UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Aurimar Bezerra Melo de Sousa Filho

Controle de Formação de Robôs Móveis

São Carlos

2018

Aurimar Bezerra Melo de Sousa Filho

Controle de Formação de Robôs Móveis

Trabalho apresentado à Escola de Engenharia de São Carlos da Universidade de São Paulo, para obtenção do título de Engenheiro.

Área de concentração: Engenharia Elétrica com ênfase em Eletrônica

Orientador: Prof Marco Henrique Terra
Coorientador: Maurício Nakai

São Carlos

2018

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO, POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Bezerra Melo de Sousa Filho, Aurimar
Controle de Formação de Robôs Móveis / Aurimar
BB574c Bezerra Melo de Sousa Filho; orientador Marco Henrique Terra; coorientador Maurício Eiji Nakai. São Carlos, 2018.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2018.

1. Controle de formação. 2. Robô móvel. 3. Líder/Seguidor. 4. ePuck. 5. Sistema Vicon. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO

Nome: Aurimar Bezerra Melo de Sousa Filho

Título: "Controle de formação de robôs móveis"

Trabalho de Conclusão de Curso defendido e aprovado em 22 / 06 / 2010,

com NOTA 8,5 (oito, cinco), pela Comissão Julgadora:

Prof. Titular Marco Henrique Terra - Orientador - SEL/EESC/USP

Prof. Adjunto Roberto Santos Inoue - UFSCar

Mestre Mauricio Eiji Nakai - Doutorando - SEL/EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

Este trabalho é dedicado aos meus pais e amigos que me apoiaram nessa jornada.

AGRADECIMENTOS

Aos meus pais pelo amor, incetivo e apoio incondicional que me permitiu morar longe de casa para alcançar meus objetivos.

Aos meus amigos por serem companheiros não só nas horas de diversão, como também nos estudos.

E a todos que direta ou indiretamente fizeram parte dessa jornada, o meu muito obrigado.

"O estudo, a busca da verdade e da beleza são domínios em que nos é consentido sermos crianças por toda a vida."

Albert Einstein

RESUMO

BEZERRA, A. **Controle de Formação de Robôs Móveis**. 2018. 75p. Trabalho de Conlus/ ao de Curso - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2018.

Este trabalho apresenta um estudo sobre o controle de formação aplicado a robôs móveis no arranjo Líder/Seguidor. Isso, por causa do avanço das tecnologias dos robôs móveis e dos veículos autônomos o que os tornam mais populares. Dessa forma, o trabalho desenvolve um software que aplica o controle de formação e do robô móvel, usando 4 robôs móveis ePucks e o sistema de câmeras Vicon. Mas antes, desenvolve um modelo do robô móvel para poder simular todos os controle empregados. Com isso, demonstra a utilização desse tipo de controle com robôs móveis na prática.

Palavras-chave: Controle de formação; Robô móvel; Líder/Seguidor; ePuck; Sistema Vicon;

ABSTRACT

BEZERRA, A. . 2018. 75p. Trabalho de Conlus/ ao de Curso - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2018.

This work presents a study about the formation control applied to mobile robots in the Leader/Follower arrangement. This happens because of the advancement of mobile robot technologies and autonomous vehicles which make them more popular. In this way, the work develops a software that applied the formation control and the mobile robot control, using 4 ePucks mobile robots and the Vicon camera system. But before, it develops a model of the mobile robot to be able to simulate all the control employees. With this, it demonstrates the use of this type of control with mobile robots in practice.

Keywords: Formation control; Mobile robot; Leader/Follower; ePuck; Vicon system;

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

VANT	Veículo aéreo não tripulado
RMR	Robô móvel com roda
EDO	Equação diferencial ordinária
PD	Proporcional Derivativo

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, veículos e robôs autônomos estão se popularizando e tornando-se cada vez mais comuns na sociedade, por causa das muitas funções que podem realizar, principalmente quando trabalham em grupo. Para o trabalho em grupo, a movimentação coordenada desses agentes constitui-se importante para atingir o objetivo desejado. Entretanto, ao aumentar o número de veículos em uma mesma operação, cresce a dificuldade de controle individual e também proporciona movimentos desnecessários ou repetidos, se não existir algo os coordenando. Esse aumento já está acontecendo, por causa dos avanços em tecnologias de alto desempenho, transmissão de dados e posição global, que faz os custos desses veículos diminuírem enormemente. Logo, estudos em controle de formação e cooperação destacam-se, porque atacam diretamente os problemas ao agir como um controle externo para a coordenação de veículos autônomos em grupos.

Ter um futuro com centenas de carros autônomos convergindo de direções diferente para uma mesma via ou um aeroporto só com aviões autônomos é possível, senão uma realidade iminente. Logo, pensar como esses veículos vão interagir e movimentar-se em conjunto para trazer um melhor resultado interessa a comunidade científica que já produz bastantes trabalhos com veículos e robôs autônomos. Exemplos são o trabalho apresentado por Augugliaro et al. (2014) que propõe a utilização de robôs quadricópteros para a construção de um muro de tijolos, demonstrando a utilização de sistemas de cooperação na construção civil e o trabalho apresentado por Quater et al. (2014), onde robôs quadricópteros e dirigíveis são usados para inspeções de placas fotovoltaicas em locais de difícil acesso, demonstrando a aplicação na área de energias sustentáveis. Outros exemplos, são a utilização de formação para entretenimento, como utilizado na abertura dos jogos olímpicos de inverno de 2018, no qual foram usados quadricópteros para formação de desenhos no ar e, também, para comboios de caminhões em que é necessário um caminhão com motorista como líder e os outros, sem intervenção humana, o seguem.

Percebe-se como o controle de formação tem papel importante para a consolidação dos veículos autônomos na sociedade e aumentar a diversidade de sua utilização. Esse controle tem como objetivo tornar os movimentos dos veículos na operação mais simples e eficazes, mais escaláveis e tolerante a falhas.

O controle de formação com robôs autônomos constitui-se de um grupo com robôs espacialmente distribuídos cuja a dinâmica de estado são acopladas em uma lei de controle comum. Essa lei pode ser abordada, principalmente, de três formas: o controle baseado no comportamento, configuração Líder/Seguidor e estrutura virtual (LAWTON; BEARD; YOUNG, 2003). Este trabalho, concentra-se na configuração Líder/Seguidor descrito em Williams, Lafferriere e Veerman (2005) e Lafferriere, Williams e Caughman (2004). Nesta

configuração, o grupo escolhe um líder, enquanto os outros robôs são os seguidores, com uma trajetória resultante da movimentação do líder e da formação (distância entre os membros do grupo e posição angular) escolhida.

No estudo de controle de formação de robôs, um grupo pode ser formado por robôs homogêneos ou heterogêneos, com essa heterogeneidade, podendo ser morfológica ou funcional. Também, os agentes da formação podem compartilhar informações entre si. Um grupo de robôs heterogêneos acontece em uma condição líder/seguidor, onde o líder tem uma diferença funcional, já que o líder não se atenta as condições de seus seguidores, porém os mesmos sabem as condições do líder para coordenar suas ações com a dele. A diferença também pode ser funcional, quando o robô líder tem melhores tecnologias de sensoreamento do que os demais para guiá-los. Um exemplo de uso de robôs heterogêneos com a formação Líder/Seguidor é o uso de RMRs com um robô quadricóptero. Esse segue o líder e dá as posições relativas entre eles a partir de uma posição mais elevada para controle de formação mais preciso, usado por Dorigo et al. (2013).

Esse estudo tem base no exemplo anterior, onde usa-se um grupo de RMRs na configuração Líder/Seguidor e imagem de câmeras para um posicionamento mais preciso de RMRs, dando a posição relativa entre eles a partir de um ponto mais elevados. Com isso, espera-se mostrar o uso do controle de formação para movimentações de grupos em vários casos, mostrando suas vantagens e sua tolerância a falhas. Mais ainda, mostra na prática seu funcionamento, com o uso de robôs ePuck e o sistema de câmeras Vicon (NAKAI et al., 2015).

Este trabalho está organizado da seguinte maneira: Primeiro, tem-se a apresentação dos RMRs usados (robôs ePucks), com suas características físicas e o porque da sua utilização. Logo após, apresenta-se o sistema de câmeras Vicon utilizado, com suas características detalhadas. Com isso, parte-se para os controladores aplicados. Primeiramente, trabalha-se o controle de formação Líder/Seguidor, mostrando a matemática necessária e a lei de controle aplicada. Em seguida, foca-se no controle dos RMRs, desenvolvendo um controlador baseado na cinemática e para obter um controle melhor, desenvolve-se um controle baseado na dinâmica dos RMRs.

Para determinar os ganhos dos controles empregados, precisa-se montar um modelo do robô ePuck para simulação. Com isso, parte-se para a implementação em software, onde é usado o ®MATLAB para obter simulações do controle de formação e do robô, determinando os seus respectivos ganhos. Dessa forma, testa-se esses controles em diversas situações de formações e trajetórias, como também testa-se a tolerância à pertubações, esperando obter erros mínimos entre a trajetória desejada e a simulada do robô ePuck. Em seguida, adapta-se o mesmo software, para conseguir usá-lo em prática e demonstrar a usabilidade desse tipo de controle, com a ajuda dos robôs ePuck e a câmera Vicon, mostrando o resultado em vídeos.

2 MATERIAIS E MÉTODOS

Este capítulo aborda o robô ePuck, o sistema de câmeras Vicon utilizada para realimentação das posições dos RMRs, os controles utilizados e os métodos utilizados para desenvolver os softwares. O software desenvolvido pode ser analisado nos anexos A e B, além dele os softwares usados para comunicação da Vicon e do robô ePuck nos apêndices A e B.

2.1 Materiais

2.1.1 Robô ePuck

O robô ePuck é um robô móvel em miniatura, desenvolvido pela *Ecole Polytechnique Fédérale de Lausanne*, onde eles tentaram reunir em um mesmo robô várias características importante, que o torne excelente para propósitos educacionais em nível universitário (MONDADA et al., 2009). Tais características são:

- Tamanho reduzido com boa estrutura, que possibilita o uso em pequenos espaços, inclusive em mesas e bancadas;
- Variabilidade de aplicação, já que tem um bom potencial de processamento e uma grande quantidade de sensores;
- Interface amigável, possuindo comunicação bluetooth que facilita a comunicação com o computador e a gravação de *firmware* sem cabos;
- Baixo custo, para maior acessibilidade pelas instituições de ensino;
- Projeto aberto, devido seu uso por diversos laboratórios, escolas e universidades;

Isso mostra, o por que de ser usado nessa pesquisa. Além disso, ele pode ser usado em outras áreas, como par exemplo em processamento de sinais, programação em tempo real e interação homem-máquina, mostrando sua versatilidade .

A estrutura do robô constitui-se em uma única peça de polímero que acomoda os motores, as duas rodas, a bateria e a placa de circuito, como mostrado na Figura 1. O ePuck tem um motor de passo com engrenagem de redução, que tem uma revolução de 20 passos e uma redução de 50:1. A roda tem diâmetro de 41mm, com uma distância entre elas de 53mm. A velocidade máxima do robô corresponde a 1000 passos por segundo aproximadamente, o que equivale a uma revolução da roda por segundo.

Figura 1: Robô ePuck

Para o correto funcionamento dos controle empregados, deve atentar-se as características físicas do robô ePuck, com isso mediu-se e calculou-se as medidas mostradas na Tabela 1.

Tabela 1: Características físicas do robô ePuck

Característica	**Valor**
Raio da roda	$20,5mm$
Centro do eixo até a roda	$26.5mm$
Massa do robô	$144g$
Distância entre centro de massa e centro do eixo	$0,5mm$
Momento de inércia da plataforma	$68,156Kg^2mm$
Momento de inércia da rodas em relação ao eixo da roda	$1,314Kg^2mm$
M. de inércia da roda em relação ao eixo no plano da roda	$0,676Kg^2mm$

2.1.2 Sistema Vicon

O sistema de câmeras Vicon consiste em um grupo de câmeras de alta resolução posicionadas para cobrir vários ângulos de um ambiente. Essas câmeras, quando corretamente posicionadas, conseguem a localização de um ponto no sisteme de coordenadas cartesianas (x, y, z), como também os ângulos de rotação sobre esses mesmos eixos. As câmeras Vicon utilizam-se da luz infravermelha, produzidos ao redor da lente, como pode ser observado na Figura 2, fazendo com que o rastreio dos pontos não dependam da iluminação do ambiente, desde que não hajam interferências no espectro infravermelho.

Figura 2: Câmera Vicon

A localização dos pontos desejados são feitos através de marcadores reflexivos em forma de esfera. Para diferenciar os pontos, é feita uma identificação a partir do padrão dos marcadores colocado no objeto(distância entre os marcadores). Dessa forma, consegue-se ter uma variedade de objetos em rastreamento, possibilitando o estudo do controle de formação.

As câmeras Vicon fazem o tratamento das imagens em uma plataforma própria, na qual se comunicam via cabo coaxial. Para se comunicar com o servidor, usa-se um cabo ethernet, ligado a uma placa de rede. Isso garante uma aquisição de dados e transmissão de dados com alta velocidade, o que permite o sistema Vicon ser usado em tempo real. Com isso, o servidor disponibiliza as posições dos objetos desejados via software através da rede local. Mais características são apresentadas na Tabela 2.

Tabela 2: Características da câmera Vicon

Câmera	T40S
Resolução	4Mp (2336 x 1728)
Máxima taxa em máxima resolução	525 fps
Sensor	Vicon Vegas S-4
Lentes	PENTAX modelo C21211KP

Esse sistema tem várias vantagens, como alta precisão posicional e angular, através de seleção e fusão de todos os dados gerados pelas câmeras, opções de baixo custo, podendo ser usado por ambientes educacionais, pode ser usado em vários ambientes, trazendo mais variabilidade de uso, opções com baixa latência para o uso em rastreio de robôs e amplamente integrado com várias aplicações que permitem seu uso com diferente

softwares. Além disso, a Vicon é amplamente usado por grandes empresas e organizações, como a ®Boing, ®Airbus, ®Ford, ®BMW, ®UPENN e ®NASA, em uma grande variedade de aplicações, como ergonomia, engenharia e pesquisa. Logo, percebe-se que o sistema Vicon constitui-se em um sistema de ponta de câmeras, com uma grande confiabilidade, sustentado não só por suas características técnicas, como também pela seu uso no mercado.

2.2 Modelo de Formação

O modelo de formação tem como objetivo gerar as trajetórias dos robôs móveis para que mantenham posições fixas em relação aos outros robôs. Com isso, apresenta-se o controlador descentralizado, onde as posições e velocidades dos N robôs são descritas pela seguinte equação de estado:

$$\dot{x}_i = A_{veh}x_i + B_{veh}u_i. \tag{2.1}$$

Para $i = 1, 2, ..., N$ e $x \in \mathbb{R}^{2n}$, sendo x_i as posições do i-ésimo RMR e suas respectivas velocidades e u_i as entradas de controle. A matrizes A_{veh} e B_{veh} têm a seguinte forma:

$$A_{veh} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & a_{22} & 0 & a_{24} \\ 0 & 0 & 0 & 1 \\ 0 & a_{42} & 0 & a_{44} \end{pmatrix}, B_{veh} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}. \tag{2.2}$$

Sendo a_{ij} constantes relacionadas a trajetória desejada para o robô líder. As colunas 1 e 3 estão zeradas para garantirem a convergência dos robôs para a formação (LAFFERRIERE; WILLIAMS; CAUGHMAN, 2005)

O vetor $x = (x_1, x_{2,...,x_N})^T$ descreve os estados de cada RMRs. Assim, tem-se $x_p = ((x_p)_1, ..., (x_p)_N)^T$ e $x_v = ((x_v)_1, ..., (x_v)_N)^T$, denotando os vetores de posição e velocidade respectivamente, obtendo:

$$x = x_p \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} + x_v \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.3}$$

A matriz de formação é:

$$h = h_p \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} \in \mathbb{R}^{2n}. \tag{2.4}$$

Um dígrafo Γ representa a topologia de comunicação entre os robôs que consiste de vértices V e arestas E. Cada vértice representa um robô e as arestas representam a comunicação entre os mesmos, com o par i representando o RMR e j o seu vizinho, sendo J_i o número de vizinhos do i-ésimo RMR. A matriz de adjacência de Γ é uma matriz quadrada Q, sendo:

$$q_{ij} = \begin{cases} 1, & \text{se } (j,i) \in E \\ 0, & \text{caso contrário} \end{cases}, (i, j \in V). \tag{2.5}$$

A matriz Laplaciana L é dado por:

$$L = L_G \otimes I_{2n},\ com\ L_G = D^{+}(D - Q). \tag{2.6}$$

Sendo D a matriz diagonal, D^{+} é sua matriz pseudo inversa e a matriz Q define a topologia de comunicação entre os robôs da formação, ou seja, a hierarquia líder/seguidor (VEERMAN et al., 2005). Nessa topologia, o líder não recebe informações dos outros integrantes da formação, o que significa que os outros integrantes da formação são forçados a se movimentar em função do líder, como mostrado na Figura 3

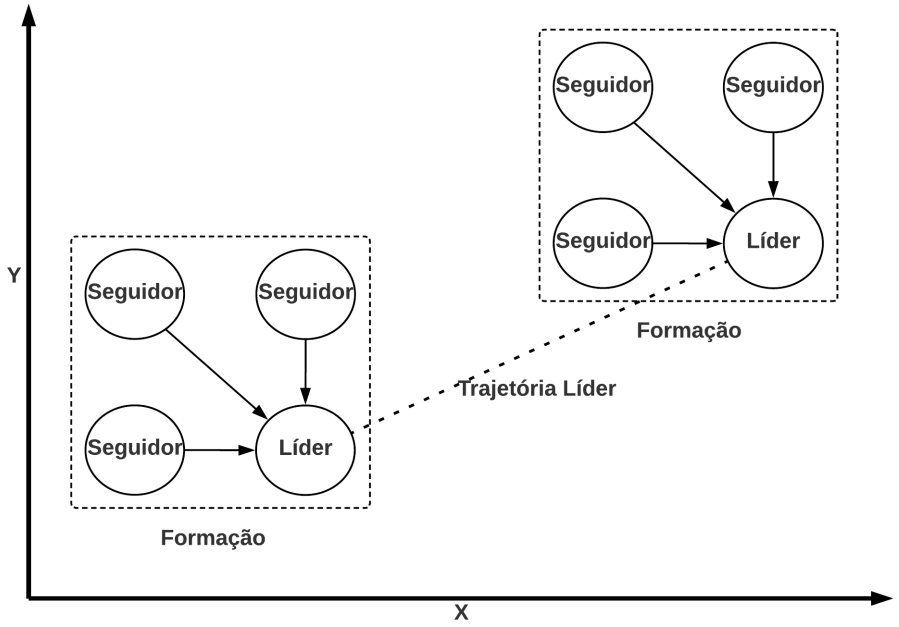

Figura 3: Formação com 4 veículos

Para a convergência da formação h, é necessário uma entrada de controle u. Para isso, tem-se que o erro de saída z_i é calculado como a média do posicionamento relativo da vizinhança.

$$z_i = (x_i - h_i - \frac{1}{J_i}\sum_{j \in J_i}(x_j - h_j)). \tag{2.7}$$

Como resultado o vetor z de saída pode ser escrito como:

$$z = L(x - h). \tag{2.8}$$

Sendo L a matriz Laplaciana do dígrafo de comunicação descrita em (2.6). Logo, A lei de controle é formada a partir de um realimentação F, da seguinte forma:

$$u = Fz = FL(x - h). \tag{2.9}$$

Com a Equação (2.9) em (2.1) obtém-se:

$$\dot{x} = Ax + BFL(x - h). \tag{2.10}$$

Com, $A = I_N \otimes A_{veh}$ e $B = I_N \otimes B_{veh}$, sendo I_N a matriz identidade de ordem N. Dessa forma, levando em consideração a estrutura de A, B e L e que a matriz $F = I_N \otimes F_veh$, tem-se:

$$F_{veh} = \begin{pmatrix} f1 & f2 & 0 & 0 \\ 0 & 0 & f1 & f2 \end{pmatrix}. \tag{2.11}$$

Em Williams, Lafferriere e Veerman (2005), demonstra-se que $f1 < 0$ e $f2 < 0$ são condições necessárias para a convergência da formação.

2.3 Controle Cinemático

A lei de controle da cinemática usada no projeto foi proposta por Kanayama et al. (1990). É dada por:

$$\begin{aligned} v^d &= v_r cos(\alpha_e) + k_x x_e, \\ w^d &= w_r + v_r(k_y y_e + k_\alpha sin(\alpha_e)). \end{aligned} \tag{2.12}$$

Sendo v^d a velocidade linear do RMR desejada, w^d é a velocidade angular desejada, k_x, k_y e k_α são os ganhos de realimentação para os erros x_e, y_e e α_e nas coordenadas polares x e y e no ângulo de orientação do robô respectivamente, dado por:

$$\begin{aligned} x_e &= (x_r - x_c)cos(\alpha_c) + (y_r - y_c)sin(\alpha_c), \\ y_e &= -(x_r - x_c)sin(\alpha_c) + (y_r - y_c)cos(\alpha_c), \\ \alpha_e &= \alpha_r - \alpha_c. \end{aligned} \tag{2.13}$$

Sendo r os valores de orientação futuros e c os valores atuais. Com isso, tem-se:

$$\begin{aligned} v_r &= \sqrt{\dot{x}_r^2 + \dot{y}_r^2}, \\ \alpha_r &= tg^{-1}(\frac{\dot{y}_r}{\dot{x}_r}), \\ w_r &= \dot{\alpha}_r. \end{aligned} \tag{2.14}$$

Sendo x_r, y_r, $\dot{x}_r$ e $\dot{y}_r$ dados pela equação de controle (2.10).

Para obter posições atuais através de simulação, utiliza-se a Equação (2.15) e depois integra-se as velocidades obtidas, sendo $c = r/2b$, r é o raio da roda e b é a distância entre o centro e as rodas, $\dot{\theta}_d$ e $\dot{\theta}_e$ as velocidades angulares aplicadas da roda direita e esquerda respectivamente.

$$\begin{aligned} \dot{x}_c &= c(bcos(\alpha_c) - dsin(\alpha_c))\dot{\theta}_d + c(bcos(\alpha_c) + dsin(\alpha_c))\dot{\theta}_e, \\ \dot{y}_c &= c(bsin(\alpha_c) + dcos(\alpha_c))\dot{\theta}_d + c(bsin(\alpha_c) - dcos(\alpha_c))\dot{\theta}_e, \\ \dot{\alpha}_c &= c(\dot{\theta}_d - \dot{\theta}_e). \end{aligned} \tag{2.15}$$

O controlador baseado na dinâmica baseia-se nas velocidades angulares das rodas. Dessa forma, a relação entre as velocidades é dada por:

$$\dot{q}_2^d = \begin{pmatrix} \dot{\theta}_d^d \\ \dot{\theta}_e^d \end{pmatrix} = \begin{pmatrix} 1/r & b/r \\ 1/r & -b/r \end{pmatrix} \begin{pmatrix} v_i^d \\ w_i^d \end{pmatrix}. \tag{2.16}$$

Sendo θ_d^d e θ_e^d as velocidades angulares desejadas da roda direita e esquerda respectivamente.

2.4 Controle Dinâmico

Nesse projeto usou-se o controle PD independente para cada roda, com realimentação u dada por:

$$\begin{aligned} u_d &= -(\theta_d - \theta_d^d)k_{p1} - (\dot{\theta}_d - \dot{\theta}_d^d)k_{d1}, \\ u_e &= -(\theta_e - \theta_e^d)k_{p2} - (\dot{\theta}_e - \dot{\theta}_e^d)k_{d2}. \end{aligned} \tag{2.17}$$

A equação do torque controlado do modelo dinâmico é:

$$\tau = B^-1(\ddot{q}_2^d - A(\dot{q}_2)\dot{q}_2^d + u). \tag{2.18}$$

Com, $A(\dot{q}_2) = -M_2^{-1}C_2(\dot{q}_2)$, $B = M_2^{-1}$ e as matrizes M_2 e C_2 dadas por:

$$\begin{aligned} M_2 &= S_c(q1)^T M(q_1) S_c(q1), \\ C_2 &= S_c(q_1)^T C(q, \dot{q}) S_c(q1) + S_c(q_1)^T M(q_1) S_c(q_1). \end{aligned} \tag{2.19}$$

Sendo a matriz S_c que transforma as velocidades angulares das rodas nas velocidades atuantes no centro de massa do RMR, C a matriz das forças coriolis e centrípeta, M a matriz de inércia e $q_1 = [x_c \ y_c \ \alpha_c \ \theta_d \ \theta_e]^T$. As matrizes foram baseadas em Coelho e Nunes (2003).

$$\begin{aligned} S(q_1)^T &= \begin{bmatrix} bcos(\alpha_c) - dsin(\alpha_c)) & c(bcos(\alpha_c) + dsin(\alpha_c)) \\ c(bsin(\alpha_c) + dcos(\alpha_c)) & c(bsin(\alpha_c) - dcos(\alpha_c)) \\ c & -c \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \\ C(q, \dot{q}) &= \begin{bmatrix} \begin{bmatrix} 0 & 0 & md\dot{\alpha}_c cos(\alpha_c) & 0 & 0 \\ 0 & 0 & md\dot{\alpha}_c sin(\alpha_c) & 0 & 0 \end{bmatrix} \\ 0_{3x5} \end{bmatrix}, \\ M(q) &= \begin{bmatrix} m & 0 & mdsin(\alpha_c) & 0 & 0 \\ 0 & 0 & -mdcos(\alpha_c) & 0 & 0 \\ mdsin(\alpha_c) & -mdcos(\alpha_c) & I & 0 & 0 \\ 0 & 0 & 0 & I_w & 0 \\ 0 & 0 & 0 & 0 & I_w \end{bmatrix}. \end{aligned} \tag{2.20}$$

Sendo os parâmetro m e I dados por $m = mc + 2m_w$ e $I = I_c + 2m_w(d^2 + b^2) + 2I_m + m_c d^2$, com m_w a massa da plataforma do robô, I_c o momento de inércia da plataforma em relação ao eixo vertical, I_w o momento de inércia da roda e I_m o momento da inércia da roda em relação ao eixo definido no plano da roda.

Baseado em Coelho e Nunes (2003), tem-se o cálculo da velocidade angulares das rodas a partir do torque obtido na Equação (2.18) e a utilização da equação diferencial apresentado em (**??**), considerando a existência de distúrbios $w_i = [w_d \ w_e]^T$

$$\ddot{q}_2 = A(\dot{q}_2)\dot{q}_2 + B\tau_i + Bw_i. \tag{2.21}$$

2.5 Métodos

Para montar o controle de formação apresentado, foi utilizado o ®MATLAB para a construção do software desejado. Ele foi escolhido por ser uma ferramenta altamente conhecida na área de engenharia e pesquisa e também por facilitar a conectividade com o robô ePuck e a câmera Vicon. Isso propicia os testes necessários para ratificar a teoria estudada.

Para teste, é utilizado uma sala com 4 câmeras Vicon instaladas em cada canto superior para obtenção das imagens dos quatros robôs ePucks usados na formação, que são processadas pelo software Tracker Vision em um servidor. Esse software envia os dados de posição para os algoritmos de controle.

2.5.1 Esquema Geral

O algoritmo de controle implementado começa com o cálculo das trajetórias dos robôs seguidores, a partir de suas posições iniciais, velocidades e pela trajetória do líder. Com isso, os controle do robôs calculam as velocidades necessárias de cada roda para que cada robô siga a trajetória estipulada. Essas velocidades são adaptadas para o motor de passos dos robôs ePucks e assim são enviadas via bluetooth para cada um. Finalmente, a Vicon realimenta o controle com as posições dos robôs para que os controles atuem no caso de pertubações ou desvios. O algoritmo é melhor representado pelo fluxograma mostrado na Figura 4.

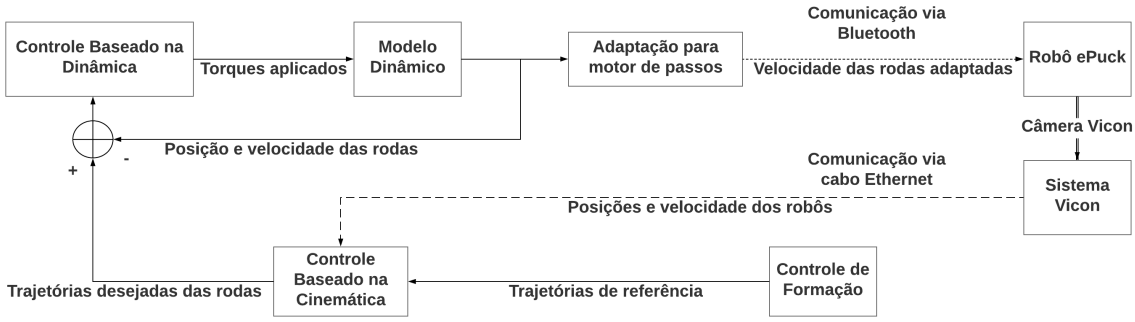

Figura 4: Fluxograma do algoritmo implementado

Entretanto, para construção do software e ajustes de ganhos dos controladores, precisa-se de um ambiente de simulação. Para isso, utiliza-se um modelo cinemático do robô ePuck que calcula a posição dos robôs, usando a velocidade das rodas calculadas pelos controladores em um tempo determinado. Esse novo algoritmo é mostrado na Figura 5.

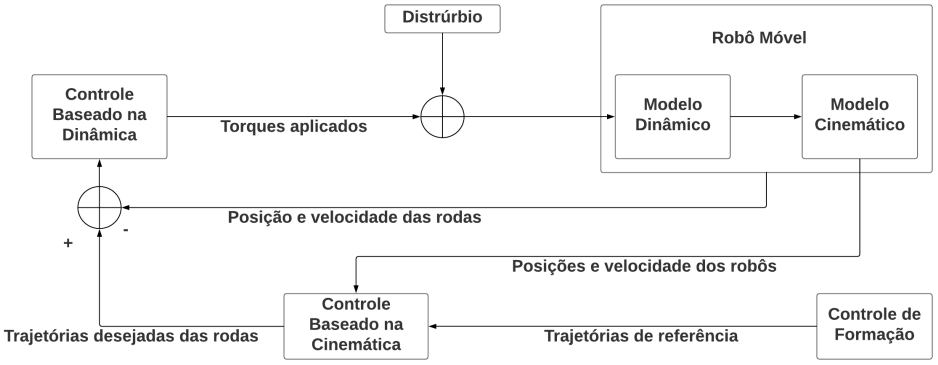

Figura 5: Fluxograma do algoritmo implementado para simulação

Para simular com maior proximidade do teste real, é necessário determinar o tempo de um ciclo do programa. Esse ciclo constitui-se da aquisição de dados da Vicon e transmissão para o software, os cálculos dos controladores e a transmissão dos dados para todos os robôs ePucks. Desta maneira, testa-se quanto tempo essas etapas levam, chegando a uma média de 0,4 segundos por ciclo. Outro fato que ocorre consisti na adaptação da velocidade angular das rodas para a frequência do motor de passos. Logo, usa-se um valor de adaptação linear, aplicado na velocidade angular das rodas. Calcula-se esse valor através da Equação (**??**). Com essa equação e as características do robôs ePuck mostradas, tem-se um valor de adaptação $q_{adpt} = 158$ aproximadamente.

$$q_{adpt} = \frac{velocidade\ máxima\ em\ passos}{velocidade\ linear\ máxima}(raio\ da\ roda). \tag{2.22}$$

2.5.2 Algoritmo do Controle de Formação

O algoritmo do controle de formação é, relativamente, simples para implementação em software. Isso, porque consiste em implementar a Equação (2.10), uma vez que o ®MATLAB já fornece ferramentas para a resolução de EDO. Dessa forma, só precisa-se determinar as matrizes necessárias para a resolução da equação e os ganhos da matriz de realimentação. O algoritmo é mostrado na Figura 6.

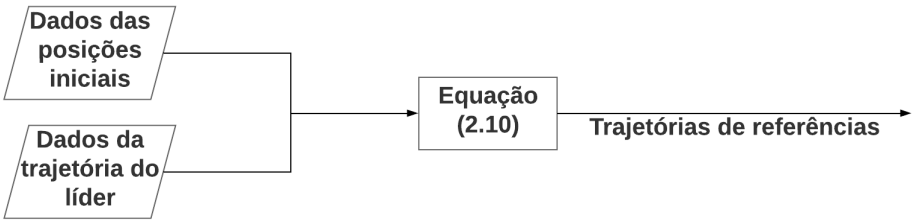

Figura 6: Fluxograma do algoritmo do controle de formação.

O controle de formação implementado é o Líder/Seguidor já comentado, ou seja, os seguidores reconhecem a posição do líder e não se comunicam entre si. Logo, a topologia de comunicação usado para a montagem da matriz de adjacência do dígrafo de comunicação Q e da matriz diagonal D é representado na Figura 7.

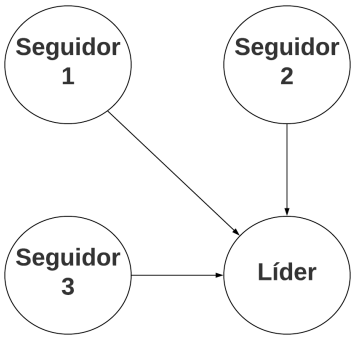

Figura 7: Topologia de comunicação

Com isso, atenta-se a matriz A_{veh}, porque os valores de a_{22}, a_{24}, a_{42} e a_{44} devem ser determinados. Isso, porque pode gerar comportamentos indesejados. Primeiramente, as constantes devem ser iguais para obter o mesmo efeito. Logo, com $a_{22} < 0$ obtém-se uma desaceleração uniforme dos robôs, com $a_{22} > 0$ tem-se uma aceleração uniforme dos robôs e com $a_{22} = 0$ os robôs tendem a alcançar uma velocidade uniforme, mostrado em Williams, Lafferriere e Veerman (2005). O último caso é o buscado para o controle desejado. Desse forma, $a_{22} = a_{24} = a_{42} = a_{44} = 0$.

Para o controle seguir a trajetória determinada pelo líder, tem que definir os valores de $f1$ e $f2$ da matriz F_{veh}. Entretanto, deve-se atentar para as limitações de velocidade das rodas dos robôs ePucks. Logo, a partir das informação do diâmetro das rodas e da velocidade de uma revolução, que são $41mm$ e $1s$ respectivamente, tem-se que a velocidade máxima vale $0.13m/s$. Para a prática, tenta-se manter a velocidade máxima por volta de $0.1m/s$. Com isso, através de várias simulações obtêm-se o valores do $f1 = -0.1$ e $f2 = -1$.

As formações usadas nas simulações e prática feita são especificadas nos resultados desse trabalho. Entretanto, a realimentação é construída para funcionar com qualquer formação escolhida e não depende das posições iniciais dos robôs.

2.5.3 Algoritmo Controle Cinemático

O algoritmo para a implementação do controle cinemático é resultado do cálculo da Equação (2.14), a partir das informações do controle de formação. Com isso e mais as posições atuais dos robôs, calcula-se os erros das coordenadas polares e do ângulo de orientação na Equação (2.13). Após isso, usa-se as informações de velocidades do controle de formação mais os erros para aplicá-los na Equação (2.12). Por fim, tanto o controle baseado na dinâmica, como os robôs ePuck, utilizam-se das velocidades angulares das rodas. Por consequência, utiliza-se a Equação (2.16) para fazer essa transformação. O algoritmo é melhor observado na Figura 8.

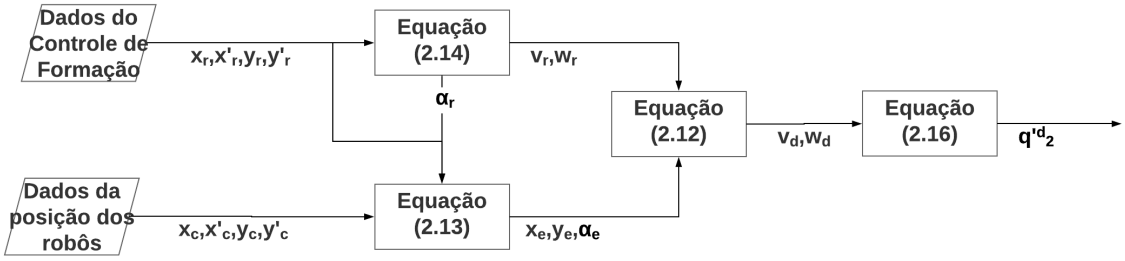

Figura 8: Fluxograma do algoritmo do controle cinemático

Nesse algoritmo e em outros, existe derivadas para calcular. Entretanto, não é necessário usar nenhum código de cálculo de derivada, porque o programa foi implementado em etapas, ou seja, o sistema torna-se discreto. Dessa forma, calcula-se a velocidade, como a diferença das distâncias percorridas pelo tempo. Esse mesmo cálculo discreto encontra-se usado em integrais que, futuramente, aparecem no código.

Para o cálculo dos ganhos k_x, k_y e k_α usados na lei de controle, usa-se o mesmo método usado, anteriormente, no controle de formação. Portanto, atingi-se os valores de $k_x = 0,1$, de $k_y = 100$ e de $k_\alpha = 35$.

2.5.4 Algoritmo Controle Dinâmico

O controle baseado na dinâmica tem uma maior complexidades, exigindo maior atenção. Primeiramente, recebe-se as velocidades das rodas desejada e as reais para integrá-las, usando o método já discutido anteriormente. Assim, pode-se usar a Equação (2.17) para determinar a realimentação. Após isso, determina-se as matrizes usadas para o cálculo do torque, descritas nas equações (2.19) e (2.20) e, depois, usando a Equação (2.18) para obter o torque. Tem-se um fluxograma para observar melhor o algoritmo, mostrado na Figura 9.

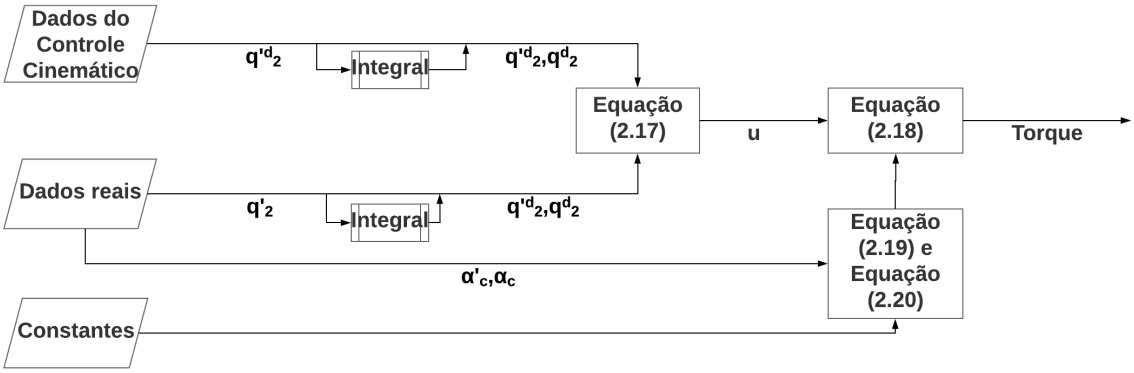

Figura 9: Fluxograma do algoritmo do controle dinâmico

Os ganhos para a roda direita e esquerda são iguais e para determiná-los usa-se métodos empíricos em simulação. Logo, obteve-se $k_p = 0.00001$ e $k_d = 0.00001$.

Para a simulação, tem-se um acréscimo de uma pertubação logo após o cálculo do torque. Isso é feito para testar a tolerância a pertubações do controle PD. Esse teste é feito em simulação para garantir o funcionamento do controle baseado na dinâmica no modelo real. Esse acréscimo da pertubação é mostrado anteriormente na Figura 5.

2.5.5 Modelos Dinâmico e Cinemático

O modelo dinâmico resulta da aplicação de uma EDO na Equação (2.21), utilizando o torque obtido no controle dinâmico. Por isso obtém-se as velocidades angulares das rodas que vão ser aplicados nos robôs ePucks.

Na simulação, ainda é necessário um modelo cinemático do robô ePuck para, a partir das velocidades angulares das rodas, calcular a velocidade linear de cada eixo e a velocidade angular, mostrado na Equação (2.15). Assim, com o tempo de um ciclo que já foi determinado, pode-se obter as posições futuras dos RMRs. Essas posições são usadas para realimentar o controle cinemático, continuando o algoritmo.

Observa-se o fluxograma desses modelos na Figura 10.

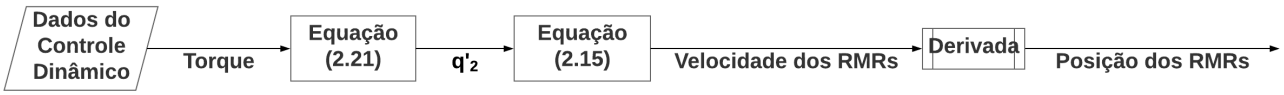

Figura 10: Fluxograma do modelo cinemático do robô ePuck

3 RESULTADOS E DISCUSSÕES

3.1 Resultados

Nessa seção, apresenta-se os resultados simulados e práticos do controle implementado.

3.1.1 Resultado Simulado

Para a simulação, fez-se vários testes com diferentes tipos de formação, tipos de trajetória do líder e diferentes posições iniciais dos robôs. Além disso, fez-se o acréscimo de pertubações como já mostrado. Todas essas condições são impostas para garantir o funcionamento de todos os controles implementados.

Para a primeira simulação, tem-se uma trajetória retilínea do líder, com $\dot{x} = 0.02m/s$ e $\dot{y} = 0.02m/s$ e a formação h_1 para 4 robôs ePuck, mostrado na Figura 11.

$$h_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.1 \\ 0 \\ 0 \\ 0 \\ 0.2 \\ 0 \\ 0 \\ 0 \\ 0.3 \\ 0 \end{bmatrix} = \begin{bmatrix} x\ líder \\ \dot{x}\ líder \\ y\ líder \\ \dot{y}\ líder \\ x\ seguidor\ 1 \\ \dot{x}\ seguidor\ 1 \\ y\ seguidor\ 1 \\ \dot{y}\ seguidor\ 1 \\ x\ seguidor\ 2 \\ \dot{x}\ seguidor\ 2 \\ y\ seguidor\ 2 \\ \dot{y}\ seguidor\ 2 \\ x\ seguidor\ 3 \\ \dot{x}\ seguidor\ 3 \\ y\ seguidor\ 3 \\ \dot{y}\ seguidor\ 3 \end{bmatrix}$$

Figura 11: Formação h_1

Logo, tem-se o gráfico mostrado na Figura 12. Como esperado, as velocidades lineares dos robôs não superam seu limite máximo, mostrando que o controlador pode ser usado em teste real. Isso é exibido na figura Figura 13.

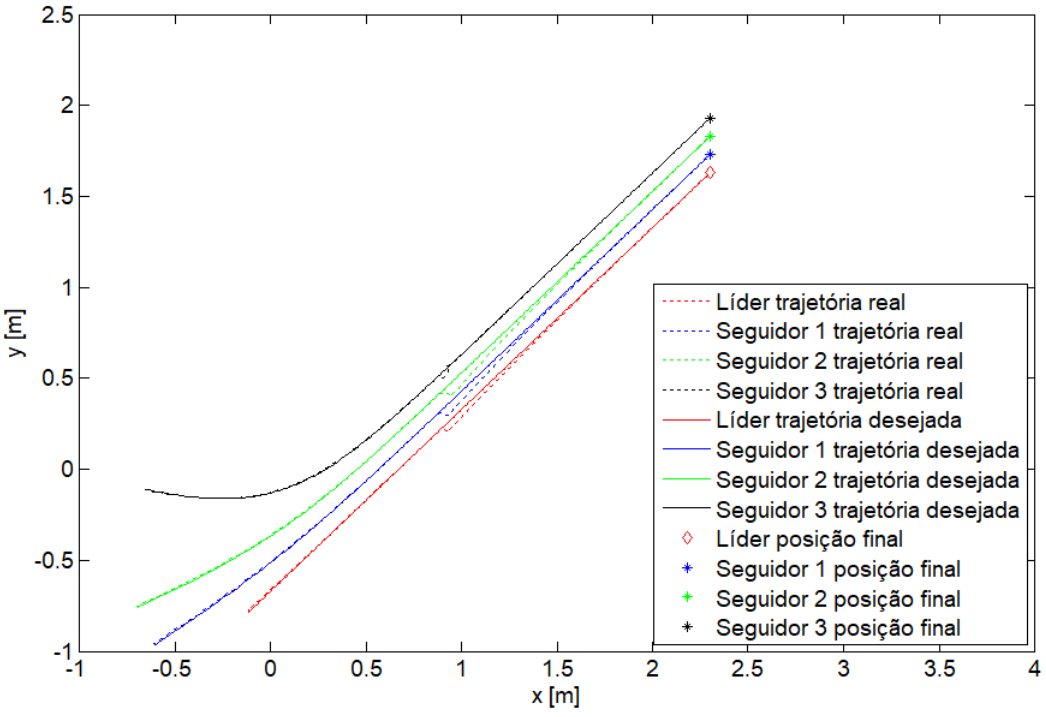

Figura 12: Gráfico da simulação para trajetória retilínea e formação h_1

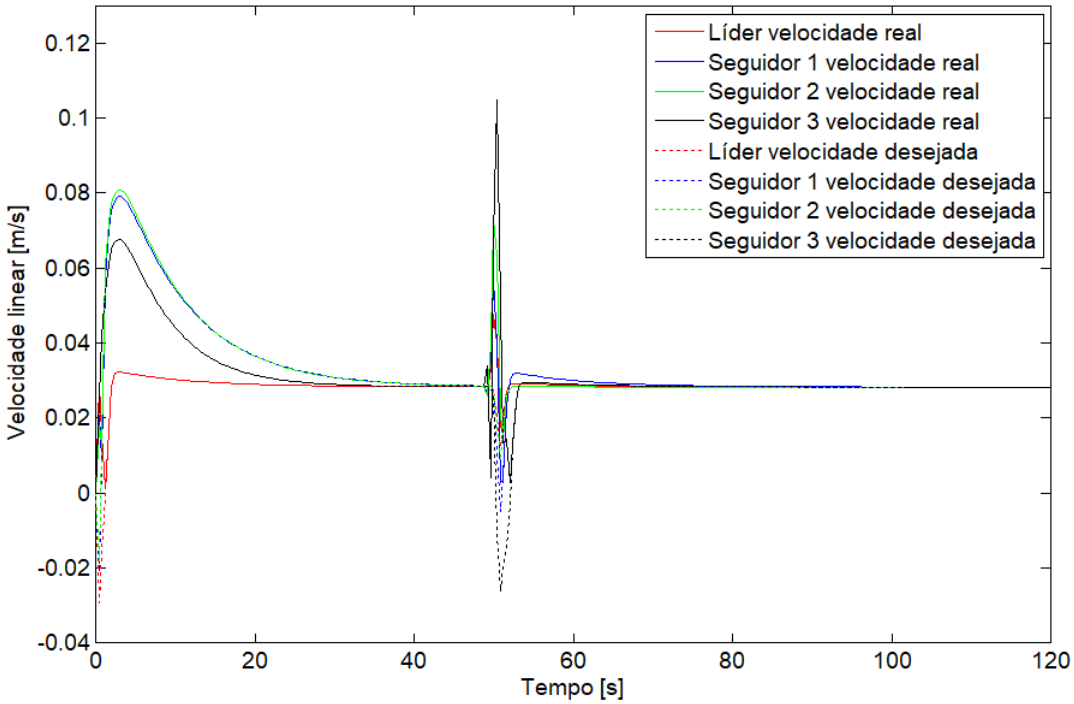

Figura 13: Gráfico da velocidade linear real para trajetória retilínea e formação h_1

Em outra simulação, tem-se uma trajetória circular do líder, com $\dot{x} = 0.02sin(2pi(tempo/120))$ e $0.02cos(2pi(tempo/120))$, onde tempo corresponde ao tempo atual da simulação e a formação h_2 para 4 robôs ePuck, mostrado na Figura 14.

$$h_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0.15 & 0 & -0.15 & 0 & 0.15 & 0 & -0.15 & 0 & 0 & 0 \end{bmatrix}^T$$

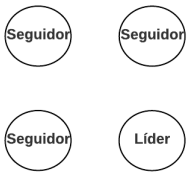

Figura 14: Formação h_2

Logo, tem-se o gráfico mostrado na Figura 15. Para essa simulação, as velocidades lineares dos robôs também não superam seu limite máximo, apresentado na figura Figura 16.

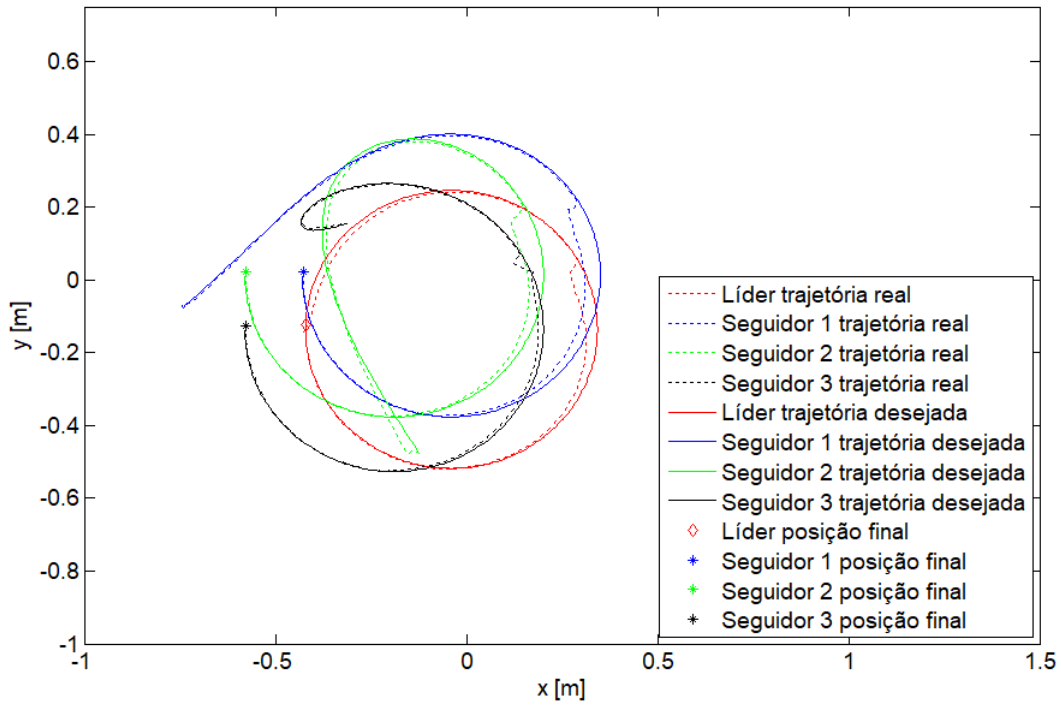

Figura 15: Gráfico da simulação para trajetória circular e formação h_2

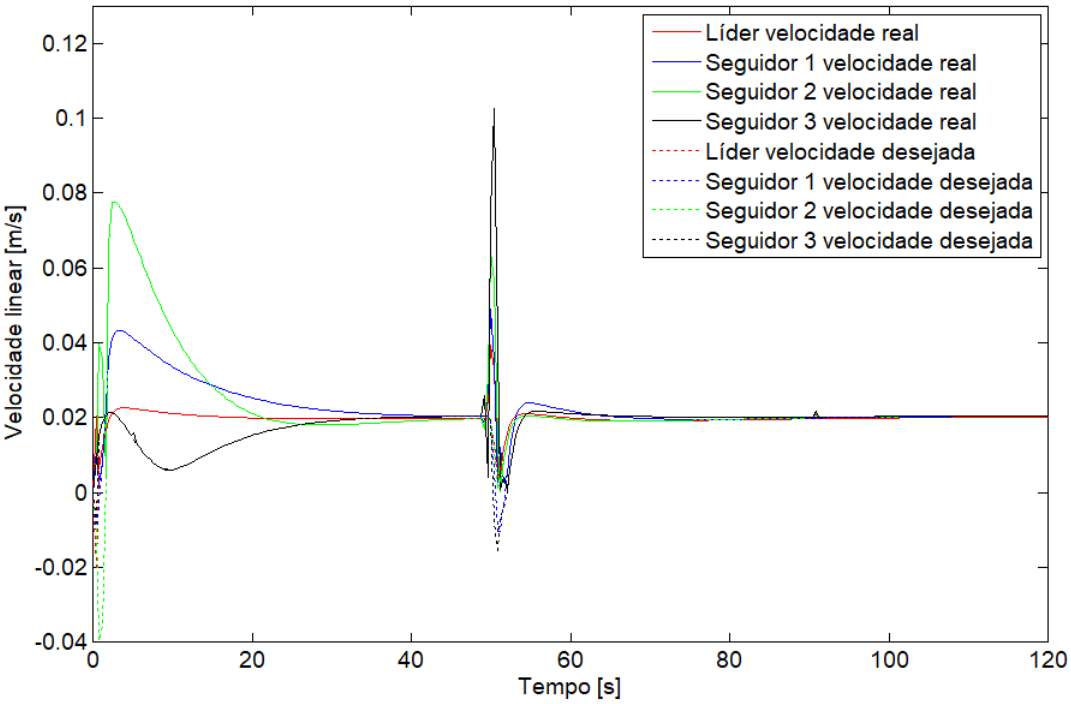

Figura 16: Gráfico da velocidade linear real para trajetória circular e formação h_2

Por último, tem-se um trajetória quadrada, com uma velocidade linear de $v = 0,3$ e a formação h_3 para os quatros robôs, mostrado na Figura 17.

$$h_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & -0.1 & 0 & -0.1 & 0 & -0.2 & 0 & -0.2 & 0 & -0.3 & 0 & -0.3 & 0 \end{bmatrix}^T$$

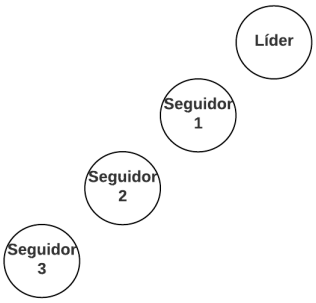

Figura 17: Formação h_3

Dessa maneira, obtém-se o gráfico mostrado na Figura 18. Nessa simulação, as velocidades lineares dos robôs, também, não superaram seu limite máximo, exibido na figura Figura 19.

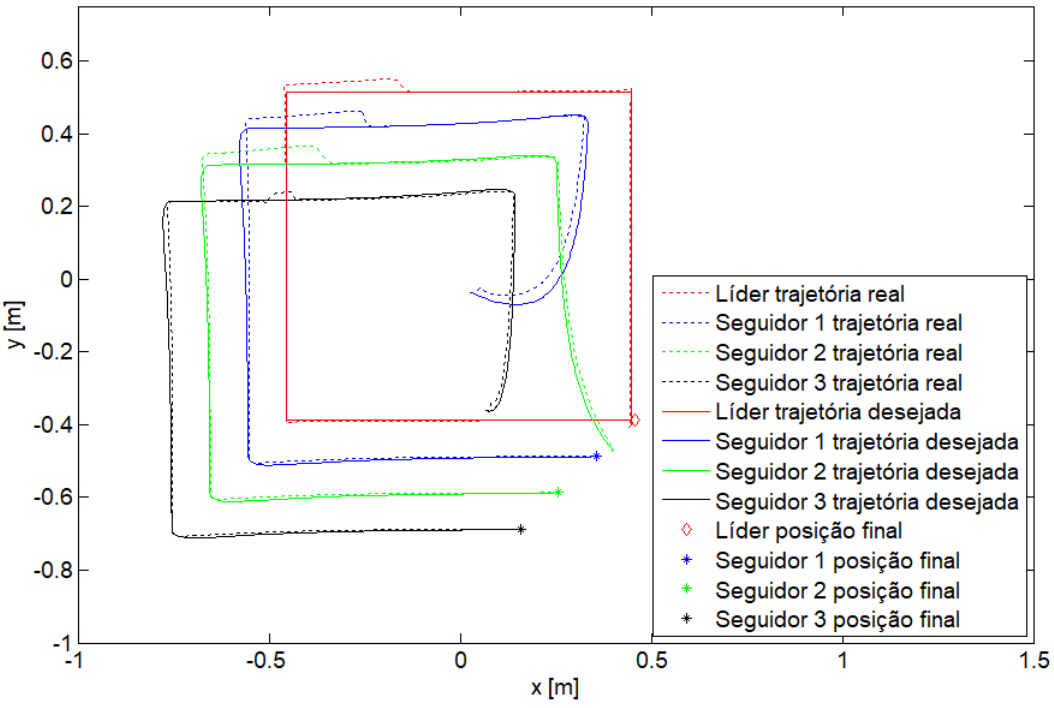

Figura 18: Gráfico da simulação para trajetória quadrada e formação h_3

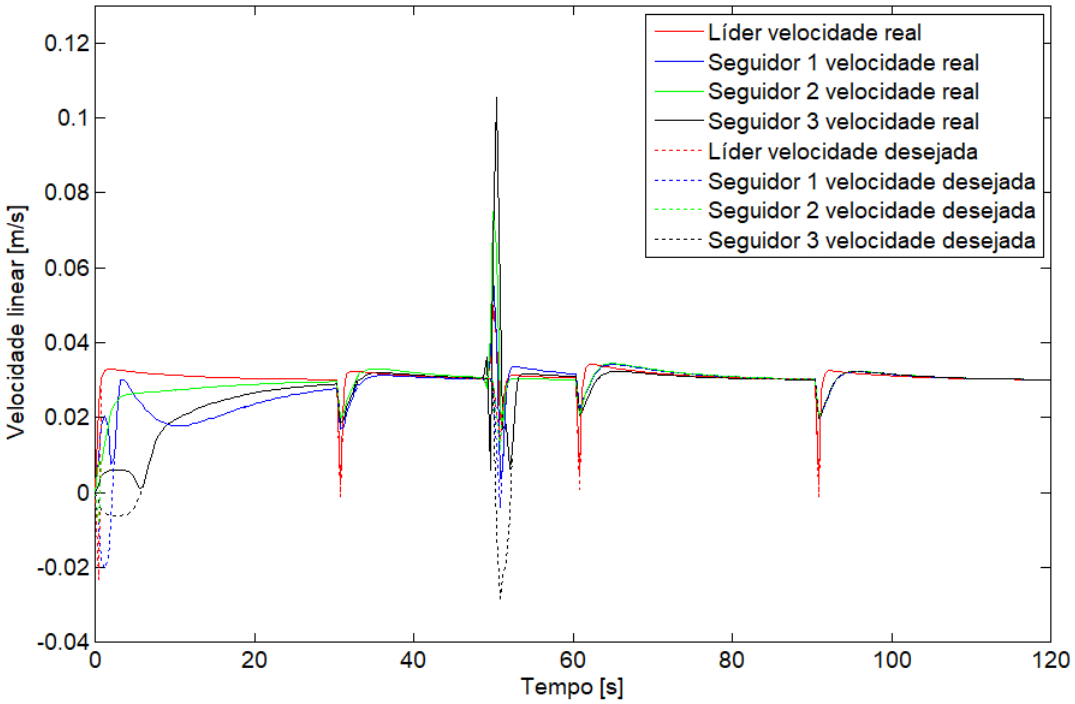

Figura 19: Gráfico da velocidade linear real para trajetória quadrada e formação h_3

3.1.2 Resultado Real

Para o teste real, utiliza-se 4 robôs ePucks e o sistema Vicon instalado em uma sala quadrada. A Vicon consegue abranger uma área de quase $2m \times 2m$. Logo, deve-se limitar as trajetórias do robô líder a esse espaçamento.

Por isso, utiliza-se uma trajetória circular, com a velocidade do líder, sendo $\dot{x} = 0.04sin(2pi(tempo/60))$ e $\dot{y} = 0.04cos(2pi(tempo/60))$ e formação h_2. Dessa forma, obteve-se o gráfico de formação exposto na Figura 20. A velocidade linear máxima do robô ePuck é respeitada, como apresentado na Figura 21

O resultado em vídeo pode ser conferido via internet pelos links descritos no anexo C.

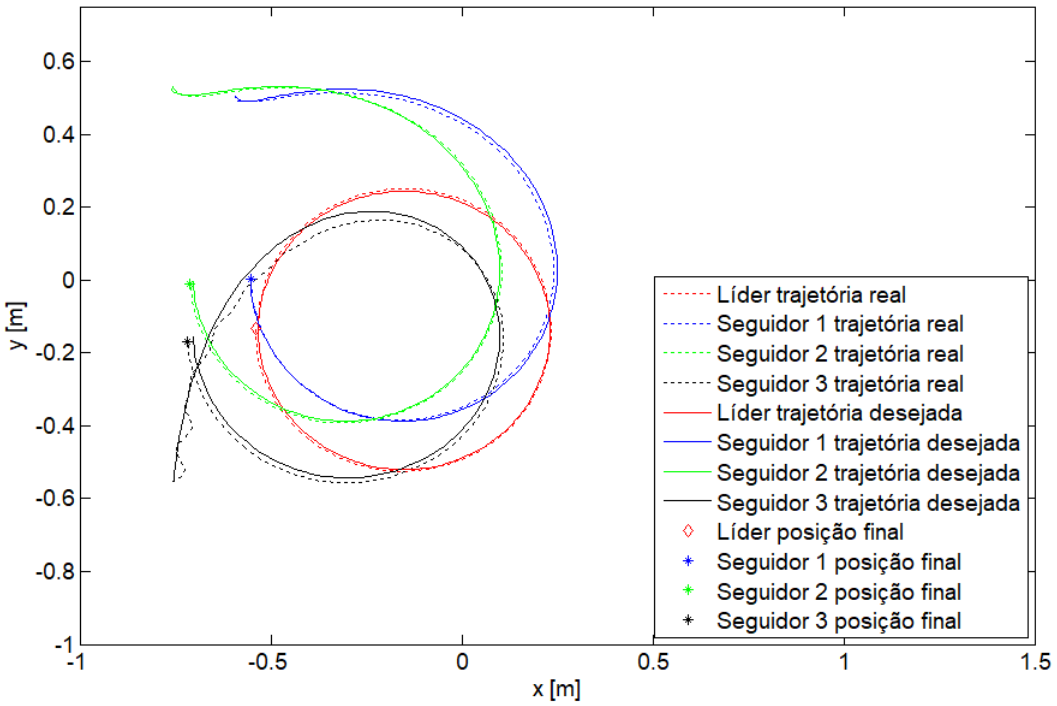

Figura 20: Gráfico do teste real para trajetória circular e formação h_2

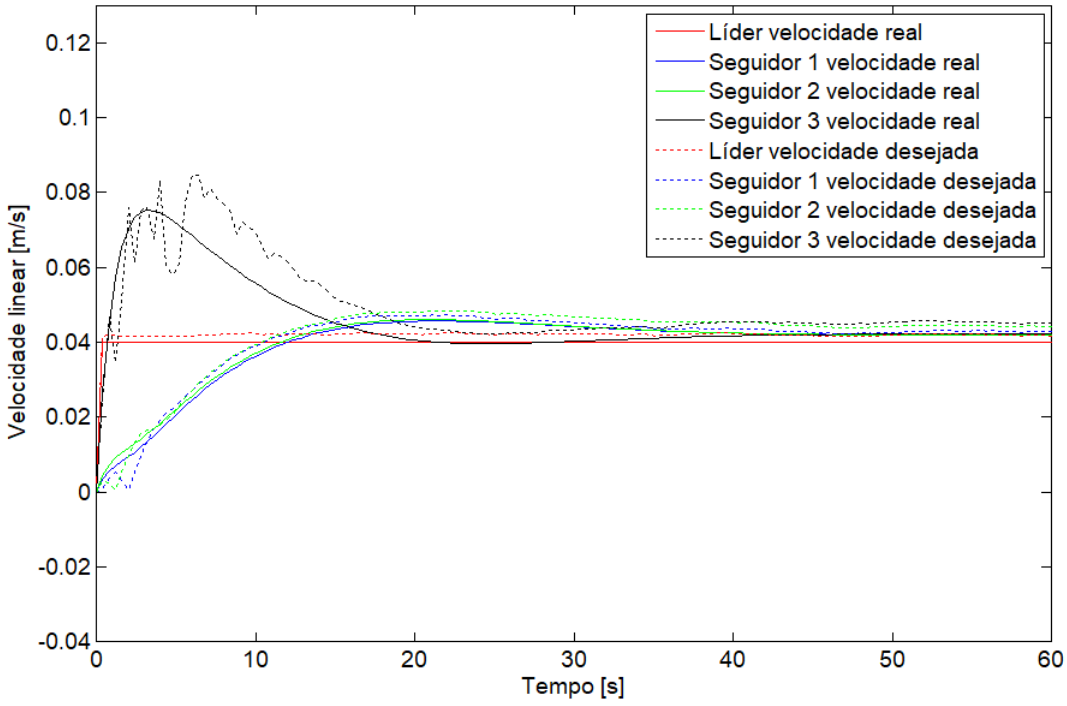

Figura 21: Gráfico da velocidade linear do teste, com trajetória circular e formação h_2

3.2 Discussões

Nessa seção, apresenta-se as discussões sobre os resultados obtidos, as dificuldades do trabalho e análises de diversas situações, usando o controle de formação.

3.2.1 Resultados Obtidos

Primeiramente, tem-se os resultados obtidos em simulação. Essa simulação abrange várias formações e trajetórias para testar os controles projetados. Observa-se nos resultados que o controle de formação consegue gerar para diferentes formações, trajetórias do líder e posições iniciais as trajetórias de todos os seguidores, mantendo a formação desejada. Como observado nas Figuras 12, 15 e 18, onde tem-se a trajetória desejada. Essa trajetória é gerada pelo controle de formação implementado, sendo a base para os controles baseado na cinemática e dinâmica do robô ePuck.

Após isso, observa-se o funcionamento dos controles baseados na cinemática e dinâmica em simulação. Dessa forma, monta-se um modelo do robô ePuck para que os controle desenvolvidos em simulação funcionem em testes reais. Além disso, acrescenta-se pertubações para ratificar o funcionamento dos mesmos controles. Logo, ao observar as mesmas figuras, percebe-se que a trajetória real, que representa o movimento do robô ePuck, aproxima-se bastante da trajetória desejada. Além disso, mesmo com a pertubação, o robô volta para a trajetória e termina a simulação na formação desejada, mostrando a tolerância a pertubação do controlador PD baseado na dinâmica desenvolvida. Também considera-se importante as velocidades imprimidas no modelo ePuck, pois, aceita-se qualquer velocidade por mais absurda que ela seja em simulação. Entretanto, como já discutido, o robô tem limitações físicas de velocidade que devem ser respeitadas, como são mostradas nas Figuras 13, 16 e 19.

Por último, tem o teste real com o uso dos robôs ePucks e o sistema Vicon para ratificar o modelo do robô desenvolvido e os controles já testados em simulação. Dessa forma, observa-se a Figura 20, que mostra a trajetória desejada e a percorrida pelo robô, obtendo uma trajetória real aproximada a desejada. Além disso, respeita-se as limitações do robô, como mostrado na Figura 21.

3.2.2 Dificuldades

Na montagem do software para esse projeto, usa-se o ®MATLAB. Entretanto, o mesmo oferece ferramentas diferentes para a montagem do algoritmo desejado. Uma delas consiste no Simulink, ferramenta usada primeiramente neste trabalho, por causa da proximidade do autor com a ferramenta e pela facilidade de se calcular derivadas, integrais e EDOs que são bastantes usadas nesse projeto. Com isso, montou-se inicialmente em Simulink, toda a estrutura do software discutida, obtendo resultados positivos para o controle de formação em simulação. Entretanto, a integração do Simulink com as

ferramentas para teste(robô ePuck e sistema Vicon) exigiam muito tempo, tornando a discretização grande o suficiente para impedir que os controladores funcionem corretamente.

Para solucionar o problema, usa-se o script de ®MATLAB para montar o programa, como discutido e apresentado. Esse método demonstra-se mais simples para integrar com as ferramentas, fazendo com que os esforços do projeto sejam gastos no controle estudado.

No código, teve-se que ter bastante cuidado com a discretização das derivadas e integrais. Para a função utilizada no cálculo de EDO, leva-se em conta a precisão do cálculo e a velocidade da função. Logo, houve um balanço dessas características para uma melhor performance.

Com isso, um problema das ferramentas de teste é a variação do tempo de comunicação com o programa desenvolvido. Isso faz com que o teste real tenha variações de tempo para cada ciclo de mais ou menos $0.04s$. Essa variação causa pequenos erros entre a posição da trajetória desejada e a real. Entretanto, percebe-se que esse tempo não trás problemas para os controladores que conseguem trabalhar com essa variação, como visto na Figura 20.

Outro problema ocorre com a comunicação do robô ePuck, que utiliza uma comunicação serial via bluetooth. Esse problema acontece pelo alcance limitado do bluetooth do computador que executa o código. Por causa disso, os comandos não alcançam o robô ou há uma alta latência na comunicação. Portanto, utiliza-se uma adaptador bluetooth, como mostrado na Figura 22, que aumenta o alcance do sinal e estabiliza a comunicação.

Figura 22: Adaptador bluetooth

Além do mais, no robô ePuck, deve-se observar as velocidades aplicadas. Isso, porque pode prejudicar o funcionamento do robô e gerar uma saturação na velocidade. Essa saturação atrapalha a trajetória gerada pelos controladores. Pode-se observar isso, em uma simulação, usando uma trajetória circular e a formação h_2, modifica-se os valores da matriz F_{veh} para $f_1 = -0.4$ e $f_2 = -1$. Com isso, obtêm-se velocidades desejadas superiores ao permitido pelo robô ePuck. Dessa forma, as trajetórias são desrespeitadas, como mostrado na Figura 23 e na Figura 24, observa-se a saturação das velocidades

lineares.

O sistema Vicon não apresenta problemas com a comunicação, porque a mesma é feita com um cabo ethernet, com uma boa taxa de velocidade. Porém, deve-se atentar aos limites da área coberta pelo sistema, uma vez que a câmera perde o rastreio do robô, quando ele ultrapassa essa região. Dessa forma, trabalha-se com trajetórias que respeitem essa área.

3.2.3 Análise de Situação

O controle de formação tem uma grande variedade de utilização, como mencionado na introdução desse trabalho. Um dos motivos para isso consiste na escalabilidade. Isso porque, mesmo aumentando a quantidade de RMRs envolvidos, tem-se um custo computacional parecido com o utilizado nos exemplos anteriores.

Para demonstrar essa situação, tem-se Figura 25 uma situação com 18 robôs e mudança de formação durante a trajetória, onde o líder é representado pela cor vermelha e todos os seguidores pela cor preta. Isso remete ao uso de quadricópteros nas olimpíadas de inverno 2018, que formaram desenhos no ar para entretenimento.

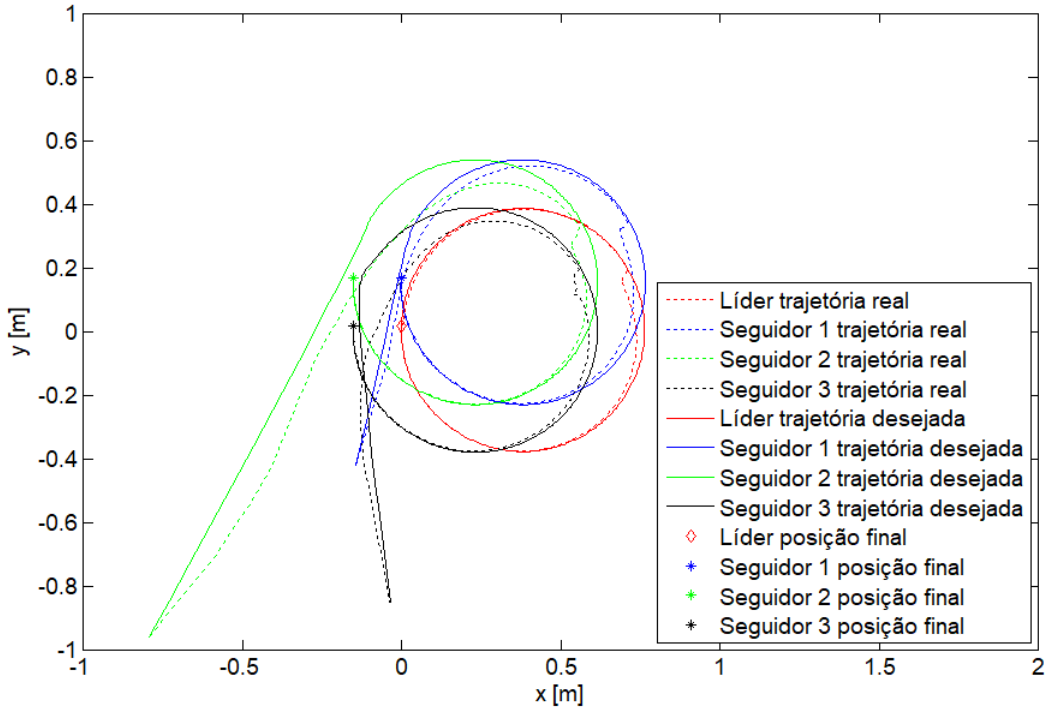

Figura 23: Gráfico de trajetória com saturação da velocidade

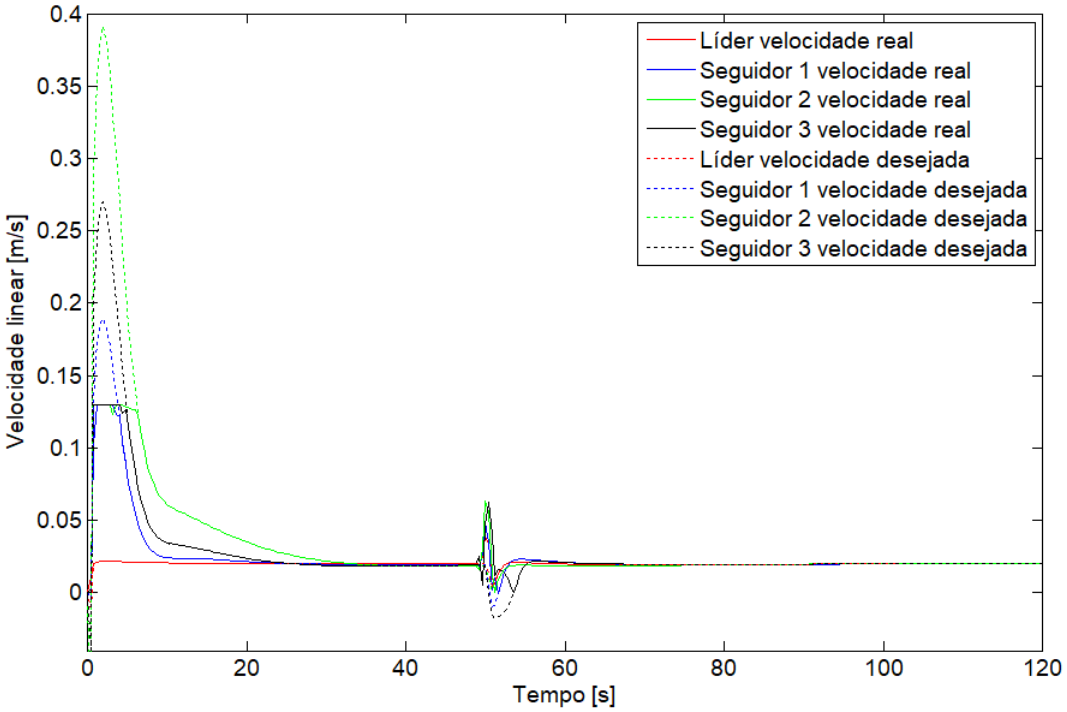

Figura 24: Gráfico da velocidade linear com saturação

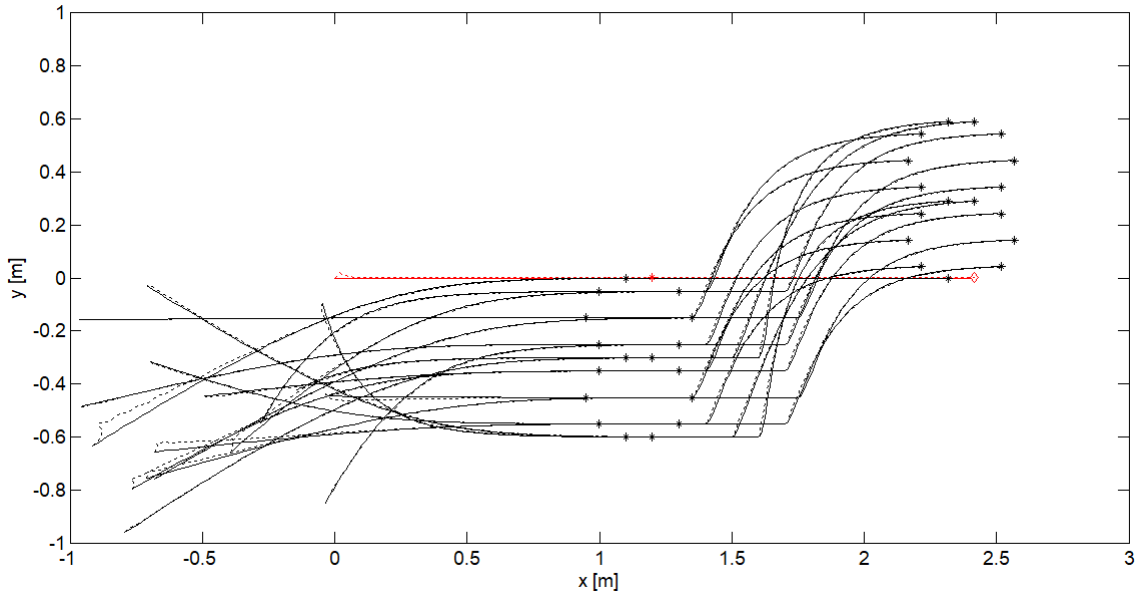

Figura 25: Gráfico de trajetória com 18 robôs ePucks

4 CONSIDERAÇÕES FINAIS

Este trabalho analisou o controle de formação na configuração Líder/Seguidor para robôs móveis, utilizando o robô ePuck e o sistema de câmeras Vicon, que substitui o quadricóptero. Para isso, foi construído um modelo do robô ePuck para simulação, permitindo construção de todos os controles envolvidos. Para assim, testar e demonstrar a utilização do controle de formação.

O trabalho tem como objetivo simular os controles e modelo desenvolvido, verificando se a formação e a trajetória estabelecidas foram respeitadas e se o sistema é tolerante a pertubações. Com isso aferido, há um teste com os robôs ePucks para corroborar o modelo e o uso na prática do controle de formação. Portanto, observa-se nos resultados que os objetivos do trabalho foram realmente alcançados.

Dessa forma, o trabalho contribui para demonstrar que este tipo de controle de formação é passível de utilização nas situações apresentadas na introdução deste trabalho. Para isso, deve-se aumentar e aprofundar as pesquisas nessa área.

No decorrer do trabalho, não demostra-se interesse no problema de colisões entre os participantes da formação, podendo ser mais uma área para estudo que contribuirá com o crescimento da pesquisa. Além disso, outros meios de comunicação devem ser estudados para permitir uma maior quantidade de robôs ePucks ou outros RMRs desejados na formação. Por fim, pode se adicionar o quadricóptero para observar a formação de um ponto elevado e substituir o sistema Vicon.

REFERÊNCIAS

AUGUGLIARO, F. et al. the flight assembled architecture installation: Cooperative construction with flying machines. **Control Systems, IEEE**, v. 34, n. 4, p. 46 – 64, 2014.

COELHO, P.; NUNES, U. Decentralized control of vehicle formations. **Lie algebra application to mobile robot control: a tutorial, Robotica**, n. 21, p. 483 – 493, 2003.

DORIGO, M. et al. Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. **Robotics Automation Magazine, IEEE**, n. 20(4), p. 60 – 71, 2013.

KANAYAMA, Y. et al. A stable tracking control method for an autonomous mobile robot. **Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on**, v. 1, p. 384 – 389, 1990.

LAFFERRIERE, G.; WILLIAMS, A.; CAUGHMAN, J. Graph theoretic methods in the stability of vehicle formations. **American Control Conference**, v. 4, p. 3729 – 3734, 2004.

_____. Decentralized control of vehicle formations. **Systems and Control Letters**, n. 54(9), p. 899 – 910, 2005.

LAWTON, J.; BEARD, R.; YOUNG, B. A decentralized approach to formation maneuvers. **Robotics and Automation, IEEE Transactions on**, n. 19(6), p. 933 – 941, 2003.

MONDADA, F. et al. The e-puck, a robot designed for education in engineering. **In Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions**, p. 59 – 65, 2009.

NAKAI, M. et al. Sobre o controle de robôs heterogêneos em formação utilizando o sistema de posicionamento vicon. **XII Simpósio Brasileiro de Autimação Inteligente(SBAI)**, 2015.

QUATER, P. et al. Light unmanned aerial vehicles (uavs) for cooperative inspection of pv plants. **Photovoltaics, IEEE Journal of**, v. 4, n. 4, p. 1107 – 1113, 2014.

VEERMAN, J. et al. Flocks and formations. **Journal of Statistical Physics**, n. 121(5-6), p. 901 – 936, 2005.

WILLIAMS, A.; LAFFERRIERE, G.; VEERMAN, J. Stable motions of vehicle formations. **Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on**, p. 72 – 77, 2005.

Apêndice A

Sistema Vicon. Código usado para comunicação com o sistema Vicon:

```
function viconconn()

addpath('C:\Program Files\Vicon\DataStream SDK\Win64\MATLAB')

% Program options
TransmitMulticast = false;

% Load the SDK
fprintf( 'Loading SDK...' );
Client.LoadViconDataStreamSDK();
fprintf( 'done\n' );

HostName = '10.235.0.161';

% Make a new client
global MyClient
MyClient = Client();

% Connect to a server
fprintf( 'Connecting to %s ...', HostName );
while ~MyClient.IsConnected().Connected
  % Direct connection
  MyClient.Connect( HostName );
  fprintf( '.' );
end
fprintf( '\n' );

% Enable some different data types
MyClient.EnableSegmentData();
MyClient.EnableMarkerData();
MyClient.EnableUnlabeledMarkerData();
MyClient.EnableDeviceData();

fprintf( 'Segment Data Enabled: %s\n',          AdaptBool(
   MyClient.IsSegmentDataEnabled().Enabled ) );
fprintf( 'Marker Data Enabled: %s\n',           AdaptBool(
   MyClient.IsMarkerDataEnabled().Enabled ) );
fprintf( 'Unlabeled Marker Data Enabled: %s\n', AdaptBool(
```

```
   MyClient.IsUnlabeledMarkerDataEnabled().Enabled ) );
fprintf( 'Device Data Enabled: %s\n',           AdaptBool(
   MyClient.IsDeviceDataEnabled().Enabled ) );

% Set the streaming mode
MyClient.SetStreamMode( StreamMode.ClientPull );

% Set the global up axis
MyClient.SetAxisMapping( Direction.Forward, ...
                         Direction.Left,    ...
                         Direction.Up );    % Z-up

Output_GetAxisMapping = MyClient.GetAxisMapping();
fprintf( 'Axis Mapping: X-%s Y-%s Z-%s\n',
   Output_GetAxisMapping.XAxis.ToString(), ...
                                           Output_GetAxisMapping
                                              .YAxis.ToString
                                              (), ...
                                           Output_GetAxisMapping
                                              .ZAxis.ToString
                                              () );

% Discover the version number
Output_GetVersion = MyClient.GetVersion();
fprintf( 'Version: %d.%d.%d\n', Output_GetVersion.Major, ...
                                Output_GetVersion.Minor, ...
                                Output_GetVersion.Point );

if TransmitMulticast
  MyClient.StartTransmittingMulticast( 'localhost', '
     224.0.0.0' );
end
```

Código usado para obtenção das posições dos robôs:

```
function [pos] = viconpos()

  addpath('C:\Program Files\Vicon\DataStream SDK\Win64\MATLAB
     ')
```

```
  persistent vetrot;
  persistent i;
  global MyClient;

  if isempty(i)
      i=1;
  end

  drawnow;
  % Get a frame
%   fprintf( 'Waiting for new frame...' );
  while MyClient.GetFrame().Result.Value ~= Result.Success
    fprintf( '.' );
  end% while
  fprintf( '\n' );

  % Get the frame number
  Output_GetFrameNumber = MyClient.GetFrameNumber();
%   fprintf( 'Frame Number: %d\n', Output_GetFrameNumber.
   FrameNumber );

  % Get the frame rate
  Output_GetFrameRate = MyClient.GetFrameRate();
%   fprintf( 'Frame rate: %g\n', Output_GetFrameRate.
   FrameRateHz );


  % Count the number of subjects
  SubjectCount = MyClient.GetSubjectCount().SubjectCount;
%   fprintf( 'Subjects (%d):\n', SubjectCount );

  for SubjectIndex = 1:SubjectCount
%     fprintf( '  Subject #%d\n', SubjectIndex - 1 );

    % Get the subject name
    SubjectName = MyClient.GetSubjectName( SubjectIndex ).
       SubjectName;
%     fprintf( '    Name: %s\n', SubjectName );
```

```
    % Get the root segment
%     RootSegment = MyClient.GetSubjectRootSegmentName(
   SubjectName ).SegmentName;
%     fprintf( '    Root Segment: %s\n', RootSegment );

    % Count the number of segments
    SegmentCount = MyClient.GetSegmentCount( SubjectName ).
       SegmentCount;
%     fprintf( '    Segments (%d):\n', SegmentCount );
    for SegmentIndex = 1:SegmentCount
%       fprintf( '      Segment #%d\n', SegmentIndex - 1 );

      % Get the segment name
      SegmentName = MyClient.GetSegmentName( SubjectName,
         SegmentIndex ).SegmentName;
%       fprintf( '        Name: %s\n', SegmentName );

      % Get the segment parent
%       SegmentParentName = MyClient.GetSegmentParentName(
   SubjectName, SegmentName ).SegmentName;
%       fprintf( '        Parent: %s\n',  SegmentParentName )
   ;

      % Get the segment's children
      ChildCount = MyClient.GetSegmentChildCount( SubjectName
         , SegmentName ).SegmentCount;
%       fprintf( '    Children (%d):\n', ChildCount );
%       for ChildIndex = 1:ChildCount
%         ChildName = MyClient.GetSegmentChildName(
   SubjectName, SegmentName, ChildIndex ).SegmentName;
%         fprintf( '      %s\n', ChildName );
%       end% for


      % Get the global segment translation
      Output_GetSegmentGlobalTranslation = MyClient.
         GetSegmentGlobalTranslation( SubjectName,
         SegmentName );
%       fprintf( '        Global Translation: (%g, %g, %g) %s
```

```
   \n',                ...
%                          Output_GetSegmentGlobalTranslation
   .Translation( 1 ), ...
%                          Output_GetSegmentGlobalTranslation
   .Translation( 2 ), ...
%                          Output_GetSegmentGlobalTranslation
   .Translation( 3 ), ...
%                          AdaptBool(
   Output_GetSegmentGlobalTranslation.Occluded ) );


      % Get the global segment rotation in EulerXYZ co-
         ordinates
      Output_GetSegmentGlobalRotationEulerXYZ = MyClient.
         GetSegmentGlobalRotationEulerXYZ( SubjectName,
         SegmentName );
%       fprintf( '         Global Rotation EulerXYZ: (%g, %g,
   %g) %s\n',                 ...
%                          radtodeg(
   Output_GetSegmentGlobalRotationEulerXYZ.Rotation( 1 )),
         ...
%                          radtodeg(
   Output_GetSegmentGlobalRotationEulerXYZ.Rotation( 2 )),
         ...
%                          radtodeg(
   Output_GetSegmentGlobalRotationEulerXYZ.Rotation( 3 )),
         ...
%                          AdaptBool(
   Output_GetSegmentGlobalRotationEulerXYZ.Occluded ) );
%         if i>5
%             i=1;
%         end
%         vetrot(i,1) = Output_GetSegmentGlobalTranslation.
   Translation( 1 );
%         vetrot(i,2) = Output_GetSegmentGlobalTranslation.
   Translation( 2 );
%         vetrot(i,3) =
   Output_GetSegmentGlobalRotationEulerXYZ.Rotation( 3 );
%         i=i+1;
```

```
%
%           pos(1)=mean(vetrot(:,1));
%           pos(2)=mean(vetrot(:,2));
%           pos(3)=mean(vetrot(:,3));

          if strcmp('e1964',SubjectName)
              nome=1;
          end
          if strcmp('e1974',SubjectName)
              nome=2;
          end
          if strcmp('e1943',SubjectName)
              nome=3;
          end
          if strcmp('e1967',SubjectName)
              nome=4;
          end

          try %#ok<TRYNC>
              pos(nome,1)= Output_GetSegmentGlobalTranslation.
                 Translation( 1 )/1000;                          %#
                 ok<AGROW>
              pos(nome,2)= Output_GetSegmentGlobalTranslation.
                 Translation( 2 )/1000;
              pos(nome,3)=
                 Output_GetSegmentGlobalRotationEulerXYZ.
                 Rotation( 3 );
          end
%           pos(SubjectIndex,1)=
    Output_GetSegmentGlobalTranslation.Translation( 1 );
%           pos(SubjectIndex,2)=
    Output_GetSegmentGlobalTranslation.Translation( 2 );
%           pos(SubjectIndex,3)=
    Output_GetSegmentGlobalRotationEulerXYZ.Rotation( 3 );

      end% SegmentIndex

    end% SubjectIndex
```

```
if SubjectCount == 0
    pos(1:3)=0;
    disp('fora do alcance da camera')
    return;
end
```

Apêndice B

Robô ePuck.

Código usado para comunicação com o robô ePuck:

```
function epuckconn()

clear all


instrfind

try
    fclose(ans)
end

global ePic1;
ePic1 = ePicKernel();
[ePic1 result] = connect(ePic1, 'COM13'); %1964

global ePic2;
ePic2 = ePicKernel();
[ePic2 result] = connect(ePic2, 'COM17'); %1974

global ePic3;
ePic3 = ePicKernel();
[ePic3 result] = connect(ePic3, 'COM14'); %1943

global ePic4;
ePic4 = ePicKernel();
[ePic4 result] = connect(ePic4, 'COM18'); %1967


% %   Inicio - Estabelece comunicacao com o epuck
% ePic=activate(ePic, 'pos');
% ePic = set(ePic,'ledON', 7);
% ePic = update(ePic);
% ePic = reset(ePic,'odom');
% %   Fim - Estabelece comunicacao com epuck


%
```

```
% ePic2 = set(ePic2,'speed',[100,0]);
% ePic=activate(ePic,'odom');
% % We want to activate encoders to use odometry
% ePic=activate(ePic,'pos');
%
% ePic2 = update(ePic2);
% [val, up] = get(ePic, 'pos')

%          ePic = disconnect(ePic);
```

Código usado para parar com o robô ePuck:

```
function epuckstop()

global ePic1
global ePic2
global ePic3
global ePic4

disp('EPUCK1')
ePic1 = set(ePic1,'speed',[0,0]);
ePic1 = update(ePic1);
disp('EPUCK2')
ePic2 = set(ePic2,'speed',[0,0]);
ePic2 = update(ePic2);
disp('EPUCK3')
ePic3 = set(ePic3,'speed',[0,0]);
ePic3 = update(ePic3);
disp('EPUCK4')
ePic4 = set(ePic4,'speed',[0,0]);
ePic4 = update(ePic4);
% %

% disconnect(ePic1)
% disconnect(ePic2)
% disconnect(ePic3)
% disconnect(ePic4)

% clear all
% while(1)
```

```
% ePic2 = set(ePic2,'speed',[1000,1000]);
% ePic2 = update(ePic2);
% end
%
```

ANEXO A

Código para simulação:

```
clc
clear all
close all

% Parametros para controle de formacao
n = 4;

Aveh = [0 1 0 0
        0 0 0 0
        0 0 0 1
        0 0 0 0];

Bveh = [0 0
        1 0
        0 0
        0 1];
Ia = eye(n);

A = kron(Ia,Aveh);

Q = zeros(n,n);
D = zeros(n,n);
for i=1:n
    if(i~=1)
        Q(i,1)=1;
        D(i,i)=1;
    end;
end;


Lt = pinv(D)*(D-Q);

%Formacao desejada
h = [ 0; 0; 0; 0
      0; 0; 0.1; 0
      0; 0; 0.2; 0
      0; 0; 0.3; 0];
```

```
Fveh = [-0.1 -1 0 0
        0 0 -0.1 -1];

X = kron(Lt,Bveh*Fveh);

% Geracao das posicoes iniciais de cada RMR
px = zeros(n,1);
py = zeros(n,1);
palfa = zeros(n,1);
for(i=1:n)
    px(i,1) = (-rand) ;
    py(i,1) = (-rand);
    palfa(i,1) = pi*rand-pi;
end;
px(1,1) = 0;
py(1,1) = 0;

x0 = zeros(4*n,1);
vx = zeros(n,1);
vy = zeros(n,1);
w = zeros(n,1);

for i=1:n;
    x0(1+4*(i-1),1) = px(i,1);
    x0(2+4*(i-1),1) = vx(i,1);
    x0(3+4*(i-1),1) = py(i,1);
    x0(4+4*(i-1),1) = vy(i,1);
end;

%Constantes
mc=0.144;           %Kg
mw=0.016;           %Kg
b=53/2*1e-3 ;       %m
d=.5*1e-3 ;         %m
r=41/2*1e-3  ;      %m
Ic=68.156*1e-6;     %Kg*m^2
Iw=1.314*1e-6 ;     %Kg*m^2
Im=0.676*1e-6 ;     %Kg*m^2
c = r/(2*b);
```

```
m = mc + 2*mw;
I = Ic + 2*mw*(d^2+b^2)+2*Im+mc*d^2;
deltat = 0.4;


% Ganhos do controlador cinetico
Kx = 0.1;
Ky = 110;
Ka = 40;




% Ganhos controlador dinamico
Kp = 0.00001;
Kd = 0.00001;


% ODE
options = odeset('RelTol',1e-3,'AbsTol',1e-6);
tic
[t,x] = ode113(@formation,0:0.2:0.4,x0,options,A,X,h);
toc

% Inicializacao de variaveis
xci = px;
yci = py;
aci = palfa;
ari = zeros(n,2);
vid = zeros(n,1);
wid = zeros(n,1);
q2ci = zeros(2*n,1);
q2di = zeros(2*n,1);
dq02 = zeros(2*n,1);
velo = zeros(2,n);
torque = zeros(2*n,1);
tempo =0;
tempo_f = 120;
```

```
% Inicializacao dos vetores de dados
data = 1;
xci_data = zeros(n,length(0:deltat:tempo_f));
yci_data = zeros(n,length(0:deltat:tempo_f));
x0_data = zeros(4*n,length(0:deltat:tempo_f));
torque_data = zeros(2*n,length(0:deltat:tempo_f));
aci_data = zeros(n,length(0:deltat:tempo_f));
wx_data = zeros(n,length(0:deltat:tempo_f));
vci_data = zeros(n,length(0:deltat:tempo_f));
vid_data = zeros(n,length(0:deltat:tempo_f));
wid_data = zeros(n,length(0:deltat:tempo_f));
q2ci_data = zeros(2*n,length(0:deltat:tempo_f));
q2di_data = zeros(2*n,length(0:deltat:tempo_f));
ari_data = zeros(n,length(0:deltat:tempo_f));

xci_data(:,data) = xci;
yci_data(:,data) = yci;
torque_data(:,data) = torque;
aci_data(:,data) = ari(:,2);
vid_data(:,data) = vid;
wid_data(:,data) = wid;
wx_data(:,data) = velo(2,:)';
vci_data(:,data) = velo(1,:)';
x0_data(:,data) = x0;
q2ci_data(:,data) = q2ci;
q2di_data(:,data) = q2di;
ari_data(:,data) = ari(:,2);



%% Loop de controle com discretizacaoo de 0.4 segundos
for tempo=0:deltat:tempo_f
    % Velocidade desejada para trajetoria do lider
%
%     x0(2,1) = 0.02*sin(2*pi*tempo/tempo_f);
%     x0(4,1) = 0.02*cos(2*pi*tempo/tempo_f);
%
%     if(tempo<=30)
%         x0(2,1) = 0;
```

```
%         x0(4,1) = 0.03;
%     end;
%     if(tempo>30 && tempo<=60)
%         x0(2,1) = -0.03;
%         x0(4,1) = 0;
%     end;
%     if(tempo>60 && tempo<=90)
%         x0(2,1) = 0;
%         x0(4,1) = -0.03;
%     end;
%     if(tempo>90 && tempo<=120)
%         x0(2,1) = 0.03;
%         x0(4,1) = 0;
%     end;

%     x0(2,1) = 0.02;
%     x0(4,1) = 0.02;


    % Resolucao da EDO para controle da Formacao
    tic
    [t1,x] = ode113(@formation,0:0.2:0.4,x0,options,A,X,h);

    % Calculo da velocidade e angulo desejados
    [velo,ari] = speed([x0_data(:,data)'; x(3,:)],deltat,n);
    x0 = x(3,:)';

    %  Controle Cinematico
    [q2di,vid,wid] = control_cin(xci,yci,aci,x(3,:),velo,Kx,
       Ky,Ka,b,r,ari(:,2));

    % Controlador Dinamico
    [torque, A2, B2 ] = din([q2di_data(:,data) q2di],[dq02 (
       q2ci - q2di)],velo(2,:)',aci,deltat,...
        Kp,Kd,I,Iw,b,c,d,m,n);

    % Diferenca antiga entre as velocidades angulare das
       rodas
    % desejadas e real
```

```
    dq02 = (q2ci - q2di);


%     Simulacao de pertubacao
    wd = 10^-4*exp(-(tempo-50)^4)*sin(pi*tempo);
    we = -10^-3*exp(-(tempo-50)^4)*sin(pi*tempo);
    torque = torque + [0.5*wd;0.2*we;0.8*wd;0.3*we;-0.4*wd
       ;0.3*we;0.1*wd;-0.7*we];


    % Modelo dinamico
    [t2, q2ci1] = ode113(@model_din,0:0.2:0.4,q2ci,options,A2
       ,B2,torque);
    q2ci = q2ci1(3,:)';


    % Modelo cinematico dos RMR para determinar sua posicao
       em simulacao
    [xci,yci,aci,vci] = model_cin(q2ci,xci,yci,aci,c,b,d,
       deltat,n);

    data = data + 1;
    xci_data(:,data) = xci;
    yci_data(:,data) = yci;
    ari_data(:,data) = ari(:,2);
    torque_data(:,data) = torque;
    aci_data(:,data) = aci;
    vid_data(:,data) = vid;
    wid_data(:,data) = wid;
    wx_data(:,data) = velo(2,:)';
    vci_data(:,data) = vci;
    x0_data(:,data) = x0;
    q2ci_data(:,data) = q2ci;
    q2di_data(:,data) = q2di;

    toc

end;
```

```
% plot do grafico

function [ dx ] = formation(t0,x,A,X,h)
dx = A*x + X*(x - h);
end

function [ vel,ari ] = speed( x,deltat,n )
% Seleciona as velocidades desejada de cada RMR
vxri = zeros(2,n);
vyri = zeros(2,n);
wri = zeros(1,n);
for i = 1:n
    vxri(:,i) = x(:,4*i-2);
    vyri(:,i) = x(:,4*i);
end;

% Calcula a velocidade linear desejada
vri = sqrt(vxri.^2 + vyri.^2);


% Caculo do angulo desejado de cada RMR
ari1 = atan2(vyri,vxri);

ari= ari1';
 for i = 1:n
    if ari1(2,i) - ari1(1,i) > pi
        wri(1,i) = (ari1(2,i) - ari1(1,i) - 2*pi)/deltat;
    elseif ari1(2,i) - ari1(1,i) < -pi
        wri(1,i) =(ari1(2,i) - ari1(1,i) + 2*pi)/deltat;
    else
        wri(1,i) = (ari1(2,i) - ari1(1,i))/deltat;
    end;
end;
%Calculo da velocidade angular desejado de cada RMR


vel = [vri(2,:); wri];

end
```

```
function [ q2id ,vid,wid] = control_cin( xci,yci,aci,x,vel,Kx
   , Ky, Ka, b, r,ari )
% Seleciona as posicoes desejadas e velocidades
n = length(xci);
xri = zeros(n,1);
yri = zeros(n,1);
for i = 1:n
    xri(i,1) = x(4*i-3);
    yri(i,1) = x(4*i-1);
end;

vri = vel(1,:)';
wri = vel(2,:)';

% Calculo do erro
xei = cos(aci).*(xri-xci)+sin(aci).*(yri-yci);
yei = -sin(aci).*(xri-xci)+cos(aci).*(yri-yci);
aei = ari - aci;

% Controlador cinematico
vid = vri.*cos(aei) + Kx*xei;
wid = wri + vri.*(Ky*yei + Ka*sin(aei));
%  for i = 1:4
%     if wid > 2*pi
%         wid(i,1) = -2*pi + wid(i,1);
%     elseif wid < -2*pi
%         wid(i,1) = 2*pi + wid(i,1);
%     else
%         wid(i,1) = wid(i,1);
%     end;
% end;
% wid= min(7*pi/4, max(-7*pi/4, wid));
Ia = eye(n);
vel = zeros(2*n,1);
for i = 1:n
    vel(2*i-1,1) = vid(i,1);
    vel(2*i,1) = wid(i,1);
end;
```

```
% Calculo das velocidades angulares de cada roda
q2id = kron(Ia,[1/r b/r ; 1/r -b/r])*vel;


end

function [ torque,A,B ] = din(q2di, dq2, wci, aci, deltat, Kp
   , Kd,I,Iw,b,c,d,m,n)
% Calculo das matrizes Aveh, Bveh e M2
[A,B,M2]=fcn( wci,aci,I,Iw,b,c,d,m,n );

% Calculo do u
tempo = [0 deltat];
dq = trapz(tempo,dq2,2);
q3di = diff(q2di')'/deltat;
ui = -Kp*dq - Kd*dq2(:,2);

% Caculo do torque
torque = M2*(q3di - A*q2di(:,2) + ui);
end

function [ A,B,M2f] = fcn( walfa,alfa,I,Iw,b,c,d,m,n )
 M = zeros(5*n,5*n);
 S = zeros(5*n,2*n);
 C = zeros(5*n,5*n);
 for i =1:n
    S(5*i-4:5*i,2*i-1:2*i)= ...
     [c*(b*cos(alfa(i))-d*sin(alfa(i))) c*(b*cos(alfa(i))+d*
        sin(alfa(i)))
      c*(b*sin(alfa(i))+d*cos(alfa(i))) c*(b*sin(alfa(i))-d*
         cos(alfa(i)))
                  c                                -c
                  1                                0
                  0                                1];
    C(5*i-4:5*i,5*i-4:5*i) = [ 0 0 m*d*walfa(i)*cos(alfa(i))
       0 0
                                0 0 m*d*walfa(i)*sin(alfa(i))
                                   0 0
                                0 0 0 0 0
                                0 0 0 0 0
```

```
                                0 0 0 0 0];
    M(5*i-4:5*i,5*i-4:5*i) =[m   0   m*d*sin(alfa(i)) 0 0
                             0   m  -m*d*cos(alfa(i)) 0 0
                             m*d*sin(alfa(i)) -m*d*cos(alfa(i
                                )) I 0 0
                             0 0 0 Iw 0
                             0 0 0 0 Iw];
 end;

% Calculo de C e M
C1 = S'*C*S;
M2f = S'*M*S;

% Calculo de A e B
A = -1*(C1 + M2f);
B = inv(M2f);

end

function [ ddq20 ] = model_din( t0, q2ci, A, B, torque )
ddq20 = A*q2ci + B*torque;
end

function [ xci,yci,aci,v,vxci,vyci ] = model_cin( q2id, x, y,
    a, c, b, d, t,n)
 right = zeros(n,1);
 left = zeros(n,1);
 q2id = min(6.34,max(-6.34,q2id));
for i = 1:n
    right(i,1) = q2id(2*i-1);
    left(i,1) = q2id(2*i);
end;

 wci = c*(right - left);
%  for i = 1:4
%     if wci > 2*pi
%         wci(i,1) = -2*pi + wci(i,1);
%     elseif wci < -2*pi
%         wci(i,1) = 2*pi + wci(i,1);
%     else
```

```
%          wci(i,1) = wci(i,1);
%      end;
% end;

 aci1 = wci*t + a;
 aci = aci1;



 vxci = c*((b*cos(aci) - d*sin(aci)).*right + (b*cos(aci) +
    ...
     d*sin(aci)).*left);

 vyci = c*((b*sin(aci) + d*cos(aci)).*right + (b*sin(aci) -
    ...
     d*cos(aci)).*left);

 v = sqrt(vxci.^2 + vyci.^2);
 xci = vxci*t + x;
 yci = vyci*t + y;
end
```

ANEXO B

Código para teste real:

```
close all

%% Parametros para comunicacao
global ePic1; %red
global ePic2; %blue
global ePic3; %green
global ePic4; %black

%% Parametros para controle de formacao
Aveh = [0 1 0 0
        0 0 0 0
        0 0 0 1
        0 0 0 0];

Bveh = [0 0
        1 0
        0 0
        0 1];
Ia = eye(4);

A = kron(Ia,Aveh);

Q = [0 0 0 0
     1 0 0 0
     1 0 0 0
     1 0 0 0];

D = [0 0 0 0
     0 1 0 0
     0 0 1 0
     0 0 0 1];

Lt = pinv(D)*(D-Q);

%Formacao desejada
h = [ 0  ; 0  ;0   ; 0
      0; 0  ;0.15; 0
     -0.15; 0  ;0.15; 0
```

```
      -0.15; 0   ;0; 0];

Fveh = [-0.1 -1 0 0
         0 0 -0.1 -1];

X = kron(Lt,Bveh*Fveh);

% Geracao das posicoes iniciais de cada RMR
n = 4;

pos = viconpos();
px(:,1) = pos(1:n,1);
py(:,1) = pos(1:n,2);
palfa(:,1) = pos(1:n,3);
vx = [0;0;0;0];
vy = [0;0;0;0];
w = [0;0;0;0];
x0 = zeros(4*n,1);
for i=1:n;
    x0(1+4*(i-1),1) = px(i,1);
    x0(2+4*(i-1),1) = vx(i,1);
    x0(3+4*(i-1),1) = py(i,1);
    x0(4+4*(i-1),1) = vy(i,1);
end;

%Constantes
mc=0.144;             %Kg
mw=0.016;             %Kg
b=53/2*1e-3 ;         %m
d=.5*1e-3 ;           %m
r=41/2*1e-3  ;        %m
Ic=68.156*1e-6;       %Kg*m^2
Iw=1.314*1e-6 ;       %Kg*m^2
Im=0.676*1e-6 ;       %Kg*m^2
c = r/(2*b);
m = mc + 2*mw;
I = Ic + 2*mw*(d^2+b^2)+2*Im+mc*d^2;
deltat = 0.4;
```

```
% Ganhos do controlador cinetico
Kx = 0.1;
Ky = 110;
Ka = 40;

% Ganhos controlador dinamico
Kp = 0.00001;
Kd = 0.00001;


%% ODE
options = odeset('RelTol',1e-3,'AbsTol',1e-6);
tic
[t,x] = ode113(@formation,0:0.2:0.4,x0,options,A,X,h);
toc

xci = px;
yci = py;
aci = palfa;
ari = zeros(n,2);
vid = zeros(n,1);
wid = zeros(n,1);
q2ci = zeros(2*n,1);
q2di = zeros(2*n,1);
dq02 = zeros(2*n,1);
velo = zeros(2,n);
torque = zeros(2*n,1);
tempo =0;
tempo_f = 60;
q2_adpt = 1000/0.13*(r);

%% Plot das posicoes iniciais de cada RMR
% Plot das posicoes iniciais de cada RMR (vermelho lider,
   azul 2,
% verde 3 e preto 4)
data = 1;
xci_data = zeros(n,length(0:deltat:tempo_f));
yci_data = zeros(n,length(0:deltat:tempo_f));
```

```
x0_data = zeros(4*n,length(0:deltat:tempo_f));
torque_data = zeros(2*n,length(0:deltat:tempo_f));
aci_data = zeros(n,length(0:deltat:tempo_f));
wci_data = zeros(n,length(0:deltat:tempo_f));
vci_data = zeros(n,length(0:deltat:tempo_f));
vid_data = zeros(n,length(0:deltat:tempo_f));
wid_data = zeros(n,length(0:deltat:tempo_f));
q2ci_data = zeros(2*n,length(0:deltat:tempo_f));
q2di_data = zeros(2*n,length(0:deltat:tempo_f));


xci_data(:,data) = xci;
yci_data(:,data) = yci;
torque_data(:,data) = torque;
aci_data(:,data) = ari(:,2);
vid_data(:,data) = vid;
wid_data(:,data) = wid;
wci_data(:,data) = velo(2,:)';
vci_data(:,data) = velo(1,:)';
x0_data(:,data) = x0;
q2ci_data(:,data) = q2ci;
q2di_data(:,data) = q2di;

%% Loop de controle com discretizacao de 0.4 segundos
for tempo = 0:deltat:tempo_f
    tic
% Velocidade do lider para a trajetoria desejada
    x0(2,1) = 0.04*sin(2*pi*tempo/tempo_f);
    x0(4,1) = 0.04*cos(2*pi*tempo/tempo_f);

%     if(tempo<=15)
%         x0(2,1) = 0;
%         x0(4,1) = 0.03;
%     end;
%     if(tempo>15 && tempo<=30)
%         x0(2,1) = -0.03;
%         x0(4,1) = 0;
%     end;
%     if(tempo>30 && tempo<=45)
```

```
%          x0(2,1) = 0;
%          x0(4,1) = -0.03;
%      end;
%      if(tempo>45 && tempo<=60)
%          x0(2,1) = 0.03;
%          x0(4,1) = 0;
%      end;

    % Resolucao da EDO para controle da Formacao
    [t1,x] = ode113(@formation,0:0.2:0.4,x0,options,A,X,h);

    % Calculo da velocidade e angulo desejados
    [velo,ari] = speed([x0'; x(3,:)],deltat,n);
    x0 = x(3,:)';

%     plot(tempo,velo(1,1),'or',tempo,velo(1,2),'ob',tempo,
   velo(1,3),'og',...
%          tempo,velo(1,4),'ok')

    %  Controle Cinematico
    q2di = control_cin(xci,yci,aci,x(3,:),velo,Kx,Ky,Ka,b,r,
       ari(:,2));

    % Controlador Dinamico
    [torque, A2, B2 ] = din([q2di_data(:,data) q2di], [dq02 (
       q2ci - q2di)]...
        ,aci,velo(2,:)',deltat,Kp,Kd,I,Iw,b,c,d,m,n);
    dq02 = (q2ci - q2di);

    % Modelo dinamico
    [t2, q2ci1] = ode113(@model_din,0:0.2:0.4,q2ci,options,A2
       ,B2,torque);
    q2ci = q2ci1(3,:)';



    % Multiplicacao pelo valor de adaptacao(o RMR devido ao
       tipo de
    % motor recebe o valor da velocidade em frequencia, por
```

```
   isso o valor
% de adaptacao)
q2ci_adpt = round(q2ci*q2_adpt);


% comunicacao com os RMRs
fprintf('tempo edo\n')
toc
fprintf('tempo comunicacao\n')
ePic1 = set(ePic1,'speed',[q2ci_adpt(1,1) , q2ci_adpt
   (2,1)]);
ePic1 = update(ePic1);
ePic2 = set(ePic2,'speed',[q2ci_adpt(3,1) , q2ci_adpt
   (4,1)]);
ePic2 = update(ePic2);
ePic3 = set(ePic3,'speed',[q2ci_adpt(5,1) , q2ci_adpt
   (6,1)]);
ePic3 = update(ePic3);
ePic4 = set(ePic4,'speed',[q2ci_adpt(7,1) , q2ci_adpt
   (8,1)]);
ePic4 = update(ePic4);
toc

%Posicao dos RMRs
fprintf('tempo vicon')
pos = viconpos();
xci(:,1) = pos(1:n,1);
yci(:,1) = pos(1:n,2);
aci(:,1) = pos(1:n,3);
toc

data = data + 1;
xci_data(:,data) = xci;
yci_data(:,data) = yci;
torque_data(:,data) = torque;
aci_data(:,data) = ari(:,2);
vid_data(:,data) = vid;
wid_data(:,data) = wid;
wci_data(:,data) = velo(2,:)';
```

```
    vci_data(:,data) = velo(1,:)';
    x0_data(:,data) = x0;
    q2ci_data(:,data) = q2ci;
    q2di_data(:,data) = q2di;



    fprintf( '\n\n');
end;
% Parada dos robos
epuckstop

% Grafico
```

As funções referentes ao sistema Vicon e ao robô ePuck encontram-se nos apêndices.

ANEXO C

Para mostrar o teste real do controle de formação, tem-se uma gravação da trajetória mostrada pela Figura 20 em um site de compartilhamento de vídeos que pode ser conferido pelo link:

https://www.youtube.com/watch?v=NRn_yaoOLGs