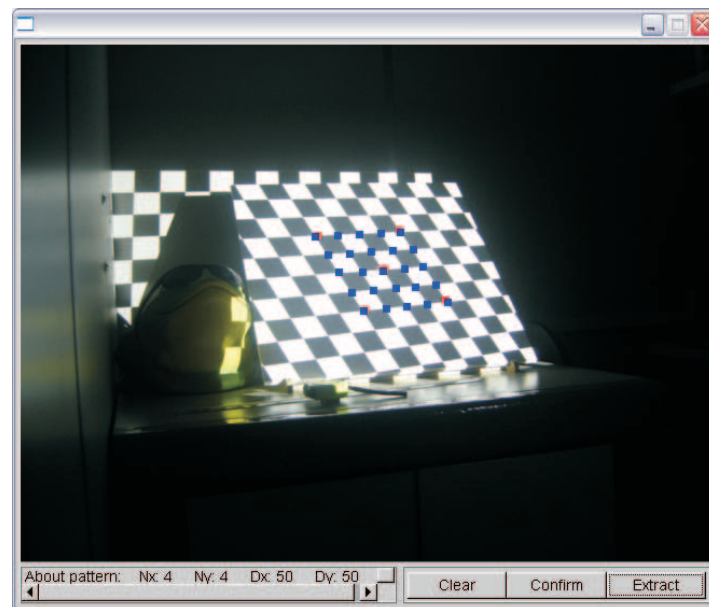


Figura 7.14: Extração dos demais pontos.

A seguir veremos uma calibração completa utilizando a câmera Cyber-Shot modelo DSC-P93A e o mesmo padrão xadrez.

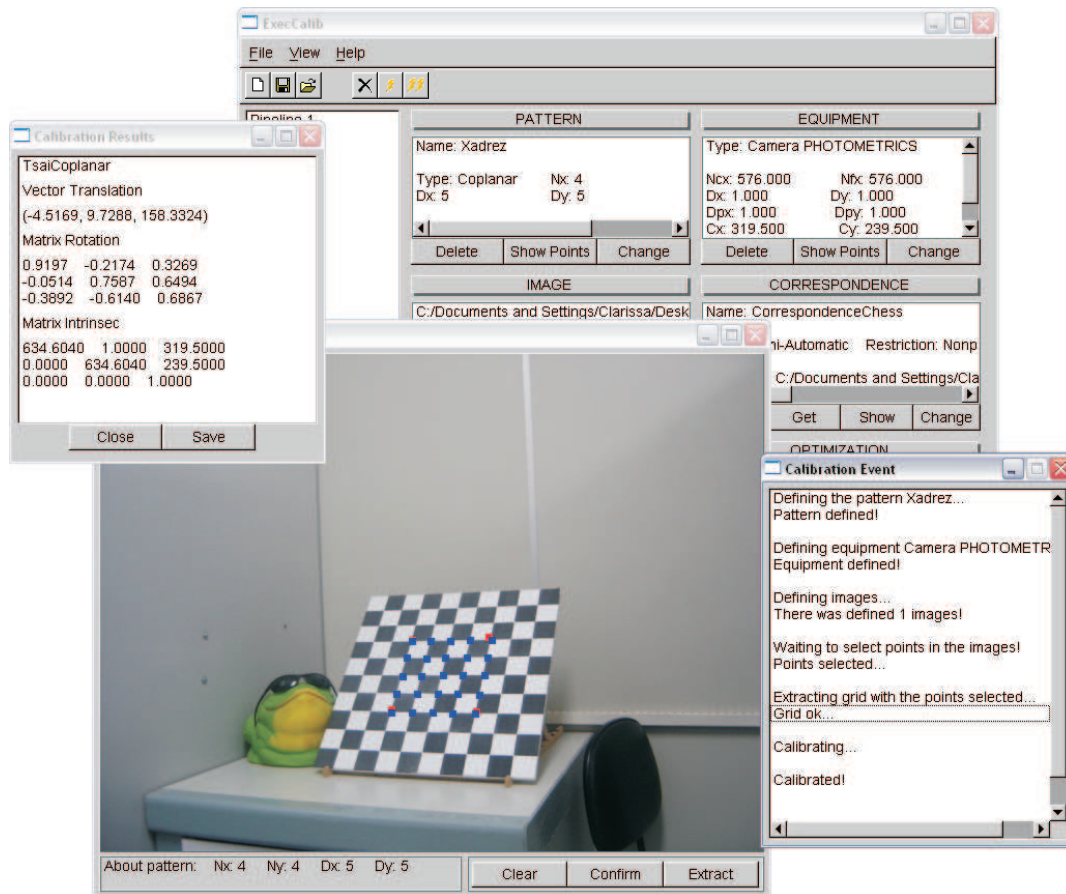


Figura 7.15: Calibração de uma câmera.

Neste mesmo teste calibramos um projetor utilizando os dados de calibração obtidos pela calibração da câmera exibida acima. Os resultados obtidos encontram-se ilustrados a figura 7.16.

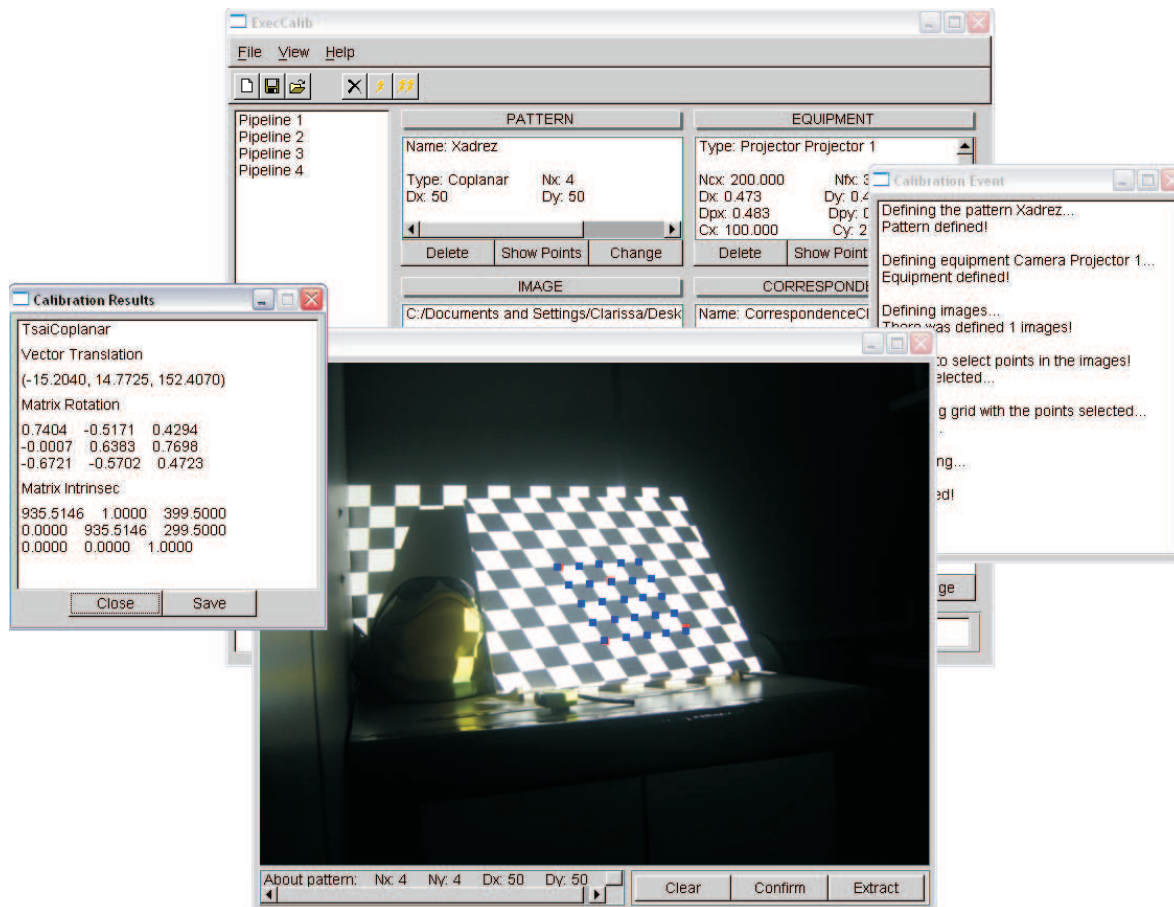


Figura 7.16: Calibração de um projetor.

A figura 7.17 mostra o resultado da execução de dois pipelines que tomam os dados de entrada da etapa de calibração em um arquivo. O método de calibração usado no pipeline 1 foi o Tsai coplanar, e no pipeline 2 o Tsai não coplanar.

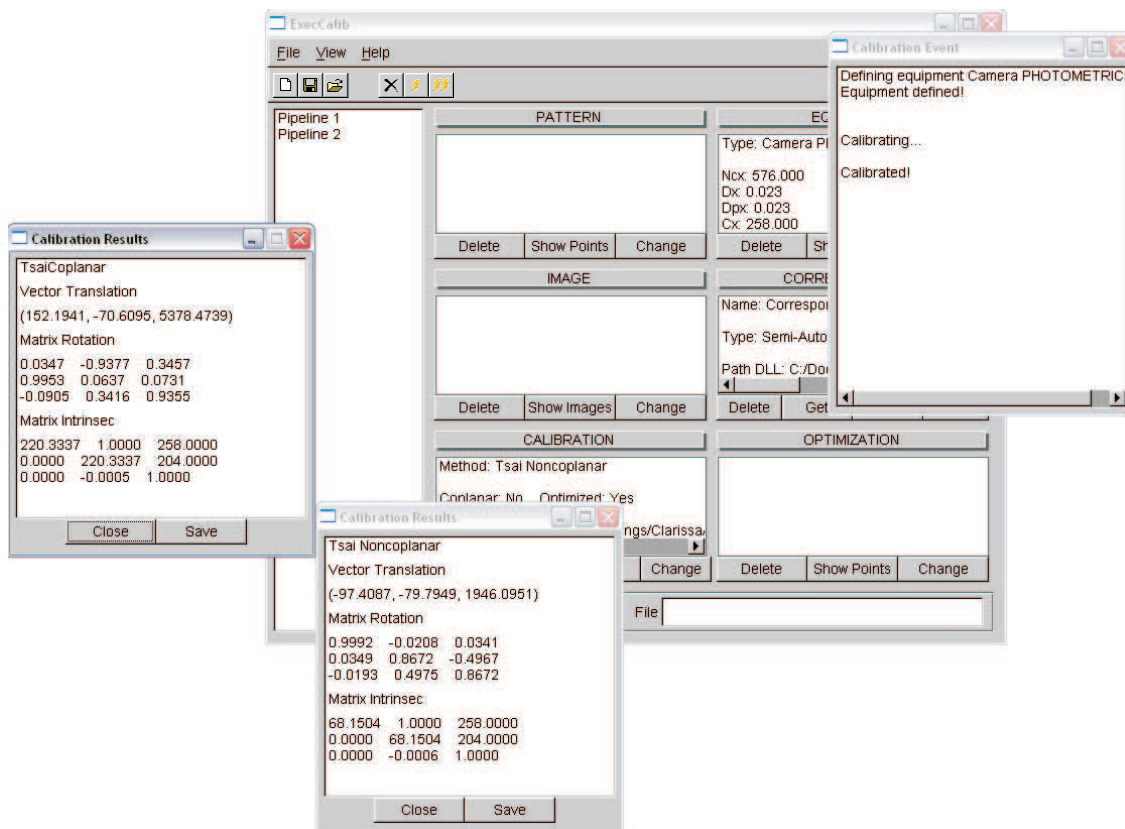


Figura 7.17: Execução de várias pipelines.

A figura 7.18 mostra testes realizados com combinações entre elementos de diferentes pipelines. A figura 7.18 mostra um exemplo de tais testes.

No exemplo citado acima um dos pipelines executa a calibração através de conjuntos de pares de pontos obtidos através de um método de correspondência, e o outro adquiriu os pontos através de um arquivo. Ambos os métodos utilizam o método Tsai Coplanar, com imagens diferentes.

Testes de Calibração

Devido à limitação de tempo não foi possível incluir outros métodos de calibração. Isso impossibilitou a realização de testes comparativos entre as matrizes resultantes da calibração. Estes testes fazem parte dos trabalhos futuros.

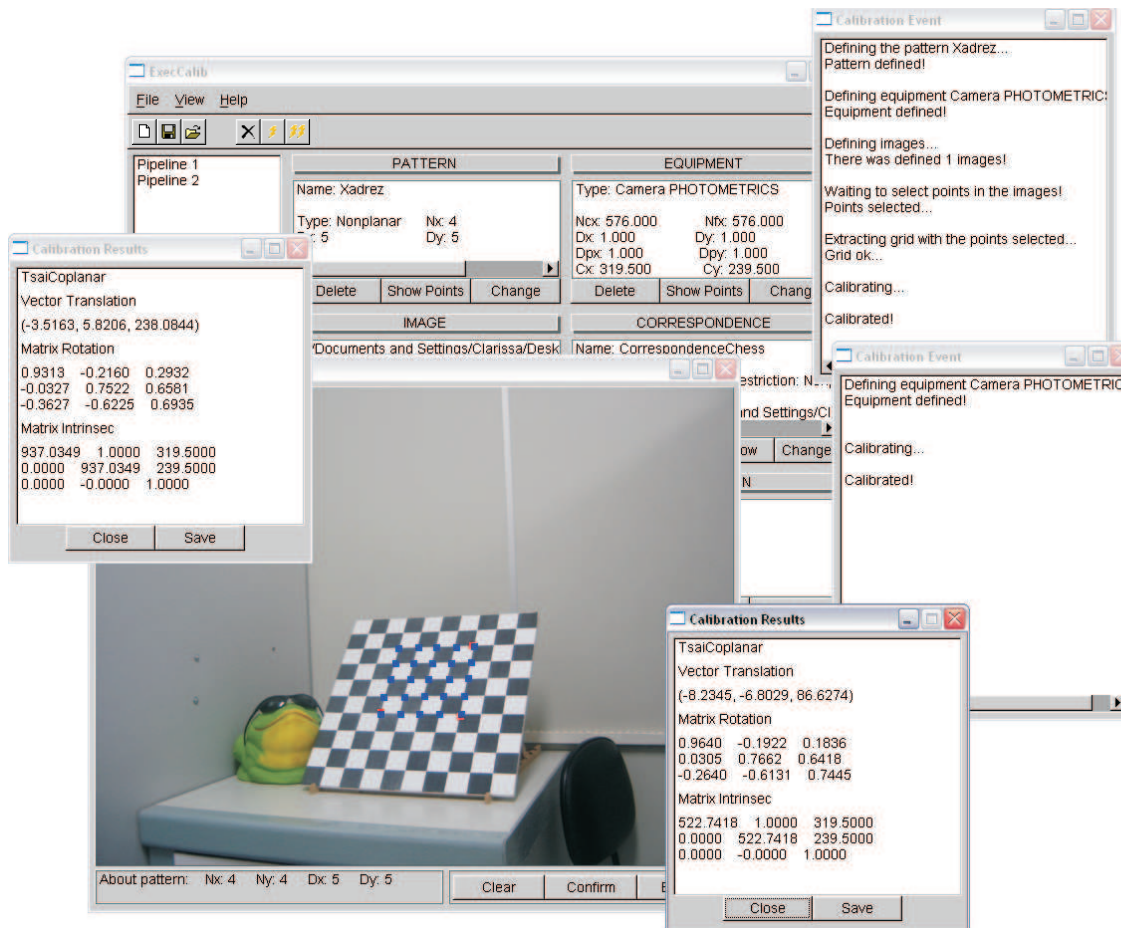


Figura 7.18: Execução de pipelines não-convencionais.

Capítulo 8

Conclusão e Trabalhos Futuros

8.1 Conclusão

Este trabalho propôs uma arquitetura genérica onde fosse possível implementar diversos métodos de calibração de câmera, criando assim a possibilidade de analisar e comparar os resultados de calibração dos diversos métodos.

O sistema desenvolvido teve como principal objetivo dar maior flexibilidade ao usuário para criar diferentes formas de calibração e assim analisar e comparar os resultados. Ele consiste em dois módulos para a edição e execução de pipelines de calibração.

A ferramenta EditCalib permite ao usuário a criação de diversos pipelines de calibração, cujos elementos podem ser totalmente personalizados para cada método. A adição de novos métodos de calibração, correspondência e otimização através de dll's (Dynamically Linked Library) de suas subclasses tornou o sistema mais compacto e simples, apenas com a desvantagem de adaptar os métodos as classes-mães correspondentes e indicar o caminho correto de seus diretórios no sistema.

O módulo ExecCalib executa os pipelines através da leitura de arquivos dos mesmos, gerados pela ferramenta EditCalib. Ainda é possível realizar possíveis trocas de dados da calibração, antes de sua execução, agilizando o processo de testes com novas informações.

O sistema possui uma interface simples de forma a facilitar o uso da ferramenta. Devido a sua flexibilidade, os mais diversos métodos podem ser adicionados ao sistema, apenas adaptando-os às classes presentes no sistema. A diversidade de elementos e de métodos que podem ser aplicados à ferramenta tornam o sistema mais versátil e útil para as aplicações que necessitem de calibração de seus equipamentos.

Para este trabalho foram implementados os métodos de Tsai coplanar e não-coplanar, utilizando padrões xadrez.

8.2 Trabalhos futuros

- Inclusão de novos métodos de calibração, como Zhang e método dos pontos circulares.
- Realização de testes comparando numericamente as matrizes dos diversos métodos de calibração.
- Inclusão de novos métodos de otimização, como o Gloptipoly, permitindo assim uma análise numérica e a comparação da convergência com outros métodos como o Levenberg-Marquardt.
- Estudar formas automáticas de realizar a restrição entre as relações dos elementos.
- Implementar um ambiente 3D que simule a reconstrução dos equipamentos calibrados.
- Implementar testes de projeção dos pontos 3D do padrão virtual, utilizando uma câmera virtual com os mesmos dados da câmera real calibrada. Em seguida, comparar estas projeções com as fotografias originais dos padrões que foram tiradas com a câmera real.

Referências Bibliográficas

- Abdel-Aziz, Y. I. & Karara, H. M. (n.d.), ‘Direct linear transformation object space coordinates in close-range photogrammetry’, *Proc. Symp. Close-Range Photogrammetry* .
- Alvarez, B. (2001), Edição tridimensional de fotografias arquitetônicas, Master’s thesis, Pontifícia Universidade Católica do Rio de Janeiro.
- Carvalho, P. C., Velho, L., Montenegro, A. A., Peixoto, A., Sá, A., Soares, E. & Escriba, L. A. R. (2005), *Fotografia 3D*, Associação Instituto de Matemática Pura e Aplicada, IMPA, Rio de Janeiro.
- Henrion, D. & Lasserre, J.-B. (2002), Gloptipoly: Global optimization over polynomials with matlab and sedumi, Technical report.
- Horn, B. K. (2000), ‘Tsai’s camera calibration method revisited’, <http://www.ai.mit.edu/people/bkph/papers/tsaiexplain.pdf>.
- Kushal, A. M., Bansal, V. & Banerjee, S. (2002), ‘A simple method for interactive 3d reconstruction and camera calibration from a single view’, *Proc. Indian Conference on Computer Vision, Graphics and Image Processing* .
- Lasserre, J.-B. (2001), ‘Global optimization with polynomials and the problem of moment’, *Journal on Control and Optimization* **3**, 796–817.
- Lourakis, M. I. A. (2005), ‘A brief description of the levenberg-marquardt algorithm implemented by levmar’, <http://www.ics.forth.gr/lourakis/levmar/levmar.pdf>.
- Photo 3D* (1999), <http://www.photo3D.com>.
- Ranganathan, A. (2004), ‘The levenberg-marquardt algorithm’, <http://citeseer.ist.psu.edu/638988.html>.
- Schweighofer, M. (2005), ‘Optimization of polynomials on compact semialgebraic sets’, *SIAM J. on Optimization* **15**(3), 805–825.

- Strobl, K., Sepp, W., Fuchs, S., Paredes, C. & Arbter, K. (n.d.), ‘Camera calibration toolbox for matlab’, http://www.vision.caltech.edu/bouguetj/calib_doc/.
- Szenberg, F. (2001), Acompanhamento de Cenas com Calibração Automática de Câmeras, PhD thesis, Pontifícia Universidade Católica do Rio de Janeiro.
- Szenberg, F., Carvalho, P. & Gattass, M. (2001), ‘Automatic camera calibration for image sequences of a football match’, *Proceedings of the Second International Conference on Advances in Pattern Recognition* pp. 301–311.
- Tapper, M., McKerrow, P. J. & Abrantes, J. (2002), ‘Problems encountered in the implementation of tsai’s algorithm for camera calibration’, *Proc.2002 Australasian Conference on Robotics and Automation* .
- Tsai, R. Y. (1987), ‘A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses’, *Ieee Journal Of Robotics And Automation* **RA-3**(4), 323–344.
- Velho, L. & Gomes, J. (2003), *Fundamentos da Computação Gráfica*, Associação Instituto de Matemática Pura e Aplicada, IMPA, Rio de Janeiro.
- Wu, Y., Li, X., Wu, F. & Hu, Z. (2006), ‘Coplanar circles, quasi-affine invariance and calibration’, *Image and Vision Computing* **24**, 319–236.
- Zhang, Z. (1998), A flexible new technique for camera calibration, Technical report, Microsoft Corporation.