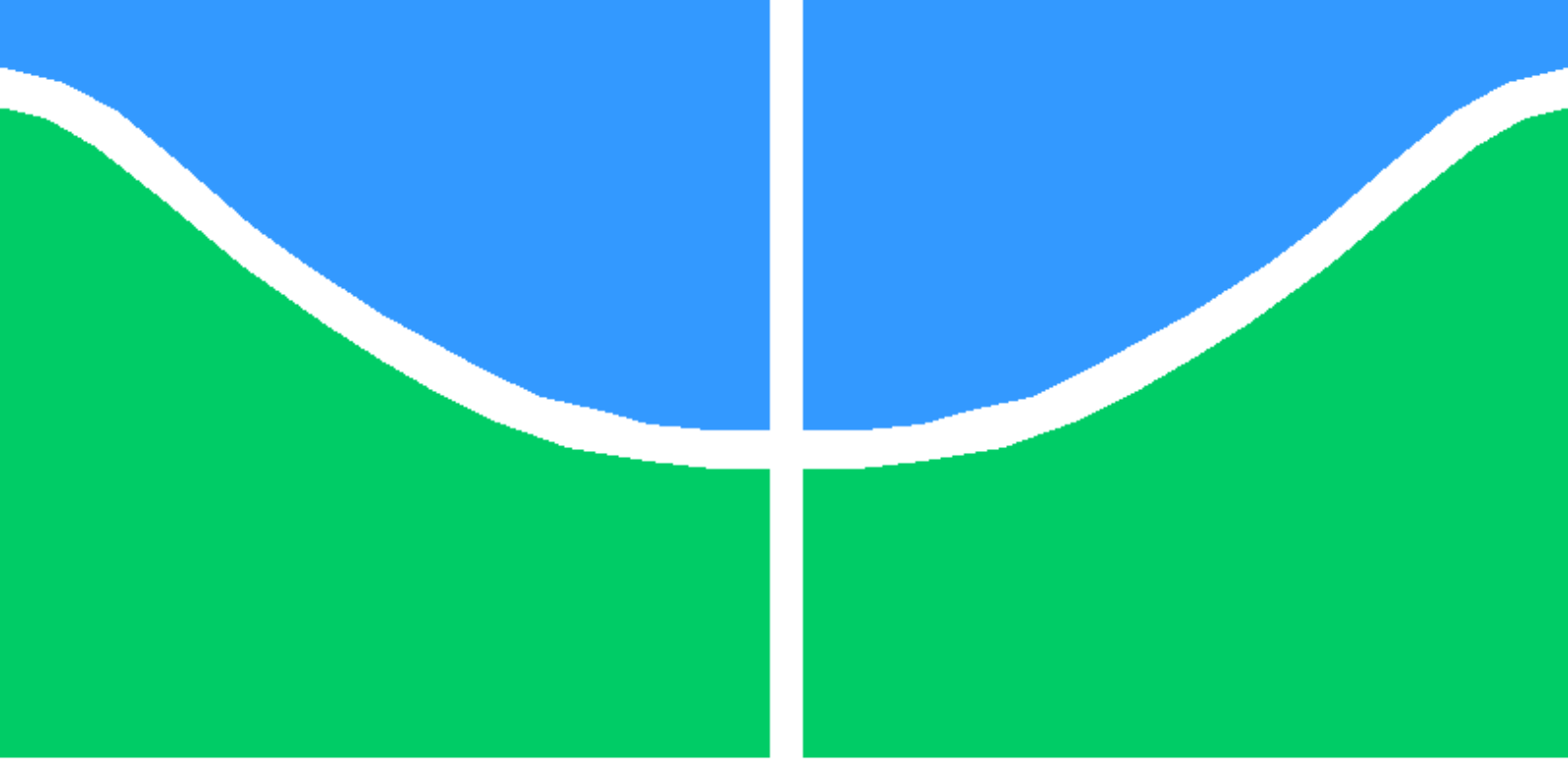


TRABALHO DE GRADUAÇÃO

EVITAMENTO DE OBSTÁCULOS COM SENSORES DE ULTRASSOM E ZONA VIRTUAL DEFORMÁVEL PARA O ROBÔ OMNI

Pedro Augusto de Carvalho Mohn

Brasília, março de 2011

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

EVITAMENTO DE OBSTÁCULOS COM SENSORES DE ULTRASSOM E ZONA VIRTUAL DEFORMÁVEL PARA O ROBÔ OMNI

Pedro Augusto de Carvalho Mohn

Relatório submetido ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Banca Examinadora

Prof. Geovany Araújo Borges, D.Sc., ENE/UnB _____
Orientador

Prof. Flávio de Barros Vidal, Dr., CIC/UnB _____
Examinador

Prof. José Alfredo Ruiz Vargas, D.Sc., ENE/UnB _____
Examinador

Dedicatória

A todos aqueles que acreditaram em mim e apoiaram-me em toda minha formação.

Pedro Augusto de Carvalho Mohn

Agradecimentos

Essencialmente a Deus,
Aos meus familiares, em especial minha mãe, Joana D'Arc, e meu pai, João Milton, que sempre torceram por mim, me acompanharam desde muito antes da universidade, e que sempre tiveram muita paciência,
Aos meus amigos,
Aos meus colegas de faculdade, aos membros do LARA/UnB pela força e paciência, e em especial ao Pedro Nehme e ao Felipe Brandão por terem me apoiado tanto, muito obrigado,
Aos meus colegas de trabalho.

Pedro Augusto de Carvalho Mohn

RESUMO

Esse projeto trata de um sistema de evitamento de obstáculos com rede de ultrassom aplicado ao robô móvel Omni. Ele tem como objetivo tornar o robô móvel capaz de tratar os sinais dos sensores de forma a calcular uma trajetória segura que evite obstáculos. Para tanto, um cinturão de sensores de ultra-som fornece medidas relacionadas à presença de obstáculos. Essas medidas são tratadas por meio de um Filtro de Kalman Estendido (EKF) e então são utilizadas para atualizar uma zona virtual deformável (ZVD) que corresponde ao espaço livre em torno do robô (zona de segurança). O evitamento é alcançado pelo comando que levar a ZVD a ficar próxima de um círculo. Os resultados foram obtidos em um ambiente de simulação e em seguida implementados no robô Omni.

ABSTRACT

This project address an obstacle avoidance system, with an ultrasound sensors array applied to the mobile robot Omni. The objective is to make the omnidirectional mobile robot capable of treating the sensors signals in a way to calculate a safe trajectory that avoid obstacles. To acomplish this task, a belt of ultrasonic sensors provide measurements related to the presence of obstacles. These measures are filtred by a Extended Kalman Filter (EKF) and used in the updating process of the Deformable Virtual Zone (DVZ), which corresponds to the free space around the robot (safety zone). The avoidance is achieved by the command that leads the trajectory become a circle. The results were obtained in a simulation environment and then implemented on the robot Omni.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

Vx	Velocidade em x do robô no sistema global de coordenadas	[m/s]
Vy	Velocidade em y do robô no sistema global de coordenadas	[m/s]
W	Velocidade angular do robô (orientação) no sistema global de coordenadas	[rad/s]
q_n	Ponto de controle da spline	[m,m]
q_n^x	Coordenada x do ponto de controle da spline	[m]
q_n^y	Coordenada y do ponto de controle da spline	[m]
$P_{virtual}$	Posição do robô virtual	[m]
P_{real}	Posição do robô real	[m]
$V_{virtual}$	Velocidade do robô virtual	[m/s]
V_{real}	Velocidade do robô real	[m/s]
A_{real}	Aceleração do robô real	[m/s^2]
X	Coordenada x da posição do robô no sistema global de coordendas	[m]
Y	Coordenada y da posição do robô no sistema global de coordenadas	[m]
θ	Ângulo de orientação do robô no sistema global de coordenadas	[rad]

Símbolos Gregos

θ	Ângulo de orientação do Omni no sistema global de coordenadas	[rad]
Δ	Variação entre duas grandezas similares	
β	Ângulo de apontamento do sensor de ultrassom	[rad]
φ	Ângulo da força de reação ao obstáculo detectado no sistema global de coordenadas	[rad]

Símbolos relacionados ao Filtro de Kalman

x_k	Estado atual (vetor coluna)
A	Matriz da dinâmica do sistema
x_{k-1}	Estado anterior (vetor coluna)
B	Matriz de controle ou de entrada
u_k	Sinal atual de controle ou de entrada
w_k	Ruído atual associado ao processo
Q_k	Matriz de covariância do ruído de processo
y_k	Saída atual
C	Matriz de resposta ou de saída
v_k	Ruído atual associado ao processo de medição
R_k	Matriz de covariância do ruído de medição
$\hat{x}_{k\|k-1}$	Estimativa do estado atual, dado o estado anterior
$\hat{x}_{k-1}$	Estimativa do estado anterior
$P_{k\|k-1}$	Matriz de incerteza associada à estimativa do estado atual dado o anterior (predição atual)
P_{k-1}	Matriz de incerteza associada à estimativa do estado anterior (correção anterior)
$\hat{x}_k$	Estimativa do estado atual
G_k	Ganho do filtro de Kalman
P_k	Matriz de incerteza associada à estimativa do estado atual (correção atual)
I	Matriz identidade
F	Matriz jacobiana de $f(\cdot)$ com respeito às variáveis do processo
H	Matriz jacobiana de $h(\cdot)$ com respeito à medição y
F_w	Matriz jacobiana de $f(\cdot)$ com respeito ao ruído w
H_v	Matriz jacobiana de $h(\cdot)$ com respeito ao ruído v

Símbolos relacionados ao Espaço de Estados do Robô

$P_{i,k}$	Ponto do iésimo landmark no instante $_k$
$x_{i,k}$	Coordenada x, no sistema global de coordenadas, do iésimo landmark no instante $_k$
$y_{i,k}$	Coordenada y, no sistema global de coordenadas, do iésimo landmark no instante $_k$
xs_i	Coordenada x, no sistema global de coordenadas, do iésimo sensor
ys_i	Coordenada y, no sistema global de coordenadas, do iésimo sensor
$c_{[i][j]}$	Elemento da matriz C localizado na linha$_i$, coluna$_j$

Grupos Adimensionais

B_n	Função polinomial básica (índice n)

Subscritos

n	índice
i	índice
k	instante atual
$_{k-1}$	instante anterior
$real$	robô real
$virtual$	robô virtual (referência)
res	resultante
med	média(o)
om	Omni (robô móvel real)

Sobrescritos

$\hat{}$	Estimativa
^{T}	Operador de transposição matricial
$\cdot$	Variação temporal
$\rightarrow$	Vetor
$-$	Valor médio

Siglas

LARA/UnB	Laboratório de Robótica e Automação da Universidade de Brasília
EKF	Filtro de Kalman Estendido (do inglês, Extended Kalman Filter)
ZVD	Zona virtual Deformável (do inglês, Deformable Virtual Zone)
LMS	Método da mínima média quadrática (do inglês, Least Mean Square)

Capítulo 1

Introdução

1.1 Contextualização

Este trabalho está inserido no campo da determinação de trajetórias em robótica móvel.

Considere o caso da movimentação de uma sonda espacial que esteja explorando Marte, e comandada por um posto localizado na Terra. A onda de rádio que comanda a sonda leva de 2 minutos e 30 segundos a 20 minutos para viajar entre os dois planetas, dependendo do alinhamento das órbitas dos planetas [1]. Logo, o sistema de navegação da sonda permanece por um tempo mínimo de 5 a 40 minutos sem receber comandos de reação do operador em relação a algum obstáculo detectado na trajetória em Marte. Caso não haja um sistema de evitamento de obstáculos implementado no sistema de navegação, a sonda estará muito vulnerável a colisões, além de ter sua velocidade de locomoção restringida a um patamar compatível com o referido lapso temporal, provavelmente alguns centímetros a cada 5 minutos, na melhor das hipóteses de alinhamento.

Considere ainda um robô móvel responsável pelo transporte de material dentro de um armazém. O operador pode traçar a trajetória do robô ou comandá-lo por um joystick até a pilha de determinado material, porém, se uma pessoa entrar na trajetória, ou caso o operador não guie precisamente o robô, é interessante que o sistema de navegação seja capaz de desviar automaticamente o robô dos obstáculos contidos na trajetória, reduzir sua velocidade, ou pará-lo, de forma a evitar colisões.

O contexto do trabalho compreende as aplicações que utilizam um robô móvel cuja integridade deve ser preservada por meio de algoritmo de controle de trajetória, e que seja comandado pelo usuário, mas que, devido ao ambiente em que está inserido, o robô esteja sujeito a colisões.

1.2 Descrição do problema

Dentre as estratégias de movimentação existentes em robótica móvel, existem duas que se destacam no problema em análise: seguimento de trajetória e evitamento de obstáculos.

[1] http://www.jpl.nasa.gov/news/fact_sheets/mpf.pdf [1]

A estratégia de seguimento de trajetória é responsável por fazer o robô seguir um sinal de referência, que pode ser uma informação de velocidade, de posição, ou de aceleração. No caso do robô móvel Omni, o sinal de referência é fornecido pelo comando do joystick, e se traduz em sinais de velocidades (Vx, Vy, e W) a serem realizadas pelo robô.

O problema a ser resolvido é garantir a integridade do robô, seguindo ao máximo a referência, os comandos do usuário. A integridade do robô deve ser priorizada de tal forma que ele evite a colisão, ainda que o usuário dê comandos ao robô que ameacem ou impossibilitem sua integridade – atribuindo menor peso à referência ou até mesmo desconsiderando-a. A invasão da zona de segurança por um obstáculo poderá ser resultado de um comando de joystick que aproxime o robô de algum obstáculo, ou devido ao movimento de algum obstáculo que se aproxime do robô.

1.3 Solução proposta

A estratégia de evitamento de obstáculos se baseia em garantir a integridade do robô, alterando a trajetória solicitada de forma a evitar colisões com obstáculos detectados por um cinturão de ultrassom. A solução prevê a especificação de uma zona de segurança ideal para o robô, e busca seu restabelecimento sempre que algum obstáculo a deforme. Essa zona é especificada como uma Zona Virtual Deformável (ZVD), ou seja, uma área virtualmente definida e tratada pelo programa responsável pela determinação da trajetória do robô real. Essa região ou zona se deforma ao serem detectados obstáculos em seu interior, de forma a retratar a realidade do robô. A zona de segurança real do robô deve ser atualizada utilizando as estimativas de um Filtro de Kalman Estendido, cujas predições são feitas com auxílio da odometria, e as correções são feitas por meio de um cinturão de 14 sensores de ultrassom. A solução também utiliza uma ZVD ideal, isto é, uma região especificada para representar a situação de segurança ideal para a integridade física do robô.

De posse da ZVD real e da ideal, o sistema quantifica as invasões de objetos, e as trata como forças de reação atuantes no robô, de modo a alterar a movimentação com base na força média de reação almejando que a ZVD real tenda à ZVD ideal, da maneira mais próxima possível do seguimento da referência.

Quando nenhum obstáculo é detectado dentro da zona de segurança, o robô se movimenta baseando-se exclusivamente na referência.

1.4 Apresentação do manuscrito

No capítulo 2 é apresentada teoria sobre Splines, sobre Filtro de Kalman e sua versão para o caso não-linear, além de teoria sobre evitamento de obstáculos em robótica móvel. O capítulo 3 apresenta a proposta do sistema de evitamento de obstáculos que se pretende implementar. Os experimentos efetuados com a solução desenvolvida, bem como seus resultados, são apresentados e discutidos no capítulo 4. Nele estão disponíveis gráficos demonstrando o seguimento da trajetória em ambientes diferentes, e imagens da trajetória em si. Esses ensaios foram realizados em um ambiente de simulação do Matlab. Por fim, as conclusões são apresentadas no capítulo 5. Os

anexos contêm os diagramas esquemáticos, feitos pelo aluno Pedro Henrique Dória Nehmé do curso de Engenharia Elétrica da UnB, referentes ao sistema de aquisição de medidas dos sonares, e a descrição do conteúdo do CD.

Capítulo 2

Fundamentação Teórica

2.1 Introdução

Nesse capítulo serão apresentadas as principais ferramentas utilizadas para o desenvolvimento desse trabalho. Uma delas, Splines [2], que são curvas parametrizadas por funções matemáticas. Foram úteis para modelar a área de segurança ao redor do robô, apresentada como uma Zona Virtual Deformável (ZVD), permitindo a utilização de algoritmos computacionais para tratar essa área. E ainda, Filtro de Kalman [3] foi a ferramenta principal de estimação dos parâmetros da ZVD, apresentada em seguida na seção 2.3. Ao final do capítulo, são apresentados conceitos relativos ao evitamento de obstáculos, tal como robôs omnidirecionais, ambientes dinâmicos, e o algorítimo de controle do princípio da ZVD, utilizado nesse projeto.

2.2 Splines

Curvas spline parametrizadas são curvas na forma $r(s) = (x(s), y(s))$, onde $x(s)$ e $y(s)$ são funções splines e s é um parâmetro que cresce ao se percorrer a curva [2].

Uma spline é uma função com intervalos de continuidade conhecidos como spans, onde cada span é determinada por uma função polinomial B_n com d coeficientes [2]. Geralmente são funções quadráticas (ordem 2, $d = 3$) ou cúbicas (ordem 3, $d = 4$). Ao se utilizar apenas polinômios de baixa ordem, contribui-se para a estabilidade e simplicidade computacional, sendo preferível aumentar o número de spans e defini-las por polinômios de baixa ordem, do que manter um baixo número de spans e aumentar a ordem dos respectivos polinômios [2].

Para calcular uma curva parametrizada $r(s)$ usando funções splines temos o seguinte:

$$r(s) = (x(s), y(s)), \tag{2.1}$$

sendo que r(s) é uma soma vetorial ponderada:

$$r(s) = \sum_{n=0}^{N_{B-1}} B_n(s) q_n \text{ para } 0 \leq s \leq L, \tag{2.2}$$

em que os pesos são os pontos de controle $q_n = (q_n^x, q_n^y)$, e os vetores são as funções básicas B_n. O somatório começa em $n = 0$ e vai até $n = N_{B-1}$, e é calculado ao longo do eixo s, para $s \in [0, L]$.

As funções componentes de $r(s)$, ou seja, $x(s)$ e $y(s)$ são somatórios análogos a (2.2), tal que, para um valor fixo de s, por exemplo $s = s_1$, a coordenada x da B-spline é dada por:

$$x(s_1) = B_0(s_1)x_0 + B_1(s_1)x_1 + B_2(s_1)x_2 + B_3(s_1)x_3 + \ldots \\ \ldots + B_{L-2}(s_1)x_{L-2} + B_{L-1}(s_1)x_{L-1}, \quad (2.3)$$

e a coordenada y da B-spline é dada por:

$$y(s_1) = B_0(s_1)y_0 + B_1(s_1)y_1 + B_2(s_1)y_2 + B_3(s_1)y_3 + \ldots \\ \ldots + B_{L-2}(s_1)y_{L-2} + B_{L-1}(s_1)y_{L-1} \quad (2.4)$$

As funções básicas B_n podem ser calculadas de forma recursiva, a partir da instância inicial definida abaixo:

$$B_{n,1}(s) = \begin{cases} 1 & \text{se } n \leq s < n+1 \\ 0 & \text{caso contrário} \end{cases} \quad (2.5)$$

e por meio do passo indutivo:

$$B_{n,d}(s) = \frac{(s-n)B_{n,d-1}(s) + (n+d-s)B_{n+1,d-1}(s)}{d-1} \quad (2.6)$$

Considere o caso em que $d = 3$, a primeira função básica de uma B-spline terá a seguinte forma:

$$B_0(s) = \begin{cases} \frac{s^2}{2} & \text{se } 0 \leq s < 1 \\ \frac{3}{4} - \left(s - \frac{3}{2}\right)^2 & \text{se } 1 \leq s < 2 \\ \frac{(s-3)^2}{2} & \text{se } 2 \leq s < 3 \\ 0 & \text{caso contrário} \end{cases} \quad (2.7)$$

As demais funções básicas são apenas cópias deslocadas:

$$B_n(s) = B_0(s-n) \quad (2.8)$$

Após o cálculo dos somatórios (2.3) e (2.4) $\forall\ s \in [0, L]$, todas as coordenadas de r(s) estão determinadas, ou seja, a curva parametrizada por funções B-spline está calculada. O motivo dessa denominação advém de os parâmetros serem as funções $x(s)$ e $y(s)$, que são funções B-splines. Essas funções recebem esse nome porque são splines baseadas em funções básicas B_n [2].

É interessante destacar um caso particular de curvas parametrizadas por B-splines. Se o intervalo $[0, L]$ de s for tomado como periódico, $r(s)$ será uma curva fechada [2]. Nesse caso, o eixo s pode ser visto como um "eixo circular", e as funções $B_n(s)$ podem ser consideradas em sua totalidade, até mesmo quando $s > L$: isso porque o trecho que se estende além de $s = L$ é calculado até $s = L$ e a parte extrapolante é calculada em $s \geq 0$. Por exemplo, no caso de uma spline de ordem $d = 3$, sua última função B_n ($B_{NB-1}(s)$) começa em $s = L - 1$, chega em $s = L$, "extrapola o limite" de $s = L$ e é não-nula para $s \in [0, 2]$. Portanto, o cálculo da B-spline em determinado

trecho deve considerar as funções básicas que começam dentro do trecho, bem como as que, embora não comecem nele, terminam dentro dele. Isso porque possuem partes extrapolantes ao limite $s = L$, que continuam em $s = 0$, e adentram esse trecho. Dessa forma, todas as funções básicas são inteiramente calculadas e utilizadas na parametrização da curva r(s) em todo o intervalo $s \in [0, L]$.

Ainda considerando a hipótese de $s = [0, L]$ ser um intervalo periódico, suponha que os pontos de controle $q_n = (q_n^x, q_n^y)$ de $r(s)$ sejam delimitadores de determinada área. Nessa situação, os pontos seriam envolvidos e a área seria delimitada por essa curva $r(s)$ [2]. Por exemplo, se os pontos-limite ao redor de um robô forem utilizados como pontos de controle para o cálculo de uma curva parametrizada por B-splines, a área livre ao seu redor poderia ser representada por meio de tal curva, e a área poderia ser abordada por meio de um modelo matemático.

A Figura 2.1 [4] ilustra essa possibilidade de utilizar Splines para especificar Zonas Virtuais Deformáveis (ZVD). O tracejado indicado por Θ (elipse externa) se refere ao alcance máximo dos sensores, e o indicado por Ψ representa o conjunto de informações captadas pelos sensores, ou seja, se refere às informações sobre o ambiente. Note que devido a presença de um obstáculo detectado pelos sensores, há uma deformação do tracejado Ψ em relação ao tracejado Θ. Essa deformação está sendo representada por I. Os tracejados Ξ_h e Ξ_I referem-se a Zonas Virtuais Deformáveis. O tracejado Ξ_h (elipse interna) representa uma ZVD controlada ou ideal. O tracejado Ξ_I se refere a ZVD real, denota a área livre ao redor do robô, e depende da deformação I. A deformação de Ξ_I em relação a Ξ_h está sendo representada por Δ.

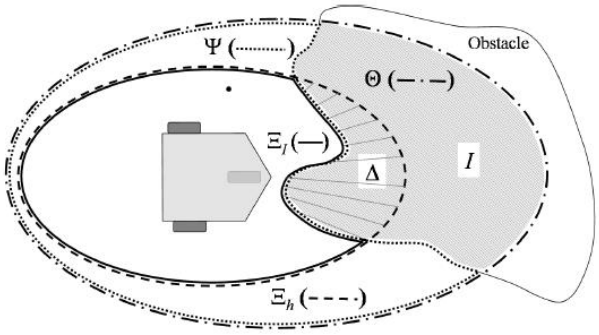


Figura 2.1: Robô e áreas especificadas referentes ao princípio da ZVD

Note que a presença de obstáculos deforma a ZVD real, porém a ZVD ideal permanece inalterada, pois sua especificação independe de obstáculos ao redor do robô.

2.3 Filtro de Kalman

O filtro de Kalman é o resultado de um processo evolucionário de ideias que surgiram ao longo de muitos séculos, e se refere ao processo de estimação do estado de um sistema dinâmico. Entre os colaboradores para essa tecnologia estão Gauss, Galilei, Newton, Riccati, Bayes, Markov, Wiener, Kolmogorov, Peter Swerling, e Rudolf Emil Kalman [3].

O filtro de Kalman faz a estimação dos estados de um sistema dinâmico linear perturbado por um ruído gaussiano de média nula, onde as medidas são relacionadas linearmente às variáveis de

estado [3]. O filtro de Kalman é uma ferramenta matemática alimentada pelo modelo do processo e respectivo ruído, pelas medições e respectivo ruído de medição, pelo estado inicial do vetor de estados e respectiva matriz de covariância (incerteza inicial). Sua fundamentação matemática envolve a teoria de probabilidade, sistemas dinâmicos, método dos quadrados mínimos, sistemas estocásticos e método da mínima média quadrática – LMS (do inglês, Least Mean Square) [3].

Trata-se de um estimador estatisticamente ótimo em relação a qualquer função quadrática de estimação de erro [3]. Essa característica motiva seu uso, quando se está diante de situações como a descrita acima. Entre as áreas de aplicação para essa ferramenta, pode-se citar controle de processos, rastreamento, e navegação [3].

Como já foi dito, o filtro de Kalman é uma ferramenta que exige o modelo do processo – cujas variáveis de estado deseja-se estimar –, o ruído associado, a entrada do processo, o estado inicial e respectiva matriz de incerteza (valores iniciais das variáveis de estado e incerteza associada a essa informação), e as medições e respectiva incerteza associada [3]. Ele também exige que o sistema seja linear, isto é, que o sinal de controle e as variáveis de estado sejam linearmente relacionadas, de tal forma que o sistema possa ser descrito conforme (2.9) e (2.10). Além disso, ele exige que o ruído do processo e o de medição, e o estado inicial sejam descorrelacionados entre si.

Considere x o vetor de variáveis de estado de um processo dinâmico que se deseja estimar, A a matriz do processo, u o sinal de entrada ou de controle, B a matriz de entrada, e w o ruído branco do processo. Considerando ainda y o vetor das medições das variáveis de estado, C a matriz de saída, v o ruído associado às medições, e v e w são descorrelacionados entre si e descorrelacionados de x_0 que podemos fazer a seguinte definição:

Modelo do sistema linear:

$$x_k = Ax_{k-1} + Bu_k + w_k,\ w_k \sim N(0, Q_k) \tag{2.9}$$

$$y_k = Cx_k + v_k,\ v_k \sim N(0, R_k) \tag{2.10}$$

Baseado no sistema descrito acima, temos as seguintes equações do Filtro de Kalman:

Fase de predição:

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1} + Bu_k \tag{2.11}$$

$$P_{k|k-1} = AP_{k-1}A^T + Q_k \tag{2.12}$$

Fase de correção:

$$\hat{x}_k = \hat{x}_{k|k-1} + G_k(y_k - C\hat{x}_{k|k-1}) \tag{2.13}$$

$$G_k = P_{k|k-1}C^T(CP_{k|k-1}C^T + R_k)^{-1} \tag{2.14}$$

$$P_k = (I - G_kC)P_{k|k-1} \tag{2.15}$$

w_k e v_k são ruídos brancos e, portanto, são variáveis aleatórias de média nula. A primeira denota o ruído do processo e possui matriz de covariância Q_k, a segunda denota o ruído de medição e possui matriz de covariância R_k. Pelo fato de terem média nula, essas variáveis não entram no cálculo da estimativa do estado. No entanto, por apresentarem matriz de covariância, elas entram no cálculo da incerteza associada à estimativa.

A equação (2.13) corrige a predição ($\hat{x}_{k|k-1}$) baseada na medição realizada (y_k). Na equação (2.14), G_k é o ganho do filtro de Kalman. Por sua vez, a equação (2.15) calcula a incerteza associada à supracitada correção. Naturalmente há incerteza associada ao estado corrigido ($\hat{x}_k$) e ao estado predito ($\hat{x}_{k|k-1}$), ensejando a seguinte argumentação:

$$\begin{cases} P_k = (I - G_k C)P_{k|k-1}, \quad P_k > 0, \quad P_{k|k-1} > 0 \qquad \Rightarrow \quad (I - G_k C) > 0 \\ \Rightarrow G_k C < I \\ \Rightarrow I - G_k C < I \\ \therefore P_k < P_{k|k-1} \end{cases} \tag{2.16}$$

Diante de (2.16), tem-se que a fase de predição aumenta a incerteza da estimativa, enquanto que a fase de correção diminui essa incerteza.

Deve-se observar que as fórmulas apresentadas são válidas apenas para sistemas dinâmicos lineares [3]. Caso o sistema dinâmico cujo estado deseja-se estimar não seja linear, há que se recorrer ao Filtro de Kalman Estendido – EKF (do inglês, Extended Kalman Filter). Essa versão do Filtro de Kalman consiste na linearização de primeira ordem do modelo dinâmico a ser estimado.

As equações descritivas do modelo (2.9) e (2.10), as equações da fase de predição (2.11) e (2.12), e as equações da fase de correção (2.13) a (2.15) são substituídas pelas seguintes:

Modelo do sistema não-linear:

$$x_k = f(x_{k-1}, u_{k-1}, w_k) \quad , w_k \sim N(0, Q_k) \tag{2.17}$$

$$y_k = h(x_k, v_k) \quad , v_k \sim N(0, R_k) \tag{2.18}$$

Fase de predição do EKF:

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1}, u_{k-1}, 0) \tag{2.19}$$

$$P_{k|k-1} = F P_{k-1} F^T + F_w Q_k F_w^T \tag{2.20}$$

Onde F é a matriz jacobiana de $f(\cdot)$ com respeito às variáveis do processo, e F_w é a matriz jacobiana de $f(\cdot)$ com respeito ao ruído w.

Fase de correção do EKF:

$$\hat{x}_k = \hat{x}_{k|k-1} + G_k(y_k - h(\hat{x}_{k|k-1}, 0)) \tag{2.21}$$

$$G_k = P_{k|k-1} H^T (H P_{k|k-1} H^T + H_v R_k H_v^T)^{-1} \tag{2.22}$$

$$P_k = (I - G_k H) P_{k|k-1} \tag{2.23}$$

Note que, quando do cálculo do estado estimado, os últimos parâmetros das funções $f(\cdot)$ e $h(\cdot)$ são nulos, devido ao fato de w_k e v_k serem ruídos brancos, conforme já comentado.

Devido às aproximações inerentes à linearização de primeira ordem do modelo dinâmico utilizada no EKF, esse estimador é sub ótimo.

2.4 Evitamento de Obstáculos

Na literatura sobre robótica móvel há diversas estratégias para o evitamento de obstáculos em diferentes situações. Dentre os algoritmos de controle e situações que se pretende destacar encontram-se: princípio da Zona Virtual Deformável (ZVD) [4] e [5], aplicações para robôs omnidirecionais [6], e aplicações envolvendo ambientes dinâmicos [5] e [7].

O princípio da ZVD corresponde à definição de uma zona de proteção deformável, sobre a qual a presença de um obstáculo induz uma informação de intrusão que impulsiona a reação do veículo [4]. Basicamente, trata-se de definir uma zona-limite ao redor do veículo, verificar se há objetos invadindo esse espaço e reagir quando for o caso, buscando manter a integridade dessa região, e consequentemente a integridade do robô.

Essa técnica apresenta duas principais vantagens: dispensa conhecimento prévio do ambiente, e o controlador pode lidar com outras restrições além do evitamento de obstáculos, por exemplo busca de alvos, manutenção de altitude, e seguimento de trajetória [5]. No entanto, essa abordagem pode apresentar o problema de mínimo local [5], resultando em bloqueio na movimentação do robô.

Robôs omnidirecionais possuem a capacidade de se movimentar em qualquer direção independentemente de sua orientação. Dessa forma, a pose do robô $\xi_k = \begin{pmatrix} X_k \\ Y_k \\ \theta_k \end{pmatrix}$ pode ser alterado para uma outra configuração $\xi_{k+1} = \begin{pmatrix} X_{k+1} \\ Y_{k+1} \\ \theta_{k+1} \end{pmatrix}$ sem que uma variável dependa de outra. Sendo assim, o robô pode andar em uma direção, apesar de estar orientado para outra.

Para explicitar essa capacidade, considere um automóvel de passeio. Somente as rodas dianteiras possuem ajuste de direção (uso do volante), de modo que a parte traseira é arrastada na direção apontada pela parte fronteira. Sendo assim, o carro só consegue se mover na direção para a qual está direcionado ("apontado"). Para que esse veículo possa estacionar em uma vaga como a da Figura 2.2 [1], é necessário que o motorista faça uma manobra de baliza, ajustando a orientação do carro sempre que desejar mudar a direção da trajetória. Se os carros de passeio fossem omnidirecionais, a solução se resumiria a posicionar o carro ao lado da vaga, sem mudar sua orientação, direcionar as rodas para à direita e então acelerar, permitindo a locomoção do carro conforme a Figura 2.3 [2].

Aproveitando-se dessa capacidade inerente a robôs omnidirecionais, pode-se fazer uma abordagem por função de penalidade exterior. Esse algoritmo baseia-se em determinar uma área ao redor dos obstáculos e quanto mais o robô invade essa área, maior é a repulsão que o objeto gera no robô [6]. Isso resulta em velocidades suaves quando o robô está próximo de riscos à sua integridade, e em movimentos tangenciais às áreas dos obstáculos. Sendo assim, essa estratégia evita colisões, afastando-se minimamente da referência. No entanto, ela apresenta a desvantagem de exigir o conhecimento prévio do ambiente [6].

Ambientes dinâmicos são aqueles cuja configuração é alterada independentemente da locomoção do robô ou independentemente de suas ações. Por exemplo, uma praça frequentada por pessoas, ou um estacionamento de carros, ou ainda um ambiente contendo um grupo de robôs móveis

[1] http://www.bancnotestonny.ro/ com adaptações
[2] http://www.bancnotestonny.ro/ com adaptações

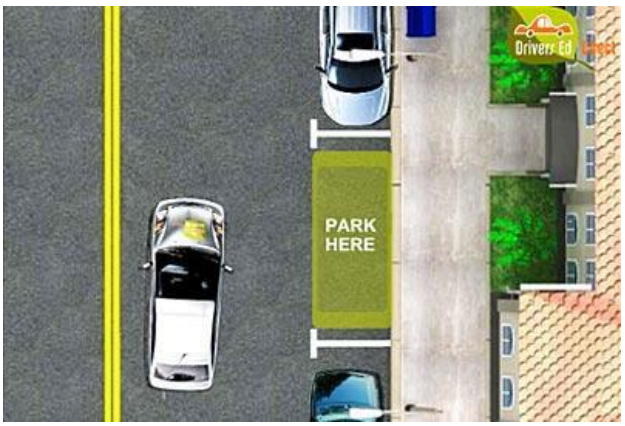


Figura 2.2: Vaga de baliza

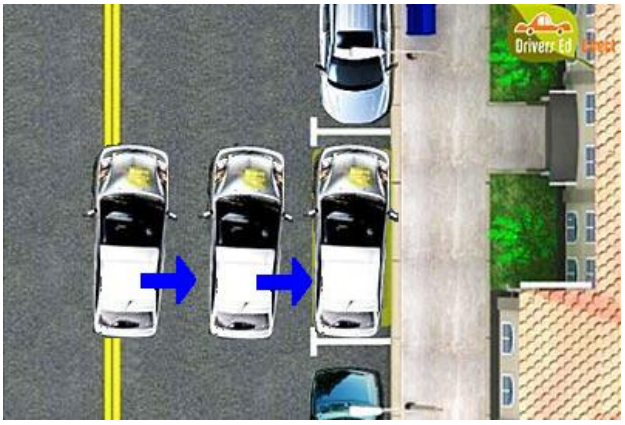


Figura 2.3: Solução omnidirecional

utilizando controle descentralizado. Uma abordagem interessante para situações como essas envolve a definição de um estado denominado Estado de Colisão Inevitável (do inglês, Inevitable Collision State – ICS) [7]. A característica dessa abordagem é que, caso o sistema atinja o ICS, uma colisão ocorrerá em um momento posterior não importa qual seja a trajetória futura do sistema [7]. O algoritmo consiste em evitar o ICS, levando em consideração o comportamento futuro dos objetos móveis, fazendo o sistema agir por antecipação.

Outra forma de solucionar o problema do ambiente dinâmico é apresentada em [5]. Trata-se de utilizar uma zona de proteção deformável e interagir com o ambiente por meio de sensores. A ZVD é parametrizada pelas variáveis de movimento do robô móvel e pode deformar diante de informações de distância. Sempre que um objeto entra no alcance do sensor, ele induz uma deformação na ZVD, que será compensada pelo controle de movimento do robô. Note que essa solução não exige conhecimento prévio sobre o ambiente, e ele pode ser estático ou dinâmico, que ainda assim o sistema de controle é capaz de reagir à situação em que o robô se envolver.

Capítulo 3

Proposta de um sistema de evitamento de obstáculos

3.1 Introdução

A metodologia utilizada para desenvolvimento do presente trabalho consiste essencialmente de três partes, quais sejam: controle de trajetória, evitamento de obstáculos, e a combinação das duas estratégias de movimentação. O controle de trajetória se baseia em fazer o robô seguir o comando do *joystick* externo (USB). O evitamento de obstáculos se refere à comunicação com os sensores, à captação de suas medidas para percepção do ambiente, e ao cálculo de um comando que reaja ao ambiente visando à integridade do robô móvel. A combinação envolve atribuir pesos a esses comandos de forma que o robô móvel se movimente de uma maneira segura, evitando obstáculos presentes na trajetória solicitada pelo operador, objetivo do trabalho.

3.2 Controle de Trajetória

3.2.1 Lei de Controle

Deseja-se controlar a posição do robô real: $\binom{x}{y}_r = P_{real}$. Considerando que o movimento do robô é uniformemente variado (M.U.V.), isto é, considerando que a aceleração do robô é constante, a equação horária abaixo descreve seu movimento.

$$S = S_0 + V_0 t + at^2/2 \tag{3.1}$$

Deseja-se ainda utilizar uma lei de controle para seguimento da referência que considere a posição e a velocidade dos robôs real e virtual, bem como que a taxa de variação do erro seja proporcional ao próprio erro, fazendo-o diminuir exponencialmente.

Para tanto, considere o seguinte erro de posição:

$$\mathbf{e} = \begin{pmatrix} x \\ y \end{pmatrix}_v - \begin{pmatrix} x \\ y \end{pmatrix}_r = P_{virtual} - P_{real} \tag{3.2}$$

E considere a seguinte taxa de variação do erro, de forma a atender o requisito anterior:

$$\dot{\mathbf{e}} = -\mathbf{K}\mathbf{e} \tag{3.3}$$

Onde $\mathbf{K}$ é uma matriz positiva a ser definida.

Uma abordagem por auto-valores e auto-vetores leva a seguinte fórmula:

$$\dot{\mathbf{e}} = \mathbf{A}\mathbf{e} \tag{3.4}$$

Matriz de auto-valores:

$$\mathbf{A} = \begin{pmatrix} -\lambda_1 & 0 \\ 0 & -\lambda_2 \end{pmatrix} \tag{3.5}$$

A escolha a seguir resulta em um sistema estável, por ter pólos reais idênticos, contidos no semi-plano esquerdo (spe):

$$K = -A = \lambda I \tag{3.6}$$

Derivando a equação (3.2) em relação ao tempo, temos o seguinte:

$$\dot{\mathbf{e}} = \dot{P}_{virtual} - \dot{P}_{real} \tag{3.7}$$

$$= V_{virtual} - V_{real} \tag{3.8}$$

Onde $V_{virtual}$ e V_{real} se referem a velocidade do robô virtual e a do real.

Então, substituindo-se 3.3 em 3.7:

$$\dot{P}_{real} = V_{real} = Ke + \dot{P}_{virtual} \tag{3.9}$$

Derivando a equação (3.9) em relação ao tempo, temos o seguinte:

$$\dot{V}_{real} = A_{real} = K\dot{e} + \ddot{P}_{virtual} = -K^2 e + \dot{V}_{virtual} \tag{3.10}$$

Com as equações (3.2), (3.5), (3.6), (3.9), e (3.10), temos todas as variáveis necessárias para determinar a posição do robô real por meio da equação horária do movimento uniformemente variado.

Assim, podemos reescrever a equação 3.1, como:

$$P_{real,_k} = P_{real,_{k-1}} + V_{real,_k} t + A_{real,_k} t^2/2 \tag{3.11}$$

Onde $P_{real,_k}$ é a posição do robô real no instante k e é a variável que queremos controlar, $V_{real,_k}$ é a velocidade do robô real no instante k e foi definida em 3.9, t é o tempo, e $A_{real,_k}$ é a aceleração do robô real no instante k e foi definida em 3.10.

Essa equação baseia-se na consideração inicial de que a aceleração do robô é constante. Tal consideração é razoável, uma vez que o intervalo entre cada cálculo de posição é suficientemente pequeno (aproximadamente 5 ms).

A equação acima é utilizada para seguimento da referência, no entanto o movimento realizado pelo Robô Omni não é determinado apenas por essa estratégia de movimentação. Ele possui um segundo componente, qual seja, o evitamento de obstáculos.

3.3 Evitamento de Obstáculos

3.3.1 Cinturão de Sonares

Para que o Robô Omni possa evitar obstáculos, é necessário que ele perceba o ambiente ao seu redor. Tal percepção é feita por meio de um cinturão de 14 sensores de ultrassom para detecção de objetos ao redor do robô, seguindo a configuração *daisy chain*. O sistema de ultrassom, bem como seu protocolo de comunicação foi desenvolvido pelos ex-alunos do curso de Engenharia Mecatrônica da UnB Daniel Fraianeli e Tiago Adnet.

O modelo dos sensores utilizados é o SRF02 da Devantech, que permite a leitura da distância a cada 70ms. A comunicação com os sensores ocorre sob o padrão serial RS-485, impossibilitando uma comunicação direta através da porta serial do computador (padrão RS-232). Para realizar a comunicação, a *daisy chain* dos sensores foi conectada a um microcontrolador Atmega 8 da Atmel conectado à porta serial (COM 1) do computador através de um transceiver Maxim MAX232, conforme esquemático contido na figura I.1, incluída no anexo. O protocolo de comunicação gravado no microcontrolador está contido no CD (anexo deste relatório). Na tabela 3.1 constam os tipos de comandos interpretados pelo protocolo e as respectivas respostas.

Tabela 3.1: Diretivas de comandos e respostas do protocolo de comunicação dos sonares

Dados enviados	**Dados obtidos**
'?S0'	tempo em microssegundos do sensor 0
'?S1'	tempo em microssegundos do sensor 1
'?S2'	tempo em microssegundos do sensor 2
⋮	⋮
'?S13'	tempo em microssegundos do sensor 13

O tempo em microssegundos é descrito por uma palavra de 16 bits e se refere ao tempo que a onda de som levou para sair do SRF02, atingir um objeto e retornar. No entanto, o microcontrolador não envia uma palavra de 16 bits, ao invés disso ele envia 2 bytes à porta serial do computador. O primeiro byte obtido pelo computador é o mais significativo e o segundo é o menos

significativo da palavra de 16 bits. Sendo assim, os 2 bytes devem ser concatenados e decodificados de binário para decimal, para então ser utilizado na determinação da distância entre o robô e o objeto detectado.

Para se obter a distância do robô ao objeto, utiliza-se a seguinte fórmula:

$$2 \cdot Distância = (tempo/1000000) \cdot v_{som} \tag{3.12}$$

Conforme informado, os módulos utilizados permitem que ocorra uma leitura a cada 70ms, no entanto a movimentação do robô ocorre a cada 5ms. Dessa forma, a latência da aquisição de dados de um sonar é 14 vezes maior que a da movimentação, e a da aquisição de todos os sonares é 196 vezes maior do que a da movimentação. Portanto, as medidas ficam desatualizadas rapidamente enquanto o robô se movimenta. Outro problema na percepção do ambiente é nem sempre o processo de medição dos sonares ocorrer, resultando em eventuais faltas de leitura, e ser ruidoso.

A Figura 3.1 mostra um diagrama de blocos referente à aquisição de dados dos sensores.

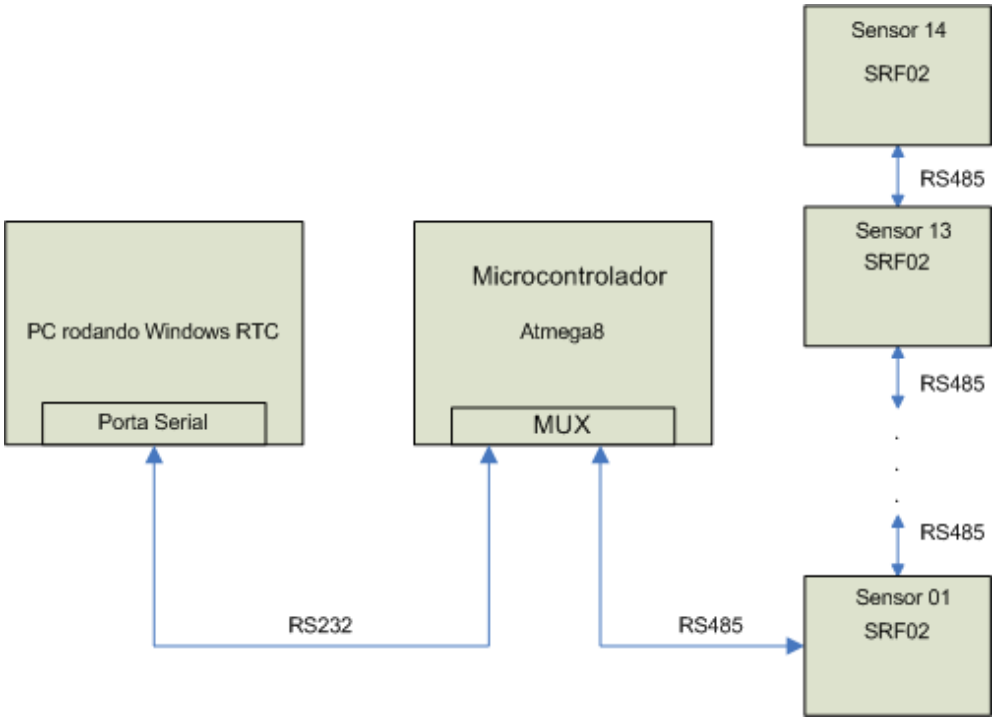


Figura 3.1: Diagrama de Blocos: Sistema de aquisição de medidas

3.3.2 Percepção do Ambiente

Diante dos problemas citados na subseção anterior, faz-se necessário tratar as medidas, de forma a se obter uma percepção mais confiável do ambiente. Para solucionar esse problema, optou-se por utilizar um estimador da posição dos *landmarks*. *Landmarks* são as extremidades observáveis do ambiente, que delimitam a zona de segurança ao redor do robô.

Inicialmente foi considerado utilizar o Filtro de Kalman para solucionar os problemas de ruído e de ausência de medição, por ele ser um estimador ótimo, conforme consta na seção 2.3. No entanto, como se escolheu os *landmarks* do ambiente para serem as variáveis de estado, o sistema dinâmico em análise não pode ser descrito por um modelo linear, o que impossibilita o uso do Filtro de Kalman.

Sendo assim, optou-se por utilizar o Filtro de Kalman Estendido (EKF), que é um estimador sub-ótimo devido às aproximações inerentes à linearização do modelo. O sistema de percepção do ambiente utiliza a fase de predição do estimador escolhido durante a latência das medições, e utiliza a fase de correção quando recebe uma nova medida. A fase de predição utiliza as informações de odometria para suprir a falta de medição, e a fase de correção atualiza as variáveis de estado a partir do estado predito e por meio da nova leitura do sonar.

As variáveis de estado são as coordenadas x e y de cada *landmark* lido pelo sensor de ultrassom, ou seja, o sistema possui 28 variáveis de estados, referentes aos 14 pares (x, y) dos *landmarks*. Abaixo segue a forma geral das variáveis de estado:

$$P_{i,k} = \begin{bmatrix} x_{i,k} \\ y_{i,k} \end{bmatrix} = \begin{bmatrix} x_{i,k-1}\cos(\Delta\theta_k) + y_{i,k-1}\sin(\Delta\theta_k) - \Delta X_{om,k}\cos(\Delta\theta_k) - \Delta Y_{om,k}\sin(\Delta\theta_k) \\ -x_{i,k-1}\sin(\Delta\theta_k) + y_{i,k-1}\cos(\Delta\theta_k) + \Delta X_{om,k}\sin(\Delta\theta_k) - \Delta Y_{om,k}\cos(\Delta\theta_k) \end{bmatrix} \tag{3.13}$$

Onde o índice i vai de 0 a 13 e se refere a cada *landmark*, e os índices $k-1$ e k se referem aos instantes anterior e atual, e onde $\Delta X_{om,k}$, $\Delta Y_{om,k}$, $\Delta\theta_k$ se referem aos sinais de controle e são respectivamente: a variação na coordenada x do robô, a variação na coordenada y do robô, a variação no ângulo de orientação do robô.

Adotando-se $\Delta X_{om,k}$, $\Delta Y_{om,k}$, $\Delta\theta_k$ como sendo os sinais de controle (u_k), temos:

$$u_k = \begin{bmatrix} \Delta X_{om,k} \\ \Delta Y_{om,k} \\ \Delta\theta_k \end{bmatrix} \tag{3.14}$$

As equações (3.13) e (3.14) permitem que o vetor de estado seja definido como:

$$x_k = \begin{bmatrix} P_{0,k} \\ P_{1,k} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ P_{12,k} \\ P_{13,k} \end{bmatrix} = \begin{bmatrix} x_{0,k} \\ y_{0,k} \\ x_{1,k} \\ y_{1,k} \\ \vdots \\ x_{12,k} \\ y_{12,k} \\ x_{13,k} \\ y_{13,k} \end{bmatrix} = f(x_{k-1}, u_{k-1}) + w_k \tag{3.15}$$

Onde w_k é o sinal de ruído do processo.

Conforme equações acima, as variáveis de estado $x_{i,k}$ e $y_{i,k}$ podem ser descritas em função das

variáveis de estado no momento anterior ($x_{i,k-1}$ e $y_{i,k-1}$), e em função do sinal de controle u_k. No entanto, note que o modelo do sistema é não-linear, pois os coeficientes que relacionam a entrada com as variáveis de estado são funções senoidais. Ressalta-se ainda que, devido às características do sistema de aquisição de dados dos sonares, conforme discutido anteriormente, há uma grande latência na disponibilização das medidas a serem utilizadas na fase de correção do estimador. Caso o estimador realize a correção apenas quando for obtida a medida de todos os sensores, o estado do sistema estará sendo estimado predominantemente pela fase de predição, que aumenta a incerteza da estimativa, conforme explicitado na seção 2.3.

Pelo sistema ser não-linear, e almejando-se minimizar o problema da referida latência, optou-se por utilizar o EKF sequencial [8] como estimador do sistema.

Essa escolha implica em algumas ponderações para se definir a equação da saída. Primeiramente, os sonares não medem diretamente o par de coordenadas $(x_{i,k}, y_{i,k})$ de um landmark, ao invés disso, no instante $_k$ o i-ésimo sensor mede a distância $D_{i,k}$. Ademais, não há que se falar em atualização de todas as variáveis de estado em um único passo, mas sim na atualização de cada par $(x_{i,k}, y_{i,k})$ por vez, isto é, a cada vez que uma nova medida se torna disponível.

Faz-se necessário determinar meios de se calcular as coordenadas $(x^s_{i,k}, y^s_{i,k})$ do landmark detectado a partir da medição do i-ésimo sensor no instante $_k$, e ainda determinar meios para que a atualização possa ocorrer a cada obtenção de uma nova medida.

As fórmulas (3.16) e (3.17) atendem ao primeiro requisito.

$$x^s_{i,k} = xs_i + D_{i,k}cos(\beta_i) \tag{3.16}$$

$$y^s_{i,k} = ys_i + D_{i,k}sin(\beta_i) \tag{3.17}$$

A partir dessas equações, e denotando (x_j, y_j) como sendo as coordenadas do landmark$_j$ do vetor de estados, pode-se atender ao segundo requisito relacionando a nova medida realizada pelo i-ésimo sensor com as variáveis de estado do sistema definindo a equação da saída como:

$$y_k = h(x_k, v_k) = \mathbf{C}x_k + v_k = \begin{bmatrix} x^s_{i,k} \\ y^s_{i,k} \end{bmatrix} + v_k \tag{3.18}$$

Onde x_k é o vetor de estado e v_k é o ruído de medição.

$$C_{2\ \mathrm{x}\ 14} = \begin{pmatrix} 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 & \cdots & 0 & 0 \end{pmatrix} \tag{3.19}$$

$$\begin{cases} c_{[1][j]} = c_{[2][j+1]} = 1 \iff j = \arg\min \sqrt{(x^s_i - x_j)^2 + (y^s_i - y_j)^2} \\ c_{[i][j]} = 0 \qquad \text{nos demais casos} \end{cases} \tag{3.20}$$

As equações de formação acima têm o objetivo de selecionar qual par de variáveis de estado deve ser atualizado ou corrigido no EKF sequencial, diante de uma nova medição ou detecção de

um *landmark*. Ela foi definida de tal forma que o par $\binom{x_{i,k}}{y_{i,k}} = P_{i,k}$ a ser atualizado seja aquele mais próximo do *landmark* detectado ou medido $\binom{x^s_{i,k}}{y^s_{i,k}}$.

Diante de (3.13) a (3.20), o sistema dinâmico não-linear em análise já pode ser estimado utilizando as equações do EKF sequencial na forma definida por (2.19) a (2.23).

3.3.3 Definição da ZVD e Cálculo da Força de Reação

A estratégia de evitamento de obstáculos proposta se baseia na especificação de uma ZVD nominal ou ideal, na verificação se a especificação está sendo atendida, e finalmente no cálculo de uma reação a eventuais discrepâncias na especificação.

É possível definir uma ZVD utilizando a teoria sobre curvas parametrizadas por B–Splines, conforme discutido na Seção 2.2. Conforme subseção 3.3.2, o sistema de navegação obtém informações sobre o ambiente físico em que o robô Omni está inserido. De posse dessas informações, pode-se verificar se a zona de segurança real está de acordo com a zona de segurança ideal, ou seja, a ZVD especificada. Quando esse requisito não é atendido, o sistema de controle emula a existência de uma força de reação necessária para que a ZVD especificada seja reestabelecida.

O sistema de controle calcula uma força de reação para cada discrepância detectada pelos sensores. Cada obstáculo percebido dentro da zona de segurança especificada gera uma força, com módulo numericamente igual à discrepância, e na mesma direção de apontamento do sensor que o detectou, porém no sentido oposto ao de apontamento, por se tratar de uma reação ao obstáculo. O sistema então calcula a força resultante e a força média de reação aos obstáculos do ambiente.

A força média terá mesma direção e mesmo sentido da força resultante (soma vetorial das componentes), porém sua intensidade será igual à da força resultante dividida pelo número de componentes, conforme explicitado a seguir:

Força de reação:

$$|\vec{F}_i| = limite - D_i, \qquad \forall_i | D_i < limite \tag{3.21}$$

$$|\vec{F}_i| = 0, \qquad \text{para os demais casos} \tag{3.22}$$

$$\varphi_i = \beta_i + \pi \tag{3.23}$$

$$\vec{F}_{res} = \sum_{i=0}^{13} \vec{F}_i, \qquad \forall_i | D_i < limite \tag{3.24}$$

$$\vec{F}_{med} = \vec{F}_{res}/n \tag{3.25}$$

Nas fórmulas acima, D_i é a distância medida pelo i-ésimo sensor até o obstáculo, $limite$ é a distância especificada como mínima para que o sistema de evitamento de obstáculo atue, φ_i é o ângulo da i-ésima força de reação (sist. coord. global), β_i é o ângulo de apontamento do i-ésimo sensor de ultrassom, e n é a quantidade de sensores que detectaram algum obstáculo invadindo a zona de segurança, ou seja, o número de forças $\vec{F}_i$ não-nulas.

Ressalte-se que os dados da componente $\vec{F}_i$ são obtidos pelos sonares, e a força média é calculada pelo sistema computacional do robô.

As figuras 3.2 e 3.3 exemplificam o conceito das forças tratadas nas fórmulas acima.

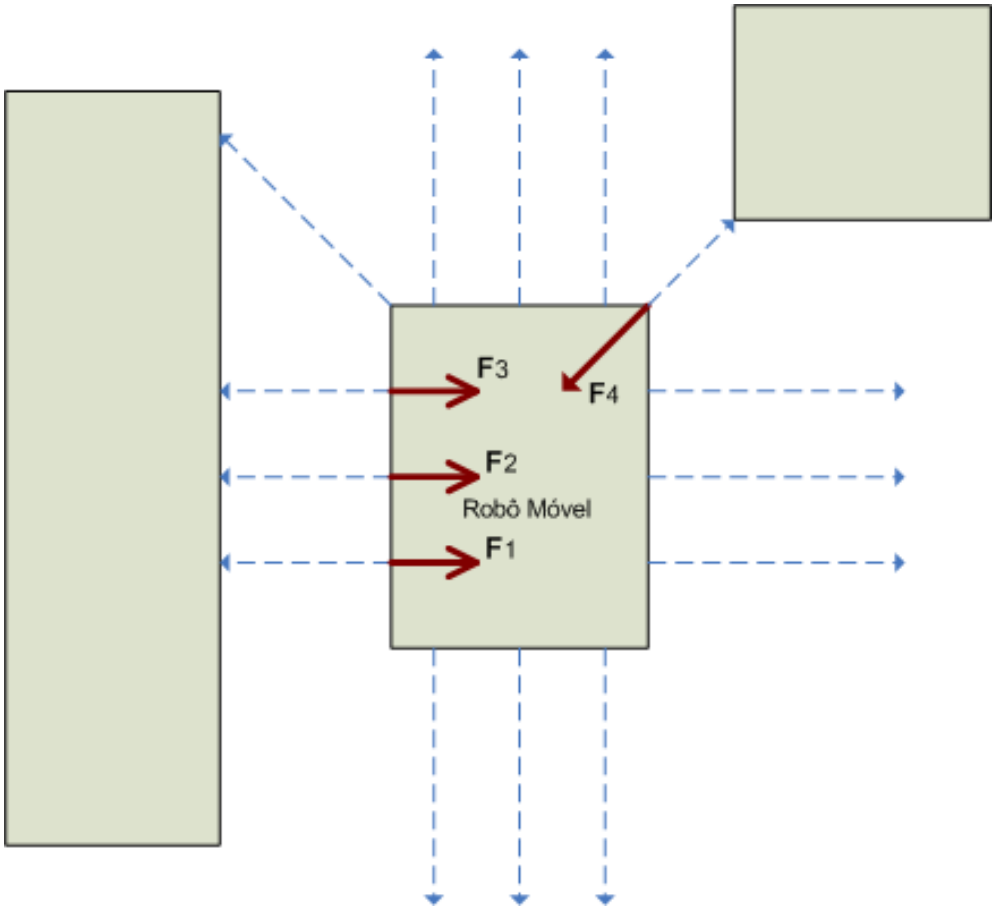


Figura 3.2: Diagrama de forças reacionais

Na figura 3.2, os vetores pontilhados se referem à leitura dos sonares. Os vetores superiores, inferiores, na diagonal superior esquerda, e os vetores à direita do robô móvel possuem comprimento igual ao *limite*, ou seja, seus respectivos sensores não estão captando objetos dos quais o robô deva se afastar. Note que cada sensor que capta um objeto a uma distância inferior ao *limite*, ou seja, os sensores à esquerda e na diagonal superior direita, gera uma força de reação ($\vec{F_1}$, $\vec{F_2}$, $\vec{F_3}$ e $\vec{F_4}$).

Na figura 3.3, a resultante das forças $\vec{F_1}$, $\vec{F_2}$, $\vec{F_3}$ e $\vec{F_4}$ está ilustrada na cor preta, e se refere a sua soma vetorial. A força média possui mesma direção e sentido, porém seu módulo é um quarto do módulo da força resultante.

Após o cálculo de $\vec{F}_{med}$, é feita a mudança do sistema de coordenadas. Isso porque todos os cálculos foram feitos no sistema de coordenadas do robô, porém ele se locomove no sistema de coordenadas global.

$$\vec{F}_{med,global} = M_{rot}\vec{F}_{med,local} \tag{3.26}$$

$$M_{rot} = \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix} \tag{3.27}$$

Há que se ressaltar que as equações utilizadas acima para definição da força de reação não possuem o rigor físico necessário para denominá-la força. Além disso, a discrepância em si pode não ser o valor ideal para o módulo da força. Para atender ao rigor físico e definir uma magnitude

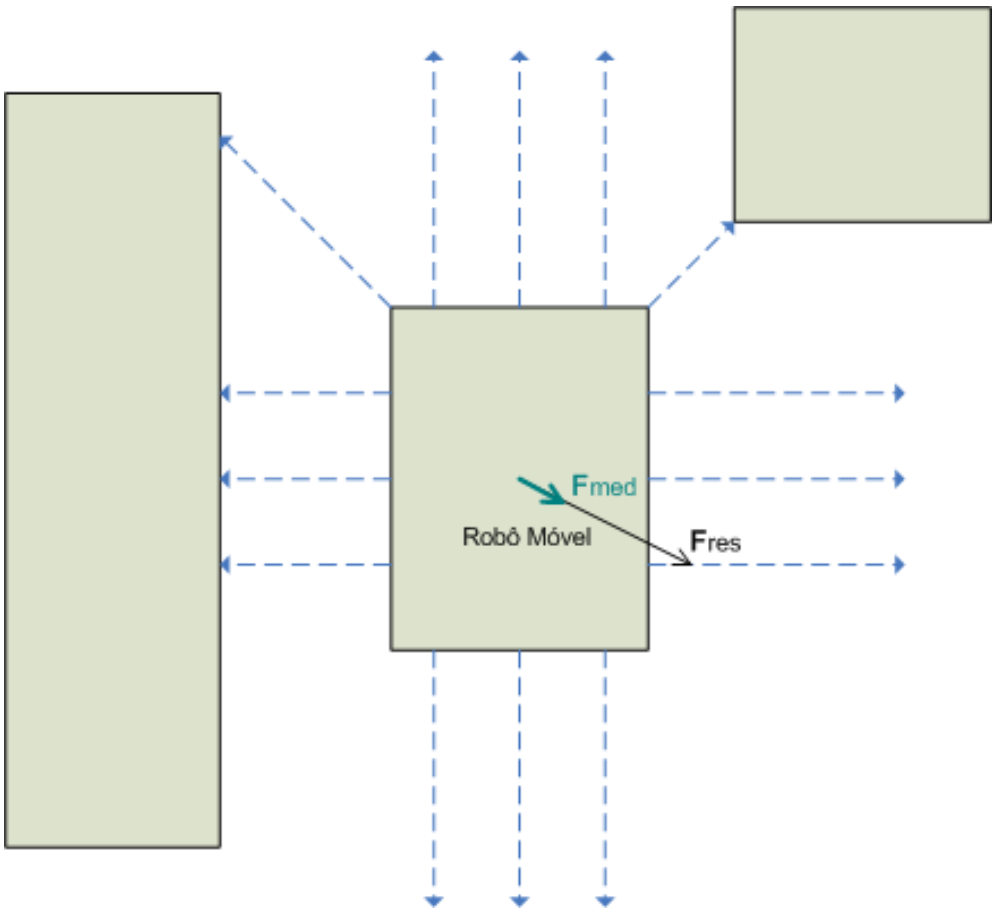


Figura 3.3: Diagrama das forças Resultante e Média

da força mais adequada ao evitamento de obstáculos, foram utilizadas as seguintes fórmulas para definir a força de repulsão:

$$\vec{F}_{rep} = K_{rep}\vec{F}_{med,global} \tag{3.28}$$

Onde K_{rep} é um escalar, e sua dimensão é definida como $[K_{rep}] = \text{Kg/s}^2$.

Além disso, deseja-se que $\vec{F}_{rep}$ seja proporcional à maior discrepância observada em cada situação (D_{max}), por exemplo $K_{rep} = a \cdot D_{max}$, onde a é um parâmetro a ser ajustado.

Definida a força e definindo a massa do robô real como $massa_{real} = 350Kg$, pode-se determinar as equações do evitamento de obstáculos baseadas nas equações horárias do movimento retilíneo uniforme. Ressalte-se que o evitamento de obstáculos não altera a orientação do robô, apenas sua posição (X_{om}, Y_{om}).

$$A_{rep} = F_{rep}/massa \tag{3.29}$$

$$V_{rep} = V_{rep,ant} + A_{rep}dt \tag{3.30}$$

$$\Delta P_{real} = V_{rep}dt + (A_{rep}dt^2)/2 \tag{3.31}$$

3.4 Combinação das Estratégias de Movimentação

3.4.1 Simulação em Matlab

Previamente à implementação da estratégia de movimentação no Omni, foi feita simulação em Matlab. A solução utilizada no robô real está descrita mais adiante.

Conforme estrutura da solução proposta, o programa também foi dividido em três partes.

O primeiro módulo de programação se refere ao seguimento da referência.

Foi necessário calcular a primeira e segunda derivadas da pose do robô virtual, tendo em vista que as fórmulas de controle utilizam a velocidade e aceleração do robô de referência, e apenas as informações da pose estão disponíveis. Essas derivadas foram obtidas conforme a seguir:

$$V_{virtual} = (P_{virtual} - P_{virtual,ant})/dt \tag{3.32}$$

$$A_{virtual} = (V_{virtual} - V_{virtual,ant})/dt \tag{3.33}$$

O segundo módulo se refere ao evitamento de obstáculos. Enquanto todos sensores detectam distâncias superiores a esse limite, o robô segue a referência de maneira inerte ao ambiente. Ao ser detectada qualquer distância inferior, o robô passa a reagir ao meio ambiente, deixando de seguir fielmente ao sinal de referência. É salutar destacar que na simulação, o robô real está equipado com 16 sensores, e que todas as medidas estão disponíveis ao mesmo tempo, sem apresentar delay, a cada iteração (50ms). Note que não há latência entre a movimentação do robô e a obtenção das medidas dos sensores.

De posse de todas as medidas, é feito o cálculo da força média de reação, conforme Equações (3.21) a (3.25). Se $\vec{F}_{med} \neq \vec{0}$, é calculada a variação da pose necessária para que o robô evite os obstáculos. Caso não haja obstáculos dentro de 1 metro, ou ainda, caso não haja força de reação, a alteração da pose devida ao evitamento de obstáculos é nula e o robô segue apenas a referência, conforme cálculo do 1^o módulo. Portanto, a estratégia de evitamento de obstáculos altera a pose que seria comandada para seguir a referência, sempre que ela não respeitar o limite de segurança definido. Para o evitamento, o valor de K_{rep} foi ajustado por calibração, sendo que o valor de $K_{rep} = 1,0$ foi o que mais se adequou ao evitamento dos obstáculos exigido ao se seguir a trajetória do robô virtual, conforme ilustrado na seção 4.3. Após esses passos, a variação na pose para evitamento de obstáculo é calculada conforme Equações (3.21) a (3.31).

No terceiro e último módulo, o programa combina o resultado dos módulos anteriores. Primeiramente ele verifica se discrepâncias foram detectadas, isto é, se houve alguma medida inferior a 1 metro. Se não houve nenhuma discrepância, a pose do robô real é igual à calculada pelo 1^o módulo (Equação (3.11)). No entanto, caso tenham sido detectados obstáculos dentro da zona de segurança, a pose do robô real é definida pela soma da pose calculada pelo 1^o módulo (Equação (3.11)) com a variação calculada pelo 2^o módulo (Equação (3.31)). Vale lembrar que o evitamento de obstáculos calcula apenas alterações de posição (X_{om}, Y_{om}) e, portanto, a variação no ângulo de orientação, devido ao evitamento de obstáculos, é nula ($\Delta\theta_{rep} = 0$).

3.4.2 Implementação no Omni

A solução discutida neste capítulo e testada em simulação foi migrada para a plataforma do robô móvel Omni.

Primeiramente, foi necessário dotar o robô de sistema destinado à percepção do ambiente. Para tal, os 14 sensores de ultrassom instalados no robô foram conectados a um microcontrolador Atmega8 da Atmel, que por sua vez foi conectado à porta serial do computador embarcado no robô Omni. Foi utilizado um protocolo de comunicação desenvolvido anteriormente em um trabalho voluntário no LARA/UnB. O padrão de comunicação com os sensores está ilustrado na Tabela 3.1, inserida na subseção 3.3.2. A comunicação foi testada, e os valores foram mostrados na tela do terminal de comunicação serial do referido computador. Os valores referentes à distância dos objetos posicionados ao redor do robô que foram informados em tela estavam compatíveis com as medidas realizadas com a trena.

Feito isso, o programa desenvolvido em Matlab foi traduzido para linguagem C e efetuados ajustes necessários ao funcionamento na plataforma do robô Omni. Foram utilizadas bibliotecas desenvolvidas em trabalhos anteriores na plataforma. O programa utilizou o sistema *multithread* de programação, sendo que uma *thread* ficou responsável por se comunicar com o *joystick* externo USB, uma outra ficou responsável pela comunicação com os sensores de ultrassom, e à terceira *thread* coube a atuação nos motores, ou melhor, o envio dos comandos ao nível inferior da plataforma, que implementa a atuação nos motores.

A primeira thread, adquire os comandos do Joystick, que são traduzidos por uma das bibliotecas do Omni como sinais de velocidade em X, Y e θ. Ressalte-se que essa é uma diferença entre a plataforma de simulação e a do robô, pois na simulação o comando de referência é uma solicitação de pose (ξ), e não de velocidades. Por meio de um mutex, essa *thread* disponibiliza os comandos de velocidade para a *thread* de atuação.

Para implementação da estratégia de evitamento de obstáculos, a *thread* de leitura dos sonares se comunica com os sensores seguindo a tabela 3.1, ou seja, enviando os caracteres reservados à solicitação de medições, e então recebe os bytes dos sensores, concatena-os, decodifica-os e determina a distância até o obstáculo que está na direção do sensor, conforme discutido na subseção 3.3.2. Há que se ressaltar outra diferença entre a plataforma de simulação e a do robô Omni. Na simulação, o sistema de evitamento de obstáculos acessava as informações de distância de todos os sensores instantaneamente, e não havia latência associada a essa aquisição de dados enquanto o robô se movimentava.

No robô Omni, a leitura dos sensores ocorre sequencialmente, conforme ordem da *daisy chain*, e a leitura de cada sensor demora aproximadamente 70ms. Enquanto a leitura dos sensores está ocorrendo, o robô está se movimentando a cada 5 ms. Sendo assim, há uma latência considerável associada ao processo de leitura dos sonares em relação à movimentação do robô. Novamente, por meio de um mutex, essa *thread* disponibiliza as leituras dos sonares para a *thread* de atuação.

A *thread* de atuação armazena o sinal de velocidades obtido pela *thread* do *joystick*, bem como armazena as informações sobre o ambiente obtidas pela *thread* dos sonares. Ela então procede ao cálculo da velocidade de repulsão do robô, conforme equação (3.30). Então é feita a combinação,

de forma similar ao que foi discutido na seção 3.4.1. Se não há discrepância, isto é, se a zona de segurança está sendo respeitada, essa *thread* envia o sinal de controle para movimentação dos motores baseando-se apenas no sinal de velocidades obtido pela *thread* de leitura do *joystick* USB. Caso haja discrepância, a *thread* soma as velocidades solicitadas pelo usuário (joystick) com as velocidades calculadas pela equação (3.30). Então o sinal resultante é enviado para efeito de controle dos motores. Vale lembrar que, assim como na simulação, o evitamento de obstáculos no robô Omni não altera a orientação do robô, ou seja, ($\Delta\theta_{rep} = 0$).

Capítulo 4

Resultados Experimentais

4.1 Introdução

Conforme comentado na seção 3.4.1, foram realizados ensaios em Matlab para averiguar o desempenho da lei de controle e para averiguar a combinação das duas estratégias de movimentação.

Primeiramente, os ensaios foram realizados para se ajustar o parâmetro λ da lei de controle apresentada na seção 3.2.1. Foram feitos diversos ensaios com valores diferentes da referida variável, e então selecionou-se o valor ótimo para λ que otimiza o seguimento de trajetória, por meio da análise dos erros rms das variáveis da pose (ξ) do robô real.

Para essa estratégia de movimentação, a referência é um robô virtual que segue a trajetória previamente definida. Ele desenvolve velocidade linear de 0.5 m/s e sua pose no instante atual e no instante anterior são acessíveis pelo sistema de controle. O robô real tem sua pose calculada conforme discutido na seção 3.2.1. A simulação utiliza um vetor para representar o tempo, que vai de 0 a 100 segundos, e apresenta resolução $dt = 50ms$. A velocidade do robô é simulada utilizando esse vetor. O valor de λ (3.6) foi ajustado por calibração, e verificou-se que a melhor solução é obtida em $\lambda = 0,87$, conforme gráficos apresentados e discutidos neste capítulo. Esse mesmo valor de λ foi utilizado no seguimento da referência no que tange ao ângulo de orientação do robô, utilizando fórmulas análogas às apresentadas na seção 3.2.1.

A cada iteração, ou a cada 50ms do vetor tempo, a pose do robô real é calculada, e a cada 1 segundo do vetor tempo, são desenhados os robôs real e virtual, e os marcadores que sinalizam suas posições (vide Figura 4.1).

O ambiente de simulação possui paredes e estruturas, que são interpretadas pelos sensores do robô real como obstáculos. O robô virtual passa próximo desses obstáculos e em determinados trechos de sua trajetória, ele não está equidistante dos obstáculos. Além disso, foi definido o valor de 1 metro como limite de segurança, que não é respeitado pelo robô virtual. Diante de tais situações, o robô real deve reagir ao ambiente, e não seguir a referência integralmente. Ressalte-se que o limite de segurança é um valor de gatilho para a atuação do sistema de evitamento de obstáculos.

Fixado o valor de λ, foram realizados outros ensaios para diferentes valores de K_{rep}. Esse

ajuste se refere ao evitamento de obstáculos discutido na seção 3.3.

Para demonstrar o resultado dos ensaios, foram incluídos gráficos e imagens da trajetória dos robôs real e virtual.

Para a plataforma do robô Omni, não foram gerados gráficos ou imagens destinados à análise. Infelizmente, a implementação da solução desenvolvida na referida plataforma se tornou um problema de programação.

Apesar de as medições ocorrerem, e serem recebidos valores plausíveis, conforme mostrado no terminal de comunicação serial do robô, essas medições não puderam ser plotadas na tela do robô, quando o programa principal estava sendo executado.

Além disso, acredita-se que o programa não foi capaz de enviar corretamente os comandos aos motores do robô Omni, resultando em comportamentos inesperados, tais como reação aos obstáculos, porém quando restabelecia a zona de segurança, ele não parava e continuava movimentando-se opostamente aos obstáculos. Em outra versão do programa o robô apresentou acelerações e desacelerações repentinas, em uma outra versão apresentou continuamente movimentos nas juntas do robô, mesmo quando não estava sendo enviados comandos pelo joystick USB, ou seja, quando não havia duas estratégias de movimentação a serem realizadas.

4.2 Avaliação da Lei de Controle para Seguimento de Trajetória

A seguir são mostradas as figuras da trajetória dos robôs virtual e do real para quatro valores distintos de λ. Essas imagens foram geradas em simulação e estavam utilizando apenas o módulo de seguimento de trajetória. Nesses ensaios configurou-se a simulação para que a posição inicial do robô real coincidisse com o início da trajetória ($t = 0.00s$), e a posição inicial do robô virtual coincidisse com o próximo ponto da trajetória ($t = 0.05s$).

A Figura 4.1 se refere às trajetórias dos robôs, e os marcadores circulares se referem à posição do robô virtual, enquanto que os quadrados se referem à do real. Essa figura mostra que a lei de controle permite o seguimento da trajetória de maneira razoável. Não é possível visualizar os dois robôs (real e de referência), porém os marcadores ao longo da trajetória evidenciam que o robô real seguiu razoavelmente o robô de referência. Note que há apenas alguns círculos que não estão inscritos nos quadrados, e ainda assim, a discrepância é muito pequena.

Para uma análise mais apurada da lei de controle, optou-se por avaliar o erro rms entre a referência e o sinal real de cada variável da pose ξ, ou seja $(x, y, \theta\ )$ para valores distintos do parâmetro λ, auto-valor da lei de controle. As fórmulas 4.1 e 4.2 foram utilizadas para o cálculo do erro rms de cada uma das referidas variáveis.

$$Erro_{rms} = \sqrt{\overline{erro^2}} \tag{4.1}$$

Onde,

$$erro = \text{Referência} - \text{Sinal real} \tag{4.2}$$

Para se obter gráficos destinados à análise do seguimento de trajetória, o programa da simulação foi configurado para rodar um ensaio completo, isto é, aproximadamente três voltas na trajetória, para cada valor de λ. Durante o ensaio, o programa armazena a pose real, a pose virtual, e o tempo associado a esses valores, para posterior geração de gráficos. Após o ensaio completo, essa versão do programa incrementa o valor de lambda e roda outro ensaio completo. Esses procedimentos são repetidos até atingir o valor de λ que foi definido como limite. Foram gerados os gráficos 4.2 a 4.9 para se analisar a influência de λ no desempenho da estratégia de movimentação referente ao seguimento de trajetória. Diante desses gráficos, definiu-se um valor para o parâmetro em questão, cujos gráficos foram incluídos ao final desta seção (gráficos das Figuras 4.11 a 4.16).

O cálculo de erro rms foi feito sobre os vetores de referência e vetores de sinal real referentes às variáveis da pose (x, y, θ), vetores gerados em cada ensaio (cada valor de λ). Os gráficos contidos na Figura 4.2 foram gerados na simulação para $\lambda = 0$ até $\lambda = 100$, variando-se λ em 0,05 entre cada ensaio. Porém, os valores para o erro rms das variáveis X e Y passaram a ser indeterminados após 800 iterações, isto é, para valores de $\lambda > 40$. Sendo assim, os resultados dessa simulação foram gerados e avaliados apenas com os dados obtidos até as 800 primeiras iterações.

Pelos gráficos da Figura 4.2, há um interesse maior na região de $\lambda = 15$ até $\lambda = 25$, para otimizar as variáveis X e Y. Os gráficos contidos na Figura 4.3 foram gerados na simulação para essa região de λ, incrementando-se seu valor com 0.01 entre cada ensaio.

A Figura 4.4 foi incluída para verificar o ponto de mínimo do erro rms de Y na região investigada. Nesse gráfico, evidencia-se que o valor de $\lambda = 20,0$ otimiza o desempenho da lei de controle com respeito ao referido erro rms, obtendo-se o valor de 0,3026, conforme legenda incluída na figura.

No entanto, também pelo gráfico *d)* da Figura 4.2, há um interesse maior na região de $\lambda = 0$ até $\lambda = 3$, para otimizar a soma normalizada do erro rms de todas variáveis da pose. Para normalização das variáveis, cada valor de erro rms foi divido pela variação máxima de sua respectiva variável, ou seja, $\Delta X_{max} = 5 - (-1) = 6$, $\Delta Y_{max} = 5 - 0 = 5$, e $\Delta\theta_{max} = \frac{3\pi}{2} - (-\frac{\pi}{2}) = 2\pi$ Os gráficos das Figuras 4.5 a 4.8 foram gerados para essa região de λ, incrementando-se seu valor com 0,01 entre cada ensaio.

A Figura 4.9 foi incluída para verificar o ponto de mínimo da soma normalizada dos erros rms das variáveis x, y, e θ na região investigada. Nesse gráfico, evidencia-se que o valor de $\lambda = 0,87$ otimiza o desempenho da lei de controle com respeito à referida soma, obtendo-se o valor de 0,03705, conforme legenda incluída na figura.

A partir desses resultados, tem-se que $\lambda = 20$ otimiza o desempenho no posicionamento com respeito à variável Y. A Figura 4.1 foi gerada com esse valor de λ, e incluída no início dessa Seção. Tem-se ainda que $\lambda = 0,87$ otimiza o desempenho com respeito à soma normalizada dos erros rms. A figura 4.10 foi gerada com esse valor de λ.

Os gráficos contidos nas Figuras 4.11 e 4.12 ilustram o desempenho da lei de controle com respeito à variável x quando $\lambda = 0,87$, valor que otimiza a soma normalizada dos erros rms.

Os gráficos contidos nas Figuras 4.13 e 4.14 ilustram o desempenho da lei de controle com respeito à variável y quando $\lambda = 0,87$, valor que otimiza a soma normalizada dos erros rms.

Os gráficos das Figuras 4.15 e 4.16 ilustram o desempenho da lei de controle com respeito à variável θ quando $\lambda = 0,87$, valor que otimiza a soma normalizada dos erros rms.

Há que se ressaltar que a escolha do valor de λ depende da aplicação a que se destina a solução desenvolvida nesse trabalho. Não se pretende definir um valor ótimo de maneira absoluta, mas sim mostrar diferentes configurações que podem ser utilizadas. A escolha da mais adequada cabe ao responsável pelo projeto que utilizará aquela que melhor supre as necessidades do projeto.

4.3 Avaliação da Combinação das Estratégias de Movimentação

A seguir são mostradas as figuras da trajetória do robô virtual e do real para diferentes valores da constante de repulsão K_{rep}, com λ fixo em $0,87$. Essas trajetórias foram geradas em simulação e contemplam a solução completa, conforme discutido na seção 3.3.3. Os ensaios duraram tempo suficiente para uma volta completa no cenário da simulação. Os marcadores circulares se referem à posição do robô virtual, enquanto que os quadrados se referem à do real. Os traços na cor preta e marcadores "+" se referem à força média de repulsão calculada.

As imagens da trajetória contidas na Figura 4.17 se referem à combinação das estratégias de movimentação de seguimento de trajetória e evitamento de obstáculos. Essas imagens foram geradas definindo-se valores diferentes de K_{rep}. Em todos os casos configurou-se a posição inicial do robô real no início da trajetória ($t = 0.00s$), a posição inicial do robô virtual um passo a frente na trajetória ($t = 0.05s$), e o limite de segurança, ou valor de gatilho para atuação do evitamento, como $limite = 1,00$ metro.

Com o aumento de K_{rep}, o robô real se afasta mais rapidamente dos obstáculos. No caso do ambiente de simulação utilizado, isso faz com que o robô se afaste mais rapidamente da trajetória. Nas trajetórias *a)* e *d)* da Figura 4.17 fica mais evidente o efeito de K_{rep} no movimento final do robô.

Note que na Figura 4.17 **c)** o robô realizou movimentos mais bruscos para priorizar sua integridade do que na Figura 4.17 **d)**. O comprimento dos traços na cor verde permite se ter noção do quanto o robô se afastou da referência.

Na última configuração, ou seja $K_{rep} = 2,5D_{max}$, o robô se afasta ainda mais da referência. Os marcadores quadrados apareceram mais distanciados entre si, na direção ortogonal aos obstáculos, indicando manobras de evasão mais bruscas do que as anteriores.

Os valores $\lambda = 0,87$ e $K_{rep} = 1,0D_{max}$ são os que proporcionam o melhor desempenho para o robô, de acordo com o que foi visto até então. A Figura 4.17 **b)** se refere a esses valores. Com esse ajuste dos parâmetros, o robô consegue se desviar dos obstáculos, garantindo sua integridade, porém ele se afasta ligeiramente da referência, realizando movimentos mais suaves.

Fixando $\lambda = 0,87$ e $K_{rep} = 1,0D_{max}$, realizaram-se outros três ensaios. Nesses ensaios, o objeto de avaliação foi o limite de segurança (gatilho), sendo testados os valores de $0,80m$ (Figura 4.18 *a)*), $1,20m$ (Figura 4.18 *b)*), e $1,50m$ (Figura 4.18 *c)*).

Para o limite de segurança igual a $0,80m$ (Figura 4.18 *a)*), o robô se afastou menos da

referência, como esperado, também em consequência dessa escolha, seus movimentos foram suaves. No entanto, o robô esteve mais vulnerável a colisões, devido a maior proximidade dos obstáculos.

Como esperado, limites menores ensejam maior proximidade à referência, e maiores fazem o percurso se afastar mais da referência. Porém para o caso do limite de segurança de $1,50$ metro ocorreram duas colisões. Esse fato não era esperado, pois quanto maior o limite de segurança, mais a solução deveria priorizar o evitamento de obstáculos, em detrimento do seguimento de trajetória. Pode-se, no entanto, afirmar que um limite de segurança maior enseja maior discrepância medida pelos sensores em relação aos obstáculos, traduzindo-se em um maior ganho para o módulo de evitamento. Acredita-se que esse alto ganho tenha resultado em maior instabilidade para o sistema, resultando em situações indesejáveis, tais como pouco seguimento da trajetória de referência e colisões com obstáculos.

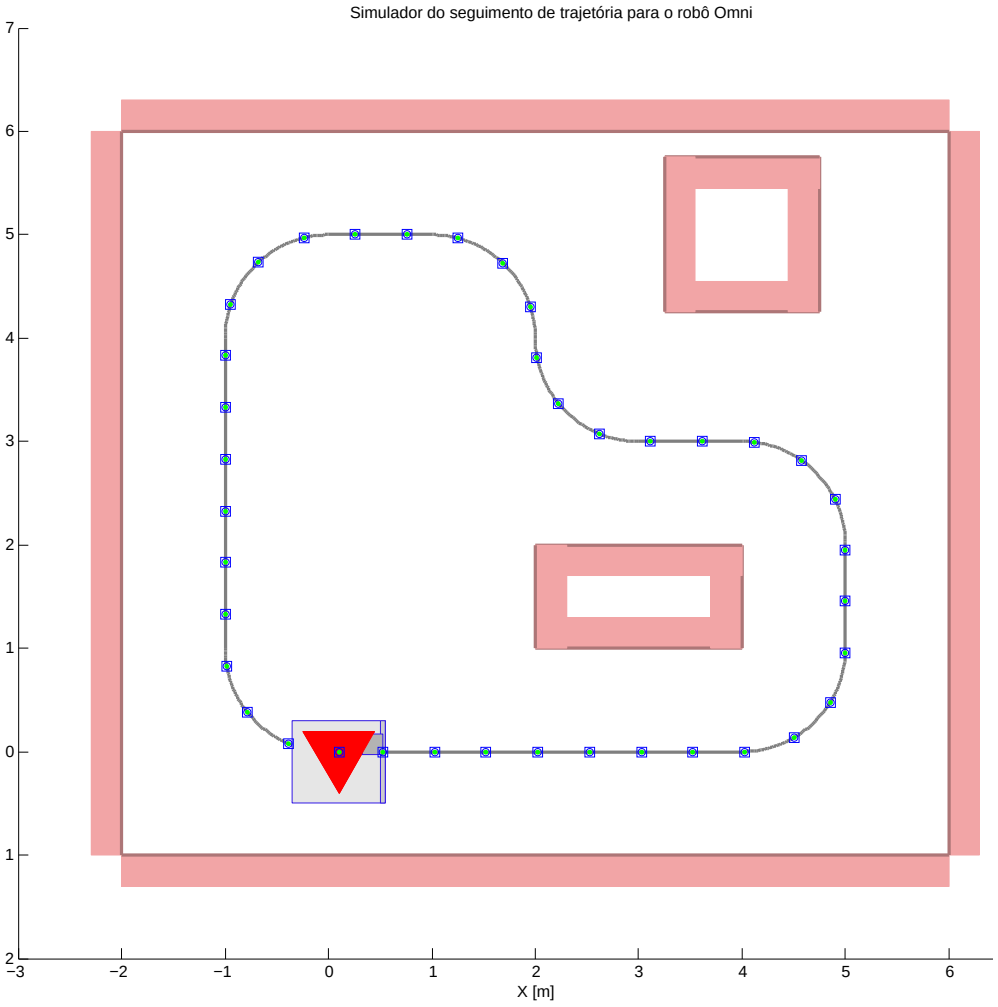
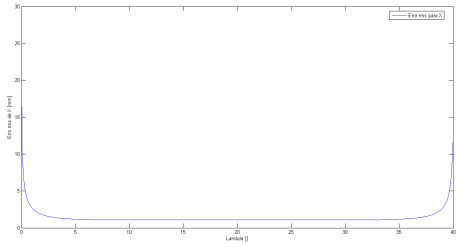
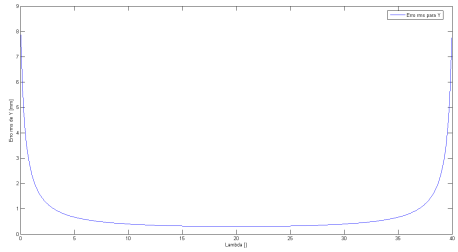


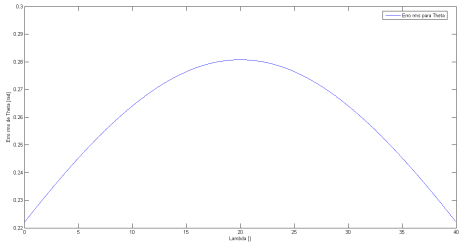
Figura 4.1: Trajetória dos robôs real e virtual para $\lambda = 20,0$



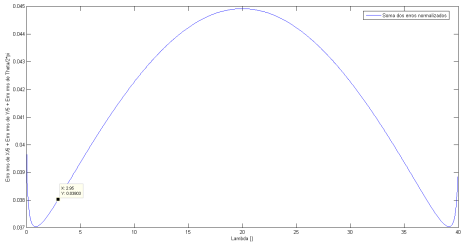
(a) Erro rms de X



(b) Erro rms de Y

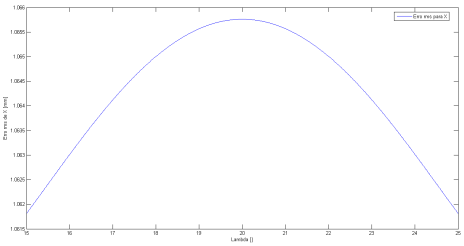


(c) Erro rms de Theta

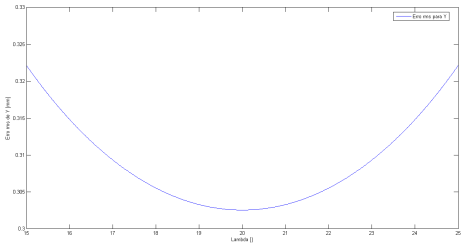


(d) Soma das normalizações dos erros rms

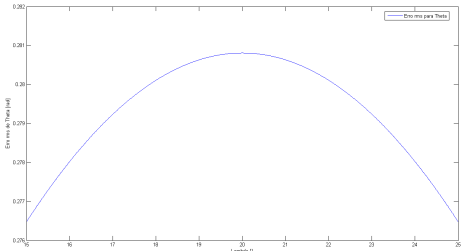
Figura 4.2: Ensaio de $\lambda = 0,0$ até $\lambda = 100,0$ (incremento de 0.05)



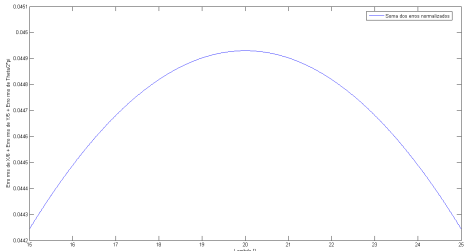
(a) Erro rms de X



(b) Erro rms de Y



(c) Erro rms de Theta



(d) Soma das normalizações dos erros rms

Figura 4.3: Ensaio de $\lambda = 15,0$ até $\lambda = 25,0$ (incremento de 0.01)

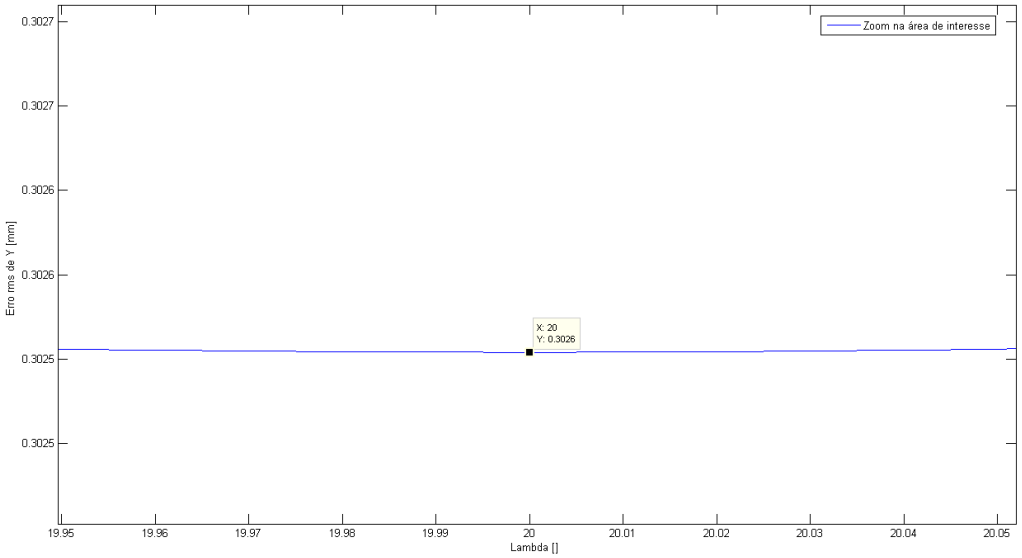


Figura 4.4: Erro rms de Y (zoom no ponto de mínimo)

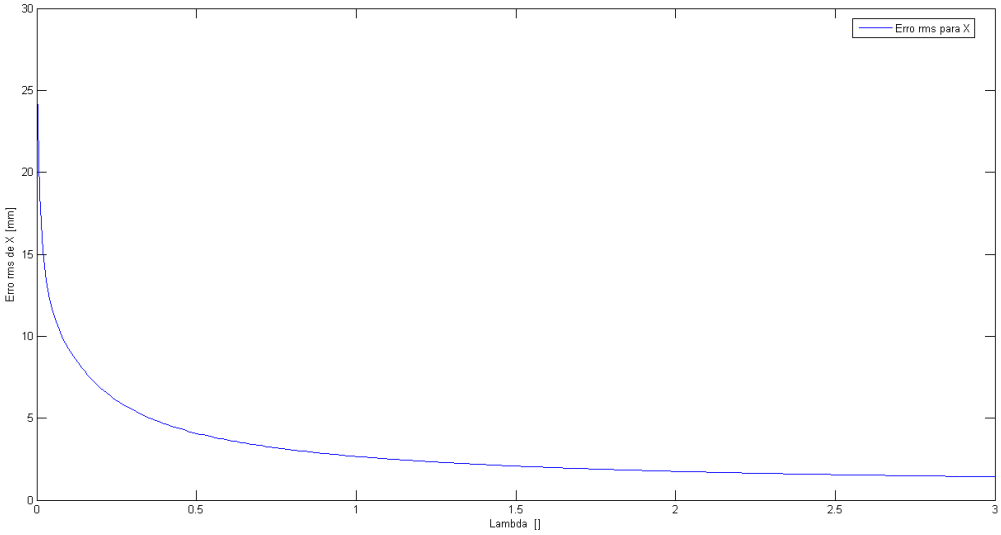


Figura 4.5: Erro rms de X ($\lambda = [0; 3]$, incremento de 0,01)

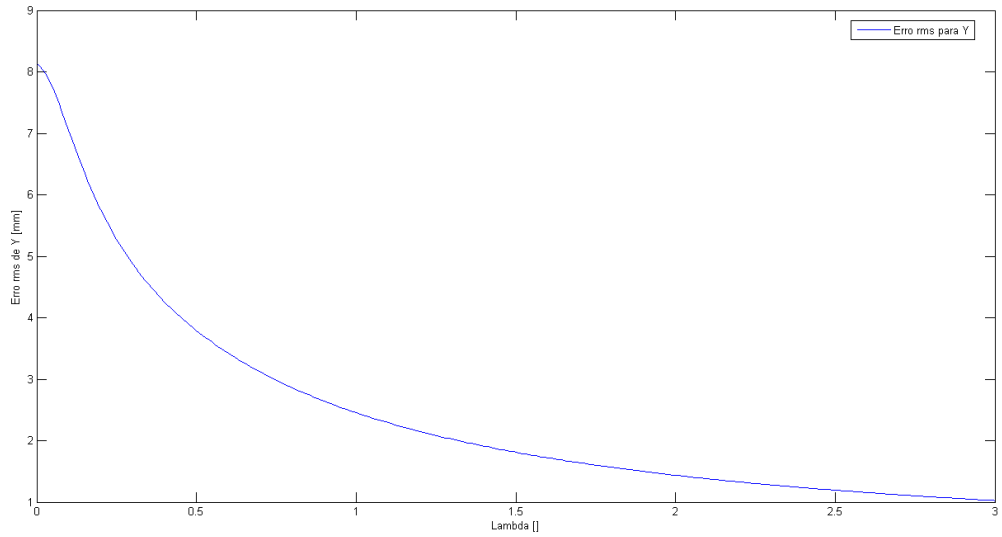


Figura 4.6: Erro rms de Y ($\lambda = [0; 3]$, incremento de 0,01)

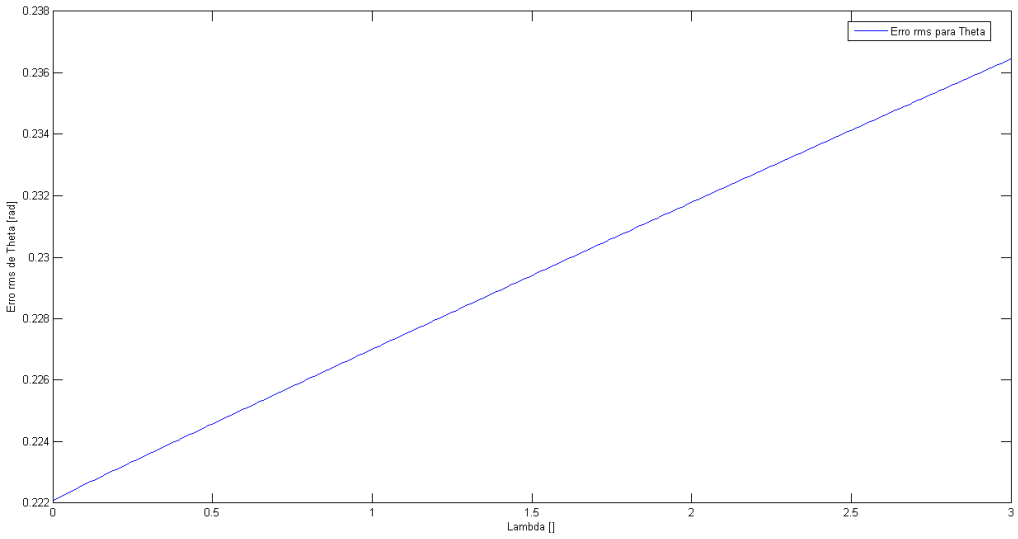


Figura 4.7: Erro rms de Theta ($\lambda = [0; 3]$, incremento de 0,01)

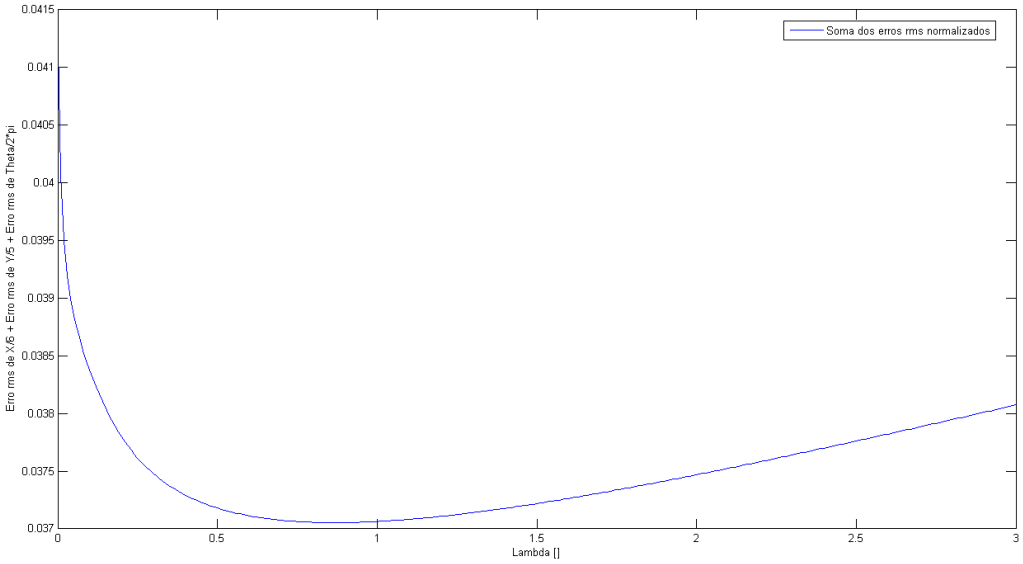


Figura 4.8: Soma das normalizações dos erros rms ($\lambda = [0; 3]$, incremento de 0,01)

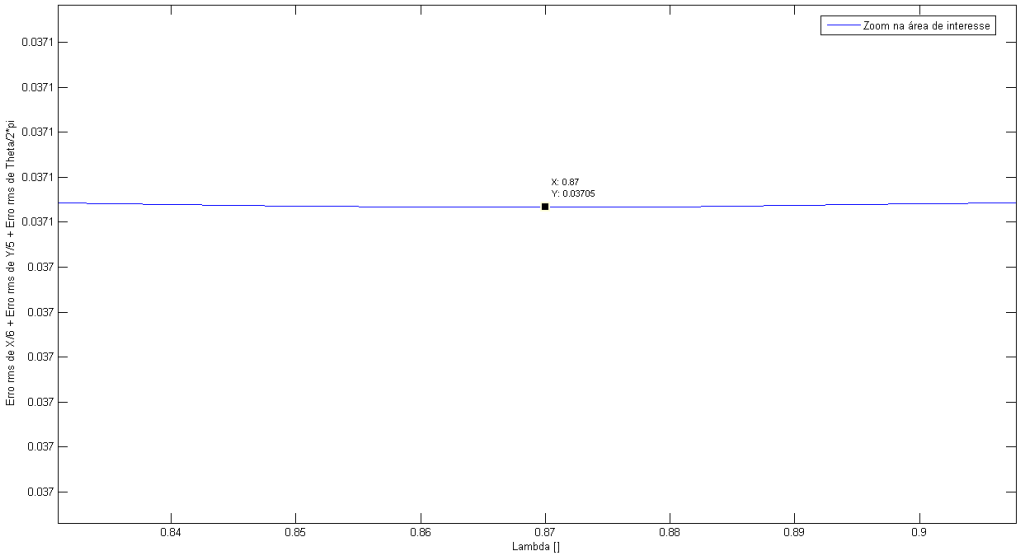


Figura 4.9: Soma das normalizações dos erros rms (zoom no ponto de mínimo)

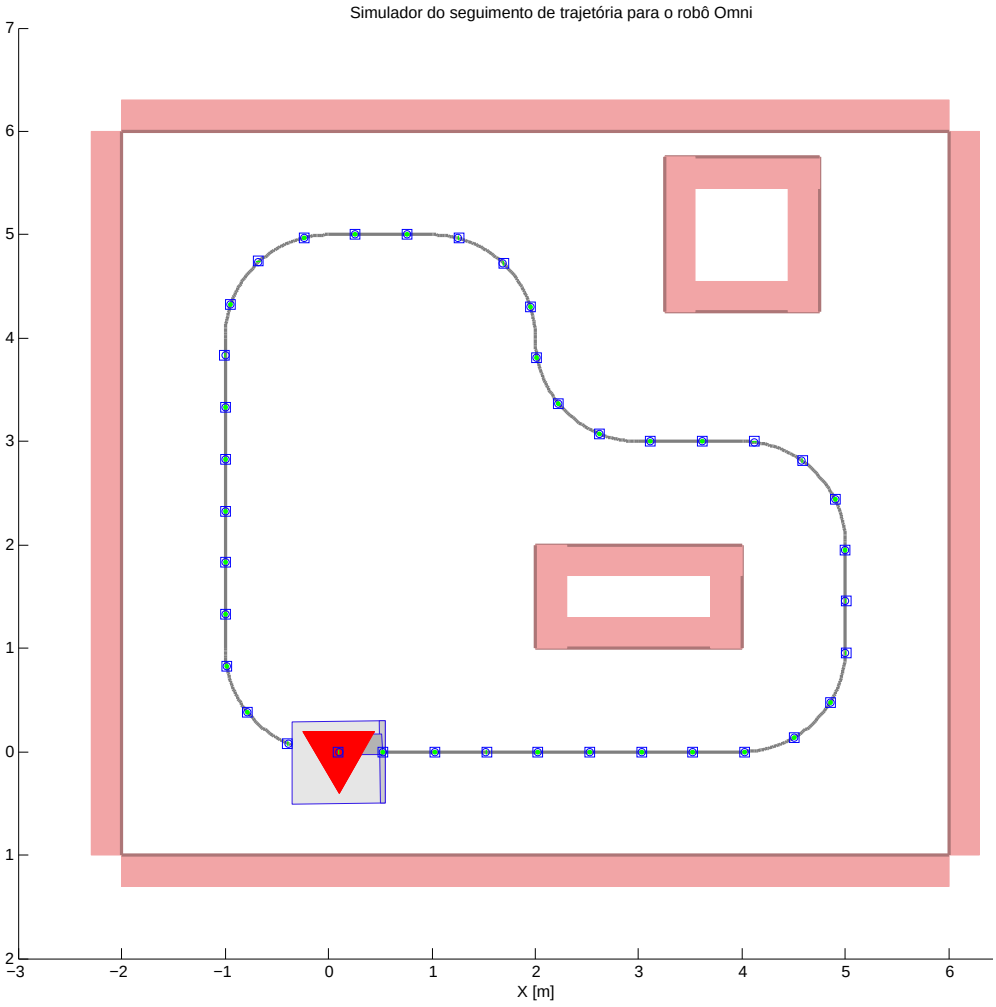


Figura 4.10: Trajetória dos robôs real e virtual para $\lambda = 0,87$

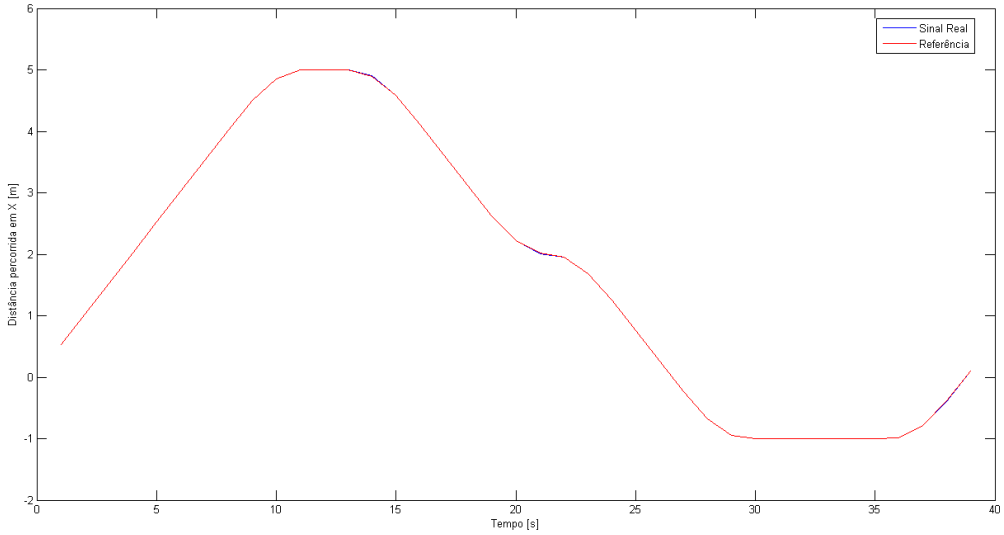


Figura 4.11: Valor de X dos robôs real e de referência ($\lambda = 0,87$)

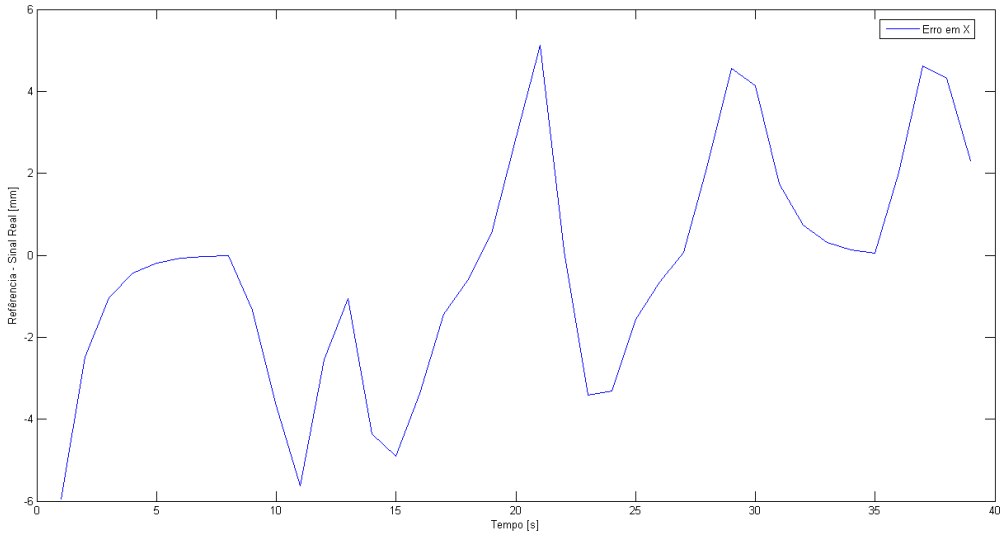


Figura 4.12: Erro em X ($\lambda = 0,87$)

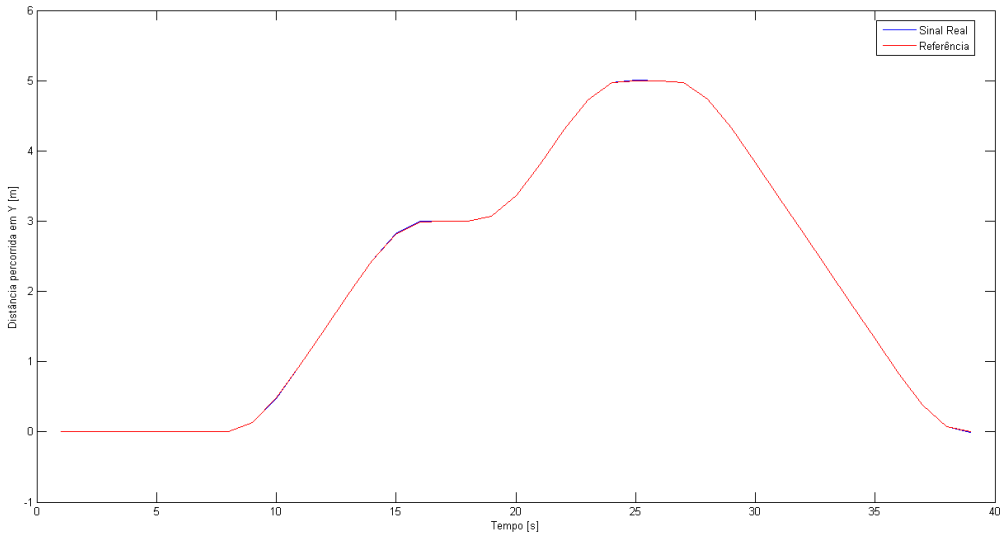


Figura 4.13: Valor de Y dos robôs real e de referência ($\lambda = 0,87$)

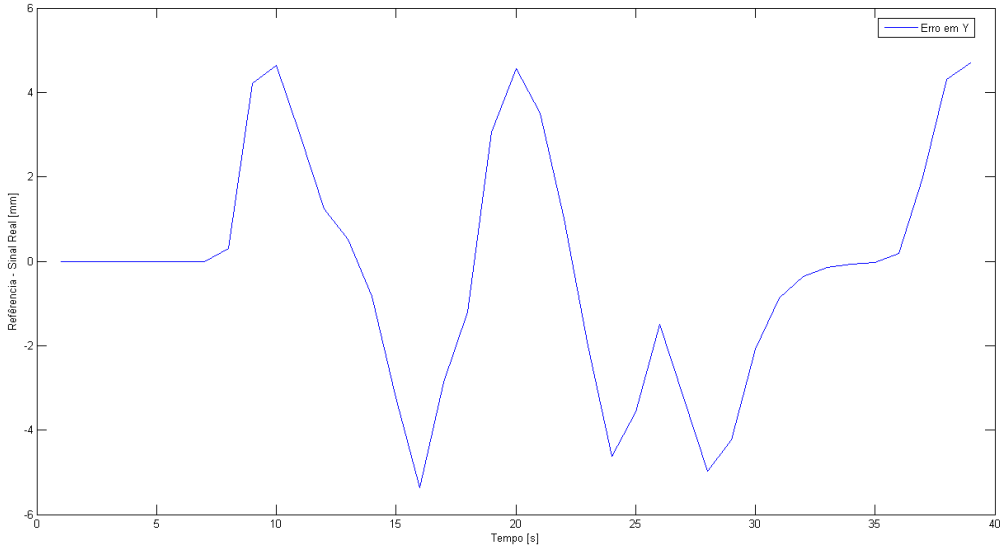


Figura 4.14: Erro em Y ($\lambda = 0,87$)

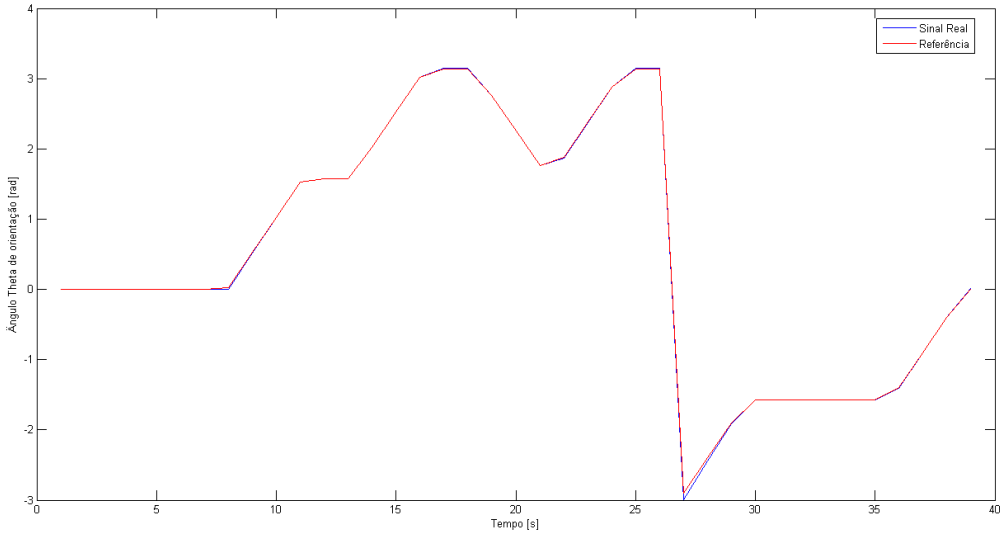


Figura 4.15: Valor de Theta dos robôs real e de referência ($\lambda = 0,87$)

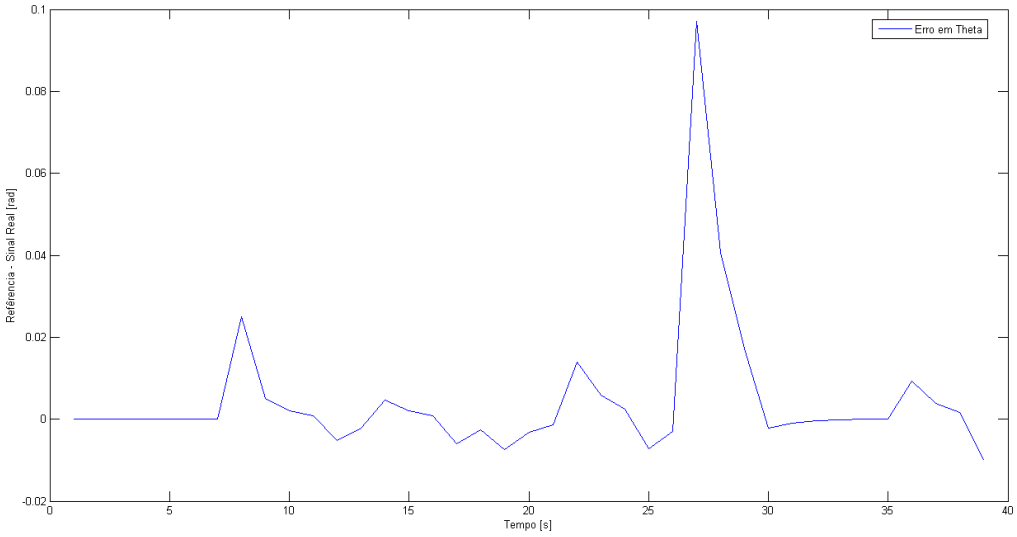
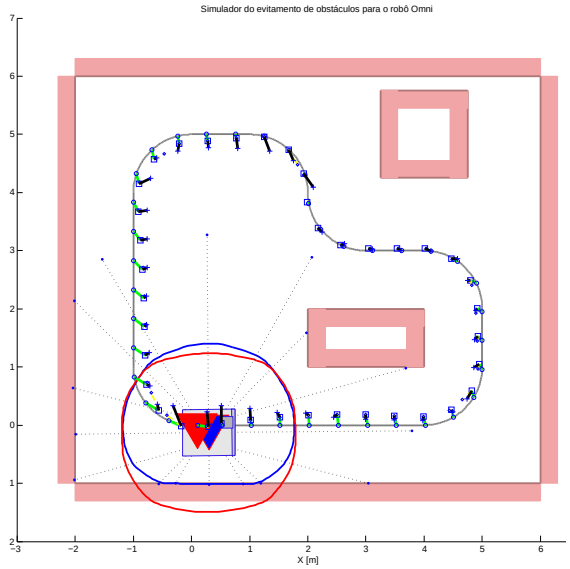
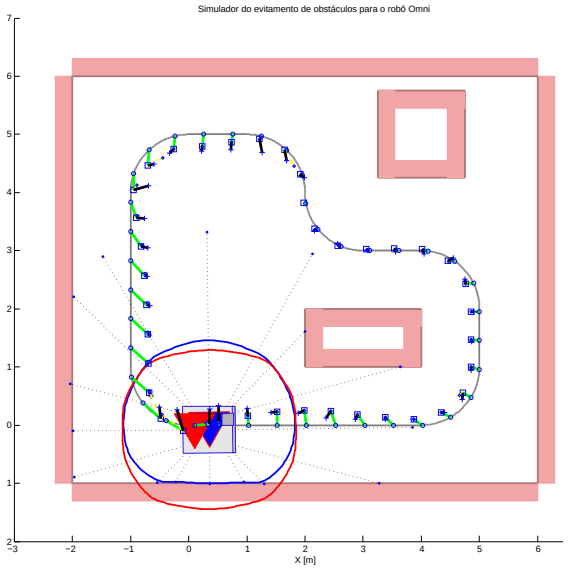


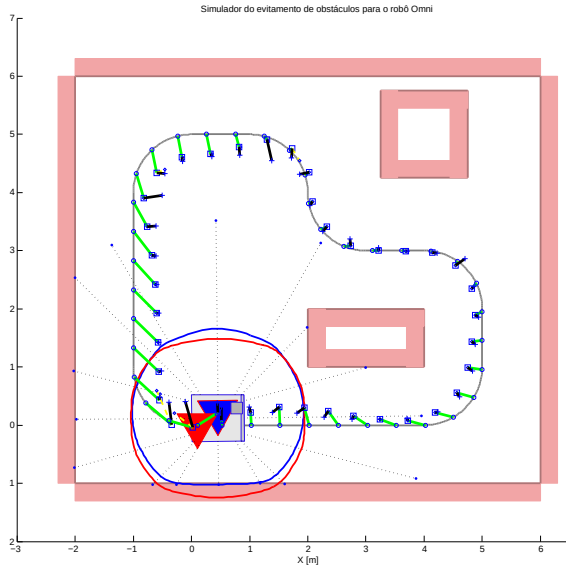
Figura 4.16: Erro em Theta ($\lambda = 0,87$)



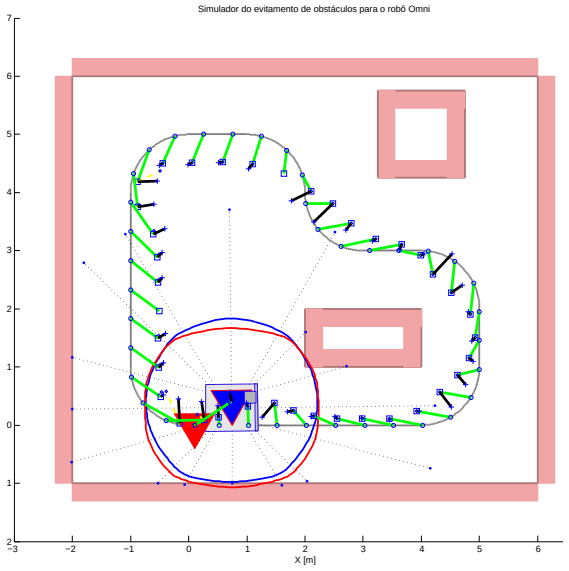
(a) $\lambda = 0,87$ e $K_{rep} = 0,5 \cdot Dmax$



(b) $\lambda = 0,87$ e $K_{rep} = 1,0 \cdot Dmax$

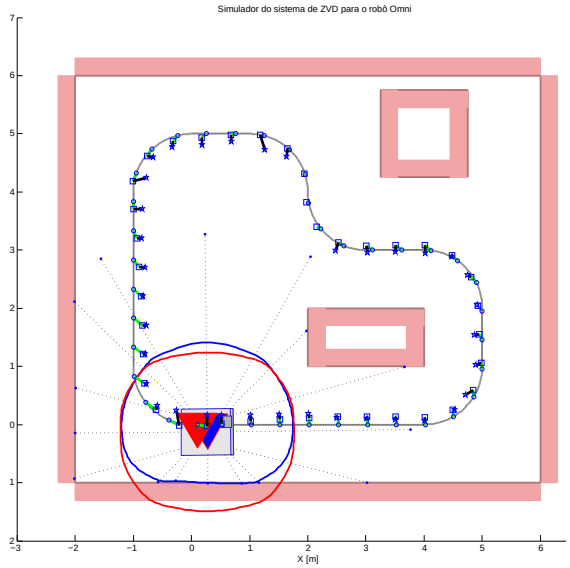


(c) $\lambda = 0,87$ e $K_{rep} = 1,5 \cdot Dmax$

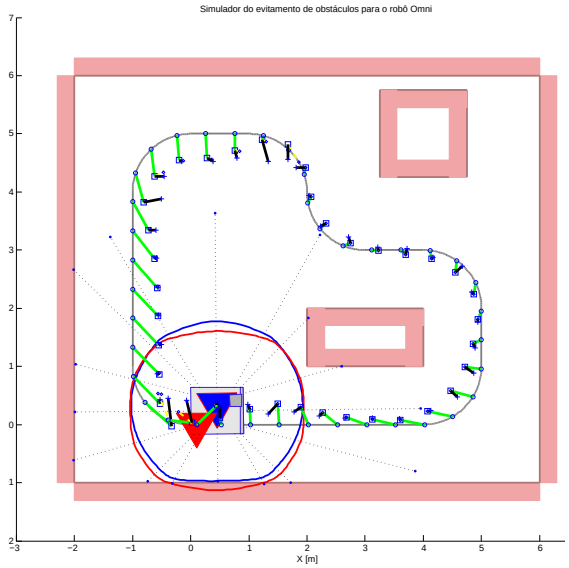


(d) $\lambda = 0,87$ e $K_{rep} = 2,5 \cdot Dmax$

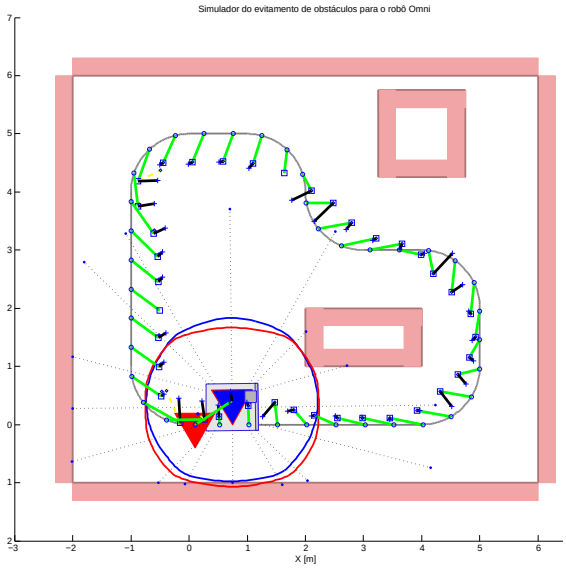
Figura 4.17: Ensaios de evitamento de obstáculos para diferentes valores de K_{rep}



(a) Limite de segurança de $0,80m$



(b) Limite de segurança de $1,20m$



(c) Limite de segurança de $1,50m$

Figura 4.18: Ensaios de evitamento de obstáculos para diferentes limites de segurança

Capítulo 5

Conclusões

Neste trabalho foram apresentadas aplicações motivadoras para o estudo e desenvolvimento de soluções para evitamento de obstáculos em robótica móvel. Baseando-se em estudos relativos a B-Splines, baseando-se em trabalhos sobre Zona Virtual Deformável, e ainda apoiado no estudo sobre teoria de controle, foi proposta uma solução que permitisse que um robô móvel seguisse um sinal de referência, mas que garantisse sua integridade, desviando da referência sempre que os comandos do usuário o levassem a uma situação de risco à sua integridade.

Foi constatado que dois parâmetros influenciavam no desempenho da solução testada em simulação, quais sejam, o valor de λ, referente ao controle de trajetória, e o valor de K_{rep}, referente ao evitamento de obstáculos. Para ajuste ótimo dos parâmetros foram feitos diversos ensaios em simulação. Diante dos ensaios, foram gerados gráficos e imagens para análise de resultados.

Após simulação, a solução foi migrada para a plataforma do robô Omni. Os sensores e respectiva comunicação via microcontrolador Atmega 8 foram configurados, obtendo-se dados referentes à percepção do ambiente em que o robô móvel Omni estava inserido. Devido às diferenças entre a plataforma de simulação e a do robô Omni, não foi possível implementar a solução desenvolvida. O trabalho se tornou um problema de programação ao invés de engenharia.

No entanto, diante dos resultados da simulação, conclui-se que é possível utilizar a lei de controle apresentada na seção 3.2.1. O algoritmo desenvolvido preencheu os requisitos estabelecidos, permitindo que o robô real do simulador (Matlab) seguisse o sinal de referência com erros na ordem de 6 mm (posicionamento) e na ordem de 100 $mrad$ ou $5,73$ (orientação).

Conclui-se também que é possível fazer um evitamento de obstáculos de acordo com o que foi discutido na seção 3.3. O robô real na simulação foi capaz de evitar obstáculos de uma maneira suave, ao mesmo tempo que desenvolveu uma trajetória próxima da referência.

Sendo assim, no âmbito da simulação, os objetivos foram alcançados, tendo o trabalho apresentado resultados satisfatórios.

Devido à generalidade do algoritmo que implementou a estratégia de movimentação, conclui-se que o presente trabalho pode ser expandido para outras aplicações onde há a necessidade do evitamento de obstáculos ser auxiliado ou realizado por meio de métodos computacionais, uma das motivações deste trabalho.

Durante o desenvolvimento do trabalho, surgiram ideias para trabalhos futuros. Trata-se de sugestões para aperfeiçoar o evitamento de obstáculos desenvolvido.

Como proposta de trabalho futuro, tem-se a investigação dos problemas de implementação, e implementar plenamente o algorítimo desenvolvido.

Feito isso, sugere-se a utilização de dois microcontroladores da série ATXMega. Cada um deles se comunicaria com sete sensores, e enviaria os dados para o computador. Isso faria a latência na aquisição de dados do ambiente ser reduzida, pois cada microcontrolador comunicaria-se quase simultaneamente com os sete sensores aos quais está conectado, e enviaria esses dados instantaneamente ao computador. Dessa forma, a latência par apercepção de todos os *landmarks* seria reduzida a um patamar inferior a $200ms$.

Propõe-se também a emulação da comunicação com a porta serial de forma a permitir a operação do robô por meio de um computador remoto.

Propõe-se ainda a utilização de 14 EKF em paralelo, cada um responsável por um par (x, y) de *landmarks*, que possivelmente apresentará maior eficiência computacional do que um filtro contendo 28 variáveis de estado.

Outra proposta é de utilizar o Ladar instalado no robô Omni como ferramenta para percepção do ambiente no qual ele está inserido. O Ladar permite ajustar a resolução do *encoder*, resultando em mais landmarks observados. Por meio do ajuste da resolução, ele é capaz de disponibilizar uma quantidade de informações sobre o ambiente muito maior do que os sensores de ultrassom. Além disso, o Ladar possui uma latência de comunicação inferior à dos sensores de ultrassom. Deve-se observar que o Ladar está instalado no robô Omni alguns centímetros mais alto do que o cinturão de ultrassom. Isso pode resultar em uma percepção diferente do ambiente, dependendo da altura dos *landmarks*.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Jet Propulsion Laboratory. *NASA Facts*. 1997. Disponível em: <http://www.jpl.nasa.gov/news/fact_sheets/mpf.pdf>.

[2] BLAKE, A.; ISARD, M. *Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion*. [S.l.]: Springer, 2000. 352 p. ISBN 3540762175.

[3] GREWAL, M. S.; ANDREWS, A. P. *Kalman Filtering: Theory and Practice Using MATLAB*. [S.l.]: Wiley-IEEE Press, 2008. 592 p. ISBN 0470173661.

[4] LAPIERRE, L.; ZAPATA, R.; LEPINAY, P. Combined Path-following and Obstacle Avoidance Control of a Wheeled Robot. *The International Journal of Robotics Research*, v. 26, n. 4, p. 361–375, abr. 2007. ISSN 0278-3649.

[5] MADDEN, J. D. Mobile robots: motor challenges and materials solutions. *Science (New York, N.Y.)*, v. 318, n. 5853, p. 271–286, nov. 2007. ISSN 1095-9203.

[6] GALICKI, M. Collision-free control of an omni-directional vehicle. *Robotics and Autonomous Systems*, v. 57, n. 9, p. 889–900, 2009. ISSN 0921-8890.

[7] MARTINEZ-GOMEZ, L.; FRAICHARD, T. *Collision avoidance in dynamic environments: An ICS-based solution and its comparative evaluation*. [S.l.]: IEEE, 2009. 100–105 p. ISBN 978-1-4244-2788-8.

[8] SIMON, D. *Optimal state estimation: Kalman, H [infinity] and nonlinear approaches*. [S.l.]: John Wiley and Sons, 2006. 526 p. ISBN 0471708585.

ANEXOS

I. DIAGRAMAS ESQUEMÁTICOS

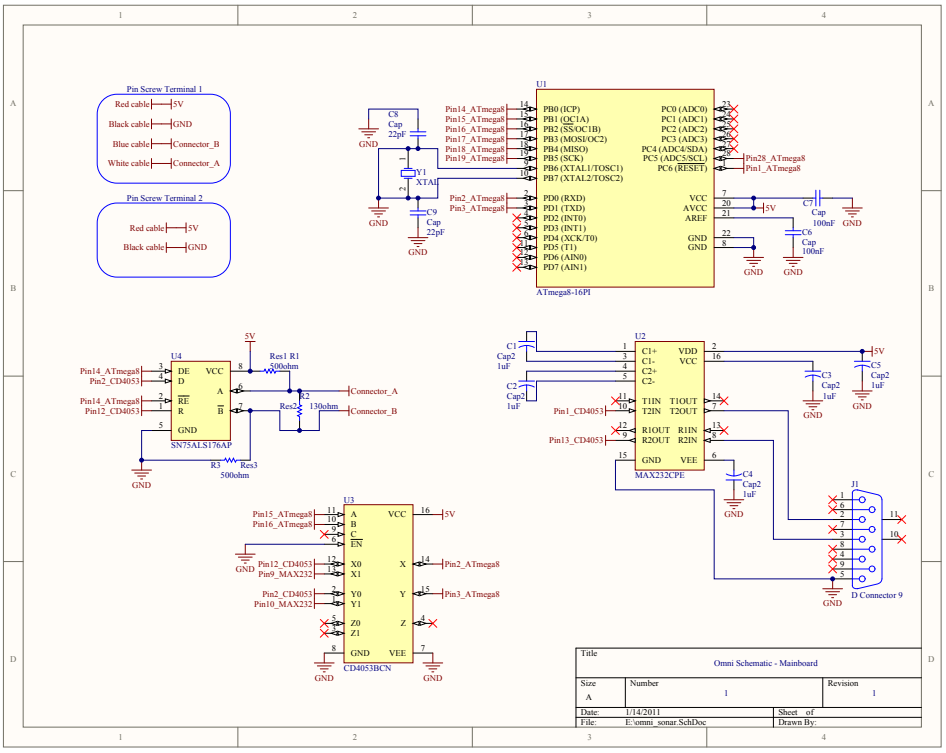


Figura I.1: Esquemático do sistema de aquisição de medidas dos sonares

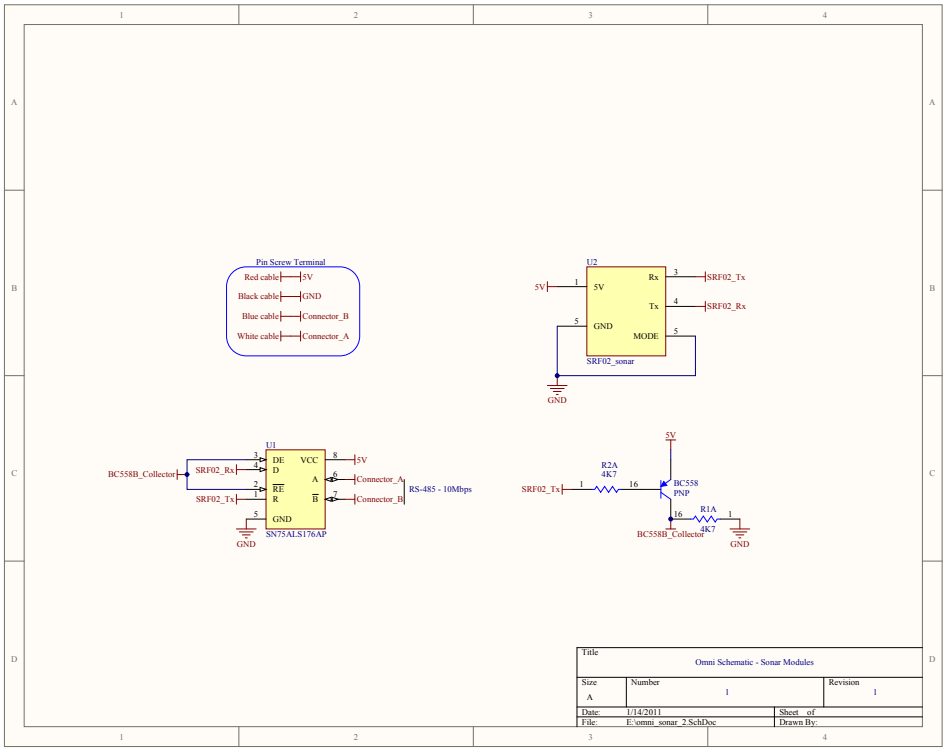


Figura I.2: Esquemático dos módulos sonares

II. DESCRIÇÃO DO CONTEÚDO DO CD

Tabela II.1: Conteúdo do CD

Nome da pasta ou arquivo	**Descrição do conteúdo**
Pasta Simulador	Contém os arquivos utilizados para simular o experimento no Matlab.
simulador.m	Programa principal
Protocolo_sensores_ultrassom.txt	Protocolo de comunicação dos sensores de ultrassom gravado no microcontrolador
Pasta Teleoperation	Contém os arquivos em linguagem C executados na plataforma do robô Omni
ApplicationExampleTeleoperation.dsw	Projeto no formato para o Visual C++ englobando os arquivos utilizados na plataforma do Omni
ApplicationExampleTeleoperation.cpp	Programa principal
RelatórioTG2.pdf	Versão do relatório do Trabalho de Graduação 2 assinada pela Banca Examinadora