INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

Sistema de auto calibração visual para robots do ISePorto, baseado em EKF

JOÃO PEDRO CARVALHO RIBEIRO
Setembro de 2013

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Sistema de auto calibração visual para robots do ISePorto, baseado em EKF

João Pedro Carvalho Ribeiro

Tese submetida no Âmbito do

Mestrado em Engenharia Electrotécnica e de Computadores

Orientador: José Miguel Soares de Almeida (Mestre)

Setembro de 2013

© João Pedro Carvalho Ribeiro, 2013

Resumo

Os sistemas de perceção existentes nos robôs autónomos, hoje em dia, são bastante complexos. A informação dos vários sensores, existentes em diferentes partes do robôs, necessitam de estar relacionados entre si face ao referencial do robô ou do mundo. Para isso, o conhecimento da atitude (posição e rotação) entre os referenciais dos sensores e o referencial do robô é um fator critico para o desempenho do mesmo. O processo de calibração dessas posições e translações é chamado calibração dos parâmetros extrínsecos.

Esta dissertação propõe o desenvolvimento de um método de calibração autónomo para robôs como câmaras direcionais, como é o caso dos robôs da equipa ISePorto. A solução proposta consiste na aquisição de dados da visão, giroscópio e odometria durante uma manobra efetuada pelo robô em torno de um alvo com um padrão conhecido. Esta informação é então processada em conjunto através de um *Extended Kalman Filter* (EKF) onde são estimados necessários para relacionar os sensores existentes no robô em relação ao referencial do mesmo.

Esta solução foi avaliada com recurso a vários testes e os resultados obtidos foram bastante similares aos obtidos pelo método manual, anteriormente utilizado, com um aumento significativo em rapidez e consistência.

Abstract

Autonomous mobile robots perception systems are complex multi-sensors systems. Information from different sensors, placed in different parts of the platforms, need to be related and fused into some representation of the world or robot state. For that, the knowledge of the relative pose (position and rotation) between sensors frames and the platform frame plays a critical role. The process to determine those is called extrinsic calibration.

This paper addresses the development of automatic robot calibration tool for robots with rotating directional cameras, such as the ISePorto team robots. The proposed solution consists on a robot navigating in a path, while acquiring visual information provided by a known target positioned in a global reference frame. This information is then combined with wheel odometry sensors, robot rotative axis encoders and gyro information within an *Extended Kalman Filter* (EKF) framework, that estimates all parameters required for the sensors angles and position determination related to the robot body frame.

We evaluated our solution, by performing several trials and obtaining similar results to the previous used manual calibration procedure, but with a much less time consuming performance and also without being susceptible to human error.

Conteúdo

Resumo **v**

Abstract **vii**

Lista de Figuras **xi**

Lista de Tabelas **xiii**

Lista de símbolos **xv**

Acrónimos **xvii**

1 Introdução **1**

1.1 Âmbito da Tese 1

1.2 Enquadramento e Motivação 2

1.3 Objectivos 3

1.4 Organização da tese 4

2 Estado da Arte **5**

2.1 Flexible Camera Calibration By Viewing a Plane From Unknown Orientations 5

2.2 Camera Calibration from Vanishing Points in Images of Architectual Scenes 6

2.3 Extrinsic Calibration of a Camera and Laser Range Finder 8

3 Caracterização do Problema **11**

3.1 Robô da Equipa de MSL ISePorto 11

3.2 Método de Calibração Manual 14

4 Fundamentos **17**

4.1 Ângulos de Euler 17

4.1.1 Rotações em *roll*, *pitch* e *yaw* 18

4.1.2 *Gimbal Lock* 20

4.2 Fundamentos de Visão 20

4.2.1 Visão Computacional 20

4.2.2 Formação da Imagem 21

4.2.3 Intrínsecos . . . 22
4.2.4 Extrínsecos . . . 25
4.2.5 Extracção de Features . . . 26
4.3 Extended Kalman Filter . . . 26

5 Projeto **29**
5.1 Arquitectura do Sistema . . . 30
5.2 Pré-requisitos do Sistema . . . 32

6 Implementação **35**
6.1 Caracterização do Hardware Utilizado Processo de Calibração . . . 35
6.1.1 Câmara da *Head* . . . 35
6.1.2 Câmara da *Kicker* . . . 36
6.1.3 *Inertial Navigation System* (INS) . . . 37
6.1.4 *Encoders* . . . 38
6.2 Aquisição de Dados . . . 39
6.3 Processamento das Imagens . . . 40
6.3.1 Principais Diferenças Entre os Algoritmos . . . 41
6.3.2 Comparação de resultados . . . 42
6.3.3 Correcção da Ordem dos Pontos do Alvo . . . 43
6.4 Implementação do EKF . . . 44

7 Resultados **49**

8 Conclusões e Trabalho Futuro **57**

Bibliografia **60**

A Maple **61**

Lista de Figuras

1.1 Equipa de futebol robótico IsePorto 3

2.1 Imagens do padrão usadas para calibração das câmaras. 6
2.2 Imagens não calibradas. 6
2.3 Cantos seleccionados pelo utilizador. 7
2.4 Os *vanishing points* representados da imagem. 7
2.5 Representação da rotação e translação entre imagens. 7
2.6 Reconstrução do modelo 3D do cenário. 8
2.7 Um esquema do método de calibração onde podemos ver a câmara, o *Laser Range Finder* (LRF) e o padrão de xadrez. 8
2.8 Imagens do xadrez com a projecção dos pontos do lazer. 9

3.1 Imagem do guarda redes da equipa ISePorto com destaque para a posição das câmaras 12
3.2 Os eixos de rotação existentes num robô da equipa ISePorto 13
3.3 Representação dos referenciais que constituem o robô 13
3.4 Método de calibração antigo. À esquerda o alvo e a *frame* mecânica usados para calibrar os parâmetros extrínsecos e ângulos do robô. À direita o robô alinhado pela linha para calibração do *offset* entre o *kicker* e a *head*. 14

4.1 Representação dos ângulos de Euler no referencial de um avião. 18
4.2 Rotação em torno do eixo dos xx's. 18
4.3 Rotação em torno do eixo dos yy's. 19
4.4 Rotação em torno do eixo dos zz's. 19
4.5 Efeito *Gimbal Lock*. 20
4.6 Modelo *pinhole* 21
4.7 Projecção de retas do mundo na imagem dependendo do tipo de distorção causada pela lente 23
4.8 *RADOCC Camera Calibration Toolbox* 24
4.9 *Imagens Intrínsecos* 24
4.10 *Erro da projeção em pixeis* 25
4.11 Representação dos ângulos de Euler do referencial da câmara. 25

5.1 Manobra efetuada pelo robô para a observação do alvo. Na figura temos o robô e o alvo vistos de cima e a descrição da manobra efetuada pelo robô. 30
5.2 Arquitectura do método de calibração automático. 30
5.3 Imagem do encoder magnético do eixo do *kicker*. 32
5.4 Alinhamento manual dos ângulos do robô usando a *frame* mecânica 33

6.1 Imagem da câmara da *head*. 36
6.2 Imagem da câmara da *kicker*. 37
6.3 Imagem do INS usado na equipa ISePorto. 38
6.4 Imagem dos *encoders* das rodas de tração [1]. 39
6.5 Imagem dos *encoders* dos eixos de rotação. *Kicker* à esquerda e *head* à direita. 39
6.6 Imagem da *gui* da equipa ISePorto, a ipshell. 40
6.7 Imagem do alvo com os cantos detectados pelo algoritmo do OpenCV [2]. 41
6.8 Método de análise de cantos candidatos. 42
6.9 Comparação da detenção de cantos entre os dois algoritmos. 42
6.10 Algoritmo de determinação e correção da ordem dos pontos do alvo. 43

7.1 Imagens do robô a realizar a manobra circular em torno do alvo. 49
7.2 Estimação do *pitch* e posição do robô. 50
7.3 Estimação dos ângulos de Euler da câmara do *kicker*. 50
7.4 Estimação da altura e deslocamentos da câmara do *kicker*. 51
7.5 Estimação dos ângulos de Euler da câmara do *kicker*. 51
7.6 Estimação da altura e deslocamentos da câmara do *kicker*. 52
7.7 Inovação do EKF comparativamente com os limites 2-σ do erro. 52
7.8 Imagens do robô a realizar a manobra simplificada. 53
7.9 Estimação do *pitch* e posição do robô. 53
7.10 Estimação dos ângulos de Euler da câmara do *kicker*. 54
7.11 Estimação da altura e deslocamentos da câmara do *kicker*. 54
7.12 Estimação dos ângulos de Euler da câmara do *kicker*. 55
7.13 Estimação da altura e deslocamentos da câmara do *kicker*. 55
7.14 Inovação do EKF comparativamente com os limites 2-σ do erro. 56

Lista de Tabelas

7.1 Média dos resultados obtidos em 4 calibrações automáticas em comparação com os valores calibração manual para o robô R2 56

Lista de símbolos

ϕ	Roll
θ	Pitch
ψ	Yaw

Acrónimos

API	*Application Programming Interface*
EKF	*Extended Kalman Filter*
FPS	*Frames Per Second*
GPS	*Global Positioning System*
GUI	*Graphical User Interface*
IEEE	*Institute of Electrical and Electronics Engineers*
IMU	*Inertial Measurement Unit*
INS	*Inertial Navigation System*
ISEP	Instituto Superior de Engenharia do Porto
KF	*Kalman Filter*
LIDAR	*Light Detection And Ranging*
LRF	*Laser Range Finder*
LSA	Laboratório de Sistemas Autónomos
MSL	*Middle Size league*
MLE	*Maximum Likelihood Estimation*
PPM	Matriz de Projeção Perspectiva

Capítulo 1

Introdução

1.1 Âmbito da Tese 1

1.2 Enquadramento e Motivação 2

1.3 Objectivos 3

1.4 Organização da tese 4

1.1 Âmbito da Tese

Esta dissertação aborda o desenvolvimento de um sistema de calibração automática de parâmetros extrínsecos[1] para sistemas robóticos cuja aplicação e validação será feita com base na equipa de futebol robótico ISePorto.

Nos dias de hoje os sistemas robóticos estão presentes nas mais diversão áreas desde a indústria, ao ramo automóvel, segurança ou até na medicina desempenhando as mais diversas funções, tais como:

- Medicina (apoio a idosos e deficientes, fabrico de próteses, simulações computacionais, cirurgias delicadas, ...)
- Indústria (transporte de carga, controlo de qualidade, monitorização de máquinas, classificação de produtos, ...)
- Sistemas de navegação e de identificação (identificação de alvos, mapeamento, ...)
- Sistemas de segurança (monitorização militar aérea e terrestre, autenticação de pessoas por reconhecimento facial, monitorização de segurança através da detecção de movimentos, ...)

Um sistema robótico, principalmente se for autónomo, para cumprir estas funções depende em grande escala da informação recolhida pelos seus sensores para poder tomar decisões e atuar de acordo

[1] Estes parâmetros serão abordados com mais detalhe no capítulo 4.2.4

com os seus objetivos. Alguns exemplos de sensores usados atualmente pelos robôs do Laboratório de Sistemas Autónomos (LSA) são:

- Sensores de visão
- Sistemas *Light Detection And Ranging* (LIDAR)
- Sonares de Varrimento Lateral
- Sondas Multi Feixe
- Radares
- INS
- *Global Positioning System* (GPS)
- entre outros...

No entanto, embora munido dos mais variados sensores, um sistema mal calibrado interpretará de forma errada a informação recolhida pelos mesmos devido ao facto de o robô ter uma ideia errada da posição dos sensores no momento em que a informação foi recolhida. Isto significa que, se a bola de jogo estiver exatamente um metro à frente do robô, é possível que o robô veja uma bola diferente com cada uma das câmaras, devido à sua má calibração, e nenhuma delas esteja correta. Esta situação é o equivalente a uma pessoa julgar que está a olhar para norte quando na realidade está a olha para oeste, isto originará que todos os cálculos relativos à sua posição estejam errados devido à noção errada da sua orientação.

Mesmo sendo um processo essencial na grande maioria dos sistemas robóticos, os processos de calibração continuam, ainda hoje, a ser processos maioritariamente manuais, trabalhosos e bastante suscetíveis a erros humanos.

A presente dissertação incide sobre a resolução deste problema, sugerindo um método de calibração automático, recorrendo à informação de imagens recolhidas de um alvo conhecido no mundo em conjunto com a odometria e a medida dos ângulos dos eixos móveis do robô. Esta informação é então processada por um EKF onde serão estimados os parâmetros extrínsecos do robô.

Esta dissertação já deu origem a um artigo cientifico [3] que foi publicado na conferência do *Institute of Electrical and Electronics Engineers* (IEEE) que decorreu durante o Festival Nacional de Robótica 2013.

1.2 Enquadramento e Motivação

O LSA do Instituto Superior de Engenharia do Porto (ISEP), possui, hoje em dia, diversos projetos na área de sistemas autónomo a nível aéreo, aquático e terrestre para aplicações nos mais diversos ambientes de operação. Um ponto comum entre estes sistemas autónomos é o uso de sensores, sendo os sensores de imagem, com por exemplo as câmaras, um dos mais comuns, para possibilitar

a perceção do mundo que os rodeia [4]. A motivação para esta dissertação surgiu da necessidade de automatizar o processo de calibração dos parâmetros extrínsecos dos robôs, de forma a aumentar a velocidade e precisão do mesmo.

O sistema utilizado nesta dissertação para implementação e teste do novo método de calibração foi a equipa de futebol robótico da *Middle Size league* (MSL) IsePorto.

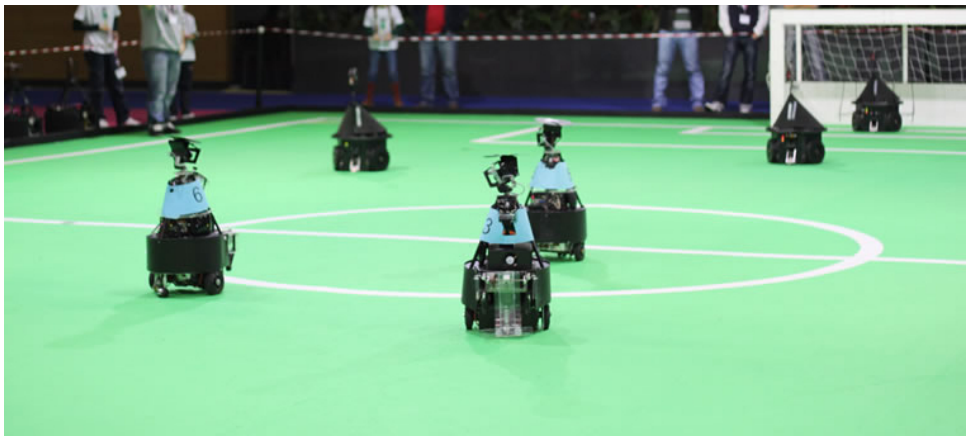


Figura 1.1: Equipa de futebol robótico IsePorto

O futebol robótico é uma área de investigação que tem evoluído nos últimos anos nas áreas de inteligência artificial, cooperação entre agentes e robótica móvel [5].

Esta competição consiste num jogo de futebol entre equipas de cinco robôs completamente autónomos, o que exige que os robôs sejam capazes de executar tarefas e tomar decisões autonomamente, com base na perceção obtida por intermédio dos sensores e da informação partilhada pelos outros robôs da equipa.

Desde o início da equipa ISePorto, os robôs tem evoluído a um ritmo elevado, tanto a nível de software como a nível de hardware, tentando sempre aumentar a performance dos robôs de forma a manter a equipa o mais competitiva possível.

Os robôs da equipa ISePorto são do mais afetados pela ineficiência do processo de calibração atual, isto deve-se à quantidade de câmaras e sensores que estão fixos em eixos rotativos do robô, cujas configurações variam com a rotação desses eixos.

Além disso, enorme contacto físico existente na MSL assim como os remates cada vez mais fortes são capazes de desalinhar a estrutura do robô e mudar a atitude das câmaras fazendo com que robô necessite constantemente de ser calibrado.

1.3 Objectivos

Nesta dissertação é abordada a problemática do processo de calibração num sistema robótico, principalmente em sistemas que necessitam de ser calibrados com alguma frequência devido à características das funções que desempenham. Um exemplo destes sistemas são os robôs da equipa ISePorto, que devido ao constante contacto com robôs adversários vão perdendo a calibração ao longo do tempo.

Assim sendo os objetivos desta dissertação passam desenvolver uma ferramenta de calibração automática que seja capaz de:

- Calibrar os parâmetros extrínsecos do robô;
- Aumentar consideravelmente a rapidez e facilidade de execução do processo;
- Melhorar os resultados obtidos pelo processo de calibração e torna-lo mais consistente e menos suscetível a erros humanos;
- Usar métodos de filtragem discretos que permitam a implementação do processo de calibração em tempo real, como é o caso do EKF, e que permitam, simultâneamente, obter uma caracterização da qualidade da calibração;
- Ser adaptável para um sistema em tempo real que seja capaz de ajustar os parâmetros dinamicamente durante o jogo.

1.4 Organização da tese

Este relatório está organizado em oito capítulos.

No capítulo seguinte são analisados alguns trabalhos existentes a nível de calibração de parâmetros extrínsecos que foram seguidos de perto durante esta dissertação.

No capítulo 3, é efetuado a caracterização e análise do problema.

No capítulo 4 são descritos alguns fundamentos teóricos necessários para uma boa compreensão do trabalho desenvolvido.

O capítulo 5 contém a projeção do trabalho onde foi estruturada a arquitetura do sistema e os procedimentos do mesmo.

Seguidamente, no capítulo 6 é proposta é descrita a implementação desde o hardware usado até à implementação do EKF.

No capítulo 7 são apresentados alguns resultados juntamente com algumas conclusões tiradas sobre os mesmos.

Por fim, no capítulo 8 estão as conclusões e o trabalho a desenvolver no futuro.

Capítulo 2

Estado da Arte

2.1 Flexible Camera Calibration By Viewing a Plane From Unknown Orientations 5

2.2 Camera Calibration from Vanishing Points in Images of Architectual Scenes 6

2.3 Extrinsic Calibration of a Camera and Laser Range Finder 8

O processo de calibração de câmaras é um passo essencial na utilização de visão computacional de forma a ser possível executar com precisão tarefas como *tracking* de objetos, localização de veículos autónomos, medição óptica, entre outros. Assim sendo este tópico tem vindo a ganhar uma grande importância nas últimas décadas, onde têm sido feitos bastantes progressos para melhorar a eficiência e qualidade dos processos de calibração, tanto para os parâmetros intrínsecos como extrínsecos. De seguida serão apresentados alguns destes trabalhos realizados nas últimas décadas que serviram de referência durante a realização desta dissertação.

2.1 Flexible Camera Calibration By Viewing a Plane From Unknown Orientations

No trabalho apresentado por Zhengyou Zhang foi proposto um processo mais flexível para calibração de câmaras [6]. Este processo apenas necessita que a câmara observe um padrão em diferentes orientações (pelo menos duas). O padrão pode ser uma folha imprimida e colada a uma superfície razoavelmente plana, como a capa de um livro por exemplo. Tanto as câmara como o padrão podem ser movidos e as suas posições não precisam de ser conhecidas.

Este método consiste no uso de uma solução de fórmula fechada seguido de uma otimização não linear baseada no algoritmo de *Maximum Likelihood Estimation* (MLE).

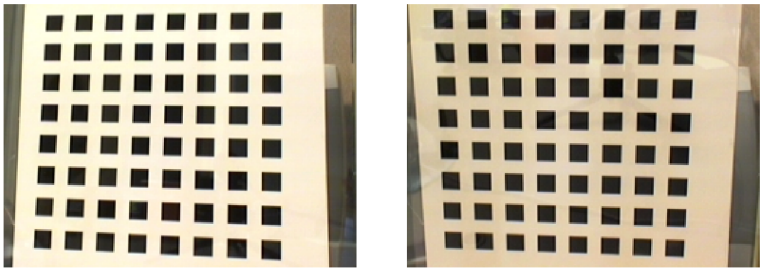


Figura 2.1: Imagens do padrão usadas para calibração das câmaras.

Após a obtenção das imagens do padrão, visto de várias posições, são detectados os pontos de interesse existentes na imagem. Estes pontos serão então usados para calcular os parâmetros intrínsecos da câmara usando a solução de fórmula fechada. Os coeficientes de distorção radial são estimados recorrendo à resolução dos mínimos quadrados. por fim é feita a optimização de todos os parâmetros [7].

Ambos os resultados obtidos, quer por simulação computacional ou através de dados reais, foram bastante satisfatórios comparados com as técnicas clássicas usadas na altura. Devido à sua simplicidade e flexibilidade, este processo aumentou consideravelmente a facilidade de expandir a visão computacional 3D do laboratório para utilização na vida real.

2.2 Camera Calibration from Vanishing Points in Images of Architectual Scenes

Este trabalho propõem um método para recuperar modelos 3D de imagens adquiridas por câmara não calibradas em cenários com padrões arquitectónicos como por exemplo cantos entre paredes e janelas onde a intercepção de linhas rectas é predominante. Este método usa o paralelismo e a ortogonalidade presente na arquitectura actual para obter as matrizes de projecção em cada posição do cenário de modo a poder usar os *vanishing points* para a calibração das câmara [8].

Este método pode ser descrito pelos seguintes passos:

- Recolher imagens do cenário em várias posições diferentes;

Figura 2.2: Imagens não calibradas.

- O utilizador selecciona na imagem uma série de cantos que sejam paralelos ou perpendiculares no mundo;

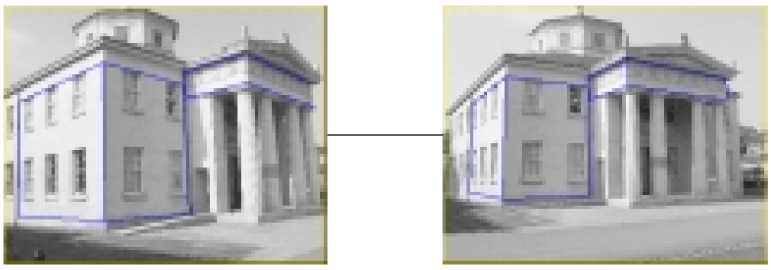


Figura 2.3: Cantos seleccionados pelo utilizador.

- Neste passo é feita a calibração dos parâmetros extrínsecos da câmara para cada imagem. Isto é feito pela determinação dos *vanishing points* associados com as linhas paralelas do mundo. São analisadas três direcções, ortogonais entre si, de forma a obter três parâmetros extrínsecos e a orientação da câmara para cada posição do cenário;

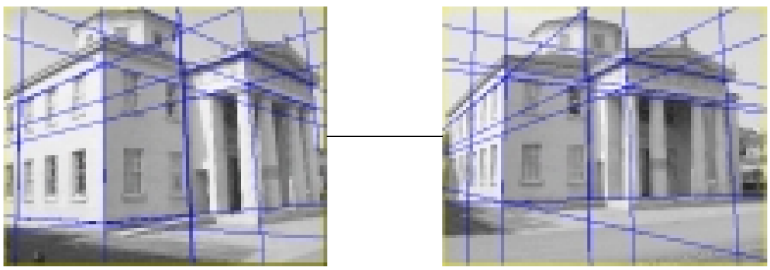


Figura 2.4: Os *vanishing points* representados da imagem.

- A matriz de projecção para cada posição do cenário é calculada com base nos *vanishing points*, isto permite a obtenção da rotação e translação entre imagens através da exploração das constantes existentes na epipolar;

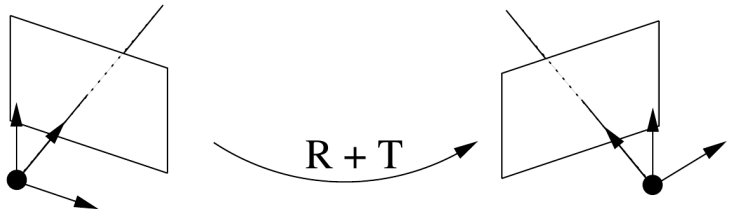


Figura 2.5: Representação da rotação e translação entre imagens.

- O último passo consiste em encontrar correspondências entre as imagens de forma a ser possível calcular o modelo 3D do cenário;

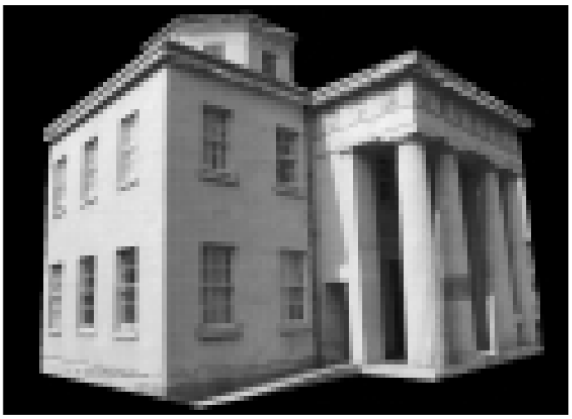


Figura 2.6: Reconstrução do modelo 3D do cenário.

De forma a facilitar o trabalho do utilizador no segundo ponto, foi desenvolvida uma aplicação chamada *PhotoBuilder* para auxiliar todo o processo de escolha dos cantos e reconstrução do modelo 3D. O processo completo demora, em média, menos de quinze minutos.

2.3 Extrinsic Calibration of a Camera and Laser Range Finder

Este trabalho desenvolvido por Qilong Zhang e Robert Pless introduziu um novo método de calibração para plataformas constituídas por uma câmara e um LRF 2D [9]. Este método consiste em observar um padrão em forma de xadrez e resolver as constantes que existem entre a imagem obtida pela câmara e o varrimento obtido pelo LRF. Desta forma o erro algébrico destas constantes é minimizado, o que consequentemente irá minimizar o erro de re-projeção entre os dois sensores.

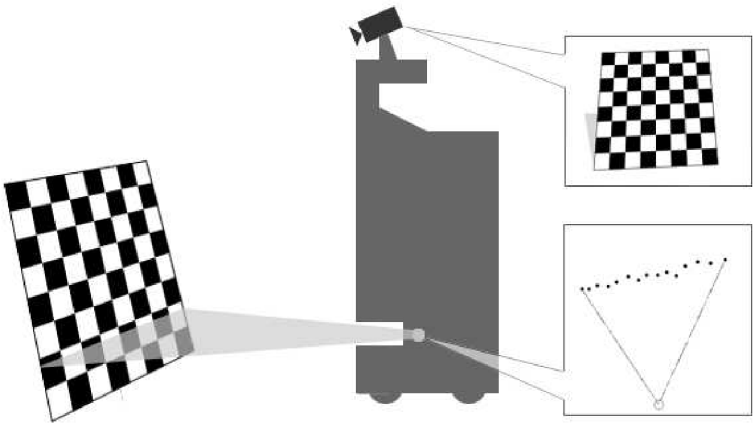


Figura 2.7: Um esquema do método de calibração onde podemos ver a câmara, o LRF e o padrão de xadrez.

De forma a poder utilizar efetivamente os dados recolhidos pela câmara e pelo LRF é importante que a posição de ambos seja conhecida, pelo que essa posição têm influência quando é feita a relação entre os dados dos dois sensores.

A calibração de cada um dos sensores pode ser decomposta em dois tipos de parâmetros, intrínsecos

e extrínsecos. Este método assume que os parâmetros intrínsecos são conhecidos e têm como principal objetivo a calibração dos parâmetros extrínsecos, no entanto, durante o processo de otimização de parâmetros, este método acaba por também fazer pequenas correções nos parâmetros intrínsecos já existentes.

Outro requisito para o uso deste método é que o padrão de xadrez têm que ser observado em simultâneo pelos dois sensores para cada posição do alvo, de forma a ser possível comparar as constantes existentes entre as imagens e os varrimentos. Na figura 2.8 podemos ver imagens do xadrez com os pontos do laser projetados.

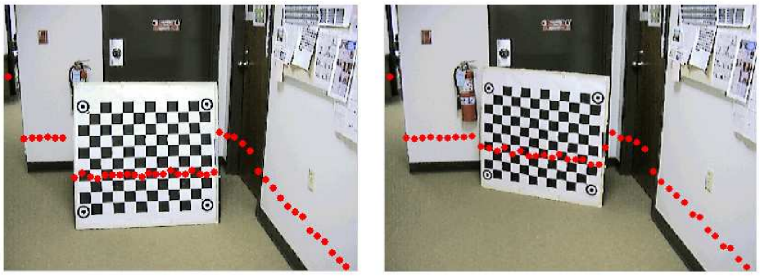


Figura 2.8: Imagens do xadrez com a projecção dos pontos do lazer.

Em modo de sumário, os procedimentos deste método podem ser descritos nos seguintes passos:

- Posicionar um padrão de xadrez numa posição visível pelos dois sensores em diferentes orientações;
- Para cada posição do xadrez, ler os pontos do laser e correlaciona-los com as *features* detectadas na imagem da câmara. Estimar a orientação R_i e posição t_i em relação ao xadrez;
- Estimar a orientação da câmara relativamente ao laser Φ assim como a sua posição relativa Δ;
- Optimizar os parâmetros;

Os resultados apresentados foram bastante satisfatórios, com melhorias na precisão dos parâmetros intrínsecos na ordem dos 30% e diminuição no erro dos parâmetros extrínsecos também na ordem dos 30%.

Capítulo 3

Caracterização do Problema

3.1 Robô da Equipa de MSL ISePorto 11

3.2 Método de Calibração Manual 14

Como foi referido no capítulo 1.2, a equipa ISePorto é uma das plataformas do LSA onde a necessidade de um novo método de calibração era mais evidente. Por este motivo vai ser usada como exemplo durante este capítulo, para caracterização da problemática abordada por esta dissertação.

3.1 Robô da Equipa de MSL ISePorto

Os robôs da equipa ISePorto são robôs complexos que constituem um desafio bastante interessante para a obtenção de uma boa calibração. Isto deve-se ao facto de o robô ser composto por diversos sensores, como por exemplo: duas câmaras para percepção visual, *encoders* de rotação em ambas as rodas e eixos, um sensor *Inertial Measurement Unit* (IMU) de baixo custo e um sensor *laser* (apenas no guarda redes), sendo que estes últimos dois sensores não são calibrado pelo método abordado nesta dissertação [10].

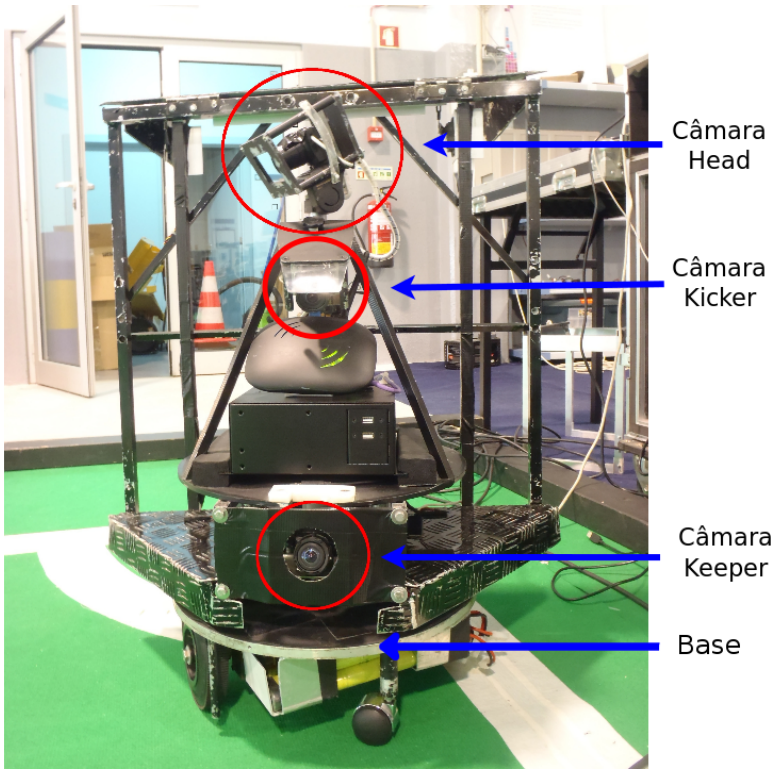


Figura 3.1: Imagem do guarda redes da equipa ISePorto com destaque para a posição das câmaras

Para além dos robôs poderem rodar sobre si próprios de forma a poderem se deslocar para todas as direcções, os robôs do ISePorto possuem mais dois eixos rotativos na zona do *kicker*[1] e na zona da *head*, ver figura 3.2. O eixo do *kicker* permite a rotação de todo o corpo superior do robô em torno da base do robô, facilitando a manobra a bola de forma mais eficiente durante o seu movimento, enquanto que o eixo da *head* permite a rotação da câmara da *head* sobre o corpo superior, para que o robô tenha uma visão completa do campo sem ter que rodar sobre si mesmo.

[1] De forma a manter a nomenclatura usada nas equações do LSA a zona do chutador do robô será tratada por *"kicker"* e parte de cima, que contêm a câmara rotativa, por *"head"*

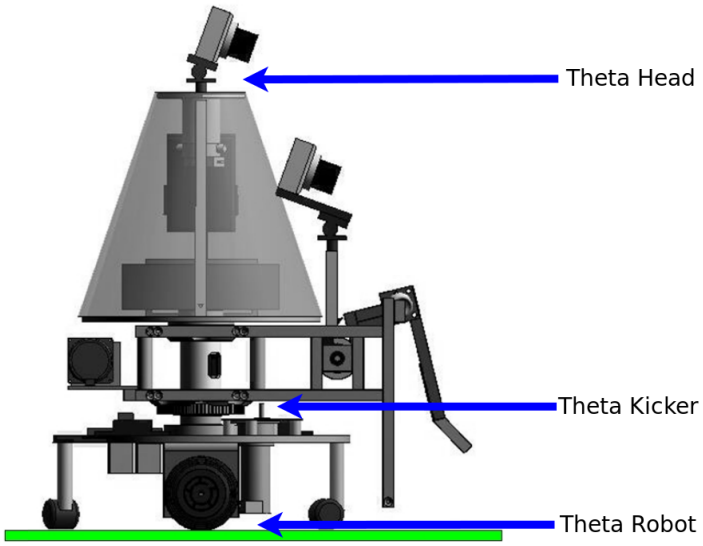


Figura 3.2: Os eixos de rotação existentes num robô da equipa ISePorto

A rotação destes eixos são medidas por *encoders* de rotação com um index que permite servir de ponto de referência para calibração da posição inicial de cada eixo.

Na figura 3.3 é possível visualizar os eixos que constituem o robô.

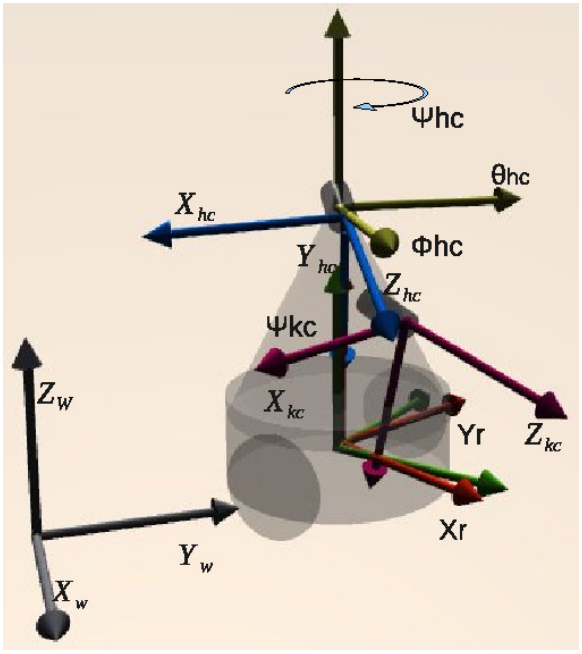


Figura 3.3: Representação dos referenciais que constituem o robô

Representado a cinzento temos o referencial do mundo, a a verde e vermelho respectivamente os referencias do robô e de rotação do *kicker*, a violeta o da câmara do *kicker* e a amarelo e azul respectivamente os referências da rotação e da câmara da *head*.

Os eixos rotativos em conjunto com os referenciais das câmaras fazem com que durante o processo de calibração sejam precisos calibrar 13 parâmetros extrínsecos do robô, esses parâmetros são visíveis na matriz 3.1,

$$\left[\theta_w, h_{kc}, dx_{kc}, dy_{kc}, \psi_{kc}, \theta_{kc}, \phi_{kc}, h_{hc}, dx_{hc}, dy_{hc}, \psi_{hc}, \theta_{hc}, \phi_{hc}\right]^T \tag{3.1}$$

onde θ_w, h_{ck}, dx_{ck}, dy_{ck}, ψ_{ck}, θ_{kc}, ϕ_{kc}, h_{hc}, dx_{hc}, dy_{hc}, ψ_{hc}, θ_{hc} e ϕ_{hc} são respectivamente o *pitch* do robô, altura da câmara do *kicker*, deslocamento em x e y da câmara do *kicker* face ao centro do robô, ângulos de Euler da câmara do *kicker*, altura da câmara da *head*, deslocamento em x e y da câmara da *head* face ao centro do robô, ângulos de Euler da câmara da *head*. O *roll* do robô é considerado constante e muito perto de zero, visto o robô as rodas de tração do robô serem iguais, e não ser possível de mensurar com o *hardware* atual do robô.

3.2 Método de Calibração Manual

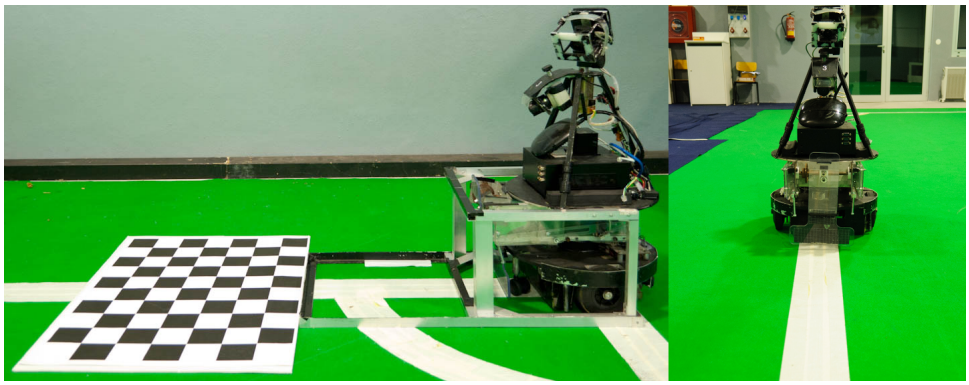


Figura 3.4: Método de calibração antigo. À esquerda o alvo e a *frame* mecânica usados para calibrar os parâmetros extrínsecos e ângulos do robô. À direita o robô alinhado pela linha para calibração do *offset* entre o *kicker* e a *head*.

Como já foi referido anteriormente, o processo de calibração anteriormente usado consistia numa série de procedimentos manuais e demorados que podem ser divididos em três grandes fases:

- Numa primeira fase o robô é colocado numa *frame* mecânica, ver figura 3.4 à esquerda, especialmente desenhada para alinhar todos os eixos rotativos, para que os valores dos *encoders* possam ser lidos com o robô alinhado. O ângulo obtido pelo *encoder* do *kicker* é usado como ângulo inicial para a direcção do *kicker*, este ângulo será usado como referência para a posição de ambas as câmaras (*kicker* e *head*).

- De seguida são recolhidas imagens de um alvo conhecido, numa posição conhecida face ao robô. Essas imagens são então processadas *offline* recorrendo a uma ferramenta para Matlab chamada *Camera Calibration Toolbox* [11] onde será possível determinar os parâmetros extrínsecos das

câmaras relativamente ao alvo. A partir destes valores, e sabendo a posição do alvo, são calculados os parâmetros extrínsecos das câmaras relativamente a cada um dos referenciais móveis do robô.

- O último passo deste processo consiste em determinar o *offset* que relaciona o ângulo de rotação do *kicker* com o ângulo de rotação da *head*. Para estimar este *offset* o robô é alinhado com uma das linhas laterais do campo, ver figura 3.4 à direita, são tiradas medidas da linha do campo vista por ambas as câmaras e é calculado a diferença entre elas através de um processo iterativo.

Os maior problemas que envolvem este método de calibração são a morosidade, entre trinta minutos a duas horas, a sua sensibilidade a erro humano e a sua difícil aprendizagem por parte dos novos elementos da equipa.

Capítulo 4

Fundamentos

4.1 Ângulos de Euler 17
4.1.1 Rotações em *roll*, *pitch* e *yaw* 18
4.1.2 *Gimbal Lock* 20
4.2 Fundamentos de Visão 20
4.2.1 Visão Computacional 20
4.2.2 Formação da Imagem 21
4.2.3 Intrínsecos 22
4.2.4 Extrínsecos 25
4.2.5 Extracção de Features 26
4.3 Extended Kalman Filter 26

Neste capítulo são apresentados alguns dos conceitos e fundamentos necessários à boa compreensão dos capítulos seguintes, nomeadamente, fundamentos associados à representação de rotações, visão computacional e estimação de parâmetros.

4.1 Ângulos de Euler

Os ângulos de Euler providenciam uma forma de representação da orientação 3D de um objeto usando uma combinação de três rotações, *roll* (*phi*), *pitch* (*theta*) e *yaw* (*psi*), sobre os eixos X, Y e Z respetivamente [12].

Os ângulos de Euler são bastante usados a nível de análise e controlo de orientação em objetos devido à sua simplicidade comparado com outras formas de representação como é o caso dos quaterniões. No entanto, os ângulos de Euler possuem uma limitação chamada *"Gimbal Lock"*[1] que impossibilita o seu uso em sistemas cujo ângulo em *pitch* possa atingir $+/-90$ graus.

[1] esta limitação será abordada com mais detalhe no capitulo 4.1.2

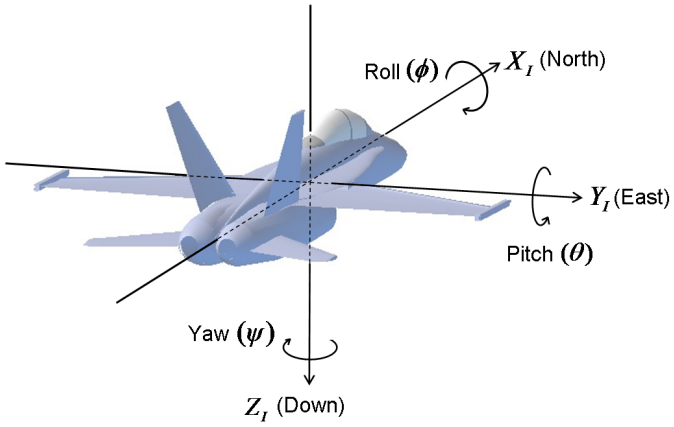


Figura 4.1: Representação dos ângulos de Euler no referencial de um avião.

4.1.1 Rotações em *roll*, *pitch* e *yaw*

A rotação em *roll* é feita através da rotação em torno do eixo dos xx's. Esta é descrita algebricamente pela seguinte relação:

$$R_{(x,\phi)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & -sen\phi \\ 0 & sen\phi & cos\phi \end{bmatrix} \tag{4.1}$$

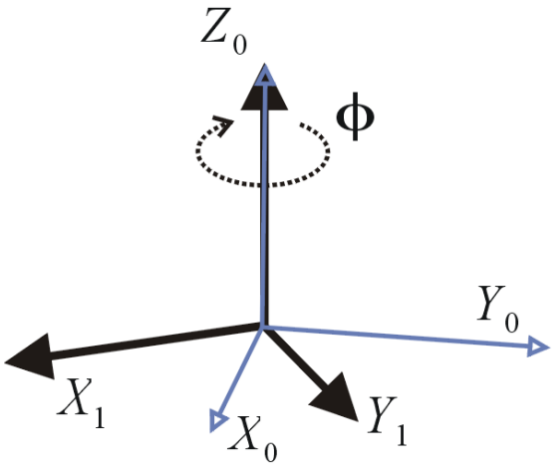


Figura 4.2: Rotação em torno do eixo dos xx's.

A rotação em torno do eixo dos yy's designada por *pitch*, é descrita algebricamente por:

$$R_{(y,\theta)} = \begin{bmatrix} cos\theta & 0 & sen\theta \\ 0 & 1 & 0 \\ -sen\theta & 0 & cos\theta \end{bmatrix} \tag{4.2}$$

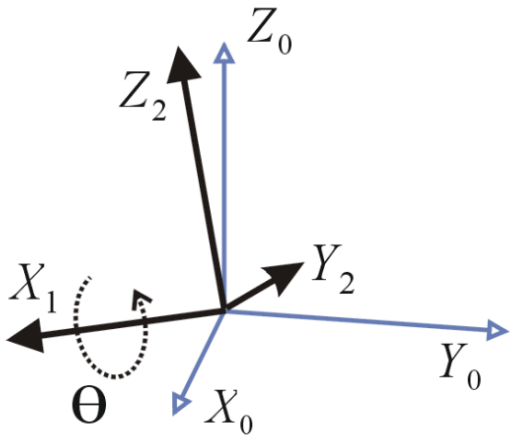


Figura 4.3: Rotação em torno do eixo dos yy's.

Por fim, a rotação no eixo dos zz's designada por yaw, que é representada pela matriz:

$$R_{(z,\psi)} = \begin{bmatrix} cos\psi & -sen\psi & 0 \\ sen\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

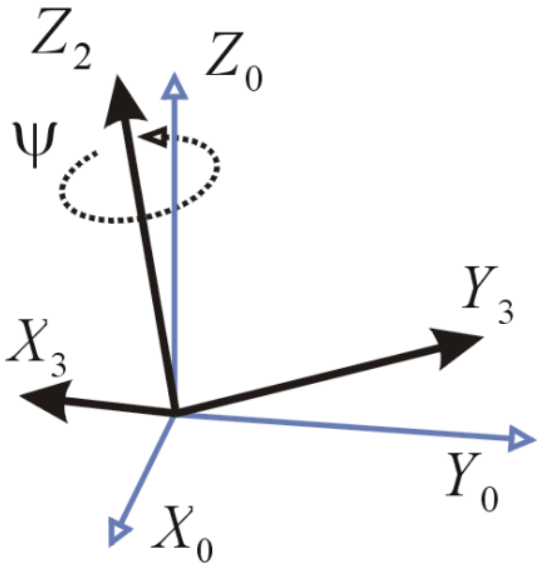


Figura 4.4: Rotação em torno do eixo dos zz's.

A matriz de rotação R em função de ϕ, θ e ψ é representada por:

$$R = R^T_{(x,\phi)} R^T_{(y,\theta)} R^T_{(z,\psi)} = \begin{bmatrix} cos\theta cos\psi & cos\theta sin\psi & -sin\theta \\ cos\psi sin\phi sin\theta - cos\theta sin\psi & sin\psi sin\phi sin\theta + cos\phi cos\psi & sin\phi cos\theta \\ cos\psi sin\theta cos\phi + sin\phi sin\psi & -sin\phi cos\psi + sin\psi sin\theta cos\theta & cos\theta cos\phi \end{bmatrix} \tag{4.4}$$

4.1.2 *Gimbal Lock*

O efeito *Gimbal Lock* ocorre quando a orientação do sensor não pode ser representada unicamente pelos ângulos de Euler, normalmente acontece quando o ângulo de *pitch* atinge $+/-90$ graus.

Figura 4.5: Efeito *Gimbal Lock*.

Este efeito é problemático porque com um ângulo de *pitch* de 90 graus, uma rotação em *yaw* ou *roll* têm exatamente o mesmo efeito no sensor. Olhando para a figura 4.5 facilmente percebemos que esta orientação do avião pode ser obtida por uma rotação em *yaw* e *pitch* ou por uma rotação em *pitch* e *roll*. Esta ambiguidade pode provocar efeitos graves no controlo do sistema.

Este problema só pode ser resolvido pela mudança do método de representação, como é o caso dos quaterniões.

4.2 Fundamentos de Visão

4.2.1 Visão Computacional

Quando falamos em sistemas autónomos, falamos quase sempre em perceção do mundo exterior que rodeia o sistema. Isto acontece porque o resultado dessa perceção vai ter uma grande influencia sobre as decisões que o robô irá tomar durante o desempenho da sua missão.

Para esse efeito existe atualmente uma grande diversidade de sensores que são usados de forma a permitir ao robô conhecer o mundo que o rodeia, como por exemplo sensores de distâncias, sonares,

bumpers e sensores de imagem. Os sensores de imagem são, hoje em dia, provavelmente o sensor mais comum a nível de sistemas autónomos devido à sua relação quantidade de informação / preço. Sendo um sensor relativamente acessível, permite a obter uma grande quantidade de informação a taxas consideráveis sem que seja preciso sacrificar na qualidade dessa mesma informação.

4.2.2 Formação da Imagem

O processo que permite mapear os pontos do mundo na imagem da câmara designa-se de projeção projetiva.

O cálculo da projeção projetiva é efetuado através da modelização do sensor de imagem. O modelo mais comum é designado por *pinhole* e considera que a passagem dos raios de luz é feita através de um pequeno orifício antes de serem projetados no plano da imagem, esta afirmação permite assumir que um ponto tridimensional no mundo corresponde apenas a um único ponto no plano da imagem.

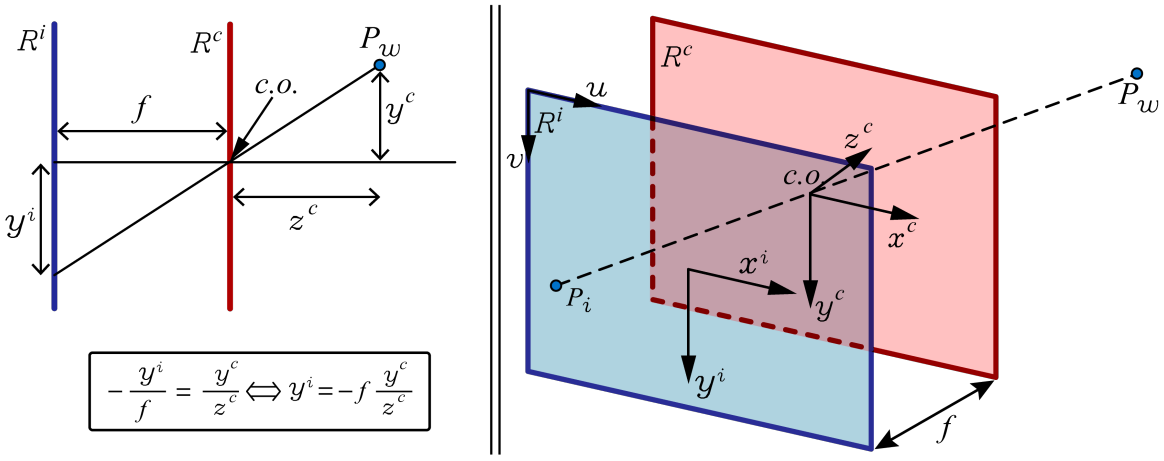


Figura 4.6: Modelo *pinhole*. À esquerda uma apresentação 2D do modelo *pinhole*. À direita a representação tridimensional acompanhada pelos sistemas referenciais da câmara e da imagem.

Na figura 4.6 é possível visualizar a projeção perspetiva de um ponto no plano da imagem R^i através da passagem pelo centro ótico *c.o.*, que se encontra no plano do centro ótico R^c. A distância que separa os dois planos é denominada por distância focal f.

Visto existir uma relação geométrica simples entre os dois planos, a projeção perspetiva pode ser descrita através da equação linear (4.5):

$$\begin{bmatrix} x \\ y \end{bmatrix} = -\frac{f}{Z}\begin{bmatrix} X \\ Y \end{bmatrix} \tag{4.5}$$

onde X, Y, Z, são as coordenadas do ponto no mundo P_w no referencial da câmara. O sinal negativo na equação indica que a imagem formada no plano da imagem está invertida. Este efeito pode ser resolvido invertendo a imagem, para isso (x,y) -> $(-x,-y) = (x_s,y_s)$, o que corresponde a colocar o plano da imagem à frente do centro ótico [13]. Reformulando então a equação (4.6) obtemos:

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \end{bmatrix} \tag{4.6}$$

A visualização de pontos do mundo no referencial da câmara pode ser obtida através de uma transformação composta por uma rotação e translação:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \Big[R|T\Big] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{4.7}$$

Juntando as equações (4.6) e (4.7), podemos relacionar um ponto no mundo com o referencial normalizado da câmara através da Matriz de Projeção Perspectiva (PPM):

$$Z \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = PPM \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{4.8}$$

onde a PPM pode ser representada por:

$$PPM = A \Big[R|T\Big] \tag{4.9}$$

A matriz A depende dos paramentos intrínsecos da câmara, que serão descritos no capítulo 4.2.3, enquanto que as componentes R e T fazem parte dos parâmetros extrínsecos da câmara, que serão tratados no capítulo 4.2.4. Após o cálculo da matriz PPM é então possível mapear pontos do mundo no referencial 2D da imagem através da seguinte relação:

$$m \approx (PPM)M \tag{4.10}$$

4.2.3 Intrínsecos

A matriz A parâmetros intrínsecos de uma câmara servem para relacionar as coordenadas do plano da imagem com o plano da câmara, ou seja relacionar o referencial da imagem (coordenadas em *pixels*) com o referencial da câmara (coordenadas normalizadas).

Estes parâmetros dependem exclusivamente das características físicas da câmara (da sua geometria interna e do tipo de lente), e incluem informações relativas à projeção perspetiva e resolução, alinhamento do sensor e os coeficientes associados à distorção da imagem.

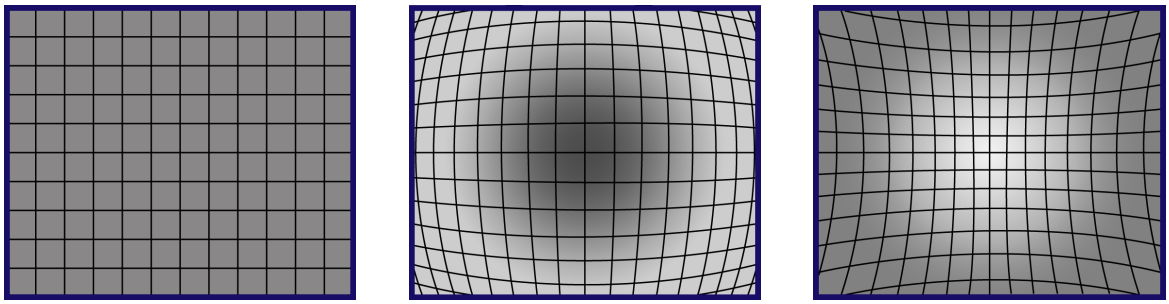
$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.11}$$

$$k = (k_1, k_2, k_3, k_4, k_5) \tag{4.12}$$

onde, f_x e f_y representam as distâncias focais, c_x e c_y as coordenadas em *pixels* do centro ótico da imagem e o vetor k os coeficientes de distorção da imagem.

Distorção da lente

Segundo o modelo pinhole, um ponto no mundo e o seu ponto projetado no plano da imagem seriam colineares. Assim sendo, linha retas no mundo originariam linhas retas na imagem, no entanto isso não acontece. Este efeito é denominado por distorção e é mais visível quando a distância focal da lente é maior ou a sua qualidade é menor [14].



(a) Ausência de distorção

(b) Distorção radial em forma de barril

(c) Distorção radial em forma de almofada

Figura 4.7: Projecção de retas do mundo na imagem dependendo do tipo de distorção causada pela lente

A distorção pode ser descrita segundo duas componentes: tangencial e radial. A distorção tangencial tem como principal origem o desalinhamento físico dos vários elementos que constituem a lente. Nas câmaras modernas a perturbação induzida na imagem por esta componente é mínima, principalmente em lentes com distância focal fixa, pelo que nestes casos pode ser desprezada.

A distorção radial deve-se à variação do ângulo de refração à medida que aumenta a distância ao centro da lente. Quando a refração é menor nos extremos da lente, a imagem sofre distorção radial em forma de barril devido à redução da ampliação com o aumento da distância ao centro. De forma oposta, quando a refração é maior nos extremos, há um aumento da ampliação com o aumento da distância ao centro, produzindo na imagem uma perturbação em forma de almofada.

Calibração dos parâmetros intrinsecos

A obtenção dos parâmetros intrínsecos de uma câmara podem ser obtidos recorrendo à utilização da ferramenta *Toolbox* de Matlab. Podemos ver na figura 4.8 o menu principal da *toolbox* que nos permite fazer, entre outras coisa, a análise do erro das imagens, calibração dos parâmetros e exportação dos mesmos.

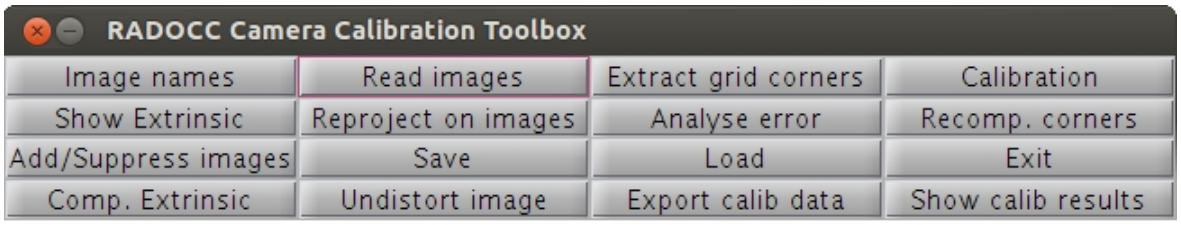


Figura 4.8: RADOCC Camera Calibration Toolbox

O processo de calibração dos parâmetros intrínsecos através da *toolbox* consiste na aquisição de imagens de um alvo com um padrão de dimensões conhecidas, em diferentes posições face à câmara.

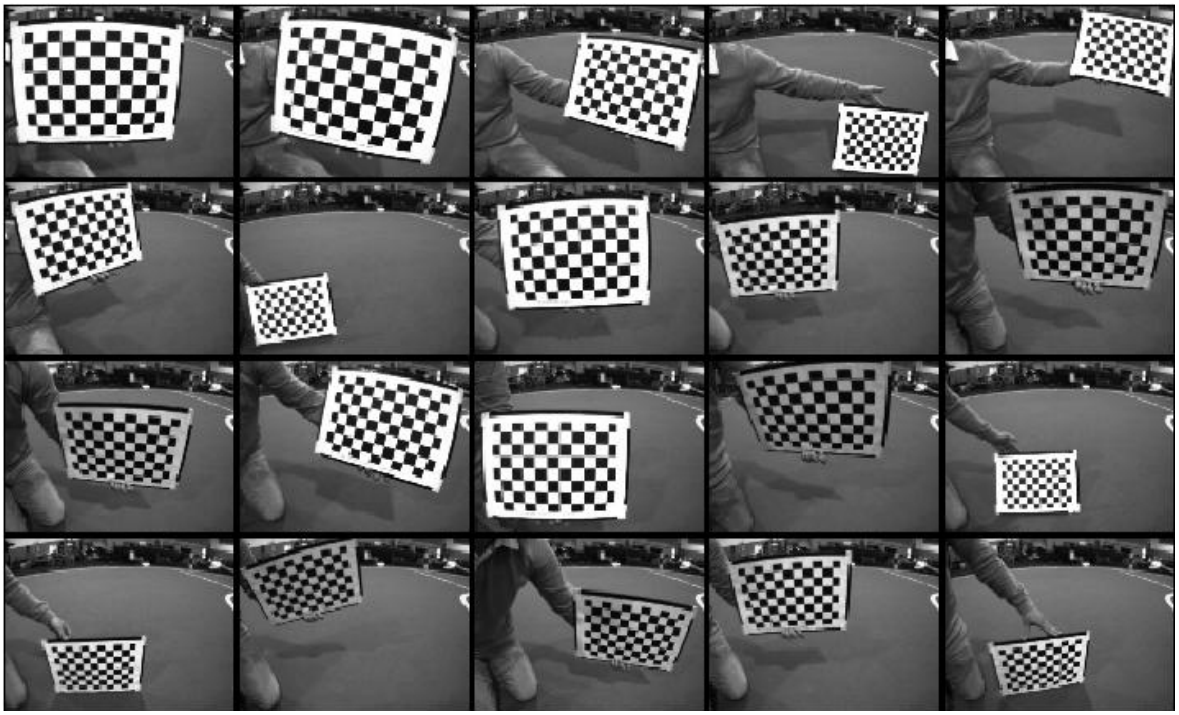


Figura 4.9: Imagens usadas para a calibração dos intrínsecos

Essas imagens são analisadas pela *toolbox*, que recorrendo a processos de optimização, determina o conjunto de parâmetros que optimizam o erro entre a projeção do ponto tridimensional do alvo nas imagens e os pontos detectados pela *toolbox*. Para diminuir o erro é possível remover manualmente as imagens cujo erro seja maior permitindo que imagens que as imagens em pior estado não induzam erro desnecessário nos parâmetros.

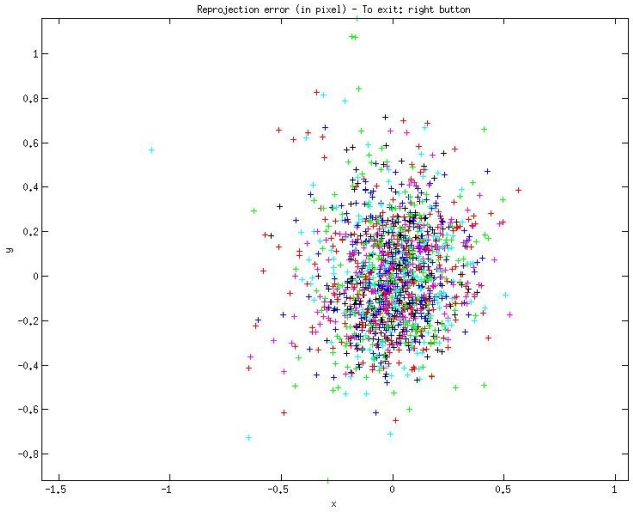


Figura 4.10: Erro da projeção em pixeis

4.2.4 Extrínsecos

Os parâmetros extrínsecos são utilizados para transformar as coordenadas tridimensionais no referencial do mundo para o referencial da câmara [15]. Essa transformação é caracterizada por seis parâmetros: os ângulos de Euler (ϕ, θ e ψ), e as três componentes do vetor de translação T. A matriz de rotação pode ser representada em função de (ϕ, θ e ψ), da seguinte forma:

$$R = R^T_{(x,\phi)} R^T_{(y,\theta)} R^T_{(z,\psi)} \tag{4.13}$$

Uma rotação pura significa que dois dos eixos estão fixos e a rotação acorre apenas em volta do restante eixo [16]. O processo de rotação pura permite obter os três ângulos de rotação: *roll* (ϕ), *pitch* (θ) e *yaw* (ψ), também conhecido por ângulos de *Euler*.

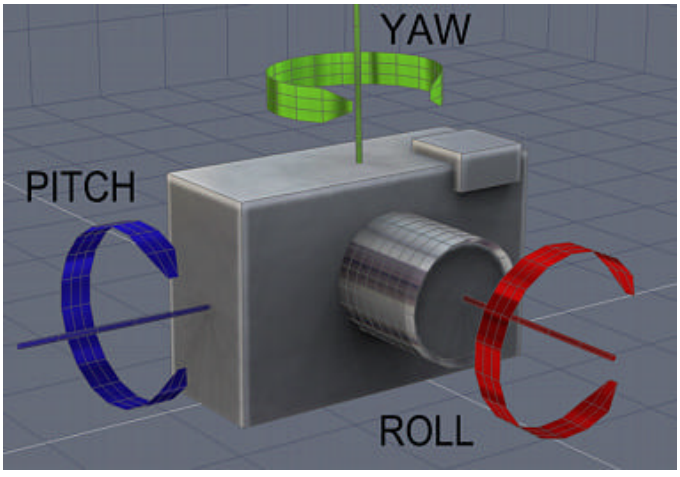


Figura 4.11: Representação dos ângulos de Euler do referencial da câmara.

A matriz de rotação R em função de (ϕ, θ e ψ) é representada por:

$$R = R^T_{(x,\phi)} R^T_{(y,\theta)} R^T_{(z,\psi)} = \begin{bmatrix} cos\theta cos\psi & cos\theta sin\psi & -sin\theta \\ cos\psi sin\phi sin\theta - cos\theta sin\psi & sin\psi sin\phi sin\theta + cos\phi cos\psi & sin\phi cos\theta \\ cos\psi sin\theta cos\phi + sin\phi sin\psi & -sin\phi cos\psi + sin\psi sin\theta cos\theta & cos\theta cos\phi \end{bmatrix} \tag{4.14}$$

E matriz de translação T, que representa o ponto da origem do referencial do mundo no referencial da câmara, é representada por:

$$T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{4.15}$$

4.2.5 Extracção de Features

No contexto da visão computacional, a designação de *feature* é atribuída a uma parcela da imagem que representa uma determinada zona de interesse com determinadas características. Como *feature* podem considerar-se, por exemplo, cantos, fronteiras, linhas, curvas e pixeis com intensidade díspar relativamente à sua vizinhança.

A extracção de *features* é uma tarefa de processamento de imagem de baixo nível que opera de forma densa sobre cada pixel da imagem, de modo a testar a existência de uma particularidade nesse ponto. A repetibilidade é das propriedades mais importantes nos algoritmos de detecção de *features*, porque a maior parte das aplicações de visão computacional depende da identificação das mesmas *features* ao longo de uma sequência de imagens. Por esse motivo, os algoritmos de extracção de *features* mais populares envolvem técnicas para fazer face a alterações de iluminação, perspetiva, escala e rotação da imagem, provocadas quer pela movimentação da câmara quer pela movimentação das próprias *features*, de modo a que estas ocorrências não se reflictam na capacidade de detecção. Em contrapartida, à medida que estes algoritmos vão ficando mais robustos, os requisitos computacionais, para a sua implementação, aumentam. Factor também determinante na escolha de um algoritmo de extracção de *features*.

Para reduzir as necessidades computacionais dos algoritmos de extracção de *features*, a sua maioria opera em imagens em escala de cinza tornando-os mais simples e consequentemente computacionalmente menos exigentes.

4.3 Extended Kalman Filter

O *Kalman Filter* (KF) é um algoritmo de estimação probabilístico bastante usado em sistemas de controlo a nível de engenharia para estimar os estados do processo (mesmo que não sejam mensuráveis diretamente através de sensores) [17, 18].

O Filtro de Kalman é um filtro Gaussiano, ou seja, a representação probabilística da incerteza, nos

estados e nas observações, é representada de forma paramétrica, pela media e covariância. E recorre a modelos matemáticos para modelizar a dinâmica do sistema e os sensores, denominados de modelo do sistema e modelo da observação.

O primeiro, permite a previsão do estado atual com base no estado anterior e nos sinais de controlo aplicados ao sistema, e pode ser representado matematicamente da seguinte forma:

$$x(k) = f(x(k), u(k), v(k)) \tag{4.16}$$

$x(k-1)$ → estado do sistema no instante anterior
$u(k)$ → vector de controlos aplicado ao sistema
$v(k)$ → representa as fontes de ruído no sistema, sendo v(k) N(0, Q(k))
$Q(k)$ → representa a variância do ruído do sistema

O modelo da observação, permite prever qual a observação de um determinado sensor, sabendo qual é o estado atual do sistema:

$$z(k) = h(x(k), w(k)) \tag{4.17}$$

$w(k)$ → representa as fontes de ruído nos sensores, sendo w(k) N(0 , R(k))
$R(k)$ → é a variância das fontes de ruído nos sensores

A nível de arquitetura, o KF é composto por duas fases: prever e corrigir. No primeiro passo, a previsão, é feita uma previsão do próximo estado e da sua covariância com base no estado atual e no modelo do sistema. No segundo passo, a correção, é feita uma comparação entre a mais recente observação por parte dos sensores e uma previsão dessa observação com base no estado previsto, utilizando o modelo da observação. Esta diferença entre a observação e a observação prevista, denominado de inovação na observação, vai ser pesada e utilizada para corrigir o estado atual.

No KF o modelo do processo f e o modelo de observação h são considerados lineares e os seus ruídos w e v como sendo Gaussianos de média zero e não correlacionados:

$$x(k) = F(k)x(k-1) + B(k)u(k) + v(k) \tag{4.18}$$

$$z(k) = H(k)x(k) + w(k) \tag{4.19}$$

$$w(k)\ N(0, Q(k)) \tag{4.20}$$

$$v(k)\ N(0, R(k)) \tag{4.21}$$

Onde $Q(k)$ e $R(k)$ representam respetivamente as incertezas no modelo do sistema e de observação.

O EKF é uma variante do KF que foi desenvolvido mais especificamente para sistemas não lineares. Devido a natureza não linear do EKF, é necessária a linearização das funções f e h, pelo que, é

necessário o uso do Jacobiano dessas mesmas funções:

$$\nabla f_x(k) = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}(k)} \tag{4.22}$$

$$\nabla h_x(k) = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}(k)} \tag{4.23}$$

Desta forma, o passo da previsão pode ser descrito matematicamente através de

$$\hat{x}(k \mid k-1) = f(\hat{x}(k-1 \mid k-1), u(k)) \tag{4.24}$$

$$P(k \mid k-1) = \nabla f_x(k) P(k-1 \mid k-1) \nabla^T f_x(k) + Q(k) \tag{4.25}$$

Enquanto que o passo da correcção é composto por

$$\hat{x}(k \mid k) = \hat{x}(k \mid k-1) + W(k)[z(k) - h(\hat{x}(k \mid k-1))] \tag{4.26}$$

$$P(k \mid k) = P(k \mid k-1) - W(k) S(k) W^T(k) \tag{4.27}$$

onde $\hat{x}$ representa o estado do sistema, o termo $z(k) - h(\hat{x}(k \mid k-1))$ representa a inovação da medida (erro entre a medida observada pelo sensor e a medida prevista), P a covariância do estado sistema, W o ganho de *Kalman* que vai pesar qual a contribuição da inovação para a correção do estado do sistema e S é a covariância da inovação. W e S são calculados através das seguintes equações:

$$W(k) = P(k \mid k-1) \nabla^T h_x(k) S^{-1}(k) \tag{4.28}$$

$$P(k \mid k) = \nabla h_x(k) P(k \mid k-1) \nabla^T h_x(k) + R(k) \tag{4.29}$$

Ao contrário do KF, o EKF não é um filtro óptimo. Isto significa que como os modelos do processo e dos sensores são linearizados em torno do estado previsto para o cálculo da propagação da incerteza, os erros no estado previsto levam a más linearizações dos modelos e consequentemente a erros na estimação das incertezas.

No entanto, na prática, quando usado com cuidado, o EKF pode originar estimações bastante precisas. Isto acontece quando o sistema é passível de ser linearizado com precisão para cada instante.

Capítulo 5

Projeto

5.1 Arquitectura do Sistema 30

5.2 Pré-requisitos do Sistema 32

Nesta fase do projeto, já com a caracterização do problema efetuada, capítulo 3, e com os parâmetros que necessitam de ser estimados bem definidos, equação 3.1, estava tudo pronto para delinear o método em si.

Após a análise do estado da arte actual, em termos de calibração de parâmetros extrínsecos, decidiu-se utilizar uma adaptação do método de Zhengyou Zhang, descrito no capítulo 2.1, em conjunto com um EKF. Esta escolha deste método teve como principais razões os bons resultados que apresentou em conjunto com o facto de não ser necessário nenhum tipo de hardware externo ao já existente no robô. Este método faz uso da visão computacional já existente no robô para extrair informação de um alvo conhecido e assim estimar os parâmetros extrínsecos. A escolha da utilização do EKF foi motivada pela possibilidade implementação, tanto *offline* como *online*.

De forma a possibilitar a observação do alvo em diferentes orientações, foi idealizada uma manobra para ser efectuada pelo robô em torno do alvo, como podemos ver na figura 5.1.

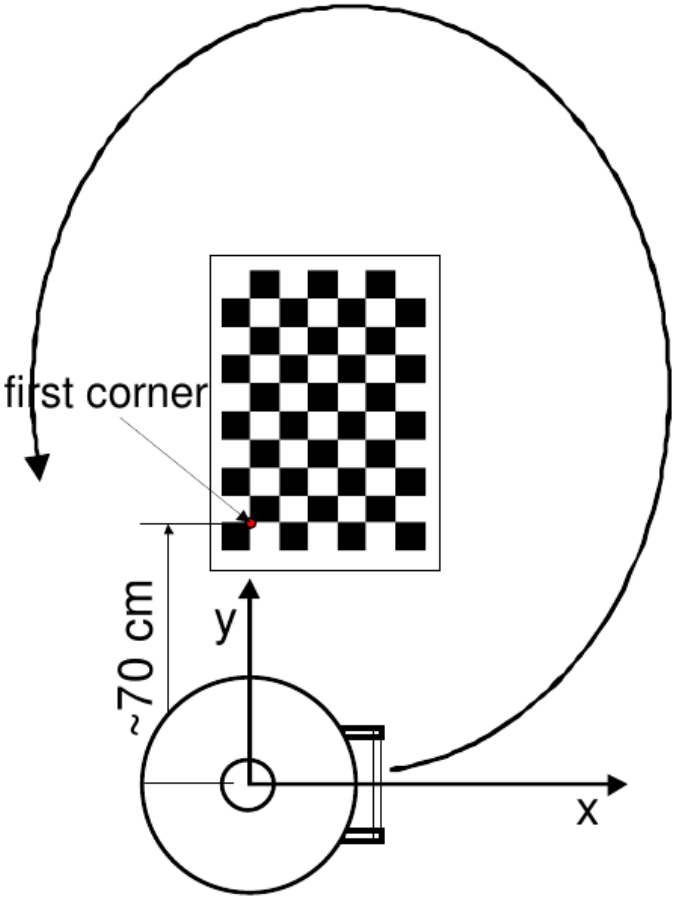


Figura 5.1: Manobra efetuada pelo robô para a observação do alvo. Na figura temos o robô e o alvo vistos de cima e a descrição da manobra efetuada pelo robô.

A aquisição dos dados durante a manobra e o seu processamento por parte do EKF para a estimação dos parâmetros serão abordados mais à frente no capítulo 6.

5.1 Arquitectura do Sistema

Na figura 5.2 está representada a arquitectura do método do calibração.

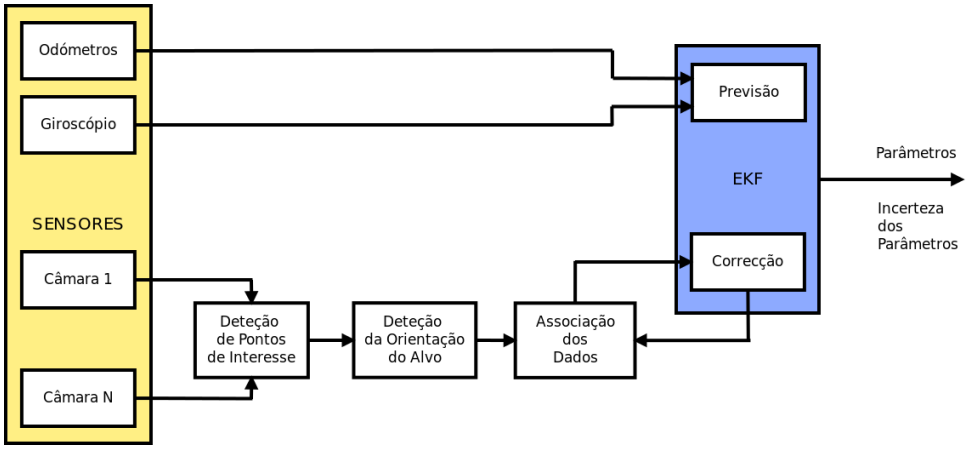


Figura 5.2: Arquitectura do método de calibração automático.

No lado esquerdo podemos ver o sensores presentes no sistema, cuja informação será processada pelo sistema de calibração de forma à obtenção dos parâmetros extrínsecos. A informação da odometria e do giroscópio entra diretamente na fase de previsão do filtro, devido à sua informação ser incremental, e permite-nos estimar a próxima posição no alvo com base na sua posição anterior. Quanto à informação das câmaras, que é usada como observação, na fase de correção do EKF, necessita de ser pré-processada para extrair a informação relevante para este processo.

Este pré-processamento passa pela extração de pontos de interesse da imagem, nomeadamente os pontos do padrão de xadrez usado como alvo, e o cálculo da orientação do alvo [1] para que possa ser calculado a diferença entre o alvo previsto e o alvo observado. Essa diferença entre os pontos do alvo observados e os previstos vão constituir a inovação do EKF, que se estiver dentro dos valores aceitáveis, vai ser passada pelo ganho do EKF e ser usada para corrigir os parâmetros de estado.

O processo de calibração têm uma duração média de cinco minutos e é composto pelos seguintes passos:

- Posicionamento do robô junto do alvo para iniciar o processo de calibração;
- Iniciar a recolha dos dados;
- Efetuar a manobra circular à volta do alvo com o robô;
- Parar a recolha dos dados e extraí-los do robô;
- Executar o EKF no matlab;
- Inserir os dados resultantes no ficheiro de configuração do robô;
- Validar os resultados;

Como podemos ver pela lista de procedimentos anterior, o tempo médio de cinco minutos também inclui a validação dos dados de calibração, pela observação de linhas do campo utilizando a Ipshell. O procedimento que se revelou mais demorado foi a manobra em si, em torno do alvo, devido ao facto de que quanto mais devagar a manobra for efetuada, melhor a nitidez das imagens para a extração dos cantos do xadrez.

Foi também desenvolvida a hipótese de uma manobra simplificada onde o robô ficava parado em frente ao alvo, na mesma posição inicial da manobra circular, e apenas eram rodados para esquerda e direita alternadamente os eixos do *kicker* e da *head* sem que ambas as câmaras perdessem os alvo de vista. Esta manobra permite-nos estimar a atitude e posição das câmaras, assim como o *offset* entre os seus eixos.

O processo de calibração com esta manobra tem exatamente os mesmos passos, mas devido à simplicidade da manobra o tempo é reduzido para cerca de quatro minutos.

[1] será abordada em mais detalhe no capítulo 6.3.3

5.2 Pré-requisitos do Sistema

A nível de pré-requisitos existem apenas dois fatores importantes que devem existir para que este método obtenha os resultados esperados. Estes fatores são a calibração prévia dos parâmetros intrínsecos e o alinhamento mínimo dos eixos do robô.

O primeiro pré-requisito deve-se ao facto de que a previsão dos pontos do alvo, nas imagens obtidas pelas câmaras, utiliza os parâmetros intrínsecos. Sumariando o que foi descrito no capítulo 4.2.3, a ausência de calibração dos parâmetros intrínsecos provoca uma conversão errada das coordenadas do referencial da câmara para o referencial do mundo.

Desta forma, as observações obtidas pelas câmaras estão cheias de erro, provocado pela conversão errada entre referencias, o que resultará em correções erradas por parte de filtro EKF, que irá muito provavelmente divergir.

O segundo pré-requisito surge da necessidade de inicializar o EKF com valor minimamente aceitáveis, isto é, se um ângulo do robô estiver 90 graus para a esquerda e for inicializado com 20 graus para a direita e uma incerteza de apenas 10 graus, será difícil para o filtro estimar corretamente o valor do ângulo.

Na prática, estas situações extremas são raras, pelo que a maior parte das vezes os ângulos do robô estão apenas ligeiramente desalinhados devido aos embates durante os jogos. Mas, como a posição do zero dos *encoders* nem sempre está posicionado para a frente do robô, onde 0 graus significaria que o robô estava alinhado para a frente, quando ocorrem trocas de peças esta situação torna-se preocupante.

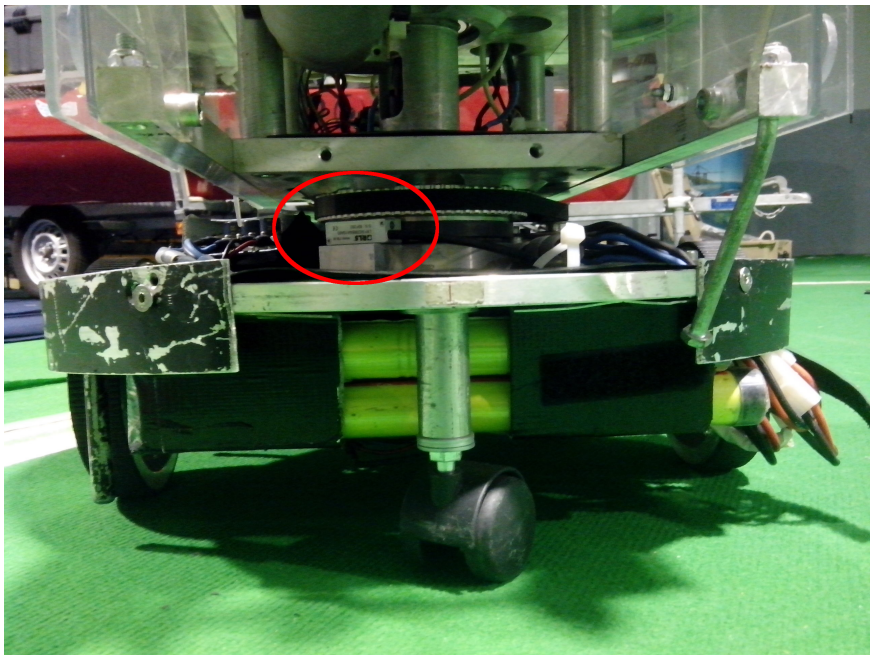


Figura 5.3: Imagem do encoder magnético do eixo do *kicker*.

Como solução para este problema, é conveniente que após a substituição de peças que tenham influência direta nos ângulos do robô seja feita uma pequena calibração manual dos ângulos dos dois

eixos do robô antes de executar o método automático. Esta pequena calibração implica apenas alinhar visualmente os ângulos do robô e inserir os dados obtidos pelos sensores no ficheiro de calibração do robô, assumindo esses valores como uma calibração inicial.

Figura 5.4: Alinhamento manual dos ângulos do robô usando a *frame* mecânica

Desta forma é assegurado que o erro da inicialização no EKF estará dentro dos limites de incerteza.

Capítulo 6

Implementação

6.1 Caracterização do Hardware Utilizado Processo de Calibração 35
6.1.1 Câmara da *Head* 35
6.1.2 Câmara da *Kicker* 36
6.1.3 INS 37
6.1.4 *Encoders* 38
6.2 Aquisição de Dados 39
6.3 Processamento das Imagens 40
6.3.1 Principais Diferenças Entre os Algoritmos 41
6.3.2 Comparação de resultados 42
6.3.3 Correcção da Ordem dos Pontos do Alvo 43
6.4 Implementação do EKF 44

6.1 Caracterização do Hardware Utilizado Processo de Calibração

Como foi mencionado no capítulo 5, este método de calibração utiliza apenas a informação extraída dos sensores já existente nos robôs da equipa ISePorto. Embora esses sensores já tenham sido enumerados no capítulo 3.1, iremos agora ver as especificações de cada um deles com mais detalhe [19].

6.1.1 Câmara da *Head*

A câmara da *head* do robô, usada pelo robô para localização e deteção da bola ou obstáculos no campo, é uma UeYE UI-1225-LE [20], ver figura 6.1.

Figura 6.1: Imagem da câmara da *head*.

Embora todas as câmaras da *head* dos robôs sejam iguais, as lentes que elas usam não o são. No entanto esta diferença não traz qualquer tipo de problema para o método abordado nesta dissertação, pelo que os parâmetros intrínsecos são calibrados individualmente.

A nível de especificações esta câmara é capaz de:

- 752X480 em resolução;
- 60 *Frames Per Second* (FPS);
- Ligação USB2;
- Possibilidade de disparo por *trigger* externo.

6.1.2 Câmara da *Kicker*

A câmara utilizada no *kicker* do robô para visualização da bola durante as fintas ou remates é uma Philips PVC740K, adaptada para uma caixa de câmaras industriais para uso de lentes *cs mount*, ver figura 6.2.

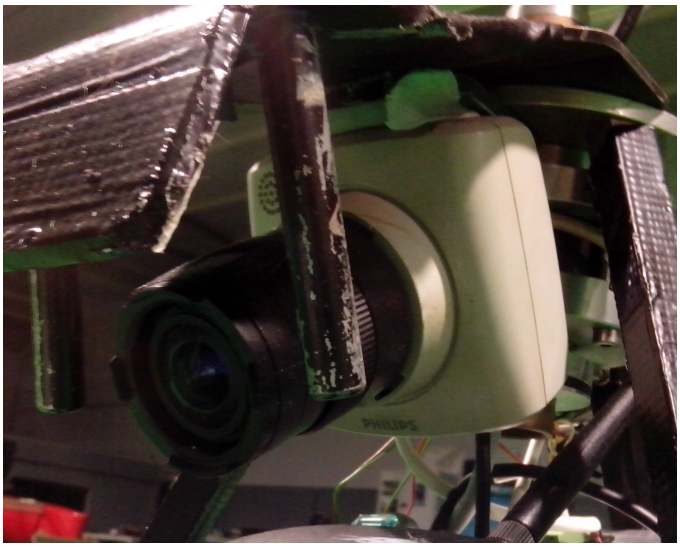


Figura 6.2: Imagem da câmara da *kicker*.

Esta câmara, que já foi no passado também utilizada na *head* dos robôs, é bastante mais antiga que a uEye e consequentemente possui umas especificações um pouco mais fracas:

- 320X240 de resolução a 30 FPS;
- 640X480 de resolução a 20 FPS;
- Ligação USB1;

6.1.3 INS

O INS existente nos robôs da equipa ISePorto têm como funcionalidade auxiliar na localização robô em campo, principalmente ao nível da orientação do robô. O INS é composto por um magnetómetro de três eixos e um giroscópio de um eixo que auxiliam a odometria e a visão a nível do ângulo em Z do robô face à baliza adversária.

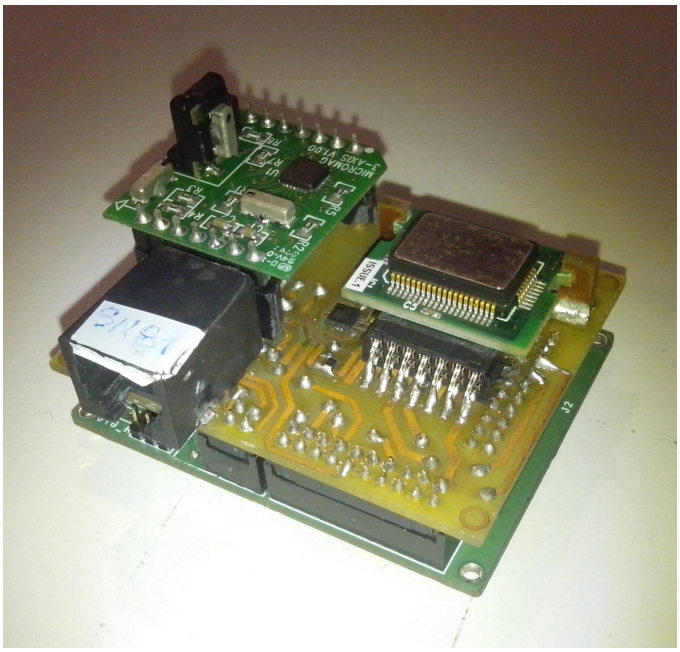


Figura 6.3: Imagem do INS usado na equipa ISePorto.

Este INS foi desenhado e desenvolvido pelo *lsa* e já existe uma nova versão em desenvolvimento que contará com um giroscópio de três eixos e um acelerómetro de três eixos. No entanto, para efeitos desta dissertação, serão listados os componentes do INS actual:

- Microcontrolador ARM STM32F103RET6da da STMicroelectronics;
- Magnetómetro Micromag v3 da PNI;
- Giroscópio CRS10 da Silicon Sensing;
- Acelerómetro MMA8451Q da Freescale.

6.1.4 *Encoders*

Cada robô da equipa ISePorto faz uso de quatro encoders, dois para as rodas, um para a rotação do *kicker* e outro para a rotação da câmara da *head*. Os encoders das rodas são usados pelo robô para o cálculo da odometria. Embora a odometria calculada pelos incremento dos encoders do robô tenha tendência a acumular grandes quantidades de erro ao longo do tempo, quando usada em conjunto com outros sensores, como INS e visão computacional, consegue fornecer informações importantes relativamente ao movimento do robô. Este é um dos motivos que, hoje em dia os encoders são dos sensores mais comuns a nível de sistemas autónomos, ver figura 6.4.

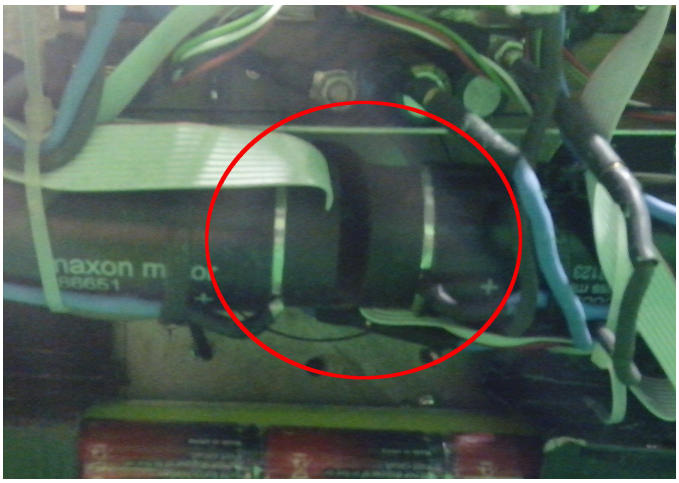


Figura 6.4: Imagem dos *encoders* das rodas de tração [1].

Quanto aos encoders do *kicker* e da *head* são usados pelo robô para saber a cada instante a posição em graus dos seus eixos rotativos face a sua base, figura 6.5.

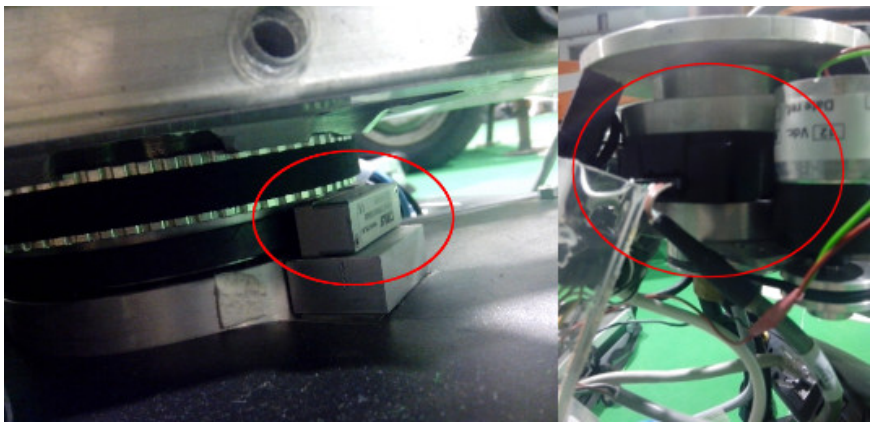


Figura 6.5: Imagem dos *encoders* dos eixos de rotação. *Kicker* à esquerda e *head* à direita.

6.2 Aquisição de Dados

A aquisição dos dados dos sensores durante a manobra de calibração foi implementando com recurso ao módulo de gravação de dados, denominado Infosys, já existente no *software* da equipa ISePorto. Este módulo está já preparado para gravar, a pedido, a informação dos sensores existentes no robô. No entanto, este módulo fornece também uma *Application Programming Interface* (API) que permite a gravação dos dados para novos sensores ou modificar o formato dos dados para os sensores já existentes.

Um dos factores importantes do Infosys é que a gravação dos dados contém o *timestamp* em que o *trigger* para a aquisição dos dados foi dado pelo sistema. O que significa que mesmo que os dados não sejam imediatamente gravados, devido a questões de prioridades da *thread* do Infosys, não existirá erro provocado por esse *delay*.

Isto permite-nos assegurar que o processamento *offline* destes dados seja feito exatamente pela ordem que a medidas foram efetuadas pelos sensores.

Após a gravação dos dados de todos os sensores estar configurada no módulo do Infosys, a gravação pode ser iniciada e parada com recurso à *Graphical User Interface* (GUI) da equipa ISePorto chamada Ipshell, ver figura 6.6.

Figura 6.6: Imagem da *gui* da equipa ISePorto, a ipshell.

Esta aplicação permite-nos criar um registo onde podemos configurar quais os sensores que queremos gravar em simultâneo e criar uma espécie de ação que nos permite iniciar e para a gravação desses sensores facilmente. Os dados obtidos após a gravação ficam num ficheiro de texto onde cada linha corresponde a uma leitura de um determinado sensor, essas linhas estão identificadas com um prefixo que nos permite saber a que sensor corresponde a leitura.

Os dados encontram-se todos no mesmo ficheiro por uma questão de eficiência, pois assim o Infosys apenas têm que lidar com um ficheiro em vês de ter que escrever em vários consoante o sensor em causa. Para separar os dados dos sensores foi desenvolvido um pequeno *script* que separa os dados dos sensores em diferentes ficheiros para que possam ser processados em Matlab pelo EKF.

6.3 Processamento das Imagens

Enquanto que a informação extraída dos outros sensores está normalmente pronta a ser usada, as imagens obtidas pelas câmara necessitam de ser processadas para ser possível extrair a informação necessária. Neste caso em particular é necessário identificar na imagem todos os cantos do padrão de xadrez de forma a obter a posição do alvo no referencial da imagem.

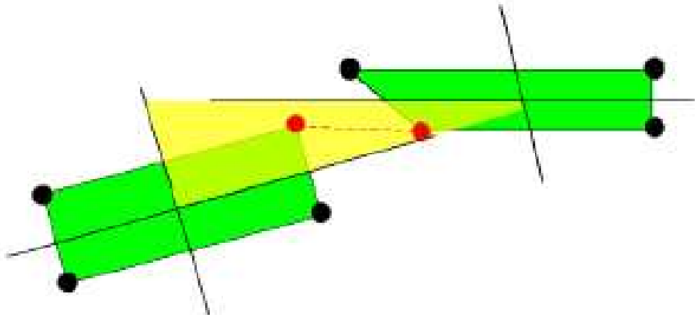


Figura 6.7: Imagem do alvo com os cantos detectados pelo algoritmo do OpenCV [2].

Após alguma pesquisa relacionada com algoritmos de reconhecimentos de padrão foi escolhida uma adaptação do *cvFindChessboardCorners* do *OpenCV* desenvolvido por David Scaramuzza [21]. O *OpenCV*, inicialmente desenvolvido pela *Intel*, é atualmente uma das maiores bibliotecas *open source* a nível de visão computacional.

Esta versão do algoritmo adaptada por David Scaramuzza sofreu uma enorme otimização para câmaras de baixa resolução e imagens de baixa qualidade, que é relevante principalmente na câmara do *kicker* nos robôs do ISePorto.

6.3.1 Principais Diferenças Entre os Algoritmos

Para preservar a forma e melhorar a identificação de todos os quadrados presentes no xadrez, o novo algoritmo implementa um método adaptativo para os *kernels* de erosão. Este método vem colmatar um problema existente com o algoritmo original que durante a fase de erosão usada para destacar os quadrados do xadrez, estes perdiam a forma e ficavam arredondados. O algoritmo de David propõem o uso de *kernels* pequenos, $3x3$, com duas formas diferentes, *"cross"* e *"rect"*. O uso destes dois tipos de *kernels* de forma alternada possibilita a preservação do rácio dos quadrados independentemente das suas orientações.

Outro problema existente no algoritmo original é a ligação dos quadrados com os seus "vizinhos"em imagens de baixa resolução que apresentem alguma distorção. Estas ligações são de extrema importância, pois sendo mal identificadas todos os passos do algoritmo que se seguem ficam comprometidos. Como solução para este problema, esta versão conta com um novo método para encontrar as ligações entre os cantos do quadrados existentes no padrão de xadrez. Esse método é composto por três fases:

- Para cada canto de cada quadrado do xadrez é calculada a distância aos outros cantos à volta. Caso essa distância seja menor que o lado mais pequeno dos dois quadrados envolvidos no cálculo, os dois cantos são considerados candidatos a "vizinhos";

- Para cada um dos cantos candidatos são desenhadas duas linhas no centro dos quadrados respetivos, como é representado na figura 6.8;

- Se ambos os cantos candidatos se encontrarem no mesmo lado de todas as quatro linhas desenhadas, são então considerados um par.

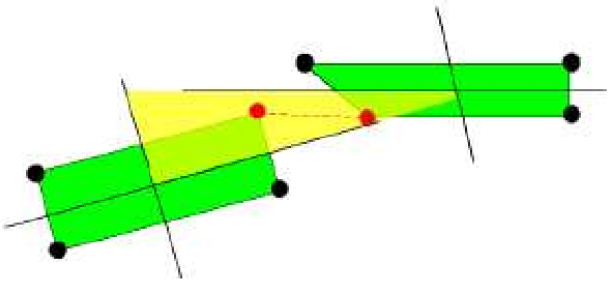


Figura 6.8: Método de análise de cantos candidatos.

Para além destas duas melhorias anteriormente descritas, esta versão de David Scaramuzza dispõem ainda de várias otimizações para o funcionamento do algoritmo em baixas resoluções e na presença de distorção. Outro fator diferenciador é o facto de, ao contrário do algoritmo original, ser possível obter resultados quando apenas alguns dos cantos são detetados. Isto é uma mais valia visto que para efeitos de calibração às vezes a totalidade dos pontos não é um fator obrigatório.

6.3.2 Comparação de resultados

A percentagem de identificação total dos cantos do xadrez com o algoritmo original era apenas de 12% mesmo escalando as imagens até duas vezes o seu tamanho original. Podemos ver na figura 6.9 do lado esquerdo os pontos a vermelhos que não foram detetados.

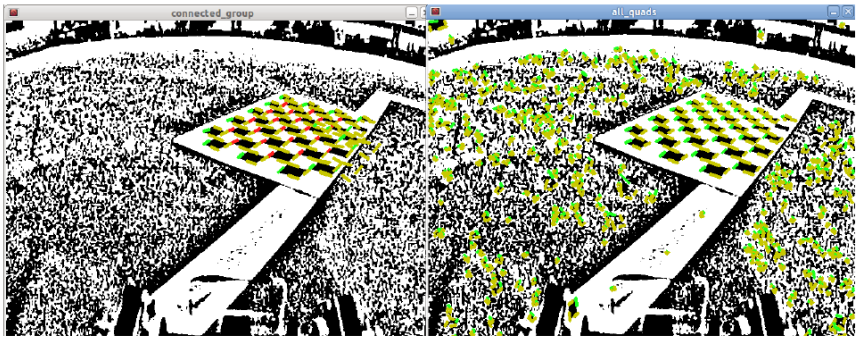


Figura 6.9: Comparação da detenção de cantos entre os dois algoritmos.

Quanto ao algoritmo utilizado os resultados demonstraram-se bastante mais satisfatórios com 33% de identificação total para o tamanho original, 70% para uma imagem escalada duas vezes e 80% para três vezes o tamanho original.

6.3.3 Correcção da Ordem dos Pontos do Alvo

Um problema encontrado em ambos os métodos foi a ordem da sequência dos pontos retornada pela função *cvFindChessboardCorners*, devido ao facto de existirem possíveis sequências e não existir forma de prever a ordem que será retornada em cada leitura.

Esta situação é problemática pelo facto que no EKF a inovação é obtida pela comparação, ponto a ponto, entre o alvo previsto pelo filtro e o alvo observado pelas câmaras. Logo, para a comparação ponto a ponto ser correta ambos os alvos têm de estar com os pontos na mesma ordem. Para resolver este problema foi desenvolvido um algoritmo que deteta a ordem retornada pelo OpenCV e converte os pontos para a ordem desejada.

Numa primeira fase, o algoritmo verifica se os pontos retornados estão no formato $6x9$ ou $9x6$ usando o produto vetorial (6.1), entre os vetores formados pelo primeiro e segundo ponto e pelo primeiro e sétimo ponto. Caso não estejam no formato correto é necessário fazer a transposição dos pontos.

$$\frac{\overrightarrow{V}_{(1,2)}.\overrightarrow{V}_{(1,7)}}{\|\overrightarrow{V}_{(1,2)}\|} > threshold \tag{6.1}$$

De seguida é necessário verificar se pontos estão na direção certa usando um produto vetorial similar (6.2). Se os pontos estiverem na direção errada é necessário espelhar os pontos verticalmente para corrigir a ordem.

$$\perp(\overrightarrow{V}_{(1,2)}).\overrightarrow{V}_{(1,7)} > 0 \tag{6.2}$$

Neste momento, os pontos já estão na posição correta, falta apenas verificar se a ordem está correta ou se é necessário inverter os pontos no *array*. Para avaliar a ordem é necessário usar como referência os pontos previsto pelo EKF para garantir que estão em sintonia.

No lado direito da figura 6.10 podemos ver a correção dos pontos numa ordem errada, linha vermelha, para a ordem esperada, linha verde usando o algoritmo anteriormente descrito.

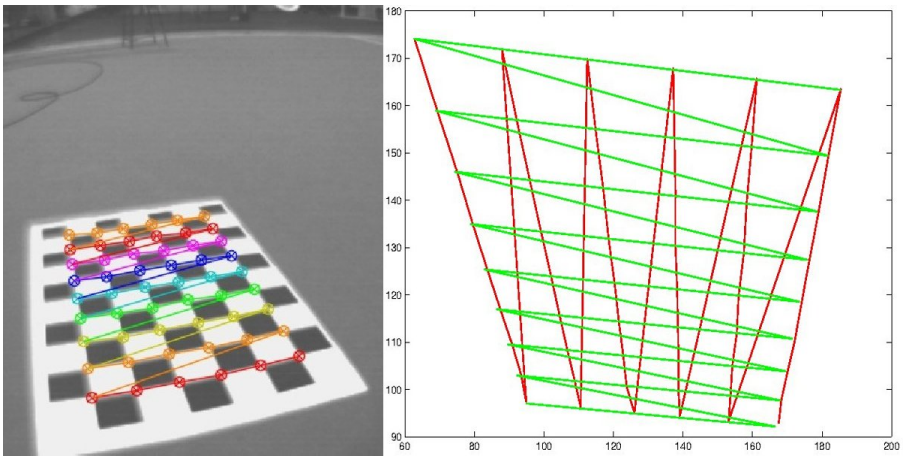


Figura 6.10: Algoritmo de determinação e correção da ordem dos pontos do alvo.

6.4 Implementação do EKF

Para a implementação do estimador, o estado tem que contar com todas as variáveis necessárias para relacionar os pontos do mundo com os pontos na imagem. Deste modo o conjunto de variáveis associadas aos parâmetros extrínsecos das câmaras, vetor (3.1), tem de ser acrescentadas as variáveis de estado do robô no mundo.

$$x = \begin{bmatrix} EstadoRobo & CameraKicker & CameraHead \end{bmatrix}^T \tag{6.3}$$

Sendo o $EstadoRobo$ composto por,

$$EstadoRobo = \begin{bmatrix} x_w & y_w & \phi_w & \theta_w & \phi_{a1} & \phi_{a2} \end{bmatrix} \tag{6.4}$$

Onde, x_w, y_w, ϕ_w, θ_w, ϕ_{a1} e ϕ_{a2} são respetivamente x, y, orientação, *pitch*, ângulo do *kicker* e e o ângulo da *head* do robô. Esta informação é obtida através dos *encoders* e será usada durante a fase de previsão do EKF para prever os pontos do alvo durante o percurso do robô.

$CameraKicker$ e $CamaraHead$ são respetivamente,

$$CamaraKicker = \begin{bmatrix} h_{kc} & d_{x_{kc}} & d_{y_{kc}} & \psi_{kc} & \theta_{kc} & \phi_{kc} \end{bmatrix} \tag{6.5}$$

$$CamaraHead = \begin{bmatrix} h_{hc} & d_{x_{hc}} & d_{y_{hc}} & \psi_{hc} & \theta_{hc} & \phi_{hc} \end{bmatrix} \tag{6.6}$$

Onde, $CamaraKicker$ e $CamaraHead$ são os parâmetros extrínsecos da câmara em relação aos referencias de rotação do *kicker* e *head*, e são constituídos pela altura h_*, distância ao eixo de rotação d_x* e d_y*, e três ângulos ψ_*, θ_*, ϕ_* que definem a rotação em relação à *frame* da câmara.

Na figura 3.3, apresentada no capítulo 3.1, é possível ter uma melhor perceção dos referenciais existentes no robô.

O modelo do processo do sistema:

$$x(k) = f(x(k-1), u(k)) + v(k) \tag{6.7}$$

$x(k)$ $\rightarrow$ vetor de estado

f $\rightarrow$ função de transição de estado

$u(k)$ $\rightarrow$ controlos (*inputs*)

$v(k)$ $\rightarrow$ o ruído do processo é considerado ser Gaussiano com média 0 e covariância Q

$$f = \begin{bmatrix} x_w + ds * cos(\phi_w + d\phi_w) & y_w + ds * sin(\phi_w + d\phi_w) & \phi_w + d\phi_w & \theta_w \\ \phi_{a1} + d\phi_{a1} & \phi_{a2} + d\phi_{a2} & h_{ck} \\ d_{ck} & \psi_{ck} & \theta_{ck} \\ \phi_{ck} & h_{ch} & d_{ch} \\ \psi_{ch} & \theta_{ch} & \phi_{ch} \end{bmatrix}^T \tag{6.8}$$

Onde ds é o deslocamento linear e $d\phi_w$ é o incremento angular vindos da odometria entre os instantes t_{k-1} e t_k.

O modelo de observação,

$$z(k) = h(x(k), w(k), P_w) \tag{6.9}$$

- $z \rightarrow$ medida do sensor
- $h \rightarrow$ função do modelo de observação
- $w \rightarrow$ o ruído da observação, que é assumido como uma distribuição Gaussiana com média nula e covariância R, onde as fontes de ruído são consideradas independentes, logo R é uma matriz diagonal
- $P_w \rightarrow$ pontos do mundo

vai permitir relacionar um conjunto de pontos do mundo com medidas em coordenadas da imagem, correspondentes à projeção de pontos no do mundo no plano da imagem. Deste modo é possível relacionar observações de um objeto na imagem com a previsão dessas observações para um determinado estado previsto.

De forma a ser possível trabalhar com todos os diferentes eixos do robô é necessário primeiro formular as suas relações. Essas relações são caracterizadas pela matriz de transformação que serão apresentadas de seguida.

As matrizes (6.10) e (6.11) permitem relacionar o referencial do robô e o referencial do mundo.

$$R_r^w = \begin{bmatrix} \cos(\psi_r) & -\sin(\psi_r) & 0 \\ \sin(\psi_r) & \cos(\psi_r) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.10}$$

$$T_r^w = \begin{bmatrix} \cos(\psi_r) & -\sin(\psi_r) & 0 & Px_r \\ \sin(\psi_r) & \cos(\psi_r) & 0 & Py_r \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.11}$$

Para melhorar a tração dos robôs da equipa ISePorto, eles dispõem de uns pequenos graus de liberdade no *pitch* da base na direção das rodas loucas. Por este facto os referenciais do robô e da sua base não podem ser considerados coincidentes.

$$R_b = R\psi_b \cdot R\theta_b \cdot R\phi_b \tag{6.12}$$

As matrizes (6.13) e (6.14) representam a rotação e transformação do referencial *kicker* face à base do robô.

$$R_k^b = \begin{bmatrix} \cos(\psi_k) & -\sin(\psi_k) & 0 \\ \sin(\psi_k) & \cos(\psi_k) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.13}$$

$$T_k^b = \begin{bmatrix} \cos(\psi_k) & -\sin(\psi_k) & 0 & 0 \\ \sin(\psi_k) & \cos(\psi_k) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.14}$$

Para relacionar a câmara do *kicker* com o referencial do robô precisamos apenas de relacionar a câmara *kicker* com o referencial do próprio *kicker*.

$$R_{kc}^k = R\psi_{kc}^k \cdot R\theta_{kc}^k \cdot R\phi_{kc}^k \tag{6.15}$$

$$T_{kc}^k = \begin{bmatrix} & & & kickerdist_x \\ & R_{kc}^k & & kickerdist_y \\ & & & h_{kc} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.16}$$

$$T_{kc}^w = T_r^w \cdot T_b^r \cdot T_k^b \cdot T_{kc}^k \tag{6.17}$$

Com estas transformações definidas e conhecendo os parâmetros intrínsecos da câmara já podemos calcular a projeção na imagem dos pontos no referencial do mundo.

Assim a matriz de projeção perspectiva fica:

$$PPMkicker = A \cdot T_{kc}^{w\,-1} \tag{6.18}$$

O mesmo processo é necessário para obtermos a projeção dos pontos vistos pela câmara da *head*. As matrizes (6.19) e (6.20) representam a rotação e transformação do *head* face à base do robô.

$$R_h^b = \begin{bmatrix} \cos(\psi_h + h_{offset}) & -\sin(\psi_h + h_{offset}) & 0 \\ \sin(\psi_h + h_{offset}) & \cos(\psi_h + h_{offset}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.19}$$

$$T_h^b = \begin{bmatrix} \cos(\psi_h + h_{offset}) & -\sin(\psi_h + h_{offset}) & 0 & 0 \\ \sin(\psi_h + h_{offset}) & \cos(\psi_h + h_{offset}) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.20}$$

A relação da câmara da *head* com o referencial do *head* é dada pela seguinte transformação.

$$R_{hc}^h = R\psi_{hc}^h \cdot R\theta_{hc}^h \cdot R\phi_{hc}^h \tag{6.21}$$

$$T_{hc}^h = \begin{bmatrix} & & & headdist_x \\ & R_{hc}^h & & headdist_y \\ & & & h_{hc} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.22}$$

$$T_{hc}^w = T_r^w \cdot T_b^r \cdot T_k^b \cdot T_h^k \cdot T_{hc}^h \tag{6.23}$$

A matriz de projeção perspectiva que relaciona os pontos do mundo com os pontos na imagem da *head*, pode ser obtida através da equação (6.24).

$$PPMhead = A \cdot T_{hc}^{w\,-1} \tag{6.24}$$

Assim, podemos projetar pontos do referencial do mundo nos referenciais das câmaras, em coordenadas homogéneas, através da seguinte equação.

$$\begin{bmatrix} P_{camx} \\ P_{camy} \\ P_{camz} \end{bmatrix} = PPM_{cam}(x).Pw \tag{6.25}$$

E para prever os ponto 2D do mundo nas imagens das câmaras usando o estado do robô e os parâmetros estimados, a equação (6.26) pode ser usada.

$$P_{img} = \begin{bmatrix} x_{imgi} \\ y_{imgi} \end{bmatrix} = h'(x, Pw) = \begin{bmatrix} \frac{Pcam_x}{Pcam_z} \\ \frac{Pcam_y}{Pcam_z} \end{bmatrix} \tag{6.26}$$

Assim, o modelo de observação h, que permite prever todos os pontos do xadrez no referencial da imagem, com base no estado do filtro (estado do robô e parâmetros extrínsecos estimados) é dado pela seguinte equação:

$$\hat{z} = \begin{bmatrix} P_{img1} \\ \vdots \\ P_{imgn} \end{bmatrix} = h\left(x, \begin{bmatrix} P_{chess1} \\ \vdots \\ P_{chessn} \end{bmatrix}\right) = \begin{bmatrix} h'(x, P_{chess1}) \\ \vdots \\ h'(x, P_{chessn}) \end{bmatrix} \tag{6.27}$$

O Jacobiano para o modelo de observação é obtido usando a seguinte equação (6.28).

$$\nabla_x h\left(x, \begin{bmatrix} P_{chess1} \\ \vdots \\ P_{chessn} \end{bmatrix}\right) \tag{6.28}$$

Todas as equações e matrizes necessárias para o EKF assim como as funções matlab para o cálculo dos Jacobianos foram geradas com recurso à ferramenta Maple. No apêndice A é possível visualizar o ficheiro Maple desenvolvido durante esta dissertação.

Capítulo 7

Resultados

Várias experiências foram efetuadas para avaliar o desempenho do método de calibração automático proposto. Estas experiências consistiram na realização da manobra descrita na figura 5.1, com os vários robôs da equipa ISePorto, recolhendo a informação necessária para ser processada com recurso ao EKF desenvolvido.

Na figura 7.1 é possível visualizar imagens do robô durante a manobra de calibração.

Figura 7.1: Imagens do robô a realizar a manobra circular em torno do alvo.

Os resultados seguintes foram obtidos pelo EKF durante os testes de calibração.

Na figura 7.2 é possível observar a posição estimada pelo método automático em comparação com a posição do robô obtida apenas dos incrementos da odometria. Ambas as trajetórias são bastante similares mas quando comparados com as marcas da trajetória, desenhadas no chão do campo para servir de *ground truth*, a posição prevista apresenta uma maior semelhança. Na parte inferior da figura é apresentação a estimação do ângulo *pitch* do robô, estimado pelo EKF com base nas observações do alvo durante a manobra. É importante salientar que os robôs da equipa ISePorto tem duas rodas loucas, à frente e a trás, com alguma folga base ao chão de forma a aumentar o atrito nas rodas laterais dos robôs e diminuir as derrapagens nas mudanças de direção. Esta inclinação era considerada nula pelo método de calibração manual e é agora estimada no método automático de forma a diminuir o erro por ela provocado.

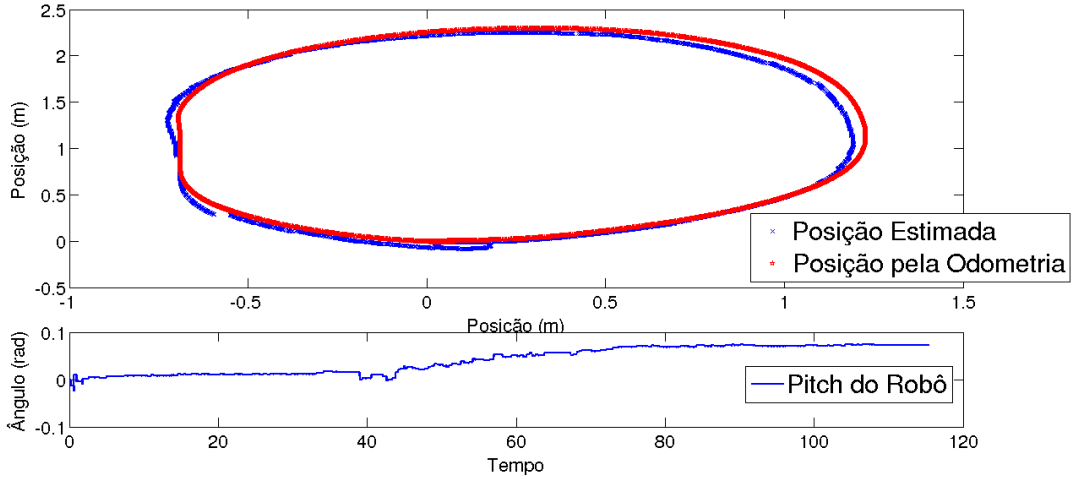


Figura 7.2: Estimação do *pitch* e posição do robô.

Na figura 7.3 é possível visualizar a estimação atitude para a câmara do *kicker*, representada pelos ângulos de Euler, e as suas covariâncias durante a manobra.

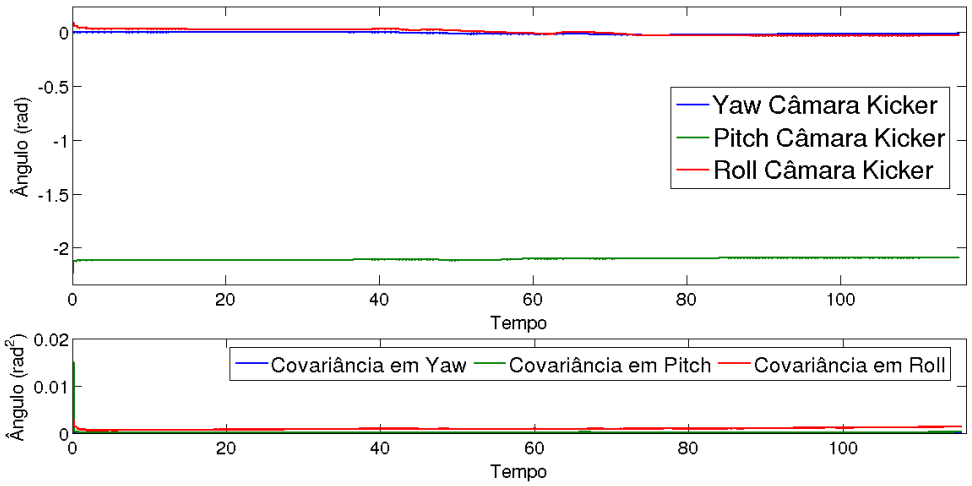


Figura 7.3: Estimação dos ângulos de Euler da câmara do *kicker*.

A altura e as distâncias ao eixo da câmara do *kicker*, bem como as suas covariâncias, podem ser visualizadas na figura 7.4.

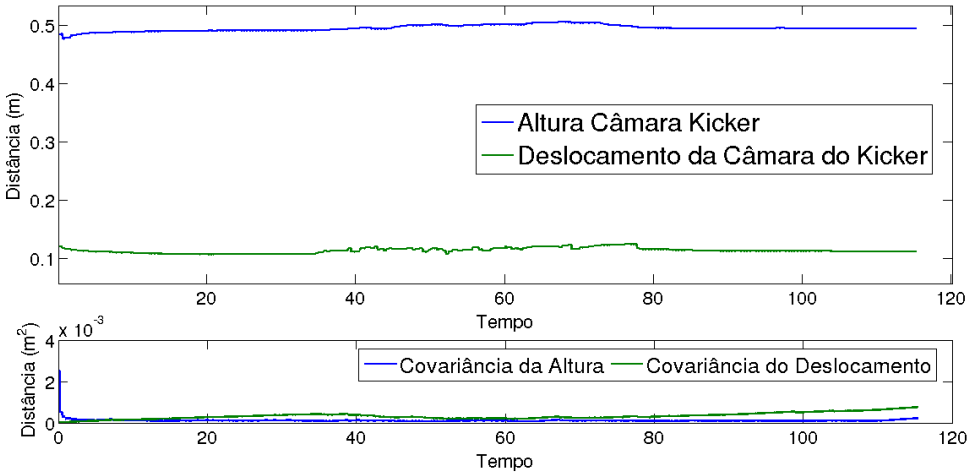


Figura 7.4: Estimação da altura e deslocamentos da câmara do *kicker*.

Da mesma forma, a atitude, altura e distâncias ao eixo da câmara do *head* e as suas covariâncias podem ser visualizadas nas figuras 7.5 e 7.6.

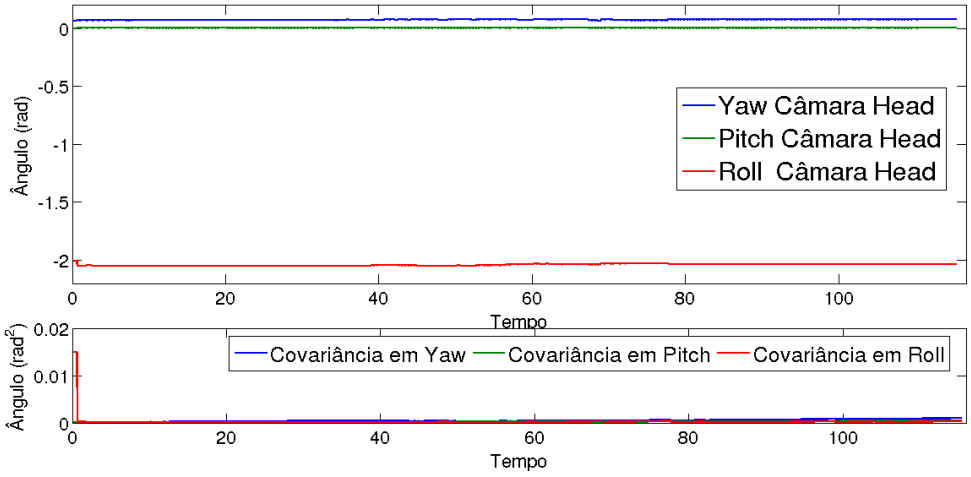


Figura 7.5: Estimação dos ângulos de Euler da câmara do *kicker*.

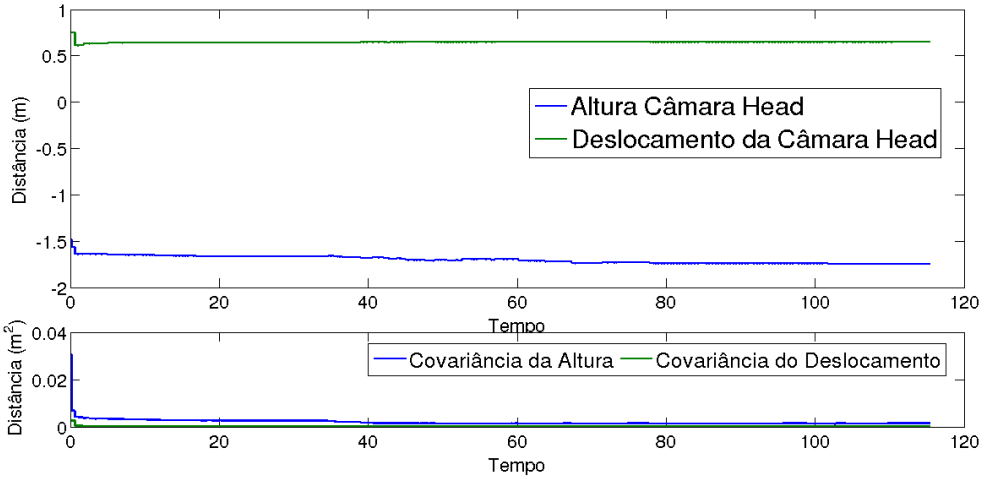


Figura 7.6: Estimação da altura e deslocamentos da câmara do *kicker*.

É possível observar que a convergência da grande maioria dos parâmetros estimados pelo EKF é bastante rápida e após as primeiras *frames* os valores mantêm-se mais ou menos constantes ao longo do tempo. Isto acontece porque os valores iniciais do filtro estão já muito próximos dos valores reais que cada robô terá e pretende-se apenas calibrar os desfasamentos provocados pelas diferentes peças entre robôs e os erros de montagem. No entanto, estes pequenos erros, que por vezes não passam de dois ou três graus, podem provocar erros de quase meio metro nas distâncias medidas pelo robô.

Para finalizar, como forma de demonstrar a inovação do EKF, a figura 7.7 demonstra a inovação em x e y dos pontos extraídos do xadrez em comparação com os limites 2-σ da covariância da inovação durante a manobra.

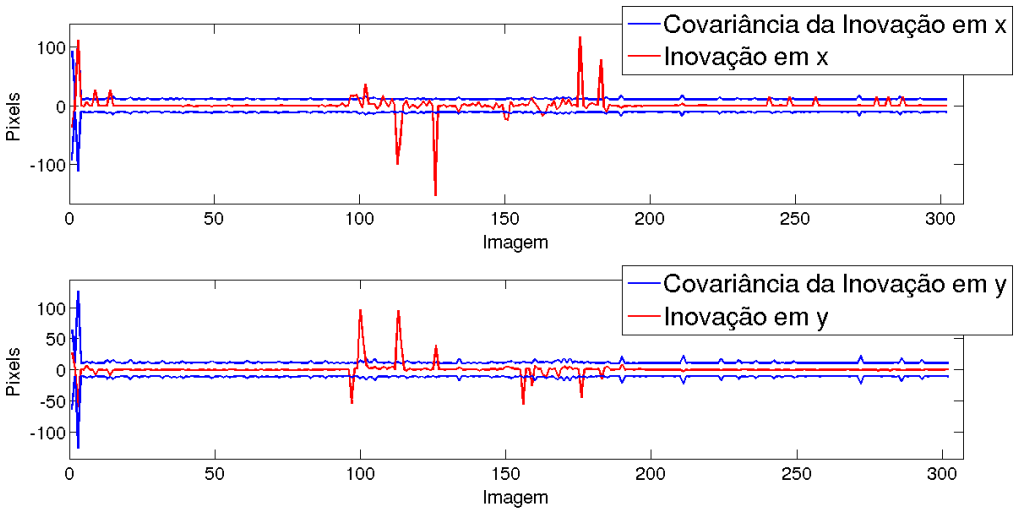


Figura 7.7: Inovação do EKF comparativamente com os limites 2-σ do erro.

Embora os resultados obtidos sejam bastante satisfatórios, é possível observar alguns picos na inovação em x e y dos pontos do alvo assim como alguns "saltos"na posição estimada pelo filtro. Após alguma investigação conclui-se que este problema era não ser possível prever com exatidão mudanças

bruscas no *pitch* do robô devido ao seu INS apenas conter um giroscópio com apenas o eixo dos zz. Desta forma, sempre que durante a manobra havia uma mudança brusca no *pitch* do robô a inovação aumentava provocando uma correção que afetava também os ângulos das câmaras e induzia alguma instabilidade no sistema.

Para combater este problema decidiu-se testar a manobra simplificada, na qual esta situação será minimizado devido à ausência de movimento.

Figura 7.8: Imagens do robô a realizar a manobra simplificada.

Na figura 7.9 temos a posição estimada pelo método de calibração em comparação com a posição do robô obtida pela odometria. Visto não existir movimento na manobra simplificada, a posição dada pela odometria encontra-se muito perto do zero, no entanto, como podemos ver pela posição estimada o robô foi mal posicionado e encontra-se ligeiramente à esquerda do primeiro ponto do alvo.

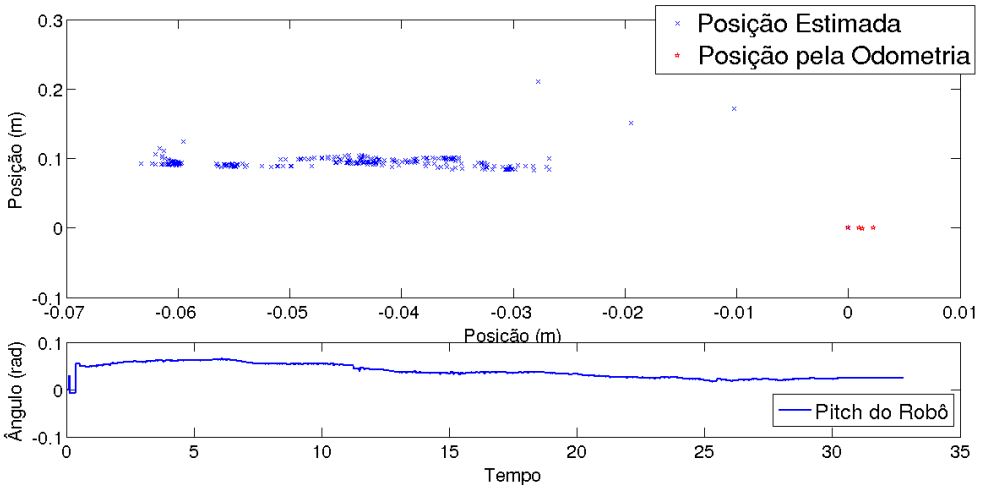


Figura 7.9: Estimação do *pitch* e posição do robô.

A figura 7.10 demonstra a estimação atitude para a câmara do *kicker* e as suas covariâncias durante a manobra simplificada.

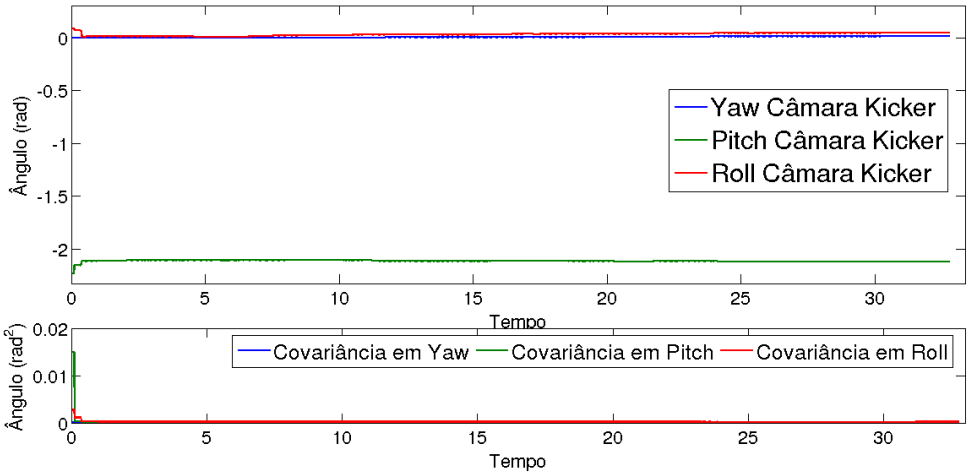


Figura 7.10: Estimação dos ângulos de Euler da câmara do *kicker*.

A altura e as distâncias ao eixo da câmara do *kicker* e as suas covariâncias estão representadas na figura 7.11.

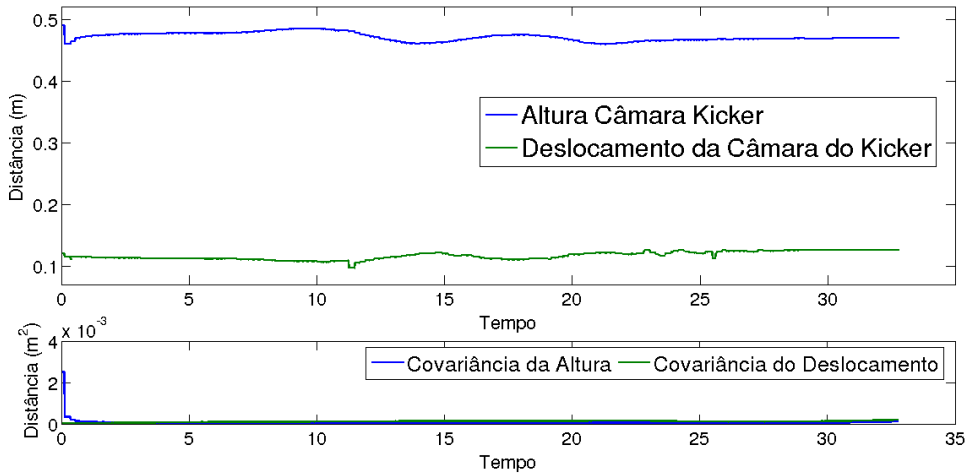


Figura 7.11: Estimação da altura e deslocamentos da câmara do *kicker*.

Nas figuras 7.12 e 7.13 temos a atitude, altura e distâncias ao eixo da câmara da *head* e as suas covariâncias.

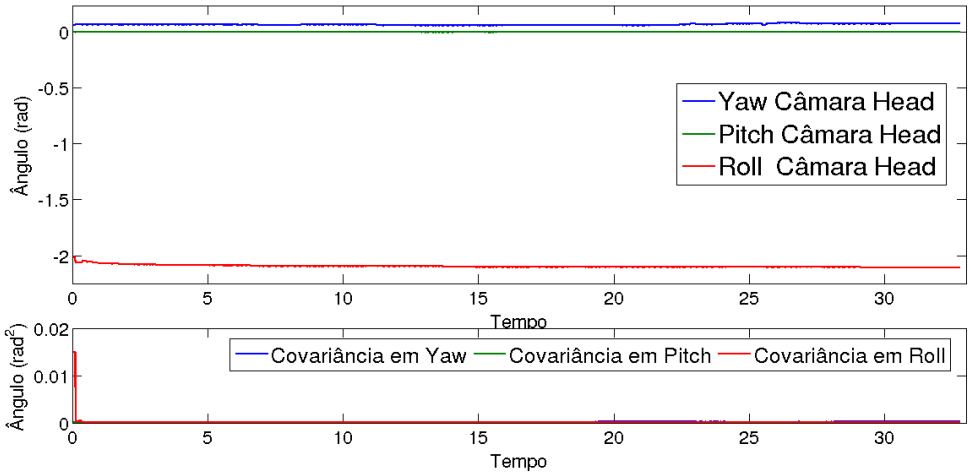


Figura 7.12: Estimação dos ângulos de Euler da câmara do *kicker*.

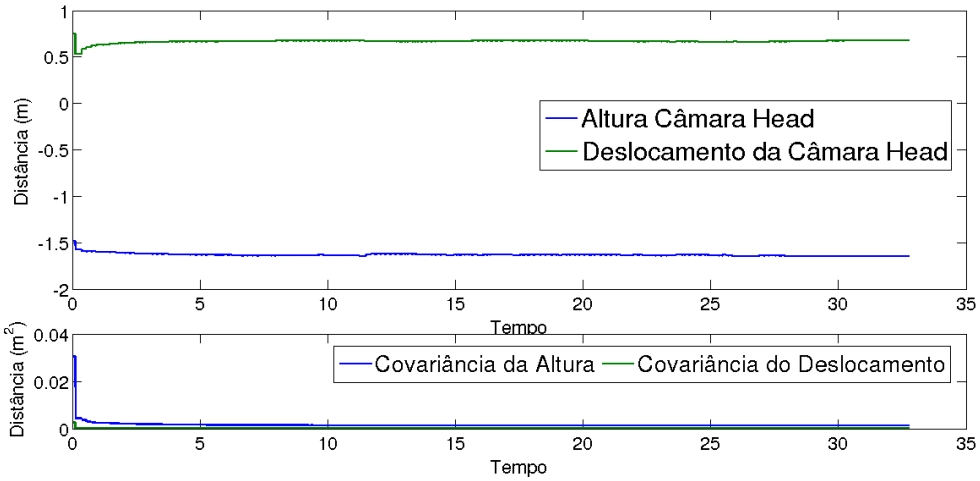


Figura 7.13: Estimação da altura e deslocamentos da câmara do *kicker*.

A figura 7.14 demonstra a inovação em x e y dos pontos em comparação com os limites 2-σ do erro de estimação durante a manobra simplificada.

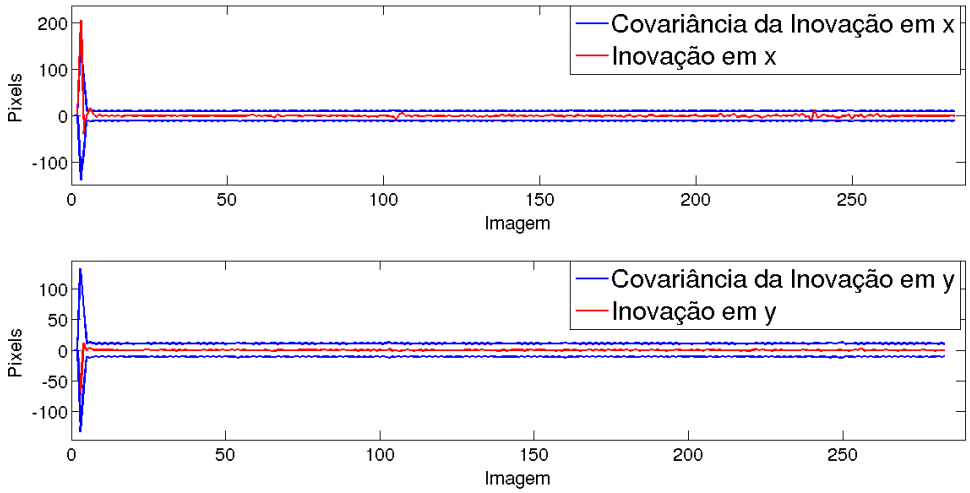


Figura 7.14: Inovação do EKF comparativamente com os limites 2-σ do erro.

Na tabela 7.1, podemos ver a comparação dos valores da calibração manual do robô R2 face à média de quatro calibrações automáticas efetuadas com o novo método. É possível verificar que os resultados obtidos com recurso ao método automático são bastante similares com os obtidos pela calibração manual. Por outro lado é também visível pelo desvio padrão que existe uma grande consistência entre as várias calibrações obtidas pelo método automático.

	h_{kc}	d_{kc}	θ_{kc}	ϕ_{kc}	ψkc	h_{hc}	d_{hc}	θ_{hc}	ϕ_{hc}	ψ_{hc}
Calib. Manual	0,5016	0,1125	0,5461	-0,0218	-0,0263	0,6638	0,0636	0,4998	-0,0426	-0,0307
Calib. Automático (Méd.)	0,5003	0,0968	0,5548	-0,0238	-0,0496	0,6587	0,0485	0,5086	-0,0429	-0,0564
Desvio Padrão	0,0020	0,0027	0,0063	0,0099	0,0125	0,0031	0,0021	0,0070	0,0121	0,0122

Tabela 7.1: Média dos resultados obtidos em 4 calibrações automáticas em comparação com os valores calibração manual para o robô R2

Os resultados obtidos durante os testes demonstraram que este método permite a obtenção de valor de calibração bastante satisfatórios em aproximadamente de 1/10 do tempo do método manual e com uma maior consistência entre calibrações. Durante o Robocup 2013 na Holanda, onde estes resultados foram recolhidos, este novo método permitiu a calibração de robôs num espaço de tempo nunca antes conseguido, como por exemplo durante o intervalo de um jogo (algo nunca antes conseguido).

Capítulo 8

Conclusões e Trabalho Futuro

Nesta dissertação foi apresentada uma solução de calibração automática para robôs com câmaras direcionais, montadas em eixos rotativos, como é o caso da equipa ISePorto.

A solução proposta permite uma calibração total dos parâmetros extrínsecos das câmaras e eixos do robô de forma autónoma num curto espaço de tempo. A calibração dos robôs da equipa ISePorto apresenta um desafio bastante interessante devido à existência de vários eixos rotativos.

O método proposto apenas necessita que o robô desempenhe uma trajetória circular em torno de um alvo com um padrão conhecido, em forma de xadrez, enquanto é recolhida informação visual, inercial e odometria.

Para tal foi implementado, com recurso à API de recolha de dados do ISePorto, um módulo de registo para guardar esta informação durante a execução da manobra. Foi também implementado um módulo que processa a informação visual em tempo real, com recurso ao OpenCV, para extração dos pontos do xadrez. Esta informação recolhida é depois fundida através de um EKF que foi desenvolvido de forma a ser possível estimar todos os parâmetros extrínsecos do robô. Foi introduzido no EKF uma validação da inovação para descartar as medidas de observação erradas (por não deteção total do alvo). Esta validação foi desenvolvida recorrendo ao teste da inovação normalizada face aos limites da inovação definidos.

O EKF foi validado e afinado com base na sequência da inovação por forma a esta apresentar uma média nula, ruído branco e baixa autocorrelação.

Os resultados obtidos pelo novo método de calibração foram bastante similares aos obtidos pelo método manual que era anteriormente utilizado. Foi também observada uma maior consistência nos resultados obtidos durante as experiências realizadas, onde o desvio padrão observado entre experiências se manteve sempre bastante baixo.

Foi também testada e validada uma manobra simplificada que não inclui movimento do robô face ao alvo, apenas movimentos nos seus dois eixos rotativos, que apresentou resultados bastante similares à manobra completa para casos em que não tenham existido grandes mudanças no robô a

nível mecânico.

Durante o Robocup 2013 na Holanda, onde foi testado o método de calibração automático, foram obtidos resultados bastante satisfatórios em tempos máximos de cinco minutos, o que representa uma diminuição de aproximadamente $10x$ no tempo de calibração face ao método manual.

Esta dissertação deu também origem a um artigo cientifico [3] que foi publicado na conferência do IEEE que decorreu durante o Festival Nacional de Robótica 2013.

Um fator importante na escolha do EKF como ferramenta de filtragem e estimação face a alternativa como o MLE foi a possibilidade de implementação do EKF numa solução de tempo real. Como durante os jogos da MSL os parâmetros dos robôs estão em constante mudança devido a colisões e movimentos bruscos, de futuro este método pode ser adaptado para uma solução em tempo real que usa as linhas do campo (ou outras *features* existentes no mundo) para manter os parâmetros calibrados durante o jogo.

Outro trabalho futuro passa pela introdução de um INS 3D no sistema para que seja possível uma melhor estimação dos ângulos de Euler do robô com especial importância para inclinação em *pitch* provocada pelas rodas loucas do robô. A implementação e teste deste método nas outras plataformas existente no LSA está também planeada para trabalho futuro.

Bibliografia

[1] Maxon Motors. http://www.maxonmotor.com/, 2013.

[2] I. Culjak, D. Abram, T. Pribanic, H. Dzapo, and M. Cifrek. A brief introduction to OpenCV, 2012.

[3] João Ribeiro, Rui Serra, Nuno Nunes, Hugo Silva, and José Almeida. EKF-based visual self-calibration tool for robots with rotating directional cameras. In *Proceedings of the 13th International Conference on Mobile Robots and Competitions*, page 6, Lisbon, 2013.

[4] Roland Siegwart and Illah R. Nourbakhsh. Introduction to Autonomous Mobile Robots. April 2004.

[5] H.D. Burkhard, D. Duhaut, M. Fujita, P. Lima, R. Murphy, and R. Rojas. The road to RoboCup 2050. *IEEE Robotics & Automation Magazine*, 9(2):31–38, June 2002.

[6] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pages 666–673 vol.1. IEEE, 1999.

[7] Yuan Shen, Xiqin Wang, Huadong Meng, and H Zhang. Camera Extrinsic Parameter Calibration by Minimizing Total Control-Point Errors. *International Conference on Computer Vision, Beijing, China*, 2005.

[8] D Robertson R. Cipolla T. Drummond. Camera Calibration From Vanishing Points in Images of Architectural Scenes. *Proceedings of the British Machine Vision Conference*, 1999.

[9] R. Pless. Extrinsic calibration of a camera and laser range finder (improves camera calibration). In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2301–2306. IEEE, 2004.

[10] André Dias, Miguel Almeida, Alfredo Martins, and Eduardo Silva. Traction characterization in the Robocup Middle Size League. *IEEE Robotica 2007 7th Conference on Mobile Robots and Competitions*, 2007.

[11] J Y Bouguet. Camera calibration toolbox for Matlab. In *http://www.vision.caltech.edu/bouguetj/*, 2008.

[12] R. Pio. Euler angle transformations. *IEEE Transactions on Automatic Control*, 11(4):707–715, October 1966.

[13] Yi Ma, Stefano Soatto, Jana Kosecká, and S. Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models (Interdisciplinary Applied Mathematics)*. Springer, 2005.

[14] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[15] Jong-Soo Lee and Yu-Ho Jeong. CCD camera calibrations and projection error analysis. In *Proceedings KORUS 2000. The 4th Korea-Russia International Symposium On Science and Technology*, volume 2, pages 50–55. IEEE, 2000.

[16] Bernd Jähne and Horst Haussecker. *Computer Vision and Applications: A Guide for Students and Practitioners*. Academic Press; 1 edition (May 15, 2000), 2000.

[17] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)*. The MIT Press, 2005.

[18] Hugh Durrant-whyte. Introduction to Estimation and the Kalman Filter. 2006.

[19] Alfredo Martins, Eduardo Silva, Carlos Almeida, and Hugo Silva. ISePorto Robotic Soccer Team for Robocup 2009:Improving Perception. 2009.

[20] UeYE Cameras. http://www.ids-imaging.com, 2013.

[21] M. Rufli, D. Scaramuzza, and R. Siegwart. Automatic detection of checkerboards on blurred and distorted images. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3121–3126. IEEE, September 2008.

Apêndice A

Maple

```
> restart;

> with(LinearAlgebra);

> with(Student[LinearAlgebra]);

> A := Matrix(3, 4, {(1, 1) = fx, (1, 2) = 0, (1, 3) = xc, (1, 4) = 0, (2, 1) = 0, (2, 2) =
  fy, (2, 3) = yc, (2, 4) = 0, (3, 1) = 0, (3, 2) = 0, (3, 3) = 1, (3, 4) = 0});

> Rrobotworld := RotationMatrix(thetarobot, (0, 0, 1));

> Trobotworld := Matrix(4, 4, Rrobotworld); Trobotworld(1, 4) := Probotx;
  Trobotworld(2, 4) := Proboty; Trobotworld(4, 4) := 1;

> Tworldrobot := simplify(MatrixInverse(Trobotworld));

> Rbaserobot := RotationMatrix(Basetheta, (0, 0, 1)) .
  RotationMatrix(Basepitch, (0, 1, 0)) .
  RotationMatrix(Baseroll, (1, 0, 0));

> Tbaserobot := Matrix(4, 4, Rbaserobot); Tbaserobot(4, 4) := 1;

> Trobotbase := simplify(MatrixInverse(Tbaserobot));

> Rkickbase := RotationMatrix(thetakick, (0, 0, 1));
```

```
> Tkickbase := Matrix(4, 4, Rkickbase); Tkickbase(4, 4) := 1;

> Tbasekick := simplify(MatrixInverse(Tkickbase));

> Rcamkickerkick := RotationMatrix(CamKickertheta, (0, 0, 1)) .
  RotationMatrix(CamKickerpitch, (0, 1, 0)) .
  RotationMatrix(CamKickerroll, (1, 0, 0));

> Tcamkickerkick := Matrix(4, 4, Rcamkickerkick);
  Tcamkickerkick(4, 4) := 1;
  Tcamkickerkick(1, 4) := Kickerdistx;

> Tcamkickerkick(2, 4) := Kickerdisty; Tcamkickerkick(3, 4) := KickerH;

> Tkickcamkicker := simplify(MatrixInverse(Tcamkickerkick));

> TWorldcamkicker := (Tkickcamkicker . Tbasekick . Trobotbase . Tworldrobot);

> PPMkicker := (A . TWorldcamkicker);

> Rheadbase := RotationMatrix(thetahead + thetaheadoffset, (0, 0, 1));

> Theadbase := Matrix(4, 4, Rheadbase); Theadbase(4, 4) := 1;

> Tbasehead := simplify(MatrixInverse(Theadbase));

> RcamheadHead := RotationMatrix(CamHeadtheta, (0, 0, 1)) .
  RotationMatrix(CamHeadpitch, (0, 1, 0)) .
  RotationMatrix(CamHeadroll, (1, 0, 0));

> TcamheadHead := Matrix(4, 4, RcamheadHead); TcamheadHead(4, 4) := 1;
  TcamheadHead(1, 4) := Headdistx;
  TcamheadHead(2, 4) := Headdisty;
  TcamheadHead(3, 4) := HeadH;

> THeadcamhead := simplify(MatrixInverse(TcamheadHead));

> TWorldcamhead := (THeadcamhead . Tbasehead . Trobotbase . Tworldrobot);
```

```
> PPMhead := (A . TWorldcamhead);

> #Simplifications

> Basetheta := 0;

> Baseroll := 0;

> Pw := Vector(4, 1, [Pwx, Pwy, Pwz, 1]);

> Pcamkicker := simplify(PPMkicker . Pw);

> PimageKicker := Vector(2, 1, [simplify(Pcamkicker(1)/Pcamkicker(3)),
  simplify(Pcamkicker(2)/Pcamkicker(3))]);

> PcamHead := simplify(PPMhead . Pw);

> PimageHead := Vector(2, 1, [simplify(PcamHead(1)/PcamHead(3)),
  simplify(PcamHead(2)/PcamHead(3))]);

> with(codegen, makeproc);

> with(CodeGeneration);

> FPimagekicker := makeproc(PimageKicker, [Probotx, Proboty, thetarobot, Basepitch, thetakick,
  CamKickertheta, CamKickerpitch, CamKickerroll, Kickerdistx,
  Kickerdisty, KickerH, fx, fy, xc, yc]);

> Matlab(FPimagekicker, optimize = true, output = "wold2camkicker.m");

> FPimageHead := makeproc(PimageHead, [Probotx, Proboty, thetarobot, Basepitch, thetahead,
  thetaheadoffset, CamHeadtheta, CamHeadpitch, CamHeadroll, Headdistx,
  Headdisty, HeadH, fx, fy, xc, yc]);

> Matlab(FPimageHead, optimize = true, output = "wold2camhead.m");

> #Jacobian
```

```
> with(MTM);

> Jacobiankicker := jacobian(PimageKicker, [Probotx, Proboty, thetarobot, Basepitch,
  thetakick, thetahead, KickerH, Kickerdistx, Kickerdisty, CamKickerroll,
  CamKickerpitch, CamKickertheta, HeadH, Headdistx, Headdisty, CamHeadroll,
  CamHeadpitch, CamHeadthetal]);

> FuncJacobiankicker := makeproc(Jacobiankicker, [Probotx, Proboty, thetarobot, Basepitch,
  thetakick, CamKickertheta, CamKickerpitch, CamKickerroll, Kickerdistx,
  Kickerdisty, KickerH, fx, fy, xc, yc]);

> Matlab(FuncJacobiankicker, optimize = true, output = "jacobkicker.m");

> Jacobianhead := jacobian(PimageHead, [Probotx, Proboty, thetarobot, Basepitch, thetakick,
  thetahead, KickerH, Kickerdistx, Kickerdisty, CamKickerroll, CamKickerpitch,
  CamKickertheta, HeadH, Headdistx, Headdistx, CamHeadroll, CamHeadpitch, CamHeadthetal]);

> FuncJacobianhead := makeproc(Jacobianhead, [Probotx, Proboty, thetarobot, Basepitch, thetahead,
  thetaheadoffset, CamHeadtheta, CamHeadpitch, CamHeadroll, Headdistx, Headdisty,
  HeadH, fx, fy, xc, yc]);

> Matlab(FuncJacobianhead, optimize = true, output = "jacobhead.m");
```