
**UNIVERSIDADE TÉCNICA DE LISBOA**

**INSTITUTO SUPERIOR TÉCNICO**

# CONTROLO DE FORMAÇÕES DE ROBOTS MOVIMENTANDO-SE NUM PLANO

Paulo Sérgio Rodrigues Gomes nº 46638

LICENCIATURA EM ENGENHARIA ELECTROTÉCNICA
E DE COMPUTADORES
Relatório de Trabalho Final de Curso
163/2002/L

Prof. Orientador: Pedro Lima
Prof. Acompanhante: Luís Custódio

Setembro de 2004

# Resumo

Neste trabalho são estudados e implementados algoritmos de controlo de formação de uma equipa de 3 robots não-holonómicos, movimentando-se numa superfície plana. O objectivo destes algoritmos é manter uma formação estável num conjunto de robots, ao mesmo tempo que cada robot evita quaisquer obstáculos que detecte. Os algoritmos estudados foram originalmente desenvolvidos pelo laboratório GRASP, da Universidade da Pennsylvania, e foram testadas várias modificações aos métodos originais. O processo de simulação foi efectuado em ambiente *Matlab*, seguido de implementação em robots reais, pertencentes ao projecto Socrob. Para a implementação dos algoritmos foram utilizados processos de localização dos robots e dos obstáculos (odometria, visão, sonares e comunicação entre os robots por *wireless Ethernet*). Foi também estudado um protocolo de coordenação, que selecciona uma dada formação de acordo com o estado do mundo em cada momento. Os resultados do trabalho consistem em simulações bem como em experiências efectuadas numa equipa de 3 robots reais.

**Palavras-chave**: controlo de formações, robots móveis não-holonómicos, cooperação entre robots móveis, localização de robots.

# Índice

Resumo ..... ii
Índice ..... iii
Lista de Figuras ..... iv
Introdução ..... 1
1. Estrutura do Controlo ..... 4
2. Controlo de formações ..... 6
  2.1 Introdução ..... 6
  2.2 Controlador $l-\psi$ (Separation-Bearing Control) ..... 7
  2.3 Controlador $l-l$ (Separation-Separation Control) ..... 10
  2.4 Controlador $l-\delta$ (Separation Distance-To-Object Control) ..... 13
  2.5 Controlador $\psi-\psi$ (Dilation Control) ..... 16
3. Protocolo de Coordenação de Formações ..... 18
  3.1 Protocolo de Coordenação na ausência de obstáculos ..... 19
  3.2 Protocolo de Coordenação na presença de obstáculos ..... 22
4. Simulação ..... 24
  4.1 Simulação dos controladores isoladamente ..... 26
  4.2 Simulação do protocolo de coordenação de formação ..... 32
5. Implementação ..... 37
  5.1 Hardware/Software ..... 37
  5.2 Localização dos robots ..... 38
  5.3 Experiências e resultados ..... 41
6. Conclusão ..... 46
Referências ..... 48
Anexo A – Actuações nos motores ..... 49
Anexo B – Utilização do Simulador ..... 50
Anexo C – Funcionamento do plugin follow ..... 54

# Lista de Figuras


Figura 1 – Estrutura do controlo de formações ..... 4
Figura 2 – Problema do controlo de formações ..... 6
Figura 3 – Controlador $l-\psi$ ..... 7
Figura 4 – Controlador $l-l$ ..... 10
Figura 5 – Restrições do controlador $l-l$ ..... 11
Figura 6 – Duas soluções para o controlador $l-l$ ..... 11
Figura 7 – Controlador $l-\delta$ ..... 13
Figura 8 – Escolha do controlador (sem obstáculos) ..... 19
Figura 9 – Exemplo do protocolo de coordenação sem obstáculos ..... 20
Figura 10 – Exemplo do protocolo de coordenação com obstáculos ..... 23
Figura 11 – Bloco Simulink do Robot ..... 24
Figura 12 – Bloco Simulink do obstáculo ..... 24
Figura 13 – Simulação 1 ($SB_{12}C$: $l_{12}^d$=3 , $\psi_{12}^d$ =210º ; $S_{13}S_{23}C$: $l_{13}^d$ =3 ; $l_{23}^d$ =4 ) ..... 26
Figura 14 – Simulação 2 ($SB_{12}C$: $l_{12}^d$ =3 , $\psi_{12}^d$ =210º ; $S_{13}S_{23}C$: $l_{13}^d$ =3 ; $l_{23}^d$ =4 ) ..... 27
Figura 15 – Simulação 3 ($SB_{12}C$: $l_{12}^d$ =3 , $\psi_{12}^d$ =210º ; $S_{13}S_{23}C$: $l_{13}^d$ =3 ; $l_{23}^d$ =4) ..... 27
Figura 16 – Simulação 4 – ($SB_{12}C$: $l_{12}^d$ =3 , $\psi_{12}^d$ =210º ; $S_{13}S_{23}C$: $l_{13}^d$ =3 ; $l_{23}^d$ =4) ..... 28
Figura 17 – Simulação 5 - ($SB_{12}C$: $l_{12}^d$ =3 , $\psi_{12}^d$ =210º ; $S_{13}S_{23}C$: $l_{13}^d$ =3 ; $l_{23}^d$ =4) ..... 29
Figura 18 – Simulação 6 - ($SB_{12}C$: $l_{12}^d$ =4 , $\psi_{12}^d$ =210º ; $S_{13}S_{23}C$: $l_{13}^d$ =3 ; $l_{23}^d$ =4) ..... 29
Figura 19 – Simulação 6 - Controlo de $R_2$ ..... 30
Figura 20 – Simulação 6 - Controlo de $R_3$ ..... 30
Figura 21 – Simulação 7 ($SB_{32}C$: $l_{32}^d$ =2 , $\psi_{32}^d$ =150º ; $SB_{13}C$: $l_{13}^d$ =3 ; $\psi_{13}^d$ =150º ) 31
Figura 22 – Simulação 8 ($SB_{32}C$: $l_{32}^d$ =2 , $\psi_{32}^d$ =90º ; $SB_{13}C$: $l_{13}^d$ =3 ; $\psi_{13}^d$ =90º ) ..... 31
Figura 23 – Simulação 9 ($SB_{12}C/S_0DC$: $l_{12}^d$ =3, $\psi_{12}^d$ =210º, $\delta_0$ =1) ..... 32
Figura 24 – Simulação 9 - Evolução de $L_{12}$ e de $\delta$ ..... 33
Figura 25 – Simulação 10 - ($SB_{12}C/S_0DC$: $l_{12}^d$ =3, $\psi_{12}^d$ =210º, $\delta_0$ =1) ..... 33
Figura 26 – Simulação 11 - ($SB_{12}C/S_0DC$: $l_{12}^d$ =3, $\psi_{12}^d$ =210º, $\delta_0$ =1) ..... 34
Figura 27 – Simulação 12 ($SB_{12}C/S_0DC$: $l_{12}^d$ =3, $\psi_{12}^d$ =210º, $\delta_0$ =1) ..... 34
Figura 28 – Simulação 12 - Evolução de $L_{12}$ e de $\delta$ ..... 35
Figura 29 – Simulação 13 - ($SB_{12}C/S_0DC$: $l_{12}^d$ =3, $\psi_{12}^d$ =270º, $\delta_0$ =1.5; ..... 36
Figura 30 – Exemplo da correcção de posição ..... 40
Figura 31 – Teste 1 – $SB_{ij}C$: $l_{ij}^d$ =1, $\psi_{ij}^d$ = - 90º ..... 41
Figura 32 – Teste 1 – $SB_{ij}C$: $l_{ij}^d$ =1 , $\psi_{ij}^d$ = - 90º; Evolução da distância entre os robots $l_{ij}$. 42
Figura 33 – Teste 1 – $SB_{ij}C$: $l_{ij}^d$ =1 , $\psi_{ij}^d$ = - 90º; Evolução do ângulo $\psi_{ij}$ ..... 42
Figura 34 – Teste 2 – $SB_{ij}C$: $l_{ij}^d$ =1 , $\psi_{ij}^d$ = - 90º; Evolução do ângulo $\psi_{ij}$ ..... 43

Figura 35 – Teste 2 – $SB_{ij}C$: $l_{ij}^{d}$ =1 , $\psi_{ij}^{d}$ =-90º; Evolução do ângulo $\psi_{ij}$ ..... 43
Figura 36 – Teste 3 – ($SB_{12}C$: $l_{12}^{d}$ =1.5 , $\psi_{12}^{d}$ =150º ; $S_{13}S_{23}C$: $l_{13}^{d}$ =1.5 ; $l_{23}^{d}$ =1.5 ) .... 44
Figura 37 – Teste 3 – ($SB_{12}C$: $l_{12}^{d}$ =1.5 , $\psi_{12}^{d}$ =150º ; $S_{13}S_{23}C$: $l_{13}^{d}$ =1.5 ; $l_{23}^{d}$ =1.5 ) – Evolução da distância entre $R_1$ e $R_3$ ..... 45
Figura 38 – Teste 3 – ($SB_{12}C$: $l_{12}^{d}$ =1.5 , $\psi_{12}^{d}$ =150º ; $S_{13}S_{23}C$: $l_{13}^{d}$ =1.5 ; $l_{23}^{d}$ =1.5 ) – Evolução da distância entre $R_2$ e $R_3$ ..... 45
Figura 39 – Menu do Simulador ..... 50
Figura 40 – Editor de trajectória do líder e dos obstáculos ..... 51

# Introdução

Com o aumento de popularidade dos sistemas de robótica móvel, tornou-se claro que existem muitas tarefas difíceis, e por vezes impossíveis, de concretizar utilizando apenas um robot. A ideia de que vários robots teriam um desempenho superior ao efectuar essas tarefas é actualmente aceite, o que conduziu a que, nos últimos anos se tenha assistido a um desenvolvimento e aumento de interesse nessa área.

Chegou-se à conclusão que existem vantagens em utilizar vários robots a trabalhar em conjunto como equipa, em oposição a ter apenas um. O conceito de ter vários robots mais simples e baratos a desempenharem a mesma tarefa em vez de apenas um mais dispendioso e complexo tornou-se atractivo para os investigadores.

As aplicações da cooperação entre robots incluem planeamento, navegação e manipulação cooperativa, exploração e mapeamento, bem como o controlo de formações. Em algumas situações, para uma equipa de robots poder trabalhar em conjunto, é necessário que actuem segundo uma determinada formação. Um caso típico é o do transporte de um objecto por uma equipa de 3 robots. Os robots têm de manter uma formação rígida em cada momento, para que o objecto fique estável e o transporte seja bem sucedido. Entre as topologias mais comuns estão a formação em linha recta, em triângulo, *follow-the-leader*, com inúmeras variações, alterando certos parâmetros.

O objectivo deste trabalho é estudar e implementar algoritmos de controlo de formações para três robots não holonómicos movimentando-se numa superfície plana. Tal objectivo pressupõe várias etapas: observação do mundo e auto-localização dos robots, escolha da formação adequada através de um protocolo de coordenação, cálculo das actuações dos motores de acordo com a formação, possibilidade de alterar a formação em tempo real e possibilidade de evitar obstáculos.

Para manter uma formação é necessário estar implementado um controlador que receba dados acerca do mundo, e calcule as velocidades que o robot deve manter. Como existem várias formações diferentes, também existem vários controladores. Este trabalho contempla os controladores estudados pelo Laboratório GRASP da Universidade da Pennsylvania, E.U.A. [8], como por exemplo em [1], [2] e [3]. Estes algoritmos são utilizados pelos membros deste laboratório como a base do seu projecto, construindo sobre ela métodos mais complexos, que contemplam, por exemplo, geradores de trajectória envolvendo campos de potencial [1], controlo de formações utilizando apenas a visão [2], localização cooperativa e transporte cooperativo de objectos [3], grafos de formações [5], etc.

As entradas do controlador que mantém a formação revelam o estado do mundo à volta do robot seguidor. Os sensores têm um papel muito importante nesta área: pode-se utilizar um sistema de visão para cada robot observar o ambiente e obter as posições e orientações dos outros robots; pode haver um sistema de comunicação entre a equipa onde são trocadas informações relevantes; os robots podem estar equipados com sonares ou lasers para detectarem obstáculos ou outros robots; pode haver um sistema de odometria para determinar a posição do robot, etc. Os métodos estudados têm a particularidade de serem extensíveis a qualquer quantidade de robots sem adição de peso de computação, dado que cada robot apenas necessita de um número limitado de informação para se localizar e manter a formação. Este trabalho contempla a implementação numa equipa com apenas três robots.

Em termos de observação de estado, pode-se considerar centralizada ou descentralizada, ao referir a forma como cada robot observa o mundo. Utilizando observação descentralizada, cada robot está isolado do mundo, só tendo acesso à informação necessária aos controladores através dos sensores. Com estes, ele pode saber onde está o resto da equipa e identificar obstáculos, mas não pode determinar a velocidade ou orientação dos outros robots. É então necessário utilizar estimadores para obter esta informação. Na observação do mundo centralizada cada robot tem a informação do estado completa, nomeadamente através de comunicação, não sendo necessário utilizar estimadores de posição ou velocidade dos outros robots. Este sistema de observação é mais robusto que o descentralizado por se dispor de dados com menos incerteza, tendo por outro lado como custo a utilização da comunicação entre a equipa, na medida em que no método centralizado não é necessário cada robot determinar a posição da equipa, bastando-lhe saber a sua própria postura, para poder comunicá-lo. Neste trabalho, e por já estar implementado um sistema de comunicação entre a equipa de robots, é utilizada a observação centralizada, sendo ainda utilizada a odometria para a auto-localização de cada robot. Além disso, é utilizada a visão e os sonares para identificar obstáculos, e para corrigir eventuais erros de odometria.

Conhecendo o robot o estado do mundo, é necessário escolher uma formação. O protocolo de coordenação de formações consiste num algoritmo que escolhe um controlador (e respectivas referências) de acordo com o estado do mundo: a presença ou não de obstáculos, a presença próxima de potenciais líderes, distâncias relativas entre a equipa, etc.

Neste trabalho são utilizados robots não-holonómicos de duas rodas, com duas entradas independentes (velocidade linear e angular), e cuja cinemática é descrita por um modelo simples. A primeira parte do trabalho refere-se ao desenvolvimento de algoritmos em ambiente de simulação, e a segunda parte à implementação dos mesmos algoritmos em robots reais.

Este relatório está organizado da seguinte forma: no primeiro capítulo é descrita a estrutura de controlo de formações a implementar numa equipa de robots reais; de seguida, no segundo capítulo, são estudados os vários tipos de controlo de formação; no terceiro é discutido o protocolo de coordenação para a escolha de formações; no quarto capítulo é referido o processo de simulação e são analisados os resultados de vários testes; no capítulo 5 é descrita a implementação dos algoritmos em robots reais e apresentados respectivos resultados; o sexto capítulo é dedicado às principais conclusões do trabalho.

# 1. Estrutura do Controlo

Neste capítulo é descrita a estrutura geral de controlo de formações, a forma como os controladores a serem desenvolvidos interagem com o sistema formado pelos robots/mundo.

Existem duas possibilidades principais na forma como um robot pode obter informação sobre o resto da formação: utilizando os sensores (câmaras omnidirecionais, etc.) ou comunicando através de uma rede *wireless* (WLAN) à qual todos os robots pertençam. No entanto, continuam a ser necessários os sensores para obter informação sobre o mundo (obstáculos). Dado que a informação necessária para controlar um robot seguidor inclui a posição, orientação e velocidade linear e angular do líder (ou líderes), informação essa que precisaria de ser processada através dos dados fornecidos pelos sensores, vemos que uma abordagem utilizando uma WLAN seria mais leve em termos de processamento, facto que não pode ser desprezado, já que tratamos de controlo em tempo real. Por outro lado, utilizar a observação do estado centralizada pode ser negativo, pois um erro de leitura num sensor é propagado pela rede, já que o comportamento de cada robot seguidor está dependente de dados referentes ao líder. Tendo isso em consideração, verifica-se que um bom compromisso seria utilizar a comunicação por WLAN, e periodicamente corrigir os valores fazendo uma observação do mundo e obtendo toda a informação necessária. Este algoritmo de localização é abordado mais à frente.

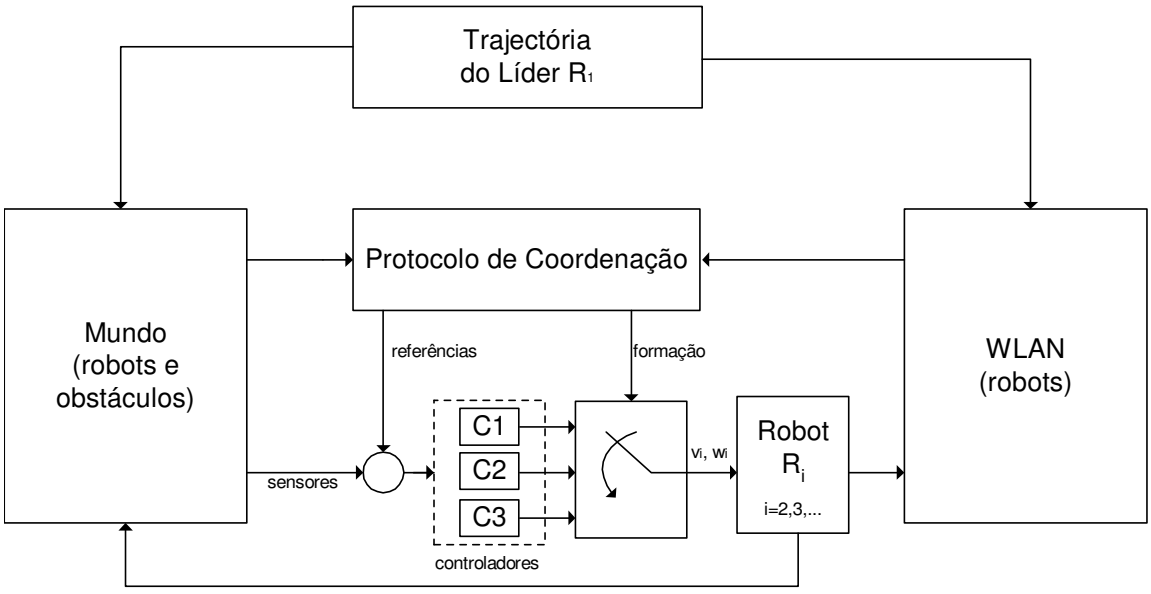


**Figura 1 – Estrutura do controlo de formações**

Na Figura 1 podemos ver uma estrutura possível para o controlo de formações. $R_1$ é o líder da formação, seguindo uma trajectória especificada exteriormente. O protocolo de coordenação determina em cada momento qual a formação a seguir pelo resto da equipa (robots $R_i$, i=2,3,...). Esta decisão vai depender da informação retirada do mundo (obstáculos) e da WLAN (configurações de todos os robots). O protocolo decide não só o tipo de controlo para cada robot, mas também os parâmetros de cada controlador. Este assunto será analisado mais profundamente no capítulo 3. Numa primeira fase de simulação, considera-se que os sensores têm um alcance e precisão suficientes para que cada robot consiga observar todo o mundo e sem erro. Numa segunda fase será incluído o alcance dos sensores e considerado o erro daí proveniente. Assim, na prática, cada robot escolhe um conjunto de leis de controlo que descrevem a sua interacção com outros robots e com obstáculos.

# 2. Controlo de formações

## *2.1 Introdução*

Numa primeira abordagem, considera-se um sistema com apenas 2 robots, em que um é o líder ($R_1$) e o outro, o seguidor ($R_2$) . O líder segue uma trajectória definida exteriormente, e o seguidor mantém uma posição relativa ao primeiro. Na Figura 2 pode-se observar o problema geral do controlo de formações:

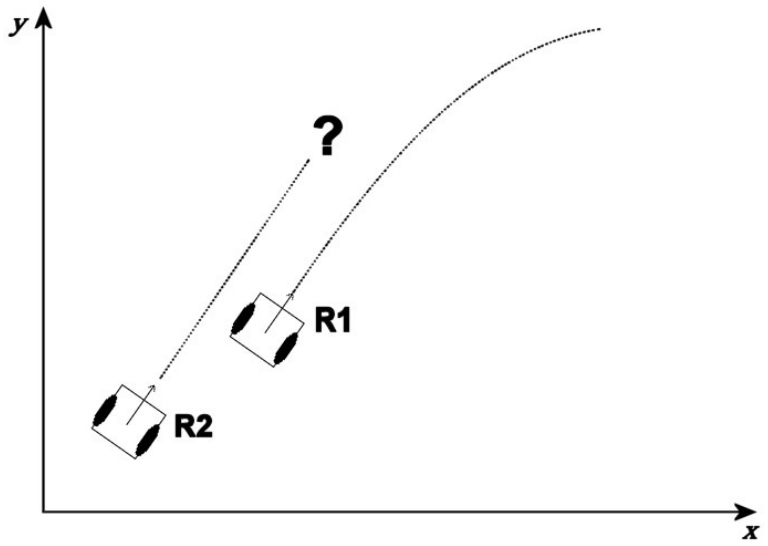


**Figura 2 – Problema do controlo de formações**

Problema: estando definida a trajectória do robot líder e a posição relativa de $R_2$ em relação a $R_1$, como determinar a trajectória de $R_2$ de modo a manter a formação?

Algumas das abordagens mais utilizadas para tentar resolver este problema foram propostas pelo laboratório GRASP da universidade da Pennsylvania, que apresenta vários trabalhos nesta área. Os controladores apresentados de seguida foram estudados por este laboratório.

Consideram-se robots não holonómicos com duas entradas independentes (velocidade linear $v_i$ e velocidade angular $\omega_i$ ), com o seguinte modelo da cinemática:

$$\dot{x}_i = v_i \cos\theta_i \qquad \dot{y}_i = v_i \sin\theta_i \qquad \dot{\theta}_i = \omega_i \tag{2.1}$$

## *2.2 Controlador* $l-\psi$ *(*Separation-Bearing Control*)*

Este controlador é chamado na literatura inglesa por *Separation-Bearing Control* ($SB_{ij}C$), cuja tradução possível seria Controlador Separação-Ângulo, pois consiste em manter controlada a distância e o ângulo que um robot faz em relação ao outro. Este primeiro controlador é do tipo seguimento de líder, em que as duas variáveis que um robot não-holonómico consegue controlar são relativas ao robot líder.

Na Figura 3 vêem-se dois robots não holonómicos (tracção diferencial), em que $R_1$ é o líder e $R_2$ é o seguidor, cada um com as configurações $(x_i,y_i,\theta_i)$ e com velocidades $v_i$ e $\omega_i$. Com este controlador consegue-se que $R_2$ siga $R_1$ com uma separação $l_{12}$ (distância medida entre a frente de $R_2$ e o centro do eixo das rodas de $R_1$) e com um ângulo $\psi_{12}$ entre $\theta_1$ e o segmento de recta cujo comprimento define $l_{12}$ . A distância entre o centro do eixo das rodas e a parte da frente do robot $R_i$ é dada por $d_i$.

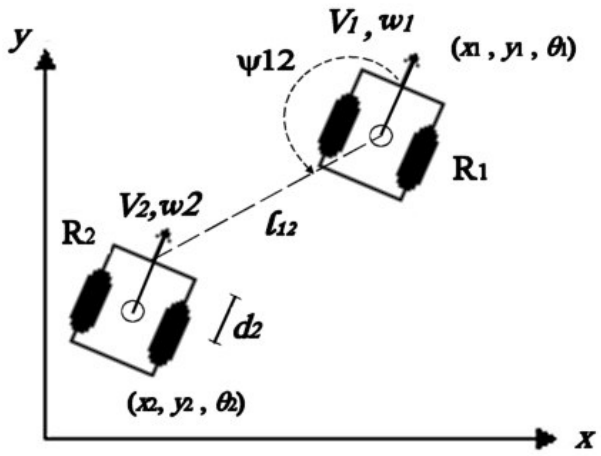


**Figura 3 – Controlador** $l-\psi$

Variando as referências de $l_{12}$ e de $\psi_{12}$, pode-se alterar a formação: para $\psi_{12}$=90º os robots andam lado a lado; para $\psi_{12}$=180º, formam uma fila; etc.

O sistema dos dois robots sofre uma mudança de coordenadas, onde as entradas são as velocidades dos dois robots, e as saídas são os valores de $l_{12}$ e de $\psi_{12}$. Assim, as equações da cinemática são dadas por:

$$\dot{z}_j = G(z_j)u_j + F(z_j,u_i) \qquad ,\dot{\theta}_j = w_j \tag{2.2}$$

onde $z_j = \begin{bmatrix} l_{ij} & \psi_{ij} \end{bmatrix}^T$ é a saída do sistema e $u_i = \begin{bmatrix} v_i & \omega_i \end{bmatrix}^T$ é a entrada para $R_i$ e $u_j = \begin{bmatrix} v_j & \omega_j \end{bmatrix}^T$ para $R_j$, e:

$$G = \begin{bmatrix} \cos\gamma_{ij} & d_j \sin\gamma_{ij} \\ \dfrac{-\sin\gamma_{ij}}{l_{ij}} & \dfrac{d_j \cos\gamma_{ij}}{l_{ij}} \end{bmatrix}$$

$$F = \begin{bmatrix} -v_i \cos\psi_{ij} \\ \dfrac{v_i \sin\psi_{ij}}{l_{ij}} - \omega_i \end{bmatrix}$$

$$\gamma_{ij} = \theta_i + \psi_{ij} - \theta_j$$

Manipulando (2.2), obtém-se:

$$u_j = G^{-1}(z_j)\left(\dot{z}_j - F(z_j, u_i)\right) = G^{-1}\left(\dot{z}_j - F\right) \qquad (2.3)$$

Fazendo

$$\dot{l}_{ij} = k_1(l_{ij}{}^d - l_{ij}), \quad \dot{\psi}_{ij} = k_2(\psi_{ij}{}^d - \psi_{ij}), \quad \dot{\theta}_j = \omega_j$$

$$p = \begin{bmatrix} k_1(l_{ij}{}^d - l_{ij}) \\ k_2(\psi_{ij}{}^d - \psi_{ij}) \end{bmatrix} = \begin{bmatrix} \dot{l}_{ij} \\ \dot{\psi}_{ij} \end{bmatrix} = \dot{z}_j \qquad \text{com } k_i > 0$$

obtém-se que:

$$u_j = G^{-1}(p - F) \qquad (2.4)$$

Calculando a matriz inversa de G:

$$G^{-1} = \begin{bmatrix} \cos\gamma_{ij} & -l_{ij}\sin\gamma_{ij} \\ \dfrac{\sin\gamma_{ij}}{d_j} & \dfrac{l_{ij}\cos\gamma_{ij}}{d_j} \end{bmatrix}$$

e seja:

$$s_{ij} = k_1(l_{ij}{}^d - l_{ij}) \qquad p = \begin{bmatrix} s_{ij} \\ b_{ij} \end{bmatrix} \qquad (2.5)$$

$$b_{ij} = k_2(\psi_{ij}{}^d - \psi_{ij})$$

Substituindo em (2.4):

$$u_j = \begin{bmatrix} \cos\gamma_{ij} & -l_{ij}\sin\gamma_{ij} \\ \dfrac{\sin\gamma_{ij}}{d_j} & \dfrac{l_{ij}\cos\gamma_{ij}}{d_j} \end{bmatrix}\left(\begin{bmatrix} s_{ij} \\ b_{ij} \end{bmatrix} - \begin{bmatrix} -v_i\cos\psi_{ij} \\ \dfrac{v_i\sin\psi_{ij}}{l_{ij}} - \omega_i \end{bmatrix}\right) \quad \Leftrightarrow$$

$$u_j = \begin{bmatrix} v_j \\ \omega_j \end{bmatrix} = \begin{bmatrix} \cos\gamma_{ij} & -l_{ij}\sin\gamma_{ij} \\ \dfrac{\sin\gamma_{ij}}{d_j} & \dfrac{l_{ij}\cos\gamma_{ij}}{d_j} \end{bmatrix}\begin{bmatrix} s_{ij} + v_i\cos\psi_{ij} \\ b_{ij} + \omega_i - \dfrac{v_i\sin\psi_{ij}}{l_{ij}} \end{bmatrix} \quad \Leftrightarrow$$

$$\begin{bmatrix} v_j \\ \omega_j \end{bmatrix} = \begin{bmatrix} \cos\gamma_{ij}\left(s_{ij} + v_i\cos\psi_{ij}\right) - l_{ij}\sin\gamma_{ij}\left(b_{ij} + \omega_i - \dfrac{v_i\sin\psi_{ij}}{l_{ij}}\right) \\ \dfrac{\sin\gamma_{ij}}{d_j}\left(s_{ij} + v_i\cos\psi_{ij}\right) + \dfrac{l_{ij}\cos\gamma_{ij}}{d_j}\left(b_{ij} + \omega_i - \dfrac{v_i\sin\psi_{ij}}{l_{ij}}\right) \end{bmatrix} \quad \Leftrightarrow$$

$$\begin{bmatrix} v_j \\ \omega_j \end{bmatrix} = \begin{bmatrix} s_{ij}\cos\gamma_{ij} - l_{ij}\sin\gamma_{ij}\left(b_{ij}+\omega_i\right) + v_i \sin\psi_{ij}\sin\gamma_{ij} + v_i\cos\gamma_{ij}\cos\psi_{ij} \\ \frac{1}{d_j}\left(s_{ij}\sin\gamma_{ij} + l_{ij}\cos\gamma_{ij}\left(b_{ij}+\omega_i\right) + v_i\sin\gamma_{ij}\cos\psi_{ij} - v_i\cos\gamma_{ij}\sin\psi_{ij}\right) \end{bmatrix}$$

Considerando as igualdades

$$\cos(A-B) = \sin(A)\sin(B) + \cos(A)\cos(B)$$
$$\sin(A-B) = \sin(A)\cos(B) - \cos(A)\sin(B)$$

vê-se que:

$$v_i \sin\psi_{ij}\sin\gamma_{ij} + v_i\cos\gamma_{ij}\cos\psi_{ij} = v_i\cos\left(\psi_{ij}-\gamma_{ij}\right)$$
$$v_i \sin\gamma_{ij}\cos\psi_{ij} - v_i\cos\gamma_{ij}\sin\psi_{ij} = v_i\sin\left(\psi_{ij}-\gamma_{ij}\right)$$

Como $\gamma_{ij} = \theta_i + \psi_{ij} - \theta_j$, então:

$$v_i\cos\left(\psi_{ij}-\gamma_{ij}\right) = v_i\cos\left(\psi_{ij}-(\theta_i+\psi_{ij}-\theta_j)\right) = v_i\cos\left(\theta_i-\theta_j\right)$$
$$v_i\sin\left(\psi_{ij}-\gamma_{ij}\right) = v_i\sin\left(\psi_{ij}-(\theta_i+\psi_{ij}-\theta_j)\right) = v_i\sin\left(\theta_i-\theta_j\right)$$

Assim, partindo das equações da cinemática (2.2), obtêm-se as expressões para as velocidades linear e angular para o robot seguidor $R_j$ de um robot líder $R_i$:

$$v_j = s_{ij}\cos\gamma_{ij} - l_{ij}\sin\gamma_{ij}(b_{ij}+\omega_i) + v_i\cos(\theta_i-\theta_j) \quad (2.6)$$

$$\omega_j = \frac{1}{d_j}\left[s_{ij}\sin\gamma_{ij} + l_{ij}\cos\gamma_{ij}(b_{ij}+\omega_i) + v_i\sin(\theta_i-\theta_j)\right] \quad (2.7)$$

onde $l_{ij}^{\ d}$ é a separação desejada e $\psi_{ij}^{\ d}$ o ângulo entre os robots desejada.

Em [2] prova-se que este controlador é estável, com algumas condições (a trajectória do líder tem de ser bem-comportada). Este controlador funciona como o mais básico entre os estudados, e tem uma gama de aplicação muito geral. Para situações em que só existe o líder na proximidade, este controlador é ideal. No entanto, quando existem por perto outros robots pertencentes à equipa, é mais vantajoso utilizar o controlador estudado de seguida, pois o controlador anterior não está preparado para os evitar.

## *2.3 Controlador $l-l$ (Separation-Separation Control)*

Este controlador (denominado $S_{ik}S_{jk}C$) é uma variação do anterior; o robot seguidor $R_3$ vai seguir $R_1$ e $R_2$ com separação $l_{13}$ e $l_{23}$, respectivamente. É portanto necessário que existam dois robots líder. Uma formação muito comum é ter $R_1$ como líder, $R_2$ a seguir $R_1$ com o controlador $l-\psi$ e $R_3$ a seguir ambos com $l-l$ .

Na Figura 4 pode-se ver um caso típico:

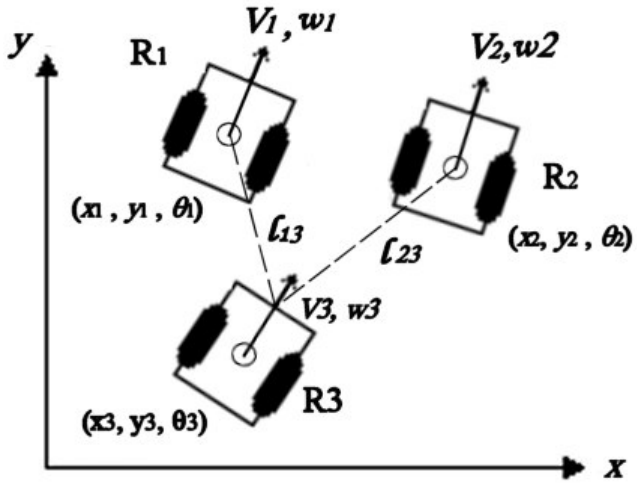


**Figura 4 – Controlador $l-l$**

O objectivo para este controlador é então manter constantes as distâncias $l_{13}$ e $l_{23}$.

Uma restrição para este controlador reside no facto de que $R_3$ está dependente não de um, mas de dois robots. Se a distância entre $R_1$ e $R_2$ for maior que $l_{13}^{d}$ + $l_{23}^{d}$, então não será fisicamente possível manter a formação, e o controlador falhará. Como os três robots formam um triângulo, existem assim três inequações que restringem o uso deste controlador:

$$l_{23}^{\ d} \le l_{12} + l_{13}^{\ d}$$
$$l_{13}^{\ d} \le l_{12} + l_{23}^{\ d}$$
$$l_{12} \le l_{13}^{\ d} + l_{23}^{\ d}$$

Representando estas restrições num gráfico $l_{13}(l_{23})$ , obtemos a Figura 5, onde a zona a cinzento representa o conjunto de formações possíveis com este controlador.

No entanto, é fácil prever uma situação em que este controlador traz vantagens em relação ao anterior: se $R_2$ e $R_3$ seguissem $R_1$ com um controlo $SB_{ij}C$, existiria

a possibilidade de haver colisões entre os dois robots seguidores se estes estivessem muito próximos; esta situação está prevista por este controlador, e todos os robots têm uma distância mínima entre eles.

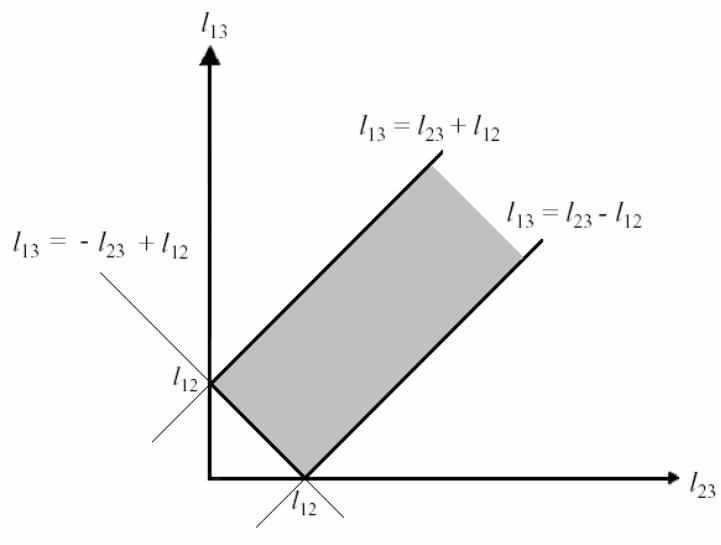


**Figura 5 – Restrições do controlador** $l-l$

Uma característica deste controlador reside no facto de que existem duas posições que satisfazem as condições de distância.

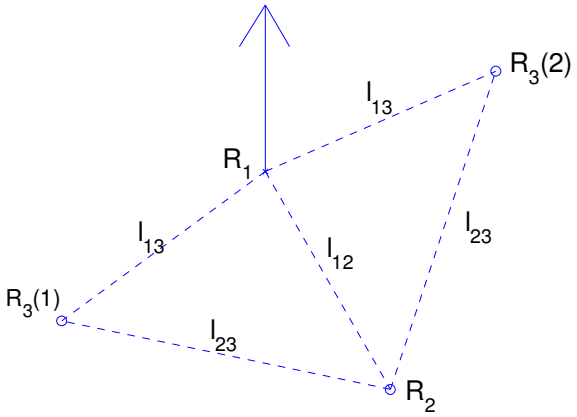


**Figura 6 – Duas soluções para o controlador** $l-l$

Na Figura 6 está um exemplo: $R_2$ está separado de $R_1$ por $l_{12}$, e ambos $R_3(1)$ e $R_3(2)$ estão separados $l_{13}$ de $R_1$ e $l_{23}$ de $R_2$. Assim, o controlador vai levar o robot a uma destas posições, dependendo da posição inicial.

O sistema é agora:

$$\dot{z}_k = G(z_k)u_k + F(z_k, u_i, u_j) \qquad ,\dot{\theta}_k = \omega_k \tag{2.8}$$

com $z_k = \begin{bmatrix} l_{ik} & l_{jk} \end{bmatrix}^T$ e $u_i = \begin{bmatrix} v_i & \omega_u \end{bmatrix}^T$ , $u_j = \begin{bmatrix} v_j & \omega_j \end{bmatrix}^T$ , $u_k = \begin{bmatrix} v_k & \omega_k \end{bmatrix}^T$ e ainda:

$$G = \begin{bmatrix} \cos\gamma_{ik} & d_k \sin\gamma_{ik} \\ \cos\gamma_{jk} & d_k \sin\gamma_{jk} \end{bmatrix}$$

$$F = \begin{bmatrix} -v_i \cos\psi_{ik} \\ -v_j \cos\psi_{jk} \end{bmatrix}$$

$$\gamma_{ij} = \theta_i + \psi_{ij} - \theta_j$$

Manipulando (2.8) e fazendo cálculos semelhantes aos efectuados para o controlador $l-\psi$, obtém-se:

$$u_k = G^{-1}(p - F) \tag{2.9}$$

onde

$$p = \begin{bmatrix} k_1({l_{ik}}^d - l_{ik}) \\ k_1({l_{jk}}^d - l_{jk}) \end{bmatrix} \qquad \text{com } k_1 > 0$$

Como as duas variáveis tratam a mesma grandeza (distância), pode-se usar os mesmos ganhos ($k_1$).

Em malha fechada fica:

$$\dot{z}_k = p, \qquad \dot{\theta}_k = \omega_k \tag{2.10}$$

ou seja:

$$\dot{l}_{ik} = k_1({l_{ik}}^d - l_{ik}), \quad \dot{l}_{jk} = k_1({l_{jk}}^d - l_{jk}), \quad \dot{\theta}_k = \omega_k$$

Assim, utilizando (2.8), obtêm-se as velocidades do robot seguidor $R_k$ com os líderes $R_i$ e $R_j$:

$$v_k = \frac{s_{ik}\sin\gamma_{jk} - s_{jk}\sin\gamma_{ik} + v_i\cos\psi_{ik}\sin\gamma_{jk} - v_j\cos\psi_{jk}\sin\gamma_{ik}}{\sin(\gamma_{jk} - \gamma_{ik})} \tag{2.11}$$

$$\omega_k = \frac{-s_{ik}\cos\gamma_{jk} + s_{jk}\cos\gamma_{ik} - v_i\cos\psi_{ik}\cos\gamma_{jk} + v_j\cos\psi_{jk}\cos\gamma_{ik}}{d\sin(\gamma_{jk} - \gamma_{ik})} \tag{2.12}$$

onde:

$$\gamma_{ij} = \theta_i + \psi_{ij} - \theta_j$$

$$s_{ij} = k_1({l_{ij}}^d - l_{ij})$$

Este controlador é muito útil quando as distâncias entre os robots são muito pequenas, evitando assim o risco de colisão. Para formações onde as distâncias são maiores, o controlador $l-\psi$ é mais adequado. Estas informações serão úteis ao elaborar um protocolo de coordenação de formações.

Em [2] é provada a estabilidade deste controlador. Apesar de ambos os controladores serem suficientes para poder controlar um conjunto de robots, estes não suportam a presença de obstáculos na trajectória. O controlador seguinte apresenta um algoritmo capaz de contornar obstáculos.

## *2.4 Controlador* $l-\delta$ *(*Separation Distance-To-Object Control*)*

A situação em que o robot seguidor encontra um obstáculo enquanto mantém uma formação não é prevista nos controladores anteriores. O controlador *Separation Distance-To-Obstacle Control* (SDoC ou SDC) permite a um robot ($R_2$) evitar obstáculos enquanto segue o líder ($R_1$). Mais concretamente, quando um robot detecta um obstáculo com os sensores, cria um robot virtual que percorre a fronteira do obstáculo, e comporta-se de uma forma análoga ao controlador Separação-Separação.

O robot virtual $R_0$ é mantido a uma distância $\delta$ de $R_2$, e move-se com velocidade linear $v_0$ e orientação $\theta_0$. Na Figura 7 é dado um exemplo deste caso. O robot virtual percorreria a fronteira do obstáculo na posição que se encontrasse mais próxima de R2, de modo a minimizar possibilidades de colisão.

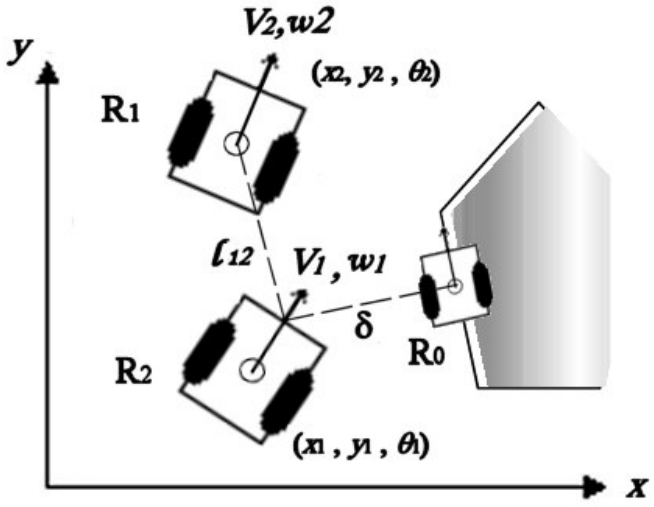


**Figura 7 – Controlador** $l-\delta$

Os objectivos deste controlador são um pouco diferentes dos controladores anteriores: em primeiro lugar, este é um controlador que não estará sempre a funcionar com obstáculos (a maior parte do tempo espera-se que estes não existam). Em segundo lugar, a distância $\delta$ é uma espécie de distância de segurança, não sendo portanto necessário que $R_2$ se encontre a uma distância mínima de $\delta$. No entanto, se um robot seguidor encontra um obstáculo, é porque o controlo de formações normal tinha criado uma trajectória que passava através do obstáculo, sendo portanto necessário evitá-lo, mas manter o percurso o mais próximo possível do inicial.

O ideal seria ter um controlador a funcionar, $l-\psi$ por exemplo, e quando fosse detectado um obstáculo a menos de uma distância próxima, haveria uma comutação do controlador.

O sistema de $R_i$, $R_j$ e $R_0$ é dado por:

$$\dot{z}_j = G(z_j)u_j + F(z_j, u_i) \qquad ,\dot{\theta}_j = w_j \tag{2.13}$$

com $z_j = \begin{bmatrix} l_{ij} & \delta \end{bmatrix}^T$ e $u_i = \begin{bmatrix} v_i & \omega_u \end{bmatrix}^T$ , $u_j = \begin{bmatrix} v_j & \omega_j \end{bmatrix}^T$ e ainda:

$$G = \begin{bmatrix} \cos\gamma_{ij} & d_j \sin\gamma_{ij} \\ \sin\gamma_{0j} & d_j \cos\gamma_{oj} \end{bmatrix}$$

$$F = \begin{bmatrix} -v_i \cos\psi_{ij} \\ 0 \end{bmatrix}$$

$$\gamma_{ij} = \theta_i + \psi_{ij} - \theta_j \ , \ \gamma_{0j} = \theta_0 - \theta_j$$

Manipulando (2.13) e efectuando cálculos semelhantes aos efectuados para os controladores anteriores, obtém-se:

$$u_j = G^{-1}\left(p \ -F\right) \tag{2.14}$$

onde:

$$p = \begin{bmatrix} k_1(l_{ij}{}^d - l_{ij}) \\ k_0(\delta_0 - \delta) \end{bmatrix} \qquad \text{com } k_i > 0$$

Em malha fechada fica:

$$\dot{z}_k = p, \qquad \dot{\theta}_k = \omega_k \tag{2.15}$$

ou seja:

$$\dot{l}_{ij} = k_1(l_{ij}{}^d - l_{ij}), \qquad \dot{\delta} = k_0(\delta_o - \delta), \qquad \dot{\theta}_j = \omega_j$$

As velocidades para o robot $R_j$ são:

$$v_j = \frac{s_{ij}\cos\gamma_{oj} + s_{oj}\sin\gamma_{ij} + v_i\cos\psi_{ij}\cos\gamma_{oj}}{\cos(\gamma_{oj} - \gamma_{ij})} \tag{2.16}$$

$$\omega_j = \frac{s_{ij}\sin\gamma_{oj} - s_{oj}\sin\gamma_{ij} + v_i\cos\psi_{ij}\sin\gamma_{oj}}{d\cos(\gamma_{oj} - \gamma_{ij})} \tag{2.17}$$

onde $s_{oj} = k_0(\delta_o - \delta)$ , e $\delta_o$ é a distância desejada entre o robot $R_j$ e o obstáculo.

É de notar que para fazer a linearização é necessário impor a restrição de que os vectores $\vec{\delta}$ e $\vec{l}_{ij}$ não podem ser colineares ($\cos(\gamma_{oj} + \gamma_{ij}) \neq 0$), isto é, o robot líder nunca pode estar entre o seguidor e o obstáculo (situação pouco usual). Este algoritmo é válido nas condições apresentadas para evitar obstáculos simples (muito estruturados). Pode-se também reparar que, se o robot seguidor estiver a fazer uma formação com dois líderes (com um controlo $l-l$ ), e detectar um obstáculo, vai ter forçosamente de abandonar a referência de um dos líderes para controlar a distância ao obstáculo. Esta característica pode ser

considerada como um ponto fraco do algoritmo. Este assunto é também discutido no capítulo sobre o protocolo de coordenação. Outro pormenor refere-se à inexistência de $v_0$ nas equações. Isso deve-se ao facto de que, para deduzir essas expressões considera-se $v_0$=0. Para tentar funcionar como um obstáculo móvel (por exemplo um robot não pertencente à equipa), talvez fosse possível considerar $v_0 \neq 0$, e deduzir novas expressões, ou simplesmente utilizar o controlador $l - l$ , mas com parâmetros diferentes para este caso especial. Quanto a $\theta_0$ , é obtido utilizando os sensores, ou mais simplesmente, determinando o ângulo de um vector perpendicular ao segmento de recta entre o robot e o ponto do obstáculo mais próximo. Este assunto é discutido no capítulo da implementação.

Alterando a relação entre $k_0$ e $k_1$, é possível configurar este controlador de modo a dar mais importância ao obstáculo (e a manter-se longe dele), ou ao líder (e fazer o possível para conservar a distância desejada constante). Deixando os dois ganhos iguais deverá criar uma situação de meio termo, em que ambos os objectivos têm o mesmo peso.

No caso da presença de dois obstáculos, o algoritmo vai só levar em conta o ponto mais próximo de qualquer um eles. Pode acontecer o caso em que esse ponto pertença a obstáculos diferentes em momentos diferentes. Este método funciona, novamente, para obstáculos simples.

Em [2] é mais uma vez provada a estabilidade deste controlador, salvo a excepção acima indicada.

## *2.5 Controlador $\psi-\psi$ (Dilation Control)*

Este controlador (denominado $SB_{ij}B_{ijk}C$) tem a particularidade de controlar não um mas dois robots. Assim, permite que $R_j$ mantenha uma separação $\rho^d$ e ângulo $\psi_{ij}^{\ d}$ com $R_i$, e ao mesmo tempo que $R_k$ siga $R_i$ e $R_j$ com ângulos relativos $\psi_{ik}^{\ d}$ e $\psi_{jk}^{\ d}$. Este controlador é, de certa forma, semelhante ao controlador $l-l$ (um seguidor com dois líderes), tendo como diferença a existência do factor de dilatação $\rho$. De facto, alterando apenas um parâmetro, consegue-se mudar a formação dos três robots, contraindo ou expandindo consoante $\rho$ diminui ou aumenta, mantendo sempre a forma (o triângulo formado pelos três robots permanece semelhante).

As equações cinemáticas reflectem agora $R_j$ e $R_k$:

$$\dot{z} = G(z)\bar{u} + F(z,u_i) \qquad ,\dot{\theta}_j = w_j \qquad ,\dot{\theta}_k = w_k \tag{2.18}$$

com $z = \begin{bmatrix}\rho & \psi_{ij} & \psi_{ik} & \psi_{jk}\end{bmatrix}^T$, $\bar{u} = \begin{bmatrix}v_j & \omega_j & v_k & \omega_k\end{bmatrix}^T$, $u_i = \begin{bmatrix}v_i & \omega_i\end{bmatrix}^T$, $u_k = \begin{bmatrix}v_k & \omega_k\end{bmatrix}^T$ e:

$$G = \begin{bmatrix} \cos\gamma_{ij} & d\sin\gamma_{ij} & 0 & 0 \\ \dfrac{-\sin\gamma_{ij}}{l_{ij}} & \dfrac{d\cos\gamma_{ij}}{l_{ij}} & 0 & 0 \\ 0 & 0 & \dfrac{-\sin\gamma_{ik}}{l_{ik}} & \dfrac{d\cos\gamma_{ik}}{l_{ik}} \\ \dfrac{\sin\psi_{jk}}{l_{jk}} & -1 & \dfrac{-\sin\gamma_{jk}}{l_{jk}} & \dfrac{d\cos\gamma_{jk}}{l_{jk}} \end{bmatrix}$$

$$F = \begin{bmatrix} -v_i\cos\psi_{ij} \\ \dfrac{v_i\sin\psi_{ij}}{l_{ij}} - \omega_i \\ \dfrac{v_i\sin\psi_{ik}}{l_{ik}} - \omega_i \\ 0 \end{bmatrix}$$

$$\gamma_{ij} = \theta_i + \psi_{ij} - \theta_j$$

Manipulando (2.18), obtém-se:

$$u_k = G^{-1}(p - F) \tag{2.19}$$

onde

$$p = \begin{bmatrix} k_1({l_{ij}}^d - l_{ij}) \\ k_2({\psi_{ij}}^d - \psi_{ij}) \\ k_2({\psi_{ik}}^d - \psi_{ik}) \\ k_2({\psi_{jk}}^d - \psi_{jk}) \end{bmatrix} \qquad \text{com } k_i > 0$$

Como três das variáveis tratam a mesma grandeza (ângulo), pode-se utilizar os mesmos ganhos ($k_2$). De notar que $l_{ij} = \rho$ (o factor de dilatação). Em malha fechada fica:

$$\dot{z} = p\,, \qquad \dot{\theta}_j = w_j \qquad ,\dot{\theta}_k = w_k \tag{2.20}$$

ou seja:

$$\dot{l}_{ij} = k_1({l_{ij}}^d - l_{ij}), \quad \dot{\psi}_{ij} = k_2({\psi_{ij}}^d - \psi_{ij}), \quad \dot{\psi}_{ik} = k_2({\psi_{ik}}^d - \psi_{ik}) \quad \dot{\psi}_{jk} = k_2({\psi_{jk}}^d - \psi_{jk})$$

$$\dot{\theta}_j = w_j, \qquad \dot{\theta}_k = w_k$$

Nota: as velocidades do robot seguidor $R_i$ com o líder $R_i$ são as mesmas que no controlador $l - \psi$ .

# 3. Protocolo de Coordenação de Formações

No capítulo anterior foram estudados vários controladores capazes de manter uma formação estável num conjunto de robots móveis. No entanto, na prática, os robots estão condicionados ao desempenho dos sensores, actuadores e às comunicações, além do próprio mundo. Neste capítulo vai ser descrito um protocolo a seguir de modo a que os robots adoptem a formação adequada em cada situação.

Primeiro de tudo convém diferenciar entre formação e controlador de formação. Uma formação resulta da aplicação de um controlador com certos parâmetros (por exemplo $l_{ij}^{\ d}$ e $\psi_{ij}^{\ d}$ para o controlador SBC). Quando se refere o conjunto das formações, fala-se do conjunto de formações que podem ser obtidas variando os parâmetros para o mesmo controlador. Já o conjunto de controladores refere os vários controladores disponíveis para cada um dos robot da formação utilizar (independentemente dos parâmetros escolhidos). Para escolher a formação é portanto necessário escolher primeiro um controlador, e de seguida obter parâmetros a utilizar como referência nesse controlador. É de notar que em algumas aplicações faz sentido alterar estes parâmetros, enquanto que noutras não. O protocolo de coordenação de formações considerado utiliza formações constantes para cada controlador, isto é, para um dado controlador escolhido existem parâmetros pré definidos a utilizar.

Seja Uj={$C_{1,j}$ , $C_{2,j}$ , ...} o conjunto de controladores disponível a um robot $R_j$ num dado momento. Para $R_j$ obter os valores para a actuação dos motores, é necessário escolher um controlador $C_{i,j} \in U_j$. Como neste trabalho é estudado o caso particular de três robots não-holonómicos, também o protocolo vai ser só aplicado à equipa de robots $R_{1,2,3}$, apesar da extensão a um grupo de *n* robots ser apenas uma generalização deste protocolo.

Considera-se que os robots entram na formação sequencialmente. Isto é, inicialmente a formação é formada por $R_1$, de seguida é introduzido um robot $R_2$, depois $R_3$, etc. $R_1$, o robot líder, segue uma dada trajectória *g(t)*. Como $R_2$ só tem $R_1$ como líder, utiliza um controlador *separation-bearing* ($U_2$={$SB_{12}C$}), que é o único controlador que necessita de apenas um líder. Já $R_3$ tem três controladores à escolha: U3={$SB_{13}C$, $SB_{23}C$ ou $S_{13}S_{23}C$}. Ou seja, neste caso simples de três robots temos três formações diferentes (se não permutarmos $R_2$ com $R_3$). O conjunto de formações é então dado por U={$U_2$ x $U_3$}. Se cada controlador é estável isoladamente, então cada formação também tem de ser estável. É necessário verificar também que o sistema é estável quando ocorre uma transição de uma formação para outra.

## *3.1 Protocolo de Coordenação na ausência de obstáculos*

O problema da escolha do controlador é condicionado primariamente pela capacidade de obtenção de informação acerca do mundo e pelos obstáculos presentes. Podem ser consideradas duas situações principais: com e sem a presença de obstáculos. Na Figura 8 pode-se ver uma estratégia para mudança de controlador para $R_3$, na ausência de obstáculos:

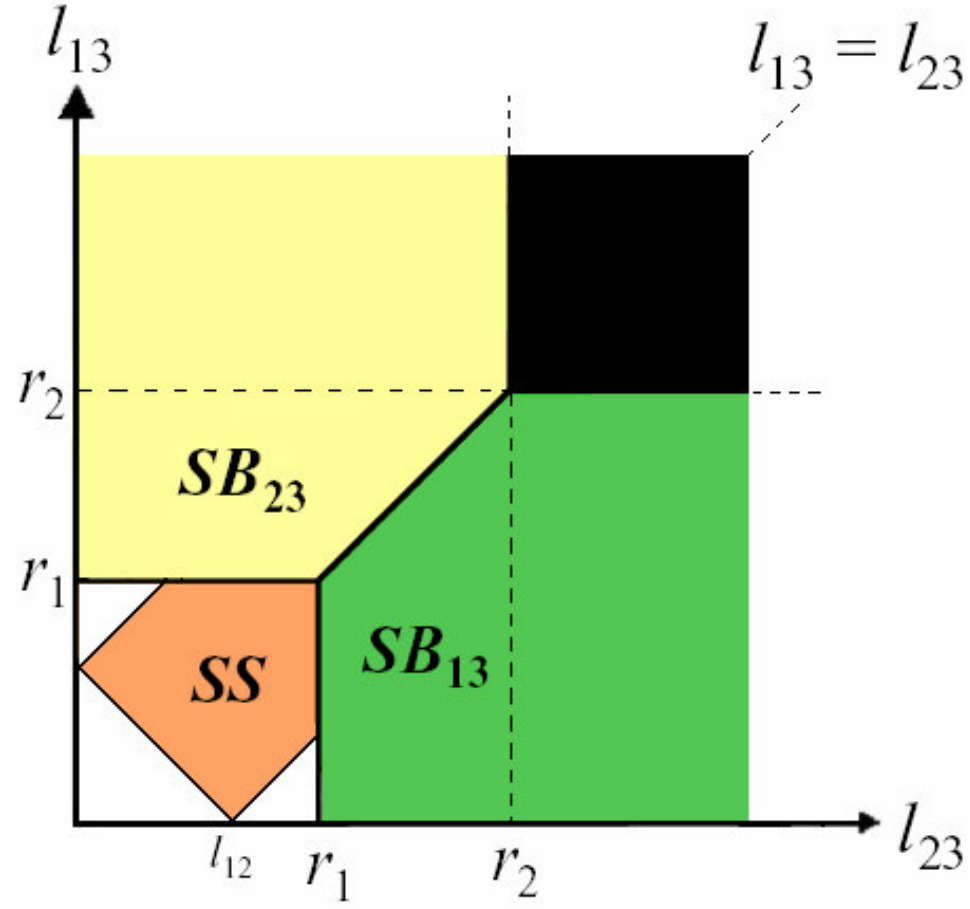


**Figura 8 – Escolha do controlador (sem obstáculos)**

Na Figura 8, $l_{ij}$ representa a distância entre o robot $R_i$ e $R_j$, $r_2$ é a distância máxima a que um robot consegue detectar outro robot utilizando os sensores. $r_1$ é um alcance prê-definido, menor que $r_2$, abaixo do qual um robot pode e deve procurar um segundo líder.

Obviamente na figura não são consideradas as restrições relativas ao espaço físico dos robots, e é evidente que $l_{13}$ e $l_{23}$ não podem ser demasiado pequenas, de forma a os robots não chocarem entre si.

Existem portanto 4 zonas diferentes: .

- $l_{23} < r_1$ e $l_{13} < r_1$ (zona a laranja) ➔ $R_3$ está bastante próximo dos outros dois robots, e deve utilizar o controlo $S_{13}S_{23}$, já que é o que minimiza o risco de colisões. De notar que é necessário respeitar as restrições para este controlador, tal como é indicado em 2.3.
- $l_{23} > r_1$ , $l_{23} < l_{13}$ e $l_{13} < r_2$ (zona a verde) ➔ $R_3$ está mais próximo de $R_1$ do que de $R_2$, e suficientemente afastado de ambos de modo a não haver risco de colisão. Nesta situação utiliza-se o controlo $SB_{13}$.
- $l_{13} > r_1$ , $l_{13} < l_{23}$ e $l_{23} < r_2$ (zona a amarelo) ➔ Idêntico à zona anterior, mas com $R_2$ em vez de $R_3$. Utiliza-se o controlo $SB_{23}$.
- $l_{13} > r_2$ e $l_{23} > r_2$ (zona a preto) ➔ Nesta zona o robot não consegue localizar nenhum líder, por ambos estarem demasiado afastados. Nesta situação o robot deveria seguir uma navegação autónoma até localizar um líder.

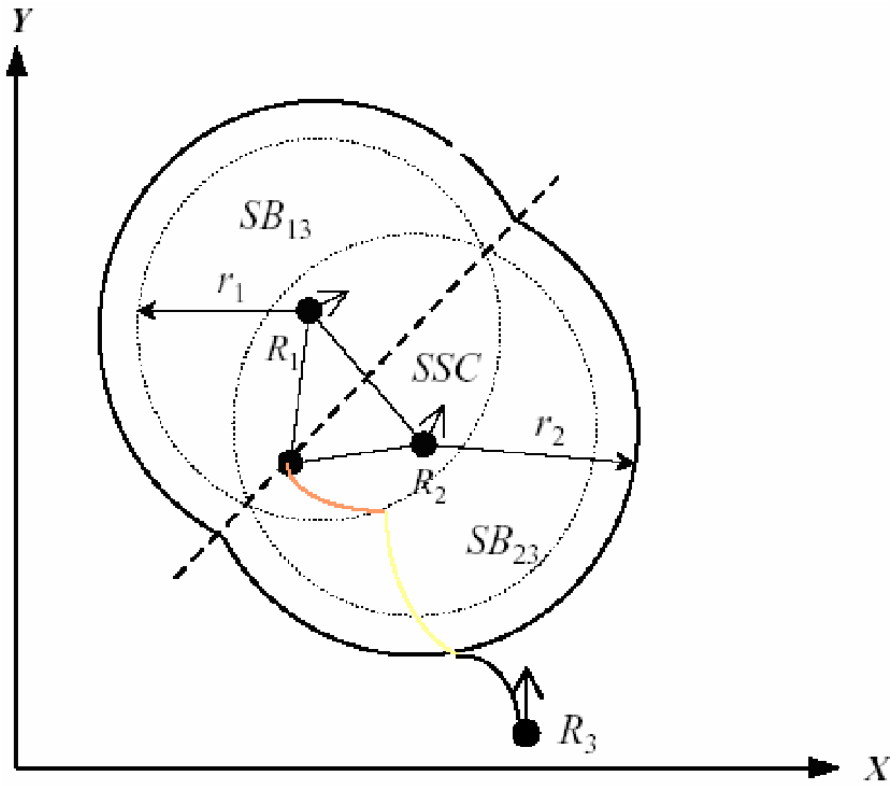


**Figura 9 – Exemplo do protocolo de coordenação sem obstáculos**

Na Figura 9 pode-se ver $R_3$ a escolher o controlador em cada situação de acordo com as regras anteriores. O percurso de $R_3$ divide-se em três (cada um com uma cor relativa à zona da Figura 8 onde se encontra: preto, amarelo e laranja).

Inicialmente, $R_3$ não tem nenhum líder sob o alcance dos seus sensores, pelo que segue uma trajectória autónoma (trajectória a preto na Figura 9, zona a preto na Figura 8), até detectar $R_2$. Nesse momento, só tem disponível um líder, pelo que a única hipótese é o controlador $SB_{23}$ (trajectória a amarelo na Figura 9, zona amarela na Figura 8). A trajectória segundo este controlador leva $R_3$ a entrar numa zona onde consegue detectar um segundo líder, $R_1$. A partir do momento em que a distância entre $R_3$ e $R_1$ é menor que $r_1$, começa a existir o risco de haver colisões entre os robots, logo é escolhido o controlador SSC (trajectória a laranja na Figura 9, zona a laranja na Figura 8). Com este exemplo consegue-se perceber o funcionamento básico deste protocolo de coordenação de formações, na inexistência de obstáculos.

É de notar que, sendo o controlador SSC o mais adequado para um terceiro robot numa equipa de 3 elementos, pode-se considerar que este protocolo tem como objectivo levar o sistema ao caso típico: $R_1$ líder, $R_2$ a seguir $R_1$ com $SB_{12}C$ e $R_3$ a seguir ambos com SSC. De facto, até seria adequado fornecer parâmetros ao controlador SB durante a altura em que $R_3$ só tem um líder (trajectória a amarelo na Figura 9) de modo a levar o robot a uma zona onde encontre um segundo líder.

Outro caso a considerar é de ter ambos $R_2$ e $R_3$ com controlador SSC. Neste caso, $R_1$ é líder dos dois seguidores, assim como $R_2$ é líder de $R_3$, e vice-versa. Nesta situação, a formação seria mantida em termos de distâncias entre robots, mas não em termos de ângulo relativo à orientação de $R_1$, dado que $\psi$ não é controlado por nenhum dos robots. Como resultado, a formação 'roda' em torno de $R_1$, já que o controlo se refere apenas às distâncias. Como tal, na prática este caso não é permitido. $R_2$ segue sempre apenas $R_1$, e é $R_3$ que utiliza este protocolo de coordenação. No caso da Figura 9, $R_2$ segue $R_1$ com o controlador $SB_{12}C$, independentemente de $R_3$ estar ou não ao alcance dos seus sensores.

Esta estratégia de coordenação de formações, proposta em [2], faz com que em cada situação o controlador mais indicado seja escolhido. No entanto, se existisse comunicação por WLAN, cada robot poderia partilhar a sua posição, e todo o estado seria observável, não havendo a possibilidade de o robot se encontrar na quarta zona, sendo nesse caso o valor de $r_2$ considerado infinito. Este protocolo funciona, tal como foi demonstrado, num ambiente sem obstáculos. Quando os robots não são os únicos componentes no mundo, é necessário que o protocolo também considere as decisões a tomar nesse caso.

## *3.2 Protocolo de Coordenação na presença de obstáculos*

Se um obstáculo é detectado, um robot vai ter de mudar para o controlador $S_OD$. Considere-se o caso típico em que $R_2$ segue $R_1$ com $SB_{12}C$ quando detecta um obstáculo. Se a distância ao obstáculo $\delta$ for menor que uma dada distância de segurança $\delta_{seg} > \delta_0$ ( sendo $\delta_0$ a distância desejada ao obstáculo) e $\beta_{10} < \pi$ (onde $\beta_{10}$ é o ângulo entre os vectores $\vec{\delta}$ e $l_{12}$ que revela se o $R_2$ está a dirigir-se na direcção do obstáculo ou não) e $R_1$ estiver dentro do alcance dos sensores ( $l_{12} < r_2$ ), então é utilizado o controlador $S_OD$. Se $\delta > \delta_{seg}$ e o líder estiver presente, então regressa ao controlador inicial $SB_{12}C$. Se o líder não estiver presente depois de o robot sair da zona de segurança, então inicia o percurso em navegação autónoma.

Existem, portanto, condições para iniciar a usar este controlador, e condições para deixar de o usar, e de considerar o obstáculo ultrapassado. Resumindo:

- $\delta < \delta_{seg}$ , $\beta_{10} < \pi$ e $l_{12} < r_2$ ➔ $R_2$ encontrou entrou na área de segurança de um obstáculo enquanto seguia $R_1$. Neste caso é escolhido o controlador $S_OD$.
- $\delta > \delta_{seg}$ e $l_{12} < r_2$ ➔ $R_2$ saiu da área de segurança do obstáculo, e continua a seguir $R_1$ com o controlador $SB_{12}C$.
- $\delta > \delta_{seg}$ e $l_{12} > r_2$ ➔ Se quando $R_2$ sai da área de segurança do obstáculo e o líder $R_1$ saiu do alcance dos sensores, então $R_2$ vai iniciar um percurso em navegação autónoma.

Na Figura 10 vê-se um exemplo deste protocolo. $R_2$ não tem, no início, nenhum líder, pelo que segue autonomamente (percurso a preto na Figura 10). Ao detectar $R_1$, muda para o controlador $SB_{12}C$ (percurso a azul escuro na Figura 10). Enquanto segue $R_1$, $R_2$ entra na zona de segurança de um obstáculo ( $\delta < \delta_{seg}$ ), e muda para o controlador $SD_OC$ (percurso a azul claro na Figura 10). Enquanto $R_2$ tiver este controlador, vai contornar o obstáculo a uma distância $\delta_0$ (contorno exterior do círculo a vermelho), comportando-se de uma forma bastante parecida com o controlador SSC, até que o percurso do líder $R_1$ faça com que $R_2$ tenha de abandonar a distância de segurança ao obstáculo, para voltar ao controlador $SB_{12}C$.

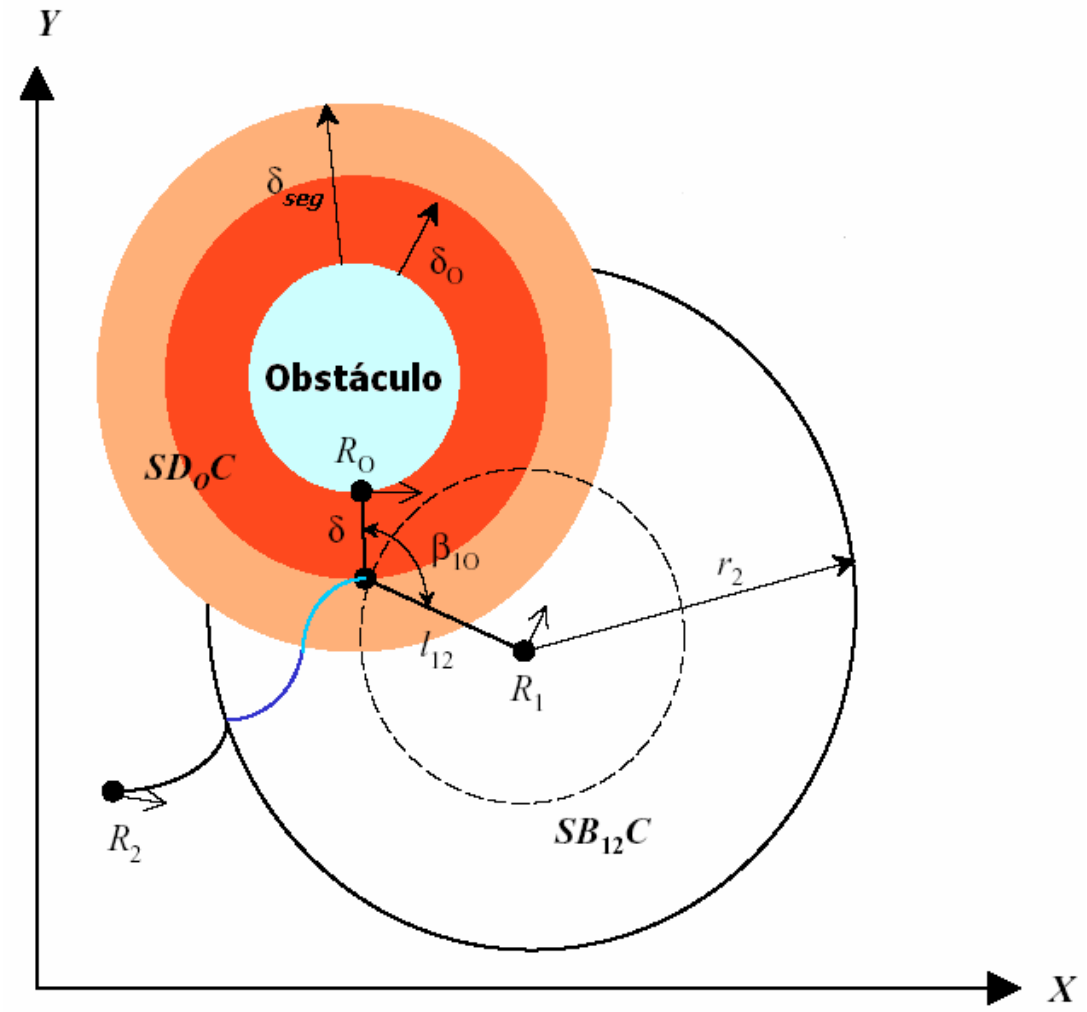


**Figura 10 – Exemplo do protocolo de coordenação com obstáculos**

A condição referente a $\beta_{10}$ pode ser alterada de modo a fazer com que o robot abandone o obstáculo mais cedo ou mais tarde que o normal. Outra condição que faria sentido seria $|\beta_{10}| < \pi/2$, que representaria se o percurso que $R_2$ teria de percorrer ao seguir o líder o levaria à zona de segurança do obstáculo. Em simulação foram testadas várias alterações, e optou-se por utilizar a condição $\beta_{10} < 2\pi/3$. Seria interessante ignorar esta condição para verificar as diferenças nos resultados.

Esta segunda parte do protocolo também foi desenvolvida em [2]. É de notar que este protocolo, tanto para o caso sem obstáculo como com obstáculos, considera uma mudança de formação a alteração do controlador para o robot, e não a alteração dos parâmetros do mesmo controlador. Fazer um sistema em que os parâmetros fossem adaptados seria interessante, mas tal não é contemplado neste trabalho.

# 4. Simulação

Para testar o desempenho dos controladores estudados procedeu-se, numa primeira fase, à simulação, desenvolvendo um simulador em ambiente *Matlab* (*Simulink*). No Anexo B é descrito o modo de funcionamento básico deste simulador, assim como as limitações do mesmo.

Neste simulador são simuladas as trajectórias de dois robots, $R_2$ e $R_3$, que têm por líder $R_1$, cuja trajectória é determinada *a priori*. É possível testar três dos controladores estudados em 2., $l-\psi$, $l-l$ e $l-\delta$. Na primeira parte deste capítulo, simulam-se os controladores $l-\psi$, $l-l$ isoladamente, sem considerar a presença de obstáculos. Nas simulações seguintes utilizam-se trajectórias simples para o líder $R_1$, já que é essa uma das hipóteses simplificativas. O tempo de simulação é sempre de 30 segundos. A formação inicial dos robots nunca é exactamente igual à formação desejada.

A trajectória de $R_1$ é determinado pelo utilizador como um vector de pontos (x,y). $\theta_1$ é obtido determinado o ângulos do vector formado entre dois pontos seguidos da trajectória de $R_1$, e $\omega_1(t)=\theta_1(t)-\theta_1(t-1)$. A velocidade linear é obtida derivando a posição em x e em y.

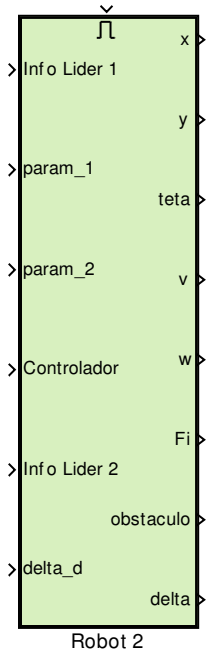


**Figura 11 – Bloco Simulink do Robot**

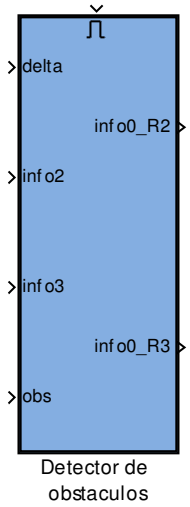


**Figura 12 – Bloco Simulink do obstáculo**

O bloco representado na Figura 11 simula o funcionamento de um robot seguidor. Tem como entradas 'Info Lider 1' e 'Info Lider 2', nomeadamente postura e velocidades dos líderes, 'Controlador', escolhido pelo utilizador, 'param_1' e 'param_2', os parâmetros do controlador, e 'delta_d', a distância desejada ao obstáculo. Como saídas temos a postura e velocidades do robot, bem como 'Fi', 'obstaculo' e 'delta', para posterior tratamento de dados. Dentro do bloco existem sub-blocos com as seguintes funções: cálculo das variáveis de controlo, assim como variáveis auxiliares (por exemplo, $l_{ij}$, $\gamma_{ij}$, $s_{ij}$, etc.); bloco de controlo, que calcula as velocidades do robot seguidor de acordo com o controlador escolhido; à saída do bloco de controlo as velocidades são saturadas (velocidade linear entre 0 e 2 metros por segundo e velocidade angular entre -5 e 5 metros por segundo); bloco com a cinemática do robot; bloco que determina se o obstáculo está perto e se é necessário utilizar o controlador de desvio de obstáculos.

Existem no simulador dois blocos semelhantes ao da Figura 11, representando $R_2$ e $R_3$. Como se pode ver, seria fácil acrescentar um quarto robot, copiando o bloco e escolhendo os líderes e parâmetros adequados. No entanto, o resto do simulador, nomeadamente a interface, não está preparada para tal alteração.

O bloco representado na Figura 12 simula a posição do obstáculo no mundo dos robots. Este bloco só está activo se o controlador escolhido para pelo menos um dos robots for o de desvio de obstáculos. As entradas são 'delta' (delta segurança), 'info2' e 'info3' (inclui posição actual dos dois robots seguidores) e 'obs', que inclui um vector com todos os pontos do obstáculo. Este bloco procura e devolve na saída ('info0_R2' e 'info0R3') o ponto do obstáculo mais próximo de cada um dos robots.

## *4.1 Simulação dos controladores isoladamente*

Num primeiro exemplo, utilizamos a formação mais simples: $R_2$ a seguir $R_1$ com $SB_{12}C$, e $R_3$ a seguir $R_1$ e $R_2$ com $S_{13}S_{23}C$. O resultado é o seguinte (Figura 13):

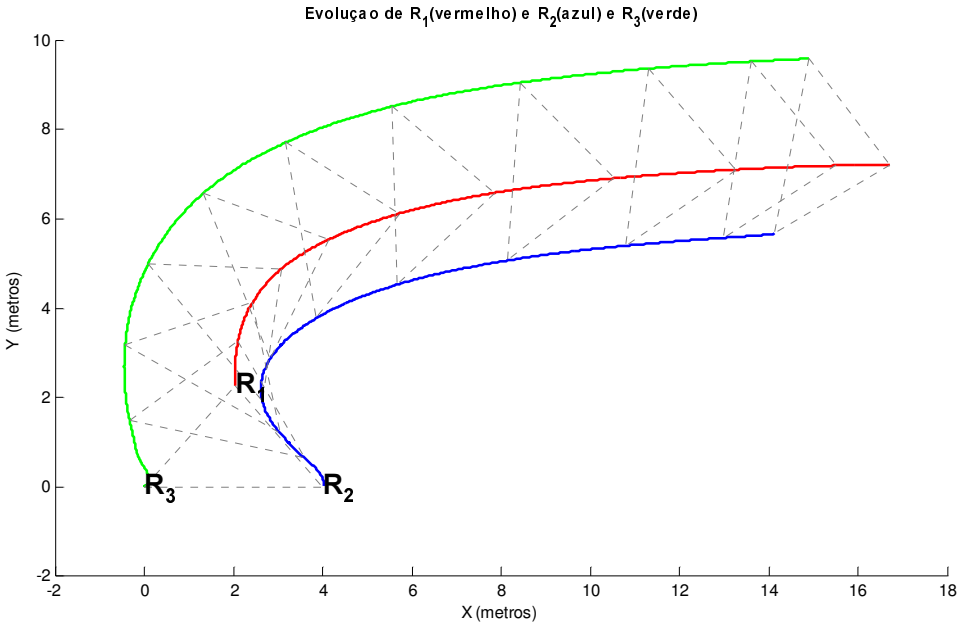


**Figura 13 – Simulação 1 ($SB_{12}C$: $l_{12}^d$=3 , $\psi_{12}^d$ =210º ; $S_{13}S_{23}C$: $l_{13}^d$ =3 ; $l_{23}^d$ =4 )**

Para melhor perceber a evolução da formação, é conveniente observar os triângulos a tracejado. Podemos ver claramente que a formação tende para a desejada. Note-se que, neste primeiro exemplo, a formação inicial é bastante semelhante à formação final desejada.

Na Figura 14 foi simulada a mesma formação que na Figura 13, com a diferença na trajectória do líder $R_1$. Como a posição inicial do líder está bastante longe de pertencer à formação desejada, consegue-se observar claramente a convergência dos robots seguidores ao mesmo triângulo que na simulação anterior.

Alterando a posição inicial dos robots (Figura 15) mas mantendo a mesma formação, verifica-se que esta converge bastante rapidamente para a desejada, apesar do erro inicial. Nesta simulação, durante os instantes iniciais, ambos os robots andam à velocidade máxima permitida, de modo a atingirem a formação o mais rapidamente possível. Isto faz sentido, ao pensarmos que nestes instantes iniciais, o erro de separação entre os robots é muito grande. $R_2$ tenta manter a formação com $R_1$, enquanto que $R_3$ tenta manter-se a uma distância relativa constante a $R_1$ e a $R_2$ (daí a curva inicial).

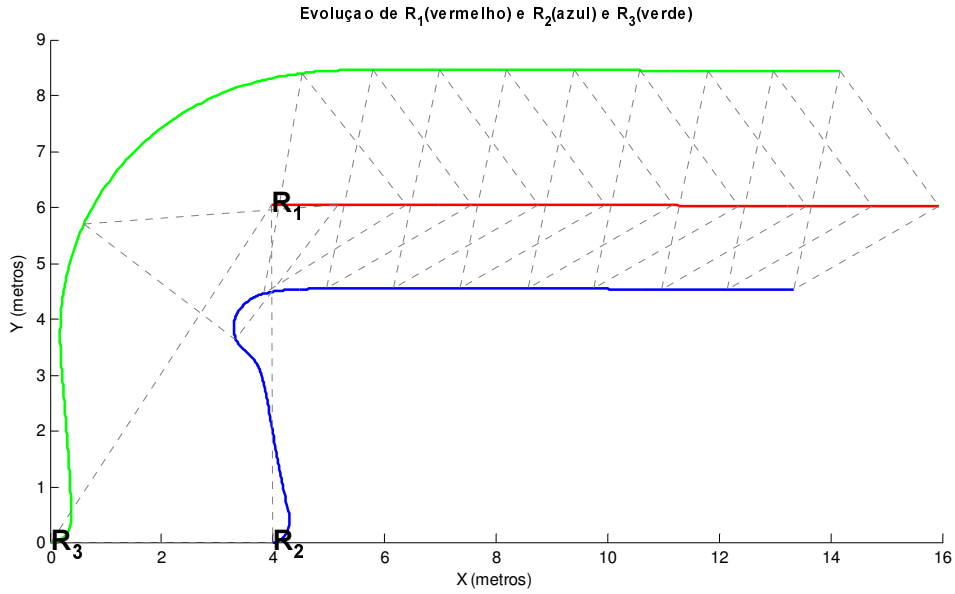


**Figura 14 – Simulação 2 ($SB_{12}C$: $l_{12}{}^d$ =3 , $\psi_{12}{}^d$ =210º ; $S_{13}S_{23}C$: $l_{13}{}^d$ =3 ; $l_{23}{}^d$ =4 )**

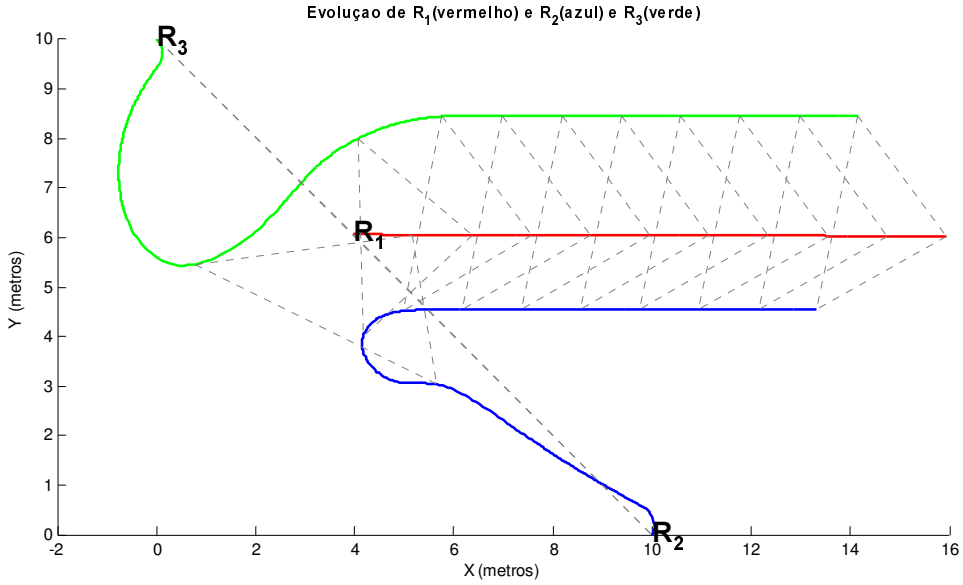


**Figura 15 – Simulação 3 ($SB_{12}C$: $l_{12}{}^d$ =3 , $\psi_{12}{}^d$ =210º ; $S_{13}S_{23}C$: $l_{13}{}^d$ =3 ; $l_{23}{}^d$ =4)**

Numa outra simulação, com resultados na Figura 16, utilizou-se a mesma formação que na anterior, mas alterou-se a trajectória do líder de modo a fazer uma curva apertada a uma certa altura.

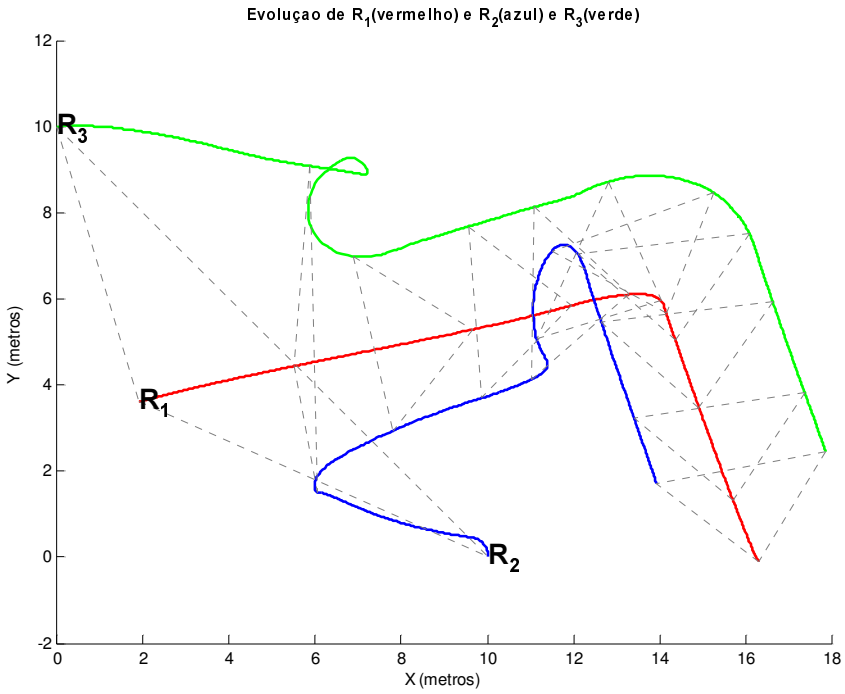


**Figura 16 – Simulação 4 – ($SB_{12}C$: $l_{12}{}^{d}$ =3 , $\psi_{12}{}^{d}$ =210º ; $S_{13}S_{23}C$: $l_{13}{}^{d}$ =3 ; $l_{23}{}^{d}$ =4)**

Pode-se verificar novamente que, após uma convergência inicial, os robots mantém a formação, mesmo depois da curva; $R_3$ (a verde) faz também uma pequena curva, enquanto $R_2$ (a azul) precisa de recuar para ficar em formação.

Esta última simulação faz-nos pensar no limite da complexidade da trajectória do líder, para que os seguidores mantenham a formação. A simulação seguinte ajuda a perceber quais essas limitações (Figura 17). Aqui, o líder faz uma curva e uma contracurva quase de seguida. Ambos os seguidores mantém as distâncias desejadas, mas os ângulos $\psi_{ij}$ sofrem uma grande perturbação, apesar de no fim a formação ser de novo alcançada. Durante as curvas, os robots seguidores têm as velocidades dos motores muito elevadas, e por vezes saturadas. Se a curva do líder tivesse sido um pouco mais rápida, poderia não ter sido possível não manter a formação, pelo menos temporariamente. Daqui vem o pressuposto que a trajectória do líder tem de ser bem-comportada, considerando um bom comportamento como uma trajectória que não envolva movimentos ou curvas com raio reduzido. Outra característica envolve a velocidade do líder; para se manter a formação, o líder tem de andar a uma velocidade não muito elevada, geralmente inferior à velocidade máxima dos robots seguidores.

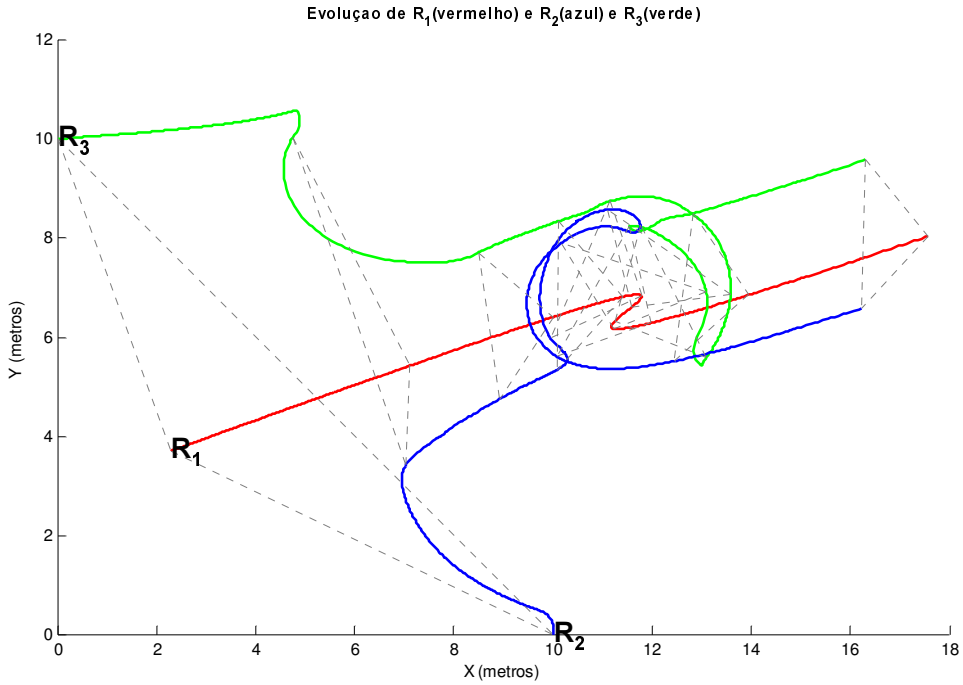


**Figura 17 – Simulação 5 - ($SB_{12}C$: $l_{12}{}^{d}$ =3 , $\psi_{12}{}^{d}$ =210º ; $S_{13}S_{23}C$: $l_{13}{}^{d}$ =3 ; $l_{23}{}^{d}$ =4)**

De seguida vejamos um caso semelhante, com o líder a fazer também duas curvas, mas muito menos apertadas, e mais distanciadas entre si (Figura 18).

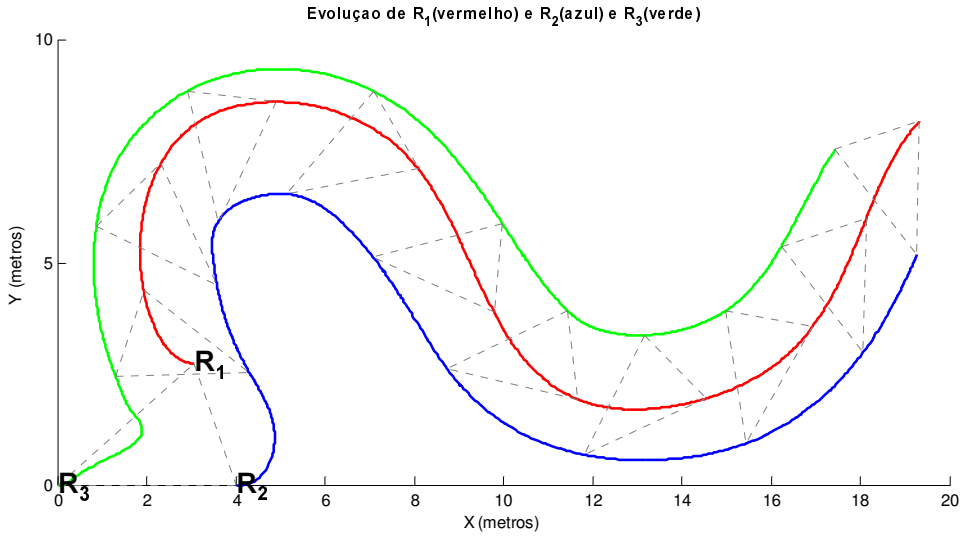


**Figura 18 – Simulação 6 - ($SB_{12}C$: $l_{12}{}^{d}$ =4 , $\psi_{12}{}^{d}$ =210º ; $S_{13}S_{23}C$: $l_{13}{}^{d}$ =3 ; $l_{23}{}^{d}$ =4)**

Para este caso, vejamos também a evolução das distâncias e ângulos em causa (Figura 19 e Figura 20). Podemos ver que enquanto para $R_3$ ambas as referências são alcançadas rapidamente, em $R_2$ o ângulo $\psi_{12}$ nunca fica muito constante. Isto deve-se ao facto de que $R_2$ tem de fazer uma curva apertada para seguir $R_1$, enquanto $R_3$ apenas tem de manter distâncias. Se fosse usado um controlador SBC em $R_3$, então os resultados seriam semelhantes aos de $R_2$.

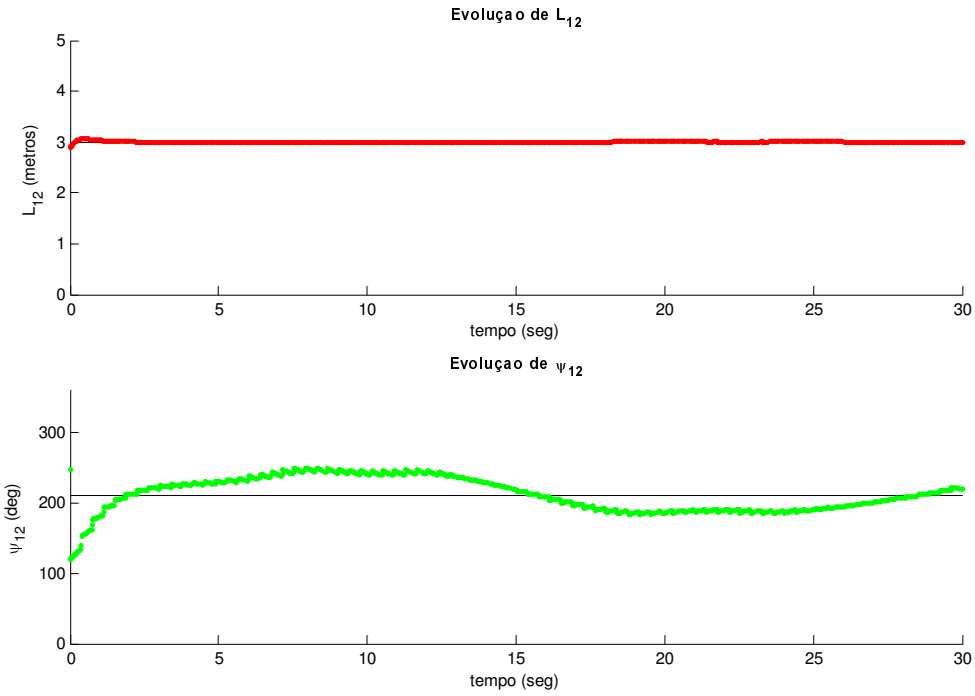


**Figura 19 – Simulação 6 - Controlo de $R_2$**

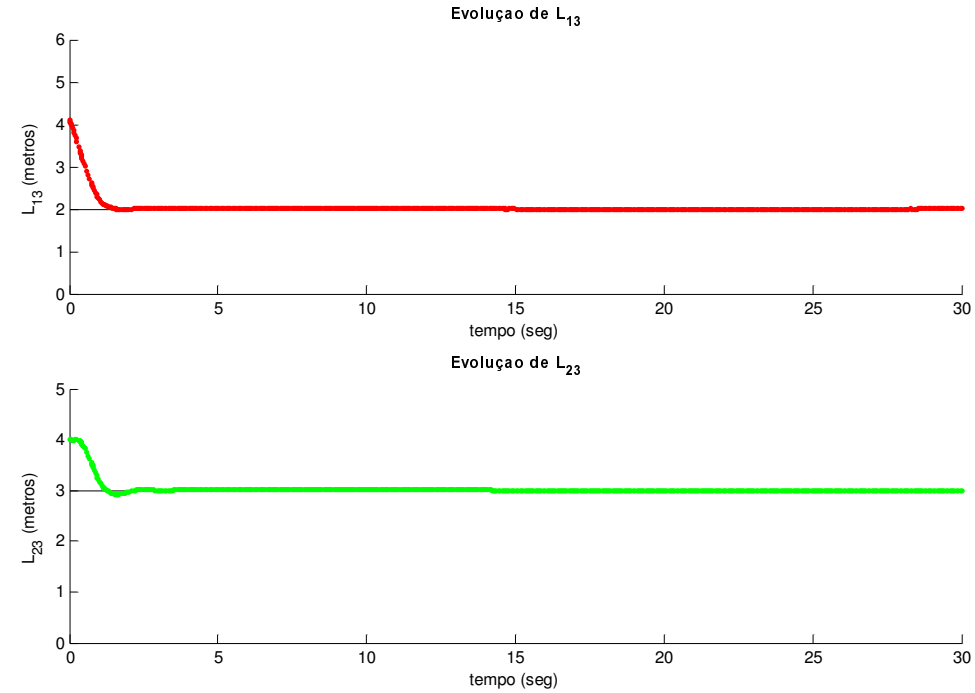


**Figura 20 – Simulação 6 - Controlo de $R_3$**

Outra simulação com estes dois controladores mostra a possibilidade de um robot seguidor ser um líder para outro robot. Na Figura 21 temos $R_1$ como líder de $R_3$, que por sua vez é líder de $R_2$ .

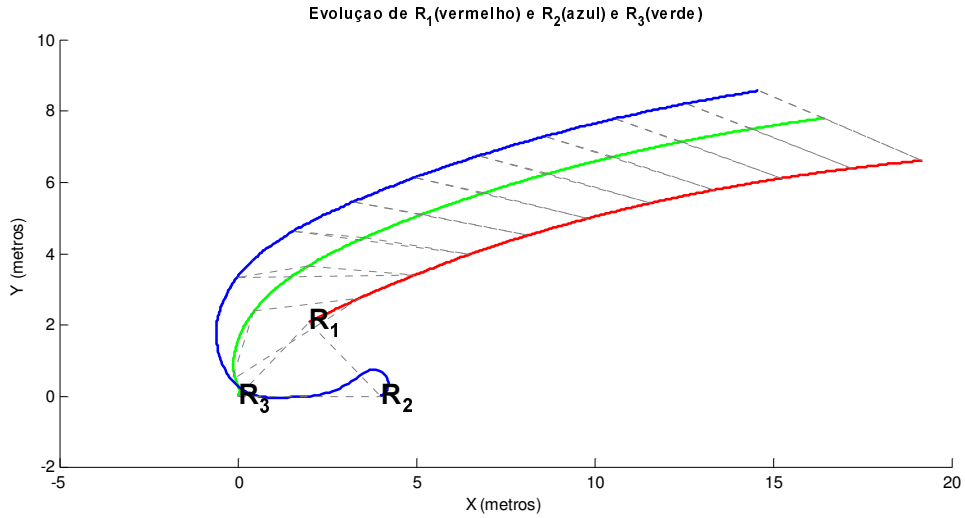


**Figura 21 – Simulação 7 ($SB_{32}C$: $l_{32}{}^{d}$ =2 , $\psi_{32}{}^{d}$ =150º ; $SB_{13}C$: $l_{13}{}^{d}$ =3 ; $\psi_{13}{}^{d}$ =150º )**

Alterando os ângulos $\psi$ para 180º, obteríamos uma fila com os três robots. Não é apresentada essa simulação, porque o percurso dos robots nessa situação sobrepõe-se, não sendo fácil perceber a formação observando uma figura. Já alterando para 90º, os robots andariam lado a lado, como se pode ver na seguinte simulação (Figura 22).

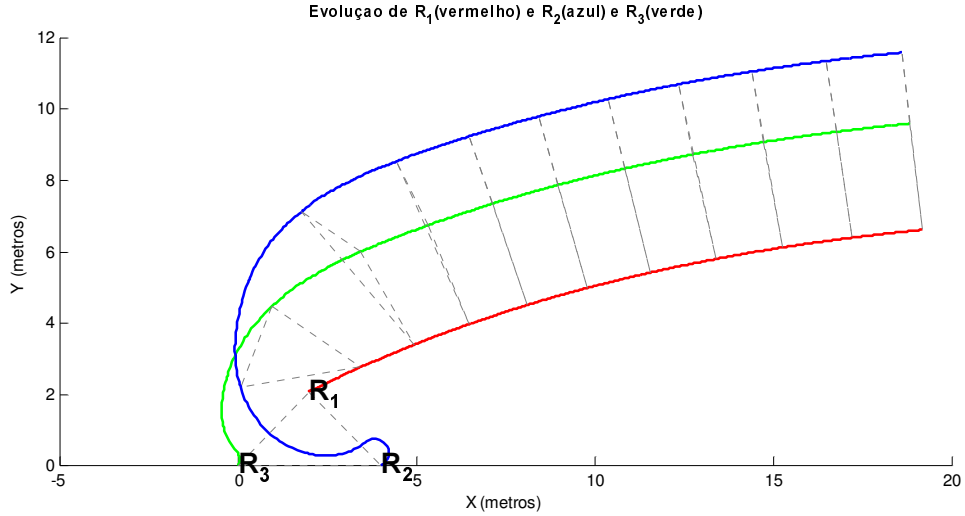


**Figura 22 – Simulação 8 ($SB_{32}C$: $l_{32}{}^{d}$ =2 , $\psi_{32}{}^{d}$ =90º ; $SB_{13}C$: $l_{13}{}^{d}$ =3 ; $\psi_{13}{}^{d}$ =90º )**

Conclui-se assim que a formação é mantida utilizando os controladores SBC e SSC, desde que a trajectória do líder seja bem comportada. De seguida são feitas simulações contemplando a presença de obstáculos.

## *4.2 Simulação do protocolo de coordenação de formação*

Nesta secção são simulados casos em que é utilizado o protocolo de coordenação de formações, estudado no capítulo 3, com a única diferença de que não foram considerados os limites dos sensores. Em todas as simulações seguintes são considerados obstáculos presentes no mundo, com o propósito de testar o controlador $l-\delta$.

Inicialmente o controlador escolhido é o SBC, e quando o robot seguidor chega a menos de uma determinada distância a um obstáculo, muda para o controlo SDC. Enquanto até agora eram controlados a distância e o ângulo relativos ao líder, agora são controlados a distância ao líder e a distância ao obstáculo.

Na Figura 23, pode-se observar a utilização típica deste controlador. A vermelho temos o percurso do líder $R_1$, a azul tracejado o percurso de $R_2$ se não tivessem sido considerados os obstáculos, e a azul a traço contínuo o percurso real de $R_2$, a evitar obstáculos.

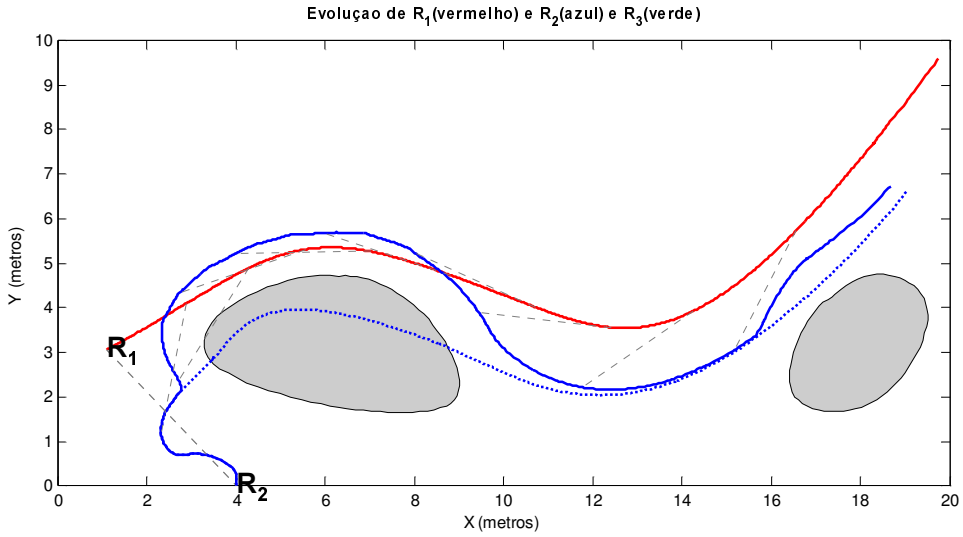


**Figura 23 – Simulação 9 ($SB_{12}C/S_0DC$: $l_{12}^{d}$ =3, $\psi_{12}^{d}$ =210º, $\delta_0$ =1)**

No início do trajecto, $R_2$ segue pelo caminho utilizando o controlador SBC. A partir do ponto em que detecta o obstáculo (no local onde a trajectória a azul se separa da trajectória a tracejado), é utilizado o controlador SDC. A evolução das distâncias $l_{12}$ e $\delta$ podem ser observadas na Figura 24. Como se pode verificar, o robot contorna o obstáculo mantendo uma distância a este praticamente constante (igual a $\delta_0$), ao mesmo tempo que segue $R_1$ a uma distância fixa. Após $R_2$ ultrapassar o primeiro obstáculo, regressa ao controlador SBC, tal como mostra a convergência da trajectória do robot para o percurso a tracejado. No entanto detecta um obstáculo uma segunda vez, e volta e trocar de controlador e

consegue evitá-lo. É de notar que, no primeiro obstáculo, se $R_2$ tivesse contornado o obstáculo pelo outro lado, não teria conseguido manter a distância constante a $R_1$, e o controlador falharia. Na seguinte simulação comprova-se isso.

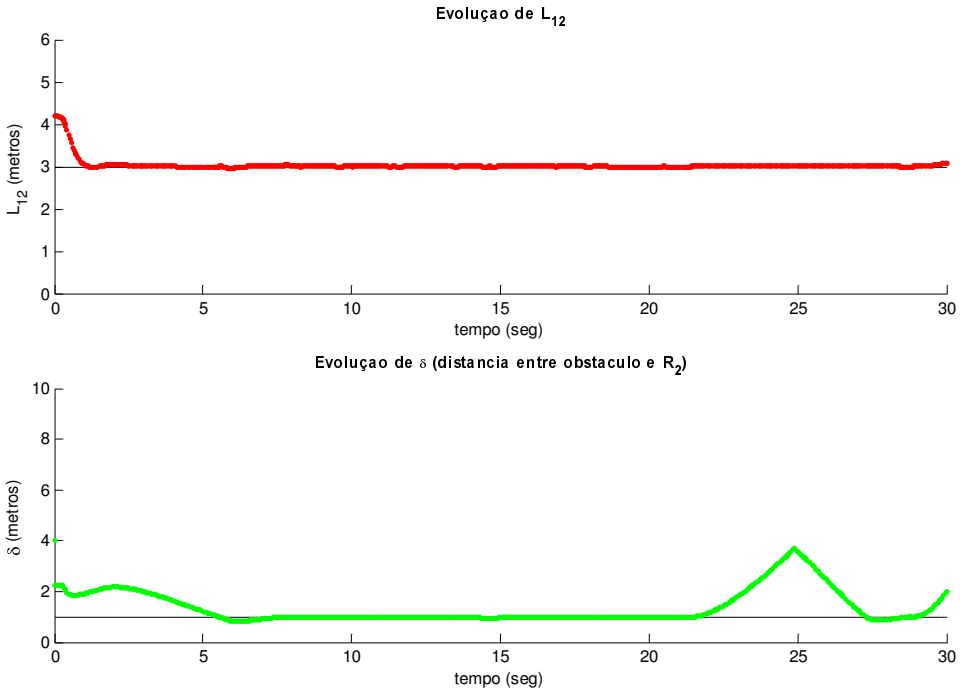


**Figura 24 – Simulação 9 - Evolução de $L_{12}$ e de $\delta$**

Mantendo a mesma formação desejada e o mesmo percurso do líder, alargou-se o primeiro obstáculo do percurso, de modo a ser impossível ao robot contorná-lo pelo mesmo lado. O resultado está na Figura 25. O percurso a tracejado representa o caminho tomado por $R_2$ na Simulação 9.

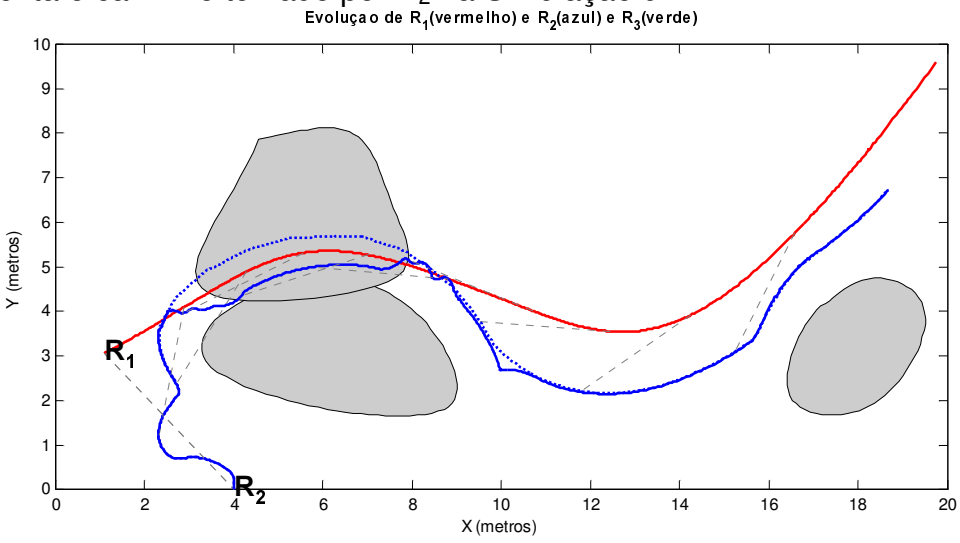


**Figura 25 – Simulação 10 - ($SB_{12}C/S_0DC$: $l_{12}^{d}$ =3, $\psi_{12}^{d}$ =210º, $\delta_0$ =1)**

Como se pode verificar, não conseguindo contornar o obstáculo ao mesmo tempo que segue o líder, o robot entra pelo obstáculo dentro, o que num caso real significaria uma colisão. No entanto, este caso nunca existiria na realidade, pois também o percurso do líder atravessaria o obstáculo. Cria-se, então, um pequeno corredor para o líder passar, e veja-se o resultado.

Na Figura 26 foi acrescentado um obstáculo, bloqueando o caminho que $R_2$ tomou na Simulação 9 (Figura 23), mas que deixa um espaço para o líder passar.

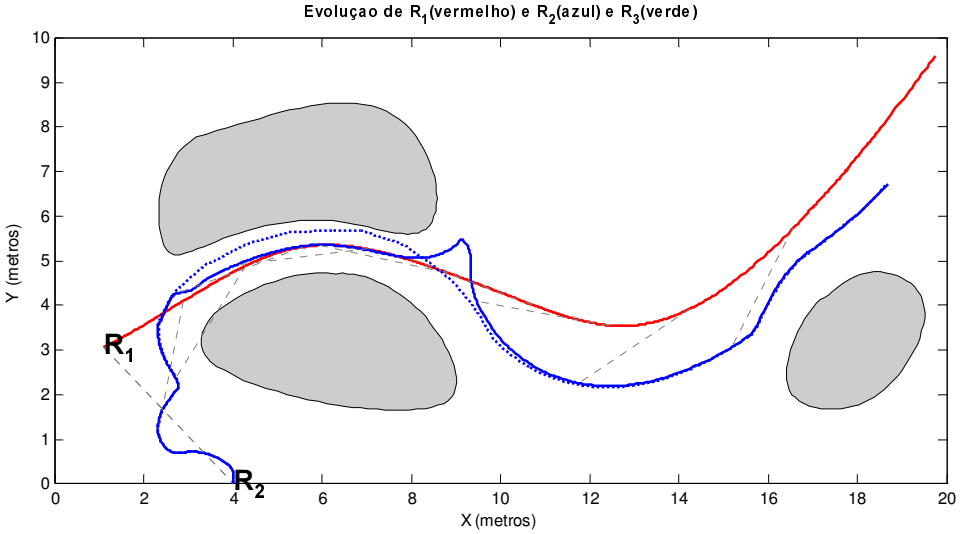


**Figura 26 – Simulação 11 - ($SB_{12}C/S_0DC$: $l_{12}{}^d$ =3, $\psi_{12}{}^d$ =210º, $\delta_0$ =1)**

O percurso a tracejado representa o caminho tomado por $R_2$ na Simulação 9, quando não havia o obstáculo a bloquear o caminho. Como se pode ver, esse antigo percurso levaria o robot a uma distância muito pequena do obstáculo. Em vez disso, $R_2$ segue pelo corredor gerado pelos dois obstáculos, numa trajectória bastante semelhante à do $R_1$, mas mantendo sempre a distância desejada a esse líder.

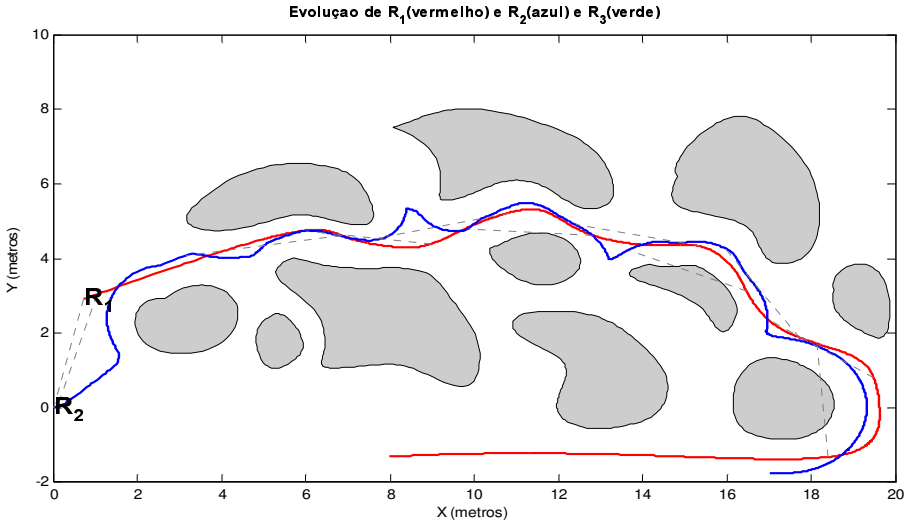


**Figura 27 – Simulação 12 ($SB_{12}C/S_0DC$: $l_{12}{}^d$ =3, $\psi_{12}{}^d$ =210º, $\delta_0$ =1)**

Um exemplo do mesmo tipo, mas mais complexo, é apresentado na Figura 27. Para esta simulação, vejam-se também as distâncias $L_{12}$ e de $\delta$ na Figura 28 (lembrar que $\delta$ é a distância ao obstáculo mais próximo, e não a um obstáculo em particular).

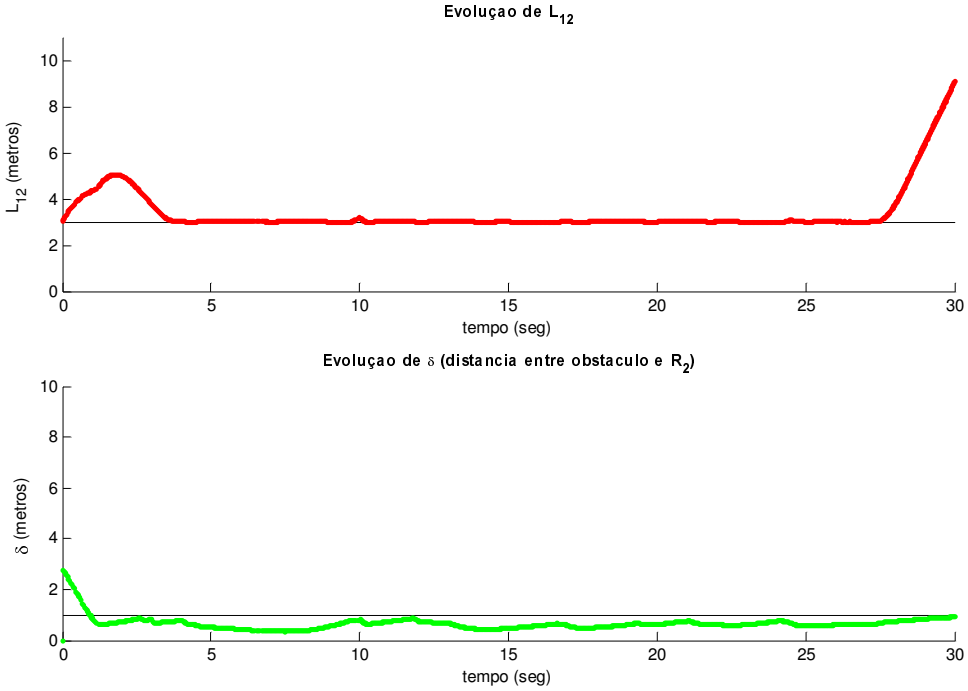


**Figura 28 – Simulação 12 - Evolução de $L_{12}$ e de $\delta$**

Vê-se que, devido à complexidade do ambiente nesta simulação, a distância ao líder tem mais perturbações que o normal, mas permanece praticamente constante. Já a distância ao obstáculo $\delta$ está quase sempre abaixo do desejável ($\delta_0$=1). Isto deve-se simplesmente ao facto de tal $\delta_0$ ser impossível para aquele ambiente, pois não existia espaço nos corredores formados pelos obstáculos para $R_2$ permanecer suficientemente afastado. Assim, tenta ficar o mais afastado possível. Note-se ainda o instante final, em que $R_1$ segue uma linha recta a uma velocidade muito maior que a máxima de $R_2$, que como resultado fica para trás (como se pode ver nos últimos segundos da evolução de $L_{12}$ na Figura 28).

Uma última simulação conta com os dois robots seguidores, ambos com o controlador SDC a seguirem $R_1$, e com um teste a uma configuração de obstáculos muito comum na literatura (Figura 29). Nesta simulação, é suposto os robots seguirem lado a lado. No entanto entram numa espécie de corredor, e são obrigados a mudar de formação. Caso ligeiramente diferente, passar-se-ia num corredor mais estreito, o que forçaria os robots a abandonarem a formação lado a lado, e os obrigaria a andar em fila. No entanto, deveria haver um mecanismo automático que escolhesse o líder mais adequado em cada momento. Se nessa situação ambos os robots continuassem com $R_1$ como líder,

ocorreria o risco de colisão entre eles, pois não teriam em conta a proximidade um com o outro.

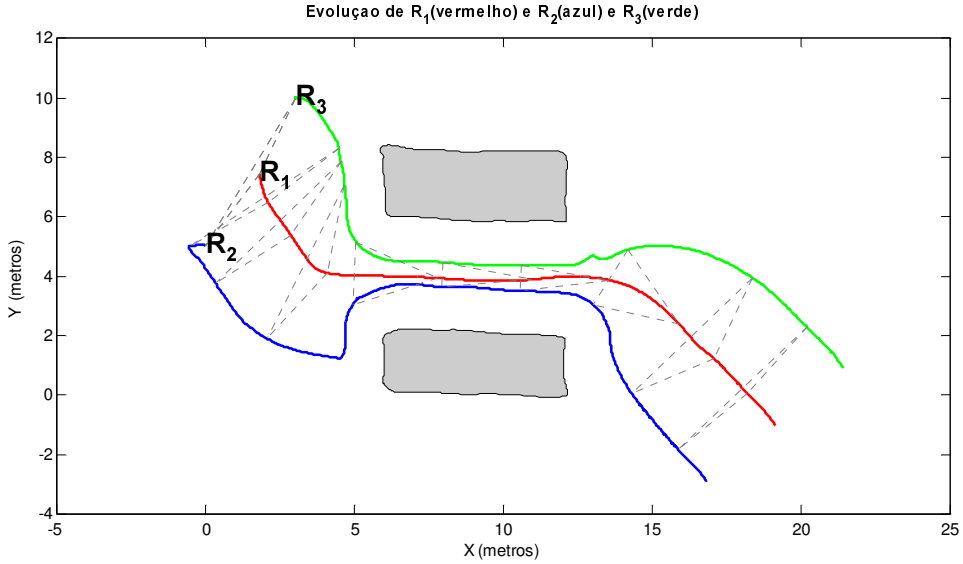


**Figura 29 – Simulação 13 - ($SB_{12}C/S_0DC$: $I_{12}{}^d$ =3, $\psi_{12}{}^d$ =270º, $\delta_0$ =1.5; $SB_{13}C/S_0DC$: $I_{13}{}^d$ =3, $\psi_{13}{}^d$ =90º, $\delta_0$ =1.5)**

Conclui-se então que os controladores estudados funcionam como previsto, e que o próximo passo é a sua implementação em robots reais, descrita no capítulo seguinte.

# 5. Implementação

## *5.1 Hardware/Software*

Na segunda fase deste trabalho, os algoritmos estudados foram implementados numa equipa de robots móveis pertencentes ao projecto Socrob. Entre as várias características dos robots referenciadas em [6], destacam-se aquelas que foram relevantes para este projecto:

- A equipa é formada por 4 robots não-holonómicos, modelo *Nomadic Super Scout II*. Cada robot contém os elementos básicos de um computador: *motherboard*, memória RAM e disco rígido. O robot move-se através de duas rodas independentes. Dos 4 robots, um deles está configurado fisicamente para ser guarda-redes nos jogos de futebol robótico, e como tal não era adequado a ser utilizado nas experiências de controlo de formações, pelo que foi apenas utilizado como robot substituto.
- Cada robot dispõe de comunicação através de uma *wireless Ethernet*, podendo assim comunicar com o resto da equipa e com computadores exteriores.
- Entre os sensores do Scout, foram utilizados os seguintes: sistema de odometria, sonares e câmara omnidirecional. Através das medições de odometria é possível obter a posição e orientação do robot; os sonares, em número de 14, conseguem localizar obstáculos e outros robots com alguma precisão até alguns metros de distância; a câmara omnidirecional está dependente da iluminação e da segmentação de cor, e consegue detectar outros robots, apesar de ter alguma incerteza nas medições.
- Os *Scouts* têm o sistema operativo Linux instalado (Suse 9.1), e a grande maioria do código do projecto é escrito na linguagem de programação C.

Devido a este trabalho funcionar paralelamente ao projecto Socrob (e haver a possibilidade de ser integrado no futuro), o *software* que controla os *Scouts* foi também utilizado como base do código para a implementação do controlo de formações. Funções como controlo dos motores, detecção de obstáculos utilizando os sonares, funcionamento da câmara omnidirecional e respectivas ferramentas, etc., já estavam disponíveis e foram utilizadas neste projecto, praticamente sem modificações relativamente ao código original. O projecto Socrob tem como base de funcionamento do controlo a existência de *plugins*, como por exemplo *getclose2bal, go, followball,* etc. A forma como a implementação foi abordada consistia em conceber um novo *plugin* respeitando a estrutura dos existentes, de modo a poder ser compatível e facilmente implementado no projecto principal, no futuro. Como tal foi criado o *plugin follow,* cujo código está no ficheiro follow.c . Pormenores acerca da execução deste *plugin* nos *Scouts* são descritos no Anexo C. A comunicação é efectuada através de uma memória partilhada por todos os robots, denominada *blackboard.*

## *5.2 Localização dos robots*

Nesta implementação os robots lêem a odometria das rodas, e comunicam-na à equipa através do *blackboard*. Se por um lado basear a localização dos robots na odometria é simples e eficaz, por outro lado traz alguns problemas, nomeadamente o aumento do erro de posição com o caminho percorrido. Por exemplo, um erro inicial de orientação de 5º significa um erro de posição de 17.5 centímetros, depois de percorridos dois metros numa qualquer direcção. Para além dessa dependência da postura inicial, há que referir os erros que surgem devido a movimentos bruscos dos robots, ou quando deslizam ou ficam bloqueados. As rodas continuam a rodar, mas o robot não move, ou pelo menos não a distância dada pela odometria. Isto faz com que este método apresente algum erro depois de um percurso longo. No projecto Socrob, este problema foi resolvido utilizando as câmaras do robot para localizar a postura no campo de futebol robótico. Este é um caso específico, dado que este é um ambiente previamente conhecido e com marcas distintas, como as linhas do campo, as balizas, as várias cores associadas a vários locais, etc. Os referenciais dos robots também são relativos ao próprio campo.

Apesar de a implementação deste trabalho ter ocorrido no mesmo laboratório onde se localiza este campo, optou-se por não utilizar este método de auto-localização, com a justificação de manter os algoritmos funcionais independentemente do ambiente. Além disso, e por esse método por vezes ter resultados com erros muito grandes, era possível o surgimento de movimentos muito bruscos nos robots, dado encontrarem-se de repente longe de onde pensavam estar. Apesar disso, foram feitos testes utilizando este método, com resultados não muito adequados ao controlo de formações. Porém, seria trivial adicionar este método de localização numa futura versão do *plugin* a funcionar activamente no projecto Socrob. Assim, é somente necessário que todos os robots que vão integrar a formação sejam inicializados no mesmo referencial. Como tal, nesta implementação o único meio de localização dos robots reside na leitura da odometria. Existe então a necessidade de corrigir o erro gerado por este método. Foram considerados algumas formas de o fazer, descritas de seguida:

- A existência de uma câmara fixa exterior aos robots que visualizasse a área em que estes se encontrassem, permitiria obter posições relativas, e portanto corrigir o erro da odometria, mas mais do que isso, tornaria a própria leitura da odometria redundante, já que a posição e orientação poderiam ser obtidas pela câmara (se necessário recorrendo a estimadores de estado). Por outro há o custo de ter uma câmara adicional e a possibilidade dos robots saírem do alcance da câmara, além da perda de autonomia da equipa. Este método poderia ser útil para obter informações sobre robots móveis, mas para o controlo de formações não é adequado.

- A utilização de sonares poderia servir para obter a distância a que estão os outros robots, distância essa que, após comparada com o mesmo valor obtido pela odometria dos outros robots, possibilitaria uma correcção de posição. Como desvantagens, o facto de não ser possível corrigir também a orientação (apenas a posição), e a existência de erros devido à falta de exactidão dos sonares. Naturalmente que seria aconselhável utilizar um método de tratamento dos dados, filtrando de alguma forma os valores errados.
- Utilizar a câmara omnidirecional para localizar os outros robots é um método muito utilizado, como por exemplo em [2] e em [3]. No caso presente, a câmara não serviria para obter posições, mas sim para corrigir as posições obtidas pela odometria. O método consistiria em procurar na imagem da câmara um robot na zona onde supostamente ele estaria (o que consistiria numa mancha preta dado que os robots são dessa cor), encontrar o ponto que correspondesse ao centro do robot, comparar essa posição com a calculada através da odometria e fazer as correcções necessárias. Novamente a orientação não seria corrigida (a não ser que fossem utilizados estimadores como o filtro de Kalman estendido). Este compromisso entre a informação obtida pela câmara e a obtida pela odometria consiste no fundo numa fusão sensorial. Existem bastantes formas de fazer esta fusão, mas um algoritmo complexo nesta área não faz parte do objectivo do trabalho. Optou-se antes por um algoritmo simples. Periodicamente são identificados (na imagem da câmara omnidirecional) os robots, representados por zonas a preto com uma certa dimensão mínima. Se a luz ambiente for pouca, ou se a superfície apresentar zonas escuras, então é possível que seja identificado um robot erroneamente. Para evitar tais situações, seria desejável utilizar os sonares para confirmar que existe realmente um objecto na posição obtida pela câmara. Chamemos a este objecto um robot virtual, ou seja, um objecto que *pode* ser um robot real. O caso do robot virtual ser na verdade um obstáculo de cor preta não é considerado. De seguida procura-se na vizinhança de cada robot virtual por um robot (considerando as medidas da odometria). Se existir, então é porque a posição dada pela odometria está errada, e a verdadeira é a dada pela câmara. A vizinhança considerada tem de ser suficientemente pequena para não trocar a identidade dos robots. Na Figura 30 vê-se um exemplo: $R_1$ utiliza a câmara omnidirecional e encontra $R_{cam}$. De seguida determina a posição de robots localizados nessa zona, utilizando a odometria, e obtém a posição de $R_{od}$. Se $R_{od}$ e $R_{cam}$ estiverem a menos de um valor *r*, então é porque a odometria contém um erro, e a posição real do robot é obtida pela câmara. Uma vez que todas estas posições são relativas a $R_1$, é possível então corrigir a sua posição. Por outro lado, $R_1$ pode também ter erro na sua localização, levando a que este se propague à determinação da posição do outro robot num referencial do mundo.

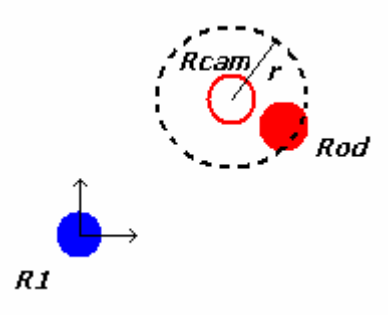


**Figura 30 – Exemplo da correcção de posição**

Partindo do princípio que a posição determinada pela câmara é correcta, ou que pelo menos tem um erro menor que a posição dada pela odometria, então este método tem a sua utilidade. Porém, não serve para corrigir a orientação relativa do robot, apenas a sua posição. Este método foi implementado no código do *plugin follow*, mas devido a um comportamento diferente do previsto não foi incluído na versão final do código, sendo a odometria o único sensor considerado para obter a posição.

- Uma maneira de evitar a necessidade de confirmação da identidade do robot encontrado pela câmara do método anterior, consiste em cada robot utilizar uma marca visual distinta, nomeadamente uma faixa colorida ([3]). Havendo três robots e três cores diferentes, seria possível identificar imediatamente a posição de cada um observando a câmara de cima. Este método é muito mais simples que o anterior, havendo porém a necessidade de identificar cada robot com uma marca, o que pode ser complicado para uma equipa maior que a utilizada. Existe a possibilidade de utilizar as marcas de cor não como factor decisivo de identificação de robots, mas apenas como ajuda.
- O método de localização cooperativa, estudado em [3], permite aos robots determinarem a sua posição e orientação relativa, determinando apenas a direcção relativa dos outros membros da equipa, usando para isso uma câmara omnidirecional. Em [3] um computador exterior obtinha toda a informação de todos os robots, e determinava a postura da equipa, sendo neste caso o controlo dos robots não autónomo. No caso presente, seria necessário cada robot saber todas as posições relativas entre a equipa, de modo a determinar a própria posição. Este método necessita de pelo menos 3 robots, e tem como grande vantagem já não ser necessário ter um estimador para obter a orientação.

As figuras da secção seguinte, respeitantes às experiências com os robots reais, contêm as posições destes tendo em conta os valores dados pela odometria (foram gravados num ficheiro os valores dados por este sensor, para posteriormente visualizar as trajectórias dos robots), pelo que não é levado em conta o erro associado aos sensores. No entanto foram feitas gravações de vídeo que demonstravam o erro de odometria a aumentar com o percurso efectuado.

## *5.3 Experiências e resultados*

Foram então realizadas algumas experiências nos robots reais, de modo a ser perceptível o desempenho dos controladores e as diferenças em relação à simulação. As experiências denotam o funcionamento dos controladores isoladamente, sem o protocolo de coordenação de formações estar activo.

O primeiro teste foi feito utilizando um robot líder (a vermelho) e um robot seguidor (a azul) utilizando um controlador SBC. O percurso dos robots pode ser observado na Figura 31. O início do percurso é marcado por uma marca redonda, e a formação dos dois robots é evidenciado a tracejado. Nota: enquanto que nas simulações foram consideradas velocidades lineares apenas positivas, nos seguintes testes foram também permitidas velocidades negativas (o robot a andar para trás). Isto devido somente a uma questão de clareza das figuras do capítulo 4, pois com velocidades negativas as trajectórias eram bastante mais confusas.

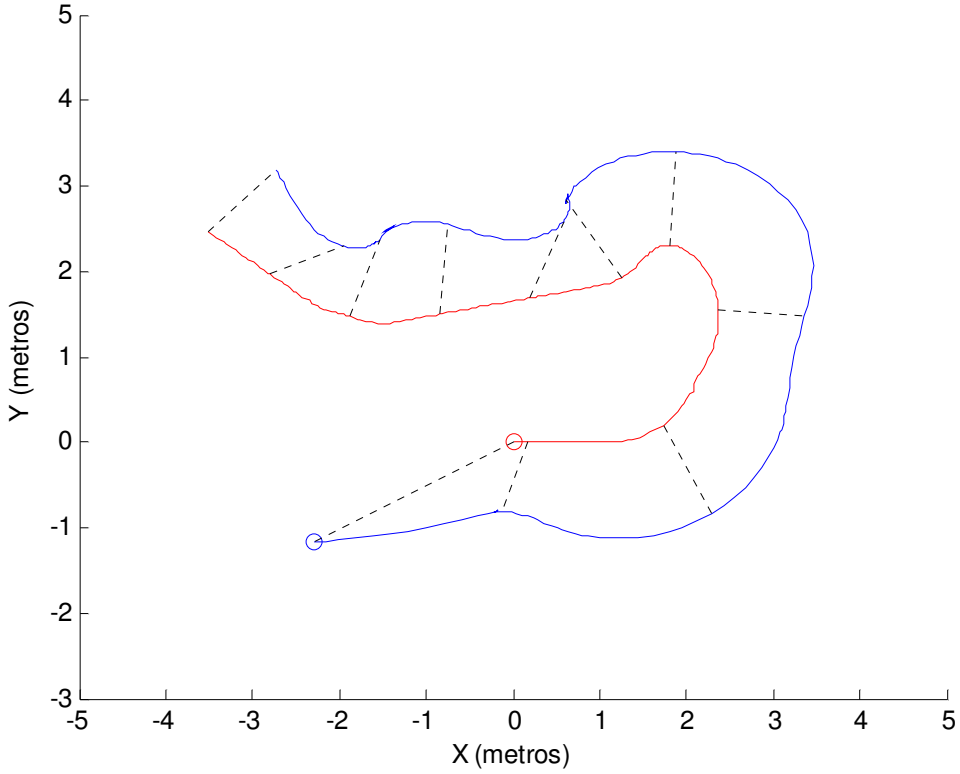


**Figura 31 – Teste 1 – SB$_{ij}$C: $l_{ij}^{d}$ =1, $\psi_{ij}^{d}$ = - 90º**

**Percurso do líder (vermelho) e do seguidor (azul)**

Neste teste, o robot líder ficou parado por uns instantes, dando oportunidade ao seguidor para se colocar em formação. De seguida a formação começa a mover-se, e o robot líder faz uma curva para a esquerda. Na Figura 32 e na Figura 33

podem-se observar a evolução da distância entre os robots e do ângulo $\psi$. Nota: nestas figuras o eixo dos xx tem a escala em segundos. Esta escala é apenas uma estimativa da relação tempo/ciclos de amostragem (15 ciclos de amostragem = 1 segundo).

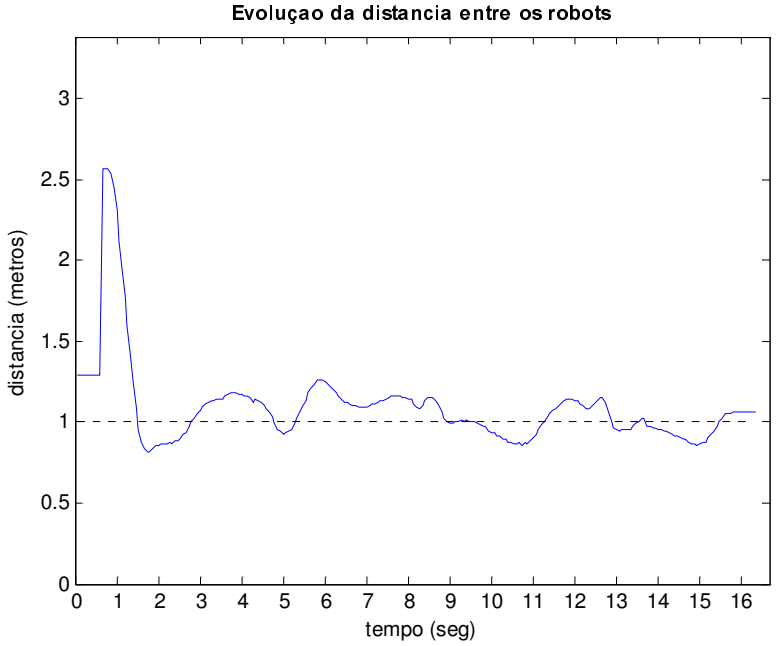


**Figura 32 – Teste 1 – $SB_{ij}C$: $l_{ij}^{d}$ =1 , $\psi_{ij}^{d}$ = - 90º; Evolução da distância entre os robots $l_{ij}$**

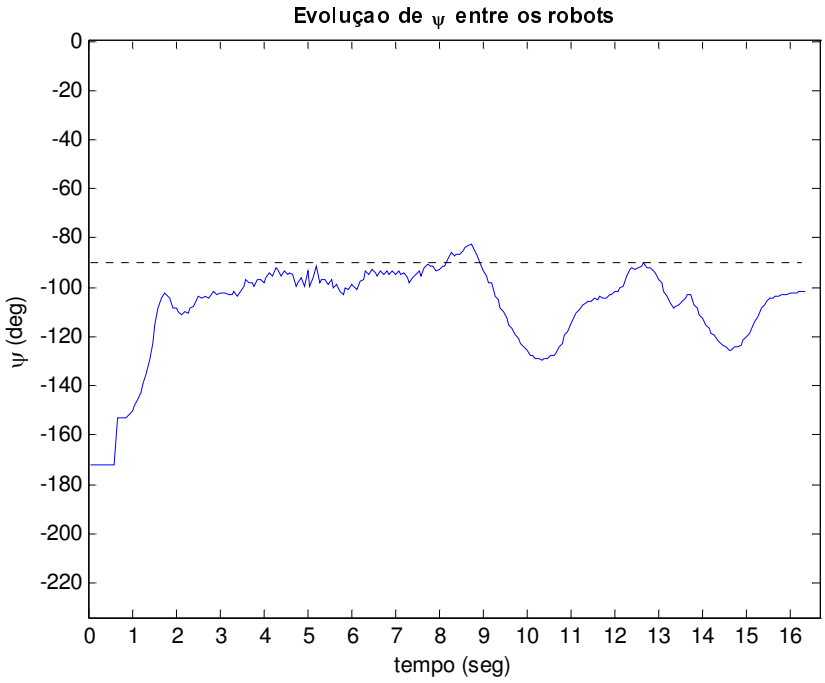


**Figura 33 – Teste 1 – $SB_{ij}C$: $l_{ij}^{d}$ =1 , $\psi_{ij}^{d}$ = - 90º; Evolução do ângulo $\psi_{ij}$**

Como se pode ver, é possível identificar a situação inicial em que o seguidor vai para a posição desejada enquanto o líder está parado. De seguida os valores oscilam na vizinhança da referência, havendo porém muitas perturbações.

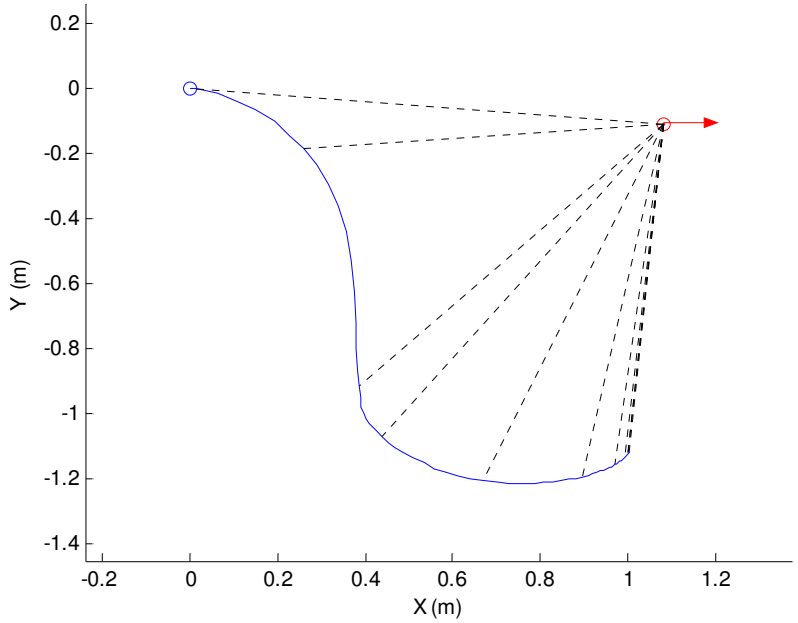


**Figura 34 – Teste 2 – $SB_{ij}C$: $l_{ij}^{d}$ =1 , $\psi_{ij}^{d}$ = - 90º; Evolução do ângulo $\psi_{ij}$**

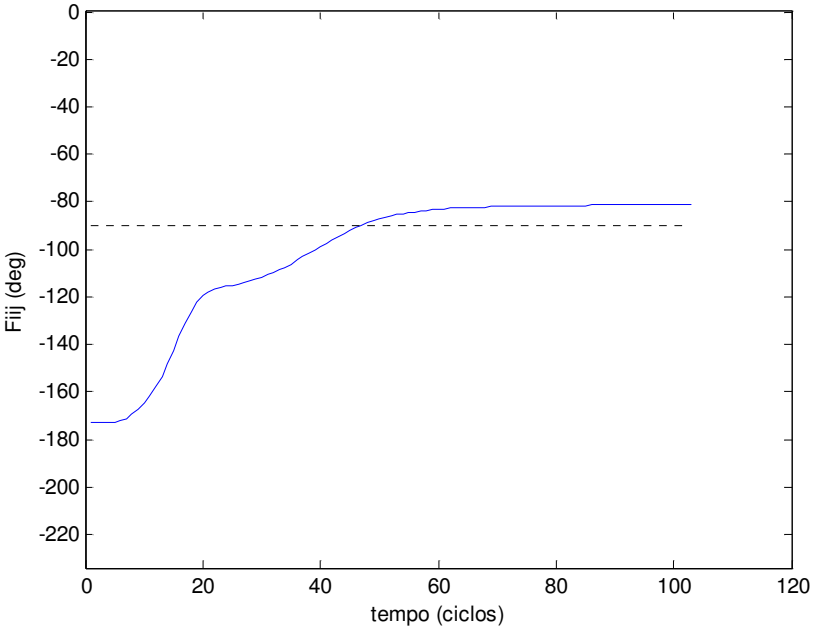


**Figura 35 – Teste 2 – $SB_{ij}C$: $l_{ij}^{d}$ =1 , $\psi_{ij}^{d}$ =-90º; Evolução do ângulo $\psi_{ij}$**

Foi feito um segundo teste para observar um pormenor do controlo de $\psi_{ij}$. Na Figura 34 observa-se o caso em que o líder está parado e o seguidor coloca-se em formação. No entanto, verifica-se pela Figura 35 que o seguidor não fica exactamente onde deve. Isto deve-se ao facto de que, quando o erro é baixo, a actuação resultante é proporcionalmente baixa. Logo, a existência de uma *dead-*

*zone* nos motores pode fazer com que haja uma situação como a verificada. Em termos práticos, nota-se que lentamente esse erro iria diminuir, o que significa que este fenómeno pode ocorrer devido às baixas actuações dos motores, obtidas devido ao baixo erro. Uma hipótese seria a alteração do ganho do controlador para erros pequenos.

De seguida foi testado o algoritmo SSC, utilizando 3 robots. O robot $R_1$ (a azul) segue o líder $R_2$ (a vermelho) com um controlador SBC, e o segundo seguidor $R_3$ (a verde) segue os dois com o controlador SSC. Na Figura 36 pode-se ver o percurso da formação.

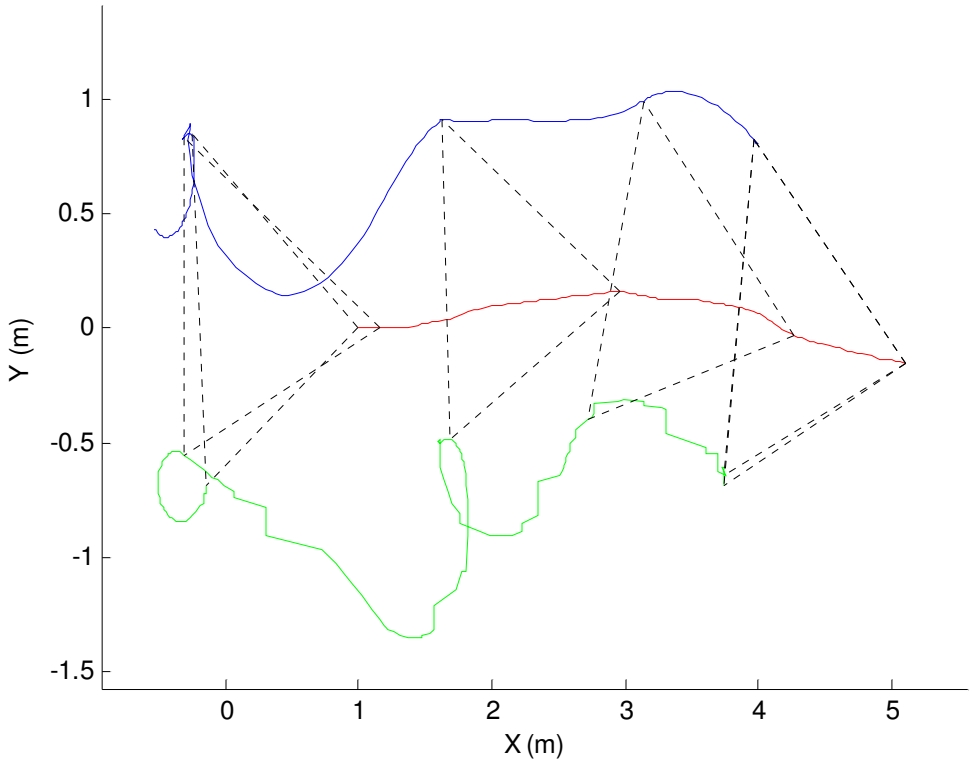


**Figura 36 – Teste 3 – ($SB_{12}C$: $l_{12}^d$ =1.5 , $\psi_{12}^d$ =150º ; $S_{13}S_{23}C$: $l_{13}^d$ =1.5 ; $l_{23}^d$ =1.5 )**

**Percurso dos robots**

Como se pode verificar a distância entre $R_1$ e $R_3$ (Figura 37) e entre $R_2$ e $R_3$ (Figura 38) mantêm-se a oscilar em torno do valor esperado, mas no entanto a formação é mantida. As diferenças destes testes em relação às simulações devem-se primariamente ao condicionamento gerado pelos sensores, pela comunicação e pelos actuadores. Existem erros ou atrasos temporários associados a cada um destes factores, fazendo com que as condições não sejam as perfeitas. Além disso, podem haver termos nas expressões que não permitam a simplificação prevista.

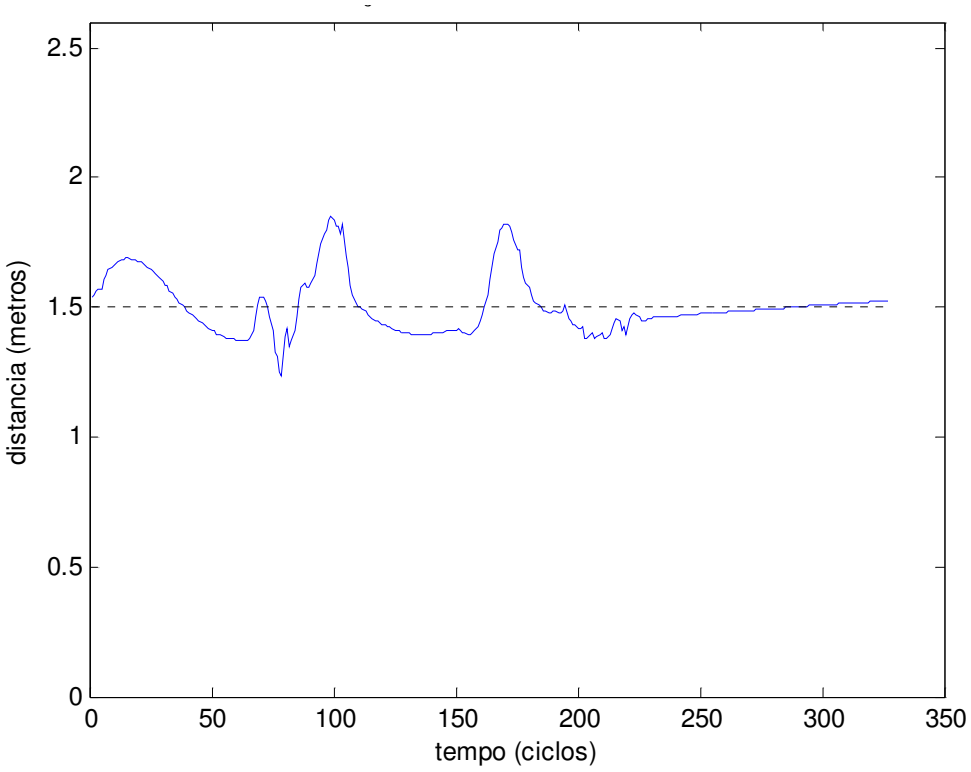


**Figura 37 – Teste 3 – ($SB_{12}C$: $l_{12}^{d}$ =1.5 , $\psi_{12}^{d}$ =150º ; $S_{13}S_{23}C$: $l_{13}^{d}$ =1.5 ; $l_{23}^{d}$ =1.5 ) – Evolução da distância entre $R_1$ e $R_3$**

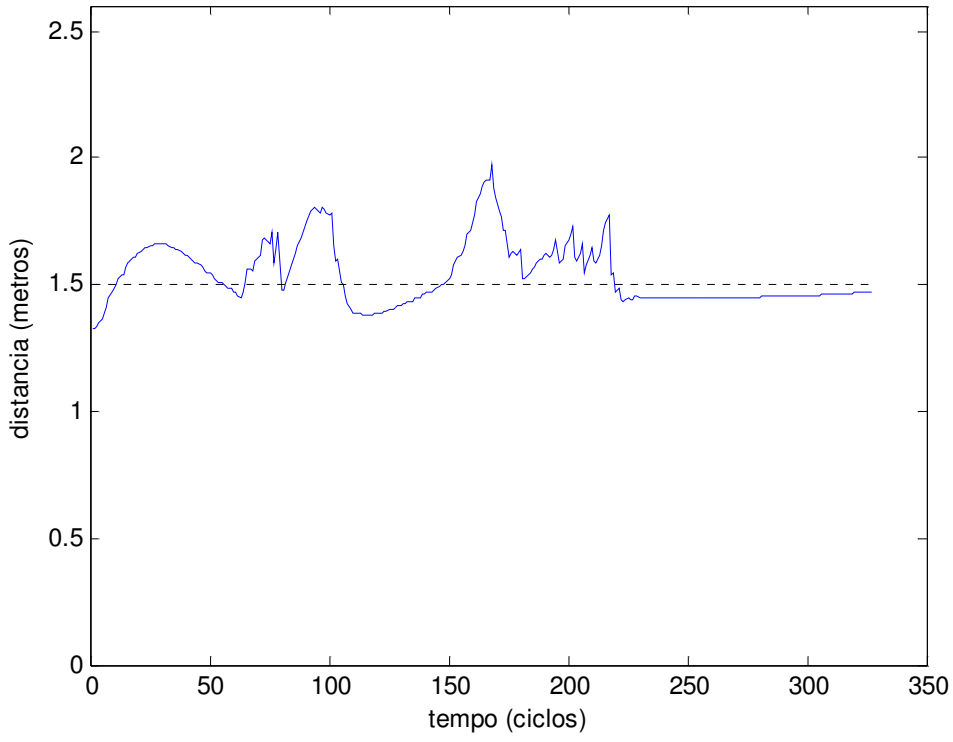


**Figura 38 – Teste 3 – ($SB_{12}C$: $l_{12}^{d}$ =1.5 , $\psi_{12}^{d}$ =150º ; $S_{13}S_{23}C$: $l_{13}^{d}$ =1.5 ; $l_{23}^{d}$ =1.5 ) – Evolução da distância entre $R_2$ e $R_3$**

# 6. Conclusão

Este trabalho consistiu num relatório de Trabalho Final de Curso no campo do Controlo de Formações de Robots Móveis. Nele foi referido o trabalho efectuado, os métodos e algoritmos estudados, o processo de simulação e o processo de implementação em robots reais.

No capítulo 2 estudaram-se vários controladores de formação. Para um robot manter uma formação, é necessário que conserve uma certa distância e ângulo relativo a um líder. Utilizando um controlador SBC é possível manter essa formação. No caso de aparecer um segundo seguidor, pode surgir o perigo de colisão. Nesta situação, deve ser utilizado um controlador SSC, que mantém distâncias mínimas entre todos os robots. Estes dois controladores são os básicos no controlo de formações, sendo possível estender uma formação a *n* robots, cada qual tendo apenas um ou dois líderes. Para o caso em que um obstáculo aparece no percurso, é necessário que o robot seguidor continue a seguir o líder, enquanto ao mesmo tempo mantém uma distância de segurança ao obstáculo, contornando-o. Utilizando um controlador SDC é possível cumprir estas especificações.

No capítulo 3 foi abordado um algoritmo capaz de escolher o controlador mais indicado em cada situação. O protocolo de coordenação de formações divide-se em duas partes, respectivamente com e sem obstáculos. A partir de alguma informação sobre o mundo, este protocolo escolhe o modo de controlo da formação mais adequado. Em vez de este protocolo escolher também os parâmetros de cada controlador, ficou decidido que escolheria apenas o controlador em si, e os parâmetros teriam valores por defeito.

No capítulo 4 foi referido o processo de simulação em ambiente *Matlab*. É descrito o simulador desenvolvido, e são apresentadas várias simulações que demostram o funcionamento teórico dos controladores estudados no capítulo 2. As conclusões deste capítulo foram que é possível manter uma formação estável com qualquer um destes controladores (e mesmo com comutações entre eles) se a trajectória do líder for bem-comportada (isto é, não ter velocidades muito elevadas, não fazer curvas muito apertadas, não atravessar obstáculos).

No capítulo 5 é explicado o modo utilizado para implementar os vários algoritmos em robots reais. Foram utilizados os robots da equipa de futebol robótico do projecto Socrob, que já contavam com toda uma base de *software,* em cima da qual foi desenvolvido um *plugin* que controlava as formações entre os robots. Este *plugin* foi programado em linguagem C, e incluía não só os vários controladores de formação, mas também o protocolo de coordenação de formações. Dos três controladores implementados, apenas dois obtiveram resultados positivos, ficando o controlador SDC por ficar a funcionar.

Devido à forma como foi desenvolvido o código utilizado na implementação, é possível uma integração do controlo de formações no código principal do projecto Socrob. Sendo um projecto de futebol robótico, poderá fazer sentido manter formações entre os jogadores enquanto estes se movimentam. Os adversários poderão ser considerados obstáculos, sendo portanto evitados. No entanto, apesar de o *plugin* ser facilmente implementado no jogo, os algoritmos implementados não dependem do jogo, ou seja, não é necessário os robots estarem dentro do campo de futebol robótico para poderem seguir em formação. Mas por outro lado, os algoritmos poderiam ser adaptados especificamente para ter um melhor desempenho no ambiente do futebol robótico.

Os novos robots holonómicos que estão a ser construídos pela equipa Socrob poderão vir a ser uma plataforma adequada para a continuação deste trabalho. Para isso seria necessário adaptar as expressões das velocidades para robots holonómicos. Assim, em vez de velocidade linear e angular, existem velocidades linear em x e linear em y, bem com velocidade angular. Tudo leva a crer que as expressões para robots holonómicos serão mais simples que para os robots não-holonómicos, devido à ausência de funções trigonométricas.

Concluindo, no geral pode-se considerar que este trabalho foi bem sucedido, já que foram estudados e simulados três controladores diferentes possibilitando formações com três robots ao mesmo tempo que se evitam obstáculos, e desses três foram ainda implementados dois controladores nos robots reais, tornando possível formações, ainda que sem o suporte para obstáculos. A área do controlo de formações revela-se muito interessante dentro da robótica móvel, e deixa a prever um futuro em que a cooperação entre veículos automatizados possa ser um processo mais eficiente e simples.

# Referências

[1] R. Fierro, P. Song, A. Das , and V. Kumar, " Cooperative control of robot formations ", Submitted to the Kluwer Series on Applied Optimization , March 2001
[2] A. Das, R. Fierro V. Kumar, J. Ostrowski, J. Spletzer and C. Taylor, “A Framework for Vision Based Formation Control”, IEEE Transactions On Robotics And Automation, Vol. XX, No. Y, Month 2001
[3] J. Spletzer, A.K. Das, R. Fierro, C.J. Taylor, V. Kumar, and J.P. Ostrowski, "Cooperative localization and control for multi-robot manipulation", IROS 2001, Hawaii, USA Dec 2001.
[4] J. Spletzer and C. J. Taylor, "Sensor Planning and Control in a Dynamic Environment", IEEE International Conference on Robotics and Automation, Washington DC, USA, May 2002
[5] Jaydev P. Desai, Jim Ostrowski and Vijay Kumar, "Controlling formations of multiple mobile robots". IEEE International Conference on Robotics and Automation, Belgium, May 1998
[6] http://socrob.isr.ist.utl.pt
[7] http://omni.isr.ist.utl.pt/~mir/cadeiras/robmovel/Kinematics.pdf
[8] http://www.grasp.upenn.edu/

# Anexo A – Actuações nos motores

Nesta secção são obtidas as expressões das velocidades obtidas pelos controladores, de modo a poderem ser actuadas em motores em robots não holonómicos, ou seja, a cinemática dos robots ([7]). A expressões das velocidades estão em 2.2, 2.3 e 2.4, respectivamente para os controladores $l-\psi$, $l-l$ e $l-\delta$.

Os robots não holonómicos foram os utilizados para implementar os algoritmos. Seja $\omega_R$ a velocidade da roda direita e $\omega_L$ (velocidades angulares das rodas). As velocidades das rodas relacionam-se com as velocidades linear $(v)$ e angular $(\omega)$ da seguinte forma:

$$v = \frac{R}{2}\left(\omega_R + \omega_L\right) \tag{A.1}$$

$$\omega = \frac{R}{D}\left(\omega_R - \omega_L\right) \tag{A.2}$$

em que R é o raio da roda, e D é a distância entre as rodas. Manipulando (A.1) e (A.2), obtêm-se as velocidades angulares das rodas:

$$\omega_L = \frac{2v - D\omega}{2R} \tag{A.3}$$

$$\omega = \frac{2v + D\omega}{2R} \tag{A.4}$$

# Anexo B – Utilização do Simulador

Neste anexo é descrito o funcionamento do simulador desenvolvido para poder estudar os controladores de formação. O simulador foi desenvolvido em *Matlab* 6.5 (R13), e em *Simulink* 5.0 (R13). Foi escolhido este ambiente devido às ferramentas existentes neste *software,* nomeadamente o *Simulink*, que serve para modelar e simular vários tipos de modelos, como por exemplo, estruturas de controlo como a estudada neste trabalho.

O simulador é composto por sete ficheiros, descritos de seguida:

- robsim.m é o ficheiro que contem o *script* principal do simulador, e é o ficheiro a executar na *prompt* do *Matlab*.
- sistema.mdl é o modelo de *Simulink* que simula a formação de robots. Existe também o ficheiro sistemaR12.mdl, compatível com a versão anterior do *Simulink*. Para utilizar esta versão, é necessário alterar o nome do sistema no ficheiro simul.m (ver à frente).
- traj_gen.m lança uma janela preparada para receber do utilizador a trajectória do líder da formação, bem como os obstáculos do percurso.
- simul.m prepara o sistema para ser simulado, e expõe os resultados sob a forma de gráficos com a evolução da formação e a evolução dos erros de controlo. Contém também no início do ficheiro uma série de parâmetros da simulação que podem ser aqui alterados.
- obs.m é um *script* lançado de dentro do modelo *Simulink*, que serve para calcular o ponto do obstáculo mais próximo do robot.
- traj.mat é um ficheiro que contém o vector que descreve a trajectória do líder, que é utilizada pelo simulador.
- obs.mat contém um vector com os vários pontos dos vários obstáculos.

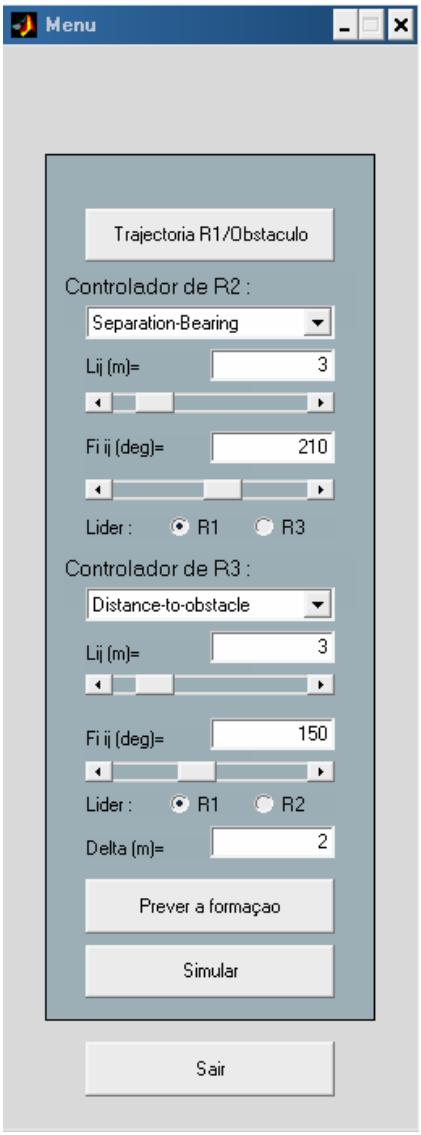


**Figura 39 – Menu do Simulador**

Para iniciar o simulador, é necessário mudar a directoria de trabalho do *Matlab* para a directoria onde estão localizados todos estes ficheiros. De seguida lança-se a janela que contém as várias opções do simulador, escrevendo *robsim* no *prompt* do Matlab. O menu (Figura 39) tem as seguintes opções: alteração da trajectória do líder $R_1$ e dos obstáculos; escolha do controlador e dos respectivos parâmetros de $R_2$ e $R_3$; escolha do líder de $R_2$ e de $R_3$, no caso de o controlador escolhido só necessitar de um controlador (SBC); alteração do parâmetro $\delta_0$ (distância desejada ao obstáculo) para ambos os robots; previsão da formação de acordo com as opções correntes. O botão 'Simular' inicia o processo de simulação, e no final são exibidas três janelas: uma para cada robot, com a evolução dos dois valores a controlar para cada controlador, e uma com o percurso de cada um dos robots. É possível também desactivar $R_2$ ou $R_3$ da simulação, escolhendo para isso 'Desactivado' na lista de controladores.

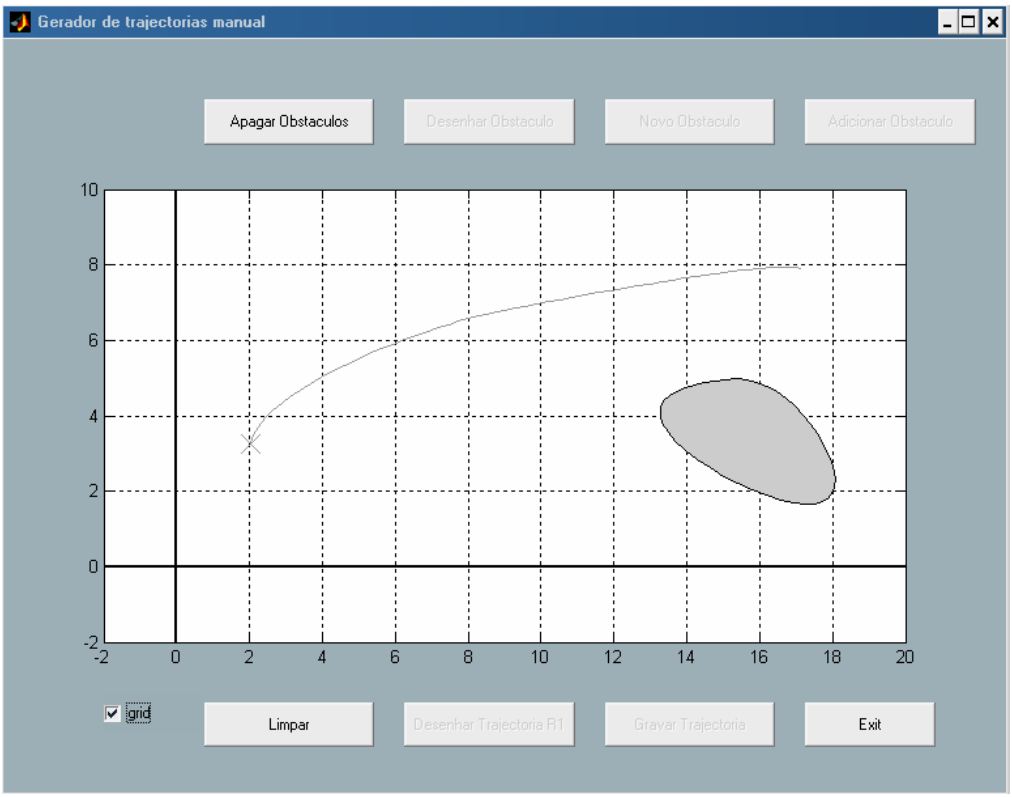


**Figura 40 – Editor de trajectória do líder e dos obstáculos**

O gerador de trajectória do líder $R_1$ e dos obstáculos (Figura 40) tem as seguintes opções:

- Para editar a trajectória de $R_1$: clicar no gráfico o ponto inicial do percurso; de seguida clicar vários pontos no mesmo gráfico (pontos a vermelho) ao longo do percurso desejado (é possível criar uma trajectória só com um ponto, que representa o robot parado); ao premir em 'Desenhar Trajectória R1', é criado um *spline* com base nos pontos inseridos

(resultado a azul); para alterar a trajectória da actual (a cinzento) para a trajectória a azul, premir em 'Gravar Trajectória' (nota: a orientação inicial de $R_1$ é sempre de 0º); para cancelar e apagar os pontos inseridos, premir em 'Limpar'.

- Para editar o obstáculo a evitar pelo controlador SDC: clicar no gráfico vários pontos, formando um obstáculo (pontos a vermelho); ao premir em 'Desenhar Obstáculo', é criada uma área fechada limitada por pontos azuis (novamente criada utilizando *splines*); para alterar o obstáculo actual (a cinzento), e substituí-lo pelo novo obstáculo a azul, premir em 'Novo Obstáculo'; para cancelar e apagar os pontos inseridos, premir 'Limpar'. É possível criar uma 'parede', criando um obstáculo com dois pontos.
- Para adicionar mais um obstáculo: seguir os mesmos passos da opção anterior, mas em vez de premir 'Novo Obstáculo', premir 'Adicionar Obstáculo'; podem ser adicionados tantos obstáculos quantos os necessários à simulação.
- Para apagar todos os obstáculos: premir 'Apagar Obstáculos'; o gráfico fica sem obstáculos, e podem ser inseridos novos obstáculos.

O modelo sistema.mdl contém 4 blocos principais: um bloco que trata a trajectória de $R_1$ inserida pelo controlador, e que calcula a postura para cada momento; dois blocos, um para cada seguidor $R_2$ e $R_3$, que simula o robot em si, os motores e a cinemática do mesmo (incluindo as saturações das actuações nos motores); as velocidades linear e angular de cada robot são limitadas, para simular os motores verdadeiros; considera-se sempre a velocidade linear também sempre positiva. Estes blocos (idênticos) têm como entrada a informação do(s) líder(es) ou obstáculos e os vários parâmetros escolhidos pelo utilizador no menu, e tem como saída a posição e velocidade do robot; finalmente o último bloco diz respeito ao robot virtual $R_0$, que representa o obstáculo. Este último bloco só está activo no caso do controlador escolhido ser SDC.

Este simulador tem como limitações:

- Devido a pormenores do funcionamento do simulador e do próprio *Simulink*, não é possível existir um *live loop* de líderes/seguidores. Ou seja, $R_2$ não pode seguir $R_3$ se este já estiver a seguir $R_2$. Esta limitação não se verifica na implementação real, mas em termos práticos o resultado pode ser imprevisível.
- O simulador só está preparado para funcionar com três robots, tendo um deles ($R_1$) a trajectória definida exteriormente pelo utilizador. A expansão para *n* robots seria possível, dado que em termos de modelo, seria apenas necessário copiar o bloco que simula o funcionamento de um robot as vezes necessárias. No entanto, o simulador não está preparado para tal alteração, e seria necessário alguma manipulação do código dos vários *scripts*.

- Alguns testes levam a crer que existem inconsistências no simulador, provavelmente devido à forma como o *Simulink* simula o modelo em tempo contínuo. Por exemplo, o percurso de $R_2$ a seguir $R_1$ e a evitar obstáculos era alterado por vezes quando se alterava o controlador de $R_3$, algo que deveria ser completamente independente. No entanto estas diferenças não são muito significativas.
- O controlador de dilatação, referenciado em 2.5, não foi simulado (e também não implementado em robots reais).

Para alterar outros parâmetros da simulação não disponíveis no menu do simulador, é possível editar o ficheiro simul.m e modificar os seguintes parâmetros (logo no início do ficheiro): posição inicial de $R_2$ e $R_3$ (x,y) (a orientação inicial é sempre de 0º); tempo de simulação (valor por defeito igual a 30 segundos); ganhos dos vários controladores ($k_1$, $k_2$ e $k_0$); nome do ficheiro de *Simulink* (valor por defeito 'sistema', compatível com *Simulink 5.0*). Outros parâmetros não se encontram disponíveis para modificação de uma forma simples, sendo necessário alterar o próprio modelo sistema.mdl.

# Anexo C – Funcionamento do plugin follow

- Como executar o plugin nos Scouts

**Instalação**: Copiar o ficheiro `socrob.tar` para cada um dos *Scouts,* e descomprimir (`tar -xvf socrob.tar`). De seguida, mudar para a directoria criada (`socrob/`) e fazer `make`. De seguida editar o ficheiro follow.c, na directoria `socrob/kernel/control/`. No início do ficheiro existem algumas definições que permitem fazer testes ao *plugin*:

```
//#define FILE_OUTPUT   //writes a file with the evolution of the state
//#define FOLLOW_DEBUG  //prints info in stdout
//#define RUN_LEADER    //this robot is the leader of the formation
//#define RUN_SBC       //forces SBC control with Ri; ignores obstacles and other leaders
//#define RUN_SSC       //forces SSC; SSC or nothing
//#define PRINT_RATE    //prints time rate of this plugin
```

Removendo as barras de comentário do início da linha, e é possível activar cada um dos modos de teste. O *Scout* 2, devido à sua impossibilidade de ler do *Blackboard*, fica limitado a ser líder, por isso para esse *Scout*, fica activa a linha:

```
#define RUN_LEADER    //this robot is the leader of the formation
```

Para os outros dois *Scouts* pode-se escolher tanto

```
#define RUN_SBC       //forces SBC control with Ri; ignores obstacles and other leaders
```

como

```
#define RUN_SSC       //forces SSC; SSC or nothing
```

No ficheiro config.dat estão declarados os vários parâmetros dos controladores. Para alterá-los, basta modificar esse ficheiro.

**Execução:** De seguida mudar para a directoria `/home/Users/ps/socrob/kernel` e executar: `./main -T follow .` Em alternativa, executar `./run` (*script* que corre o *main* com as mesmas opções em *loop* até ser interrompido). Este comando inicia o projecto Socrob no modo teste. Neste caso, é o *plugin follow* que é testado. Na prática, isto significa que este *plugin* é o único a ser executado pelo micro-agente relativo ao controlo. Este teste é executado indefinidamente, até o programa ser interrompido (Ctrl-C).

Dos três *Scouts,* o *Scout* 2 é sempre o líder. Assim, é necessário iniciar este *Scout* antes dos seguidores. No caso de ser utilizado o *Scout* 3 como substituto, é necessário adicionar o parâmetro `-id=<id>`, onde `<id>` é a identidade do *Scout* substituído (*Scout* 1: `bp`; *Scout* 2: `ph`; *Scout* 4: `vet`). Os robots seguidores podem ser iniciados por qualquer ordem. É ainda necessário no início inicializar a odometria (se possível antes de executar o programa), de modo a todos os robots terem o mesmo referencial.

- Como alterar o código do projecto Socrob de modo a incluir este *plugin*

De seguida são referenciados os passos necessários a inserir este *plugin* no código principal do projecto Socrob. Depois de criar uma directoria nova no *Scout*, fazer `cvs checkout <cvsbranch>` (substituir `<cvsbranch>` pela directoria do CVS que contenha a mais recente versão do projecto Socrob). De seguida criar um ficheiro chamado follow.c na directoria `socrob/kernel/control`. O código a incluir neste ficheiro é o listado no Anexo C. De seguida é necessário acrescentar as seguintes linhas a alguns ficheiros do projecto:

➔Ficheiro `main-scout.c`, na directoria `/socrob/kernel/`

No início do ficheiro

```
#define VELSTEPC 0.1
#define VELSTEPD 0.05
```

No início da função `main()`

```
int fd, s;
char buffer[ 1 ];
float vCommon = 0;
float vDiff = 0;
int velChange = FALSE;
float velStepC = VELSTEPC;
float velStepD = VELSTEPD;
float xPos, yPos, thPos;
```

No fim da função `main()`, substituir a linha

```
select( 1, NULL, NULL, NULL, NULL );
```

por

```
{
    read( fd, buffer, 1 );
    switch ( buffer[ 0 ] )
    {
        case 'Q':
        case 'q':
            vCommon  += velStepC;
            velChange = TRUE;
         break;

        case 'A':
        case 'a':
            vCommon  -= velStepC;
            velChange = TRUE;
            break;

        case 'O':
        case 'o':
            vDiff    -= velStepD;
            velChange = TRUE;
            break;

        case 'P':
        case 'p':
            vDiff    += velStepD;
            velChange = TRUE;
            break;

        case ' ':
            vCommon   = 0;
```

```
                vDiff     = 0;
                velChange = TRUE;
                break;

            case 'M':
            case 'm':
                vDiff     = 0;
                velChange = TRUE;
                break;

            case 'W':
            case 'w':
                odometryGetPosition( &xPos, &yPos, &thPos );
                printf( "Odometry: (x,y,th) = ( %1.3f, %1.3f, %1.3f )\n\r", xPos, yPos,
thPos );
                break;

            case 'S':
            case 's':
                odometrySetPosition( 0, 0, 0 );
                printf( "Odometry: Set position to (0,0,0)\n\r" );
                break;

            case 'x':
            case 'X':
                kill(getpid(),SIGINT);
                break;
        }   // switch

        /* -- Resend the speeds -- */
        if ( velChange == TRUE )
        {
            motorsSet( vCommon + vDiff, vCommon - vDiff );
            velChange = FALSE;
        }
    }
```

No início da função `terminateProgram()`

```
    char szAlive[ 120 ];
    //sets the heartbeat dead
    sprintf( szAlive, "%s.heartbeat", bbGetLocalString( "id", "none" ) );
    bbSetGlobalString( szAlive, "dead" );
```

➔Ficheiro `machine.c`, na directoria `/socrob/kernel/logicMachine/`

Na função `machineMALoop()`
substituir

```
    if ( bbGetLocalBool( "boot.test", FALSE ) == TRUE )
    {
        testMode( bbGetLocalString( "boot.test.major", "ERROR" ),
            bbGetLocalString( "boot.test.minor", "ERROR" ) );
        /* does not return from it */
    }
```

por

```
    if ( bbGetLocalBool( "boot.test", FALSE ) == TRUE )
    {
        printf( "ua LogicMachine" RC_RUNNING "\n\n");
        bbSetLocalBool("machine.running?", TRUE);
        testMode( bbGetLocalString( "boot.test.major", "ERROR" ),
        bbGetLocalString( "boot.test.minor", "ERROR" ) );
        /* does not return from it */
    }
```

➔Ficheiro `testmode.c`, na directoria `/socrob/kernel/logicMachine/`

No início do ficheiro, acrescentar a linha

```
       { "follow", &test_follow, "Test formation control plugin"},
```

à variável global

```
       STestMode majorModes[]
```

Acrescentar ao fim do ficheiro

```
    /*****************************************************************
     @Function: test_follow()
     @Description: Test formation control plugin
     @Author: Paulo Gomes
     *****************************************************************/

    static void
    test_follow( char *minor )
    {
       test_behaviour( "follow" );
    }
```

➔Ficheiro `testmode.h`, na directoria `/socrob/kernel/logicMachine/`

Acrescentar a declaração da seguinte função

```
static void test_follow( char *minor );
```

➔Ficheiro `Makefile`, na directoria `/socrob/kernel/control/`

Adicionar a

```
## Object List
PLUGINS = \
```

a linha

```
  follow.so \
```

➔Ficheiro `config.dat`, na directoria `/socrob/kernel/`

Adicionar junto às variáveis relativas ao *plugin control*

```
############################################################
### Control: FOLLOW

*.formation.deltasafe float 1.0
*.formation.delta0  float 0.8

*.formation.bp.x       float   0
*.formation.bp.y  float       0
*.formation.bp.th  float      0
*.formation.bp.v       float   0
*.formation.bp.w       float 0
*.formation.bp.leader1 string  ph
*.formation.bp.leader2  string  vet
*.formation.bp.l1d  float 1.0
*.formation.bp.fi1d  float -90
*.formation.bp.l2d  float 1
*.formation.bp.controler      string  sbc

*.formation.ph.x  float 0
*.formation.ph.y  float 0
*.formation.ph.th  float 0
*.formation.ph.v  float 0
*.formation.ph.w  float 0
*.formation.ph.leader1  string  bp
*.formation.ph.leader2  string  vet
*.formation.ph.l1d  float 1
*.formation.ph.fi1d  float 0
*.formation.ph.l2d  float 1
*.formation.ph.controler      string  leader
```

```
*.formation.vet.x  float 0
*.formation.vet.y  float 0
*.formation.vet.th  float 0
*.formation.vet.v  float 0
*.formation.vet.w  float 0
*.formation.vet.leader1  string  ph
*.formation.vet.leader2  string  bp
*.formation.vet.l1d  float 1
*.formation.vet.fi1d  float -180
*.formation.vet.l2d  float 1
*.formation.vet.controler     string  sbc
```

Depois de alterar todos estes ficheiros, basta fazer `make` na directoria `/socrob/` e o *plugin* é instalado.

- Funcionamento do *plugin follow*

No modo de teste, este *plugin* é executado indefinidamente. O método principal do *plugin* é o seguinte:

- Obtém informação (posição, orientação, velocidades) de toda a equipa (verificando se está a funcionar ou não), incluindo do próprio robot, através da função `getPopInfo()`.
- Para cada robot da equipa que estiver activo, determina a que distância está dele.
- Actualiza a sua própria posição no *blackboard* do sistema (função `setInfo()`).
- Obtém a lista de obstáculos através dos sensores (função `getObstacles()`).
- Procura entre os obstáculos encontrados, os que podem ser outros robots da equipa, e remove-os da lista de obstáculos a considerar. Tenta corrigir os valores da sua própria odometria comparando a localização que tem dos outros robots, com a localização que obtém dos sensores.
- Entre os obstáculos restantes, procura o mais próximo, e verifica se está mais perto que o delta de segurança. Para este obstáculo calcula o delta e o beta10.
- Verifica o estado do mundo (que robots é que estão activos, a que distância estão, se existem obstáculos, etc.) e escolhe um controlador de acordo com o protocolo de coordenação de formações. De seguida executa uma função de acordo com o controlador e com os líderes escolhidos (funções `separationBearingControl()`, `separationSeparationControl()`, `separationDistanceObstacleControl()`).
- Se não existirem líderes, entra em navegação autónoma, que neste caso, consiste em manter o robot parado, até encontrar outro líder perto.

- No caso de ter sido calculada uma actuação por um controlador, envia as velocidades linear e angular para os motores.
- Reinicia o processo.

Neste modo de teste, o robot líder é controlado pelo utilizador. A forma de manobrar o robot é semelhante ao modo de funcionamento da ferramenta `zmanual`, um programa que serve para controlar o robot, que é muito utilizado no projecto Socrob. Com a janela onde foi executado o programa do *Scout* 2 (o líder) seleccionada, utilizar as teclas 'q', 'a', 'o', 'p', respectivamente aumentar velocidade do modo comum, diminuir velocidade do modo comum, aumentar velocidade da roda direita (fazendo o robot rodar para a esquerda), aumentar velocidade da roda esquerda (fazendo o robot rodar para a direita). A barra de espaços para o robot, a tecla 'w' imprime no ecrã a odometria actual e a tecla 's' faz o *reset* à odometria. Com estes comandos, é possível controlar a trajectória do líder.