Uma Estratégia de Navegação para Robôs Móveis em Ambientes Interiores

Raquel Frizera Vassallo

Dissertação de Mestrado em Engenharia Elétrica
Automação

Mestrado em Engenharia Elétrica - Automação

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Vitória, ES - BRASIL

Dezembro de 1998

Uma Estratégia de Navegação para Robôs Móveis em Ambientes Interiores

Raquel Frizera Vassallo

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

Aprovada em 07/12/98 por:

Prof. Dr. Hans Jörg Andreas Schneebeli - Orientador, UFES

Prof. Dr. José Santos-Victor - Co-orientador, ISR/IST

Prof. Dr. Mário Sarcinelli Filho, UFES

Prof. Dr. Mário Fernando Montenegro Campos, UFMG

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Vitória, ES - BRASIL

Dezembro de 1998

Vassallo, Raquel Frizera, 1973

Uma Estratégia de Navegação para Robôs Móveis em Ambientes Interiores. [Vitória] 1998

xiii, 97p., 29,7 cm (UFES, M. Sc., Engenharia Elétrica, 1998)

Dissertação, Universidade Federal do Espírito Santo, PPGEE.
I. Robótica - Visão Computacional
I. PPGEE/UFES II. Título (série)

Aos meus pais, Dalton e Angela,

que viveram intensamente comigo

todos os passos dados até este momento.

O presente trabalho foi realizado no Programa de Pós-Graduação em Engenharia Elétrica da UFES - Universidade Federal do Espírito Santo, com a colaboração do IST - Instituto Superior Técnico da Universidade Técnica de Lisboa e auxílio financeiro recebido deste instituto durante estágio lá realizado, com o financiamento de passagens aéreas para Portugal através do Programa CYTED - Comisión Interministerial de Ciencia y Tecnología para el Desarrollo.

Agradecimentos

Ninguém vive sozinho e não produz nada isoladamente. Ao final deste trabalho, eu gostaria de agradecer imensamente àqueles que me ajudaram a galgar mais este degrau na vida.

Aos meus pais, Dalton e Angela, meus maiores incentivadores e exemplos de dedicação, pelo amor e segurança, sempre presentes nas mãos firmes em que pude me apoiar durante toda esta jornada.

Ao Prof. Hans Jörg Andreas Schneebeli, orientador desta dissertação, pelo incentivo, atenção e disponibilidade. Já se vão quase sete anos de orientação, desde a minha primeira participação em um projeto seu durante a minha graduação. A ele devo a minha formação científica, fruto do seu investimento e confiança na minha pessoa.

Ao Prof. José Santos-Victor, meu co-orientador, pela atenção e carinho a mim dispensados durante o período em que estive sob sua orientação no Vislab, em Lisboa. Pela paciência em ensinar a esta "marinheira de primeira viagem" a arte de navegar através da Visão Robótica, sempre me estimulando a navegar incansavelmente nos corredores do ISR até que pudesse ultrapassar o "Cabo da Boa Esperança" e alcançar as "Índias".

Ao Prof. Mário Sarcinelli Filho, que com sua "força" e "insistência", sempre impulsiona seus alunos a seguir em frente.

Aos demais professores deste programa de pós-graduação, pela participação na minha formação.

Ao amigo Fransérgio, pela alegria e agradável companhia de todos os momentos. Ao Carlos De Vincenzo, pela amizade e o "largo sorriso" de sempre. Ao Roger, pela parceria nas batalhas iniciais em visão computacional. Ao Jan, pelas mais diversas histórias de aventuras e exemplo de praticidade. Ao Danilo, pela atenção nas poucas, mas decisivas noites em que foi meu "anjo da guarda" para que a escrita deste trabalho se concretizasse. À Prof^a. Jussara, pela amizade em vários momentos e paciência em me orientar no LaTex. À Marlene, secretária da pós-graduação, sempre disposta a resolver com grande agilidade toda a burocracia, além de bater bons papos. E aos demais amigos do LAI, pela certeza dos agradáveis momentos que sempre compartilhamos.

Ao Nuno e à Maria João, por me adotarem nos dias solitários em Lisboa. Ao Antonio, pelas eternas implicâncias nas brincadeiras e disponibilidade sincera para trabalhar ao lado da Labmate. Ao Carlos, ou Espanhol, pelo carinho e amizade constantes que preencheram minha estada em Lisboa. Ao Jordão e ao Etienne, pelo grande incentivo e amizade, mesmo depois do meu regresso ao Brasil. À toda equipe do Vislab, que recebeu de braços abertos esta "brazuca" como se já fosse uma deles.

Ao mais que amigo Alexandre, ou KBlo, que sempre me incentivou e esteve presente em grandes momentos desta caminhada e de muitas outras. Ao amigo Lucilinho, que esteve tão próximo, apesar da distância, através dos freqüentes telefonemas em que conversamos sobre nossos trabalhos. À torcida dos amigos que compreendem a grandeza desta etapa.

À minha irmã Paula que, mesmo distante, esteve sempre presente.

À minha madrinha, Tia Márcia, pela boa vontade em realizar uma correção relâmpago de português.

Às duas instituições de ensino superior que proporcionaram as condições para que esta dissertação se realizasse.

À UnED/Colatina da ETFES - Escola Técnica Federal do Espírito Santo, pelo apoio e facilidades a mim concedidos para que pudesse efetuar os estudos e trabalhos desta dissertação. Aos meus alunos que souberam compreender a minha ausência em certos momentos.

Finalmente, a Deus, que graças a Seu grande amor de Pai, não se importa de ser o último citado neste agradecimento.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Este trabalho aborda o problema de navegação baseada em visão para um robô móvel em ambientes interiores. Inicialmente, as soluções para problemas de navegação com visão computacional se baseavam na reconstrução 3D do ambiente como primeira etapa, antes de uma tomada de decisão. Porém, muitos problemas nessa área podem ser resolvidos de uma maneira mais direta e rápida se a estrutura do ambiente de trabalho e as características da tarefa a ser executada forem exploradas. Neste trabalho, a idéia principal é manter o robô centrado em um corredor, à medida que ele se movimenta e, além disto, torná-lo capaz de identificar a sua posição atual em relação a um percurso pré-definido, desempenhando tarefas específicas dependendo da sua posição. O controle da navegação é baseado em um sistema monocular constituído por uma câmara que captura imagens, em tons de cinza, de um corredor. A estratégia de navegação une dois paradigmas: o servo controle e o método baseado em aparência. O servo controle se baseia no ponto de fuga calculado a partir das retas do corredor e na assimetria entre as inclinações das mesmas para controlar a orientação e posição do robô. Sua atuação se concentra na tarefa de controle dos motores. O método baseado em aparência é utilizado para monitorar o percurso desenvolvido pelo robô através da definição de um mapa topológico do ambiente. Esta representação permite definir locais onde se executam tarefas especiais, como por exemplo, entrar por uma porta, girar ao final de um corredor, identificar um objeto, etc. Uma seqüência de imagens de referência, previamente capturadas, define o mapa interno do ambiente e indica a trajetória a ser percorrida pelo robô. Os resultados dos experimentos realizados mostram que a estratégia de navegação implementada foi capaz de solucionar adequadamente o problema proposto. Além disto, o mapa topológico do ambiente permitiu que as tarefas de virar ao final de um corredor e identificar o final do percurso fossem executadas com sucesso. Os resultados obtidos são encorajadores e mostram a funcionalidade da combinação dos dois paradigmas utilizados. A união dessas duas técnicas possibilitou a implementação de um sistema mais autônomo, robusto e de resposta rápida, sem a necessidade de uma reconstrução 3D do ambiente.

Abstract

This work addresses the problem of visual-based navigation of a mobile robot in indoors environments. Earlier vision research was based on full or partial 3D reconstruction of the environment before defining the robot navigation or taking a decision. However, many problems in this area can be solved by using a more direct and purposive approach, exploiting the working area structure and the characteristics of the task to be performed. In this work the main idea is to maintain the robot in the middle of a corridor during navigation and, besides that, make it capable to identify its position related to a pre-defined trajectory, in order to perform special tasks on specific locations. The robot control is based on a single camera which observes the area and provides grayscale images of the corridor. The navigation strategy merges two distinct paradigms, visual servoing and appearance based methods. The visual servoing uses the vanishing point defined from the corridor guidelines and the asymmetry among their slopes to control the robot orientation and position. It is used locally to perform the motor control of the robot. The appearance based method is used for monitoring the navigation process by constructing a topological map of the environment. This internal representation enables the system to define places where special tasks like entering doors, making turns at the end of a corridor, identifying an object, etc, can be performed. A sequence of reference images is previously captured to provide the topological description of the environment and to define the robot trajectory. The results show that the navigation strategy was able to correct the robot trajectory when needed and the topological map made the tasks of turning at the end of a corridor and stopping when the final position was met to be performed successfully. The experiments results are encouraging and actually show the advantages of combining both paradigms. The union of these two techniques increaesed the autonomy, robustness and speed of the system without needing a 3D model of the area.

Capítulo 1

Introdução

Valeu a pena? Tudo vale a pena
Se a alma não é pequena.
Quem quere passar além do Bojador
Tem que passar além da dor.

("Mar Portuguez" - Fernando Pessoa)

Na robótica móvel, a interação de um robô com o ambiente de trabalho pode ser conseguida através de diversos sistemas de sensoriamento. As respostas obtidas a partir dos sensores influenciam o comportamento e as ações efetuadas por um robô. Com o aumento da complexidade dos trabalhos em robótica, a cada dia aumenta a necessidade de maior precisão nos sistemas de controle, não sendo suficientes as medidas e referências obtidas através das próprias leituras dos atuadores envolvidos. Sensores adicionais, como ultrasom [23, 26] e visão computacional [28, 32], são freqüentemente utilizados para melhorar a interatividade com o meio de trabalho [7]. Estes sistemas são capazes de extrair do ambiente informações que auxiliam na definição da orientação, posição e velocidade, na detecção de obstáculos, etc; melhorando a realimentação nas malhas de controle dos robôs. O tipo de sensores aplicados e a natureza das suas respostas definem a maneira como a informação extraída é trabalhada e utilizada [6, 34] nestes sistemas.

Sendo a visão um dos sentidos mais poderosos do ser humano, aumentam, a cada dia, o interesse e os esforços com relação ao uso de sistemas de visão na robótica móvel. Informações sobre o mundo real em três dimensões (3D) são extraídas a partir da sua projeção em uma imagem de duas dimensões (2D) [19, 45]. Todo sistema de visão, seja natural ou artificial, colabora com o movimento em pelo menos duas maneiras. Ele permite que o

seu possuidor seja capaz de detectar, medir e interpretar o movimento de objetos externos, e além disto, controlar, planejar e coordenar o seu próprio movimento. É muito comum, ainda, a cooperação entre diferentes sistemas sensoriais e a definição de comportamentos ou agentes como, por exemplo, procurar um marcador, vagar no ambiente, acertar orientação, evitar obstáculos, etc [36, 50]. A prioridade e a cooperação entre os comportamentos são definidas conforme a situação encontrada a cada momento do movimento [24, 61].

O uso de sensores visuais tem sido muito útil e importante, já que possibilita a realização de medições de sinais e a extração de características do ambiente, sem a necessidade de haver contato físico. Duas tarefas constantemente abordadas em robótica podem ser resumidas como o controle e posicionamento de uma garra, no caso de manipuladores, e o controle e posicionamento em relação ao ambiente, no caso de robôs móveis para a navegação [30]. Desta forma, a aplicação de visão artificial tem possibilitado a definição de sistemas de controle mais robustos e versáteis, com a vantagem de serem mais adaptáveis às mudanças e incertezas do local e, por isso, facilitarem a interação de robôs em ambientes menos estruturados. Garantir o sucesso da tarefa de navegação de um robô de uma maneira autônoma não é um problema de fácil solução e tem sido o centro de atenção para uma série de trabalhos na área [47].

Muitos dos trabalhos envolvendo visão computacional na robótica móvel se concentram na idéia da reconstrução geométrica do ambiente. Nestes casos, os problemas de navegação são resolvidos através de um modelo matemático do ambiente em três dimensões [13, 20]. À medida que o robô se movimenta, imagens do ambiente são capturadas de diferentes posições e utilizadas na reconstrução 3D do local. A partir desta representação métrica da região, define-se a trajetória a ser executada pelo robô. Além disto, torna-se fácil a determinação da posição absoluta do robô em relação a um referêncial fixo no mundo. Como desvantagem deste método, está o fato de o processo de reconstrução 3D requerer muito tempo de processamento e esforço computacional. O tempo gasto com o processamento das imagens para a extração das características através de segmentação, detecção de contornos, correspondência entre pontos de imagens estéreo e definição de mapas de disparidade e profundidade, é muito grande devido à complexidade destes cálculos que, juntos, tornam-se computacionalmente dispendiosos. Na maioria das vezes, ainda se torna necessário o conhecimento prévio de uma série de parâmetros e uma calibração precisa das camâras que serão utilizadas. A determinação destes parâmetros e a definição de diferentes métodos de calibração das câmaras são constantemente abordados em trabalhos como em [59, 62].

Porém, em muitos casos, não se torna necessária a reconstrução 3D para a solução do problema de navegação. Basta extrair das imagens aquilo que realmente é necessário e relevante para o sucesso da missão a ser realizada ou para a execução de tarefas, como passar por uma porta, desviar de um obstáculo, reduzir a velocidade em determinadas situações, etc. Por exemplo, para ir de um ponto A para um ponto B em uma sala, não se precisa realizar uma reconstrução do ambiente, mas simplesmente ter a noção relativa entre um ponto e outro e evitar os obstáculos à medida que se caminha no trajeto. É com esta idéia que muitos problemas em visão computacional podem ser resolvidos, encarando-se o problema de uma forma mais direta e simples [1], e com o enfoque realmente relevante para a tarefa em questão. Isto possibilita a utilização de técnicas mais qualitativas e robustas para a solução do problema de navegação.

Esta idéia sugere que a visão não seja usada simplesmente com a intenção de observar o ambiente, mas como uma forma de coletar informações e verificar a realização da tarefa continuamente. Esta forma de utilização da visão computacional durante o movimento do robô, representa uma abordagem conhecida por "visão ativa" [2, 47]. É como no caso do ser humano, onde a percepção não é passiva, mas sim ativa. Não se observa simplesmente, mas se explora, procura e extrai do ambiente a informação desejada ou que seja importante.

A aplicação de visão ativa durante o movimento de um robô móvel permite uma precisão maior no controle dos motores, sensores e atuadores envolvidos, além de melhorar a interação com ambientes menos estruturados. A detecção de objetos e eventos esperados ou inesperados torna-se mais simples e melhora-se a possibilidade de solucionar o problema ou tarefa de uma maneira mais direta e menos complexa. Esta abordagem da visão, num modo contínuo de controle de ações do robô, pode ser observada em trabalhos como [4, 18, 52].

No caso da navegação de robôs móveis, podem ser encontrados vários sistemas diferentes de aplicação de visão computacinal. O tipo de características e informações exploradas nas imagens variam conforme o ambiente de trabalho (estruturado ou não, externo ou interno) e a tarefa a ser executada pelo robô [33]. São encontrados sistemas de visão que exploram de diversas formas as disparidades entre as imagens capturadas e permitem recuperar a informação de profundidade perdida na projeção do ambiente em uma imagem 2D [13, 37]. Ou, ainda, sistemas monoculares que procuram encontrar e explorar características específicas do ambiente. Muitas vezes, marcadores artificiais ou naturais são utilizados para guiar, indicar e, em alguns casos, até controlar o posicionamento do robô à medida que são detectados durante o movimento [56, 60]. Alguns sistemas se utilizam do

fluxo óptico ou campo de movimento para definir a navegação do robô [52].

O trabalho desenvolvido nesta dissertação aborda o problema da navegação de robôs móveis em ambientes interiores, mais precisamente, em corredores. Os principais objetivos são:

- **Desenvolver uma estratégia de navegação baseada em um sistema monocular de visão, com imagens em tons de cinza.** A intenção é abordar o problema de uma maneira simples sem realizar a reconstrução 3D do ambiente. O sistema de visão contará com apenas uma câmara para observar e explorar as características geométricas dos ambientes de corredores, cujas imagens fornecerão as informações necessárias para o controle do robô. Isso permitirá que o sistema tenha uma resposta rápida e robusta, permitindo uma maior estabilidade na navegação.
- **Manter o robô no centro do corredor durante o seu deslocamento.** A estratégica de navegação deverá promover o controle de posição e orientação do robô de tal forma que este se mantenha centralizado no corredor à medida que se movimenta. A idéia básica é usar o ponto de fuga obtido a partir da projeção das retas do corredor no plano da imagem [40] e a assimetria existente entre estas retas para definir o controle.
- **Monitorar o processo de navegação com possibilidade de definir pontos ou locais para a execução de tarefas.** O sistema deverá ser capaz de identificar a posição atual do robô com relação a um percurso total pré-definido. Desta forma, será possível especificar pontos onde tarefas deverão ser executadas, aumentando-se, assim, a autonomia do sistema.
- **Unir dois paradigmas diferentes para definir a estratégia de navegação no interior de corredores, explorando as vantagens que cada um apresenta.** Os dois paradigmas empregados serão: o método de servo controle baseado em visão e método baseado em aparências. A união desses dois paradigmas se baseia no fato de o método de servo controle permitir explorar de uma maneira simples a geometria e a estrutura do ambiente de navegação, enquanto que o método baseado em aparências possibilita monitorar a localização do robô e ainda definir locais para a execução de tarefas mais gerais ou especiais, através de uma representação interna não métrica do ambiente.

Quando características do ambiente são extraídas e utilizadas para definir sinais de controle nos atuadores e motores do robô, pode-se dizer que se tem o método de servo controle [18, 30]. Caso as imagens sejam utilizadas para comparação com outras de referência e, a partir deste resultado, o controle ou localização são definidos, tem-se o método de aparência [3, 46].

O servo controle baseado no ponto de fuga será o responsável por garantir o controle de orientação e posição do robô através do controle das velocidades angular e linear do mesmo. Por sua vez, o método baseado em aparências terá como objetivo definir uma descrição interna e não métrica do ambiente, a qual será usada como referência para monitorar o percurso desenvolvido pelo robô e definir o desempenho de tarefas. O próximo capítulo apresentará uma discussão mais detalhada a respeito destes dois métodos adotados.

1.1 Trabalhos Similares

Trabalhos anteriores já abordaram o problema da navegação em interiores ou corredores explorando caracteríticas encontradas nestes ambientes, geralmente estruturados. Há ainda uma série de trabalhos em ambientes externos onde retas traçadas no chão foram usadas para guiar o movimento de robôs móveis. Geralmente, nesses casos, adotou-se um ou outro dos dois paradigmas citados anteriormente.

No trabalho apresentado em [10], o contorno das retas encontradas em uma autoestrada é utilizado como base do sistema de navegação de um automóvel. Este sistema foi denominado de *GOLD - Generic Obstacle and Lane Detection* e apresentado por Bertozzi e Broggi [9]. Nesse caso, princípios de fluxo óptico e mapeamento perspectivo inverso são aplicados. Também no trabalho apresentado por Schneiderman e Nashman [54], o contorno das retas de uma estrada é detectado e usado para a atualização de um modelo usado para guiar um automóvel.

Já para o caso de interiores, marcadores, imagens e contornos do ambiente são utilizados para definir as variações a serem aplicadas na orientação do robô. No trabalho apresentado em [21], são detectadas as retas de um corredor para a definição do movimento; em [31], os contornos e a variação na escala de objetos são utilizados; em [37], os contornos verticais presentes em um corredor são explorados por um sistema estéreo de visão, onde, além do controle dos motores, o mapeamento do ambiente é realizado. Interessantes, ainda, são o sistema e a representação utilizados em [43], denominado de *VSRR - View-sequenced*

Route Representation, um método baseado em aparência aplicado no interior de corredores.

Diferente dos trabalhos citados acima, o trabalho desenvolvido nesta dissertação procurou unir dois paradigmas, explorando de cada um as suas vantagens e facilidades. Isso possibilitou desenvolver uma estratégia de navegação mais simples, rápida, robusta e também mais autônoma para o caso de interiores de corredores.

1.2 Estrutura do Trabalho

Esta dissertação está organizada basicamente em 5 capítulos. O Capítulo 1 corresponde a essa Introdução, onde foi abordada a aplicação de visão computacional na róbotica móvel, definido o objetivo deste trabalho e citados alguns trabalhos relacionados com a navegação de robôs móveis em corredores. No Capítulo 2 são apresentados os princípios e as características dos dois paradigmas utilizados na estratégia de navegação. Nesse capítulo são, ainda, ressaltadas quais características do ambiente foram exploradas e extraídas a partir de imagens para a solução do problema proposto. A seguir, o Capítulo 3 mostra como foi implementada a estratégia de navegação através da descrição dos procedimentos envolvidos e da definição do controle do robô. A maneira como cada paradigma foi aplicado é apresentada separadamente e a validade da sua utilização é afirmada através de simulações. Depois disto, no Capítulo 4, o equipamento utilizado e os resultados experimentais são apresentados. Finalmente, o Capítulo 5 traz uma discussão do que os resultados obtidos representam e as conclusões que se pode obter deste trabalho. Além disto, são sugeridas idéias para trabalhos futuros que poderão dar seqüência ao trabalho apresentado nesta dissertação.

Capítulo 2

Os Dois Paradigmas de Navegação

Aos cinqüenta ou aos dezesseis anos independentemente,
existe no coração de todo ser humano o amor pelo prodigioso,
a doce contemplação pelas estrelas e pelas coisas
e pensamentos radiantes,
o imprescrutável desafio dos acontecimentos,
o incessante apetite infantil pelo que virá depois,
e a alegria e o jogo da vida.

(S. Ullmee)

Neste capítulo serão apresentados os dois paradigmas de navegação utilizados neste trabalho de pesquisa: o método de servo controle baseado em visão e o método baseado em aparências. Além de seus conceitos e definições, será discutida a abordagem adotada para estes métodos no caso da navegação de robôs móveis em corredores.

O método de servo controle será analisado na secção 2.1, assim como os princípios de geometria projetiva envolvidos, o modelo da câmara utilizado e as transformações entre os diferentes referenciais envolvidos.

Já o método de aparências será abordado na secção 2.2, contando com uma discussão sobre a utilização de marcadores naturais ou artificiais para a construção de representações do ambiente sob a forma de grafos ou mapas topológicos.

2.1 O Método de Servo Controle Baseado em Visão

O sensoriamento por visão tem sido amplamente aplicado em sistemas de controle para robôs manipuladores e robôs móveis. Muitas vezes, o sensoriamento por visão é utilizado em malhas abertas e mais externas de controle, ou seja, primeiro "ver" e depois "mover". Uma alternativa para se melhorar a precisão e o desempenho desses sistemas de controle é usar a visão em malhas de realimentação mais internas, influenciando diretamente no controle de motores e atuadores do robô. Isso é conhecido como Servo Controle baseado em visão [30].

Portanto, no caso de navegação, servo controle baseado em visão pode ser entendido como a ação de "controlar" a posição do robô com relação ao ambiente, usando as informações fornecidas pelo sistema de visão, ao invés de simplesmente "observar" o ambiente. Essas informações são, então, utilizadas como sinais de entrada em malhas mais internas do sistema de controle, contribuindo diretamente para o controle motor.

A estratégia consiste em explorar a estrutura do ambiente de navegação de tal forma que se possa extrair do mesmo as características que podem ser utilizadas na solução do problema de navegação ou posicionamento do robô. Essa abordagem mais direta torna o problema a ser resolvido mais simples e dá mais ênfase aos pontos realmente relevantes. Sendo mais simples, o problema pode ser resolvido com um sistema de servo controle baseado nas informações coletadas, que seja simples o suficiente para trabalhar na freqüência desejada e de maneira mais robusta, imune às incertezas provenientes dos sensores e do ambiente [1, 18].

Quando a intenção é manter um objeto no centro do corredor, geralmente o que se faz é observar as retas ou as paredes do corredor e tentar posicionar o objeto a uma mesma distância dessas retas ou paredes. Considere, agora, um robô móvel com um sistema monocular, ou seja, uma câmara, sempre apontando para a frente no sentido do movimento e centralizada em relação ao robô. No espaço 3D, retas que são paralelas se encontram em um ponto no infinito. Devido à projeção perspectiva, em uma imagem 2D, as projeções dessas retas se encontram em um ponto chamado "ponto de fuga". Manter o ponto de fuga no centro da imagem e as projeções das retas o mais simétricas possível, significa manter o robô no centro do corredor durante o seu movimento. A posição do ponto de fuga na imagem não depende da translação, mas apenas da rotação da câmara. Já as inclinações das retas projetadas variam com a translação da câmara na direção perpendicular às retas no mundo

real. Portanto, uma maneira mais direta e simples de encarar o problema é analisar a projeção das retas do corredor no plano da imagem ao invés de tentar uma reconstrução 3D do ambiente.

Logo, o sistema de servo controle implementado deve atuar no movimento do robô de tal forma que o ponto de fuga seja sempre mantido no centro da imagem. Isso é conseguido através do controle de posição e orientação do robô, a partir das velocidades angular e linear do mesmo [57].

2.1.1 Geometria Projetiva e Modelo da Câmara

Para se controlar a navegação de um robô móvel, a partir das informações obtidas por uma câmara, é necessário se conhecer os aspectos geométricos da projeção do mundo real 3D em uma imagem 2D [19]. Essa projeção elimina a informação de profundidade existente no espaço tridimensional.

O modelo de câmara mais comumente utilizado em visão computacional é o modelo pontual - *pinhole model*. Esse modelo se baseia na passagem dos raios luminosos por um pequeno orifício na entrada da câmara e na sua projeção em uma superfície plana, como mostrado na Figura 2.1. As câmaras baseadas nesse princípio são chamadas câmaras perspectivas. Esta representação é uma maneira simples e funcional de se modelar as câmaras CCD (*Charge Coupled Device*).

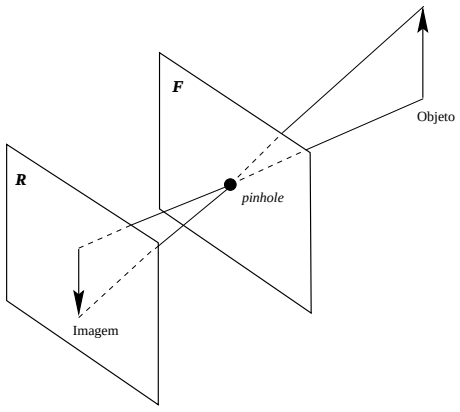

Figura 2.1: Formação da imagem em uma câmara perspectiva.

Considere o modelo geométrico para uma câmara perspectiva definido pela Figura 2.2. O plano $\mathcal{R}$ é chamado de plano da imagem ou plano de retina, no qual se forma

a imagem através da projeção por perspectiva. O ponto C é o centro óptico, localizado a uma distância f do plano $\mathcal{R}$, correspondente à distância focal. Através da projeção perspectiva, a imagem m de um ponto M no mundo 3D é formada através da interseção da reta $< C, M >$ com o plano $\mathcal{R}$.

A reta que passa pelo centro óptico C e é perpendicular ao plano da imagem $\mathcal{R}$ é chamada de eixo óptico. O eixo óptico intercepta o plano da imagem no ponto principal c. O plano $\mathcal{F}$ que contém C e é paralelo ao plano $\mathcal{R}$ é conhecido como plano focal. Os pontos M localizados no plano focal não possuem imagem no plano de retina, pois as retas $< C, M >$ formadas são sempre paralelas a este plano e, portanto, não o interceptam em ponto algum. Estas retas interceptam o plano $\mathcal{R}$ apenas em pontos no infinito. Pontos no infinito, embora especiais e de difícil consideração em geometria afim ou euclidiana, podem ser facilmente manipulados em geometria projetiva.

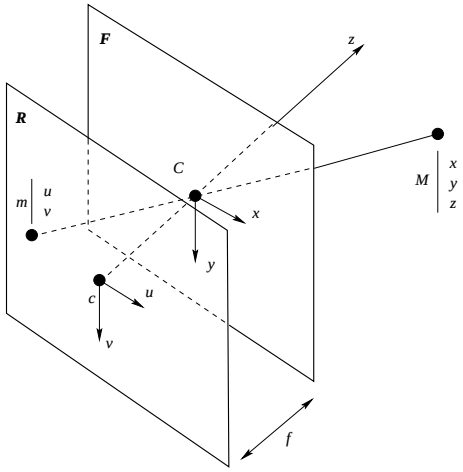

Figura 2.2: Modelo da câmara perspectiva. O plano focal $\mathcal{F}$ é paralelo ao plano da imagem $\mathcal{R}$, e a distância f corresponde à distância focal.

Considere os sistemas de coordenadas da câmara $\{C\}$ e do plano da imagem $\{I\}$ definidos conforme a Figura 2.2. O sistema de coordenadas da câmara $\{C\}$ tem como origem o ponto C e o seu eixo z ao longo do eixo óptico, enquanto que o sistema de coordenadas da imagem $\{I\}$ tem como origem o ponto principal c.

Um ponto M com as coordenadas representadas no referencial da câmara (x, y, z), será projetado no plano da imagem no ponto m com coordenadas (u, v). É fácil mostrar que, por semelhança de triângulos, a relação entre as coordenadas de M e m nos dois referenciais pode ser escrita por:

$$u = -\frac{f\,x}{z} \qquad v = -\frac{f\,y}{z} \tag{2.1}$$

Isto pode ser escrito como uma relação linear através de coordenadas homogêneas:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.2}$$

Os pontos no espaço 3D com $z = 0$, correspondem a pontos no plano focal. As coordenadas u e v das suas projeções não poderão ser definidas. Pode-se entender esta idéia através de um ponto M inicialmente localizado fora do plano focal que sofre continuamente uma aproximação a este plano. Os pontos com $z = 0$ e, portanto, $\lambda = 0$, são chamados de pontos no infinito no plano da imagem. A expressão $\lambda = 0$ é a equação da linha no infinito no plano da imagem e corresponde à imagem do plano focal.

A equação (2.2) é projetiva e está definida a não ser pelo fator de escala λ. Esta equação pode ser reescrita utilizando-se coordenadas projetivas e, portanto, pode ser interpretada como uma transformação linear entre o espaço projetivo $\mathcal{P}^3$ e o plano projetivo $\mathcal{P}^2$:

$$\begin{bmatrix} U \\ V \\ S \end{bmatrix} = \begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ T \end{bmatrix} \tag{2.3}$$

Sua representação na forma matricial é dada por:

$$\widetilde{\mathbf{m}} = \widetilde{\mathbf{P}}\widetilde{\mathbf{M}} \tag{2.4}$$

onde $\widetilde{\mathbf{m}}$ e $\widetilde{\mathbf{M}}$ são as coordenadas projetivas dos pontos m e M, e $\widetilde{\mathbf{P}}$ é conhecida como *matriz de projeção perspectiva.*

É importante notar que o uso da geometria projetiva permite que a projeção por perspectiva seja descrita por uma equação linear, o que simplifica bastante a abordagem, pois ao invés de se trabalhar com equações não lineares, como a equação (2.1), usa-se uma relação linear (2.4). Uma câmara, portanto, pode ser vista como sendo um sistema capaz de realizar uma transformação linear do espaço projetivo $\mathcal{P}^3$ para o plano projetivo $\mathcal{P}^2$.

Normalmente, a origem do referencial da imagem não é o ponto principal, mas sim o canto superior esquerdo do plano da imagem. Além disto, em alguns casos, há diferença entre as escalas dos eixos u e v devido a características eletrônicas do processo de aquisição nas câmaras CCD.

Portanto, considere a mudança de coordenadas mostrada na Figura 2.3.

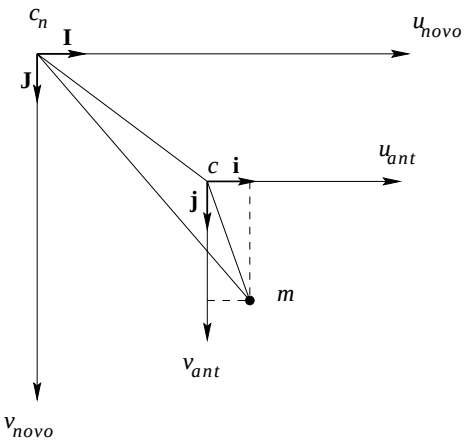

Figura 2.3: Mudança de coordenadas no plano da imagem.

Para um pixel m tem-se:

$$c_n m = c_n c + cm \tag{2.5}$$

Escrevendo $cm = u_{ant}\mathbf{i} + v_{ant}\mathbf{j}$ no referencial antigo e considerando a mudança de escala entre os dois referenciais $(\mathbf{i}, \mathbf{j})$ e $(\mathbf{I}, \mathbf{J})$, tem-se:

$$\mathbf{i} = \mathbf{sI} \qquad \mathbf{j} = \mathbf{sJ} \qquad \text{com} \qquad \mathbf{s} = \begin{bmatrix} k_u & 0 \\ 0 & k_v \end{bmatrix} \tag{2.6}$$

Substituindo $c_n c$ por $\mathbf{t}$ no novo sistema de coordenadas, pode-se reescrever a equação (2.5) em coordenadas projetivas como:

$$\widetilde{\mathbf{m}}_{\mathbf{novo}} = \widetilde{\mathbf{H}}\widetilde{\mathbf{m}}_{\mathbf{ant}} \qquad \text{onde} \qquad \widetilde{\mathbf{H}} = \begin{bmatrix} \mathbf{s} & \mathbf{t} \\ 0_2^T & 1 \end{bmatrix} \tag{2.7}$$

A partir da equação (2.4), tem-se:

$$\widetilde{\mathbf{m}}_{\mathbf{ant}} = \widetilde{\mathbf{P}}_{\mathbf{ant}}\widetilde{\mathbf{M}}$$

Substituindo $\widetilde{\mathbf{m}}_{\mathbf{ant}}$ em (2.7):

$$\widetilde{\mathbf{m}}_{\mathbf{novo}} = \widetilde{\mathbf{H}}\widetilde{\mathbf{P}}_{\mathbf{ant}}\widetilde{\mathbf{M}}$$

Para o novo referencial, tem-se:

$$\widetilde{\mathbf{P}}_{\mathbf{novo}} = \widetilde{\mathbf{H}}\widetilde{\mathbf{P}}_{\mathbf{ant}} \tag{2.8}$$

Representando as coordenadas de $\mathbf{t}$ por u_0 e v_0 e de acordo com as equações (2.2) e (2.8), pode-se escrever a matriz $\widetilde{\mathbf{P}}$ de forma mais geral:

$$\widetilde{\mathbf{P}} = \begin{bmatrix} -fk_u & 0 & u_0 & 0 \\ 0 & -fk_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.9}$$

Os parâmetros que aparecem na expressão acima não dependem da posição e orientação da câmara no espaço e por isso são chamados de parâmetros intrínsecos.

Para simplificar a expressão, considere $S_u = -fk_u$ e $S_v = -fk_v$. E para especificar a relação entre as coordenadas de um ponto 3D expresso no referencial da câmara e as coordenadas da sua projeção na imagem, dada pela matriz de projeção perspectiva $\widetilde{\mathbf{P}}$, será adotada a seguinte representação:

$${}^{I}T_C = \begin{bmatrix} S_u & 0 & u_0 & 0 \\ 0 & S_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.10}$$

2.1.2 Transformação entre os Sistemas de Coordenadas Envolvidos

Estando definida a relação entre os referenciais da câmara e da imagem, resta agora definir como ocorrem as transformações entre os demais referenciais envolvidos no problema da navegação do robô no interior de corredores. A intenção é definir, ao final, uma matriz de transformação capaz de relacionar os pontos no espaço 3D do mundo real com as suas projeções na imagem 2D formada pela câmara.

Para isto, suponha dois referenciais diferentes no mundo, $\{R_0\}$ e $\{R_1\}$, e considere P_0 e P_1 as coordenadas homogêneas (ou projetivas) de um mesmo ponto nos dois referenciais.

Sabe-se que, para o caso de movimentos rígidos, a transformação entre os dois referenciais pode ser definida através de translações e rotações nos diferentes eixos representadas pelo conjunto básico de transformações:

$$Trans_{x,a} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} ; \; Trans_{y,b} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} ; \; Trans_{z,c} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

para translações nos eixos, e

$$Rot_{x,\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_\alpha & -s_\alpha & 0 \\ 0 & s_\alpha & c_\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} ; \; Rot_{y,\phi} = \begin{bmatrix} c_\phi & 0 & s_\phi & 0 \\ 0 & 1 & 0 & 0 \\ -s_\phi & 0 & c_\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} ; \; Rot_{z,\theta} = \begin{bmatrix} c_\theta & -s_\theta & 0 & 0 \\ s_\theta & c_\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

para as rotações nos eixos[1].

A transformação mais geral representada a seguir pela equação (2.11) é então conseguida multiplicando-se as matrizes básicas, na ordem em que ocorrem as transformações, quando se muda de um referencial para o outro.

$$P_0 \; = \; {}^0T_1 \, P_1 \qquad \text{com } {}^0T_1 \; = \; \begin{bmatrix} \mathbf{R} & \mathbf{d} \\ 0_3^T & 1 \end{bmatrix} \tag{2.11}$$

onde **R** representa o conjunto de rotações sofridas e **d** corresponde à translação entre os dois referenciais, expressa em coordenadas do referencial de origem $\{R_0\}$. A demonstração das definições feitas acima pode ser encontrada em [55].

2.1.3 Transformação do Referencial do Mundo para o do Robô

Agora, para o caso da navegação em corredores, considere o referencial do mundo fixo no canto esquerdo no início do corredor, como mostrado na Figura 2.4. O robô possui 3 graus de liberdade definidos pela sua posição e orientação com relação ao referencial do mundo, ou seja, possui liberdade de movimento nas direções dos eixos X e Y, e ainda

[1] Usa-se (s_α, c_α) para representar $(\sin\alpha, \cos\alpha)$ para simplificar a notação

a possibilidade de rotação em torno do eixo Z. A direção Y é tomada como a direção do movimento no sentido do corredor e a diferença angular θ entre os dois referenciais corresponde à orientação do robô.

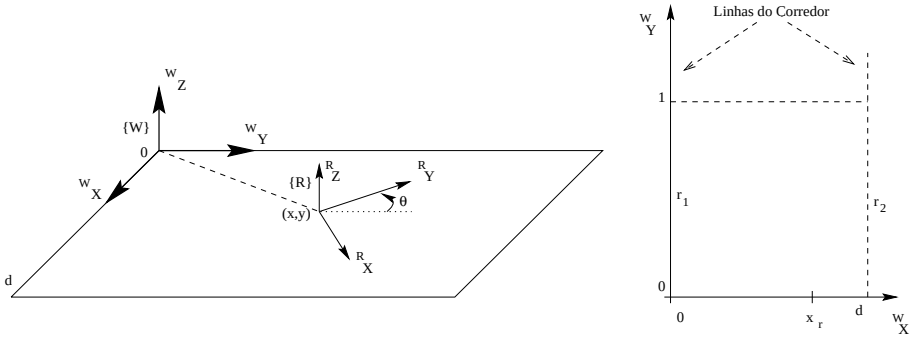

Figura 2.4: À esquerda: Referenciais do mundo e do robô. O robô apresenta 3 graus de liberdade com relação ao referencial do mundo. À direita: Representação das linhas do corredor.

Como mostrado na Figura 2.4, as linhas do corredor podem ser definidas através dos pontos:

$$\begin{aligned} &\text{reta } r_1\text{:} \quad (x_1, y_1) = (0, 0); \quad (x_2, y_2) = (0, 1) \\ &\text{reta } r_2\text{:} \quad (x_3, y_3) = (d, 0); \quad (x_4, y_4) = (d, 1) \end{aligned} \tag{2.12}$$

onde d representa a largura do corredor.

À medida que o robô se desloca no plano $X\,Y$, a mudança entre os referenciais do mundo e do robô pode ser descrita através de uma rotação do eixo ${}^{W}Z$ por um ângulo θ e por uma translação da origem dada por (x_r, y_r), onde θ representa a orientação do robô e ${}^{W}P_{OR}$ será a posição da origem do referencial do robô $\{R\}$ no referencial do mundo $\{W\}$. Portanto, a transformação entre o referencial do mundo e o referencial do robô pode ser representada por:

$${}^{W}P = {}^{W}T_R \, {}^{R}P \qquad \text{com} \qquad {}^{W}T_R = \begin{bmatrix} c_\theta & -s_\theta & 0 & x_r \\ s_\theta & c_\theta & 0 & y_r \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.13}$$

Essa matriz ${}^{W}T_{R}$ permite conhecer as coordenadas de um ponto no mundo ${}^{W}P$ a partir das suas coordenadas ${}^{R}P$ no referencial do robô. Logo, para o processo contrário, deve-se inverter a expressão acima, obtendo-se a equação:

$$
\begin{aligned}
{}^{R}P &= {}^{R}T_{W}\,{}^{W}P \\
\text{com} \qquad {}^{R}T_{W} &= {}^{W}T_{R}{}^{-1} = \begin{bmatrix} c_\theta & s_\theta & 0 & -x_r c_\theta - y_r s_\theta \\ -s_\theta & c_\theta & 0 & x_r s_\theta - y_r c_\theta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{2.14}
$$

2.1.4 Transformação do Referencial do Robô para o da Câmara

Considere, agora, os sistemas de coordenadas da câmara $\{C\}$ e do robô $\{R\}$, conforme mostrado na Figura 2.5.

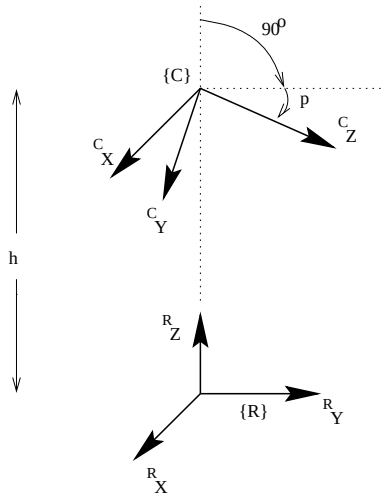

Figura 2.5: A mudança entre os referenciais do robô e da câmara pode ser descrita por uma rotação no eixo x da câmara - *pitch*, p, e uma translação vertical h.

Para se passar do referencial do robô para o da câmara, o primeiro sofreu uma translação h no eixo ${}^{R}Z$ e uma rotação negativa de um ângulo $(90^o + p)$ no eixo ${}^{R}X$. O ângulo p é conhecido por *pitch-angle*. A matriz de transformação homogênea entre os dois referenciais partindo-se do referencial do robô pode ser escrita como:

$$
{}^{R}T_{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{-(90^o+p)} & -s_{-(90^o+p)} & 0 \\ 0 & s_{-(90^o+p)} & c_{-(90^o+p)} & h \\ 0 & 0 & 0 & 1 \end{bmatrix} =
$$

$$
= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -s_p & c_p & 0 \\ 0 & -c_p & -s_p & h \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.15}
$$

Como no caso anterior, para se obter as coordenadas de um ponto no referencial da câmara ${}^{C}P$ a partir das suas coordenadas no referencial do robô ${}^{R}P$, inverte-se a matriz definida em (2.15).

$$
{}^{C}T_{R} = {}^{R}T_{C}{}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -s_p & -c_p & hc_p \\ 0 & c_p & -s_p & hs_p \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.16}
$$

2.1.5 Transformação do Referencial do Mundo para o da Imagem

Com as transformações vistas até o momento, é possível se converter as coordenadas de um ponto representado no referencial do mundo para as suas coordenadas no referencial da câmara. Falta, porém, incluir a última relação, a projeção do ponto no plano da imagem. Essa transformação já foi definida na subsecção 2.1.1 pela equação (2.10), que se repete aqui por conveniência:

$$
{}^{I}T_{C} = \begin{bmatrix} S_u & 0 & u_0 & 0 \\ 0 & S_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}
$$

onde S_u, S_v, u_o e v_o são os parâmetros intrínsecos da câmara.

Uma vez estando definidas todas as transformações citadas anteriormente, pode-se definir a matriz de transformação que relaciona as coordenadas (${}^{w}x$, ${}^{w}y$, ${}^{w}z$) de um ponto

${}^{W}P$ qualquer do espaço 3D no referencial do mundo com as coordenadas (${}^{i}x$, ${}^{i}y$) de sua projeção no plano da imagem:

$$\begin{bmatrix} \lambda\, {}^{i}x \\ \lambda\, {}^{i}y \\ \lambda \end{bmatrix} = {}^{I}T_{W} \begin{bmatrix} {}^{w}x \\ {}^{w}y \\ {}^{w}z \\ 1 \end{bmatrix} \qquad \text{com} \qquad {}^{I}T_{W} = {}^{I}T_{C}\, {}^{C}T_{R}\, {}^{R}T_{W}$$

Resolvendo para ${}^{I}T_{W}$:

$${}^{I}T_{W} = \begin{bmatrix} S_u\, c_\theta - u_0\, c_p s_\theta & S_u\, s_\theta + u_0\, c_p c_\theta & -u_0\, s_p \\ -\left(-S_v\, s_p + v_0\, c_p\right) s_\theta & \left(-S_v\, s_p + v_0\, c_p\right) c_\theta & -S_v\, c_p - v_0\, s_p \\ -c_p s_\theta & c_p c_\theta & -s_p \end{bmatrix}$$

$$\begin{bmatrix} S_u\, \left(-x_r\, c_\theta - y_r\, s_\theta\right) + u_0\, c_p\, \left(x_r\, s_\theta - y_r\, c_\theta\right) + u_0\, h s_p \\ \left(-S_v\, s_p + v_0\, c_p\right) \left(x_r\, s_\theta - y_r\, c\theta\right) + S_v\, h c_p + v_0\, h s_p \\ c_p\, \left(x_r\, s_\theta - y_r\, c_\theta\right) + h s_p \end{bmatrix} \tag{2.17}$$

A partir da relação entre os referenciais do mundo e do plano da imagem, será possível determinar como a posição e orientação do robô se relacionam com as informações obtidas nas imagens do ambiente. Como a intenção é controlar a posição e a orientação, é necessário utilizar mais de um sinal extraído dessas informações. A partir das imagens do ambiente, é possível se obter informações do ponto de fuga e dos parâmetros das retas do corredor. Isto servirá de base para definir o algoritmo de servo controle para a navegação do robô.

Sem perda de generalidade, pode-se considerar o sistema de coordenadas da imagem centralizado, ou seja, $(u_0, v_0) = (0, 0)$ e desprezar a coordenada y_r da origem do referencial do robô, adotando $y_r = 0$. Isto se deve ao fato de que y_r, que representa uma translação no sentido do corredor, é irrelevante para a análise das mudanças que ocorrem com o ponto de fuga na imagem quando o robô muda de posição. Apenas a coordenada x_r exerce influência nesse caso. Desta forma, a matriz ${}^{I}T_{W}$ pode ser simplificada para:

$$ {}^{I}T_{W} = \begin{bmatrix} S_u c_\theta & S_u s_\theta & 0 & -S_u c_\theta x_r \\ S_v s_\theta s_p & -S_v s_p c_\theta & -S_v c_p & S_v(s_p s_\theta x_r - h c_p) \\ -s_\theta c_p & c_p c_\theta & -s_p & c_p s_\theta x_r + h s_p \end{bmatrix} \tag{2.18} $$

Uma vez que o robô se movimenta em uma superfície plana e que a posição vertical da câmara se mantém fixa, apenas os pontos no plano XY são interessantes. Portanto, considerando ${}^{w}z = 0$, pode-se definir a homografia representada pela matriz ${}^{I}H_{W}$, relacionando os pontos do plano do chão com as suas projeções na imagem:

$$ \begin{bmatrix} \lambda\, {}^{i}x \\ \lambda\, {}^{i}y \\ \lambda \end{bmatrix} = {}^{I}H_{W} \begin{bmatrix} {}^{w}x \\ {}^{w}y \\ 1 \end{bmatrix} $$

$$ \text{com} \qquad {}^{I}H_{W} = \begin{bmatrix} S_u c_\theta & S_u s_\theta & -S_u c_\theta x_r \\ S_v s_\theta s_p & -S_v s_p c_\theta & S_v(s_p s_\theta x_r - h c_p) \\ -s_\theta c_p & c_p c_\theta & c_p s_\theta x_r + h s_p \end{bmatrix} \tag{2.19} $$

A transformação definida por ${}^{I}H_{W}$ depende da orientação e da posição do robô (θ, x_r), dos parâmetros intrínsecos da câmara, que definem a sua geometria interna, e dos seus parâmetros extrínsecos, que definem a sua orientação externa (ângulos e vetores de translação da câmara).

2.1.6 A Influência das Retas do Corredor na Posição e na Orientação do Robô

As projeções das retas do corredor podem ser determinadas a partir das suas equações e da matriz ${}^{I}H_{W}$. Sabe-se que, no plano projetivo $\mathcal{P}^2$, as coordenadas projetivas de uma reta podem ser definidas através do produto vetorial de dois pontos pertencentes a ela. A prova disto pode ser vista em [19]. Portanto, para o caso das retas do corredor $\tilde{r}_1$ e $\tilde{r}_2$, no plano do chão, tem-se:

$$ \tilde{r}_1 \sim \tilde{u}_1 \times \tilde{u}_2 = [1 \;\; 0 \;\; 0]^T $$

$$ \tilde{r}_2 \sim \tilde{u}_3 \times \tilde{u}_4 = [1 \;\; 0 \;\; d]^T $$

onde $\sim$ representa uma igualdade a não ser por um fator de escala, e $\tilde{u}_1$ a $\tilde{u}_4$ são as coordenadas homogêneas no plano XY dos pontos definidos em (2.12).

Da mesma forma que uma reta em $\mathcal{P}^2$ pode ser definida pelo produto vetorial de dois de seus pontos, as coordenadas do ponto de encontro de duas retas podem ser determinadas pelo produto vetorial destas retas. Logo, para as retas paralelas do corredor, o seu ponto de encontro definido no infinito $\tilde{u}_\infty$ é dado por:

$$\tilde{u}_\infty \sim \tilde{r}_1 \times \tilde{r}_2 \sim [0 \;\; 1 \;\; 0]^T$$

A projeção de $\tilde{u}_\infty$ no plano da imagem, define o ponto de fuga $({}^ix_v, {}^iy_v)$ das retas do corredor. Isso é conseguido através da matriz ${}^IH_W$ em (2.19):

$$\begin{aligned} ({}^ix_v, {}^iy_v) &= {}^IH_W \, \tilde{u}_\infty \\ ({}^ix_v, {}^iy_v) &= (S_u \frac{\tan\theta}{c_p}, \;\; -S_v \tan p) \end{aligned} \tag{2.20}$$

A equação (2.20) mostra que a coordenada horizontal ${}^ix_v$ do ponto de fuga depende da orientação do robô θ, da escala de pixels do eixo S_u e do ângulo *pitch* de inclinação p. Anular essa coordenada, representa anular o ângulo θ. Em termos do controle, isto se traduz na intenção de manter o ponto de fuga na coluna central da imagem e, portanto, a orientação do robô paralela ao corredor.

Para se obter informação sobre a posição x_r do robô, será necessário utilizar as projeções das linhas do corredor no plano da imagem. Essa informação será conseguida através das inclinações m_i das retas projetadas.

Para a reta $\tilde{r}_1$, primeiramente projetam-se os pontos $\tilde{u}_1$ e $\tilde{u}_2$ através do produto da homografia ${}^IH_W$ com as suas coordenadas homogêneas no plano do chão:

$${}^i\tilde{u}_1 = {}^IH_W \, [0 \;\; 0 \;\; 1]^T \qquad \text{e} \qquad {}^i\tilde{u}_2 = {}^IH_W \, [0 \;\; 1 \;\; 1]^T$$

A reta $\tilde{r}_1$ projetada será o produto vetorial dos seus pontos:

$${}^i\tilde{r}_1 \sim {}^i\tilde{u}_1 \times {}^i\tilde{u}_2 = [A \;\; B \;\; C]^T$$

Escrevendo a equação para a reta na imagem:

$$A\,x + B\,y + C = 0 \qquad \text{ou} \qquad y = m_1\,x + b_1$$

Sendo comum, no caso de imagens, referenciar colunas em função de linhas, vale a pena, ainda, ressaltar a representação[2]:

$$x = m_1^{'}\, y + b_1^{'} \qquad \text{onde } x = coluna \text{ e } y = linha$$

Separando m_1:

$$y \;=\; -\,\frac{A}{B}\,x \;-\; \frac{C}{B} \qquad \Rightarrow \qquad m_1 \;=\; -\frac{A}{B} \qquad \text{onde} \qquad m_1^{'} \;=\; \frac{1}{m_1}$$

Que resulta em:

$$m_1 \;=\; -\frac{S_v}{S_u}\frac{h\, c_\theta}{(c_p x_r \;+\; h\, s_p s_\theta)}$$

De maneira semelhante, para a reta $\tilde{r}_2$:

$${}^{i}\tilde{u}_3 \;=\; {}^{I}H_W\,[\,d\;\;0\;\;1\,]^T \qquad \text{e} \qquad {}^{i}\tilde{u}_4 \;=\; {}^{I}H_W\,[\,d\;\;1\;\;1\,]^T$$

$${}^{i}\tilde{r}_2 \;\sim\; {}^{i}\tilde{u}_3 \times {}^{i}\tilde{u}_4 \;=\; [\,D\;\;E\;\;F\,]^T$$

$$D\,x + E\,y + F = 0 \qquad \text{ou} \qquad y \;=\; m_2\,x + b_2$$

$$\text{e ainda} \qquad x \;=\; m_2^{'}\,y + b_2^{'}$$

Separando m_2:

$$y \;=\; -\,\frac{D}{E}\,x \;-\; \frac{F}{D} \qquad \Rightarrow \qquad m_2 \;=\; -\frac{D}{E} \qquad \text{onde} \qquad m_2^{'} \;=\; \frac{1}{m_2}$$

Que resulta em:

$$m_2 \;=\; -\frac{S_v}{S_u}\,\frac{h\, c_\theta}{(c_p x_r \;-\; c_p d \;+\; h\, s_p s_\theta)}$$

[2]Esse tipo de representação será empregado mais adiante neste trabalho

Definindo δ_x como a soma dos inversos de m_1 e m_2, tem-se:

$$\begin{aligned}
\delta_x &= \left(\frac{1}{m_1} + \frac{1}{m_2}\right) = \left(m_1^{'} + m_2^{'}\right) \\
&= -\frac{S_u}{S_v}\frac{c_p(2x_r - d)}{hc_\theta} - 2\,\frac{S_u}{S_v}\tan\theta s_p \\
&= -\frac{S_u}{S_v}\frac{c_p(2x_r - d)}{hc_\theta} - \frac{{}^{i}x_v\ 2\ s_p c_p}{S_v} \\
&= -\frac{S_u}{S_v}\frac{c_p(2x_r - d)}{hc_\theta} - \frac{{}^{i}x_v \sin 2p}{S_v}
\end{aligned} \tag{2.21}$$

A equação (2.21) mostra que o sinal δ_x, o qual pode ser extraído a partir da projeção das retas do corredor no plano da imagem, contém informação sobre a posição do robô com relação ao centro do corredor $(2x_r - d)$. Porém, δ_x também depende da orientação do robô. Essa dependência pode ser eliminada se o ângulo p for conhecido. O parâmetro S_v pode ser utilizado para calcular o ângulo p a partir de ${}^{i}y_v$ e, daí, p pode ser usado para calcular θ através de ${}^{i}x_v$ na equação (2.20).

A Figura 2.6 mostra a análise da coordenada ${}^{i}x_v$ do ponto de fuga, quando o robô está centrado no corredor e sofre apenas variação na sua orientação. Da mesma forma, a figura mostra também como δ_x varia quando se mantém a orientação $\theta = 0^o$ e apenas se modifica a posição x_r no corredor (o centro do corredor é dado por $x_r = 0.5$).

Com isso, mostra-se que o sistema de servo controle implementado para o robô deve incluir informações tanto do ponto de fuga quanto dos parâmetros das projeções das retas. Simulações e tentativas de se aplicar apenas o ponto de fuga no controle foram executadas. Porém, quando o robô se encontrava afastado do centro do corredor, o resultado foi um afastamento cada vez maior da posição ideal, apesar de se conseguir corrigir sua orientação. Isso confirma a necessidade de se usar uma combinação de ${}^{i}x_v$ e δ_x no controle da navegação, já que a intenção é de se controlar tanto a posição quanto a orientação do robô.

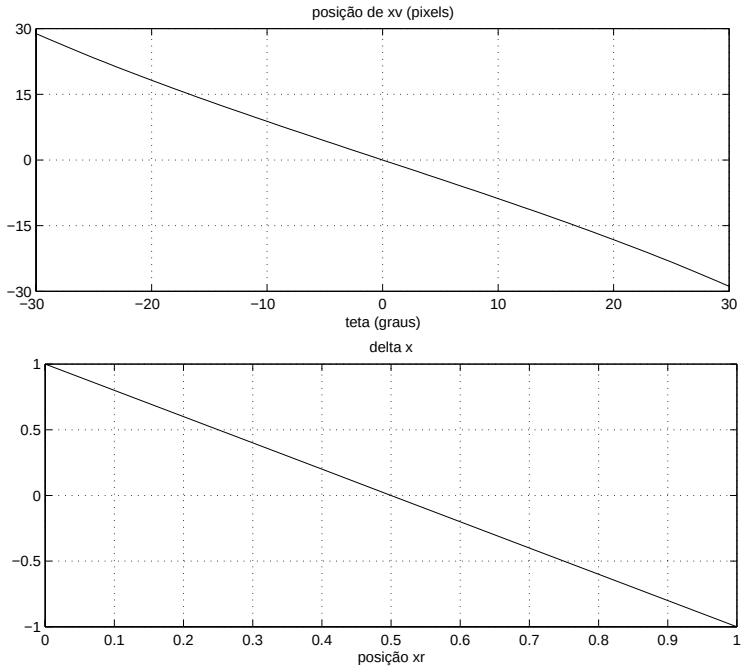

Figura 2.6: Superior: mudanças em ${}^{i}x_v$ quando a orientação θ varia. Inferior: variação de δ_x com a posição x_r.

2.2 O Método Baseado em Aparências

Geralmente, a aplicação de visão computacional no sistema de sensoriamento de um robô não se restringe apenas ao controle motor do processo de navegação ou posicionamento. Existe, ainda, a intenção de se monitorar a posição do robô no ambiente de trabalho. Dessa forma, o robô se torna capaz de executar tarefas ou tomar decisões mais gerais em determinados pontos do percurso, que não estejam exclusivamente ligadas ao controle motor.

Em muitos casos, usam-se métricas baseadas na reconstrução 3D para a obtenção de um modelo do ambiente, onde fica fácil a definição da posição absoluta do robô. Esta opção, porém, tem a desvantagem de ser computacionalmente dispendiosa, como já comentado anteriormente. Em outras configurações, para a definição da posição atual do robô, aplicam-se métricas baseadas em informações provenientes de medições de odometria ou dos próprios motores ou atuadores envolvidos. Porém, também neste caso, sabe-se que a navegação não pode se basear inteiramente nessas medidas por causa de erros cumulativos devido ao deslizamento das rodas e às imprecisões nas informações conseguidas a partir dos sensores de posição desses equipamentos.

Soluções para esse problema já foram propostas e discutidas em outros trabalhos [5, 47], atráves da identificação ou comparação de imagens do ambiente com referências para a confirmação da posição do robô. Uma destas formas de solução consiste na utilização de marcadores artificiais [35, 56] ou de objetos do próprio ambiente como marcadores naturais [3, 11, 43]. Dessa forma, a localização do robô no ambiente é feita de maneira relativa a um ou mais marcadores que se encontrarem presentes no campo visual do robô. Dependendo da estratégia usada para a navegação, a utilização de marcadores para a localização pode ser encarada de duas formas distintas: para confirmar a posição absoluta do robô no ambiente, servindo, assim, como apoio a uma métrica de posicionamento absoluto, ou para definir apenas a posição do robô em relação a um ou mais marcadores, o que para muitos casos já é o suficiente para o sucesso da navegação.

A diferença pode ser explicada com um exemplo simples. Como é o caso deste trabalho, suponha que a tarefa a ser executada por um robô seja percorrer um corredor e virar à direita quando atingir o seu final. Para que isto aconteça, coloca-se um marcador natural ou artificial no final desse corredor. Uma possível estratégia seria: *prosseguir por X metros no corredor, identificar e verificar a posição atual com relação ao marcador.* A outra possibilidade seria: *prosseguir até se atingir uma certa posição em relação ao marcador.* Essa segunda estratégia requer um monitoramento contínuo do progresso do robô no ambiente. Porém, o interessante nesta abordagem é o fato de não se tornar necessário o conhecimento da posição absoluta do robô em um referencial fixo e nem um modelo métrico do ambiente para que a tarefa seja executada com sucesso. O importante é que o robô tenha a habilidade de reconhecer e localizar os marcadores na sua vizinhança, pois não é necessário saber qual a sua posição exata e absoluta em uma sala para que se possa contornar um obstáculo, passar por uma porta ou planejar o deslocamento em um ambiente.

Geralmente, durante a navegação, o reconhecimento dos marcadores do ambiente é realizado através de correlação com imagens de referência previamente armazenadas ou através de busca e identificação de características específicas no caso de marcadores artificiais.

É nessa idéia que se apóia o método de navegação baseado em aparências. O processo é definido através da associação de comandos ou eventos com a aparência da cena capturada pela câmara do robô. Portanto, com essa abordagem, imagens do ambiente são traduzidas em comandos ou eventos de controle ou execução de tarefas durante a navegação. Tornam-se possíveis o monitoramento do processo de navegação, correções de posição e orientação e

ainda a definição de posições ou momentos que devam corresponder à execução de tarefas especiais no ambiente, como por exemplo, entrar em uma sala, procurar e pegar determinado objeto, virar à esquerda, passar pelo lado direito do marcador, parar, etc. Tudo isso definido, considerando-se um posicionamento relativo e qualitativo, através de uma representação não métrica e não necessariamente com grande precisão.

Utilizando-se o método baseado em aparência, pode-se construir um mapa topológico do ambiente, como em [8, 60]. Outra maneira de representação é sob a forma de um grafo, o qual permite, além da localização do robô no ambiente, a aplicação de técnicas de planejamento de trajetórias como em [35, 56]. Em ambos os casos de representação, os nós correspondem aos locais ou pontos identificados por certos marcadores onde o robô deverá executar determinada tarefa ou tomar uma decisão. Os arcos, por sua vez, correspondem às regiões onde são aproveitadas as características do ambiente para se definir e aplicar as estratégias de navegação entre um nó e outro. As Figuras 2.7 e 2.8 mostram um mapa topológico e um grafo de posição como formas de representação de um ambiente de navegação. O mapa topológico traz uma representação do ambiente com a indicação de locais, objetos ou tarefas a serem percorridos ou executados. Já o grafo de posição representa todas as possibilidades de navegação entre um nó e outro com a finalidade de possibilitar o planejamento da trajetória a ser seguida pelo robô. A escolha entre uma e outra representação depende do trabalho a ser realizado pelo robô.

Em alguns trabalhos, há ainda a preocupação de se garantir ao robô a capacidade de escolher de forma autônoma os marcadores a serem utilizados e a executar a atualização dos mapas ou grafos armazenados à medida que navega no ambiente [8, 17, 60].

Para o problema abordado nesta dissertação, o método de aparências aplicado utiliza imagens do ambiente como marcadores naturais. Não são necessários a distribuição e o reconhecimento de marcadores artificiais. Abordagens semelhantes no interior de corredores podem ser vistas em [8, 37, 43]. Todo o processo é desenvolvido apenas com imagens do próprio ambiente de navegação. Nesse caso, imagens de certos lugares são previamente capturadas e memorizadas, formando, assim, um conjunto de referência que corresponde a uma representação não métrica da região de trabalho. Durante o movimento, as imagens obtidas do ambiente são constantemente comparadas com as de referência, através de um processo de correlação. Quando uma dessas imagens de referência é reconhecida, o robô é capaz de identificar qualitativamente a sua posição. O conjunto de imagens de referência é tomado seqüencialmente segundo a tarefa a ser realizada pelo robô. Portanto, durante o

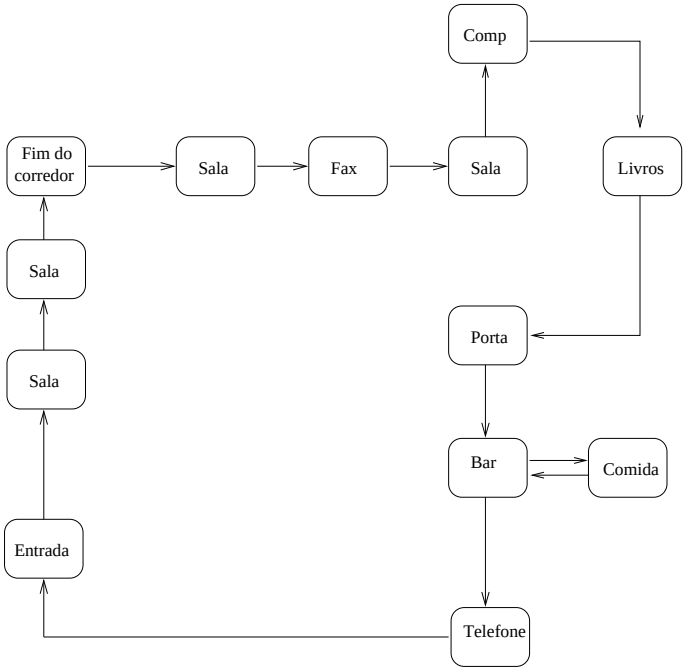

Figura 2.7: Representação de um ambiente de navegação sob a forma de mapa topológico.

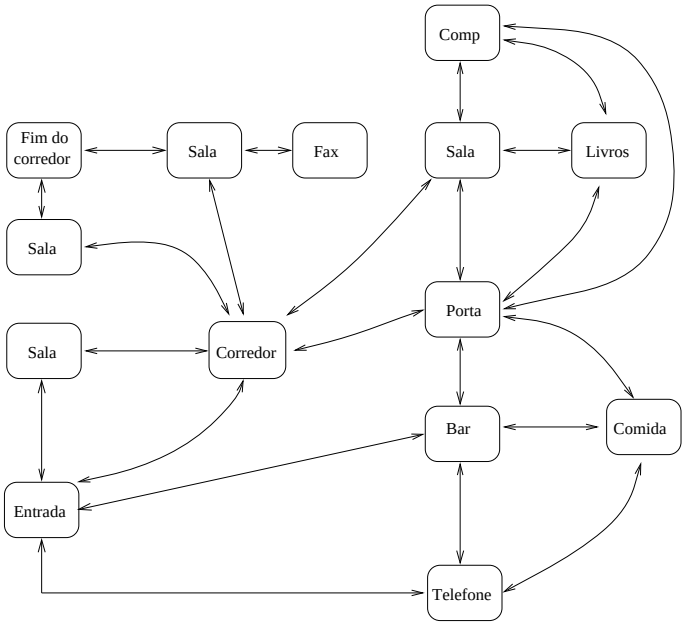

Figura 2.8: Representação de um ambiente de navegação sob a forma de mapa topológico.

movimento, o robô realiza o percurso planejado à medida que compara a imagem obtida do ambiente com as imagens de referência correspondentes à posição atual e a uma ou mais posições seguintes. Dessa forma, além de se reconhecer a posição em que o mesmo se encontra, garante-se a execução da tarefa proposta. O conjunto de imagens de referência pode ser ainda utilizado para a construção de um mapa topológico do ambiente como descrito anteriormente.

Porém, diferente de algumas abordagens anteriores já citadas, em que se aplica apenas o método de aparência para o controle da navegação de um robô, o esquema proposto neste trabalho combina os dois paradigmas apresentados [58].

Por isso, na representação topológica do ambiente construída neste trabalho, os nós correspondem aos pontos de execução de tarefas e tomadas de decisão, enquanto que os arcos representam as regiões em que o método de servo controle também é usado [53]. O método baseado em aparências é constantemente aplicado para o monitoramento do processo como um todo.

Capítulo 3

A Estratégia de Navegação Implementada

Na planta, a inteligência dormita;
no animal, sonha;
só no homem acorda, conhece-se,
possui-se e torna-se consciente.

(Léon Denis)

Neste capítulo será apresentado o sistema de navegação implementado. A estratégia adotada uniu os dois paradigmas de navegação apresentados no capítulo anterior: o servo controle baseado em visão e o método de aparências. Serão descritas as etapas envolvidas na estratégia e a forma como cada um dos dois métodos foi aplicado para a solução do problema da navegação no interior de corredores. A intenção foi explorar as características e vantagens desses métodos para a implementação de um sistema adequado e robusto.

3.1 Descrição do Processo de Navegação

O método de servo controle baseado em visão foi implementado a partir do ponto de fuga das retas do corredor, e o método de aparências se baseia na comparação das imagens capturadas com imagens de referência. O primeiro garante o controle da posição e orientação do robô através das velocidades linear e angular. O ponto de fuga calculado a partir das retas do corredor, as inclinações dessas retas e o desvio padrão do ponto de fuga são usados para definir os parâmetros do controle. Já o método baseado em aparências é aplicado para se monitorar a posição do robô em relação a um percurso pré-definido e determinar o momento de se executar tarefas especiais, como por exemplo, virar para a

direita ou esquerda, entrar em uma sala, pegar um objeto, etc. O percurso a ser cumprido pelo robô foi mapeado por uma seqüência de imagens de referência, previamente adquiridas e armazenadas. Essas imagens fornecem um mapa interno do ambiente de navegação e indicam os locais de execução das tarefas definidas para o robô.

O ciclo do sistema de navegação implementado pode ser descrito da seguinte forma:

1. Aquisição de uma imagem do corredor;
2. Detecção de contornos e definição das equações das retas do corredor;
3. Cálculo do ponto de fuga e do seu erro em relação à coluna central da imagem (ambos em *pixels*);
4. Definição da mudança na velocidade do robô através do servo controle baseado no desvio do ponto de fuga calculado. A variação da velocidade angular é calculada a partir de um controlador PD, e a variação na velocidade linear é definida aplicando-se uma função com características exponenciais;
5. Aplicação do método baseado em aparências para monitorar o progresso do robô em relação ao percurso a ser realizado e determinar se alguma tarefa especial deve ser executada.

Esse procedimento é constantemente repetido enquanto o robô se move pelos corredores. As imagens são capturadas, processadas e utilizadas pelos processos de servo controle e monitoramento baseado no método de aparências. Essas imagens não são armazenadas de forma permanente, sendo descartadas logo após a sua utilização.

3.1.1 Detecção das Retas do Corredor e Cálculo do Ponto de Fuga

Para a detecção dos contornos das retas do corredor, são utilizados os operadores de gradiente de Sobel [28, 32]. Além disto, são adotadas algumas condições na seleção dos pontos de interesse. Determinadas as equações das retas, o ponto de fuga é calculado. Enfoques similares foram descritos em outros trabalhos [9, 10, 54]. O procedimento usado pode ser resumido da seguinte maneira:

1. Aplicação dos filtros de Sobel para as direções X e Y;
2. Seleção dos pontos da imagem resultante que possuem ambos os gradientes nas direções X e Y com valor superior a um limite estabelecido. Dessa maneira, descartam-se os pontos pertencentes a linhas unicamente verticais ou horizontais, selecionando-se apenas os pontos de retas diagonais;
3. Separação dos pontos selecionados em duas imagens binárias. Uma delas corresponde ao conjunto de pontos com gradiente de direção positiva, enquanto a outra contém os pontos de gradiente com direção negativa. Cada grupo corresponde a uma das duas retas do corredor;
4. Erosão de cada imagem binária individualmente por uma máscara diagonal correspondente a uma das retas do corredor. Isso elimina parte dos *outliers*, ou seja, pontos que geralmente correspondem a erros e que devem ser desprezados;
5. Utilização dos dois conjuntos finais de pontos para a determinação das equações das retas do corredor. O método de ajuste e interpolação utilizado é o *RANSAC - Random Sample Consensus* apresentado por Fischler e Bolles [22]. As equações das duas retas encontradas são representadas sob a seguinte forma:

$$x = m_1^{'} y + b_1^{'} \tag{3.1}$$

$$x = m_2^{'} y + b_2^{'} \tag{3.2}$$

   onde x e y correspondem, respectivamente, aos valores da coluna e da linha da imagem, ambos dados em pixels.
6. Cálculo do ponto de fuga através da interseção das duas retas do corredor, igualando-se as equações (3.1) e (3.2), e cálculo do desvio horizontal do ponto de fuga em relação à coluna central da imagem, através da equação (3.3):

$$erro(n) = 96 - x_v \tag{3.3}$$

   onde x_v corresponde à coordenada horizontal ou coluna do ponto de fuga, medida a partir do referencial no topo esquerdo do plano da imagem, e o valor 96 corresponde à coluna central da imagem (192 x 144 *pixels*).

A Figura 3.1 mostra o resultado gráfico do processo de detecção das retas de um corredor e a determinação do ponto de fuga a partir de uma imagem capturada durante a

navegação do robô. A primeira imagem mostrada corresponde a uma imagem capturada do corredor. Em seguida, pode-se observar o resultado da aplicação dos filtros de Sobel e seleção dos pontos com gradiente acima de um *threshold*. Na seqüência, estão as imagens binárias contendo os pontos de gradiente positivo e negativo após a erosão com as máscaras diagonais correspondentes. E por fim, as retas e o ponto de fuga calculados e sobrepostos à imagem original.

Figura 3.1: Detecção das retas do corredor e determinação do ponto de fuga.

Valores númericos resultantes deste procedimento serão analisados no Capítulo 4, referente aos resultados experimentais. Além disto, pode-se encontrar, no Apêndice A, uma rápida descrição do método *RANSAC* e a maneira como esse foi aplicado no ajuste das equações das retas.

3.2 A Aplicação do Método de Servo Controle

É o método de servo controle que determina as mudanças necessárias na velocidade do robô, tanto na sua componente linear quanto na angular, garantindo o controle da sua posição e orientação.

O desvio do ponto de fuga em relação à coluna central da imagem foi adotado como base para o servo controle aplicado. Porém, para alguns casos, quando o robô está muito afastado do centro do corredor, ou seja, mais encostado a uma das paredes, a utilização apenas do ponto de fuga para controlar a sua orientação pode resultar em um maior desvio do mesmo em relação ao centro da trajetória. Isso se deve ao fato de ser necessário controlar tanto a orientação quanto a posição do robô. Para solucionar esse problema, tanto o ponto de fuga quanto as inclinações das retas do corredor foram utilizados na implementação do controle.

As Figuras 3.2 e 3.3 mostram como a posição do ponto de fuga e as inclinações das retas de um corredor variam quando a orientação e a posição do robô mudam. Os valores para ${}^{i}x_v$ e δ_x, apresentados nos gráficos, foram obtidos através do processamento

das imagens capturadas nas diferentes posições e orientações. Estes gráficos podem ser comparados com aqueles apresentados na Figura 2.6, os quais foram traçados a partir das equações (2.20) e (2.21) definidas para ${}^{i}x_v$ e δ_x. Note que o comportamento tanto para ${}^{i}x_v$, quanto para δ_x, obtidos através do processamento das imagens, confirma o que foi definido pelas equações e a maneira como estes sinais variam com as mudanças na orientação e posição do robô. A diferença, basicamente, se deve apenas a um fator de escala nos eixos dos sinais. Isso mostra a vantagem e a validade em se utilizar essas duas características, o ponto de fuga e as inclinações das retas, para o controle do robô.

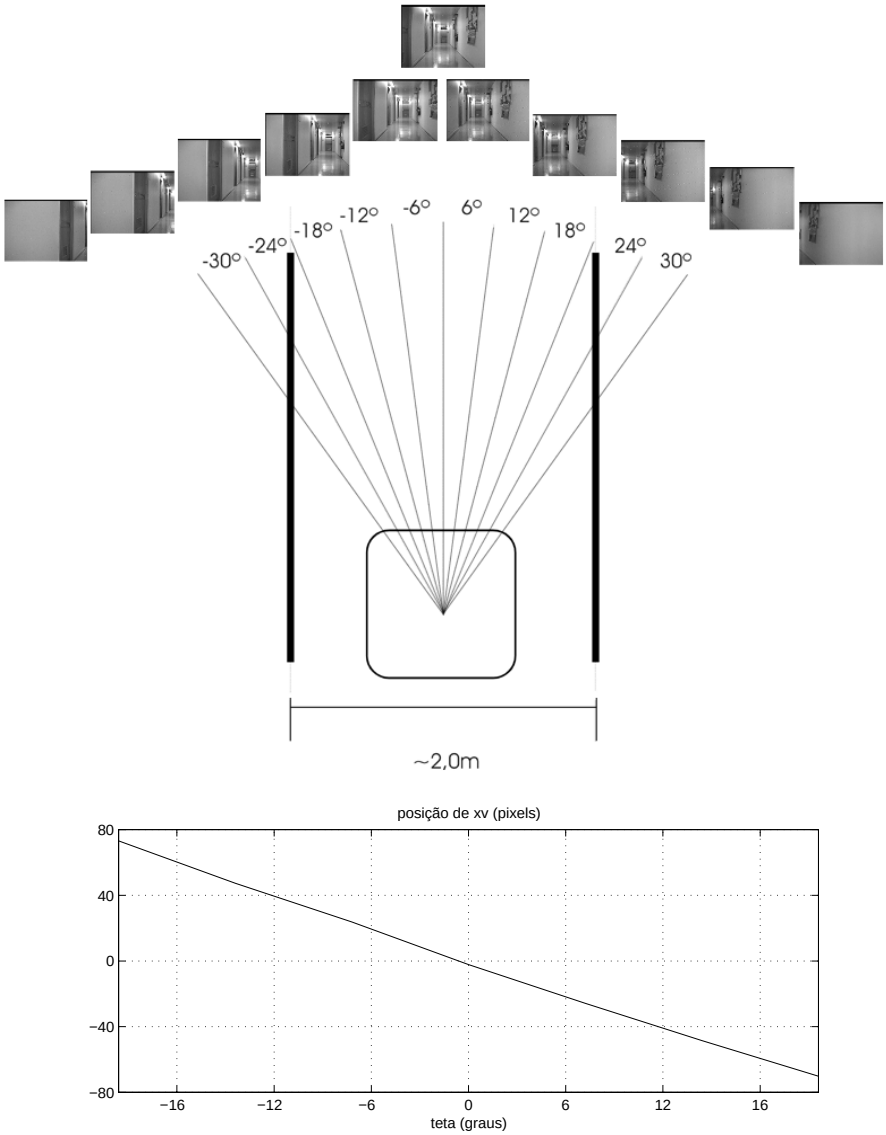

Figura 3.2: Variação da posição do ponto de fuga com a mudança na orientação do robô.

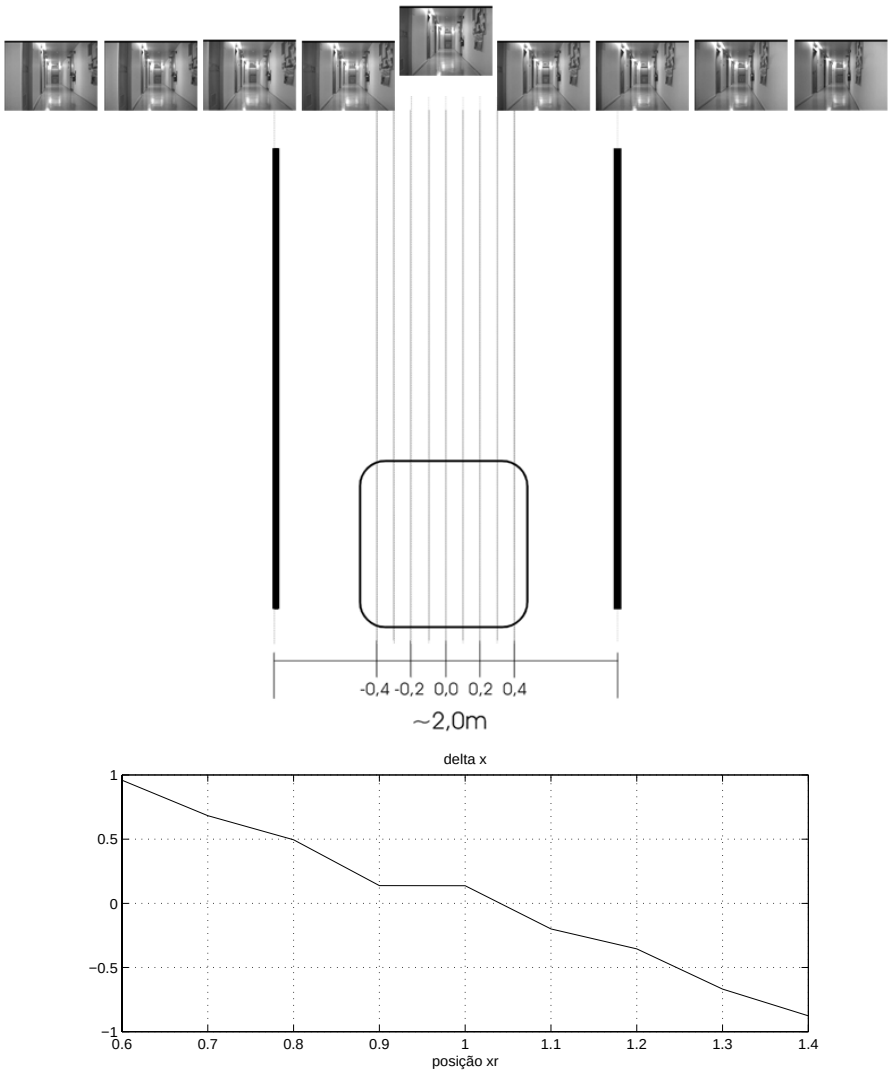

Figura 3.3: Variação das inclinações das retas com a mudança na posição do robô.

O desvio do ponto de fuga foi utilizado como o sinal de erro para o controlador, e a assimetria entre as inclinações das retas foi utilizada para modular os ganhos do controlador aplicado. Essa assimetria é representada por δ_x, definido pela soma dos valores obtidos para $m_1^{'}$ e $m_2^{'}$ nas equações (3.1) e (3.2).

Outro aspecto relevante é a incerteza existente no cálculo do ponto de fuga. Esta incerteza, a qual pode ser representada pelo desvio padrão, mede a confiabilidade dos valores determinados e, por isso, a sua influência também foi considerada. Inicialmente calculou-se a incerteza existente nos parâmetros das retas, utilizando-se os pontos que foram selecionados e aplicando-se o procedimento apresentado por Haralick [27]. Essa incerteza foi representada pelas matrizes de covariância desses parâmetros. Uma vez obtida a incerteza presente nos parâmetros das retas, efetuou-se a propagação dessa para o ponto de fuga, sendo, então, representada sob a forma de desvio padrão. O procedimento para o cálculo das covariâncias dos parâmetros das retas a partir dos pontos selecionados na imagem e a propagação dessa incerteza para o ponto de fuga estão descritos no Apêndice B.

3.2.1 O Controle da Velocidade Linear

O controle da velocidade linear foi considerado independente do controle da velocidade angular. Foram adotados aspectos diferentes na implementação desses controladores, apesar de os mesmos sinais serem usados para seus cálculos e modulação.

Para o caso da velocidade linear, a idéia aplicada é simples. Esta abordagem, representada pela equação (3.4), tem como objetivo garantir que a velocidade linear sofra reduções todas as vezes que o desvio do ponto de fuga for elevado ou a assimetria entre as retas for significativa, indicando que o robô se encontra desviado do centro do corredor. Esta equação corresponde a uma sigmóide, onde o limite superior para a velocidade é dado por VEL e o parâmetro de saturação para a exponencial é dada por E_0.

$$v(n) \;=\; VEL\,\frac{1}{\left(1+e^{(\,|erro(n)|\,-\,E_0)}\right)\,\left(1+\left|m_1^{'}(n)+m'_2(n)\right|\right)} \tag{3.4}$$

onde $VEL = 75\,mm/s$, $E_0 \;=\; 25\,pixels$, $erro(n)$ representa o desvio do ponto de fuga em relação à coluna central da imagem e $m_1^{'}$ e $m_2^{'}$ correspondem as inclinações das retas do corredor.

A modulação conseguida nos valores de velocidade linear através da sigmóide aplicada pode ser observada no gráfico 3.4.

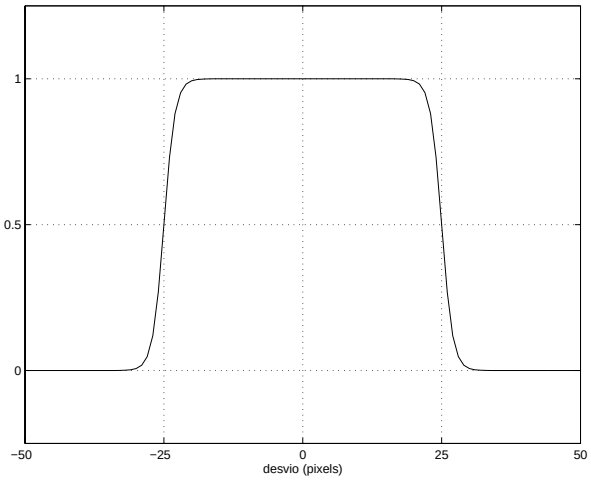

Figura 3.4: Modulação na velocidade linear conseguida através da aplicação da sigmóide.

Pelo gráfico, nota-se a existência das duas regiões de interesse. A primeira, para desvios menores que 25 $pixels$, corresponde à região central do gráfico, onde a velocidade assume valores próximos a VEL, variando inversamente com a assimetria das retas representada por $(m_1^{'} + m_2^{'})$. A segunda, para desvios maiores que $25 pixels$, corresponde às duas regiões laterais onde a velocidade diminui seguindo uma transição suave e confirmando a intenção em se reduzir a velocidade quando o desvio aumentasse muito.

Para se calcular o valor final da velocidade linear a ser aplicada no robô, foi considerada, ainda, a influência do valor da velocidade do ciclo anterior de controle. Isso é feito para garantir que a velocidade linear sofra variações suaves, à medida que o erro no seu posicionamento vai se modificando. O valor final da velocidade linear é calculado através da equação (3.5) dada por:

$$v_f(n) \;=\; \gamma\, v_f(n-1) + (1-\gamma)\, v(n) \tag{3.5}$$

onde $\gamma \;=\; 0.15$ representa a quantidade de "memória" considerada.

Além disto, para que a velocidade não seja reduzida a valores muito baixos, causando praticamente a parada do robô, foi estabelecido um limite inferior. O valor desse limite é de 50 mm/s. Isso significa que o robô se movimentava com velocidade linear uniforme e igual a 50 mm/s todas as vezes que o valor calculado é inferior a esse limite.

3.2.2 O Controle da Velocidade Angular

Para o caso da velocidade angular, chegou-se à conclusão que um controlador PD era suficiente [14, 38]. O erro utilizado como sinal de controle é o desvio do ponto de fuga em relação à coluna central da imagem. Os ganhos do controlador são modulados utilizando-se a assimetria entre as inclinações das retas do corredor e o desvio padrão do ponto de fuga.

$$\omega_r(n) = k_p \{ erro(n) + k_d [erro(n) - erro(n-1)] \} \tag{3.6}$$

onde k_p e k_d são os ganhos proporcional e derivativo e $erro(n)$ é o desvio do ponto de fuga em relação à coluna central da imagem. O valor de k_p é modulado como mostra a equação (3.7) e adota-se $k_d = 0.05$.

$$k_p \;=\; 1.2 \,(\, m_1^{'} + m_2^{'} \,) \frac{5}{(\, 1 + std \,)} \tag{3.7}$$

onde $m_1^{'}$ and $m_2^{'}$ são as inclinações das retas do corredor e std é o desvio padrão do ponto de fuga.

Com o valor de w_r calculado pela equação (3.6), o controle ainda não era eficaz. Notou-se que para desvios do ponto de fuga menores que 25 $pixels$, o valor calculado era satisfatório, mas para valores de desvio maiores que aquele limite, o sinal para w_r deveria coincidir com o sinal do $erro$, ou seja, do próprio desvio. Isso pode ser ilustrado através do esquema na Figura 3.5:

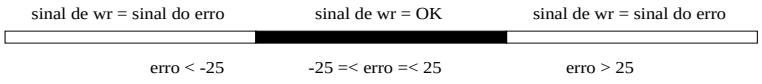

Figura 3.5: Regiões de mudança do sinal da velocidade angular w_r.

Para garantir essas faixas de mudança de sinal para w_r, e ainda criar uma transição suave entre uma faixa e outra, o valor de w_r foi modulado através de uma sigmóide, gerando o valor final w_f definido pela função:

$$w_f(n) \;=\; s(\, erro(n) \,)\, |w_r| \, \frac{erro(n)}{|\, erro(n) \,|} \;+\; (1 - s(\, erro(n) \,))\, w_r \tag{3.8}$$

onde $s(\, erro(n) \,)$ é a sigmóide dada por:

$$s(\, erro(n) \,) \;=\; \frac{1}{1 + \, e^{(E_0 - |\, erro(n) \,|)\alpha}} \tag{3.9}$$

onde $E_0 \;=\; 25\, pixels$ e $\alpha \;=\; 0.5$.

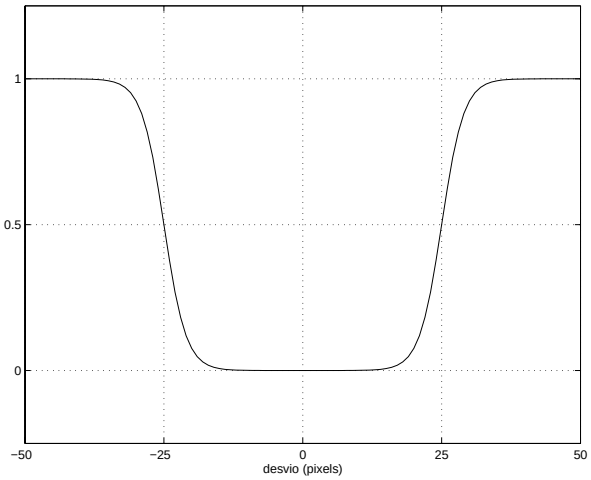

Figura 3.6: Modulação das regiões de troca de sinal de w_r de acordo com a variação do desvio do ponto de fuga.

O gráfico apresentado na Figura 3.6 mostra como se consegue definir a mudança entre uma região e outra de maneira suave, aplicando-se a função $s(\, erro(n)\,)$.

Esse sistema de controle foi escolhido por fornecer bons resultados, uma vez que os parâmetros estejam bem ajustados. Além disto, o processo como um todo não exige grande esforço computacional.

A Figura 3.7 mostra uma simulação da aplicação das leis de controle para a velocidade linear e a velocidade angular que acabaram de ser apresentadas. Foram escolhidas posições iniciais para o robô descentralizadas e com orientações incorretas: $x_r = 0.3$ e $\theta = -15^o$ para o primeiro caso, e $x_r = 0.7$ e $\theta = 15^o$ para o segundo caso. Através das trajetórias obtidas, pode-se observar que o controle implementado permite que o robô corrija tanto a sua posição quanto a sua orientação à medida que se move no interior de um corredor.

Esse sistema de controle foi desenvolvido e implementado durante um estágio de três mêses e meio no Vislab, Laboratório de Visão Computacional do ISR, em cooperação com o LAI, Laboratório de Automação Inteligente da UFES. Devido ao curto espaço de tempo, o controlador implementado não representa a melhor solução para o problema, apesar de resolvê-lo satisfatoriamente. Uma outra opção para o controlador da velocidade angular foi pensada e simulada posteriormente ao término do estágio. Esta outra lei será apresentada a seguir.

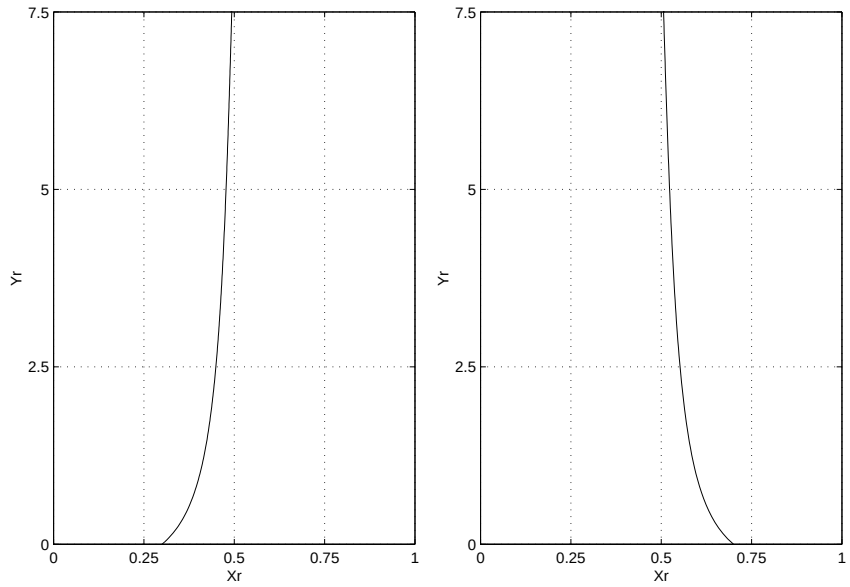

Figura 3.7: Simulação das leis de controle para as velocidades linear e angular quando o robô inicia o movimento descentralizado e com orientação incorreta. As distâncias são medidas em relação à largura do corredor.

Considere a equação de controle para a posição angular θ_c apresentada em (3.10). Os sinais de entrada para o controlador são a coordenada horizontal do ponto de fuga no referencial da imagem ${}^{i}x_v$ e a assimetria entre as duas retas calculadas representada por δ_x, definido na equação (2.21).

$$\theta_c(k) = \theta_c(k-1) + K_p({}^{i}x_v + \delta_x), \qquad \text{com } K_p \sim 0.3 \tag{3.10}$$

A Figura 3.8 mostra uma simulação da aplicação dessa lei de controle quando o robô inicia seu movimento deslocado do centro do corredor, $x_r = 0.8$, e com uma orientação de $\theta = 15^o$. O gráfico do lado esquerdo mostra as mudanças na posição e orientação do robô se apenas o ponto de fuga fosse considerado na lei de controle. O robô seria capaz de corrigir sua orientação, mas não conseguiria se mover para o centro do corredor, corrigindo com sucesso a sua posição. Já o gráfico à direita mostra os resultados quando se considera a influência da assimetria das retas dada por δ_x. O robô corrige tanto a posição quanto a orientação no corredor.

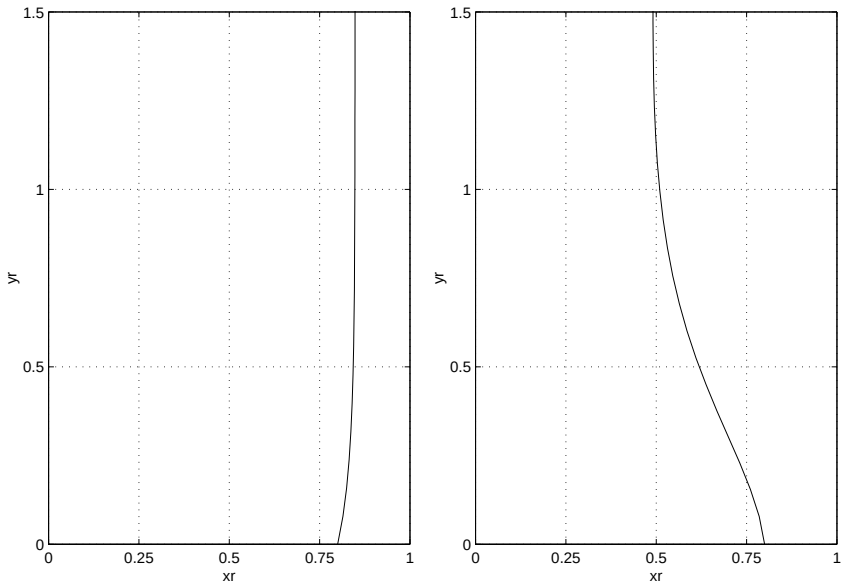

Figura 3.8: Simulação da lei de controle, quando apenas o ponto de fuga é considerado (à esquerda) e quando se inclui também o δ_x (à direita).

3.3 Aplicação do Método Baseado em Aparências

O método de servo controle, por si só, não é capaz de solucionar satisfatoriamente problemas de navegação que necessitam de uma representação mais global do ambiente. Esta representação global permite definir os momentos ou posições em que devem ser tomadas decisões ou realizadas tarefas que não estejam diretamente ligadas ao processo de controle do movimento. Nesse aspecto, a autonomia do robô é ampliada adicionando-se uma representação do ambiente conseguida com a aplicação do método baseado em aparências.

Para o sistema de navegação implementado, o mapa do ambiente de trabalho foi construído a partir de uma seqüência de imagens previamente adquiridas e armazenadas. A ordem destas imagens corresponde à seqüência de locais por onde o robô deve passar ao longo do percurso definido para a sua navegação.

Portanto, à medida que se move no interior de um corredor, a posição do robô é constantemente monitorada usando-se a seqüência de imagens de referência. O processo pode ser descrito através dos seguintes passos:

1. Comparação da imagem capturada do corredor pela câmara com um conjunto de imagens de referência;
2. Definição da posição atual do robô, em relação ao conjunto total de imagens, a partir do resultado obtido no processo de comparação;
3. Determinação se a posição atual corresponde a algum ponto para a execução de uma tarefa ou tomada de uma decisão.

A comparação entre a imagem capturada e as imagens de referência é feita através de correlação. O método de correlação utilizado foi a soma dos quadrados das diferenças ou *SSD - Sum of Squared Differences* [28, 32]. Para o caso de uma imagem 2D discreta, o valor para essa correlação pode ser calculado pela equação (3.11):

$$SSD = \sqrt{\frac{\sum_{l=1}^{H} \sum_{c=1}^{W} (I_1(l,c) - I_2(l,c))^2}{N}} \tag{3.11}$$

onde H e W correspondem, respectivamente, ao número de linhas e colunas, N corresponde à dimensão em pixels $N = H\text{x}W$ das imagens e, I_1 e I_2 são as imagens que estão sendo comparadas.

Os valores obtidos pela aplicação do método de correlação definem a qual das imagens de referência a imagem capturada se assemelha mais. Quanto menor é o valor *SSD* calculado, maior a indicação de semelhança entre as imagens comparadas. Portanto, a melhor correlação obtida, ou seja, a de menor valor como resultado, indica a proximidade do robô à posição representada pela imagem de referência correspondente.

A Figura 3.9 mostra algumas imagens de referência para um dos corredores onde foram feitos testes (o conjunto completo é formado por 15 imagens). A Figura 3.10 apresenta uma das imagens adquiridas pela câmara e o gráfico do resultado da correlação *SSD* com todo o conjunto de referência para aquele corredor, onde o melhor resultado foi para a posição 8. Isso mostra como o uso de métodos de correlação, em particular, o *SSD*, realmente possibilita identificar qualitativamente a posição do robô.

Para garantir uma maior eficiência e rapidez no ciclo de navegação, a imagem capturada pela câmara é comparada com apenas 3 imagens, ao invés de todo o conjunto de referência. Estas imagens correspondem à imagem de referência da posição atual em que o robô se encontra e às imagens de referência das duas posições seguintes. Desse modo,

Figura 3.9: Algumas imagens de referência $\{1^a, 3^a, 5^a, 7^a, 9^a, 11^a\}$ usadas pelo método baseado em aparência no sistema de navegação.

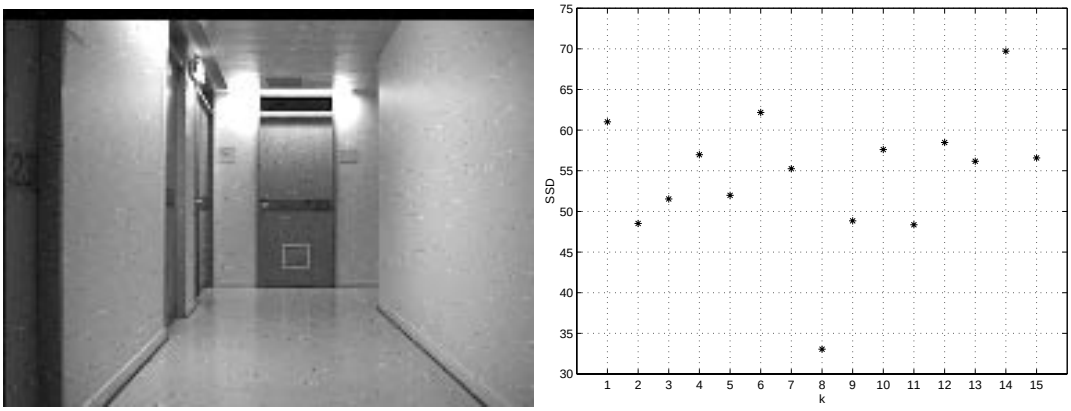

Figura 3.10: À esquerda: Imagem capturada durante o percurso. À direita: Resultado da correlação *SSD* entre a imagem capturada e o conjunto de 15 imagens de referência para todo um corredor.

além de se garantir uma maior rapidez no processo, diminuem-se os erros na estimação da nova posição atual, já que, para corredores, é muito grande a semelhança entre as diversas posições. Ainda com o intuito de diminuir os erros, a posição atual do robô só é atualizada com um novo valor quando uma nova posição é indicada três vezes consecutivas pelo método de correlação nos ciclos de controle.

O método aplicado não tem a capacidade de indicar a posição exata do robô no corredor. O que se consegue é a informação qualitativa que possibilita determinar satisfatoriamente o momento de se executar determinada ação ou atitude. Por exemplo, as imagens no final do corredor foram utilizadas para determinar o momento em que o robô deveria realizar uma curva à direita, antes de prosseguir no corredor seguinte. Além disto, a última imagem da seqüência total indicava que o robô já havia atingido o final da sua trajetória e que, portanto, deveria parar.

A utilização das imagens de referência na ordem do movimento permite a representação do ambiente por um mapa topológico onde se pode observar a combinação dos dois

métodos de navegação aplicados. Os arcos correspondem a segmentos de trajetória onde o esquema de servo controle baseado em visão é utilizado, e o método de aparências monitora de maneira qualitativa a posição do robô. Os nós correspondem aos pontos onde tarefas especiais devem ser executadas.

Os resultados obtidos durante um dos testes de navegação para o processo de correlação entre as imagens de referência e uma das imagens capturadas, assim como a representação topológica do ambiente construída, podem ser observados no Capítulo 4, referente aos resultados experimentais.

Capítulo 4

Resultados Experimentais

Se não houver frutos,
valeu a beleza das flores,
Se não houver flores,
valeu a sombra das folhas,
Se não houver folhas,
valeu a intenção da semente.

(Henfil)

Neste capítulo serão apresentados os resultados obtidos com o sistema de navegação implementado. Primeiramente será descrito o equipamento utilizado para a realização dos experimentos. Logo a seguir, serão apresentados os resultados do processamento de uma imagem capturada durante a navegação. E, por fim, serão mostrados e discutidos os gráficos e mapas traçados a partir dos resultados obtidos com a estratégia de controle aplicada.

4.1 Equipamento Utilizado

As experiências descritas neste trabalho foram realizadas com um TRC Labmate, uma plataforma móvel da HelpMate Robotics Inc., mostrado na Figura 4.1 e pertencente ao Laboratório de Visão Computacional do Instituto Superior Técnico & Instituto de Sistemas e Robótica de Lisboa, Portugal. O sistema de visão utilizado é monocular, e as imagens processadas têm dimensões de 192 x 144 pixels, sendo representadas em 256 tons de cinza.

A câmara empregada para capturar as imagens foi uma câmara da Sony com foco automático, instalada no topo do robô, sendo utilizada uma placa DT3852 da Data

Figura 4.1: O robô utilizado nos experimentos: TRC Labmate.

Translation para a aquisição. A porta serial RS-232 serviu para a comunicação entre o microprocessador do robô e um computador de bordo, responsável pelo processamento das imagens e cálculos envolvidos no controle da navegação. O computador de bordo utilizado foi um Pentium $150MHz$ com $32MBytes$ de memória. A programação foi praticamente toda desenvolvida em Matlab 4.2 $c1$. As rotinas para a estratégia de navegação estão listadas no Apêndice C. As interfaces com o robô e com a placa de aquisição de imagem foram implementadas usando-se a linguagem C.

Os experimentos foram realizados nos corredores do andar do laboratório de Visão do ISR, mostrado na Figura 4.2. A máxima trajetória percorrida era constituída pelos três corredores compridos ($\approx$ 15 m) e pelo pequeno corredor ($\approx$ 5 m), totalizando aproximadamente 50 m. Ao final de cada corredor, o robô executaria a tarefa de girar 90^o no sentido horário para que pudesse prosseguir em frente, até que identificasse a última imagem de referência, significando o final da missão.

4.2 Resultados do Processamento de Imagens

Durante o movimento, cada imagem adquirida era processada para gerar os sinais de controle da navegação, além de ser comparada com as imagens de referência com o objetivo de monitorar o percurso e determinar o momento para a execução de tarefas.

A primeira etapa corresponde à detecção das retas do corredor. Deste processo resultam as equações das retas e as coordenadas do ponto de fuga e, conseqüentemente, o seu desvio em relação à coluna central da imagem e o seu desvio padrão, proveniente da propagação do erro existente no processo de detecção das retas.

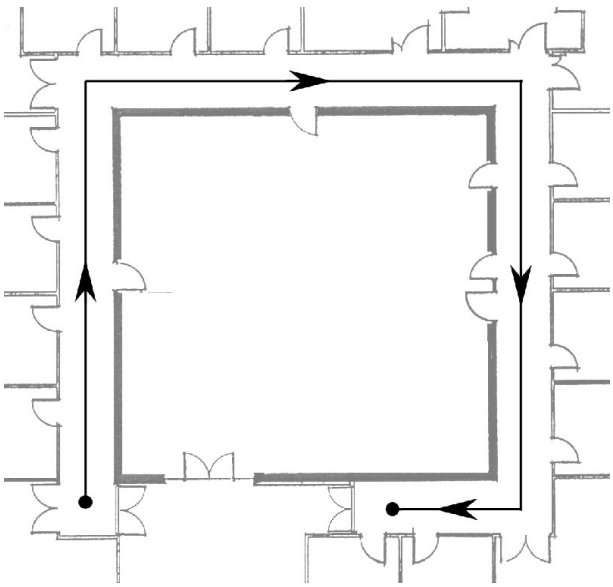

Figura 4.2: Mapa do andar onde foram realizados os testes.

A Figura 4.3 é uma imagem capturada do corredor durante a navegação. O resultado do processo de detecção das retas para esta imagem está mostrado na Figura 4.4, enquanto que a tabela 4.1 mostra os valores numéricos obtidos a partir dos cálculos. O referencial utilizado foi posicionado no topo esquerdo do plano da imagem conforme a Figura 4.5. As coordenadas são medidas em *pixels*.

Figura 4.3: Imagem do corredor capturada durante a navegação.

A partir dos valores mostrados na tabela 4.1, os sinais para a velocidade linear e a velocidade angular foram calculados através da lei de controle definida pelas equações (3.4) e (3.6). O resultado está mostrado na tabela 4.2. É interessante comentar que o valor real calculado para *vel* foi de 26 mm/s, mas valores menores que 50 mm/s são ignorados e substituídos por este valor limite.

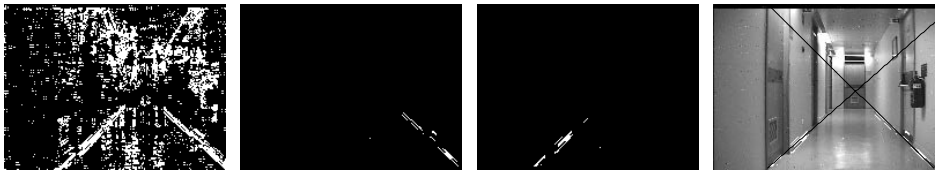

Figura 4.4: Resultado do processo de detecção das retas do corredor.

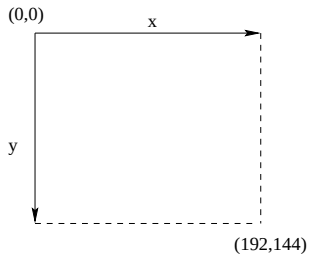

Figura 4.5: O referencial utilizado no plano da imagem.

Parâmetros da reta 1	$m_1^{'} = 1.09$ $b_1^{'} = 39.05$
Parâmetros da reta 2	$m_2^{'} = -1.12$ $b_2^{'} = 208.00$
Ponto de fuga	$ptf = (122, 77)\ pixels$
Desvio do ptf em relação à coluna central	$erro = -26\ pixels$
Desvio Padrão do ptf	$std = 5.76\ pixels$

Tabela 4.1: Resultados do processo de Detecção das Retas.

Velocidade Angular	$w_r = -0.6256\ ^o/s$
Velocidade Linear	$vel = 50\, mm/s$

Tabela 4.2: Valores das velocidades angular e linear calculados e aplicados ao robô.

Além de gerar os sinais de controle, a imagem da Figura 4.3 foi comparada com três imagens do conjunto de referência. Estas imagens correspondem à posição atual em que o robô se encontrava e às duas posições seguintes. Para este caso, a posição atual era a posição 3 no conjunto total de referência. O resultado da correlação entre a imagem capturada e as imagens de referência das posições 3, 4 e 5 pode ser observado na Figura 4.6. Nota-se que o menor, e, portanto, o melhor valor foi para a posição 4, o que significa que o robô já se encontrava nas imediações da posição seguinte à posição atual 3.

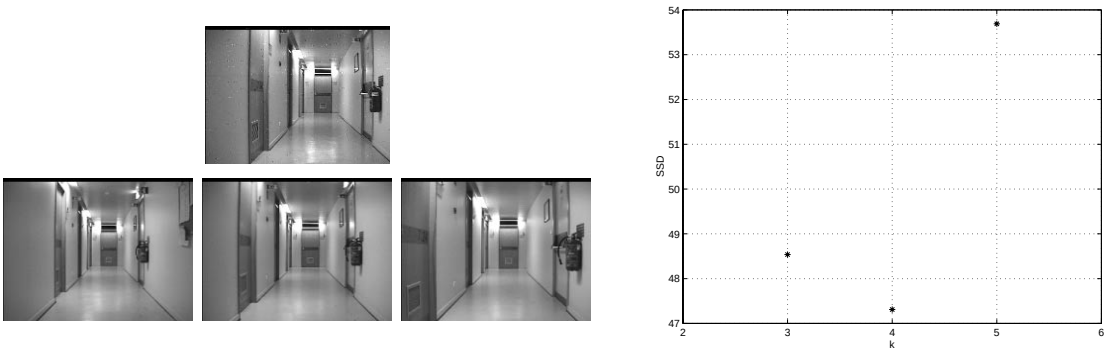

Figura 4.6: À esquerda: Imagem capturada (acima) e as imagens de referência 3, 4 e 5. À direita: Resultado da correlação.

Todos esses passos eram sempre repetidos para cada imagem capturada durante a navegação. Os resultados do processamento das imagens forneciam os parâmetros de entrada para o servo controle da posição e orientação do robô e para o método de aparência no monitoramento do percurso.

4.3 Resultados da Navegação no Interior dos Corredores

Durante os testes de navegação no interior dos corredores, o sistema implementado foi monitorado para a avaliação da sua funcionalidade e desempenho. A intenção foi observar a sua capacidade em controlar a posição e orientação do robô durante o movimento e ainda identificar os momentos corretos para a rotação no final dos corredores. A freqüência obtida para o ciclo completo de controle foi cerca de 2 Hz.

A Figura 4.7 mostra os resultados obtidos durante um experimento realizado apenas no primeiro corredor. A trajetória, mostrando a variação da posição do robô durante o movimento, foi reconstruída a partir de medidas de odometria. Pode-se observar ainda a evolução temporal do erro do ponto de fuga em relação à coluna central da imagem, do desvio padrão do ponto de fuga e da orientação do robô. Interessante ressaltar que, devido ao equipamento utilizado, a orientação do robô foi medida no sentido anti-horário, ou seja, rotações à esquerda correspondem ao sentido positivo e crescente dos ângulos. Além disso, o gráfico de odometria não corresponde exatamente à trajetória desenvolvida devido a erros geralmente causados por deslizamentos das rodas do robô. O tempo, nos gráficos, foi medido em segundos e o comprimento do corredor é de aproximadamente 15 m.

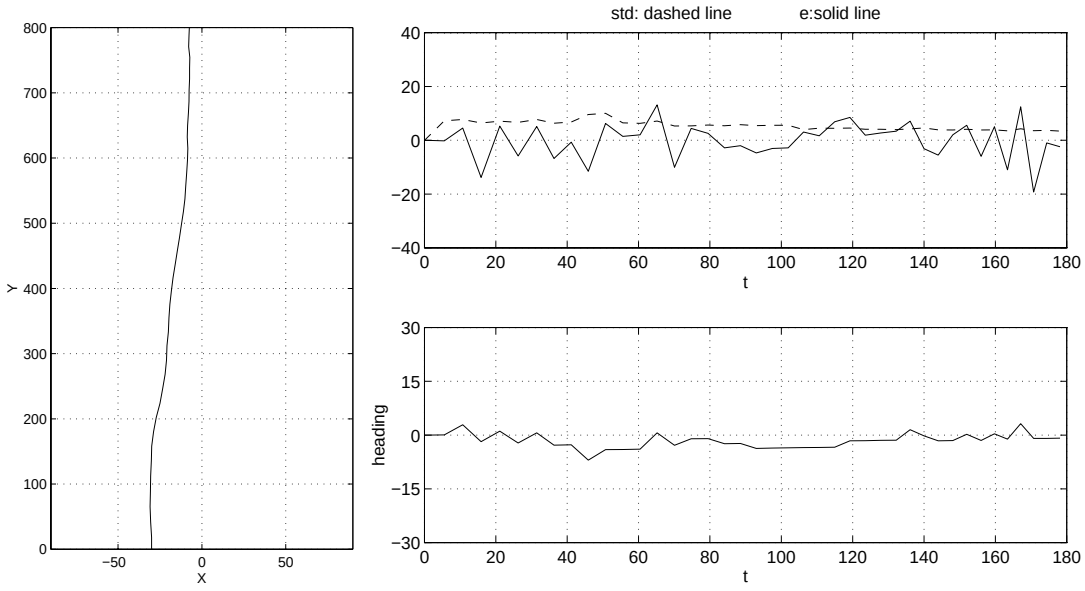

Figura 4.7: À esquerda: Trajetória do robô reconstruída a partir de medidas de odometria (cm). Acima, à direita : Desvio do ponto de fuga e o seu desvio padrão (ambos em *pixels*). Abaixo, à direita: Variação na orientação robô ($graus$).

Pode-se observar que o robô foi capaz de acertar a sua posição, movimentando-se em direção ao centro do corredor a partir de uma posição inicial próxima à parede esquerda e que, depois disto, manteve-se centralizado no resto do percurso.

As Figuras 4.8 a 4.14 mostram os resultados obtidos para uma volta completa no interior dos corredores do andar do laboratório. A Figura 4.8 mostra a trajetória desenvolvida pelo robô, reconstruída a partir das medidas de odometria. O percurso total é de aproximadamente 50 m.

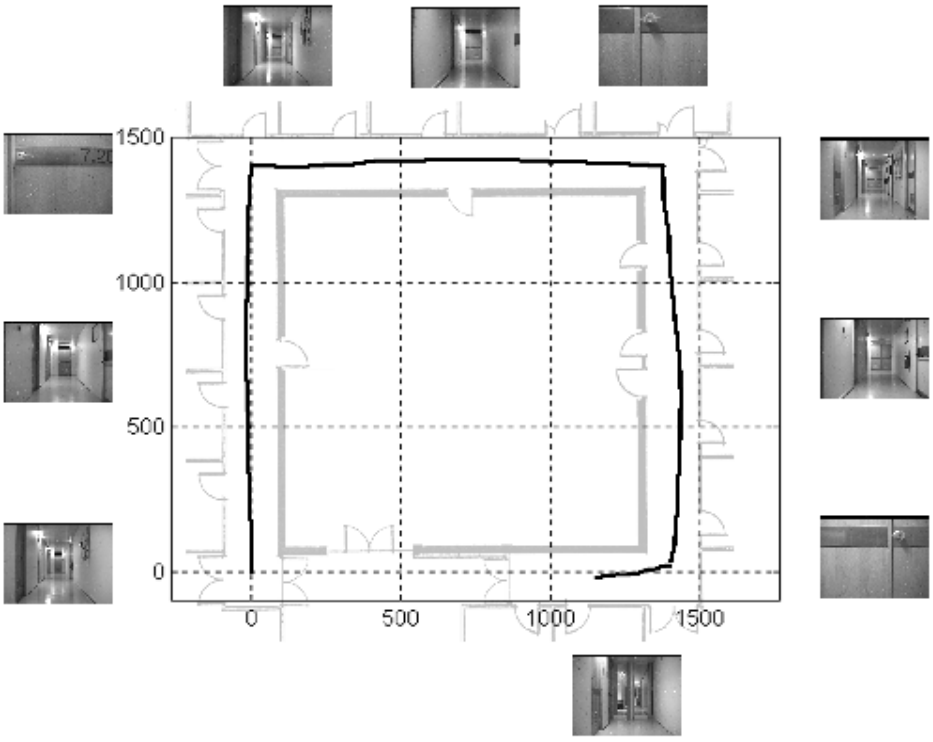

Figura 4.8: Percurso desenvolvido pelo robô durante uma volta completa pelos corredores do andar.

Foi utilizado um conjunto de 51 imagens como referência. A partir destas imagens, foi possível gerar um mapa topológico do ambiente como o mostrado na Figura 4.9. Este mapa é constituído por 5 nós e 4 arcos. O primeiro e o último dos nós correspondem às posições inicial e final do trajeto percorrido pelo robô. Os três nós intermediários correspondem às posições finais de corredores, onde ficou definida a tarefa de uma rotação de 90^o à direita para que o robô prosseguisse no corredor seguinte. Os arcos, por sua vez, representam os corredores propriamente ditos onde as características do ambiente (os contornos das

retas) foram capturadas e utilizadas pelo método de servo controle para controlar o robô. Enquanto isto, durante toda a navegação, o método de aparências fazia o monitoramento da posição do robô.

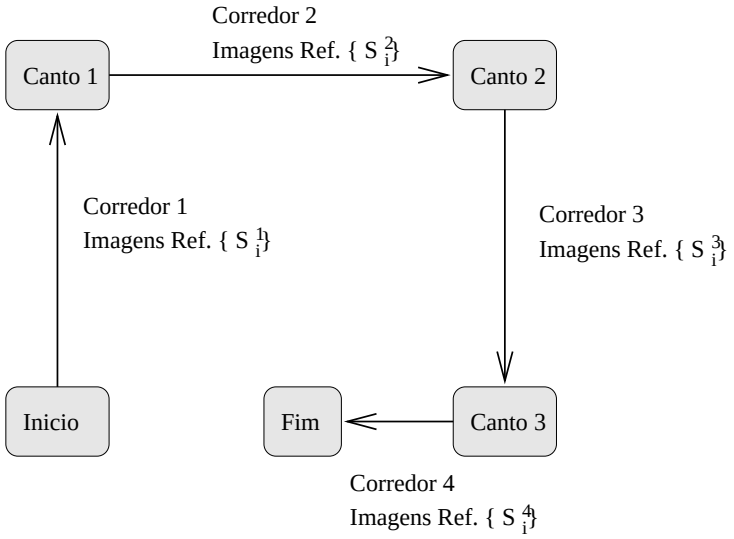

Figura 4.9: Mapa topológico do andar do laboratório construído a partir das imagens de referência e das posições escolhidas para a execução de tarefas. Os nós correspondem aos pontos onde o robô deve efetuar uma rotação de 90^o, sentido horário. E os arcos representam as regiões dos corredores onde se aplica o método de servo controle.

A Figura 4.10 mostra as mudanças na orientação do robô. Já as Figura 4.11 e 4.12 mostram, respectivamente, as variações nos sinais aplicados para a velocidade linear e a velocidade angular. A Figura 4.13 mostra as variações no desvio do ponto de fuga em relação à coluna central da imagem e no seu desvio padrão. E por fim, a Figura 4.14 traz o gráfico para a posição relativa no conjunto de imagens de referência ao longo do percurso. Todos os gráficos foram traçados em função do tempo de movimento do robô, medido em segundos.

No gráfico da Figura 4.10, onde se observa a orientação do robô, podem ser indentificados os momentos em que o robô atingiu o final dos corredores, ou seja, os três nós intermediários no mapa do ambiente. Note que as esquinas entre dois corredores correspondem às regiões de mudança em 90^o na orientação do robô. Nesses locais, enquanto a orientação muda de um valor para outro, a velocidade linear se anula e a velocidade angular é mantida constante, sendo igual a $5graus/s$ (o sinal negativo é devido ao tipo de orientação adotada), como mostram os gráficos das Figuras 4.11 e 4.12.

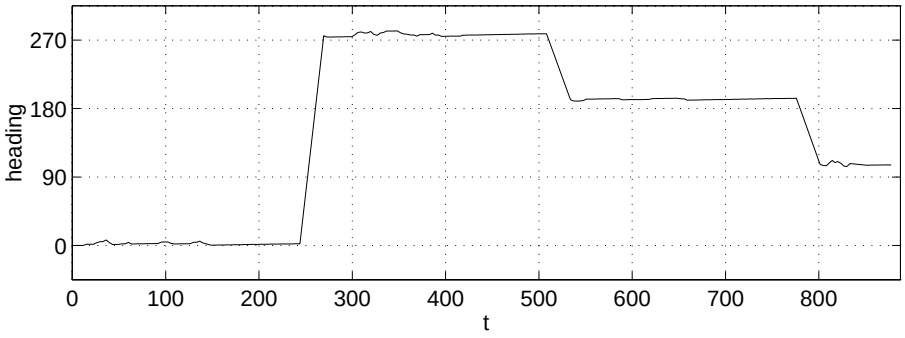

Figura 4.10: Orientação do robô ($graus$) medida no sentido anti-horário.

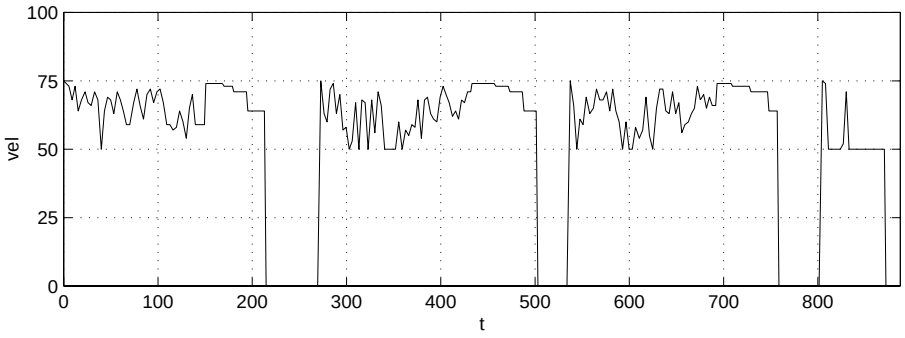

Figura 4.11: Sinal de referência calculado para a velocidade linear do robô (mm/s).

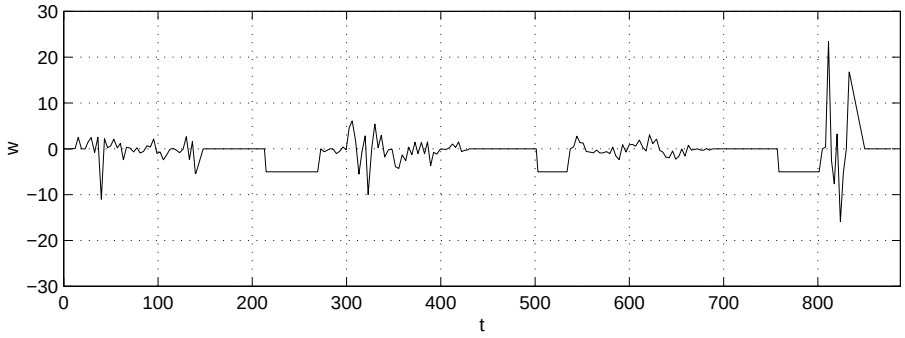

Figura 4.12: Sinal de referência calculado para a velocidade angular do robô ($graus/s$).

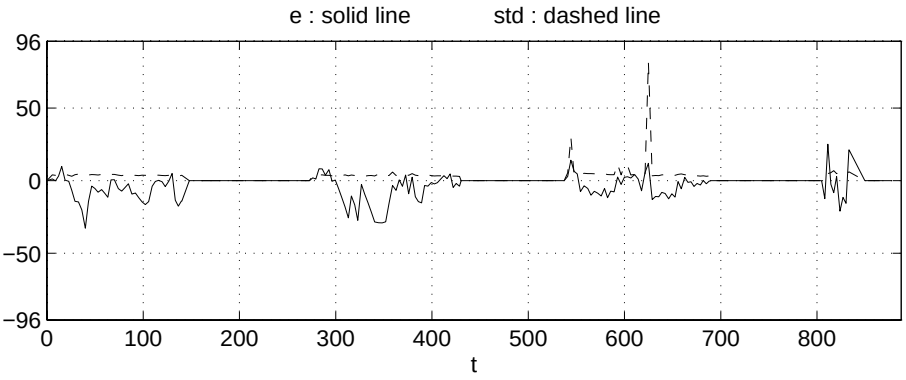

Figura 4.13: Desvio do ponto de fuga em relação à coluna central da imagem e o seu desvio padrão (ambos em *pixels*).

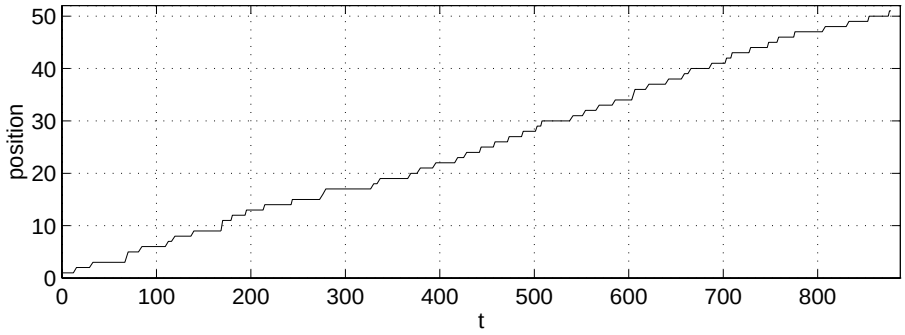

Figura 4.14: Gráfico das posições identificadas em relação ao conjunto de imagens de referência durante o movimento.

Ainda no gráfico da Figura 4.11, nas regiões um pouco antes do final dos corredores, pode-se notar uma redução gradativa da velocidade. Esta redução corresponde à aproximação da posição de rotação. Durante esta aproximação, a velocidade angular é reduzida a zero para que o robô caminhe em linha reta até a posição de rotação. Vale a pena ressaltar, ainda, o fato da velocidade linear não ultrapassar o valor de $75\,mm/s$, limite superior na lei de controle empregada. Além disso, nota-se a limitação em $50\,mm/s$ como seu valor mínimo ao longo dos corredores. Como dito no Capítulo 3, isso foi feito para garantir o mínimo de movimento do robô.

Comparando-se o gráfico da Figura 4.13, que mostra o desvio do ponto de fuga, com o gráfico da Figura 4.12, nota-se que as variações no sinal aplicado para a velocidade angular acompanham as variações no desvio do ponto de fuga. Os maiores valores aplicados para a velocidade angular correspondem a pontos onde o desvio do ponto de fuga calculado

foi também significativo, indicando um razoável erro na orientação do robô. Já a incerteza do ponto de fuga, representada pelo seu desvio padrão, manteve-se praticamente constante, apresentando apenas um pico, onde provavelmente ocorreu um erro na detecção das retas. Os valores para o desvio do ponto de fuga e seu desvio padrão se anulam nas regiões correspondentes à aproximação do final de um corredor e ao período de rotação, pois nestas regiões, não são realizados a detecão das retas e o cálculo do ponto de fuga.

Os valores elevados para a velocidade angular que ocorrem no final do percurso são decorrentes dos erros encontrados na detecção das retas no último corredor. Isso geralmente acontece porque o comprimento deste último é muito pequeno e, portanto, quase não se obtém imagem das retas no campo visual suficiente para se efetuar a sua detecção sem grandes erros.

Finalmente, no gráfico da Figura 4.14, pode-se observar a mudança gradativa nas posições percorridas pelo robô durante o movimento. A permanência em cada uma das posições corresponde ao intervalo de tempo que o robô gasta para fazer a transição entre uma posição atual e a seguinte, ou seja, considerar válida uma nova posição após identificá-las três vezes consecutivas. A diferença de tamanho para alguns intervalos se deve à grande semelhança entre certas posições do corredor, o que, às vezes, dificulta e causa um atraso na confirmação de uma nova posição.

De um modo geral, os resultados obtidos mostram que o robô foi capaz de corrigir e manter sua posição centralizada no corredor, durante o movimento. Além disso, a comparação com as imagens de referência permitiu determinar qualitativamente a sua posição de maneira suficiente para que o robô executasse com sucesso as tarefas definidas ao longo do percurso.

Capítulo 5

Conclusões

Talvez não tenhamos conseguido fazer o melhor,
mas lutamos para que o melhor fosse feito...
não somos o que deveríamos ser,
não somos o que iremos ser,
mas, graças a Deus, não somos o que éramos.

(Martin Luther King)

Nesta dissertação foi abordado o problema da navegação de robôs móveis em ambientes interiores com enfoque para o caso de corredores. O objetivo foi manter o robô sempre no centro do corredor durante o movimento e aumentar a sua autonomia na navegação através da definição de tarefas em pontos especiais do trajeto. O sistema de sensoriamento utilizado foi o de visão computacional baseado no processamento de imagens em tons de cinza capturadas por uma única câmara. Dois paradigmas foram combinados: o servo controle e o método de aparências para a implementação da estratégia de navegação do robô.

O servo controle se baseou no ponto de fuga e na assimetria das inclinações das retas do corredor, detectadas, sempre que possível, nas imagens capturadas. Já que esse paradigma não é capaz de definir ou realizar ações de âmbito mais geral, sua tarefa se limitou em corrigir o movimento do robô, garantindo o controle da sua orientação e posição com relação ao centro do corredor.

A inclusão do método de aparências tornou possível o monitoramento da posição do robô quanto ao percurso total a ser percorrido e, ainda, a definição de ações ou tarefas mais gerais desvinculadas do controle motor do robô. Este método permite associar comandos

ou eventos com a aparência de imagens capturadas do ambiente. Em particular, para o trabalho desenvolvido nesta dissertação, foram definidas duas tarefas: as ações de virar ao final de cada corredor e cessar o movimento ao se identificar o ponto final da trajetória. O mapeamento do ambiente foi realizado a partir de uma seqüência de imagens armazenadas previamente à navegação do robô. Durante o movimento, a cada ciclo de controle, a imagem capturada foi comparada com um conjunto de imagens de referência.

O controle para a posição e orientação do robô obtido com o servo controle apresentou resposta de forma robusta e estável, e foi capaz de corrigir e manter adequadamente o robô centrado no corredor durante o movimento [57]. O sucesso conseguido na execução das tarefas definidas, comprova a validade da aplicação do método de aparências. As imagens, utilizadas como referência, constituem uma representação não métrica do ambiente. Esta representação se baseia na aparência das diversas posições e localidades que constituem a área de navegação. A partir das imagens, conseguiu-se mapear o ambiente e monitorar a posição do robô de maneira relativa, sem a necessidade do conhecimento preciso do valor métrico da sua posição em um referencial ou a reconstrução 3D do ambiente [53].

A partir dos resultados obtidos e mostrados no Capítulo 4, pôde-se comprovar a funcionalidade da união desses dois métodos e as vantagens que se obtém quando esta combinação é aplicada como estratégia de navegação [58]. O método de servo controle possibilita a execução de tarefas locais de uma maneira bem robusta. Tarefas locais são executadas apenas nas regiões onde se consegue extrair informações das imagens do ambiente, pois dependem destas características para serem definidas, como foi o caso do controle da posição e da orientação do robô nesta dissertação. O método de aparência permite a execução de tarefas mais globais, definidas em diferentes posições dentro do mapeamento do ambiente. Essas tarefas não dependem da extração de características, mas sim, da identificação das posições através da aparência das suas imagens correspondentes. Desta forma, os dois métodos se complementam, aumentando, assim, a autonomia do sistema de navegação do robô.

O tempo de resposta obtido foi também satisfatório para o problema proposto. Os atrasos mais significativos são devidos ao processamento das imagens do corredor durante a comparação com as imagens de referência, já que as rotinas para detecção das retas não necessitam de grande demanda computacional.

5.1 Trabalhos Futuros

Apesar de os resultados atingirem as expectativas, alguns detalhes ainda faltam ser solucionados e melhorados. Uma das idéias é ampliar o sistema de navegação, tornando-o mais abrangente, não se restringindo apenas a corredores, mas sim, promovendo a navegação do robô em ambientes estruturados. Isto corresponderá a melhorias e aprimoramentos na estratégia atual implementada. Há também a intenção de se incluir novas tarefas e funções para o robô, como por exemplo a detecção de obstáculos. O aumento da velocidade de resposta do sistema e, conseqüentemente, o aumento da velocidade de navegação do robô também representam objetivos futuros. As idéias para trabalhos futuros podem ser resumidas em:

- **Aumentar a velocidade e as possibilidades de utilização deste sistema de navegação em diferentes ambientes, tornando-o mais robusto e autônomo.** Espera-se conseguir isso empregando-se novas formas de representação das imagens de referência para otimizar ou até modificar o método de comparação e correlação utilizado. Além disto, para melhorar a lei de controle da posição e orientação, pode-se tentar um estudo mais aprofundado das características do ambiente de trabalho e suas projeções no plano da imagem.
- **Definir um método de escolha automática de marcadores naturais ou artificiais no ambiente e incluir métodos de planejamento de trajetórias.** Até agora, a escolha das imagens de referência foi definida antes do movimento e o sistema não tem autonomia nenhuma em descartar ou modificar as imagens escolhidas. Alguns trabalhos apresentam diferentes maneiras para se escolher e descartar esses marcadores e efetuar, a partir deles, o planejamento da trajetória a ser seguida [60]. Geralmente, esses métodos de escolha se baseiam na presença de objetos, construção de vetores de características, diferenças de contornos, cores e outras informações extraídas das imagens que garantem diferenças marcantes entre elas [17, 41]. A partir das posições dos marcadores escolhidos, diferentes formas de mapeamento do ambiente [8, 42, 63] e de planejamento de trajetórias [11, 56] podem ser definidas. Isso também contribuirá para a navegação, não só em corredores, mas em outros ambientes estruturados.
- **Adicionar novos comportamentos ao robô, como por exemplo, a capacidade de detecção de obstáculos.** Isso pode ser conseguido através do cálculo de

fluxo óptico [12, 29, 39], como nos trabalhos em [15, 16, 25, 49]. A utilização do fluxo óptico ainda pode ser aplicada no cálculo do tempo de colisão, o que permite reduzir a velocidade do robô gradativamente até a sua parada completa ou estacionamento em locais determinados [51, 48, 44].

- **Definir novas tarefas para o robô.** A partir do mapeamento adotado, espera-se executar tarefas como entrar em uma sala, procurar por um objeto específico, pegá-lo, etc; aumentando mais ainda a autonomia do sistema.

O problema abordado neste trabalho é apenas um pequeno foco da área de navegação de robôs móveis. Essa área representa um amplo campo de estudo, cujo interesse aumenta a cada dia. A cada novo passo dado, surgem novas idéias, curiosidades e dúvidas, as quais garantem um contínuo trabalho de pesquisa e experimentações.

Apêndice A

RANSAC - Random Sample Consensus

Os pontos das retas detectados e selecionados nas imagens foram utilizados para determinar as equações das retas do corredor. O método de estimação aplicado como método de ajuste de curva foi o *RANSAC - Random Sample Consensus* [22].

Geralmente, os métodos de estimação procuram utilizar o maior número de pontos para uma estimativa inicial e então, ir eliminando os pontos inválidos. O *RANSAC*, ao contrário desses métodos, diferencia-se pelo fato de utilizar apenas o número mínimo e suficiente de pontos necessários para uma primeira estimativa, aumentando o conjunto com novos pontos sempre que possível.

Suponha um conjunto de P pontos de amostras para se instanciar um modelo que requer o mínimo de n pontos, tal que $P > n$. O método de *RANSAC* pode ser descrito por:

1. Seleciona-se aleatoriamente um conjunto S_1 formado por n pontos a partir do conjunto P para uma primeira estimativa do modelo M_1;
2. Usa-se o modelo M_1 para determinar o novo conjunto S_1^* de pontos de P que se aproximam desse modelo com erros dentro de uma certa tolerância;
3. Se o número de pontos em S_1^* é maior que um certo limite t, que é função da estimativa do nível de erros em P, usa-se S_1^* para uma nova e melhor estimativa M_1^* do modelo, usando um método de ajuste como por exemplo, Mínimos Quadrados ou *Least Squares*;
4. Se o número de pontos em S_1^* é menor que o limite t, seleciona-se aleatoriamente um novo conjunto S_2 de n pontos e repete-se o procedimento descrito;

5. Se depois de um certo número de tentativas, não se obtiver um conjunto S_i de t ou mais pontos, pode-se fazer uma estimativa final do modelo com o maior conjunto encontrado, ou terminar o procedimento indicando-se erro.

Esse método requer que 3 parâmetros sejam especificados:

- A tolerância do erro que é utilizada para definir se um ponto é ou não compatível com o modelo instanciado;
- O número de tentativas do procedimento;
- O número t que representa o número mínimo de pontos que um conjunto S_i deve ter para que o modelo estimado a partir dos seus pontos seja considerado correto.

Para o trabalho desenvolvido nesta dissertação, o procedimento adotado para estimação das retas de um corredor foi uma versão ligeiramente modificada, com o número de tentativas fixado em 8. A intenção foi determinar a equação de cada uma das retas a partir dos pontos selecionados na imagem. A primeira estimativa era sempre feita com $n = 2$, já que bastam dois pontos para definir uma reta em duas dimensões. O procedimento pode ser descrito por:

1. Ordena-se o conjunto de pontos de acordo com a coordenada correspondente à sua coluna na imagem;
2. Escolhem-se aleatoriamente 2 pontos pertencentes, respectivamente, à primeira e à última terça parte do conjunto total. Isso garante a escolha de dois pontos que apresentem uma certa distância entre si;
3. Define-se a equação da reta a partir dos dois pontos escolhidos;
4. Medem-se os erros de distância dos demais pontos do conjunto em relação à reta calculada;
5. Calcula-se a média dos erros;
6. Se a média for menor que o melhor valor encontrado até então, guardam-se todos os pontos que possuem erros menores que a média calculada;
7. Se ainda não terminaram as 8 tentativas, volta-se ao segundo passo e repete-se o procedimento;

8. Ao final das 8 tentativas, tem-se armazenado o melhor conjunto de pontos. Faz-se, então, uma nova estimativa da reta, usando-se todos os pontos desse conjunto com o método de Mínimos Quadrados.

O método de *RANSAC* demostrou ser bem mais robusto do que outros, sofrendo muito menos influência dos *outliers* presentes no conjunto de pontos selecionados na detecção das retas do corredor. Maiores detalhes em relação a esse método e alguns aprimoramentos são apresentados em [22].

Apêndice B

Propagação da Covariância

A incerteza relacionada ao ponto de fuga foi calculada através da propagação das covariâncias dos parâmetros das retas dos corredores.

Para o cálculo das covariâncias dos parâmetros das retas, foi utilizado o método exposto por Haralick [27]. Nesse método, são calculadas as covariâncias correspondentes aos parâmetros θ e ρ de uma reta, a partir das médias e variâncias dos pontos selecionados no processo de detecção das retas do corredor.

Considere a seguinte representação para uma reta:

$$y \sin\theta + x \cos\theta = \rho \tag{B.1}$$

E no caso da detecção das retas, considere os pontos selecionados $(\hat{x_n}, \hat{y_n})$ como amostragens ruidosas de coordenadas (x_n, y_n). Os pontos $(\hat{x_n}, \hat{y_n})$ se relacionam com os pontos (x_n, y_n) pelo modelo:

$$\begin{pmatrix} \hat{x_n} \\ \hat{y_n} \end{pmatrix} = \begin{pmatrix} x_n \\ y_n \end{pmatrix} + \xi_n \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} \tag{B.2}$$

onde ξ_n são independentes e normalmente distribuídos com $N(0, \sigma^2)$.

As médias e as variâncias dos pontos amostrados, os quais foram utilizados para a interpolação da reta, podem ser calculadas por:

$$\mu_x = \frac{1}{N}\sum_{n=1}^{N} x_n \tag{B.3}$$

$$\mu_y = \frac{1}{N}\sum_{n=1}^{N} y_n \tag{B.4}$$

$$\sigma_x^2 = \sum_{n=1}^{N} (x_n - \mu_x)^2 \tag{B.5}$$

$$\sigma_y^2 = \sum_{n=1}^{N} (y_n - \mu_y)^2 \tag{B.6}$$

$$\sigma_{xy} = \sum_{n=1}^{N} (x_n - \mu_x)(y_n - \mu_y) \tag{B.7}$$

A matriz de covariância dos parâmetros θ e ρ da reta é dada por:

$$\sum_{\theta\rho}^{2x2} = \begin{bmatrix} \sigma_{\theta\theta} & \sigma_{\theta\rho} \\ \sigma_{\rho\theta} & \sigma_{\rho\rho} \end{bmatrix} \tag{B.8}$$

onde

$$\sigma_{\theta\theta} = \frac{4\sigma^2(\sigma_y^2 \cos^2\theta + \sigma_x^2 \sin^2\theta - \sigma_{xy}\sin 2\theta)}{2NT^2}$$

$$\sigma_{\theta\rho} = \sigma_{\rho\theta} = \frac{-4\sigma^2[\,(\mu_x \sin\theta - \mu_y \cos\theta)\,(\sigma_y^2 \cos^2\theta + \sigma_x^2 \sin^2\theta - \sigma_{xy}\sin 2\theta)\,]}{2NT^2}$$

$$\sigma_{\rho\rho} = \frac{4\sigma^2[\,(\mu_x \sin\theta - \mu_y \cos\theta)^2\,(\sigma_y^2 \cos^2\theta + \sigma_x^2 \sin^2\theta - \sigma_{xy}\sin 2\theta) + N(\,(\mu_x \sin\theta - \mu_y \cos\theta)^2 - T)^2\,]}{2NT^2}$$

$$T = \frac{\sigma_y^2 - \sigma_x^2}{N}\cos 2\theta - \frac{2\sigma_{xy}}{N}\sin 2\theta - (\mu_y \sin\theta + \mu_x \cos\theta)^2 + \rho(\mu_y \sin\theta + \mu_x \cos\theta)$$

Representando as duas equações das retas do corredor no plano da imagem, com $y = lin$ e $x = col$:

$$r1 \quad : \quad lin \sin\theta_1 + col \cos\theta 1 = \rho_1 \tag{B.9}$$

$$r2 \quad : \quad lin \sin\theta_2 + col \cos\theta 2 = \rho_1 \tag{B.10}$$

O ponto de fuga é definido igualando-se as equações (B.9) e (B.10). Suas coordenadas são dadas por:

$$lin = \frac{\rho_2 \cos\theta_1 - \rho_1 \cos\theta_2}{\sin(\theta_2 - \theta_1)} \longrightarrow f_1(\theta_1, \rho_1, \theta_2, \rho_2) \tag{B.11}$$

$$col = \frac{\rho_1 \sin\theta_2 - \rho_2 \sin\theta_1}{\sin(\theta_2 - \theta_1)} \longrightarrow f_2(\theta_1, \rho_1, \theta_2, \rho_2) \tag{B.12}$$

A propagação das incertezas dos parâmetros das retas para o ponto de fuga é efetuada através da expressão:

$$\sum_{Pontodefuga}^{2x2} = H \sum_{\theta_1,\rho_1,\theta_2,\rho_2}^{4x4} H^T \tag{B.13}$$

onde $\sum$ significa a matriz de covariância e H é uma matriz definida a partir das derivadas parciais das coordenadas do ponto de fuga em relação a cada um dos parâmetros θ_1, ρ_1, θ_2 e ρ_2:

$$H = \begin{bmatrix} \frac{\partial f_1}{\partial \theta_1} & \frac{\partial f_1}{\partial \rho_1} & \frac{\partial f_1}{\partial \theta_2} & \frac{\partial f_1}{\partial \rho_2} \\ \frac{\partial f_2}{\partial \theta_1} & \frac{\partial f_2}{\partial \rho_1} & \frac{\partial f_2}{\partial \theta_2} & \frac{\partial f_2}{\partial \rho_2} \end{bmatrix} \tag{B.14}$$

onde

$$\frac{\partial f_1}{\partial \theta_1} = \frac{\cos\theta_2[\rho_2 - \rho_1 \cos(\theta_2 - \theta_1)]}{\sin^2(\theta_2 - \theta_1)}$$

$$\frac{\partial f_1}{\partial \rho_1} = -\frac{\cos\theta_2}{\sin(\theta_2 - \theta_1)}$$

$$\frac{\partial f_1}{\partial \theta_2} = \frac{\cos\theta_1[\rho_1 - \rho_2\cos(\theta_2 - \theta_1)]}{\sin^2(\theta_2 - \theta_1)}$$

$$\frac{\partial f_1}{\partial \rho_2} = \frac{\cos\theta_1}{\sin(\theta_2 - \theta_1)}$$

$$\frac{\partial f_2}{\partial \theta_1} = \frac{\sin\theta_2[\rho_1\cos(\theta_2 - \theta_1) - \rho_2]}{\sin^2(\theta_2 - \theta_1)}$$

$$\frac{\partial f_2}{\partial \rho_1} = \frac{\sin\theta_2}{\sin(\theta_2 - \theta_1)}$$

$$\frac{\partial f_2}{\partial \theta_2} = -\frac{\sin\theta_1[\rho_1 + \rho_2\cos(\theta_2 - \theta_1)]}{\sin^2(\theta_2 - \theta_1)}$$

$$\frac{\partial f_2}{\partial \rho_2} = -\frac{\sin\theta_1}{\sin(\theta_2 - \theta_1)}$$

A matriz de covariância $\sum_{\theta_1,\rho_1,\theta_2,\rho_2}^{4x4}$ para os parâmtros θ_1, ρ_1, θ_2 e ρ_2 é dada por:

$$\sum_{\theta_1,\rho_1,\theta_2,\rho_2}^{4x4} = \begin{bmatrix} \theta_1\theta_1 & \theta_1\rho_1 & 0 & 0 \\ \rho_1\theta_1 & \rho_1\rho_1 & 0 & 0 \\ 0 & 0 & \theta_2\theta_2 & \theta_2\rho_2 \\ 0 & 0 & \rho_2\theta_2 & \rho_2\rho_2 \end{bmatrix} \tag{B.15}$$

Através da expressão (B.13), as incertezas relacionadas com o ponto de fuga podem ser calculadas e o desvio padrão std será dado por:

$$\sum_{Pontodefuga}^{2x2} = \begin{bmatrix} \sigma_{linlin} & \sigma_{lincol} \\ \sigma_{collin} & \sigma_{colcol} \end{bmatrix} \tag{B.16}$$

$$std = \sqrt{\sigma_{lincol}} \tag{B.17}$$

Apêndice C

As Rotinas para Matlab da Estratégia de Navegação

Neste Apêndice estão listados os comandos e rotinas definidos para a estratégia de navegação desenvolvida nesta dissertação. As rotinas aqui apresentadas foram implementadas para Matlab 4.2*c*1.

As rotinas de nível mais baixo que definem a interface com a placa de aquisição utilizada, com o microcontrolador do robô e a comunicação via porta serial foram desenvolvidas por outro componente da equipe do laboratório, o aluno Antonio Bastos. Por esse motivo, essas rotinas não serão apresentadas e descritas neste Apêndice, porém, encontram-se disponíveis no laboratório. Neste caso, os programas foram desenvolvidos utilizando-se a linguagem C.

C.1 Comandos de interface com a placa de aquisição de imagens

Os comandos para a inicialização, finalização e aquisição de uma imagem, apresentados a seguir representam apenas a interface *shell* para Matlab das rotinas que foram definidas em C.

Comando "acqinit"

Inicializa a placa de aquisição.

```
% Inicializacao da placa de aquisicao
acqmat(0);
```

Comando "acq"

Faz a aquisição de uma imagem.

```
function [i]= acq
% function [i]= acq
% Rotina de aquisicao de imagem
[i1,i2,i3,i4]=acqmat(1);
i = [i1,i2,i3,i4];
```

Comando "acqexit"

Finaliza a comunicação com a placa de aquisição.

```
% Finalizacao da placa de aquisicao
acqmat(3);
```

C.2 Comandos de interface com o robô

Assim como no caso anterior, alguns dos comandos representam apenas uma interface *shell* para o Matlab.

Comando "labinit"

Inicializa a comunicação com o robô.

```
% Inicializacao da comunicacao com a Labmate
labmat(0);
```

Comando "labreset"

Reinicializa o robô. Anula as velocidades linear e angular, assim como as demais variáveis de estado do robô.

```
% Reset da labmate
labmat(7);
```

Comando "labexit"

Finaliza a comunicação com o robô.

```
% Finalizacao da comunicacao com a Labmate
labmat(9);
```

Comando "set_vel(v,wr)"

Aplica os valores de referência para as velocidades linear e angular no robô.

```
function set_vel(v,wr)
% Carrega os valores para velocidade linear e velocidade angular
% da labmate
labmat(1,v,wr);
```

Comando "set_acel(al,ar)"

Aplica os valores de referência para as acelerações linear e angular no robô.

```
function set_acel(al,ar)
% Carrega os valores para aceleracao linear e aceleracao angular
% da labmate
labmat(2,al,ar);
```

Comando "get_status"

Lê o registrador de estado do robô.

```
function s=get_stat
% function s=get_stat
% Le o status da labmate, retorna s=[heading x y vel_left vel_right]
% onde os valores de velocidades sao das rodas esquerda e direita
s=labmat(8);
```

Comando "clr_head"

Anula o valor da orientação armazenado no registrador de estado do robô.

```
% Anula o heading da Labmate
labmat(6);
```

Comando "turn_abs(ang,r)"

Rotaciona o robô até se atingir a orientação de `ang` graus no seu referencial absoluto. O raio da rotação é dado por `r`.

```
function turn_abs(ang,r)
% function turn_abs(ang,r)
% Rotacao Absoluta da Labmate
% O comando de rotacao e' acompanhado do resto do programa para que se
% garanta que o comando seja executado por completo antes de se
% terminar a rotina
% Ideia e' manter a Labmate rodando enquanto nao se atingir o valor
% do angulo pedido +- um erro
% OBS.: Para as rotinas internas da Labmate o valor do angulo devera
% ser multiplicado por 100. Ex.: 45 graus = 4500

% erro=2 graus
erro=200;
```

```
% Caso o angulo seja 0 ou 360 e' necessario um tratamento especial
if (ang==0 | ang==360)
a1=36000;
a2=0;
% Rotacao Absoluta
labmat(3,a1,r);
% Get status
s=labmat(8);
while(1)
if(s(1)>(a1-erro))
break;
elseif (s(1)<(a2+erro))
break;
end
% Get status
s=labmat(8);
end
else
a1=ang*100;
% Rotacao Absoluta
labmat(3,a1,r);
% Get status
s=labmat(8);
while( s(1)<(a1-erro) | s(1)>(a1+erro))
% Get status
s=labmat(8);
end
end
```

C.3 Rotinas da estratégia de navegação nos corredores

Comando "missão"

Define a navegação do robô nos corredores utilizados para testes durante os experimentos. A partir dessa rotina é que são chamadas todas as outras necessárias para o movimento e controle do robô nos corredores.

```
%  Rotina de Navegacao da Labmate
% - Inicializacao
% * Inicializacao das variaveis
% * Inicializacao da placa de aquisicao e da labmate
% * Aquisicao da 1 imagem
% - Navegacao
% * Percurso do 1 corredor
% * Execucao de tarefa (Rotacao de -90 graus)
% * Percurso do 2 corredor
% * Execucao de tarefa (Rotacao de -90 graus)
% * Percurso do 3 corredor
% * Execucao de tarefa (Rotacao de -90 graus)
% * Percurso do 4 corredor
%
% Variaveis globais
err_sum=0; % Acumulador dos erros dos Ptf's calculados ao centro da imagem
dist_ant=0; % Variavel que guarda o valor de teta para ser usado novamente
REF=96; % Valor da coluna onde o Ptf ideal deve ser encontrado
VEL=75; % Valor da velocidade linear
cont=1; % Contador de amostragens para os dados armazenados
posatual=1; % Indicador da posicao atual
pos_ant=1; % Indicador da posicao anterior
wr=0; % Velocidade angular
ssd=0; % SSD - sum of squared differences metric entre duas imagens
vel=VEL;
heading_ref=0;
cont_foto=1;
```

```
ind_foto=1;
prefixo='foto';

%****** Inicializacao

% Inicializacao da Labmate e da acquisicao
acqinit;
labinit;
% Anula o heading
clr_head;
% Define as aceleracoes
set_acel(15,5);
% Adquire a primeira imagem
colormap(gray);
image=acq;
imshow(image);
truesize;
%Grava uma imagem para criar um AVI
% ind_foto=saveimg(ind_foto,prefixo);
% cont_foto=1;
%Comeca movimento
set_vel(VEL,0);
% Inicializacao dos dados de acompanhamento
tempo=clock;
Status(cont,:)=get_stat;
Time_seg(cont)=0;
Wr(cont)=wr;
Posatual(cont)=posatual;
Pos(cont)=posatual;
SSD(cont)=ssd;
Poly1(cont,:)=[0 0];
Poly2(cont,:)=[0 0];
Ptf(cont,:)=[0 0];
Erro(cont)=dist_ant;
Covptf(cont)=0;
```

```
Std(cont)=0;
VelAplic(cont)=vel;
heading_ref=Status(cont,1)/100;

%****** Percurso dos corredores
% Looping n 1 - Primeiro Corredor
Nref=9;
loop;
savedata;
% Primeira virada
Nref=15;
atitude1;
savedata;
% Looping n 2 - Segundo Corredor
posatual=16;
pos=16;
Nref=24
loop;
savedata;
% Segunda virada
Nref=30
atitude1;
savedata;
% Looping n 3 - Terceiro Corredor
posatual=31;
pos=31;
Nref=41
loop;
savedata;
% Terceira virada
Nref=47
atitude1;
savedata;
% Looping n 4 - Restinho do  Corredor
posatual=48;
```

```
pos=48;
Nref=49;
loop;
savedata;
% Aproximacao final
Nref=51
atitude4;
savedata;

%******* Finalizacao
% Finaliza a placa de aquisicao
acqexit;
% Anula as velocidades
vel=0;
set_vel(vel,0);
savedata;
% Destruicao das variaveis auxiliares
clear poly1 poly2 cov1 cov2 ptf covptf  imagem ssd;
clear posatual pos wr dist_ant err_sum tr1 tr2;
clear image pos_ant tempo Nref vel heading_ref prefixo;
clear cont_foto;
return;
```

Comando "atitude1"

Esta rotina define a tarefa de aproximação da posição de rotação no final de cada corredor e a execução de um giro de 90^o no sentido horário.

```
% Tomada de atitude: aproximacao da posicao de rotacao e giro de 90 graus

% Acerta o heading, caso seja necessario
if (abs(dist_ant)>7)
turn_abs(heading_ref,0);
end
```

```
% Anula alguns dados para manter coerencia
% no processo de amarzenamento de dados
wr=0;
poly1=[0;0];
poly2=[0;0];
ptf=[0;0];
covptf=zeros(2,2);
dist_ant=0;

% Mantem o movimento em linha reta mas prossegue fazendo
% matching das posicoes ate encontrar a posicao de rotacao
cont_pos=0;
while(posatual<Nref)
imagem=acq;
[pos,ssd]=matchpos(imagem,posatual,Nref)
if (pos==pos_ant)
cont_pos=cont_pos+1;
if (abs(posatual-pos)<=2 & cont_pos==3)
posatual=pos
cont_pos=0;
vel=floor(VEL*(1-exp(pos-Nref)))
if (vel<50)
vel=50;
end
set_vel(vel,0);
end
end
pos_ant=pos;
imshow(imagem);drawnow;
savedata;

%Grava imagem para fazer AVI
% if(cont_foto==5)
% ind_foto=saveimg(ind_foto,prefixo);
% cont_foto=1;
```

```
% else
% cont_foto=cont_foto+1;
% end
end

% Rotaciona -90 graus
turn_for(-90,0);
savedata;

% Espera-se tempo necessario para que o clr_head nao interfira
% no turn_for
tic;
while(toc<2)
end
% clr_head;

% Guarda o valor do novo heading como referencia
s=get_stat;
heading_ref=s(1)/100;

% Carrega os valores de aceleracao novamente porque o
% clr_head muda os valores anteriores
set_acel(15,5);

% e os valores de velocidades porque a rotina turn_for
% anula os mesmos
vel=VEL;
set_vel(vel,0);

% Mostra a nova posicao e destroi variaveis auxiliares
imagem=acq;
imshow(imagem);drawnow;
clear cont_pos s;

%Grava uma imagem para fazer um AVI
```

```
% ind_foto=saveimg(ind_foto,prefixo);
% cont_foto=1;
```

Comando "atitude4"

Esta rotina define a tarefa de aproximação da posição final até encontrar o momento exato de cessar o movimento e finalizar a missão.

```
% Tomada de atitude4: aproximacao da posicao final para cessar o movimento

% Acerta o heading, caso seja necessario
if (abs(dist_ant)>7)
turn_abs(heading_ref,0);
end

% Anula alguns dados para manter coerencia
% no processo de amarzenamento de dados
wr=0;
poly1=[0;0];
poly2=[0;0];
ptf=[0;0];
covptf=zeros(2,2);
dist_ant=0;

% Mantem o movimento em linha reta, mas prossegue fazendo
% matching das posicoes ate encontrar a posicao final
cont_pos=0;
while(posatual<Nref)
imagem=acq;
[pos,ssd]=matchpos(imagem,posatual,Nref)
if (pos==pos_ant)
cont_pos=cont_pos+1;
if (abs(posatual-pos)<=2 & cont_pos==3)
posatual=pos
cont_pos=0;
```

```
vel=floor(VEL*(1-exp(pos-Nref)))
if (vel<50)
vel=50;
end
set_vel(vel,0);
end
end
pos_ant=pos;
imshow(imagem);drawnow;
savedata;

%Grava imagem para fazer um AVI
% if(cont_foto==5)
% ind_foto=saveimg(ind_foto,prefixo);
% cont_foto=1;
% else
% cont_foto=cont_foto+1;
% end
end

% Mostra a nova posicao e destroi variaveis auxiliares
imagem=acq;
imshow(imagem);drawnow;
clear cont_pos;

%Grava imagem para fazer AVI
% ind_foto=saveimg(ind_foto,prefixo);
% cont_foto=1;
```

Comando "loop"

Define o ciclo de controle da estratégia de navegação.

```
% Loop de controle para o percurso dos corredores do ISR
%  * Aquisicao de nova imagem
```

```
% * Processamento da imagem, extracao das retas e calculo do ponto de fuga
% * Controle da velocidade angular e velocidade linear
% * Determinacao da posicao atual
%  * Verificacao da condicao de parada

while(posatual<Nref)

% Aquisicao da imagem
imagem=acq;
% Deteccao das linhas do corredor
[poly1,poly2,tr1,tr2,cov1,cov2]=linedet(imagem);
% Calculo do ponto de fuga
[ptf,covptf]=ptfdetec(tr1,tr2,cov1,cov2,poly1,poly2);
% Calculo do erro e do controle
[dist_ant,err_sum,wr,vel]=ctrlwr(ptf,REF,dist_ant,err_sum,poly1(2),poly2(2),
                                        sqrt(covptf(2,2)),vel);
% Machting para determinacao da posicao atual
[pos,ssd]=matchpos(imagem,posatual,Nref);
if (pos==pos_ant)
if (abs(posatual-pos)<=2)
posatual=pos
end
end
pos_ant=pos;
% Armazenamento dos dados
savedata;
%Grava imagem para criar um AVI
% if(cont_foto==5)
% ind_foto=saveimg(ind_foto,prefixo);
% cont_foto=1;
% else
% cont_foto=cont_foto+1;
%
% end
```

```
end
```

Comando "linedet(im)"

Faz a detecção das retas do corredor. São aplicados os filtros de Sobel e máscaras de erosão para a seleção dos pontos pertencentes ao contorno das retas. O método de *RANSAC* é utilizado para determinar as equações e, além disso, é calculada a matriz de covariância dos parâmetros das retas.

```
function [p1,p2,tr1,tr2,cov1,cov2]=linedet(IM)
%function [p1,p2,tr1,tr2,cov1,cov2]=linedet(IM)
% Detecta os pontos do contorno de uma reta
% Os pontos de interesse sao selecionados comparando-se os modulos de
% Ex e Ey com um valor de threshold e aplicando-se uma erosao com
% mascaras diagonais. As retas sao interpoladas, usando-se o metodo
% de Ransac
% Retorna os  parametros das retas e as covariancias de teta e ro

%****** Inicializacao
th=0.008;
eout=IM;
IM=double(IM)/255;
[m,n]=size(IM);
mask1=[ 1 1 0 0 0 0;
0 1 1 0 0 0;
0 0 1 1 0 0;
0 0 0 1 1 0;
0 0 0 0 1 1];
mask2=[ 0 0 0 0 1 1;
0 0 0 1 1 0;
0 0 1 1 0 0;
0 1 1 0 0 0;
1 1 0 0 0 0];

%****** Definicoes do filtro de Sobel
```

```
fx=[-1 0 1;-2 0 2; -1 0 1]/8;
fy=[1 2 1; 0 0 0; -1 -2 -1]/8;

%****** Aplicacao do filtro
Ex=filter2(fx,IM);
Ey=filter2(fy,IM);
% figure;
% imshow(Ex+Ey);

%****** Escolha dos pontos para as novas estimativas
% Selecao dos pontos com gradiente acima de um threshold
IM=abs(Ex)>th & abs(Ey)>th;
% figure;
% imshow(IM);
% Determinacao do sinal do gradiente
IS=sign(Ex).*sign(Ey);
% IM1 = matriz com pontos de gradiente>th e inclinacao positiva
IM1=(IM.*IS)==1;
% IM2 = matriz com pontos de gradiente>th e inclinacao negativa
IM2=(IM.*IS)==-1;
clear Ex Ey IS fx fy;
% Refinamento da escolha atraves de erosao com uma mascara.
% A mascara e' aplica apenas na parte inferior da imagem e o
% restante da mesma e' igualado a zero
IM1(90:m,1:n)=erode(IM1(90:m,1:n),mask1);
IM1(1:90,1:n)=zeros(90,n);
figure;
imshow(IM1);
IM2(90:m,1:n)=erode(IM2(90:m,1:n),mask2);
IM2(1:90,1:n)=zeros(90,n);
figure;
imshow(IM2);

%******* Separa os indices de linhas e colunas onde se encontrar 1's
%   nas matrizes IM1 e IM2
```

```
[newl1,newc1]=find(IM1);
[newl2,newc2]=find(IM2);
newl1=newl1';
newc1=newc1';
newl2=newl2';
newc2=newc2';

%****** Estimativa das novas retas
p1=ransac2d(newl1,newc1);
p2=ransac2d(newl2,newc2);
% Calculo de teta e ro e montagem da Variavel tr1 = [teta1 ro1]
tr1(1)=atan(-p1(2));
tr1(2)=abs(p1(1)*cos(tr1(1)));
cov1=covar_tr(newc1,newl1,tr1,1);
% Calculo de teta e ro e montagem da Variavel tr2 = [teta2 ro2]
tr2(1)=atan(-p2(2));
tr2(2)=abs(p2(1)*cos(tr2(1)));
cov2=covar_tr(newc2,newl2,tr2,1);

%****** Display
% figure;
% imshow(IM1+IM2);
dispret(p1,p2,IM1+IM2,eout);
return
```

Comando "ransac2d(x,y)"

Corresponde à rotina de interpolação da equação de uma reta a partir de um conjunto de pontos de amostragens usando o método de *RANSAC*.

```
function solution = ransac2d(x,y)
% function solution = ransac2d(x,y)
% Faz a interpolacao dos pontos para determinacao de uma reta
% usando Ransac - Random Sample Consensus
N = size(x,2);
```

```
% 99% chances of getting two inliers at least once, if 25% of x's are
% outliers .

ntrial = 8;
[x,I] = sort(x);
y = y(I);
best = -1 ;
for t = 1:ntrial ,
i1 = ceil(N*rand(1)/3);
i2 = ceil(N*(2+rand(1))/3);
slope = (y(i2)-y(i1))/(x(i2)-x(i1)) ;
height= y(i2) - x(i2)*slope ;
error = (-y+slope*x + height).^2 ;
med   = median(error);
% Selection best inliers
if (best<0) | (med < best) ,
good = error <= med ;
best = med ;
end
end
x = x(good);
y = y(good);
solution = lscov([ones(size(x));x]',y',eye(size(x,2)));
```

Comando "covar_tr(x,y,tr,vsig)"

Calcula a matriz de covariância dos parâmetros das retas.

```
function [eout]=covar_tr(x,y,tr,vsig)
% function [eout]=covar_tr(x,y,tr,vsig)
% Esta funcao calcula a matriz de covariancia entre os parametros da
% reta, teta e ro, a partir da covariancia entre os pontos x e y
% usados na interpolacao da mesma
%  medx e medy : medias
% cvarxy      : matriz de covariancia dos sinais x e y
```

```
% vsig     : variancia do signal
% teta        : angulo
% ro          : distancia
% N     : numero de amostras de x e y
%****** Calculo das medias e matriz de covariancia entre x e y
% Os valores de x e y sao subtraidos do menor valor encontrado (shift)
teta=tr(1);
ro=tr(2);
[l,N]=size(x);
xmin=min(x);
ymin=min(y);
x=x-xmin;
y=y-ymin;
medy=mean(y);
medx=mean(x);
cvarxy=cov(x,y)*(N-1);
% Senos e Cossenos
st=sin(teta);
ct=cos(teta);
s2t=sin(2*teta);
c2t=cos(2*teta);

%****** Definicao das Variancias de x e y, e Covariancia xy
vx=cvarxy(1,1);
vy=cvarxy(2,2);
cvxy=cvarxy(1,2);

%****** Calculo dos termos da matriz de Covariancia entre Teta e Ro
% Variaveis Auxiliares
term0=medy*st+medx*ct;
T=((vy-vx)/N)*c2t - (2*cvxy/N)*s2t - ((term0)^2) + ro*(term0);
term1=vy*(ct^2)+vx*(st^2)-cvxy*s2t;
term2=medx*st-medy*ct;
numer=4*vsig;
denom=2*N*(T^2);
```

```
eout(1,1)=term1;
eout(2,2)=((term2)^2)*(term1)+N*(((term2)^2) -T)^2;
eout(1,2)=-(term2*term1);
eout(2,1)=eout(1,2);
eout=(eout)*(numer/denom);
return
```

Comando "dispret(poly1,poly2,IM,img)"

Corresponde a uma rotina auxiliar para se traçar as retas detectadas sobrepondo-as à imagem do corredor.

```
function  dispret(poly1,poly2,IM,img)
% function  dispret(poly1,poly2,IM,img)
% Faz o display das retas detectadas sobrepostas a imagem original

[m,n]=size(img);
% Calculo dos pontos das retas
l=1:m;
c1=round(poly1(2)*l+poly1(1));
c2=round(poly2(2)*l+poly2(1));

% Marcacao em branco dos pontos usados na interpolacao
[lin,col]=find(IM);
t=size(lin);
for i=1:t
img(lin(i),col(i))=255;
end
% Display
figure;
imshow(img);
hold on;
plot(c1,l,'w');
plot(c2,l,'w');
hold off;
```

```
drawnow;
return
```

Comando "ptfdetec(tr1,tr2,cov1,cov2,p1,p2)"

Calcula as coordenadas e a matriz de covariância do ponto de fuga. A partir dessa matriz é calculado o seu desvio padrão.

```
function [ptf,covptf]= ptfdetec(tr1,tr2,cov1,cov2,p1,p2)
% function [ptf,covptf]= ptfdetec(tr1,tr2,cov1,cov2,p1,p2)
% Calcula o ponto de fuga dado duas retas definidas por seus respectivos
% parametros.
% Alem disto, calcula a matriz de covariancia do ponto de fuga a partir
% das matrizes de covariancia de teta e ro das duas retas
% tr = [teta  ro]

% Variaveis Auxiliares
teta1=tr1(1);
ro1=tr1(2);
teta2=tr2(1);
ro2=tr2(2);
ct1=cos(teta1);
ct2=cos(teta2);
st1=sin(teta1);
st2=sin(teta2);

%****** Calculo do ponto de fuga
lixo1= [1 -p1(2); 1 -p2(2)];
lixo2=[p1(1); p2(1)];
ptf=inv(lixo1)*lixo2;

%****** Calculo da matriz de covariancia
   %*** Calculo de H
% Variaveis Auxiliares
st21=sin(teta2-teta1);
```

```
ct21=cos(teta2-teta1);
den=st21^2;

% F1 e' define a coordenada linha do ptf em funcao de teta1 ro1 teta2 ro2
% F2 e' define a coordenada coluna do ptf em funcao de teta1 ro1 teta2 ro2
% Calculo das derivadas para a criacao de H
df1dt1=ct2*(ro2 - ro1*ct21)/den;
df1dr1=-ct2/st21;
df1dt2=ct1*(ro1 - ro2*ct21)/den;
df1dr2=ct1/st21;
df2dt1=st2*(ro1*ct21-ro2)/den;
df2dr1=st2/st21;
df2dt2=-st1*(ro1 + ro2*ct21)/den;
df2dr2=-st1/st21;
%
% Especifica{\c c}ao de H:
%
%       H =[  df2 df2 df2 df2 (coluna=x)
%               dt1 dr1 dt2 dr2
%               df1 df1 df1 df1 (linha=y)
%               dt1 dr1 dt2 dr2 ]
%
        H=[df2dt1 df2dr1 df2dt2 df2dr2;df1dt1 df1dr1 df1dt2 df1dr2];

% *** Propagacao da Covariancia dos tetas e ros para o ponto de fuga ****
aux=zeros(4,4);
aux(1:2,1:2)=cov1;
aux(3:4,3:4)=cov2;
covptf=H*aux*H';
return
```

Comando "ctrlwr(ptf,ref,dist_ant,err_sum,m1,m2,std,vel_ant)"

Define o controle do robô. Primeiramente, o desvio do ponto de fuga em relação à coluna central da imagem é calculado e, depois, são definidos e aplicados ao robô os sinais de referência para a velocidade angular e a velocidade linear.

```
function [dist_ant,err_sum,wr,vel_ant]=
ctrlwr(ptf,ref,dist_ant,err_sum,m1,m2,std,vel_ant)
% function [dist_ant,err_sum,wr,vel_ant]=
%ctrlwr(ptf,ref,dist_ant,err_sum,m1,m2,std,vel_ant)
% Calcula e executa o controle da velocidade angular e aplica
% um filtro de primeira ordem para calcular a velocidade linear
% da Labmate baseado no erro do ponto de fuga estimado em relacao a
% posicao ideal, na inclinacao das retas do corredor e ainda, na
% covariancia do ponto de fuga.
% O controle da velocidade angular e' tipo PID.
% As novas velocidades sao aplicadas a Labmate

%  Definicao dos ganhos do PID
kp=1.2*(5/(1+std));
kd=0.05;
ki=0;

%****** Desvio do ponto de fuga atual em relacao ao centro da imagem
dist=ref-ptf(1)

%****** Controle da velocidade angular a partir do desvio
err_sum = err_sum + dist;
E0=25;
a=0.5;
sigm=1/(1+exp(E0-abs(dist))*a);
wc=(kp)*(m1+m2)*(dist + kd*(dist - dist_ant) + ki*err_sum);
wr=sigm*abs(wc)*sign(dist)+(1-sigm)*wc

%****** Controle da velocidade linear
```

```
y=0.15;
vel=y*vel_ant+(1-y)*75*1/((1+exp(abs(dist)-25))*(1+abs(m1+m2)));
vel=round(vel)
if(vel<=50)
vel=50;
end

%****** Manda pra Labmate comandos das velocidades linear e angular
% set_vel(vel,wr);
tic;
while(toc<0.8)
end
% set_vel(vel,0);
dist_ant=dist;
vel_ant=vel;
return
```

Comando "matchpos(img,patual,NumRef)"

Compara a imagem capturada do corredor com o conjunto de imagens de referência correspondentes à posição atual e às duas posições seguintes à que o robô se encontra. O método de correlação aplicado é o *SSD*.

```
function [pos,ssd]=matchpos(img,patual,NumRef)
% function [pos,ssd]=matchpos(img,patual,NumRef)
% Executa o matching da imagem img com as imagens de referencia
% das posicoes atual e as duas seguintes.
% Utiliza para isto correlacao calculada atraves do
% SSD - sum of squared differences metric

cont=1;
if (patual>NumRef)
patual=1;
end
lim=patual+2;
```

```
if (lim>NumRef)
lim=NumRef;
end
for i=patual:lim
nome=['ref' num2str(i) '.bmp'];
ref=imread(nome);
SSD(cont)=ssd2d(img,ref);
x(cont)=i;
cont=cont+1;
end
figure;
plot (x,SSD,'k*');
set(gca,'position',[0.13 0.13 0.700 0.700])
[m,pos]=min(SSD);
ssd=SSD(pos);
pos=x(pos);
xlabel('k');
ylabel('SSD');
set(gca,'xtick',[0 1 2 3 4])
set(gca,'xlim',[0 4])
grid on
return
```

Comando "ssd2d(im1,im2)"

Efetua o cálculo do *SSD* em duas dimensões entre duas imagens.

```
function [out]=ssd2d (im1,im2)
% function [out]=ssd2d (im1,im2)
% Calcula o SSD - sum of squared differences metric entre duas imagens

[l,c]=size(im1);
im1 = double (im1);
im2 = double (im2);
N=l*c;
```

```
out=sqrt((sum(sum((im1-im2).^2)))/N);
return;
```

Comando "saveimg(cont,str)"

Rotina auxilar para salvar imagens do corredor para que se possa criar animações do movimento do robô.

```
function [cont]=saveimg(cont,str)
% function [cont]=saveimg(cont,str)
% Rotina auxiliar para a aquisicao e armazenamento das
% imagens de referencia
% OBS.: Esta rotina esta' adaptada para o trabalho com a Labmate, por
% isso, devera' ser alterada para ter uma aplicacao mais geral
[a,b,c,d]=acqmat(1);
nome=[str num2str(cont) '.gif']
acqmat(2,nome);
cont=cont+1;
%imshow([a,b,c,d]);
```

Comando "savedata"

Rotina auxiliar para salvar informações do estado do robô para posterior análise e criação de gráficos.

```
% Rotina para armazenamento de dados
cont=cont +1;
Status(cont,:)=get_stat;
Time_seg(cont)=etime(clock,tempo);
Wr(cont)=wr;
Posatual(cont)=posatual;
Pos(cont)=pos;
SSD(cont)=ssd;
Poly1(cont,:)=poly1';
Poly2(cont,:)=poly2';
```

```
Ptf(cont,:)=ptf';
Erro(cont)=dist_ant;
Covptf(cont)=covptf(2,2);
Std(cont)=sqrt(covptf(2,2));
VelAplic(cont)=vel;
```

Referências Bibliográficas

[1] Y. Aloimonos. Purposive and qualitative active vision. In *Proc. of the 10th. IEEE International Conference on Pattern Recognition*, Atlantic City, NJ - USA, June 1990.

[2] Y. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active vision. *International Journal of Computer Vision*, 1(4):333–356, January 1988.

[3] C. Andersen, S.D. Jones, and J.L. Crowley. Appearance based processes for visual navigation. In *Proc. of the 5th International Symposium on Intelligent Robotic Systems - SIRS97*, Stockholm, Sweden, July 1997.

[4] R. Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):996–1005, August 1988.

[5] S. Basri, E. Rivlin, and I. Shimshoni. Visual homing: Surfing on the epipoles. In *ICCV 98*, pages 863–869, Banbay, India, January 1998.

[6] T. F. Bastos. Sensores de proximidad en robotica. In *Reunion del Proyecto SISPER (Sistema de Percepcion Modular y Reconfigurable para Robotica)*, Maracaibo, Venezuela, September 7-11 1998.

[7] T. F. Bastos and V. Dynnikov. Aplicacion de robots y sensores en manufactura. In *Tercer Congreso Internacional de Manufactura*, Queretaro, Mexico, October 22-24 1998.

[8] G. Beccari, S. Caselli, F. Zanichelli, and D.Diemmi. Inducing topological maps from task-oriented perception and exploratory behaviors. In *Proc. of the Second Euromicro Workshop on Advanced Mobile Robots (EUROBOT'97)*, pages 134–140, Brescia, Italy, October 22-24 1997.

[9] M. Bertozzi and A. Broggi. Gold: a parallel real-time stereo vision system for generic obstacle and lane detection. *IEEE Transactions on Image Processing*, 7(1):62–81, January 1998.

[10] M. Bertozzi, A. Broggi, G. Conte, and A. Fascioli. Vision-based automated vehicle guidance: the experience of the argo vehicle. Technical report, Universita di Parma-Dipartimento di Ingegneria dell'Informazione, Parma, Italy, 1998.

[11] G. Bianco, R. Cassinis, A. Rizzi, N. Adami, and P. Mosna. A bee-inspired robot visual homing method. In *Proc. of the Second Euromicro Workshop on Advanced Mobile Robots (EUROBOT'97)*, pages 141–146, Brescia, Italy, October 22-24 1997.

[12] M. J. Black and P. Anandan. A framework for the robust estimation of optical flow. In *Proc. Fourth International Conference on Computer Vision - ICCV'93*, Berlim, Germany, May 1993.

[13] M. Buffa, O. Faugeras, and Z. Zhang. A stereo vision-based navigation system for a mobile robot. Technical Report RR-1895, INRIA-Institut National de Recherche en Informatique et en Automatic, Sophia-Antipolis, France, April 1993.

[14] C. Carreira and J. Santos-Victor. Vision based teleoperated cellular robots. In *Proc. of the 5th International Symposium on Intelligent Robotic Systems - SIRS97*, Stockholm, Sweden, July 1997.

[15] D. Coombs, M. Herman, Tsai Hong, and Marilyn Nashman. Real-time obstacle avoidance using central flow divergence and peripherical flow. Technical Report NISTIR 5605, National Institute of Standards and Technology - Intelligent Systems Division, Gaithersburg, 1995.

[16] V. Costa, M. Sentieiro, and J. Santos-Victor. Ciclope : a normal flow based approach for real time obstacle detection. Technical Report TR05/95, Instituto Superior Tecnico - Instituto de Sistemas e Robotica, Lisboa, Portugal, March 1995.

[17] A.J. Davison and D.W. Murray. Mobile robot localisation using active vision. In *Proc. of 5th European Conference on Computer Vision*, volume 1, pages 809–825, Freiburg, Germany, June 2-6 1998.

[18] B. Espiau, F. Chaumette, and Patrick Rives. A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8(3):313–326, June 1992.

[19] O. Faugeras. *Three-Dimensional Computer Vision- A Geometric Viewpoint.* MIT Press, 1993.

[20] O. Faugeras, S. Laveau, L. Robert, G. Csurka, and C. Zeller. 3d reconstruction of urban scenes from sequences of images. Technical Report RR-2572, INRIA-Institut National de Recherche en Informatique et en Automatic, Sophia-Antipolis, France, June 1995.

[21] X. Feng and W. Liu. Robot autonomous navigation. CDS Technical Report CDS97-009, California Institute of technology, 1997.

[22] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of ACM*, 24(6), 1981.

[23] E. O. Freire, T. F. Bastos-Filhos, J. E. M. Xavier, and H. A. Schneebeli. Agent-based ultrasonic sensing system for mobile robots. In *7th Latin-American Congress of Automatic Control*, pages 910–915, Buenos Aires, Argentina, September 1996. in portuguese.

[24] M. C. Garcia-Alegre and F. Recio. Basic agents for visual/motor coordination of a mobile robot. In *Proc. of the First International Conference on Autonomous Agents*, pages 429–434, Marina del Rey, CA, USA, February 1997.

[25] J. Gaspar. Visao para robotica movel: Deteccao de obstaculos sobre pavimento plano. Master's thesis, Istituto Superior Tecnico, Lisbon, Portugal, September 1994. in portuguese.

[26] D. Guinea, G. Sanchez, P. Bustos, and M. Garcia-Alegre. A distributed architecture for active perception in autonomous robots. In *Proc. of the IEEE International Conference on Systems, Man & Cybernetics*, pages 22–25, Vancouver, Brithish Columbia, Canada, October 1995.

[27] R. Haralick. Propagating covariance in computer vision. In *Proc. ECCV Workshop of Performance Characteristics of Vision Algorithms*, Cambridge, UK, April 1996.

[28] B. Horn. *Robot Vision*. MIT Press, 1986.

[29] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1–3):185–203, August 1981.

[30] S. Hutchinson, G.D. Hager, and P.I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, October 1996.

[31] D. P. Huttenlocher, M. E. Leventon, and W. J. Rucklidge. Visually-guided navigation by comparing edge images. In K. Goldberg, D. Halperin, J. Latombe, and R. Wilson, editors, *Algorithmic Foundations of Robotics*, pages 85–95. A K Peters, 1995.

[32] R. Jain, R. Kastwi, and B.G. Schunck. *Machine Vision*. McGraw-Hill, 1995.

[33] R. A. Jarvis. A perspective on range finding techniques for computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):122–139, March 1983.

[34] J. G. Jimenez and A. O. Baturone. Estimacion de la posicion de un robot movil. *Informatica y Automatica*, 29(4):3–18, 1996.

[35] J. Kosecka. Visually guided navigation. *Robotics and Autonomous Systems*, 21(1):37–50, July 1997.

[36] J. Kosecka and R. Bajcsy. Cooperation of visually guided behaviors. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, N. York, USA, June 1993.

[37] D. Kriegman, E. Triendl, and T. O. Binford. Stereo vision and navigation in buildings for mobile robots. *IEEE Transactions on Robotics and Automation*, 5(6):792–803, December 1989.

[38] B. C. Kuo. *Automatic Control Systems*. Prentice-Hall International Editions, 6th edition, 1991.

[39] S. H. Lai and B. C. Vemuri. Robust and efficient computation of optical flow. Technical Report TR-95-012, University of Florida - Computer and Information Sciences Department, Gainesville, Florida, April 1995.

[40] X. Lebegue and J.K. Aggarwal. Significant line segments for an indoor mobile robot. *IEEE Transactions on Robotics and Automation*, 9(6):801–806, December 1993.

[41] S. Li and S. Tsuji. Selecting distinctive scene features for landmarks. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, pages 53–58, Nice, France, May 1992.

[42] S. Li, S. Tsuji, and A. Hayashi. Qualitative representation of outdoor environment along route. In *Proceedings of ICPR'96*, pages 176–180, 1996.

[43] Y. Matsumoto, M. Inaba, and H. Inoue. Visual navigation using view-sequenced route representation. In *Proc. of the 1996 IEEE International Conference on Robotics and Automation*, pages 83–88, Minneapolis, Minnesota, USA, April 1996.

[44] F.G. Meyer. Time-to-collision from first-order models of the motion field. *IEEE Transactions on Robotics and Automation*, 10(6):792–798, December 1994.

[45] R. Mohr and B. TRiggs. *Projective Geometry for Image Analysis - A Tutorial given at ISPRS*. INRIA, Vienna, July 1996.

[46] J. Nielsen and G. Sandini. Learning mobile robot navigation: A behavior-based approach. In *IEEE Int. Conf. on Systems, Men and Cybernetics*, San Antonio, Texas, USA, October 1994.

[47] G. Sandini, F. Gandolfo, E. Grosso, and M. Tistarelli. Vision during action. In Y. Aloimonos, editor, *Active Perception*. Lawrence Erlbaum Associates, 1993.

[48] J. Santos-Victor and G. Sandini. Docking behaviors via active perception. In *Proc. of the International Symposium on Intelligent Robotic Systems - SIRS95*, Pisa, Italy, July 1995.

[49] J. Santos-Victor and G. Sandini. Visual-based obstacle detection : a purposive approach using the normal flow. In *Proc. of the International Conference on Intelligent Autonomous Systems - IAS95*, Karlsruhe, Germany, March 1995.

[50] J. Santos-Victor and G. Sandini. Embedded visual behaviors for navigation. *Robotics and Autonomous Systems*, 19(3-4):299–313, March 1997.

[51] J. Santos-Victor and G. Sandini. Visual behaviors for docking. *Computer Vision and Image Understanding*, 67(3):223–238, September 1997.

[52] J. Santos-Victor, G. Sandini, F. Curotto, and S. Garibaldi. Divergent stereo in autonomous navigation : From bees to robots. *International Journal of computer Vision*, 14:159–177, 1995.

[53] J. Santos-Victor, R. F. Vassallo, and H. J. Schneebeli. Topological maps for visual navigation. In *Proc. of the 1st International Conference on Computer Vision Systems*, Las Palmas, Canarias, Spain, January 1999.

[54] H. Schneiderman and M. Nashman. A discriminating feature tracker for visual-based autonomous driving. *IEEE Transactions on Robotics and Automation*, 10(6):769–775, December 1994.

[55] M. W. Spong and M. Vidyasagar. *Robot Dynamics and Control*. John Wiley and Sons, 1989.

[56] C. Taylor and D. Kriegman. Vision-based motion planning and exploration algorithms for mobile robots. In K. Goldberg, D. Halperin, J. Latombe, and R. Wilson, editors, *Algorithmic Foundations of Robotics*, pages 69–82. A K Peters, 1995.

[57] R. F. Vassallo, H. J. Schneebeli, and J. Santos-Victor. A purposive strategy for visual-based navigation of a mobile robot. In *41th Midwest Symposium on Circuits and Systems*, Notre Dame, Indiana, USA, August 1998.

[58] R. F. Vassallo, H. J. Schneebeli, and J. Santos-Victor. Visual navigation: Combining visual servoing and appearance based methods. In *Proc. of the 6th Intl. Symp. on Intelligent Robotics Systems - SIRS98*, Edinburg, Scotland, July 1998.

[59] L. A. Vicente. *Estrategias de Correspondencia jerarquica y metodos deirectosde autocalibracion para un sistema estereoscopico binocular*. PhD thesis, Instituto de Automatica Industrial, Madrid, Spain, October 1996.

[60] G. Wichert. Mobile robot localization using a selforganized visual environment representatio. In *Proc. of the Second Euromicro Workshop on Advanced Mobile Robots (EUROBOT'97)*, pages 29–36, Brescia, Italy, October 22-24 1997.

[61] J. E. Xavier and H. A. Schneebeli. A structure for constructing agent-based control systems for mobile robots. In *2nd Brazilian Symposium on Intelligent Automation*, pages 97–102, Curitiba, Brazil, September 1995. in portuguese.

[62] C. Zeller and O. Faugeras. Camera self-calibration from video sequences: the kruppa equations revisited. Technical Report RR-2793, INRIA-Institut National de Recherche en Informatique et en Automatic, Sophia-Antipolis, France, February 1996.

[63] J. Y. Zheng and S. Tsuji. Panoramic representation for route recognition by a mobile robot. *International Journal of Computer Vision*, 1(9):55–76, 1992.