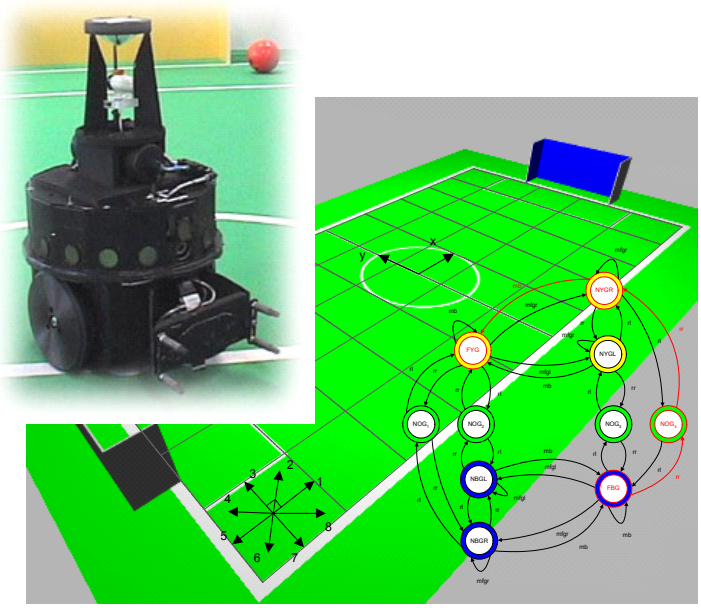
UNIVERSIDADE TÉCNICA DE LISBOA

INSTITUTO SUPERIOR TÉCNICO



NAVEGAÇÃO TOPOLÓGICA PARA FUTEBOL ROBÓTICO

Gonçalo Neto, nº 45340, Ramo de Controlo e Robótica
Hugo Costelha, nº 44747, Ramo de Controlo e Robótica

LICENCIATURA EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES
Relatório de Trabalho Final de Curso
627/2001/L

Prof. Orientador: Pedro Almeida Lima
Prof. Acompanhante: José Santos-Victor

Janeiro de 2003

Trabalho orientado por:

Pedro Manuel Urbano de Almeida Lima
Docente Responsável
Secção de Sistemas e Controlo
Departamento de Engenharia Electrotécnica e de Computadores
Instituto Superior Técnico

José Alberto Rosado dos Santos-Victor
Docente Acompanhante
Secção de Sistemas e Controlo
Departamento de Engenharia Electrotécnica e de Computadores
Instituto Superior Técnico

Agradecimentos

Hugo: Em primeiro lugar gostaria de agradecer à Fernanda, minha namorada sem a qual eu não teria conseguido ultrapassar estes últimos anos (não desesperes, já estou quase a acabar...) ♡

Ao meu Pai, à minha Mãe e ao meu *Brother* que desde sempre me apoiaram no meu percurso estudantil, um especial obrigado.

À malta da *SCDEEC*, em especial ao JO, ao RMS e ao DM, que foram companhia em muitos trabalhos, e também à *SCDEEC*, local sagrado para a execução de todos os meus projectos.

Gonçalo: Depois de todas as horas em frente ao computador, estes minutos parecem os mais difíceis: apenas porque é complicado escrever agradecimentos sem parecer a entrega dos Óscares, onde tudo soa a frase feita.

Bem... suponho que as pessoas a quem vou agradecer sabem que é sincero!

Assim, começo por agradecer ao Hugo, por não desesperar nos meus atrasos e despertadores que não tocaram. Olhando para todos os trabalhos, acho que no fim de contas ainda nos safámos bem!

Aos meus amigos... Ao Miguel e ao Pedro que já me aturam à mais tempo do que eu próprio me lembro! Miguel, aquelas Barcas Velhas tão quase a ser despachadas!! Ao pessoal da ursalhada – Ricardo, Bruno, Melro, Pigas e Cajó – porque eu sei que daqui a 40 anos ainda vamos estar a fazer jantaradas e a contar as mesmas histórias de acampamentos.

Aos meus pais, por me estarem sempre a chatear a cabeça e, por vezes, me acordarem para o mundo, e à minha irmã por não me chatear a cabeça quando precisei do carro! Um agradecimento especial ao meu pai pela revisão do trabalho.

À minha *gorda* Filipa (qual será a razão de estares aqui no fim?), pela paciência quando passei mais tempo com o computador do que com ela... A nossa ida ao *campsite* do Rover no Golfzito tem que ser para breve para compensar essas horas sem ti!!!!

Ambos: Em primeiro lugar ao Prof. Pedro Lima um especial agradecimento pela dedicação ao longo deste último ano e meio. Aos restantes membros do projecto *ISocRob*, ao Prof. Custódio, ao Miguel Garção, ao Bruno Damas, ao Hugo Furtado, ao Pedro Pinheiro e ao Cláudio Gil pela partilha de experiências e conhecimento. Aos novatos do grupo pelo empenho demonstrado – não desesperem!!

Um especial abraço ao pessoal de Controlo e Robótico, com o qual passámos muitas horas, bons e maus momentos neste curso.

Resumo

O objectivo deste Trabalho Final de Curso enquadra-se no desenvolvimento de um sistema de navegação topológica, baseado na descrição de locais-chave na forma de parâmetros representativos de imagens desses locais, e a aplicação do sistema desenvolvido ao futebol robótico, através da implementação de algoritmos no âmbito do projecto SocRob (*Soccer Robots* ou *Sociedade de Robots*).

Para atingir esse objectivo, associou-se um mapa topológico a um grafo, em que cada nó correspondia a um local-chave. Desta forma, a navegação resume-se à procura de um caminho no grafo. A associação de imagens pré-capturadas aos locais-chave foi feita usando Análise de Componentes Principais.

O método apresentou um desempenho promissor na navegação entre locais-chave e mostrou ser adaptável a diferentes grafos. São apresentados resultados de simulação.

Palavras-Chave

Navegação Topológica, Mapa Topológico, Análise de Componentes Principais, Futebol Robótico.

Abstract

The goal of this Final Year Project is to develop a topological navigation system, based on the description of key-places by parameters that represent the images associated with those spots, and the application of the developed system to robotic soccer, through the implementation of algorithms in the scope of the SocRob project (*Soccer Robots* or *Society of Robots*).

To achieve this goal, a topological map was associated with a graph, were each node corresponded to a key-place. This way, navigation is reduced to a graph path search. Principal Components Analysis was used to link the pre-acquired images to the key-places.

The method revealed a promising performance navigating between key-places and was able to adapt itself to different graphs. Simulation results are presented.

Keywords

Topological Navigation, Topological Map, Principal Component Analysis, Robotic Soccer.

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O problema da navegação de robots móveis consiste em conduzir um robot num dado meio envolvente, mais ou menos bem estruturado, por forma a evitar obstáculos, até uma zona objectivo, utilizando retroacção sensorial na forma de informação do meio.

Na sua versão mais "clássica", o problema pode ser resolvido através da determinação periódica da posição do robot num dado sistema de coordenadas, a partir da sua odometria e/ou das distâncias ou ângulos relativamente a balizas de referência, ou ainda analisando a correlação entre um mapa do mundo pré-existente e as observações dos sensores do robot em cada instante. No entanto, para ambientes menos estruturados e/ou quando se pretende uma descrição mais qualitativa da localização e da zona para onde se deve dirigir o robot, são desejáveis alternativas que mais se assemelhem à forma como os seres humanos "navegam". Uma dessas alternativas é a **Navegação Topológica** [3]. Esta consiste em associar imagens a locais-chave que serão depois usadas para navegar num mapa topológico, i.e., um grafo que interliga os locais-chave por caminhos possíveis.

Citando o exemplo dado em [3], e de acordo com a Figura 1.1, para ir do Rossio para a Praça Duque de Saldanha, não é necessário pensar em termos métricos. Para tal, basta seguir em frente até chegar à Rotunda, virar à direita na direcção de Picoas e prosseguir até referida Praça. Os locais turísticos representam neste problema os locais-chave.

Transpondo o exemplo dado para o futebol Robótico, ao invés de se enviar o robot para a posição $x = 2$, $y = 1$ e $\theta = -45°$, a tarefa passa a ser ir para o lado direito da baliza azul e virar para o centro da mesma. Assim, usando navegação topológica, o robot passa a executar as tarefas dum modo qualitativo.

Neste trabalho pretendeu-se iniciar o desenvolvimento de um método de navegação para aplicação a um robot futebolista, mais concretamente aos robots da equipa *ISocRob*. Para alcançar tal objectivo, foi necessário resolver de uma forma integrada diversos problemas:

Construção de um Mapa Topológico: nesta fase do desenvolvimento, realizada *off-line*, procurou-se construir um mapa topológico que servisse de base à navegação. O mapa foi identificado com um grafo de auxílio à navegação, cuja construção está descrita no Capítulo 3. Para a representação dos locais do mapa em causa, compararam-se duas técnicas que têm por base a Análise de Componentes Principais (ACP), como apresentada em [7, 8]. No Capítulo 2 apresentam-se as referidas técnicas de ACP.

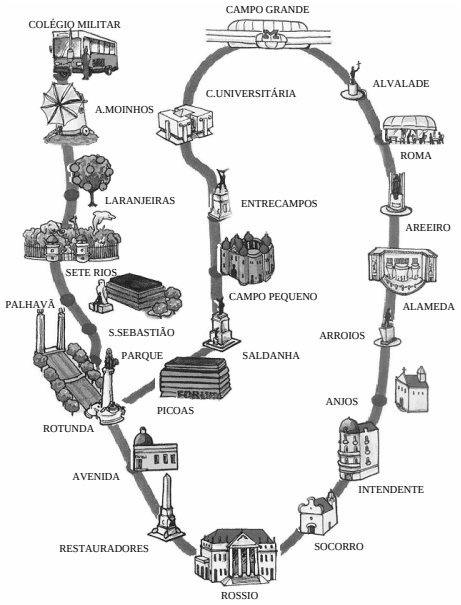


Figura 1.1 - Mapa topológico de locais turísticos de Lisboa

Localização no Mapa: esta fase integra-se no processo de navegação e decorre em tempo de execução. Fundamentalmente, resume-se a um problema de classificação de um padrão de entrada em grupos gerados por padrões de treino, associados aos locais do mapa topológico. As técnicas usadas foram o método dos *k-vizinhos mais próximos* [6] e o método *MKL* (do acrónimo inglês *Multispace Karhunen-Loeve* [2]). A forma de usar esses métodos para a localização num mapa topológico é descrita no Capítulo 3.

Navegação: esta fase consiste, por sua vez, essencialmente, na procura de um caminho no grafo associado ao mapa topológico. Foram utilizadas técnicas de procura em árvore, descritas em [9]. No Capítulo 4 são apresentadas as referidas técnicas e, além disso, é descrita a construção de um supervisor simples para a navegação, o qual permita detectar falhas na execução dos caminhos gerados.

O encadeamento dos passos apresentados, em termos de desenvolvimento do método, está resumido no fluxograma apresentado no Anexo D.

No Capítulo 5 é descrita a forma como os métodos apresentados se podem adequar aos robots futebolistas ou, neste caso, ao simulador desenvolvido. Esta implementação seguiu a linha de orientação definida pelo fluxograma já referido.

No Capítulo 6 são apresentados os resultados respeitantes aos testes conduzidos no Capítulo 5 e são apresentados alguns comentários acerca dos mesmos. O Capítulo 7 apresenta as conclusões principais deste trabalho.

Capítulo 2

Representação dos Dados

A representação dos locais é uma característica fundamental da Navegação Topológica. Neste trabalho, usa-se como base a análise de imagens. Contudo, estas imagens podem ter dimensões elevadas, especialmente se forem imagens com cor, levando a um peso computacional extremamente grande. Para superar este problema são usados algoritmos de compressão e/ou representação de dados.

Vários investigadores têm usado técnicas diferentes para a representação/compressão de dados em navegação por recurso à visão. Em [5] é efectuada a localização analisando as imagens no domínio da frequência. Em [4] são usadas imagens originais, enquanto que em [1] se recorre à análise dos histogramas das imagens. Outros autores recorrem a técnicas de Análise de Componentes Principais (ACP) [3].

Optou-se por uma esta última abordagem para fazer a compressão das imagens, já que o seu uso não só possibilita a compressão dos dados como faz uma selecção das principais características de um conjunto de imagens (seleccionando as que melhor distinguem cada uma delas). Neste capítulo descreve-se a aplicação de algoritmos de ACP ao problema em questão.

2.1 Construção de um Espaço Principal

Foi utilizado um método de ACP que é conhecido como expansão de *Karhunen-Loeve* (*KL*), o qual se encontra descrito em [7]. Neste, começa-se por capturar um conjunto de M imagens e colocá-las na forma vectorial $x^{(i)}$, com $i = 1, \ldots, M$. De notar que cada um dos vectores é N-dimensional, sendo N, no caso das imagens, dado pela multiplicação do número de *pixels* pelo número de canais (3 canais, no caso RGB)[1].

O conjunto de vectores pode ser representado na forma matricial X, de dimensão $[N \times M]$, em que cada coluna corresponde a um dos vectores $x^{(i)}$.

Da mesma forma, pode definir-se a matriz $\tilde{X}$, em que cada uma das colunas é dada por $\tilde{x}^{(i)} = (x^{(i)} - \bar{x})$. O vector $\bar{x}$ representa a média sobre o conjunto $\left\{x^{(i)}\right\}_{i=1,\ldots,M}$

A matriz de covariância do conjunto de vectores é dada, então, por $R = \frac{1}{M}\tilde{X}\tilde{X}^T$. Os seus vectores próprios formam uma base ortogonal para o conjunto de todos os vec-

[1] Na realidade, não é necessário aplicar o método a imagens, apesar de essa ser a aplicação mais difundida. Qualquer conjunto de vectores com a mesma dimensão é passível de ser analisado. Assim outros tipos de sensores poderão ser utilizados, eventualmente em conjugação uns com os outros.

tores $\tilde{x}^{(i)}$. No âmbito da ACP, também é usual referir-se aos vectores desta base como *componentes principais* ou, no caso das imagens, *imagens principais*.

No entanto, para a matriz R, cuja dimensão é $[N \times N]$, o cálculo de uma base de vectores próprios pode tornar-se computacionalmente impraticável[2]. Para contornar este aspecto de computação, utiliza-se o facto dos valores próprios significativos (diferentes de 0) de R serem iguais aos valores próprios de $A = \frac{1}{M}\tilde{X}^T\tilde{X}$ sendo os vectores próprios de R, $\left\{\phi^{(i)}\right\}_{i=1,\ldots,M}$, obtidos a partir dos de A, $\left\{a^{(i)}\right\}_{i=1,\ldots,M}$ através de [7]:

$$\phi^{(i)} = \lambda^{(i)^{-\frac{1}{2}}}\tilde{X}a^{(i)} \qquad i = 1, \ldots, M \tag{2.1}$$

com

$$\phi^{(i)^T}\phi^{(i)} = 1, a^{(i)^T}a^{(i)} = 1 \qquad i = 1, \ldots, M$$

Se o número de imagens for muito menor que a dimensão delas então, seguramente ($M \ll N$), este procedimento torna-se bastante vantajoso. Contudo, continua a implicar o armazenamento da matriz $\tilde{X}$.

Existem, todavia, alguns procedimentos que permitem evitar esse outro aspecto. Estes tomam como base o facto de os vectores usados para gerar o espaço, $\left\{x^{(i)}\right\}_{i=1,\ldots,M}$, poderem ser aproximados por uma base constituída pelos η vectores próprios de maior importância, ou seja, correspondentes aos η maiores valores próprios. Pode efectuar-se esta aproximação porque cada um dos valores próprios representa a variância dos dados segundo a direcção definida pelo respectivo vector próprio. Assim, é importante que sejam as direcções de maior variância a serem representadas, dado que contêm maior informação sobre os dados que aproximam.

A Figura 2.1 apresenta um exemplo simples (para dados bidimensionais) no qual a direcção definida por ϕ_1 é claramente mais importante na representação dos dados do que aquela definida por ϕ_2. De facto, a dispersão dos dados segundo ϕ_1 (quantificada pelo respectivo valor próprio) será muito maior que segundo ϕ_2, o que se pode constatar visualmente.

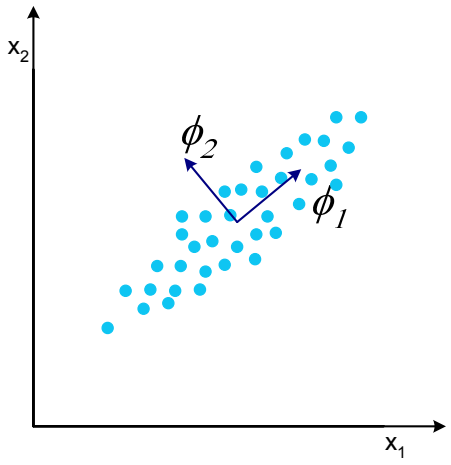


Figura 2.1 - Exemplo de *KL* para dados bidimensionais

Optou-se, neste trabalho, pelo procedimento iterativo apresentado em [7] que, após a sua conclusão, devolve uma aproximação aos η vectores próprios mais significativos da

[2] A título de exemplo, pode referir-se que para uma imagem RGB, de dimensão 320×240, se tem $N = 230400$.

matriz R. O número de imagens principais obtidas torna-se, então, um dos parâmetros a ter em conta, equilibrando os recursos computacionais e a exactidão dos resultados.

De seguida, apresenta-se um resumo desse procedimento.

Passo de Inicialização: A partir do conjunto $\left\{x^{(i)}\right\}_{i=1,\ldots,M}$ escolhem-se $k = \eta + 1$ imagens (onde η é o número de imagens principais que se pretende extrair) a partir das quais se constrói uma matriz A_k:

$$A_k = \frac{1}{k}\tilde{X}_k^T\tilde{X}_k \tag{2.2}$$

À semelhança do procedimento tomado na Equação (2.1), obtém-se o primeiro conjunto de imagens próprias (não normalizadas), $\left\{\psi_k^{(i)}\right\}_{i=1,\ldots,\eta}$, através da seguinte equação:

$$\psi_k^{(i)} = \tilde{X}_k a_k^{(i)} \qquad i = 1, \ldots, \eta \tag{2.3}$$

onde $\left\{a_k^{(i)}\right\}_{i=1,\ldots,\eta}$ representam os vectores próprios normalizados de A_k associados aos η maiores valores próprios.

De notar que a matriz A_k produz até $\eta + 1$ vectores próprios linearmente independentes. Ao tomar apenas os η mais importantes (correspondentes aos maiores valores próprios), considera-se que as imagens usadas para o cálculo de A_k, e implicitamente de R_k, podem ser aproximadas por uma combinação linear de η vectores próprios.

Passo de Iteração: Incrementa-se k e retira-se uma nova imagem, $x^{(k)}$, do conjunto inicial de M imagens, a partir da qual se calcula $\tilde{x}^{(k)} = x^{(k)} - \bar{x}$. Constrói-se, então, uma nova matriz $\tilde{X}_k$ em que $\tilde{x}^{(k)}$ e $\left\{\psi_{k-1}^{(i)}\right\}_{i=1,\ldots,\eta}$ correspondem à colunas desta matriz. Pode, então, calcular-se A_k usando (2.2) e, desta matriz, obter uma nova base de vectores próprios (não normalizados) $\left\{\psi_k^{(i)}\right\}_{i=1,\ldots,\eta}$, da mesma forma que na Equação (2.3). Note-se que, mais uma vez, apenas se retiveram η dos $\eta+1$ vectores próprios.

Passo de Normalização: Repete-se o passo anterior até que $k = M$. Nesta altura, usa-se a Equação (2.4) para normalizar as imagens próprias e obter a base de η vectores próprios para o sub-espaço principal.

$$\phi_M^{(i)} = (M\lambda_M^{(i)})^{-\frac{1}{2}}\psi_M^{(i)} \qquad i = 1, \ldots, \eta \tag{2.4}$$

2.2 Operações e Distâncias no Espaço Principal

Depois de construído o sub-espaço principal, é de extrema utilidade definir algumas operações, tais como a projecção e reconstrução de vectores, e medidas de distância.

Tomando o conjunto $\left\{\phi^{(i)}\right\}_{i=\ldots,\eta}$ como uma base para o espaço em questão, representado por S, a componente i da projecção de um vector N-dimensional x sobre S pode ser calculada por:

$$y_i = (x - \bar{x}_T)^T \phi^{(i)} \qquad i = 1, \ldots, \eta \tag{2.5}$$

onde $\bar{x}_T$ representa a média de todos os vectores de treino $\left\{x_T^{(i)}\right\}_{i=1,\ldots,M}$, ou seja, todos os vectores usados para criar S.

Por outro lado, é importante dispor de uma forma obter a reconstrução de um vector η-dimensional de S para o espaço das imagens:

$$x = \left(\sum_{i=1}^{\eta} y_i \phi^{(i)}\right) + \bar{x}_T \tag{2.6}$$

Quanto a distâncias no espaço S, usaram-se duas métricas distintas, expressas nas Equações (2.7) e (2.8), onde Λ representa uma matriz diagonal com todos os valores próprios.

$$d_E(y^{(1)}, y^{(2)}) = (y^{(1)} - y^{(2)})^T (y^{(1)} - y^{(2)}) \tag{2.7}$$

e

$$d_C(y^{(1)} - y^{(2)}) = (y^{(1)} - y^{(2)})^T \Lambda (y^{(1)} - y^{(2)}) \tag{2.8}$$

A primeira distância apresentada é, apenas, o quadrado da distância euclidiana entre os dois vectores $y^{(1)}$ e $y^{(2)}$. A segunda distância, usada para navegação topológica em [3], pretende pesar as componentes das projecções das imagens sobre o sub-espaço principal, de forma a que às associadas aos maiores valores próprios seja dada maior importância. Pode mostrar-se que, quando $\eta = M$, d_C corresponde à distância entre dois vectores no espaço original, pesada pela matriz de covariância, ou seja:

$$(y^{(1)} - y^{(2)})^T \Lambda (y^{(1)} - y^{(2)}) = (x^{(1)} - x^{(2)})^T R (x^{(1)} - x^{(2)}) \tag{2.9}$$

No caso de $\eta < M$, apenas se considera a aproximação à matriz de covariância definida pelos η vectores próprios e η valores próprios considerados.

Com a introdução de uma métrica é possível desenvolver uma forma de classificar um vector x, ou seja, identificar a sua posição no espaço. Associando $\left\{y_T^{(i)}\right\}_{i=1,\ldots M}$, ou seja, as projecções de todos os vectores que foram utilizados para determinar o espaço S (ver Capítulo 2.1), com posições num mapa, pode dizer-se que o vector x se encontra na posição C_j se o vector de $\left\{y_T^{(i)}\right\}_{i=1,\ldots M}$ que se encontra mais próximo da sua projecção é $y^{(j)}$[3]. Esta classificação, expressa na Equação (2.10), é dependente da distância escolhida, o que pode levar a resultados de classificação algo diferentes para as diversas métricas.

$$j = \arg\min_i d(y, y_T^{(i)}) \tag{2.10}$$

[3]Este tipo de abordagem permite guardar apenas as projecções dos vectores usados para construir o espaço (cuja dimensão é muito menor que a dos vectores em si) para caracterizar totalmente o mapa.

É interessante definir, ainda, a aproximação efectuada quando se projecta um vector x no espaço S. Nesta ordem de ideias, a Equação (2.11) permite calcular o erro quadrático entre o vector original (x) e a sua reconstrução depois de projectado ($\hat{x}$):

$$\varepsilon^2(x) = (x - \hat{x})^T(x - \hat{x}) \tag{2.11}$$

Em [2] este valor é ainda definido como a distância do vector ao espaço em causa.

O valor médio dos erros quadráticos cometidos na reconstrução de todos os vectores $x_T^{(i)}$ que geraram o espaço pode ser dado pela soma dos valores próprios que foram desprezados (Equação (2.12)). Este facto fornece um critério de escolha do número de valores próprios que vão ser usados na representação dos espaços. Assim, um procedimento comum [2] consiste em estabelecer um valor de ε a usar e derivar um valor de η que resulte num erro menor que o ε escolhido.

$$E[\varepsilon^2] = \sum_{i=\eta+1}^{M} \lambda_i \tag{2.12}$$

Para obter o erro quadrático médio em percentagem (ξ), basta dividir o valor obtido na Equação (2.12) pela soma de todos os valores próprios.

$$\xi = \frac{\sum_{i=\eta+1}^{M} \lambda_i}{\sum_{i=1}^{M} \lambda_i} \tag{2.13}$$

2.3 Agrupamento e Classificação de Dados

Como já foi apontado, uma das aplicações, no âmbito da navegação topológica, dos métodos de ACP referidos, é a associação de cada uma das imagens de entrada a uma posição de um mapa topológico. Assim, o vector obtido pela projecção de uma determinada imagem de treino será suficiente para caracterizar o local-chave em questão.

Há, contudo, outro tipo de aplicação desses mesmos métodos: cada local-chave poderá ser associado a um grupo de imagens de treino, por oposição a uma única imagem. A escolha entre cada uma das abordagens só pode ser feita tendo como base um problema específico.

Para o problema da navegação topológica num ambiente de futebol robótico, a associação de uma imagem a cada local-chave (sendo o número de locais algo elevado) poderá não ser a mais adequada. Isto porque a distinção correcta de todos os locais-chave implica um erro quadrático médio de reconstrução bastante baixo, o que se traduz, de acordo com a Equação (2.12), num valor de η relativamente elevado, face ao utilizado na maior parte das aplicações.

Tomando o caminho definido pela segunda abordagem, torna-se, então, necessário definir formas de agrupar as imagens e representar a informação nelas implícita. O problema pode ser posto na seguinte forma: tomando um conjunto de imagens de entrada I e uma partição desse conjunto $\mathcal{I} = \{I_1, I_2, \ldots, I_P\}$ com $I_i \cap I_j = \emptyset,\ _{i \neq j}$, $\bigcup_i I_i = I$ e em que P é o número de locais-chave, pretende-se encontrar uma representação que associe a cada local C_i a informação contida em I_i.

É interessante verificar que, desta forma, a localização de um vector no local-chave em questão se torna um problema de classificação.

Vários autores abordaram já este problema usando métodos de ACP. Em [8], são identificados os locais-chave com variedades no espaço das projecções (sub-espaço principal S) e utilizam-se *splines* para interpolar os pontos destas variedades; a vantagem deste método é a obtenção de uma descrição paramétrica dos espaços associados a cada local-chave.

Outra solução, a partir da qual foram ensaiadas algumas experiências neste trabalho, é a de usar a ACP sobre o conjunto de imagens I_i para associar o sub-espaço resultante (S_i) ao local C_i, à semelhança do que é descrito em [2]. A este método é dado o nome de *MKL* (do acrónimo anglo-saxónico *Multispace Karhunen-Loeve*).

A classificação de um vector x num local C_j torna-se, deste modo, uma questão de verificar qual dos espaços produz o menor erro quadrático de reconstrução para esse vector, como se pode ver pela Equação (2.14), na qual $\varepsilon(x, S_i)$ representa o erro quadrático associado ao espaço S_i. A Figura 2.2 esquematiza esta operação. De notar que o vector x, a amarelo, será classificado no espaço S_2 e o vector de treino $x_T^{(k)}$, apesar de ter sido usado para gerar S_2, foi classificado em S_1. Não há, pois, garantias de que um vector de treino seja classificado no espaço que ajudou a gerar o que, contudo, seria útil para filtrar falsos positivos.

$$j = \arg\min_i \varepsilon^2(x, S_i) \tag{2.14}$$

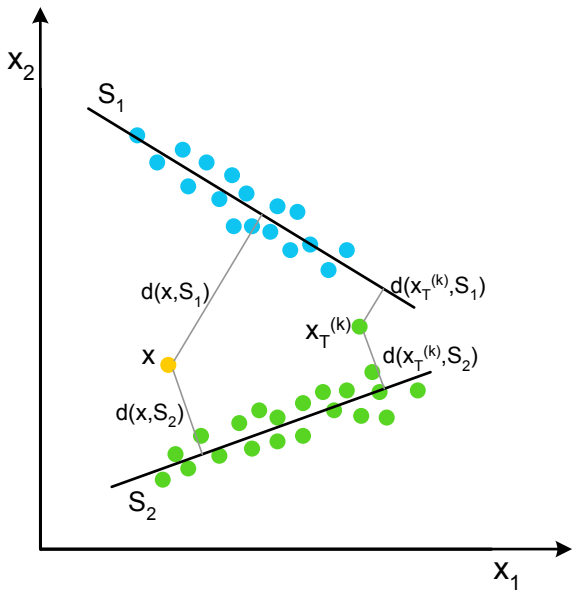


Figura 2.2 - Exemplo de *MKL* para espaços bidimensionais

Pode definir-se uma medida do erro quadrático médio de reconstrução (em percentagem), quando se projecta um vector no conjunto de espaços obtido. Tal medida é dada pela média ponderada dos respectivos erros quadráticos médios (em percentagem) associados, respectivamente, a cada espaço, da seguinte forma:

$$\xi\left(\{S_j\}_{j=1,\ldots,P}\right) = \frac{1}{M}\sum_{i=1}^{P}\left(M_i\xi(S_i)\right) \tag{2.15}$$

em que M representa o número total de imagens de treino e M_i o número de imagens usadas para gerar S_i.

Além desta forma de agrupar e classificar os resultados, foi utilizado um outro método para o agrupamento das imagens. Nele, começa-se, por utilizar o conjunto de treino $I = \left\{x_T^{(i)}\right\}_{i=1,\ldots,M}$ para criar o espaço S, conforme foi anteriormente descrito. O passo seguinte é o de projectar cada imagem x_T sobre o espaço criado e, consoante o conjunto I_j em que se encontra, definir o local-chave C_j ao qual a sua projecção y_T pertence, ou seja:

$$y_T \in C_j \Leftrightarrow x_T \in P_j$$

A classificação de um novo vector x pode, então, ser implementada através de um método de *k-vizinhos mais próximos*. Assim, começa-se por projectar x sobre S obtendo y. De seguida, calcula-se a distância deste vector a todas as projecções das imagens de treino $\left\{y_T^{(i)}\right\}_{i=1,\ldots,M}$, segundo uma determinada métrica, e retêm-se as k menores distâncias. O vector de entrada é, então, classificado na classe mais representada no conjunto das k projecções mais próximas de y.

Neste trabalho, foram feitos realizados testes, para diferentes valores de k e para as duas métricas apresentadas no Capítulo 2.2.

Capítulo 3

Mapa Topológico

Em oposição à navegação métrica, a Navegação Topológica baseia-se numa descrição mais qualitativa da posição do robot e dos vários objectivos a atingir. Deste contexto surge a base do método de navegação proposto: o *Mapa Topológico*.

Pode observar-se na Figura 1.1 um mapa topológico de locais turísticos de Lisboa. Neste mapa é possível observar vários locais-chave com as ligações entre estes que serão usados pelo turista para navegar em Lisboa e visitar os vários locais. Esta visita pode ser feita sempre sem ter de recorrer à localização exacta dos locais, sabendo apenas como os mesmos estão ligados entre si. Veja-se o exemplo mencionado na Introdução.

No contexto da Navegação Topológica há que definir o mapa topológico onde se quer navegar. No Capítulo 3.1 deste capítulo retrata-se a descrição do mapa topológico, o modo como este é guardado e caracterizado, enquanto que no Capítulo 3.2 se descreve o modo de associar cada nó à realidade física em causa, usando imagens, bem como a forma de o robot se localizar num determinado nó.

3.1 Caracterização do Mapa Topológico

Um mapa topológico é representado por um grafo constituído por nós e transições com várias características especiais associadas aos mesmos:

Nós: estão associados a locais (ou regiões) específicos no mundo. Podem definir um local onde uma determinada tarefa deve ser desenvolvida, identificando assim os pontos de importância para os robots. Através da auto-localização (ver Capítulo 3.2) o robot determina em que nó se encontra;

Transições: Estas caracterizam a ligação entre os nós e, como tal, estão associadas ao movimento do robot de um nó para o outro.

Em [3] associam-se características geométricas simples do meio envolvente às transições, i.e., ao caminho a seguir entre os nós. Na abordagem usada, o movimento que o robot deve fazer para partir dum nó e atingir outro é definido não só pela transição mas também pelo nó a atingir. Para ilustrar esta ideia, segue-se um exemplo dum mapa topológico associado ao interior duma casa.

Na Figura 3.1 está presente parte desse mapa topológico, segundo a abordagem usada. No grafo são visíveis dois nós ($SALA$ e $QUARTO$) e duas transições (ambas $r\&mf$). Assume-se que o robot é capaz de se localizar num ou noutro nó, consoante se encontra

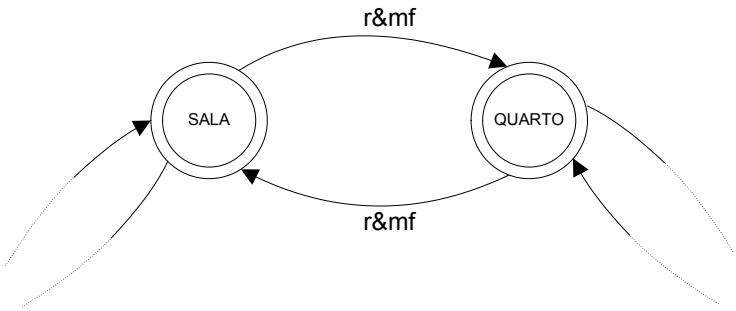


Figura 3.1 - Exemplo dum grafo associado a um mapa topológico

junto a uma ou outra porta (ver Capítulo 3.2) e as duas portas estão de frente uma para a outra. Neste caso, a transição $r\&mf$ é caracterizada por uma rotação de 180°, seguida dum movimento com uma velocidade linear (pode ser constante) e uma velocidade angular que depende da orientação do robot em relação à porta a atingir. O robot, quando parte da *SALA* para o *QUARTO*, começa por "inverter"a sua orientação para, de seguida, navegar em direcção à outra porta, fazendo seguimento visual de caminho. Durante o percurso dum nó para o outro o robot, vai efectuando continuamente a auto-localização no mapa topológico, parando o seu movimento quando se localiza no nó objectivo. O mesmo seria feito se a tarefa a cumprir fosse navegar do nó *QUARTO* para o nó *SALA*.

O movimento fica não só descrito pela transição mas também pelo nó objectivo. Deste modo, poucas transições são suficientes para definir um mapa complexo.

O mapa topológico pode ser descrito duma forma bastante simples num ficheiro de texto constituído por três secções: uma primeira e uma segunda contendo as características associadas a cada nó e a cada transição, respectivamente, e a terceira com a descrição dos vários caminhos possíveis. Seguindo o exemplo que vem sendo utilizado, apresenta-se na Figura 3.2 uma descrição da parte do grafo da casa apresentado na Figura 3.1.

```
Nós:
Nó          | Geometria da Porta | Orientação da Porta | Descrição
-------------------------------------------------------------------------------------------
1. SALA           1.40 x 1              0.0          Junto à porta da Sala, de frente para a mesma.
2. QUARTO         2.80 x 1              0.0          Junto à porta do Quarto, de frente para a mesma.


Transições:
Transição | Custo | Rotação Inicial [º] | Vel. Linear [m/s] | Vel. Angular [rad/s] | Descrição
-------------------------------------------------------------------------------------------------------------
1. r&mf       1            180                 1.0                 0.0             Roda 180º e move-se em frente.


Grafo:
Origem | Transição | Destino
-----------------------------
SALA:
           r&mf      QUARTO

QUARTO:
           r&mf       SALA
```

Figura 3.2 - Descrição de parte do grafo da casa

No exemplo referido, o robot detecta as portas recorrendo à razão entre a altura e largura (no caso da porta da sala essa razão é de 1.4 para 1), pelo que, para cada nó, está descrita essa característica geométrica. Está ainda descrita a orientação com que se pretende atingir o nó objectivo em relação à porta. Nas transições estão referidas as características básicas do movimento. Como se pode ver, no caso da transição $r\&mf$, é indicada a rotação inicial a imprimir ao robot, a velocidade linear e a velocidade angular que o robot deve usar. Tal como foi mencionado anteriormente, o movimento do robot não

fica completamente definido pelas características da transição. Contudo, é possível, ainda assim, controlar a velocidade linear e/ou angular, a fim de cumprir objectivos descritos pelo nó a atingir (neste caso a velocidade angular seria controlada de forma a manter a orientação do robot face à porta igual a 0). Existe ainda um campo que define o custo da respectiva transição, de forma a orientar a planificação de caminhos, conforme explicado no Capítulo 4.

Esta abordagem permite definir as características dos nós e das transições duma forma bastante simples. Se no exemplo anterior as portas fossem de cores diferentes e fosse tomada a opção de usar também a cor como característica das portas, bastaria, para definir o grafo através dessa característica, incluir mais um campo nos nós. A descrição do grafo apresentado na Figura 3.1 passaria a ser a que se ilustra na Figura 3.3. Obviamente, quando se incluem novas características, torna-se necessário implementar novas funções capazes de as detectar e usar.

```
Nós:
Nó          | Geometria da Porta |   Cor    | Orientação da Porta | Descrição
-------------------------------------------------------------------------------------------------------
1. SALA            1.40 x 1       Castanha            0.0          Junto à porta da Sala, de frente para a mesma.
2. QUARTO          2.80 x 1       Branca              0.0          Junto à porta do Quarto, de frente para a mesma.


Transições:
Transição | Custo | Rotação Inicial [º] | Vel. Linear [m/s] | Vel. Angular [rad/s] | Descrição
-------------------------------------------------------------------------------------------------------
1. r&mf       1            180                  1.0                  0.0             Roda 180º e move-se em frente.


Grafo:
Origem | Transição | Destino
-----------------------------
SALA:
            r&mf      QUARTO

QUARTO:
            r&mf       SALA
```

Figura 3.3 - Descrição de parte do grafo da casa (incluindo a característica *cor*.)

Na última secção da descrição do mapa topológico estão as ligações possíveis ente os vários nós do grafo. Para cada nó indicam-se as transições possíveis e os nós a que essas transições levam. No exemplo referido, os nós *SALA* e *QUARTO* estão ligados entre si pela transição $r\&mf$, estando essas ligações representadas na secção denominada *Grafo* na Figura 3.2 e na Figura 3.3 (as ligações são as mesmas, pelo que esta parte da descrição é igual em ambos os casos).

No exemplo dado todo os nós eram descritos segundo as mesmas características (geometria, ou geometria e cor). No entanto nada obriga a tal: é possível num caso usar uma característica e noutros casos usar outra. Basta para tal usar um símbolo no ficheiro para indicar que a característica não é usada. O mesmo se passa para os movimentos associados às transições. No exemplo dado, definiu-se que se a velocidade, angular ou linear, tiver um valor associado, então deve-se usar esse valor, caso seja 0, deve ser controlada. Apenas depende dos critérios usados no desenvolvimento do grafo.

Do mesmo modo que para o exemplo apresentado é bastante simples, poder-se-ia ter um caso mais complexo usando a mesma abordagem. Se fosse considerado, por exemplo, um mapa topológico em que houvesse vários nós definidos como cidades em Portugal, associadas às transições poderiam estar funções de elevada complexidade. Imaginando que três dos nós representavam as cidades de Évora, Funchal e Porto, ter-se-ia que a transição entre o nó *FUNCHAL* e o nó *PORTO* seria uma função *ir_de_avião* e a transição entre o nó *PORTO* e o nó *ÉVORA* seria *ir_de_carro*. As funções associadas às transições implementariam as viagens nos respectivos transportes.

No âmbito do presente trabalho, os nós poderão ser associados a certas zonas do campo, como o meio-campo ou a zona junto das balizas, enquanto que as transições representarão funções de navegação simples, semelhantes às indicadas para o exemplo da *SALA* e do *QUARTO*.

3.2 Representação dos Nós e Localização no Grafo

Quaisquer que sejam os nós do grafo construído, é importante saber como associar a sua conceptualização à realidade física dos mesmos. Uma abordagem possível, e utilizada no presente trabalho, consiste em retirar um conjunto de imagens de treino e, com base nos métodos referidos no Capítulo 2, associar grupos específicos de imagens aos nós do grafo.

De uma forma genérica, poder-se-á retirar um conjunto de imagens dos locais onde se pretende navegar para depois as agrupar segundo as características de cada nó. Obter-se-á, então, uma partição do conjunto de imagens iniciais, sendo que cada um dos conjuntos desta partição definiria um nó do grafo. É interessante referir que, com as mesmas imagens iniciais, diferentes partições irão ter resultados diferentes, ou seja, poderão ser definidos vários grafos a partir do mesmo conjunto de imagens iniciais.

No presente trabalho, as zonas associadas a cada nó do grafo foram áreas distintas do campo de futebol robótico. Tendo como base a partição do conjunto de imagens do campo, foram experimentadas duas abordagens distintas, descritas no Capítulo 2.3. Na primeira, utilizou-se o método de *MKL* para associar sub-espaços principais aos conjuntos de imagens que definiam cada um dos nós. Na segunda, gerou-se o espaço principal de todas as imagens e, depois de projectar as imagens de treino no espaço gerado, associaram-se os conjuntos das projecções aos nós do grafo.

Na Figura 3.4 esquematiza-se a execução do método de associação das imagens de treino e respectiva projecção no espaço principal para o exemplo da *SALA* e do *QUARTO*, de acordo com o descrito para a segunda abordagem.

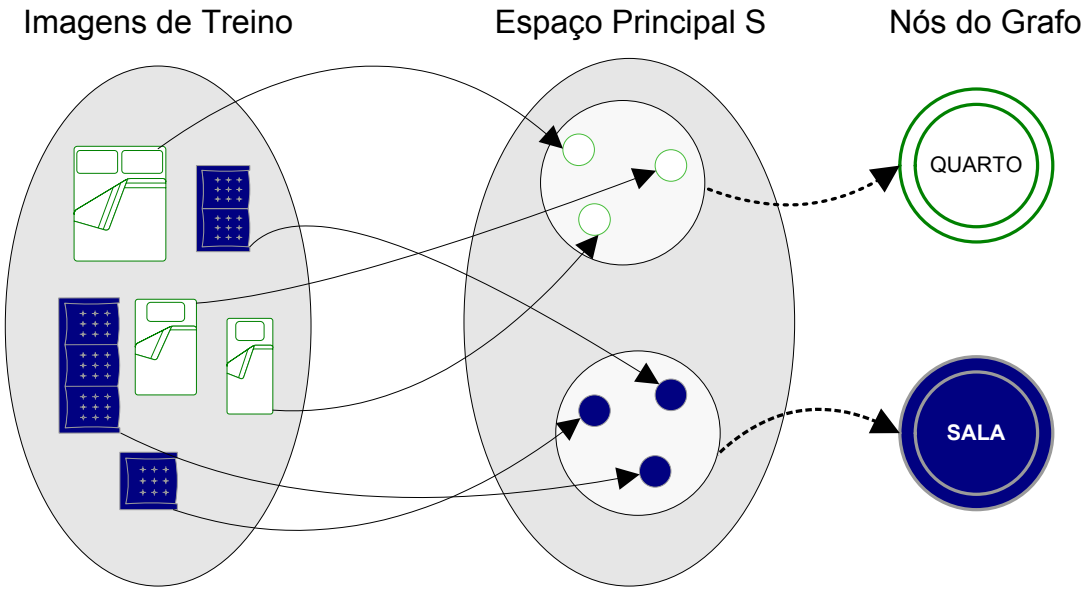


Figura 3.4 - Esquema de associação de imagens de treino com os nós do grafo

Uma aplicação deste método será a de permitir, com o mesmo conjunto de imagens de treino, criar vários grafos com finalidades distintas.

A desvantagem do método de *MKL*, quando usado para este fim, é o facto de implicar o cálculo de novos sub-espaços principais para cada um dos nós, o que é bastante dispendioso a nível computacional. O agrupamento simples das projecções das imagens de treino apenas implica a construção do espaço próprio global, a qual pode ser efectuada uma única vez para o mesmo conjunto de imagens, permitindo uma flexibilidade de utilização muito maior.

Além disso, este último método de associação dos dados permite não só a construção relativamente rápida de novos grafos, mas também a existência de grafos dinâmicos, nos quais a partição das imagens de treino poderá ser função de algum parâmetro, em tempo de execução, ou controlada por algum supervisor externo à navegação. Permite, também, não só a variação do tamanho dos conjuntos da partição mas ainda do número de conjuntos da partição em si, o que se traduz na introdução e remoção de nós no grafo.

No caso do futebol robótico, poder-se-á pensar em grafos distintos para situações de defesa e ataque, bem como grafos dinâmicos para usar em cooperação.

Uma vez definido o significado físico de cada um dos nós, pela associação de um conjunto de imagens (ou projecções das mesmas) que o caracterizam, torna-se necessário saber como, dada uma imagem, se pode identificar a que nó ela pertence.

Este problema pode ser resolvido através dos métodos referidos no Capítulo 2.3 para a classificação de padrões. Assim, no caso de ter sido associado um sub-espaço principal a cada um dos nós, através do método de *MKL*, a imagem será classificada no nó cujo espaço esteja a menor distância, ou seja, produza um erro quadrático de reconstrução menor. No caso de ter sido feito um agrupamento simples das imagens, poderá ser usado o método dos *k-vizinhos mais próximos* no espaço S das projecções, com alguma métrica associada, para identificar o nó no qual a imagem se insere.

Capítulo 4

Navegação Topológica

Como já antes se salientou, o objectivo principal deste trabalho visa a obtenção de um sistema de navegação para um robot futebolista, baseado numa descrição qualitativa da sua posição e dos seus movimentos.

Neste capítulo aborda-se a questão da navegação propriamente dita: a geração de caminhos e, finalmente, a execução desses caminhos. Tendo em conta que os locais-chave são descritos por nós de um grafo, a navegação será também efectuada sobre o suporte do grafo.

4.1 Geração de Caminhos

A principal pré-condição para ser possível efectuar a navegação é que seja possível, ao robot, determinar onde está. No presente trabalho, a localização corresponde à identificação do nó do grafo em que o robot se encontra, assunto já abordado no Capítulo 3.2.

Contudo, há ainda um melhoramento nesse possível nesse processo de localização que pode ser feito antes de gerar o caminho propriamente dito. Este consiste na introdução de um filtro na decisão do robot acerca da sua posição no mapa topológico, ou seja, no nó do grafo. Este procedimento, não sendo muito significativo num ambiente estático, torna-se essencial num ambiente dinâmico como é o caso do futebol robótico. O filtro destina-se, então, a reter os últimos p resultados da função de localização e considerar que o local actual corresponde ao nó mais votado. Para o grafo do exemplo da *SALA* e do *QUARTO*, descrito no capítulo 3, e para $p = 7$, o resultado da filtragem da situação ilustrada na Figura 4.1 seria *SALA*, apesar de ter ocorrido alguma oscilação na decisão imediata.

Quanto à geração de caminhos para usar na navegação, ela pode ser implementada por um algoritmo de procura de caminho num grafo. O algoritmo usado foi o A^*, tal como descrito em [9], sendo o nó inicial definido por aquele em que o robot se encontra e o nó final fornecido por um supervisor.

Este algoritmo de procura segue uma estratégia de *Procura Melhor Primeiro*, escolhendo sempre o nó a explorar (n) como aquele que minimiza uma função $f(n) = g(n) + h(n)$. Nesta, $g(n)$ corresponde à soma do custo de todas as transições desde o nó inicial até n e $h(n)$ é uma função heurística que estima o custo de viajar desde n até ao nó objectivo. Diz-se que a função $h(n)$ é admissível se não sobre-estimar o custo real da

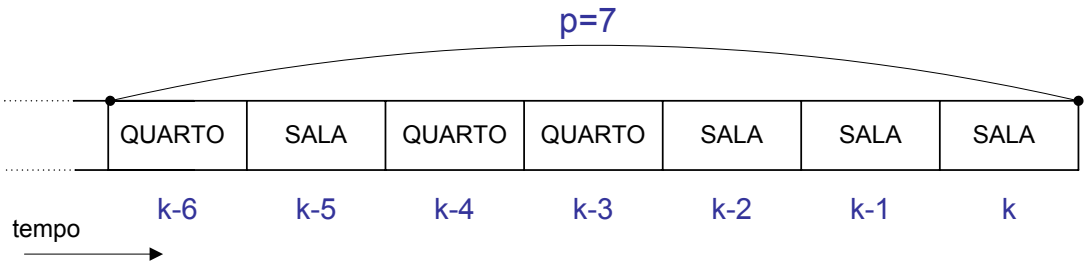


Figura 4.1 - Filtragem da decisão de localização

melhor solução para ir desde n até ao nó objectivo. Neste caso, a procura é completa e óptima no sentido do custo do caminho.

Inicialmente, o algoritmo foi ensaiado para um grafo com uma quantidade de nós muito mais elevada (≈ 500), já que cada nó correspondia a uma imagem. Nesta situação, a cada transição estava associado um custo e a cada nó um valor heurístico, o qual tentava prever o custo desse nó até ao nó objectivo. Teve-se o cuidado de tornar a heurística admissível, para garantir o sucesso do algoritmo em encontrar o caminho óptimo.

No entanto, para a aplicação final passou-se a usar um grafo com muito menos nós (≈ 10), em que a cada nó estava associado um conjunto de projecções das respectivas imagens de treino. Nesta situação, em que a procura é suficientemente rápida, não seria muito vantajoso definir uma heurística para cada nó. Assim, utilizou-se apenas o custo das transições, o que transforma o A^* numa procura de custo uniforme. Como caso particular, colocaram-se todos os custos com valor igual, o que faz degenerar a procura de custo uniforme na procura em largura. De notar que, em ambas as procuras referidas, a heurística continua a ser admissível, pelo que continuam a garantir completude e optimalidade no custo do caminho. Para grafos maiores não serão, no entanto, óptimas no tempo.

4.2 Execução de Caminhos

Dado um nó inicial (n_i) e um nó final (n_f), e depois de procurado no grafo o caminho mais próximo para chegar de um a outro (C), a execução do caminho deve ser equivalente à execução sequencial das transições entre os nós de C.

Assim, em cada instante seria executada uma transição entre dois nós e o único trabalho do supervisor de navegação seria detectar quando o robot se tinha localizado no nó de chegada para essa transição e, nesse caso, iniciar a execução da transição seguinte. Este supervisor será designado *Supervisor de Navegação em Cadeia Aberta*.

Considerando *Nó Partida* o nó de partida em cada transição, *Nó Chegada* o nó de chegada em cada transição e *Caminho* o caminho que ainda falta percorrer até ao nó final, o fluxograma da Figura 4.2 esquematiza esta operação.

No entanto, há uma questão por resolver no modelo apresentado que se pode tornar bastante problemática. O supervisor parte do pressuposto que, em cada transição, se o robot não se encontra em *Nó Partida* é porque já completou a transição até *Nó Chegada*. Esta forma de proceder não seria grave se fosse garantida a execução da transição por completo.

Contudo, se essa transição falhar, o robot pode localizar-se num nó diferente de *Nó*

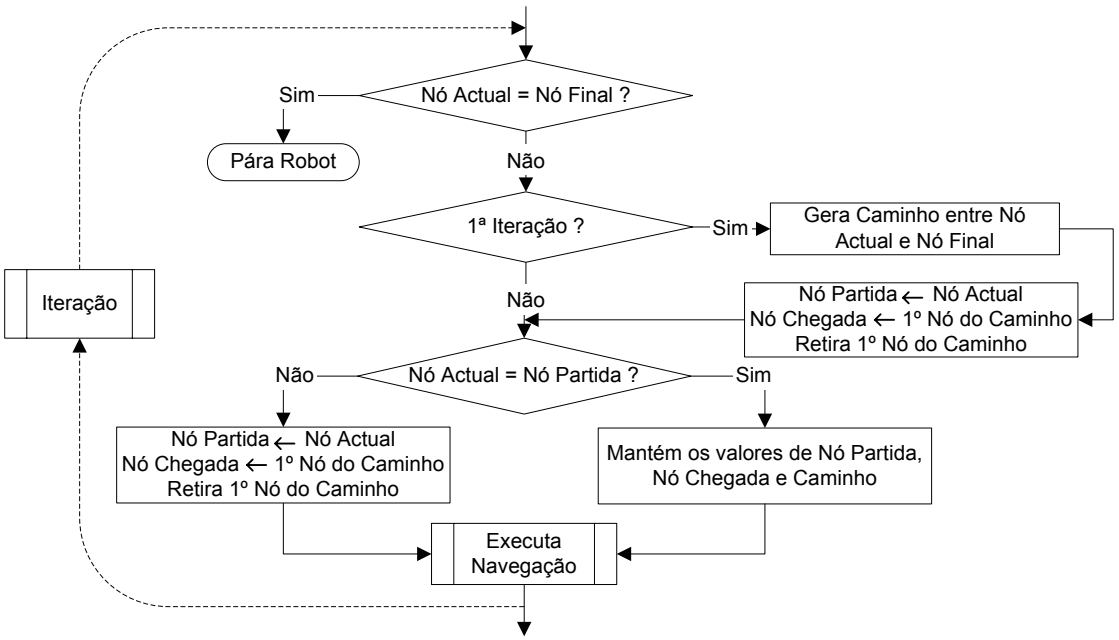


Figura 4.2 - Supervisor de Navegação em Cadeia Aberta

Partida ou *Nó Chegada*, e continuar a executar a transição anterior, apesar de neste momento ela não fazer sentido.

É necessário, então, introduzir no modelo uma forma de controlar as falhas nas transições, de modo a permitir gerar um novo caminho cada vez que a execução do actual não for bem sucedida. Esse melhoramento, que pode ser designado por *Supervisor de Navegação em Cadeia Fechada*, está expresso no fluxograma da Figura 4.3. Parte deste supervisor é igual ao anterior (antes do bloco de decisão *Nó Actual = Nó Partida?* e depois da função *Executa Navegação*), pelo que se julgou dispensável esquematizá-lo totalmente na Figura 4.3.

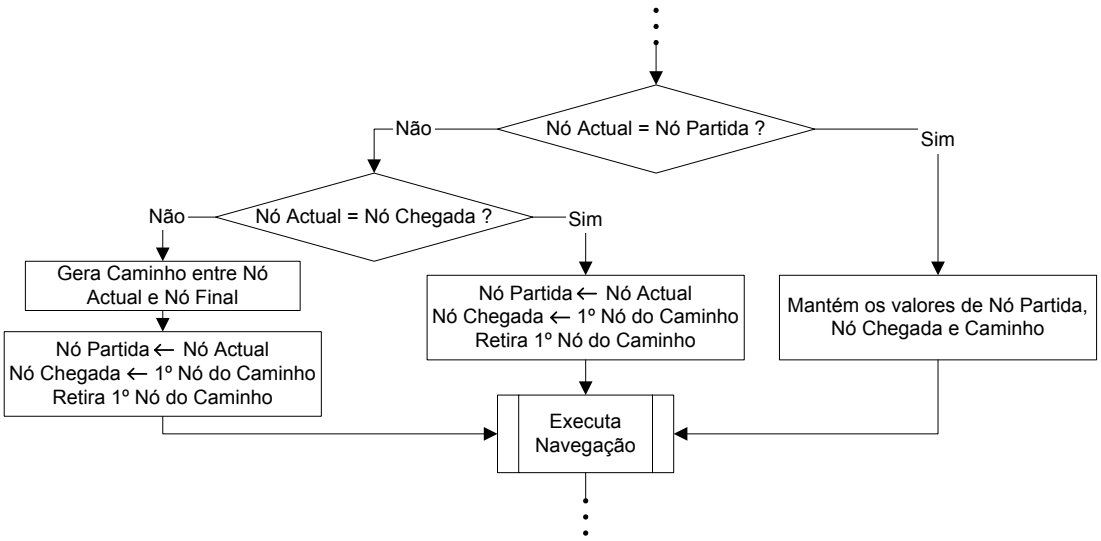


Figura 4.3 - Supervisor de Navegação em Cadeia Fechada

Capítulo 5

Aplicação ao Futebol Robótico

O ambiente do futebol robótico é constituído por um campo de futebol com cerca de 10% da área dum campo de futebol de 11, duas balizas coloridas, uma bola colorida e duas equipas de robots, sendo o principal objectivo num jogo conseguir derrotar a equipa adversária marcando mais golos que esta. A Navegação Topológica pode inserir-se no futebol de várias perspectivas. A perspectiva abordada neste trabalho consiste, todavia, em ter uma navegação *mais global*, a qual será usada em conjunto com outros tipos de navegação, *mais locais*, de forma a atingir os fins pretendidos. Neste capítulo apresentam-se os vários algoritmos anteriormente descritos, aplicados, segundo esta perspectiva, ao Futebol Robótico, bem como diversas experiências com o objectivo de avaliar diferentes parametrizações dos referidos algoritmos e o seu impacto na qualidade da navegação.

Como ferramenta de desenvolvimento e teste do projecto, foi desenvolvido um simulador no Matlab© (ver Anexo A). Dado que neste trabalho as imagens usadas correspondem a imagens da câmara da frente dum robot da equipa *ISocRob*, o simulador foi ajustado com os parâmetros das câmaras usadas nos robots reais. Foi ainda introduzido ruído do tipo *sal e pimenta* [10] para tornar a geração de imagens mais real. O nível de ruído utilizado foi de 12%, o que significa que em cada imagem 12% dos *pixels* têm a cor errada.

Para melhor perceber a ligação entre os vários componentes da navegação topológica (aplicada ao futebol robótico), criou-se o fluxograma apresentado no Anexo D.

5.1 Mapa Topológico

5.1.1 Definição

De acordo com o descrito no Capítulo 3, o primeiro passo no desenvolvimento do método consiste em elaborar o mapa topológico, definindo os nós e as transições do respectivo grafo. Na perspectiva mencionada, procurou-se definir um grafo em que os nós estivessem associados a várias zonas relevantes do campo e em que a navegação entre os nós fosse feita recorrendo a objectos do campo estáticos e de fácil detecção.

As regiões associadas a cada nó foram definidas recorrendo à *posição relativa* das balizas em relação aos robots, segundo a descrição apresentada na Tabela 5.1. A *posição relativa* da baliza é definida pela coordenada horizontal do centro da parte visível na imagem, pelo que pode ser especificada em termos de percentagem da largura da imagem. Se o valor dessa coordenada for 0, a baliza encontra-se no extremo esquerdo da imagem, enquanto que se for 1 (largura da imagem), encontra-se no extremo direito da imagem.

Nó	Cor	Posição da baliza	Descrição
NBGL	Azul	0.25	*Near Blue Goal Left* (A baliza azul é visível à esquerda do robot e está perto deste).
NBGR	Azul	0.75	*Near Blue Goal Right* (A baliza azul é visível à direita do robot e está perto deste).
FBG	Azul	0.50	*Far Blue Goal* (A baliza azul é visível e está longe do robot).
NYGL	Amarelo	0.25	*Near Yellow Goal Left* (A baliza amarela é visível à esquerda do robot e está perto deste).
NYGR	Amarelo	0.75	*Near Yellow Goal Right* (A baliza amarela é visível à direita do robot e está perto deste).
FYG	Amarelo	0.50	*Far Yellow Goal* (A baliza amarela é visível e está longe do robot).
NOG1	(nenhuma)	0	*No Goal 1* (Caso 1 em que não é visível nenhuma baliza).
NOG2	(nenhuma)	0	*No Goal 2* (Caso 2 em que não é visível nenhuma baliza).
NOG3	(nenhuma)	0	*No Goal 3* (Caso 3 em que não é visível nenhuma baliza).
NOG4	(nenhuma)	0	*No Goal 4* (Caso 4 em que não é visível nenhuma baliza).

Tabela 5.1 - Descrição dos nós

Estando os nós identificados, definiram-se as transições necessárias para poder navegar entre os vários nós. Procurou-se, no entanto, definir um número básico de transições que possibilitasse uma grande navegabilidade. Tendo por base os 4 principais movimentos do robot, definiram-se 5 transições cuja descrição se encontra na Tabela 5.2.

Usando estas transições para ligar os nós definidos anteriormente construiu-se o grafo representado na Figura 5.1. Da análise deste grafo percebe-se que o robot é capaz de realizar uma grande quantidade de movimentos, usando apenas 10 nós e 5 transições básicas. Embora possa não estar ainda muito claro quais as regiões definidas por cada nó, ao longo do presente capítulo serão fornecidos outros detalhes que irão clarificar esta questão.

O ficheiro de texto elaborado com a descrição do grafo usando o método apresentado no Capítulo 3 está presente no Anexo B.

5.1.2 Associação de Imagens aos Nós

Uma vez definidos os nós, torna-se importante desenvolver métodos para associar imagens com determinadas características a determinados nós. Conforme se assinalou no Capítulo 2, cada nó fica identificado recorrendo a um conjunto de imagens a ele associ-

Transição	Custo	Vel. Linear	Vel. Angular	Movimento
rr	1	0.0	-0.7	*rotate right* (Rodar para a direita).
rl	1	0.0	0.7	*rotate left* (Rodar para a esquerda).
mfgr	1	0.5	0.0	*move forward goal right* (Movimentar-se em frente mantendo a baliza visível do seu lado direito).
mfgl	1	0.5	0.0	*move forward goal left* (Movimentar-se em frente mantendo a baliza visível do seu lado esquerdo).
mb	1	0.5	0.0	*move backward* (Movimentar-se para trás mantendo a baliza visível ao centro).

Tabela 5.2 - Descrição das transições

adas. Torna-se, assim, necessário desenvolver algoritmos que sejam capazes de associar cada imagem a um nó especifico.

A descrição dos nós esboçada na secção anterior indicia o modo de associar imagens aos grupos. Usando a cor das balizas, é possível determinar o centro de massa das mesmas na imagem, a altura (em *pixels*) e a área visível na imagem em *pixels* (veja-se a Figura 5.2). O centro de massa é usado para decidir em que lado (esquerdo ou direito, dentro dos limites da imagem) se encontra a baliza, enquanto que a altura e a área são usados para determinar se a baliza se encontra perto, longe ou fora do alcance visual do robot. Estas três características adequam-se perfeitamente à definição dos nós feita na Tabela 5.1.

Com esta descrição torna-se óbvio que não há distinção na classificação entre os vários nós onde a baliza não é visível (*NOG1*, *NOG2*, *NOG3* e *NOG4*). Não obstante, é possível navegar para um *NOG* específico dado que o grafo é determinístico e podem ser gerados caminhos que levem ao nó pretendido, mesmo que seja um *NOG*. Esta similaridade entre os nós *NOG* também não é muito relevante, tendo em conta que os nós onde as baliza não são visíveis não são especialmente relevantes para o comportamento do robot no âmbito do futebol robótico.

Mesmo usando as características da baliza definidas anteriormente, não é necessário qualquer conhecimento das grandezas reais da baliza. No caso deste trabalho, os limiares de decisão entre as regiões podem sempre ser obtidos com base em imagens de locais específicos do campo. Por exemplo, para determinar a altura (um dos parâmetros usados para distinguir entre estar perto ou longe duma baliza) que define o limiar entre a região, usou-se a altura da baliza detectada numa imagem capturada com o robot no meio campo. Definiu-se, deste modo, que o limiar de decisão entre estar perto e estar longe da baliza se identifica com a linha do meio campo.

5.1.3 Extracção de Dados

Tendo desenvolvido as funções para associar imagens a determinados nós (também designados por grupos), há que descrever os grupos recorrendo às técnicas de extracção de dados descritas no Capítulo 2.

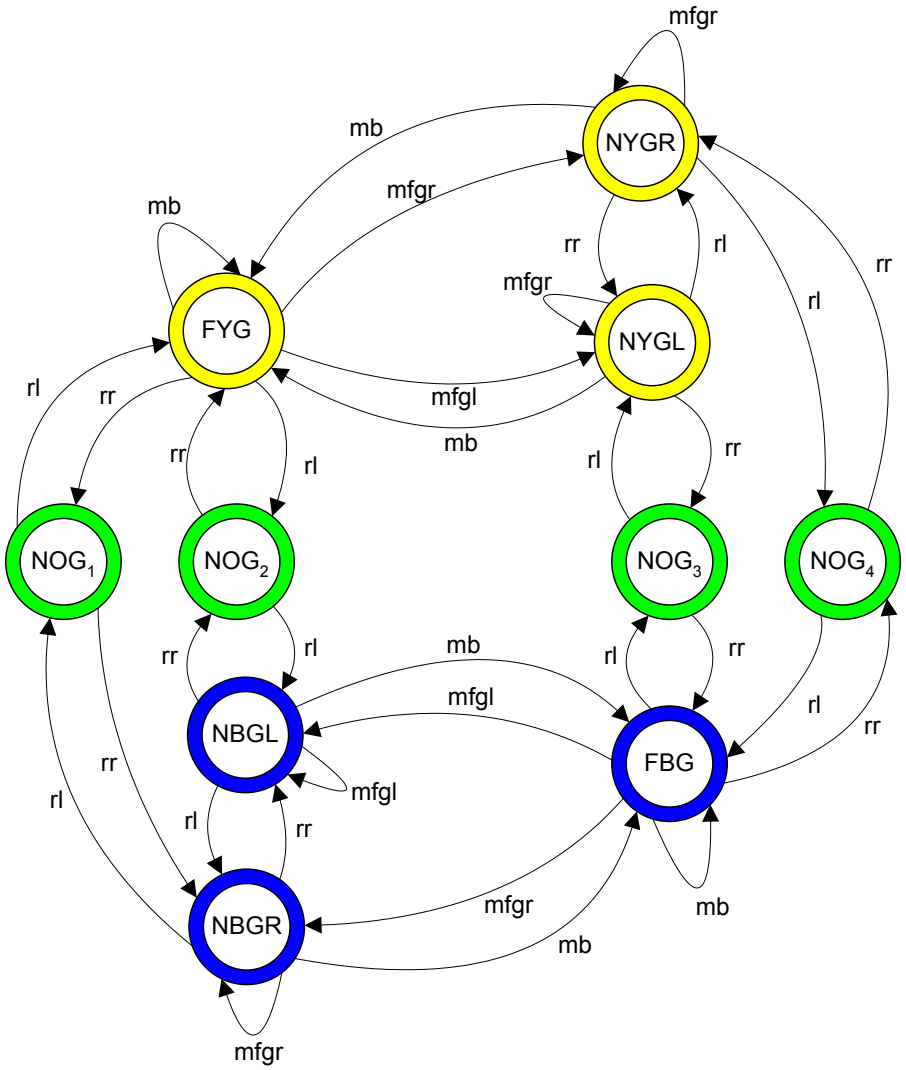


Figura 5.1 - Mapa Topológico para Futebol Robótico

Usando os dois algoritmos descritos anteriormente, o *KL* e o *MKL*, procedeu-se à extracção dos dados. A abordagem usada consistiu em construir um conjunto de imagens retiradas em várias posturas no campo, uniformemente distribuídas ao longo de x, y e θ. Na Figura 5.3 estão ilustradas as posturas onde foram adquiridas as imagens de treino no caso do espaçamento entre posições ser de $1m$ em x e em y e de 45° em θ, num campo com $5m$ de largura e $10m$ comprimento.

Agrupou-se, então, cada uma das imagens geradas ao longo da grelha especificada, de acordo com o modo indicado na secção anterior (apresenta-se no Capítulo 6 uma tabela de frequências das imagens por grupo, indicando o número de imagens atribuídas a cada nó). Aos conjuntos de imagens assim definidos, aplicaram-se os métodos *KL* e o *MKL*, gerando o espaço principal (ou espaços principais no caso do *MKL*):

***KL*:** Gerou-se um espaço principal usando todas as imagens capturadas, guardando as projecções nesse espaço principal de cada uma delas. A projecção de cada imagem foi associada ao grupo em que a imagem se insere. O conjunto de projecções de cada grupo foi guardado de forma a ser possível, em tempo-real, classificar as imagens que vão sendo adquiridas.

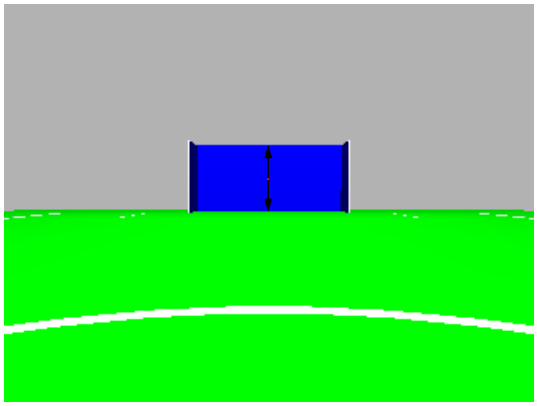


Figura 5.2 - Características da baliza na imagem

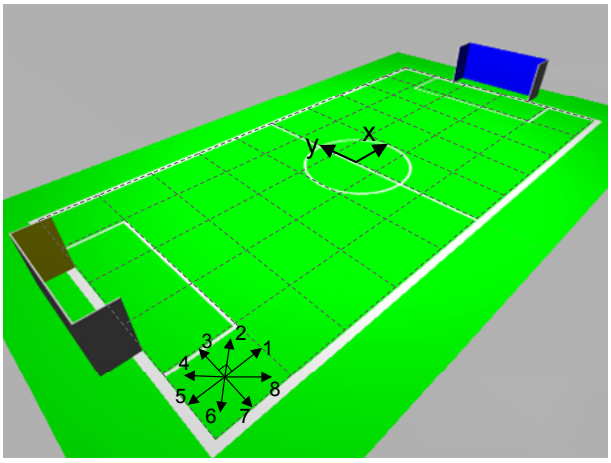


Figura 5.3 - Posturas onde são adquiridas as imagens de treino

***MKL*:** Para cada nó, pegou-se no respectivo conjunto de imagens e gerou-se o espaço principal por elas criado. Foram gerados 7 espaços principais (ter em conta que para o *NOG1*, *NOG2*, *NOG3* e *NOG4* o conjunto de imagens associado é o mesmo, logo o espaço principal gerado é o mesmo).

Para determinar o número de valores próprios a usar para gerar cada espaço principal, obtiveram-se primeiro todos os valores próprios para cada espaço e seleccionou-se o número de valores próprios de modo que o erro quadrático médio de reconstrução (em percentagem) fosse inferior a 20

No Capítulo 6 apresentam-se os gráficos da evolução do número de valores próprios em função do erro quadrático médio de reconstrução percentual (ξ), para o caso do método *KL*, indicando o número de valores próprios no caso de ξ ser igual a 20% e 40%. Mostra-se ainda um gráfico com a função inversa, mas obtida de forma experimental através do cálculo, para um conjunto de 1000 posturas aleatórias, do erro quadrático médio de reconstrução em casos com valores próprios diferentes. Em relação ao método *MKL*, apresenta-se para cada um dos espaços principais associados a cada nó, o número mínimo de valores próprios necessários para que ξ seja inferior a 20%.

Para ter uma ideia visual do efeito da expansão *KL* sobre uma imagem, foram usadas duas imagens distintas, uma de treino (usada para gerar o espaço) e outra de teste (não pertencendo ao conjunto de imagens de treino) e, no Capítulo 6, são apresentadas

as reconstruções de ambas as imagens, depois de terem sido projectadas em 4 espaços principais. Estes espaços foram obtidos usando 10, 25, 40 e 55 valores próprios.

5.2 Localização e Navegação

Tendo todo o conhecimento *a priori*, definem-se agora os métodos usados em tempo de execução.

As imagens foram tratadas usando os métodos definidos nos capítulos anteriores, resultando na localização do robot num nó do mapa topológico. A localização foi implementada usando os dois métodos enunciados, o *KL* e o *MKL*, sendo a sua implementação algo diferente. No Capítulo 3.2 é feito um estudo da aplicação destes dois métodos, pelo que agora se retrata a sua implementação ao caso do Futebol Robótico, segundo o mapa topológico definido na secção anterior.

No caso do método *KL*, determina-se a distância da projecção da imagem capturada em tempo de execução às projecções de todas as imagens capturadas *a priori* (para gerar as imagens para os nós usou-se a mesma grelha de posturas que se usou para as imagens de treino). A menor das distâncias determina qual das imagens capturadas *a priori* está mais perto da imagem capturada no momento, pelo que o nó onde se vai localizar o robot será o nó ao qual está associada a imagem capturada *a priori*.

Para minimizar o erro de localização, em vez de usar a imagem mais próxima, recorreu-se ao método dos *k-vizinhos mais próximos*. Deste modo, a localização do robot não se baseia apenas numa distância, numa imagem, mas sim nas *k imagens mais próximas*, sendo atribuído o nó mais frequente entre elas.

No caso do método *MKL* a localização foi feita de modo diferente, determinando o espaço principal (nó) mais próximo da imagem capturada pelo robot (ver Capítulo 3.2).

No Capítulo 6 apresenta-se uma tabela com a comparação do custo temporal dos vários métodos. Compara-se o tempo de processamento do método *KL* usando número de vizinhos e métricas diferentes (ver os parâmetros usados na Tabela 5.3) com o tempo de processamento do método *MKL*, na classificação dum conjunto de 10, 20 e 40 posturas diferentes.

k	**Métrica**
1	Euclidiana
1	Ponderada
5	Euclidiana
5	Ponderada
10	Euclidiana
10	Ponderada

Tabela 5.3 - Parâmetros usados para testar a localização com o método *KL*

No Anexo C apresentam-se ainda os gráficos com o resultado da localização usando o método *MKL* e o método *KL* (associado ao método dos *k-vizinhos mais próximos*), para as situações descritas na Tabela 5.4 e usando os valores de k e as métricas sintetizados na Tabela 5.3. Para cada situação, uma das variáveis foi fixa e as outras variadas em intervalos do valor especificado.

	X	**Y**	**Θ**
Situação 1	Fixo, $x = 3m$	Variável, $\Delta y = 0.2m$	Variável, $\Delta\theta = 6°$
Situação 2	Variável, $\Delta x = 0.2m$	Fixo, $y = 0m$	Variável, $\Delta\theta = 6°$
Situação 3	Variável, $\Delta x = 0.2m$	Variável, $\Delta y = 0.2m$	Fixo, $\theta = 138°$

Tabela 5.4 - Posturas usadas para testar a localização.

Para tentar obter ainda melhores resultados na localização em tempo de execução, integrou-se no sistema o filtro descrito em 4.1, de modo a eliminar o efeito de localizações esporádicas erradas que poderiam aparecer durante o movimento do robot.

Depois de localizado, o robot, para navegar, apenas precisa de saber qual o caminho a seguir e qual a transição que deve executar.

Tal como indicado no Capítulo 4.1, foi implementado um algoritmo de procura em grafos para gerar o caminho entre o nó onde se localizou o robot e o nó onde se pretende chegar (nó objectivo). Dada a forma usada para descrever o grafo, é bastante simples alterar os custos associados às transições, variando estes entre a *procura de custo uniforme* e a *procura em largura*, no caso em que os custos são todos iguais (ver Capítulo 4.1, para mais detalhes). No Capítulo 6 apresenta-se um caso comparativo do resultado do caminho encontrado, usando a *procura em largura* e a *procura de custo uniforme*, na qual os custos das transições utilizados foram os definidos na Tabela 5.5.

	Procura em largura	**Procura de custo uniforme**
rr	1	2
rl	1	2
mfgr	1	1
mfgl	1	1
mb	1	2

Tabela 5.5 - Custos usados para comparar os métodos de procura.

Uma vez implementado o método de navegação, torna-se imprescindível efectuar alguns testes com todos os blocos a funcionar. Assim, após testar os comportamentos individuais de cada módulo do sistema, é possível obter um visão geral do seu desempenho executando-os em conjunto.

Uma das questões sobre a qual era importante mostrar resultados é a das falhas na execução dum caminho em tempo real. Para apresentar resultados sobre esta matéria, desenhou-se um supervisor de objectivos que verifica, em cada momento, se o robot já atingiu o objectivo proposto (o *Nó Final* pretendido) e, se for o caso, lhe atribui um novo objectivo, escolhido aleatoriamente do conjunto de nós existentes. Deste modo, foi possível deixar o robot a navegar durante um tempo considerável e contabilizar o numero de falhas por caminho (em percentagem) através de:

$$Falhas_{i,j} = \frac{N_F(i,j)}{N_C(i,j)} * 100 \tag{5.1}$$

onde $N_F(i,j)$ é o número de falhas entre o nó i e o nó j e $N_C(i,j)$ é o número de vezes que foi executado o caminho entre o nó i e o nó j.

Note-se que só foram calculadas as percentagens para caminhos entre nós consecutivos, pois todos os outros casos são compostos por estes.

De forma a apresentar o resultado da aplicação do método na sua totalidade, foi executado o simulador em inúmeras situações distintas, mostrando-se o resultado desses ensaios no Capítulo 6.

Realizou-se ainda um teste para demonstrar a flexibilidade do método desenvolvido. Este consistiu em atribuir um conjunto de objectivos, mas definindo um novo limiar entre os nós *perto da baliza* e o nó *longe da baliza*, para cada cor, na zona central de cada metade do campo.

Capítulo 6

Resultados

Neste capítulo são apresentados os resultados obtidos no estudo experimental descrito no Capítulo 5. Tal como indicado anteriormente, todos os resultados foram obtidos em Matlab©, usando o simulador desenvolvido.

6.1 Mapa Topológico

Seguindo a ordem utilizada no capítulo anterior, apresenta-se de seguida uma tabela com a frequência de associação das imagens de treino aos nós. Esta associação foi feita usando as funções de processamento de imagem desenvolvidas para o efeito.

Nó	**Nº Imagens**	**Em percentagem**
NBGL	35	6.6
NBGR	39	7.4
FBG	56	10.6
NYGL	35	6.6
NYGR	39	7.4
FYG	56	10.6
NOG	268	50.8
Total	**528**	**100**

Tabela 6.1 - Frequência da associação das imagens de treino aos nós

Dado que as balizas ocupam um pequeno espaço na área do campo, é natural que mais de 50% das imagens apareçam associadas ao nó em que nenhuma baliza é visível. De notar ainda que os números de imagens obtidas longe da baliza e perto da baliza são semelhantes, para cada cor, dado que a linha de transição entre as respectivas regiões é sobre o meio-campo. Obviamente que estas afirmações só fazem sentido devido ao espaçamento uniforme usado para capturar as imagens de treino.

Os resultados seguintes dizem respeito à relação entre o erro quadrático médio de reconstrução e número de valores próprios para o método *KL*. Na Figura 6.1 apresenta-se o gráfico com o número de valores próprios em função do erro quadrático médio de reconstrução, em percentagem.

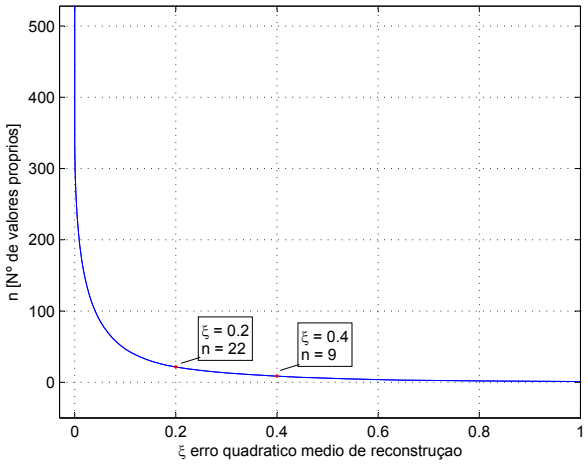


Figura 6.1 - Evolução do número de valores próprios em função do erro quadrático médio de reconstrução

Tal como seria de esperar existe um conjunto de valores próprios com grande peso, permitindo que, com apenas 22 valores próprios, se tenha conseguido atingir o limiar de erro de 20%.

No gráfico da Figura 6.2 compara-se o erro quadrático médio de reconstrução, como função do número de valores próprios, em duas situações distintas. Numa a curva foi obtida experimentalmente, enquanto que na outra a curva corresponde ao erro esperado usando apenas imagens do conjunto de treino.

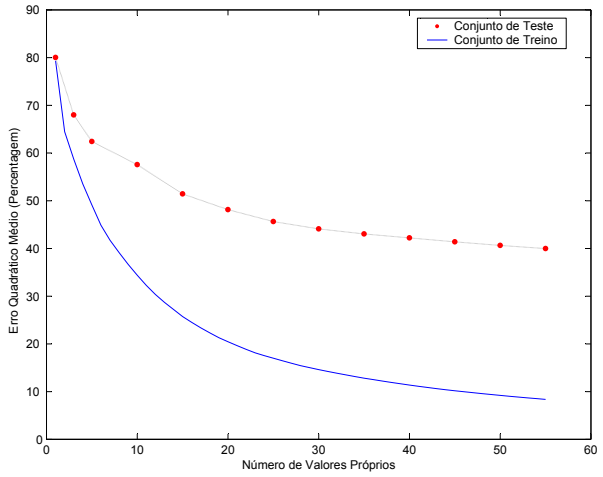


Figura 6.2 - Evolução do erro quadrático médio de reconstrução em função do número de valores próprios (Conjuntos de treino vs teste)

O resultado patente na Figura 6.2 é bastante interessante já que, à parte um factor de escala, a evolução das duas curvas é bastante semelhante. Tal deve-se ao facto do erro quadrático médio de reconstrução ser definido para as imagens de treino, no espaço criado usando essas mesmas imagens. No caso das imagens capturadas em posturas aleatórias no campo, a maior parte (se não todas) não coincide com as imagens de treino. Assim sendo, está-se a reconstruir imagens usando um espaço do qual estas poderão não fazer parte. Desta forma, o erro tem sempre que ser superior ao erro obtido quando usadas as

imagens de treino.

Apresentam-se agora os resultados em relação ao número de valores próprios para o método *MKL*. A Tabela 6.2 contém o número de imagens associado a cada nó, o número de valores próprios (para o caso em que $\xi = 20\%$) e a razão entre o número de valores próprios e o número de imagens.

Nó	**Nº Valores Próprios**	**Nº Imagens**	λ/N_{img}
NBGL	7	35	0.20
NBGR	4	39	0.10
FBG	16	56	0.26
NYGL	8	35	0.23
NYGR	5	39	0.13
FYG	15	56	0.27
NOG	8	268	0.03

Tabela 6.2 - Número de valores próprios no caso do método *MKL*

De notar, que apesar de o número de imagens associadas ao nó *NOG* ser elevado, o número de valores próprios necessários para que o erro quadrático médio de reconstrução seja 20% é bastante baixo. De facto, este conjunto apresenta a razão entre número de vectores próprios e números de imagens mais baixa. Este resultado era expectável neste caso, uma vez que as imagens associadas ao nó *NOG* são muito semelhantes (razão pela qual todos os *NOG*s são classificados no *NOG*).

A situação mais delicada é a dos nós em que a baliza está longe, pois as imagens, embora sensivelmente parecidas, possuem uma grande variação na posição da baliza, levando a que sejam necessários mais valores próprios para garantir o mesmo erro quadrático médio de reconstrução.

Um resultado relevante é a comparação do número de valores próprios necessários para cada espaço no caso do método *MKL* e o número de valores próprios no caso do método *KL*. Enquanto que com o método *MKL* o número máximo de valores próprios necessários é 16, com o método *KL* são necessários 22. Esta diferença não é surpreendente já que, além do número de imagens para cada espaço ser muito menor no *MKL* que no *KL*, há ainda que ter em consideração a grande semelhança entre as imagens de cada grupo.

A reconstrução das imagens apresentadas nas Figuras 6.3 e 6.4 (imagens de treino e de teste respectivamente) está ilustrada nas Figuras 6.5 e 6.6.

Em ambos os casos é nítida a melhoria na reconstrução das imagens com o aumento do número de valores próprios. De notar ainda que esse aumento, sendo significativo até aos 25 valores próprios, passa depois a ser muito inferior, já quase não se notando a diferença entre usar 40 ou 55 valores próprios. Apesar de não ser acentuada a diferença, percebe-se que, com 25 valores próprios, a reconstrução da imagem de treino é melhor sucedida que a reconstrução da imagem de teste. Era de esperar este facto, já que a perten ça ao conjunto de imagens de treino assegura à partida que, com um erro maior ou menor, a imagem estará mais próxima do espaço principal.

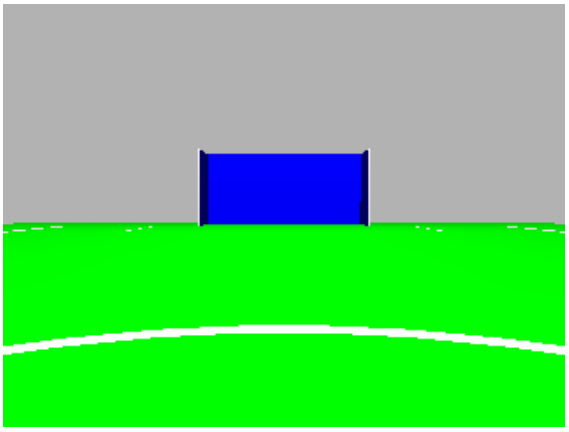


Figura 6.3 - Imagem de treino usada para reconstrução.

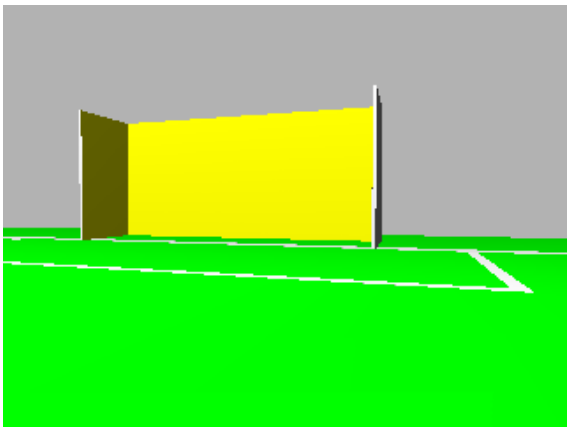
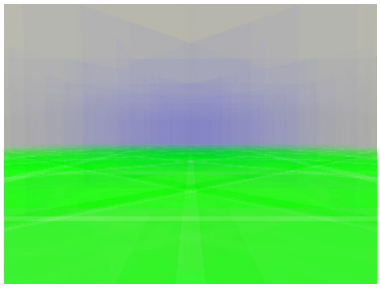
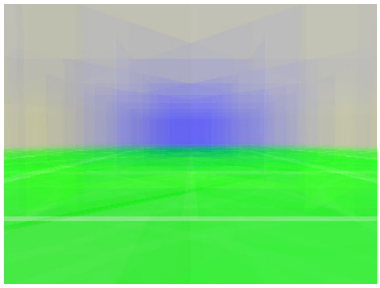


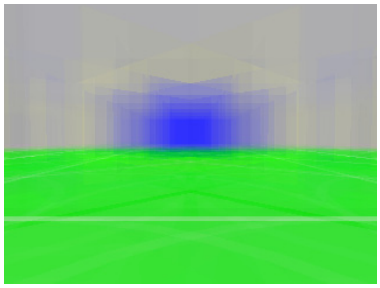
Figura 6.4 - Imagem de teste usada para reconstrução.



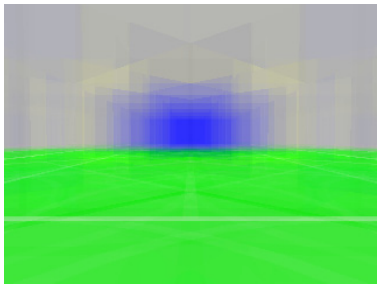
(a) 10 Valores Próprios



(b) 25 Valores Próprios

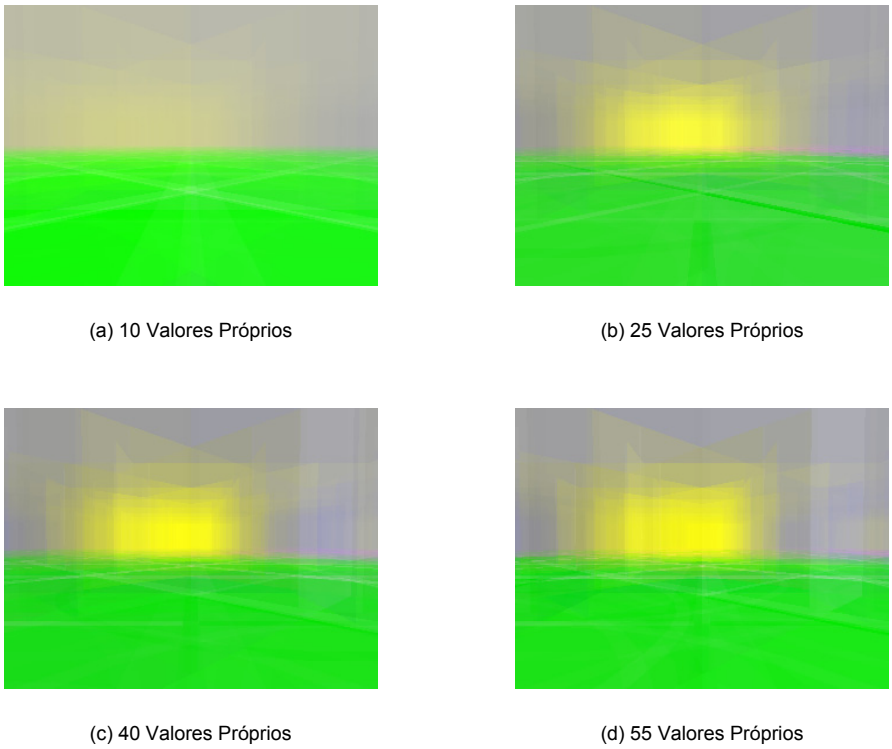


(c) 40 Valores Próprios



(d) 55 Valores Próprios

Figura 6.5 - Reconstrução de uma imagem de treino.



(a) 10 Valores Próprios

(b) 25 Valores Próprios

(c) 40 Valores Próprios

(d) 55 Valores Próprios

Figura 6.6 - Reconstrução de uma imagem de teste.

6.2 Localização e Navegação

Na Tabela 6.3 apresentam-se os tempos de processamento da localização, usando ambos os métodos e variando também os diferentes parâmetros na localização.

Métodos	***k***	**Métrica**	**Número de posturas**			**Tempo Médio de Classificação**
			10	**20**	**40**	
KL	1	Eucl.	1.28	2.56	5.20	0.129
	1	Pond.	1.55	3.06	6.13	0.153
	5	Eucl.	1.41	2.81	5.64	0.141
	5	Pond.	1.72	3.34	6.70	0.169
	10	Eucl.	1.64	3.15	6.31	0.160
	10	Pond.	1.83	3.67	7.31	0.183
MKL			7.06	14.08	28.17	0.705

Tabela 6.3 - Custo temporal (em s) dos métodos (com variação de parâmetros)

Da análise da tabela podem ser extraídas, para o método *KL*, algumas dependências dos parâmetros. Assim, pode dizer-se que o tempo de classificação cresce com o aumento de k, o que era previsível, já que essa variação acarreta um aumento na computação (ordenação do vector com os *k-vizinhos mais próximos* em cada iteração). Quanto à dependência da métrica usada, é natural que a métrica euclidiana seja menos pesada, pois não envolve a multiplicação por uma matriz, no cálculo da distância.

Efectuando, ainda, uma comparação com a classificação com recurso ao *MKL*, é bastante nítida a diferença no tempo de computação necessário entre os dois métodos. Resta, desse modo, verificar os resultados da classificação em si, de forma a saber se os algorit-

mos mais pesados justificam o consumo temporal que apresentam.

Para poder fazer um boa opção na escolha de um método de localização foram realizadas localizações recorrendo, para os vários parâmetros, aos valores indicados na Tabela 5.3 para cada uma das situações definidas na Tabela 5.4. Dada a dimensão dos resultados, optou-se por apresentar os gráficos com os resultados da localização no Anexo C.

Comparando, para os vários valores de k, o resultado da classificação com base no método *KL* e a métrica euclidiana, retira-se imediatamente que, para $k = 1$, há mais localizações erradas que nos outros casos. As situações em que $k = 5$ e em que $k = 10$ são mais semelhantes, mas, no entanto, para $k = 10$ o limiar entre as regiões de decisão torna-se menos claro, tornando a localização especialmente difícil nestes casos.

No caso em que se usa a distância ponderada, também facilmente se infere que o pior caso corresponde a $k = 1$, em que ocorrem muitas falsas localizações. Nesta situação torna-se mais difícil concluir qual dos casos, $k = 5$ ou $k = 10$, é o melhor, pois são bastante parecidos. Apesar disso, através de uma análise mais detalhada e visualizando os resultados da localização noutras posturas, pôde igualmente concluir-se que os resultados parecem ser melhores para $k = 5$.

Quanto à classificação baseada no *MKL*, verificou-se que o método funciona bem para lidar com falsos positivos, embora, por outro lado, as classes obtidas não correspondam ao que seria de desejar para esta aplicação.

Na Tabela 6.4 resume-se a qualidade dos resultados em termos de sucesso na localização e em termos de custos computacionais. A conclusão final é que o método *KL* associado aos *k-vizinhos mais próximos*, com $k = 5$, e usando a métrica euclidiana será o melhor a integrar neste trabalho.

Método	**Localização**	**Tempo/Recursos**
KL, eucl., $k = 5$	++	+++
KL, pond., $k = 5$	++	++
MKL	+	–

Tabela 6.4 - Comparação dos métodos de localização

Antes de apresentar todo o sistema em funcionamento, mostram-se ainda resultados referentes à navegação. Assim, na Figura 6.7 está indicado o caminho gerado entre o nó *FBG* e o nó *FYG* utilizando os custos indicados na Tabela 5.5.

O caminho denominado 1 corresponde ao caminho gerado pela procura em largura. Dado que nesta procura os custos são os mesmos, e sabendo que a primeira transição que se guarda em cada nó (caso exista) é *rr*, a procura tem tendência a gerar os caminhos que começam por rodar à direita, como se pode ver. As restantes opções feitas pela procura são óbvias, tendo em conta que os custos são iguais para qualquer transição. No caso da procura de custo uniforme, como se deu um custo inferior aos movimentos para a frente, a procura gerou um caminho em que optou por uma transição desse tipo, em lugar de uma em que se deslocasse para trás.

6.3 Análise Global da Implementação

Quanto à aplicação global do método ao problema em questão, tentou-se apresentar resultados descritivos das situações possíveis e do geral comportamento do método.

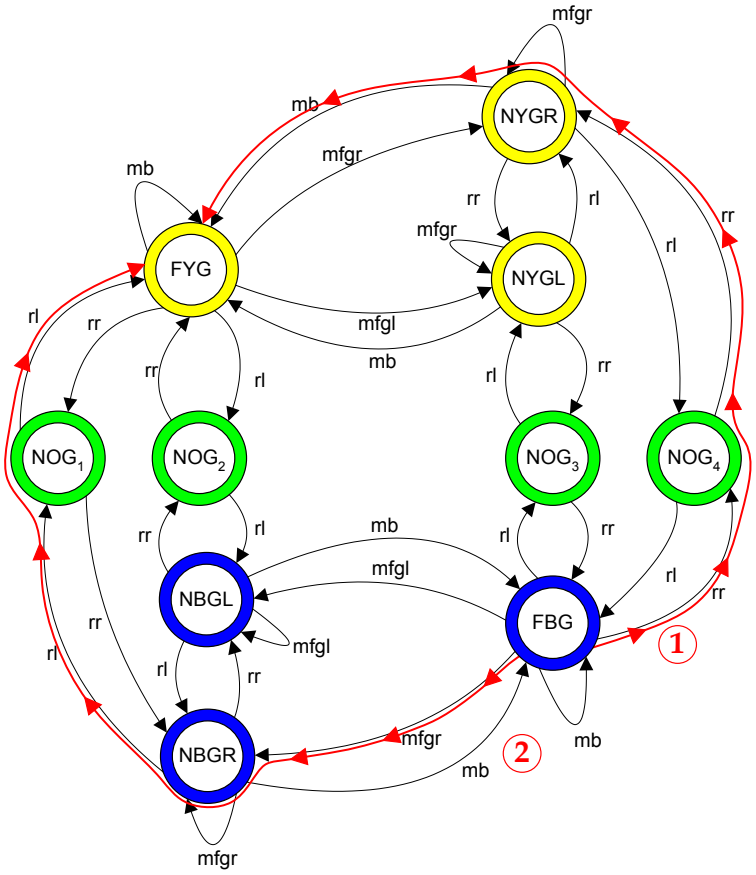


Figura 6.7 - Caminho gerado pela *procura em largura* (1) vs caminho gerado pela *procura de custo uniforme* (2).

O resultado da percentagem das falhas detectadas através do teste descrito em 5.2 é apresentado na Tabela 6.5.

	NBGL	**NBGR**	**FBG**	**NYGL**	**NYGR**	**FYG**	**NOG1**	**NOG2**	**NOG3**	**NOG4**
NBGL		0	0					58		
NBGR	29		0				44			
FBG	19	0							55	62
NYGL					25	5			15	
NYGR				20		7				0
FYG				10	20		63	56		
NOG1		15				27				
NOG2	20					63				
NOG3			50	25						
NOG4			50		0					-

Tabela 6.5 - Percentagem de Falhas nas transições entre nós

No teste foram realizadas 386 transições, o que significa que cada transição foi realizada 14.3 vezes, em média. Analisando a tabela, percebe-se que o número de falhas mais elevado aconteceu sempre que há um *NOG* envolvido, quer este seja um nó de partida ou um nó de chegada. Isto deve-se ao facto de as transições dos, ou para os, *NOG*s corresponderem a situações em que a baliza a aparece, ou desaparece, do alcance visual do robot. Ora, nos gráficos de localização, pode ver-se que estas posturas (posturas em que aparece apenas um pouco de um dos lados) correspondem a limiares de decisão ruidosos,

ou seja, menos claros. Após sair da zona de limiar, o robot tem muito menos problemas, tendo-se inclusivamente, observado diversos casos em que não houve falhas.

De salientar ainda que, mesmo nos casos com bastantes falhas, o robot conseguiu sempre atingir o seu objectivo final, o que parece constituir um bom indicador da robustez do método desenvolvido.

São apresentadas de seguida três situações: uma em que o robot percorre o caminho do nó inicial até ao nó objectivo sem problemas; uma outra em que lhe é fornecido um plano mais elaborado, com vários objectivos; e uma última na qual o robot entra numa situação de *live lock*.

Exemplo 1: Nesta situação o robot partiu do nó $NYGL$, na postura de $x = -3.5m$, $y = 1.7m$ e $\theta = 206°$, tendo como objectivo o nó $NBGL$ e gerou o seguinte caminho:

$$NYGL \rightarrow rr\ NOG_3 \rightarrow rr\ FBG \rightarrow mfgl\ NBGL$$

Este foi executado sem falhas nas transições, estando a trajectória real correspondente evidenciada na Figura 6.8. Nela, a trajectória do robot é indicada a cinzento escuro, as transições efectuadas com sucesso (correspondentes a localizações no nó esperado) com triângulos verdes e os locais onde foi gerado um novo caminho (tanto por falha na execução de uma transição como por recepção de um novo objectivo) com triângulos cor-de-laranja, ambos orientados segundo a trajectória. Estas ocorrências estão relacionadas com o fluxograma da Figura 4.3, que foi utilizado como supervisor de navegação.

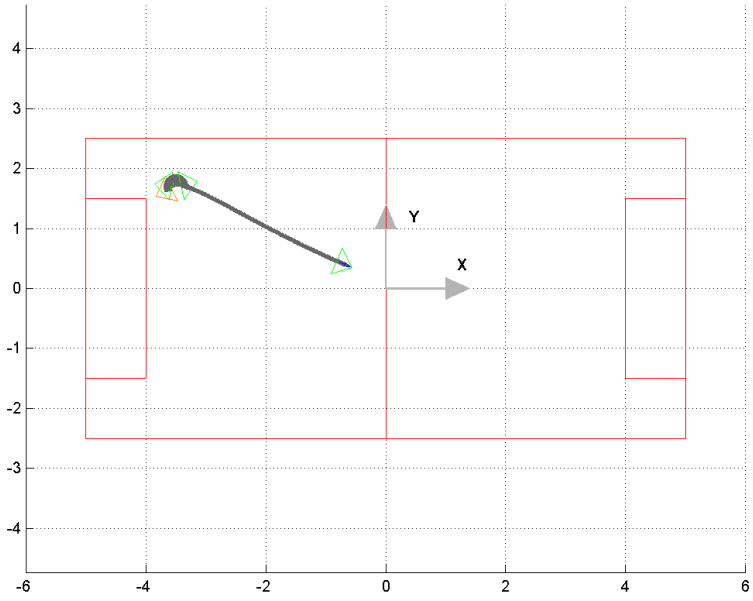


Figura 6.8 - Resultado de Navegação Topológica numa situação simples e bem sucedida.

De notar que, apesar do robot ter parado na zona do meio-campo, esta encontra-se no limiar do nó $NBGL$, tal como foi definido pelo conjunto de treino.

Exemplo 2: Neste caso, mais elaborado, foi fornecido ao robot um conjunto de objectivos a atingir. Sempre que o robot atingia um objectivo proposto, automaticamente

passava a perseguir o objectivo seguinte (ver a implementação do *Gerador de Planos* em A.1). De notar que não existe nenhuma obrigatoriedade para que isto aconteça. O supervisor poderá decidir que o robot deve abandonar o objectivo actual antes de o ter atingido, se essa for a função pretendida.

Desta forma, forneceu-se ao robot o conjunto de objectivos $\{FYG, NYGL, NOG1, NBGL, NYGR\}$ e atribuiu-se-lhe a postura inicial $x = 4.1m$, $y = -0.7m$ e $\theta = 278°$. Este começou por procurar atingir o objectivo FYG, depois de se ter localizado em *NOG1*. Foi, então, gerado o caminho:

$$C_1 : \quad NOG_1 \rightarrow rl\ FYG$$

Durante a transição o robot localizou-se, no entanto, em *NBGL*, pelo que surgiu a necessidade de gerar um novo caminho. De realçar que, na realidade, a primeira localização num nó do tipo *NOG* é ambígua, pois todos os nós são descritos da mesma forma e só são distinguíveis pela análise do passado:

$$C_2 : \quad NBGL \rightarrow rr\ NOG_2 \rightarrow rr\ FYG$$

A execução de C_2 decorreu sem erros e, após ter atingido FYG, o robot passou a deslocar-se para *NYGL* (segundo *Nó Objectivo* especificado inicialmente), pelo que foi gerado um novo caminho:

$$C_3 : \quad FYG \rightarrow mfgl\ NYGL$$

Após ter atingido *NYGL*, o novo objectivo passou a ser *NOG1* e foi igualmente gerado um novo caminho:

$$C_4 : \quad NYGL \rightarrow mb\ FYG \rightarrow rr\ NOG_1$$

Na execução da transição rr o robot localizou-se, todavia, em *NYGL*, pelo que foi gerado um novo caminho $C_5 = C_4$. Desta vez o caminho executou-se, no entanto, sem problemas. Após cumprido o 3º objectivo, passou para o 4º, *NBGL*, para o qual se gerou C_6, cuja execução decorreu também sem problemas.

$$C_6 : \quad NOG_1 \rightarrow rr\ NBGR \rightarrow rr\ NBGL$$

Uma vez atingido o penúltimo objectivo, *NBGL*, foi gerado o caminho C_7, a fim de alcançar o último objectivo, *NYGR*. A execução deste último caminho decorreu de novo sem problemas.

$$C_7 : \quad NBGL \rightarrow rr\ NOG_2 \rightarrow rr\ FYG \rightarrow mfgr\ NYGR$$

Ao todo, foram atribuídos 5 objectivos e gerados 7 caminhos, sendo os 2 caminhos excedentes correspondentes a situações de excepção.

A trajectória pode ser analisada na Figura 6.9, cujas marcas têm o mesmo significado que na Figura 6.8. Para clarificação estão ainda indicados na Figura 6.9 os caminhos apresentados anteriormente, nas posturas onde foram gerados.

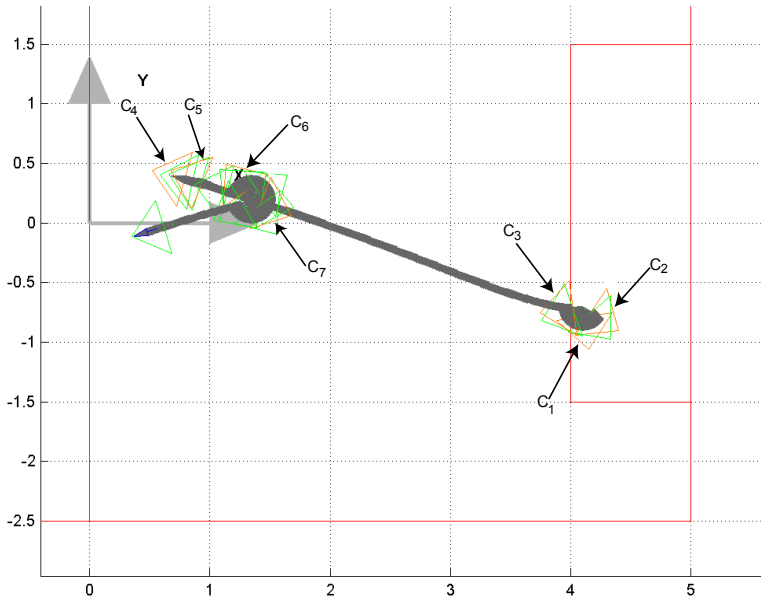


Figura 6.9 - Resultado de Navegação Topológica numa situação multi-objectivo.

Exemplo 3: No último exemplo, o robot foi colocado na postura $x = 0.5m$, $y = -2.4m$ e $\theta = 52°$, tendo-se localizado no nó *NBGR* e tendo como objectivo o nó *FYG*. Desta forma, o caminho gerado foi:

$$C_1: \quad NBGR \rightarrow rl\ NOG_1 \rightarrow rl\ FYG$$

No entanto, na execução da 1ª transição o robot localizou-se em *FBG*. Esta situação, não esperada, levou a que fosse gerado um novo caminho para o nó objectivo:

$$C_2: \quad FBG \rightarrow rr\ NOG_4 \rightarrow rr\ NYGR \rightarrow mb\ FYG$$

Na execução da 1ª transição voltou a suceder uma situação à partida não prevista e o robot voltou a localizar-se em *NBGR*, o que conduziu a uma nova geração de C_1. Assim, o robot permaneceria neste ciclo até que uma das primeiras transições, tanto em C_1 como em C_2, fosse efectuada com sucesso.

O gráfico de trajectória para esta situação específica está representado na Figura 6.10.

A fim de testar a flexibilidade do algoritmo, alterou-se a região de decisão entre os nós *perto da baliza* e os nós *longe da baliza* do metade do campo para $^1/_4$ do campo, para ambos os lados. Deste modo, os nós correspondentes a zonas perto da baliza passaram a ter regiões menores e cujos limites eram ainda bastante perto da baliza. Testou-se o comportamento do robot com este novo mapa topológico para os mesmos objectivos e postura inicial do **Exemplo 2**, obtendo-se o resultado apresentado na Figura 6.11.

Tal resultado foi bastante interessante, revelando as capacidades do método desenvolvido. Em termos de execução de caminhos, o resultado foi semelhante ao obtido no **Exemplo 2**, sendo, todavia, de apontar as seguintes diferenças:

1. O caminho C_4 foi percorrido sem falhas, tornando desnecessária a geração de C_5;

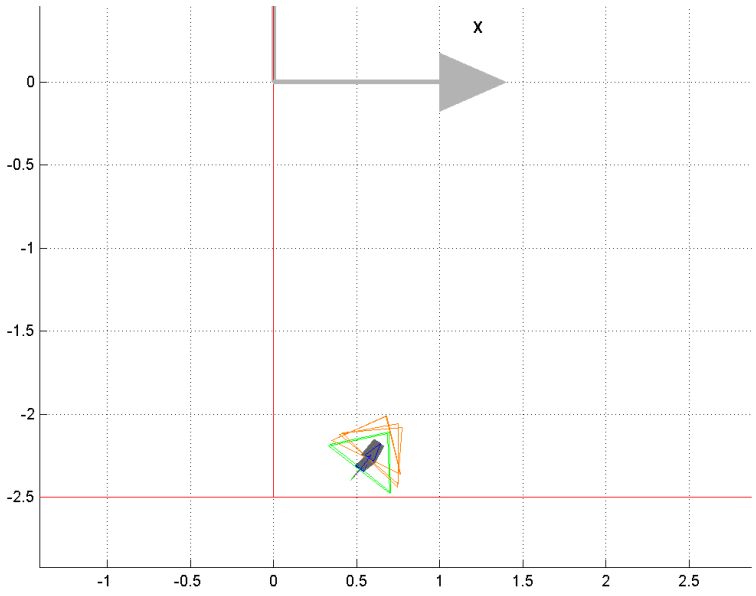


Figura 6.10 - Resultado de Navegação Topológica numa situação de *live lock*.

2. Durante a execução do caminho C_6 (neste caso foi o 5º caminho a ser executado), no decorrer da transição rr para atingir o nó $NBGR$, o robot localizou-se no nó FBG, gerando um novo caminho: $FBG \rightarrow mfgl\,NBGL$.

De referir que a segunda diferença aconteceu devido ao facto de o grafo ter sido inicialmente pensado com base na previsão de que as zonas associadas aos nós *perto da baliza* e *longe da baliza* se separariam na zona central do campo. Dado que, nesta situação, se alteraram as zonas associadas a cada nó mas não se alterou o grafo, surgiram algumas diferenças, como é exemplo a situação em que o robot se localizou num nó *perto da baliza* e depois se movimentou para o nó *longe da baliza* associado à baliza da mesma cor, ficando ainda relativamente perto dessa baliza (o limiar está no centro da metade do campo correspondente). Para, de seguida, o robot atingir um dos nós perto da baliza oposta, bastou descrever um movimento de rotação, de acordo com o grafo actual. Contudo, ao efectuar esta operação, o robot localizou-se num nó longe da baliza, o que não era expectável pela análise do grafo.

Esta situação é natural, devido à mudança das regiões associadas aos nós do grafo, pelo que se deve verificar se este continua a ser aplicável. Neste caso não houve problemas, pois o método consegue superar esta falha; mas noutros casos, todavia, as falhas nas transições definidas poderiam tornar o grafo impraticável.

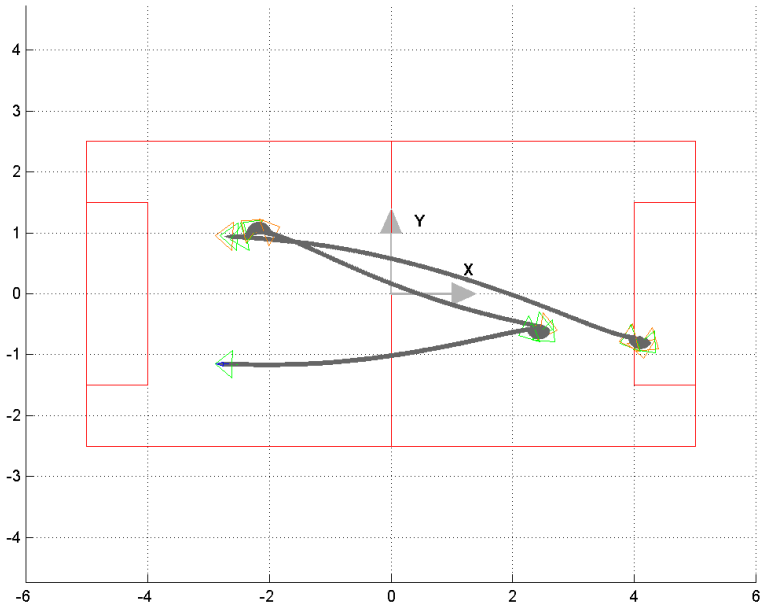


Figura 6.11 - Resultado de Navegação Topológica com um Mapa Topológico diferente.

Capítulo 7

Conclusões

Foi apresentada uma abordagem diferente à navegação num ambiente de Futebol Robótico, baseada em considerações qualitativas. O sistema implementado retém informação do ambiente na forma de um Mapa Topológico, utilizando técnicas de Análise de Componentes Principais. Para efectuar a localização no referido Mapa, implementaram-se métodos de classificação de dados, sendo o resultado destes usado para gerar caminhos no Mapa, através de algoritmos de procura em grafos. São ainda usadas características geométricas simples do mundo para executar os referidos caminhos.

A extracção de dados, efectuada sobre um conjunto de imagens de um campo de futebol robótico virtual, foi aplicada com êxito através da utilização do método de Análise de Componentes Principais, também designado por expansão de *Karhunen-Loeve*. Este tira partido do facto de as imagens capturadas num determinado ambiente apresentarem muitas características comuns, ou seja, estarem bastante correlacionadas.

Verificou-se que, após aplicação da expansão de KL a um conjunto de 528 imagens, um número bastante pequeno de imagens principais (que constituem uma base para o espaço, designado por espaço principal, no qual estão contidas as imagens utilizadas para treino) é suficiente para representar esse conjunto com uma fiabilidade adequada ao propósito deste trabalho.

Para conseguir a descrição dos locais do Mapa Topológico, particionou-se um conjunto de imagens do campo, de forma a agrupar no mesmo sub-conjunto as imagens que partilhassem certas características visuais. Na compressão dos dados de cada um destes sub-conjuntos, foram utilizados dois procedimentos diferentes. No primeiro, designado por KL, foi gerado um único espaço principal e associadas, a cada local, as projecções das imagens correspondentes sobre esse espaço; no segundo, designado por MKL, foi, por sua vez, associado um espaço principal a cada local do mapa.

Foi implementada a localização no Mapa Topológico como um processo de classificação de padrões, tendo como base as duas formas de representação do conhecimento apresentados. Para o método KL concluiu-se que a utilização dos *k-vizinhos mais próximos* com $k \neq 1$ melhorava significativamente a localização do robot. A classificação associada ao MKL não pareceu, ao contrário, melhorar significativamente o desempenho da localização. Além disso, verificou-se que este segundo método requeria um custo computacional bastante mais elevado, podendo torná-lo impraticável. De uma forma geral, os métodos de localização foram bem sucedidos na classificação de uma imagem no local esperado.

O bloco de navegação foi implementado recorrendo a algoritmos amplamente usados, nomeadamente a procura de custo uniforme, sobre um grafo relativamente simples. Desta forma, foi directa e bastante eficiente a sua integração no método. Recorreu-se ainda às características geométricas das balizas (objectos de fácil detecção no Futebol Robótico), para navegar ao longo do caminho gerado.

O método desenvolvido provou, em suma, ser aplicável ao ambiente de Futebol Robótico, no simulador desenvolvido para o efeito, revelando-se eficaz no cumprimento dos objectivos de navegação. Um dos factores de sucesso do método pode atribuir-se ao facto de as falhas na execução de transições serem, usualmente, detectadas e corrigidas prontamente.

Apesar dessas indicações favoráveis, foi detectado um problema na execução do método: este poderia entrar, por vezes, em situações de *live lock*. A introdução de um supervisor da navegação mais elaborado que o utilizado actualmente poderia, contudo, identificar e corrigir este tipo de situação. Uma abordagem possível à implementação deste supervisor seria a aplicação de um detector de *live lock*, através do estudo dos caminhos gerados ao longo do tempo. Outra abordagem, porventura mais interessante, seria a utilização de um algoritmo de aprendizagem, capaz de analisar falhas e sucessos nas transições e alterar os custos das mesmas, de forma a favorecer o uso das melhor sucedidas.

Uma das características mais interessantes do método é a possibilidade de permitir a definição de vários grafos a serem executados e, inclusive, mudá-los em tempo real. Um exemplo típico é o da implementação de grafos distintos para um defesa e um atacante. Esta situação dinâmica pode, ainda, ser bastante vantajosa quando implementada como um esquema de auxílio à cooperação no ambiente de futebol robótico. Pesa neste último caso o facto de os robots poderem dialogar numa linguagem comum, qualitativa, em que o mundo onde navegam, o mapa topológico, é flexível e pode adaptar-se às necessidades de cada situação.

A implementação do método de Navegação Topológica nos robots da equipa *ISocRob* seria o passo a tomar de seguida, tentando migrar o método desenvolvido neste projecto para a plataforma dos robots.

Para isso, seria necessário garantir que a execução dos algoritmos associados à Navegação Topológica seria possível de ser efectuada em tempo real e em paralelo com os outros módulos da arquitectura de *software* dos robots.

Um dos factores a ter em conta é o tempo necessário para a execução completa do algoritmo. Os testes feitos num *Pentium IV* mostraram que se porta de forma bastante rápida. No entanto, apesar de, nos robots reais, o desenvolvimento ser feito em linguagem *C* e não em *Matlab*, mesmo assim o método poderá não ser rápido o suficiente para correr na plataforma actual sem baixar a frequência actual de execução dos algoritmos.

A memória necessária para a execução do método equivale ao número de componentes principais mais a média das imagens de treino (cuja dimensão é idêntica à das imagens). Para o caso de uma imagem de dimensão $[240 \times 320]$, com 3 canais (*RGB*, por exemplo) e com uma compressão usando 25 imagens principais, s memória necessária é de, aproximadamente, 6Mb, um valor que para o nível de computação dos computadores actuais, é aceitável.

A equipa *IScoRob* espera actualizar o seu poder de computação nos tempos próximos, pelo que, mesmo que, durante a aplicação do método, se concluísse não haver poder de

computação suficiente, seria certo que o método poderia ser aplicado sem qualquer problema nas novas plataformas.

A título conclusivo, pode afirmar-se que o método de Navegação Topológica apresenta resultados promissores. No entanto, não devem ser colocados de parte outros métodos de navegação. A melhor prática consiste em aliar os métodos de navegação mais clássicos à Navegação Topológica, usando esta última num contexto mais global e a navegação métrica a um nível mais local. Aproveita-se assim o melhor dos dois tipos de navegação, ou seja, faz-se uma gestão racional das potencialidades específicas de cada um deles.

Bibliografia

[1] Blaer, P. e Allen, P., "Topological Mobile Robot Localization Using Fast Vision Techniques", *Proceedings of the 2002 IEEE Int. Conference on Robotics & Automation*.

[2] Cappelli, R., Mario, D. e Maltoni, D., "Multispace KL for Pattern Representation and Classification", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 23, nº 9, 2001, págs. 977–996.

[3] Gaspar, J., Winters, N. e Santos-Victor, J., "Vision-based Navigation and Environmental Representations with an Omnidirectional Camera", *IEEE Trans. on Robotics and Automation*, volume 16, nº 6.

[4] Horswill, I., "Polly: A Vision-based Artificial Agent", *Proc. Nat. Conf. Artificial Intelligence*, págs. 824–829.

[5] Ishiguro, H. e Tsuji, S., "Image-based Memory of Environment", *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, págs. 634–639.

[6] Marques, J., *Reconhecimento de Padrões: Métodos Estatísticos e Neuronais*, Ensino da Ciência e da Tecnologia, IST Press, Lisboa, 1st ed., 1999.

[7] Murakami, H. e Kumar, V., "Efficient Calculation of Primary Images from a Set of Images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 4, nº 5, 1982, págs. 511–515.

[8] Murase, H. e Nayar, S. K., "Visual Learning and Recognition of 3-D Objects from Appearence", *International Journal of Computer Vision*, volume 14, nº 1, 1995, págs. 5–24.

[9] Russel, S. e Norvig, P., *Artificial Intelligence: a Modern Approach*, Prentice Hall Series on Artificial Intelligence, Prentice-Hall, New Jersey, 1st ed., 1995.

[10] Shapiro, L. e Stockman, G., *Computer Vision*, Prentice Hall, 2001.

[11] Ventura, R., Toscano, F., Marques, C., Custódio, L. e Lima, P., "ISocRob-2000: Technical Report", Rel. téc., Instituto de Sistemas e Robótica, Lisboa-Portugal, 2000.

Anexo A

Simulador

No decorrer deste projecto foi desenvolvido um Simulador. A importância deste Simulador proveio da necessidade de:

1. Acelerar o processo de desenvolvimento através do uso de uma ferramenta rápida de produção de imagens de teste e de verificação de resultados;

2. Tornar o desenvolvimento independente dos robots, da sua estrutura física e do software. Dado o momento de adaptação que a equipa *SocRob* atravessa, evitou-se deste modo lidar com problemas de implementação durante a fase de desenvolvimento;

3. Poder obter bastantes resultados, nas mais variadas situações e de um modo bastante rápido e necessitando apenas de recorrer para o efeito a um simples computador.

O Simulador foi desenvolvido em Matlab©, usando como linguagem de modelação do mundo real *VRML* (Virtual Reality Modelling Language). A escolha recaiu sobre esta linguagem devido a vários factores:

1. É uma linguagem independente da plataforma de trabalho, podendo nomeadamente ser usada da mesma forma em Linux e em Windows©, o que, no caso presente, se torna bastante útil;

2. É uma linguagem poderosa (e no entanto simples), capaz de reproduzir bastante fielmente as condições reais do ambiente (cor, luz, material, etc.);

3. Pode ser facilmente integrada numa página interactiva usando $JAVA^{TM}$, o que se revela bastante interessante, tendo em conta o grupo de investigação onde se insere este projecto (inclusive foi feita esta integração);

4. O Matlab© possui uma *toolbox* de Realidade Virtual que, em conjunto com *VRML*, proporciona uma ferramenta de desenvolvimento e de teste com elevada capacidade e funcionalidade.

A parte do simulador em *VRML* foi desenvolvida para simular a câmara da frente dos robots. Os parâmetros usados nesse âmbito estão descritos no Capítulo A.2, descrevendo-se no Capítulo A.1 o modelo criado em Matlab para simular o robot.

A.1 Modelo do Matlab

Para poder simular o trabalho no *Matlab*© desenvolveu-se um modelo em *Simulink*©.

O modelo desenvolvido integra as funções de simulação do robot, o simulador da câmara e as funções desenvolvidas para implementar a navegação topológica. Na Figura A.1, pode visualizar-se o modelo criado e os seus blocos principais.

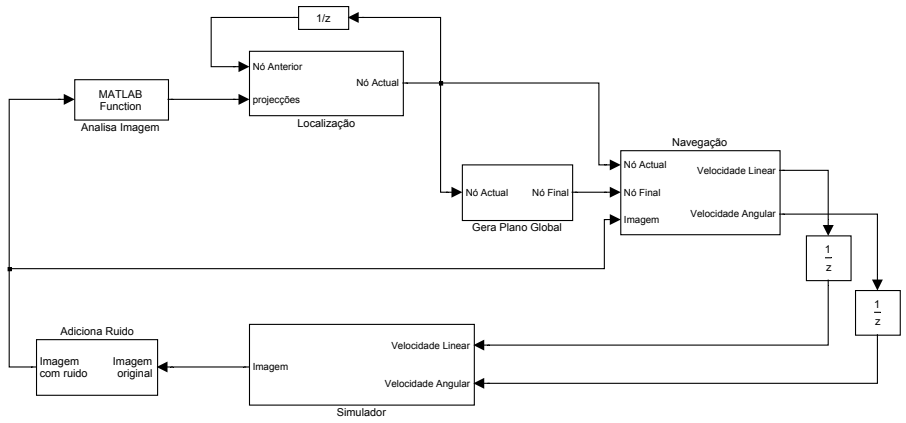


Figura A.1 - Modelo de Simulink

Descrevem-se, de seguida, as funções realizadas por cada um dos blocos que constituem o modelo:

Simulador: Este bloco simula o robot, o movimento e a câmara frontal. Recebe a velocidade linear e a velocidade angular determinados pelo algoritmo de navegação e simula cinemática do robot para determinar a sua postura. Inclui-se ainda a dinâmica dos motores (não garantindo no entanto que seja a mesma dinâmica dos motores usados nos robots reais) na simulação para obter maior realismo. Determinada a sua postura, este simula a câmara da frente e devolve a imagem do robot capturada nessa mesma postura.

Adiciona ruído: O ruído é adicionado à imagem capturada ao robot, de modo a tornar a simulação mais real. O ruído adicionado é do tipo *sal e pimenta* pois será o tipo de ruído adequado a uma imagem capturada por uma câmara digital. O nível de ruído é especificado em termos de percentagem de *pixels*.

Analisa Imagem: Este bloco projecta a imagem capturada no espaço principal, gerado *a priori*, devolvendo a projecção calculada.

Localização: A localização do robot nos locais-chave é realizada neste bloco, sendo nele que se aplicam os métodos de localização descritos ao longo do trabalho.

Navegação: Os algoritmos descritos anteriormente para efectuar a navegação do robot ao longo dos nós (logo no campo real) estão implementados neste bloco, o qual recebe o *Nó Actual* e o *Nó Final*, para gerar o caminho entre estes. Recebe ainda a imagem actual do robot para ser usada no cálculo da velocidade linear e angular durante a execução das transições, pois, como foi referido no Capítulo 5, além dos valores das velocidades associados às transições, são usadas ainda as balizas para fazer seguimento visual de caminho.

Gera Plano Global: Este bloco gere o *Nó Final* que o robot deve atingir. Pode ter qualquer função associada para esse efeito. No decurso do projecto usaram-se duas funções diferentes:

- Uma em que o utilizador fornece um vector com os nós a serem atingidos, sendo que, nesse caso, o bloco em questão altera o *Nó Final* para o seguinte, sempre que for atingido o *Nó Final* actual;
- Na outra situação é gerado um novo *Nó Final*, sempre que é atingido o *Nó Final* actual.

(Outros): Foram inseridos dois elementos de atraso (correspondentes a um atraso real) para simular o existente ao longo da malha de realimentação, inerente a qualquer sistema do género.

As simulações foram efectuadas usando um intervalo de iteração fixo e processamento sequencial.

A.2 Simulação da Câmara

Usando o VRML, criou-se um modelo $3D$ do ambiente do futebol robótico, tal como se apresenta na Figura A.2, onde se pode se ver, em perspectiva, o campo de futebol.

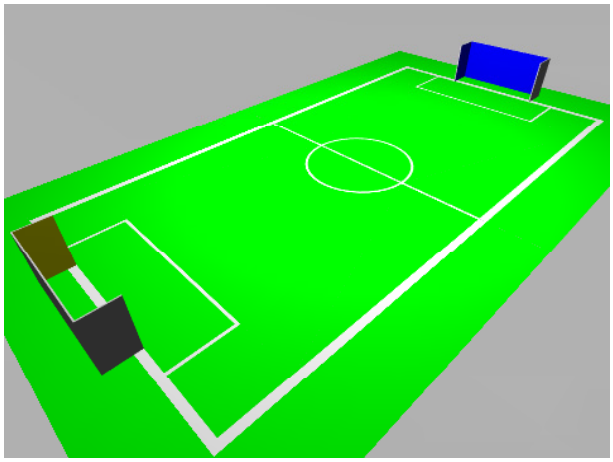


Figura A.2 - Perspectiva do modelo do campo

Nos robots existem duas câmaras, uma no topo do robot, omnidireccional, e uma outra frontal. No simulador simulou-se a câmara frontal, usando para tal as propriedades desta:

1. Posição da câmara;
2. Pitch da câmara;
3. Ângulo de visão (vertical).

Na imagem A.3 pode observar-se a relação entre os vários parâmetros da câmara, onde f é a distância focal, h é a altura da câmara, θ é o ângulo de visão da câmara, c a distância (horizontal) real do centro da imagem ao robot e b corresponde na imagem à distância no mundo $x - c$.

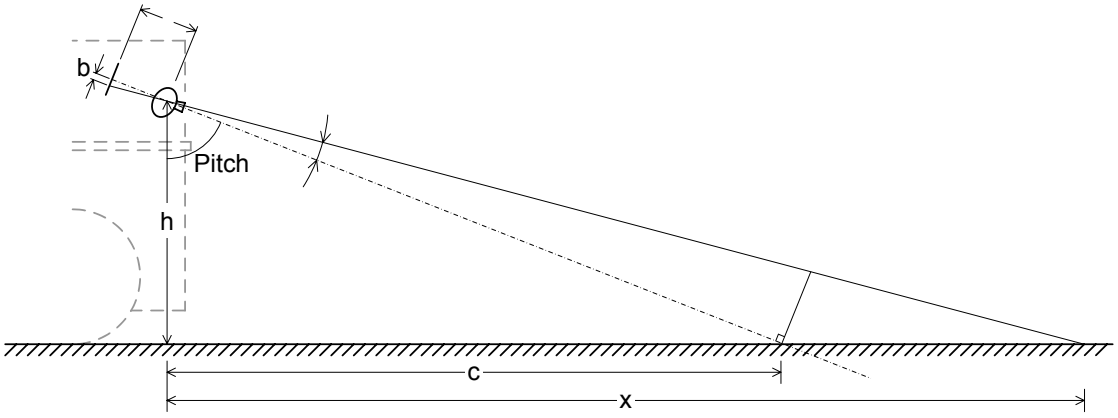


Figura A.3 - Parâmetros da Câmara

A posição da câmara é definida à partida pela sua construção e é conhecida. O *pitch* é tal que o centro da imagem corresponde a uma distância real ao robot pré-determinada. O ângulo de visão é determinado através da seguinte relação:

$$\theta = 2\tan^{-1}\frac{b}{f} \tag{A.1}$$

A distância focal (f) é calculada durante o processo de calibração (que é feito apenas uma vez e não se altera se não se alterarem os parâmetros da câmara) como indicado em [11]. O parâmetro b corresponde à altura do *CCD*. Assim, os valores obtidos foram:

- Posição da Câmara = $\left[\begin{array}{ccc} 0.200 & 0 & 0.243 \end{array}\right] cm$
- Pitch = $82.0°$
- Ângulo de Visão = $54.7°$

Dado que o trabalho recorre a imagens segmentadas (embora também possa usar imagens não segmentadas), o principal objectivo do simulador para o trabalho é produzir imagens segmentadas.

Em condições normais, a segmentação não tem, obviamente, a qualidade apresentada, pelo que o algoritmo teve de ser desenvolvido de modo a ser robusto a essa alterações, aspecto para que o simulador também contribuiu, pois pode-se facilmente introduzir vários tipos de ruído nas imagens geradas e testar o algoritmo.

Anexo B

Ficheiro com Descrição do Grafo

```
Nodes:
Nodes   | Goal color | Goal position | Description
-------------------------------------------------------------------------------
1.  NBGL      blue           0.25        near the blue goal, left from the goal
2.  NBGR      blue           0.75        near the blue goal, right from the goal
3.  FBG       blue           0.5         far from the blue goal
4.  NYGL     yellow          0.25        near the yellow goal, left from the goal
5.  NYGR     yellow          0.75        near the yellow goal, right from the goal
6.  FYG      yellow          0.5         far from the yellow goal, right from the goal
7.  NOG1      none           0.0         no goal (1)
8.  NOG2      none           0.0         no goal (2)
9.  NOG3      none           0.0         no goal (3)
10. NOG4      none           0.0         no goal (4)


Transitions:
Transition | Cost | Linear speed [m/s] | Angular speed [rad/s] | Description
-------------------------------------------------------------------------------------------------------
 1. rr         1          0.0                  -0.7             rotate right
 2. rl         1          0.0                   0.7             rotate left
 3. mfgr       1          0.5                   0.0             move forward using the goal (on the right)
 4. mfgl       1          0.5                   0.0             move forward using the goal (on the left)
 5. mb         1         -0.5                   0.0             move backward using the goal


Graph:
Origin | Transition | Destination
---------------------------------
NBGL:
            rr          NOG2
            rl          NBGR
            mb          FBG

NBGR:
            rr          NBGL
            rl          NOG1
            mb          FBG

FBG:
            rr          NOG4
            rl          NOG3
            mfgr        NBGR
            mfgl        NBGL

NYGL:
            rr          NOG3
            rl          NYGR
            mb          FYG

NYGR:
            rr          NYGL
            rl          NOG4
            mb          FYG

FYG:
            rr          NOG1
            rl          NOG2
            mfgr        NYGR
            mfgl        NYGL

NOG1:
            rr          NBGR
            rl          FYG

NOG2:
            rr          FYG
            rl          NBGL

NOG3:
            rr          FBG
            rl          NYGL

NOG4:
            rr          NYGR
            rl          FBG
```

Figura B.1 - Descrição do grafo usado

Anexo C

Análise Comparativa dos Métodos de Localização (extensão)

Para uma melhor leitura e compreensão do relatório, remeteu-se parte dos resultados das comparações dos métodos de localização para este anexo.

As Figuras C.2 a C.22 são o resultado da localização das posturas referidas na Tabela C.1. A Figura C.1 indica a leitura de um dos gráficos, onde os triângulos representam as orientações do robot. De notar que, sendo uma das varáveis um ângulo, a superfície que melhor representaria o gráfico seria um cilindro e não um plano. De facto, é notória a continuidade entre os nós (definidos por cores) em 0° e em 360°.

	X	**Y**	Θ
Situação 1	Fixo, $x = 3m$	Variável, $\Delta y = 0.2m$	Variável, $\Delta\theta = 6°$
Situação 2	Variável, $\Delta x = 0.2m$	Fixo, $y = 0m$	Variável, $\Delta\theta = 6°$
Situação 3	Variável, $\Delta x = 0.2m$	Variável, $\Delta y = 0.2m$	Fixo, $\theta = 138°$

Tabela C.1 - Posturas usadas para testar a localização.

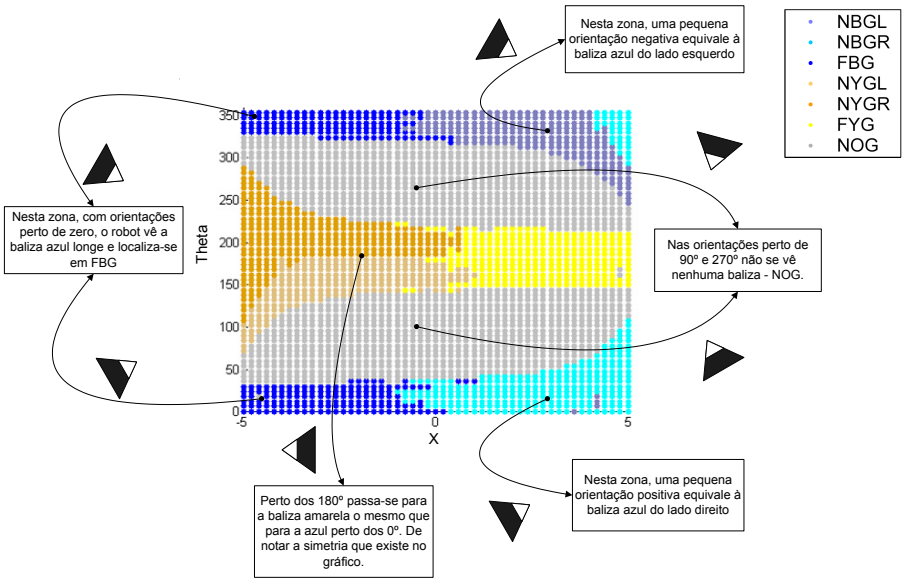


Figura C.1 - Leitura de um gráfico de localização.

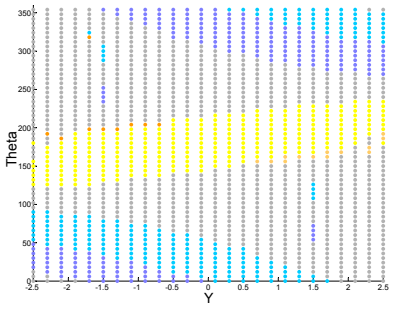


Figura C.2 - KL, eucl. e $k = 1$ para a Situação 1.

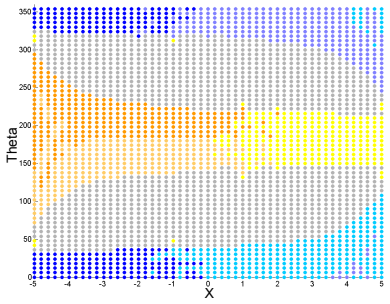


Figura C.3 - KL, eucl. e $k = 1$ para a Situação 2.

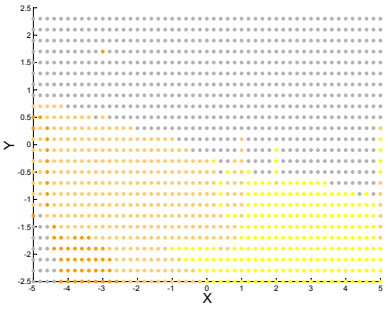


Figura C.4 - KL, eucl. e $k = 1$ para a Situação 3.

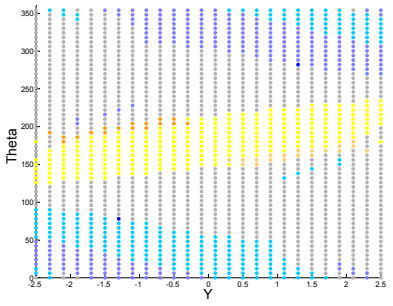


Figura C.5 - KL, pond. e $k = 1$ para a Situação 1.

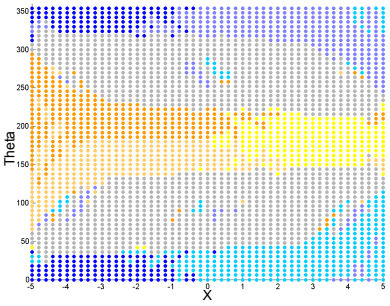


Figura C.6 - KL, pond. e $k = 1$ para a Situação 2.

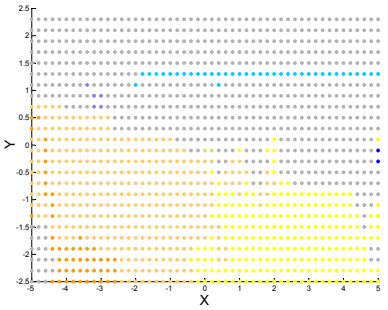


Figura C.7 - KL, pond. e $k = 1$ para a Situação 3.

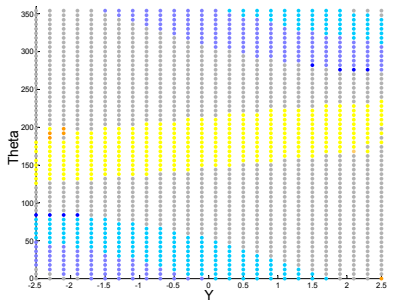


Figura C.8 - KL, eucl. e $k = 5$ para a Situação 1.

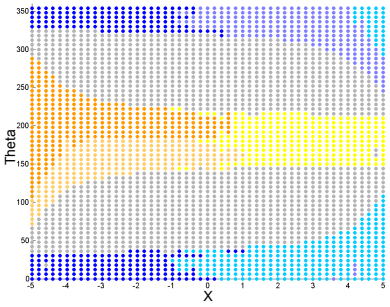


Figura C.9 - KL, eucl. e $k = 5$ para a Situação 2.

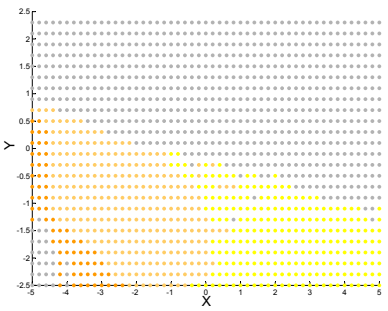


Figura C.10 - KL, eucl. e $k = 5$ para a Situação 3.

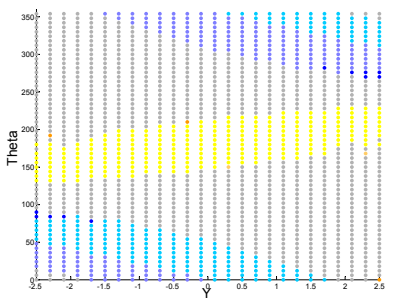


Figura C.11 - KL, pond. e $k = 5$ para a Situação 1.

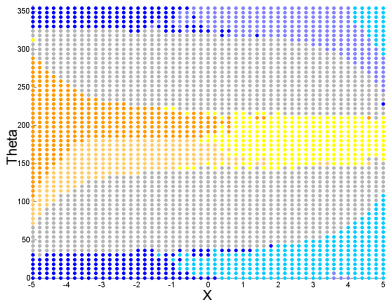


Figura C.12 - KL, pond. e $k = 5$ para a Situação 2.

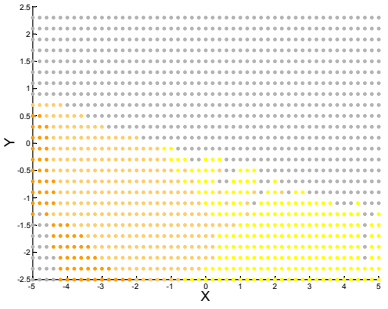


Figura C.13 - KL, pond. e $k = 5$ para a Situação 3.

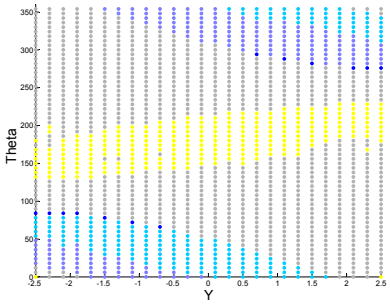


Figura C.14 - *KL*, eucl. e $k = 10$ para a Situação 1.

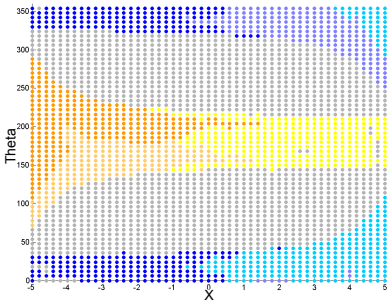


Figura C.15 - *KL*, eucl. e $k = 10$ para a Situação 2.

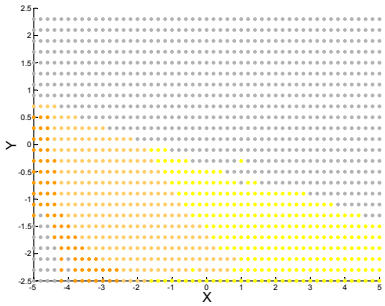


Figura C.16 - *KL*, eucl. e $k = 10$ para a Situação 3.

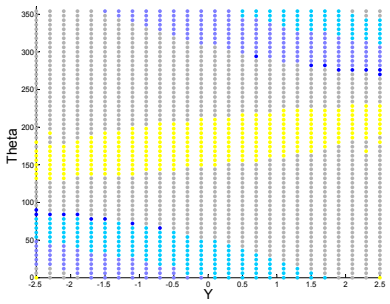


Figura C.17 - *KL*, pond. e $k = 10$ para a Situação 1.

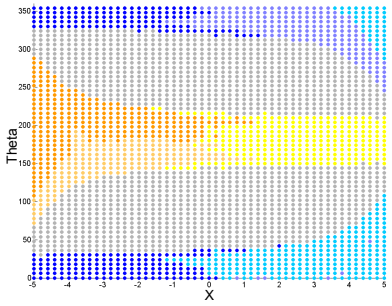


Figura C.18 - *KL*, pond. e $k = 10$ para a Situação 2.

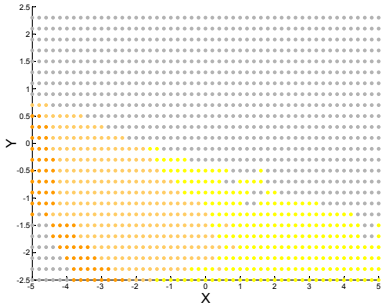


Figura C.19 - *KL*, pond. e $k = 10$ para a Situação 3.

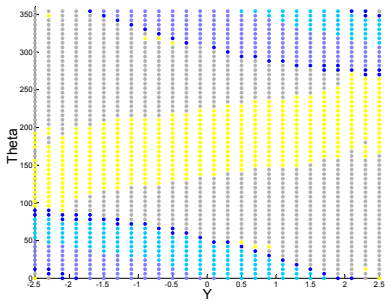


Figura C.20 - *MKL* para a Situação 1.

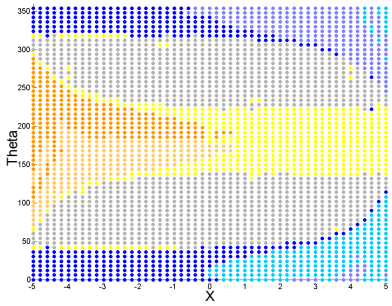


Figura C.21 - *MKL* para a Situação 2.

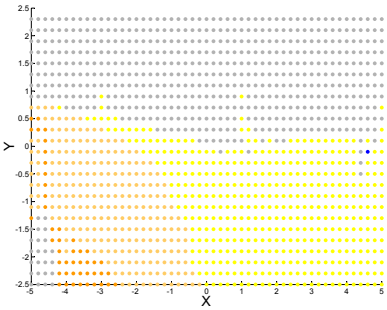


Figura C.22 - *MKL* para a Situação 3.

Anexo D

Fluxograma da Navegação Topológica para Futebol Robótico

Neste anexo apresenta-se um fluxograma descritivo dos vários componentes da navegação topológica, segundo a aplicação feita.

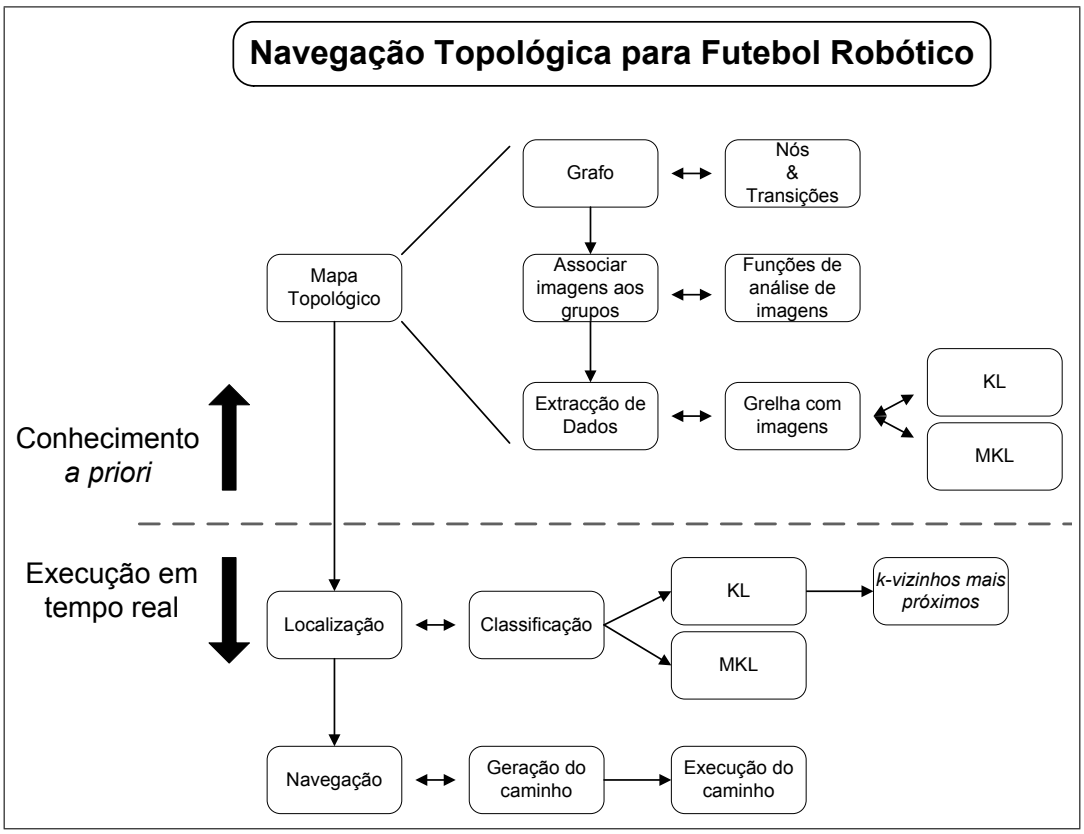


Figura D.1 - Fluxograma descritivo dos vários componentes da Navegação Topológica aplicada ao Futebol Robótico

Torna-se assim simples de ter uma ideia global do método e do modo como se encaixam as várias componentes.