ALEXANDRE SHIROTA

Movimento Coordenado de Múltiplos Robôs Móveis em nível de Simulação

Monografia do Trabalho de
Conclusão de Curso

São Paulo
2008

ALEXANDRE SHIROTA

Movimento Coordenado de Múltiplos Robôs Móveis em nível de Simulação

Monografia do Trabalho de Conclusão de Curso

Área de concentração:
Engenharia Mecatrônica

Orientador:
Prof. Dr. Jun Okamoto Jr.

São Paulo
2008

FICHA CATALOGRÁFICA

Shirota, Alexandre
Movimento coordenado de múltiplos robôs móveis em nível de simulação / A. Shirota. -- São Paulo, 2007.
44 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1. Robôs 2. Robótica 3. Inteligência artificial I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II. t.

Sumário

Lista de Figuras

Resumo

Sistemas de múltiplos robôs tornaram-se, recentemente, um importante campo de estudo na área de robótica e inteligência artificial [6]. Considerando elementos móveis, o movimento coordenado é uma abordagem natural para o estudo da cooperação de robôs. Portanto, como projeto de trabalho de formatura, este documento trata da implementação de algoritmos de controle de formação de um grupo de múltiplos robôs em movimento coordenado.

Agradecimentos

Agradeço à Universidade de São Paulo e à Escola Politécnica por oferecerem condições de uma formação universitária de alto nível.

Aos familiares e amigos que estiveram presentes nos momentos de estudos, aprendizado e conquistas durante os anos de graduação.

Ao meu professor orientador, Dr. Jun Okamoto Jr. por oferecer condições e apoio para exercer um trabalho na área de cooperaçao de robôs.

A todos os demais que, de maneira direta ou indireta, me auxiliaram na minha formação e na realização do presente trabalho.

Capítulo 1

Introdução

1.1 Cooperação de Robôs

A utilização de robôs para realizar determinadas tarefas abre diversas possibilidades antes não aproveitadas, ou em auxílio a outras atividades já realizadas pelo homem. Como exemplo, pode-se citar aplicações em:

- Ambientes agressivos, onde o envio de um ser humano seja perigoso ou de alto custo [1];
- Ambientes inacessíveis ao homem: muito longes (extraterrestres) ou muito pequenos (escala microscópica) [1];
- Tarefas excessivamente repetitivas [1];
- Tarefas exigentes em termos de processamento de informação ou velocidade e precisão de execução.

Assim, uma característica desejável para essas máquinas é o da liberação de agentes humanos para a realização de outras atividades onde são mais eficientes. Em alguns casos pode ser até impossível a presença de monitoração humana dos robôs. Por isso é interessante que sejam *autônomos*.

Além disso, a quebra do paradigma de um único robô interagindo com seu ambiente em contraposição a *sistemas multi-robôs* ([4], [5], [3]) tem sido objeto de muito estudo nos últimos anos. Isso se deve às diversas vantagens potenciais oferecidas por esse novo modo de enxergar os sistemas robóticos, dentre as quais:

- Sucesso na execução de tarefas essencialmente muito complexas ou até impossíveis de serem realizadas por um único robô;

- Implementação física e/ou computacionalmente mais simples, de menor custo, mais flexível ou mais tolerante a falhas;

- Pode ser possível obter ganhos de desempenho na execução de determinadas tarefas.

No estudo desse tipo de sistema, é importante que se diferencie um sistema *coletivo* de um sistema *cooperativo*. O primeiro trata de qualquer sistema multi-robô, mesmo que os seus elementos não reconheçam os demais agentes do ambiente nem os interpretem como tal, basta que haja mais de um agente no sistema. No segundo caso, além dessa característica, observa-se particularmente um *comportamento de cooperação* entre seus agentes. Assim, não apenas existe um reconhecimento de outros elementos atuantes no ambiente, mas a eles é também identificado algum comportamento e a possibilidade de associação deles à execução de uma tarefa. Devido às interpretações subjetivas a que essas afirmações possam levantar, toma-se de [4] a seguinte definição:

> *"Dada uma especificação de tarefa, um sistema de múltiplos robôs apresentará comportamento cooperativo se, através de algum mecanismo de cooperação, houver um ganho na utilidade total do sistema."*

É esse tipo de sistema que tem-se, aqui, interesse em estudar.

Capítulo 2

Revisão Bibliográfica

2.1 Movimento Coordenado

Percebe-se que a habilidade de navegar propositadamente é fundamental para grande parte dos animais e qualquer organismo inteligente [1]. Tal é a sua importância, que o movimento coordenado consitui uma das principais subdivisões de estudo da cooperação de robôs [5]. É uma atividade essencial para diversas das aplicações de cooperação apresentadas em estudos taxonômicos ([4] e [6]), como, por exemplo, a implementação de atividades de *"foraging"* (busca) e *"box pushing"* (manipulação cooperativa).

Neste último, é necessário que o sistema robótico forme e mantenha uma determinada configuração geométrica durante a execução da sua tarefa. Esse requisito está presente também em outros tipos de aplicação, como posicionamento de satélites para medições de precisão, aplicações militares de posicionamento estratégico, entre outros. A obtenção desse tipo de comportamento em sistemas de múltiplos robôs forma um de seus ramos de estudo e recebe a denominação de *Controle de Formações*.

Em [11], apresenta-se a seguinte definição:

> *"Formações são definidas como grupos de robôs móveis na geração e manutenção de alguma forma geométrica pré-determinada por meio do controle da posição e orientação de cada robô com relação ao grupo, enquanto permite-se que o grupo se mova como um todo."*

Os mesmos autores sugerem uma possível estrutura de classificação dos trabalhos na área segundo critérios que julgam importantes para a comparação da estratégia de coordenação do sistema robótico. Por não se ter interesse aqui de detalhar cada ponto da classificação proposta, uma abordagem mais geral será

adotada na apresentação de algumas das estratégias utilizadas para o tratamento da questão do controle de formações. Essa abordagem é também reconhecida por [12] como uma possível classificação dos trabalhos na área e segundo a classificação de [11], refere-se apenas à parte da *estratégia de controle*.

2.1.1 Estruturas Virtuais

Um exemplo de desenvolvimento teórico, simulação e implementação desse método pode ser encontrado em [7].

A idéia fundamental é a de que pontos movendo-se no espaço relacionados por condições geométricas fixas comportam-se da mesma maneira que os pontos de um corpo rígido que apresente o mesmo movimento. Se os robôs se comportarem como esse conjunto de pontos, formarão uma estrutura virtual. Assim, no artigo define-se:

> *"Uma estrutura virtual (EV) é uma coleção de elementos a qual mantém uma relação geométrica (semi) rígida entre si e relativa a um determinado sistema de referência."*

O problema, em si, é definido como o de projetar um algoritmo de controle que seja capaz de fazer com que um grupo de n robôs mova-se em formação (ou seja, mantenha-se em coerência com a estrutura virtual) numa determinada direção. A proposta é a de obter esse resultado utilizando o conceito de *EV*. A vantagem desse método é que a formação e movimentação do sistema (estrutura virtual) é bastante robusta. Porém, em situações em que se necessita uma reestruturação da forma ou configuração da formação, esse método pode não ser a melhor escolha.

O algoritmo utilizado apresenta a seguinte estrutura geral:

1. Alinhamento da EV com a configuração atual dos robôs;
2. Deslocamento da EV de um Δx e $\Delta\theta$ cujos valores são determinados por um gerador de trajetórias;
3. Calcular as trajetórias individuais dos robôs para interceptar a EV nos pontos desejados;
4. Estabelecer os comandos do atuação (dos motores) para obter as trajetórias determinadas;
5. *Goto* 1;

O alinhamento da EV é feito através de um método numérico de otimização. Procura-se uma transformação I_r^w composta de uma translação e uma rotação, a qual minimize determinadas métricas que levam em consideração a diferença entre a posição dos robôs e os pontos correspondentes da EV.

Outra característica importante da solução adotada é a existência de um fluxo de controle bidirecional entre a EV e a formação, de maneira que tanto a EV controla a posição dos robôs quanto a posição dos robôs determina a posição da estrutura virtual. Assim, o controle da formação não é prejudicado pela falha em algum dos robôs à custa do risco de não se atingir o alvo final num caso de falha completa de algum dos componentes. Nesse caso, propõe-se que algum processo de alto nível fique responsável por detectar tal situação e tomar as decisões convenientes.

2.1.2 Comportamento Reativo e Motor-Schemas

O paradigma do comportamento reativo baseia-se em modelos biológicos de inteligência: acopla-se diretamente os sensores aos atuadores segundo determinadas regras, os *comportamentos*. Com isso, a ação resultante do robô é composta pela combinação de uma série de comportamentos pré-definidos ao invés de um processo racional rigoroso. Algumas das vantagens desse modo de implementar o controle do robô são [1]:

- Capacidade de obter vários comportamentos simultâneos;
- Diminuição dos efeitos das restrições de tempo real impostas aos sensores, devido ao seu acoplamento direto com os atuadores.

Por outro lado, não é possível definir clara e precisamente o comportamento do grupo durante a execução da tarefa.

Os "motor-schemas" são modos de acoplar os motores diretamente aos sensores segundo comportamentos desejados. O resultado de tal ação de controle é um vetor de velocidade com uma determinada direção, sentido e módulo, o qual determina a reação do robô aos estímulos ambientes. Em [8], esse é o método utilizado para implementar o controle de formações. Combina-se comportamentos de navegação aos de controle de formação. Assim, obtém-se comportamentos de desvio de obstáculos, navegação rumo a um alvo e geração e manutenção de uma formação.

As formações desejadas são pré-determinadas, estabelecendo-se um identificador para cada robô, o qual corresponde à sua posição na formação. Essa posição é

determinada pela geometria da formação e pelo *unit-center* (centro da formação), o qual pode ser de 3 tipos:

- *Unit-center referenced*: o referencial da formação é um ponto calculado por cada robô segundo a média das posições x e y de todos os elementos.
- *Leader-referenced*: cada robô determina a sua posição na formação com relação um único líder, o qual apenas estabelece a trajetória, sem procurar manter-se na formação.
- *Neighbor-referenced*: a referência de cada robô é qualquer outro robô pré-determinado.

Os *motor-schemas* utilizados foram 4, além de um adicional para considerar efeitos de ruído no sistema. São eles: "move-to-goal" (atingir alvo), "avoid-static-obstacles" (evitar obstáculos), "avoid-robot" (evitar colisão com outro robô), "maintain-formation" (manter formação). A cada um deles é estabelecido um ganho, indicando a importância relativa deles no comportamento final do robô e, conseqüentemente, da formação.

Basicamente, são duas as etapas envolvidas no algoritmo utilizado: a de *percepção*, na qual cada elemento calcula a sua posição ideal na formação, e a *motora*, na qual os comandos de movimento são geradas e executadas. O vetor de movimento sempre é direcionado do centro da formação para o alvo, mas a magnitude depende da distância a que o robô se encontra da sua posição ideal (calculada na etapa motora).

O algoritmo apresentado é implementado tanto em robôs reais quanto em nível de simulação. São apresentados e discutidos o comportamento do sistema em trajetórias curvas e em campos com obstáculos. Além disso, é feita uma discussão sobre como o sistema é afetado pela escolha do centro da formação.

2.1.3 Leader-Follower

Nesse método de controle da formação, o grupo é dividido entre robôs líderes e robôs seguidores. Geralmente, existe um líder específico o qual realiza o papel do planejamento de trajetória, de maneira autônoma ou através do recebimento de comandos de um operador. Os robôs seguidores por sua vez seguem a trajetória de seu líder segundo determinadas regras (geralmente uma distância e orientação relativas). Uma das vantagens desse método é a sua implementação prática. Por outro lado, o fato de depender unicamente de um líder torna-o um ponto de falha

crucial. Além disso, não existe, a princípio, nenhuma realimentação de informação sobre o estado dos seguidores para o líder.

Um exemplo de utilização dessa técnica de controle de formação está na referência [9]. A abordagem proposta consiste na obtenção de comportamentos de controle de formação, a princípio complexos, a partir da implementação de modos simples de controle e uma estratégia de mudança entre eles que resulte num sistema estável. Esses controladores são descentralizados (podendo ser também centralizados) e reativos. Uma discussão mais detalhada apresenta-se no capítulo 4.

2.2 Exploração

Além de ser capaz de gerar e manter uma formação, decidiu-se estudar e acrescentar ao sistema funcionalidades relativas à exploração do ambiente. A revisão bibliográfica levantou alguns trabalhos voltados nessa direção, porém nada tão esclarecedor a ponto de guiar o desenvolvimento no trabalho. Portanto, algumas idéias próprias de implementação foram exploradas. Essencialmente, a mudança na geometria da formação é o que permitiu agregar ao sistema alguma funcionalidade de exploração.

A primeira abordagem implementada utiliza um controlador humano para determinar o momento adequado de realizar a mudança da formação. Em contraposição à simplicidade de implementação, existe a necessidade constante de um supervisor do sistema para controlar a trajetória da formação e alterá-la conforme necessário. Na segunda abordagem, a mudança de formação é disparada por uma lógica implementada nos sensores do robô a qual automaticamente determina quando é necessário mudar a formação e, portanto, dispensa a presença do controlador para tal.

Em ambos os casos, porém, a trajetória da formação é determinada pelo controlador humano. Devido ao tempo e os recursos disponíveis, não se implementou um gerador de trajetórias, mas existe uma literatura extensa sobre o tema, como pode ser encontrado em [2].

Capítulo 3

Modelos

3.1 Modelo do Robô

O modelo de robô utilizado durante a implementação foi o *Robolab*, do Laboratório de Microprocessadores da Escola Politécnica da USP, desenvolvido no Laboratório de Percepção Avançada. As dimensões relevantes estão na tabela 3.1.

Tabela 3.1: Parâmetros do *Robolab*

Parâmetro	Valor
Largura	100(mm)
Comprimento	130(mm)

Uma foto do robô, assim como sua representação no software de simulação, junto com os sensores estão nas figuras 3.1 e 3.2, respectivamente.

3.2 Modelo do Sensor

O sensor utilizado no robô é o GP2D120, da fabricante *Sharp*. Trata-se de um sensor infra-vermelho, cujas especificações técnicas podem ser encontradas no seu data sheet [13]. Para implementação no simulador, as informações de interesse estão na tabela 3.2.

Tabela 3.2: Parâmetros do array de sensores infra-vermelho

Parâmetro	Valor
Alcance (mm)	400
Ângulo de Abertura	26^o

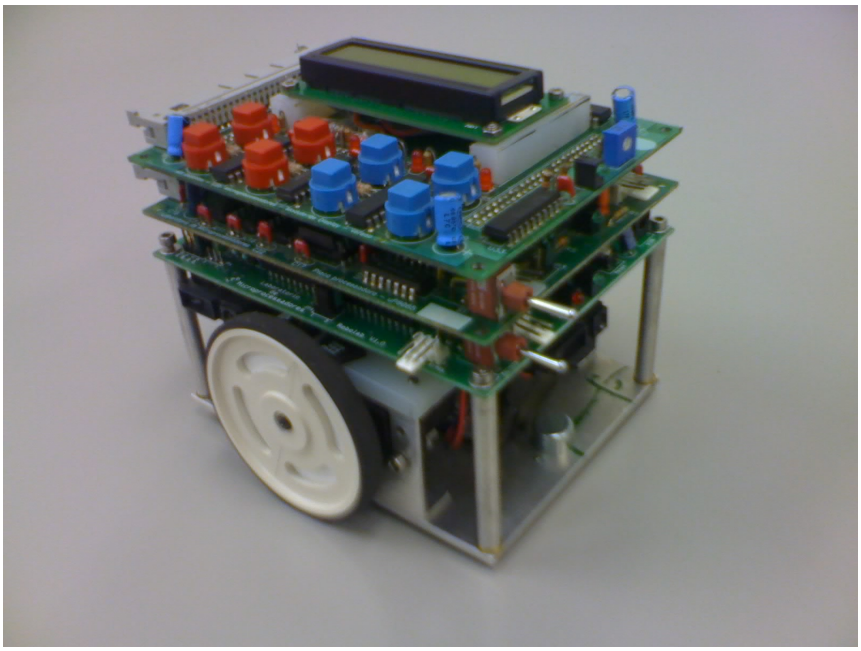

Figura 3.1: Foto do *Robolab*

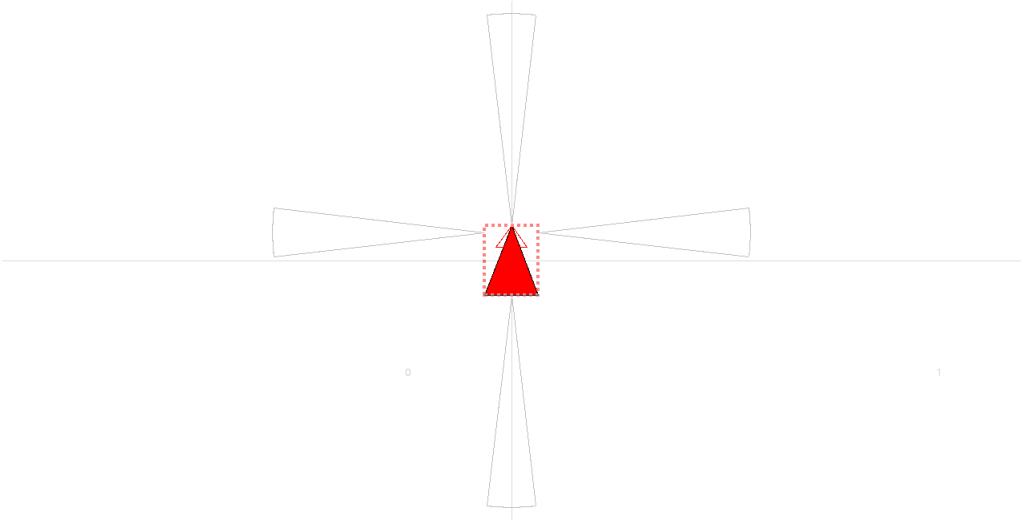

Figura 3.2: Representação do *Robolab* no *Stage*

Capítulo 4

Implementação

4.1 Movimento Coordenado

Como opção de estudo aprofundado e implementação em nível de simulação, optou-se pelo método proposto na referência [9]. Essa escolha foi baseada no detalhamento presente no artigo em questão, o que permitiu um melhor entendimento do modelamento, da teoria e, conseqüentemente, da sua implementação.

Alguns paradigmas da abordagem proposta são importantes no contexto da descrição do sistema, como:

- Presença de líder, o qual determina a trajetória principal do sistema;
- A relação entre os diversos robôs é determinada por um grafo, no qual cada robô é descrito por um nó e os arcos representam uma restrição entre eles. Consequentemente associa-se a esse arco um controlador o qual procura manter a restrição satisfeita.
- O controle dos robôs (não-holonômicos) é feito através de comandos de velocidades linear e angular (v_i, ω_i).

O modo $SB_{ij}C$, consiste na descrição do seguimento de um líder, com base em dois parâmetros de controle: distância (l_{ij}) e ângulo relativo (ψ_{ij}). O objetivo é controlar essas variáveis para os valores desejados $(l_{ij}^d,\ \psi_{ij}^d)$. O modelamento realizado e os parâmetros relevantes podem ser observados na figura 4.1. O segundo modo de controle, SD_oC, implementa uma estratégia de desvio de obstáculos que se baseia no seguimento de um robô virtual que se encontra tangente ao obstáculo.

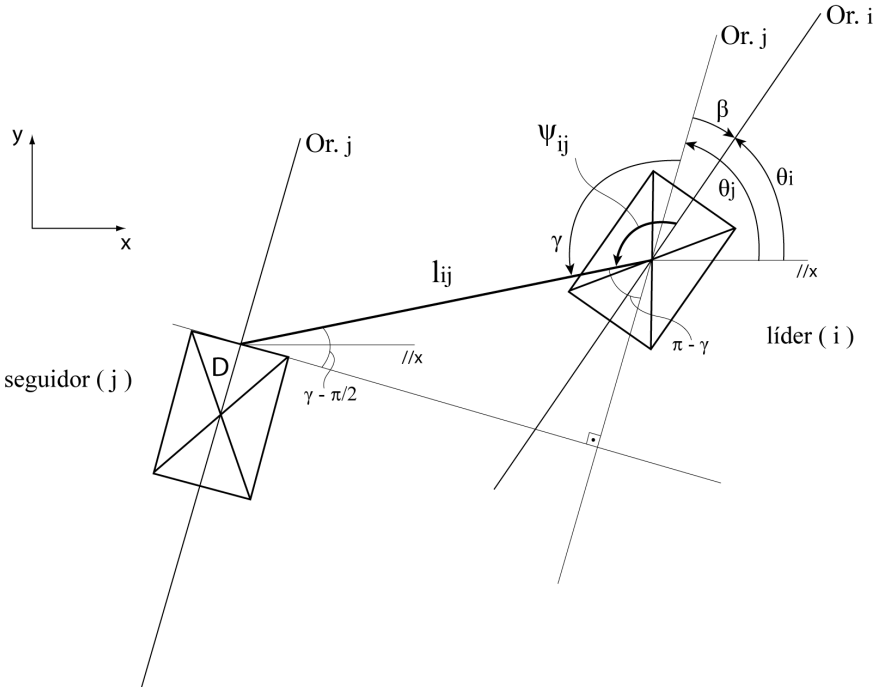

Figura 4.1: Modelamento físico $SB_{ij}C$

4.2 Equacionamento

4.2.1 Controlador SBijC

As equações[1] que descrevem o sistema são obtidas representando os parâmetros dos robôs em termos das variáveis de controle. Isso é feito pela decomposição das velocidades lineares e angulares dos robôs nas direções paralela e transversal de l_{ij}, obtendo-se[2]:

$\dot{l_{ij}}$ = projeção das velocidades lineares e angulares na direção de l_{ij}

$$\dot{l_{ij}} = -v_j \cos(\pi - \gamma) + v_i \cos(\pi - \psi) + D\omega_j \cos(\gamma - \frac{\pi}{2}) \quad (4.1)$$

$$= v_j \cos(\gamma) - v_i \cos(\psi) + D\omega_j \sin(\gamma) \quad (4.2)$$

[1]Por conveniência, foi eliminada a referência ij de algumas variáveis. Subentende-se que as variáveis relativas têm como líder o robô i e seguidor o robô j.

[2]$\gamma = \beta + \psi_{ij}$

$\dot{\psi_{ij}}$ = projeção das velocidades lineares e angulares na direção perpendicular a l_{ij}

$$\dot{\psi_{ij}} = -\frac{D\omega_j \sin(\gamma - \frac{\pi}{2})}{l_{ij}} - \frac{v_j \sin(\pi - \gamma)}{l_{ij}} - \omega_i + \frac{v_i \sin(\pi - \psi)}{l_{ij}} \quad (4.3)$$

$$= \frac{D\omega_j \cos(\gamma)}{l_{ij}} - \frac{v_j \sin(\gamma)}{l_{ij}} - \omega_i + \frac{v_i \sin(\psi)}{l_{ij}} \quad (4.4)$$

Dessa maneira, o seguinte sistema pode ser montado:

$$\dot{z_{ij}} = \mathbf{G_1}(z_{ij}, \beta_{ij})\mathbf{u_j} + \mathbf{F_1}(z_{ij})\mathbf{u_i} \quad (4.5)$$

onde $z_{ij} = [l_{ij} \quad \psi_{ij}]^T$ é a saída do sistema, $\mathbf{u_i} = [v_i \quad \omega_i]^T$ é o vetor de velocidades do robô i, $\mathbf{u_j} = [v_j \quad \omega_j]^T$ é o vetor de velocidades do robô j e as matrizes são definidas por:

$$\mathbf{G_1} = \begin{pmatrix} \cos(\gamma_{ij}) & D\sin(\gamma_{ij}) \\ \frac{-\sin(\gamma_{ij})}{l_{ij}} & \frac{D\cos(\gamma_{ij})}{l_{ij}} \end{pmatrix}, \qquad \mathbf{F_1} = \begin{pmatrix} -\cos\psi_{ij} & 0 \\ \frac{\sin(\gamma_{ij})}{l_{ij}} & -1 \end{pmatrix} \quad (4.6)$$

Como pode ser facilmente constatado, esse sistema é não-linear (presença de senos, cossenos e produtos cruzados das variáveis de controle). A abordagem adotada para resolver esse problema foi o de realizar uma linearizaçao por entrada e saída (*"input-output feedback linearization"*). Esse método mantém as não-linearidades essenciais do sistema, mas do ponto de vista da entrada e saída, o sistema passa a ser linear. A aplicação de tal procedimento resulta na seguinte equação linearizada:

$$\dot{z_{ij}} = \mathbf{p_1} = \mathbf{k}(z_{ij}^d - z_{ij}) \quad (4.7)$$

onde,

$$\mathbf{p_1} = \begin{pmatrix} k_1(l_{ij}^d - l_{ij}) \\ k_2(\psi_{ij}^d - \psi_{ij}) \end{pmatrix} \quad (4.8)$$

Aplicando-se esse resultado na equação 4.5, obtém-se para o robô j:

$$\mathbf{u_j} = \mathbf{G_1^{-1}}(p_1 - \mathbf{F_1}\mathbf{u_i}) \quad (4.9)$$

Onde $\mathbf{p_1}$ pode ser visto como um controlador proporcional de ganhos k_1 e k_2.

Portanto, realizando a inversão de $\mathbf{G_1}$ [3] , tem-se:

$$\mathbf{G_1^{-1}} = \frac{l_{ij}}{D}\begin{pmatrix} \frac{D\cos(\gamma_{ij})}{l_{ij}} & -D\sin(\gamma_{ij}) \\ \frac{\sin(\gamma_{ij})}{l_{ij}} & \cos(\gamma_{ij}) \end{pmatrix} = \begin{pmatrix} \cos(\gamma_{ij}) & -l_{ij}\sin(\gamma_{ij}) \\ \frac{D\sin(\gamma_{ij})}{l_{ij}} & \frac{l_{ij}\cos(\gamma_{ij})}{D} \end{pmatrix} \tag{4.10}$$

Aplicando esse resultado em (4.9):

$$\begin{aligned} \mathbf{u_j} &= \mathbf{G_1^{-1}}\left\{ \begin{bmatrix} k_1(l_{ij}^d - l_{ij}) \\ k_2(\psi_{ij}^d - \psi_{ij}) \end{bmatrix} - \begin{bmatrix} -\cos(\psi_{ij}) & 0 \\ \frac{\sin(\psi_{ij})}{l_{ij}} & -1 \end{bmatrix} \begin{bmatrix} v_i \\ \omega_i \end{bmatrix} \right\} \\ &= \begin{bmatrix} \cos(\gamma_{ij}) & -l_{ij}\sin(\gamma_{ij}) \\ \frac{D\sin(\gamma_{ij})}{l_{ij}} & \frac{l_{ij}\cos(\gamma_{ij})}{D} \end{bmatrix} \begin{bmatrix} k_1(l_{ij}^d - l_{ij}) + \cos(\psi_{ij})v_i \\ k_2(\psi_{ij}^d - \psi_{ij}) - \frac{\sin(\psi_{ij})}{l_{ij}} + \omega_i \end{bmatrix} \end{aligned}$$

E, portanto, as equações das velocidades linear e angular do robô j, são:

$$v_j = \cos(\gamma_{ij})[k_1(l_{ij}^d - l_{ij}) + \cos(\psi_{ij})v_i] - l_{ij}\sin(\gamma_{ij})[k_2(\psi_{ij}^d - \psi_{ij}) - \frac{\sin(\psi_{ij})}{l_{ij}}v_i + \omega_i] \tag{4.11}$$

$$\omega_j = \frac{\sin(\gamma_{ij})}{D}[k_1(l_{ij}^d - l_{ij}) + \cos(\psi_{ij})v_i] + \frac{l_{ij}\cos(\gamma_{ij})}{D}[k_2(\psi_{ij}^d - \psi_{ij}) - \frac{\sin(\psi_{ij})}{l_{ij}}v_i + \omega_i] \tag{4.12}$$

4.2.2 Controlador SDoC

A partir da fig 4.2, tem-se:

$$\dot{z_{ij}} = \mathbf{G_1}\mathbf{u_j} + \mathbf{F_1}\mathbf{u_i} \tag{4.13}$$

Onde:

$$\mathbf{G_1} = \begin{pmatrix} \cos\gamma_{ij} & D\sin\gamma_{ij} \\ \sin\gamma_{oj} & D\cos\gamma_{oj} \end{pmatrix}, \qquad \mathbf{F_1} = \begin{pmatrix} -\cos\psi_{ij} \\ 0 \end{pmatrix} \tag{4.14}$$

Desenvolvendo, obtém-se:

$$\mathbf{u_j} = \mathbf{G_1^{-1}}(p_2 - \mathbf{F_1}\mathbf{u_i}) \tag{4.15}$$

[3] $seja\ \mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix},\ então\ \mathbf{A^{-1}} = \frac{1}{detA}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix} = \frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$

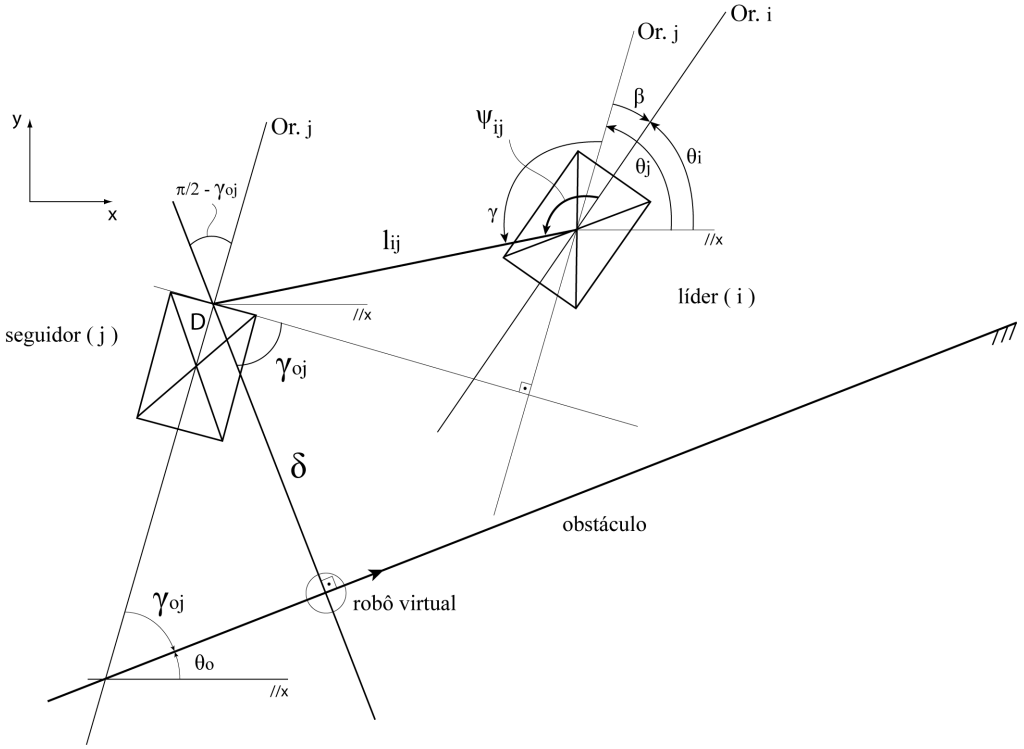

Figura 4.2: Modelamento físico SD_oC

Com $\mathbf{p_2} = [k_1(l_{ij}^d - l_{ij}) \quad k_0(\delta^d - \delta)]^T$. Realizando a inversão de $\mathbf{G_1}$:

$$\mathbf{G_1^{-1}} = \frac{1}{D\cos(\gamma_{ij} - \gamma_{oj})} \begin{pmatrix} D\cos(\gamma_{oj}) & -\sin(\gamma_{oj}) \\ -D\sin(\gamma_{ij}) & \cos(\gamma_{ij}) \end{pmatrix} \tag{4.16}$$

$$= \frac{1}{\cos(\gamma_{ij} - \gamma_{oj})} \begin{pmatrix} \cos(\gamma_{oj}) & -\frac{\sin(\gamma_{oj})}{D} \\ -\sin(\gamma_{ij}) & \frac{\cos(\gamma_{ij})}{D} \end{pmatrix} \tag{4.17}$$

Assim:

$$\begin{aligned} \mathbf{u_j} &= \mathbf{G_1^{-1}} \left\{ \begin{bmatrix} k_1(l_{ij}^d - l_{ij}) \\ k_0(\delta^d - \delta) \end{bmatrix} - \begin{bmatrix} -\cos\psi_{ij} \\ 0 \end{bmatrix} \begin{bmatrix} v_i \\ \omega_i \end{bmatrix} \right\} \\ &= \frac{1}{\cos(\gamma_{ij} - \gamma_{oj})} \begin{bmatrix} \cos(\gamma_{oj}) & -\frac{\sin(\gamma_{oj})}{D} \\ -\sin(\gamma_{ij}) & \frac{\cos(\gamma_{ij})}{D} \end{bmatrix} \begin{bmatrix} k_1(l_{ij}^d - l_{ij}) + \cos(\psi_{ij})v_i \\ k_0(\delta^d - \delta) \end{bmatrix} \end{aligned}$$

E, portanto, as equações das velocidades linear e angular do robô j são:

$$v_j = \frac{1}{\cos(\gamma_{ij}-\gamma_{oj})}\{\cos(\gamma_{oj})[k_1(l_{ij}^d - l_{ij}) + \cos(\psi_{ij})v_i] - \frac{\sin(\gamma_{oj})}{D}k_0(\delta^d - \delta)\} \quad (4.18)$$

$$\omega_j = \frac{1}{\cos(\gamma_{ij}-\gamma_{oj})}\{-\sin(\gamma_{ij})[k_1(l_{ij}^d - l_{ij}) + \cos(\psi_{ij})v_i] - \frac{\cos(\gamma_{ij})}{D}k_0(\delta^d - \delta)\} \quad (4.19)$$

Este controlador utiliza, para seguimento do obstáculo, o valor da tangente da superfície alvo. No caso do artigo de referência [9], este valor é extraído do ambiente através de uma câmera omnidirecional e um algoritmo de cálculo. No caso implementado neste trabalho, a leitura de câmera foi substituída pela leitura de 4 sensores infra-vermelho posicionados a 90 graus entre si. Isso foi feito por razões dentre as quais:

- Inviabilidade de se implementar um algoritmo de extração da imagem e cálculo da tangente da superfície do obstáculo a partir de uma câmera omnidirecional;
- Estudo do comportamento dos algoritmos de cooperação em modelos semelhantes ao robô *Robolab* desenvolvido no *Laboratório de Percepção Avançada* da Escola Politécnica da USP.

Essa modificação motivou, evidentemente, a substituição do algoritmo de detecção da tangente do obstáculo por um novo algoritmo compatível com a leitura dos sensores infra-vermelho. A idéia dessa algoritmo é o de fazer uma estimativa inicial sobre o ângulo do obstáculo e atualizar essa informação de acordo com as leituras do sensor.

Devido a essa alteração no sensoriamento do ambiente, algumas simplificações são feitas:

- O obstáculo é assumido como sendo, localmente, uma parede reta;
- O ângulo inicial do obstáculo é assumido como sendo a 90^0 do sensor que retornar a menor leitura de distância.

Listing 4.1: Algoritmo de detecção do ângulo da parede

```
if( sensor frontal ){
  desvie do obstáculo;
  if( leitura sensor diminui ){
    troque sinal da estimativa do angulo;
  }
}
```

```
else{
  desvie do obstáculo;
  if( leitura sensor aumenta ){
    troque sinal da estimativa do angulo;
  }
  if( taxa de variacao do sinal do sensor inverte sinal ){
    captura angulo parede;
  }
}
```

Na listagem 4.1, apresenta-se o algoritmo utilizado para fazer a detecção do ângulo da parede. Existem, basicamente, dois casos:

- Sensor frontal retorna menor distância: nesse caso, sabe-se que o obstáculo mais urgente está à frente do robô. Portanto, troca-se o modo do controlador para o SD_oC com uma estimativa inicial para o ângulo (a 90^0 para a esquerda do robô, por exemplo). Caso o valor da distância devolvida pelo sensor diminua, significa que o robô está se virando em direção à parede, ou seja, o sinal da estimativa inicial não foi boa, portanto, inverte-se esse valor. Com uma estimativa com o sinal correto, eventualmente o robô irá desviar da parede e sua frente estará livre.

- Sensor lateral retorna menor distância: nessa situação, a intenção é seguir a parede, a 90^0 do sensor. Portanto, deseja-se estabilizar o ângulo em torno desse valor. Se a estimativa inicial estiver correta, não há necessidade de se atualizar esse valor. Portanto, o que se monitora é a taxa de variação desse sinal. Se esse valor estiver diminuindo, significa que o robô deve continuar girando na mesma direção para atingir a posição relativa desejada. Caso contrário, deve-se inverter o sinal de rotação. A inflexão do sinal da taxa de variação acusa que passou pelo angulo de 90^0, portanto o valor do angulo na inversão é armazenado.

4.3 Mudança de Formação

Uma vez verificada a possibilidade de realizar o movimento coordenado, procurou-se utilizar os métodos de controle de formação para uma especificação de mais alto nível, tornando essa funcionalidade útil. Particularmente, isso foi feito voltado para a capacidade de exploração. A abordagem da exploração neste trabalho não se refere rigorosamente ao estudo de métodos ou algoritmos específicos que implementem a exploração no sistema, mas sim o de utilizar a formação de maneira prática durante essa atividade.

Fundamentalmente, a maneira como a formação pode auxiliar na exploração é por meio da sua geometria. Como hipótese, deve-se assumir um ambiente bidimensional e que os robôs não tenham outro método de transposição de obstáculos a não ser pelo desvio.

Num primeiro caso, os robôs podem ser distribuídos de maneira a ocupar uma larga região, ampliando-se a área percorrida (explorada). Porém, ao se deparar com um obstáculo, nem sempre a mesma configuração geométrica pode ser mantida, principalmente frente a uma passagem estreita. Nesse caso, o robô líder continua determinando a trajetória pela qual os demais devem passar e a formação é alterada de maneira a estreitar a área frontal ocupada pelos robôs. No caso limite, forma-se uma linha, na qual os robôs seguidores executam a mesma trajetória do líder.

De maneira geral, durante a exploração, é desejável que os robôs ocupem uma larga região para aumentar a quantidade de informação coletada do ambiente.

4.3.1 Supervisionada

Neste modo, o sistema não atua de maneira autônoma, mas depende de um controlador para determinar o momento certo de mudar de formação. Isso é feito através de um comando executado pelo supervisor sempre que achar necessário.

No modo supervisionado, um controlador humano deve constantemente monitorar o sistema, não só para determinar a trajetória percorrida pelos robôs, mas também para decidir quando deve ser necessário mudar a formação. O controlador pode escolher por entre formações pré-determinadas, conforme demonstrado nas figuras 5.10, 5.11, 5.12 e 5.13.

4.3.2 Automática

A fim de implementar a mudança de formação automática, foram necessárias, essencialmente, duas alterações no sistema:

- Implementação de lógica aplicada sobre os sensores para determinar situações de desvio de obstáculo;
- Criação de estrutura computacional que suporte as funcionalidades necessárias, em particular, uma estrutura que permita escolher e "carregar" formações no sistema robótico.

A estrutura computacional utilizada foi, propriamente, a de utilização de uma *struct* contendo os parâmetros que definem uma formação. Ao configurar esses

parâmetros, basta passá-los ao sistema para que haja um chaveamento e estabilização em torno dos valores desejados de acordo com o mecanismo de movimento coordenado, já descrito.

Mesmo sendo automática, esse modo de mudança de formação não libera o sistema totalmente da presença do controlador visto que é necessário ainda determinar a trajetória do grupo de robôs. Como já foi mencionado anteriormente, isso não é visto como uma deficiência do sistema, já que esse não é o intuito do trabalho. Uma referência de literatura sobre o assunto pode ser encontrado em [2].

4.4 Implementação Computacional

Uma vez obtido o equacionamento, passou-se à implementação desse sistema e dos controladores em nível de simulação computacional. Os *softwares* utilizados foram o *Player* e o *Stage*, ambos desenvolvidos [10] na *University of South California* (USC, EUA). O *Player* pode ser entendido como um *Hardware Abstraction Layer*, cuja função é mascarar os detalhes de hardware do robô, provendo conexões abstratas entre o software de controle desenvolvido e os robôs a serem controlados. *Stage* é um simulador em 2-D que possibilita criar um ambiente onde pode ser colocado um sistema de múltiplos robôs. Esses robôs, os seus sensores e atuadores podem ser acessados através do *Player* assim como se fossem robôs reais. Uma vez obtido o equacionamento, passou-se à implementação desse sistema e dos controladores em nível de simulação computacional. Os *softwares* utilizados foram o *Player* e o *Stage*, ambos desenvolvidos [10] na *University of South California* (USC, EUA). O *Player* pode ser entendido como um *Hardware Abstraction Layer*, cuja função é mascarar os detalhes de hardware do robô, provendo conexões abstratas entre o software de controle desenvolvido e os robôs a serem controlados. *Stage* é um simulador em 2-D que possibilita criar um ambiente onde pode ser colocado um sistema de múltiplos robôs. Esses robôs, os seus sensores e atuadores podem ser acessados através do *Player* assim como se fossem robôs reais.

Para poder entender e utilizar esses programas, é fundamental o conhecimento de 3 conceitos básicos:

- *interface*: especifica como interagir com uma determinada classe de sensor, atuador ou algoritmo. Define a semântica e sintaxe de toda mensagem que pode ser trocada entre entidades da mesma classe.

- *driver*: um trecho de código que se comunica com o sensor, atuador ou

algoritmo, traduzindo suas entradas e saídas, adaptando-as ao determinado tipo de *interface*.

- *device*: um driver em conjunto com sua interface atribuído a um endereço adequado. No *Player*, as informações são sempre trocadas entre *devices* através de *interfaces*.

Existem diversos *interfaces* e *drivers* pré-definidos pelos desenvolvedores do Software. Mesmo que não haja um driver para o dispositivo do usuário (ou este não esteja satisfeito com o existente), pode-se criá-los, porém isso não será tratado aqui.

Dessa maneira, o *Player* pode ser entendido também como um servidor de *devices* através de uma rede. A comunicação com os programas de controle ocorre através do protocolo TCP. Assim, a interface externa do *Player* é simplesmente uma porta TCP, o que permite que os programas de controle sejam escritos em qualquer linguagem que suporte tal padrão. Como a maioria das linguagens de programação atualmente atendem a esse requisito, proporciona-se liberdade na escolha da linguagem de desenvolvimento. No caso do presente trabalho, optou-se pela utilização da linguagem de programação C, por ser utilizada extensamente durante o curso de graduação e, portanto, de maior familiaridade.

Capítulo 5

Resultados

5.1 Trajetória Linear

A trajetória mais simples possível, uma reta, foi executada durante 40s, estabelecidas velocidades $v_i = 1,0m/s$ e $\omega_i = 0rad/s$ para o líder. Os resultados da simulação estão na figura 5.1. Os gráficos das distâncias e orientações relativas estão nos gráficos 5.3 e 5.4, respectivamente.

5.2 Trajetória Circular

Para demonstrar a validade do algoritmo em situações mais genéricas, foi executada simulação do sistema em uma trajetória circular completa. As velocidades do robô líder foram $v_i = 1,0m/s$ e $\omega_i = 0,05rad/s$, durante 2m40s. Uma simulação idêntica foi realizada para $\omega_i = -0,05rad/s$, obtendo-se resultados semelhantes, replicados convenientemente. Por simplicidade, apresenta-se apenas a primeira configuração.

Nas figuras 5.3, 5.4, 5.7 e 5.8, pode-se observar a estabilização dos parâmetros relativos em torno dos valores dados como desejados (l_{ij}^d e ψ_{ij}^d), conforme apresentado na tabela 5.1. No caso da trajetória circular, o desempenho da distância relativa foi pior por se tratar de uma situação mais dinâmica que a linear e pelo ganho mais baixo referente a essa variável.

5.3 Desvio de obstáculos

O código que gerou os resultados anteriores foi acrescido do algoritmo e das equações de desvio de obstáculos. Um exemplo de resultado obtido está apresentado na figura 5.9.

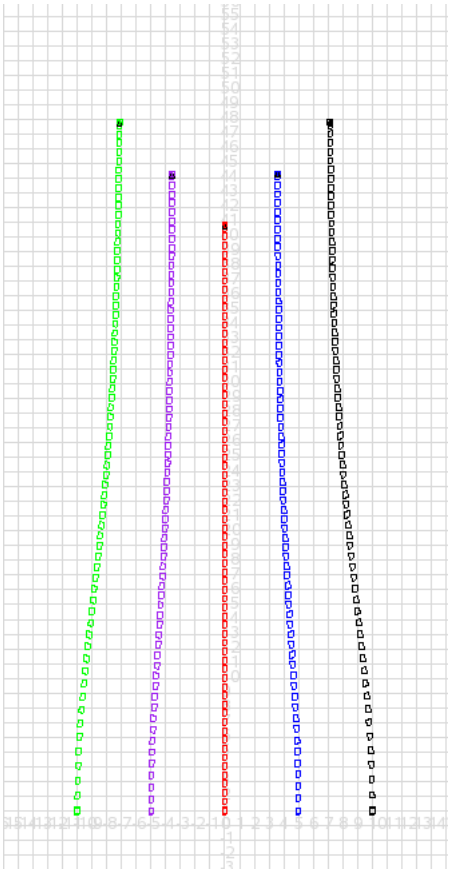

Figura 5.1: Trajetórias dos robôs

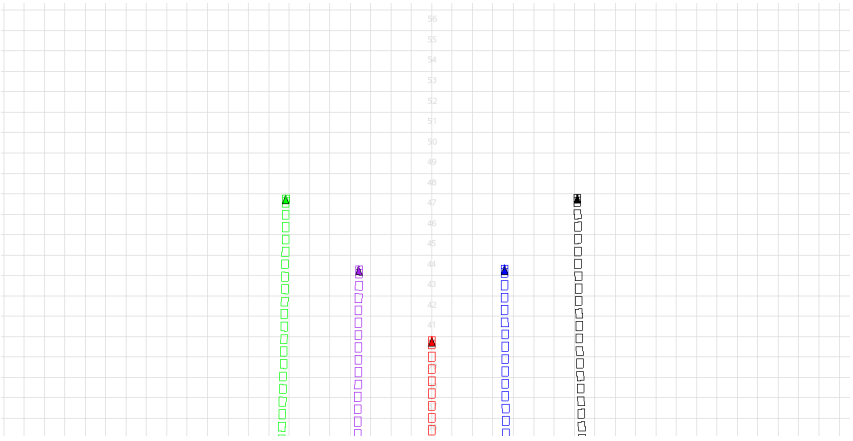

Figura 5.2: Configuração final dos robôs (próximo à estabilização)

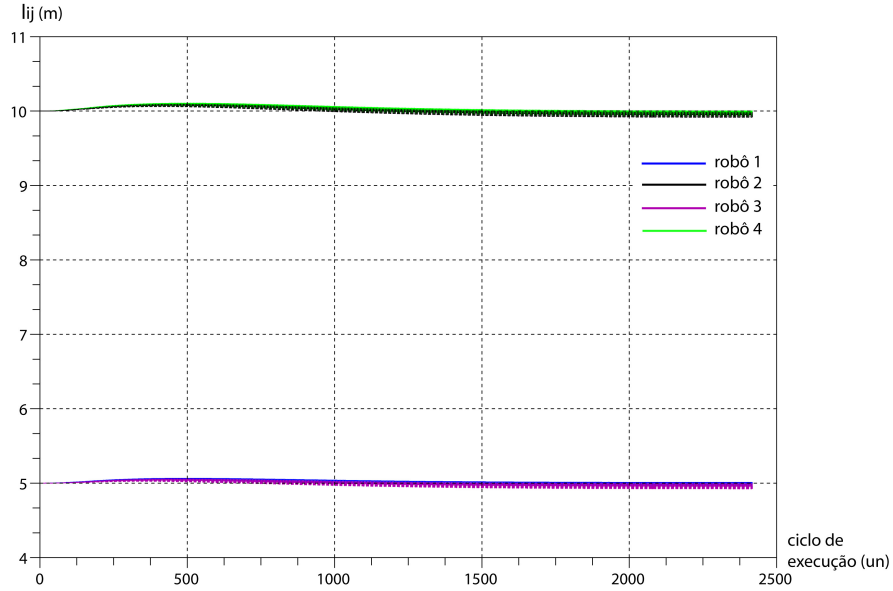

Figura 5.3: Distância relativa dos robôs com relação ao líder por ciclo de execução

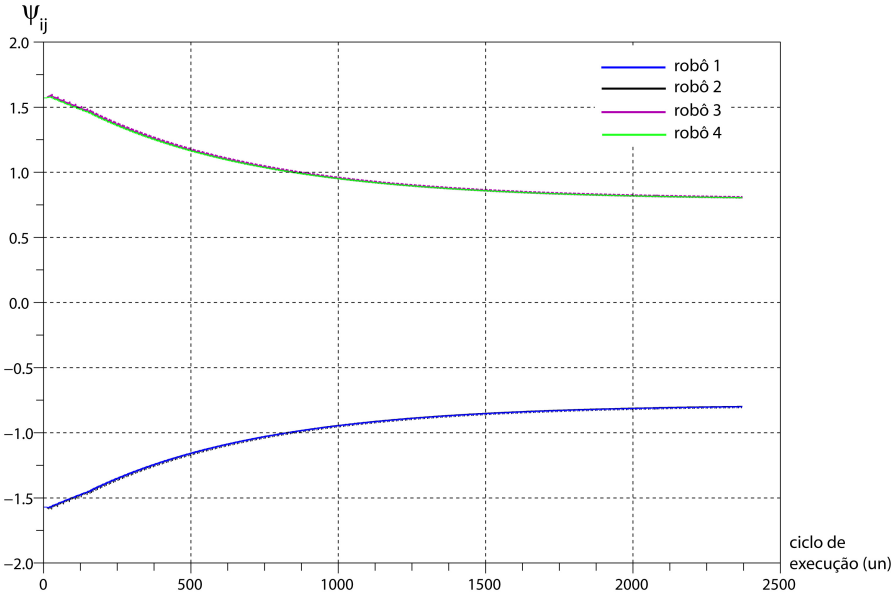

Figura 5.4: Orientação relativa dos robôs com relação ao líder por ciclo de execução

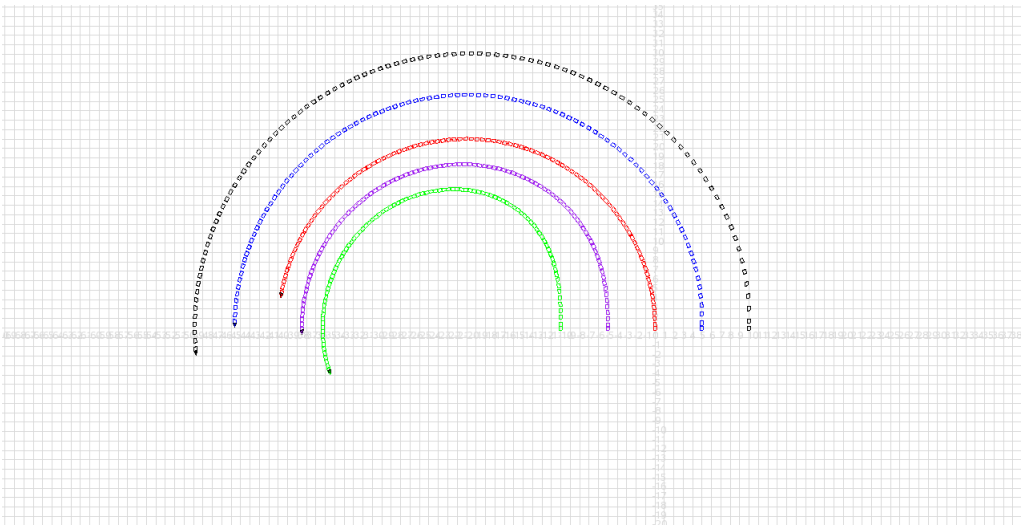

Figura 5.5: Trajetória intermediária dos robôs em formação

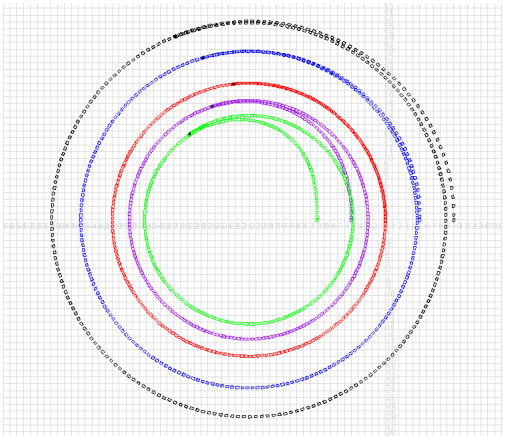

Figura 5.6: Trajetória do sistema após completar o ciclo

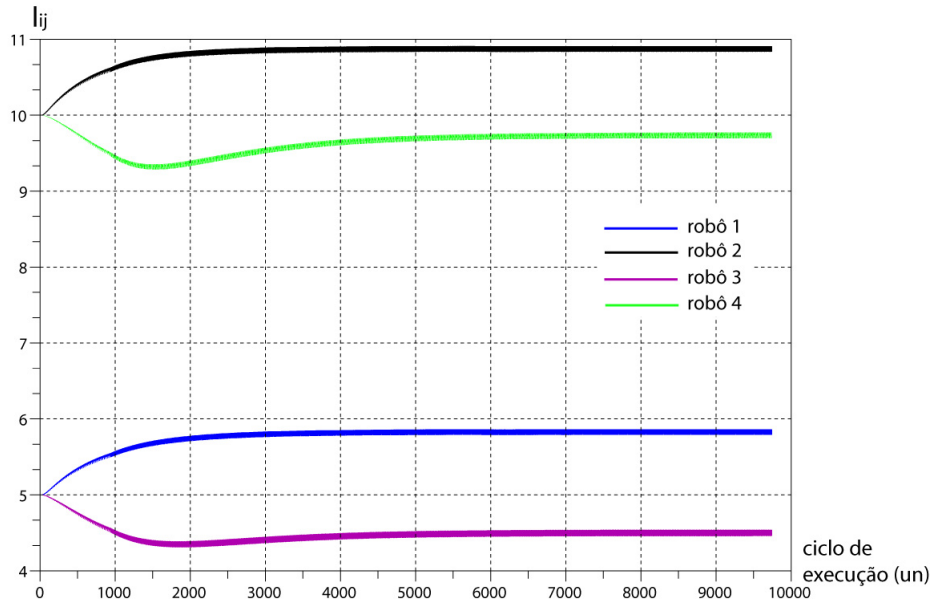

Figura 5.7: Distância relativa dos robôs com relação ao líder por ciclo de execução

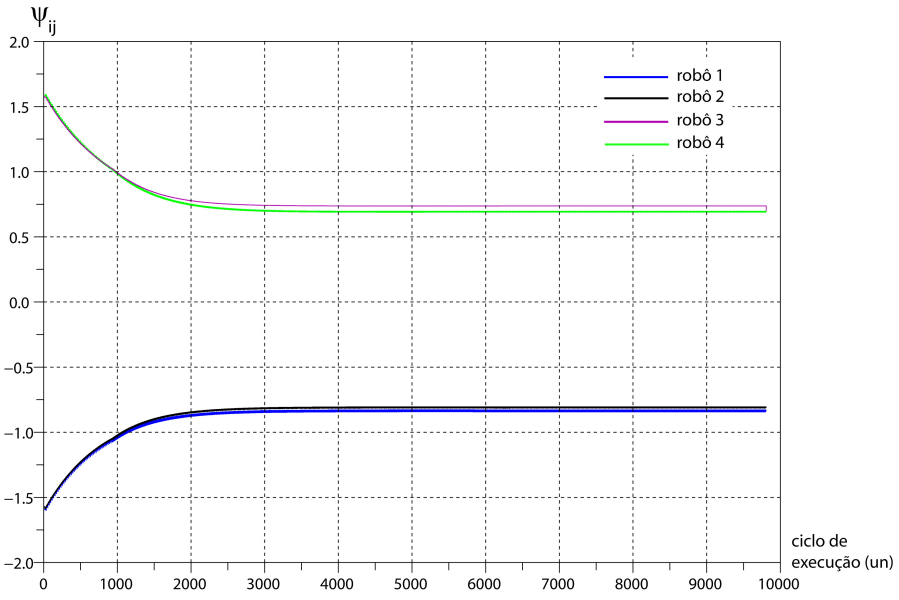

Figura 5.8: Orientação relativa dos robôs com relação ao líder por ciclo de execução

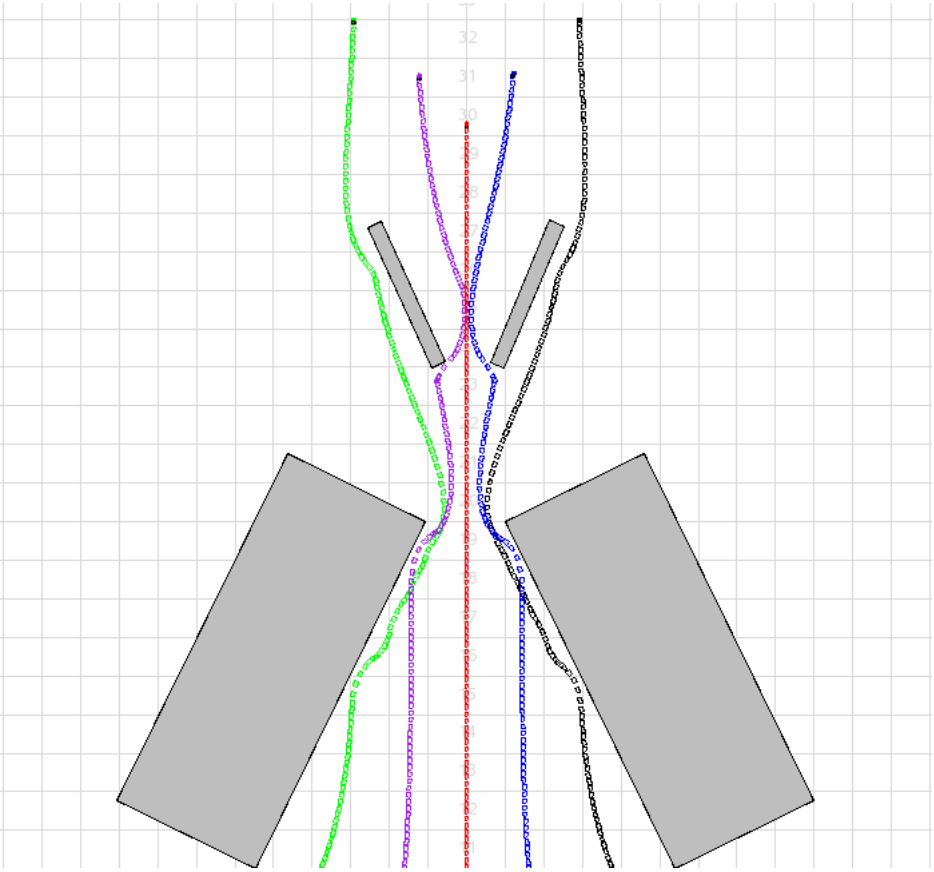

Figura 5.9: Resultado de simulação para desvio de obstáculos

Os parâmetros utilizados foram:

Tabela 5.1: Distâncias e Orientações relativas para os robôs seguidores

	robô 1	robô 2	robô 3	robô 4
l_{ij}^d (m)	2	4	2	4
ψ_{ij}^d (rad)	$-\frac{\pi}{4}$	$-\frac{\pi}{4}$	$\frac{\pi}{4}$	$\frac{\pi}{4}$

Os ganhos aplicados foram $k_1 = 0,07$, $k_2 = 0,1$, $k_0 = 1,5$. O líder seguiu sua trajetória com $v_i = 0,3m/s$ e $\omega_i = 0,0rad/s$. Utilizou-se o valor de 40cm como a distância máxima detectável pelo sensor e a distância desejada para a parede $\delta_d = 0.35cm$.

Pode-se observar que o algoritmo implementado consegue gerar de maneira satisfatória um desvio dos obstáculos e retornar ao modo de controle anterior de maneira estável.

5.4 Mudança de Formação supervisionada

No sistema supervisionado, foram geradas diversas formações entre as quais é possível alternar durante a operação e, assim, melhor utilizar a distribuição dos robôs para navegar. Foram implementadas quatro formações com 3 robôs e o sistema foi ensaiado em diversas situações diferentes, mostrando-se que é possível utilizar a formação para obter uma melhor movimentação pelo ambiente, podendo-se atingir regiões do espaço ou aproveitar melhor a região do espaço para exploração.

As quatro formações implementadas foram: coluna, linha, cunha e V. Elas estão apresentadas nas figuras 5.10, 5.11, 5.12, 5.13.

Figura 5.10: Robôs em formação coluna

Em primeiro lugar, foi feito um teste quanto à estabilidade e velocidade das transições entre algumas das formações. Como exemplo, tem-se nas figuras 5.14,

Figura 5.11: Robôs em formação linha

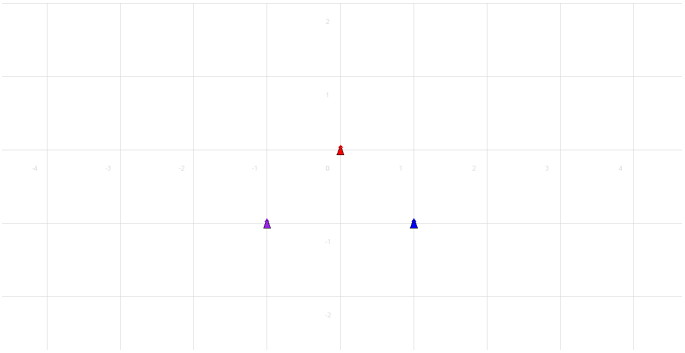

Figura 5.12: Robôs em formação cunha

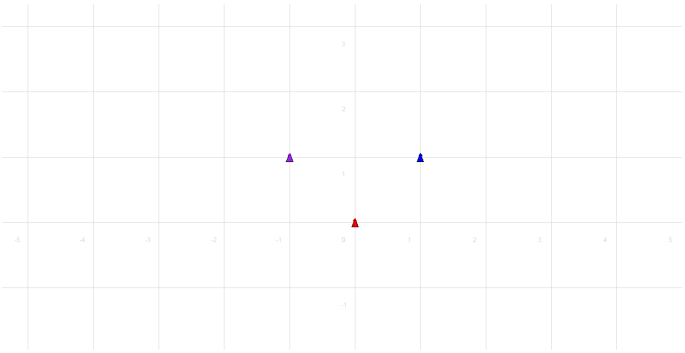

Figura 5.13: Robôs em formação V

5.15, 5.16, as transições entre a formação linha para as formações coluna, cunha e V, respectivamente. É possível notar que o sistema se estabiliza com razoável precisão, o que foi feito em tempo hábil (estabilização em torno de 3s). Como o sistema provavelmente estará em constante movimento, apresenta-se também , nas figuras 5.17, 5.18, 5.19, exemplos de transição em movimento, o que têm desempenho melhor ainda, notando-se uma boa convergência para a formaçao desejada.

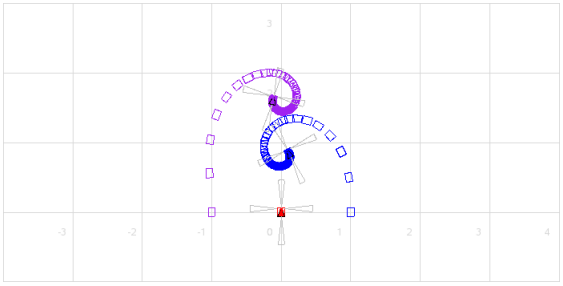

Figura 5.14: Transição estática de formação entre linha e coluna

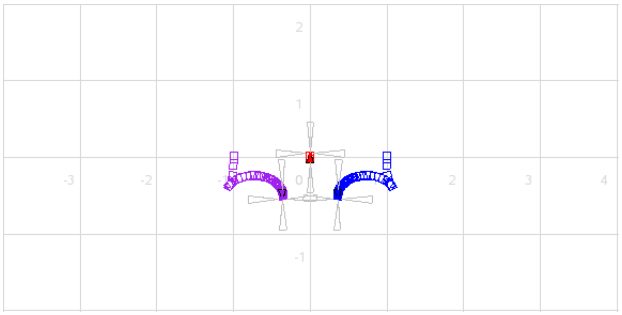

Figura 5.15: Transição estática de formação entre linha e cunha

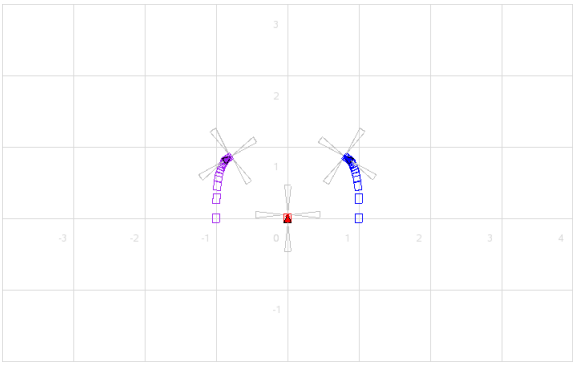

Figura 5.16: Transição estática de formação entre linha e V

Em seguida, o sistema foi testado num corredor hipotético gerado no simulador. Inicia-se numa formação em linha, a qual, pela sua largura, não permite que os robôs ultrapassem o corredor. Dessa maneira, ligeiramente antes de se chegar à entrada do corredor, a formação é alterada para a formação em coluna, adequada para tal situação. Após o corredor ter sido ultrapassado, a formação é novamente alterada para a formação em V. Um exemplo está na figura 5.20.

Como exemplo e teste final, o sistema foi testado em dois ambientes aleatórios simulando ambientes reais com diversas salas, corredores e portas. Nota-se que é possível utilizar a mudança de formação para atingir regiões que em formação

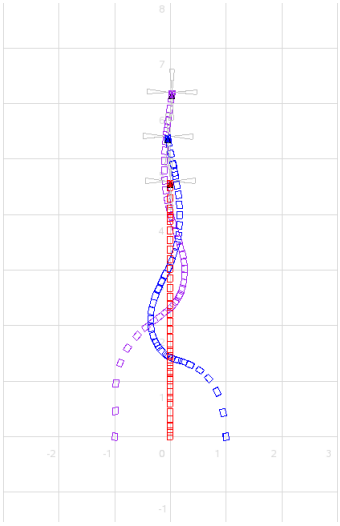

Figura 5.17: Transição dinâmica de formação entre linha e coluna

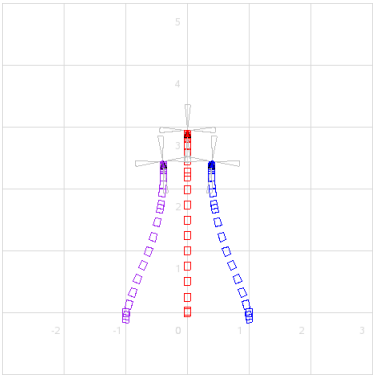

Figura 5.18: Transição dinâmica de formação entre linha e cunha

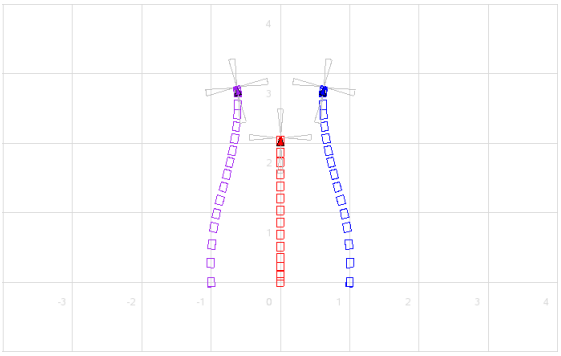

Figura 5.19: Transição dinâmica de formação entre linha e V

fixa não seria possível atingir. Além disso, a distribuição dos robôs pode ser manipulada pelo operador para se obter outros objetivos, como uma distribuição

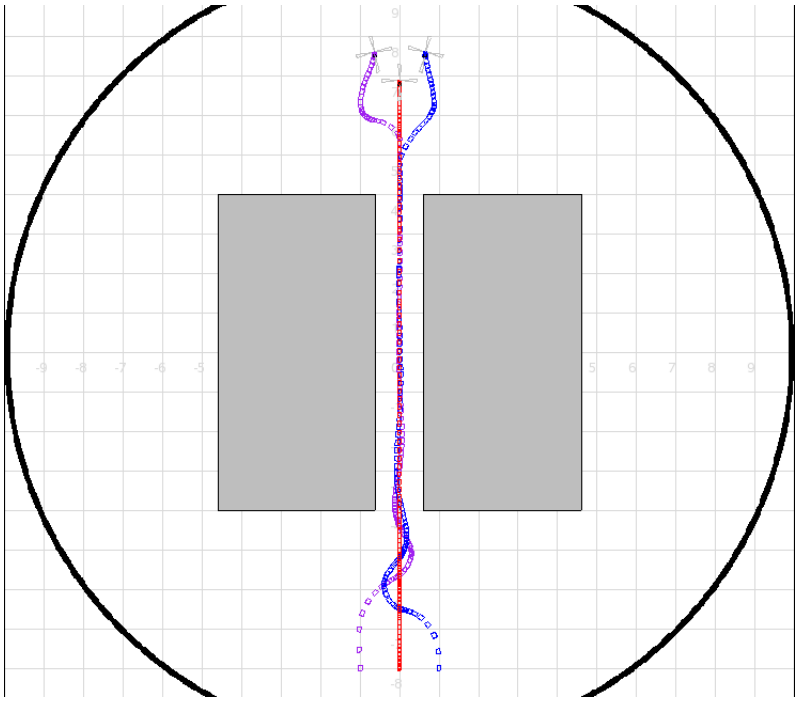

Figura 5.20: Ultrapassagem de corredor com robôs em formação

mais ampla para melhor aproveitar a varredura dos sonares e, assim, explorar mais rapidamente o ambiente. Os resultados estão na figuras 5.21 e 5.22.

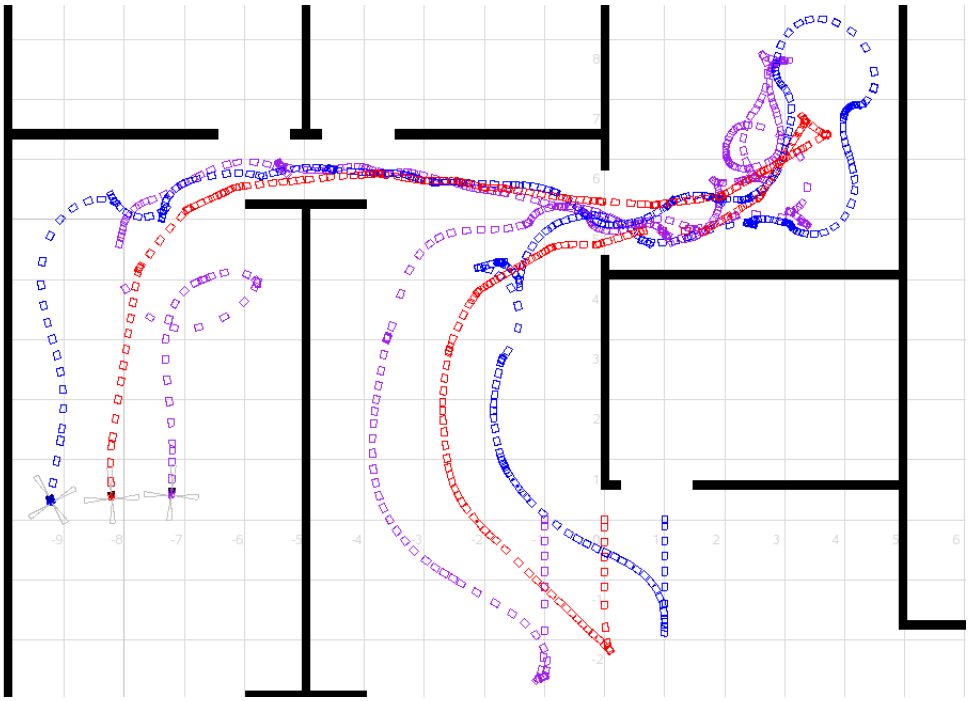

Figura 5.21: Demonstração do sistema em ambiente simulado - exemplo 1

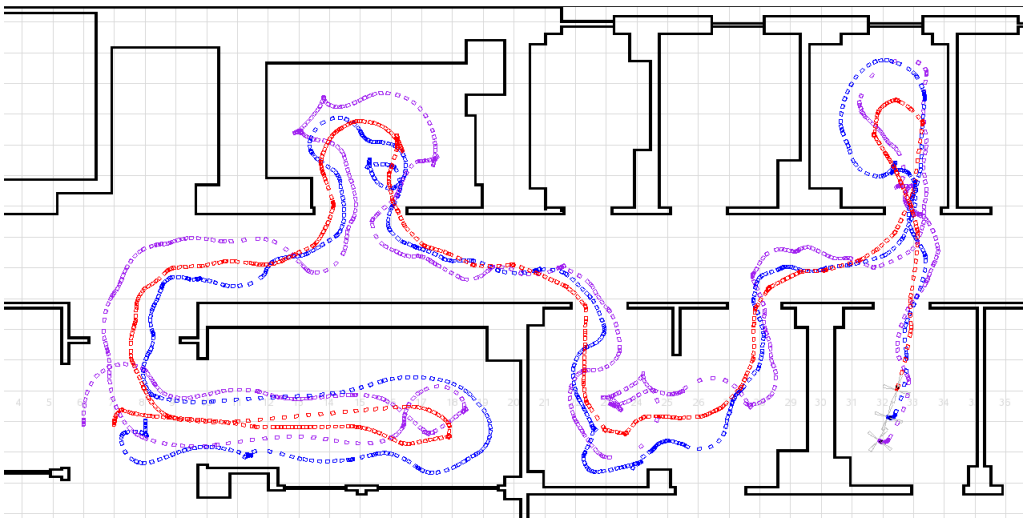

Figura 5.22: Demonstração do sistema em ambiente simulado - exemplo 2

Detalhes das transições entre as formações durante o trajeto estão nas figuras 5.23 e 5.24.

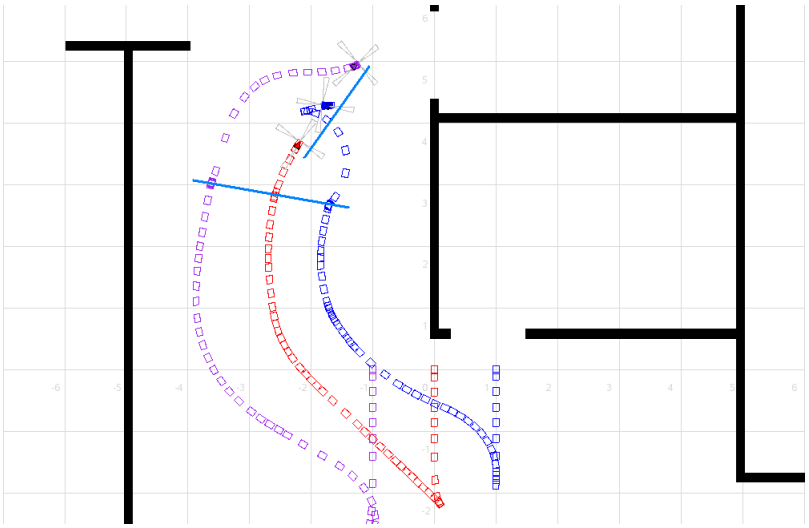

Figura 5.23: Detalhe da transição entre linha e coluna para passagem por corredor

5.5 Mudança de Formação autônoma

A mudança de formação autônoma baseou-se na utilização de uma lógica aplicada sobre os sinais dos sensores para determinar quando é necessário alterar a geometria da formação. O sistema decide entre três formações, progressivamente, linha (5.11), cunha (5.12) e coluna (5.10). Ao realizar essa mudança de geometria,

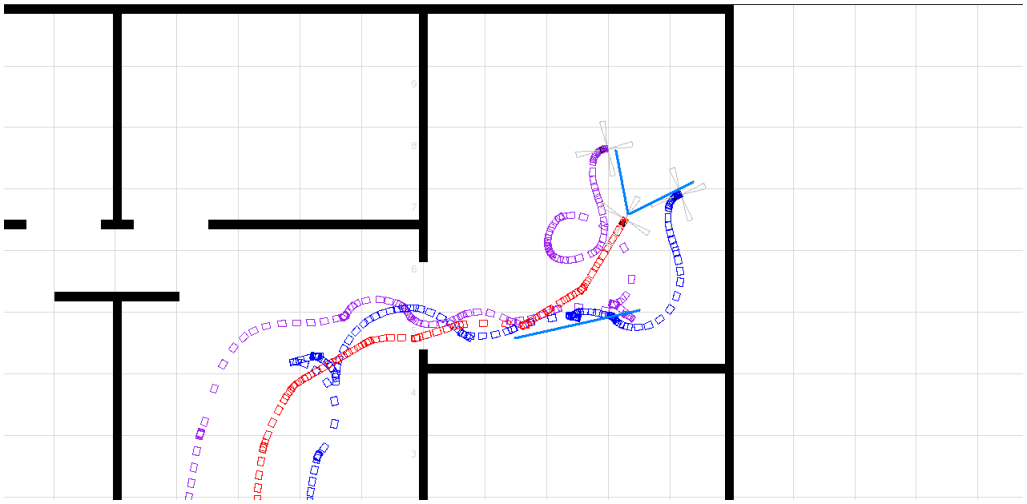

Figura 5.24: Detalhe da transição entre coluna e cunha para exploração da sala

muda-se a área frontal da formação, o que permite ir de uma configuração adequada para exploração para uma configuração adequada para transpor passagens estreitas.

Um exemplo de simulação desse modo está nas figuras 5.25, 5.26 e 5.27. Nessas figuras, é possível perceber os pontos de mudança de formação, indicados por setas nas figuras 5.26 e 5.27. No primeiro caso, trata-se da transição entre a formação cunha e a formação linha. O robô havia mudado sua formação para cunha devido ao obstáculo e, após tê-lo ultrapassado, voltou para a formação em linha. No segundo caso, o robô manteve-se por mais tempo na formação cunha pois, durante a trajetória, detectou a presença de um obstáculo lateral, impedindo a sua volta à formação linha. Somente depois de ter ultrapassado o obstáculo é que o robô voltou para a geometria inicial.

Um outro exemplo está demonstrado na figura 5.28. Nesse caso, ambos os lados da formação estão barrados por obstáculos, de maneira que é impossível continuar com a trajetória sem mudar a geometria. Percebe-se claramente a utilidade do algoritmo implementado. Deve-se notar que todo o comportamento do sistema foi autônomo, exceto pela determinação da trajetória, o que foi feito através de comandos humanos. O progressivo estreitamento dos obstáculos é acompanhado do estreitamento da geometria da formação, pela mudança da formação linha para cunha e depois coluna, o que está indicado na figura.

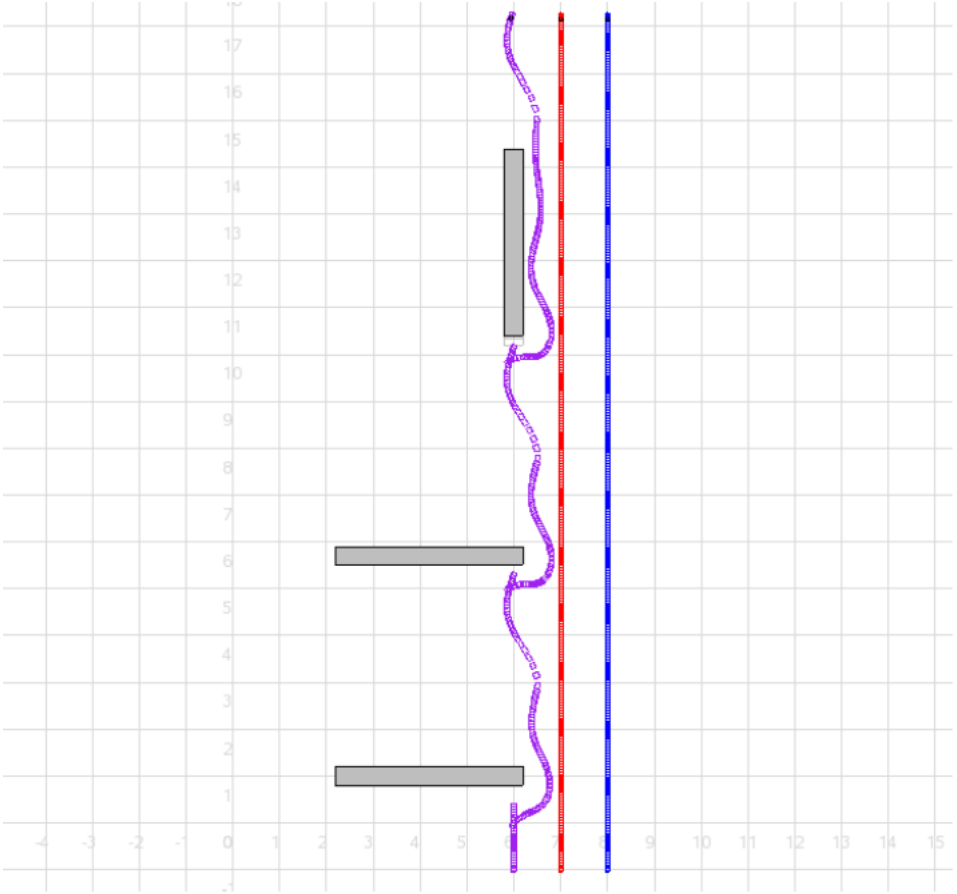

Figura 5.25: Trajetória completa da mudança de formação automática - exemplo 1

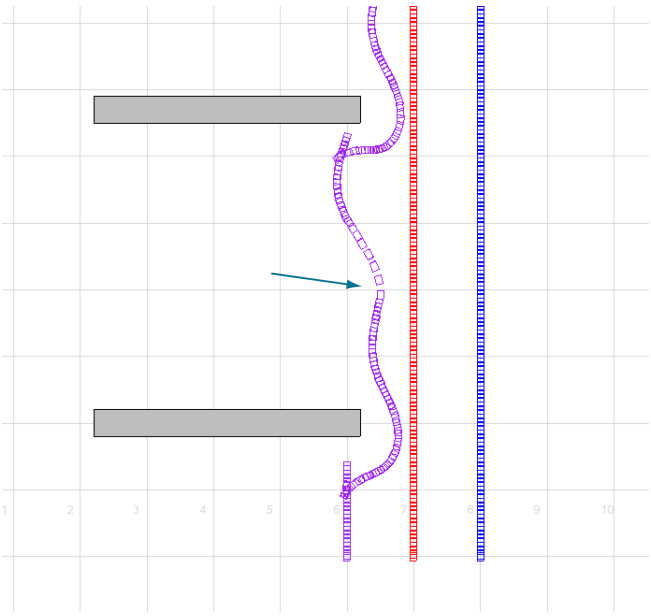

Figura 5.26: Retorno automático à formação original durante a trajetória

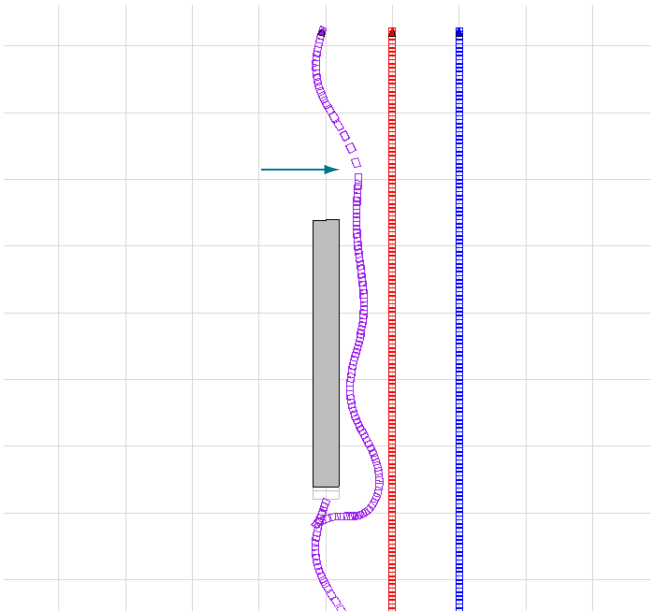

Figura 5.27: Retorno automático à formação original após transpor os obstáculos

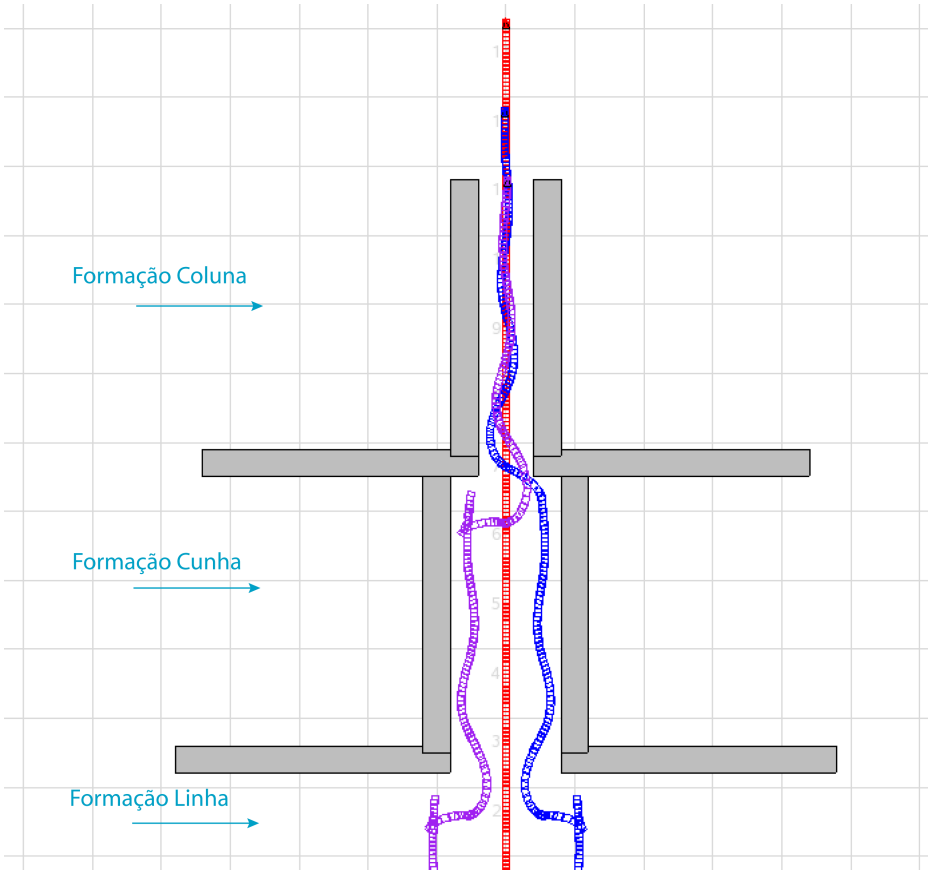

Figura 5.28: Trajetória completa da mudança de formação automática - exemplo 2

Capítulo 6

Conclusão

Notou-se ser possível gerar e manter formações estáveis e fazê-los em tempo hábil. Mesmo já sendo suficiente para diversas aplicações, decidiu-se extender as funcionalidades do sistema, implementando-se mecanismos que permitem alternar entre formações durante a execução de uma atividade qualquer com os robôs. Isso é de particular interesse para navegar por ambientes complexos, porém, do modo como foi implementado, depende-se de um supervisor humano para determinar o momento ideal de realizar as transições de formação.

Tendo em vista essa limitação, estudou-se maneiras de liberar o sistema da necessidade de um agente humano da constante monitoração do sistema. Apesar de não livrar totalmente o sistema do supervisor, foi possível desenvolver uma funcionalidade que determina o momento adequado de se mudar a formação, limitando o supervisor a controlar a trajetória do sistema. Mesmo não sendo uma tarefa simples, esse controlador pode ser substituído por um gerador de trajetórias, tema sobre o qual já existe uma extensa literatura ([2]).

6.1 Trabalhos Futuros

Muito trabalho pode ser ainda acrescentado ao sistema criado, com limites apenas para a criatividade. Enumera-se a seguir algumas das possibilidades para futuros interessados em pesquisar e aprofundar os assuntos e materiais aqui discutidos:

- Desenvolvimento de algoritmo de desvio de obstáculos mais robusto
- Implementação do modelo de erros no sensor
- Simulação de aplicações práticas, como *box-pushing*
- Introdução de novos mecanismos de cooperação, como comunicação

- Implementação nos robôs *Robolab* reais

Referências Bibliográficas

[1] DUDEK, G.; JENKIN, M. "Computational principles of mobile robotics", *Cambridge University Press*, 2000.

[2] LATOMBE, J. C. "Robot Motion Planning", *Kluwer Academic Publishers*, 1991.

[3] DUDEK, G. et. al. "A Taxonomy for Multi-Agent Robotics", *Autonomous Robots*, vol. 3, pp. 375-397, 1996.

[4] CAO Y. U.; FUKUNAGA, A. S.; KAHNG, A. B. "Cooperative Mobile Robotics: Antecedents and Directions", in *Proceedings of 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS' 95)*, 1995.

[5] PARKER, E. L. "Current State of the Art in Distributed Autonomous Mobile Robotics", in *Distributed Autonomous Robotic Systems 4, Proceedings of the 5th International Symposium on Distributed Autonomous Robotic Systems, DARS 2000*, Outubro 2000.

[6] IOCCHI, L.; NARDI, D.; SALERNO, E. "Reactivity and Deliberation: a survey on Multi-Robot Systems", *Balancing Reactivity and Deliberation in Multi-Agent Systems (LNAI 2103)*, Springer, pp. 9-32, 2001

[7] LEWIS, M. A.; TAN, K. "Virtual Structures for High-Precision Cooperative Mobile Control", *Autonomous Robots*, 4:387-403, Outubro 1997

[8] ARKIN, R. C.; BALCH, T. "Behavior-based Formation Control for Multirobot Teams", in *IEEE Transactions on Robotics and Automation*, 1999.

[9] KUMAR, V. et. al., "A Vision-Based Formation Control Framework", in *IEEE Transactions on Robotics and Automation*, Outubro 2002.

[10] GERKEY, B. P.; HOWARD, A.; VAUGHAN, R. T. "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems", in *Proceedings*

of the International Conference on Advanced Robotics, ICAR 2003), Junho 2003.

[11] MICHAUD, F. et. al., "Dynamic robot formations using directional visual perception", in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol.3, 2002

[12] DO, K. D.; PAN, J., "Nonlinear formation control of unicycle-type mobile robots", in *Robotics and Autonomous Systems*, v.55, pp. 191-204, 2007.

[13] Sharp, "GP2D120", Component Data sheet obtido em www.dorukan.com/files/GP2D120-DATA-SHEET.pdf, acessado em 10/10/2007.