

**FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO**

# **Odometria visual em robôs para a agricultura com câmara(s) com lentes olho de peixe**

**Sérgio Miguel Vieira Pinto**

**RELATÓRIO DE PROGRESSO**



**Mestrado Integrado em Engenharia Eletrotécnica e de Computadores**

**Orientador: Prof. Dr. Armando Jorge Sousa**

**Co-orientador: Prof. Dr. Filipe Neves dos Santos**

**1 de Junho de 2018**



# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Enquadramento . . . . .	1
1.2	Motivação e Objetivos . . . . .	2
<b>2</b>	<b>Revisão Bibliográfica</b>	<b>3</b>
2.1	Contextualização . . . . .	3
2.2	Parâmetros Intrínsecos . . . . .	6
2.2.1	Lente olho de peixe . . . . .	7
2.3	Parâmetros Extrínsecos . . . . .	8
2.4	Detetores de Características . . . . .	9
2.4.1	Detetores de Cantos . . . . .	10
2.4.1.1	Harris . . . . .	10
2.4.1.2	FAST . . . . .	11
2.4.2	Detetores de blobs . . . . .	11
2.4.2.1	SIFT . . . . .	11
2.4.2.2	SURF . . . . .	14
2.5	Descritor de Características . . . . .	15
2.5.1	ORB . . . . .	16
2.5.2	SIFT . . . . .	17
2.5.3	SURF . . . . .	17
2.6	Associação de Características . . . . .	18
2.6.1	Correspondência de Características . . . . .	18
2.6.1.1	Brute force . . . . .	18
2.6.2	Seguimento de Características . . . . .	18
2.6.2.1	FLANN . . . . .	18
2.7	Estimação do movimento . . . . .	20
2.7.1	2D-para-2D . . . . .	21
2.7.2	3D-para-3D . . . . .	22
2.7.3	3D-para-2D . . . . .	23
2.8	Métodos de ajuste de erros . . . . .	24
2.8.1	Windowed bundle adjustment . . . . .	24
2.8.2	RANSAC . . . . .	25
<b>3</b>	<b>Sistema de Odometria Visual</b>	<b>27</b>
3.1	Software . . . . .	27
3.1.1	ROS . . . . .	27
3.1.1.1	Nó VisodoAgro . . . . .	28
3.1.2	Desenvolvimento em Matlab . . . . .	35

3.2	Hardware . . . . .	36
3.2.1	Raspberry Pi . . . . .	36
3.2.2	Câmara com lente olho de peixe . . . . .	36
3.2.2.1	Tipos de lentes . . . . .	36
<b>4</b>	<b>Resultados Experimentais</b>	<b>39</b>
4.1	Calibração da câmara . . . . .	39
4.2	Diferença entre detetores de características . . . . .	40
4.3	Comparação das combinações . . . . .	41
4.4	Testes . . . . .	43
4.4.1	Teste de parâmetros . . . . .	43
4.4.2	Sequência sem Movimento . . . . .	43
4.4.3	Deteção da trajetória . . . . .	44
<b>5</b>	<b>Conclusões e Trabalho Futuro</b>	<b>45</b>
5.1	Conclusões . . . . .	45
5.2	Trabalho Futuro . . . . .	45

# Lista de Figuras

2.1	Ilustração do problema de OV para visão estéreo. [8]	4
2.2	Etapas de um sistema de Odometria Visual.	5
2.3	Esquema do modelo de projeção perspectiva. [9]	6
2.4	Projeção de um ponto no mundo no plano de imagem. Onde o ponto $p'$ representa projeção perspectiva e o ponto $p$ com o modelo equiangular com distorção da lente olho de peixe. [11, 12]	7
2.5	Algoritmo de construção da imagem com lente olho de peixe. [13]	8
2.6	Parâmetros que definem a posição e orientação do sistema de coordenadas da câmara com um sistema de coordenadas global.	9
2.7	Detetor de cantos Harris. [15]	10
2.8	Detetor de canto FAST, círculo em volta do ponto $p$ . [15]	11
2.9	Computação da DoG. [15]	12
2.10	O <i>pixel</i> marcado com um $x$ é avaliado em relação aos seus vizinhos no mesmo nível e nos níveis superiores e inferiores. [15]	13
2.11	Ilustração do método do integral da imagem. [15]	15
2.12	Descritor SIFT. As setas designam a soma dos gradientes nas caixas individuais. O círculo azul representa a região que é considerada na atuação do filtro Gausiano. Este exemplo ilustra uma matriz 8x8 resultando num 2x2 descritor. [15]	17
2.13	A imagem da esquerda ilustra a decomposição 2-d, note que as linhas vermelhas são para o eixo $x$ e as linhas azuis para o eixo $y$ . A imagem da direita ilustra a árvore 2-d correspondente.	19
2.14	Algoritmo de procura NN baseado nas árvores 2-d.	20
2.15	Uma ilustração da restrição epipolar. [8]	21
3.1	Diagrama de ação do nó implementado em ROS.	29
3.2	Representação da linha epipolar.	29
3.3	Representação do plano epipolar.	31
3.4	Representação da razão entre frames.	33
3.5	Representação da transformação homogénea.	33
3.6	Representação dos tópicos elaborados no nó visodoAgro.	34
3.7	Conteúdo da mensagem <i>msgInfo.msg</i> .	34
3.8	Conteúdo da mensagem <i>msgTime.msg</i> .	35
3.9	Conteúdo da mensagem <i>visodom.msg</i> .	35
3.10	Exemplar da Raspberry Pi 3 modelo B.	36
3.11	Diferença de ângulos de visão de lentes olho de peixe.	37
3.12	Câmara com lente olho de peixe e suporte para visão noturna.	38
4.1	Quadrado de xadrez para calibrar câmara.	39

4.2	Imagem da calibração dos parâmetros da câmara. . . . .	40
4.3	Distribuição dos pontos característicos para os quatro detetores de características, FAST , SURF, SIFT, HARRIS. . . . .	41
4.4	Tabela de comparação das combinações dos métodos. . . . .	42

# Abreviaturas e Símbolos

INESC-TEC	Institute for Systems and Computer
Engineering, Technology and Science	
GPS	Global Position System
GNSS	global Navigations Satellite System
INS	Inertial Navigation System
OV	Odometria Visual
UAV	Unmanned aerial vehicle
NASA	National Aeronautics and Space Administration
RANSAC	RANdom SAmple Consensus
FAST	Feature from Accelerated Segment Test
SIFT	Scale Invariant Feature Transform
DoG	Diference of Gaussians
SURF	Speeded Up Robust Features
ORB	Oriented FAST and Rotation-Aware BRIEF
BRIEF	Binary Robust Independent Elementary Features
SVD	Singular Value Decomposition
FLIR	Foreard Looking Infra-Red
ROS	Robot Operating System
IDE	Integrated Development Environment





# Capítulo 1

## Introdução

Este capítulo levanta algumas questões a serem respondidas durante a dissertação e fornece uma visualização dos objetivos e alcance do documento.

### 1.1 Enquadramento

A necessidade de uma maior eficiência nos trabalhos do quotidiano, levou a um desenvolvimento da robótica, tornando os robôs mais autónomos e eficazes. Com o avanço da robótica é possível aumentar o número de horas de trabalho sem perdas de eficiência e desgaste que um ser humano teria, obtendo uma diminuição de custos.

Em Portugal a vinicultura tem vindo a ter um crescimento constante e com ela um aumento da necessidade da robótica. A monitorização das vinhas é essencial para qualidade do produto final, sendo esta, muitas vezes difícil de se realizar com eficácia. Assim, através da robótica essa eficácia pode ser conseguida, realizando trabalhos de maior dificuldade para o Homem a um custo mais reduzido, tal como a monitorização dia e noite, 24h por dia, dos terrenos.

Desta forma, o projeto *RoMoVi*, desenvolvido pelo centro de Robótica do INESC-TEC, pretende desenvolver um robô, no âmbito da robótica para a agricultura, capaz de podar, monitorizar e fertilizar as encostas de vinhas inclinadas [1]. Este projeto tem disponíveis três plataformas: o *AGROB VI6*, *AGROB VI5*, *AGROB VI4*, sendo a presente dissertação desenvolvida no contexto do projeto *RoMoVi* e linha de investigação apelidada de *AGROB* [2].

Neste projeto é necessário a implementação de um sistema de localização que seja de baixo custo e adequado à aplicação em contexto de vinha de encosta.

A precisão na localização de robôs é fundamental para a sua eficácia e funcionamento. Existem vários tipos de localizações, tais como *Global Position System* (GPS), *Global Navigations Satellite System* (GNSS), *Inertial Navigation System* (INS), Odometria através das rodas, laser/ultrassónico Odometria e Odometria Visual (OV). Contudo, cada tecnologia tem as suas fraquezas. Odometria das rodas é uma tecnologia simples para estimação da posição, mas a inclinação do terreno e deslizamento das rodas no pavimento causa erros grandes. Laser/ultrassónico Odometria utiliza energia acústica para detetar objetos e medir distâncias entre os sensores e os objetos, contudo

estes sensores são muito sensíveis ao ruído. INS é ótimo para a acumulação de deslizamento e têm grande precisão, mas em contrapartida é muito dispendiosa a nível monetário e não é a ideal para soluções comerciais. Apesar de a GPS ser a solução mais comum para localização absoluta por causa da não acumulação de erro, não é útil para este projeto devido ao fraco sinal dos satélites. Tal, deve-se à inclinação dos terrenos e possibilidades de céu nublado. Para além disso, a utilização de GPS este também se torna cara para a precisão em centímetros. [3]

Por último, OV é uma tecnologia que envolve um fluxo de imagens adquiridas através de uma câmara e posteriormente analisadas permitindo obter a estimação da localização. É um método barato e com grande precisão, que se enquadra neste projeto. As desvantagens desta localização passam pela necessidade de um bom processamento e a possibilidade de existência de erros causados por objetos dinâmicos e/ou sombras.

## 1.2 Motivação e Objetivos

Com as dificuldades de localização já mencionadas no capítulo anterior, a tecnologia mais indicada é a Odometria Visual (OV). Esta será implementada num raspberry pi com uma câmara com lente olho de peixe para obter maiores ângulos de captura.

Assim, utilizando a câmara com lente de olho de peixe com um ângulo de cerca de 180° é possível capturar um fluxo de imagens com imensa informação. Informação essa que será analisada por um algoritmo implementado num raspberri pi, da qual resultará uma estimativa de localização.

A utilização de visão será sensível às condições de iluminação e reflexões. Com isto, o processamento de imagem terá de ser rigoroso para que os erros sejam mínimos.

Na presente dissertação pretende-se desenvolver um sistema de auxílio à localização que seja de baixo custo e adequado à aplicação em contexto de vinha de encosta. Neste sentido será utilizada a OV na qual resultará uma avaliação da precisão do método.

## Capítulo 2

# Revisão Bibliográfica

Odometria Visual (OV) é um método de estimação da posição através de um fluxo de imagens de uma câmara (Sistema monocular) ou várias câmaras (Sistema estéreo) [4]. OV foi durante muitos anos, um dos estudos pioneiros de Nister, Naroditsky and Bergen [5]. O trabalho deles resultou num algoritmo em que extrai pontos característicos de uma imagem, cruzavam com outros e , no fim, usavam para obter uma estimativa de movimento.

A implementação de OV em locais interiores é bem sucedida enquanto que em locais exteriores são notórios alguns problemas. Alguns fatores, como sombras e objetos dinâmicos, causam dificuldades de localização em locais exteriores. Para OV funcionar eficientemente é necessário iluminação suficiente e ambiente estático para permitir que o movimento seja extraído corretamente.

OV tem uma ampla aplicação e foi implementada em diferentes campos eficientemente. Os domínios de aplicação incluem robótica e automação. Os tipos de aplicação são sistemas robóticos móveis, tais como terrestre, subaquáticos, aéreos e espaciais.

Em termos terrestres, a OV é usada para a navegação e deteção de objetos eficientemente. Quanto aos sistemas robóticos móveis subaquáticos, a OV tem um papel significativo nos veículos autónomos subaquáticos e sistemas de inspeção de recifes de corais [3]. A nível aéreo, a OV é aplicada a *drones* (UAVs) para melhorar a performance de navegação autónoma. Por último, na exploração espacial , a OV é usada para estimar o movimento do robô NASA Mars [6] .

Na indústria , OV desempenha um papel importante. É aplicada em sistemas de apoio à condução, tal como em sistema de travagem assistida baseados em visão. Em grande parte dos veículos são utilizados sensores de visão para navegação e deteção de obstáculos devido ao sinal de GPS ser fraco, ser de leve implementação (em veículos pequenos é crucial) e ser suficientemente eficaz para o baixo custo. Na agricultura é utilizada OV para estimação relativa em relação às culturas [7, 1].

### 2.1 Contextualização

Nesta secção é feito um breve resumo da computação em sistemas de OV.

Com base no artigo [8], um determinado agente ligado a um sistema de câmaras move-se num determinado ambiente e captura imagens em instantes de tempo discreto  $k$ . Caso o sistema seja composto por visão monocular, o conjunto de imagens capturadas em intervalos de  $k$  é representado por  $I_{0:n} = \{I_0, \dots, I_n\}$ . No caso de ter visão estéreo, então existem duas imagens (esquerda e direita) em cada instante de tempo, representadas por  $I_{e,0:n} = \{I_{e,0}, \dots, I_{e,n}\}$  e  $I_{d,0:n} = \{I_{d,0}, \dots, I_{d,n}\}$ . Na Figura 2.1 é apresentada uma ilustração deste cenário.

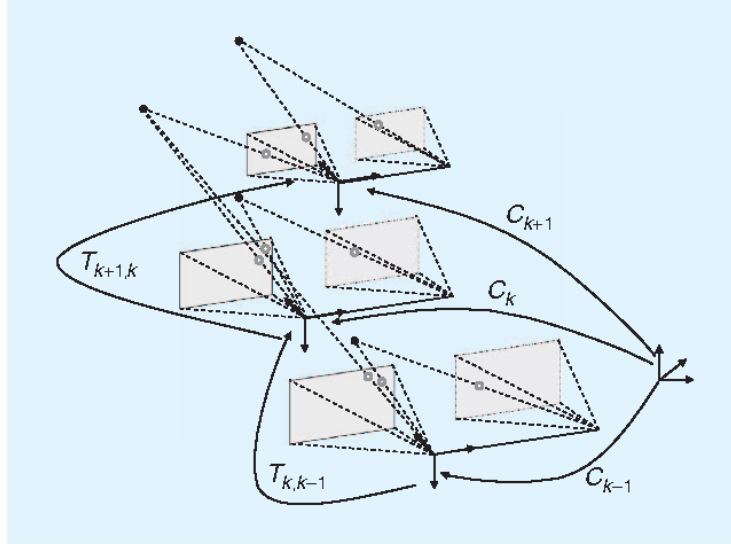


Figura 2.1: Ilustração do problema de OV para visão estéreo. [8]

Por simplificação, é assumido que o sistema de coordenadas da câmara coincide com o sistema de coordenadas do agente. No caso de sistema estéreo, considera-se que o sistema de coordenadas da câmara esquerda pode ser utilizado como origem sem perda de aspetos gerais.

As posições de uma câmara em instantes de tempo adjacentes  $k-1$  e  $k$  estão relacionadas por uma transformação de corpo rígido  $T_{k,k-1} \in R^{4 \times 4}$  da seguinte forma:

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix}$$

onde  $R_{k,k-1} \in R^{3 \times 3}$  é a matriz de rotação e  $t_{k,k-1} \in R^{3 \times 1}$  é o vetor de translação. O conjunto  $T_{1:n} = \{T_{1,0}, \dots, T_{n,n-1}\}$  contém todos os movimentos sequenciais da câmara. Para simplificar a notação, a partir de agora,  $T_k$  será usado em vez de  $T_{k,k-1}$ . Finalmente, o conjunto das posições da câmara  $C_{0:n} = \{C_0, \dots, C_n\}$  contém as transformações da câmara em relação à coordenada da frame inicial no instante  $k = 0$ . A posição atual  $C_n$  pode ser calculada juntando todas as transformações  $T_k$  ( $k = 1 \dots n$ ), e assim,  $C_n = C_{n-1}T_n$ , com  $C_0$  sendo a posição da câmara no instante  $k = 0$ , que pode ser definida arbitrariamente pelo utilizador.

A principal tarefa em um sistema OV é calcular a transformação relativa  $T_k$  a partir das imagens  $I_k$  e  $I_{k-1}$ , e depois juntar as transformações para recuperar a trajetória completa  $C_{0:n}$  da câmara.

Isto significa que o sistema consegue recuperar de forma incremental a trajetória, posição após posição.

A fim de efetuar um sistema de OV é necessário realizar várias etapas. A Figura 2.2 ilustra um diagrama de blocos com a metodologia a utilizar.

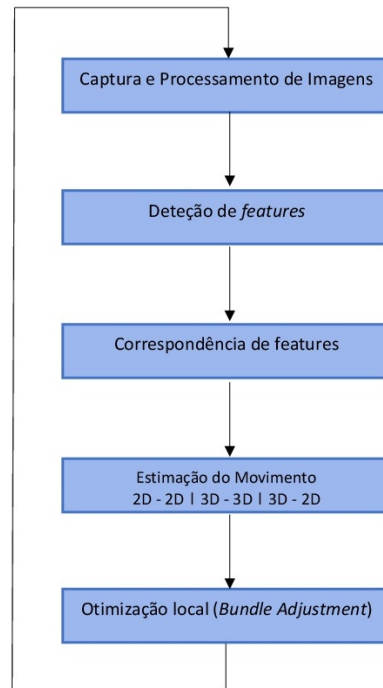


Figura 2.2: Etapas de um sistema de Odometria Visual.

Desta forma, a primeira etapa ilustra o processo de captura de uma nova imagem  $I_k$ , ou um par de imagens no caso de um sistema estéreo. Nesta mesma etapa é necessário o processamento da imagem para remover possíveis deformações das imagens ou efeitos de lentes.

A segunda etapa apresenta dois métodos diferentes para encontrar características <sup>1</sup>, que consistem na procura de características visuais salientes que possam corresponder noutras imagens.

A terceira etapa apresenta dois métodos diferentes para correlacionar características. Nesta mesma etapa são aplicados algoritmos de correção para remover eventuais erros de correspondência. Uma solução para o problema é a utilização do algoritmo RANSAC.

A quarta etapa consiste no cálculo do movimento relativo  $T_k$  entre os instantes de tempo  $k - 1$  e  $k$ . Após a obtenção do  $T_k$  então é possível calcular a posição da câmara  $C_k$  por junção do  $T_k$  com a posição anterior  $C_{k-1}$ .

Finalmente, a última etapa descreve a aplicação do algoritmo de otimização local, *Buddle Adjustment*, ao longo das últimas características com o objetivo de obter uma estimativa mais precisa da trajetória local.

<sup>1</sup> é uma região na imagem que difere da sua vizinhança em termos de intensidade, cor e textura.

## 2.2 Parâmetros Intrínsecos

A formação de uma imagem em uma câmara ocorre com a entrada de feixes de luz através de uma abertura na câmara e a projeção desses feixes em uma tela, também chamada de plano de imagem.

Em uma câmara real, um ponto no mundo reflete diversos feixes de luz. Se todos os feixes refletidos por esse ponto convergirem para um mesmo ponto no plano de imagem, então é dito que a imagem está focada. O modelo de projeção perspectiva é uma simplificação da câmara real.

O modelo de projeção perspectiva é apresentado na Figura 2.3. O centro de projecção  $\mathbf{O}$  é a origem do sistema de coordenadas da câmara e também o centro da câmara. O eixo-z do sistema de coordenadas da câmara é chamado eixo-principal. O plano  $z = f$  é o plano de imagem e a intersecção do plano de imagem com o eixo-principal é chamado ponto principal. Considere  $\mathbf{X} = [X, Y, Z]^T$  as coordenadas de um ponto no mundo referentes ao sistema de coordenadas da câmara. A intersecção do plano de imagem com o segmento de reta ligando  $\mathbf{X}$  e  $\mathbf{O}$  é a projeção de  $\mathbf{X}$  e é referenciada como  $\mathbf{x}$ . Por semelhança de triângulos observa-se que  $\mathbf{x} = [f \frac{X}{Z}, f \frac{Y}{Z}, f]$  em relação à câmara. Como a última coordenada de  $\mathbf{x}$  será sempre  $f$ , ela será desconsiderada nas equações daqui em diante.

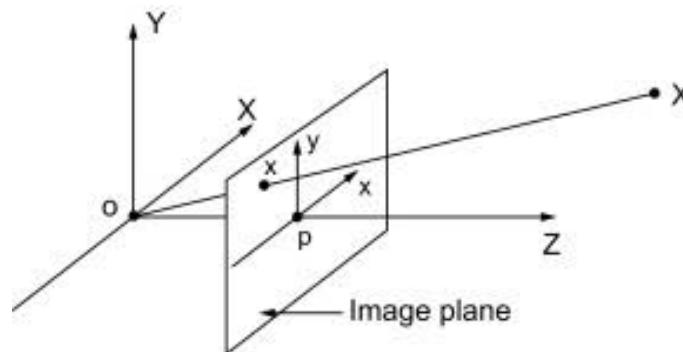


Figura 2.3: Esquema do modelo de projeção perspectiva. [9]

Assim, o mapeamento do mundo 3-D para uma imagem 2-D é obtido através da equação de projeção perspectiva :

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = KX = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix},$$

onde  $\lambda$  é o facto de profundidade,  $\alpha_u$  e  $\alpha_v$  são as distâncias focais, e  $u_0$  e  $v_0$  são as coordenadas centrais da projeção da imagem. Estes parâmetros são conhecidos como parâmetros intrínsecos que dependem de câmara para câmara. Quando o campo de visão da câmara é superior a  $45^\circ$ , os efeitos da distorção radial são visíveis e causam erros superiores.

Note que a última coordenada  $w$  é a escala da coordenada homogênea e não a distância focal  $f$ , que como já foi dito, é desconsiderada daqui em diante. Daqui em diante as coordenadas homogêneas serão representadas por  $\mathbf{x}$  e  $\mathbf{X}$

### 2.2.1 Lente olho de peixe

Em câmaras com lentes olho de peixe os parâmetros intrínsecos são diferentes. Estes tipos de lentes são similares à visão humana, onde a imagem resultante tem grande resolução central e baixa resolução nos pontos mais distantes do centro. O modelo mais comum de lentes olho de peixe é o modelo equiangular ,

$$r = k\theta,$$

onde  $k$  é uma constante [10].

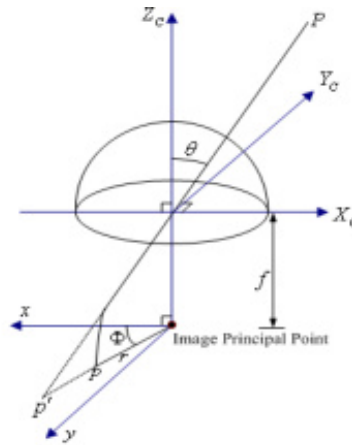


Figura 2.4: Projeção de um ponto no mundo no plano de imagem. Onde o ponto  $p'$  representa projeção perspectiva e o ponto  $p$  com o modelo equiangular com distorção da lente olho de peixe. [11, 12]

A Figura 2.4 representa a diferença entre a representação de um ponto  $P$  na imagem captura . Desta forma, o ponto  $p'$  usa o modelo da projeção perspectiva e o ponto  $p$  o modelo equiangular onde se visualiza a distorção da lente olho de peixe.

Assim, o algoritmo é descrito pela Figura 2.5

Passo 1:

$$\mathbf{P}_C = \mathbf{R}\mathbf{P}_W + \mathbf{t},$$

onde  $\mathbf{P}_C$  é o ponto capturado na imagem,  $\mathbf{P}_W$  é o ponto no mundo,  $\mathbf{R}$  é a matriz de orientação e  $\mathbf{t}$  é o vetor da posição. Os parâmetros  $\mathbf{R}$  e  $\mathbf{t}$  são parâmetros extrínsecos.

Passo 2: O ponto  $\mathbf{P}_C$  é projetado com perspectiva de uma esfera resultando o ponto  $\mathbf{p}$

$$\mathbf{p} = \frac{\mathbf{P}_C}{\|\mathbf{P}_C\|} = (\sin\phi\cos\theta, \sin\phi\sin\theta, \cos\phi)^T.$$

Passo 3: O ponto  $\mathbf{p}$  é mapeado para  $m$  no plano de imagem com distorção da lente da câmara

$$m = D(p),$$

onde  $m = (x, y)$  e  $D$  é o modelo de distorção da lente olho de peixe.

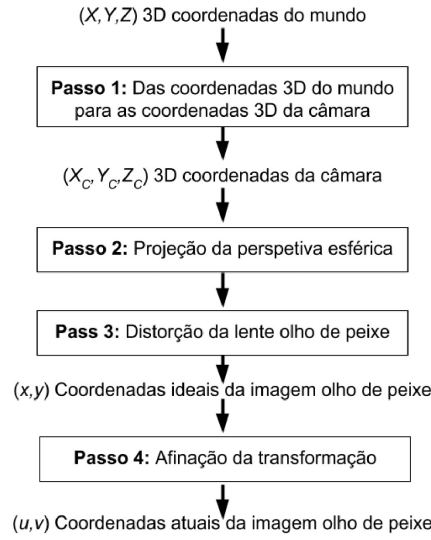


Figura 2.5: Algoritmo de construção da imagem com lente olho de peixe. [13]

Passo 4: O ponto  $m$  é transformado em  $m'$  usando a transformação de afinidade:

$$m' = K_A(m),$$

onde  $m' = (u, v)$ . A imagem obtida é a imagem atual da câmara com lente olho de peixe que é igual :

$$\tilde{m}' = K_A \tilde{m}',$$

onde  $\tilde{m}' = (x, y, 1)^T$  e  $\tilde{m}' = (u, v, 1)^T$  e  $K_A = \begin{bmatrix} r & s & u_0 \\ 0 & 1 & v_0 \\ 0 & 0 & 1 \end{bmatrix}$

Desta forma, concluímos que existem 12 parâmetros para as lentes olho de peixe: 4 parâmetros de transformação de afinidade, 4 radiais e 4 parâmetros de distorção tangencial. Estes parâmetros são os parâmetros intrínsecos das lentes olho de peixe.

## 2.3 Parâmetros Extrínsecos

Em geral os pontos do mundo são descritos em relação a um sistema de coordenadas global. A relação entre os dois sistemas é dada por uma transformação de corpo rígido do tipo

$$\mathbf{X} = \mathbf{R}\mathbf{X}_W + \mathbf{T},$$

onde  $\mathbf{X}_W$  são as coordenadas do ponto  $\mathbf{X}$  em relação ao sistema de coordenadas global. A matriz  $\mathbf{R} \in SO(3)$  é a rotação que alinha o sistema de coordenadas global com o sistema de coordenadas da câmara e  $\mathbf{T} \in \mathbf{R}^3$  é o vetor de translação entre os dois sistemas de coordenadas. Os parâmetros de  $\mathbf{R}$  e  $\mathbf{T}$  são chamados de parâmetros extrínsecos e a matriz



$$[R|T] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

é a matriz de parâmetros extrínsecos. A Figura 2.6 mostra a relação entre os dois sistemas de coordenadas.

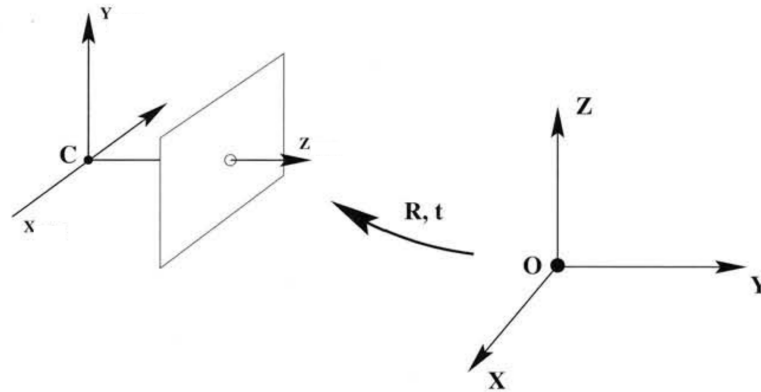


Figura 2.6: Parâmetros que definem a posição e orientação do sistema de coordenadas da câmara com um sistema de coordenadas global.

## 2.4 Detetores de Características

A detecção de características é o processo responsável por determinar características numa imagem. Uma característica pode ser uma região de uma cor específica, um determinado padrão na imagem, um ponto onde ocorre a variação de cores ou outra informação que possa ser extraída da imagem. Estas características devem ser escolhidas de forma a serem possíveis de encontrar e recuperar futuramente.

Relativamente a sistemas de OV, foi descoberto que a distribuição de características numa imagem afeta de uma forma considerável os resultados obtidos. Em particular, quanto mais características encontradas mais estáveis são os resultados da estimação do movimento, mas o tempo de processamento é maior e pode causar perdas de *frames* importantes. As características são procuradas numa imagem através de um detetor de características. Para ser um bom detetor de características têm de obedecer às seguintes características [14]:

- Precisão na localização, tanto em posição como em escala.
- Repetibilidade, garante que uma grande percentagem das características visíveis a duas imagens serão identificadas em ambas as imagens.
- Eficiência computacional, está relacionada ao tempo que o algoritmo de detecção necessita para identificar as características de uma imagem.

- Robustez ao ruído e desfocagem
- Distinção , de modo que as características possam ser correspondidas precisamente entre imagens diferentes.
- Invariância à iluminação e mudanças geométricas, garante que a característica será identificada igualmente com ou sem a transformação.

Na área de OV os detetores de características, tais como *cantos*<sup>2</sup> ou *blobs*<sup>3</sup>, são muito importantes porque é possível saber com precisão a sua posição numa imagem.

Entre estes dois tipos de detetores, os detetores de canto são computacionalmente mais rápidos mas menos distintos. Adicionalmente, detetores de canto estão melhores localizados em posições de imagens mas são menos localizados em escala. Isto significa que os detetores de canto podem não ser tão bem detetados que os detetores de *blob* quando existem grandes variações de escala e pontos visíveis. Contudo, detetores de *blob* não são bons em certos ambientes [14].

### 2.4.1 Detetores de Cantos

Um canto em uma imagem é o ponto onde há uma grande variação de intensidade dos *pixels* em duas direções dominantes.

#### 2.4.1.1 Harris

O detetor de cantos Harris é um dos métodos mais recentes (1988). A ideia base no algoritmo de Harris é analisar um ponto através das características e de uma pequena vizinhança. Através da alteração da janela em várias direções do ponto , a média da intensidade da janela deve alterar significativamente se o ponto for um canto. Os cenários possíveis são ilustrados na Figura 2.7

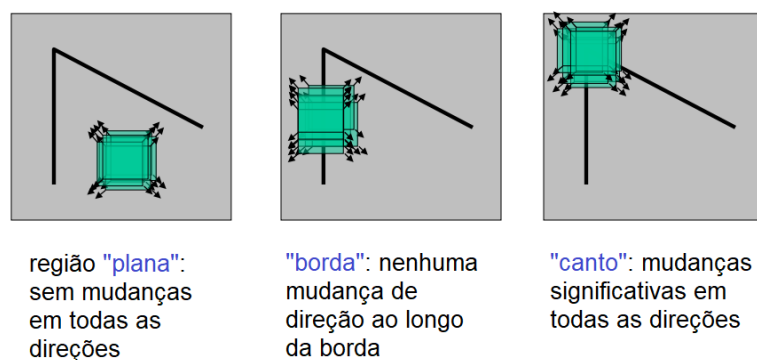


Figura 2.7: Detetor de cantos Harris. [15]

Desde que a janela seja alterada em várias direções , o algoritmo deve obter o mesmo resultado mesmo que a imagem tenha sofrido uma rotação. O mesmo não acontece caso a imagem sofra

<sup>2</sup>é definido como um ponto de intersecção de duas ou mais arestas.

<sup>3</sup>são regiões mais brilhantes (ou mais escuras) do que o meio circundante.

um zoom , o algoritmo não fornece os mesmos resultados que a imagem original. Assim, o algoritmo é não-invariante à escala. Além de ser computacionalmente pesado no cálculo da média da intensidade da janela.

#### 2.4.1.2 FAST

O detetor de canto FAST (do inglês , *Feature from Accelerated Segment Test*) é um detetor rápido. O algoritmo usa um círculo de 16 pixels com um raio de 3 pixels em volta do pixel  $p$  , de forma a detetar se  $p$  é um canto, ver Figura 2.8.

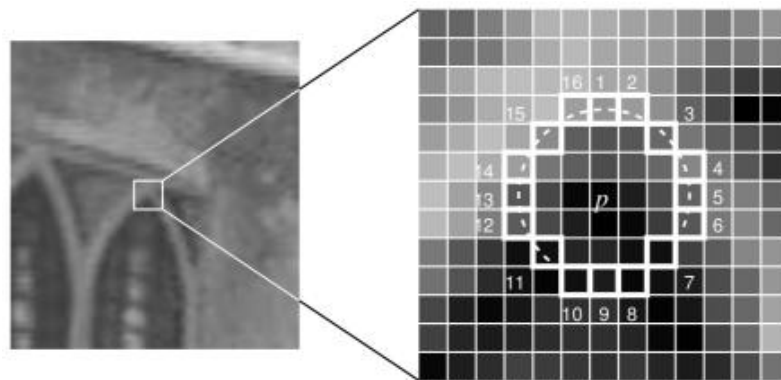


Figura 2.8: Detetor de canto FAST, círculo em volta do ponto  $p$ . [15]

Analisando a Figura 2.8 , os pixels 11-16 e 1-6 têm maior intensidade que os pixel a examinar, pixel  $p$ . Isto significa que o pixel  $p$  é um canto. O teste realizado pode ser resumido a duas condições:

1. Se existirem  $N$  pixels adjacentes no anel na qual todos têm intensidade superior ao pixel  $p$ , então  $p$  é um canto.
2. Se existirem  $N$  pixels adjacentes no anel na qual todos têm intensidade inferior ao pixel  $p$ , então  $p$  é um canto.

Assim se alguma condição for verdadeira , o pixel  $p$  é classificado como canto. O parâmetro  $N$  é usualmente 12 de forma a obter uma detecção com alta qualidade.  $N$  pode ser 9 de forma a detecção ser mais rápida. O algoritmo depende da intensidade de threshold , o qual em diferentes ambientes deve originar diferentes resultados e o threshold deve ser ajustado para a melhor performance. Desde que o algoritmo use um círculo de pixels para determinar se o pixel examinado é um canto, o algoritmo é invariante à escala e rotação.

### 2.4.2 Detectores de blobs

#### 2.4.2.1 SIFT

A diferença de gaussianas (DoG, do inglês *Difference of Gaussians*) foi utilizada para identificar características invariantes à escala, rotação e parcialmente invariantes à variação na luminosidade

e transformações afins. O método SIFT (do inglês, *Scale Invariant Feature Transform*) é composto por um detetor de blobs baseado em DoG.

O primeiro passo no algoritmo de SIFT é utilizar a função gaussiana sobre uma imagem com escalas diferentes para gerar o espaço de escalas da imagem.

A função

$$L(x, y, \sigma) = G(x, y, \sigma) * I_{x,y} \quad (2.1)$$

define o espaço de escalas da imagem  $I_{x,y}$ . Na equação 2.1, o operador  $*$  define a convolução da função gaussiana onde  $I_{x,y}$  é a imagem de entrada e o núcleo Gaussiano  $G(x, y, \sigma)$  é igual a

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}.$$

O passo é repetido com diferentes  $\sigma$ , escalados por um fator  $k$ .

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma).$$

Desde que todas as DoG estão calculadas para a escala corrente, designada como oitava, a imagem cujo valor de  $\sigma$  é duas vezes o valor de  $\sigma$  inicial na oitava é selecionada para ser reproduzida. O processo é ilustrado na Figura 2.9.

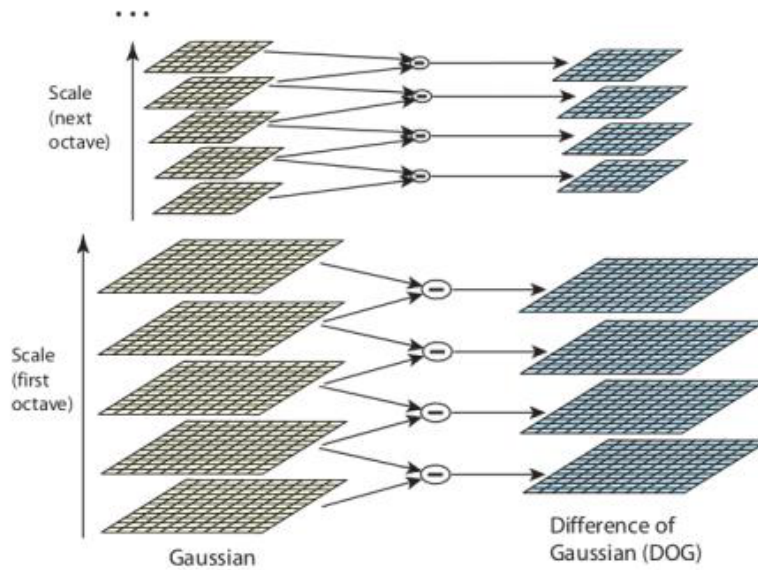


Figura 2.9: Computação da DoG. [15]

Calculadas as DoGs, deseja-se obter os máximos e mínimos locais em todas as escalas de  $D(x, y, \sigma)$ . Para isso avalia-se cada ponto da primeira oitava com seus 26 vizinhos, 8 no mesmo nível e 9 nos níveis superiores e inferiores de escala. A Figura 2.10 apresenta essa comparação. Se o ponto for extremo nessa oitava, sua posição é estimada na oitava seguinte e o processo é repetido para esse ponto. As informações referentes à escala e oitava alcançadas são guardadas e farão parte da característica.

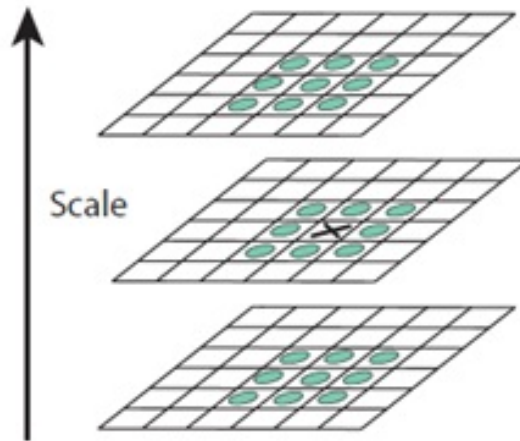


Figura 2.10: O *pixel* marcado com um  $x$  é avaliado em relação aos seus vizinhos no mesmo nível e nos níveis superiores e inferiores. [15]

Uma vez obtido o ponto de extremo é refinada a posição para obter uma posição em *subpixels*. É utilizada uma expansão de Taylor em torno do ponto de extremo

$$D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x \quad (2.2)$$

onde  $c$  é o deslocamento em torno do ponto de extremo. O ponto extremo  $\hat{x}$  é determinado derivando a equação 2.2 em relação a  $x$  e igualando a zero. O resultado é dado por

$$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial^2 x^2} \frac{\partial D}{\partial x}. \quad (2.3)$$

Se houver variação maior que 0.5 em qualquer uma das direções o ponto muda de pixel e o valor de extremo é interpolado no novo ponto. O valor do ponto de extremo é refinado para remover extremos com baixo contraste. Essa operação é feita substituindo a equação 2.3 na equação 2.2, resultando em

$$D(\hat{x}) = D + \frac{1}{2} \frac{\partial D^T}{\partial x} \hat{x}.$$

Outro filtro usa uma ideia semelhante para remover pontos de borda. A matriz hessiana

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

descreve a curvatura principal da imagem em torno do ponto de extremo. Analisando a relação entre os autovalores de  $H$  avalia-se que se o determinante de  $H$  é negativo o ponto é descartado. Assim a avaliação é feita sobre a relação dos autovalores, como mostra a seguinte equação:

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha + \beta)^2}{\alpha\beta} = \frac{(r\beta + \beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}, \quad (2.4)$$

onde  $\alpha = r\beta$ . Um valor extremo será descartado caso a relação

$$\frac{Tr(H)^2}{Det(H)} < \frac{(\tau + 1)^2}{\tau},$$

para um limiar  $\tau$  a ser definido.

Até este ponto foram obtidas localizações e escalas dos pontos de extremos e foram removidos extremos com baixo contraste e em bordas. Resta obter uma orientação para o ponto. Para isso escolhe-se em cada oitava a imagem gaussiana  $L$  cuja escala mais se aproxima da escala do ponto de extremo. Para cada uma das imagens  $L(x, y)$  (note que o valor de  $\sigma$  não aparece pois é definido como o mesmo do ponto de extremo) a magnitude  $m(x, y)$  e a orientação  $\theta(x, y)$  são calculadas usando

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

$$\theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y))).$$

Calcula-se as orientações em torno do ponto de extremo e um histograma é montado com 36 orientações possíveis para a característica, cada um respondendo por  $10^\circ$  dos  $360^\circ$  possíveis para  $\theta(x, y)$ . Cada uma das orientações  $\theta(x, y)$  é pesada utilizando a magnitude  $m(x, y)$  e uma janela gaussiana com  $\sigma$  sendo 1.5 vezes a escala do ponto de extremo antes de ser adicionada à orientação da característica.

Picos no histograma das orientações em torno do ponto de extremo correspondem a direções dominantes do gradiente local. O maior pico de histograma é identificado e uma característica é gerada com localização, escala  $a$  e orientação. Picos no histograma que sejam máximos locais e cuja magnitude seja maior que 80% da do pico máximo também geram características com a mesma localização e escala, mas com a orientação diferente. Por fim, para cada pico que gerou uma característica, uma parábola é traçada pelo pico e os dois valores do histograma adjacentes a ele. Para se obter maior precisão o pico máximo é então tomado como o máximo da parábola gerada pela interpolação dos três picos.

#### 2.4.2.2 SURF

O detetor SURF (do inglês, *Speeded Up Robust Features*) é similar ao SIFT no sentido que os passos são iguais, mas são feitos diferentemente. A imagem em análise é filtrada com janelas baseada no método do integral da imagem, como uma aproximação Gaussiana

$$I_\Sigma(x, y) = \sum_i^x \sum_j^y I(i, j). \quad (2.5)$$

O benefício do uso deste método é a redução do numero de cálculos e a diferença no tamanho da janela que influenciam no tempo de cálculos. Uma adição e duas subtrações usando os pontos de canto são requeridas para o calculo da soma, Figura 2.11.

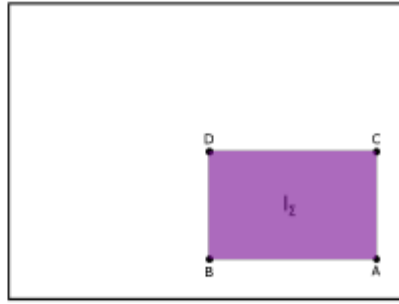


Figura 2.11: Ilustração do método do integral da imagem. [15]

A soma de todos os pixels como visto em 2.5 é igual a  $I_2 = A - B - C + D$ , dado que a origem da imagem original é no canto superior esquerdo. Detecção blob é usado e é realizada usando a matriz Hessian. Através de uma imagem  $I$  e um ponto  $\mathbf{x} = (x, y)$  na imagem, a matriz Hessian  $H(x, \sigma)$  é calculada. Onde  $\sigma$  é a escala atual.

$$H(x, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix}.$$

Onde  $L_{xx,xy,yy}(x, \sigma)$  são as derivadas de segunda ordem no ponto  $x$  da imagem  $I$ . Estas derivadas parciais são aproximadas usando filtros para diferentes escalas e oitavas. Depois de suavizar e reduzir a amostragem da imagem para cada  $\sigma$  como no algoritmo SIFT, o algoritmo SURF aumenta o tamanho do filtro. Uma vez que o Hessian é formado para o ponto  $\mathbf{x}$  numa certa escala  $\sigma$ , o determinante do Hessian é calculado e ponderado de forma a obter a melhor aproximação. O determinante aproximado é guardado em um mapa de resposta para a escala atual. Uma vez calculada a matriz Hessian para todas as oitavas, o algoritmo procura o máximo local e o máximo detetado é guardado e interpolado na escala e no espaço da imagem.

Assim, o detetor de SURF têm pontos chaves:

- Localização
- Escala
- Orientação

## 2.5 Descritor de Características

Como mencionado anteriormente, uma boa característica tem como propriedade a diferenciabilidade. A diferenciabilidade permite que a característica seja identificada de maneira única. Porém se for utilizado somente o ponto de interesse onde localiza-se a característica, a diferenciação torna-se difícil, uma vez que existe pouca informação para gerar um identificador para aquele ponto.

Desta forma, os descritores codificam uma área da imagem em torno do ponto característico, para cada ponto característico. Ao fazer isto, os pontos característicos entre duas imagens são comparáveis uns com os outros.

### 2.5.1 ORB

O descritor ORB (do inglês, *Oriented FAST and Rotation-Aware BRIEF*) precisa de pontos característicos para serem detetados pelo detetor oFAST. A deteção é realizada como descrita na secção 2.4.1.2 com a adição da orientação do ponto característico. Isto é realizado calculando o centroide de intensidade. Primeiro, o momento do patch é calculado como

$$m_{pq} = \sigma x^p y^q I(x, y),$$

onde  $p$  e  $q$  são parâmetros dos pixels em ordem do momento. E o centroide do patch é encontrado com a seguinte equação

$$C = \left( \frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right).$$

Em seguida o vetor  $\vec{OC}$  é construído do ponto central do ponto de característica, O, para o centroide obtido anteriormente, C. O ângulo de  $\vec{OC}$  é obtido com a orientação do patch.

$$\theta = \text{atan}^2(m_{01}, m_{10}).$$

Uma vez atribuída uma orientação ao ponto característico, o descritor BRIEF (do inglês, *Binary Robust Independent Elementary Features*) realiza um teste binário,  $\tau$ , da intensidade entre pontos de um patch de imagem suavizada. O teste é definido como

$$\sigma(p; x, y) = \begin{cases} 1, & p(x) < p(y) \\ 0, & p(x) \geq p(y) \end{cases},$$

onde  $\mathbf{p}$  é o patch de imagem que foi suavizada e  $(x, y)$  é o ponto de teste. O resultado da característica é um vetor de 256 testes binários

$$f_{256}(p) = \sum_{1 \leq i \leq 256} 2^{i-1} \tau(p; x_i, y_i).$$

De forma a realizar um descritor de recurso elementares independentes robustos binários (BRIEF) invariantes à rotação, é orientar o descritor para a orientação  $\theta$  dos pontos caraterísticos. Isto é feito construindo uma versão direcionada dos testes binários na localização do recurso

$$\mathbf{S}_\theta = \mathbf{R}_\theta \mathbf{S},$$

onde  $\mathbf{R}_\theta$  é a matriz rotação e  $\mathbf{S}$  é igual a :

$$\mathbf{S} = \begin{bmatrix} \mathbf{x}_1 & \dots & \mathbf{x}_{256} \\ \mathbf{y}_1 & \dots & \mathbf{y}_{256} \end{bmatrix}.$$



Desta forma , o BRIEF dirigido é igual a :

$$g_n(\mathbf{p}, \theta) = f_n(\mathbf{p}) |(\mathbf{x}_i, \mathbf{y}_i) \in S_\theta.$$

Uma desvantagem com o BRIEF dirigido é a baixa variação e alta correlação entre os testes binários. Isso é reduzido pela busca entre os testes binários para testes com alta variação e não correlação.

### 2.5.2 SIFT

Primeiro o gradiente e a orientação são analisados para os pontos numa janela que é centrada pelo ponto característico. O tamanho da janela é  $16 \times 16$  caixas em uma matriz. Esta matriz é dividida em  $4 \times 4$  sub-regiões, onde o histograma baseado na orientação é construído para cada região. Em seguida, as coordenadas desses pontos e orientações do gradiente são rodadas de forma a incorporar a invariância rotacional. Os pontos são futuramente aperfeiçoados por uma função Gaussiana com peso  $\sigma = \frac{w}{2}$ , onde  $w$  é o tamanho da janela do descritor. Como ilustra a Figura 2.12

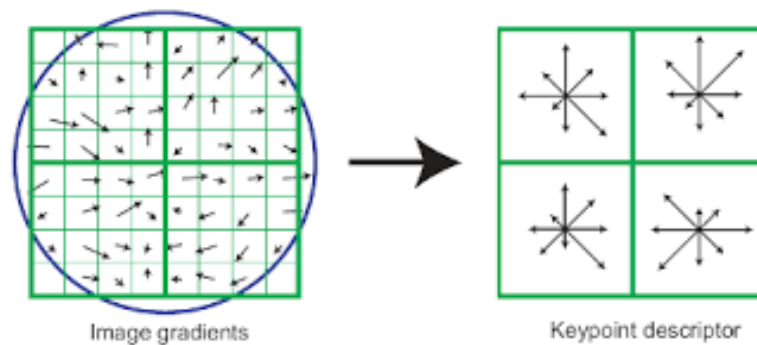


Figura 2.12: Descritor SIFT. As setas designam a soma dos gradientes nas caixas individuais. O círculo azul representa a região que é considerada na atuação do filtro Gaussiano. Este exemplo ilustra uma matriz  $8 \times 8$  resultando num  $2 \times 2$  descritor. [15]

### 2.5.3 SURF

Uma janela quadrada é centrada no ponto de característica, com a mesma orientação. O tamanho da janela é  $20s \times 20s$ , onde  $s$  é a escala onde o ponto característico foi detetado. De seguida, a janela é dividida em sub-regiões  $4 \times 4$ . As respostas das ondas de Haar e seu valor absoluto são resumidas para direção horizontal e vertical em cada sub região. O descritor resultante consiste em um vetor 4D que contém as quatro somas para cada sub-região.

## 2.6 Associação de Características

O processo de correspondência de características é um requisito para muitas aplicações relacionadas com imagens, tais como, recuperação de uma imagem, detecção do movimento e reconstrução da forma. Como mencionado no início da Seção 2.4, espera-se que uma boa característica tenha como propriedade a diferenciabilidade. A diferenciabilidade permite que a característica seja identificada (idealmente) de maneira única. As diferentes características determinadas pela etapa anterior precisam ser correspondidas, ou seja, é preciso determinar quais características são as mesmas em imagens distintas. Geralmente existem dois métodos diferentes para encontrar suas correspondências: correspondência de características e seguimento de características.

### 2.6.1 Correspondência de Características

O método de correspondência de características consiste na detecção de características de maneira independente em todas as imagens com um determinado detetor de características. Idealmente, as correspondências detetadas representam a mesma informação nas imagens. Depois são efetuadas as correspondências entre as características detetadas usando uma estratégia de comparação baseada em medidas de semelhança. Este método é muito usado em ambientes exteriores de grandes dimensões, em que as imagens são capturadas partir de posições distantes entre si, com a finalidade de limitar os problemas relacionados com desvio de movimento.

#### 2.6.1.1 Brute force

O método *Brute force* compara um descritor de um conjunto de pontos característicos da primeira imagem com todos os descritores dos pontos característicos da segunda imagem. Geralmente, o descritor com pequena distância Euclidiana é comparado com o descritor da primeira imagem. A distância do descritor é limitada por um threshold de forma a tornar a comparação válida. O tempo entre duas imagens devem ser pequenas de forma à comparação do descritor ser correta.

### 2.6.2 Seguimento de Características

O método de seguimento de características, com alternativa, deteta características em apenas uma imagem com um determinado detetor de características. Na próxima imagem a característica correspondente é procurada na área em torno da localização da característica detetada. Este método de detecção e seguimento é muito usado em ambientes interiores de pequenas dimensões e é adequado para aplicações de OV. Uma vez que as imagens são capturadas a partir de posições próximas e a quantidade de movimentos entre imagens sucessivas geralmente é pequena.

#### 2.6.2.1 FLANN

O método FLANN (do inglês, *Fast Library for Approximate Nearest Neighbours*) é considerado um algoritmo que procura correspondência dos descritores numa vizinhança próxima do ponto característico. O OpenCV utiliza uma correspondência baseada no FLANN que utiliza a estrutura

de dados da árvore k-d e vizinhos próximos. K-d significa  $k$  dimensões, onde  $k$  é um inteiro positivo. A ideia base é :

- Escolher uma dimensão  $k$ .
- Encontrar o valor mediano dessa dimensão.
- Dividir a dimensão em duas partes com base no valor mediano.
- Repetir, até  $k-1$  atingir 0 e começar de novo com  $k$  original até que todos elementos no conjunto de dados sejam examinados.

O processo é ilustrado na figura 2.13 . Este exemplo inicia-se no ponto (7,2).

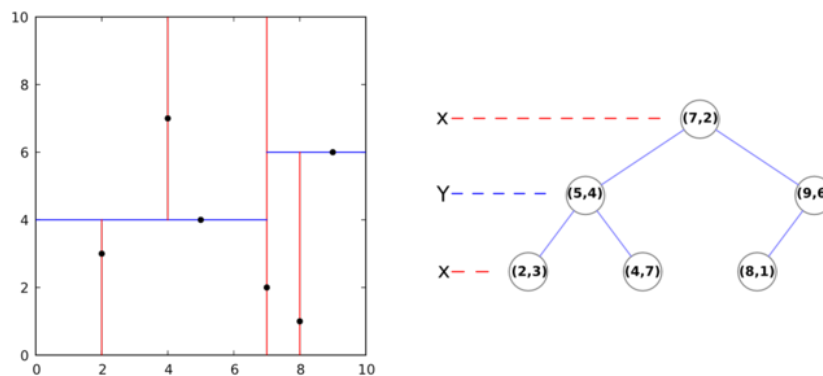


Figura 2.13: A imagem da esquerda ilustra a decomposição 2-d, note que as linhas vermelhas são para o eixo x e as linhas azuis para o eixo y. A imagem da direita ilustra a árvore 2-d correspondente.

Uma vez que a árvore k-d é formada, o próximo passo é usar pontos do segundo conjunto de dados e ver em qual nó da árvore k-d que o ponto examinado está mais próximo. O procedimento é chamado vizinho mais próximo (NN, do inglês, *Nearest Neighbour*). Começando no nó raiz (7,2), como ilustrado, o algoritmo percorre a árvore 2-d construída recursivamente. Escolhe o nó da esquerda ou direita se o ponto examinado tiver um valor maior ou menos que o nó atual na dimensão atual, respectivamente. Uma vez que o algoritmo tenha atingido um nó final, o algoritmo guarda esse nó como o melhor atualmente. A recursão é então finalizada e o algoritmo começa a atravessar de volta ao nó raiz. A distância até ao ponto examinado é verificada em cada nó, se a distância for menor então é atualizado o melhor valor. O algoritmo também verifica se o hiperplano do outro lado da árvore está dentro do raio da menor distância atual, se não , esse hiperplano é descartado. Se estiver dentro, o algoritmo percorre esse ramo da árvore k-d também. O procedimento pode ser visto na figura ilustrada 2.14.

O ponto (2,8) é examinado para o seu vizinho mais próximo, marcado com uma estrela na figura. O algoritmo começa no nó raiz (7,2) e a distância , isto é , o raio do grande círculo centrado na estrela cobre todos os hiperplanos. Isso significa que não podemos descartar nenhum

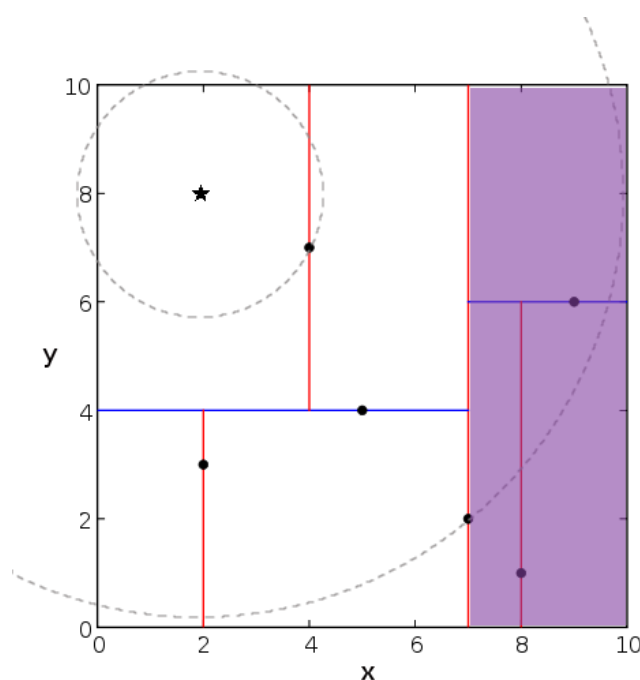


Figura 2.14: Algoritmo de procura NN baseado nas árvores 2-d.

hiperplano. O algoritmo percorre a árvore, ignorando os hiperplanos roxos da figura. O nó da folha (4,7) é definido como o melhor e o algoritmo começa a atravessar de volta para o nó raiz, verificando a distância em cada nó. Desde que o raio do pequeno círculo centrado em torno da estrela é a melhor combinação atual e não está se cruzando com os hiperplanos roxos, pode-se descartar esses hiperplanos e o algoritmo não procura por eles. Uma vez no nó raiz, o algoritmo termina e (4,7) era de fato o mais próximo vizinho.

## 2.7 Estimação do movimento

A etapa de estimação do movimento representa o núcleo da computação em um sistema de OV. Ela efetua o cálculo do movimento da câmara entre a imagem atual e a imagem anterior. A trajetória da câmara e do agente (assumindo que estão ligados rigidamente) pode ser recuperada completamente através da concatenação de todos os movimentos individuais.

Esta secção explica com a transformação relativa  $T_k$  pode ser calculada entre duas imagens  $I_{k-1}$  e  $I_k$  a partir de dois conjuntos de pontos característicos correspondentes  $f_{k-1}$ ,  $f_k$  nos instantes de tempo  $k-1$  e  $k$ , respetivamente. Existem três métodos diferentes para o cálculo do  $T_k$  dependendo se as correspondências dos pontos característicos são especificadas em três ou duas dimensões. Os métodos referidos são:

- 2D-para-2D: Neste caso, ambos  $f_{k-1}$  e  $f_k$  são especificados pelas coordenadas 2D da imagem.

- 3D-para-3D: Neste caso, ambos  $f_{k-1}$  e  $f_k$  são especificados pelas coordenadas 3D da imagem. Para realizar esta acção, é necessário a triangulação de pontos 3D em cada instante de tempo, isto pode ser feito a partir de um sistema com câmaras estéreo, por exemplo.
- 3D-para-2D: Neste caso,  $f_{k-1}$  são especificados em 3D e  $f_k$  são as suas correspondentes reprojeções em 2D na imagem  $I_k$ . No caso da utilização de um sistema monocular, a estrutura 3D necessita de triangular duas câmaras adjacentes (isto é  $I_{k-2}$  e  $I_{k-1}$ ) e assim a construção da imagem 2D.

### 2.7.1 2D-para-2D

A relação geométrica entre duas imagens  $I_k$  e  $I_{k-1}$  da calibração da câmara é descrita pela matriz essencial  $E$ . Esta contém os parâmetros de movimento da câmara para um fator de escala desconhecido.

$$E_k \approx \hat{t}_k R_k$$

onde  $t_k = [t_x, t_y, t_z]^T$  e

$$\hat{t}_k = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}.$$

A correspondência dos pontos característico, rotações e translações podem ser obtidos através da matriz essencial. A principal propriedade da estimação de movimento através de 2D-para-2D é a restrição epipolar, que determina a linha em cada ponto de característica correspondente em outra imagem, como ilustrado na Figura 2.15.

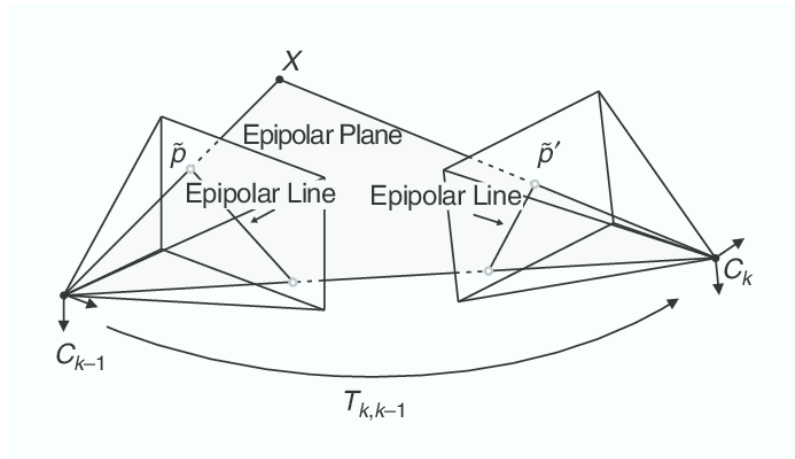


Figura 2.15: Uma ilustração da restrição epipolar. [8]

Esta restrição pode ser formulada por  $p'^T E p = 0$ , onde  $p$  é um ponto característico numa imagem e  $p'$  é o correspondente ponto característico noutra imagem.

Através da estimação da matriz  $E$ , a rotação e translação pode ser extraída.

$$R = U(\pm W^T)V^T,$$

$$\hat{t} = U(\pm W)SU^T,$$

onde

$$W^T = \begin{bmatrix} 0 & \pm 1 & 0 \\ \mp 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

e U, S, V são obtidas através do USV (do inglês, *Singular Value Decomposition*)

$$E = USV^T$$

Para recuperar a trajetória de uma sequência de imagens é necessário a concatenação das diferentes transformações  $T_{0:n}$ . Para isso é necessário uma escala relativa obtida através de :

$$r = \frac{\|X_{k-1,i} - X_{k-1,j}\|}{\|X_{k,i} - X_{k,j}\|}$$

Desta forma, o algoritmo de OV com correspondência 2D-para-2D é sumariado de seguida:

1. Capturar uma imagem  $I_k$ .
2. Extrair e conjugar pontos característicos entre  $I_{k-1}$  e  $I_k$ .
3. Obter a matriz essencial do par de imagens  $I_{k-1}, I_k$ .
4. Decompor a matriz essencial em  $R_k$  e  $\hat{t}_k$ .
5. Computorizar a escala relativa  $\hat{t}_k$ .
6. Concatenar a transformação  $C_k = C_{k-1} T_k$ .
7. Repetir 1.

### 2.7.2 3D-para-3D

A estimação do movimento através da correspondência 3D-para-3D é usada em sistemas estéreo.

A solução geral é encontrar  $T_k$  que minimiza a distância  $L_2$  entre duas 3D características

$$\arg \min_{T_k} \sum_i \|\tilde{X}_k^i - T_k \tilde{X}_{k-1}^i\|$$

onde  $i$  é a característica  $i$  e  $X_k, X_{k-1}$  são as coordenadas homogêneas dos pontos 3D.

Como analisado anteriormente, o número mínimo de soluções são três correspondências não colineares. Desta forma, se  $n \geq 3$ , uma solução possível é realizar a diferença dos centroides das características 3D e uma rotação usando SVD. A translação é dada por

$$t_k = \bar{X}_k - R\bar{X}_{k-1}$$

onde  $\bar{\cdot}$  é a média aritmética.

A rotação é eficientemente calculada através do SVD como

$$R_k = VU^T$$

onde  $USV^T = \text{svd}((X_{k-1} - \bar{X}_{k-1})(X_k - \bar{X}_k)^T)$  e  $X_k$  corresponde ao ponto 3D.

O algoritmo de OV com correspondência 3D-para-3D é sumariado de seguida:

1. Capturar duas pares de imagens num sistema estéreo,  $I_{l,k-1}, I_{r,k-1}$  e  $I_{l,k}, I_{r,k}$
2. Extrair e conjugar pontos característicos entre  $I_{l,k-1}$  e  $I_{l,k}$
3. Triangular os pontos característicos conjugados para cada par de estéreo.
4. Computacional  $T_k$  de pontos característicos 3D,  $X_{k-1}$  e  $X_k$
5. Concatenar a transformação  $C_k = C_{k-1} T_k$
6. Repetir 1

### 2.7.3 3D-para-2D

A estimação do movimento de 3D-para-2D é mais precisa do que 3D-para-3D porque são minimizados os erros de reprojeção. A transformação  $T_k$  é analisada num sistema estéreo através do  $X_{k-1}$  e  $p_k$ . No caso de um sistema monocular, através da triangulação das medições de imagens  $p_{k-1}$  e  $p_{k-2}$ .

A formula general neste caso é encontrar  $T_k$  que minimize o erro de reprojeção da imagem

$$\arg \min_{T_k} \sum_i \|p_k^i - \hat{p}_{k-1}^i\|^2$$

onde  $p_{k-1}$  é a reprojeção do ponto 3D  $X_{k-1}^i$  na imagem  $I_k$  de acordo com a transformação  $T_k$ .

Como no problema da correspondência 3D-para-3D são necessários um número mínimo de pontos. Com um número de pontos igual a 6,  $P$  é calculado através do uso de SVD e a rotação e translação é obtida através de  $P_k = [R|t]$ .

Assim, a estimação através de 3D-para-2D assume que os pontos são obtidos de apenas uma câmara. Desta forma, num sistema estéreo o ponto 2D é obtido através da junção da câmara da esquerda com a câmara da direita. Obviamente a captura das imagens têm de ser no mesmo instante de tempo. Para o caso de um sistema monocular é algoritmo inicializa após a captura das duas primeiras imagens, desta forma é necessário triangular duas imagens capturadas em instantes diferentes para obter o ponto.

O algoritmo de OV com correspondência 3D-para-2D é sumariado de seguida:

1. Realizar na primeira vez:
  - 1.1. Capturar duas imagens  $I_{k-2}, I_{k-1}$

- 1.2. Extrair e conjugar pontos característicos entre eles
- 1.3. Triangular os pontos característicos  $I_{k-2}, I_{k-1}$
2. Realizar a cada iteração:
  - 2.1. Capturar nova imagem  $I_k$
  - 2.2. Extrair e conjugar pontos característicos com a imagem anterior  $I_{k-1}$
  - 2.3. Computorizar posição da câmara através do calculo 3D-para-2D
  - 2.4. Triangular todos os novos pontos característicos entre  $I_k$  e  $I_{k-1}$
  - 2.5. Repetir 2.1

## 2.8 Métodos de ajuste de erros

A associação de pontos , usualmente, está contaminada por *outliers*<sup>4</sup>, isto é, associação de dados errada. As possíveis causas de *outliers* são ruídos da imagem, exclusões, pouca nitidez, alteração de posições e iluminação das quais os modelos matemáticos dos detetores de características não detetam.

Para a estimação da movimentação correta da câmara é importante que o *outliers* sejam removidos.

### 2.8.1 Windowed bundle adjustment

O *bundle adjustment* é um algoritmo criado no campo da fotometria em meados da década de 1950 e tornou-se muito utilizado no campo da visão por computador, explicitamente na área de reconstrução 3D.

A função principal é tentar otimizar ao mesmo tempo os parâmetros (intrínsecos e extrínsecos) da câmara, bem como os parâmetros dos pontos 3D de referência. Ele é aplicado para os casos em que as características detetadas numa imagem são procuradas em mais do que duas características. Este algoritmo considera uma "janela" de  $n$  características da imagem e depois faz uma otimização dos parâmetros das posições da câmara e dos pontos 3D de referência para este conjunto de características da imagem.

No *bundle adjustment*, a função de erro a minimizar é o erro de reprojeção da imagem.

$$\arg\min_{X^i, C_k} \sum_{i,k} \|p_k^i - g(X^i, C_k)\|^2,$$

onde  $p_k^i$  é o  $i$ -ésimo ponto da imagem dos pontos 3D de referência  $X^1$  medido na  $k$ -ésima imagem e  $g(X^i, C_k)$  é a sua imagem de reprojeção de acordo com a posição atual da câmara  $C_k$ . O erro de reprojeção é uma função não-linear.

Para concluir, o principal objetivo do *bundle adjustment* em sistemas de OV é a redução do desvio entre a trajetória estimada e a real.

<sup>4</sup>valor atípico, é uma observação que apresenta um grande afastamento das demais series ou é inconsistente.



### 2.8.2 RANSAC

Tal como referido anteriormente, uma das funções do algoritmo RANSAC é a remoção dos *outliers*, que permite a estimação do movimento da câmara de uma forma mais precisa. A etapa de rejeição dos *outliers* é a mais delicada em sistema de OV.

De uma forma geral, este algoritmo é utilizado quando se pretende estimar um modelo (por exemplo os parâmetros de rotação e translação de uma câmara) na presença de *outliers*.

A ideia fundamental do RANSAC é calcular candidatos modelo a partir de amostras de conjuntos de pontos de dados. A escolha das amostras é feita aleatoriamente. Posteriormente com os outros dados é seleccionada como solução.

O esboço deste algoritmo é apresentado em seguida:

O número de iterações  $N$  necessário para garantir a solução correta é calculado pela equação

$$N = \frac{\log(1 - p)}{\log(1 - (1 - \varepsilon)^s)}$$

O valor  $s$  representa o número de pontos de dados a partir do qual o modelo pode ser instanciado,  $\varepsilon$  é a percentagem de *outliers* nos pontos de dados e  $p$  representa a probabilidade de sucesso pretendida.

Notar que o algoritmo é um método iterativo e não determinístico de tal forma que calcula uma diferente solução em cada execução. Contudo, a solução tende a estabilizar à medida que o número de iterações aumenta.



## Capítulo 3

# Sistema de Odometria Visual

Neste capítulo serão apresentados os procedimentos implementados para a realização dos testes. O capítulo aborda o *software* e *hardware* utilizados no desenvolvimento.

### 3.1 Software

Neste subcapítulo serão descritas as ferramentas de *software* utilizadas assim como o código produzido.

#### 3.1.1 ROS

ROS (*Robot Operating System*) é uma *framework* para desenvolvimento de *software* de robótica. ROS consiste num compêndio de bibliotecas, ferramentas e outros recursos que visa facilitar a criação de comportamentos complexos alcançando uma vasta gama de plataformas robóticas distintas.

O ROS fornece serviços idêntico a um sistema operativo, incluindo abstração de *hardware*, baixo nível de controlo, implementação de funcionalidades e mensagens que são transmitidas entre nós. Além disto, é constituído por bibliotecas e ferramentas para obter, criar, gravar e executar código em vários computadores.

Desta forma, ROS é caracterizado por:

- **Pacotes** - O *software* desenvolvido em ROS organiza-se em pacotes. Os pacotes são módulos que podem conter nós, bibliotecas independentes, *datasets*, ficheiros de configuração, elementos de *softwares* de terceiros, entre outros elementos que constituem o módulo. Os pacotes constituem a unidade base de compilação.
- **Nós** - Os nós são processos em ROS que executam computação, funcionando como programas. Os nós comunicam entre si através da publicação e subscrição de mensagens, serviços e através do *Parameter Server*. Um sistema robótico em execução utiliza um conjunto de nós que cooperam para o seu funcionamento. A arquitetura distribuída de ROS permite que os nós em execução não necessitem de operar no mesmo equipamento, possibilitando a

simbiose de diferentes plataformas comunicando entre si. O nó mestre pertence ao conjunto de nós laçando pelo *roscore*, e tem a função de possibilitar aos nós que se localizem uns aos outros permitindo o estabelecimento de comunicações.

- **Tópicos** - Os tópicos são os recipientes através dos quais os nós trocam mensagens entre si. Os tópicos utilizam um modelo de publicação/subscrição que separa a produção de informação do seu consumo. Os nós não têm conhecimento de outros nós que publiquem determinado tópico ou o subscrevam. Um nó apenas necessita de subscrever os dados específicos a um tópico ou de publicar dados num tópico específico. A estrutura permite a existência de múltiplos publicadores e subscritores associados a cada tópico.
- **Serviços** - A unidirecionalidade existente no modelo publicador/subscritor não possibilita interações do tipo "pedido/resposta". Para possibilitar este tipo de interações existem os serviços em ROS. Um serviço é composto por um par de mensagens, uma mensagem definida para o pedido e outra para a resposta. Um nó pode oferecer um serviço ativado por um nó cliente ao submeter a mensagem de pedido.

A biblioteca de ROS suporta o desenvolvimento ROS em linguagem C++ e Python.

### 3.1.1.1 Nó VisodoAgro

O algoritmo implementado neste nó pode ser resumido no diagrama de ação [3.1](#)

No primeiro frame o algoritmo lê a imagem e transforma em escala de cinzas. De seguida, calcula os pontos característicos da imagem. Visto que no primeiro frame não é possível realizar uma correspondência com o frame anterior, devido à inexistência deste, o algoritmo avança para o próximo frame. A partir do segundo frame os passos são sempre iguais. Assim, o algoritmo obtém o próximo frame e extraí a imagem em escala de cinzas. Seguidamente, caso se utilize o método de **brute force**, explicito no capítulo [2.6.1.1](#), calcula-se os pontos característicos deste frame que corresponderam aos pontos característicos do anterior. Caso o método seja **FLANN**, explicito no capítulo [2.6.2.1](#), utiliza-se uma pequena região à volta dos pontos característicos do frame anterior como forma de se encontrar o ponto característico do frame atual.

Obtida a matriz com as devidas correspondências é necessário calcular a matriz Fundamental. Esta, é obtida através de um loop em que escolhe um conjunto de correspondências aleatoriamente de forma a calcular a matriz fundamental. De seguida testa a qualidade dessa matriz Fundamental através do numero de inliers<sup>1</sup>, guardando o conjunto de correspondências com maior número de inliers. No fim calcula a matriz Fundamental com o melhor conjunto de correspondência.

Como representado na figura [3.2](#) o ponto  $x$  corresponde ao ponto no plano de imagem do frame anterior,  $x'$  corresponde ao ponto no plano de imagem do frame atual que corresponde os dois ao ponto  $X$  no mundo. Desta forma,  $Fx$  descreve uma linha (linha epipolar) na qual o ponto correspondente  $x'$  na outra imagem deve estar. Isto significa que, para todos os pares de pontos

<sup>1</sup> pontos onde está a maior concentração de pontos.

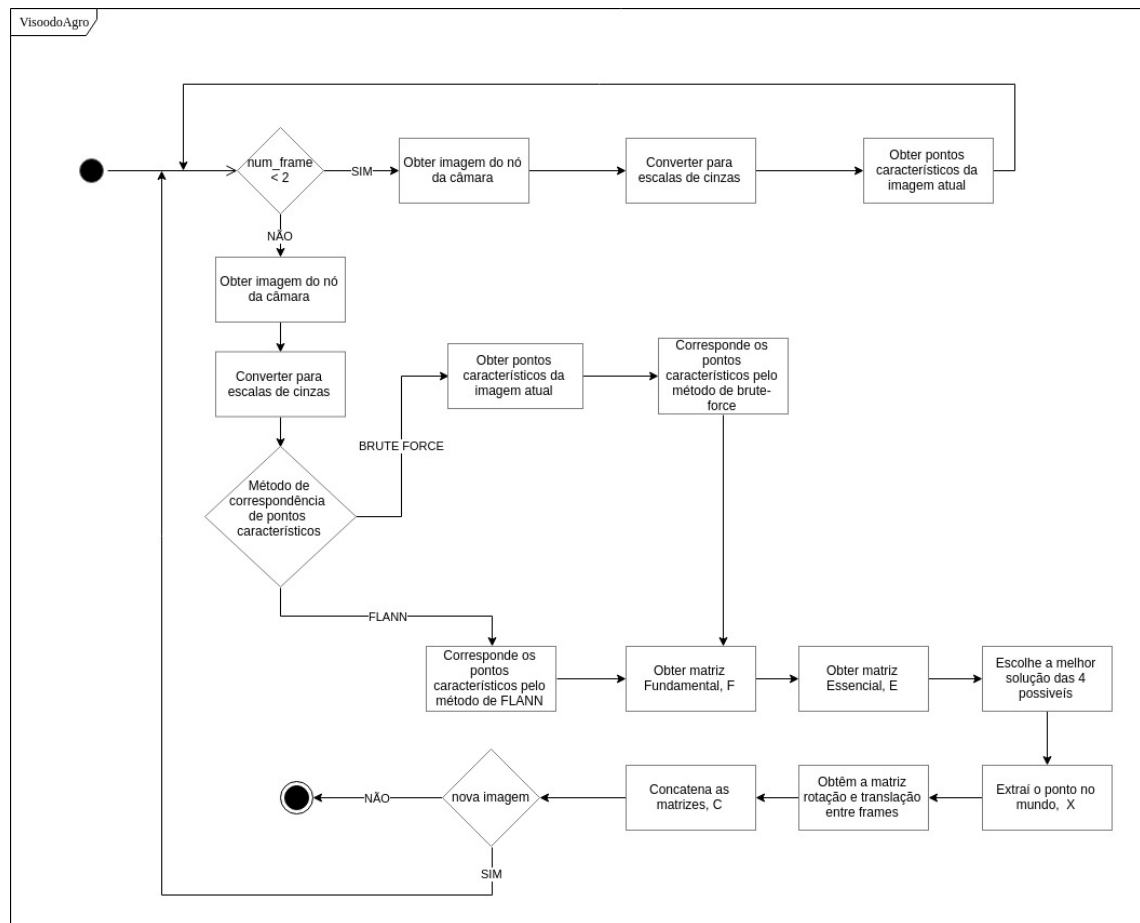


Figura 3.1: Diagrama de ação do nó implementado em ROS.

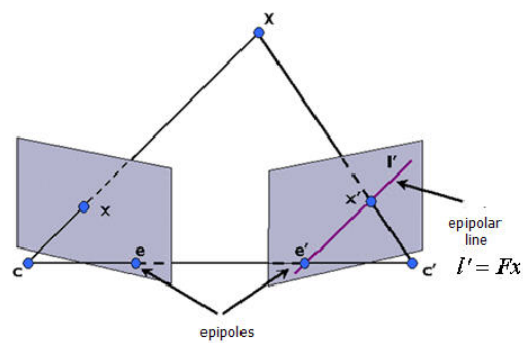


Figura 3.2: Representação da linha epipolar.

correspondentes:

$$x'^T F x = 0$$

sendo,

$$F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix},$$

$$x' = \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} \text{ e } x = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}.$$

Desenvolvendo, obtemos :

$$u'u f_{11} + u'v f_{12} + u' f_{13} + v'u f_{21} + v'v f_{22} + v' f_{23} + u f_{31} + v f_{32} + f_{33} = 0$$

$$\Leftrightarrow (u'u, u'v, u', v'u, v'v, v', u, v, 1) \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = 0$$

Para todos os pontos característicos :

$$\begin{bmatrix} u'_1 u_1 & u'_1 v_1 & u'_1 & v'_1 u_1 & v'_1 v_1 & v'_1 & u_1 & v_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u'_n u_n & u'_n v_n & u'_n & v'_n u_n & v'_n v_n & v'_n & u_n & v_n & 1 \end{bmatrix} \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = 0$$

$$\Leftrightarrow \mathbf{A}\mathbf{F} = \mathbf{0}. \quad (3.1)$$

Sendo a matriz  $\mathbf{A}$  invertível, singular e o número de pontos característicos superior a 8, a solução é obtida através da decomposição em valores singulares (SVD, do inglês *singular value decomposition*).

$$F = UDV^T.$$

Conseguida a matriz Fundamental e conhecida a matriz de parâmetros Intrínsecos,  $\mathbf{K}$ , a matriz

Essencial, que representa o plano epipolar na imagem 3.3, é obtida pela seguinte equação:

$$E = K'^T F K,$$

sendo

$$E = R \begin{bmatrix} t \\ \end{bmatrix}_x$$

, onde  $\mathbf{R}$  é a matriz rotação e  $\mathbf{t}$  é a matriz translação.

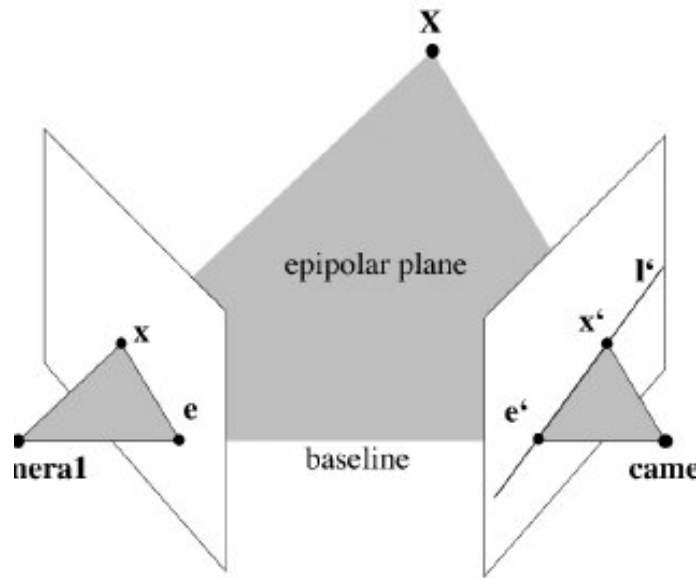


Figura 3.3: Representação do plano epipolar.

Sendo a matriz Essencial descrita desta forma, é possível obter os valores das matrizes rotação e translação através da decomposição dos valores singulares (SVD). Sendo :

$$E = U \text{diag}(1, 1, 0) V^T$$

onde as duas soluções possíveis para a matriz rotação,  $\mathbf{R}$ :

$$R = U W V^T$$

$$R = U W^T V^T$$

e a matriz translação,  $\mathbf{t}$ :

$$t = \pm u_3$$

.

Obtidas as matrizes rotação e translação existem 4 possíveis soluções, tal como verificado

anteriormente. Desta forma, é necessário escolher a solução certa. Para testar as hipóteses é realizada a triangulação por regressão ortogonal e escolhida a solução mais consistente.

Assim, é possível verificar que existe uma relação entre os pontos correspondentes entre imagens. Se um ponto desconhecido no mundo,  $\mathbf{X}$ , representado na frame anterior como  $\mathbf{x}$  e no frame atual como  $\mathbf{x}'$ , as coordenadas de  $\mathbf{X}$  podem ser calculadas. Isto, requer a matriz Intrínseca e a matriz Essencial entre frames.

A relação entre os pontos da imagem,  $\mathbf{x}$  e  $\mathbf{x}'$  e o ponto no mundo  $\mathbf{X}$  com os parâmetros da câmara  $P$  é expressa

$$\mathbf{x} = P\mathbf{X}$$

$$\mathbf{x}' = P'\mathbf{X}$$

, onde  $P$  e  $P'$   $\in \mathbb{R}^{3 \times 4}$  é a combinação da matriz Intrínseca e Essencial do frame anterior e atual, respetivamente.

Sendo,

$$\mathbf{x} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \mathbf{x}' = \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix}, \mathbf{X} = \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

,

$$E = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}, K = \begin{bmatrix} f_x & 0 & x_c \\ 0 & f_y & y_c \\ 0 & 0 & 1 \end{bmatrix}$$

.

E considerando  $p^{iT}$  as linhas da matriz  $P$  e devido à estrutura da matriz Intrínseca,  $K$ ,  $\mathbf{X}$  é expresso pela seguinte equação linear :

$$\begin{bmatrix} up^{3T} - p^{1T} \\ vp^{3T} - p^{2T} \\ u'p'^{3T} - p'^{1T} \\ v'p'^{3T} - p'^{2T} \end{bmatrix} \mathbf{X} = 0. \quad (3.2)$$

A equação 3.2 é resolvida através da decomposição em valores singulares como explicito anteriormente na equação 3.1.

Desta forma,

$$\mathbf{X} = UDV^T.$$

Onde,  $\mathbf{X} \in \mathbb{R}^{1 \times n}$  e  $n$  igual ao número de pontos característicos com correspondência. Através da matriz  $\mathbf{X}$  é obtida a matriz Rotação e Translação resultante da transformação .

Como ilustrado na figura 3.4 a escala entre frames é obtida pela razão,  $r$ , entre a distância de dois pontos em frames diferentes. Considerando  $x_{1,k-1}$ ,  $x_{2,k-1}$ ,  $x_{1,k}$ ,  $x_{2,k}$  pontos na imagem com



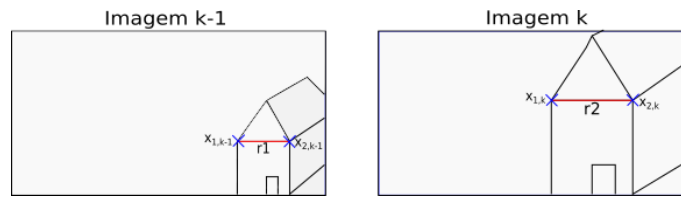


Figura 3.4: Representação da razão entre frames.

coordenadas (u,v) resulta :

$$r = \frac{\|x_{1,k-1} - x_{2,k-1}\|}{\|x_{1,k} - x_{2,k}\|}$$

.

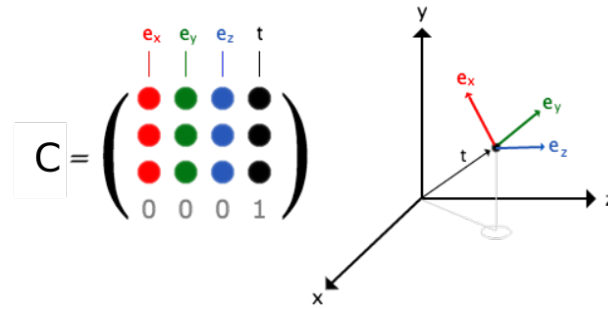


Figura 3.5: Representação da transformação homogênea.

Desta forma, considerando a matriz rotação entre frames

$$R_f = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix}$$

e a matriz translação entre frames

$$t_f = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

Onde , a matriz rotação global

$$R = R * R_f$$

e a matriz translação global

$$t = t + R * (t_f * r).$$

Originando a matriz concatenação  $C$ ,

$$C = [R|t] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Em termos de tópicos desenvolvidos no nó **visodoAgro**, foram criadas as seguintes mensagens customizadas : "*msgInfo*", "*msgTime*" e "*visodom*", como representado na figura 3.6

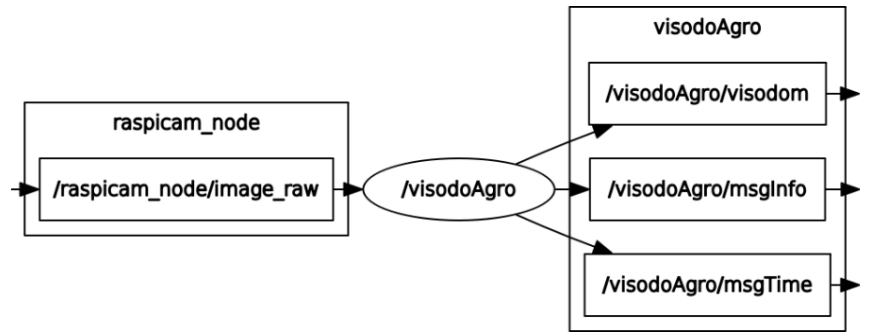


Figura 3.6: Representação dos tópicos elaborados no nó visodoAgro.

Desta forma, o formato da mensagem "**msgInfo**" é representado pela figura 3.7

```

msgInfo.msg - Mousepad
Header header

sensor_msgs/Image prevImage
sensor_msgs/Image currImage

P_match[] p_match

int32[] inliers

Matrix4x4 C
Matrix3x3 Rf
Vector3 tf|

```

Figura 3.7: Conteúdo da mensagem *msgInfo.msg*.

Esta mensagem é utilizada para publicar todos os dados importantes do algoritmo, para posteriormente ser realizada uma análise em Matlab e uma representação dos resultados obtidos nesta dissertação. Desta forma, nos campos **prevImage** e **currImage** são publicadas as imagens utilizadas para o cálculo das matrizes rotação e transformação nesse instante. Estas matrizes são publicadas no campos **Rf** e **tf**, respetivamente. O Campo **C** é usado para publicar a matriz concatenação, enquanto que o campo **p\_match** é utilizado para publicar as correspondências dos pontos característicos.

Além disso, existem ainda os *inliers*, pontos utilizados para o cálculo das matrizes resultantes, removendo as correspondências que provocavam erros nos resultados.

Por outro lado, a mensagem "**msgTime**" é utilizada para comparar os tempos de execução, o número imagens perdidas entre o processamento e quantidade de imagens processadas por diferentes tipos de algoritmos. Esta, é representada pelos seguintes campos da figura 3.8

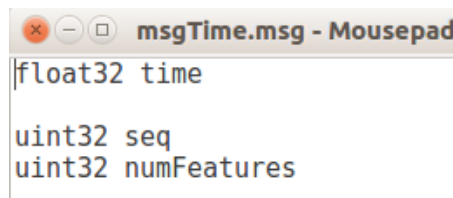


Figura 3.8: Conteúdo da mensagem *msgTime.msg*.

Por fim, a mensagem "**visodom**" é utilizada para publicar a localização em relação ao primeiro frame. Esta, é constituída pelos seguintes parâmetros da figura 3.9

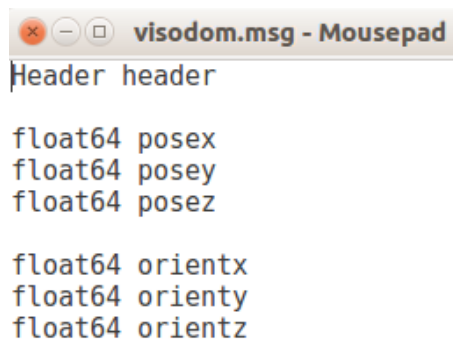


Figura 3.9: Conteúdo da mensagem *visodom.msg*.

As coordenadas x, y e z são publicadas nos parâmetros **posex**, **posey**, **posez**, respetivamente e a orientação x,y e z nos parâmetros **orientx**, **orienty**, **orientz**, respetivamente.

### 3.1.2 Desenvolvimento em Matlab

O Matlab é um *software* de alto desempenho desenvolvido para o cálculo numérico. Caracteriza-se por implementar uma linguagem própria, que resulta de uma combinação de linguagens como C, Java e Basic.

O código desenvolvido em Matlab teve como principal função o processamento dos dados guardados nos *rosbags*. Estes *rosbags* contêm as mensagens publicadas para cada teste realizado. Foram desenvolvidos *scripts* para filtrar os dados de interesse e gerar dados possíveis de analisar e inferir conclusões.

## 3.2 Hardware

Este subcapítulo visa apresentar os componentes de hardware, utilizado no desenvolvimento do projeto.

### 3.2.1 Raspberry Pi

O Raspberry Pi é um computador do tamanho de um cartão de crédito, em que todo o hardware é integrado numa única placa. O principal objetivo é promover o ensino em Ciência da Computação básica em escolas, mas devido à sua excelente qualidade / preço é bastante usado em grandes projetos de robótica, programação e até aplicações industriais.

O modelo utilizado nesta dissertação é o Raspberry Pi 3 model B. Este modelo contém um processador 1.2 GHz 64-bit quad-core ARMv8 CPU, 1 GB de RAM e Bluetooth 4.1. Além disto, este computador é compatível com ROS Kinetic e com vários módulos, como a câmara com lente olho de peixe.

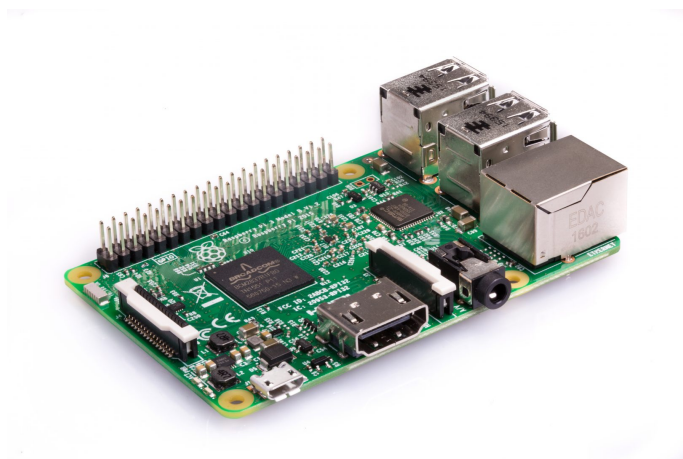


Figura 3.10: Exemplar da Raspberry Pi 3 modelo B.

### 3.2.2 Câmara com lente olho de peixe

A lente olho de peixe é uma lente grande-angular, uma vez que alcança ângulos de visão extremamente amplos. Além disso, produz uma forte distorção visual destinada a criar uma imagem panorâmica ou hemisférica ampla. Em vez de produzir imagens com linhas retas de perspectiva, as lentes olho de peixe usam um mapeamento especial (por exemplo: ângulo equisólido), que dá às imagens uma aparência convexa não retilínea.

#### 3.2.2.1 Tipos de lentes

Existem 2 tipos de lentes olho de peixe : circular e *full-frame*.

Os primeiros tipos de lentes olho de peixe desenvolvidos foram circulares, lentes que tomaram um hemisfério de  $180^\circ$  projetando um círculo no plano de imagem. Estas, têm  $180^\circ$  de ângulo de visão vertical, horizontal e diagonal.

As lentes olho de peixe *full-frame* ampliam o círculo da imagem para cobrir toda a estrutura retangular, designado por "olho de peixe de moldura completa". Desta forma, as lentes têm um ângulo de visão diagonal de  $180^\circ$  enquanto os ângulos vertical e horizontal de visão são menores.

A figura 3.11 representa as diferenças de plano de imagens obtidos com lente olho de peixe circular e olho de peixe *full-frame*. Além disso também representa a diferenças em distâncias focais das lentes, na qual afeta o tamanho das imagens na plano de imagem.

A lente é aplicada num ambiente agrícola. Ao se utilizar a lente olho de peixe circular, esta mostra-nos um maior plano de imagem, contudo contém informação desnecessária, como o céu. Assim, torna-se mais vantajoso o uso da lente olho de peixe *full-frame*, uma vez que fornece mais informação central e horizontal do campo de visão.

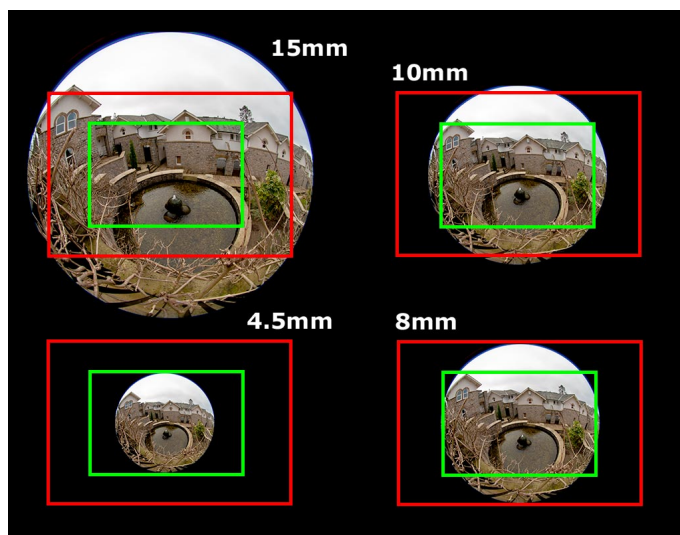


Figura 3.11: Diferença de ângulos de visão de lentes olho de peixe.

A lente especificamente escolhida é a representada na figura 3.12, com as seguintes especificações: 5 megapixel, ângulo de abertura de 160 graus (câmaras têm tipicamente 72 graus), resolução 1080p e com suporte para LED infravermelho para visão noturna.

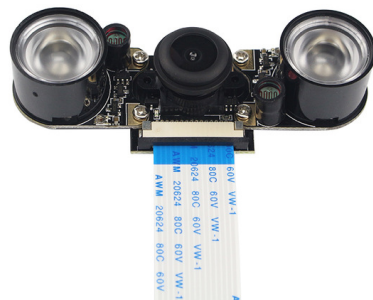


Figura 3.12: Câmara com lente olho de peixe e suporte para visão noturna.

## Capítulo 4

# Resultados Experimentais

Os testes desta secção têm como objetivo encontrar o melhor detetor / descritor / matcher , determinando a robustez do subsistema de localização para o ruído, iluminação variante e rotação.

Os parâmetros utilizados nos vários detetores para limitar o número de pontos característicos são definidos com base no valor médio de intensidade da imagem. Desta forma, os detetores trabalham sob várias condições de brilho.

Para obter resultados constantes, a plataforma ROS permite ao usuário reproduzir uma sequência de mensagens nas mesmas condições em que foram adquiridas. Esta funcionalidade permite ao desenvolvedor resultados constantes, ou seja, diferentes combinações de parâmetros podem ser testadas para a mesma entrada exata.

### 4.1 Calibração da câmara

Como mencionado anteriormente, no capítulo [2.2](#), as câmara e lente têm que ser calibrada para não causar erros de distorção nos resultados finais.

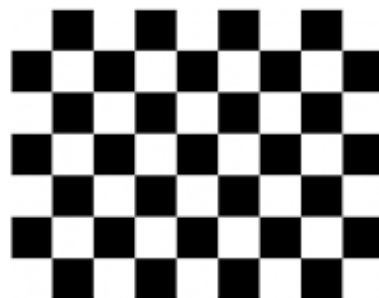


Figura 4.1: Quadrado de xadrez para calibrar câmara.

Desta forma, é usado o algoritmo desenvolvido em [?]. O algoritmo baseia-se no uso de um quadrado de xadrez, figura 4.1. Este é composto por quadrados com a mesma largura e devido à sua composição em xadrez é fácil a determinação de pontos de intreseção cruzamento. Com estes pontos é possível criar linhas com tamanhos conhecidos e através destas obter os parâmetros intrínsecos da câmara e lente.

A figura 4.2 ilustra a calibração da devida câmara do raspberry pi com uma lente olho de peixe *full-frame*. O resultado final é a matriz de parâmetros intrínsecos composta por :

$$\begin{bmatrix} 306.442 & 0 & 330.165 \\ 0 & 306.27 & 234.44 \\ 0 & 0 & 1 \end{bmatrix}$$

e matriz de parâmetros de distorção

$$\begin{bmatrix} -0.27 & 0.0445 & 0.000077 & 0.000007 & 0 \end{bmatrix}$$

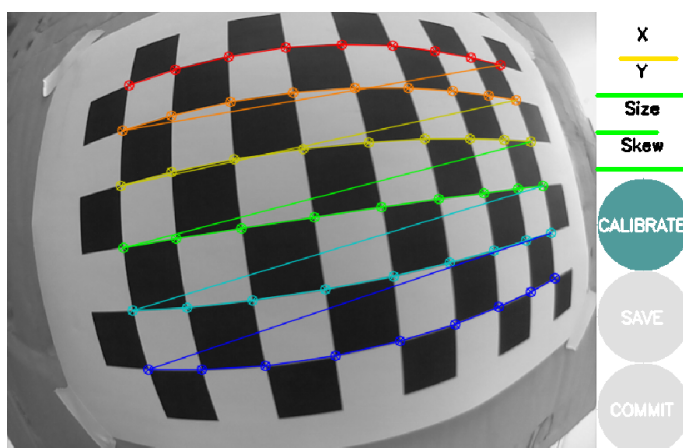


Figura 4.2: Imagem da calibração dos parâmetros da câmara.

## 4.2 Diferença entre detetores de características

Nesta dissertação serão comparados 4 detetores de características, explicitos em 2.4. Os testes são realizados num ambiente agrícola para maior coerência com o objectivo da dissertação, como ilustrado na figura 4.3.

De notar que os pontos azuis são pontos de características que têm correspondências corretas no próximo frame e os pontos vermelhos são *outliers*, pontos com correspondência de características erradas e/ou sem correspondências.

Os métodos têm as suas diferenças, desde logo a quantidade dos pontos com correspondência no frame seguinte. Desta forma, o método HARRIS é péssimo, devido às poucas correspondências, e o método SIFT descarta muitos pontos característicos de curta distância. A distribuição dos pontos



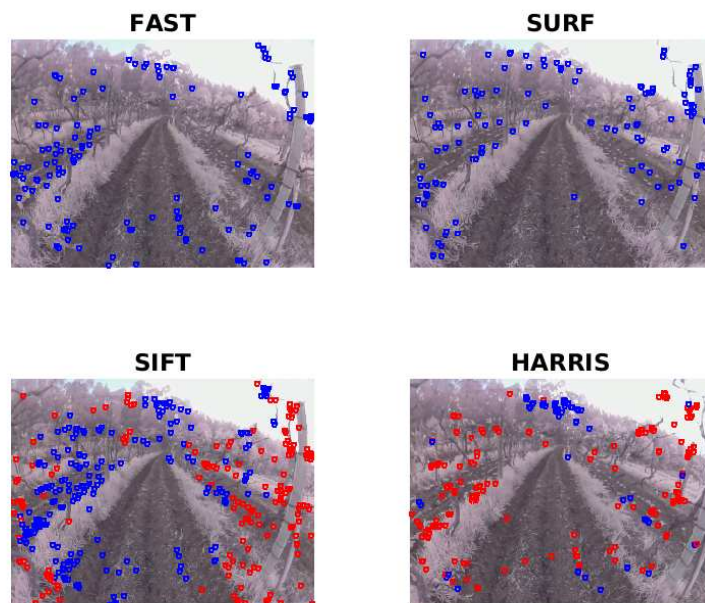


Figura 4.3: Distribuição dos pontos característicos para os quatro detetores de características, FAST , SURF, SIFT, HARRIS.

característicos é favorável no método SURF , FAST e SIFT, sendo o primeiro o melhor, com maior seleção dos pontos na videiras, sendo estes mais estáveis e recomendados devido ao ambiente vinícola. Em termos de quantidade de pontos o método SIFT têm demasiados o que implica em maior tempo de processamento e maior número de frames perdidos entre processamento. Esta quantidade podia ser diminuída mas em cenários mais precários o algoritmo fica sujeito a encontrar um número reduzido de pontos característicos que impossibilitem o funcionamento do algoritmo.

Assim, o método HARRIS é o pior devido a fraca qualidade e distribuição dos pontos característicos. O método SIFT tem um grande tempo de processamento que caso a velocidade do robô seja baixa é uma solução plausível devido a boa quantidade de pontos característicos e boa distribuição. Por fim, o método FAST e SURF são os mais indicativos. Sendo, o último com melhor distribuição dos pontos característicos.

### 4.3 Comparação das combinações

Como explicito no capítulo 2, existem vários detetores de características, descritores de características e métodos de associação de características. Desta forma, nesta dissertação são usados 4 detetores de características, FAST, HARRIS, SIFT e SURF, 3 descritores de característica, ORB, SIFT e SURF e 2 métodos de associação de características, FLANN e Brute-Force, totalizando 24 combinações.

Os métodos são comparados pelos critérios :

- **Imagens** - numero de imagens *frames* processadas.
- **Tempo** - minimo, máximo e média de tempo nos ciclos de processamento.
- **Imagens perdidas (Lost Frames)** - quantidade de imagens perdidas(não analisadas) entre ciclos de processamento.
- **Número de pontos característicos** - quantidade minima, máxima e média de pontos característicos nos ciclos de processamento.

De forma a obter resultados coerentes é utilizado sempre a mesma movimentação, ficheiro *bag*, em todos os teste.

MAIN												
FEUP_jardim_frente_2.bag												
			IMAGENS	TEMPO			Imagens Perdidas			Número Características		
				MIN	MAX	MEDIA	MIN	MAX	MEDIA	MIN	MAX	MEDIA
FAST	ORB	FLANN	33	0,32161	0,50455	0,38236	2	5	5	238	789	441
		Brute-Force	-	-	-	-	-	-	-	-	-	-
	SIFT	FLANN	27	0,38742	0,82033	0,45041	2	8	5	246	949	496
		Brute-Force	-	-	-	-	-	-	-	-	-	-
	SURF	FLANN	32	0,32970	0,63913	0,39328	2	7	5	257	999	483
		Brute-Force	-	-	-	-	-	-	-	-	-	-
HARRIS	ORB	FLANN	24	0,29596	1,62281	0,32311	2	19	5	162	406	269
		Brute-Force	6	1,46739	1,52516	1,50374	21	26	24	259	280	272
	SIFT	FLANN	22	0,26623	1,70619	0,31482	2	19	4	96	493	258
		Brute-Force	8	1,61741	1,65778	1,64259	7	24	19	300	300	300
	SURF	FLANN	17	0,30195	1,66072	0,32749	2	26	6	162	405	292
		Brute-Force	8	1,51844	1,60151	1,56593	16	21	19	300	300	300
SIFT	ORB	FLANN	-	-	-	-	-	-	-	-	-	-
		Brute-Force	-	-	-	-	-	-	-	-	-	-
	SIFT	FLANN	22	0,31915	1,79668	0,37545	2	22	5	245	582	419
		Brute-Force	7	1,68539	1,76291	1,74433	9	24	22	500	501	500
	SURF	FLANN	27	0,30361	1,42223	0,35846	2	14	5	202	710	382
		Brute-Force	10	1,22044	1,27926	1,26471	5	17	14	500	501	500
SURF	ORB	FLANN	32	0,30555	0,76352	0,34905	2	7	5	233	550	356
		Brute-Force	16	0,57653	0,97567	0,95483	7	13	9	245	478	426
	SIFT	FLANN	26	0,31670	1,61984	0,35134	2	14	5	243	664	358
		Brute-Force	9	1,08626	1,99224	1,65027	9	22	16	321	500	436
	SURF	FLANN	34	0,31244	1,05094	0,33764	2	9	5	244	659	311
		Brute-Force	14	0,70223	1,23049	0,94099	5	16	10	271	500	433

Page 1

Figura 4.4: Tabela de comparação das combinações dos métodos.

A figura 4.4 ilustra os resultados obtidos nos testes. As combinações em que o algoritmo resulta em erro por baixo número de pontos característicos e / ou baixa quantidade de associação de pontos característicos durante uma grande quantidade de ciclos de processamento é representada pelo simbolo -. De forma a melhor interpretação da tabela os valores com bons resultados estão coloridos de verde.

Assim, na coluna **Imagens** a maior quantidade analisadas no algoritmo melhor. Nas colunas **minimos, máximo e médios do Tempo** , **Imagens Perdidas** e **numero de pontos característicos** quanto menor o valor melhor os resultados.

Analisando a tabela, as combinações com o símbolo - ou sem células coloridas a verde são péssimas soluções. Desta forma, a associação **Brute-Force** não têm nenhuma solução devido ao maior tempo de processamento em relação ao método **FLANN** e a menor coerência de correspondência de pontos característicos. Assim, das restantes combinações, os detetores de **HARRIS** e **SIFT** têm algumas soluções possíveis de utilizar mas com parâmetros fracos, tais como a quantidade de Imagens, tempo máximo e máxima quantidade de imagens perdidas, tendo estes parâmetros absurdamente maiores. Dos restantes métodos, sendo todos combinações razoáveis de implementar, é de realçar as combinações **FAST/ORB/FLANN** e **SURF/ORB/FLANN** com os melhores parâmetros, sendo as utilizadas nos restantes testes.

## 4.4 Testes

De forma a validar o algoritmo de localização os testes requerem a quantificação do erro. O erro é descrito com a diferença entre o valor real e o valor medido.

As coordenadas reais têm de ser obtidas através de um sensor externo, tal como a laser baseado em localização, odometria das rodas or mesmo uma fita métrica. *Devido ao declive dos terrenos e estrutura é utilizada uma fita métrica para comparação dos resultados.*

Os devidos testes foram realizados em ambiente agrícola, precisamente numa vinha. Os percursos efetuados foram : estático, movimento em linha reta, movimento em L (semi-quadrado), percurso quadrangular, percurso retangular e percurso circular. De notar que os percursos foram realizados com velocidades baixas. Assim, a câmara consegue extrair um maior número de características e por consequência o erro da trajetória estimada será menor.

### 4.4.1 Teste de parâmetros

Quatro parâmetros foram escolhidos para comparar a combinação dos detetores / descritores e matchers. Eles são :

- **Imagens processadas** - Corresponde ao número de imagens que o programa conseguiu processar. É proporcional à velocidade do ciclo de processamento.
- **Média de frames perdidos** - Quantidade de frames perdidos entre ciclos.
- **Média matches por imagem** - Corresponde à média de correspondências encontradas para cada imagem depois de filtrar outliers.
- **Média de processamento** - Corresponde à média da duração de processamento de ciclo.

### 4.4.2 Sequência sem Movimento

Devido à fita métrica não produzir medidas em tempo real do movimento, o primeiro teste é feito com uma sequência de frames, or bags, onde o robô não se movimenta.

#### **4.4.3 Detecção da trajetória**

*Devido ao erro deste teste não poder ser quantificado, este serve apenas como demonstração ou provas de conceito. A validação do resultado é visual.*

## **Capítulo 5**

# **Conclusões e Trabalho Futuro**

### **5.1 Conclusões**

### **5.2 Trabalho Futuro**



# Bibliografia

- [1] J. Mendes, F. N. dos Santos, N. Ferraz, P. Couto, and R. Morais, “Vine trunk detector for a reliable robot localization system,” in *2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, IEEE, may 2016.
- [2] “Agrob v16 – robotics for agriculture and forestry – that works in steep slope viticulture.” <http://agrob.inesctec.pt/>. [Online; accessed 05-February-2018].
- [3] M. O. A. Aqel, M. H. Marhaban, M. I. Saripan, and N. B. Ismail, “Review of visual odometry: types, approaches, challenges, and applications,” *SpringerPlus*, vol. 5, oct 2016.
- [4] S. K. Ericson and B. S. Åstrand, “Analysis of two visual odometry systems for use in an agricultural field environment,” *Biosystems Engineering*, vol. 166, pp. 116–125, feb 2018.
- [5] D. Nister, O. Naroditsky, and J. Bergen, “Visual odometry,” in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, IEEE.
- [6] Y. Cheng, M. Maimone, and L. Matthies, “Visual odometry on the mars exploration rovers,” in *2005 IEEE International Conference on Systems, Man and Cybernetics*, IEEE.
- [7] E. Ericson and B. Astrand, “Visual odometry system for agricultural field robots,” in *Anonymous Proceedings of the World Congress on engineering and computer science*, 2008.
- [8] D. Scaramuzza and F. Fraundorfer., *Visual odometry part I: The first 30 years and fundamentals. IEEE Robotics & Automation Magazine*, pages 80–92. 2011.
- [9] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, “An overview to visual odometry and visual SLAM: Applications to mobile robotics,” *Intelligent Industrial Systems*, vol. 1, pp. 289–311, nov 2015.
- [10] P. Hansen, P. Corke, and W. Boles, “Wide-angle visual feature matching for outdoor localization,” *The International Journal of Robotics Research*, vol. 29, pp. 267–297, dec 2009.
- [11] P. Srestasathiern and N. Soontranon, “A novel camera calibration method for fish-eye lenses using line features,” *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XL-3, pp. 327–332, aug 2014.

- [12] J. Kannala and S. Brandt, “A generic camera calibration method for fish-eye lenses,” in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, IEEE, 2004.
- [13] X. Ying, Z. Hu, and H. Zha, “Fisheye lenses calibration using straight-line spherical perspective projection constraint,” in *Computer Vision – ACCV 2006*, pp. 61–70, Springer Berlin Heidelberg, 2006.
- [14] F. Fraundorfer and D. Scaramuzza, “Visual odometry : Part II: Matching, robustness, optimization, and applications,” *IEEE Robotics & Automation Magazine*, vol. 19, pp. 78–90, jun 2012.
- [15] H.Berg and R.Haddad., *Visual odometry for road vehicles using a monocular camera. Master thesis, Department of Signals and Systems, Chalmers University of Technology, Gothenburg, Sweden, 2016. [Online].*