UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

DANIEL FERNANDO TELLO GAMARRA

CONTROLE DA NAVEGAÇÃO DE UM ROBÔ MÓVEL EM UM CORREDOR COM REDUNDÂNCIA DE CONTROLADORES

VITÓRIA
2004

DANIEL FERNANDO TELLO GAMARRA

CONTROLE DA NAVEGAÇÃO DE UM ROBÔ MÓVEL EM UM CORREDOR COM REDUNDÂNCIA DE CONTROLADORES

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Engenharia Elétrica, na área de concentração em Automação.
Orientador: Prof. Dr. Mário Sarcinelli Filho.

VITÓRIA
2004

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Central da Universidade Federal do Espírito Santo, ES, Brasil)

Gamarra, Daniel Fernando Tello, 1976-

G186c Controle da navegação de um robô móvel em um corredor com redundância de controladores / Daniel Fernando Tello Gamarra. – 2004.

97 f. : il.

Orientador: Mário Sarcinelli Filho.

Co-Orientador: Teodiano Freire Bastos Filho.

Dissertação (mestrado) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Robôs móveis. 2. Robôs - Sistemas de controle. I. Sarcinelli Filho, Mário. II. Bastos Filho, Teodiano Freire. III. Universidade Federal do Espírito Santo. Centro Tecnológico. IV. Título.

CDU: 621.3

DANIEL FERNANDO TELLO GAMARRA

CONTROLE DA NAVEGAÇÃO DE UM ROBÔ MÓVEL EM UM CORREDOR COM REDUNDÂNCIA DE CONTROLADORES

Dissertação submetida ao programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para a obtenção do Grau de Mestre em Engenharia Elétrica - Automação.

Aprovada em 29 de junho de 2004.

COMISSÃO EXAMINADORA

Prof. Dr. Mário Sarcinelli Filho
Universidade Federal do Espírito Santo
Orientador

Prof. Dr. Teodiano Freire Bastos Filho
Universidade Federal do Espírito Santo
Co-orientador

Prof. Dr. Eduardo Oliveira Freire
Universidade Federal de Sergipe

Profa. Dra. Eliete Maria de Oliveira Caldeira
Faculdade Novo Milênio

Dedico esta dissertação a minha mãe Vilma, sol de minha vida,
a minha tia Yolanda, por sua ajuda, carinho e preocupação constante desde sempre,
e a todos aqueles que lutam por ideais de verdade, beleza,
virtude e maior justiça social neste mundo.

Agradecimentos

Dedico meus sinceros agradecimentos para:

–o professor doutor Mário Sarcinelli Filho, meu orientador e ao professor doutor Teodiano Freire Bastos Filho, meu co-orientador, meu mais profundo e sincero agradecimento pela orientação, dedicação e tempo dedicados ao desenvolvimento deste trabalho, dentro desta linha igualmente desejaria agradeçer ao professor Hans-Jorg Andreas Schneebeli, por sua ajuda e dicas na área de programação que foram determinantes e fundamentais para a construção da plataforma do robô;

– o CNPQ, pela bolsa concedida a qual foi básica e indispensável para viver neste grande país continente chamado Brasil; também a meus irmãos Jorge Estuardo e Vilma Lorena, e meu Tio Sergio pela força e apoio desde a distância;

– a meus amigos Marco Antonio, pessoa à qual admiro e aprecio por sua capacidade e como ser humano, pessoa da qual meu país pode esperar só grandes coisas, ao Francisco pela amizade e conselho, a meu amigo Andre Ferreira, companheiro de luta durante tudo este tempo, sua amizade, conhecimento e ajuda foram importantes durante a dissertação;

– os amigos do laboratrio os quais têmse convertido em minha familia durante este estágio de minha vida especialmente ao Jonathan; grande amigo meu e companheiro na luta também, Rodrigo; pela amizade, gentileza, troca de ideias, caronas e bons papos; Wanderlei pela amizade e correções ortográficas, Josemar pela amizade e as dicas de informática, igualmente gostaria agradecer com igual entusiasmo a meus amigos Elizeu Pandolfi, Raquel, Roger, Jaines, Paulo André, Leo Simas e demais amigos do laboratório, aos meninos de iniciação científica pela colaboração no trabalho e amizade, especialmente a Flavio, Rodolfo e Rosenfeld, também a Sandra e Espindula pela amizade;

–finalmente gostaria agradecer a duas famílias que em vitôria me têm tratado com extrema gentileza e consideração não merecida com respeito a minha pessoa, a família de Marco especialmente a Tia Mariulza, a Tia Marina, o tio Pinheiro e Marcos Paulo; e a família de meu amigo André especialmente a Idmar e Penha, a todos eles muito obrigado.

"Yo prefiero mil veces que miremos hacia nosotros con exageración a que, nos perdamos en un internacionalismo simplista y necio o en un europeísmo de remedo, vicio de nuestros intelectuales, barniz de nuestras mediocridades. Soy indoamericanista porque creo con Engels que la realidad no se inventa, se descubre. No pertenezco a los que buscan el remedio de nuestros males fuera de nosotros mismos [...] He vuelto de Europa más indoamericano que nunca. He visto desde lejos a nuestra América con interés y con admiración. Convencido de la urgencia de su unidad, para defendernos del imperialismo amenazador, creo que cada país debe buscar sus verdaderos valores, reivindicarlos y ofrecer a la gran tarea histórica de luchar contra el enemigo del Norte y de afirmar nuestra soberanía. " **Victor Raul Haya de la Torre**

Resumo

Este trabalho apresenta uma estratégia de controle para que um robô móvel possa navegar ao longo da linha média de um corredor. A estratégia de controle é baseada na fusão de dois sinais de controle gerados por dois controladores distintos, os quais são redundantes, no sentido de que têm o mesmo objetivo de controle. Estes sinais correspondem a um sistema de controle baseado em visão, que utiliza a técnica de fluxo óptico, e um sistema de controle baseado em sensores ultra-sônicos. Estes sinais de controle são fusionados utilizando um filtro de informação descentralizado, pelas vantagens que tal mecanismo de fusão de dados apresenta.

Abstract

This work presents a control strategy for a mobile robot perform the task of navigating along the middleline of the corridor. The control strategy is based on fusing two control signals generated by two different controllers, which are redundant, in the sense that they have the same control goal. These control signals correspond to a vision-based controller that uses the optical flow technique and to an ultrasonic sensor-based controller. These signals are fused using a decentralized information filter, for the advantages such mechanism of date fusion presents.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução p. 15

1.1 Trabalhos Relacionados . . . . . p. 16

1.1.1 Fusão de Sinais de Controle Baseados em Visão . . . . . p. 16

1.1.2 Navegação Visual: Combinando Servocontrole e Métodos Baseados na Aparência . . . . . p. 16

1.1.3 Sistema de Navegação Estéreo Divergente . . . . . p. 18

1.1.4 Sistema para Navegação e Guiagem de Robôs Móveis Autônomos . . . . . p. 18

1.2 Definição do Problema . . . . . p. 19

1.2.1 Objetivo do Trabalho . . . . . p. 20

1.2.2 Contextualização . . . . . p. 20

1.3 Estrutura da Dissertação . . . . . p. 22

2 Os Controladores Utilizados p. 23

2.1 Fluxo Óptico . . . . . p. 24

2.2 Técnicas para o Cálculo do Fluxo Óptico . . . . . p. 26

2.2.1 Métodos Diferenciais . . . . . p. 26

2.2.2 Métodos Baseados em Freqüência . . . . . p. 27

2.2.3 Métodos Baseados em Correlaçao . . . . . p. 27

2.2.4 Métodos de Movimento Múltiplo . . . . . p. 28
2.3 Algoritmo de Mínimos Quadrados . . . . . p. 28
2.4 Modelos do Robô e da Câmara . . . . . p. 30
2.4.1 Modelo do Robô . . . . . p. 30
2.4.2 Modelo da Câmara . . . . . p. 32
2.4.3 Modelo Diferencial Câmara-Robô . . . . . p. 32
2.5 Controlador Baseado em Fluxo Óptico . . . . . p. 33
2.6 Controlador Baseado em Ultra-Som . . . . . p. 36

3 Fusão de Dados p. 38
3.1 Fusão Sensorial . . . . . p. 38
3.2 O Filtro de Kalman . . . . . p. 39
3.3 O Filtro de Kalman Descentralizado . . . . . p. 42
3.4 O Filtro de Informação . . . . . p. 42
3.5 O Filtro de Informação Descentralizado . . . . . p. 43
3.6 Aplicação para o Cálculo de Fluxo Óptico . . . . . p. 44
3.7 Fusão de Sinais de Controle . . . . . p. 46

4 Resultados Experimentais p. 49
4.1 Descrição do *"Hardware"* . . . . . p. 49
4.2 Movimento no Corredor . . . . . p. 49
4.2.1 Movimento do robô perto da parede esquerda . . . . . p. 50
4.2.2 Movimento do robô perto da parede direita . . . . . p. 50
4.2.3 Movimento do robô no centro do corredor . . . . . p. 52
4.3 O Algoritmo Proposto para Cálculo de Fluxo Óptico . . . . . p. 52
4.3.1 Aplicação do algoritmo proposto . . . . . p. 55
4.3.2 Complexidade computacional do algorimo proposto . . . . . p. 55

4.4 Experimentos de Navegação em corredores . . . . . p. 58

5 Conclusões p. 67

Referências p. 69

Apêndice A – Código do programa fusioncontrolertotal.c p. 72

Lista de Figuras

1 Seqüência de quadros de imagem para cálculo do fluxo óptico. . . . . p. 24

2 Fluxo óptico calculado a partir dos quadros da imagem. . . . . p. 25

3 Um exemplo do campo de fluxo óptico calculado pelo Método de Mínimos Quadrados. . . . . p. 30

4 Outro exemplo de campo de fluxo óptico calculado pelo Método de Mínimos Quadrados. . . . . p. 31

5 Sistemas de coordenadas do ambiente ([W]), do robô ([R]) e da câmara ([C]). . . . . p. 31

6 Obtenção do modelo da câmara. . . . . p. 32

7 Sistema de controle baseado em fluxo óptico proposto. . . . . p. 33

8 Componentes horizontais do fluxo óptico medido nas paredes do corredor, quando o robô está no seu centro. . . . . p. 34

9 Controlador baseado em ultra-som utilizado. . . . . p. 36

10 O filtro de Kalman . . . . . p. 40

11 O filtro de informação descentralizado. . . . . p. 44

12 Divisão da imagem em subregiões para o cálculo do fluxo óptico . . . . p. 45

13 O esquema de controle adotado. . . . . p. 47

14 O robô usado nos experimentos. . . . . p. 50

15 Experimento de movimento no corredor com o robô perto da parede esquerda. . . . . p. 51

16 Experimento de movimento no corredor com o robô perto da parede direita. p. 51

17 Experimento de movimento em corredor com o robô no centro. . . . . p. 52

18 Seqüência de imagens “NASA”. . . . . p. 56

19 Seqüência de imagens “rubic”. . . . . p. 56

20 Seqüência de imagens “Táxi de Hamburgo”. . . . . p. 57

21 Variâncias resultantes do algoritmo de mínimos quadrados com fusão e do algorimo de mínimos quadrados modificado com filtro de informação descentralizado para a seqüência de imagens “NASA”. . . . . p. 57

22 Variâncias resultantes do algoritmo de mínimos quadrados com fusão e do algorimo de mínimos quadrados modificado com filtro de informação descentralizado para a seqüência de imagens “rubic”. . . . . p. 58

23 Variâncias resultantes do algoritmo de mínimos quadrados com fusão e do algorimo de mínimos quadrados modificado com filtro de informação descentralizado para a seqência de imagens “táxi de Hamburgo”. . . . . p. 58

24 Trajetória do robô ao longo do corredor. . . . . p. 61

25 Covariâncias associadas aos controladores sobrepostas. . . . . p. 62

26 Velocidade angular produzida pelo controlador de fluxo óptico. . . . . p. 62

27 Velocidade angular produzida pelo controlador de ultra-som. . . . . p. 63

28 Velocidade angular resultante do processo de fusão e velocidade angular medida pelo robô. . . . . p. 63

29 Posição do robô no final do corredor, quando a informação de fluxo óptico não está mais disponível. . . . . p. 64

30 Trajetória do robô ao longo do corredor. . . . . p. 64

31 Covariâncias associadas aos controladores sobrepostas. . . . . p. 65

32 Velocidade angular produzida pelo controlador de fluxo óptico. . . . . p. 65

33 Velocidade angular produzida pelo controlador de ultra-som. . . . . p. 66

34 Velocidade angular resultante do processo de fusão e velocidade angular medida pelo robô. . . . . p. 66

Lista de Tabelas

1 Valores do fluxo óptico e suas variâncias. . . . . p. 46

2 Valor do fluxo óptico resultante da fusão e sua variância . . . . . p. 46

3 Comparação do esforço computacional para algoritmos de cálculo do fluxo óptico. . . . . p. 59

4 Comparação de tempos de execução de algoritmos de cálculo de fluxo a bordo de um robô móvel. . . . . p. 59

1 Introdução

Resultados recentes na literatura, assim como trabalhos desenvolvidos pelo grupo de Robótica e Visão Computacional da Universidade Federal do Espírito Santo, têm abordado o problema de navegação de um robô num corredor processando a informação proveniente somente de sensores ultra-sônicos (FREIRE et al., 2004) ou somente de câmaras de vídeo (VASSALLO; SCHNEEBELI; SANTOS-VICTOR, 2000) (SANTOS-VICTOR et al., 1995).

No caso de câmaras de vídeo, a justificativa para seu uso é porque a visão é o sentido que fornece ao robô informação mais rica com respeito ao ambiente com o qual ele interage. Porém, mesmo sendo a visão um sentido tão rico e poderoso, à medida que o ambiente se torna mais complexo é necessária maior robustez nas informações fornecidas por uma câmara de video instalada em um robô, e se faz necessário contar com um maior número de sensores, o que faz com que as medições e o desempenho do sistema sensorial sejam melhores (HONG, 1999). Por isso, além de utilizar a visão, neste trabalho utilizaremos também os sensores de ultra-som.

Definidos os sensores a utilizar em nosso sistema, será adotada a arquitetura baseada na fusão de sinais de controle gerados por distintos controladores, proposta em (FREIRE, 2002), sendo que um dos controladores a ser utilizado é baseado nos sensores de ultra-som, e o outro é baseado no cálculo do fluxo óptico. O sinal de controle final enviado aos motores do robô é obtido pela fusão dos sinais gerados por cada controlador individual. Para realizar tal fusão de controladores, usaremos um filtro de informação descentralizado. É importante mencionar que praticamente todos os experimentos até aqui realizados com tal arquitetura de controle (FREIRE, 2002) usavam controladores responsáveis por fases de navegação distintas, todos eles recebendo informação proveniente apenas de sensores ultra-sônicos. Neste ponto é que surgiu a idéia de utilizar dois controladores com o mesmo objetivo de fazer o robô seguir a linha média de um corredor, e fazer a fusão da saída de ambos para gerar o sinal de controle efetivo para o robô, ou seja, ter dois controladores redundantes (no sentido de ter o mesmo objetivo de controle). Os controladores utilizados,

porém se baseiam em informação sensorial absolutamente distinta, com características de complementariedade.

1.1 Trabalhos Relacionados

A tarefa de navegação por corredores e a técnica de fusão sensorial ja foram abordadas anteriormente. A seguir detalharemos alguns trabalhos de relevância desenvolvidos neste campo.

1.1.1 Fusão de Sinais de Controle Baseados em Visão

(CARELLI et al., 2002) apresenta uma estratégia de controle para navegação de robôs móveis em corredores usando a fusão de sinais de controle, gerados por dois controladores baseados em visão. Tais controladores são um controlador baseado no cálculo do fluxo óptico e outro baseado nas linhas de perspectiva do corredor.

No caso do controlador baseado nas linhas de perspectiva, as linhas paralelas formadas pela confluência entre as paredes do corredor e o chão são projetadas no plano da imagen, e se interceptam no ponto de fuga. Já o controlador de fluxo óptico utiliza o fluxo em duas regiões simétricas no plano da imagem, e busca igualá-los nas regiões citadas.

Os controladores implementados são também redundantes, pelo fato de terem o mesmo objetivo de controle, que é guiar o robô pela linha média do corredor, mas sua natureza não é totalmente diferente, pois ambos se baseiam em visão. A fusão de ambos os sinais de controle, em tal trabalho, é feita utilizando um filtro de Kalman descentralizado.

Esta arquitetura é muito interessante, mas usa três computadores para realizar os cálculos necessários para a navegação do robô: um primeiro computador realiza o cálculo do fluxo óptico, um segundo realiza o cálculo das linhas de perspectiva do corredor, as quais são calculadas utilizando a transformada de Hough, e um terceiro calcula os sinais de controle e implementa sua fusão. O sinal resultante é enviado ao robô por um enlace de radiofrequência. Como se pode perceber, o custo computacional é bastante elevado, e os recursos exigidos são muitos, além do fato de que todo o controle depende de uma mesma fonte sensorial, o sensor de visão, que caso venha a falhar deixa o robô sem nenhum tipo de informação de seu entorno.

1.1.2 Navegação Visual: Combinando Servocontrole e Métodos Baseados na Aparência

Em (VASSALLO; SCHNEEBELI; SANTOS-VICTOR, 2000) é feito uso de um robô com sistema de visão monocular. O sistema de controle se baseia em dois paradigmas distintos: de um lado há um servocontrole baseado no ponto de fuga, que é obtido pela interseção das linhas de perspectiva do corredor no plano da imagem, e por outro lado é usada a aparência para monitorar a posição do robô, tanto para mantê-lo em sua trajetória como para executar diferentes tarefas de navegação.

O servocontrole baseado na localização do ponto de fuga utiliza a informação proveniente das linhas do corredor para controlar o movimento de rotação do robô. As linhas do corredor são extraídas utilizando o operador gradiente de Sobel, e utilizando um conjunto de restrições para selecionar os pontos de interese, correspondentes às linhas do corredor. Basicamente, o procedimento utilizado é:

- extrair as componentes vertical e horizontal dos gradientes da imagem;
- selecionar os pontos que contêm informação do gradiente horizontal e vertical;
- separar este conjunto de pontos em subconjuntos de pontos de gradientes com orientação positiva e gradientes com orientação negativa (cada subconjunto corresponde a uma das linhas do corredor);
- erodir cada imagem com uma máscara correspondente às linhas do corredor;
- usar os pontos selecionados para estimar robustamente as linhas do corredor.

Uma vez que as duas linhas estejam determinadas, é calculado o ponto de fuga, cujo desvio em relação à coluna central da imagem define o sinal de erro, que ativa o controlador de orientação do robô.

O método baseado na aparência complementa este trabalho, que utiliza uma representação topológica do ambiente. Tal representação pode ser vista como um grafo, no qual existem "enlaces" que correspondem a segmentos de trajétoria, onde o servo-controle visual pode ser utilizado, e "nós", que correspondem a pontos onde ações especiais têm que ser tomadas, como virar, entrar por uma porta, evitar um obstáculo, etc.

O progresso ao longo dos "enlaces" é monitorado usando um conjunto de imagens referenciais adquiridas em posições estratégicas ao longo do corredor. Durante a operação

normal, uma imagem tomada pelo robô é comparada com as imagens de referência guardadas por ele, através do método de somas de quadrados de diferenças. Esta comparação permite descobrir de qual imagem a imagem corrente está mais próxima, e, de acordo com isso, tomar uma ação previamente definida (virar, entrar por uma porta, etc).

1.1.3 Sistema de Navegação Estéreo Divergente

Em (SANTOS-VICTOR et al., 1995) é proposto um sistema que se enquadra no campo de navegação guiada visualmente, e se baseia no cálculo do fluxo óptico. O uso do fluxo óptico foi motivado por estudos e experimentos realizados com abelhas (LEHRER et al., 1988). O uso que elas dão a esta informação permite resolver problemas de navegação e aterrissagem em superfícies.

Seguindo essa linha de pensamento, um robô móvel (Robee) foi equipado com duas câmaras laterais, em direções opostas, em analogia com a posição dos olhos das abelhas e outros insetos, de tal modo que os campos visuais não se sobrepõem. Tal técnica é chamada Navegação Estéreo Divergente.

Uma particularidade deste sistema é que a rotação das câmaras é implementada de forma simples. Ambas são sempre giradas em conjunto (ângulo de *pan*), para ficarem na direção perpendicular à orientação do robô . Outra característica interessante deste trabalho é a implementação de um mecanismo de sustentação (*sustaining mechanism*) que estabiliza a velocidade angular do robô quando ocorre uma perda de informação de fluxo, causada pela falta de textura ou por mudanças na estrutura do ambiente.

O sistema tem dois laços de controle: um laço de controle de navegação, que controla a velocidade de rotação para balancear o fluxo à esquerda e à direita, e um laço de controle da velocidade, que controla a velocidade linear, acelerando ou desacelarando o robô em função da amplitude dos campos de fluxo laterais.

Esse trabalho apresenta muitos aspectos interessantes, mas também possui a desvantagem que o robô é completamente cego na direção do movimento.

1.1.4 Sistema para Navegação e Guiagem de Robôs Móveis Autônomos

Em (SANDI-LORA; HEMERLY; LAGES, 1998) é utilizada a fusão sensorial via filtro de Kalman de dois subsistemas independentes, o *dead-reckoning*, que utiliza a leitura

de dois *encoders* localizados nas rodas esquerda e direita do robô, e uma bússola digital, montada no centro do eixo das rodas dianteiras, que permite medir a orientação do veículo. A orientação corrigida é realimentada para calcular a posição, de modo a se reduzir o acúmulo de erros. A precisão do *dead-reckoning* depende diretamente da qualidade dos sensores utilizados, sendo inevitável o acúmulo do erro de posição, o que requer a fusão com outros sensores para compensar este erro.

Os sensores utilizados tinham períodos de amostragem diferentes, o que gerou a implementação de um ambiente multitarefa baseado em Linux. A leitura da bússola e as leituras dos encoders são feitas por tarefas periódicas diferentes, cada uma com seu tempo de amostragem. Quando uma tarefa necessita gerar um retardo, ela é posta para "dormir"durante este tempo, liberando o processador para as demais tarefas. As tarefas foram implementadas como *"threads"*, que oferecem a vantagem de compartilhar o mesmo espaço de endereço, além de vários outros recursos, com o que a troca de dados é feita mais facilmente.

O controlador utilizado para a guiagem do veículo é do tipo *"fuzzy"*. Tal controlador é baseado na posição e orientação obtidas na fase de navegação. As regras do controlador são obtidas utilizando um esboço em papel para visualizar os caminhos que devem ser seguidos para se atingir o objetivo. As regras de inferência do controlador são obtidas levando-se em consideração o raio de giro do veículo e a sua velocidade média. Assim, utiliza-se um esboço em papel das trajetórias a serem seguidas, desde o ponto de partida até se atingir o ponto de chegada.

Esse trabalho é um caso muito interesante de fusão sensorial para o caso de navegação de robôs, mas é mais adequado para o caso de navegação de ponto a ponto que para o caso de navegação em corredores. Também, deve-se considerar o fato de que a odometria não é uma fonte sensorial na qual se possa confiar, dada a grande variação possível no erro acumulado.

Arquiteturas utilizando fusão sensorial têm sido utilizadas em tarefas de navegação de robôs essencialmente no controle de ponto a ponto. Após uma revisão exaustiva da literatura não achamos aplicações no caso específico de navegação em corredores, embora tenhamos encontrado aplicações de navegação de robôs usando tal arquitetura, tal como pode ser visto em (FUKE; KROTKOV, 1996) onde se integra a informação do número de pulsos detectados por *encoders* instalados nas rodas do veículo com informação de acelerômetros e giroscópios.

1.2 Definição do Problema

O problema que se deseja resolver neste trabalho é o problema de navegação de robôs em corredores, num ambiente estruturado, utilizando uma arquitetura de controle reativa baseada na redundância de dois controladores distintos, os quais utilizam fontes sensoriais diferentes, obtendo-se um único sinal de controle pela fusão dos sinais de controle gerados pelos dois controladores independentes.

Entenda-se por ambiente estruturado aquele onde os parâmetros necessários à operacionalidade do sistema robótico podem ser identificados e quantificados (ROMANO, 2002). Nosso ambiente apresenta algumas características que foram definidas a priori, como o fato de ter um chão plano e utilizar texturas nas paredes dos corredores para melhorar o cálculo de fluxo.

1.2.1 Objetivo do Trabalho

O objetivo deste trabalho é controlar um robô móvel de forma a mantê-lo navegando ao longo da linha média do corredor. Para tanto, será usada informação proveniente de sensores ultra-sônicos e de uma câmara de vídeo instalada a bordo do robô.

Assim, será usada uma estrutura de dois controladores redundantes para cumprir a mesma finalidade, ou seja, conduzir o robô ao longo do corredor, e o sinal de controle global será gerado através de um mecanismo de fusão de dados. Os controladores a serem usados se baseiam em informação sensorial absolutamente distinta, com características de complementaridade.

Para cumprir tal objetivo, definimos as seguintes tarefas a serem realizadas:

1. analisar o Filtro de Kalman (MUTAMBARA, 1998), o Filtro de Informação (MUTAMBARA, 1998), e o Filtro de Informação Descentralizado (FREIRE, 2002);
2. analisar o fluxo óptico e as técnicas utilizadas para calculá-lo (CALDEIRA, 2002), e a partir daí propor a técnica a ser utilizada por nosso sistema;
3. implementar um controlador baseado em ultra-som já disponível na literatura (FREIRE, 2002);
4. implementar um controlador baseado em fluxo óptico (CARELLI et al., 2002);

5. descrever e implementar o controlador para navegar em corredores utilizando fusão de sinais de controle (FREIRE, 2002) (FREIRE et al., 2004) a partir de dois controladores redundantes;

6. analisar o desempenho dos controladores e da fusão dos sinais de controle, através de experimentos usando o robô móvel PIONEER 2DX.

1.2.2 Contextualização

Conhecidas as etapas a serem seguidas para atingir o objetivo proposto, e considerando os trabalhos apontados na Seção 1.1, podemos destacar alguns aspectos do trabalho aqui apresentado :

- com relação aos cálculos realizados, apenas o computador de bordo do robô é utilizado, pois não há necessidade de segmentar a informação e os cálculos em vários computadores, como em (CARELLI et al., 2002);

- diferentemente de (VASSALLO; SCHNEEBELI; SANTOS-VICTOR, 2000) e (CARELLI et al., 2002), a plataforma de trabalho empregada neste projeto é baseada no sistema operacional LINUX, a qual permite, efetivamente trabalhar em tempo real;

- em comparação com o sistema proposto em (SANTOS-VICTOR et al., 1995), é preciso mencionar que nosso sistema utiliza uma só câmara, e não duas. Mesmo assim é garantido que o robô navega pela linha média do corredor (CARELLI et al., 2002) (DEV; KROSE; GROEN, 1997);

- também é importante mencionar que nosso sistema não necessita a reconstrução 3D do ambiente de trabalho, sendo baseado somente nas estimativas de fluxo óptico e medições por ultra-som;

- outra vantagem de nossa plataforma é que ela não requer um sistema de controle para a câmara cada vez que o robô gire, o que simplifica a estratégia de controle a utilizar;

- com respeito a (FREIRE et al., 2004) e (FREIRE, 2002), a diferença básica é que os controladores aqui utilizados são redundantes, ou seja, têm o mesmo objetivo. Neste sentido, os experimentos aqui relatados complementam aqueles ali relatados, onde somente controladores responsáveis por fases diferentes de navegação foram utilizados. É importante, também, ressaltar que a plataforma de trabalho de nosso

sistema é baseada no ARIA, que é uma plataforma de nível mais baixo que o Saphira para fazer a interface com o robô PIONEER 2DX, o qual é utilizado em (CARELLI et al., 2002), (FREIRE et al., 2004) e (FREIRE, 2002);

- com relação ao trabalho relatado em (SANDI-LORA; HEMERLY; LAGES, 1998), há bastante analogia, já que um esquema baseado em fusão de dados é empregado, assim como uma plataforma baseada em LINUX. As principais diferenças ficam, no caso, por conta do uso do filtro de informação descentralizado (FREIRE, 2002) como máquina de fusão, e do uso da informação visual.

1.3 Estrutura da Dissertação

A descrição do sistema implementado e experimentado nesta dissertação se dá através de uma estrutura distribuída em capítulos. Tais capítulos são os seguintes:

- **Capítulo 1: Introdução**

  Neste capítulo é apresentada uma introdução geral, assim como os trabalhos relacionados ao tema anteriormente desenvolvidos. É definido o problema a ser estudado e o objetivo a atingir, assim como são definidas as tarefas a serem realizadas e a estrutura da Dissertação.

- **Capítulo 2: Controladores Utilizados**

  Neste capítulo é apresentado o conceito de fluxo óptico, assim como as diversas técnicas utilizadas para o seu cálculo. Também são apresentados o controlador de fluxo óptico e o controlador à base de ultra-som que serão utilizados.

- **Capítulo 3: Fusão de Dados**

  Neste capítulo se faz uma referência à fusão sensorial e às distintas técnicas utilizadas em fusão sensorial, à teoria do Filtro de Kalman, do Filtro de Informação e do Filtro de Informação Descentralizado, assim como o uso da fusão para nossos sinais de controle.

- **Capítulo 4: Resultados Experimentais**

Neste capítulo se descreve o robô PIONEER 2DX e a plataforma de trabalho desenvolvida em Linux. Apresentam-se os testes feitos com o robô para avaliar o desempenho de nosso sistema, e são discutidos os resultados experimentais obtidos.

- **Capítulo 5: Conclusões**

Neste capítulo são apresentadas as conclusões obtidas do desenvolvimento deste trabalho, assim como os trabalhos futuros que poderiam ser implementados dentro desta linha.

2 Os Controladores Utilizados

Os insetos têm uma surpreendente habilidade de navegar com precisão e rapidez no espaço tridimensional, mesmo possuindo olhos relativamente simples e um sistema nervoso e cérebro pequenos para processar a informação disponível. Este fato tem servido como fonte de inspiração aos cientistas para pesquisar as técnicas que os insetos utilizam para navegar (ZWANN; SANTOS-VICTOR, 1997), e a conclusão comum é que eles utilizam fluxo óptico para detectar obstáculos e estimar distâncias. Neste capítulo, então, é feito um pequeno estudo sobre o fluxo óptico, que será uma das ferramentas da qual nos serviremos para a realização de nosso trabalho. O fluxo óptico é apresentado, assim como as diferentes técnicas usadas para o seu cálculo. É dada atenção especial aos métodos diferenciais, e, dentre eles ao método de mínimos quadrados (LUKAS; KANADE, 1981). Com base no cálculo do fluxo óptico, é descrito o controlador de fluxo que será utilizado, e que é baseado no trabalho de (CARELLI et al., 2002), assim como é descrito o controlador à base de ultra-som utilizado em nosso trabalho, que é o mesmo controlador desenvolvido em (FREIRE, 2002).

Este trabalho utiliza duas fontes diferentes de informação sensorial para implementar dois controladores distintos, baseados nas informações por elas fornecidas. Uma das fontes é uma câmara de vídeo CCD, usada em conexão com a técnica de fluxo óptico. É, então, implementado um controlador que utiliza a disparidade entre os valores do fluxo óptico nas paredes à esquerda e à direita do corredor para obter o sinal de controle. A segunda fonte de informação sensorial é um conjunto de quatro transdutores ultra-sônicos. A disparidade de distâncias às paredes por eles medidas é, então, utilizada para gerar outro sinal de controle. Note-se que estes dois controladores são redundantes, pelo fato de que ambos têm o mesmo objetivo de controle, ou seja, fazer com que o robô navegue pela linha média do corredor(FREIRE, 2002).

Também é apresentado o modelo matemático do robô e da câmara, necessário para poder entender com maior clareza o controlador à base de fluxo óptico.

2.1 Fluxo Óptico

Se tomarmos uma seqüência de imagens no tempo, e se existem objetos movendo-se na cena ou se a câmara está a bordo de um veículo em movimento, informação importante sobre o que a imagem contém pode ser obtida analisando e entendendo a diferença entre duas imagens, causada pelo movimento. De uma seqüência de imagens em movimento (Figura 1) podemos calcular uma nova função, chamada de fluxo óptico. O fluxo óptico é definido como a distribuição de velocidades aparentes do movimento do padrão de brilho de uma imagem, e aparece, geralmente, devido ao movimento relativo entre os objetos e a câmara, sempre que se considera uma seqüência de quadros de imagem ao longo do tempo (HORN; SCHUNCK, 1981). Ele é expresso, como podemos visualizar na Figura 2, na forma de vetores que caracterizam tal movimento.

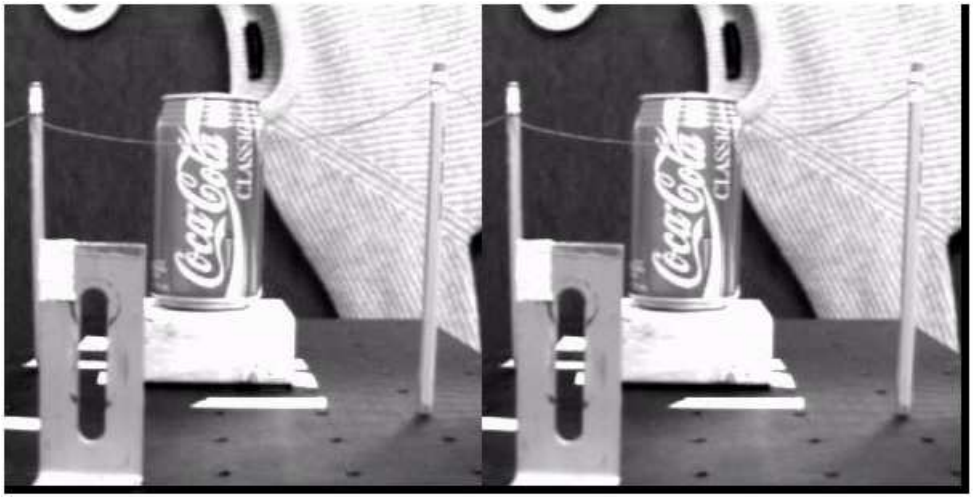

Figura 1: Seqüência de quadros de imagem para cálculo do fluxo óptico.

Durante muitos anos, o problema do processamento de seqüências de imagens para calcular o fluxo óptico tem sido objeto de estudo. O interesse por fluxo óptico resultou em sua utilização em aplicações as mais diversas, tais como prover informação importante com respeito ao arranjo temporal dos objetos vistos e a velocidade de mudança deste arranjo (GIBSON, 1977), recuperar informações da cena tridimensional e parâmetros de movimento, a partir de uma única câmara em movimento (ZHANG; FAUGERAS, 1992) e (ZHENG; CHELLAPPA, 1993), segmentar movimento (BLACK; ANANDAN, 1990), calcular o foco de expansão e o tempo de colisão (SARCINELLI-FILHO; SCHNEEBELI; CALDEIRA, 2002), calcular a disparidade estéreo (LANGLEY et al., 1991) e medir o fluxo sanguíneo e o movimento das paredes do coração em imagens médicas (PRINCE; MCVEIGH, 1992), e, em alguns casos, recuperar informações acerca da forma dos objetos

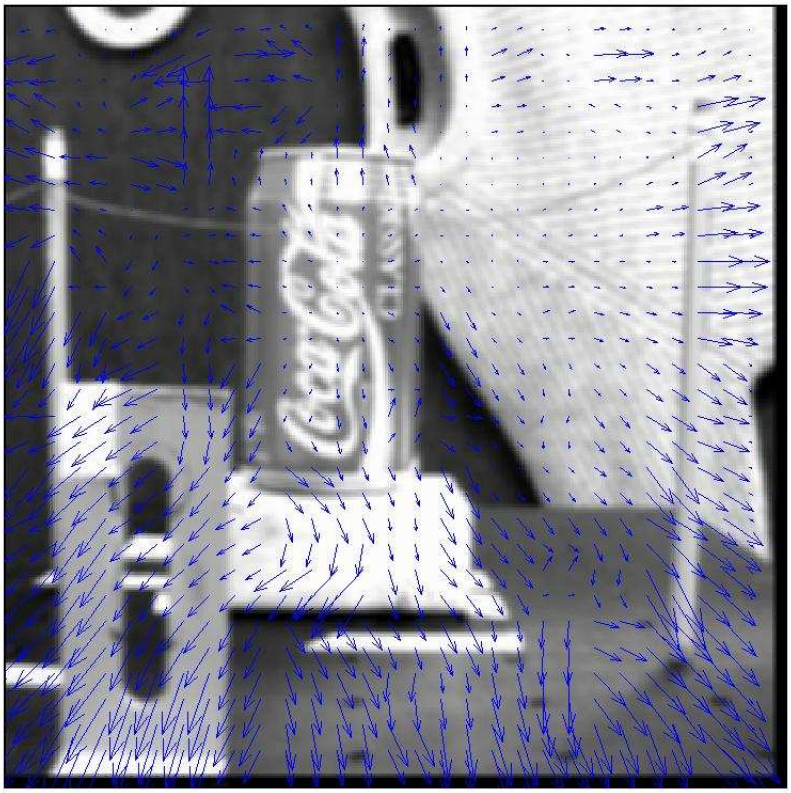

Figura 2: Fluxo óptico calculado a partir dos quadros da imagem.

(KOENDERINK; DOORN, 1976) (CLOCKSIN, 1978). Mais recentemente, o fluxo óptico também tem sido aplicado na navegação de robôs móveis (SARCINELLI-FILHO; SCHNEEBELI; CALDEIRA, 2002)(DEV; KROSE; GROEN, 1997) (CARELLI et al., 2002), o que também é o objetivo final deste trabalho.

Para o cálculo do fluxo óptico, é obtido um vetor velocidade $V = (u, v)$ em cada *pixel* ou grupo de *pixels* considerado. Vamos supor que a intensidade do brilho da imagem é dada por $I(x, y, t)$, como uma função do tempo, t, assim como de x e y. Em outro ponto da imagem infinitesimalmente distante e num tempo infinitesimalmente depois, a intensidade é dada por

$$I(x + dx, y + dy, t + dt) = I(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt + ..., \qquad (2.1)$$

onde as reticências representam as derivadas de ordem mais elevada. Agora, suponhamos que parte de um determinado objeto está na posição (x, y) na imagem, no tempo t, e

que para um tempo dt depois aquela parte moveu-se uma distância (dx, dy) ao longo da imagem. Agora, consideremos que a intensidade do brilho é a mesma antes e depois. Portanto, a equação (2.1) é valida nos dois instantes e nas duas posições da imagem, ou seja,

$$I(x + dx, y + dy, t + dt) = I(x, y, t). \tag{2.2}$$

Então,

$$\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt + ... = 0. \tag{2.3}$$

Dividindo a equação (2.3) por dt temos que

$$\frac{dx}{dt} = u, \qquad \frac{dy}{dt} = v, \tag{2.4}$$

são as velocidades com que o objeto está se movendo nas direções x e y, respectivamente. Assim, no limite quando dt tende para zero, temos que

$$-\frac{\partial I}{\partial t} = \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v,$$

ou, equivalentemente,

$$-I_t = I_x u + I_y v, \tag{2.5}$$

equação esta que é chamada **equação de restrição de fluxo óptico**. Nela $\frac{\partial I}{\partial t} = I_t$, num determinado *pixel*, significa a velocidade com a qual a intensidade está mudando em relação ao tempo, enquanto $\frac{\partial I}{\partial x} = I_x$ e $\frac{\partial I}{\partial y} = I_y$ representam a velocidade com a qual a intensidade está mudando em relação ao espaço. Estas três componentes podem ser calculadas para cada *pixel* da imagem, e, a partir delas as componentes u e v de cada vetor de fluxo óptico podem ser calculadas. O único problema é que a restrição de fluxo óptico (equação(2.5)) sozinha não é suficiente para obter u e v em cada *pixel*. Daí, várias técnicas para cálculo do fluxo óptico foram propostas, cada uma acrescentando alguma nova restrição.

2.2 Técnicas para o Cálculo do Fluxo Óptico

Os algoritmos implementados para o cálculo do fluxo óptico podem ser divididos em quatro grandes grupos (BEAUCHEMIN; BARRON, 1995), como caracterizado a seguir. Entretanto, como nosso objetivo não é propor novas técnicas de obtenção de fluxo óptico, tais técnicas não serão aqui detalhadas.

2.2.1 Métodos Diferenciais

Os métodos diferenciais calculam a velocidade da imagem a partir de derivadas espaço-temporais, ou seja, a partir das derivadas da intensidade da imagem. O domínio da imagem é suposto como contínuo (diferenciável) no tempo e no espaço. Existem diversos métodos diferenciais, tais como:

- os métodos globais;
- os métodos locais;
- métodos de modelo de superfície ou de contorno;
- métodos de multipla restrição;
- enfoques hierárquicos.

2.2.2 Métodos Baseados em Freqüência

Estes métodos são baseados no uso de filtros. Tais técnicas usam filtros de orientação sensitiva no domínio de Fourier em imagens variantes no tempo. Uma das vantagens destes métodos é que um mecanismo sensível ao movimento opera na energia orientada espaço-temporalmente no espaço de Fourier, e pode estimar movimento em imagens para as quais os enfoques baseados em emparelhamento (matching) falhariam. Podemos mencionar os seguintes métodos:

- os métodos globais;
- os métodos locais;
- métodos de filtragem seletiva orientada;
- filtragem baseada na fase;
- enfoques hierárquicos.

2.2.3 Métodos Baseados em Correlaçao

Na prática, pelo fato de utilizar somente alguns poucos quadros de imagem ou uma relação sinal-ruido pobre, os enfoques de diferenciação ou de freqüência não são apropriados. Os métodos diferenciais muitas vezes são pouco robustos pelo fato de trabalhar

com as derivadas da intensidade, e é natural considerar métodos baseados na correlação. Tais métodos tentam estabelecer correspondências entre características invariáveis entre dois quadros de imagem, onde estas características típicas podem ser quinas, bordas, etc. Podemos mencionar os seguintes métodos:

- emparelhamento baseado em correlação;
- enfoques hierárquicos;

2.2.4 Métodos de Movimento Múltiplo

Muitos fenômenos podem gerar movimentos múltiplos na imagem. Entre eles, oclusão e transparência são importantes, devido à sua constante ocorrência e sua significação nas imagens. Além disso, o conteúdo de informação é útil para níveis posteriores de processamento, tais como segmentação de movimento e reconstrução de uma superfície tridimensional. Quando se apresenta uma agudização destes problemas dentro de dois quadros de imagens, os métodos tradicionais falham, e é precisso ter um enfoque que lida com estas perturbações. Para superar estes problemas, têm surgido diversos métodos baseados no movimento múltiplo, tais como:

- método Procesamento de Linha;
- método de distribuição de velocidade mista;
- método de modelos paramétricos.

Para aplicação em tempo real, buscou-se o algoritmo mais conveniente, o qual deve gastar pouco tempo computacional e pouca memória para armazenamento de imagens. Baseando-nos nos estudos de (BARRON; BEAUCHEMIN, 1994), que fizeram um estudo comparativo entre diversos algoritmos de cálculo de fluxo óptico, concluímos que o algoritmo de mínimos quadrados (LUKAS; KANADE, 1981), dentre alguns outros mais, é um dos algoritmos que apresentam melhores resultados, em termos de precisão e robustez, fato este que é confirmado em (CALDEIRA, 2002). Ali, é realizado um estudo mais exaustivo e detalhado dos métodos para cálculo de fluxo óptico que são mais apropriados para o caso de navegação de robôs, e também é demonstrado que o método de mínimos quadrados, por sua robustez, simplicidade, e, principalmente, baixo custo computacional, é adequado para aplicações em tempo real na navegação de robôs.

2.3 Algoritmo de Mínimos Quadrados

Este algoritmo requer o uso de apenas dois quadros de imagem, e, por ser um método diferencial, também é computacionalmente simples, o que o torna muito atrativo (BARRON; BEAUCHEMIN, 1994). Nele, a restrição de fluxo óptico (2.5), que não é suficiente para determinar os valores de u e v para todos os *pixels*, é aplicada partindo do pressuposto adicional de que o fluxo óptico é constante em uma região constituída por um grupo de N por N pixels (LUKAS; KANADE, 1981). Voltando a escrever a equação (2.5), incluindo esta condição, temos o conjunto de equações

$$\begin{aligned} I_{x_1}u + I_{y_1}v + I_{t_1} &= 0, \\ I_{x_2}u + I_{y_2}v + I_{t_2} &= 0, \\ \vdots \qquad & \quad \vdots \\ I_{x_{N^2}}u + I_{y_{N^2}}v + I_{t_{N^2}} &= 0, \end{aligned} \tag{2.6}$$

onde I_x, I_y e I_t são aqui calculadas como proposto em (HORN; SCHUNCK, 1981). Daí o fluxo em todos os *pixels* da região considerada é dado pela estimativa de mínimos quadrados

$$(\hat{u}, \hat{v}) = (\mathbf{A}^t\mathbf{A})^{-1}\mathbf{A}^t\mathbf{b}, \tag{2.7}$$

onde a matriz $\mathbf{A}$ é dada por

$$\mathbf{A} = \begin{bmatrix} I_{x_1}, \cdots, I_{x_{N^2}} \\ I_{y_1}, \cdots, I_{y_{N^2}} \end{bmatrix}^t \tag{2.8}$$

e o vetor $\mathbf{b}$ é dado por

$$\mathbf{b} = -\begin{bmatrix} I_{t_1}, \cdots, I_{t_{N^2}} \end{bmatrix}^t. \tag{2.9}$$

À estimativa $(\hat{u}, \hat{v})$ do fluxo óptico está associada uma variância dada por $\mathbf{e}^t\mathbf{e}$ (SORIA; CARELLI; SARCINELLI-FILHO, 2003), onde

$$\mathbf{e} = \begin{bmatrix} e_1, \cdots, e_{N^2} \end{bmatrix}^t, \tag{2.10}$$

com $e_i = I_{x_i}\hat{u} + I_{y_i}\hat{v} + I_{t_i}$.

Nos exemplos da Figura 3 e da Figura 4 podemos observar o fluxo óptico calculado usando o método de mínimos quadrados na seqüência do Táxi de Hamburgo e em um dos

corredores utilizados nos testes de navegação que foram feitos neste trabalho.

Figura 3: Um exemplo do campo de fluxo óptico calculado pelo Método de Mínimos Quadrados.

2.4 Modelos do Robô e da Câmara

2.4.1 Modelo do Robô

A Figura 5 representa o sistema de coordenadas do robô [R], do ambiente [W] e da câmara [C]. As equações da cinemática do robô, expressas no seu sistema de coordenadas ([R]), são dadas por

$$\begin{aligned} w &= \dot{\theta} \\ \dot{y} &= v cos\theta \\ \dot{x} &= v sen\theta \end{aligned} \tag{2.11}$$

onde v é a sua velocidade linear e w é a sua velocidade angular. Para a dinâmica do robô, utilizamos os mesmos parâmetros obtidos experimentalmente por análise da resposta ao degrau em (CARELLI et al., 2002), por estar trabalhando com o mesmo robô. A relação entre a velocidade angular de referência, gerada pelo controlador e enviada ao

Figura 4: Outro exemplo de campo de fluxo óptico calculado pelo Método de Mínimos Quadrados.

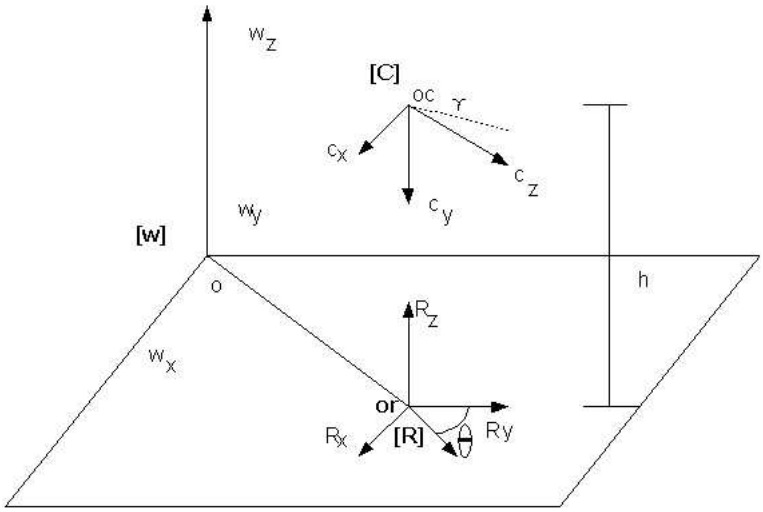

Figura 5: Sistemas de coordenadas do ambiente ([W]), do robô ([R]) e da câmara ([C]).

robô, w_R, e a velocidade angular efetivamente desenvolvida pelo robô, w, foi obtida, a qual é representada por um modelo linear de segunda ordem, a saber

$$w = \frac{k_w a_w}{s^2 + b_w s + a_w} w_R, \tag{2.12}$$

com $k_w = 0.45$, $a_w = 104.6$ e $b_w = 9.21$.

A dinâmica relativa à velocidade linear também foi obtida, mas ela não será utilizada neste trabalho, já que apenas a velocidade angular w será controlada.

2.4.2 Modelo da Câmara

A partir da Figura 6 é obtida a relação

$$r = \lambda \frac{P}{c_{pz}}, \tag{2.13}$$

que expressa o modelo da câmara, onde r é a projeção de um ponto P do espaço no plano da imagem e λ é a distância focal da câmara.

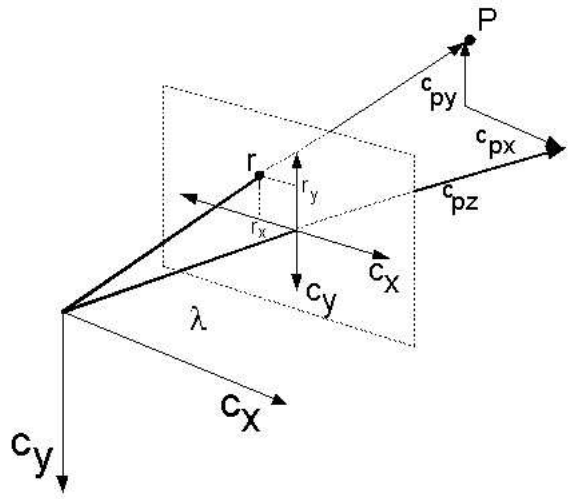

Figura 6: Obtenção do modelo da câmara.

2.4.3 Modelo Diferencial Câmara-Robô

Tomando como base a equação de Coriolis que descreve o movimento de um ponto P em um sistema de coordenadas, o qual se move com velocidade linear V e velocidade angular ω, ou seja,

$$\dot{P} = -V - \omega \times P, \tag{2.14}$$

derivando (2.13) e usando (2.14), as componentes de $\dot{r}$ no plano da imagem são dadas por (CARELLI et al., 2002)

$$\dot{r}_x = \frac{-\lambda V_x + r_x V_z}{c_{pz}} + \frac{w_x}{\lambda} r_x r_y - w_y(\lambda + \frac{r_x^2}{\lambda}) - w_z r_y \tag{2.15}$$

e

$$\dot{r}_y = \frac{-\lambda V_y + r_y V_z}{c_{pz}} + w_x(\lambda + \frac{r_y^2}{\lambda}) - \frac{w_y}{\lambda} r_x r_y - w_z r_x. \tag{2.16}$$

Tomando em conta as simplificações características do sistema sob análise, ou seja, $V_x = V_y = 0$ e $w_x = w_z = 0$, e renomeando as variáveis $V_z = v$ e $w_y = w$, as equações (2.15) e (2.16) podem ser reescritas como

$$\dot{r}_x = \frac{r_x v}{c_{pz}} - w(\lambda + \frac{r_x^2}{\lambda}) \quad \dot{r}_y = \frac{r_y v}{c_{pz}} - \frac{w}{\lambda}(r_x r_y), \tag{2.17}$$

que representam as equações diferenciais cinemáticas para a câmara montada no robô.

2.5 Controlador Baseado em Fluxo Óptico

Este controlador se baseia no cálculo do fluxo óptico horizontal nas duas paredes do corredor, a saber, $\dot{r}_{x1}$ e $\dot{r}_{x2}$. A Figura 7 permite interpretar o seu funcionamento. Note-se que quando o robô estiver sobre a linha média do corredor, e orientado na mesma direção do referido corredor, então $r_{x1} = -r_{x2}$. Para uma posição qualquer do robô no corredor,

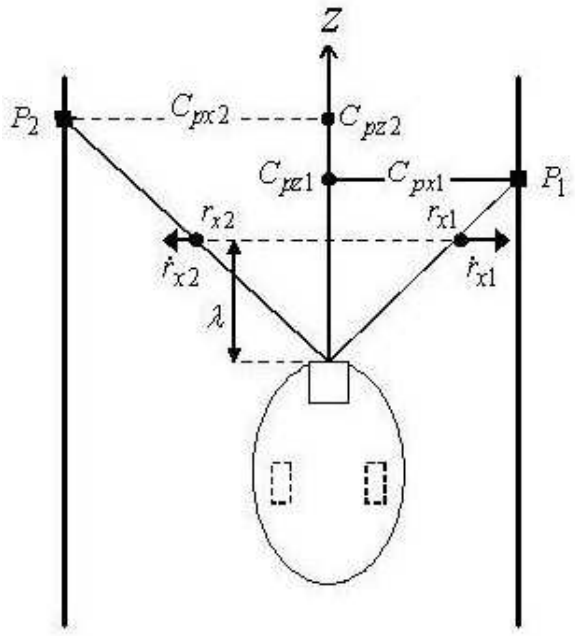

Figura 7: Sistema de controle baseado em fluxo óptico proposto.

tem-se, a partir de (2.17), que

$$\dot{r}_{x1} = \frac{r_{x1} v}{c_{pz1}} - w(\lambda + \frac{r_{x1}^2}{\lambda}) \tag{2.18}$$

e

$$\dot{r}_{x2} = -\frac{r_{x2}v}{c_{pz2}} - w(\lambda + \frac{r_{x2}^2}{\lambda}). \tag{2.19}$$

Observe-se que, neste caso, os vetores correspondentes às componentes horizontais do fluxo óptico, $\dot{r}_{x1}$ e $\dot{r}_{x2}$, são simétricos ($\dot{r}_{x1} = -\dot{r}_{x2}$), o que pode ser confirmado na Figura 8, onde é mostrado o fluxo óptico horizontal calculado nas paredes do corredor quando o robô está na sua linha média.

Figura 8: Componentes horizontais do fluxo óptico medido nas paredes do corredor, quando o robô está no seu centro.

Portanto, para fazer o robô navegar pela linha central do corredor, é necessário igualar os módulos das componentes horizontais do fluxo óptico nas duas paredes, ou seja, variar a velocidade angular w do robô até obter $\dot{r}_{x1} = -\dot{r}_{x2}$. Portanto, para o robô na linha central do corredor, obtém-se obter, das equações (2.18) e (2.19),

$$r_{x1}v(\frac{1}{c_{pz1}} - \frac{1}{c_{pz2}}) = 2w(\lambda + \frac{r_{x1}^2}{\lambda}). \tag{2.20}$$

Se a velocidade angular do robô, neste momento, também for nula ($w = 0$), então, da equação (2.20), obtém-se $c_{pz1} = c_{pz2}$, e daí $c_{px1} = c_{px2}$, por semelhança de triângulos (ver Figura 7), o que quer dizer que o robô está navegando ao longo da linha média do corredor, como desejado. Utilizando a equação (2.17), e substituindo a expressão $r_{x1} = -r_{x2}$ nas

equações (2.18) e (2.19), pode-se obter

$$\begin{bmatrix} \dot{r}_{x1} \\ \dot{r}_{x2} \end{bmatrix} = \begin{bmatrix} \frac{r_{x1}}{c_{pz1}} & -(\lambda + \frac{r_{x1}^2}{\lambda}) \\ \frac{-r_{x2}}{c_{pz2}} & -(\lambda + \frac{r_{x2}^2}{\lambda}) \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}, \tag{2.21}$$

ou seja,

$$\begin{bmatrix} \dot{r}_{x1} \\ \dot{r}_{x2} \end{bmatrix} = J \begin{bmatrix} v \\ w \end{bmatrix}, \tag{2.22}$$

onde J é chamado o Jacobiano do sistema câmara-robô.

Retomando a equação (2.12), e aplicando uma lei de controle de dinâmica inversa (KHALIL, 1996), obtém-se

$$w_R = \frac{1}{k_w a_w}(\eta + b_w \dot{w} + a_w w), \tag{2.23}$$

onde η é definido como

$$\eta = \ddot{w}_d + k_{pw}(w_d - w) + k_{dw}(\dot{w}_d - \dot{w}), \tag{2.24}$$

sendo w_d definida como a velocidade angular desejada, a qual é mantida constante em zero quando o robô percorre a linha média do corredor. Por sua vez, k_{pw} e k_{dw} são ganhos positivos.

A partir da equação (2.22), tem-se que

$$\begin{bmatrix} v \\ w \end{bmatrix} = J^{-1}\dot{r}_x \qquad com \qquad J^{-1} = \{j_{i,j}^{-1}\}, i = 1, 2, j = 1, 2 \tag{2.25}$$

e, a partir daí, se obtém que $w = j_{21}^{-1}\dot{r}_{x1} + j_{22}^{-1}\dot{r}_{x2}$. Substituindo este valor da velocidade angular instantânea do robô, obtido através do fluxo óptico medido, no valor da velocidade angular da equação (2.24), com $w_d = 0$, $\dot{w}_d = 0$ e $\ddot{w}_d = 0$, tem-se que

$$\begin{aligned} w_R = \frac{1}{k_w a_w}[&-k_{pw}(j_{21}^{-1}\dot{r}_1 + j_{22}^{-1}\dot{r}_2) - k_{dw}\dot{w} \\ &+b_w \dot{w} + a_w w]. \end{aligned} \tag{2.26}$$

Combinando agora as equações (2.12) e (2.23), obtém-se que a dinâmica de malha fechada do sistema, para a velocidade angular w do robô, é definida por

$$\ddot{w} + k_{dw}\dot{w} + k_{pw}w = 0, \tag{2.27}$$

o que permite afirmar que o sistema de controle proposto é estável, ou seja, $w \to 0$ quando $t \to \infty$ (CARELLI et al., 2002). Em outras palavras, é possï¿½el levar o robô para a linha média do corredor, e mantê-lo em tal posição com velocidade angular nula, que é o objetivo de controle.

2.6 Controlador Baseado em Ultra-Som

Levando em consideração a equação (2.11) e definindo as variáveis de estado do sistema como $\tilde{x}$ e ϕ, onde $\tilde{x}$ é a distância do robô à linha média do corredor e ϕ é o ângulo de orientação do robô com relação ao eixo do corredor (ver Figura 9), tem-se que

$$\begin{aligned} \dot{\tilde{x}} &= u.sen\varphi \\ \dot{\varphi} &= w, \end{aligned} \tag{2.28}$$

sendo que $\dot{\tilde{x}}$ e $\dot{\varphi}$ são determinados a partir das leituras dos sensores ultra-sônicos $S0$, $S15$, $S7$ e $S8$ (ver Figura 9). Quanto maior a disparidade das leituras dos pares $S0 - S7$ e $S8 - S15$ maior $\tilde{x}$, e quanto maior a disparidade das leituras dos pares $S0 - S15$ e $S7 - S8$ maior o ângulo φ. Assumindo que a velocidade linear v do robô é constante, o objetivo é obter um sinal de controle $w(t)$ tal que as variáveis de estado $\tilde{x}$ e φ tendam assintoticamente para zero, ao mesmo tempo em que o sinal de controle permanece dentro de limites especificados. Para tanto, é proposta a lei de controle (FREIRE, 2002)

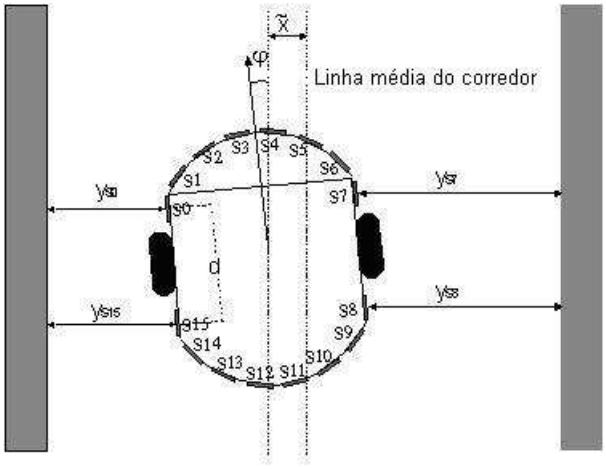

Figura 9: Controlador baseado em ultra-som utilizado (FREIRE, 2002).

$$w = -k_1(\varphi)\varphi - k_2(\tilde{x})\tilde{x}v\frac{sen\varphi}{\varphi}, \tag{2.29}$$

onde $k_1(\varphi)$ e $k_2(\tilde{x})$ são funções positivas adequadamente selecionadas, que devem ser definidas para evitar a saturação da variável de controle w. As equações de malha fechada correspondentes a (2.28) e (2.29) são, então,

$$\begin{aligned} \dot{\varphi} &= -k_1(\varphi)\varphi - k_2(\tilde{x})\tilde{x}v\frac{sen\varphi}{\varphi} \\ \dot{\tilde{x}} &= vsen\varphi, \end{aligned} \tag{2.30}$$

onde as funções $k_1(\varphi)$ e $k_2(\tilde{x})$ são definidas como (KELLY; CARELLI, 1996)

$$k_2(\tilde{x}) = \frac{k_2}{a_2 + \mid \tilde{x} \mid} \quad k_1(\varphi) = \frac{k_1}{a_1 + \mid \varphi \mid}. \tag{2.31}$$

É mostrado em (FREIRE, 2002) que tal controlador atende os objetivos especificados, e por isto ele é aqui utilizado. É importante salientar que tal sistema de controle possui demonstração de estabilidade, a qual também é descrita na tese mencionada.

3 Fusão de Dados

A fusão de dados é um termo usado amplamente para representar qualquer processo que combina informação, proveniente de várias fontes, em um só valor (MURPHY, 2000). Neste capítulo será apresentado, inicialmente, um estudo panorâmico acerca da fusão de dados, e sobre as diversas técnicas utilizadas para realizá-la. Posteriormente, serão focalizadas técnicas de fusão de dados utilizando o filtro de Kalman e o filtro de Kalman Descentralizado (MUTAMBARA, 1998), para cada uma das quais é apresentada a formulação matemática correspondente.

Também dentro deste capítulo são definidas as equações do filtro de informação (MUTAMBARA, 1998) e do filtro de informação descentralizado, esta última versão originalmente proposta em (FREIRE, 2002). Com o fim de complementar os pontos abordados neste capítulo, é feita a aplicação da fusão de dados no cálculo do fluxo óptico, em que se pode perceber que a fusão de dados resulta em uma estimativa do valor do fluxo com uma variância menor que a menor das variâncias dos valores de fluxo introduzidos na máquina da fusão, ou seja, o resultado da fusão é uma estimativa mais confiável do que o dado medido (no caso, o fluxo óptico).

Fechando o capítulo, é introduzida a técnica de fusão de sinais de controle, recentemente proposta como uma arquitetura viável para controlar a navegação de um robô móvel em ambientes estruturados, e que será utilizada neste trabalho para controlar a navegação do robô móvel em um corredor.

3.1 Fusão Sensorial

A fusão sensorial é um método de integrar sinais provenientes de múltiplos sensores, permitindo extrair informação de várias fontes diferentes e integrá-las em um só sinal. Quando os sensores estão todos retornando o mesmo tipo de percepção, eles são considerados redundantes (MURPHY, 2000). Um exemplo de redundância física é dado por um robô que tem dois anéis de sonares, um posicionado na sua parte superior e outro na sua

parte inferior. O software do sonar retorna, por exemplo, a leitura da distância mínima a um objeto. Assim, a leitura é feita por cada um dos dois anéis, fornecendo uma leitura mais confiável no caso de objetos de menor tamanho, que, em circunstâncias normais, provavelmente refletiriam o sinal acústico dos sensores colocados na parte superior, distorcendo sua medida. Os sensores podem ser também logicamente redundantes, quando eles retornam percepções idênticas, mas usam algoritmos e formas diferentes de processar a informação. Um exemplo deste tipo de sensores está naquelas aplicações onde o robô extrai imagens a partir das informações provenientes de sensores de visão e sensores laser.

Algumas vezes sensores redundantes são chamados também sensores competitivos. Por outro lado, sensores complementares provêem informação de diferentes tipos com relação a uma percepção do ambiente. Por exemplo numa fusão sensorial hipotética em missões de busca e resgate, um robô pode procurar por sobreviventes fusionando observações provenientes de um sensor piroelétrico que detecta a presença humana e uma câmara capaz de detectar movimento. O que se quer detetar é a mesma coisa, a presença humana, sendo usados dois tipos de sensores, um térmico e outro visual.

Os algoritmos mais comunmemte adotados para realizar a fusão sensorial podem ser classificados em três diferentes grupos (SASIADEK, 2004):

- o primeiro grupo corresponde à fusão baseada em modelos probabilísticos. Entre os modelos probabilísticos, temos os métodos bayesianos, a teoria da evidência e operadores recursivos, por exemplo;
- o segundo corresponde à fusão baseada no método de mínimos quadrados, e entre as técnicas de mínimos quadrados podemos mencionar o filtro de Kalman e a teoria de otimização, por exemplo;
- o terceiro corresponde à fusão baseada em técnicas de sistemas inteligentes, entre as quai temos métodos baseados em lógica nebulosa, redes neurais e algoritmos genéticos.

3.2 O Filtro de Kalman

As medições experimentais nunca são perfeitas, mesmo utilizando instrumentos modernos e sofisticados. Assim, o problema de estimar o estado de um sistema dinâmico estocástico a partir de observações ruidosas é de importância central na Engenharia, bem como a filtragem de ruído é uma parte importante do processamento de uma seqüência de

amostras de um sinal real. Existem muitas classes de filtros que podem ser utilizados para propósitos de estimação, como o filtro de média, o filtro de mediana, o filtro gaussiano, o filtro de Kalman, etc (SASIADEK; KHE, 2001). Neste trabalho, será utilizado o filtro de Kalman (Figura 10) e suas variações.

Existem dois processos que são modelados pelo filtro de Kalman. O primeiro processo é um modelo descrevendo como o vetor de erro de estado muda no tempo. Este modelo é o modelo dinâmico do sistema. O segundo modelo define a relação entre o erro do vetor de estado e qualquer medida processada pelo filtro, que é o modelo de medida. Intuitivamente, o filtro de Kalman classifica informação e pesos da contribuição relativa das medidas e da dinâmica do comportamento do vetor de estado. As medidas e o vetor de estado são quantificadas por suas correspondentes matrizes de covariância.

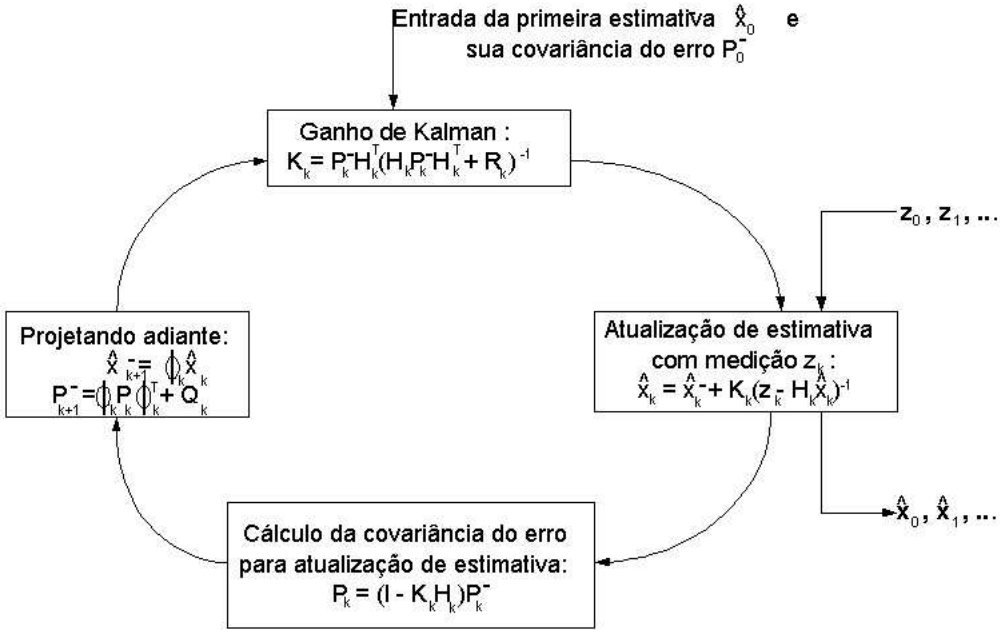

Figura 10: O filtro de Kalman (BROWN; HWANG, 1997).

Na caracterização do filtro de Kalman, é assumido que o processo aleatório a ser estimado pode ser modelado na forma

$$\mathbf{x}_{k+1} = \mathbf{\Phi}_k \mathbf{x}_k + \mathbf{w}_k, \tag{3.1}$$

e que as observações (medições) do processo, que acontecem em pontos discretos no tempo, são descritas de acordo com a relação linear

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k, \tag{3.2}$$

onde $\mathbf{x}_k$ é o vetor de estado $(n \times 1)$ do processo no tempo t_k, $\boldsymbol{\phi_k}$ é a matriz $(n \times n)$ de transição de estados, $\mathbf{z}_k$ é o vetor de observações $(m \times 1)$ no instante t_k, e $\mathbf{H}_k$ é a matriz de observação $(m \times n)$. O vetor $(m \times 1)$ $\mathbf{w}(k)$ consiste em uma seqüência de ruído gaussiano com covariância conhecida, e $\mathbf{v}(k)$ é um vetor $(m \times 1)$ representando o erro de medição, que é também uma seqüência de ruído gaussiano com covariância conhecida. As matrizes de covariância para os vetores $\mathbf{w}_k$ e $\mathbf{v}_k$, são dadas por

$$E[\mathbf{w}_k \mathbf{w}_i^T] = \begin{cases} \mathbf{Q}_k, & i = k \\ 0, & i \neq k, \end{cases} \tag{3.3}$$

$$E[\mathbf{v}_k \mathbf{v}_i^T] = \begin{cases} \mathbf{R}_k, & i = k \\ 0, & i \neq k, \end{cases} \tag{3.4}$$

e

$$E[\mathbf{w}_k \mathbf{v}_i^T] = 0, \tag{3.5}$$

para todo k e i.

As equações utilizadas para o cálculo do filtro de Kalman são, para a predição,

$$\hat{\mathbf{x}}_{k+1}^- = \boldsymbol{\Phi}_k \hat{\mathbf{x}}_k \tag{3.6}$$

e

$$\mathbf{P}_{k+1}^- = \boldsymbol{\Phi}_k \mathbf{P}_k \boldsymbol{\Phi}_k^T + \mathbf{Q}_k, \tag{3.7}$$

onde $\hat{\mathbf{x}}_{k+1}^-$ e $\mathbf{P}_{k+1}^-$ são a predição de $\hat{\mathbf{x}}_k^-$ e $\mathbf{P}_k^-$, respectivamente, dadas as observações feitas até o instante k. Para a estimação, temos que

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1}, \tag{3.8}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{z}_k + \mathbf{H}_k \hat{\mathbf{x}}_k^-) \tag{3.9}$$

e

$$\mathbf{P}_k = (I - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^- \tag{3.10}$$

onde $\mathbf{K}_k$ é o ganho de Kalman, $\mathbf{P}_k$ é a covariância do erro estimado e $\hat{\mathbf{x}}_k$ é o estado estimado, dadas as observações feitas até o instante k.

3.3 O Filtro de Kalman Descentralizado

No caso do filtro de Kalman, todas as entradas são direcionadas a um só filtro. Este modo de operação é usualmente chamado filtro de Kalman centralizado, mas existem aplicações nas quais um filtro centralizado não pode ser implementado (BROWN; HWANG, 1997), utilizando-se como alternativa, nestes casos, arquiteturas descentralizadas. Neste caso, tem-se n filtros locais que se comunicam com um filtro global. Para o i-ésimo filtro local temos que

$$\hat{\mathbf{x}}_i = \mathbf{P}_i(\mathbf{M}^{-1}\mathbf{m} + \mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{z}_i) \tag{3.11}$$

e

$$\mathbf{P}_i^{-1} = \mathbf{M}^{-1} + \mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{z}_i, \tag{3.12}$$

onde $\mathbf{m}$ é o vetor de estados estimado e $\mathbf{M}$ é a matriz de covariância associada a tal vetor de estados, ambos os termos num instante anterior ao atual. Da mesma forma, para o filtro global temos que

$$\mathbf{P}^{-1} = \sum_{i=1}^{n} \mathbf{P}_i^{-1} - (n-1)\mathbf{M} \tag{3.13}$$

e que

$$\hat{\mathbf{x}} = \mathbf{P}[\sum_{i=1}^{n} \mathbf{P}_i^{-1}\hat{\mathbf{x}}_i - (n-1)\mathbf{M}^{-1}\mathbf{m}]. \tag{3.14}$$

3.4 O Filtro de Informação

O filtro de informação é basicamente um filtro de Kalman expresso em termos de medidas de informação sobre os estados de interesse, ao invés de estimativas dos estados e suas covariâncias associadas (MUTAMBARA, 1998). Com base nisso, transcrevemos as equações do filtro de informação definidas em (FREIRE, 2002):

Matriz de informação:

$$\mathbf{Y}(k) = \mathbf{P}^{-1}(k). \tag{3.15}$$

Vetor de informação de estado:

$$\hat{\mathbf{y}}(k) = \mathbf{P}^{-1}(k)\hat{\mathbf{x}}(k) = \mathbf{Y}(k)\hat{\mathbf{x}}(k) \tag{3.16}$$

e

$$\mathbf{Y}(k) = [\phi(k)\mathbf{Y}^{-1}(k-1)\phi^T(k) + \mathbf{Q}(k)]^{-1}. \tag{3.17}$$

Estimação:

$$\hat{\mathbf{y}}(k) = \hat{\mathbf{y}}(k \mid k-1) + \mathbf{i}(k) \tag{3.18}$$

e

$$\mathbf{Y}(k) = \mathbf{Y}(k \mid k-1) + \mathbf{I}(k), \tag{3.19}$$

onde

$$\mathbf{L}(k) = \mathbf{Y}(k)\phi(k)\mathbf{Y}^{-1}(k-1),$$

$$\mathbf{i}(k) = \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{z}(k), \tag{3.20}$$

e

$$\mathbf{I}(k) = \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{H}(k) \tag{3.21}$$

são o coeficiente de propagação da informação, a contribuição da informação do estado e a matriz de informação associadaa cada estado, respectivamente.

3.5 O Filtro de Informação Descentralizado

O filtro de informação é um equivalente algébrico do Filtro de Kalman (MUTAMBARA, 1998), com a diferença que ele é mais rápido e sua inicialização é mais fácil (inicializa com zero), simples e segura, sobretudo para sistemas não-lineares, como aquele aqui considerado. Por estas razões é que o filtro de informação descentralizado foi o escolhido para fazer a fusão dos dois controladores utilizados neste trabalho, assim como em (FREIRE, 2002). Um diagrama de blocos descritivo do referido filtro pode ser visto na Figura 11. As equações que o governam se distribuem em dois grupos: as equações dos filtros locais e as equações do filtro global. As equações utilizadas para os filtros locais são

$$\hat{\mathbf{y}}_i(k) = \hat{\mathbf{y}}(k-1) + \mathbf{i}_i(k) \tag{3.22}$$

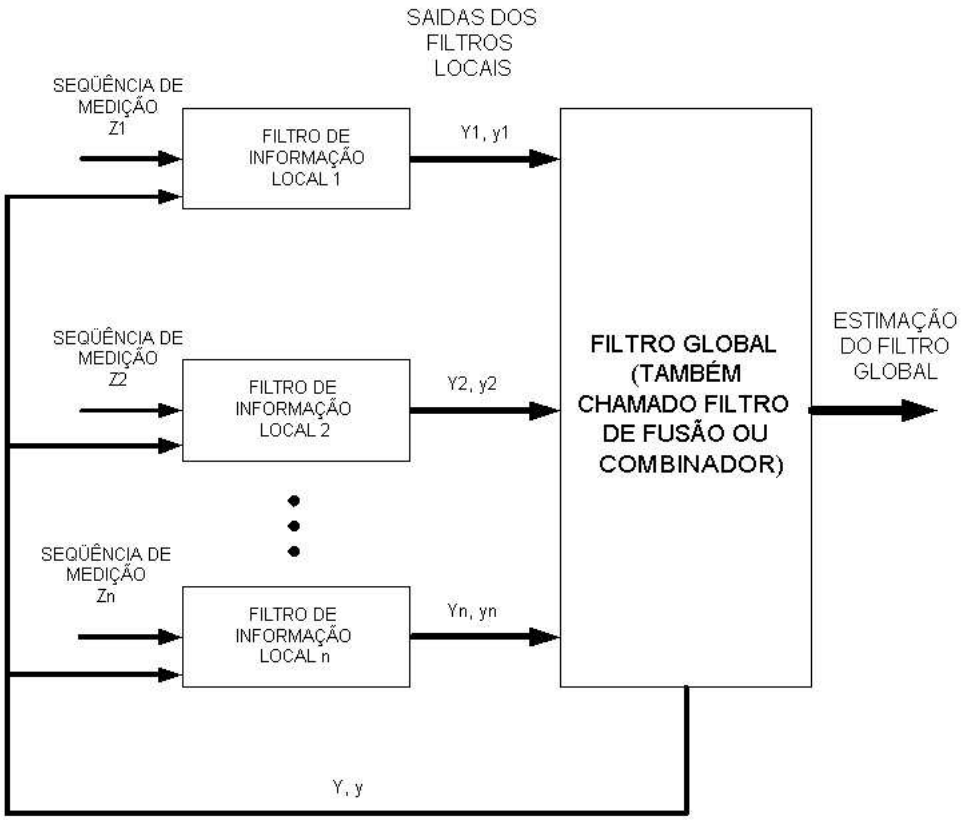

Figura 11: O filtro de informação descentralizado.

e

$$\hat{\mathbf{Y}}_i(k) = \hat{\mathbf{Y}}(k-1) + \mathbf{I}_i(k), \tag{3.23}$$

onde $\hat{\mathbf{Y}}_i$ é a matriz de informação local, $\hat{\mathbf{y}}_i$ é o vetor de informação local, $\hat{\mathbf{Y}}$ é a matriz de informação global e $\hat{\mathbf{y}}$ é o vetor de informação global. Para o filtro global, tem-se que

$$\hat{\mathbf{y}}(k) = \sum_{i}^{n} \hat{\mathbf{y}}_i(k) - (n-1)\hat{\mathbf{y}}(k-1) \tag{3.24}$$

e

$$\hat{\mathbf{Y}}(k) = \sum_{i}^{n} \mathbf{Y}_i(k) - (n-1)\mathbf{Y}(k-1), \tag{3.25}$$

onde k indica o instante de tempo considerado e n é o número de filtros locais.

3.6 Aplicação para o Cálculo de Fluxo Óptico

Com o objetivo de ilustrar a potencialidade da fusão de dados e dos algoritmos de estimação como o filtro de Kalman, utilizaremos a técnica para o cálculo de fluxo óptico proposta em (SORIA; CARELLI; SARCINELLI-FILHO, 2003). Esta técnica calcula o fluxo

óptico de duas imagens, utilizando a técnica de mínimos quadrados (LUKAS; KANADE, 1981) numa região de 20×20 pixels, da seguinte forma: tal região é dividida em 16 regiões de 5×5 pixels e a técnica de mínimos quadrados é utilizada em cada uma destas regiões, obtendo-se, ao final, 16 valores de fluxo, com suas respectivas variâncias, que são, então, consideradas como 16 estimativas do mesmo valor de fluxo (ver seção 2.3). Esta técnica é ilustrada na Figura 12, onde podemos observar a divisão da imagem em regiões para o cálculo do fluxo óptico em cada região, e a subdivisão de cada região em 16 regiões menores, a variância associada aos valores de fluxo é cálculada utilizando a equação (2.10), e naquelas regiões onde o valor de fluxo é igual a zero, utilizamos um valor constante da variância (neste exemplo 125), que é um valor numérico alto que expressa a incerteza da região.

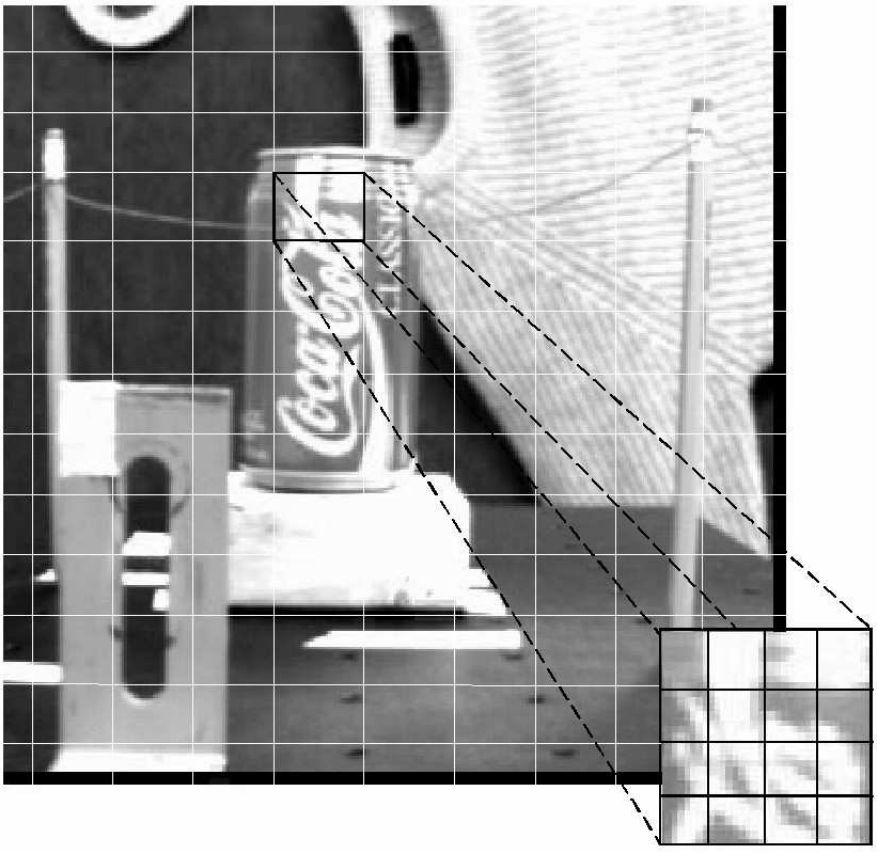

Figura 12: Divisão da imagem em subregiões para o cálculo do fluxo óptico.

Na Tabela 1 podemos observar as dezesseis estimativas do mesmo valor de fluxo, com suas respectivas variâncias. Tais valores de fluxo, assim como suas variâncias, são inseridos em uma máquina de fusão de dados, que neste caso é um filtro de Kalman descentralizado, obtendo como saída, tal como se pode ver na Tabela 2, um só valor de fluxo e uma só variância. É importante observar que o valor da variância obtida na saída do algoritmo

de fusão é menor que a menor das variâncias inseridas no filtro (Tabelas 1 e 2). Assim, a fusão de dados melhora a estimativa do fluxo, o que é constatado pelo fato de se obter uma estimativa de fluxo com variância menor.

O fato, neste exemplo, é que cada uma das 16 sub-regiões em que se produziu uma estimativa de fluxo pode ser interpretada como a informação proveniente de um sensor visual diferente, e a constatação é que, de fato, a inserção de mais sensores com característica de redundância sempre melhora a percepção do mundo (HONG, 1999).

Tabela 1: Valores do fluxo óptico e suas variâncias.

Fluxo óptico	**Variância**
0,2834	1,6591
0,2026	2,7444
0,1527	2,6940
0	125,0000
0,2093	6,1890
0,1634	1,3950
0,0716	2,1727
0	125,0000
0,1023	1,4112
0,1479	5,2418
-0,3947	125,0000
0	5,0480
0,1213	0,8270
-0,2922	4,0241
0,2215	3,4557
0	125,0000

Tabela 2: Valor do fluxo óptico resultante da fusão e sua variância

Fluxo óptico	**Variância**
0,1163	0,0180

3.7 Fusão de Sinais de Controle

A partir do resultado do exemplo apresentado na seção anterior, percebe-se que o uso da redundância de sensores é importante para se obter uma informação sensorial mais fidedigna. Um exemplo extremo da importância de tal redundância seria no caso em que um sensor sofre pane. Havendo outra fonte sensorial, o sistema pode continuar operando, ainda que obtendo dados sensoriais de pior qualidade, o que não poderia ocorrer no caso de se usar um só sensor. Porém, quando os sensores usados são complementares, o que

é comum em robótica móvel (por exemplo, um robô móvel pode ser dotado de sonar e câmara de vídeo), é difícil realizar a fusão, já que a informação desejada está codificada em sinais totalmente distintos. Nestes casos, foi proposto em (FREIRE, 2002) um paradigma de fusão denominado fusão de sinais de controle, em que distintos controladores usam a informação proveniente dos diversos sensores para gerar diversas estimativas do sinal de controle a ser entregue aos atuadores do robô. Assim, uma máquina de fusão pode ser utilizada para gerar o sinal de controle efetivo a ser enviado aos atuadores, sinal de controle este que é uma estimativa bem mais adequada à situação reportada pelos sensores do que aquela gerada por um só dos controladores.

Tal paradigma de fusão de dados será utilizado para implementar a estrutura de dois controladores redundantes para guiar o robô ao longo de um corredor, que é o que este trabalho se propõe implementar. O sistema proposto, então, é aquele mostrado na Figura 13, em que o controlador 1 usa o fluxo óptico calculado a partir das duas imagens adquiridas em seqüência pela mesma câmara (instalada a bordo do robô) e o controlador 2 usa as distâncias medidas por quatro sonares localizados nos lados do robô (ver Figura 9). Note-se que estes são os dois controladores estáveis discutidos no Capítulo 2. Note-se,

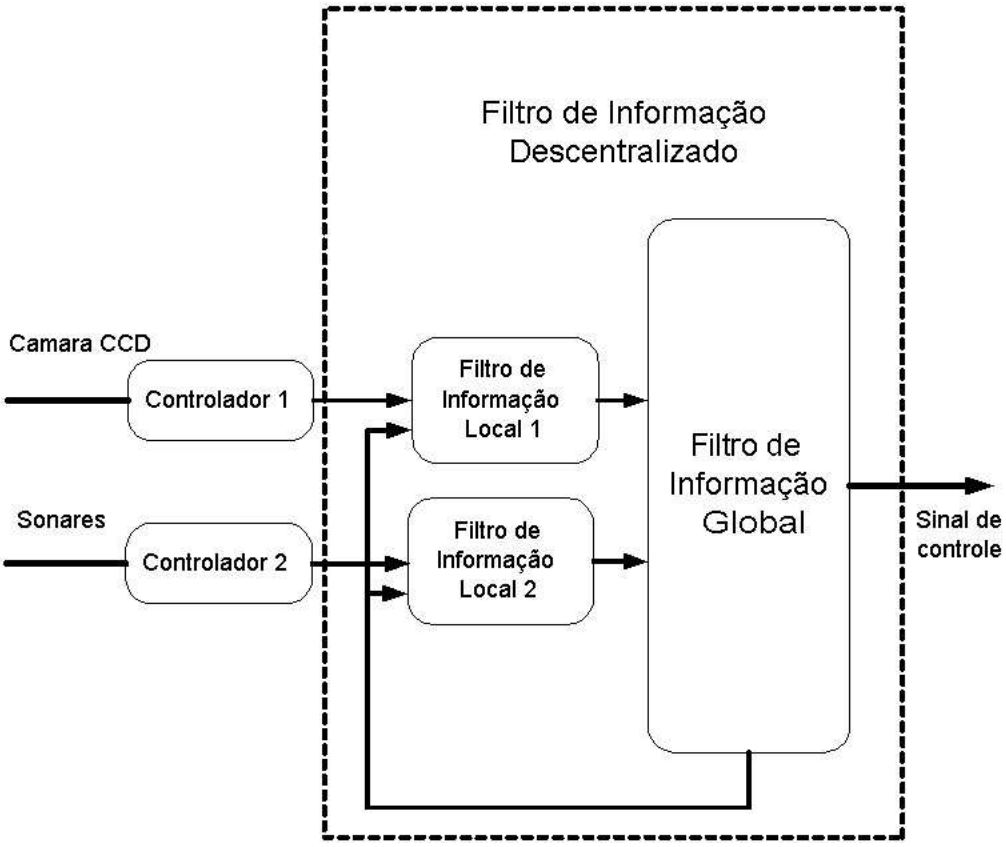

Figura 13: O esquema de controle adotado.

também que a máquina de fusão adotada é o filtro de informação descentralizado, pelas vantagens que ele apresenta sobre o filtro de Kalman descentralizado, como discutido em (FREIRE, 2002).

É importante salientar que exite também uma demonstração formal de que a fusão das saídas de controle de controladores redundantes produz um resultado melhor do que o que seria obtido usando cada controlador separado, demonstração esta que pode ser encomtrada em (FREIRE, 2002).

Na seqüência, o próximo capítulo trata da implementação do sistema de controle proposto a bordo do robô Pioneer 2-DX, assim como da sua experimentação.

4 *Resultados Experimentais*

Para a implementação da arquitetura proposta neste trabalho foi feito um programa em linguagem C em sistema operacional Linux. Este capítulo faz uma breve descrição da plataforma de *"hardware"* utilizada, mostra os resultados do cálculo de fluxo óptico em corredores, para poder analisar a factibilidade do seu uso na navegação de robôs, propõe um novo algoritmo para o cálculo de fluxo óptico, o qual é utilizado nos experimentos, e, finalmente, apresenta os resultados obtidos com o robô no processo de navegação em corredores.

4.1 Descrição do *"Hardware"*

Para validar o funcionamento dos controladores individuais e da máquina de fusão discutidos nos Capítulos 2 e 3, a arquitetura de controle adotada foi implementada num robô Pioneer 2DX, o qual possui um computador embutido, com processador Intel Pentium MMX de frequência de relógio de 266MHz, um sistema de visão constituído por um *frame grabber* Imagenation PXC200 e uma câmara PTZ Sony D30, além de 16 sensores ultra-sônicos, dos quais apenas os dois pares localizados nas laterais foram utilizados. Tal equipamento pode ser visto na Figura 14.

4.2 Movimento no Corredor

Este experimento tem por objetivo verificar a viabilidade de se usar a estratégia proposta no Capítulo 2 para a navegação do robô no corredor, usando fluxo óptico. Para isto o robô é transladado entre duas madeiras que simulam as paredes do corredor, calculando-se o fluxo óptico à medida que ele se move. Foram realizados três percursos do robô, que abordam três situações diferentes. No primeiro percurso o robô navega perto da parede esquerda, no segundo percurso ele navega perto da parede direita, e no último ele navega pelo meio do corredor, ou seja, no centro da região entre as duas paredes.

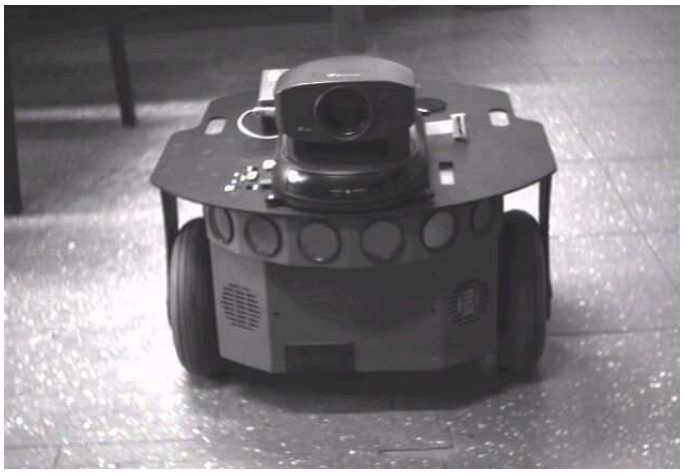

Figura 14: O robô usado nos experimentos.

O fluxo foi medido utilizando imagens de 240×320 pixels, nas quais o fluxo óptico é calculado em janelas de 140×20 pixels, tomados em posições eqüidistantes do centro da imagem. As madeiras usadas para construir os corredores tinham uma altura de 75 cm, e a separação entre elas era de 90 cm.

4.2.1 Movimento do robô perto da parede esquerda

Neste experimento o robô é posicionado mais perto da parede esquerda (Figura 15), obtendo-se os seguintes vetores de fluxo: $(-0.8574, 1.0257)$ na janela esquerda e $(0.4426, 0.0021)$ na janela direita. O método usado para o cálculo de fluxo será discutido na próxima seção, bastando, por ora, considerar tais valores. Então, a diferença entre as componentes horizontais de tais vetores é de (-0.4148), o que implica a ação de controle deverá ser fazer com que o robô vire para o lado direito, proporcionalmente a esta diferença.

4.2.2 Movimento do robô perto da parede direita

Neste novo percurso o robô é posicionado mais perto da parede direita (Figura 16), obtendo-se os seguintes vetores de fluxo: $(-0.5621, 0.2853)$ na janela esquerda e $(0.9933, 0.1257)$ na janela direita. Isto resulta na diferença de (0.4312) entre as componentes horizontais de tais vetores o que implica a ação de controle agora deverá fazer com que o robô gire para o lado esquerdo, proporcionalmente a esta diferença de fluxos.

Figura 15: Experimento de movimento no corredor com o robô perto da parede esquerda.

Figura 16: Experimento de movimento no corredor com o robô perto da parede direita.

Figura 17: Experimento de movimento em corredor com o robô no centro.

4.2.3 Movimento do robô no centro do corredor

Neste novo experimento o robô é posicionado no meio das duas paredes (Figura 17), obtendo-se os seguintes vetores de fluxo: $(-0.2835, -0.2061)$ na janela esquerda e $(0.3547, 0.5422)$ na janela direita, resultando na diferença de (0.0712) entre as componentes horizontais de tais vetores, a qual é uma diferença muito pequena (só em condições ideais poderíamos conseguir uma diferença igual a 0 estando o robô no meio). Como conseqüência, o robô deve continuar sua trajetória atual, sem executar nenhuma ação de controle.

4.3 O Algoritmo Proposto para Cálculo de Fluxo Óptico

O novo método de cálculo do fluxo óptico aqui proposto resulta da utilização do método de mínimos quadrados para obter várias estimativas do vetor de fluxo óptico correspondente a uma dada região da imagem, seguida da utilização do filtro de informação descentralizado para obter uma única estimativa a partir das diversas estimativas disponíveis. O algoritmo assim descrito foi denominado algoritmo de mínimos quadrados modificado com filtro de informação descentralizado.

Aqui note-se que a fusão de dados assim realizada resultará numa estimativa final melhor do que a melhor das estimativas fornecidas ao filtro de informação descentralizado, conforme discutido no Capítulo 3.

Embora a idéia básica de usar fusão de dados para obter uma estimativa de melhor qualidade já tenha sido utilizada em (SORIA; CARELLI; SARCINELLI-FILHO, 2003), a proposta aqui apresentada possui duas diferenças em relação àquela, ambas lhe propiciando maior rapidez de cálculo, como se verá na seqüência desta seção. A primeira diferença é o uso do filtro de informação descentralizado como algoritmo de fusão de dados, ao invés do filtro de Kalman descentralizado. Como o filtro de informação descentralizado possui equações mais simples, o tempo para computar a estimativa final do fluxo é menor. A segunda diferença é que cada estimativa usada como entrada do algoritmo de fusão usa menos pixels da imagem como observação, sendo, por isto, calculadas com maior rapidez. Neste caso, pelo menor número de observações, a qualidade das estimativas fornecidas ao filtro de informação descentralizado é pior, mas a etapa de fusão se incumbe de gerar uma estimativa final de melhor qualidade.

Assim como em (SORIA; CARELLI; SARCINELLI-FILHO, 2003), a implementação aqui realizada consiste em dividir a imagem em regiões de 20 por 20 pixels, nas quais se supõe o fluxo constante (LUKAS; KANADE, 1981), sendo cada uma destas regiões dividida em subregiões de 5 por 5 pixels (Figura 12, Capítulo 3). Aqui, porém, dentro de cada uma destas subregiões o número de pixels a ser tomado não será 25, tal como em (SORIA; CARELLI; SARCINELLI-FILHO, 2003). Com o objetivo de diminuir o tempo de cálculo, tomar-se-á somente 4 pixels de cada subregião de 5 por 5 pixels, obtendo-se, finalmente, um total de 64 pixels em cada região de 20 por 20 pixels, em lugar dos 400 pixels possíveis.

Com os 4 pixels, selecionados aleatoriamente em cada subregião de 5 por 5 pixels, se calcula o fluxo óptico em tal subregião, utilizando o método de mínimos quadrados. Os valores de fluxo das 16 subregiões são introduzidos em um filtro de informação descentralizado, com a finalidade de melhorar a qualidade da estimativa final, a partir das 16 estimativas disponíveis, e assim poder obter um fluxo mais confiável para a região considerada.

Na seqüência, são apresentadas as equações do filtro de Kalman e do filtro de informação descentralizado para o caso do fluxo óptico. Seja

$$z = -I_t, \tag{4.1}$$

$$R = \sigma^2, \tag{4.2}$$

$$\mathbf{H}_k = [I_x I_y], \tag{4.3}$$

$$\mathbf{x} = \begin{bmatrix} u \\ v \end{bmatrix}, \tag{4.4}$$

e

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix}. \tag{4.5}$$

Voltando a escrever as equações (3.1) e (3.2), obtemos

$$\mathbf{x}_{k+1} = \mathbf{\Phi}_k \mathbf{x}_k + \mathbf{w}_k$$

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k.$$

De (BROWN; HWANG, 1997) pegamos a equação

$$\mathbf{P}_k^{-1} = (\mathbf{P}_k^-)^{-1} + \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{H}_k, \tag{4.6}$$

que é outra forma de expressar a covariância inversa, onde $\mathbf{P}_k^-$ é o valor da covariância anterior ao valor da covariância atual $\mathbf{P}_k$, onde

$$\mathbf{P}_k^- = \begin{bmatrix} P_{11}^- & P_{22}^- \\ P_{33}^- & P_{44}^- \end{bmatrix},$$

tal que

$$det(P_k^-) = P_{11}^- P_{44}^- - P_{22}^- P_{33}^-. \tag{4.7}$$

Substituindo as equações (4.1), (4.2), (4.3) e (4.7) na equação (4.6) obtemos a covariância inversa

$$\mathbf{P}(k)^{-1} = \begin{bmatrix} \frac{P_{44}^-}{det(\mathbf{P}_k^-)} + \frac{I_x^2}{\sigma^2} & \frac{-P_{22}^-}{det(\mathbf{P}_k^-)} + \frac{I_x I_y}{\sigma^2} \\ \frac{-P_{33}^-}{det(\mathbf{P}_k^-)} + \frac{I_x I_y}{\sigma^2} & \frac{P_{11}^-}{det(\mathbf{P}_k^-)} + \frac{I_y^2}{\sigma^2} \end{bmatrix}. \tag{4.8}$$

Voltando a escrever a equação (3.15), ou seja,

$$\mathbf{Y}(k) = \mathbf{P}_k^{-1},$$

obtemos

$$\mathbf{Y}(k) = \begin{bmatrix} \frac{P_{44}^-}{det(\mathbf{P}_k^-)} + \frac{I_x^2}{\sigma^2} & \frac{-P_{22}^-}{det(\mathbf{P}_k^-)} + \frac{I_x I_y}{\sigma^2} \\ \frac{-P_{33}^-}{det(\mathbf{P}_k^-)} + \frac{I_x I_y}{\sigma^2} & \frac{P_{11}^-}{det(\mathbf{P}_k^-)} + \frac{I_y^2}{\sigma^2} \end{bmatrix}, \tag{4.9}$$

e escrevendo novamente a equação (3.16) temos que

$$\hat{\mathbf{y}}(k) = \mathbf{P}_k^{-1}\hat{\mathbf{x}}(k) = \mathbf{Y}(k)\hat{\mathbf{x}}(k).$$

Agora, substituindo as equações (4.5) e (4.9) na equação anterior, obtemos que

$$\hat{\mathbf{y}}(k) = \begin{bmatrix} \frac{P_{44}^-}{det(\mathbf{P}_k^-)} + \frac{I_x^2}{\sigma^2} & \frac{-P_{22}^-}{det(\mathbf{P}_k^-)} + \frac{I_x I_y}{\sigma^2} \\ \frac{-P_{33}^-}{det(\mathbf{P}_k^-)} + \frac{I_x I_y}{\sigma^2} & \frac{P_{11}^-}{det(\mathbf{P}_k^-)} + \frac{I_y^2}{\sigma^2} \end{bmatrix} \begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix},$$

donde, finalmente, obtemos a equação

$$\hat{\mathbf{y}}(k) = \begin{bmatrix} (\frac{P_{44}^-}{det(\mathbf{P}_k^-)} + \frac{I_x^2}{\sigma^2})\hat{u} & (\frac{-P_{22}^-}{det(\mathbf{P}_k^-)} + \frac{I_x I_y}{\sigma^2})\hat{v} \\ (\frac{-P_{33}^-}{det(\mathbf{P}_k^-)} + \frac{I_x I_y}{\sigma^2})\hat{u} & (\frac{P_{11}^-}{det(\mathbf{P}_k^-)} + \frac{I_y^2}{\sigma^2})\hat{v} \end{bmatrix}, \tag{4.10}$$

que representa formulação do filtro de informação relativa ao fluxo óptico.

4.3.1 Aplicação do algoritmo proposto

Foram utilizadas as seqüências clássicas de imagens "NASA", "rubic"e "táxi de Hamburgo", para comparar os desempenhos do algoritmo de mínimos quadrados modificado com filtro de informação descentralizado, aqui proposto, e do algoritmo de mínimos quadrados com fusão (SORIA; CARELLI; SARCINELLI-FILHO, 2003). Tais seqüências estão representadas nas Figuras 18, 19 e 20, pelo seu primeiro quadro de imagem. O resultado de tal comparação está nas Figuras 21, 22 e 23, que mostram as variâncias das estimativas de fluxo obtidas para as seqüências das Figuras 18, 19 e 20, respectivamente.

Em todos os casos percebe-se que a variância associada ao método aqui proposto é ligeiramente maior que aquela correspondente ao método proposto em (SORIA; CARELLI; SARCINELLI-FILHO, 2003), o que se deve ao fato de que poucos pixels em cada subregião são usados para obter as estimativas iniciais do fluxo. Entretanto, vê-se claramente que a diferença entre as variâncias não é comprometedora.

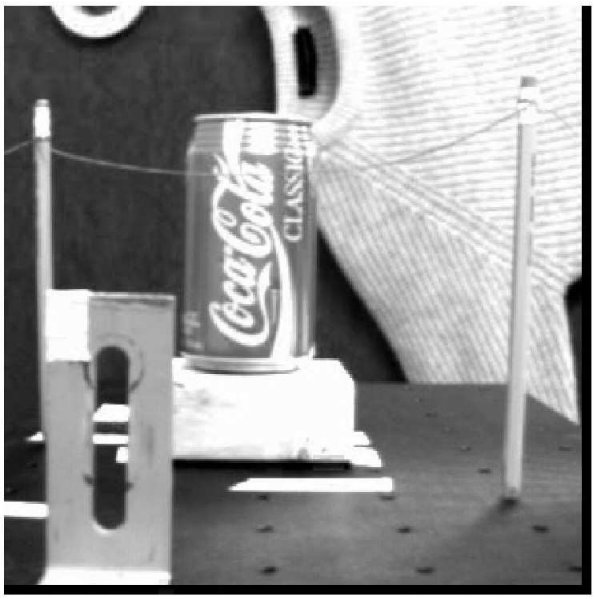

Figura 18: Seqüência de imagens "NASA".

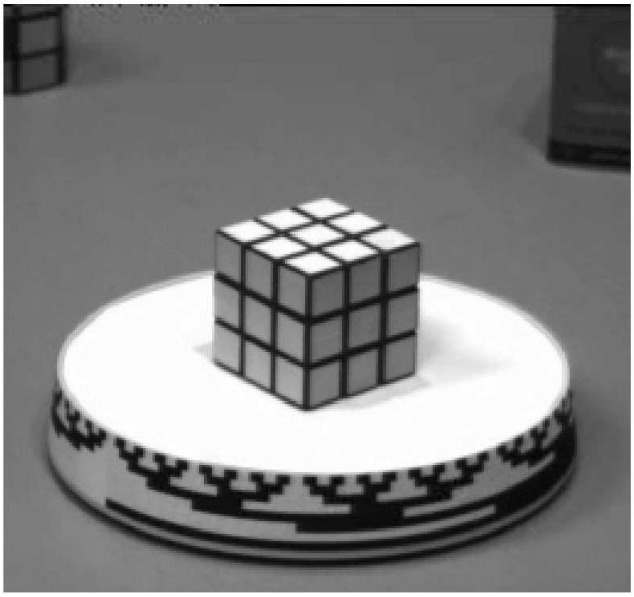

Figura 19: Seqüência de imagens "rubic".

4.3.2 Complexidade computacional do algorimo proposto

Continuando a comparação, implementou-se os dois algoritmos em Matlab, e determinouse o número de *flops* necessários para calcular a estimativa final do fluxo, em cada caso, usando a seqüência da Figura 18. Tais dados são mostrados na Tabela 3. Vê-se, ali, que o algoritmo aqui proposto tem a vantagem de ser computacionalmente bem menos complexo, exigindo, assim, menor tempo para seu processamento. Note-se que, mais uma

Figura 20: Seqüência de imagens "Táxi de Hamburgo".

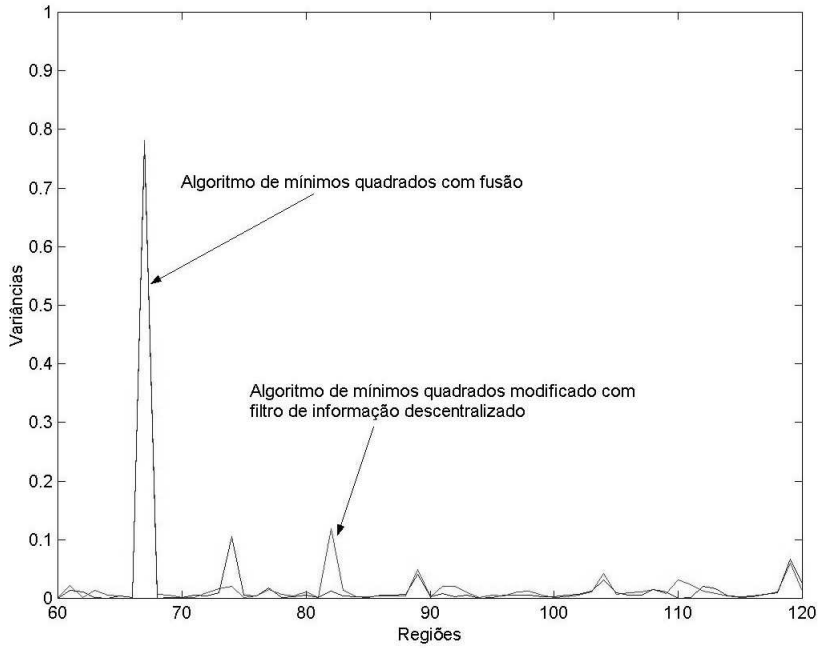

Figura 21: Variâncias resultantes do algoritmo de mínimos quadrados com fusão (SORIA; CARELLI; SARCINELLI-FILHO, 2003) e do algorimo de mínimos quadrados modificado com filtro de informação descentralizado para a seqüência de imagens "NASA".

vez, a principal causa para esta diferença é a utilização de poucos pixels para obter as estimativas iniciais do fluxo.

Para concluir a comparação, ambos os algoritmos foram implementados no computador de bordo do robô Pioneer 2-DX da ActivMedia, trabalhando com o sistema operacional Linux. Então, usando imagens capturadas pelo robô em tempo real, o tempo de cálculo

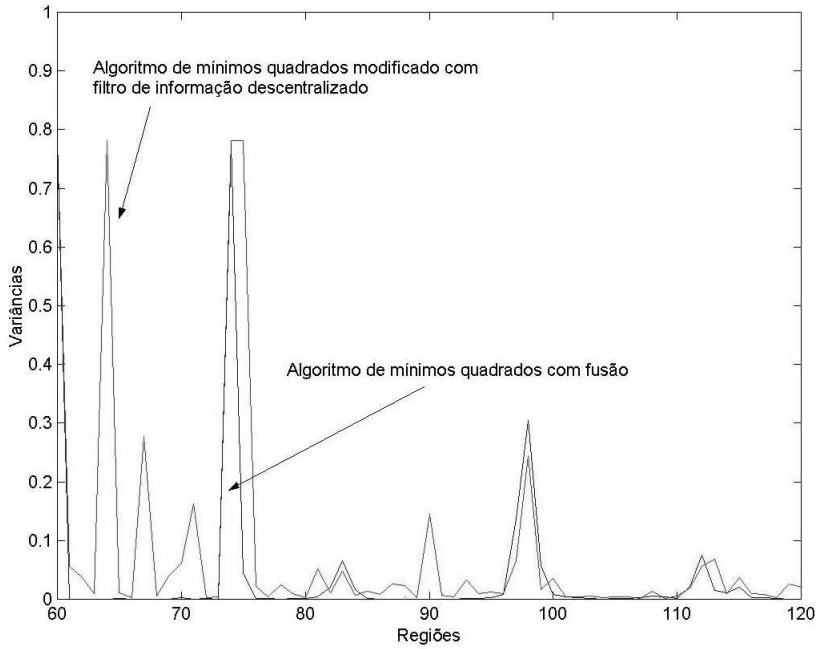

Figura 22: Variâncias resultantes do algoritmo de mínimos quadrados com fusão (SORIA; CARELLI; SARCINELLI-FILHO, 2003) e do algorimo de mínimos quadrados modificado com filtro de informação descentralizado para a seqüência de imagens "rubic".

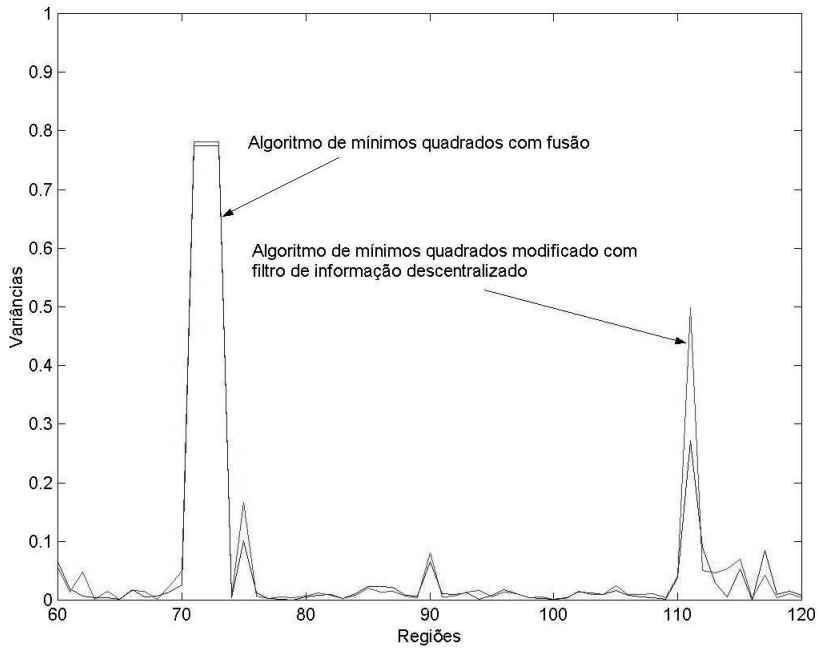

Figura 23: Variâncias resultantes do algoritmo de mínimos quadrados com fusão (SORIA; CARELLI; SARCINELLI-FILHO, 2003) e do algorimo de mínimos quadrados modificado com filtro de informação descentralizado para a seqência de imagens "táxi de Hamburgo".

Tabela 3: Comparação do esforço computacional para algoritmos de cálculo do fluxo óptico.

Algoritmo	**Esforço computacional (flops)**
Mínimos quadrados com fusão (SORIA; CARELLI; SARCINELLI-FILHO, 2003)	12.829.310
Mínimos quadrados modificado com filtro de informação descentralizado	7.766.405

Tabela 4: Comparação de tempos de execução de algoritmos de cálculo de fluxo a bordo de um robô móvel.

Algoritmo	**Computação**
Mínimos quadrados com fusão (SORIA; CARELLI; SARCINELLI-FILHO, 2003)	60 ms
Mínimos quadrados modificado com filtro de informação descentralizado	20 ms

dos algoritmos foi medido, resultando nos dados da Tabela 4. Mais uma vez, nota-se a vantagem do algoritmo aqui proposto, cujo cálculo, de fato, consome bem menos tempo.

4.4 Experimentos de Navegação em corredores

Projetados os dois controladores discutidos no Capítulo 2, eles foram incorporados como entradas para um filtro de informação descentralizado formado por dois filtros locais e um filtro global, como descrito no Capítulo 3, implementando uma arquitetura de controle da forma daquela implementada em (FREIRE, 2002) (ver Figura 13).

A diferença em relação àquele trabalho, porém, está no fato de que lá o controlador baseado em fluxo óptico não foi utilizado. Mais ainda, não ficou clara toda a potencialidade da arquitetura de controle ali proposta, pois a mesma não foi utilizada em nenhum experimento com sensores distintos (apenas sensores ultra-sônicos foram considerados). O objetivo deste trabalho, então, é exatamente validar a arquitetura de controle proposta em (FREIRE, 2002) para o caso em que outros sensores são utilizados, como também é feito em (CARELLI et al., 2002).

Note-se, porém, que em (CARELLI et al., 2002) apenas visão é utilizada como fonte de informação sensorial, o que causa uma falha geral de sensoriamento em caso de alguma deficiência na captura de imagens (deficiência de iluminação, por exemplo). Portanto, a experimentação aqui conduzida, utilizando sensores ultra-sônicos e visão, é mais genérica do que aquelas conduzidas em (FREIRE, 2002) e em (CARELLI et al., 2002), devendo este trabalho ser visto como complementação daqueles.

Para validar o funcionamento dos controladores individuais discutidos no Capítulo 2 e da máquina de fusão discutida no Capítulo 3, o cálculo do fluxo foi feito utilizando o algoritmo de mínimos quadrados modificado com filtro de informação descentralizado, pela sua reduzida complexidade computacional.

Também é necessário salientar que para realizar a fusão dos controladores (controlador de fluxo óptico e controlador de ultra-som) utilizados por esta arquitetura de controle, cada controlador deve estar associado a um determinado valor de covariância. Para a covariância do controlador de ultra-som utilizamos o mesmo esquema de covariâncias baseadas em lógica nebulosa apresentado em (FREIRE, 2002), enquanto que para o controlador do fluxo óptico utilizamos a covariância gerada pelo método de mínimos quadrados.

Foram feitos vários testes usando a plataforma descrita neste trabalho, obtendo-se como resultado principal que o robô é capaz de seguir a linha média do corredor. Num deles, o corredor utilizado tem uma largura de 1,40 m e 5 m de comprimento (Figura 24) e este experimento é que será aqui detalhado. Durante sua navegação, o robô mantém uma velocidade linear $v = 0,075m/s$, e os parâmetros do controlador baseado em fluxo óptico são $k_{pw} = 20$ e $k_{dw} = 1$. Por sua vez, os parâmetros do controlador baseado em sensores ultra-sônicos são $k_1 = 0,8$, $k_2 = 0,8$, $a_1 = 2$ e $a_2 = 2$.

A trajetória do robô ao longo do corredor é mostrada na Figura 24, enquanto a Figura 25 mostra o gráfico das duas covariâncias associadas aos controladores, superpostas uma à outra. Já na Figura 26 temos a velocidade angular gerada pelo controlador de fluxo óptico implementado, e na Figura 27 temos a velocidade angular gerada pelo controlador de ultra-som, enquanto na Figura 28 podemos observar a velocidade angular gerada pelo algoritmo de fusão superposta à velocidade angular medida pela odometria do robô.

Pode-se perceber, a partir da Figura 24, que o robô consegue seguir a linha média do corredor, como esperado, mesmo quando inicia a navegação deslocado dela. Mais ainda, o robô segue o comportamento desejado mesmo em situações em que um dos controladores não tem bom desempenho. Como um exemplo desta situação, pode-se citar o momento em que o robô se aproxima do final do corredor, como ilustrado na Figura 29. Neste caso, a

informação de fluxo óptico nas paredes do corredor já não está disponível, e o controlador baseado em fluxo óptico não gera mais um sinal de controle adequado. Porém, como o robô ainda está no corredor, os seus sensores ultra-sônicos laterais ainda conseguem medir as distâncias às paredes, como indicado na Figura 30, permitindo que o controlador que usa sua informação gere um sinal de controle adequado, e assim o robô permanece na trajetória desejada. Em outras palavras, o uso de informação sensorial proveniente de fontes distintas permite maior versatilidade e autonomia ao sistema de navegação, como esperado.

Na Figura 30 é mostrado um segundo experimento no qual o robô é deslocado do centro do corredor, bem mais que na Figura 24, mas a fusão dos dois controladores consegue igualmente levar o robô até a linha média do corredor e manter uma trajetória ao longo dela, até acabar o corredor, que é quando começa a perder a informação de ultra-som e de fluxo. Para este experimento, a Figura 31 mostra as covariâncias superpostas, a Figura 32 mostra a velocidade angular produzida pelo controlador de fluxo óptico, a Figura 33 mostra a velocidade angular produzida pelo controlador de ultra-som, e a Figura 34 a velocidade resultante do processo de fusão de velocidades angulares e a velocidade angular medida pela odometria do robô.

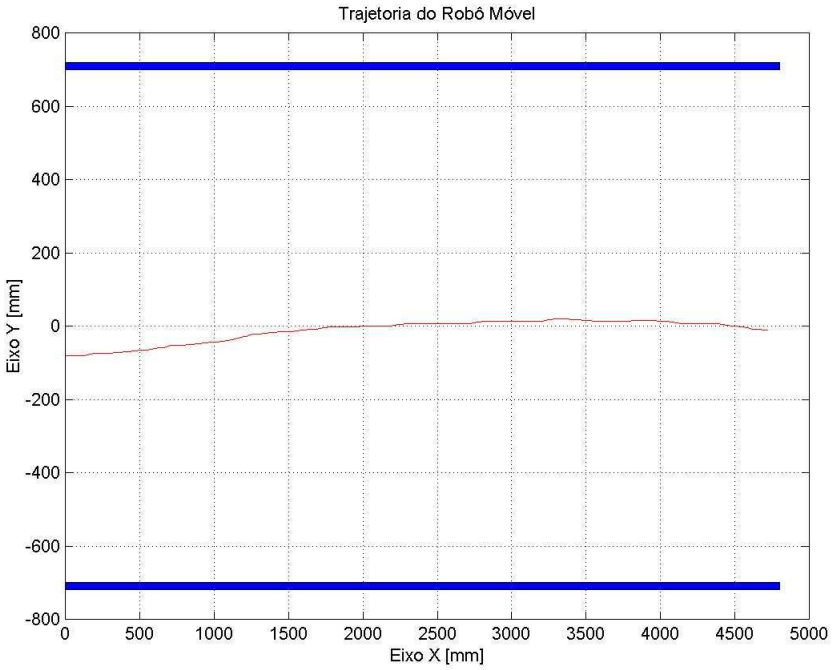

Figura 24: Trajetória do robô ao longo do corredor.

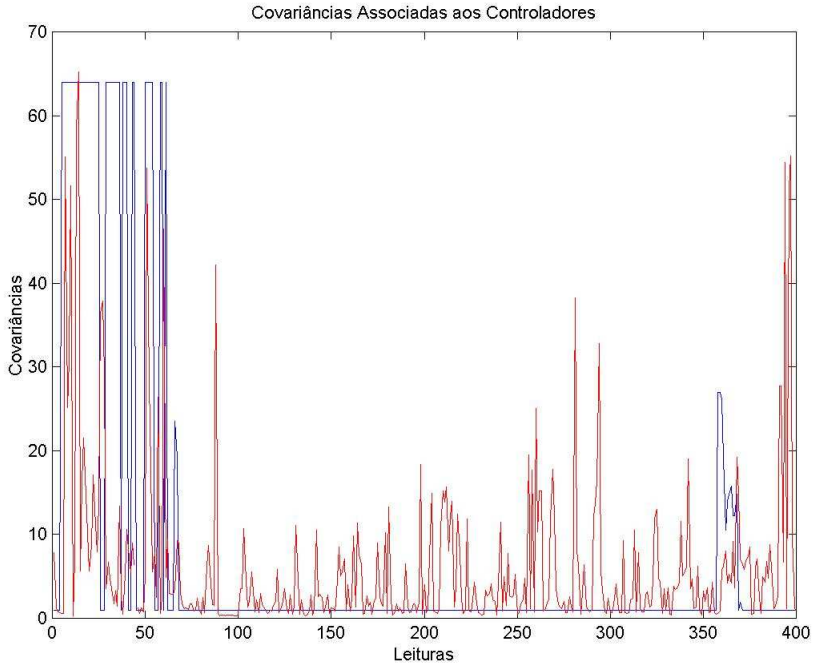

Figura 25: Covariâncias associadas aos controladores sobrepostas.

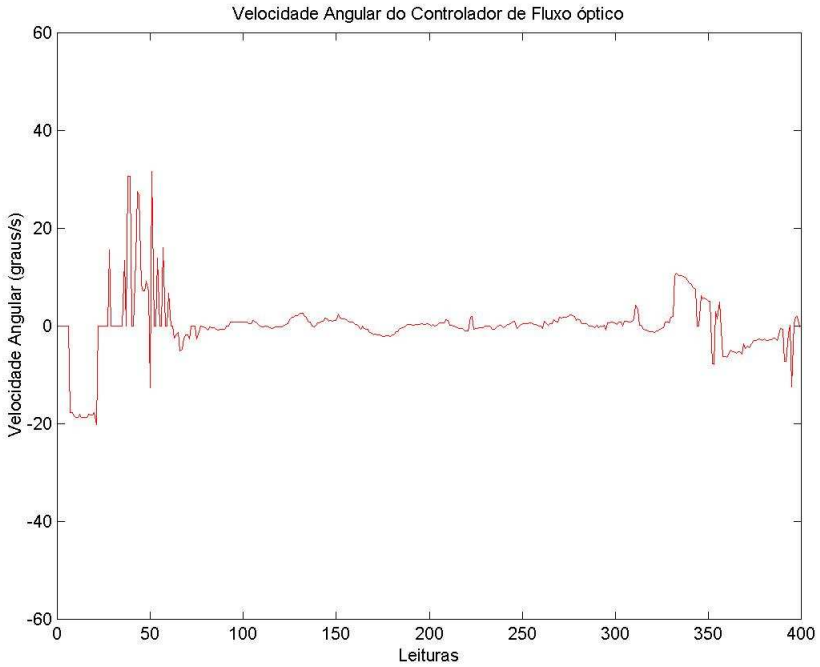

Figura 26: Velocidade angular produzida pelo controlador de fluxo óptico.

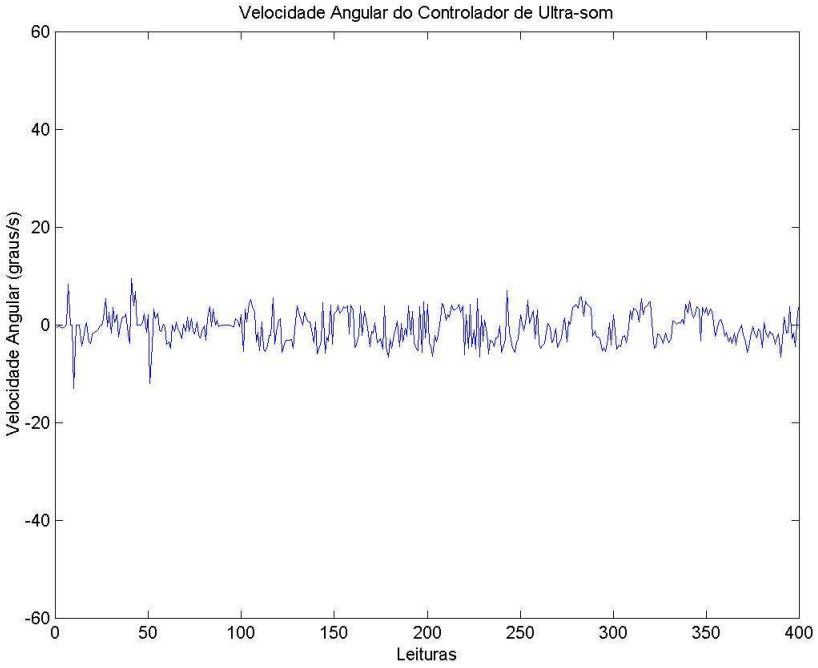

Figura 27: Velocidade angular produzida pelo controlador de ultra-som.

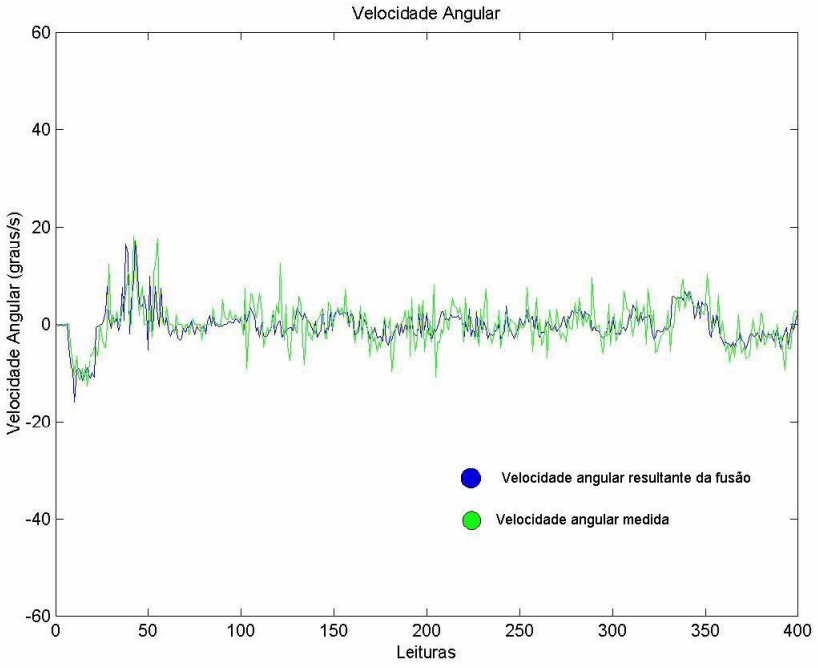

Figura 28: Velocidade angular resultante do processo de fusão e velocidade angular medida pelo robô.

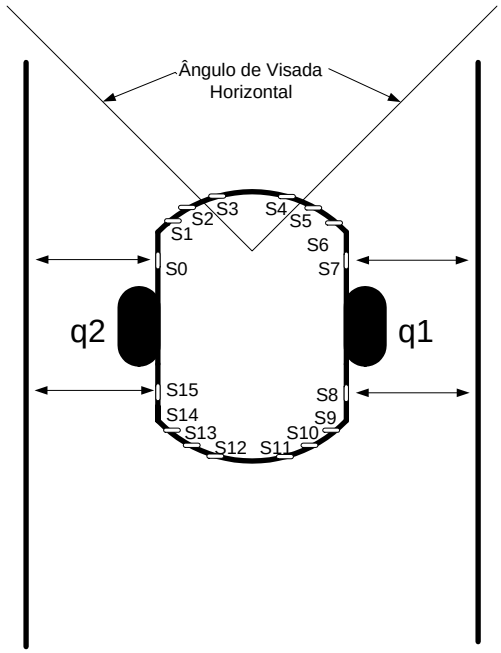

Figura 29: Posição do robô no final do corredor, quando a informação de fluxo óptico não está mais disponível.

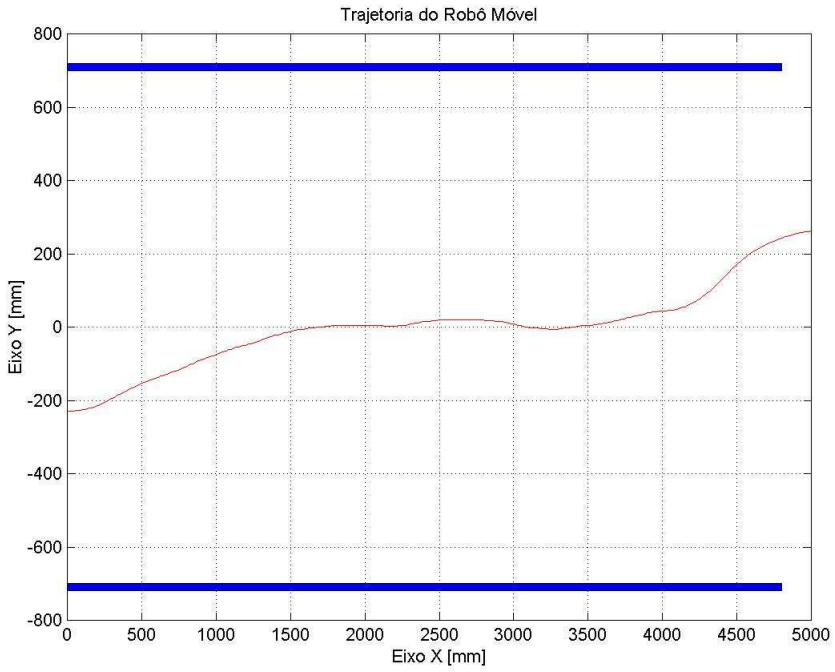

Figura 30: Trajetória do robô ao longo do corredor.

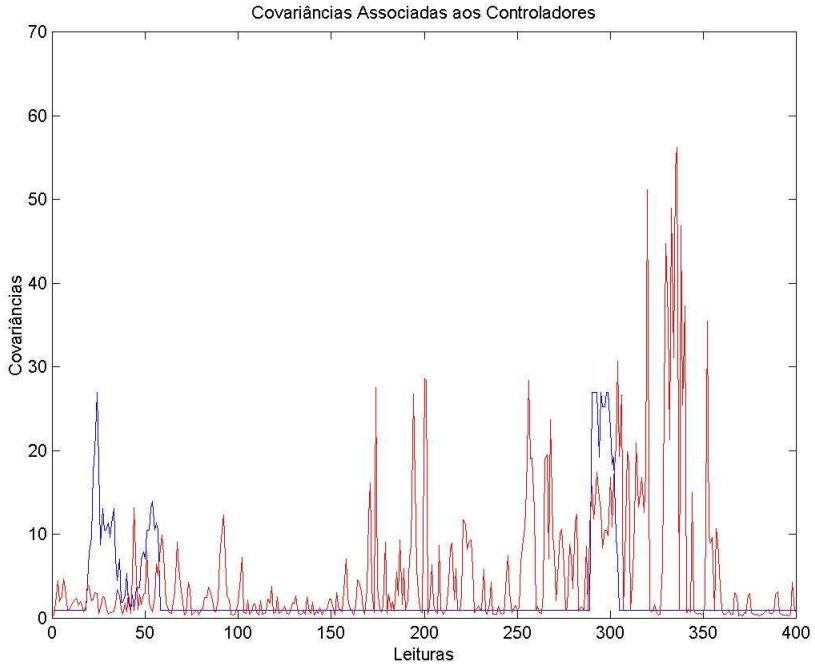

Figura 31: Covariâncias associadas aos controladores sobrepostas.

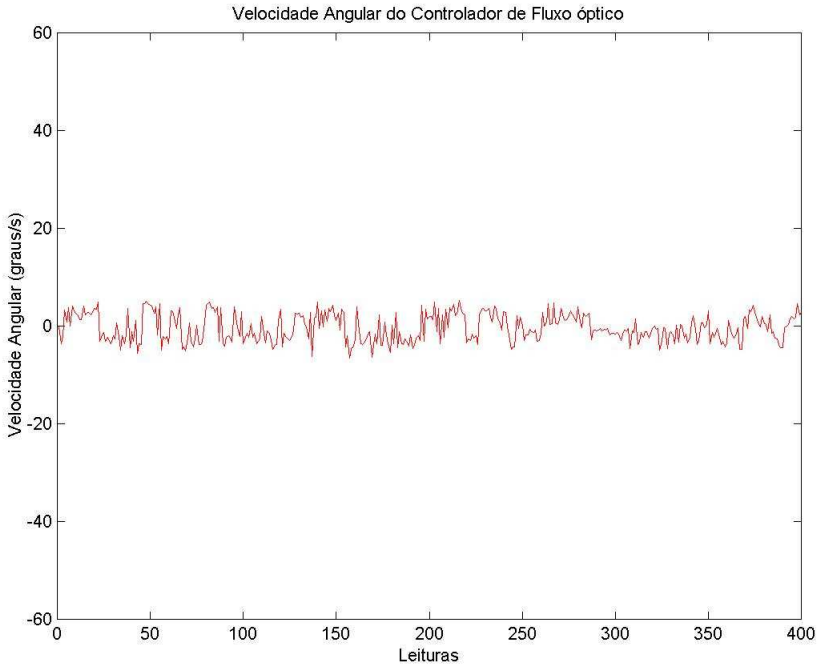

Figura 32: Velocidade angular produzida pelo controlador de fluxo óptico.

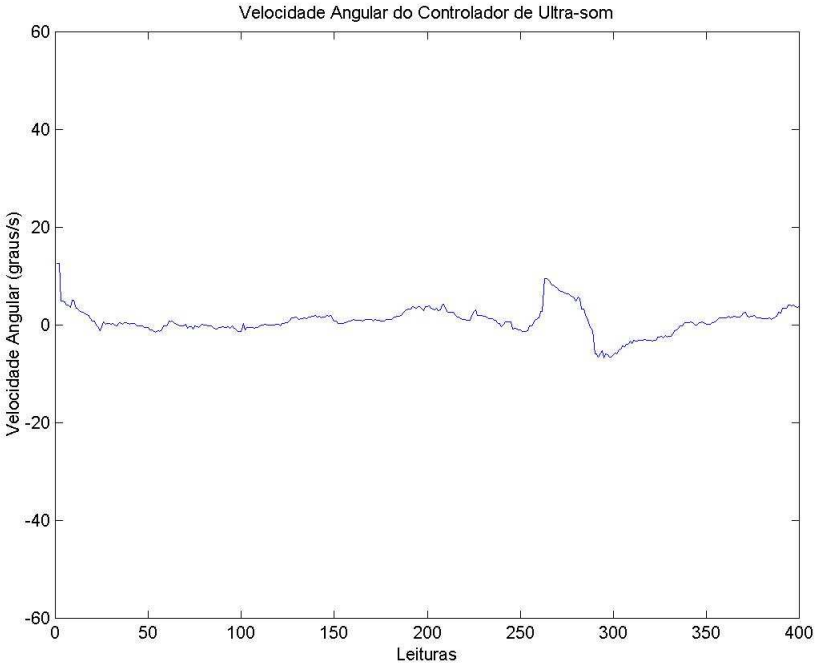

Figura 33: Velocidade angular produzida pelo controlador de ultra-som.

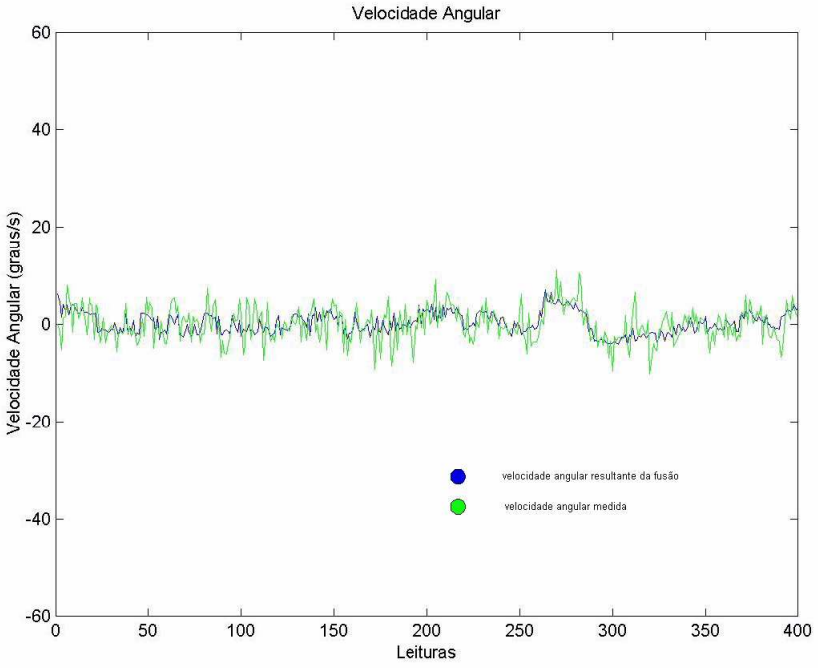

Figura 34: Velocidade angular resultante do processo de fusão e velocidade angular medida pelo robô.

5 Conclusões

O objetivo pretendido durante o desenvolvimento deste trabalho foi atingido, no sentido de que o robô consegue seguir e se manter navegando ao longo da linha média de um corredor, aproveitando a informação obtida via fusão de dois controladores, os quais têm fontes sensoriais diferentes. A arquitetura proposta conseguiu implementar, assim, dois controladores diferentes e redundantes, a fim de gerar um só sinal de controle.

Foram estudadas e exploradas técnicas estocásticas de fusão de controladores, desde a mais clássica, o filtro de Kalman, até técnicas propostas mais recentemente, como o filtro de informação descentralizado, o qual foi implementado e utilizado como um dos pilares do desenvolvimento deste trabalho. Uma vez feito um estudo e exploração da teoria do fluxo óptico e das distintas técnicas utilizadas para seu cálculo, foram combinados dois trabalhos anteriormente implementados com a finalidade de obter um novo algoritmo de cálculo de fluxo óptico, o qual foi implementado e testado, a fim de comprovar sua utilidade para a tarefa de navegação em corredores.

Também foi implementado e aplicado um controlador à base de sensores ultra-sônicos, desenvolvido anteriormente, cuja estabilidade é comprovada. Este controlador foi implementado sobre a plataforma Aria e no sistema operacional Linux, da mesma maneira que o controlador baseado em fluxo óptico, o qual também tem estabilidade comprovada. Ambos os controladores conseguiram manter o robô, separada e independentemente, navegando ao longo da linha média do corredor, mas sujeitos às desvantagens e limitações próprias de cada um deles.

O controlador obtido a partir da fusão dos sinais de controle, que tem como característica principal tanto o fato de utilizar sensores redundantes quanto de complementar os dois controladores antes desenvolvidos, foi implementado, e demonstrou, em vários experimentos de navegação, um bom desempenho, servindo plenamente para a tarefa para a qual foi projetado.

É importante salientar que este trabalho está dentro do contexto de uma área de

grande importância dentro da robótica, que é a área de fusão sensorial, a qual deve ser explorada e analisada em toda sua potencialidade. Também é necessário considerar que a plataforma de trabalho, que inclui o sistema operacional e os softwares utilizados, teve que ser estudada e adaptada às necessidades deste trabalho, o qual também deixa como saldo o desenvolvimento de uma nova plataforma de trabalho dentro do LAI (Laboratório de Automação Inteligente da UFES), plataforma esta que pode ser a base de novos trabalhos a serem desenvolvidos no laboratório.

Esta dissertação também representa uma continuação de trabalhos anteriormente desenvolvidos e, ao mesmo tempo, deixa abertas janelas para trabalhos futuros, visando gerar sistemas com maior autonomia. Dentre estes trabalhos, podemos citar o uso de um maior número de sensores e controladores, e fazer a fusão dos mesmos (por exemplo sensores laser, de luz infravermelha, etc) e aplicar esta redundância não somente à tarefa de navegação em corredores, mas a outras fases de navegação, como evitar obstáculos ou seguir pessoas.

Também seria interessante e plausível a implementação e uso de um sistema de controle usando imagens omnidirecionais e sensores à base de ultra-som, num esquema de redundância, em diferentes fases de navegação.

Referências

BARRON, J. L.; BEAUCHEMIN, S. S. Performance of optical flow techniques. *International Journal of Computer Vision*, v. 12, n. 1, p. 43–77, 1994.

BEAUCHEMIN, S. S.; BARRON, J. L. The computation of optical flow. *ACM Computing Surveys*, v. 27, n. 3, p. 433–467, 1995.

BLACK, M. J.; ANANDAN, P. A model for the detection of motion over time. In: *Proceedings of the International Conference on Computer Vision*. Osaka, Japan: [s.n.], 1990. p. 33–37.

BROWN, R. G.; HWANG, P. Y. C. *Introduction to Random signals ans Applied Kalman Filtering: with MATLAB exercises and solutions*. 3rd. ed. [S.l.]: John Wiley and Sons, 1997.

CALDEIRA, E. M. O. *Navegação Reativa de Robôs Móveis com Base no Fluxo Óptico*. Tese (Doutorado) — Universidade Federal do Espírito Santo, Vitória, ES, Brasil, 2002.

CARELLI, R. et al. Stable agv corridor navigation with fused vision-based control signals. In: *Proceedings of the 28th Annual Conference of the IEEE Industrial Electronics Society - IECON 02*. Sevilla, Spain: [s.n.], 2002. p. 2433–2438.

CLOCKSIN, W. Determining the orientation of surfaces from optical flow. In: *Proceedings Third AISB Conference*. Hamburg: [s.n.], 1978. p. 93–102.

DEV, A.; KROSE, B. J. A.; GROEN, F. C. A. Navigation of a mobile robot on the temporal development of the optic flow. In: *Proceedings of the 1997 IEEE/RSJ/GI International conference on Intelligent Robots and Systems (IROS'97)*. Grenoble, France: [s.n.], 1997. p. 558–563.

FREIRE, E. et al. A new mobile robot control approach via fusion of control signals. *IEEE Transactions on Systems, Man and Cybernetics, part B*, v. 34, n. 1, p. 419–429, 2004.

FREIRE, E. O. *Controle de Robôs Moveis por Fusão de Sinais de Controle Usando Filtro de Informação Descentralizado*. Tese (Doutorado) — Universidade Federal do Espírito Santo, Vitória, ES, Brasil, 2002.

FUKE, Y.; KROTKOV, E. Dead reckoning for a lunar rover on uneven terrain. In: *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*. Minneapolis, Minnesota, USA: [s.n.], 1996. p. 411–416.

GIBSON, J. On the analysis of change in the optic array. *Scandinavian Journal of Psychology*, v. 18, p. 161–163, 1977.

HONG, L. Sense your world better: Multisensor/information fusion. *IEEE Journal of Circuits and Sistems*, v. 10, n. 3, p. 7–8;12–15;28, 1999.

HORN, B. K. P.; SCHUNCK, B. G. Determining optical flow. *Artificial Intelligence*, v. 17, p. 185–204, 1981.

KELLY, R.; CARELLI, R. A class of non-linear pd-type controllers for robot manipulators. *Journal of Robotic Systems*, v. 13, p. 793–802, 1996.

KHALIL, H. K. *Non-Linear Systems*. second edition. USA: Prentice-Hall, 1996.

KOENDERINK, J.; DOORN, A. J. van. Visual perception of rigidity of solid shape. *Journal of Mathematical Biology*, v. 79, p. 79–85, 1976.

LANGLEY, K. et al. Vertical and horizontal disparities from phase. *Image and Vision Computing*, v. 9, p. 296–302, 1991.

LEHRER, M. et al. Motion cues provide the bee's visual world with a third dimension. *Nature*, n. 6162, p. 356–357, 1988.

LUKAS, B.; KANADE, T. An iterative image registration technique with an application to stereo vision. In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI'81)*. Vancouver, BC, Canada: [s.n.], 1981. p. 674–679.

MURPHY, R. R. *Introduction to AI Robotics*. 1st. ed. [S.l.]: MIT Presss, 2000.

MUTAMBARA, A. G. O. *Decentralized Estimation and Control for Multi-sensor Systems*. USA.: CRC Press, 1998.

PRINCE, J. L.; MCVEIGH, E. R. Motion estimation from tagged mr image sequences. *IEEE Transactions on Medical Images*, v. 11, p. 238–249, 1992.

ROMANO, V. F. (Ed.). *Robótica Industrial*. São Paulo.: Editora Edgard Blücher Ltda, 2002.

SANDI-LORA, F. A.; HEMERLY, E. M.; LAGES, W. F. Sistema para navegação e guiagem de robôs móveis autônomos. *Revista Controle e Automação*, v. 9, n. 3, 1998.

SANTOS-VICTOR, J. A. et al. Divergent stereo in autonomous navigation: From bees to robots. *International Journal of Computer Vision*, v. 14, p. 159–177, 1995.

SARCINELLI-FILHO, M.; SCHNEEBELI, H. A.; CALDEIRA, E. M. O. Using optical flow to control mobile robot navigation. In: *Proceedings of the 15th IFAC World Congress on Automatic Control*. Barcelona, Espanha: [s.n.], 2002.

SASIADEK, J. *Sensor Fusion. IFAC Professional Briefs, disponível em. http : //www.ifac − control.org: World Wide Web*, 2004.

SASIADEK, J.; KHE, J. Sensor fusion based on fuzzy kalman filter. In: *Proceedings of the Second International Workshop on Robot Motion Control*. [S.l.: s.n.], 2001. p. 275–283.

SORIA, C. M.; CARELLI, R.; SARCINELLI-FILHO, M. Optical flow estimation using data fusion. In: *Anais do VI Simpósio Brasileiro de Automação Inteligente - VI SBAI.* Bauru, SP, Brasil: [s.n.], 2003. p. 259–265.

VASSALLO, R. F.; SCHNEEBELI, H. A.; SANTOS-VICTOR, J. A. Visual navigation using visual servoing and appearance based methods. *Robotics and Autonomos Systems*, v. 31, p. 87–97, 2000.

ZHANG, Z.; FAUGERAS, O. Three-dimensional motion computation and object segmentation in a long sequence in stereo frames. *International Journal of Computer Vision*, v. 7, n. 3, p. 211–241, 1992.

ZHENG, Q.; CHELLAPPA, R. Automatic feature point extraction and tracking in image sequences for unknown image motion. In: *Proceedings of the International Conference on Computer Vision.* Berlin, Germany: [s.n.], 1993. p. 335–339.

ZWANN, S. van der; SANTOS-VICTOR, J. An insect inspired visual sensor for the autonomous navigation of a mobile robot. In: *Proceedings of the Seventh International Symposium on Intelligent Robotic Systems.* Oxford: [s.n.], 1997. p. 226–248.

APÊNDICE A – Código do programa fusioncontrolertotal.c

```
//Program to calculate optical flow using the Modified Mean Squares
// diferential technique  and the Decentralized Information filter,
//taking pixels in each subregion of 5*5 pixels (see chapter 3)
#include "video.h"
#include "capturateste.h"
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include "caria.h"
#include  <math.h>
#define PI 3.14159265358979323846264338327795
void varifuzzy(double dist,double dif,double *Re1);
int microseconds(void);
struct Camera camera0;
static char *im0;//pointer for the first image
static char *im1;//pointer for the second image
int err, errcode;
float ulfusion,urfusion;//left and right horizontal flows
float ulsigma,ursigma; //left and right variances

/*******************************************/
/*FUNCTION capturaIm0                */
/*******************************************/
/*Function to capture images*/
int capturaIm0(){
  err = 0;
```

```
  errcode = -1;
  if ((err = CameraInit (&camera0, "/dev/video0")) >= 0)
    {
      errcode = -2;
      camera0.bpp = 1;
      camera0.mode = MODE_GRAY;
      camera0.win.width=320;
      camera0.win.height=240;
      camera0.size = camera0.bpp * camera0.win.width * camera0.win.height;
      camera0.pic.palette = VIDEO_PALETTE_GREY;
      camera0.pic.depth = 8;
      if ((err = VideoInit (&camera0)) >= 0)
{
  errcode = -3;
  if ((err = VideoAlloc (&camera0, &im1)) >= 0)
               {
                errcode = -4;
               }
}
    }
    return errcode;
}

/*********************************************/
/*FUNCTION finalizaCapturaIm0             */
/*********************************************/
/*function that ends the process of image capture*/

int finalizaCapturaIm0(){
  if (errcode != -3){
    VideoDealloc(&camera0, &im1);
    }
  VideoClose (&camera0);
  return 0;
}
```

```
/*********************************************/
/* FUNCTION CalculateFlow                    */
/*********************************************/
/*function to calculate the optical flow using the method
modified mean squared and the decentralized information filter*/
int CalculateFlow()
{
float Ix0,Iy0,It0;//intensity derivatives
float a11,a12,a22;
float a1,a2,a4,d1,d2,deta;
float error,erter,sig2;
float fUMean,fVMean;
float xeg,rhoeg;
float rho[3],u[3],v[3];
float *result,*xresult;
int px1,px2,px3,px4,px5,px6,px7,px8;
int k=0,iterations;
int lincuado,colcuado;
int columnsiterations;
int q;
int i;
int n,auxiliar1;
int counter1,counter2,counter3;
float matrixul[11],matrixur[11],matrixrhol[11],matrixrhor[11];
int centinela;
//Decentralized Information Filter Variables
int c;
float matY[11],maty[11],matz[11];
float Y,y,yfinal;
float Aux1,Aux2;
        a11=0;
        a12=0;
        a22=0;
        lincuado=20;
```

```
        colcuado=10;
      columnsiterations=92;
      counter3=0;
      counter2=1;
      counter1=1;
      centinela=0;
      k=0;
      for(q=1;q<columnsiterations;q+=1){
           n=1;
         auxiliar1=1;
         a1=0;
         a2=0;
         a4=0;
         d1=0;
         d2=0;
         for (iterations=1;iterations<5;iterations+=1){
//Calculation of the Derivatives
px1=(int) ((unsigned char) (im0[(colcuado+auxiliar1+1)+(lincuado+n)*w]));
px2=(int) ((unsigned char) (im0[(colcuado+auxiliar1)+(lincuado+n)*w]));
px3=(int) ((unsigned char) (im0[(colcuado+auxiliar1+1)+(lincuado+n+1)*w]));
px4=(int) ((unsigned char) (im0[(colcuado+auxiliar1)+(lincuado+n+1)*w]));
px5=(int) ((unsigned char) (im1[(colcuado+auxiliar1+1)+(lincuado+n)*w]));
px6=(int) ((unsigned char) (im1[(colcuado+auxiliar1)+(lincuado+n)*w]));
px7=(int) ((unsigned char) (im1[(colcuado+auxiliar1+1)+(lincuado+n+1)*w]));
px8=(int) ((unsigned char) (im1[(colcuado+auxiliar1)+(lincuado+n+1)*w]));

Ix0=(float)(px1-px2+px3-px4+px5-px6+px7-px8)/4.;

px1=(int) ((unsigned char) im0[(colcuado+auxiliar1)+(lincuado+n+1)*w]));
px2=(int) ((unsigned char) (im0[(colcuado+auxiliar1)+(lincuado+n)*w]));
px3=(int) ((unsigned char) (im0[(colcuado+auxiliar1+1)+(lincuado+n+1)*w]));
px4=(int) ((unsigned char) (im0[(colcuado+auxiliar1+1)+(lincuado+n)*w]));
px5=(int) ((unsigned char) (im1[(colcuado+auxiliar1)+(lincuado+n+1)*w]));
px6=(int) ((unsigned char) (im1[(colcuado+auxiliar1)+(lincuado+n)*w]));
px7=(int) ((unsigned char) (im1[(colcuado+auxiliar1+1)+(lincuado+n+1)*w]));
```

```
px8=(int) ((unsigned char) (im1[(colcuado+auxiliar1+1)+(lincuado+n)*w]));

Iy0=(float)(px1-px2+px3-px4+px5-px6+px7-px8)/4.;

px1=(int) ((unsigned char) (im1[(colcuado+auxiliar1)+(lincuado+n)*w]));
px2=(int) ((unsigned char) (im0[(colcuado+auxiliar1)+(lincuado+n)*w]));
px3=(int) ((unsigned char) (im1[(colcuado+auxiliar1)+(lincuado+n+1)*w]));
px4=(int) ((unsigned char) (im0[(colcuado+auxiliar1)+(lincuado+n+1)*w]));
px5=(int) ((unsigned char) (im1[(colcuado+auxiliar1+1)+(lincuado+n)*w]));
px6=(int) ((unsigned char) (im0[(colcuado+auxiliar1+1)+(lincuado+n)*w]));
px7=(int) ((unsigned char) (im1[(colcuado+auxiliar1+1)+(lincuado+n+1)*w]));
px8=(int) ((unsigned char) (im0[(colcuado+auxiliar1+1)+(lincuado+n+1)*w]));

It0=(float)(px1-px2+px3-px4+px5-px6+px7-px8)/4.;

            n=n+1;
            auxiliar1=auxiliar1+1;
            a1+=Ix0*Ix0;
            a2+=Ix0*Iy0;
            a4+=Iy0*Iy0;
            d1+=Ix0*(-It0);
            d2+=Iy0*(-It0);
 }
 n=1;
 auxiliar1=1;
 //Calculation of the inverse of A
                deta=a1*a4-a2*a2;  //Determinant of A

 if(deta!=0)
                                  {
                                    a11=a4/deta;    //Inversion Result
                                    a12=-a2/deta;
                                    a22=a1/deta;
                                    fUMean=a11*d1+a12*d2;
                                    fVMean=a12*d1+a22*d2;
```

```
                                    u[k]=fUMean;
                                    v[k]=fVMean;
                                    //error calculation
                                    erter=0;
for(iterations=1;iterations<5;iterations+=1){//repetitions=1:4
px1=(int) ((unsigned char) (im0[(colcuado+auxiliar1+1)+(lincuado+n)*w]));
px2=(int) ((unsigned char) (im0[(colcuado+auxiliar1)+(lincuado+n)*w]));
px3=(int) ((unsigned char) (im0[(colcuado+auxiliar1+1)+(lincuado+n+1)*w]));
px4=(int) ((unsigned char) (im0[(colcuado+auxiliar1)+(lincuado+n+1)*w]));
px5=(int) ((unsigned char) (im1[(colcuado+auxiliar1+1)+(lincuado+n)*w]));
px6=(int) ((unsigned char) (im1[(colcuado+auxiliar1)+(lincuado+n)*w]));
px7=(int) ((unsigned char) (im1[(colcuado+auxiliar1+1)+(lincuado+n+1)*w]));
px8=(int) ((unsigned char) (im1[(colcuado+auxiliar1)+(lincuado+n+1)*w]));

          Ix0=(float)(px1-px2+px3-px4+px5-px6+px7-px8)/4.;
px1=(int) ((unsigned char) (im0[(colcuado+auxiliar1)+(lincuado+n+1)*w]));
px2=(int) ((unsigned char) (im0[(colcuado+auxiliar1)+(lincuado+n)*w]));
px3=(int) ((unsigned char) (im0[(colcuado+auxiliar1+1)+(lincuado+n+1)*w]));
px4=(int) ((unsigned char) (im0[(colcuado+auxiliar1+1)+(lincuado+n)*w]));
px5=(int) ((unsigned char) (im1[(colcuado+auxiliar1)+(lincuado+n+1)*w]));
px6=(int) ((unsigned char) (im1[(colcuado+auxiliar1)+(lincuado+n)*w]));
px7=(int) ((unsigned char) (im1[(colcuado+auxiliar1+1)+(lincuado+n+1)*w]));
px8=(int) ((unsigned char) (im1[(colcuado+auxiliar1+1)+(lincuado+n)*w]));

           Iy0=(float)(px1-px2+px3-px4+px5-px6+px7-px8)/4.;
px1=(int) ((unsigned char) (im1[(colcuado+auxiliar1)+(lincuado+n)*w]));
px2=(int) ((unsigned char) (im0[(colcuado+auxiliar1)+(lincuado+n)*w]));
px3=(int) ((unsigned char) (im1[(colcuado+auxiliar1)+(lincuado+n+1)*w]));
px4=(int) ((unsigned char) (im0[(colcuado+auxiliar1)+(lincuado+n+1)*w]));
px5=(int) ((unsigned char) (im1[(colcuado+auxiliar1+1)+(lincuado+n)*w]));
px6=(int) ((unsigned char) (im0[(colcuado+auxiliar1+1)+(lincuado+n)*w]));
px7=(int) ((unsigned char) (im1[(colcuado+auxiliar1+1)+(lincuado+n+1)*w]));
px8=(int) ((unsigned char) (im0[(colcuado+auxiliar1+1)+(lincuado+n+1)*w]));

It0=(float)(px1-px2+px3-px4+px5-px6+px7-px8)/4.;
```

```
n=n+1;
                auxiliar1=auxiliar1+1;
                error=Ix0*fUMean+Iy0*fVMean+It0;
erter+=error*error;

  }
                //Calculation of the  variance
                                  sig2=erter/4.;
                                  rho[k]=sig2;
                                  k++;
                                 }
else{
                                        u[k]=0.;
                                        v[k]=0.;
                                        rho[k]=125.;
                                        k++;

                                 }
                 counter2=counter2+1;
                 colcuado=colcuado+5;
                 counter1=counter1+1;
 if (counter1==3){
                                  colcuado=colcuado-10;
                                  lincuado=lincuado+5;
                                  counter1=1;
                             }

if (counter2==5){
                                  counter2=1;
                                  k=0;
                                  result=Fusion(4,u,rho);
                                  xeg=result[1];
                           rhoeg=result[2];
                           if(counter3==0){
              matrixul[centinela]=xeg;
```

```
              matrixrhol[centinela]=rhoeg;
              urfusion=xeg;
                                          }else{
              matrixur[centinela]=xeg;
              matrixrhor[centinela]=rhoeg;
              urfusion=xeg}
              centinela=centinela+1;

            }

   if (q==48){
                     lincuado=20;
                     colcuado=300;
                     centinela=0;
                     counter3=1;
                    }
        }
//LEFT WINDOW
//Global Filter Initial Values
y=0;
Y=0;
//Local Filters Initial Values

for(i=0;i<12;i++){
   maty[i]=0;
   matY[i]=0;
}
for (c=0;c<10;c++){
   // Local Filters
   for (i=0;i<n;i++){
      matz[i]=1*matrixul[i];
      maty[i]=maty[i]+(1/matrixrhol[i])*matz[i];
      matY[i]=matY[i]+1/matrixrhol[i];
   }
   // Global Filter
```

```
   Aux1=0;
   Aux2=0;
   for (i=0;i<n;i++){
      Aux1=Aux1+maty[i];
      Aux2=Aux2+matY[i];
   }
   y=Aux1-(i-1)*y;
   Y=Aux2-(i-1)*Y;
   // local filters actualization
   for ( i=0;i<n;i++){
      maty[i]=y;
      matY[i]=Y;
   }
}
    ulfusion=(1/Y)*y;
    ulsigma=(1/Y);

    //right Window
    //Global Filter Initial Values
y=0;
Y=0;
//Local Filters Initial Values
for(i=0;i<12;i++){
   maty[i]=0;
   matY[i]=0;
}
for (c=0;c<10;c++){
   // Local Filters
   for (i=0;i<n;i++){
      matz[i]=1*matrixur[i];
      maty[i]=maty[i]+(1/matrixrhor[i])*matz[i];
      matY[i]=matY[i]+1/matrixrhor[i];
   }
   // Global Filter
   Aux1=0;
```

```
   Aux2=0;
   for (i=0;i<n;i++){

      Aux1=Aux1+maty[i];
      Aux2=Aux2+matY[i];
   }
   y=Aux1-(i-1)*y;
   Y=Aux2-(i-1)*Y;
   // local filters actualization
   for ( i=0;i<n;i++){
      maty[i]=y;
      matY[i]=Y;
   }
}
    urfusion=(1/Y)*y;
    ursigma=(1/Y);
    return 0;
}

float*Fusion(int n,float matrixu[],float matrixrho[]){
int c,i;
float matY[11],maty[11],matz[11];
float Y,y,yfinal;
float Aux1,Aux2;
float *xreturn;
//Global Filter Initial Values
y=0;
Y=0;
//Local Filters Initial Values
for(i=0;i<12;i++){
   maty[i]=0;
   matY[i]=0;
}
for (c=0;c<10;c++){
   // Local Filters
```

```
   for (i=0;i<n;i++){
      matz[i]=1*matrixu[i];
      maty[i]=maty[i]+(1/matrixrho[i])*matz[i];
      matY[i]=matY[i]+1/matrixrho[i];
   }
   // Global Filter
   Aux1=0;
   Aux2=0;
   for (i=0;i<n;i++){
      Aux1=Aux1+maty[i];
      Aux2=Aux2+matY[i];
   }
   y=Aux1-(i-1)*y;
   Y=Aux2-(i-1)*Y;
   // local filters actualization
   for ( i=0;i<n;i++){
      maty[i]=y;
      matY[i]=Y;
   }
}
    xreturn[2]=(1/Y)*y;
    xreturn[1]=(1/Y);
    return xreturn;
}

int main()
{
  float omegaVis,sigmaVis;
  float omegaFusion,sigmaFusion;
  double Re1,Re2;
  double dist[16];
  int i,c,mindist,minsen,p;
  double k1,k2,v,wwmax,
        x=0,  //Distance from Corridor Center to Robot Center
        teta, //Robot orientation with respect to the Corridor Centerline
```

```
        A1,A2,//Num;
        vel=100;
const double d = 230,a1 = 2,a2 = 2 ;//k3 = 100;
double xi,yi,fii; // Robot initial Coordinates
double x0,y0,fi0; // Initial Coordinates System
double xd,yd,fid; // Destination coordinates
double yp,ang; // Ultrasonics Measures
double ye,fie;    // States Stimation
float xop,yop,fiop;// Personal Odometry for the Corridor
float xo,yo,fio; //  personal Odometry
  double dizq,dder,dif,dif1,dif2;
  double auxdist,auxmd,auxd;
  double cordinatex,cordinatey,phi,angularvelocity,linearvelocity;
  float omegaUsom,sigmaUsom;
  float errorfluxo,errorfluxoant=0;
  float Kp=0.5,Kd=0.25;//
  float matY[1],maty[1],matz[1];
  float rho[1],u[1];
  float Y,y,yfinal;
  float Aux1,Aux2;
  //Kalman Filter Derivatives Variables
double ph11,ph12,pp11,pp12,pp21,pp22,K1,K2,paux1,paux2,pq11,pq22,pr,x1,x2,z;
double omega,derivativeomega;
double T=0.1;
double factor1;
int    wrFlow;
int    T_initial,T_final;
//mobile robot parameters
double lambda=5.4;
double rx1=3.05;
double J11,J12,J21,J22,Ji11,Ji12,Ji21,Ji22,detJ;
//Declaration of Controller Local Variables
double aux,aux1,aux2;
//Robot Dynamical Parameters
double aw=104.6;
```

```
double bw=9.21;
double kw=0.45;
//Adjusting Parameters of the Flow Controller
double  KpF=10;
double  KdF=20;
        float *xreceive;
clock_t tempo0, tempo1,tempo2;
FILE *pFile,*pChez,*pFondateur,*pcordinatex,*pcordinatey;
FILE *pphi,*plinearvelocity,*pangularvelocity;
FILE *pflowvariance,*pultrasonicsvariance;
        FILE *pvelocities;
        FILE *pultrasonicsvelocity, *pflowvelocity;
FILE *pfusionvelocity, *pleftflow, *prightflow;
wwmax = PI/3; // Maximum Angular Velocity = 60 degrees/second
v= 0.100;
k1=0.8;
k2=0.8;
 //End of the Initialization  of Controller Local Variables
 //Inicialization of the  Kalman filter
ph11=1;
ph12=0;
pp11=10;
pp12=0;
pp21=0;
pp22=10;
x1=0;
x2=0;
//Adjusting parameters
pq11=0.00001;
pq22=0.00001;
pr=1.;
  pcordinatex=fopen("cordinatex18.dat","w");
  pcordinatey=fopen("cordinatey18.dat","w");
  pphi=fopen("phi18.dat","w");
  plinearvelocity=fopen("linearvelocity18.dat","w");
```

```
pangularvelocity=fopen("angularvelocity18.dat","w");
pultrasonicsvariance=fopen("ultrasonicsvariance18.dat","w");
pflowvariance=fopen("flowvariance18.dat","w");
pvelocities=fopen("velocities18.dat","w");
pultrasonicsvelocity=fopen("ultrasonicsvelocity18.dat","w");
pflowvelocity=fopen("flowvelocity18.dat","w");
pfusionvelocity=fopen("fusionvelocity18.dat","w");
pleftflow=fopen("leftflow18.dat","w");
prightflow=fopen("rightflow18.dat","w");
T_initial = microseconds();

if (capturaIm0() == -4) {
      }
      else{
               printf("Error en Alocacion de Imagen 0\n");
                return 0;
      }

     w = camera0.win.width;
     h = camera0.win.height;
     printf("w=%d   h=%d\n",w,h);
     im0 = (char*) calloc(w*h,sizeof(char));
     Aria_Init();
     Aria_CreateConnection();
     Aria_Sleep(8000);
     //  Initial  Position
   xi=01.0;
   yi=0.0;
   fii=0.0;
  //  Final  Position
   xd=5.0;
   yd=5.5;
   fid=0.0;
   //Initial Coordinates System
   x0=0.0;
```

```
     y0=0.0;
     fi0=0.0;
     // Personal odometry variables Inicialization
    xo = (float)(xi);
    yo = (float)(yi);
    fio = (float)((fii)*(PI/180));
    xop = (float)(xi-x0);
    yop = (float)(yi-y0);
    fiop = (float)((fii-fi0)*(PI/180));
      //state estimation variables inicialization
    ye = (yi-y0);
    fie = ((fii-fi0)*(PI/180));
      Aria_SetLinVelocity(75);

for(p=0;p<520;p++)
{
tempo0=clock();
        //pFile=fopen("premier.pgm","w");
//pChez=fopen("deuxieme.pgm","w");
  VideoGrab(&camera0);
//fprintf(pFile,"P2\n");
//fprintf(pFile,"%d\n",w);
//fprintf(pFile,"%d\n",h);
//fprintf(pFile,"%d\n",255);
//fprintf(pChez,"P2\n");
//fprintf(pChez,"%d\n",w);
//fprintf(pChez,"%d\n",h);
//fprintf(pChez,"%d\n",255);
for(i=0;i<h*w;i++){
                im0[i]=(unsigned char) (im1[i]);
//fprintf(pFile,"%d ",(unsigned char)im0[i]);
}

       //fclose(pFile);
       VideoGrab(&camera0);
```

```
       for(i=0;i<h*w;i++){
                im1[i]=(unsigned char) (im1[i]);
   }
       tempo1=clock();
CalculateFlow();

//Non Linear Flow Controller//
        ulfusion;
urfusion;
//Measure
z=(Aria_GetRotVel()/180.)*3.14159;
       //K=PH'inv(HPH'+R)
aux=ph11*pp11*ph11+ph11*pp12*ph12+ph12*pp21*ph11+ph12*pp22*ph12;
aux1=aux+pr;
aux2=1/aux1;
k1=pp11*ph11*aux2+pp12*ph12*aux2;
        k2=pp21*ph11*aux2+pp22*ph12*aux2;
//X=X+K(Z-HX)
x1=x1+K1*(z-ph11*x1-ph12*x2);
x2=x2+K2*(z-ph11*x1-ph12*x2);
omega=x1;
derivativeomega=x2/T;
//P=(I-KH)P
pp11=(1-K1*ph11)*pp11-K1*ph12*pp21;
pp12=(1-K1*ph11)*pp12-K1*ph12*pp22;
  pp21=-K2*ph11*pp11+(1-K2*ph12)*pp21;
  pp22=-K2*ph11*pp12+(1-K2*ph12)*pp22;
//X=AX
x1=x1+T*x2;
  x2=x2;
//P=APA'+Q
pp11=pp11+T*pp12+T*pp21+T*T*pp22+pq11;
  pp21=pp21+T*pp22;
  pp22=pp22+pq22;
```

```
J11=(double)((urfusion*0.011101+(lambda+(rx1*rx1/lambda))*omega*3.14159/180)
/Aria_GetVel());
        J12=(double)(-(lambda+rx1*rx1/lambda));
   J21=(double)((ulfusion*0.011101+(lambda+(rx1*rx1/lambda))*omega*3.14159/180)
/Aria_GetVel());
        J22=(double)(J12);

detJ=(double)(J11*J22-J12*J21);
Ji11=(double)(J22/detJ);
   Ji12=(double)(-J12/detJ);
  Ji21=(double)(-J21/detJ);
   Ji22=(double)(J11/detJ);
   //Flow controller
factor1=Ji21*(-urfusion*0.01101)+Ji22*(-ulfusion*0.01101);
  wrFlow=(int)((1/(kw*aw)*(-KdF*derivativeomega+KpF*factor1)+bw/(kw*aw)*
derivativeomega+omega/kw)/3.14159*180.);
  omegaVis=(float)wrFlow;
//Flow Controler Variance
if(ulsigma>ursigma) sigmaVis=ulsigma*7.;
        else sigmaVis=ursigma*7.;
        rho[0]=sigmaVis;
omegaVis=(omegaVis*180.)/PI;
if ((omegaVis>22.5)||(omegaVis<-22.5)){
omegaVis=0.001;
}
        u[0]=omegaVis;
        //End of flow Controller
        //Ultrasonics Controller
        mindist = 100000;
        for (c=0;c<16;c++)
                //tempo1=clock();
 {
    dist[c] = Aria_GetSonarRange(c);
if ((c>0)&&(c<7)){
if (dist[c]<mindist){
```

```
                                mindist = (int)dist[c];
minsen = c;
}
}
        }
dif1=(int)(dist[7]-dist[8]);
dif2=(int)(dist[15]-dist[0]);

if (dif1<dif2)
dif = dif1;
else
dif = dif2;

        if (dist[0]>=dist[15])
dizq =(int) dist[0];
else
                  dizq = (int) dist[15];
if (dist[7]>=dist[8])
dder = (int)dist[7];
else
dder = (int)dist[8];


         auxmd = ((double)mindist/1000);
 auxd = ((double)(dizq*dder)/1000000);

         if (fabs(dif)<75)
{

teta = asin(dif/230);
x  = (dder-dizq)/2000;
}
else
{
       teta=fiop;
```

```
        x=yop;
        dizq = 1000;
        dder = 1000;
}
         auxdist = (dder*dizq)/1000000;
         varifuzzy(auxdist,dif,&Re1);
         sigmaUsom=(float)Re1;
         rho[1]=sigmaUsom;
         teta=asin(dif/d);
         A1 = k1/(a1+fabs(teta));
         A2 = k2/(a2+fabs(x));
         if(teta==0)
       omegaUsom=-A1*teta-A2*x*vel/100.;
       else
       omegaUsom=-A1*teta-A2*x*vel/100.*sin(teta)/teta;
               omegaUsom=(omegaUsom*180.)/PI;
       u[1]=omegaUsom;
//End of Ultrasonics Controller

//Fusion of Control signals
//Information Filter
//global filter initial values
y=0;
Y=0;
//local filters initial values
for(i=0;i<2;i++){
   maty[i]=0;
   matY[i]=0;
}
for (c=0;c<10;c++){
   //Local filters
   for (i=0;i<2;i++){
      matz[i]=1*u[i];
      maty[i]=maty[i]+(1/rho[i])*matz[i];
      matY[i]=matY[i]+1/rho[i];
```

```
   }
   //Global Filter
   Aux1=0;
   Aux2=0;

   for (i=0;i<2;i++){
      Aux1=Aux1+maty[i];
      Aux2=Aux2+matY[i];
   }

   y=Aux1-(i-1)*y;
   Y=Aux2-(i-1)*Y;
   // local filters actualization
   for ( i=0;i<2;i++){
      maty[i]=y;
      matY[i]=Y;
   }
}
        yfinal=(1/Y)*y;
omegaFusion=yfinal;
   fprintf (pfusionvelocity, "%f\n",omegaFusion);
   fprintf(pvelocities,"omegaVis=%f omegaUsom=%f sigmaVis=%f sigmaUsom=%f

     omegaFusion=%f\n",omegaVis,omegaUsom,sigmaVis,sigmaUsom,omegaFusion);

   T_final = microseconds();

   printf(" T_final-Tinitial=%d\n", T_final -T_initial);
   printf(" T_initial=%d\n", T_initial);
 printf(" T_final=%d\n", T_final);

tempo2=clock();
printf("%f\n",(float)(tempo0)/CLOCKS_PER_SEC);
printf("%f\n",(float)(tempo1)/CLOCKS_PER_SEC);
printf("%f\n",(float)(tempo2)/CLOCKS_PER_SEC);
```

```
        printf("%f\n",(float)(tempo2-tempo0)/CLOCKS_PER_SEC);
  printf("p=%d\n",p);
cordinatex=Aria_GetX();
        cordinatey=Aria_GetY();
        phi=Aria_GetTh();
        angularvelocity=Aria_GetRotVel();
        linearvelocity=Aria_GetVel();

  fprintf (pcordinatex, "%f\n",cordinatex);
  fprintf (pcordinatey, "%f\n",cordinatey);
  fprintf (pphi, "%f\n",phi);
  fprintf (plinearvelocity, "%f\n",linearvelocity);
  fprintf (pangularvelocity, "%f\n",angularvelocity);
  fprintf(pflowvariance,"%f\n",sigmaVis);
  fprintf(pultrasonicsvariance,"%f\n",sigmaUsom);
  fprintf(pultrasonicsvelocity,"%f\n",omegaVis);
  fprintf(pflowvelocity,"%f\n",omegaUsom);
  fprintf(pleftflow,"%f\n",ulfusion);
  fprintf(prightflow,"%f\n",urfusion);


}
       //fclose(pFondateur);
       fclose(pcordinatex);
       fclose(pcordinatey);
       fclose(plinearvelocity);
       fclose(pangularvelocity);
       fclose(pflowvariance);
       fclose(pultrasonicsvariance);
       fclose(pultrasonicsvelocity);
       fclose(pflowvelocity);
       fclose(pfusionvelocity);
       fclose(pleftflow);
       fclose(prightflow);
       finalizaCapturaIm0();
```

```
printf("Leaving\n");
        Aria_Shutdown();

  return 0;
}

/*********************************************/
/* ARIFUZZY FUNCTION       */
/*********************************************/
/*Function that calculates the ultrasonics variance with
logical Fuzzy*/
void varifuzzy(double dist,double dif,double *Re1)
{
double Re1mat[2][4],Re2mat[2][4],mat[2][4];
double difvec[2],distvec[4],den;
int i,j,MP,P,G,MG;

den=0;
*Re1=0;
MP = 1;
P  = 8;
G  = 27;
MG = 64;
Re1mat[0][0] = MP;
Re1mat[0][1] = MP;
Re1mat[0][2] = G;
Re1mat[0][3] = MG;
Re1mat[1][0] = G;
Re1mat[1][1] = MG;
Re1mat[1][2] = MG;
Re1mat[1][3] = MG;
```

```
if (dif <= 30)
{
difvec[0]=1;
difvec[1]=0;
}
else if((dif>30)&&(dif<=60))
{
difvec[0]=-dif/30+2;
difvec[1]=dif/30-1;
}
else if(dif>60)
{
difvec[0]=0;
difvec[1]=1;
}

if (dist <= 0.5)
{
distvec[0]=1;
distvec[1]=0;
distvec[2]=0;
distvec[3]=0;
}
else if((dist>0.5)&&(dist<=1))
{
distvec[0]=-2*dist+2;
distvec[1]=2*dist-1;
distvec[2]=0;
distvec[3]=0;
}
else if((dist>1)&&(dist<=1.5))
{
distvec[0]=0;
distvec[1]=-2*dist+3;
distvec[2]=2*dist-2;
```

```
distvec[3]=0;
}
else if((dist>1.5)&&(dist<=2))
{
distvec[0]=0;
distvec[1]=0;
distvec[2]=-2*dist+4;
distvec[3]=2*dist-3;
}
else if(dist>2)
{
distvec[0]=0;
distvec[1]=0;
distvec[2]=0;
distvec[3]=1;
}

for (i=0;i<2;i++)
{
for (j=0;j<4;j++)
{
if(difvec[i]<distvec[j])
mat[i][j] = difvec[i];
else
mat[i][j] = distvec[j];
}
}
for (i=0;i<2;i++)
{
for (j=0;j<4;j++)
{
*Re1 = *Re1 + Re1mat[i][j]*mat[i][j];
den = den+mat[i][j];
}
}
```

```
*Re1 = *Re1/den;
}

/*********************************************/
/*            TIME FUNCTION                  */
/*********************************************/
/*Function that measures the time*/
int microseconds(void)
{
      struct timeval tval;
      int us;
      gettimeofday( &tval, NULL);
      us=tval.tv_usec;
      return us;
}
```