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

ELIETE MARIA DE OLIVEIRA CALDEIRA

NAVEGAÇÃO REATIVA DE ROBÔS MÓVEIS COM BASE NO FLUXO ÓPTICO

VITÓRIA
2002

Navegação Reativa de Robôs Móveis com Base no Fluxo Óptico

Eliete Maria de Oliveira Caldeira

Tese submetida ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Doutor em Engenharia Elétrica – Automação.

Aprovada em 11/12/2002 por:

_____
Prof. Dr. Mário Sarcinelli Filho - Orientador, UFES

_____
Prof. Dr. Hans Jörg Andreas Schneebeli - Co-orientador, UFES

_____
Prof. Dr. Mário Fernando Montenegro Campos, UFMG

_____
Prof. Dr. Márcio Rillo, USP

_____
Prof. Dr. Teodiano Freire Bastos Filho, UFES

_____
Prof. Dr. Evandro Ottoni Teatini Salles, UFES

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
Vitória, Dezembro de 2002

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

Caldeira, Eliete Maria de Oliveira, 1971-

C127n Navegação reativa de robôs móveis com base no fluxo óptico / Eliete Maria de Oliveira Caldeira. – 2002.

103 f. : il.

Orientador: Mário Sarcinelli Filho.

Co-Orientador: Hans Jörg Andreas Schneebeli.

Tese (doutorado) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Fluxo óptico. 2. Robôs móveis. 3. Veículos autônomos. 4. Desvio de obstáculos. 5. Navegação de robôs móveis. 6. Visão por computador. I. Sarcinelli Filho, Mário. II. Schneebeli, Hans Jörg Andreas. III. Universidade Federal do Espírito Santo. Centro Tecnológico. IV. Título.

CDU: 621.3

Dedico este trabalho a Deus,
a Tony Marcos, meu amado esposo,
a minhas queridas filhas Thaís e Tainá
e a meus pais Manoel (*in memorian*) e Iraci.

Agradecimentos

Vou começar agradecendo aos meus orientadores, Prof. Dr. Mário Sarcinelli Filho e Prof. Dr. Hansjörg Andreas Schneebeli, por terem acreditado que eu poderia desenvolver este trabalho e por todo o apoio que me deram em todas as fases do curso de doutorado.

Desejo também agradecer aos professores Dr. Ricardo Carelli, Dr. Oscar Nasisi e ao estudante de Doutorado Carlos Miguel Soria, do Instituto de Automática (INAUT)/ Universidad Nacional de San Juan (UNSJ), pelo suporte dado durante minha estadia em San Juan-Argentina, onde foi realizada a parte inicial deste trabalho. Gostaria de agradecer também aos demais professores, aos funcionários e aos alunos do INAUT por toda a hospitalidade.

Gostaria de agradecer ao CNPq por ter financiado este projeto através da concessão de bolsa de estudos e pelo auxílio à pesquisa para a aquisição do robô móvel Pioneer II usado nos experimentos, à CAPES pelos seis meses de bolsa de Doutorado, pelo estágio em San Juan em 1998, e pela bolsa de estágio sanduíche na UNSJ em 1999, à Fundación Antorchas da Argentina pelo apoio financeiro em minha primeira estadia em San Juan, em 1998, à fundação SETCIP da Argentina que junto à CAPES deu suporte ao projeto de cooperação entre a UFES e a UNSJ, através do qual estive por mais um tempo em San Juan em 1999. À Automática Tecnologia S.A. pela bolsa de estudos concedida nos primeiros meses deste trabalho.

Quero agradecer aos colegas e amigos do LAI (Laboratório de Automação Inteligente) com quem compartilhei bons momentos e que me deram força para superar as dificuldades do trabalho.

Gostaria de agradecer ainda aos professores e funcionários do Departamento de Engenharia Elétrica (DEL), que me acompanharam em minha caminhada durante a graduação, o mestrado e o doutorado.

A todas as pessoas que, de uma maneira ou de outra, contribuíram para que o meu trabalho se realizasse, seja com a ajuda certa, seja torcendo por mim, o meu agradecimento sincero.

Finalmente, desejo agradecer a Tony, meu esposo, por toda a compreensão, me apoiando em todas as horas, vibrando com minhas vitórias e dividindo comigo todas as minhas angústias. Agradeço também a minhas filhas Thais e Tainá que souberam entender as minhas ausências, a minha mãe e a minhas irmãs Eliana e Luciana que sempre se orgulharam do meu esforço. Afinal, foi com o apoio incondicional da minha família que eu consegui atingir este objetivo tão importante.

Resumo

Esta Tese de Doutorado propõe um sistema de sensoriamento baseado em um modelo de percepção onde as mudanças em imagens obtidas consecutivamente (por uma câmara de vídeo fixa sobre um robô móvel) permitem obter informação sobre o ambiente. Também é discutida a utilização deste sistema de sensoriamento para controlar a navegação de um robô móvel, sendo que para isto foi implementado um sistema de controle que usa exclusivamente a informação sensorial obtida a partir das imagens adquiridas. Tal sistema de controle caracteriza um comportamento reativo do robô, ou seja, assegura que ele possa vagar pelo ambiente sem colidir com nenhum objeto. Uma das características do sistema de sensoriamento proposto é que não há necessidade de modificação do ambiente, quer pela adição de texturas quer por controle de iluminação. A informação sobre o ambiente é obtida, a partir das imagens, pela técnica do fluxo óptico, o qual é sempre calculado para imagens adquiridas com o robô em movimento de translação. Então, um algoritmo de segmentação de movimento com base no fluxo óptico é proposto e implementado, de forma a detectar os objetos presentes na cena. Uma estimativa da distância a cada objeto é então obtida, calculando-se o tempo para contato a partir do fluxo óptico. O sistema de controle implementado utiliza a informação de tempo para contato a cada objeto para determinar uma nova direção segura para o robô, e envia a ele os comandos necessários para mudar sua direção atual. Os sistemas de sensoriamento e de controle aqui tratados foram implementados a bordo do robô Pioneer 2DX e utilizados em diversos experimentos, cujo resultado mostra que o sistema apresenta bom desempenho.

Abstract

This Ph. D. Thesis proposes to perform the navigation of mobile robots using a sensing system completely based on the optical flow which is calculated from a sequence of two images acquired by a camera fixed onboard the robot. To check the feasibility of using the proposed sensing system, a control system is designed to give the robot the capability of reacting to environmental changes, while executing the wander behavior with obstacle avoidance. The control system uses only data coming from the visual sensing system based on the optical flow. No modification is required to the robot's environment, such as adding suitable textures or adjusting illumination. The optical flow is calculated considering that images are acquired when the robot is moving ahead with a null angular velocity. Then, a segmentation algorithm based on the optical flow field is proposed and implemented to detect distinct objects present in the scene. The distance from the robot to each object is then estimated based on the time to impact calculated from the optical flow field. The control system uses the time to impact corresponding to each object to determine a new direction to be followed by the robot. It then sends to the robot actuators the suitable commands to change the robot's orientation. Both the sensing and control systems are implemented onboard a Pioneer 2-DX mobile robot and are used in a set of experiments, and the results show that the navigation system implemented presents very good performance.

Sumário

Capítulo 1 - Introdução

Cresce a cada dia o interesse pelo desenvolvimento de robôs que sejam capazes de substituir o trabalho humano em determinadas situações que envolvem que apresentam risco à saúde e à vida humana ou que não podem ser executadas pelo homem.

Entre as primeiras, estão as tarefas que envolvem risco imediato para o homem, como o trabalho em ambientes com presença de radiação, em campos minados, em operações de combate e de salvamento em incêndios, em zonas de grande profundidade em mares ou, ainda, que exigem manipulação de materiais tóxicos. Outras envolvem trabalhos que, se executados por muito tempo, podem causar problemas de saúde, como é o caso do trabalho em linhas de montagem, que podem causar algum tipo de lesão por esforço repetitivo, e do trabalho em indústrias químicas, onde há contínua exposição a produtos químicos.

Entre as tarefas que exigem habilidades excepcionais para o ser humano, estão a inspeção visual microscópica, a movimentação de cargas pesadas, e tarefas que necessitam de grande precisão e repetitividade, como o controle de qualidade em uma linha de montagem ou a soldagem de componentes em placas de circuito impresso.

Porém, para possibilitar a execução de determinadas tarefas, é necessário que o robô possa se locomover dentro do ambiente de trabalho. Um dos primeiros robôs móveis, chamado SHAKEY [38], foi desenvolvido no final dos anos 60, no Stanford Research Institute, para navegar em um ambiente especialmente preparado para facilitar o reconhecimento de objetos usando um sistema de visão. Depois dele, vários outros robôs móveis foram e estão sendo construídos, alguns dos quais são apresentados em [3] e [34].

Um robô móvel necessita obter informações do seu entorno, para navegar em segurança em seu ambiente de trabalho, mesmo que este seja totalmente estruturado. Ele precisa saber, por exemplo, se existem objetos e onde eles estão localizados. Além disso, em alguns locais o robô precisa detectar desníveis no piso, como a presença de escadas ou rampas. Estas informações podem ser obtidas pelo robô através do seu sistema de sensoriamento, o qual pode consistir de sensores de ultra-som, laser, luz infravermelha, luz visível, entre outros [3] [33] [34].

Neste trabalho, estamos interessados na navegação de robôs móveis utilizando sensoriamento visual. Assim, o sistema de sensoriamento deve fornecer dados sobre a presença de objetos no campo visual do robô, bem como informações para que ele possa navegar sem colidir com estes objetos. A proposta é utilizar apenas uma câmara de vídeo,

realizando um sistema de visão monocular, em uma configuração compatível com o equipamento disponível nos robôs de pequeno porte, para permitir ao robô navegar em um ambiente como o de um laboratório. Isto contrasta como o uso de visão estéreo, onde as imagens de duas câmaras de vídeo são usadas para recuperar informação tridimensional do ambiente, e com sistemas em que o ambiente é mapeado inicialmente e as informações passadas ao robô.

1.1 O Exemplo da Natureza

Como a terra recebe forte iluminação propiciada pelo sol, há uma grande facilidade de utilização da visão pelos animais. Sendo assim, este sentido foi bastante desenvolvido, tornando-se, em muitos casos, uma das principais formas de percepção do ambiente. Inclusive para o homem, a visão é um dos sentidos mais poderosos e complexos [3]. Através dela, ele é capaz de reconhecer objetos, avaliar distâncias entre objetos, estimar a própria velocidade e a de objetos ao seu redor. Sendo assim, é bastante natural pensar na implementação de um sistema de sensoriamento visual em um robô móvel.

Em um sistema de visão estéreo, como nos seres humanos, a informação sobre distância a objetos no campo visual é obtida usando as diferenças entre as imagens de um mesmo objeto, obtidas por cada um dos olhos, para estimar a profundidade em que ele se encontra [25] [41]. Para facilitar esta forma de processamento, os olhos humanos são posicionados de tal maneira que a maior parte do seu campo visual seja captada pelos dois olhos ao mesmo tempo. Assim, o campo visual do homem tem cerca de 180 graus, dos quais 140 são binoculares [41].

Essa configuração é compatível com a função natural do homem e é comum aos animais carnívoros e onívoros, que precisam de habilidades especiais para localizar precisamente suas presas. Por outro lado, as presas precisam saber se o predador está se aproximando e, para isso, precisam ter um campo visual mais amplo [41]. Sendo assim, seus olhos são posicionados de maneira a obterem um máximo de campo visual, com pouca sobreposição das imagens dos dois olhos. Os cavalos, por exemplo, podem ver cerca de 300 graus em torno de seu corpo [9] [27] [54]. Eles utilizam visão monocular e binocular. Com visão binocular, os cavalos usam ambos os olhos para focalizar um determinado objeto, similarmente ao sistema de visão humano. Por outro lado, a visão monocular é usada quando um cavalo em vigília olha para o lado e para trás. Cada olho vê uma imagem diferente, sendo que ambas são transmitidas simultaneamente ao cérebro.

De maneira semelhante, a maioria dos pássaros possui os olhos posicionados nas laterais da cabeça, com um grande campo de visão e uma pequena área de visão binocular [41]. As exceções são alguns pássaros caçadores, que necessitam de visão binocular para a determinação de distância, os quais possuem os olhos mais à frente, como os demais predadores. Porém, nos pássaros, em geral, os olhos são maiores, proporcionalmente à cabeça, quando comparados ao homem, e mais encaixados na mesma. Desta maneira, o pássaro quase não pode mover os olhos, precisando mover a cabeça a fim de modificar sua relação visual com algum objeto [23] [41].

Uma pessoa que perde a visão em um dos olhos sente uma diminuição no seu campo visual e tem que mover lateralmente a cabeça para compensar essa diferença. Além disso, a pessoa apresenta problemas com percepção de profundidade. Entretanto, ela pode aprender a

orientar-se no espaço usando características visuais que podem ser obtidas a partir de imagens monoculares [38] [42].

Para melhor entender que características são estas, considere uma pessoa caminhando por um ambiente. À medida que a pessoa se move, o brilho que chega à sua retina muda continuamente de acordo com a configuração do ambiente. Pode-se notar, por exemplo, que os objetos mais próximos a ela se movem mais rapidamente em seu campo visual do que os objetos mais distantes. A variação temporal do padrão de brilho na retina do observador é o chamado fluxo óptico, e, a partir dele, é possível, sob certas condições, recuperar a estrutura 3D do ambiente [22] [31].

De maneira semelhante, em um estudo realizado com pássaros, principalmente com pombos, foi observado que eles estimam distância e identificam objetos usando o fluxo óptico gerado quando movem suas cabeças [16]. Além disto, estudando o comportamento de abelhas durante o vôo em um corredor, notou-se que elas sempre se mantinham no meio do corredor, independentemente da sua largura. Esta observação levou à conclusão de que as abelhas medem o fluxo óptico à sua direita e à sua esquerda e que se movem no corredor tentando igualar o valor obtido nos dois lados [47]. Estes e outros estudos foram realizados com o intuito de encontrar um processo visual mais simples entre os animais para servir de modelo para implementação de sistemas de visão em dispositivos robóticos [2] [23] [47].

Os animais possuem em seus olhos células receptoras especiais otimizadas para detecção de estímulos em movimento, que funcionam independentemente em cada olho. Na verdade, muitos neurônios do sistema visual dos animais respondem mais rapidamente a um estímulo em movimento do que a um estacionário [25].

O olho humano tem dois tipos de foto-receptores: os cones, que são altamente sensíveis a cores e fornecem imagens com alta resolução espacial, e os bastonetes, que proporcionam uma imagem de baixa resolução espacial que persiste mesmo sob condições de baixa iluminação [25]. Estes receptores se conectam, ainda dentro do olho, às células ganglionares dividindo a imagem em dois canais, P ou M, de acordo com a célula ganglionar. As células ganglionares M recebem sinais de um grande número de bastonetes, possuindo boa sensibilidade a luz, ou seja, boa resolução temporal, porém tem uma baixa resolução espacial. Por outro lado, as células ganglionares P recebem estímulos de um pequeno número de cones, apresentando uma boa resolução espacial e sensibilidade a cor. Além da diferença no número e no tipo dos sinais de entrada, as células ganglionares M são maiores que as P, com terminações mais largas e que conseqüentemente conduzem sinais mais rapidamente ao nervo óptico. Assim, as células M são especializadas na percepção de movimento, bem como na detecção de estímulos súbitos a fim de redirecionar a atenção. Enquanto as células P são úteis no reconhecimento de objetos, onde padrão, cor e textura são características importantes [25]. As células M não são seletivas de direção, são apenas especializadas em detectar estímulo que varia. Por outro lado, espécies inferiores possuem células seletivas de direção nas partes mais periféricas do seu sistema visual [25].

No sistema de percepção visual humano, as células M agem principalmente na visão periférica, o que faz com que giremos a cabeça em resposta a um pequeno movimento nesta área do nosso campo visual [25]. Da mesma maneira, no cavalo estas células são responsáveis pela detecção de movimentos enquanto o mesmo utiliza visão monocular para cobrir seu extenso campo visual. Assim, ao detectar o menor movimento, o cavalo gira sua cabeça na

direção onde ele foi detectado para ver com mais clareza, usando visão estéreo, quem está se aproximando [9] [27] [41] [54].

1.2 Navegação com Sistema de Visão Monocular

Para a implementação em um robô móvel, a utilização do modelo humano em um sistema de visão estéreo é pouco interessante, devido à necessidade de utilização de duas câmaras de vídeo (fator custo) e à dificuldade de se extrair informações de pares de imagens estéreo (fator complexidade computacional). Porém, para utilizar um sistema monocular, é necessário encontrar uma forma de extrair características tridimensionais das imagens obtidas utilizando uma única câmara de vídeo. Assim, voltando ao exemplo dos animais, pode-se perguntar: quando uma pessoa perde a visão em um dos olhos fica impossibilitada de estimar distâncias a objetos no seu campo visual? Ou ainda: os animais são capazes de estimar distância e movimento relativo a objetos, utilizando a visão monocular?

Baseado nos exemplos animais apresentados observa-se que, se existe movimento relativo entre o observador e o objeto no campo visual (e desde que haja iluminação no ambiente e textura nos objetos), é possível obter informação tridimensional usando apenas visão monocular. Em princípio, é possível construir um sistema de sensoriamento visual para um robô móvel com inspiração no sistema de visão dos animais, sendo necessário apenas garantir que haja movimento relativo entre a câmara de vídeo e os objetos na cena, o que é, em geral, compatível com os objetivos deste tipo de plataforma. Este é a proposta central deste trabalho.

Para isto, é proposto que se obtenha o fluxo óptico a partir de uma seqüência de imagens adquiridas por uma câmara de vídeo instalada a bordo de um robô móvel, e se extraia daí as informações necessárias para controlar a sua navegação, fazendo com que o mesmo evite colidir com os obstáculos presentes em seu ambiente de trabalho. Note-se, inclusive, que isto equivale ao comportamento reflexo do ser humano, correspondente ao ato de desviar-se de objetos à medida que se caminha por um ambiente.

Assim, nesta tese é proposto um novo sistema de detecção de obstáculos baseado exclusivamente no uso do fluxo óptico, para ser usado no controle da navegação de um robô móvel de pequeno porte, o qual, em geral, possui capacidade restrita de processamento.

Quando se considera um sistema de controle da navegação baseado em comportamentos, ou mesmo híbrido [3], a camada de nível mais baixo implementa um comportamento totalmente reativo, o qual evita que o robô se choque com obstáculos, ou seja, é responsável pela sua integridade. Assim, para comprovar a viabilidade da utilização do sistema de sensoriamento baseado no fluxo óptico proposto, será implementado um sistema de controle da navegação imprimindo ao robô o comportamento reativo correspondente a vagar pelo ambiente sem colidir com algum objeto que porventura surja em seu caminho. Este comportamento pode ser, futuramente, associado a outros que levem o robô a executar alguma tarefa mais complexa, de acordo com algum objetivo previamente definido.

Implementando a bordo do robô um sistema de sensoriamento visual baseado no fluxo óptico, e utilizando os dados obtidos a partir deste sistema para controlar a sua navegação, podemos comprovar que, a exemplo do que acontece com os animais, é possível obter as informações necessárias para o controle da navegação de um robô móvel utilizando um sistema de visão monocular.

1.3 Definição do Problema

O problema tratado neste trabalho consiste em fazer o robô navegar evitando obstáculos usando o fluxo óptico como base do sistema sensorial constituído por uma única câmara de vídeo, ou seja, um sistema de visão monocular.

O ambiente onde o robô navega é fechado e com piso plano, e o robô não possui nenhuma informação *a priori* sobre os contornos do ambiente ou sobre a presença e localização de objetos em seu interior (um ambiente apresentando estas características é chamado semi-estruturado [3] [34]). Desta maneira, toda a informação necessária à navegação do robô deve ser obtida através do sistema de sensoriamento visual proposto.

Não deverá ser necessária nenhuma modificação do ambiente, como a adição de textura aos objetos nem controle de iluminação para facilitar o cálculo do fluxo óptico. O robô deve usar uma única câmara de vídeo e uma placa digitalizadora para a aquisição das imagens, sem nenhum recurso adicional especializado no processamento de imagens.

O controle da navegação do robô deve utilizar as informações obtidas pelo sistema de sensoriamento visual com o intuito de evitar que ele colida com os objetos presentes no ambiente, implementando no robô o comportamento vagar que é a camada de nível mais baixo em uma estrutura de controle baseada em comportamentos [3].

Além disto, todo o processamento demandado pelo sistema de sensoriamento visual, bem como pelo sistema de controle, deve ser realizado a bordo do robô, apesar da reduzida capacidade de processamento geralmente disponível. Desta forma, o robô não estará limitado a uma distância máxima de um computador de controle, nem sujeito a problemas de comunicação, seja ela por cabo ou por sinal de rádio, aumentando sua autonomia.

1.4 Trabalhos Relacionados

Muitos autores têm desenvolvido métodos para determinar, a partir de uma seqüência de imagens obtidas por uma câmara de vídeo, o movimento relativo entre a câmara e os objetos na cena [5]. A técnica utilizada, que é chamada fluxo óptico, corresponde a estimar a velocidade aparente dos *pixels* na imagem [31]. Esta técnica surgiu no final dos anos setenta, e a literatura especializada já traz vários relatos da sua aplicação na navegação de robôs móveis [1] [4] [14] [15] [17]-[19] [24] [43] [46] [48] [50].

O sistema de navegação baseado em fluxo óptico aqui proposto difere, em vários aspectos, dos sistemas para controle da navegação de robôs móveis encontrados na literatura. Em primeiro lugar, o sistema é realizado com apenas uma câmara de vídeo com um ângulo de abertura horizontal normal ($48{,}8^o$). O sistema apresentado em [16] utiliza duas câmaras: a primeira com um ângulo de visada de 60^o, responsável pela parte frontal; e a outra, uma grande angular com ângulo de visada de 115^o, responsável pela obtenção de campos periféricos. O trabalho apresentado em [43] faz uma montagem em analogia ao sistema visual das abelhas, colocando duas câmaras iguais posicionadas lateralmente. Estes sistemas apresentam como desvantagens, em relação ao sistema proposto neste trabalho, o custo de aquisição de uma segunda câmara e o aumento do tempo para processar imagens provenientes de duas câmaras. O segundo trabalho [43] apresenta ainda a desvantagem de não ter uma forma de detecção de obstáculos à frente, restringindo sua aplicação à navegação em

corredores. Um arranjo com uma câmara e dois espelhos é apresentado em [18], a fim de gerar os campos periféricos e central com apenas uma câmara, mas a solução é bastante complexa, além de restringir a parte da imagem usada para a detecção frontal. Uma outra solução é apresentada em [14], onde uma câmara grande angular é utilizada para gerar os campos visuais central e periféricos. Porém, as imagens obtidas por esse tipo de câmara são bastante distorcidas, tornando mais difícil a obtenção de informações a partir delas.

Alguns dos trabalhos encontrados na literatura [14] [16] controlam a direção para onde o eixo óptico da câmara está apontado, de forma que as rotações executadas pelo robô não sejam percebidas entre uma imagem e outra (*gaze stabilization*). Este é um processo complicado de ser implementado, e que dificulta a interpretação dos resultados obtidos, porque depende sempre de uma transformação entre os sistemas de referência da imagem e do robô. O sistema proposto neste trabalho, porém, não considera o problema da rotação. Para isto, sempre que um comando de giro é enviado ao robô, aguarda-se que ele seja completado antes de fazer novas aquisições de imagens. No sistema proposto neste trabalho, a câmara está fixa no robô, com o seu eixo óptico posicionado na direção instantânea de translação.

O sistema apresentado em [16] utiliza o PIPE (*Pipelined Image Processing Engine*) para calcular as derivadas do brilho e obter uma estimativa do fluxo normal para as imagens das duas câmaras. O sistema apresentado em [48] utiliza o MEP (*Motion Estimation Processor*) padrão MPEG 1/2, que é capaz de medir os deslocamentos dos *pixels* entre uma imagem e outra dando, portanto, uma estimativa do fluxo óptico. Os trabalhos apresentados em [15] [18] [46] utilizam uma placa da DataCube, a qual executa processamentos na imagem, como filtragem e cálculo de derivadas. O sistema proposto não requer qualquer *hardware* adicional para processamento de imagens, usando uma simples placa de digitalização de imagens.

Nos sistemas apresentados em [1] [15] [16] [18] [19] e [46] as imagens são transmitidas, seja por cabo umbilical ou sinal de radiofreqüência, a um computador externo, para processamento e extração de características, e os resultados obtidos são enviados ao robô em forma de comandos. Este tipo de sistema tem o inconveniente de restringir a autonomia do robô a regiões próximas da base, ou seja, dentro do alcance do sistema de transmissão, além dos problemas relacionados ao processo de comunicação em si. Neste trabalho todo o processamento é feito em um computador a bordo do robô.

Com relação ao método de cálculo do fluxo óptico, os sistemas propostos em [1] [4] [14] [24] [48] e [50] utilizam algoritmos de cálculo do fluxo óptico baseados em correlação. Apesar dos métodos baseados em correlação apresentarem resultados bastante satisfatórios, eles foram descartados para uso neste trabalho por serem computacionalmente muito complexos e inadequados para implementação a bordo do robô. Os métodos diferenciais se mostraram mais adequados a este tipo de plataforma, e por isto são bastante estudados. O sistema apresentado em [46] utiliza o algoritmo de Mínimos Quadrados, que foi melhorado neste trabalho [45]. O sistema utilizado em [17] para cálculo do fluxo óptico usando o modelo afim foi implementado neste trabalho, mas mostrou desempenho inferior ao apresentado pelo algoritmo de Mínimos Quadrados Modificado proposto [45], conforme será visto no Capítulo 2.

Além do cálculo do fluxo óptico, há outros aspectos no processamento que devem ser destacados. O sistema apresentado em [1] propõe um sistema de correlação bastante eficiente, porém a imagem deve ser apropriadamente filtrada antes, através da convolução da imagem com uma matriz que pode ter até 41×41 *pixels*. O sistema proposto em [4] propõe que as

imagens sejam inicialmente mapeadas por uma transformação espaço-variante, para depois calcular o fluxo óptico. Da mesma maneira, o sistema apresentado em [50] propõe que o fluxo óptico seja calculado em imagens SLOG, as quais são obtidas pelo sinal resultante da convolução da imagem com o Laplaciano da Gaussiana. O sistema aqui proposto não necessita qualquer preprocessamento na imagem, como preparação para o cálculo do fluxo óptico.

No caso dos sistemas apresentados em [14] e [17], os dados necessários ao controlador são obtidos a partir da divergência do fluxo óptico. No primeiro caso, ela é calculada a partir do fluxo normal. No segundo, para ser utilizado no cálculo da divergência, o fluxo óptico obtido é interpolado no tempo, e depois quantizado. A divergência é obtida, nos dois casos, por meio da convolução do fluxo óptico com uma série de matrizes. Alternativamente, o sistema aqui proposto executa um processamento com os vetores de fluxo óptico para fazer a segmentação. Porém, o processamento consiste em uma só varredura das regiões em que a imagem é dividida, sendo um processamento bastante simples e rápido.

Os resultados apresentados em [15] foram obtidos em simulações, os sistemas em [19] e [43] foram testados em situações restritas como em corredores, em [1] o sistema foi testado com o robô executando apenas translação, e os resultados apresentados em [18] foram obtidos em experimentos controlados, onde o fluxo óptico medido não é usado para controlar as ações de um robô. O sistema proposto neste trabalho é utilizado em um robô para a navegação no ambiente do laboratório.

Com relação ao ambiente de trabalho do robô, os sistemas apresentados em [17] [18] [43] [46] e [50] são testados em ambientes com textura nas paredes, nos objetos e, em alguns casos, no chão. O sistema implementado controla a navegação do robô sem que nenhuma preparação do ambiente seja feita. De maneira mais direta, o sistema proposto não requer adição de textura nem controle da iluminação para facilitar o cálculo do fluxo óptico.

Com relação ao método de extração de características do fluxo óptico, alguns dos sistemas utilizam a estratégia do equilíbrio, onde o robô é controlado de forma a igualar o valor de fluxo das regiões à esquerda e à direita. O sistema em [43] utiliza na comparação o valor médio do fluxo óptico nas imagens das câmaras da direita e da esquerda. Em [16] é usado o valor médio do fluxo óptico nas regiões periféricas obtidas das imagens da câmara grande angular. Nos sistemas apresentados em [24] e [50], é usado o valor médio do fluxo óptico de cada lado do foco de expansão. O sistema apresentado em [4] equilibra o fluxo óptico diretamente no comportamento implementado para manter o robô no centro de um “corredor conceitual”. A estratégia de equilíbrio não tem êxito em todas as situações possíveis no ambiente de um laboratório, não sendo capaz de levar o robô a evitar colisões com uma parede à frente, por exemplo.

Outros trabalhos procuram determinar o tempo para contato a partir do fluxo óptico [15] [18] [46] [48], ou da divergência do fluxo [14] [17].

Neste trabalho, o tempo para contato é calculado não para um determinado *pixel* ou região, mas para cada objeto resultante do processo de segmentação. Desta maneira, é possível obter um valor mais confiável do tempo para contato em cada direção do campo visual do robô. Usando estes valores, é implementado um sistema de controle que faz com que o robô gire um ângulo tal que seja suficiente para evitar os obstáculos mais próximos. A estratégia implementada consiste em determinar o menor tempo para contato em cada direção

do campo visual do robô, associando um ângulo de giro que seja suficiente para desviar do obstáculo nessa direção. Um valor único de ângulo de giro é obtido usando um filtro de Kalman descentralizado. Esta estratégia é semelhante à que foi utilizada em [46]; porém, ali apenas as regiões na linha horizontal passando pelo meio da imagem são utilizadas e o sinal de controle gerado consiste em um valor de velocidade angular.

Além disto, caracterizando os objetos resultantes do processo de segmentação em relação ao tempo para contato, ao seu tamanho e à sua posição no campo visual, é possível determinar se o robô está em rota de colisão com uma parede ou com um objeto grande, onde uma estratégia de evitar parede é executada, consistindo em fazer o robô girar 180^o, fazendo-o retornar por onde veio.

1.5 Metodologia

Neste trabalho é adotada a metodologia teórico-experimental, ou seja, é proposto um sistema de sensoriamento capaz de propiciar toda a informação necessária à navegação segura do robô, num nível teórico, e é feita uma validação experimental do referido sistema. Esta proposição e validação constituem o corpo básico deste texto.

Cada um dos subproblemas em que se divide o problema tratado neste trabalho pode ser dividido em tarefas, estabelecendo a metodologia utilizada. Assim, a obtenção do sistema de sensoriamento pode ser dividida nas seguintes partes:

- **Selecionar um algoritmo de fluxo óptico que atenda às restrições da tarefa.**

Nesta etapa foram analisados diversos algoritmos de cálculo de fluxo óptico já disponíveis na literatura, alguns dos quais foram implementados, para efeito de comparação de desempenho, usando alguns experimentos básicos, a fim de selecionar o mais adequado para a aplicação em tempo real desejada. Inclusive, foram também propostas e analisadas algumas modificações de versões clássicas de algoritmos, que mostraram desempenho adequado.

- **Obter informações sobre a presença de objetos no ambiente a partir do fluxo óptico.**

Aqui foram estudadas estratégias de segmentação de movimento baseadas no fluxo óptico já propostas na literatura, tendo sido proposta, implementada e testada uma estratégia adequada para tal segmentação.

- **Encontrar uma métrica que possa ser utilizada para determinar se os objetos no ambiente oferecem risco ao robô.**

O tempo para contato do robô a cada objeto detectado na cena pelo sistema de sensoriamento foi determinado, assim como foi calculado o fluxo óptico médio correspondente a cada um destes objetos. Estes valores dão uma estimativa indireta da distância e será usado no trabalho como medida para determinar os riscos para a navegação do robô.

O uso das informações do sistema de sensoriamento no controle da navegação do robô, para demonstrar a sua consistência, é dividido em:

- **Definir uma estratégia de controle das velocidades linear e angular do robô usando as informações do sistema visual.**

Tendo determinado os objetos na cena e uma medida da posição de cada em relação à câmara, foram estudadas diversas estratégias para o controle da navegação do robô móvel. Propôs-se, então, uma estratégia específica, a qual foi usada para projetar o sistema de controle a ser implementado a bordo.

- **Implementar o sistema visual e de controle em um robô e realizar os experimentos.**

Finalmente, o sistema global consistindo dos sistemas de sensoriamento e de controle da navegação foi implementado a bordo do robô e testado através da realização de várias experiências de navegação.

Para isto, foi utilizado o robô PIONEER 2DX, que é uma plataforma móvel a rodas com tração diferencial. Tal robô tem a bordo um computador Pentium MMX 233MHz, com 128 MBytes de memória RAM, conectado pela porta serial a um microcontrolador Siemens 88C166, cuja freqüência de relógio é 20MHz, o qual é responsável pelo controle dos motores que tracionam as rodas. Ele também possui a bordo uma câmara de vídeo CCD EVI-D30, da Sony, com ângulo de visada de $48{,}8^o$ na horizontal e de $37{,}6^o$ na vertical. A saída da câmara, um sinal de vídeo composto, passa por uma placa digitalizadora (*frame-grabber*) Imagenation modelo PXC200, fornecendo imagens em formato *bitmap* de tamanho máximo 640×480 *pixels*. O computador de bordo utiliza o sistema operacional Windows 98, e se comunica com o sistema de controle das rodas, que é executado no microcontrolador, através de uma interface gráfica incluída no programa Saphira, que faz parte da plataforma.

O ambiente para os testes de navegação é o Laboratório de Automação Inteligente (LAI), que é uma sala fechada de $4{,}5 \times 9{,}0$ m^2 com uma única porta e alguns objetos comuns a um escritório em seu interior, como mesas e cadeiras. A iluminação da sala é proveniente de lâmpadas fluorescentes colocadas no teto e de janelas localizadas em uma das paredes de 4,5m do laboratório.

1.6 Estrutura da Tese

Todas as etapas acima descritas estão contempladas no restante deste texto, da forma descrita em seguida.

O Capítulo 2 apresenta os vários algoritmos para o cálculo do fluxo óptico implementados, propõe dois novos algoritmos, e mostra o resultado da comparação entre eles, baseada em dados provenientes de alguns experimentos realizados. O capítulo é concluído com a seleção de um dos algoritmos propostos para executar o cálculo do fluxo óptico a bordo de um robô móvel de pequeno porte.

O Capítulo 3 trata do processo de segmentação da imagem baseada no fluxo óptico implementado, que resulta na caracterização de diversos objetos por área, posição na imagem e tempo para contato.

O Capítulo 4 mostra como os resultados do cálculo do fluxo óptico e da segmentação das imagens são usados para implementar um sistema de controle da navegação do robô móvel, implementando o comportamento vagar.

O Capítulo 5 descreve, em detalhes, a implementação do sistema proposto a bordo do robô móvel Pioneer 2DX, e apresenta os resultados de alguns experimentos realizados.

O Capítulo 6 apresenta os principais resultados obtidos, apontando as contribuições deste trabalho, e delineando algumas possibilidades para trabalhos futuros.

Capítulo 2 - Cálculo do Fluxo Óptico

O fluxo óptico é definido como a distribuição de velocidades aparentes do movimento do padrão de brilho através do plano da imagem, em um sistema de visão por computador, ou na retina do olho, em um sistema de visão biológico. O fluxo óptico aparece, geralmente, devido ao movimento relativo entre os objetos e a câmara (ou olho), mas pode, também, ser gerado pelo movimento de alguma das fontes de luz que iluminam a cena [31]. O campo de fluxo óptico é um campo vetorial representando o fluxo óptico em cada ponto na imagem.

Em um sistema de visão computacional, o campo de fluxo óptico pode ser obtido considerando-se uma seqüência de quadros de imagem, adquirida ao longo do tempo, e é representado associando-se um vetor (u, v) a cada *pixel*, onde u é a velocidade na direção x (horizontal) e v é a velocidade na direção y (vertical), no instante t correspondente a um dado quadro de imagem.

O campo de movimento, por sua vez, é um campo vetorial representando o movimento de pontos pertencentes a objetos no mundo real, projetados no plano da imagem. Em muitos casos, o campo de fluxo óptico e o campo de movimento são iguais ou muito semelhantes, mas podem ser muito distintos em algumas situações específicas, como, por exemplo, quando há movimento de alguma fonte de iluminação da cena. Porém, apesar de existirem tais situações, processando-se uma seqüência de imagens pode-se obter o campo de fluxo óptico. Portanto, ao contrário do campo de movimento, o campo de fluxo óptico é observável. Assim, obtendo-se o fluxo óptico e considerando-se que ele seja semelhante ao campo de movimento, é possível utilizá-lo para estimar a profundidade relativa de alguns pontos na cena, obtendo, assim, informação útil para muitas tarefas em visão computacional.

A Figura 2.1 mostra, de forma simplificada, o que são os vetores de fluxo óptico. Nesta figura, são apresentadas duas imagens do mesmo objeto (o quadrado com textura), em dois instantes diferentes, mostrando o deslocamento do objeto entre uma imagem e a outra. No quadro inferior são mostrados detalhes de uma parte da imagem, onde o vetor de fluxo óptico e suas componentes, u e v, são representados, para o *pixel* localizado no canto inferior esquerdo do objeto.

O cálculo do fluxo óptico é uma tarefa bastante complexa em termos computacionais. Por outro lado, este trabalho trata da aplicação de fluxo óptico na navegação de um robô móvel de pequeno porte, que, em geral, possui reduzida capacidade de processamento a bordo. Assim, um primeiro problema é a seleção de um algoritmo adequado para o cálculo do fluxo óptico a bordo do robô. O algoritmo selecionado deve demandar pouca memória para armazenagem de

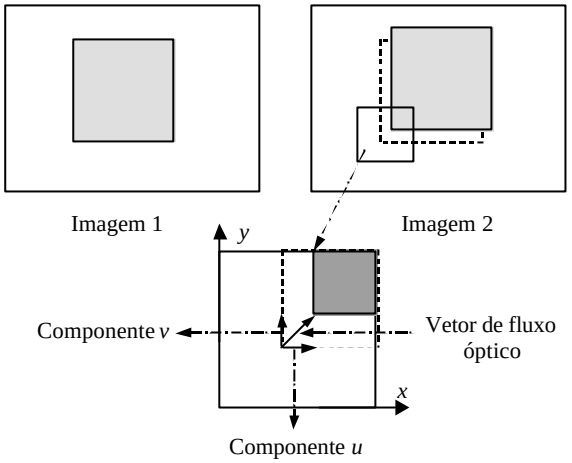


Figura 2.1 - O vetor de fluxo óptico e suas componentes.

dados e ser tão simples quanto possível em termos computacionais, mas deve gerar vetores de fluxo óptico suficientemente precisos. Desta maneira, os critérios utilizados para orientar a seleção de um algoritmo dentre os vários apresentados na literatura foram:

- o número de quadros de imagem necessários para estimar o fluxo óptico;
- a simplicidade do método;
- a robustez, principalmente no que se refere ao cálculo em imagens com oclusão e com descontinuidade, porque essas são características comuns nas imagens obtidas no ambiente de trabalho do robô.

Para encontrar um algoritmo com tais características, foram estudados vários métodos apresentados na literatura, dentre os quais alguns algoritmos foram selecionados. Estes algoritmos foram implementados e comparados usando dois experimentos básicos [44] [45], onde a câmara adquire uma seqüência de imagens de uma parede: primeiro com o robô se movendo paralelamente a ela e, depois, em um movimento de aproximação. No primeiro experimento, considerando-se que o robô se move a uma distância fixa da parede, o fluxo óptico esperado é horizontal e de módulo constante. No segundo, o tempo para contato é calculado e usado para comparar os algoritmos. Estes experimentos foram selecionados por ser possível, em ambos os casos, prever o resultado do fluxo óptico.

A seguir, são apresentadas algumas definições e características dos métodos para cálculo do fluxo óptico estudados.

2.1 Os Métodos de Cálculo do Fluxo Óptico

Vários autores propuseram métodos para a determinação do fluxo óptico, sendo que de maneira geral, este algoritmos podem ser divididos em quatro grupos principais, de acordo com o método de obtenção da estimativa do fluxo óptico [5]:

- Diferenciais ou baseados no gradiente, onde o fluxo óptico é calculado a partir das derivadas espacial e temporal do brilho da imagem. Entre estes estão os algoritmos de Horn e Schunck [31] [32], de Lai e Vemuri [35], de Lucas e Kanade [36], de Nesi et. al. [37], e de Grossmann e Santos Victor [30].
- Baseados em correlação ou *matching* por regiões, os quais definem o fluxo óptico como sendo o deslocamento de uma região entre duas imagens obtidas em instantes consecutivos. Barron, Fleet e Beauchemin em [5] comparam dois destes algoritmos,

propostos por Anandan e Singh. Também são desta classe o algoritmo desenvolvido por Ancona [1], que usa correlação 1 D em imagens pré-filtradas e o algoritmo proposto por Ted Camus em [13].

- Baseados na energia de saída de filtros *velocity-tuned*, também chamados métodos baseados em freqüência, porque o projeto destes filtros é feito no domínio de Fourier. Entre estes está o algoritmo desenvolvido por Heeger [5].
- Baseados na fase, onde o fluxo óptico é definido em termos do comportamento da fase da saída de filtros passa-banda. O algoritmo desenvolvido por Fleet e Jepson [26] se enquadra nesta classe de algoritmos, assim como os métodos baseados em *zero-crossings* (cruzamentos por zero).

Barron, Fleet e Beauchemin [5] fizeram um estudo comparativo entre diversos algoritmos de cálculo de fluxo óptico. Como resultado de seus experimentos, usando imagens sintéticas e reais, eles concluíram que os algoritmos de Horn e Schunck [31] [32], uma versão modificada dele [5] e o algoritmo de Fleet e Jepson [26] apresentaram os melhores resultados em relação à densidade do fluxo óptico e à precisão da estimativa das velocidades. O algoritmo de Mínimos Quadrados [36] também apresentou bons resultados [5]. O algoritmo de Fleet e Jepson é um método baseado em fase e os outros algoritmos são todos métodos diferenciais.

Para a aplicação em questão neste trabalho, o algoritmo de Fleet e Jepson deve ser descartado, porque usa simultaneamente muitos quadros de imagem (dezenas, em alguns exemplos [35]), demandando grande capacidade de armazenamento e apresentando uma grande complexidade de cálculos. Da mesma maneira, o algoritmo de Horn e Schunck modificado [5] se revela inadequado, porque utiliza cinco quadros de imagem no cálculo da derivada temporal, demandando muita memória e esforço computacional.

Portanto, dadas as restrições da tarefa e os critérios estabelecidos anteriormente, foram selecionados, entre os algoritmos estudados em [5], o algoritmo clássico de Horn e Schunck [31] [32] e o algoritmo de Mínimos Quadrados [5] [36], por serem simples, utilizarem apenas dois quadros de imagem e por haverem apresentado bons resultados nos testes.

Ainda com base na boa performance apresentada pelos métodos diferenciais estudados em [5] e nos critérios estabelecidos neste trabalho, também foram selecionados os algoritmos de Grossmann e Santos-Victor [30], de Paolo Nesi et al. [37] e de Lai e Vemuri [35], da literatura mais recente.

Uma análise simplificada dos fundamentos teóricos de cada um destes algoritmos, assim com uma discussão sobre a implementação feita neste trabalho, são apresentadas a seguir. Uma seqüência, com dois quadros de imagem, foi adquirida a 30 quadros por segundo com a câmara de vídeo do robô, enquanto o mesmo se movia a 60 mm/s, aproximando-se da cena. Esta seqüência foi armazenada e é usada para mostrar os resultados dos algoritmos. A seqüência mostra as imagens de uma parede revestida com um papel apresentando uma textura, o que favorece o cálculo do fluxo óptico, ainda que o desenho contenha muitos contornos, que fazem aumentar o problema de abertura.

2.2 Métodos Baseados no Gradiente

Os métodos selecionados para serem implementados neste trabalho são, como dito anteriormente, métodos diferenciais ou baseados no gradiente. Nestes métodos considera-se

que o brilho da imagem é constante no tempo. Assim, caracterizando-se as imagens pelo padrão de brilho $I(x,y,t)$, onde x e y são as coordenadas de um *pixel* em um quadro de imagem e t designa o tempo de aquisição de um quadro de imagem específico em uma seqüência, pode-se escrever

$$\frac{dI(x,y,t)}{dt} = 0 \quad \text{ou} \quad I_x u + I_y v + I_t = 0 \tag{2.1}$$

onde I_x, I_y e I_t são as derivadas do brilho em relação a x, y e ao tempo, e u e v são as componentes horizontal e vertical do vetor de fluxo óptico, para o *pixel* (x,y) sob consideração, no instante t (ver Figura 2.1). Vale ressaltar que I_x, I_y e I_t assim como u e v são funções de x, y e t, porém esta informação está sendo omitida por questão de simplicidade.

Entretanto, a equação (2.1), chamada *restrição do fluxo óptico*, não é suficiente para determinar u e v em qualquer *pixel* da imagem. Isto caracteriza um problema mal condicionado, e para determinar u e v deve-se adicionar uma outra restrição. Por outro lado, esta equação pode ser usada para determinar a componente do fluxo óptico em cada *pixel* na direção do gradiente espacial da imagem f_n. Isto pode ser visto, analiticamente, se a equação for reescrita como

$$\begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u & v \end{bmatrix}^T = \begin{bmatrix} -I_t \end{bmatrix} \tag{2.2}$$

$$(\nabla I)^T . \begin{bmatrix} u & v \end{bmatrix}^T = \begin{bmatrix} -I_t \end{bmatrix} \tag{2.3}$$

$$\frac{(\nabla I)^T . \begin{bmatrix} u & v \end{bmatrix}^T}{\|\nabla I\|} = f_n = -\frac{I_t}{\|\nabla I\|} \tag{2.4}$$

onde $\|\nabla I\| = \sqrt{I_x^2 + I_y^2}$ é a norma do gradiente de I. A componente f_n é chamada fluxo óptico normal, porque tem a mesma direção do gradiente espacial da imagem, o qual é normal à direção espacial ao longo da qual o brilho na imagem permanece constante, ou seja, normal aos contornos da imagem.

Os algoritmos escolhidos para a implementação neste trabalho podem ser divididos em dois grupos, de acordo com a forma como eles inserem uma nova restrição ao fluxo óptico, para determiná-lo. No primeiro grupo, é adicionada uma restrição de suavidade global e, no segundo, a imagem é dividida em pequenas regiões, dentro das quais é feita alguma suposição para o fluxo óptico.

2.3 Algoritmos que Usam Critério de Suavidade Global

Alguns dos algoritmos estudados neste trabalho adicionam uma restrição de suavidade global ao sistema, a fim de determinar de maneira única os vetores de fluxo óptico. O que se considera, em geral, é que o fluxo óptico varia suavemente por toda a imagem. Dentre os algoritmos implementados, os que utilizam uma restrição deste tipo são apresentados a seguir.

2.3.1 Algoritmo de Suavidade Global

O primeiro método implementado, o algoritmo clássico de Horn e Schunck [31] [32], utiliza uma restrição de suavidade global, que se resume a minimizar a soma dos laplacianos de *u* e *v*. Isto significa que a soma

$$\nabla^2 u + \nabla^2 v = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \tag{2.5}$$

deve ser minimizada.

Em sua implementação, Horn e Schunck calculam o laplaciano para cada *pixel* como [31]

$$\nabla^2 u_{i,j} \approx 3\left(\bar{u}_{i,j} - u_{i,j}\right) \quad \text{e} \quad \nabla^2 v_{i,j} \approx 3\left(\bar{v}_{i,j} - v_{i,j}\right) \tag{2.6}$$

onde *i* e *j* são, respectivamente, a linha e a coluna do *pixel* sob análise, e $\bar{u}_{i,j}$ e $\bar{v}_{i,j}$ são os valores médios das componentes do fluxo óptico tomados na vizinhança do *pixel* sob consideração, sendo calculados como

$$\begin{aligned} \bar{u}_{i,j} &= \frac{1}{6}\left(u_{i-1,j} + u_{i,j+1} + u_{i+1,j} + u_{i,j-1}\right) + \frac{1}{12}\left(u_{i-1,j-1} + u_{i-1,j+1} + u_{i+1,j+1} + u_{i+1,j-1}\right) \\ \bar{v}_{i,j} &= \frac{1}{6}\left(v_{i-1,j} + v_{i,j+1} + v_{i+1,j} + v_{i,j-1}\right) + \frac{1}{12}\left(v_{i-1,j-1} + v_{i-1,j+1} + v_{i+1,j+1} + v_{i+1,j-1}\right) \end{aligned} \tag{2.7}$$

Os valores de $u_{i,j}$ e $v_{i,j}$ podem ser, então, obtidos iterativamente para cada *pixel* de coordenadas (*i,j*), através das equações [31]

$$\begin{aligned} u^{n+1} &= \bar{u}^n - I_x \left(I_x \bar{u}^n + I_y \bar{v}^n + I_t\right) \Big/ \left(a^2 + I_x^2 + I_y^2\right) \\ v^{n+1} &= \bar{v}^n - I_y \left(I_x \bar{u}^n + I_y \bar{v}^n + I_t\right) \Big/ \left(a^2 + I_x^2 + I_y^2\right) \end{aligned} \tag{2.8}$$

onde a é uma constante, $n+1$ é a iteração atual, n é a iteração anterior e u^0 e v^0 são valores inicialmente atribuídos às componentes do vetor de fluxo óptico do *pixel* sob consideração.

Na equação (2.8), as derivadas do brilho em cada *pixel*, I_x, I_y e I_t, são aproximadas por diferenças finitas [31], ou seja

$$\begin{aligned} I_x &= \left(I_{i,j+1,k} - I_{i,j,k} + I_{i+1,j+1,k} - I_{i+1,j,k} + I_{i,j+1,k+1} - I_{i,j,k+1} + I_{i+1,j+1,k+1} - I_{i+1,j,k+1}\right) \\ I_y &= \left(I_{i+1,j,k} - I_{i,j,k} + I_{i+1,j+1,k} - I_{i,j+1,k} + I_{i+1,j,k+1} - I_{i,j,k+1} + I_{i+1,j+1,k+1} - I_{i,j+1,k+1}\right) \\ I_t &= \left(I_{i,j,k+1} - I_{i,j,k} + I_{i+1,j,k+1} - I_{i+1,j,k} + I_{i,j+1,k+1} - I_{i,j+1,k} + I_{i+1,j+1,k+1} - I_{i+1,j+1,k}\right) \end{aligned} \tag{2.9}$$

onde *i* e *j* são as coordenadas do *pixel* sob consideração, e *k* identifica cada quadro de imagem.

O inconveniente deste procedimento iterativo para a determinação de *u* e *v*, é que ele demanda um número de iterações que depende das condições iniciais. Horn e Schunck sugerem que inicialmente seja utilizado o valor $u^0 = 0$ e $v^0 = 0$ para cada *pixel*. Entretanto, à

medida que novos quadros de imagem vão sendo processados, os valores obtidos para a seqüência atual podem ser usados como condição inicial do fluxo óptico para a próxima, reduzindo assim o número de iterações e tornando o método mais atrativo.

Na implementação feita neste trabalho, utiliza-se um valor de $a = 0{,}5$, como sugerido em [31], os valores iniciais das componentes dos vetores de fluxo óptico em cada *pixel* são sempre iguais a zero ($u^0 = 0$ e $v^0 = 0$) e o número de iterações n_{iter} é igual a 10.

A Tabela 2.1 apresenta os custos computacionais associados à obtenção da estimativa do fluxo óptico pelo método de Horn e Schunck para todos os *pixels* da imagem. Nesta tabela, são apresentados, separadamente, os custos das etapas de cálculo das derivadas do brilho e de obtenção da estimativa pelo processo iterativo, sendo que, na segunda coluna, o custo é representado como o número de somas e produtos, e na terceira coluna, como a função O(n) correspondente.

Tabela 2.1 - Complexidade do algoritmo de Horn e Schunck

Etapa considerada	Custos	Complexidade
Cálculo das derivadas	21.m.n somas + 3.m.n produtos	O(m.n) somas + O(m.n) produtos
Iterações do método	20.m.n.n_{iter} somas + 12.m.n.n_{iter} produtos	O(n_{iter}.m.n) somas + O(n_{iter}.m.n) produtos
Custo total*	67.891.200 somas + 37.785.600 produtos	

m = número de colunas da imagem
n = número de linhas da imagem
n_{iter} = número de iterações
* Custo obtido para m = 640, n = 480 e n_{iter} = 10

A Figura 2.2 apresenta os resultados do cálculo do fluxo óptico usando o método de Horn e Schunck [31] para a seqüência de teste descrita anteriormente, para a qual espera-se que os vetores de fluxo óptico sejam radiais. Nesta figura, os vetores de fluxo óptico são desenhados sobre o primeiro quadro de imagem da seqüência e um pequeno círculo representa o foco de expansão, ou seja, o ponto de onde todos os vetores de fluxo óptico partem. As coordenadas do foco de expansão foram estimadas a partir dos vetores de fluxo óptico. Sendo assim, a precisão da sua determinação depende da qualidade dos vetores de fluxo óptico obtidos.

2.3.2 Algoritmo de Suavidade Global com Seguimento de Contornos

Em áreas onde a função de brilho apresenta descontinuidades, a restrição do fluxo óptico não se verifica, e o algoritmo de Horn e Schunck [31] apresenta problemas. Estas áreas ocorrem com freqüência em imagens adquiridas durante a navegação de um robô como, por exemplo, em imagens que mostram objetos em diferentes profundidades, e nos contornos, de maneira geral.

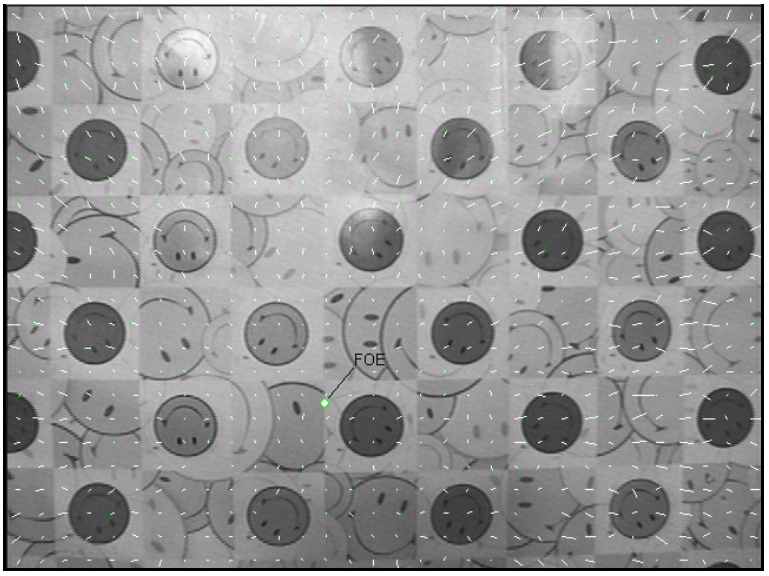


Figura 2.2 - Fluxo óptico obtido pelo método de Horn e Schunck para a seqüência de teste.

Para sobrepor este problema, Lai e Vemuri [35] sugerem um algoritmo mais robusto do que o de Horn e Schunck [31], o qual usa a restrição do fluxo óptico em regiões da imagem onde o brilho varia suavemente, e o que eles chamam de restrição do fluxo óptico baseada em contornos nos locais das descontinuidades.

Os métodos baseados em contornos computam o fluxo óptico ao longo dos *zero-crossings* usando a seguinte restrição para o fluxo óptico [35]

$$S_x u + S_y v + S_t = 0 \tag{2.10}$$

onde

$$S = \nabla^2 G * I \tag{2.11}$$

é a convolução entre o laplaciano do filtro gaussiano 2D e o brilho da imagem I, e S_x, S_y e S_t são, respectivamente, as derivadas de S nas direções x e y e no tempo. Novamente, deve-se notar que S_x, S_y e S_t são funções de x, y e t.

O algoritmo resultante minimiza a função [35]

$$\sum_p \left(E_x u + E_y v + E_t\right)^2 + l\left(u_x^2 + u_y^2 + v_x^2 + v_y^2\right) \tag{2.12}$$

onde u_x, u_y, v_x e v_y são as derivadas das componentes do fluxo óptico nas direções x e y, p denota cada um dos *pixels* no quadro de imagem e a magnitude de l reflete a influência do termo que representa a restrição de suavidade. Nesta equação, $E = b_1 I$ quando o *pixel* p não está em uma região de descontinuidade e $E = b_2 S$ quando o *pixel* p está em um contorno da imagem, sendo que b_1 e b_2 são constantes. Os contornos podem ser obtidos fazendo-se uma operação de *zero-crossing* (cruzamentos por zero) em S, ou seja, verificando-se os *pixels* onde S muda de sinal.

Nesta formulação, o quadrado do módulo do gradiente espacial ($I_x^2 + I_y^2$ ou $S_x^2 + S_y^2$) é usado como peso em cada *pixel p*, o que é, segundo os autores, um contra-senso, porque a restrição do fluxo óptico não é mais confiável em locais onde o gradiente espacial é maior, nem menos confiável onde ele é menor [35]. Sendo assim, eles sugerem que as derivadas de *I* e *S* (I_x, I_y e I_t ou S_x, S_y e S_t) sejam normalizadas, dividindo-as pelo módulo do gradiente espacial adequado ($\sqrt{I_x^2 + I_y^2}$ ou $\sqrt{S_x^2 + S_y^2}$) em cada *pixel*, gerando $\overline{E}_x$, $\overline{E}_y$ e $\overline{E}_t$.

Lai e Vemuri [35] propõem a utilização do método do Gradiente Conjugado com Pré-Condicionador Incompleto de Cholesky para resolver o sistema **KU** = **B,** onde $\mathbf{K} \in \Re^{2n^2 \times 2n^2}$. Porém, mesmo trabalhando com a estrutura em blocos proposta esta solução se torna proibitiva. Por exemplo, uma área quadrada de 480×480 *pixels*, obtida de uma imagem de 640×480 *pixels*, gera um sistema descrito por uma matriz 460.800×460.800!

Sendo assim, optou-se por determinar os vetores de fluxo óptico usando a solução iterativa proposta por Horn e Schunck [31]. Desta maneira, as componentes do vetor de fluxo óptico são determinadas para cada *pixel* da imagem como

$$\begin{aligned} u^{n+1} &= \overline{u}^n - \overline{E}_x \left(\overline{E}_x \overline{u}^n + \overline{E}_y \overline{v}^n + \overline{E}_t \right) \Big/ \left(\alpha^2 + \overline{E}_x^2 + \overline{E}_y^2 \right) \\ v^{n+1} &= \overline{v}^n - \overline{E}_y \left(\overline{E}_x \overline{u}^n + \overline{E}_y \overline{v}^n + \overline{E}_t \right) \Big/ \left(\alpha^2 + \overline{E}_x^2 + \overline{E}_y^2 \right) \end{aligned} \tag{2.13}$$

onde $\overline{u}$ e $\overline{v}$ são os valores médios das componentes do fluxo óptico, tomados na vizinhança do *pixel* sob consideração, como mostrado nas equações (2.7), α é uma constante, $n+1$ é a iteração atual, n é a iteração anterior e u^0 e v^0 são valores inicialmente atribuídos às componentes do vetor de fluxo óptico em cada *pixel* da imagem.

Um aspecto computacional importante deste algoritmo é que ele demanda calcular e armazenar a matriz *S*, dada pela equação (2.11), para cada quadro de imagem na seqüência. Além disso, é necessário detectar, em cada imagem, os *pixels* onde *S* muda de sinal, ou seja, os *zero-crossings*, e calcular as derivadas de *S* nestes pontos.

A máscara utilizada na determinação de *S* é quadrada, e seu tamanho *w* é dado em função do valor de desvio padrão *s* da função gaussiana, através da expressão $w = 2(3{,}35s + 0{,}33) + 1$. Neste trabalho, foi usado o valor de desvio padrão $s = 2$, resultando em uma máscara de tamanho 15×15. Assim, a obtenção de *S* é realizada pela convolução desta máscara com o brilho da imagem *I*.

Na implementação deste algoritmo, da mesma maneira que no anterior, o valor de a é dado por $a = 0{,}5$, os valores iniciais das componentes dos vetores de fluxo óptico são zero ($u^0 = 0$ e $v^0 = 0$) e o número de iterações n_{iter} também é fixo em 10. Além disso, os valores de b_1 e b_2 são, ambos, unitários. As derivadas de *I* são calculadas por diferenças finitas, como mostrado nas equações (2.9) e as derivadas de *S* são calculadas de maneira similar.

A Tabela 2.2 apresenta os custos de obtenção da estimativa do fluxo óptico em cada *pixel* da imagem por este método. Esta tabela apresenta, separadamente, os custos de diversas etapas do processo de obtenção da estimativa do fluxo óptico. Na segunda coluna da tabela são apresentados os custos de cada etapa do processamento em termos do número de

operações de soma e produto necessários e na última coluna é apresentada a função O(n) correspondente.

Tabela 2.2 - Complexidade do algoritmo de Lai e Vemuri

Etapa considerada	Custos	Complexidade
Obtenção de S_1 e S_2	$2.m.n.w^2$ somas $+ 2.m.n.w^2$ produtos	$O(m.n.w^2)$ somas $+ O(m.n.w^2)$ produtos
Obtenção de *zero-crossings*	$4.m.n$ produtos	$O(m.n)$ produtos
Cálculo das derivadas	$21.m.n$ somas $+ 3.m.n$ produtos	$O(m.n)$ somas $+ O(m.n)$ produtos
Normalização das derivadas	$m.n$ somas $+ 6.m.n$ produtos	$O(m.n)$ somas $+ O(m.n)$ produtos
Iterações do método	$20.m.n.n_{iter}$ somas $+ 12.m.n.n_{iter}$ produtos	$O(m.n.n_{iter})$ somas $+ O(m.n.n_{iter})$ produtos
Custo total*	206.438.400 somas + 179.097.600 produtos	

m = número de colunas da imagem
n = número de linhas da imagem
n_{iter} = número de iterações
w = tamanho da máscara Laplaciano da Gaussiana
* Custo obtido para m = 640, n = 480, w = 15 e n_{iter} = 10

A Figura 2.3 apresenta o resultado do cálculo do fluxo óptico obtido pelo método de Lai e Vemuri, para a seqüência de teste adquirida pela câmara CCD a bordo do robô. Para este algoritmo, o fluxo óptico obtido é radial com esperado. Nesta figura, os vetores de fluxo óptico são desenhados sobre o primeiro quadro de imagem da seqüência e um pequeno círculo representa o foco de expansão, ou seja, o ponto de onde todos os vetores de fluxo óptico partem. As coordenadas do foco de expansão são calculadas a partir dos vetores de fluxo óptico obtidos pelo método.

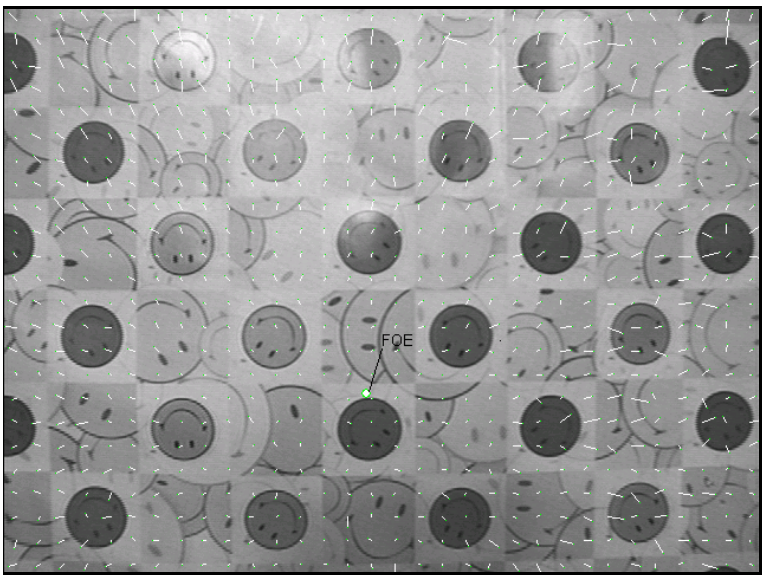


Figura 2.3 - Fluxo óptico obtido pelo método de Lai e Vemuri para a seqüência de teste.

2.4 Algoritmos que Usam Critério de Suavidade Local

Os outros algoritmos selecionados da literatura para executar o cálculo do fluxo óptico dividem a imagem em regiões, e fazem considerações sobre ele apenas em cada região. O

fluxo óptico pode ser considerado constante ou linear dentro da região sob consideração, ou seja, sua derivada primeira pode ser considerada nula ou então constante na região sob análise. Os algoritmos implementados, usando tais considerações sobre o fluxo óptico, são apresentados a seguir.

2.4.1 Algoritmo de Fluxo Óptico Constante com Solução de Mínimos Quadrados

O método proposto por Lucas e Kanade [5] [36] considera que o fluxo óptico é constante em uma região de N×N *pixels*. Portanto, escrevendo-se a restrição de fluxo óptico (equação (2.1)) para cada *pixel* na região, obtém-se o conjunto de equações dado por

$$I_{xp}u + I_{yp}v + I_{tp} = 0 \text{ para } p = 1,..., \mathrm{N}\times\mathrm{N} \tag{2.14}$$

Estas equações são usadas para gerar uma estimativa de mínimos quadrados do vetor de fluxo óptico nesta região. A estimativa $(\hat{u}, \hat{v})$ pode ser obtida por esse método como

$$(\hat{u}, \hat{v}) = \left(\mathbf{A}^T\mathbf{W}^2\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{W}^2\mathbf{b} \tag{2.15}$$

onde a matriz $\mathbf{A}$ é dada por

$$\mathbf{A} = \begin{bmatrix} I_{x_1}, I_{x_2}, ..., I_{x_{\mathrm{N}\times\mathrm{N}}} \\ I_{y_1}, I_{y_2}, ..., I_{y_{\mathrm{N}\times\mathrm{N}}} \end{bmatrix}^T \tag{2.16}$$

o vetor $\mathbf{b}$ é dado por

$$\mathbf{b} = -\left[I_{t_1}, I_{t_2}, ..., I_{t_{\mathrm{N}\times\mathrm{N}}}\right]^T \tag{2.17}$$

e $\mathbf{W}^2$ é uma matriz que atribui pesos aos valores na vizinhança do *pixel* sob consideração.

Neste trabalho, este método foi implementado usando $\mathbf{W}^2 = \mathbf{a}^T * \mathbf{a}$ onde

$$\mathbf{a} = [0{,}0625, \quad 0{,}25, \quad 0{,}375, \quad 0{,}25, \quad 0{,}0625] \tag{2.18}$$

como sugerido em [5], o que corresponde a usar N = 5, ou seja, regiões de 5×5 pixels. Note-se que a influência dos *pixels* é aumentada à medida que se caminha para o *pixel* central da região. Um fator digno de nota, neste ponto, é que não é trivial obter uma matriz de pesos adequada, quando se aumenta o valor de N. Por outro lado, o tempo de cálculo melhora sensivelmente quando N cresce.

A solução apresentada na equação (2.15) é chamada solução de Mínimos Quadrados Ponderada e uma solução chamada simplesmente solução de Mínimos Quadrados pode ser obtida, considerando-se uma matriz $\mathbf{W}^2$ que atribua o mesmo peso a todos os *pixels* na vizinhança. Neste caso a estimativa $(\hat{u}, \hat{v})$ é dada por

$$(\hat{u},\hat{v}) = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b} \qquad (2.19)$$

onde **A** e **b** são dados pelas equações (2.16) e (2.17), respectivamente. Utilizando-se esse algoritmo, pode-se variar livremente o tamanho da região.

As Figuras 2.4 e 2.5 mostram os resultados obtidos para a seqüência de imagens de teste pelo algoritmo de Mínimos Quadrados Ponderado e pelo algoritmo de Mínimos Quadrados.

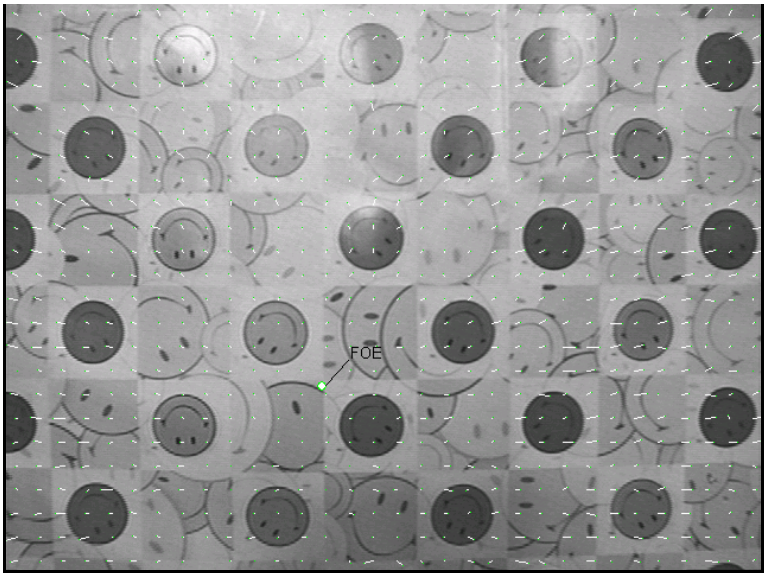


Figura 2.4- Fluxo óptico obtido pelo método de Mínimos Quadrados Ponderado para a seqüência de teste (N = 5).

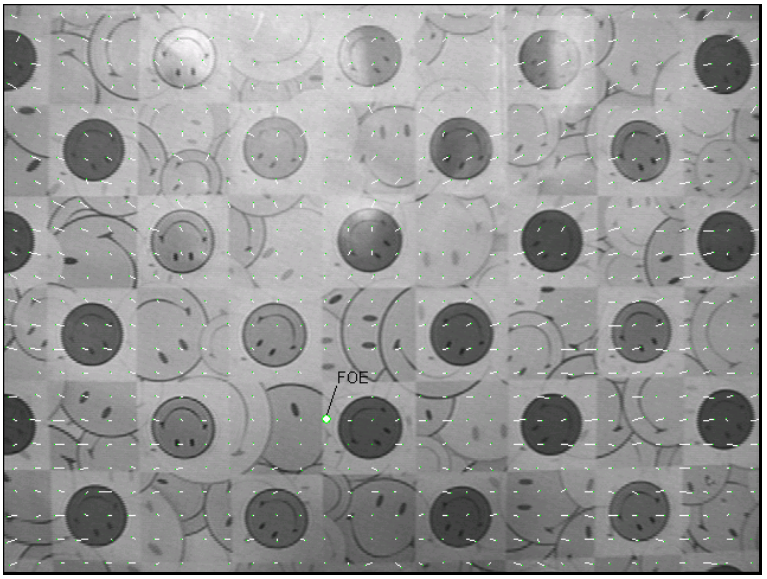


Figura 2.5- Fluxo óptico obtido pelo método de Mínimos Quadrados para a primeira seqüência de teste (N = 5).

O fluxo óptico foi calculado considerando-se regiões com N = 5 nos dois casos, para que fosse possível comparar os resultados. Nas duas figuras os vetores de fluxo óptico são desenhados sobre a primeira imagem da seqüência.

Em relação ao método de Mínimos Quadrados, pode-se dizer que ele se mostra muito atrativo quando se considera o custo computacional necessário para se obter a estimativa do fluxo óptico. Isto se deve ao fato deste não ser um método iterativo, como o de Horn e Schunck [31], e à simplicidade do método em si, que envolve apenas a inversão de uma matriz 2×2, independentemente do tamanho da região considerada. É importante notar que este método se torna mais atrativo quando o tamanho da região é aumentado, o que é mais simples de ser feito utilizando a versão sem pesos do algoritmo. Entretanto, há um compromisso entre o tamanho da região e a precisão da estimativa, porque quanto maior a região maior a probabilidade de erro na estimativa do fluxo óptico obtida para ela, devido à suposição de que o fluxo óptico é constante em toda a região. Além disto, é importante notar que as derivadas devem ser calculadas em todos os *pixels* da imagem, independentemente do tamanho da região. Sendo assim, mesmo aumentando o valor de N, a redução no tempo de cálculo ainda é limitada. Esta afirmação pode ser comprovada através da Tabela 2.3 que apresenta os custos de obtenção da estimativa do fluxo óptico para o algoritmo de Mínimos Quadrados, em cada etapa do processamento, em função do tamanho das imagens e de N. Na segunda coluna desta tabela são apresentados os custos em termos do número de somas e produtos e na última coluna é apresentada a função O(n) correspondente. Os custos da versão ponderada do algoritmo são maiores do que os apresentados na Tabela 2.3 especificamente na segunda etapa de obtenção da matriz $\mathbf{A}^T\mathbf{W}^2\mathbf{A}$ e do vetor $\mathbf{A}^T\mathbf{W}^2\mathbf{b}$.

Tabela 2.3 - Complexidade do algoritmo de Mínimos Quadrados

Etapa considerada	Custos	Complexidade
Cálculo das derivadas	21.m.n somas + 3.m.n produtos	O(m.n) somas O(m.n) produtos
Obtenção da matriz $\mathbf{A}^T\mathbf{A}$ e do vetor $\mathbf{A}^T\mathbf{b}$	6.m.n somas + 6.m.n produtos	O(m.n) somas + O(m.n) produtos
Obtenção de u e v	$3.m.n/N^2$ somas + $10.m.n/N^2$ produtos	$O(m.n/N^2)$ somas + $O(m.n/N^2)$ produtos
Custo total*	8.295.936 somas + 2.772.480 produtos	
m = número de colunas da imagem n = número de linhas da imagem N = número de linhas e colunas de cada região * Custo obtido para m = 640, n = 480 e N = 20		

2.4.2 Algoritmo de Fluxo Óptico Constante com Solução por Votação

A fim de reduzir os erros de uma estimativa de mínimos quadrados, no sentido de que ela tende a produzir uma solução média em vez da solução mais provável, Nesi et al. [37] propuseram um método que eles chamaram solução multiponto com estimador de máxima verossimilhança. Este método é baseado em uma versão modificada da Transformada de Hough Combinacional [7] e, nos estudos realizados, os autores concluíram que ele gera a solução de máxima verossimilhança, mesmo em alguns casos críticos [37].

O algoritmo proposto por Nesi et al. é similar ao algoritmo de Mínimos Quadrados [5] [36], no sentido em que ele também supõe um valor constante dos vetores de fluxo óptico em uma região de N×N *pixels*. Assim, o mesmo conjunto de equações dado em (2.14) pode ser obtido. Para tais equações, isolando-se *u* obtém-se o conjunto de equações

$$u = -\frac{I_{yp}}{I_{xp}}v - \frac{I_{tp}}{I_{xp}} \text{ para } p = 1,2,...,\mathrm{N}\times\mathrm{N} \text{ e para } I_{xp} \neq 0 \quad (2.20)$$

que pode ser reescrito como

$$u = m_i v + c_i \text{ para } i = 1,2,...,\mathrm{N}\times\mathrm{N} \quad (2.21)$$

Portanto, cada *pixel* na região determina uma reta no plano $u\times v$, cuja inclinação é dada por m_i e cujo ponto de interseção com o eixo v é dado por c_i. Porém, ao serem representadas no espaço dos parâmetros ($m\times c$), cada equação corresponde a um ponto (m_i, c_i).

Assim, a estimativa do fluxo óptico na região considerada corresponde ao ponto comum a todas as linhas geradas no plano $u\times v$, ou seja, ao ponto onde todas as linhas se interceptam. Por outro lado, no plano $m\times c$, o fluxo óptico pode ser estimado a partir da reta que melhor aproxima a série de pontos (m_i, c_i). No método proposto por Nesi et al. [37] faz-se a determinação desta reta considerando um *pixel* central e uma região quadrada com lado N ≥ 3 (N ímpar). A partir da reta, determina-se o vetor de fluxo óptico para este *pixel*. Então, o *pixel* seguinte é considerado, e o processo é repetido até que o fluxo óptico seja determinado para todos os *pixels* na imagem.

Pelo método proposto por Nesi et al. [37], os parâmetros da reta que melhor representa os pontos são escolhidos por um processo de votação. Inicialmente, a inclinação mais provável da reta é determinada pelo processo de votação. Nesta fase, o processo consiste em gerar uma reta no plano $m\times c$, usando o ponto gerado pelo *pixel* central da região N×N e cada um dos pontos gerados pelos outros *pixels* na região, e em verificar qual a inclinação que aparece mais vezes. O passo seguinte consiste em passar uma reta com a inclinação mais provável, obtida no passo anterior, em cada ponto no plano $m \times c$, e determinar a distância mais provável da reta à origem, também por meio de votação. O fluxo óptico para o *pixel* central da região N×N pode ser então calculado a partir da reta obtida neste processo.

A Figura 2.6 apresenta o resultado obtido com a implementação do algoritmo de Nesi et

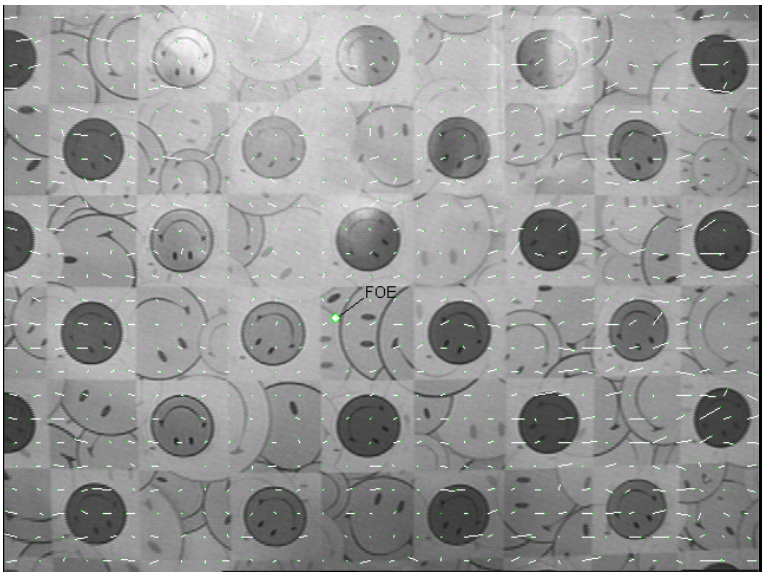


Figura 2.6 - Fluxo óptico obtido pelo método de Nesi et al. para a seqüência de teste (N = 5).

al., calculando-se o fluxo óptico para cada *pixel*, para a seqüência de imagens de teste, usando um valor de N = 5.

Uma outra implementação foi feita para este método, onde se considera que o vetor de fluxo óptico determinado usando a vizinhança N×N em torno de um certo *pixel* é válido para todos os *pixels* na vizinhança. Assim, uma nova vizinhança é escolhida como a região contígua à região já considerada. Desta maneira, o algoritmo é muito mais rápido, porque o processo de determinação da reta mais provável é executado um número menor de vezes, mas os resultados são muito piores, em termos da exatidão do fluxo óptico calculado, e por isso essa solução foi descartada.

2.4.3 Algoritmo de Fluxo Óptico Afim

O método proposto por Grossmann e Santos-Victor [30] se baseia na consideração de que o fluxo óptico em determinada região pode ser aproximado por um modelo afim dado por

$$\begin{aligned} u(x,y) &= u_0 + (x - x_0)u_x + (y - y_0)u_y \\ v(x,y) &= v_0 + (x - x_0)v_x + (y - y_0)v_y \end{aligned} \tag{2.22}$$

onde x_0 e y_0 são as coordenadas de um *pixel* de referência, u_0 e v_0 são as componentes do fluxo óptico neste *pixel*, e u_x, u_y, v_x e v_y são valores constantes, correspondentes às derivadas das componentes do vetor de fluxo óptico nas direções x e y. Esta é uma aproximação, uma vez que, em geral, não há vetor afim que satisfaça a restrição de brilho constante expressa pela equação (2.1) [30].

Em seu artigo, Grossmann e Santos-Victor [30] definem o fluxo óptico em seqüências discretas de imagens como

$$I(x+u, y+v, t+1) = I(x, y, t) + r \tag{2.23}$$

onde r representa um termo de ruído. Esta equação envolve apenas valores de brilho da imagem e não derivadas, assim como nas técnicas baseadas em correlação. Porém, a implementação do algoritmo é feita utilizando a equação

$$-I_t = I_x u + I_y v + g \tag{2.24}$$

que é uma versão mais realista da equação (2.1), utilizada nos métodos diferenciais, em que se considera um termo g correspondente ao ruído aditivo aleatório. Estas equações vão ser utilizadas para determinar as equações de observação, usadas na obtenção dos vetores de fluxo óptico afim.

Considerando a aproximação de primeira ordem da imagem obtém-se

$$I(x+u, y+v, t+1) = I(x+\tilde{u}, y+\tilde{v}, t+1) + \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u-\tilde{u} & v-\tilde{v} \end{bmatrix}^T + h(u-\tilde{u}, v-\tilde{v}) \tag{2.25}$$

onde $(\tilde{u}, \tilde{v})$ é um par arbitrariamente escolhido e h é o erro da expansão de primeira ordem.

Combinando-se as equações (2.24) e (2.25) obtém-se

$$I(x,y,t)-I(x+\tilde{u},y+\tilde{v},t+1)+I_x\tilde{u}+I_x\tilde{v}+h(u-\tilde{u},v-\tilde{v})+g=I_xu+I_xv \tag{2.26}$$

Uma vez que os quadros de imagem nos instantes sucessivos *t* e *t+1* são obtidos, as derivadas I_x e I_y podem ser calculadas. Assim, os únicos termos desconhecidos desta equação são *h* e *g*, que são a parte aleatória do modelo de observação. Grossmann e Santos-Victor [30] mostram que a variância dos termos aleatórios diminui quando se usa uma estimativa de (*u*,*v*), dada por $(\hat{u},\hat{v})$ para substituir o valor de $(\tilde{u},\tilde{v})$. Esta estimativa pode ser obtida pelo fluxo óptico determinado para as imagens prévias, ou de maneira iterativa, usando regressão robusta.

Substituindo-se o modelo afim de *u* e *v* apresentado nas equações (2.23) na equação (2.26), e considerando que o fluxo óptico e suas derivadas em relação às coordenadas da imagem são agrupados em um vetor de parâmetros definido por

$$\mathbf{T}=\begin{bmatrix}u_0 & u_x & u_y & v_0 & v_x & v_y\end{bmatrix}^T \tag{2.27}$$

a equação de observação torna-se

$$\begin{aligned}&I(x,y,t)-I(x+\hat{u},y+\hat{v},t+1)+I_x\hat{u}+I_x\hat{v}=\\&\begin{bmatrix}I_x & (x-x_0)I_x & (y-y_0)I_x & I_y & (x-x_0)I_y & (y-y_0)I_y\end{bmatrix}\mathbf{T}\end{aligned} \tag{2.28}$$

e, uma vez que uma estimativa $(\hat{u},\hat{v})$ é conhecida, Q pode ser calculado para toda a imagem ou para uma região da mesma.

Um número de observações N_{obs} maior que seis deve ser considerado, a fim de estimar o modelo afim do fluxo óptico. Mas, em geral, trabalha-se com um sistema sobre-determinado. Assim, se a equação (2.28) for escrita para cada *pixel* de observação *i* como $\mathbf{y}_i = \mathbf{x}_i$*Q, o sistema gerado para todos os N_{obs} pontos de observação pode ser escrito como $\mathbf{Y} = \mathbf{X}\mathbf{Q}$.

Supondo-se que o ruído na equação de observação seja gaussiano, pode-se obter uma estimativa de mínimos quadrados para o sistema [30], que minimiza o somatório

$$\sum_{i=1}^{N_{obs}}(\mathbf{y}_i-\mathbf{x}_i\mathbf{T})^2 \tag{2.29}$$

a qual é dada por

$$\hat{\mathbf{T}}=(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}\,. \tag{2.30}$$

Entretanto, Grossmann e Santos Victor [30] propõem que o sistema seja resolvido usando regressão robusta, porque em seus experimentos eles concluíram que o ruído neste sistema não é gaussiano [30].

Uma solução robusta é definida de forma a minimizar o somatório

$$\sum_{i=1}^{N_{obs}} ?\left(\frac{\mathbf{y}_i - \mathbf{x}_i\mathbf{T}}{s}\right) \tag{2.31}$$

para alguma função r que tenha um único mínimo em zero; ou equivalentemente, a fim de encontrar um zero de

$$\sum_{i=1}^{N_{obs}} ?\left(\frac{\mathbf{y}_i - \mathbf{x}_i\mathbf{T}}{s}\right)\mathbf{x}_i \tag{2.32}$$

onde a função Y é a derivada de primeira ordem de r, e s é o desvio padrão dos resíduos $R_i^k = \mathbf{y}_i - \mathbf{x}_i\hat{\mathbf{T}}^k$ da estimativa *k* de Q. Uma nova estimativa de Q pode ser obtida por regressão robusta como

$$\hat{\mathbf{T}}^{k+1} = \hat{\mathbf{T}}^k - \left(\mathbf{X}^T\mathbf{W}^k\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{W}^k\left(\mathbf{Y} - \mathbf{X}\hat{\mathbf{T}}^k\right) \tag{2.33}$$

onde *k* é a iteração anterior, *k*+1 é a iteração atual e **W** é uma matriz diagonal onde

$$W_{ii}^k = y\left(\frac{\mathbf{y}_i - \mathbf{x}_i\hat{\mathbf{T}}^k}{s}\right)\frac{s}{\mathbf{y}_i - \mathbf{x}_i\hat{\mathbf{T}}^k} \tag{2.34}$$

é um peso dado sucessivamente a cada observação. Uma outra solução pode ser obtida, onde Q é dado por

$$\hat{\mathbf{T}}^{k+1} = \hat{\mathbf{T}}^k - \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\left(\mathbf{R}_m\right)^k \tag{2.35}$$

onde

$$\left(R_m\right)_i^k = y\left(\frac{R_i^k}{s}\right)s \tag{2.36}$$

Usando a função $r(x) = x^2/2$ nas equações (2.33) ou (2.35) obtém-se, diretamente, a solução de mínimos quadrados para o sistema. Grossmann e Santos-Victor [30] sugerem o uso do chamado estimador Huber, onde

$$y(x) = \begin{cases} x & \text{se } |x| < 1 \\ |x|/x & \text{caso contrário} \end{cases} \tag{2.37}$$

ou uma aproximação dada por

$$y(x) = \frac{2x^2}{\left(x^2+1\right)} \tag{2.38}$$

Seja usando a equação (2.33) ou (2.35), a estimativa de mínimos quadrados é obtida na primeira iteração, e refinada nas iterações seguintes do processo de regressão robusta. No

primeiro método Q converge mais rapidamente, porém a cada iteração é necessário atualizar a matriz de pesos **W** e calcular uma inversão da matriz ($\mathbf{X}^T\mathbf{W}\mathbf{X}$) que tem dimensão 6×6. Por outro lado, no segundo método pode-se determinar Q calculando a inversa $(\mathbf{X}^T\mathbf{X})^{-1}$ apenas uma vez.

Grossmann e Santos Victor [30] propõem dois estimadores para o fluxo óptico afim, denominados estimadores anisotrópicos, os quais são baseados em um esquema de diferenciação direcional das imagens ao longo da direção da estimativa corrente do fluxo óptico. No primeiro, chamado Anisotrópico, eles atualizam as equações de observação a cada nova imagem e usam o fluxo óptico obtido para a seqüência anterior como valor inicial. No segundo, chamado Anisotrópico-M as equações de observação são atualizadas entre duas seqüências de imagens e a cada iteração do método de regressão robusta.

A implementação feita neste trabalho considerou a restrição de fluxo óptico afim para pequenas regiões da imagem, sendo que, nas equações (2.27) e (2.28), x_0 e y_0 são as coordenadas do *pixel* central da região sob consideração e u_0 e v_0 são as componentes do fluxo óptico neste *pixel*. Para cada uma destas regiões, escreveu-se um sistema sobre-determinado usando pontos de observação obtidos de maneira aleatória. O tamanho da região podia ser variado livremente, assim como o número de observações.

A Figura 2.7 apresenta o resultado obtido pelo método Anisotrópico, usando a equação (2.35) para realizar a regressão robusta sem a utilização de pesos, resolvendo as iterações para a seqüência de imagens de teste, onde as regiões utilizadas são de tamanho 20×20 *pixels* e foram tomadas 50 observações em cada uma delas.

Em relação aos resultados obtidos por este método, pode-se dizer que ele se mostra atrativo no sentido em que as derivadas do brilho devem ser calculadas apenas em alguns

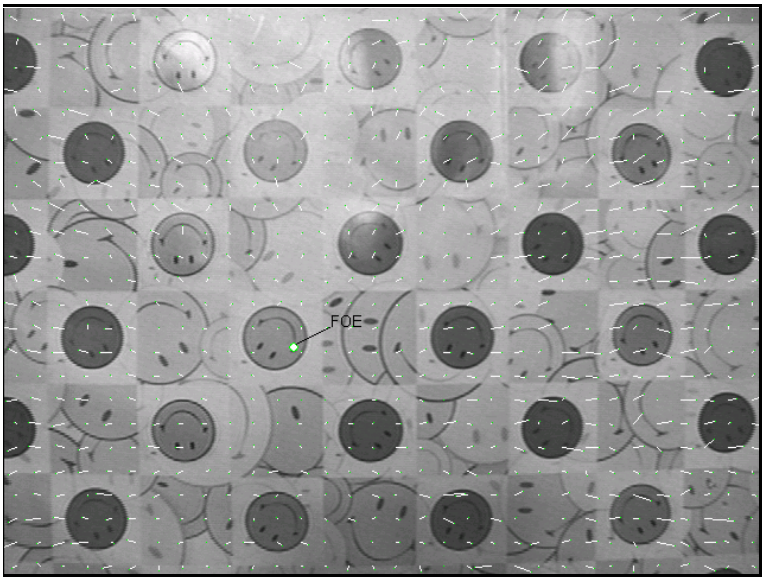


Figura 2.7 - Fluxo óptico obtido pelo método de Grossmann e Santos-Victor para a seqüência de teste ($N = 20$ e $N_{obs} = 50$).

pixels em cada região, o que reduz sensivelmente o custo computacional. Por outro lado, tem-se a necessidade de inverter uma matriz 6×6 pelo menos uma vez durante o processo de regressão robusta (no caso do algoritmo Anisotrópico-M ou no caso do algoritmo com pesos,

é necessário inverter uma matriz 6×6 a cada iteração do método de regressão robusta). Aumentando-se o tamanho da região e diminuindo-se o número de observações, o sistema se torna mais rápido, porém é necessário manter um certo compromisso entre precisão e tempo de processamento.

2.5 Algoritmos Propostos para a Aplicação em Navegação Usando Sistemas Embarcados

Dadas as restrições da tarefa tratada neste trabalho, foram propostos dois algoritmos, um para cada grupo anterior, aproveitando os aspectos positivos de alguns algoritmos da literatura para produzir resultados mais adequados.

2.5.1 Algoritmo de Suavidade Global Normalizado

O algoritmo de suavidade global proposto se baseia na utilização do procedimento de normalização das derivadas usado no algoritmo de Lai e Vemuri [35] para o algoritmo de Horn e Schunck [31], a fim de permitir que este trabalhe melhor com imagens contendo descontinuidades. Ao novo algoritmo resultante designou-se algoritmo de Horn e Schunck Normalizado [44].

Assim, neste algoritmo, o fluxo óptico em cada *pixel* pode ser obtido iterativamente como

$$\begin{aligned} u^{n+1} &= \overline{u}^n - \overline{E}_x \left(\overline{E}_x \overline{u}^n + \overline{E}_y \overline{v}^n + \overline{E}_t \right) \Big/ \left(\alpha^2 + \overline{E}_x^2 + \overline{E}_y^2 \right) \\ v^{n+1} &= \overline{v}^n - \overline{E}_y \left(\overline{E}_x \overline{u}^n + \overline{E}_y \overline{v}^n + \overline{E}_t \right) \Big/ \left(\alpha^2 + \overline{E}_x^2 + \overline{E}_y^2 \right) \end{aligned} \tag{2.39}$$

onde

$$\overline{E}_x = \frac{I_x}{\|\nabla I\|},\ \overline{E}_y = \frac{I_y}{\|\nabla I\|} \text{ e } \overline{E}_t = \frac{I_t}{\|\nabla I\|} \tag{2.40}$$

são os valores normalizados das derivadas do brilho, dividindo cada uma pela norma do gradiente espacial do brilho $\|\nabla I\|$, dada por $\sqrt{I_x^2 + I_y^2}$, no *pixel* considerado. Neste trabalho, as derivadas do brilho em cada *pixel*, I_x, I_y e I_t, são aproximadas por diferenças finitas [31] como mostrado nas equações (2.9).

Na implementação realizada, utilizou-se um valor de $\alpha = 0{,}5$, como sugerido em [31], os valores iniciais das componentes dos vetores de fluxo óptico em cada *pixel* são sempre iguais a zero ($u^0 = 0$ e $v^0 = 0$) e o número de iterações n_{iter} é igual a 10.

A Figura 2.8 mostra o resultado obtido por este algoritmo para a seqüência de imagens de teste. Verifica-se, novamente, que o fluxo óptico é predominantemente radial.

A Tabela 2.4 apresenta os custos associados à obtenção da estimativa do fluxo óptico pelo método de Horn e Schunck Normalizado. Esta tabela apresenta separadamente os custos das diversas etapas do processamento, sendo que na segunda coluna os custos são tomados como

o número de somas e produtos e na última coluna é apresentada a função O(n) correspondente. Desta tabela pode-se notar que no algoritmo de Horn e Schunck Normalizado

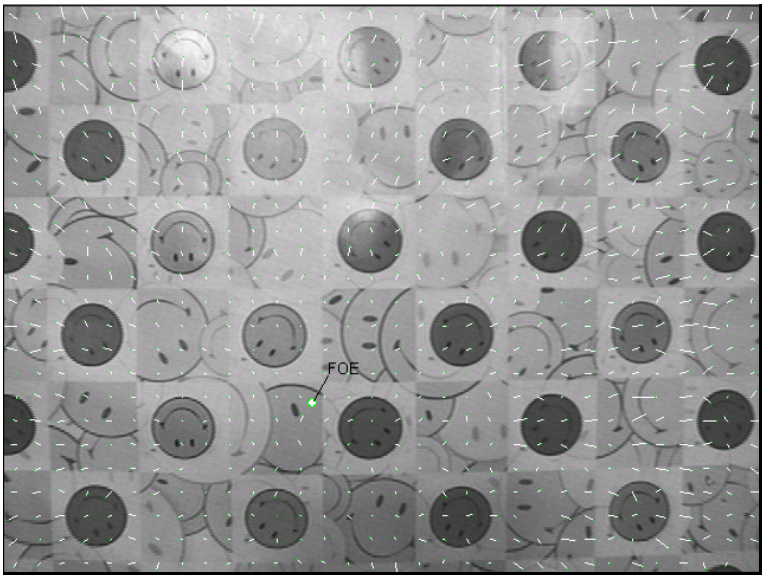


Figura 2.8 - Fluxo óptico obtido pelo método de Horn e Schunck Normalizado para a seqüência de teste.

Tabela 2.4 - Complexidade do algoritmo de Horn e Schunck Normalizado

Etapa Considerada	Custos	Complexidade
Cálculo das derivadas	21.m.n somas + 3.m.n produtos	O(m.n) somas + O(m.n) produtos
Normalização das derivadas	m.n somas + 6.m.n produtos	O(m.n) somas + O(m.n) produtos
Iterações do método	$20.m.n.n_{iter}$ somas + $12.m.n.n_{iter}$ produtos	$O(n_{iter}.m.n)$ somas + $O(n_{iter}.m.n)$ produtos
Custo total*	68.198.400 somas + 39.628.800 produtos	
m = número de colunas da imagem n = número de linhas da imagem n_{iter} = número de iterações * Custo obtido para m = 640, n = 480 e n_{iter} = 10		

o número de quadros de imagens a serem armazenados é o mesmo que no algoritmo original, bem como o número de derivadas a serem calculadas na imagem. A quantidade de cálculo é maior devido ao passo adicional de normalização das derivadas. Estas características de implementação, associadas à robustez verificada no algoritmo ao processar imagens contendo descontinuidades, fazem dele um algoritmo interessante para a tarefa em estudo [44].

2.5.2 Algoritmo de Fluxo Óptico Constante com Solução Modificada de Mínimos Quadrados

Considerando os métodos propostos por Lucas e Kanade [5] [36] e por Grossmann e Santos-Victor [30], descritos anteriormente, foi proposto neste trabalho um outro algoritmo, no qual se considera a aproximação de fluxo óptico constante em uma região de N×N *pixels*, como no primeiro, mas onde a solução de mínimos quadrados é obtida tomando-se um

número de observações menor do que o número total de *pixels* na região, como é feito no segundo método. Isto faz com que as derivadas do brilho da imagem sejam calculadas apenas em alguns *pixels*, reduzindo sensivelmente o tempo de cálculo da estimativa dos vetores de fluxo óptico. Por outro lado, uma redução do número de *pixels* de observação torna os resultados do método de mínimos quadrados menos precisos, porque nem todos os *pixels* estão contribuindo para a estimativa. Assim, é necessário manter um compromisso entre o número de *pixels* utilizado para obter a estimativa e a precisão do resultado.

A Tabela 2.5 apresenta os custos de obtenção da estimativa do fluxo óptico pelo método de Mínimos Quadrados Modificado. Na primeira coluna da tabela são discriminadas as fases do processamento que estão sendo consideradas, na segunda coluna é apresentado o número de operações de somas e produtos realizados em cada etapa e na última coluna, aparece a função O(n) correspondente. Pode-se ver da tabela que o tempo de processamento diminui com o aumento de N e com a redução do número de pontos de observação N_{obs}, confirmando o que foi dito anteriormente.

Tabela 2.5 - Complexidade do algoritmo de Mínimos Quadrados Modificado

Etapa considerada	Custos	Complexidade
Cálculo das derivadas	$21.m.n.N_{obs}/N^2$ somas $+ 3.m.n.N_{obs}/N^2$ produtos	$O(m.n.N_{obs}/N^2)$ somas $+ O(m.n.N_{obs}/N^2)$ produtos
Obtenção da matriz $\mathbf{A}^T\mathbf{A}$ e do vetor $\mathbf{A}^T\mathbf{b}$	$6.m.n.N_{obs}/N^2$ somas $+ 6.m.n.N_{obs}/N^2$ produtos	$O(m.n.N_{obs}/N^2)$ somas $+ O(m.n.N_{obs}/N^2)$ produtos
Obtenção de u e v	$3.m.n/N^2$ somas $+ 10.m.n/N^2$ produtos	$O(m.n/N^2)$ Somas $+ O(m.n/N^2)$ produtos
Custo total*	1.039.104 somas + 353.280 produtos	

m = número de colunas da imagem
n = número de linhas da imagem
N = número de linhas e colunas de cada região
N_{obs} = número de pontos de observação tomados em cada região
* Custo obtido para m = 640, n = 480, N = 20 e N_{obs} = 50

Na implementação realizada neste trabalho, o valor de N_{obs} podia ser variado livremente. Porém, pensando em uma redução substancial do custo computacional, os resultados apresentados foram obtidos tomando-se aleatoriamente 50 *pixels* de observação em cada região de 20×20 *pixels*. As Figuras 2.9 e 2.10 mostram os resultados obtidos pelos algoritmos de Mínimos Quadrados e de Mínimos Quadrados Modificado, respectivamente, para a seqüência de imagens de teste obtida pela câmara no robô, considerando em ambos regiões de 20×20 *pixels*.

Nestas imagens, pode-se verificar que os resultados obtidos no cálculo dos vetores de fluxo óptico pelo algoritmo de Mínimos Quadrados Modificado são semelhantes aos resultados obtidos para o algoritmo original. Por outro lado, o tempo necessário para a obtenção de uma estimativa por este método é sensivelmente menor, como se verá a seguir.

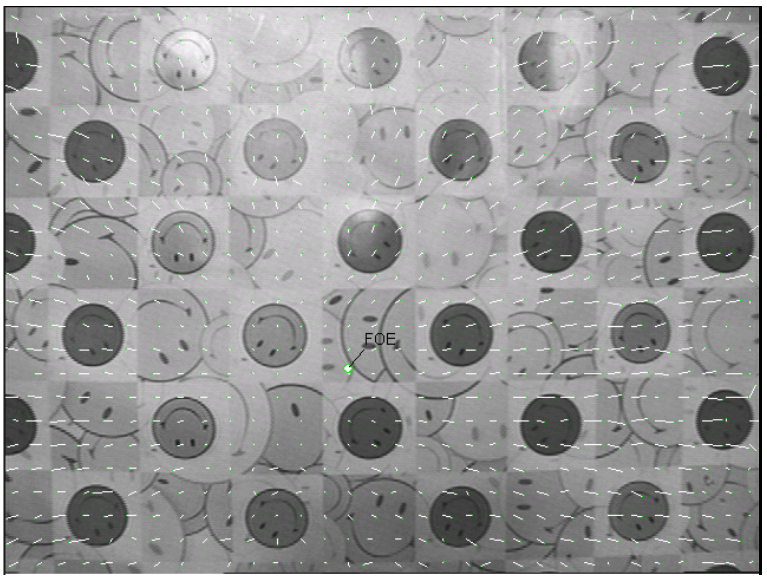


Figura 2.9- Fluxo óptico obtido pelo método de Mínimos Quadrados para a seqüência de teste (N=20).

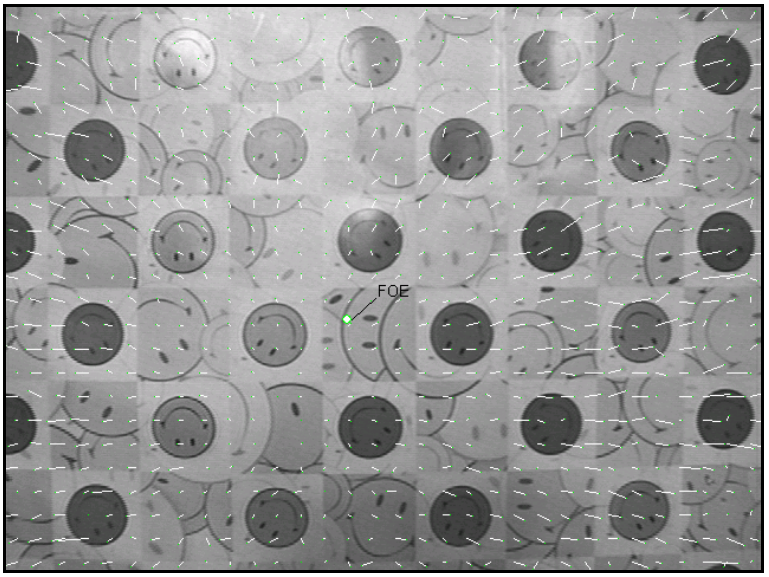


Figura 2.10- Fluxo óptico obtido pelo método de Mínimos Quadrados Modificado para a seqüência de teste (N = 20 e N_{obs} = 50).

2.6 Análise de Desempenho dos Algoritmos Implementados

Para comparar os métodos acima descritos, foram realizados dois experimentos usando o robô PIONEER 2DX. As seqüências de imagens adquiridas, usando a câmara de vídeo que fica em cima do robô, foram armazenadas, para serem posteriormente utilizadas com cada um dos algoritmos. Estes experimentos foram selecionados, basicamente, porque, em ambos, é

possível prever o resultado do cálculo do fluxo óptico. Assim, é possível utilizar uma medida comparativa do desempenho dos algoritmos implementados.

No primeiro experimento, o robô se move a 30 mm/s paralelamente a uma parede, com a câmara voltada para ela, adquirindo uma seqüência de imagens. Neste caso, considerando que o robô esteja realmente se movendo paralelo à parede, o fluxo óptico esperado apresenta valores diferentes de zero apenas na direção horizontal. A Figura 2.16 mostra uma imagem do robô durante um destes experimentos, onde a parede está à direita do robô.

Figura 2.11 - Primeiro experimento: movimento paralelo à parede.

A Tabela 2.6 apresenta os resultados de um destes experimentos, onde a parede está à esquerda do robô. Assim, a direção dos vetores de fluxo óptico deve ser de 180^o, e o seu módulo deve ser constante. Estes resultados foram obtidos para os métodos sob análise, onde os algoritmos de Mínimos Quadrados, de Grossmann e Santos-Victor e de Mínimos Quadrados Modificado trabalham com uma região de 20×20 *pixels*. O algoritmo de Paolo Nesi et al. considera uma região de 5×5 *pixels*. Os algoritmos de Grossmann e Santos-Victor e o de Mínimos Quadrados Modificado consideram 50 observações, tomadas aleatoriamente em cada região. Os valores representados na tabela são o valor médio $\bar{q}$ do ângulo do fluxo óptico (em graus), o erro médio e_q associado a ele (também em graus), o valor médio

Tabela 2.6 - Resultados numéricos para o primeiro experimento.

Algoritmo	$\bar{q}$	e_q	$\bar{f}$	σ_f
Horn e Schunck	178,62	21,88	0,72	0,38
Horn e Schunck Normalizado	176,82	22,96	0,74	0,38
Lai e Vemuri	183,44	22,92	0,76	0,39
Mínimos Quadrados	178,56	6,15	1,30	0,48
Paolo Nesi et al.	178,65	18,20	0,90	0,48
Grossmann e Santos-Victor	179,99	12,99	1,03	0,54
Mínimos Quadrados Modificado	182,54	10,42	1,26	0,52

$\bar{f} = \sqrt{u^2 + v^2}$ dos vetores de fluxo óptico (em *pixels*/segundo) e o desvio padrão s_f associado a ele (também em *pixels*/segundo). O erro médio do ângulo do fluxo óptico foi calculado tomando-se a diferença entre o ângulo estimado e o ângulo esperado (180^o), e fazendo-se a média dos valores obtidos para toda a imagem.

Como pode ser visto a partir da Tabela 2.6, os valores do ângulo médio dos vetores de fluxo óptico estão próximos ao valor esperado, para todos os algoritmos. O valor médio dos erros é menor para o algoritmo de Mínimos Quadrados, seguido pelo algoritmo de Mínimos Quadrados Modificado. O desvio padrão associado ao módulo do vetor de fluxo óptico é menor no algoritmo de Horn e Schunck, seguido pelo algoritmo de Horn e Schunck Normalizado. Resultados semelhantes aos apresentados na Tabela 2.6 foram obtidos em outros experimentos realizados, onde foram utilizadas outras texturas, o robô foi posicionado a diferentes distâncias da parede e comandado a desenvolver diferentes valores de velocidade.

No segundo experimento, a câmara de vídeo adquire uma seqüência de imagens enquanto o robô se move com velocidade constante, de 60 mm/s, aproximando-se da parede. A Figura 2.12 mostra uma imagem do robô durante este experimento.

Neste experimento, o robô é comandado a mover-se com velocidade linear constante, executando assim um movimento de translação. Em tal situação é esperado que os vetores de fluxo óptico tenham direção radial, sendo o Foco de Expansão (*Focus of Expansion* - FOE) o ponto de onde estes vetores divergem. O foco de expansão é caracterizado por apresentar as componentes do fluxo óptico iguais a zero ($u = 0$ e $v = 0$). Porém, sua determinação é feita, em geral, a partir dos vetores de fluxo óptico calculados, buscando-se o ponto onde as suas direções se cruzam [20]. Pode-se obter informação de profundidade na cena usando-se o FOE e o fluxo óptico para estimar o tempo para contato t, o qual pode ser dado por [20]

Figura 2.12 - Segundo experimento: movimento perpendicular à parede.

$$t = \frac{(x - x_{FOE})}{u} = \frac{(y - y_{FOE})}{v} = \frac{\sqrt{(x - x_{FOE})^2 + (y - y_{FOE})^2}}{\sqrt{u^2 + v^2}} \tag{2.41}$$

onde x e y são as coordenadas do *pixel* sob consideração, x_{FOE} e y_{FOE} são as coordenadas do *FOE*, e u e v são os valores das componentes do vetor de fluxo óptico neste *pixel*.

A utilização do tempo para contato como uma medida da profundidade relativa dos objetos na cena e a câmara de vídeo, parte do pressuposto que o campo de fluxo óptico é semelhante ao campo de movimento, e que pode, portanto, dar informações sobre a estrutura do ambiente.

Neste trabalho, usando-se o fluxo óptico gerado por cada um dos algoritmos sob análise, determinou-se o foco de expansão (*FOE*) considerando-se o valor médio do fluxo óptico em regiões quadradas de 20×20 *pixels*.

O procedimento implementado para determinar a coluna do *FOE* consiste em verificar, linha por linha de regiões, em que coluna da imagem o vetor de fluxo óptico passa de um ângulo maior a um menor que 90^o, ou vice-versa, ou de um ângulo maior a um menor que 270^o, ou vice-versa. O valor de x_{FOE} pode ser, então, obtido como a média dos valores de coluna encontrados para cada linha. De maneira similar, o valor de y_{FOE} pode ser estimado verificando-se, coluna por coluna de regiões, em que linha da imagem o vetor de fluxo óptico passa de um ângulo maior a um menor que 0^o, ou vice-versa, ou de um ângulo maior a um menor que 180^o, ou vice-versa. Finalmente, o valor de y_{FOE} pode ser calculado como a média dos valores de linha obtidos para cada coluna.

Tendo determinado as coordenadas do FOE, calculou-se o tempo para contato usando a terceira das igualdades na equação (2.41), para cada uma das regiões de 20×20 *pixels* da imagem. Tomou-se, então, o valor mediano de todos estes valores para representar o tempo para contato na imagem. O fluxo óptico foi calculado quando o robô estava a 30 cm da parede, e assim o tempo para contato esperado é de 5 segundos. Os valores de tempo para contato, para cada algoritmo, estão representados na Tabela 2.7.

Neste caso, as duas implementações do algoritmo de Mínimos Quadrados apresentam os resultados que mais se aproximam do esperado. Estes algoritmos, em comparação com os demais, apresentam ainda uma outra diferença: eles subestimam o tempo para contato, o que é

Tabela 2.7 - Resultados numéricos para o segundo experimento.

Algoritmo	τ (s)
Horn e Schunck	7,783820
Horn e Schunck Normalizado	7,540172
Lai e Vemuri	8,773588
Mínimos Quadrados	4,933316
Paolo Nesi et al.	5,506741
Grossmann e Santos-Victor	5,921020
Mínimos Quadrados Modificado	4,897517

muito melhor para garantir a segurança do robô, dado que o mesmo vai iniciar qualquer manobra de evasão com antecedência, e não com atraso.

A fim de concluir a análise dos algoritmos considerados, a Tabela 2.8 apresenta o valor relativo do tempo gasto para executá-los, tomando o algoritmo de Horn e Schunck como referência. Para se ter uma idéia da possibilidade de se utilizar um destes algoritmos para uma aplicação em tempo real, o algoritmo mais rápido executa o cálculo em cerca de 130 ms, sendo este tempo medido usando a função *timeGetTime* do Visual C++ funcionando no Windows NT, e considerando que, neste momento, estava também ativo um programa antivírus e os demais processos normais do sistema. Há que se mencionar que o computador utilizado foi um Pentium II 400 MHz, com 128 MBytes de memória RAM.

Tabela 2.8 – Tempos de execução.

Algoritmo	Tempo (%)
Horn e Schunck	100
Horn e Schunck Normalizado	110
Lai e Vemuri	446
Mínimos Quadrados	32
Paolo Nesi et al.	429
Grossmann e Santos-Victor	35
Mínimos Quadrados Modificado	2

A Tabela 2.9 apresenta um resumo dos estudos de complexidade realizados para alguns dos algoritmos implementados. Nas Tabelas 2.1 a 2.5 estes resultados foram apresentados de maneira mais detalhada. Na Tabela 2.9 são apresentados os resultados numéricos para as operações de soma e produto em cada imagem, considerando imagens de 640×480 *pixels*, 10 iterações nos algoritmos de iterativos, regiões de 20×20 *pixels* nos métodos locais e 50 pontos

Tabela 2.9 - Complexidade dos algoritmos em termos do número de somas e produtos.

Algoritmo	Somas	Produtos
Horn e Schunck	67.891.200	37.785.600
Lai e Vemuri	137.318.400	109.977.600
Horn e Schunck Normalizado	68.198.400	39.628.800
Mínimos Quadrados	8.296.704	2.785.480
Mínimos Quadrados Modificado	1.039.104	353.280
* Para m = 640, n = 480, N = 20, n_{iter} =10 N_{obs} = 50.		

de observação no algoritmo de Mínimos Quadrados Modificado. Esta tabela confirma os resultados de tempo da Tabela 2.8 obtidos para estes mesmos algoritmos.

Pelos resultados apresentados nos experimentos, assim como a partir das Tabelas 2.8 e 2.9, conclui-se que o algoritmo mais adequado para o cálculo do fluxo óptico a bordo de um robô móvel de pequeno porte é, sem dúvida, o algoritmo de Mínimos Quadrados Modificado. Ele explora o fato de não precisar calcular as derivadas do brilho em todos os *pixels* da imagem, o que o torna bastante rápido, e a simplicidade do método, que envolve apenas a necessidade de inversão de uma matriz 2×2 na determinação da estimativa, sem a necessidade de qualquer processo iterativo.

Por outro lado, como foi dito anteriormente, o fluxo óptico obtido por este método para os dois experimentos realizados é pior do que aquele obtido através de outros algoritmos testados. Porém, ainda assim, os resultados são satisfatórios para a tarefa de navegação de robôs móveis, onde não é necessária grande precisão. Finalmente, é importante mencionar que os experimentos apresentados foram repetidos dezenas de vezes, com o robô a diferentes distâncias da parede e a diferentes velocidades. Na maioria dos casos os resultados são similares aos apresentados, ou seja, os resultados apresentados são representativos.

2.7 Medida de Confiabilidade do Fluxo Óptico

Tendo como base o algoritmo de Mínimos Quadrados Modificado, escolhido para o cálculo do fluxo óptico neste trabalho, e buscando diminuir os efeitos de estimativas com grande quantidade de erro, estudou-se uma forma de medir a confiabilidade das estimativas em cada região.

Barron et al. [5] sugerem que os autovalores da matriz $\mathbf{A}^T\mathbf{A}$ ($l_1 \geq l_2$), cujos valores dependem da magnitude do gradiente espacial da imagem, sejam usados como medida de confiabilidade. Assim, se l_1 e l_2 forem maiores que um determinado limiar, o fluxo óptico é calculado como a estimativa de mínimos quadrados dada pelas equações (2.15) ou (2.20); se apenas l_1 for maior que o limiar, então é calculado o fluxo óptico normal f_n, ou seja, o fluxo óptico na direção do gradiente espacial que é dado pela equação (2.4), e, finalmente, se l_1 e l_2 forem menores que o limiar, o fluxo óptico não é calculado.

Dev et al. [20] sugerem uma medida de confiabilidade baseada em uma análise das principais fontes de erro da estimativa do fluxo óptico, que são a sensibilidade do modelo a ruídos, o problema de abertura e o fato da restrição do fluxo óptico não ser sempre verificada. De acordo com eles, uma estimativa da variância máxima, no cálculo do fluxo óptico pelo método de mínimos quadrados, pode ser dada por

$$s^2_{max} = \frac{\mathbf{e}^T\mathbf{e}}{M l_{min}\left(\mathbf{A}^T\mathbf{A}\right)} \tag{2.42}$$

onde

$$e_i = I_{x_i}\hat{u} + I_{y_i}\hat{v} + I_{t_i} \tag{2.43}$$

é o erro resultante da aproximação do fluxo óptico em cada *pixel i* pelos valores $\hat{u}$ e $\hat{v}$, M é igual ao número de *pixels* usados no cálculo do fluxo óptico, ou seja, igual ao número de observações e l_{min} é o menor autovalor da matriz $\mathbf{A}^T\mathbf{A}$. Esta medida pode ser usada para descartar estimativas muito grosseiras do fluxo óptico, minimizando os efeitos dos vetores de fluxo óptico com pouca confiabilidade em futuras etapas de processamento [22].

Neste trabalho foi implementada a medida sugerida em [21], onde o valor da variância máxima foi calculado para cada região de 20×20 *pixels* da imagem e as regiões com variância maior que um determinado limiar foram excluídas de qualquer processamento posterior. O limiar para valor máximo da variância foi escolhido considerando o compromisso entre a confiabilidade desejada dos vetores de fluxo óptico e a esparsidade causada pela exclusão dos vetores não confiáveis.

A Figura 2.13 mostra o resultado desta operação aplicada ao fluxo óptico obtido pelo método de Mínimos Quadrados Modificado, para a primeira seqüência de imagens teste. Apenas os vetores de fluxo óptico que atendem ao critério de confiabilidade, ou seja, apenas os vetores com variância menor que o limiar, são desenhados. Pode-se ver nesta figura que muitos vetores de fluxo óptico são descartados deixando a matriz bastante esparsa.

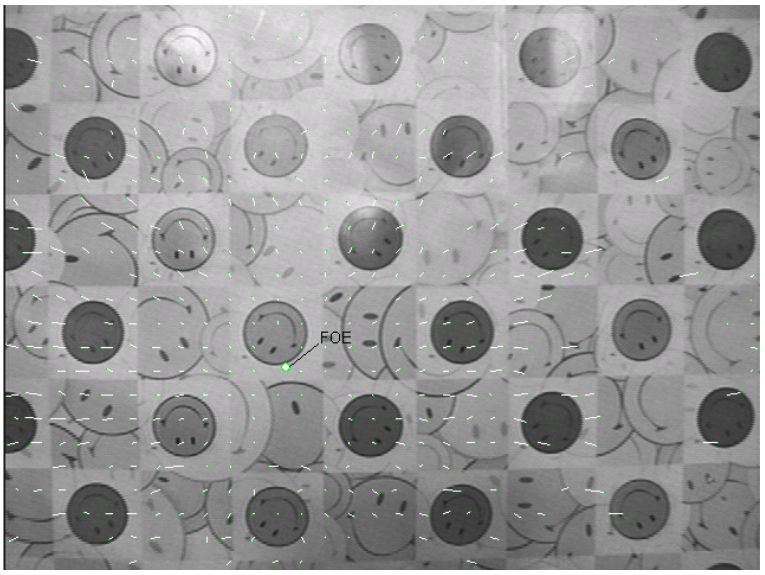


Figura 2.13 – Vetores de fluxo óptico obtidos pelo método de Mínimos Quadrados Modificado que satisfizeram ao critério de confiabilidade implementado.

2.8 Considerações Finais

Tendo em mente que de os recursos computacionais disponíveis a bordo de um robô móvel de pequeno porte são escassos, a seleção dos algoritmos para possível implementação deve considerar a sua simplicidade e restringir o número de imagens ao mínimo, ou seja, duas em cada seqüência. Assim, foi realizado um estudo comparativo de diversos algoritmos de cálculo do fluxo óptico disponíveis na literatura, para a possível implementação a bordo de um robô móvel. As implementações realizadas possibilitaram testar todos os métodos com as mesmas imagens, assim como um mesmo método com as mesmas imagens, mas variando uma ou mais características, como número de iterações ou tamanho da região. Desta maneira foi possível verificar quais as melhores condições de utilização de cada algoritmo, a fim de obter uma estimativa do fluxo óptico para a aplicação desejada.

Todos os algoritmos implementados são classificados como métodos diferenciais, ou seja, utilizam a derivada do brilho da imagem para obter a estimativa do fluxo óptico. Dentre estes existem os algoritmos iterativos: de Horn e Schunck, que utiliza uma restrição de suavidade global, e de Lai e Vemuri, que combina restrição de suavidade global e seguimento de contornos, foram implementados inicialmente.

Foi desenvolvido neste trabalho um novo algoritmo iterativo onde as derivadas são normalizadas, que apresentou uma melhoria expressiva em relação ao algoritmo original, em se tratando de imagens com descontinuidade, e que foi chamado algoritmo de Horn e Schunck Normalizado.

Como estes três métodos são iterativos, o tempo para a obtenção de uma estimativa aumenta sensivelmente com o número de iterações, prejudicando seu uso na aplicação devido à restrição temporal.

Assim, passou-se à análise de métodos não iterativos, com a implementação de algoritmos diferenciais que adicionam uma restrição de suavidade local ao fluxo óptico. No primeiro, proposto por Lucas e Kanade, considera-se que o fluxo óptico é constante na região e obtém-se uma solução de mínimos quadrados. No segundo, proposto por Paolo Nesi e co-autores, o fluxo óptico é novamente considerado constante na região e é obtida uma solução de máxima verossimilhança, por um processo de votação. No terceiro algoritmo, proposto por Grossmann e Santos-Victor, considera-se que o fluxo óptico é linear dentro da região, seguindo um modelo afim. Obtém-se uma estimativa inicial por mínimos quadrados, que é posteriormente refinada em um processo de regressão robusta.

No desenvolvimento, chegou-se a um novo algoritmo neste trabalho, chamado algoritmo de Mínimos Quadrados Modificado, que consiste em uma modificação do algoritmo de Lucas e Kanade, onde apenas alguns *pixels* de observação (mais que dois e menos que o número total de *pixels* na região) são utilizados na obtenção da estimativa do fluxo óptico em cada região da imagem pelo método de mínimos quadrados. Por este método, uma estimativa do vetor de fluxo óptico na região pode ser obtida por um processo simples e determinístico, que envolve apenas a inversão de uma matriz 2×2, e onde as derivadas do brilho não precisam ser calculadas para todos os *pixels* da imagem, mas apenas para os pontos de observação.

Como resultado da comparação dos algoritmos, em termos dos resultados obtidos nos experimentos e do tempo de obtenção da estimativa do fluxo óptico, o algoritmo desenvolvido neste trabalho, chamado algoritmo de Mínimos Quadrados Modificado, foi selecionado para executar o cálculo do fluxo óptico a bordo do robô, pois ele atende às restrições permitindo obter as estimativas dos vetores de fluxo óptico rapidamente, com precisão similar aos outros.

Capítulo 3 - Detecção de Objetos a Partir do Fluxo Óptico

Com o objetivo de facilitar a obtenção de informações sobre o ambiente, para posterior utilização no controle do robô móvel, implementou-se um processo de segmentação da imagem, com base no fluxo óptico, que resulta na determinação dos objetos presentes na cena. A seguir, são apresentados alguns conceitos relacionados a tal segmentação, assim como o algoritmo proposto neste trabalho.

3.1 Segmentação da Imagem Baseada no Fluxo Óptico

A segmentação da imagem com base no fluxo óptico é um processo diretamente relacionado à segmentação de movimentos, que consiste em agrupar os *pixels* da imagem que participem do mesmo movimento. Assim, cada grupo de *pixels* obtido no processo de segmentação está associado a cada um dos diferentes objetos movendo-se na cena 3D captada pela câmara de vídeo [10] [48]. Portanto, a segmentação baseada no fluxo óptico corresponde a agrupar os *pixels* cujos vetores de fluxo óptico sejam semelhantes, ou seja, aqueles que correspondem ao mesmo movimento dentro da imagem. Assim, considerando-se que o campo de fluxo óptico seja semelhante ao campo de movimento, cada grupo de *pixels* obtido no processo está associado a uma mesma estrutura no espaço tridimensional. Portanto, a precisão dos resultados da segmentação depende da precisão do fluxo óptico estimado [48].

Em geral, cada vetor de fluxo óptico corresponde à projeção do movimento 3D de um único objeto na cena. Assim, cada movimento distinto pode ser precisamente descrito por uma série de parâmetros associados a um modelo do movimento. Desta maneira, para realizar a segmentação do campo de fluxo óptico é necessário associar a ele um modelo e calcular os parâmetros deste modelo, o que é feito usando os vetores de fluxo óptico estimados dentro de determinada região. A segmentação do fluxo óptico pode, então, ser feita agrupando-se os vetores de fluxo óptico com parâmetros semelhantes aos do modelo.

Os métodos existentes para realização da segmentação baseada no fluxo óptico podem ser classificados como pertencendo a um dos seguintes grupos [10]:

- os que usam *affine clustering* [53], onde os agrupamentos são feitos usando um modelo afim para o fluxo óptico;
- os métodos baseados no movimento dominante [8], onde, em cada iteração, a região de movimento dominante é identificada e o movimento dentro da região é estimado, sendo eliminado da iteração seguinte;

- e, por fim, uma aproximação na qual se executa, simultaneamente, estimação e segmentação de movimentos.

O método proposto por Wang e Adelson [53] usa a segmentação de movimentos baseada em *affine clustering*. O método supõe, basicamente, que o fluxo óptico na imagem pode ser descrito como uma série de regiões planares no espaço de velocidade, ou seja, o fluxo óptico segue o modelo afim nas regiões. Por este método, o fluxo óptico é estimado para toda a imagem, que é, então, dividida em regiões retangulares, onde os parâmetros do modelo afim são estimados. Um teste de confiabilidade é usado para medir se os vetores de fluxo óptico dentro do bloco são bem representados pelo modelo. Como os modelos de regiões que pertencem a um mesmo objeto têm parâmetros similares, os modelos que passarem no teste de confiabilidade são agrupados em um pequeno número de classes usando um algoritmo *k-means* adaptativo. Um modelo afim é derivado para cada grupo de regiões que apresentem modelos similares e, finalmente, cada vetor de fluxo óptico é agrupado em uma das classes resultantes.

O método de segmentação afim multi-estágios proposto por Borshukov et al. [10] combina o método de segmentação baseado no movimento dominante, proposto por Bergen et al. [8], com o processo de agrupamento usando o modelo afim, proposto por Wang e Adelson [53]. Neste método, a imagem é dividida em blocos retangulares, para os quais os parâmetros do modelo afim do fluxo óptico são estimados. O vetor de parâmetros associado com o movimento dominante é, então, determinado, combinando-se os parâmetros de diferentes regiões. A seguir, as regiões cujos parâmetros podem ser bem representadas por este modelo são identificadas, e reunidas em um mesmo grupo. O processo é repetido para as regiões que ainda não pertencem a nenhum grupo, gerando outro vetor de movimento dominante, e assim por diante.

3.2 O Método de Segmentação Proposto

O procedimento proposto neste trabalho utiliza um modelo constante do fluxo óptico em cada região de N×N *pixels* da imagem, e agrupa aquelas regiões contíguas que tenham vetores de fluxo óptico semelhantes, de acordo com a medida utilizada em [10] e [53]. Esta medida é dada pela distância

$$\left[\left(u_{kl} - u_{ij}\right)^2 + \left(v_{kl} - v_{ij}\right)^2\right] < T_{ij} \tag{3.1}$$

onde i e j indicam, respectivamente, a linha e a coluna da região r_{ij} sob consideração, k e l indicam, respectivamente, a linha e a coluna de uma das regiões vizinhas, u e v são as componentes do fluxo óptico na direção x e y, respectivamente, e T_{ij} é o valor de limiar para a região r_{ij}. A equação (3.1) representa o quadrado do módulo do vetor diferença entre os vetores de fluxo óptico das duas regiões, quando representados em um mesmo referencial, como mostrado na Figura 3.1.

A escolha do valor de limiar T_{ij} é importante, porque um valor muito pequeno pode fazer com que regiões pertencentes a um mesmo objeto não sejam agrupadas pelo processo de segmentação. Por outro lado, um valor muito grande de limiar pode fazer com que regiões pertencentes a objetos diferentes sejam agrupadas. Além disto, como o robô está na maior parte do tempo se aproximando dos objetos na cena, espera-se que o fluxo óptico seja radial e

que seu módulo diminua à medida que a região se aproxima do foco de expansão. Desta maneira, não é possível utilizar um mesmo valor de limiar para todas as regiões. Assim, foram realizados experimentos com diversos valores para o limiar, onde se verificou que o valor de T_{ij} da ordem de 20% do quadrado da norma do vetor de fluxo óptico na região r_{ij} apresenta bons resultados.

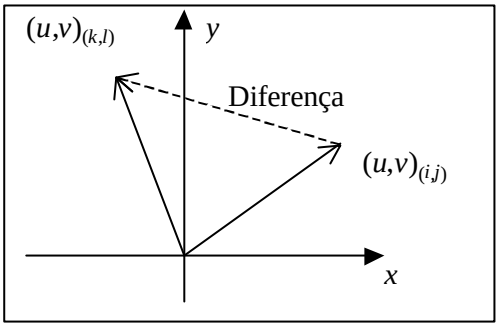


Figura 3.1 - Referencial utilizado para as componentes do fluxo óptico.

3.3 O Algoritmo de Segmentação Utilizado

Tendo o fluxo óptico sido calculado pelo método de Mínimos Quadrados Modificado para toda a imagem, tem-se um vetor (u,v) que representa cada região N×N da imagem, ou seja, que é constante para todos os *pixels* dentro da região considerada. Então, um procedimento de varredura por linhas é iniciado, de região em região, com o intuito de agrupar regiões vizinhas com fluxo óptico semelhante. No algoritmo implementado, são consideradas vizinhas as oito regiões mostradas na Figura 3.2.

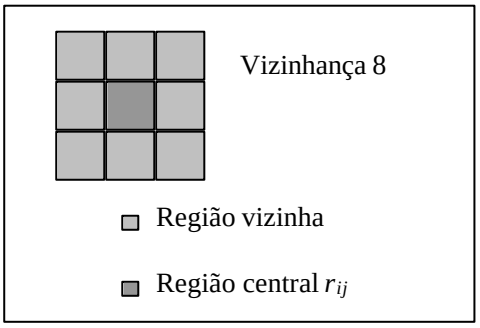


Figura 3.2 – Região e vizinhança usando conectividade-oito.

Cada região da imagem é identificada por r_{ij}, onde i é a linha e j é a coluna correspondentes à região, e cujo vetor de fluxo óptico é dado por $(u,v)_{ij}$. O marcador de cada região é identificado como c_{ij}, sendo que o valor -1 indica um valor inválido e os valores válidos são números inteiros maiores que zero. O algoritmo de segmentação é apresentado na Figura 3.3.

As regiões para as quais não é possível determinar o vetor de fluxo óptico, porque a matriz gerada no método de mínimos quadrados apresenta determinante igual a zero, são eliminadas do processo de segmentação permanecendo com marcador -1. Como resultado do processo, cada marcador representa um objeto diferente na imagem. Todo o procedimento é realizado

em um único processo de varredura, sendo, portanto, bastante rápido. Além disto, deve-se ressaltar que a análise não é feita *pixel* a *pixel*, mas região por região.

A Figura 3.4 mostra o resultado do processo de segmentação para uma seqüência de

Algoritmo de Segmentação

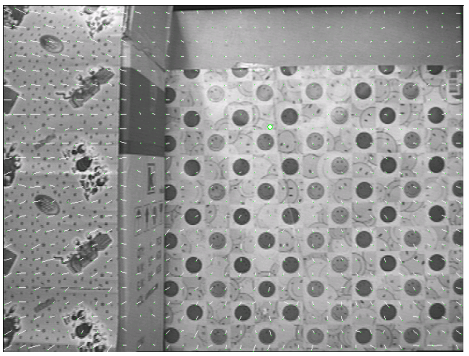
Iniciar o contador de objetos n com zero (n = 0).
Iniciar o marcador c_{ij} de cada região r_{ij} com –1.
Iniciar nl com o número de linhas de regiões.
Iniciar nc com o número de colunas de regiões.
Para i de 1 a nl(
  Para j de 1 a nc
    a)Marcar a região r_{ij} como pertencente a um objeto:
      Se r_{ij} é a primeira região (n = 0) então
        Fazer o contador de objetos n igual a 1.
        Fazer o marcador da região c_{ij} igual a n.
        Ir ao item b.
      Senão se r_{ij} já tem um marcador válido, ou seja, c_{ij} é diferente de –1 então
        Ir ao item b.
      Senão
        Buscar na vizinhança de r_{ij} por alguma região com marcador válido, cujo vetor de fluxo óptico seja semelhante ao seu, usando a medida dada na equação (3.1):
          Se não existir nenhuma região semelhante a r_{ij} com marcador válido então
            Atualizar o número de objetos (n = n + 1).
            Fazer c_{ij} igual a n.
            Ir ao item b.
          Senão
            Fazer c_{ij} igual ao menor marcador encontrado na vizinhança.
            Ir ao item b.
          Fim;
    b)Copiar o marcador c_{ij} para as regiões vizinhas de r_{ij} com fluxo óptico semelhante, que ainda não pertencerem a nenhum objeto (ou seja, com marcador -1) ou com marcador maior que c_{ij}.
  Fim para;
Fim para.

Figura 3.3- O algoritmo de segmentação baseado no fluxo óptico.

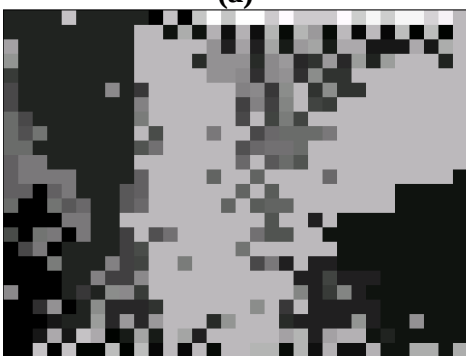
imagens. Em (a) é apresentado um quadro da seqüência de imagens considerada, com os vetores de fluxo óptico sobrepostos. Em (b) é mostrado o resultado do processo de segmentação baseada no fluxo óptico, descrito anteriormente nesta seção, sendo que as regiões vizinhas com mesmo tom de cinza representam um mesmo objeto. Cada marcador obtido pelo processo de segmentação, descrito anteriormente, foi convertido em um valor entre 0 e 255. Assim, podem existir dois objetos com o mesmo tom de cinza, porém a relação de vizinhança também deve ser considerada. Quando a medida de confiabilidade, descrita no Capítulo 2, é aplicada ao fluxo óptico, são consideradas apenas as regiões que atendem ao critério, sendo que as demais permanecem com marcador -1. Em (c) é apresentado o resultado do processo de segmentação para este caso. Novamente, cada grupo de regiões vizinhas com

mesmo tom de cinza representa um objeto, com exceção para as regiões mostradas em branco, que são as regiões onde o fluxo óptico estimado pelo método de Mínimos Quadrados Modificado não atende à medida de confiabilidade considerada.

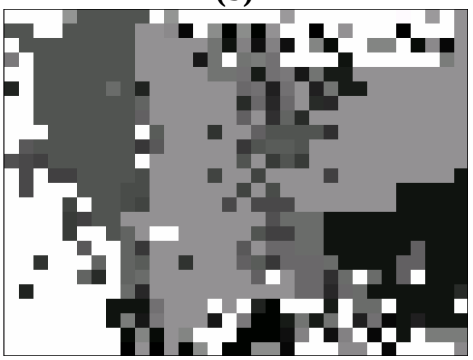
Como pode ser visto das Figuras 3.4 (b) e (c), um objeto no mundo real corresponde a mais de um objeto no resultado da segmentação, ou seja, algumas partes do objeto foram marcadas separadamente. (Este problema no processo de segmentação é conhecido como *oversegmentation* [10]). Porém, neste trabalho, cada grupo de regiões resultante do processo de segmentação, ou seja, cada grupo de regiões que possuem o mesmo marcador, será chamado objeto. Quando for feita alguma referência a um objeto no mundo real isto será colocado claramente no texto.



(a)



(b)



(c)

Figura 3.4 – (a) Primeira imagem da seqüência; (b) resultado da segmentação; (c) resultado da segmentação utilizando medida de confiabilidade.

Duas estratégias podem ser adotadas para reduzir o número de objetos no resultado da segmentação [10]. A primeira consiste em aumentar o valor de limiar considerado para determinar se duas regiões são semelhantes (equação (3.1)). Porém, aumentando-se muito este valor, pode-se causar uma perda de sensibilidade no algoritmo, que passa a formar um único

grupo com regiões representando objetos distintos da cena, o que é conhecido com *undersegmentation*. A segunda estratégia corresponde a introduzir uma nova etapa no processo de segmentação, para combinar estes objetos, o que implica encontrar um modelo que represente o objeto e uma nova medida de semelhança entre os objetos. Esta alternativa pode melhorar os resultados, mas causa um aumento indesejável no tempo de processamento. Por outro lado, como cada objeto no resultado da segmentação corresponde a uma parte de um mesmo objeto no mundo real, neste trabalho os objetos são utilizados da maneira como são obtidos no processo de segmentação proposto.

3.4 Diagrama de Tempos para Contato

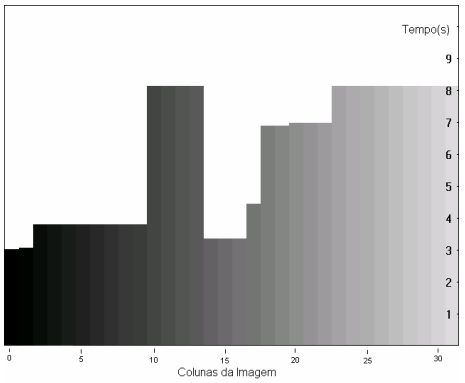
Para que o robô possa navegar utilizando apenas o sistema de sensoriamento visual, é necessário determinar, a partir das imagens, os objetos presentes na cena, o que é feito pelo processo de segmentação baseada no fluxo óptico, e determinar a distância dos objetos ao robô, o que pode ser feito a partir do cálculo do tempo para contato t [20], usando a equação (2.41).

Sendo assim, o tempo para contato foi calculado para cada região da imagem. Posteriormente, um único valor de tempo para contato foi determinado para representar cada um dos objetos obtidos pelo processo de segmentação. Uma medida conservativa, neste caso, consiste em escolher o menor tempo para contato de todas as regiões pertencentes ao objeto para representá-lo. Porém, nos experimentos observou-se que os tempos para contato estimados eram menores do que os valores reais. Assim, convencionou-se escolher o valor mediano dos tempos para contato das regiões pertencentes ao objeto.

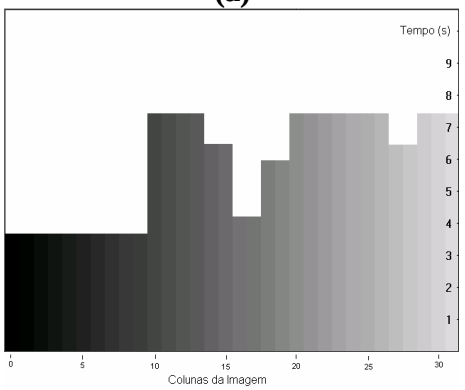
Com os valores de tempo para contato a cada objeto, foi montado um diagrama, mostrando qual o menor valor de tempo para contato encontrado entre os objetos posicionados em cada coluna de regiões, representando, portanto, o objeto mais próximo naquela direção do campo visual.

Os valores de tempo para contato no diagrama podem ser usados para determinar se algum dos objetos no campo de visão do robô oferece risco ao mesmo, caso sua trajetória seja mantida, e, em caso de necessidade de mudança de direção, qual a melhor estratégia para evitar estes obstáculos.

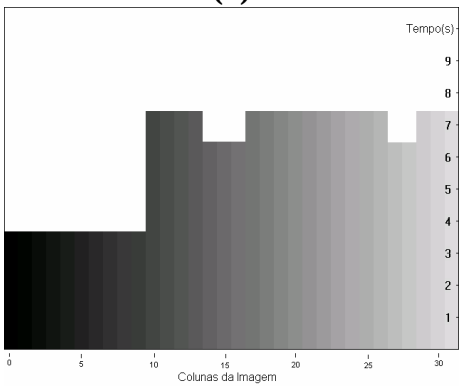
A Figura 3.5(a) mostra o diagrama de tempos para contato para as 32 colunas de regiões em que foi dividida a imagem da Figura 3.4(a). Pela figura, pode-se perceber que existe um objeto bastante próximo ao robô, ocupando a região mais à esquerda do seu campo visual, entre as colunas 0 e 9, o que era esperado, como pode ser confirmado examinando-se a Figura 3.4(a). O sistema detecta um objeto entre as regiões de 10 a 31 com um tempo para contato maior, correspondente ao objeto posicionado à direita da Figura 3.4(a), tal como esperado. Porém, o sistema detecta, erroneamente, alguns objetos com tempo para contato menor entre as regiões 14 e 22. A Figura 3.5(b) mostra o resultado obtido para a mesma seqüência, ao descartar os vetores de fluxo óptico não confiáveis, de acordo com a equação (2.42). A Figura 3.5(c) mostra o diagrama obtido ao descartar os objetos compostos por apenas uma ou duas regiões. Isto porque, nos experimentos realizados, notou-se que eles são resultantes de vetores de fluxo óptico errôneos. Entretanto, deve-se levar em conta que descartar objetos pode não ser uma boa estratégia, quando se trabalha em um ambiente com objetos pequenos, em comparação com o tamanho das regiões em que foi dividida a imagem.



(a)



(b)



(c)

Figura 3.5 – (a) Diagrama de tempos para contato, (b) mesmo diagrama usando medida de confiabilidade; (c) diagrama obtido descartando objetos pequenos.

3.5 Exemplos

Para mostrar a capacidade do procedimento implementado para a detecção de obstáculos na cena, são apresentados, a seguir, alguns exemplos. As seqüências de imagens apresentadas foram obtidas pela câmara de vídeo a bordo do robô e foram, em seguida, armazenadas.

3.5.1 Primeiro Exemplo

O primeiro exemplo, mostrado na Figura 3.6, apresenta o resultado obtido para a seqüência de imagens que foi utilizada na descrição do método. Este é um experimento onde a textura adicionada aos objetos facilita o cálculo do fluxo óptico. Na parte (a) da figura é apresentado o primeiro quadro de imagem na seqüência, com os vetores de fluxo óptico superpostos e o FOE indicado como um pequeno círculo. Na parte (b) é mostrado o resultado do processo de segmentação, sendo que regiões vizinhas de mesma tonalidade de cinza representam um mesmo objeto, e que as regiões em branco foram descartadas do processo, porque correspondem a vetores de fluxo óptico não confiáveis, segundo a medida apresentada no Capítulo 2. Na parte (c) são mostrados apenas os objetos mais próximos ao robô, e na parte (d) é mostrado o diagrama de tempos para contato resultante.

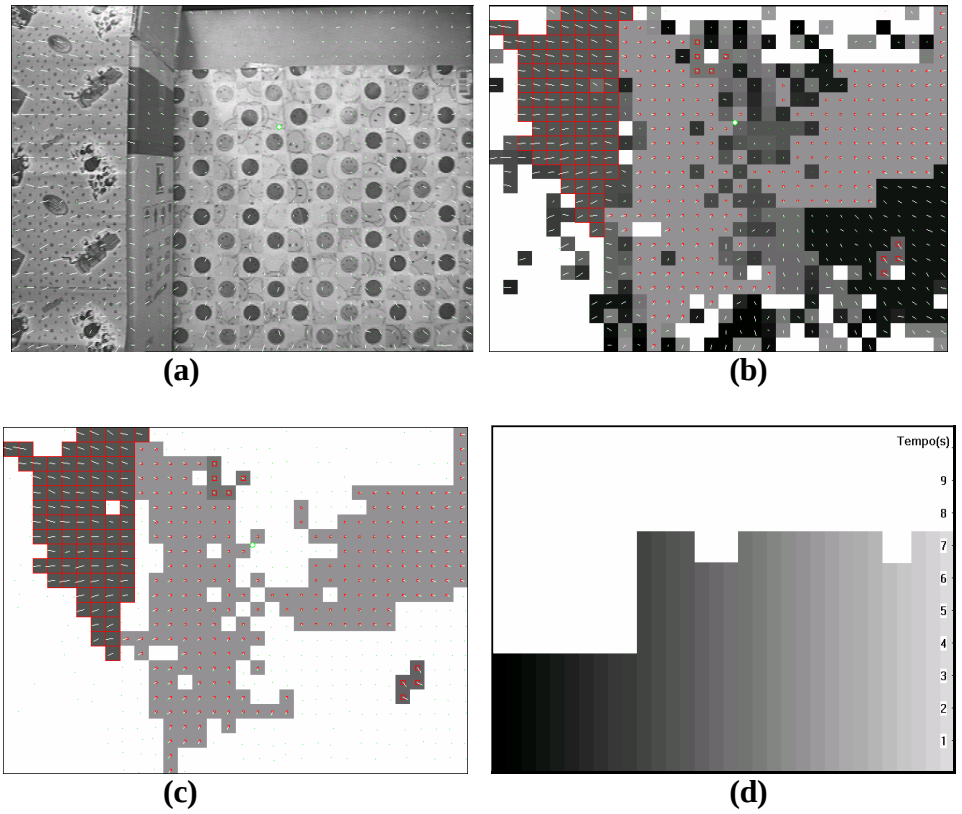


Figura 3.6 – Primeiro exemplo: (a) primeira imagem ; (b) resultado da segmentação; (c) objetos mais próximos ao robô; (d) diagrama de tempos para contato.

Nestas figuras, pode-se ver que, apesar do processo de segmentação não levar a uma identificação precisa dos objetos na cena, o diagrama de tempos para contato resultante do processamento representa bem o perfil do ambiente diante do robô, fornecendo, portanto, informação que pode ser utilizada para controlar a sua navegação.

3.5.2 Segundo Exemplo

Este segundo exemplo apresenta o resultado do processo de segmentação para uma seqüência de imagens obtida no laboratório. Este é um experimento interessante, porque mostra uma situação semelhante às que o robô vai encontrar durante a navegação. A Figura 3.7 ilustra o exemplo. Da mesma forma que na Figura 3.6, na parte (a) da Figura 3.7 é

mostrada a primeira imagem na seqüência, na parte (b) é mostrado o resultado do processo de segmentação (mais uma vez, regiões contíguas com mesma tonalidade de cinza representa um objeto, e as regiões em branco não foram utilizadas no processo). Na parte (c) da figura são mostrados apenas os objetos mais próximos ao robô e na parte (d) é mostrado o diagrama de tempos para contato correspondente.

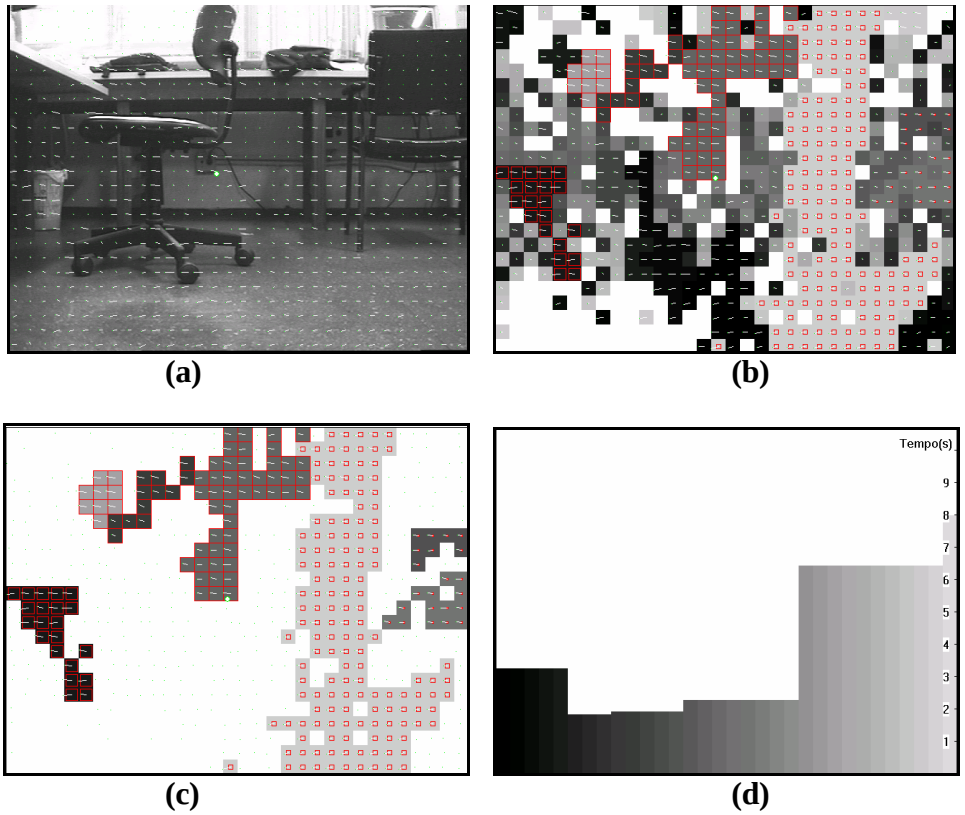


Figura 3.7 – Segundo exemplo: (a) primeira imagem; (b) resultado da segmentação; (c) objetos mais próximos ao robô; (d) diagrama de tempo para contato.

Na Figura 3.7, pode-se notar como a textura natural no piso causa o aparecimento de vetores de fluxo óptico espúrios, que acabam gerando problemas na segmentação. Além disto, há o problema associado à falta de textura em alguns objetos presentes no ambiente e à iluminação irregular com áreas muito claras, como as janelas, e outras mal iluminadas, como ocorre embaixo das mesas. Mas, apesar dessas imprecisões no cálculo do fluxo óptico e na segmentação, o diagrama de tempos para contato é capaz de refletir o perfil do ambiente em determinado instante.

3.5.3 Terceiro Exemplo

Este terceiro exemplo apresenta o resultado obtido para uma outra seqüência de imagens obtida no laboratório. É importante salientar, mais uma vez, que nenhuma textura foi adicionada aos objetos no ambiente com o intuito de facilitar o cálculo do fluxo óptico. Os resultados obtidos são mostrados na Figura 3.8, cuja leitura deve ser feita da mesma forma que no caso das figuras 3.6 e 3.7.

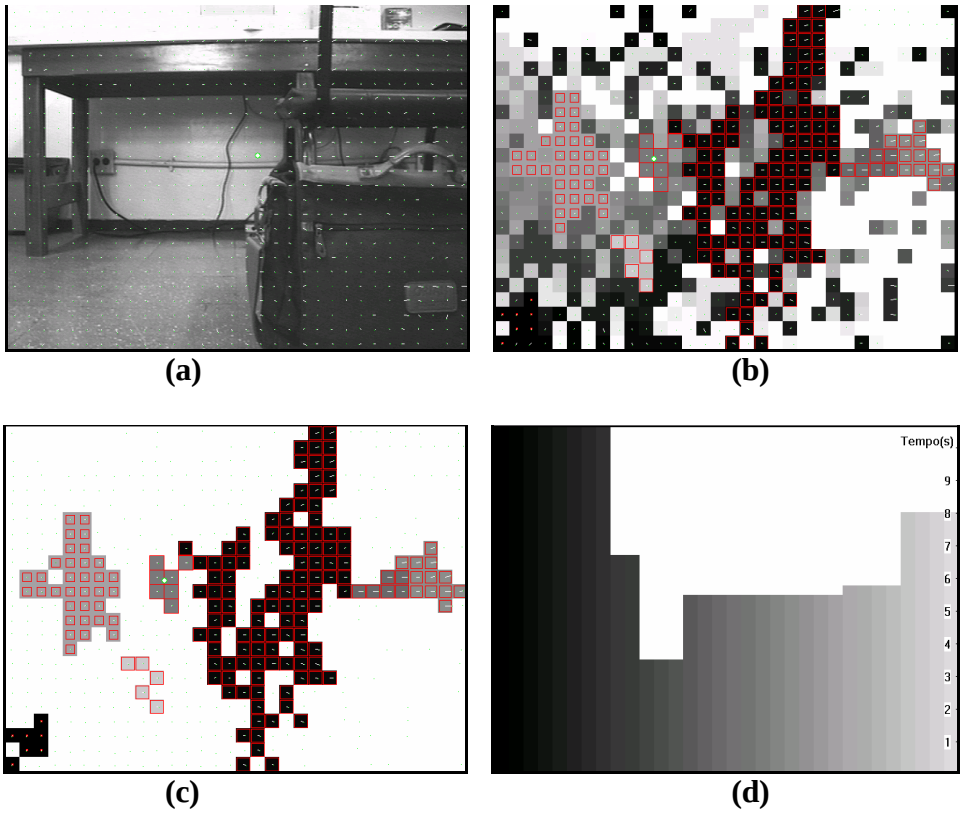


Figura 3.8 – Terceiro exemplo: (a) primeira imagem; (b) resultado da segmentação; (c) objetos mais próximos ao robô; (d) diagrama de tempo para contato.

Em tal figura, pode-se notar, uma vez mais, os problemas causados pela textura do piso do laboratório. Além disto, como neste caso a parede branca ao fundo da cena aparece bastante, é possível notar, na segmentação, o efeito do contraste com objetos mais escuros, como os pés da mesa ou os fios. Nestes casos, um vetor de fluxo óptico de maior magnitude é gerado, devido ao aumento no valor das derivadas, causando uma redução no tempo para contato, como pode ser visto na parte (d) da figura.

3.6 Considerações Finais

De acordo com os resultados obtidos no Capítulo 2, adquirindo-se uma seqüência de imagens com a câmara de vídeo a bordo do robô, os vetores de fluxo óptico podem ser obtidos pelo algoritmo de Mínimos Quadrados Modificado proposto neste trabalho. Mas para que o robô possa navegar no ambiente, é necessário determinar, usando o fluxo óptico, quais os objetos presentes na cena e a proximidade destes objetos à câmara de vídeo.

Considerando que regiões pertencentes a um mesmo objeto possuem vetores de fluxo óptico semelhantes, pode-se usar um método de segmentação de imagens com base no fluxo óptico para determinar os objetos na cena.

Tendo-se o vetor de fluxo óptico é possível se determinar o tempo para contato, ou seja, o tempo que o robô levaria para colidir com aquele ponto se mantivesse o movimento atual. Com base nesta informação, é possível determinar a proximidade do objeto e, portanto, se ele oferece risco ao robô.

Assim, foram estudados diversos métodos de segmentação de imagens com base no fluxo óptico. A maioria dos algoritmos disponíveis na literatura propõe a divisão da imagem em regiões e a associação de um modelo ao fluxo óptico dentro delas, sendo que o modelo afim é o mais comumente utilizado. As regiões são então agrupadas, usando uma medida de semelhança entre os parâmetros do modelo de cada região, através, em geral, de processos iterativos.

Entretanto, o algoritmo de cálculo do fluxo óptico considera que os vetores de fluxo óptico em cada região são constantes. Assim, foi proposto um método de segmentação que utiliza o modelo de fluxo óptico constante em cada região e que, em uma única varredura, reconhece como pertencentes a um mesmo objeto as regiões contíguas com fluxo óptico semelhante.

A seguir, os objetos obtidos no processo de segmentação são caracterizados pelo tempo para contato (o tempo para que o robô se choque com aquele objeto, se continuar seu movimento atual), e estes valores de tempo foram utilizados para montar um diagrama que informa, em cada direção do campo visual do robô, qual o menor tempo para contato, ou seja, permite inferir qual o objeto mais próximo ao robô. Este diagrama representa uma espécie de mapa de potenciais invertido, uma vez que o robô deve navegar sempre buscando as regiões de maior potencial, ou seja, buscando os objetos mais distantes.

Assim, neste capítulo determina-se o diagrama de tempos para contato, que é o resultado final do sistema sensorial proposto, o qual fornece as informações necessárias para o sistema de controle, que as utilizará para controlar a navegação do robô, para que ele navegue no ambiente de um laboratório evitando colisões com os objetos presentes no ambiente.

Capítulo 4- O Sistema de Controle de Navegação

O objetivo deste capítulo é mostrar como as informações obtidas a partir do sistema de sensoriamento visual baseado no fluxo óptico podem ser utilizadas para controlar a navegação de um robô móvel. Como foi dito no primeiro capítulo, onde foram apresentadas as características do problema, o robô deve apresentar um comportamento reativo [3], navegando pelo ambiente e evitando chocar-se com quaisquer objetos.

Sendo assim, o sistema de controle deveria ser projetado de maneira que fosse capaz de levar o robô a navegar pelo ambiente, mudando sua direção cada vez que fosse detectado algum risco à sua integridade, utilizando para isto as informações e restrições do sistema de sensoriamento implementado.

Como o sistema de sensoriamento utiliza a técnica do fluxo óptico, é necessário que haja movimento relativo entre o robô e os objetos presentes na cena. Sendo assim, considerando-se que os objetos estão estáticos, uma das restrições do sistema de sensoriamento é a de que o robô deve estar em movimento. Mais especificamente, neste sistema, o movimento deve ser de translação, ou seja, o robô deve estar caminhando em linha reta e se aproximando da cena capturada por sua câmara de vídeo.

Quando o robô está em um movimento de aproximação, todos os vetores de fluxo óptico estimados a partir das imagens adquiridas por sua câmara são radiais estão partindo de um ponto único, o chamado foco de expansão (FOE). Além disto, determinando-se o foco de expansão é possível estimar o tempo para contato a cada *pixel* na imagem, obtendo assim uma estimativa das distâncias aos objetos na cena. Portanto, a consideração de que o robô se move em translação simplifica grandemente a extração de informações das imagens adquiridas pela câmara de vídeo a bordo do robô.

Sendo assim, quando o robô estiver executando um giro para desviar-se de algum obstáculo, ele não pode adquirir imagens para executar o processamento. Por outro lado, durante a execução de um comando de rotação de um ângulo, o sistema de controle das rodas reduz a velocidade linear e aumenta a velocidade angular do robô, fazendo com que ele gire, praticamente, sem avançar. Tendo efetuado o giro, o sistema de controle reduz a velocidade angular e retorna a velocidade linear ao valor comandado antes do giro. Desta maneira, não existe risco de colisão ao deixar de processar imagens durante o movimento de rotação.

Um sistema de controle adequado para uma tarefa com estas restrições pode ser implementado usando a seqüência *Iniciar movimento de translação → Adquirir e processar imagens → Determinar o ângulo de giro necessário para desviar de obstáculos → Girar → Iniciar movimento de translação*. Partindo desta seqüência, foi desenvolvido o sistema de controle apresentado no diagrama de estados da Figura 4.1.

Neste sistema, o estado Avaliação é responsável por adquirir e processar as imagens, e analisar os resultados deste processamento, a fim de decidir qual a ação a ser executada para manter o robô navegando em segurança. O sistema de avaliação se baseia apenas nas imagens capturadas pela câmara de vídeo a bordo do robô, para decidir qual a ação mais apropriada. O sistema supõe que o robô está em translação durante a aquisição das imagens. Assim, ao detectar um obstáculo, o robô é comandado a girar de um determinado ângulo, que pode ser positivo ou negativo, voltando depois ao movimento de translação.

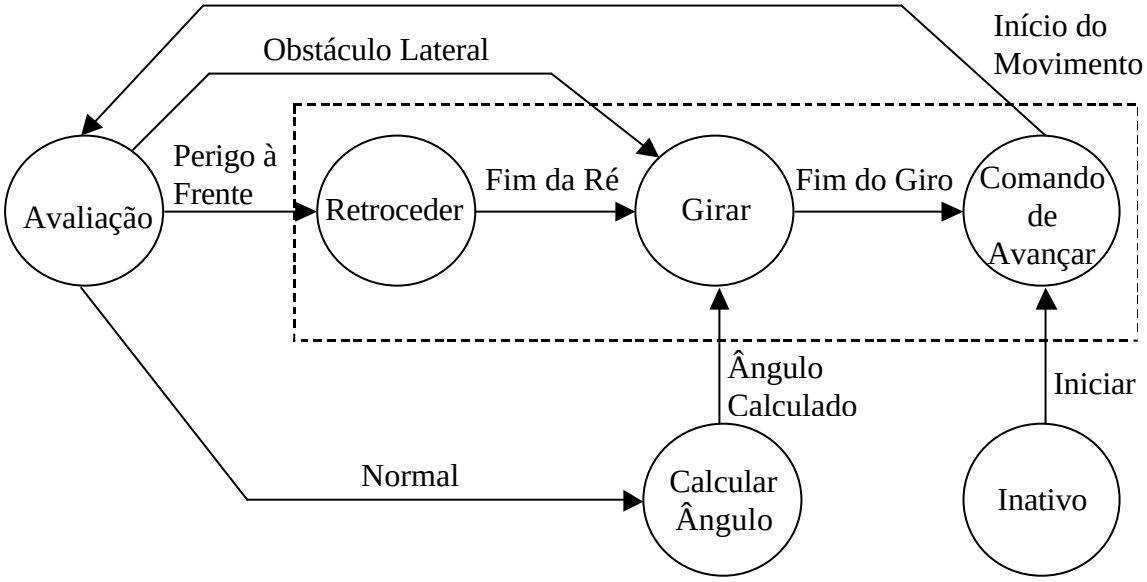


Figura 4.1 – Diagrama de estados do sistema.

Em funcionamento normal, o robô parte do estado Inativo através do comando Iniciar. Então, ele é comandado a avançar, entrando no estado de Avaliação, onde é calculado o diagrama de tempos para contato discutido no Capítulo 3. O sistema vai, então, para o estado Calcular Ângulo, onde, a partir do diagrama de tempos para contato, é calculado o ângulo necessário para o robô desviar-se dos obstáculos mais próximos. O sistema vai, então, ao estado Girar, onde o robô varia sua direção do ângulo calculado. Ao terminar o giro, o robô é novamente comandado a avançar, recomeçando o ciclo.

As outras transições na máquina de estados da Figura 4.1 referem-se a situações de anormalidade, onde algum perigo à segurança do robô é detectado, e uma ação evasiva mais rápida se faz necessária. Além disto, o sistema avalia as condições de saída dentro dos estados relacionados aos comandos, os quais estão localizados dentro da área pontilhada, para uma possível interrupção do ciclo do sistema.

No Capítulo 3 foi mostrado como é possível inferir, ainda que de maneira qualitativa, quais os objetos mais próximos, usando o fluxo óptico calculado para uma seqüência de imagens obtidas pela câmara do robô. A seguir, será mostrado como o diagrama de tempos para contato pode ser usado para calcular o ângulo de giro necessário para evitar os obstáculos detectados. Além disto, serão apresentadas outras formas de utilização das informações do diagrama de tempos para contato e do resultado do processo de segmentação.

4.1 Cálculo do Ângulo de Giro Usando o Diagrama de Tempos para Contato

Neste trabalho, cada coluna de regiões de N×N *pixels* da imagem é considerada como um sensor. Sendo assim, cada coluna no diagrama de tempos para contato representa o obstáculo mais próximo naquela direção do campo visual do robô. O controlador utilizado associa a cada sensor um ângulo de giro, de forma a fazer com que o robô desvie do obstáculo localizado na direção em que o sensor está posicionado. O ângulo total que o robô deve girar pode ser determinado, então, por um método de fusão de sensores [28] [29] [46].

O termo fusão de sensores se refere a um caso particular de integração de sensores, em que os dados de vários dispositivos de sensoriamento são combinados em um padrão único, para depois serem fusionados. Note-se que isto ocorre no caso dos diversos sensores aqui considerados, de forma que a fusão é absolutamente factível. O Filtro de Kalman e o Filtro de Informação podem ser usados para fazer a fusão de dados vindos de dois ou mais sensores de maneira ótima [28] [29]. Desta forma, os dados resultantes da fusão são mais precisos e confiáveis que os dados de cada sensor individual.

Para implementar o controlador do robô, foi utilizado um Filtro de Kalman Descentralizado [11]. Este filtro consiste em um nível de filtros locais, que tomam os dados dos sensores, processam e entregam a um filtro global, que combina todas essas informações, produzindo um único resultado, como mostrado na Figura 4.2.

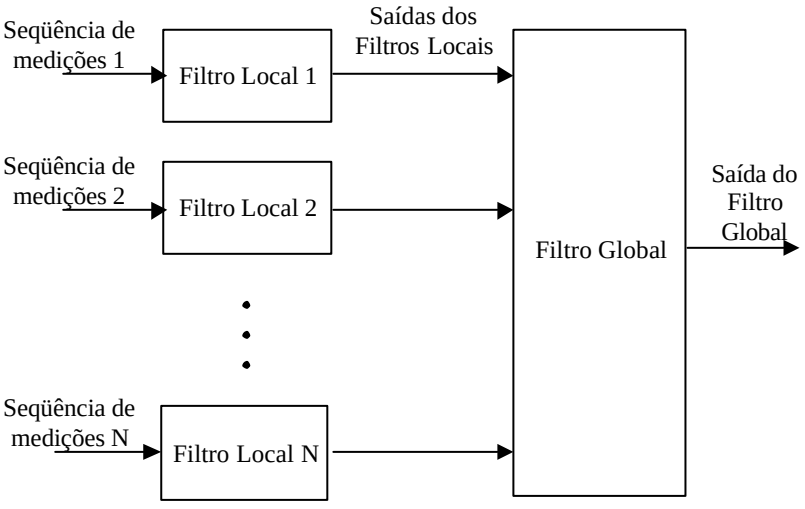


Figura 4.2 – Filtro de Kalman Descentralizado [11].

Para projetar o controlador utilizando as informações do diagrama de tempos para contato, a primeira idéia foi associar um ângulo a cada tempo para contato t_i no diagrama, dado por

$$q_i = sgn(i - 15.5)\frac{a_i}{t_i + d} \quad \text{para } i = 0, 1, ..., 31 \tag{4.1}$$

onde

$$a_i = \begin{cases} i & \text{para } i \leq 15 \\ 31 - i & \text{para i} > 15 \end{cases} \tag{4.2}$$

e *sgn*(*i*-15.5) dá o sinal da expressão. Esta função faz com que os valores de ângulo da metade da esquerda da imagem ($i \leq 15$) sejam negativos, causando ao robô um giro para a direita. Da mesma forma, os ângulos gerados pela equação são positivos para as colunas à direita na imagem, causando um giro do robô para a esquerda. O valor de t_i está no denominador porque, como ele é inversamente proporcional à distância ao objeto na cena, quanto menor seu valor mais próximo está o objeto, e maior deve ser o ângulo de giro, a fim de que o robô evite colisões com o obstáculo. Neste denominador, o t_i aparece somado a um valor d para evitar divisão por zero. Além disso, alterando o valor de d é possível controlar o ganho do controlador, ou seja, para um valor próximo a zero o sistema gera um ângulo maior e com um valor maior de d o ângulo se torna menos sensível a variações em t_i. Nos experimentos realizados com este controlador foi utilizado $d = 1$.

Para a fusão dos valores de ângulo de cada coluna da imagem, a fim de gerar um único ângulo de giro, usando um Filtro de Kalman Descentralizado, é necessário saber a variância associada a cada valor de ângulo. Como, neste caso, os ângulos não são valores medidos, é preciso usar uma estimativa da variância.

Uma forma de estabelecer um valor para a variância é utilizar uma medida da relevância de cada coluna com relação à segurança do robô. Sendo assim, o valor associado a cada ângulo na equação (4.1) pode ser constante para cada coluna e diminuir à medida que se aproxima do centro da imagem, dando, portanto, mais importância às colunas do centro do campo visual do robô. Neste trabalho, o valor utilizado como variância nos experimentos realizados com este controlador, é dado por

$$s_i^2 = |i - 15.5| \quad \text{para } i = 0, 1, ..., 31 \tag{4.3}$$

No entanto, este controlador não levou o robô a um comportamento adequado, porque anula a influência de objetos posicionados em simetria em relação ao centro da imagem, e com mesmo valor de tempo para contato, gerando um ângulo de saída zero no caso de simetria perfeita. Em geral, em situações reais a simetria não é perfeita, mas, ainda assim, o ângulo resultante é muito pequeno. Como um exemplo de uma situação em que o controlador falha, podemos mencionar o caso de uma aproximação a uma parede. Independentemente da distância à parede, a diferença de tempos para contato à direita e à esquerda no campo visual do robô é muito pequena, de forma que o controlador gera um ângulo de desvio muito pequeno, e o robô acaba por colidir.

Para contornar o problema de simetria, foi implementado um outro controlador, que consiste em calcular o ângulo de giro necessário para evitar os obstáculos presentes na cena, considerando uma única direção de giro por vez. Ou seja, calcular o ângulo que o robô deveria girar se lhe fosse permitido girar apenas para a direita e depois o ângulo necessário para evitar os obstáculos girando para a esquerda. Os dois ângulos globais são obtidos independentemente usando um filtro de Kalman descentralizado para cada sentido de giro. Enfim, para evitar que o robô execute giros muito bruscos desnecessariamente, o menor dos dois ângulos obtidos é comandado ao robô.

Os ângulos para giro à direita q_{di}, correspondentes aos tempos para contato t_i de cada coluna da imagem, foram calculados como

$$q_{di} = -\frac{a_{di}}{t_i + 1} \text{ para } i = 0, 1,...,31 \tag{4.4}$$

onde $a_{di} = ki$ para k constante. Ou seja, considerando-se um mesmo valor de tempo para contato em todas as colunas da imagem, quanto mais à direita estiver a coluna, maior o ângulo que o robô deve girar para a direita a fim de evitar colisões com o obstáculo naquela direção. Na equação acima, o sinal negativo implica em giros para a direita.

De maneira similar, os ângulos de giro para a esquerda q_{ei}, correspondentes aos tempos para contato t_i de cada coluna da imagem, são dados por

$$q_{ei} = \frac{a_{ei}}{t_i + 1} \text{ para } i = 0, 1,...,31 \tag{4.5}$$

onde $a_{ei} = k(31 - i)$ para k constante. Neste caso, para um mesmo valor de tempo para contato, quanto mais à esquerda estiver a coluna, maior o ângulo que o robô deve girar para a esquerda, a fim de evitar colidir com o objeto naquela direção. Nos experimentos realizados neste trabalho, foi utilizado um valor de k = 3 para obter os ângulos de giro à direita e à esquerda.

Além disto, nos experimentos realizados, notou-se que, como o ângulo de visada da câmara do robô é muito pequeno (48.8^o), dar mais importância a uma coluna do que a outra no cálculo do ângulo não seria muito adequado, invalidando, assim, o valor de variância dado na equação (4.3). Desta maneira, neste controlador, o valor utilizado como variância é o próprio valor de tempo para contato t_i. Assim, para uma dada coluna, quanto menor o valor do tempo para contato, mais importante é o valor de ângulo desta coluna no ângulo global calculado pelo filtro de Kalman descentralizado. Ou seja,

$$s_i^2 = t_i \text{ para } i = 0, 1, ..., 31 \tag{4.6}$$

Assim, usando dois filtros de Kalman descentralizados, são obtidos um ângulo de giro global para a direita q_d e outro para a esquerda q_e. O ângulo de giro que deve ser comandado ao robô é obtido como o menor valor entre os dois, ou seja,

$$q = \min(q_d, q_e) \tag{4.7}$$

permitindo, assim, controlar o robô com movimentos mais suaves, evitando, sempre que possível, giros com ângulos muito grandes.

4.2 Detecção de Paredes

O controlador apresentado na seção anterior foi projetado para manter o robô navegando pelo ambiente, desviando dos obstáculos que porventura apareçam no seu campo visual. Para executar essa função de maneira apropriada, é necessário que os dados do sensoriamento visual sejam bastante confiáveis.

Nos experimentos realizados neste trabalho, nenhuma modificação foi feita no ambiente no sentido de facilitar o cálculo do fluxo óptico, como em ambientes com iluminação controlada, onde os objetos possuam textura, ou seja, em ambientes artificialmente preparados para a navegação. Por isso, os dados obtidos a partir do sistema de sensoriamento têm um pouco menos de robustez do que no caso ideal. Sendo assim, foi necessário fazer a relação entre o tempo para contato medido e o ângulo comandado com um ganho bastante pequeno para evitar que o robô apresente reações muito bruscas sem necessidade. Desta maneira, apesar de o sistema medir um tempo para contato bem pequeno, o ângulo de giro é bastante suave, possibilitando, assim, que o robô navegue pelo ambiente, desviando-se de obstáculos de maneira satisfatória, quando eles estão a uma certa distância do robô.

Assim, em situações de iminência de colisão, o sistema não é capaz de tomar uma ação evasiva suficientemente forte para evitar colisões. Por este motivo, foi projetado um outro controlador, responsável pela detecção de superfícies planas verticais, tais como paredes ou objetos muito grandes, usando os resultados obtidos no processamento das imagens. Além disto, a magnitude dos vetores de fluxo óptico foi utilizada para determinar a presença de objetos muito próximos ao robô.

4.2.1 Detecção de Paredes Usando o Diagrama de Tempos Para Contato

No Capítulo 2 foi apresentado um experimento em que o robô se aproxima de uma parede, com o eixo óptico da câmara perpendicular a ela. Os tempos para contato calculados a partir dos vetores de fluxo óptico têm, aproximadamente, o mesmo valor em todas as regiões da imagem. Sendo assim, o diagrama de tempos para contato esperado tem um valor constante em todas as direções do campo visual. Este conhecimento pode ser aplicado ao sistema para a detecção de paredes ou objetos planos verticais que ocupem todo campo visual do robô.

Para determinar se o diagrama de tempos para contato obtido representa a situação descrita acima, basta tomar o valor médio entre os valores no diagrama em todas as direções, e a variância desses valores em relação ao valor médio. Se o valor médio dos tempos para contato no diagrama é pequeno, e sua variância também é pequena, o robô está se aproximando de uma parede.

4.2.2 Detecção de Paredes Usando o Resultado da Segmentação

Uma outra característica que pode ser utilizada para a detecção de paredes é obtida do resultado do processo de segmentação. Como no experimento de aproximação a uma parede, como aquele apresentado no Capítulo 2, o fluxo óptico é predominantemente radial e aumenta

gradativamente em magnitude à medida que as regiões se afastam do foco de expansão, no processo de segmentação, há o aparecimento de poucos objetos na cena. Em geral, um ou dois objetos ocupam a maior parte da imagem. Sendo assim, identificando o maior ou os dois maiores objetos resultantes do processo de segmentação, e verificando se eles ocupam uma área muito grande da imagem, pode-se concluir que existe um objeto grande ou uma parede à frente do robô.

4.2.3 Exemplos

Para demonstrar a utilização destas estratégias para a detecção de paredes, foram realizados dois experimentos de aproximação a uma parede, semelhantes aos apresentados no Capítulo 2. Para simular esta situação, no sistema da Figura 4.1 o estado Calcular Ângulo passou a gerar ângulos iguais a zero, independentemente dos valores de tempo para contato medidos. Desta maneira, apenas os comandos vindos do estado Retroceder, correspondendo a um giro de 180^o, eram enviados ao robô, permitindo, assim, que ele evitasse a parede. Em ambos os experimentos, o ciclo foi interrompido pelo usuário assim que o robô detectou a parede e retornou. No primeiro experimento, foi usada uma superfície plana coberta com um papel com textura para simular uma parede. No segundo experimento, o robô foi colocado diante de uma parede do laboratório.

Na Figura 4.3 em (a) é apresentada a primeira imagem adquirida, em (b) o primeiro quadro da quarta seqüência de imagens processada, em (c) a imagem onde foi detectada a parede e, em (d), a primeira imagem adquirida depois de retornar.

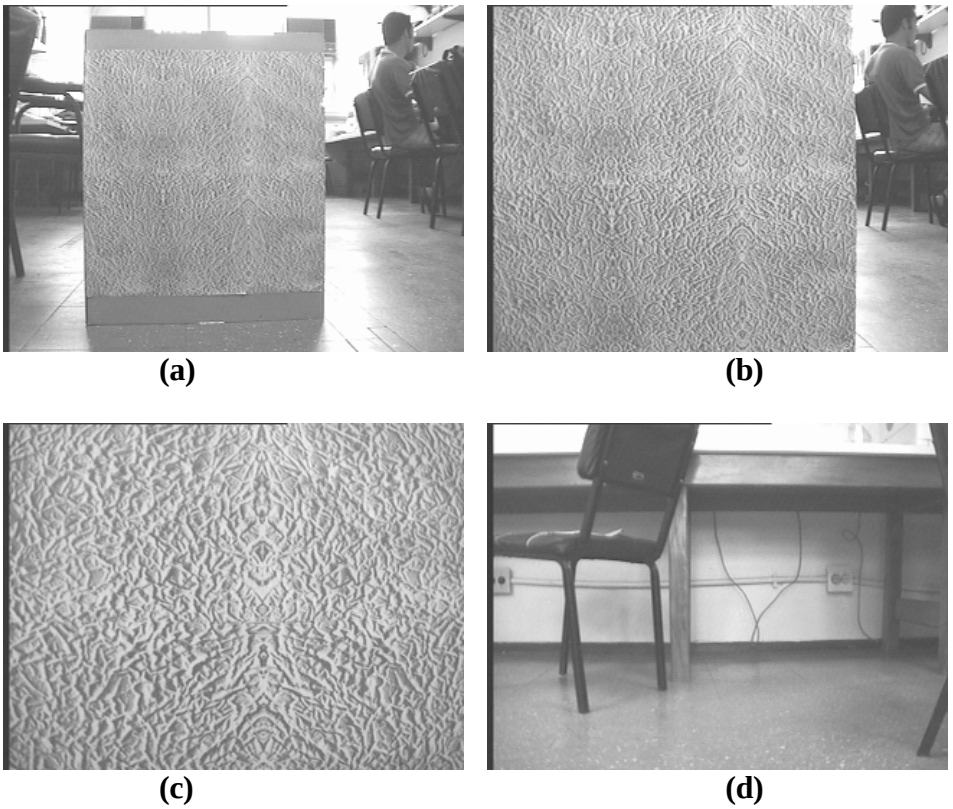


Figura 4.3 – Detecção de parede: (a) primeira imagem; (b) imagem da quarta seqüência; (c) imagem onde foi detectada a parede; (d) imagem obtida após o giro de 180^o.

A Figura 4.4 mostra o diagrama de tempos para contato e o resultado da segmentação para a imagem onde foi detectada a parede. Nesta figura, em (a) e (b), é apresentado o quadro de imagem onde a parede foi detectada, com os vetores de fluxo óptico desenhados, em (c) é

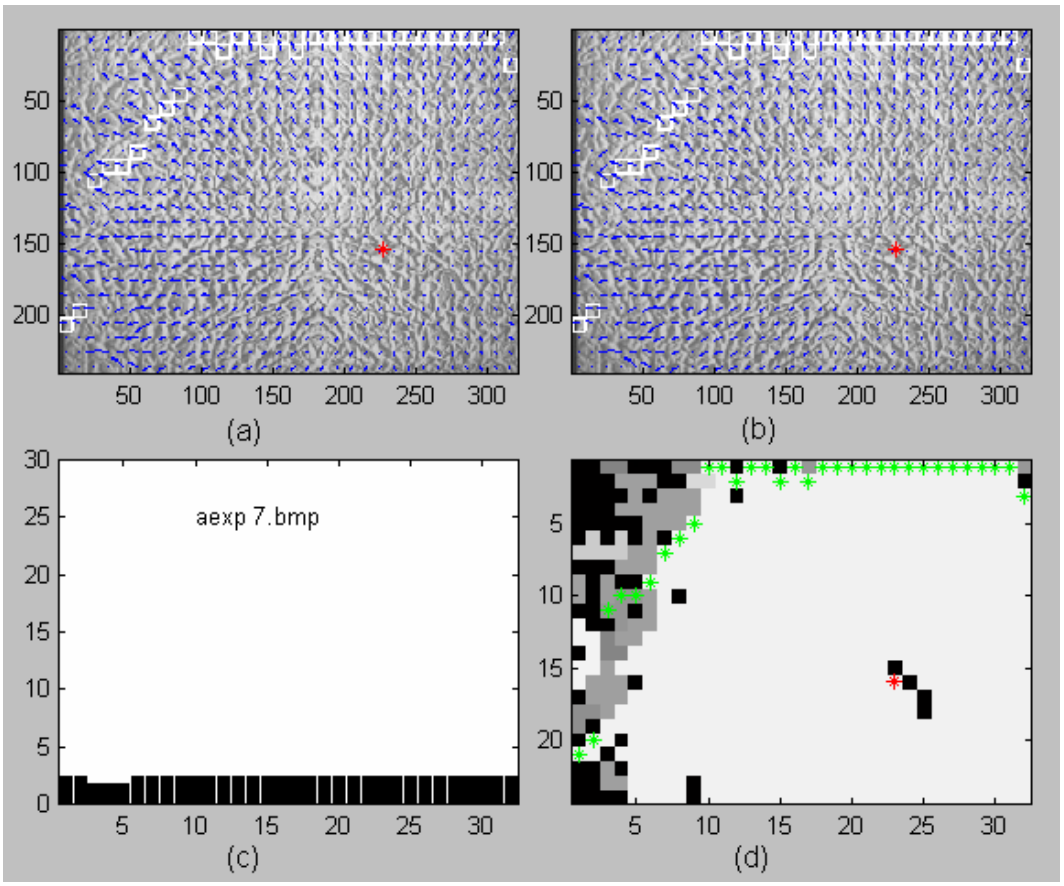


Figura 4.4 – Detecção de parede: (a) e (b) imagem onde a parede é detectada com vetores de fluxo óptico; (c) diagrama de tempos para contato; (d) resultado da segmentação.

mostrado o diagrama de tempos para contato, e, em (d), é mostrado o resultado do processo de segmentação.

É importante notar que este experimento foi realizado com uma parede onde foi adicionada textura, o que facilita o cálculo do fluxo óptico. No diagrama de tempos pode-se notar que os valores são, praticamente, os mesmos para todas as direções do campo visual do robô. No resultado da segmentação, pode-se verificar que um único objeto ocupa quase toda a imagem.

A Figura 4.5 mostra as imagens de um experimento semelhante, agora com o robô se aproximando de uma parede do laboratório. Nesta figura, em (a) é mostrado o primeiro quadro de imagem obtido no experimento, em (b), o primeiro quadro da quarta seqüência de imagens adquirida, em (c), o quadro de imagem onde é detectada a parede, e, em (d), o primeiro quadro de imagem obtido após o giro de 180^{o}.

Na Figura 4.6 são apresentados os resultados da segmentação e o diagrama de tempos para contato no instante em que a parede foi detectada. Em (a) e (b) é apresentado o quadro de imagem analisado com o fluxo óptico desenhado, em (c) é mostrado o diagrama de tempos para contato e, em (d), o resultado da segmentação.

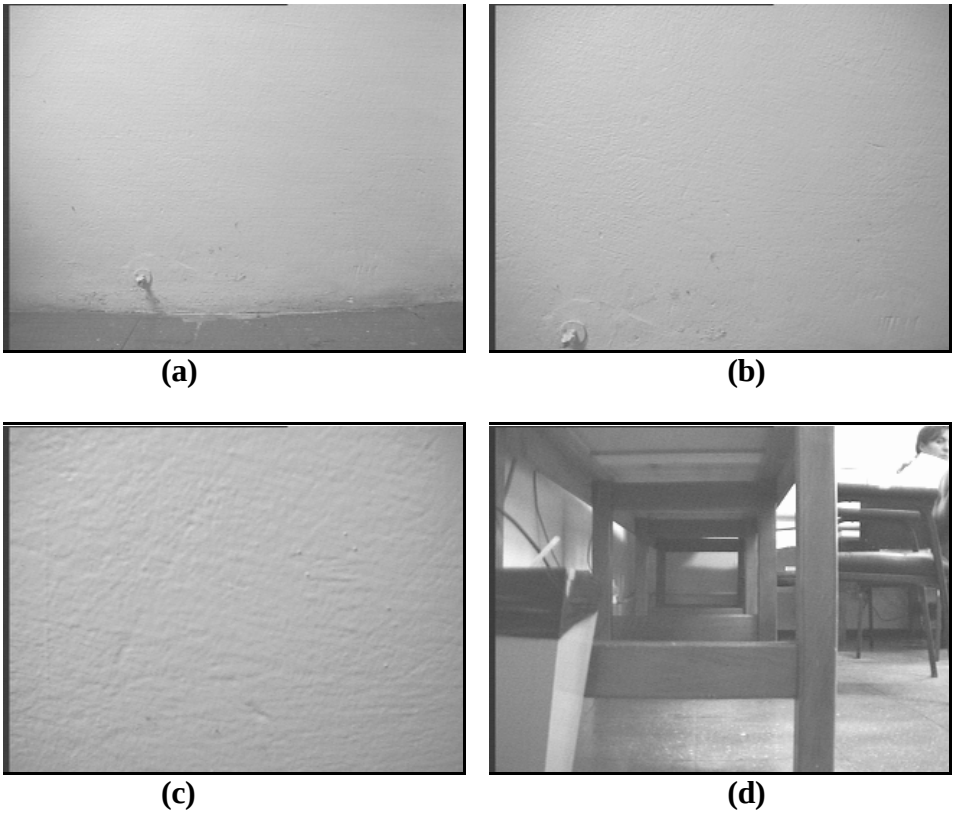


Figura 4.5 – Detecção de parede: (a) primeira imagem obtida; (b) primeiro quadro da quarta seqüência; (c) imagem onde foi detectada a parede; (d) primeira imagem obtida após giro de 180º.

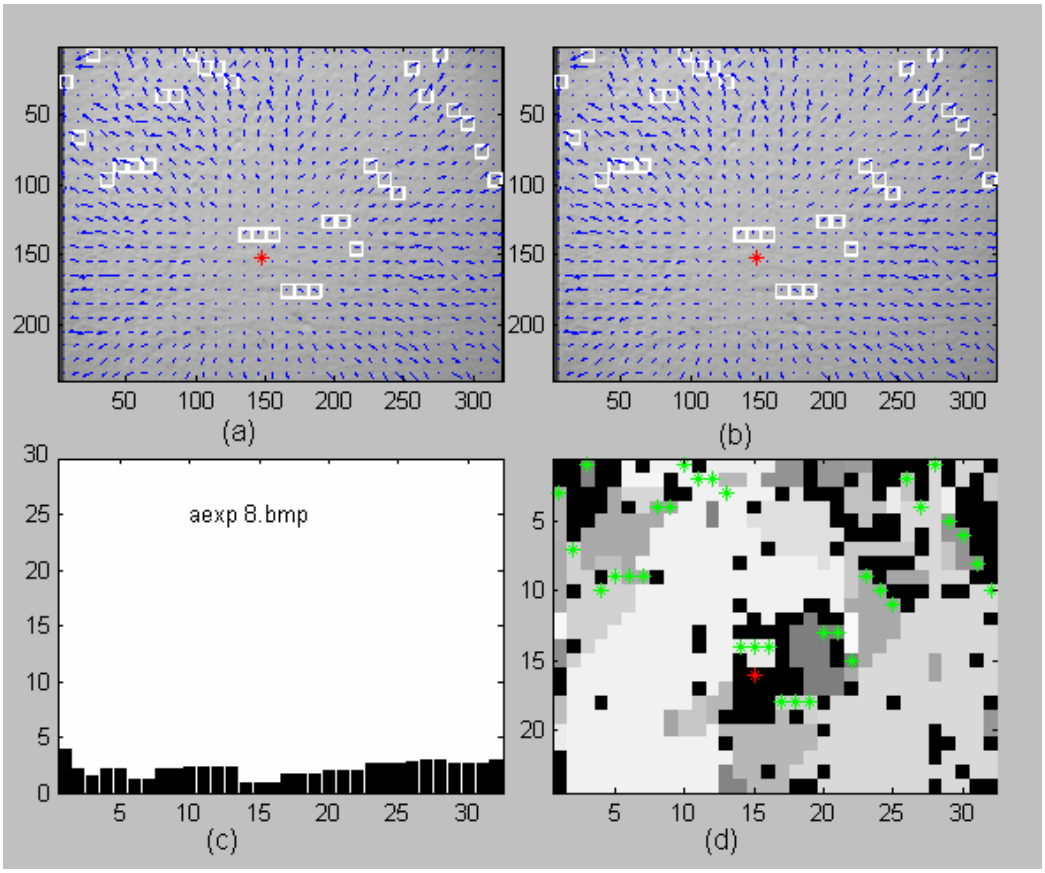


Figura 4.6 – Detecção de parede: (a) e (b) imagem onde a parede foi detectada; (c) diagrama de tempos para contato; (d) resultado da segmentação.

Nestas figuras, pode-se notar que o sistema consegue detectar a parede, baseando-se apenas nas imperfeições naturalmente presentes, ou seja, sem que nenhuma textura seja adicionada. O diagrama de tempos para contato não é tão uniforme quanto no exemplo anterior, porém o valor médio e variância dos tempos representados são suficientemente pequenos. Além disto, embora o processo de segmentação gere mais objetos, dois deles ocupam grande parte da imagem. Portanto, ambos os sistemas levam o robô à decisão esperada.

4.2.4 Influência do Movimento de Rotação

Um problema encontrado com os sistemas para a detecção de objetos planos verticais apresentados anteriormente é que eles supõem que o robô está executando um movimento de translação. Porém, em muitas situações práticas o robô desenvolve algum tipo de movimento de rotação, ainda que tenha sido comandado a manter uma velocidade linear constante, com velocidade angular e ângulo de giro iguais a zero.

O movimento de rotação mais comum acontece em torno do eixo vertical do robô, e pode ser conseqüência de um erro na verificação do término de um comando de giro (a verificação de conclusão de um comando de giro é feita de maneira indireta, verificando-se se o ângulo desejado foi alcançado e se a velocidade angular foi reduzida a zero) ou da incapacidade do controlador das rodas de manter uma velocidade linear constante.

Um outro fator gerador deste tipo de rotação é conseqüência da presença da roda livre sobre o movimento do robô. Isto acontece porque, depois de um giro do robô, apenas as rodas da tração diferencial estão alinhadas com o sentido de movimento. A terceira roda não está alinhada com o sentido de movimento e, como uma boa parte do peso do robô está sobre ela, no processo de alinhamento ela impede o movimento do robô na direção correta.

Um outro tipo de movimento de rotação ocorre quando alguma das rodas do robô passa por um defeito no piso, como uma depressão, uma emenda, uma fissura. Esta rotação causa uma variação do ângulo entre o eixo óptico da câmara e o eixo horizontal. O efeito nas imagens é semelhante ao que seria obtido se a câmara executasse uma variação no ângulo *tilt* durante o processo de aquisição.

No caso de uma rotação sobre o eixo vertical do robô, o fluxo óptico medido se assemelha àquele obtido nos experimentos de movimento paralelo à parede do Capítulo 2, ou seja, quase todos os vetores de fluxo óptico são horizontais e de módulo constante. Sendo assim, na presença deste tipo de rotação, o diagrama de tempos para contato poderá ser similar ao que se obtém quando há uma parede à frente. Além disso, como os vetores de fluxo óptico têm mesma direção e módulo em toda a imagem, o processo de segmentação pode agrupar a maior parte das regiões em um único objeto, caracterizando também a presença de uma parede à frente.

No caso de rotação devido a imperfeições do piso, em geral, ocorre o aparecimento de vetores de fluxo óptico praticamente verticais e de mesma magnitude em toda a imagem. Da mesma maneira que no caso anterior, na presença deste tipo de rotação o diagrama de tempos para contato e o resultado do processo de segmentação podem ser semelhantes aos resultados que se obtém quando o robô está se aproximando de uma parede.

A Figura 4.7 mostra um exemplo, em que a presença de rotação leva à detecção equivocada de uma parede. Nas partes (a) e (b) da figura pode-se ver que o fluxo óptico é praticamente horizontal em toda a imagem. Em (c) pode-se ver que os valores no diagrama de tempos para contato são bastante pequenos, e, em (d), que o resultado do processo de segmentação apresenta a maior parte da imagem como pertencente a um único objeto.

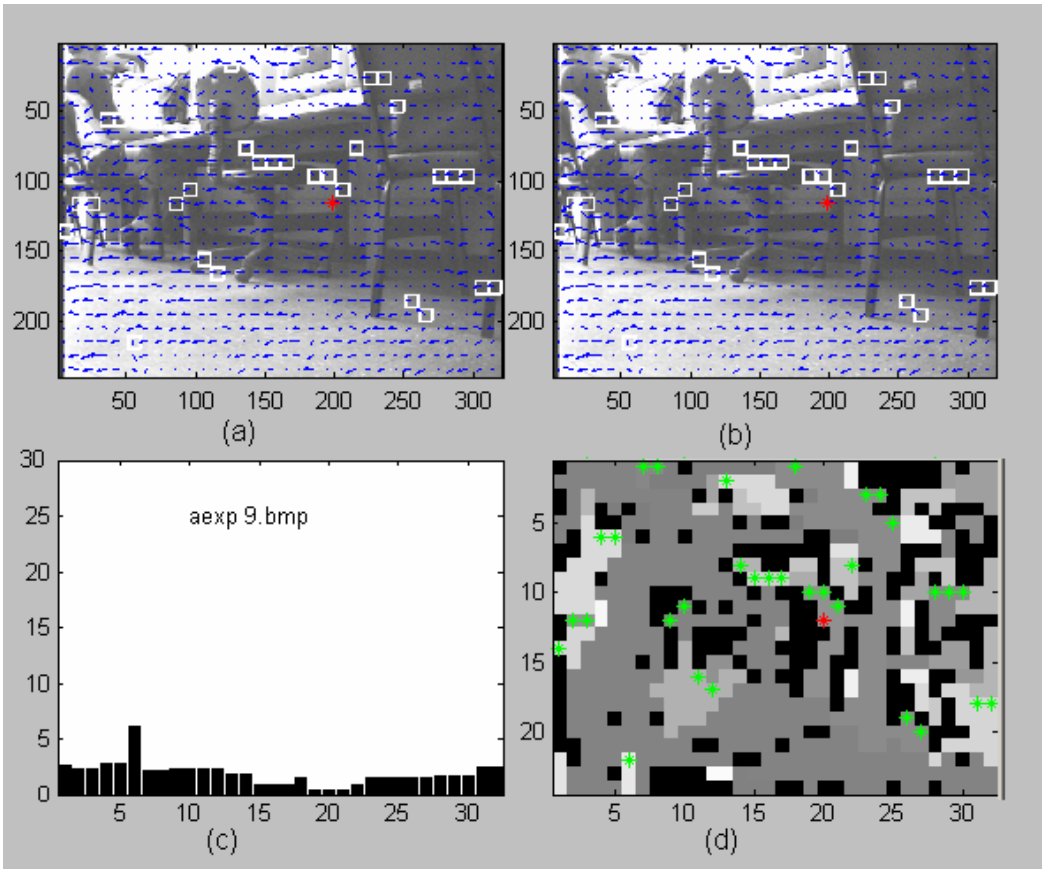


Figura 4.7 – Problemas na detecção de paredes devido a rotação: (a) e (b) imagem onde uma parede foi indevidamente detectada; (c) diagrama de tempos para contato ;(d) resultado da segmentação.

A velocidade de rotação em torno do eixo vertical do robô pode ser estimada, a partir dos vetores de fluxo óptico, e o termo devido à rotação pode ser eliminado, deixando apenas a componente devido ao movimento de translação do robô [20]. O processo, em geral, consiste em determinar a velocidade angular para cada vetor de fluxo óptico na imagem, e depois calcular o valor médio para ser usado como estimativa da velocidade angular para a imagem inteira. Depois, calcula-se e subtrai-se do vetor de fluxo óptico de cada região a componente devida a essa rotação, deixando apenas a componente devida ao movimento de translação. O problema em executar tal procedimento é o aumento no tempo necessário para o processamento, além dos erros que podem ser adicionados ao processo.

Sendo assim, neste trabalho, verifica-se a presença de rotação em torno do eixo vertical do robô nas imagens. Havendo rotação, os procedimentos apresentados anteriormente para detecção de paredes são desabilitados, por não produzirem resultados confiáveis. A rotação devido a imperfeições no piso é menos comum, por isso seu efeito não é considerado.

4.3 Detecção de Objetos pela Magnitude do Fluxo Óptico

Uma outra característica que pode ser explorada no resultado do processo de segmentação é que objetos mais próximos ao robô apresentam vetores de fluxo óptico maiores e que objetos mais distantes apresentam vetores de fluxo óptico de menor magnitude. Esta colocação considera que os objetos estão estáticos e que apenas o robô está em movimento.

Pode-se utilizar esta informação, calculando-se o valor médio da magnitude do fluxo óptico para cada objeto resultante do processo de segmentação e comparando-se o valor obtido a um determinado limiar, acima do qual o objeto representa um risco de colisão para o robô. O procedimento consiste em calcular a magnitude do fluxo óptico em cada região pertencente ao objeto e, então, calcular o valor médio das magnitudes de todas estas regiões. Se o valor calculado para um objeto resultante da segmentação excede o valor de limiar, conclui-se que o objeto que ele representa no mundo real está bastante próximo ao robô.

Como foi visto na seção anterior, os métodos de detecção de paredes apresentados anteriormente funcionam, satisfatoriamente, apenas se o robô não estiver executando nenhum tipo de rotação durante o processo de aquisição das imagens. Por outro lado, o método de detecção de obstáculos pelo valor médio da magnitude do fluxo óptico trabalha bem, mesmo em presença de rotação. Ao contrário, no caso de objetos verticais, por exemplo, a rotação favorece a detecção por este método.

Neste trabalho, com o intuito de utilizar esta informação, foi projetado um controlador que verifica a presença de objetos muito próximos ao robô por este método e comanda o robô a desviar-se destes obstáculos. Para uso neste controlador, a imagem foi dividida em três faixas de regiões: esquerda, central e direita. A primeira faixa compreende as oito colunas de regiões da esquerda, a segunda faixa é composta das dezesseis colunas de regiões ao centro da imagem, e a terceira contém as oito colunas de regiões restantes, localizadas à direita. Cada uma destas três regiões recebe uma marca, se existe algum objeto detectado por este método em alguma de suas colunas. Esta informação é, então, usada para controlar a navegação do robô, fazendo-o desviar destes obstáculos.

Os objetos localizados na região central são tratados com maior prioridade, porque têm uma relação muito direta com a segurança do robô, e, por isso, neste caso, o robô é comandado a girar 180^o, retornando por onde veio. Por outro lado, se os objetos mais próximos estiverem concentrados nas oito colunas da esquerda, o robô é comandado a girar 15^o para a direita e, se estiverem nas oito colunas da direita, o robô é comandado a girar 15^o para a esquerda. O valor de ângulo de giro, para estes casos, foi determinado por experimentação. Como o ângulo de visada da câmara é muito pequeno ($48{,}8^o$ apenas), se existirem objetos próximos ao robô nas duas regiões laterais, em geral, não será possível para o robô passar entre os dois obstáculos com segurança. Portanto, neste caso o tratamento será o mesmo como se houvesse um objeto na região central, e o robô é comandado a retornar.

A Figura 4.8 mostra um exemplo de detecção de objetos pela magnitude média do fluxo óptico. Em (a) é apresentado um dos quadros de imagem da seqüência processada e em (b) são mostradas em branco as regiões do(s) objeto(s) com magnitude média do fluxo óptico acima do valor de limiar. Nesta figura pode-se notar que o sistema detecta o movimento de uma mão, mostrando sua habilidade para detectar movimento de pessoas no ambiente. A reação programada para o robô, neste caso, consiste em girar 180^o, porque o objeto foi detectado na região central da imagem.

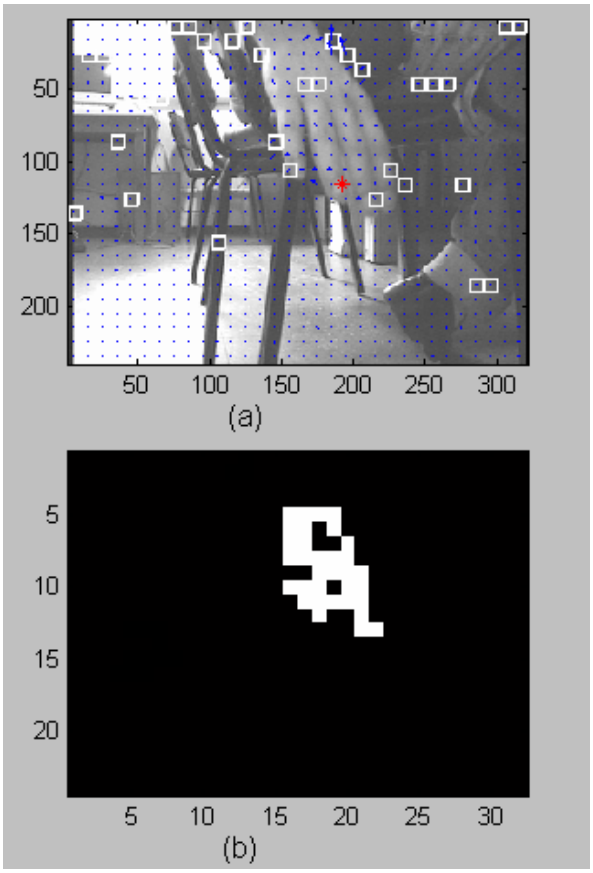


Figura 4.8 –Primeiro exemplo: (a) primeira imagem da seqüência; (b) regiões pertencentes ao(s) objeto(s) detectado(s) (em branco).

A Figura 4.9 apresenta um outro exemplo de detecção pelo método do fluxo óptico médio. Uma vez mais, em (a) é apresentado um dos quadros de imagem da seqüência para a qual foi calculado o fluxo óptico, e em (b) estão mostradas, em branco, as regiões pertencentes aos objetos com a magnitude média dos vetores de fluxo óptico acima do valor de limiar estabelecido. Neste exemplo, as regiões pertencentes aos objetos detectados ocupam a faixa delimitada à esquerda e parte da região ao centro. Sendo assim, o sistema deve dar prioridade ao desvio do objeto na região central, retornando na direção por onde veio.

Um terceiro exemplo é apresentado na Figura 4.10, onde é detectado um objeto localizado na faixa mais à direita da imagem, correspondente, mais uma vez, a uma pessoa se movendo. Nestes casos, em geral, ocorre o aparecimento de vetores de fluxo óptico de grande magnitude e, por isso, a detecção é feita mesmo a uma distância maior do que aquela em que haveria perigo real para a integridade do robô. Como, neste exemplo, o objeto detectado ocupa a faixa da direita, o robô é programado a girar 15^{o} na direção oposta, ou seja, para a esquerda.

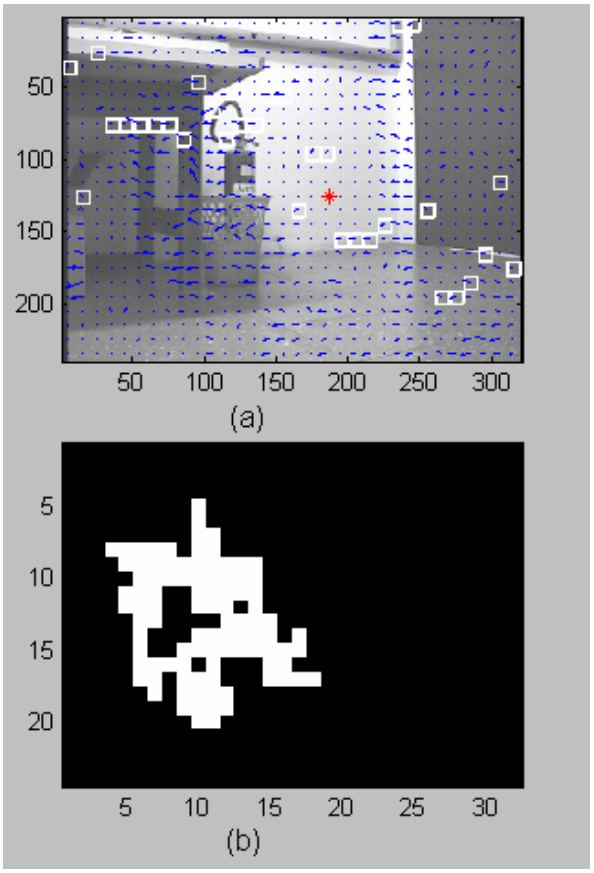


Figura 4.9 –Segundo exemplo: (a) primeira imagem da seqüência; (b) regiões pertencentes ao(s) objeto(s) detectado(s) (em branco).

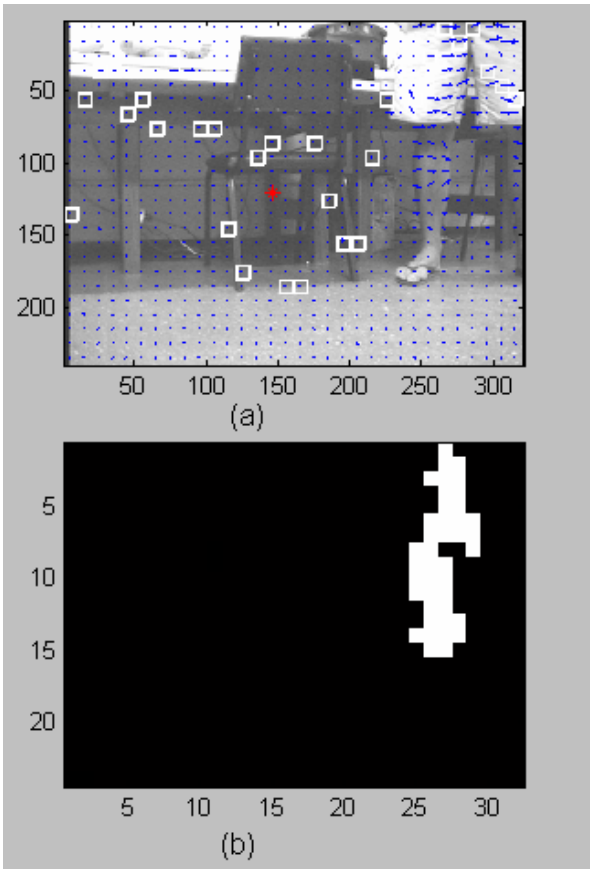


Figura 4.10 –Terceiro exemplo: (a) primeira imagem da seqüência analisada e (b) as regiões pertencentes ao(s) objeto(s) detectado(s) (em branco).

4.4 Descrição do Estado de Avaliação

Com foi dito no início deste capítulo, no estado de Avaliação o sistema adquire e analisa os dados sensoriais. O procedimento neste estado consiste em:

- Adquirir dois quadros de imagem;
- Calcular os vetores de fluxo óptico;
- Verificar se há rotação em torno do eixo vertical do robô;
- Segmentar a imagem baseando-se no fluxo óptico;
- Caracterizar os objetos pelas linhas e colunas de início e de fim;
- Selecionar os dois maiores objetos resultantes do processo de segmentação;
- Calcular o valor médio da magnitude do fluxo óptico por objeto;
- Calcular o foco de expansão (FOE);
- Determinar o tempo para contato por objeto;
- Montar o diagrama de tempos para contato;
- Determinar o valor médio e a variância dos tempos para contato no diagrama;
- Verificar se existe algum objeto com magnitude média do fluxo óptico acima do limiar;
- Determinar a condição detectada.

De acordo com os valores obtidos neste estado, vão ocorrer as mudanças de estado mostradas no diagrama da Figura 4.1.

As mudanças de estado a partir do estado Avaliação são apresentadas abaixo, por ordem de prioridade.

Perigo à Frente – Se o sistema detecta algum objeto à frente, que represente perigo iminente de colisão para o robô, ou seja, com um tempo para contato abaixo de um limiar, ele deve ser comandado a uma ação de evitar colisão à frente, retornando ao estado de Avaliação. Este deve ser o comportamento do sistema quando for detectada uma parede à frente, ou seja, quando, não havendo detectado rotação, o processo de segmentação resulta em objetos grandes ou o diagrama de tempos para contato tem pequenos valores de média e variância. Uma outra situação de perigo é detectada quando existe um objeto com o valor médio da magnitude do fluxo óptico maior que um determinado valor de referência, localizado na faixa central da imagem, ou então objetos, simultaneamente, nas duas regiões laterais.

Objeto Lateral – Se o sistema detecta um ou mais objetos com valor médio de magnitude do fluxo óptico acima do valor de tolerância, limitados a uma das faixas da direita ou da esquerda no campo visual, o robô deve ser comandado a girar 15^{o} na direção oposta, e retornar depois ao estado de Avaliação.

Normal – Se nenhum obstáculo que represente perigo iminente de colisão for detectado, o sistema vai para o estado Calcular Ângulo. Neste estado, o diagrama de tempos para contato obtido no estado de Avaliação é utilizado para calcular um ângulo de giro usando filtro de Kalman, como mostrado na seção 4.1. Este ângulo de giro é, então, comandado ao robô, e após o término da rotação, o sistema deve retornar ao estado de Avaliação.

4.5 Considerações Finais

Utilizando o método proposto no Capítulo 3 obtém-se um diagrama de tempos para contato que informa em cada direção do campo visual do robô qual o menor tempo para contato, que é o tempo que o robô levaria para colidir com o objeto naquela direção se continuasse com o movimento atual. Esta informação pode ser usada para controlar a navegação do robô porque informa sobre a proximidade dos objetos ao robô.

Assim, foi proposto um sistema de controle que utiliza o diagrama de tempos para contato para calcular uma nova direção segura para o robô e envia comandos de ângulos de giro a ele. O robô é colocado em movimento de translação, duas imagens são adquiridas e processadas, e, com base nos dados obtidos do sensoriamento visual, é determinado um ângulo de giro capaz de fazer o robô móvel desviar-se dos obstáculos detectados.

Ao contrário do que é usado normalmente, são enviados comandos de ângulo de giro ao robô e não de velocidade angular, respeitando assim uma restrição do sistema de sensoriamento visual que precisa garantir que o robô esteja em translação durante o processo de aquisição das imagens, para assegurar que os vetores de fluxo óptico obtidos sejam precisos na caracterização do movimento relativo entre o robô e os objetos, que é feita por meio do cálculo do tempo para contato. Durante o giro o sistema de sensoriamento visual é mantido inativo e o laço interno de controle é responsável por comandar o giro.

Com o sistema de controle proposto, as informações do sistema de sensoriamento apresentado no Capítulo 3 são utilizadas para imprimir no robô o comportamento reativo vagar, que é a camada de controle de mais baixo nível em um sistema de controle baseado em comportamentos, e que é responsável pela segurança do robô fazendo com que ele navegue desviando-se dos objetos presentes no ambiente, evitando colisões.

Capítulo 5- A Implementação a Bordo do Robô e os Resultados Obtidos

O sistema de sensoriamento desenvolvido foi utilizado para controlar o robô móvel Pioneer 2DX, o qual possui um computador Pentium MMX 233MHz a bordo e um microcontrolador da Siemens conectado em sua porta serial. O programa Saphira, que acompanha o robô, e é executado no computador de bordo, proporciona um canal de comunicação com o programa do microcontrolador, através da porta serial. Usando a interface do programa Saphira, é possível enviar dados ao robô, para controlar seus movimentos e as variáveis da câmara de vídeo. O robô possui, também, uma placa de digitalização de imagens (*frame-grabber*) da Imagenation. O acesso a esta placa é feito usando o programa PXC, que possui todas as funções necessárias para a aquisição de imagens. A Figura 5.1 apresenta de maneira esquemática o sistema disponível no robô.

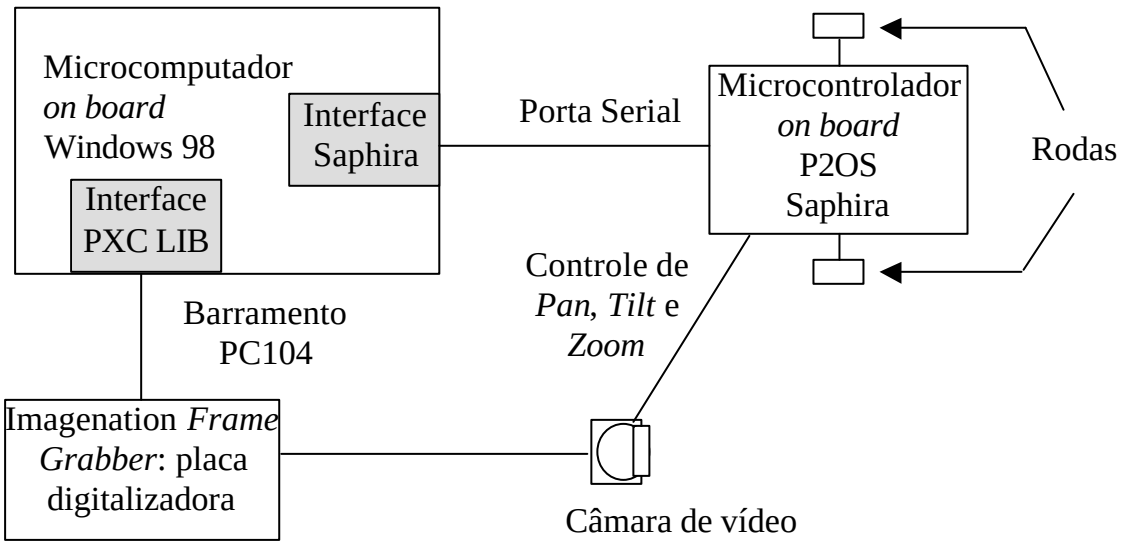


Figura 5.1- A estrutura de *hardware* disponível no robô usado nos experimentos.

Todo o sistema de controle foi implementado em linguagem C++, usando as classes do MFC (*Microsoft Foundation Class*) disponíveis no ambiente do Visual C++. Este programa, assim como todos os outros que acompanham o robô, é executado no sistema operacional Windows 98.

O programa principal apresenta três funções básicas: *StartProcess*, *StartMotors* e *StopProcess*. A função *StartProcess* é responsável pela criação das *threads* que vão fazer o processamento das imagens e enviar comandos ao robô, para controlar a sua navegação. A função *StartMotors* é responsável, exclusivamente, por habilitar o funcionamento dos motores do robô por *software*. Os motores podem, também, ser habilitados pressionando-se um botão (o branco) no console do robô. Por motivos de segurança, esta função fica disponível apenas quando as *threads* de comando e de cálculo estão ativas. Executando a função *StopProcess* o usuário pode interromper o ciclo de controle, porque esta função torna verdadeira uma variável de saída que é testada pelo sistema.

A função *StartProcess* executa, primeiramente, um procedimento que inicia a interface Saphira. Este programa, como foi mencionado anteriormente, é responsável pela comunicação com o microcontrolador do robô, e trabalha em uma *thread* separada do programa principal. Depois, usando as funções do Saphira, é feita a conexão com o microcontrolador através da porta serial do PC, e são realizadas todas as configurações necessárias para colocar o robô em movimento. Entre as variáveis que precisam ser configuradas, estão os valores de velocidade linear e angular, além dos valores máximos que essas velocidades podem assumir, para garantir a segurança do robô. A seguir, são iniciadas duas *threads* criadas neste trabalho: a primeira é responsável pela parte de processamento das imagens para extração de características, e a segunda é responsável pelo envio de comandos ao robô, usando as funções do programa Saphira. As duas *threads* se comunicam usando eventos, e, além disto, possuem uma área de parâmetros comum, que pode ser usada para transferência de dados entre si e para o programa principal. A relação entre as diversas *threads* é apresentada de maneira esquemática na Figura 5.2.

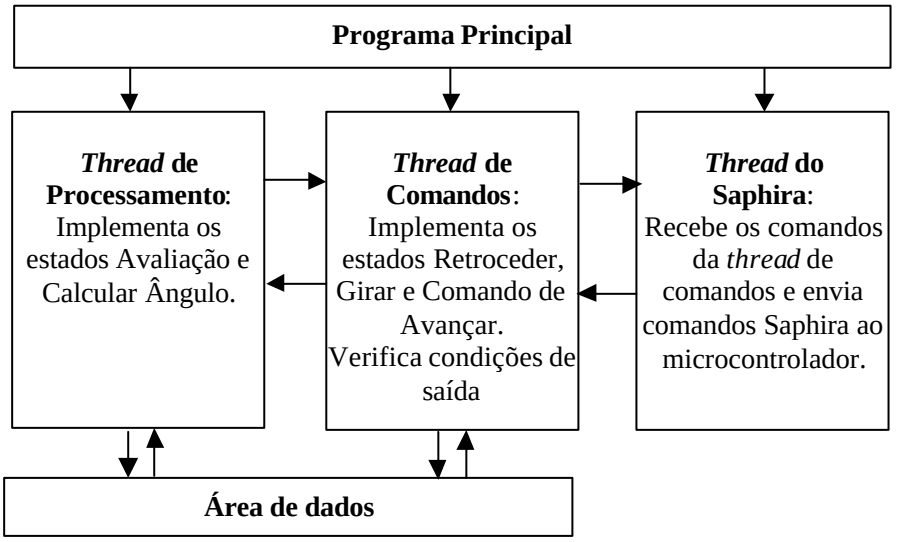


Figura 5.2- As *threads* do programa.

A estrutura utilizada para passagem de parâmetros fornece para a *thread* de processamento acesso à classe *CFrameGrabber*, que encapsula as funções do programa PXC.

Esta classe é instanciada na criação da aplicação, e possui todos os métodos necessários para configurar as imagens a serem adquiridas, em termos de resolução, brilho, contraste, cores, etc., e para fazer a aquisição das imagens.

5.1 A *Thread* de Processamento

Nesta *thread* são implementados os estados relacionados ao processamento de imagens da máquina de estados da Figura 4.1, ou seja, o estado de Avaliação, onde as imagens são processadas, e o estado Calcular Ângulo, onde é calculado o ângulo que o robô deve girar. Além disso, de acordo com os resultados obtidos nos cálculos executados nestes estados, são comandadas as devidas transições de estado. O corpo da *thread* de processamento consiste em um *loop* que pode ser interrompido pelo usuário, através da função *StopProcess*, ou quando é detectada alguma situação de emergência. Cada vez que o sistema inicia um ciclo do loop, duas imagens são adquiridas e processadas. De acordo com o resultado obtido, o procedimento pode ser calcular um novo ângulo de giro e chamar a *thread* de comandos para enviar ao robô, ou, então, pode ser necessário chamar a *thread* de comandos, diretamente, para executar um comando que retire o robô de uma situação de perigo. Cada vez que um comando deve ser enviado ao robô, esta *thread* dispara um evento Comando que aciona a *thread* de comandos. A *thread* de processamento entra, então, em um estado de espera que somente é interrompido quando ocorre um evento Continue. Este evento é disparado pela *thread* de comandos em duas situações: quando verifica que o robô finalizou a execução do comando, e está novamente em movimento de translação, a fim de liberar a *thread* de processamento para iniciar um novo ciclo de processamento; ou quando a *thread* de comandos é interrompida por alguma condição de saída, para sinalizar à *thread* de processamento que proceda a liberação de memória e finalize a *thread*.

A Figura 5.3 apresenta o algoritmo implementado nesta *thread* em pseudo-código.

Por questões de eficiência e clareza, os cálculos necessários nesta *thread* são implementados em uma classe à parte, chamada *CCalcParam*. A *thread* de processamento instancia essa classe em sua parte inicial chamando seu construtor, assim como chama seu destrutor quando vai finalizar. O construtor da classe *CCalcParam* recebe um apontador para o *CFrameGrabber* (instância da classe responsável pelo acesso à placa digitalizadora e, portanto, capaz de fazer a aquisição de imagens) e o tamanho da imagem e, com esses dados, faz a alocação de memória para armazenar os dados e os resultados dos cálculos, evitando, assim, o alto custo de alocar memória dinamicamente no decorrer do programa. O destrutor é responsável por liberar a memória alocada inicialmente. A classe implementa dois métodos, correspondentes aos cálculos necessários nos estados de Avaliação e Calcular Ângulo.

Os capítulos anteriores apresentaram os processos desenvolvidos neste trabalho para extrair informações do sensoriamento visual. Porém, na implementação a bordo do robô, algumas rotinas implementadas apresentaram um aumento no tempo de cálculo devido às diferenças entre o sistema a bordo do robô e o sistema utilizado preliminarmente. Um fator que contribui para isto é a diferença de desempenho do computador utilizado fora do robô, um Pentium II 400MHz, e o computador de bordo, um Pentium MMX 233MHz. Outro fator é a placa de aquisição de imagens do robô, Imagenation, que possui características diferentes da placa Data Translation utilizada nos experimentos iniciais. Além disto, o sistema embarcado funciona com Windows 98, enquanto o sistema usado externamente funciona com Windows NT4. Considerando estas diferenças e buscando alcançar uma redução no tempo total de

Thread de Processamento - Algoritmo

```
Alocar variáveis;
Fazer FIM igual a FALSO;
Esperar o evento Continue;
Enquanto (FIM igual a FALSO) faça
    Adquirir duas imagens;
    Processar as imagens;
    Destacar os obstáculos detectados;
    Se objeto ou parede à frente então
           Fazer AÇÃO igual a RETORNAR;
      Senão se objeto na lateral direita então
           Fazer ÂNGULO igual a –15;
           Fazer AÇÃO igual a GIRAR;
      Senão se objeto na lateral direita então
           Fazer ÂNGULO igual a +15;
           Fazer AÇÃO igual a GIRAR;
      Senão
           Calcular ângulo de giro;
           Fazer ÂNGULO igual ao ângulo calculado;
           Fazer AÇÃO igual a GIRAR;
    Fim;
    Disparar o evento Comando;
    Esperar o evento Continue;
Fim;
Desalocar Variáveis;
Enviar mensagem ao programa principal;
Sair.
```

Figura 5.3- Descrição da *thread* de processamento em pseudo-código.

cálculo, algumas modificações foram efetuadas, na implementação a bordo. Estas modificações se baseiam, especialmente, na sub-amostragem de imagens e na busca de algoritmos mais eficientes para executar alguns dos processos de cálculo. A seguir, são apresentados alguns detalhes de implementação de cada parte do processamento efetuado, mostrando as alterações realizadas. Vale ressaltar que cada modificação foi feita com o cuidado de não comprometer a qualidade dos resultados. Assim, os resultados obtidos antes e depois de cada modificação foram comparados, mantendo-se apenas as alterações que não comprometeram o desempenho do sistema.

5.1.1 A Aquisição de Imagens

O programa implementado inicialmente para comparar os algoritmos de cálculo de fluxo óptico fazia a aquisição das imagens utilizando a biblioteca Frame Grabber SDK, correspondente à placa Data Translation. As funções desta biblioteca foram encapsuladas em uma classe, sendo disponibilizadas, assim, para o programa.

A aquisição de imagens a bordo do robô Pioneer 2DX levou à necessidade de adaptação do programa para o sistema de aquisição presente a bordo, consistindo da placa Imagenation com interface pela biblioteca PXC. Esta adaptação consistiu em criar uma classe base e duas classes derivadas, uma para cada sistema de aquisição, de forma que fosse transparente para o usuário qual o sistema de aquisição utilizado e que o programa fosse portável entre os dois sistemas.

O sistema no computador externo fornece imagens em formato colorido RGB, com tamanho máximo de 640×480 *pixels*, mesmo que a câmara utilizada seja monocromática. Por isso, as imagens devem ser adquiridas neste formato e convertidas para níveis de cinza para serem, posteriormente, utilizadas no cálculo do fluxo óptico. Por outro lado, a placa de aquisição Imagenation disponível no robô pode fornecer imagens diretamente em níveis de cinza, com 8 bits por *pixel*. Ainda assim, para manter a portabilidade, o mesmo procedimento era adotado em ambos os sistemas.

Desta maneira, uma das modificações realizadas a bordo, quando o sistema completo foi implementado, consistiu na modificação de algumas estruturas de dados e de algumas rotinas, retirando a portabilidade do programa, tornando, porém, mais eficiente a interface com a placa de aquisição de imagens presente no robô.

Uma segunda modificação realizada a bordo do robô, com relação à aquisição de imagens, consistiu na redução do tamanho das mesmas. Enquanto o sistema testado fora do robô trabalha com imagens de 640×480 *pixels*, as imagens adquiridas a bordo do robô consistem em imagens de 320×240 *pixels* obtidas com 8 bits por *pixel*. Essas imagens são obtidas configurando-se o programa de aquisição para tomar apenas um dos campos (o par) na saída de vídeo composto, e reduzindo pela metade a resolução horizontal. Isto foi feito por três motivos: primeiro, porque o número de *pixels* a serem processados fica reduzido a um quarto, embora a imagem continue dando informação sobre a mesma porção do ambiente; segundo, para reduzir os erros nas estimativas do fluxo óptico, que em parte são gerados pela diferença entre os tempos de varredura das linhas pares e ímpares na imagem; e, terceiro, porque o tempo de aquisição de uma imagem com apenas um dos campos, ou seja, com metade das linhas, é menor que o tempo gasto para adquirir uma imagem com os dois campos entrelaçados.

5.1.2 O Cálculo do Fluxo Óptico

No Capítulo 2, o algoritmo de Mínimos Quadrados Modificado [45], proposto neste trabalho, foi selecionado para executar o cálculo de fluxo óptico a bordo do robô. Para reduzir o tempo de cálculo da estimativa do fluxo óptico por este método, a implementação a bordo foi feita usando a biblioteca OpenCV da Intel, a qual explora as capacidades do processador Pentium MMX, presente no robô, para acelerar os cálculos matriciais. Desta maneira, o cálculo do fluxo óptico passou a ser executado em um tempo menor, sem qualquer alteração na qualidade da estimativa obtida.

Além disto, no Capítulo 2 os testes foram feitos usando imagens de 640×480 *pixels*, digitalizadas a partir de um sinal de vídeo composto, dividindo-se a imagem em regiões de 20×20 *pixels*. Porém, como foi dito na seção anterior, as imagens utilizadas na implementação a bordo são de 320×240 *pixels*. Assim, para manter as regiões representando objetos do mesmo tamanho, a imagem foi dividida em regiões de 10×10 *pixels*.

Uma outra modificação realizada, em conseqüência da redução das dimensões da imagem e das regiões, foi uma diminuição no número, assim como uma modificação na forma de obtenção, dos *pixels* de observação usados no cálculo da estimativa do fluxo óptico em cada região. Nos experimentos apresentados no Capítulo 2, o número de observações feitas em cada região de 20×20 *pixels* era de 50 *pixels* tomados aleatoriamente. Nas primeiras versões do programa a bordo do robô, utilizando ainda imagens em 640×480 *pixels*, os pontos de observação foram dispostos em um espaçamento fixo dentro da região. Assim, foi possível eliminar o tempo gasto pelo gerador aleatório, diminuindo o tempo total de cálculo. Os pontos de observação foram reduzidos para 48 e escolhidos de forma a abranger o máximo possível a região, tomando, também, o cuidado de evitar problemas de contorno. Assim, foram escolhidos *pixels* internos a cada região e, conseqüentemente, internos à imagem, garantindo que fosse sempre possível calcular as derivadas do brilho no *pixel* utilizando diferenças finitas, como foi mostrado no Capítulo 2. Na implementação final a bordo do robô, utilizando imagens de 320×240 *pixels*, para cada região de 10×10 *pixels*, são tomados 12 pontos de observação distribuídos de forma fixa dentro da região, como mostrado na Figura 5.4.

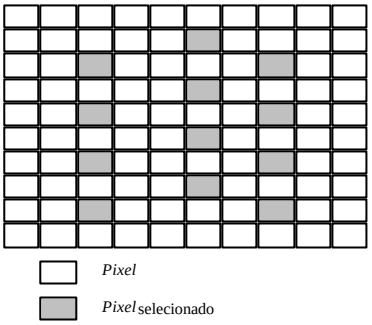


Figura 5.4 – *Pixels* selecionados como pontos de observação em cada região de 10 ´ 10 *pixels*.

Portanto, na implementação a bordo do robô, as derivadas do brilho da imagem, I_x, I_y e I_t, são calculadas para cada um dos *pixels* mostrados na Figura 5.4 e são usadas para montar a matriz **A** e o vetor **b**, como mostrado na descrição do algoritmo de Mínimos Quadrados Modificado no Capítulo 2. Sendo que as matrizes usadas na implementação são estruturas de dados da biblioteca OpenCV da Intel. Usando as estruturas de dados e as funções disponíveis nesta biblioteca é possível multiplicar, transpor, inverter matrizes e, assim, determinar os vetores de fluxo óptico para cada região, resolvendo de maneira rápida o sistema linear obtido.

5.1.3 O Algoritmo de Segmentação Baseado no Fluxo Óptico

O algoritmo que realiza a segmentação das imagens com base no fluxo óptico foi apresentado no Capítulo 3. A implementação inicial deste algoritmo, feita no computador externo, foi realizada usando vizinhança oito. Assim, o vetor de fluxo óptico em cada região era comparado com os vetores nas oito regiões contíguas (Figura 3.2). Porém, uma segunda implementação foi feita, onde o vetor de fluxo óptico em uma região é comparado apenas aos vetores dos quatro vizinhos mostrados na Figura 5.5. As seqüências de imagens para as quais foram realizados testes da implementação usando vizinhança oito, foram utilizados para testar a segunda implementação. Destes testes, verificou-se que os resultados obtidos utilizando-se vizinhança quatro eram suficientemente precisos. Por outro lado, a modificação de vizinhança oito para vizinhança quatro se traduz em uma redução significativa do tempo de processamento, devido principalmente ao cálculo das diferenças entre os vetores de fluxo óptico (equação 3.1), durante os testes de comparação realizados no algoritmo.

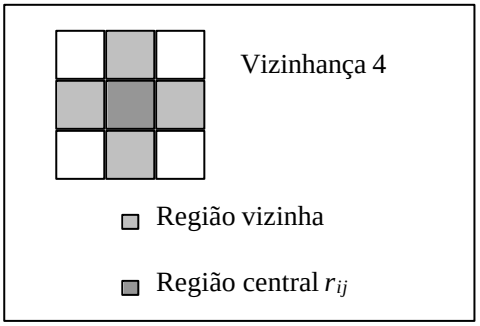


Figura 5.5 – Região e vizinhança usando conectividade-quatro.

Além disto, na implementação do algoritmo de segmentação realizada no computador a bordo do robô, não foi utilizada nenhuma medida de confiabilidade do fluxo óptico. Embora a

medida apresentada no Capítulo 2 tenha se mostrado útil nos testes realizados com objetos maiores, onde alguma textura havia sido adicionada, nos testes realizados com o robô em seu ambiente de trabalho não foi possível utilizá-la, porque ela causa uma perda muito grande de informação, tornando a matriz de fluxo óptico muito esparsa.

Da mesma maneira que foi mostrado no Capítulo 3, aquelas regiões para as quais não é possível determinar o vetor de fluxo óptico são eliminadas do processo de segmentação, permanecendo com marcador -1. Além disto, os objetos compostos de apenas uma região representam, em geral, situações de erro na estimativa do fluxo óptico e por isso são descartados de todos os processamentos realizados com base no resultado da segmentação.

5.1.4 A Determinação do Foco de Expansão (FOE)

Como foi discutido no Capítulo 2, se o robô estiver em movimento de aproximação da cena, o fluxo óptico medido deve ser divergente, ou seja, os vetores de fluxo óptico devem ser radiais, partindo do foco de expansão (FOE). Neste caso, é possível determinar as coordenadas do foco de expansão (x_{FOE}, y_{FOE}), a partir dos vetores de fluxo óptico estimados.

No procedimento implementado a bordo do robô, a coordenada x_{FOE} é determinada em dois passos. Primeiro, verificando-se, linha por linha de regiões, em que coluna da imagem os vetores de fluxo óptico passam de um ângulo maior a um menor que 90^o, ou vice-versa, ou de um ângulo maior a um menor que 270^o, ou vice-versa. A seguir, o valor de x_{FOE} é obtido como a média dos valores de coluna encontrados para cada linha. De maneira similar, o valor de y_{FOE} é estimado verificando-se, coluna por coluna de regiões, em que linha da imagem os vetores de fluxo óptico passam de um ângulo maior a um menor que 0^o, ou vice-versa, ou de um ângulo maior a um menor que 180^o, ou vice-versa. Finalmente, o valor de y_{FOE} é calculado como a média dos valores obtidos para cada coluna.

Assim, o mesmo procedimento utilizado no Capítulo 2 foi implementado a bordo. A única diferença está no fato de que, naqueles experimentos, todos os vetores de fluxo óptico disponíveis eram utilizados na determinação do FOE, enquanto na implementação a bordo do robô os vetores de fluxo óptico representando regiões que, durante o processo de segmentação, ficaram isoladas, ou seja, que pertencem a um objeto que tem apenas uma região, foram eliminados do processo. Isto porque foi observado, durante os experimentos, que a maior parte dos objetos reais no ambiente do robô são segmentados em mais de uma região, e que, em geral, objetos com apenas uma região correspondem a algum erro na estimativa do fluxo óptico.

5.1.5 A Determinação dos Tempos para Contato por Objeto

Tendo obtido os vetores de fluxo óptico, realizado o processo de segmentação e determinado o foco de expansão, é possível calcular o tempo para contato a cada região da imagem e, a partir destes valores, obter um único valor de tempo para contato para cada objeto resultante do processo de segmentação. Assim, na implementação a bordo do robô, o tempo para contato foi calculado para cada região da imagem usando a equação 2.43, ou seja, a mesma utilizada nos outros capítulos desta tese.

Neste ponto é importante observar que existem várias possibilidades para se obter um único valor representando todo o objeto a partir dos valores calculados para cada região

pertencente a ele. Assim, é necessário estabelecer um critério para escolher uma destas formas.

A maneira mais conservativa seria escolher o menor valor de tempo para contato entre todas as regiões pertencentes ao objeto. Porém, nos experimentos realizados, verificou-se que os valores de tempo para contato obtidos pelo sistema eram, em geral, subestimados, ou seja, menores que o valor real. Assim, nos testes realizados utilizando-se o valor mínimo, o sistema comandava o robô a executar giros de ângulos muito grandes. Desta maneira, o robô apresentava reações muito bruscas em situações onde não havia perigo real, em muitos casos, girando repetitivamente sem conseguir se deslocar significativamente no ambiente.

Por este motivo, no Capítulo 3 foi utilizado o valor mediano dos tempos para contato das regiões pertencentes ao objeto, onde se tentou encontrar uma medida menos suscetível aos erros nos valores mínimos e máximos. Assim, os valores de tempo para contato calculados para cada região eram armazenados em um vetor. Estes valores eram classificados usando um algoritmo bastante rápido resultante da adaptação do algoritmo *quick sort*, onde o processo de classificação é interrompido quando se obtém o valor que ocupa a posição do meio do vetor, ou seja, o valor mediano. Nos testes realizados usando o valor mediano dos tempos para contato das regiões, os resultados foram satisfatórios. Porém, ainda que utilizando um algoritmo eficiente, o custo de obtenção deste valor é significativo.

Sendo assim, na implementação a bordo do robô, o valor de tempo para contato a cada objeto foi obtido através do valor médio dos tempos para contato das regiões pertencentes a ele. Desta maneira, não é necessário armazenar nem classificar os valores de tempo para contato das regiões. À medida que estes valores são calculados, eles vão sendo adicionados, e ao fim é calculado o valor médio. Os experimentos realizados utilizando o valor médio mostraram que, apesar da influência dos valores mínimos e máximos, os valores de tempo obtidos levam o robô à reação desejada.

5.1.6 A Determinação do Diagrama de Tempos para Contato

Procurando em cada coluna de regiões pelo objeto com menor tempo para contato, é possível determinar quais são os objetos mais próximos em cada direção do campo visual do robô. O resultado deste processo de varredura por colunas de regiões pode ser representado pelo diagrama de tempos para contato, apresentado no Capítulo 3.

Neste processo de obtenção do diagrama são considerados os valores de tempo médio, calculados para cada objeto (Seção 5.1.5), e não os tempos para contato calculados para cada uma das regiões. Além disso, os objetos com apenas uma região não são considerados, por serem, em geral, resultado de vetores de fluxo óptico com erro de estimativa.

Como o valor de fluxo óptico esperado nas regiões próximas ao foco de expansão é próximo a zero, o tempo para contato nestas regiões é mais sensível a ruído [20]. Assim, os objetos pequenos e muito próximos ao FOE foram eliminados do processo de obtenção do diagrama. Desta maneira, foram tomados os objetos com menos de quatro regiões, e verificou-se se pelo menos uma de suas regiões tinha o centro localizado a uma distância inferior a 30 *pixels* do FOE. Os objetos maiores, mesmo que estejam próximos ao foco de expansão, são considerados, porque mantêm uma consistência do fluxo óptico em suas regiões, incluindo as regiões com vetores de fluxo óptico de menor módulo. Ou seja, as

regiões onde o fluxo óptico apresenta maior módulo garantem a qualidade dos vetores de menor magnitude.

5.2 A *Thread* de Comandos

A *thread* de comandos é responsável por verificar as condições de saída do programa e por comandar o robô na execução das tarefas da navegação, implementando os estados dentro da área pontilhada na máquina de estados da Figura 4.1.

Esta *thread* possui em sua implementação uma parte de inicialização, que espera a liberação dos motores do robô para movimentos, disparando a *thread* de processamento pela primeira vez assim que o robô começa a avançar. É possível saber se os motores estão habilitados monitorando uma variável de *status* dos motores, ou os dados de posição, direção e velocidades no refletor de estados do Saphira. Na verdade, essa é uma leitura indireta, sendo necessário, antes, comandar o movimento dos motores e depois monitorar se o robô está se movendo. A habilitação dos motores pode ser feita pressionando um botão (o branco) no console do robô ou por *software*, usando uma função do Saphira, que neste programa foi utilizada para implementar a função *StartMotors* do programa principal.

Depois o sistema entra em um *loop* que espera por um evento Comando que, da primeira vez, é disparado pela *thread* de processamento. Depois, porém, tal evento pode ser disparado pela *thread* de processamento ou pela própria *thread* de comandos.

Cada vez que este evento ocorre, são avaliadas as condições de saída do programa, as quais são apresentadas a seguir:

- A função *StopProcess* no programa principal foi executada pelo usuário. Como foi dito no início deste capítulo, esta função altera uma variável na área de parâmetros para sinalizar que o usuário deseja encerrar a aplicação. A *thread* de comandos verifica esta variável antes de começar a execução dos comandos correspondentes ao estado atual do sistema.
- O programa Saphira foi desconectado do robô pelo usuário. O programa Saphira, que é colocado em execução na função *StartProcess*, permite que o usuário interfira no processo usando a sua janela de interface. Assim, o usuário pode desconectar o robô, interrompendo o ciclo de comunicação com ele, e impedindo, portanto, o envio de comandos. Para saber se isto aconteceu, o programa deve testar a função *sfIsConnected* do Saphira.
- O programa Saphira foi desligado pelo usuário. Com foi dito no item anterior, o usuário pode interferir no processo, inclusive fechando a janela do programa Saphira. Esta ação interrompe a conexão com o robô, impedindo seu controle pelo sistema. O programa deve testar, neste caso, a função *sfIsExited*.
- Alguma das rodas do robô está impedida de se mover. Esta situação acontece quando algum obstáculo impede que alguma de suas rodas execute a velocidade que está sendo comandada. Ao detectar esta situação o microcontrolador muda o *status* das rodas no Saphira. Para testar essa condição de impedimento a função *sfStalledMotor* deve ser testada para cada uma das rodas, ou seja, usando *sfLEFT* e *sfRIGHT* como parâmetros.

Se alguma das condições acima ocorre, a *thread* de comandos procede com todos os passos necessários para sua própria finalização. Além disso, ela é responsável por avisar à *thread* de processamento para terminar sua execução. Para isto ela torna verdadeira uma

variável de cancelamento na área de parâmetros e dispara um novo evento Continue. Se nenhuma condição de saída ocorre, o sistema continua a execução normalmente, comandando a próxima ação do robô. As ações possíveis neste sistema são: avançar, retroceder e girar, correspondendo aos estados entre linhas pontilhadas da máquina de estados da Figura 4.1. A Figura 5.6 mostra o algoritmo executado nesta *thread*, em pseudo-código.

Thread de Comandos – Algoritmo

```
Alocar as variáveis necessárias;
Esperar o robô iniciar movimento;
Disparar evento Continue;
Fazer SAIR igual a FALSO;
Esperar evento Comando;
Enquanto (SAIR igual a FALSO) faça
    Se alguma condição de saída é verdadeira então
            Fazer FIM igual a VERDADEIRO;
            Fazer SAIR igual a VERDADEIRO;
            Disparar evento Continue;
    Senão
            Se ação igual a AVANÇAR então
                    Acionar Saphira para comandar velocidade de 100mm/s;
                    Esperar o robô começar a avançar;
            Senão se ação igual a GIRAR
                    Ler ÂNGULO;
                    Acionar Saphira para comandar giro deste ângulo;
                    Esperar o robô terminar de girar;
                    Fazer ação igual a AVANÇAR;
                    Disparar o evento Comando;
            Senão se ação igual a RETROCEDER
                    Acionar o Saphira para comandar o robô para retroceder 10cm;
                    Esperar execução do comando pelo robô;
                    Fazer ÂNGULO igual a 180;
                    Fazer ação igual a GIRAR;
                    Disparar evento Comando;
                    Fim;
    Fim;
    Esperar o evento Comando;
Fim;
Desalocar todas as variáveis;
Sair.
```

Figura 5.6- O algoritmo da *thread* de comandos em pseudo-código.

As ações são executadas enviando-se comandos para a interface do programa Saphira, que está sendo executada em uma *thread* separada. O Saphira está conectado ao microcontrolador Siemens via porta serial, enviando e recebendo pacotes de dados a cada 100ms. Em mais baixo nível, um comando do Saphira escreve em uma área de memória os valores de velocidade linear, velocidade angular, de ângulo de giro e de distância. Estes valores são enviados ao robô e servem como *setpoints* para o controlador das rodas, que está sendo executado no microcontrolador. A cada 100ms o Saphira recebe dados do microcontrolador sobre a posição e a orientação do robô, e sobre suas velocidades linear e angular. Como as ações comandadas ao robô usando as funções do Saphira seguem este protocolo de comunicação, a única maneira de garantir que um determinado comando foi recebido e executado pelo robô é verificando as informações enviadas pelo microcontrolador ao Saphira. Portanto, os comandos não são realizados no momento em que são disparados pela *thread* de comandos, o que torna lento o processo de execução.

A *thread* de comandos é chamada para executar as ações de acordo com as situações encontradas na *thread* de processamento, durante o estado de avaliação. As seqüências de ações são basicamente três:

- retornar: o objetivo, neste caso, é que o robô retorne na direção por onde chegou. Assim, seguindo a máquina de estados da Figura 4.1, o sistema entra no estado Retroceder de

onde vai para o estado Girar e depois para o estado Comando de Avançar. No primeiro passo o robô é comandado a retroceder de uma determinada distância (no caso 10 cm). Isto é necessário porque o robô tem o centro de giro posicionado mais à frente em sua plataforma, e não no centro geométrico. Assim, o giro de 180^o do passo seguinte causaria uma colisão da parte traseira com o objeto detectado à frente do robô. A opção por girar 180^o parte do pressuposto de que o robô pode voltar pelo mesmo caminho por onde veio. O terceiro passo visa preparar o robô para o novo passo de avaliação, onde o sistema supõe que o robô está em translação;

- girar: essa ação consiste em dois passos, girar de um determinado ângulo e voltar a avançar, correspondendo aos estados Girar e Comando de Avançar na Figura 4.1. Essa ação pode ser requisitada quando um objeto lateral é detectado, ou depois que uma nova direção segura para o robô é calculada no estado Calcular Ângulo;
- avançar: essa ação é a mais simples e consiste em colocar o robô em movimento de translação, fornecendo ao microcontrolador a velocidade linear desejada e eliminando qualquer comando de giro ou velocidade angular impressa anteriormente ao robô. Na máquina de estados da Figura 4.1, esta ação corresponde ao estado Comando de Avançar.

Como o sistema considera que o robô está em translação durante a aquisição de imagens, é necessário que os comandos de giro sejam executados completamente antes de disparar um novo evento Continue, acionando a *thread* de processamento. Este é o motivo pelo qual a *thread* de comando tem que esperar que os comandos sejam executados até o fim, e porque, depois de um giro, o robô é sempre comandado a avançar. Além disso, depois de um giro grande, como o de 180^o, é comum que as imagens da câmara fiquem fora de foco. Assim, é necessário permitir um tempo de acomodação ao sistema de foco automático da câmara. As mudanças de estado entre Retroceder, Girar e Comando de Avançar são coordenadas pela própria *thread* de comando, que só dispara um evento Continue, chamando a *thread* de processamento, após ter finalizado a ação, recolocando o robô em movimento de translação.

5.3 Resultados Experimentais

A seguir são apresentados os resultados obtidos em alguns experimentos realizados no laboratório, utilizando o sistema proposto:

5.3.1 Primeiro Experimento de Navegação

Neste primeiro experimento, o robô pode girar para desviar-se dos obstáculos detectados pelo sistema de sensoriamento proposto neste trabalho, de acordo com o sistema de controle implementado. O ciclo do sistema é interrompido pelo usuário depois que o robô executa o primeiro giro de 180^o, completando, assim, um experimento bastante simples.

A Figura 5.7 mostra a trajetória obtida pela odometria do robô, enquanto a Figura 5.8 mostra a primeira imagem das seqüências adquiridas pela câmara a bordo do robô, nas posições indicadas na Figura 5.7 pelas letras (a), (b), (c) e (d).

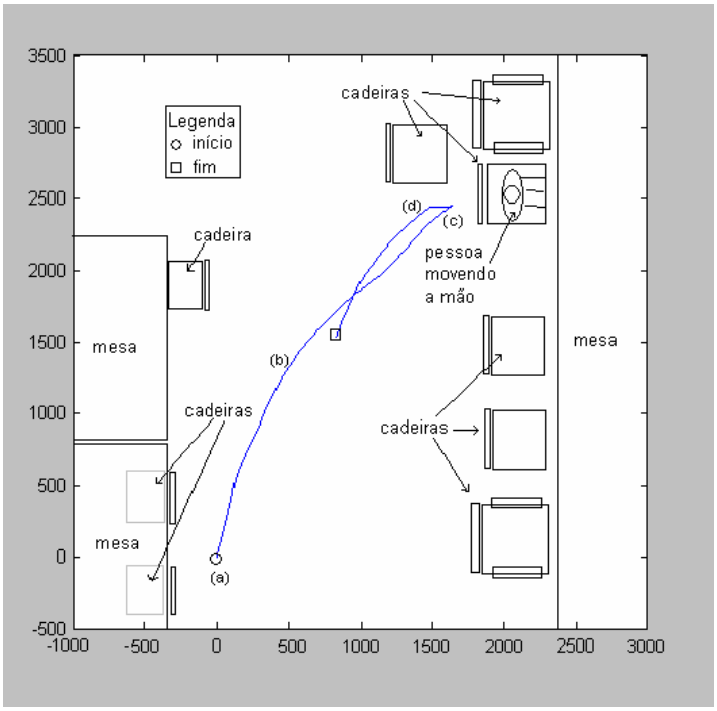


Figura 5.7- Trajetória do robô durante o primeiro experimento de navegação.

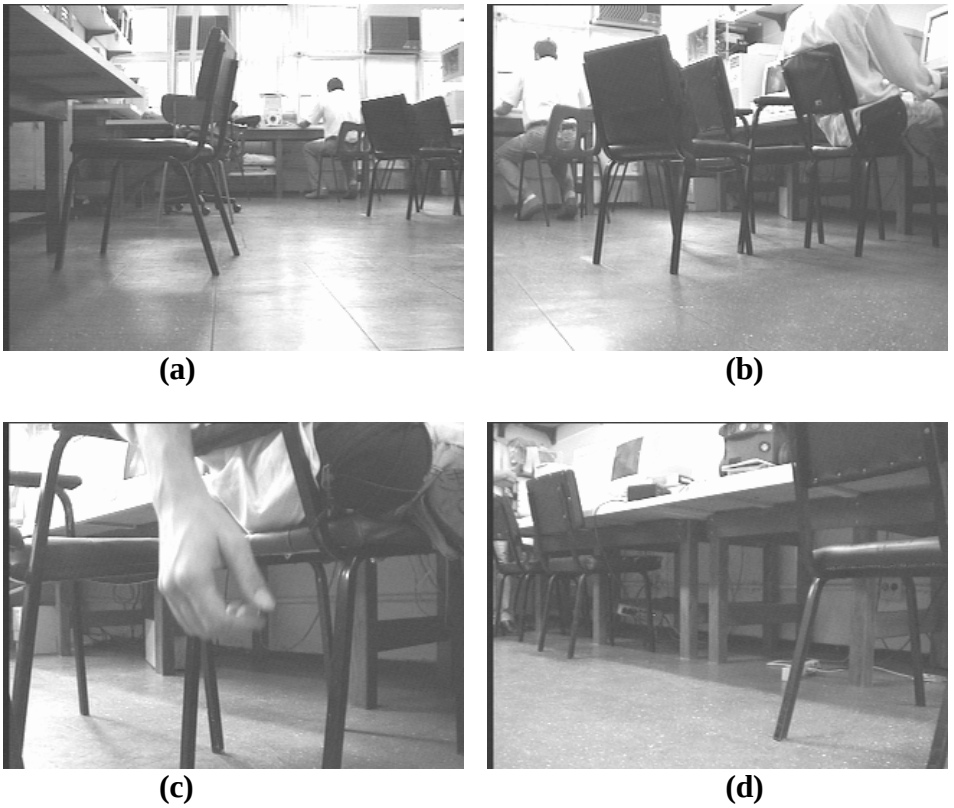


Figura 5.8 – Primeiro experimento de navegação: imagens adquiridas pela câmara do robô nas posições indicadas pelas letras (a), (b), (c) e (d) na Figura 5.7.

5.3.2 Segundo Experimento de Navegação

Neste experimento, o robô navega por um tempo maior, tendo a oportunidade de circular mais pelo laboratório, e desviar-se de diversos tipos de obstáculos, como pés de cadeira e de mesa, paredes e pessoas. A Figura 5.9 mostra a trajetória executada dentro do laboratório, a qual é obtida pela odometria do robô.

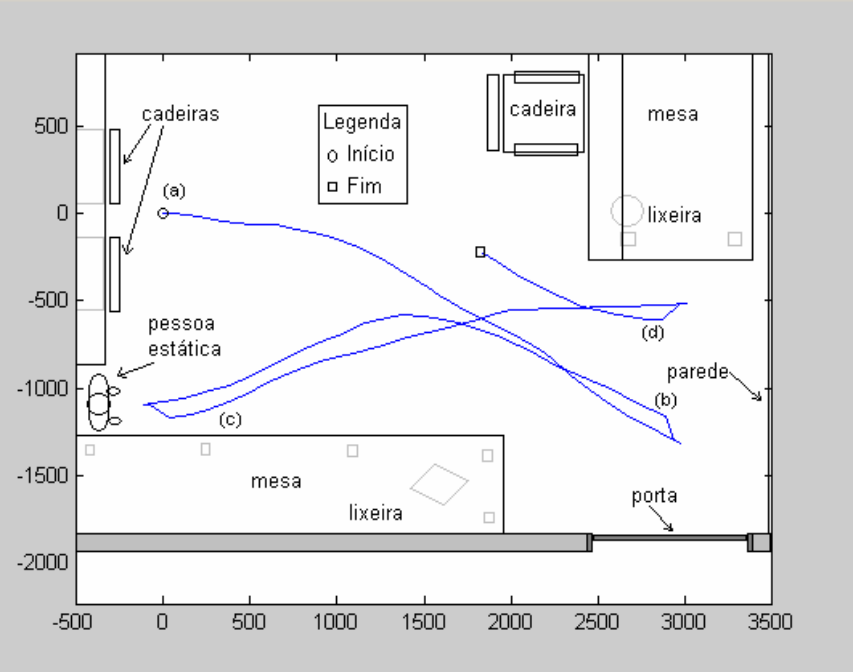


Figura 5.9 - Trajetória do robô durante o segundo experimento de navegação.

Como se pode ver na Figura 5.9, o robô faz um trajeto um pouco mais complicado, desviando-se dos objetos com suavidade, quando possível, e retornando quando algum objeto muito próximo é detectado.

A Figura 5.10 apresenta alguns instantes do percurso através de imagens adquiridas pela câmara do robô nas posições indicadas pelas letras de (a) a (d) na Figura 5.9. Neste experimento, assim como no anterior, pode-se perceber a eficiência do sistema implementado na condução do robô, principalmente considerando que não há qualquer preparação do ambiente.

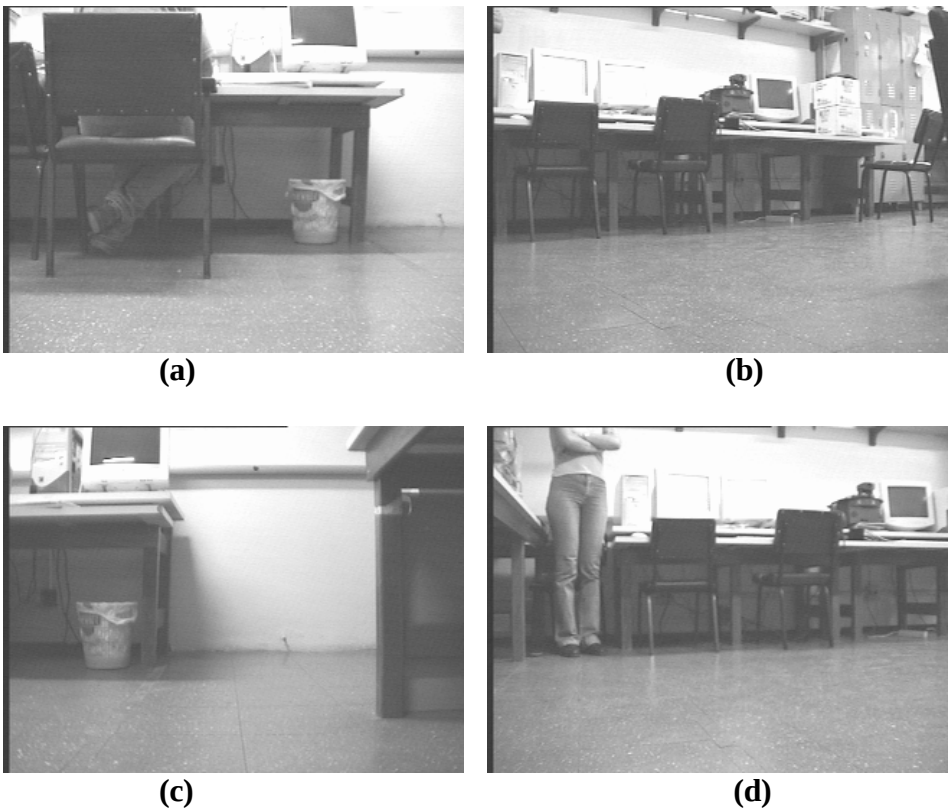


Figura 5.10 – Segundo experimento de navegação: imagens adquiridas pela câmara do robô nas posições indicadas pelas letras (a), (b), (c) e (d) na Figura 5.9

5.3.3 Terceiro Experimento de Navegação

Este experimento mostra a habilidade do sistema para detectar o movimento de pessoas no ambiente. A Figura 5.11 mostra a trajetória obtida pela odometria do robô.

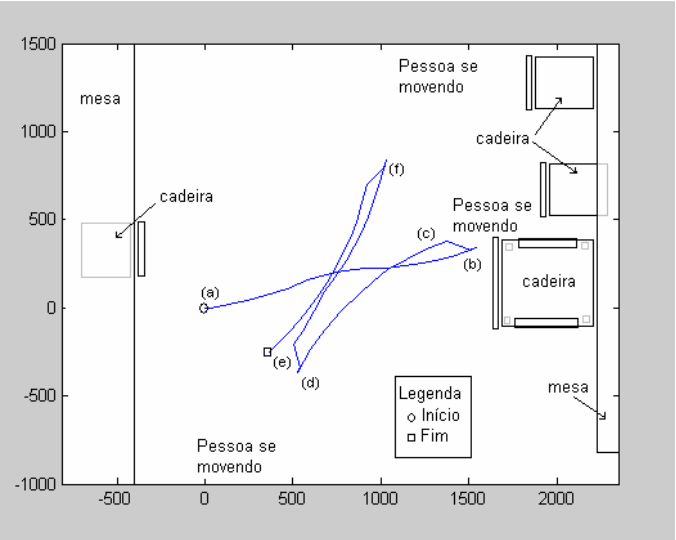


Figura 5.11- Trajetória do robô durante o terceiro experimento.

A Figura 5.12 apresenta o primeiro quadro de imagem das seqüências adquiridas pela câmara de vídeo do robô nas posições indicadas pelas letras de (a) a (f) na Figura 5.11.

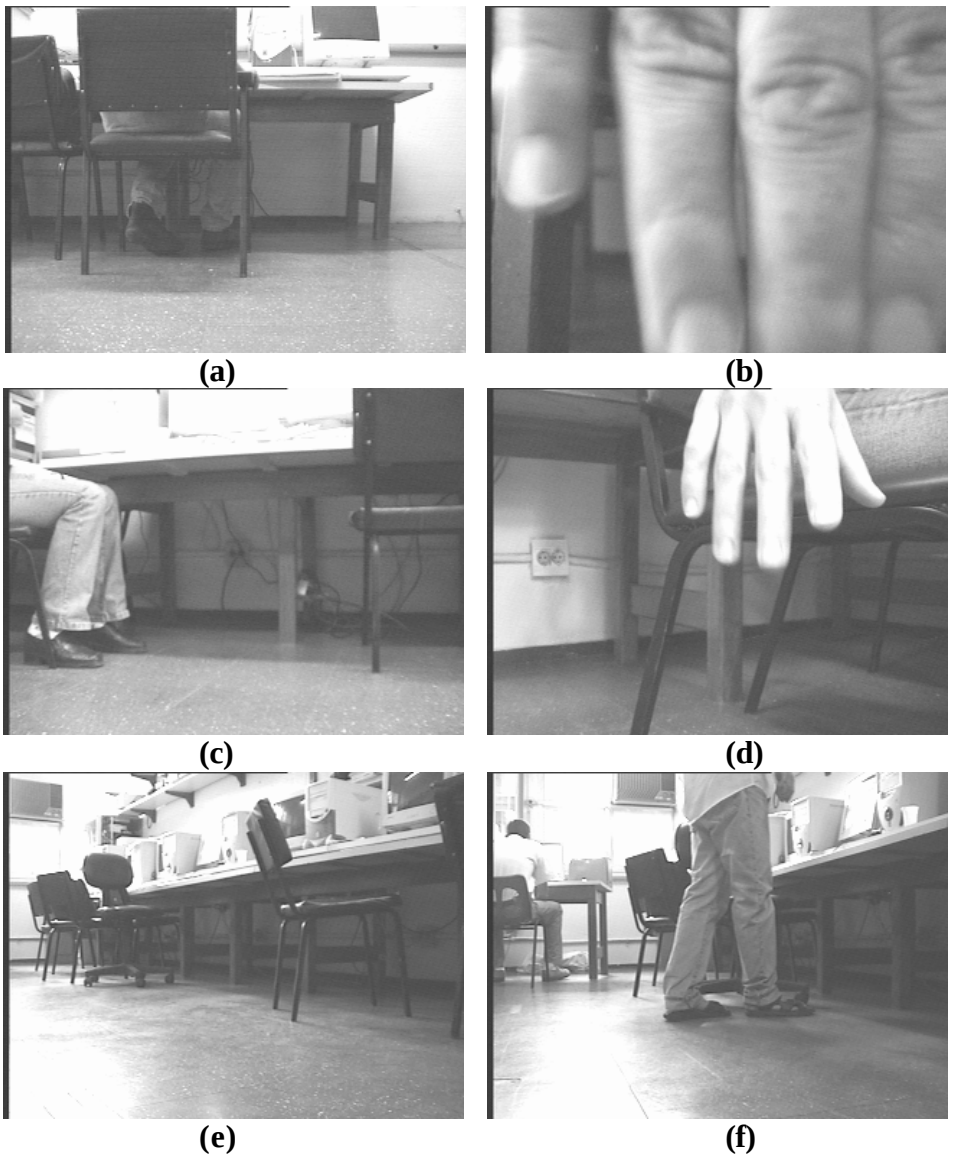


Figura 5.12 – Terceiro experimento de navegação: imagens adquiridas pela câmara do robô nas posições indicadas pelas letras (a), (b), (c), (d), (e) e (f) na Figura 5.11.

5.3.4 Experimento de Evitar Parede

Para comprovar a eficiência da estratégia montada para evitar paredes, usando o diagrama de tempos para contato e o resultado do processo de segmentação, foi montado um experimento onde o robô está impedido de girar, a menos que seja para desviar de uma parede. Desta maneira, os comandos de giro originados no estado Retroceder são repassados ao robô, enquanto os ângulos de giro gerados no estado Calcular Ângulo são ignorados.

O robô é, então, colocado entre uma parede do laboratório e um objeto plano revestido com um papel apresentando uma textura, representando uma outra parede. As duas paredes são paralelas e o robô se move entre elas, na direção perpendicular. A Figura 5.13 apresenta a trajetória executada pelo robô durante o experimento, a qual foi obtida pelo sistema de odometria do robô. A Figura 5.14 mostra as imagens obtidas pela câmara do robô nas posições indicadas pelas letras de (a) a (d) na Figura 5.13. Nela, pode-se notar que o robô vai e volta quase pelo mesmo caminho, tal como esperado.

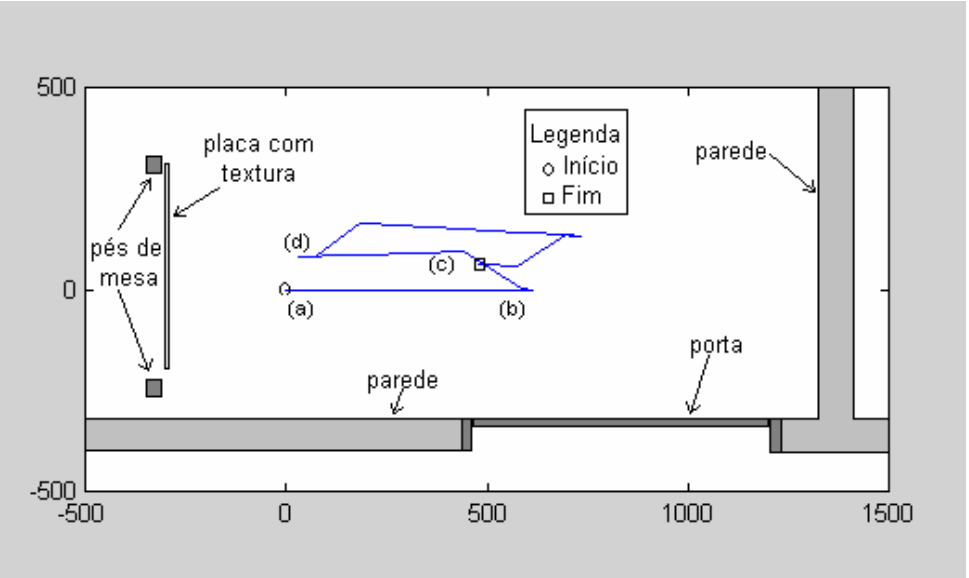
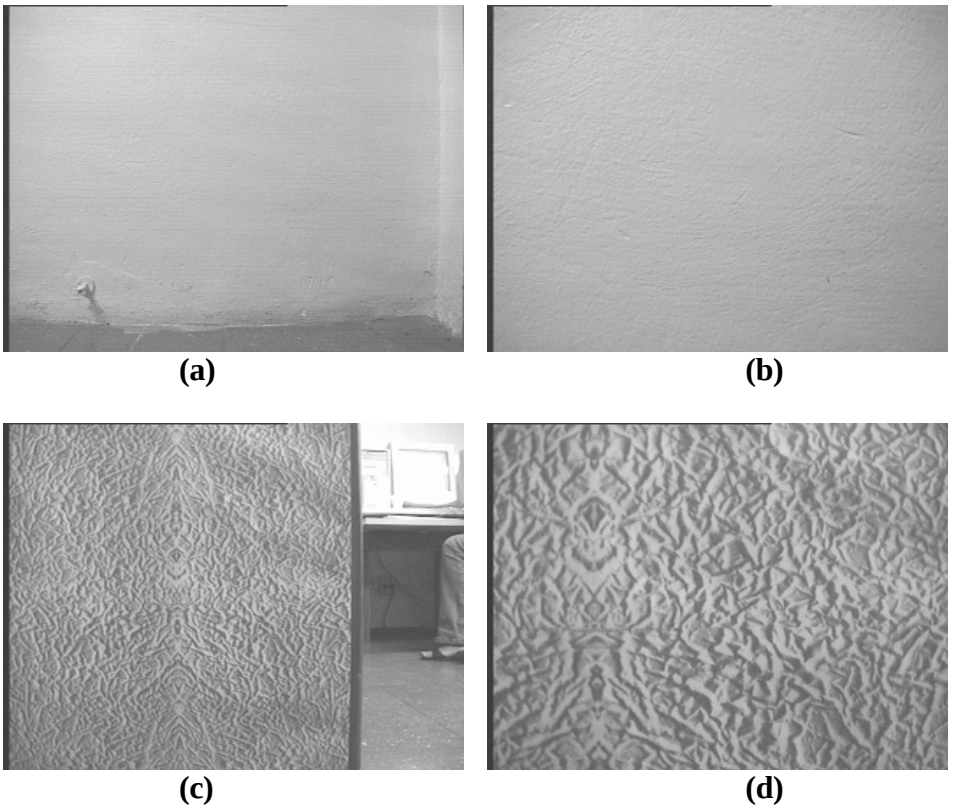


Figura 5.13- Trajetória do robô durante o experimento de evitar paredes.



(a) **(b)**

(c) **(d)**

Figura 5.14 – Evitar paredes: imagens adquiridas pela câmara do robô nas posições indicadas na Figura 5.13 pelas letras (a), (b), (c) e (d).

5.4 O Tempo de Processamento

Neste trabalho é proposto um sistema de sensoriamento visual para ser usado no controle da navegação de um robô móvel. Para esta aplicação, é imprescindível que todo o processo de aquisição e processamento de imagens seja feito de maneira rápida e eficiente, gerando resultados confiáveis em um tempo suficientemente pequeno, para ser usado no controle em tempo real do robô.

Buscando alcançar este objetivo, os algoritmos de cálculo do fluxo óptico foram estudados, implementados e comparados experimentalmente. Além disto, o algoritmo de Mínimos Quadrados Modificado foi proposto, como uma adequação de um algoritmo existente na literatura para as necessidades desta tarefa, o que ficou comprovado nos experimentos realizados, onde ele demonstrou ser eficiente e rápido. Além disto, a bordo do robô o algoritmo foi implementado usando imagens sub-amostradas, de 320×240 *pixels*, utilizando regiões menores e um menor número de pontos de observação. Foram, também, utilizadas as estruturas de dados e as funções da biblioteca OpenCV da Intel, para acelerar os cálculos matriciais necessários à obtenção da estimativa. Tudo isto levou a uma redução significativa do tempo de cálculo, preservando também a qualidade dos resultados.

Um outro ponto a considerar na questão do tempo de processamento é relacionado ao processo de segmentação das imagens com base no fluxo óptico. Apesar de ser um processamento bastante simples, executado em uma única varredura das regiões da imagem, este processamento gera um acréscimo no tempo de processamento que não pode ser desprezado.

Da mesma maneira, a determinação do foco de expansão deve ser considerada como uma parcela de contribuição para o tempo de processamento, apesar de ser realizada em uma só varredura das regiões da imagem.

A caracterização dos objetos, a qual consiste na determinação do tempo para contato por objeto e da magnitude média do fluxo óptico por objeto é uma outra operação executada durante o processamento, e que contribui para o aumento do tempo de processamento.

Uma outra parcela deste tempo é devida ao processo de caracterização do ambiente, através da geração do diagrama de tempos para contato, a partir dos valores de tempo para contato dos objetos, e da verificação da existência de objetos com magnitude média do fluxo óptico acima de um determinado limiar. Além disto, neste processo de caracterização, tendo verificado que não existe rotação, faz-se o cálculo do valor médio e da variância dos tempos do diagrama, e verifica-se a presença de objetos muito grandes no resultado da segmentação.

A Tabela 5.1 mostra os tempos gastos para executar cada uma destas tarefas no computador a bordo do robô. O programa implementado para fazer a aquisição e o processamento das imagens e para controlar a navegação do robô executa sobre o sistema Windows 98. Nesta tabela, pode-se notar que o tempo gasto para executar a aquisição das imagens é um fator limitante na redução do tempo total de processamento. Em uma situação ideal, o tempo de aquisição seria correspondente a uma taxa de 30 quadros adquiridos por segundo, porém mesmo que não fosse esse o tempo para aquisição, seria mais razoável que este tempo fosse constante. Isto não acontece porque, a cada comando de aquisição, é necessário que a câmara termine a varredura que está executando e chegue ao ponto onde deve começar a gravar o quadro desejado.

Tabela 5.1 – Os tempos gastos nas fases do processamento das imagens (em ms).

Aquisição das imagens	Cálculo do fluxo óptico	Segmentação da imagem	Determinação do FOE	Análise de características	Cálculo do ângulo de giro	Total
99	44	6	2	2	8	161
53	44	6	1	3	7	114
62	44	6	1	2	8	123
60	50	6	2	2	-	121
69	44	6	1	3	7	130
72	43	6	2	2	7	132
52	43	6	2	2	7	112
81	44	6	1	2	-	134
56	44	8	1	2	7	118
75	44	6	1	2	7	135
79	44	6	1	2	-	132

Sendo assim, comandando a aquisição de uma imagem de 320×240 *pixels* correspondente ao campo par, como foi o caso, pode ser necessário esperar a câmara terminar de fazer uma varredura do campo par (que já teria iniciado antes do comando chegar ao sistema), depois uma varredura do campo ímpar, chegando ao ponto onde o campo par pode ser adquirido. Sendo assim, o tempo de espera pode ser de até 33,33 ms, ou seja, o tempo que a câmara gasta para adquirir uma imagem inteira com os dois campos par e ímpar. Este tempo de espera pode causar uma variação bastante grande no tempo de aquisição de uma imagem.

Uma outra razão para uma variação no tempo de aquisição das imagens é relacionada a uma característica do sistema de aquisição disponível no robô. No sistema montado no computador externo, utilizando a placa Data Translation e a biblioteca Frame Grabber SDK, para adquirir n imagens em seqüência é necessário alocar memória para as *n* imagens e comandar a aquisição dizendo o número de imagens e passando o endereço da área de memória. O sistema adquire, então, as *n* imagens em seqüência e armazena na área de memória alocada. No sistema embarcado, usando a placa Imagenation e o programa PXC, deve-se, da mesma maneira, alocar a memória necessária para as *n* imagens e, depois, enviar *n* comandos de aquisição. Ou seja, no sistema disponível no robô não existe uma forma de adquirir uma seqüência contínua de imagens. Assim, no sistema externo existe uma parcela de tempo variável para o acesso à placa e depois, para cada imagem da seqüência, uma parcela fixa de tempo para a aquisição em si, enquanto no sistema embarcado todas as imagens são adquiridas com a parcela de tempo variável. Além disto, o Windows não é um sistema de tempo real, de forma que os comandos são passados ao *hardware* de acordo com a posição das tarefas na pilha do sistema, seguindo as prioridades que ele atribui a cada tipo de comando. Sendo assim, os comandos não são passados à placa de aquisição imediatamente após serem requisitados pelo programa.

Uma outra parcela de grande peso no tempo de processamento é o cálculo do fluxo óptico, que, embora seja feito por regiões e por um método bastante simples, é, ainda, um processo que consome bastante tempo. Este tempo pode ser reduzido aumentando-se o tamanho da região e diminuindo-se o número de pontos de observação usados na determinação do fluxo

óptico. Porém, como já foi discutido, isto diminui a confiabilidade dos resultados, tornando o sensoriamento visual mais sujeito a falhas.

Como foi mostrado nas seções anteriores, o sistema de controle implementado neste trabalho executa em duas *threads* separadas, sendo uma responsável pelos cálculos e a outra responsável por enviar comandos ao robô e verificar a sua execução. Além disto, foi também dito que os comandos são enviados ao robô através da interface fornecida pelo programa Saphira. Tendo isto em vista, foram feitas duas medições de tempo nas *threads*, a primeira no fim da *thread* de processamento e a segunda no fim da *thread* de comando. Assim, é possível medir o tempo gasto pelo sistema para executar os cálculos e gerar um sinal de controle, e o tempo gasto para executar os comandos.

A Tabela 5.2 mostra os tempos de cálculo e ação, medidos desta maneira para o mesmo experimento apresentado na tabela anterior. Como se pode ver a partir desta tabela, o tempo gasto no cálculo é bem menor do que o tempo gasto para executar mesmo a tarefa mais simples, ou seja, os tempos de cálculo são totalmente compatíveis com a dinâmica do robô. Como se pode notar, comparando as duas tabelas, o tempo de cálculo medido na segunda tabela é um pouco maior que na primeira. Isto se deve ao fato de que, como uma *thread* dispara um evento para chamar a outra, existe um tempo de comunicação embutido nas medições.

Tabela 5.2 – Os tempos gastos no processamento e na execução dos comandos.

Tempo de cálculo (ms)	Tempo para retroceder (ms)	Tempo de giro (ms)	Ângulo de giro (graus)	Tempo para avançar (ms)
178	-	592	-3	1004
124	-	1071	10	1006
131	-	733	7	1006
130	1005	2590	180	1005
138	-	1057	-20	1005
141	-	504	6	1010
115	-	1020	12	1005
137	1003	2584	180	1006
121	-	1018	-10	1011
138	-	721	-4	1006
135	1005	2474	180	1007

5.5 Generalidade e Escalabilidade do Sistema Proposto

Ainda considerando o sistema de sensoriamento e controle proposto, uma característica importante é a sua generalização, ou seja, a capacidade de utilização deste mesmo sistema em outro robô móvel, com outras capacidades e restrições. Ou seja, é plenamente possível implementar o mesmo sistema em outra plataforma, bastando para isto algumas poucas adaptações, que correspondem a uma reconfiguração do sistema.

Considere-se, inicialmente, a aquisição e o processamento das imagens. As imagens podem ser adquiridas com qualquer placa de aquisição que possua uma interface que as torne

disponíveis em algum formato de vídeo conhecido. Para se ter uma idéia, este trabalho foi desenvolvido usando três placas de digitalização de imagens, em suas diferentes fases. Os primeiros algoritmos foram desenvolvidos em San Juan, Argentina, no Instituto de Automática da Universidad Nacional de San Juan, onde a aquisição de imagens e parte do processamento (como o cálculo de derivadas) era realizado por intermédio de uma placa DataCube. A seqüência do trabalho, já no Brasil, foi realizada com uma placa Data Translation DT 3153. Por sua vez, a implementação a bordo do robô envolve a aquisição de imagens por meio de uma placa Imagenation, através do programa de interface PXC. Além disto, nas duas primeiras fases as imagens eram capturadas por uma câmara CCD monocromática, enquanto no robô a câmara disponível é CCD colorida. O formato fornecido pela placa de aquisição de imagens da Data Translation é RGB, com 24 bits por *pixel*, independentemente da câmara. Porém, no robô é possível adquirir imagens diretamente em nível de cinza, com 8 bits por *pixel*. Estas diferenças de hardware podem até gerar algum problema de adaptação inicial, o qual, depois de resolvido, influirá muito pouco no resultado. A possibilidade de executar determinados procedimentos com hardware dedicado, tal como o cálculo das derivadas das imagens adquiridas, porém, pode reduzir significativamente o tempo de processamento, e representa, praticamente, uma nova implementação.

Com relação ao sistema de controle, ele pode ser implementado em qualquer plataforma móvel em que se possam executar comandos de velocidade linear constante, velocidade angular constante, e giro de um ângulo pré-determinado. Isto porque o sistema de controle envia ao robô um comando para manter uma velocidade linear constante, enquanto as imagens são adquiridas e processadas. Já no instante da aquisição, obedecendo a uma restrição do sistema de sensoriamento, é necessário que o robô tenha velocidade angular zero. Ao fim do processamento das imagens, os objetos detectados são considerados para calcular quanto o robô deve mudar sua direção, a fim de evitar colisões. O desvio é feito através de comandos para girar de um determinado ângulo para a direita ou para a esquerda, sendo que a definição do valor limite do ângulo de giro deve considerar as dimensões do robô móvel.

Em relação ao ambiente de trabalho do robô, considerou-se na definição do problema que seria fechado, de piso plano e com objetos comuns em um escritório, ou seja, semi-estruturado. Nenhuma outra restrição foi adicionada, visando produzir um sistema de sensoriamento robusto. Assim, nenhuma textura foi adicionada aos objetos no ambiente para facilitar o cálculo do fluxo óptico. Também não foi feito qualquer controle da iluminação, e, conseqüentemente, haviam áreas muito iluminadas, como janelas, e áreas bastante escuras, como regiões sob as mesas. O sistema foi testado em outros ambientes dentro do prédio, com outras características, mantendo o mesmo comportamento de dentro do laboratório. Ainda em relação ao ambiente, em momento algum o sistema recebe informação do ambiente que não seja a partir do sistema de sensoriamento visual.

A utilização deste sistema de sensoriamento em um outro tipo de robô é analisada a seguir. Em um robô submarino, se existir alguma forma de garantir que o robô esteja executando apenas movimento de translação na direção do eixo óptico da câmara, é possível utilizar o sistema de sensoriamento proposto neste trabalho para detectar a presença de objetos, mesmo que o eixo de translação não seja horizontal. Além disto, se o robô executa um movimento de rotação conhecido, a componente do fluxo óptico devida a este movimento pode ser calculada e retirada do fluxo total, deixando apenas a componente devida à translação. Com esta adaptação, é também possível utilizar este sistema de sensoriamento. Vale ressaltar que, em um sistema submarino, existem outras fontes de força agindo sobre o sistema, o que dificulta grandemente o controle. Existem, por exemplo, correntes marítimas

que podem influenciar, em muito, a direção para onde vai o robô. Existe também a influência do cordão umbilical, que normalmente é utilizado neste tipo de plataforma para fornecer energia elétrica ao sistema embarcado. Ao tentar mover o robô é necessário arrastar o cordão, que oferece resistência por causa do atrito com a água. Desta maneira, pode ser bastante complicado atender à restrição de movimento translacional neste tipo de plataforma.

A utilização deste sistema em um robô de patas e em um robô a rodas preparado para ambientes externos é bastante restrita. Em um robô com patas, em geral, a plataforma onde a câmara é fixada pode mover-se para cima e para baixo (durante a caminhada em um piso plano, por exemplo), e pode inclinar-se de acordo com a inclinação do piso no ambiente (como quando o robô sobe ou desce uma rampa ou uma escada). No caso do robô a rodas navegando em um ambiente externo, a presença de irregularidades no piso torna a análise das imagens difícil, porque o movimento executado no plano da imagem, entre uma imagem e outra na seqüência, é muito mais complexo.

Uma outra característica que deve ser analisada em relação ao sistema de sensoriamento e controle proposto é a sua escalabilidade, ou seja, a capacidade do sistema funcionar em caso de aumento ou diminuição da velocidade do robô, do tamanho da imagem e das dimensões do ambiente, entre outras coisas.

Quanto à variação no número de *pixels* da imagem, por exemplo, o sistema foi implementado inicialmente com imagens entrelaçadas de vídeo de 640×480 *pixels* e depois com metade da resolução, ou seja, com imagens de 320×240 *pixels*, obtidas com a varredura par e metade da resolução horizontal das imagens usadas inicialmente. Independentemente desta mudança, o sistema continuou a obter os resultados necessários, mostrando-se robusto a este tipo de variação. No caso contrário, utilizando-se uma câmara de vídeo com mais *pixels*, desde que a imagem retratada fosse a mesma, seria necessário apenas adaptar o tamanho das regiões. O que se deve levar em conta nestes casos é o aumento no número de dados para processar, que deveria ser compensado com um aumento na capacidade de processamento e armazenamento para manter o tempo de processamento.

Uma outra mudança pode ser analisada, consistindo na troca da câmara de vídeo por uma outra com um ângulo de visada maior, posto que um ângulo de apenas 48,8º limita a faixa visível do ambiente a uma área restrita à frente do robô. Com um ângulo pequeno é possível considerar um único modelo para o fluxo óptico em todos os *pixels* na imagem. Porém, à medida que se aumenta o ângulo de visada, há um aumento de distorção no fluxo óptico esperado, principalmente nas faixas mais laterais do campo de visão. Considere, por exemplo, que o robô esteja em movimento de translação durante a aquisição das imagens com uma câmara de vídeo de 180º de visada. Espera-se que na parte central da imagem o fluxo óptico seja radial (movimento de aproximação, Capítulo 2), enquanto que nas laterais ele tende a ser mais horizontal (movimento paralelo a uma parede, Capítulo 2). Assim, para utilizar uma câmara de vídeo com maior ângulo de visada neste sistema de sensoriamento, deve-se considerar apenas a região da imagem onde o fluxo óptico segue o modelo radial esperado, a fim de obter resultados confiáveis. As regiões laterais da imagem podem ser usadas para alimentar algum outro sistema de sensoriamento, ou, então, ser simplesmente desprezadas.

Considerando a possibilidade de aumentar a velocidade do robô, a limitação está associada ao sistema de sensoriamento, mais diretamente ao tempo necessário para adquirir e processar as imagens, ou seja, à velocidade do sistema de aquisição e à capacidade de processamento e armazenamento do sistema de hardware utilizado. No caso do sistema de

controle há que se considerar a velocidade com que os comandos são passados ao robô, e em quanto tempo ele é capaz de executá-los. Assim, para utilização do sistema proposto com o robô a uma velocidade maior, deve-se pensar em uma implementação com um sistema operacional adaptado para aplicações em tempo real, e em uma forma de reduzir o tempo de aquisição das imagens.

5.6 Considerações Finais

Este capítulo descreve a implementação do sistema global proposto no Capítulo 4 a bordo do robô PIONEER 2-DX. Devido a restrições de tempo de processamento, algumas modificações foram realizadas como a redução no tamanho das imagens adquiridas de 480×640 *pixels* para 240×320 *pixels*, a mudança de vizinhança-oito para vizinhança-quatro no processo de segmentação e a utilização da biblioteca *OpenCV* da Intel que explora a extensão MMX disponível no computador PENTIUM de bordo, para a realização de cálculos matriciais, permitindo obter a estimativa do fluxo óptico em cada região de maneira mais rápida, entre outras coisas. Estas modificações levaram a uma redução considerável no tempo de processamento correspondente ao sistema de sensoriamento, como mostrado ao longo deste capítulo.

O sistema de controle implementado, o qual utiliza as informações fornecidas pelo sistema sensorial proposto no Capítulo 3, é tal que quando os objetos na cena estão muito próximos, de forma que uma colisão é iminente, o robô é parado e girado cento e oitenta graus, retornando por onde veio. Isto é necessário porque a câmara a bordo do robô tem uma abertura muito pequena (ângulo de visada horizontal de 48,8°). Em uma situação em que não é percebida iminência de colisão, por outro lado, o sistema de controle apenas gira o robô de um ângulo relativamente pequeno, de forma que o robô se desvie suavemente dos obstáculos.

Para finalizar, tendo o sistema global sido implementado a bordo do robô, foram realizados experimentos de navegação, para validação dos sistemas de sensoriamento e de controle propostos, cujos resultados foram apresentados neste capítulo. O sistema mostrou-se capaz de detectar objetos, fazendo com que o robô navegasse de maneira segura pelo laboratório sem que o ambiente fosse modificado, quer pela adição de textura quer pelo controle de iluminação.

Na análise dos tempos gastos em cada fase do processamento, ao longo dos experimentos, verificou-se que todo o ciclo é realizado em um tempo pequeno, em comparação com a dinâmica do robô, sendo, assim, possível fornecer dados ao sistema de controle em tempo real. Deve-se ressaltar que uma parte expressiva do tempo de processamento da informação sensorial é gasta exclusivamente na captura dos dois quadros de imagem consecutivos necessários ao cálculo do fluxo óptico (o tempo de aquisição da seqüência de imagens é função do tempo de varredura). Mesmo usando resolução de 240×320 *pixels* para cada quadro, na maioria dos casos o tempo total para aquisição dos dois quadros foi superior a 50% do tempo total gasto para capturar as imagens, calcular o fluxo óptico e determinar o incremento a ser imposto ao ângulo de *heading* atual do robô (houve casos em que tal tempo excedeu a 60% do total). Isto impõe uma restrição sobre o tempo de processamento, e conseqüentemente sobre a velocidade máxima de translação do robô. Assim, seria extremamente interessante implementar um sistema de captura de vídeo mais rápido consistindo de uma câmara digital rápida (padrão IEEE 1394 – *fire-wire*).

Os resultados obtidos nos experimentos apresentados neste capítulo mostram que, da mesma maneira que no exemplo dos animais, é possível controlar a navegação de um robô móvel usando um sistema de visão monocular. Fica demonstrada a viabilidade de usar o fluxo óptico como forma de propiciar o sensoriamento necessário à navegação reativa autônoma e segura de um robô móvel, mesmo considerando a capacidade de processamento limitada normalmente disponível a bordo de robôs de pequeno porte, o que constitui a mais importante contribuição deste trabalho.

Capítulo 6 - Conclusões e Trabalhos Futuros

Um sistema de sensoriamento visual monocular, ou seja, utilizando uma única câmara de vídeo, é usado neste trabalho como fonte de toda a informação utilizada para controlar um robô móvel durante sua navegação, de forma que o mesmo pode evoluir por seu espaço de trabalho sem colidir com nenhum obstáculo. Um sistema mínimo de controle, fazendo uso da informação sensorial disponibilizada, foi projetado e implementado para permitir a validação experimental do sistema sensorial proposto. O sistema de controle projetado implementa no robô um comportamento reativo, fazendo-o navegar desviando-se dos obstáculos que apareçam à sua frente. Os experimentos realizados com o sistema proposto comprovam que, da mesma maneira que no exemplo dos animais, é possível obter as informações necessárias para controlar a navegação de um robô móvel usando visão monocular, realizando todo o processamento a bordo de um robô móvel de pequeno porte, apesar da reduzida capacidade de processamento disponível, e sem qualquer alteração no ambiente.

Para obter informações de proximidade utilizando as seqüências de imagens obtidas pela câmara de vídeo foi usada a técnica do fluxo óptico. Neste trabalho, foi proposto um algoritmo não iterativo para o cálculo do fluxo óptico, chamado algoritmo de Mínimos Quadrados Modificado, que nos experimentos comparativos realizados apresentou os melhores resultados, sendo selecionado como base para o sistema de sensoriamento visual.

Além disto, uma outra contribuição deste trabalho foi o novo algoritmo iterativo proposto para o cálculo do fluxo óptico, que embora não tenha sido selecionado para a implementação a bordo do robô apresentou uma melhoria expressiva em relação ao algoritmo original, em se tratando de imagens com descontinuidade, o qual foi chamado algoritmo de Horn e Schunck Normalizado.

Aproveitando os vetores de fluxo óptico calculados, foi proposto também neste trabalho um sistema eficiente para segmentação de movimento, que permite ao robô identificar as distintas profundidades de diversos objetos presentes numa cena, de forma que o sistema de controle pode, então, desviar o eixo de translação do robô de forma a evitar os obstáculos mais próximos, buscando uma direção de deslocamento mais segura.

Uma vez que o sistema de controle implementado consiste na camada de mais baixo nível em um sistema de controle baseado em comportamentos, pode-se sugerir como continuação deste trabalho a implementação de outros comportamentos no robô, utilizando o sistema proposto como base.

Além disto, os resultados experimentais apresentados no Capítulo 5 permitem perceber que o sistema sensorial proposto detecta muito bem o movimento de pessoas no ambiente de trabalho do robô, pelo incremento que isto causa nos vetores de fluxo óptico calculados. Assim, uma proposta para a continuidade deste trabalho seria discutir a utilização do mesmo, em sistemas de vigilância, por exemplo, para seguir pessoas que surjam no ambiente de trabalho do robô, ou mesmo numa situação em que a câmara esteja fixa, podendo apenas ser girada.

Referências Bibliográficas

[1] Ancona, N., e Poggio,T., "Optical Flow from 1D Correlation: Application to a Simple Time-to-Crash Detector", *International Journal of Computer Vision,* vol. 14, pp. 131-146, 1995.

[2] Arbib, M A., "Perceptual Structures and Distributed Motor Control", *Handbook of Physiology – The Nervous System II: Motor Control*, Ed. V. B. Brooks, American Physiological Society, Bethesda, MD, pp. 1449-1480, 1981.

[3] Arkin, R. C., *Behavior-Based Robotics*, MIT Press, 1998.

[4] Baratoff, G., Toepfer, C., e Neumann, H., "Combined Space-Variant Maps for Optical Flow Based Navigation", *Biological Cybernetics (Special Issue on Navigation in Biological and Artificial Systems)*, vol. 83, nº. 3, pp. 199-209, 2000.

[5] Barron, J. L., Fleet, D. J., e Beauchemin, S. S., "Performance of Optical Flow Techniques", *International Journal of Computer Vision*, vol. 12, nº. 1, pp. 43-77, 1993.

[6] Beauchemin, S. S., e Barron, J. L., "The Computation of Optical Flow", *ACM Computing Surveys*, vol. 27, nº. 3, pp. 433-467, 1995.

[7] Ben-Tzvi, D. e Sandler, M., "A Combinatorial Hough Transform", *Pattern Recognition Letters*, vol. 11, nº. 3, pp. 167-174, 1990.

[8] Bergen, J. R., Burt, P. J., e Hanna, K., "Dynamic Multiple-Motion Computation", *Artificial Intelligence and Computer Vision*, Elsevier, Holanda, pp. 147-156, 1992.

[9] Blazer, D., "A Horse, of Course", *Horse News from The Equiworld Ezine Online Magazine*, disponível em (www.equiword.net/uk/ezine/0202/ahorseofcourse2.htm), 31 de Janeiro de 2002.

[10] Borshukov, G. D., Bozdagi, G., Altunbasak, Y., e Tekalp, A. M., "Motion Segmentation by Multi-Stage Affine Classification", *IEEE Transactions on Image Processing*, vol. 6, nº. 11, pp. 1591-1594, Novembro de 1997.

[11] Brown, R. G., e Hwang, P. Y. C., *Introduction to Random Signals and Applied Kalman Filtering*, John Willey and Sons, 3ª edição, 1997.

[12] Campani, M., e Verri, A., "Computing Optical Flow from an Overconstrained System of Linear Algebraic Equations", *Proceedings of the 3rd International Conference on Computer Vision*, Osaka, Japão, pp. 22-26, 1990.

[13] Camus, T., *Real-Time Optical Flow*, PhD Tesis, Department of Computer Science, Brown University, Providence, EUA, 1994.

[14] Camus, T., Coombs, D., Herman, M. e Hong, T. H., "Real-time Single-Workstation Obstacle Avoidance Using Only Wide-Field Flow Divergence", *Videre: Journal of Computer Vision Research*, vol. 1, nº. 3, MIT Press, 1999.

[15] Carelli, R., Secchi, H., Mut, V. e Nasisi, O., "Stable Algorithms for the Navigation of Mobile Robots in Corridors Using Optical Flow", *Proceedings of the 8th Workshop on Information Processing and Control*, Mar del Plata, Argentina, vol. 2, pp. 79-7 a 86-7 Setembro de 1999.

[16] Coombs, D., Herman, M., Hong, T., and Nashman M., "Real-time Obstacle Avoidance Using Central Flow Divergence and Peripheral Flow", *IEEE Transactions on Robotics and Automation*, vol. 14, nº. 1, pp. 49-59, Fevereiro de 1998.

[17] Costa, V. M. S., *Integração de Comportamentos Visuais para Robótica Móvel*, Dissertação de Mestrado, Universidade Técnica de Lisboa, Instituto Superior Técnico, Dezembro de 1998.

[18] De Micheli, E., e Verri, A., "Vehicle Guidance from One-Dimensional Optical Flow" Proceedings of the IEEE Symposium on Intelligent Vehicles, Tóquio, Japão, pp. 183-188, 1993.

[19] Dev, A., Kröse, B. J. A., e Groen, F. C. A., "Navigation of a Mobile Robot on the Temporal Development of the Optic Flow", *Proceedings of the 1997 IEEE/RSJ/GI International Conference on Intelligent Robots and Systems (IROS'97)*, Grenoble, França, pp. 558-563, Setembro de 1997.

[20] Dev, A., Kröse, B. J. A., e Groen, F. C. A., "Confidence Measures for Image Motion Estimation", *Proceedings of the 1997 RWC Symposium*, Japão, pp. 199-206, 1997.

[21] Dev, A., Kröse, B. J. A., e Groen, F. C. A., "Where Are You Driving to? Heading Direction for a Mobile Robot from Optical Flow", *Proceedings of the IEEE International Conference on Robotics and Automation*, Leuven, Bélgica, pp. 1578-1583, Maio de 1997.

[22] Dijkstra, T. M. H., Argante, E., e Gielen, C. C. A. M., "Motion Parallax from Catastrophies in Scale-space", *Proceedings of the International Conference on Artificial Neural Networks*, pp. 237-240, Amsterdã, Holanda, 1993.

[23] Dittrich, W. H., e Lea, S. E. G., "Motion Discrimination and Recognition", In R. G. Cook (Ed.), *Avian Visual Cognition* [On-line], Disponível em www.pigeon.psy.tufts.edu/avc/dittrich/, 2001.

[24] Duchon, A., Warren, W., e Kaelbling, L. P., "Ecological Robotics: Controlling Behavior with Optical Flow", *Proceedings of the 17th Annual Cognitive Science Conference*, pp. 164-169, 1995.

[25] Farah, M. J., *The Cognitive Neuroscience of Vision – Fundamentals of Cognitive Neuroscience*, Blackwell Publishers Inc., EUA, 2000.

[26] Fleet, D. J., e Jepson, A. D., "Computation of Component Image Velocity from Local Phase Information", *International Journal of Computer Vision*, vol. 5, pp. 77-104, 1990.

[27] Foushee, K. A., "Understanding Equine Vision", *NCSU Extension Horse Husbandry*, disponível em http://www.nchorsenews.com/articles_pg8.htm, Janeiro de 2000.

[28] Freire, E. O., Carelli, R., Mut, V., Soria, C. M., Bastos-Filho, T. F., e Sarcinelli-Filho, M., "Mobile Robot Navigation Based on the Fusion of Control Signals from Different Controllers", *Proceedings of European Control Conference 2001- ECC 01*, Porto, Portugal, pp. 1828-1833, Setembro de 2001.

[29] Freire, E. O., *Controle de Robôs Móveis por Fusão de Sinais de Controle Usando Filtro de Informação Descentralizado*, Tese de Doutorado, Universidade Federal do Espírito Santo, Agosto de 2002.

[30] Grossmann, E., e Santos-Victor, J. A., "Performance Evaluation of Optical Flow Estimators: Assessment of a New Affine Method", *Robotics and Autonomous Systems*, vol. 21, pp. 69-82, 1997.

[31] Horn, B. K. P., e Schunck, B. G., "Determining Optical Flow", *Artificial Intelligence*, vol. 17, pp. 185-203, 1981.

[32] Horn, B. K. P., e Schunck, B. G., "Determining Optical Flow: A Retrospective", *Artificial Intelligence*, vol. 59, pp. 81-87, 1993.

[33] Jones, J. L., Seiger, B. A., e Flynn, A. M., *Mobile Robots: Inspiration to Implementation*, A K Peters, Natick, MA, EUA, 1999.

[34] Kortenkamp, D., Bonasso, R. P., Murphy, R., *Artificial Intelligence and Mobile Robots – Case Studies of Successful Robot System*, AAAI Press/MIT Press, 1998.

[35] Lai, S. H., e Vemuri, B. C., "Robust and Efficient Computation of Optical Flow", *International Journal of Computer Vision*, vol. 29, pp. 87-105, 1998.

[36] Lucas, B., e Kanade, T., "An Iterative Image Registration Technique with an Application to Stereo Vision", *Proceedings of DARPA IU Workshop*, pp. 121-130, 1981.

[37] Nesi, P., Del Bimbo, A., e Ben-Tzvi, D., "A Robust Algorithm for Optical Flow Estimation", *Journal on Computer Vision, Graphics and Image Processing: Image Understanding*, Academic-Press, vol. 61, nº. 2, pp. 59-68, 1995.

[38] Neuro-Optometric Rehabilitation Association (NORA), "Patient Area - Implications of Acquired Monocular Vision (Loss of One Eye)", disponível em http://www.nora.cc/campus/monocular_vision.html, 2001.

[39] Nilsson, N. J., "A Mobile Automaton: An Application of AI Techniques", *Proceedings of the First International Joint Conference on Artificial Intelligence*, pp. 509-520, San Francisco, Morgan Kaufmann Publishers, 1969.

[40] Poggio, T., Verri, A., e Torre, V., "Green Theorems and Qualitative Properties of the Optical Flow", *AI Memo 1289*, Massachusetts Institute of Technology, Cambridge, MA, EUA, Abril de 1991.

[41] Ramel, G. J. L., "The Incredible World of Mammals -The Sensory World of Mammals – Sight and Vision - Sight and the Mammal Eye", *The Earth Life Web*, disponível em http://www.earthlife.net/mammals/vision.html, Abril de 2000.

[42] Royal National Institute of the Blind (RNIB offers practical support and advice to anyone with a sight problem), "Your Benefits and Rights - Monocular Vision: Information for people with sight in one eye only", disponível em http://www.rnib.org.uk/sightlos/brit/monoc.htm.

[43] Santos-Victor, J. A., Sandini, G., Curotto, F., e Garibaldi, S., "Divergent Stereo in Autonomous Navigation: From Bees to Robots", *International Journal of Computer Vision*, vol. 14, pp. 159-177, Março de 1995.

[44] Sarcinelli-Filho, M., Schneebeli, H. A., e Caldeira, E. M. O., "On the Use of Optical Flow in Mobile Robot Navigation: The Search for a Suitable Algorithm", *Proceedings of the 43rd Midwest Symposium on Circuits and Systems - MWSCAS 2000*, Lansing, MI, EUA, CD-ROM, 2000.

[45] Sarcinelli-Filho, M., Schneebeli, H. A., e Caldeira, E. M. O., "On Real-Time Optical Flow Calculation for Mobile Robot Navigation", *Proceedings of the 5th World Multi-Conference on Systemics, Cybernetics and Informatics - SCI2001*, vol. IX, pp. 60-65, Orlando, FL, EUA, Julho de 2001.

[46] Soria, C. M., Sarcinelli-Filho, M., Bastos-Filho, T. F., Carelli, R., e Nasisi, O., "Obstacle Detection and Avoidance using Optic Flow", *Proceedings of the 5th World Multi-Conference on Systemics, Cybernetics and Informatics - SCI2001*, vol. IX, pp. 66-71, Orlando, FL, EUA, Julho de 2001.

[47] M.V.Srinivasan, M.Lehrer, W.H.Kirchner, e S.W.Zhang, "Range Perception through Apparent Image Speed in Freely Flying Honeybees", *Visual Neuroscience*, vol. 6, pp. 519-535, 1991.

[48] Stöffler, N. O., Burkert, T., e Färber, G., "Real-Time Obstacle Avoidance Using an MPEG-Processor-Based Optic Flow Sensor", *Proceedings of the 15th International Conference on Pattern Recognition*, vol. 4, pp. 161-166, Barcelona, Espanha, Setembro de 2000.

[49] Tekalp, A. M., *Digital Video Processing*, Prentice-Hall, EUA, 1995.

[50] Temizer, S., *Optical Flow Based Local Navigation*, Tese de Mestrado, Massachusetts Institute of Technology, Cambridge, MA, EUA, Setembro de 2001.

[51] Verri, A., e Poggio, T., "Qualitative Information in the Optical Flow", *Proceedings of the Image Understanding Workshop*, McLean, VA, EUA, pp. 825-834, 1987.

[52] Verri, A., e Poggio, T., "Against Quantitative Optical Flow", *Proceedings of the First International Conference on Computer Vision (ICCV)*, Washington, DC, EUA, pp. 171-180, 1987.

[53] Wang, J. Y. A., e Adelson, E. H., "Representing Moving Images with Layers", *IEEE Transactions on Image* Processing, vol. 3, nº. 5, pp. 625-638, Setembro de 1994.

[54] Waran, N. K., “The Social Behavior of Horses”, In: *Social Behavior in Farm Animal*, Edited by L. Keeling, Swedish University of Agricultural Sciences, Skara, Sweden, Cap. 9.1.3, pp. 247-274, Abril de 2001.