ESTE EXEMPLAR CORRESPONDE A REDAÇÃO FINAL DA TESE DEFENDIDA POR LUCIANA DIÓGENES

..... E APROVADA

PELA COMISSÃO JULGADORA EM 30/07/2008

.....
ORIENTADOR

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA MECÂNICA

COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

Uso de Filtro de Kalman e Visão Computacional para a Correção de Incertezas de Navegação de Robôs Autônomos

Autor: Luciana Claudia Martins Ferreira Diogenes
Orientador: Prof. Dr. Paulo Roberto Gardel Kurka
Co-Orientador: Prof. Dr. Helder Anibal Hermini

07/08

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA MECÂNICA

COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

DEPARTAMENTO DE PROJETO MECÂNICO

Uso de Filtro de Kalman e Visão Computacional para a Correção de Incertezas de Navegação de Robôs Autônomos

Autor: Luciana Claudia Martins Ferreira Diogenes
Orientador: Prof. Dr. Paulo Roberto Gardel Kurka
Co-Orientador: Prof. Dr. Helder Aníbal Hermini

Curso: Engenharia Mecânica
Área de Concentração: Mecânica dos Sólidos e Projeto Mecânico

Trabalho Final de Doutorado apresentado à comissão de Pós Graduação da Faculdade de Engenharia Mecânica, como requisito para a obtenção do título de Doutora em Engenharia Mecânica.

Campinas, 2008
S.P . – Brasil

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

Diogenes, Luciana Claudia Martins Ferreira
D621u Uso de filtro de Kalman e visão computacional para a correção de incertezas de navegação de robôs autônomos / Luciana Claudia Martins Ferreira Diogenes. -- Campinas, SP: [s.n.], 2008.

Orientadores: Paulo Roberto Gardel Kurka, Helder Anibal Hermini.
Tese de Doutorado - Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Navegação de robôs móveis. 2. Visão Robo. 3. Veículos Autônomos. 4. Robos - Sistema de controle. 5. Kalman, Filtragem de. I. Kurka, Paulo Roberto Gardel. II. Hermini, Helder Anibal. III. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. IV. Título.

Titulo em Inglês: Use of Kalman filter and computational vision for the correction of uncertainties in the navigation of autonomous robots
Palavras-chave em Inglês: Navigations, Robotic Vision, Vehicle Autonomous, Control
Área de concentração: Mecânica dos Sólidos e Projeto Mecânico
Titulação: Doutor em Engenharia Mecânica
Banca examinadora: Paulo Roberto Gardel Kurka, Humberto Ferasoli Filho, Cleudmar Amaral de Araújo, Eurípedes Guilherme de Oliveira Nobrega, Milton Dias Júnior
Data da defesa: 30/07/2008
Programa de Pós Graduação: Engenharia Mecânica

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA MECÂNICA

COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

DEPARTAMENTO DE PROJETO MECÂNICO

TESE DE DOUTORADO

Uso de Filtro de Kalman e Visão Computacional para a Correção de Incertezas de Navegação de Robôs Autônomos

Autor: **Luciana Claudia Martins Ferreira Diogenes**
Orientador: **Paulo Roberto Gardel Kurka**
Co-Orientador: **Helder Anibal Hermini**

A Banca Examinadora composta pelos membros abaixo aprovou esta Tese:

Prof. Dr. Paulo Roberto Gardel Kurka, Presidente
DPM/FEM/UNICAMP

Prof. Dr. Humberto Ferasoli Filho
Unesp/Bauru

Prof. Dr. Cleudmar Amaral de Araújo
UFU/Uberlândia

Prof. Dr. Eurípedes Guilherme de Oliveira Nobrega
DMC/FEM/UNICAMP

Prof. Dr. Milton Dias Júnior
DPM/FEM/UNICAMP

Campinas, 30 de julho de 2008.

Dedicatória

Dedico este trabalho a universidade da qual tenho muito orgulho: a Unicamp. Durante minha caminhada acadêmica, foram dez anos me dedicando a pesquisa nessa instituição: graduação, mestrado e com esse trabalho mostrado aqui, finalizo o doutorado. Dez anos convivendo com os grandes pesquisadores do Brasil, os quais me embriagaram de sabedoria e conhecimento durante as diversas conversas nessa universidade.

Agradecimentos

Agradeço ao meu orientador, prof. Paulo Kurka, pela oportunidade dada para que esse trabalho fosse realizado, pelas horas de discussão e por todo o apoio, principalmente em minha ida a Portugal, inicialmente um pouco confusa, mas sempre esteve empenhado para que todos os problemas fossem resolvidos. Além disso, o parabenizo por ser um excelente profissional, sempre concentrado e dedicado aos nossos trabalhos.

Agradeço a CAPES, agente financiadora deste trabalho tanto no Brasil como em Portugal.

Agradeço ao prof. Fernando Lobo por todo o apoio que pode me conceder na Universidade do Porto, em Portugal, durante um ano que fiz meu doutorado sanduíche.

Ao prof. Helder Hermini, agradeço por ter me dado a primeira oportunidade e incentivo de entrar na faculdade de Engenharia Mecânica da Unicamp.

A equipe de trabalho dos orientados do prof. Kurka: Eduardo, Elvira, Jaime, Karen e Luiz.

A todos os meus amigos e amigas que sempre estiveram comigo e que me presentearam com suas amizades, me trazendo alegria e conforto em todos os momentos que precisei.

Agradeço aos meus pais, Telma e Francisco, por estarem lá na arquibancada da minha vida, sempre torcendo por mim e me dando apoio a todo o momento.

E por fim, ela que é mais que especial, minha querida e amada vovó Marieta, pelo seu infinito amor, bondade, carinho e por a mestra que não me indica só o caminho certo, mas que me dá o atalho para que eu possa chegar mais rápido nele.

"A mudança é a lei da vida. E aqueles que apenas olham para o passado ou para o presente irão com certeza perder o futuro."

(John Kennedy)

Resumo

DIOGENES, Luciana Claudia Martins Ferreira, *Uso de Filtro de Kalman e Visão Computacional para a Correção de Incertezas de Navegação de Robôs Autônomos,* Campinas: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2008. 140 p. Tese (Doutorado).

O presente trabalho tem como finalidade estabelecer um conjunto de procedimentos básicos de navegação e controle de robôs autônomos, baseado em imagens. Mapas de intensidade provenientes das imagens de duas câmeras são convertidos em mapas de profundidade, que fornecem ao robô informações sobre o seu posicionamento em um ambiente composto de objetos distintos. O modelo de robô de duas rodas com acionamento diferencial é usado, permitindo que o processo de navegação se dê através da fusão sensorial das informações obtidas pelas câmeras e dos dados de odometria do seu movimento. O filtro linear de Kalman é usado nesse processo de fusão, para obter estimativas ótimas de posição do robô, baseados nas imagens observadas pelas câmeras e pela informação de odometria medida pelos *encoders* de rotação das rodas. Realizam-se simulações computacionais da tarefa de obtenção e processamento da imagem de um ambiente bidimensional simplificado, bem como do procedimento de navegação com fusão sensorial. As simulações têm por finalidade testar a viabilidade e robustez do procedimento de navegação, na presença de incertezas nas medidas de posição através de câmeras bem como nas medidas de odometria.

Palavras Chave:

- Navegação para robôs móveis, visão robô, veículos autônomos, robôs-sistema de controle, Kalman, filtragem de .

Abstract

DIOGENES, Luciana Claudia Martins Ferreira, *Use of Kalman Filter and Computational Vision for the Correction of Uncertainties in the Navigation of Autonomous Robots,* Campinas: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2008. 140 p. Doctor Thesis.

The work establishes a collection of procedures for image based navigation control of an autonomous robot. Intensity maps obtained from cameras are transformed in depth maps, which provide information about the robot's localization in an environment, comprised of distinct objects. A two wheeled, differential powered robot model is used, allowing the navigation process to combine double source information from the camera and odometry sensors. The Kalman filter technique is used in this information combination in order to yield optimal estimates of position of the robot, based on the camera and odometry information. Computational simulations are used to validate the image capture and processing, as well as the sensorial fusion technique, in a simplified bi-dimensional environment. The simulations are also useful in accessing the viability and robustness of the navigation process, in the presence of measurement uncertainties associated to the camera and odometry measurements.

Keywords:

Navigation for mobile robots, robot vision, vehicle autonomous, robots- control system, Kalman filter.

Sumário

Lista de Figuras

Lista de Tabelas

Nomenclatura

$\mathbf{X}_k$: pose real na k.

$\mathbf{Z}_k$: pose medida por visão na k.

$\hat{\mathbf{X}}_k^-$: pose apriori na k.

$\hat{\mathbf{X}}_k$: pose aposteriori na k.

σ : desvio padrão do erro de odometria.

$\overline{\sigma}$: desvio padrão do erro de câmera.

Q: matriz covariância do ruído do processo.

$\overline{\mathbf{R}}$: matriz covariância do ruído de medida.

R: matriz rotação.

NP: nova passagem ou portal.

NPins: nova passagem instantânea, ou, portal instantâneo.

O: objetivo ou porta.

γ : ângulo formado pela reta que une dois pontos em relação ao eixo X do mundo.

α : ângulo formado pela reta que une dois pontos em relação ao eixo z do robô.

Z_ϕ : o ângulo entre z (eixo z do robô) e X (eixo X do mundo) .

.

Capítulo 1

Introdução

O termo "robô" tem sua origem nas línguas européias orientais, especialmente no idioma tcheco (robota significa trabalho forçado, escravidão) (Tomatis, 2001).

Já a palavra "navegar" (Siciliano, 2006) vem do latim navigare: *navis* + *igare* = navio + dirigir, ou seja, no entendimento mais direto da palavra, navegar é percorrer com um navio o mar. Porém, analisado em sentido mais amplo, o conceito de navegar se confunde com o de guiar e pode ser entendido como conduzir um veículo de um ponto a outro, usando recursos próprios.

Nesse contexto surge a "navegação robótica móvel", dedicada a pesquisar e desenvolver métodos que permitam aos robôs se locomoverem de forma autônoma de um ponto de origem a um ponto de destino, cumprindo uma tarefa específica.

1.1 – Motivação

A navegação autônoma de robôs é um dos problemas de engenharia mais complexos, desafiadores e propícios à pesquisa de sistemas autônomos inteligentes. O problema em uma maneira fácil de se traduzir consiste em desenvolver estratégias de tomada de decisão para um ou mais robôs móveis, os quais são inseridos em um ambiente arbitrário junto ao qual devem atuar de forma autônoma, visando cumprir certas tarefas. Muito embora a navegação de robôs possa ser descrita com base nesta breve definição, existem muitos aspectos de projeto envolvidos:

configurações do ambiente, modelo do robô, elenco de tarefas e critérios de desempenho. Sendo assim, o desenvolvimento de sistemas de navegação autônomos envolve desafios extremamente complexos para qualquer projeto (Cazangi, 2004).

Para que um robô seja capaz de ir de um ponto ao outro, ele precisa saber sua posição em um dado ambiente e o caminho que ele precisa percorrer até o objetivo. Através dessas informações, o robô é capaz de planejar e executar a tarefa de se deslocar até um local específico. Tal conjunto de condições é chamado de navegação (Heinen, 2000). Há uma classificação em três grupos para a navegação robótica: global, local e híbrida. A seguir são apresentadas as definições de navegação:

Navegação global: na navegação global, o robô planeja o caminho que irá percorrer baseado no posicionamento absoluto ou relativo a um mapa de ambiente. O robô irá se movimentar através dessas informações. Com a navegação global, estima-se a posição do robô sem nenhum conhecimento de sua posição inicial e há habilidade do robô se relocalizar quando este se perde no ambiente.

Navegação local: na navegação local, a posição é sempre relativa a objetos (estáticos ou em movimento) do ambiente. A navegação por imagens é considerada local pois ela será relativa a pontos de referência contidos em uma sequência de imagens.

Navegação híbrida: nesse tipo de navegação existe a integração da navegação global e local. Assim o robô é capaz de planejar seu deslocamento e também pode se desviar de obstáculos presentes no caminho.

O uso de robôs móveis vem se tornando cada vez mais comum para substituir operações onde envolvam seres humanos, seja para facilitar o trabalho aumentando o seu rendimento ou substituindo as pessoas em operações que envolvam risco de morte. São aplicados em várias ocasiões como: no transporte para levar cargas, vigilância, limpeza de casas, exploração espacial, auxílio a deficientes físicos, entre outros diversos.

Um exemplo de aplicação de um robô móvel é o robô Sojourner (figura 1.1) usado na exploração do planeta Marte (Wilt et al., 1997).

Figura 1.1: Robô Sojourner..

Em casos onde ambientes são considerados perigosos onde é colocada em risco a vida do ser humano, como por exemplo um campo radioativo, o robô torna-se uma ferramenta essencial para substituir a penetração de uma pessoa nesse tipo de ambiente. O robô Pioneer (figura 1.2) é um exemplo de robô usado em exploração radioativa (Josias, 1960).

Figura 1.2: Robô Pioneer.

Para a navegação de robôs móveis é necessário extrair informações do ambiente de maneira que essas possam dar subsídios para a navegação. A aquisição de informação é feita através de sensores. A escolha do sensor mais adequado depende do tipo de tarefa de navegação que se pretende implementar e torna a navegação mais autônoma (Deccó, 2004).

Inúmeros sensores são utilizados, como por exemplo, lasers, infravermelho, GPS, ultra-som bússolas, encoders e câmeras. Sensores de ultra-som e infravermelho são sensíveis aos objetos refletores, tornando-se úteis para medir distâncias quando ocorre a reflexão. Sensor de infravermelho possui boa resolução angular, tornando-se útil para detectar a presença ou ausência dos objetos. Para o sistema de localização do robô, GPS ou encoders são usados. Sensores baseados em câmeras são capazes de capturar imagens de forma contínua e fornecer a posição dos objetos.

Para proporcionar uma navegação autônoma, os sensores têm que ter a capacidade de extrair as informações do ambiente. Aumentando a quantidade de informações, por meio de fusão de sensores, torna uma navegação com mais autonomia.

Existem dois tipos de classificação para a localização de robôs móveis: métodos de localização relativa e métodos de localização absoluta (Borenstein,et al, 1996). Os métodos de localização relativa utilizam as localizações obtidas em instantes anteriores para estimar a localização atual do robô, como a odometria por exemplo. Porém, a odometria pode gerar erros que se propagam cumulativamente com o tempo (Valgas, 2002), o que torna este método ineficiente quando utilizado isoladamente. Já os métodos de localização absoluta, como as câmeras, utilizam apenas as informações atuais dos sensores para determinar a localização do robô em relação a um referencial fixo absoluto. Uma localização absoluta pode ser obtida através do reconhecimento de marcos no ambiente, que apesar de não gerar erros cumulativos, pode necessitar de um tempo de processamento bem maior do que o uso de odometria.

Neste trabalho, os sensores empregados para localização de um robô autônomo móvel são câmeras e encoders. Esses dois tipos de sensores são de custo baixo e eficientes quando usados em conjunto. A câmera pode ser usada para corrigir o erro acumulativo do enconder, já que ela dá a medida absoluta. Porém, o robô estando perante objetos distantes, o encoder pode dar uma

informação mais precisa. Torna-se necessário neste tipo de navegação, a fusão sensorial, buscando a melhor informação de localização.

1.2 - Revisão da Literatura

Para navegar de forma confiável em ambientes internos (indoors) é necessário que o robô conheça sua localização (posição e orientação) dentro do ambiente. Uma localização correta não se restringe a ajudar na navegação, mas também para a exploração e para a adaptação do mapa do ambiente pelo robô. A estimativa da localização de um robô baseado em dados sensoriais é um dos problemas fundamentais da robótica móvel.

Diversos trabalhos têm sido escritos sobre a fusão de vários sensores, incluindo visão, para construir um mapa do ambiente para a navegação de robôs móveis (Hebert e Tanade, 1988; Kriegman, et al., 1989; Thorpe, et al., 1988; Turk et al., 1988; Waxman, 1987). A principal proposta de tais trabalhos é obter uma precisão maior na determinação da posição do robô. Em muitos outros trabalhos, a câmera é usada para investigar o problema relacionado à estabilização de trajetória.

Em Costa, et al., 2003 há a descrição de um robô que utiliza imagens para seguir a trajetória desenhada em uma superfície. O centro da pista é calculado a partir da imagem coletada e uma atitude é tomada, ou seja, um sistema de controle decide a direção do robô. Nesse caso, somente uma câmera é utilizada para a obtenção das imagens.

Em Castro, et al., 2001 são desenvolvidos três sistemas de navegação autônoma baseados em imagens, utilizando lógica nebulosa e redes neurais artificiais para determinar a direção a ser seguida e a velocidade a ser empregada. Um robô navega com o uso de uma câmera sobre uma pista para controlar a trajetória e as informações extraídas da faixa da pista são obtidas para cálculo de velocidade. O ângulo das faixas é medido por um algoritmo detector de faixas (ADF) baseado em autômato finito (AF) que analisa os níveis de cinza da imagem. Dois vetores contendo as coordenadas pertencentes a ambas as faixas são obtidos no final do processo e o ADF calcula a inclinação das faixas da esquerda e da direita.

Um sistema de inferência nebulosa (SIN) foi desenvolvido para controlar a direção e a velocidade transformando o valor numérico dos ângulos das faixas em variáveis numéricas como muito a esquerda e muito a direita.

A rede neural determina a direção a ser tomada para cada ponto da pista. Duas arquiteturas de redes neurais foram utilizadas: uma rede com perceptrons múltiplas camadas (PMC) e rede de funções de base radial (FBR). Utilizou-se a retro propagação de erro como o treinamento das redes.

Para efetuar o mapeamento da velocidade, os três sistemas descritos tiveram a implementação de dois subsistemas: um linear e outro não linear. O melhor desempenho foi obtido nos sistemas baseados em redes neurais.

No trabalho de Bianchi, et al., 2001 é descrito o projeto de um sistema para controle de um robô móvel. O robô se move sobre um caminho desenhado baseado nas informações visuais e se desviando dos obstáculos usando sensores infravermelhos. O mini-robô Khepera possui oito sensores, cada um com um emissor e receptor de luz infravermelha que permite medir a proximidade dos objetos através da reflexão da luz.

Para a aquisição de imagens é usada apenas uma câmera sobre o robô. Um objeto pode ter suas cores decompostas no espaço RGB através de um gráfico tridimensional. A técnica conhecida como imposição de limiares agrupa as regiões no espaço 3D que tenham a mesma cor, em um paralelepípedo, sendo necessário a fixação de limites nesses paralelepípedos para definir uma cor. Assim que são identificadas as cores, então, identificam-se também os objetos monocromáticos.

Sobre o robô foi inserido um rótulo contendo dois círculos de diferentes cores: um no centro de gravidade do robô e um outro um pouco mais a frente, cada um com uma cor específica. O vetor de direção do robô é um vetor que passa pelos dois centros circulares, com os centros a serem determinados pela identificação das cores (técnica de imposição de limiares) e utilizado o processo de aglutinação dos pixels vizinhos conhecido por “crescimento da região”.

Para a determinação da direção do caminho a ser seguido fixou-se um quadrado com centro de gravidade coincidente com a do centro do robô. O quadrado passa duas vezes pelo caminho a ser seguido, uma região pela a qual o robô passou e outra que ainda passará. O vetor de direção desejada é o vetor que une o centro geométrico do quadrado com o ponto que ele intercepta o caminho a frente. O deslocamento do robô é extraído das informações entre ambos os vetores.

Em Schmidt and Lages, 2000 é desenvolvido um método baseado em informações visuais de um corredor para a navegação de um robô autônomo utilizando a transformada de Hough.

O primeiro passo é identificar visualmente as laterais de um corredor e localizar o seu centro. A localização das laterais é feita aplicando a definição de vetor gradiente. Dada a imagem de gradiente, utiliza-se a transformada de Hough. No caso do corredor, procura-se por duas retas.

A transformada de Hough (HT) é um método corrente para a detecção de formas que são facilmente parametrizadas em imagens computacionais (linhas, círculos, elipses, etc.) (Duda e Hart, 1972; Forsberg et al., 1995). Em geral, a transformada é aplicada após a imagem sofrer um pré-processamento, mais comumente a detecção de bordas, em que se consegue descrever analiticamente uma curva para utilizar essa transformada.

Uma variação do método da média dos máximos é utilizada no processo de defuzzyficação, levando-se em conta a área de abrangência de cada função de pertinência como peso na composição da média. Com base nas informações de média dos máximos, determinam-se os parâmetros da função de pertinência. Utiliza-se a função de custo para sintonizar o controlador a trajetórias suaves ou não.

Em Ma et al., 1999 é considerado um robô móvel que segue uma curva analítica e arbitrária no plano e o controle está justamente nos parâmetros da curva na imagem. A cinemática é dada considerando um robô uniciclo. Uma câmera é montada sobre o robô a uma distância do solo e com um ângulo de inclinação onde tal câmera faz a captação da imagem da curva.

É proposto também ao robô seguir pedaços de curvas contínuas (não necessariamente analíticas) por aproximações de curvas suaves analíticas, ou seja, seguir uma curva virtual não vista na imagem usando as leis de controle, devido a melhor economia computacional. Um exemplo é apresentado, fazendo o robô móvel seguir uma intersecção de retas, que no caso, poderia ser também a intersecção de corredores.

O trabalho de Fang et al., 2005 é semelhante ao que está descrito em Chen et. al, 2006, mas com a utilização de um referencial a menos.

Em Chen et al., 2006 descreve-se um controle de sistema visual para um robô móvel com uma câmera montada sobre ele. A trajetória desejada é encontrada através de três imagens: uma no instante k, outra no estado k-1 e uma no estado estacionário através das relações de projeções. O objeto a ser visualizado é uma estrutura rígida que contém três leds, que possuem intensidades que contrastam com o fundo. Um simples algoritmo de "thresholding" é usado para determinar as coordenadas do centro desses leds. A informação dada pela homografia euclidiana é usada para o desenvolvimento do controle.

Em Maia, 2004 aviões deslocam-se livremente enquanto permanecerem fora da zona de proteção e são regidos pelas equações de movimento. Quando um dos aviões invade a zona de proteção do outro há a necessidade de atuação de um controle. Na tentativa de fazer com que um

avião saia da zona de proteção, o controlador deve atuar sobre as velocidades dos dois aviões. A alteração no movimento do avião é uma maneira de descrever um controle híbrido.

1.3 – Objetivos do Presente Trabalho

O presente trabalho tem como objetivo a escolha e simulação de um conjunto de técnicas que utilizem o processamento de imagens provenientes de um par de câmeras na tarefa de navegação e controle de trajetória de um robô autônomo móvel.

Um projeto de pesquisa de controle de robôs móveis através do uso de imagens, em parceria com a Faculdade de Engenharia da Universidade do Porto, recebeu o nome de Projeto PILGRIM, (apenas divulgado internamente no grupo de pesquisa do prof. Paulo Kurka, ainda não oficializado). Nesse projeto, estão englobados este trabalho e outros trabalhos dos alunos do prof. Paulo Kurka que englobam um sistema de navegação de um robô de duas rodas com acionamento diferencial. O nome fictício "PILGRIM" é dado ao robô simulado, e num futuro próximo, será o nome do robô construído. PILGRIM, que em inglês significa peregrino, representa um robô em peregrinação, isto é, em navegação. A plataforma robótica proposta em tal projeto combina dois tipos de sensores: duas câmeras e um *encoder*. A técnica de controle implementada utiliza o filtro de Kalman (Choomuang and Afzulpurkar, 2005; Xia, et al., 2006; Moballegh, et al., 2004; Sousa, et al., 2005; Surrécio, et al., 2005; Kriegman, et al., 1989; Karlsson, et al., 2005). Tem-se a finalidade de assegurar alguma precisão de estimativa da posição de um robô autônomo que navega em um ambiente com obstáculos. Esses obstáculos servem também como pontos de referência pelos quais o robô se guia para sair de sua posição inicial e chegar até um objetivo final, percorrendo trajetórias retilíneas ideais. Para a chegada ao objetivo final, ou alvo, o robô utiliza "portais" de passagem, escolhidos dentre os obstáculos presentes em sua trajetória.

O robô em sua posição inicial colhe informações do ambiente a sua frente e busca localizar os objetos que são vistos por ambas as câmeras. Os objetos em comum localizados por um processamento dos mapas de intensidade, são projetados no espaço, dando origem a um mapa de profundidade que permite a determinação dos portais de passagem do robô.

Pretende-se futuramente abordar esse problema com a metodologia de sistemas híbridos, pois o mesmo envolve dois diferentes tipos de interações: um estado dinâmico contínuo e um estado dinâmico discreto onde os componentes discretos são os obstáculos que interrompem o movimento

do robô, mudando a sua direção, sua forma de prosseguir no ambiente. O caráter dinâmico do movimento é descrito pela locomoção do robô até um ponto já determinado dentro do ambiente.

A estrutura de apresentação do presente trabalho é descrita a seguir.

O modelo cinemático de movimento com o uso de odometria para a localização de posição, para um robô constituído de duas rodas, é mostrado no capítulo 2.

No capítulo 3 é descrito o método de projeção de objetos no espaço, através de imagens. Pontos correspondentes em duas imagens diferentes representam o mesmo ponto no espaço, e mostra-se que esses pontos podem ser representados tridimensionalmente pela geometria epipolar.

No capítulo 4 é mostrado o procedimento simplificado de localização do robô no espaço bidimensional, dando sua posição e sua orientação no referencial do mundo. O ambiente bidimensional produz imagens unidimensionais. Sendo assim, o procedimento apresentado no capítulo 3 torna-se mais simples e é mostrado como a posição do robô é encontrada através da imagem de pontos correspondentes,

No capítulo 5 é mostrado o processo de estimativa da posição ótima do robô a partir da fusão de dois sensores, (câmeras e *encoders*). Para estimar tal posição ótima, faz-se uso do filtro de Kalman.

No capítulo 6 é mostrada a teoria de controle híbrido. O controle híbrido pode ser usado na navegação para provocar mudança na estratégia de controle do robô, sempre que se deseja alterar alguma prioridade em seu objetivo de navegação.

No capítulo 7 são apresentadas as simulações numéricas de um robô navegando em um ambiente, com estimativa de localização através dos objetos presentes no caminho, via câmeras e por informações provenientes de *encoders*. Além disso, dois outros modelos são propostos, um dos quais envolvendo controle híbrido

Na primeira simulação, o ambiente é constituído de dois pontos. Numa segunda simulação, o ambiente é constituído de circunferências representativas de obstáculos e através do mapa de intensidade é possível encontrar o mapa de profundidade e localizar os pontos de extremo dos objetos vistos por ambas as câmeras. Em ambas as simulações pode-se analisar os resultados da navegação, variando-se os erros associados à visão das câmeras e à odometria.

No capítulo 8 são apresentadas as conclusões e sugestão de trabalhos futuros.

Capítulo 2

Navegação de Robôs Autônomos Através de Odometria

Nesse capítulo é apresentado o modelo cinemático do robô autônomo de duas rodas, considerado no presente trabalho. Além da equação cinemática de deslocamento do robô, é feito o cálculo da estimativa de erro de posicionamento, baseado na incerteza das medidas de odometria.

2.1 Modelo de Movimento do Robô

A posição (X,Z) e orientação (θ) de um robô navegando em um plano pode ser determinada pelo deslocamento de suas rodas, através do método conhecido como odometria (Bezerra, 2004; Victorino, 1998). Utilizam-se sensores (*encoders*) que medem a rotação das rodas e desse modo pode-se calcular o seu deslocamento.

No presente trabalho é considerado o emprego de *encoders* óticos incrementais (Santos, et al, 2002; Plácido, 2005).

Um exemplo simplificado de odometria é apresentado na figura (2.1). Um robô se encontra no instante inicial t_0, na posição X_0, e inicia a sua navegação pelo ambiente. Depois de um tempo t_1, a posição medida é X_1, que pode ser calculada como a soma da posição anterior X_0 e o deslocamento d_1, que ocorreu entre os tempos t_0 e t_1. No deslocamento posterior, no tempo t_2, a posição medida é X_2, dada pela soma do deslocamento anterior d_2 e pela posição anterior X_1. Assim, a posição atual pode ser calculada com base no acúmulo dos deslocamentos efetuados pelo robô, com relação a posição inicial X_0.

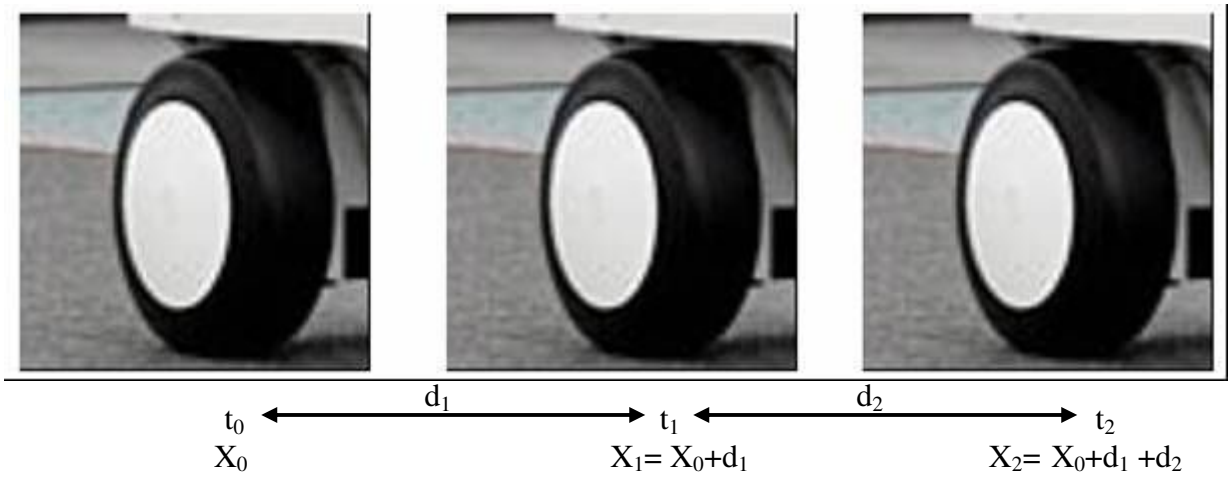


Figura.2.1: Utilização da odometria.

O processo de odometria, entretanto, pode conter erros que se propagam acumulativamente com o tempo, o que torna este método ineficiente quando utilizado isoladamente.

Neste estudo é usado um modelo de um robô constituído por duas rodas laterais, acionadas por motores independentes. A posição (X,Z) e a orientação (θ) são obtidas a partir do ponto médio do eixo que une as rodas, em relação a um referencial fixo, fora do robô, conforme mostrado na figura (2.2). Ainda na figura, considere-se:

- r_e e r_d como os raios da roda da esquerda e da direita, respectivamente;
- v_e e v_d como as velocidades dos centros das rodas da esquerda e da direita, respectivamente;
- ω_e e ω_d como as velocidades angulares da roda da esquerda e da direita, respectivamente;
- b como o comprimento do eixo de união das rodas;
- v como a velocidade linear do centro de gravidade do robô, que fica equidistante das duas rodas sobre o eixo que as une;
- ω como a velocidade angular do robô.

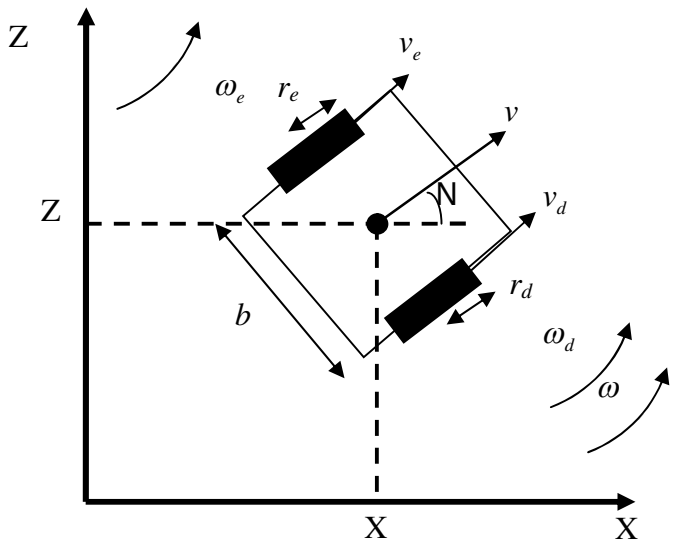


Figura 2.2: Modelo do robô.

Considerando-se o movimento instantâneo do robô como uma circunferência de raio r, mostrado na figura (2.3) tem-se que a velocidade linear do ponto situado entre as rodas é dada por:

$$v = \omega r \tag{2.1}$$

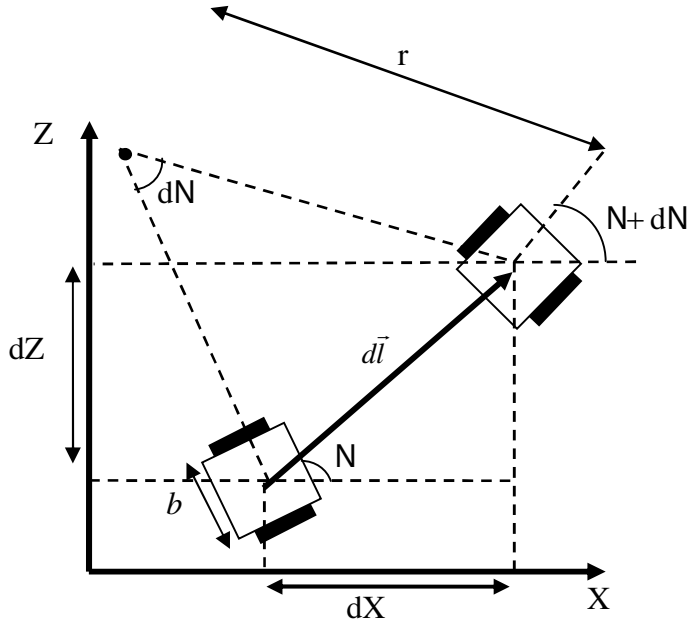


Figura 2.3: Movimento do robô de duas rodas .

A velocidade υ no centro de gravidade é dada pela velocidade das rodas da direita e da esquerda:

$$v = \frac{v_e + v_d}{2} \tag{2.2}$$

A velocidade do robô dada em função das duas rodas é encontrada substituindo-se (2.1) em (2.2). A velocidade angular é dada por:

$$\omega = \frac{v_d - v_e}{b} \tag{2.3}$$

O vetor de deslocamento do robô em um tempo infinitesimal dt, é dado por $d\vec{l}$, sendo decomposto em duas componentes dX e dZ. Tal vetor vem do uso da aproximação de $d\vec{l}$ por $d\vec{r}$, o qual é o vetor que realmente indica deslocamento entre duas posições sucessivas. Essas componentes bem como o deslocamento angular infinitesimal $d\theta$, são dados por:

$$\begin{aligned} dX &= dlCos(\theta) = vdtCos(\theta) \\ dZ &= dlSen(\theta) = vdtSen(\theta) \\ d\theta &= \omega dt \end{aligned} \tag{2.4}$$

As equações (2.4) são discretizadas pelo método de Euler, levando a:

$$\begin{aligned} X(t+\Delta t) &= X(t) + v\Delta tCos(\theta(t)) \\ Z(t+\Delta t) &= Z(t) + v\Delta tSen(\theta(t)) \\ \theta(t+\Delta t) &= \theta(t) + \omega\Delta t \end{aligned} \tag{2.5}$$

As velocidades de ambas as rodas, no deslocamento infinitesimal, são determinadas como:

$$v_d = \frac{dl_d}{dt} = \frac{\Delta l_d}{\Delta t}$$

$$v_e = \frac{dl_e}{dt} = \frac{\Delta l_e}{\Delta t} \tag{2.6}$$

Seja N_{res} o número de passos do *encoder* que corresponde a uma volta completa da roda do robô. O número de passos medidos pelos encoders quando um número arbitrário de volta é imposto às rodas direita e esquerda é dado por N_d e N_e, respectivamente. Assim, os deslocamentos são dados por:

$$\begin{aligned} \Delta l_d &= \frac{N_d}{N_{res}} 2\pi r_d \\ \Delta l_e &= \frac{N_e}{N_{res}} 2\pi r_e \end{aligned} \tag{2.7}$$

Substituindo-se (2.7) em (2.6), tem-se:

$$\begin{aligned} v_d &= \frac{N_d}{N_{res}} \frac{2\pi r_d}{\Delta t} \\ v_e &= \frac{N_e}{N_{res}} \frac{2\pi r_e}{\Delta t} \end{aligned} \tag{2.8}$$

A localização do robô, dadas as medidas dos *encoders*, pode ser encontrada substituindo (2.8) em (2.5):

$$\begin{aligned} X(t+\Delta t) &= X(t) + (N_d r_d + N_e r_e)\frac{\pi}{N_{res}} Cos\theta(t) \\ Z(t+\Delta t) &= Z(t) + (N_d r_d + N_e r_e)\frac{\pi}{N_{res}} Sen\theta(t) \\ \theta(t+\Delta t) &= \theta(t) + (N_d r_d - N_e r_e)\frac{2\pi}{bN_{res}} \end{aligned} \tag{2.9}$$

A expressão anterior leva ao modelo de deslocamento do robô que pode ser, finalmente, escrito como:

$$\begin{bmatrix} X_k \\ Z_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} X_{k-1} \\ Z_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{\pi}{N_{res}} Cos\theta_{k-1} & \frac{\pi}{N_{res}} Cos\theta_{k-1} \\ \frac{\pi}{N_{res}} Sen\theta_{k-1} & \frac{\pi}{N_{res}} Sen\theta_{k-1} \\ \frac{2\pi}{bN_{res}} & -\frac{2\pi}{bN_{res}} \end{bmatrix} \begin{bmatrix} r_d & 0 \\ 0 & r_e \end{bmatrix} \begin{bmatrix} N_d \\ N_e \end{bmatrix} \quad (2.10)$$

ou seja, o deslocamento do robô é dado por:

$$\mathbf{X}_k = \mathbf{X}_{k-1} + \mathbf{G}_{k-1}\mathbf{u}_k \quad (2.11)$$

onde:

$$\mathbf{X}_k = \begin{bmatrix} X_k \\ Z_k \\ \theta_k \end{bmatrix} \quad (2.12)$$

$$\mathbf{X_{k-1}} = \begin{bmatrix} X_{k-1} \\ Z_{k-1} \\ \theta_{k-1} \end{bmatrix} \quad (2.13)$$

$$\mathbf{G}_k = \begin{bmatrix} \frac{\pi}{N_{res}} Cos\theta_{k-1} & \frac{\pi}{N_{res}} Cos\theta_{k-1} \\ \frac{\pi}{N_{res}} Sen\theta_{k-1} & \frac{\pi}{N_{res}} Sen\theta_{k-1} \\ \frac{2\pi}{bN_{res}} & -\frac{2\pi}{bN_{res}} \end{bmatrix} \begin{bmatrix} r_d & 0 \\ 0 & r_e \end{bmatrix} \quad (2.14)$$

$$\mathbf{u}_k = \begin{bmatrix} N_d \\ N_e \end{bmatrix} \quad (2.15)$$

As medidas de odometria, na prática, estão sujeitas a erros do tipo sistemático e erro não sistemático (Valgas, 2002; Borenstein, et al., 1996).

Os erros sistemáticos são aqueles causados por imperfeições no modelo cinemático do robô, tais como uma medida incorreta dos raios das rodas ou do comprimento do seu eixo. Esse tipo de erro ocorre durante toda a navegação e acumula-se constantemente, gerando grandes distorções na determinação da sua localização. Já os erros não sistemáticos são imprevisíveis, causados por situações que surgem inesperadamente, como por exemplo, terrenos irregulares, a presença de objetos inesperados no chão, escorregamento das rodas, etc.

A equação (2.11) é reescrita considerando-se os erros de odometria, como mostrada na equação (2.16):

$$\mathbf{X}_k = \mathbf{X}_{k-1} + \mathbf{G}_{k-1}\mathbf{u}_k + \mathbf{G}_{k-1}\beta_k \tag{2.16}$$

onde β_k é o vetor de ruído do processo, cujos elementos correspondem ao erro do número de passos das rodas da direita $\overline{N}_{dk}$ e da esquerda $\overline{N}_{ek}$, sendo que cada erro possui distribuição de probabilidade normal e desvio padrão do erro de odometria σ:

$$p(\beta_k(i)) \sim N(0,\sigma) \tag{2.17}$$

onde i=1 ou 2, e os elementos de β_k são dados por:

$$\beta_k = \begin{bmatrix} \overline{N}_{dk} \\ \overline{N}_{ek} \end{bmatrix} \tag{2.18}$$

A matriz de covariância do produto $\mathbf{G}_{k-1}\beta_k$, por sua vez, é dada pelo valor esperado conforme mostrado em (2.19):

$$\mathrm{cov}(\mathbf{G}_{k-1}\beta_k) = E\left[(\mathbf{G}_{k-1}\beta_k)(\mathbf{G}_{k-1}\beta_k)'\right] = \mathbf{Q}_k \tag{2.19}$$

Usando a equação (2.14) com $r_d = r_e$ =r, e sendo definido a, b e c como:

$$a = \frac{\pi r}{N_{res}} Cos\theta_{k-1}$$

$$b = \frac{\pi r}{N_{res}} Sin\theta_{k-1} \tag{2.20}$$

$$c = \frac{2\pi r}{bN_{res}}$$

então, a cov($\mathbf{G}_{k-1}\beta_k$) pode ser reescrita como:

$$\mathbf{Q}_k = E\left[\left(\begin{bmatrix} a & a \\ b & b \\ c & -c \end{bmatrix}\begin{bmatrix} \overline{N}_{dk} \\ \overline{N}_{ek} \end{bmatrix}\right)\left(\begin{bmatrix} a & a \\ b & b \\ c & -c \end{bmatrix}\begin{bmatrix} \overline{N}_{dk} \\ \overline{N}_{ek} \end{bmatrix}\right)'\right] \tag{2.21}$$

Expandindo-se a equação (2.21):

$$\mathbf{Q}_k = E\left[\begin{bmatrix} a\overline{N}_{dk} + a\overline{N}_{ek} \\ b\overline{N}_{dk} + b\overline{N}_{ek} \\ c\overline{N}_{dk} - c\overline{N}_{ek} \end{bmatrix} * \begin{bmatrix} a\overline{N}_{dk} + a\overline{N}_{ek} & b\overline{N}_{dk} + b\overline{N}_{ek} & c\overline{N}_{dk} - c\overline{N}_{ek} \end{bmatrix}\right] \Rightarrow$$

$$\mathbf{Q}_k = E\begin{bmatrix} \left(a\overline{N}_{dk} + a\overline{N}_{ek}\right)^2 & \left(a\overline{N}_{dk} + a\overline{N}_{ek}\right)\left(b\overline{N}_{dk} + b\overline{N}_{ek}\right) & \left(a\overline{N}_{dk} + a\overline{N}_{ek}\right)\left(c\overline{N}_{dk} - c\overline{N}_{ek}\right) \\ \left(b\overline{N}_{dk} + b\overline{N}_{ek}\right)\left(a\overline{N}_{dk} + a\overline{N}_{ek}\right) & \left(b\overline{N}_{dk} + b\overline{N}_{ek}\right)^2 & \left(b\overline{N}_{dk} + b\overline{N}_{ek}\right)\left(c\overline{N}_{dk} - c\overline{N}_{ek}\right) \\ \left(c\overline{N}_{dk} - c\overline{N}_{ek}\right)\left(a\overline{N}_{dk} + a\overline{N}_{ek}\right) & \left(c\overline{N}_{dk} - c\overline{N}_{ek}\right)\left(b\overline{N}_{dk} + b\overline{N}_{ek}\right) & \left(c\overline{N}_{dk} - c\overline{N}_{ek}\right)^2 \end{bmatrix} \Rightarrow$$

$$\mathbf{Q}_k = E\begin{bmatrix} a^2\overline{N}_{dk}^2 + 2a^2\overline{N}_{dk}\overline{N}_{ek} + a^2\overline{N}_{ek}^2 & ab\overline{N}_{dk}^2 + 2ab\overline{N}_{dk}\overline{N}_{ek} + ab\overline{N}_{ek}^2 & ac\left(\overline{N}_{dk}^2 - \overline{N}_{ek}^2\right) \\ ab\overline{N}_{dk}^2 + 2ab\overline{N}_{dk}\overline{N}_{ek} + ab\overline{N}_{ek}^2 & b^2\overline{N}_{dk}^2 + 2b^2\overline{N}_{dk}\overline{N}_{ek} + b^2\overline{N}_{ek}^2 & bc\left(\overline{N}_{dk}^2 - \overline{N}_{ek}^2\right) \\ ac\left(\overline{N}_{dk}^2 - \overline{N}_{ek}^2\right) & bc\left(\overline{N}_{dk}^2 - \overline{N}_{ek}^2\right) & c^2\overline{N}_{dk}^2 - 2c^2\overline{N}_{dk}\overline{N}_{ek} + c^2\overline{N}_{ek}^2 \end{bmatrix} \Rightarrow$$

$$\mathbf{Q}_k = \begin{bmatrix} a^2E\left[\overline{N}_{dk}^2\right] + 2a^2E\left[\overline{N}_{dk}\overline{N}_{ek}\right] + a^2E\left[\overline{N}_{ek}^2\right] & abE\left[\overline{N}_{dk}^2\right] + 2abE\left[\overline{N}_{dk}\overline{N}_{ek}\right] + abE\left[\overline{N}_{ek}^2\right] & acE\left[\overline{N}_{dk}^2\right] - E\left[\overline{N}_{ek}^2\right]\right) \\ abE\left[\overline{N}_{dk}^2\right] + 2abE\left[\overline{N}_{dk}\overline{N}_{ek}\right] + abE\left[\overline{N}_{ek}^2\right] & b^2E\left[\overline{N}_{dk}^2\right] + 2b^2E\left[\overline{N}_{dk}\overline{N}_{ek}\right] + b^2E\left[\overline{N}_{ek}^2\right] & bcE\left[\overline{N}_{dk}^2\right] - E\left[\overline{N}_{ek}^2\right]\right) \\ acE\left[\overline{N}_{dk}^2\right] - E\left[\overline{N}_{ek}^2\right]\right) & bcE\left[\overline{N}_{dk}^2\right] - E\left[\overline{N}_{ek}^2\right]\right) & c^2E\left[\overline{N}_{dk}^2\right] - 2c^2E\left[\overline{N}_{dk}\overline{N}_{ek}\right] + c^2E\left[\overline{N}_{ek}^2\right] \end{bmatrix}. \tag{2.22}$$

Assumindo-se a hipótese de distribuição normal das variáveis $\overline{N}_{dk}$ e $\overline{N}_{ek}$, tem-se que $E\left[\overline{N}_{dk}^2\right] = E\left[\overline{N}_{dk}\overline{N}_{dk}^{'}\right] = \sigma^2$ e $E\left[\overline{N}_{ek}^2\right] = E\left[\overline{N}_{ek}\overline{N}_{ek}^{'}\right] = \sigma^2$.

$$\mathbf{Q}_k = \begin{bmatrix} a^2\sigma^2 + a^2\sigma^2 & ab\sigma^2 + ab\sigma^2 & 0 \\ ab\sigma^2 + ab\sigma^2 & b^2\sigma^2 + b^2\sigma^2 & 0 \\ 0 & 0 & c^2\sigma^2 + c^2\sigma^2 \end{bmatrix}$$

$$\mathbf{Q}_k = \begin{bmatrix} 2a^2\sigma^2 & 2ab\sigma^2 & 0 \\ 2ab\sigma^2 & 2b^2\sigma^2 & 0 \\ 0 & 0 & 2c^2\sigma^2 \end{bmatrix} = 2\sigma^2 \begin{bmatrix} a^2 & ab & 0 \\ ab & b^2 & 0 \\ 0 & 0 & c^2 \end{bmatrix} \tag{2.23}$$

Dessa forma, a covariância do produto $\mathbf{G}_k\beta_k$ é dada pela matriz $\mathbf{Q}_k$, definida como:

$$\mathbf{Q}_k = 2\left(\frac{\pi r\sigma}{N_{res}}\right)^2 \begin{bmatrix} (Cos\theta_{k-1})^2 & (Cos\theta_{k-1})(Sin\theta_{k-1}) & 0 \\ (Cos\theta_{k-1})(Sin\theta_{k-1}) & (Sin\theta_{k-1})^2 & 0 \\ 0 & 0 & \left(\frac{2}{b}\right)^2 \end{bmatrix} \tag{2.24}$$

O modelo cinemático descrito neste capítulo determina a posição do robô obtido por odometria. A matriz covariância de erro do processo dada pela equação (2.24) é calculada baseada no modelo real, envolvendo erros As equações que envolvem a posição $\mathbf{X_k}$ e $\mathbf{Q_k}$ serão usadas no capítulo 5, no que se refere ao filtro de Kalman para estimar a posição ótima do robô. Para o uso do filtro, basta agora descrever o processo de como se pode obter a posição do robô via um par de câmeras e a matriz de covariância proveniente do sistema de visão. Os capítulos 3 e 4 descrevem toda a parte de visão necessária para localizar o robô no ambiente.

Capítulo 3

Reconstrução 3-D a Partir de Imagens

Nesse capítulo é desenvolvido o conceito de visão estéreo em um ambiente tridimensional, que é um caso mais realístico, embora a simulação apresentada nesse trabalho considere um ambiente bidimensional, onde a coordenada y do sistema de coordenadas é eliminada. A estratégia de usar duas câmeras para visão também pode, por exemplo, ser usada para reconstruir objetos no espaço, em forma de paralelepípedos baseados na técnica de "bouding boxes" (Kurka and Rudek, 2006; Rudek, 2006). O princípio da geometria epipolar (Forster, 2004) é usado para obter a posição do ponto no espaço que é visto simultaneamente por duas câmeras. O "bounding box" engloba por inteiro o objeto cujo contorno deve ser evitado pelo robô durante a sua navegação.

3.1 - Um Modelo Simples de Imagem

O termo imagem refere-se a uma função intensidade luminosa bidimensional, chamada de $I(x, y)$, isso é, a amplitude I, associada às coordenadas da imagem (x, y). A função intensidade é sempre positiva e finita, ou seja:

$$0 < I(x, y) < \infty \tag{3.1}$$

As imagens observadas pelo olho humano consistem de luz visível refletida pelos objetos. A função $I(x, y)$ pode ser calculada por duas diferentes componentes: pela quantidade de luz incidindo em objeto sendo observado e pela quantidade de luz refletida pelo objeto. As

componentes recebem o nome de iluminação e reflectância, e respectivamente são representadas por $i(x, y)$ e $r(x, y)$. O produto dessas duas funções dá a função intensidade:

$$I(x, y) = i(x, y)r(x, y) \tag{3.2}$$

onde $0 < i(x, y) < \infty$ e $0 < r(x, y) < 1$.

Quando se admite que as imagens tenham suas intensidades monocromáticas, ou em nível de cinza, pode-se dizer que esta intensidade I varia entre um nível $I_{mínimo}$ e $I_{máximo}$, onde:

$$I_{mínimo} \leq l \leq I_{máximo} \tag{3.3}$$

$$I_{mínimo} = i_{mínimo} r_{mínimo} \tag{3.4}$$

$$I_{máximo} = i_{máximo} r_{máximo} \tag{3.5}$$

Para se adequar ao uso do processamento computacional, a função $I(x, y)$ necessita ser digitalizada tanto espacialmente quanto em amplitude. Considera-se que uma imagem contínua $I(x, y)$ possa ser representada por amostras igualmente espaçadas, arranjadas na forma de uma matriz N × M, do tipo:

$$I(x, y) = \begin{bmatrix} I(0,0) & I(0,1) & \cdots & I(0, M-1) \\ I(1,0) & I(1,1) & \cdots & I(1, M-1) \\ \vdots & \vdots & \cdots & \vdots \\ I(N-1,0) & I(N-1,1) & \cdots & I(N-1, M-1) \end{bmatrix} \tag{3.6}$$

onde $I(x, y)$ possui uma quantização em L níveis de cinza.

A imagem digital é caracterizada pelo lado direito da equação (3.6). Cada elemento dessa matriz representa um seção da imagem sendo denominado por *pixel,* uma abreviação de *picture element*, ou do português, elemento de figura.

3.2 - Modelos de Projeção

Câmeras são dispositivos de mapeamento de objetos de um espaço tridimensional em mapas planos das intensidades luminosas de tais objetos. Modelos definem a transformação de um dado ponto no espaço, que estão em coordenadas tridimensionais P(X,Y,Z) para uma representação bidimensional p(x,y) de sua imagem projetada na câmera. Segundo (Banerjee, 2001) e (Forsyth, 2003) as câmeras podem ser modeladas através do modelo em perspectiva (*perspective projection*), modelo afim (*affine projection)*, perspectiva fraca (*weak perspective)* e modelo ortográfico (*orthographic projection*). No presente trabalho considera-se a projeção em perspectiva.

3.2.1 - Projeção em Perspectiva

Uma transformação em perspectiva projeta pontos tridimensionais sobre um plano (Gonzalez e Woods, 1992, Shak, 1997). Ela é baseada numa representação denominada *pinhole*, onde todos os raios de projeção do objeto convergem para um centro ótico, apresentado na figuras (3.1) e (3.2), como em (Forsyth e Ponce, 2003) e Jain (1995).

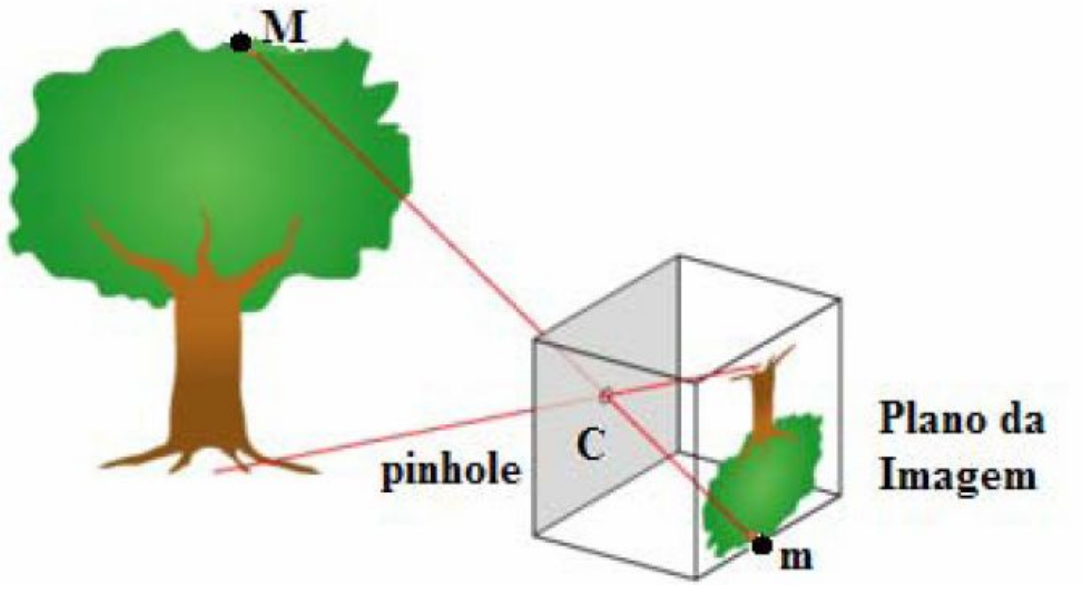


Figura 3.1: Representação do modelo pinhole.

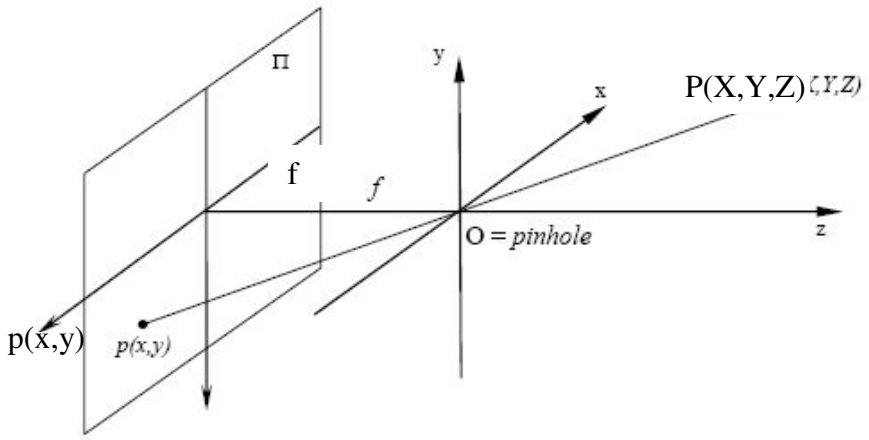


Figura 3.2: Modelo da projeção perspectiva adaptado de Forsyth (2003).

Na figura 3.2 um ponto P no espaço projeta-se no ponto p do plano de imagem π, onde f é a distância focal.

A projeção perspectiva considera a representação virtual do objeto posicionada atrás do pinhole, sobre um plano chamado de "plano de imagem". Por conveniência, o plano de imagem pode ser representado à frente do centro ótico da câmera, como ilustra a figura 3.3. Neste caso não é necessário operar com a imagem invertida, o que facilita a interpretação da cena. Qualquer objeto tem sua perspectiva formada sobre o plano de imagem Π' da câmera.

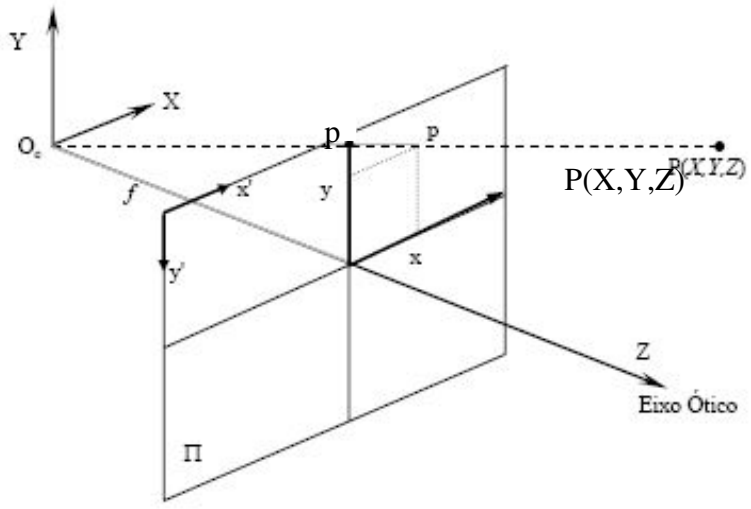


Figura 3.3: Imagem formada a frente do eixo ótico.

Representam-se as coordenadas do mundo de qualquer ponto em uma cena 3-D por (X,Y,Z) e assume-se que Z>f para garantir que todos os pontos estejam na frente da câmera. Formula-se, em seguida, a relação que fornece as coordenadas do ponto p (x,y) da projeção do ponto (X,Y,Z) sobre o plano de imagem. Jain (1995) descreve que a posição da projeção do ponto **P** no plano de imagem pode ser obtida por semelhança de triângulos, como apresentado na figura (3.4).

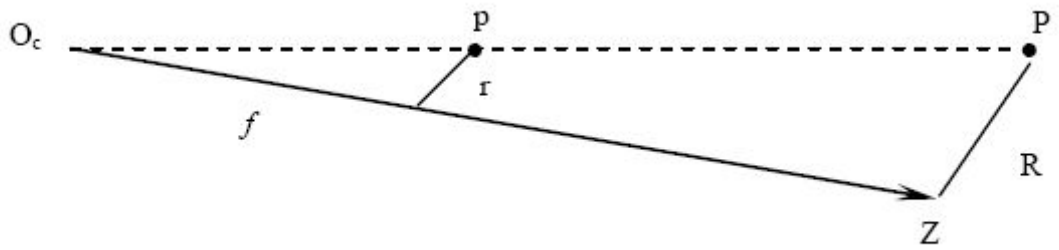


Figura 3.4: Triangulação projetiva da imagem.

Fazendo-se a semelhança de triângulos baseado na figura (3.4), tem-se:

$$\frac{f}{Z} = \frac{r}{R} \tag{3.7}$$

$$\frac{x}{X} = \frac{y}{Y} = \frac{r}{R} \tag{3.8}$$

Das equações (3.7) e (3.8) tem-se a seguinte relação:

$$\frac{x}{X} = \frac{f}{Z} \tag{3.9}$$

$$\frac{y}{Y} = \frac{f}{Z} \tag{3.10}$$

Sendo assim, as coordenadas de p em função das de P, são representadas como:

$$x = \frac{f}{Z} X \tag{3.11}$$

$$y = \frac{f}{Z} Y \tag{3.12}$$

As equações (3.11) e (3.12) representam a transformação de um ponto no espaço, em coordenadas 3D, para um ponto no plano, em coordenadas 2D. Nesta representação, as posições de x e y são relativas ao referencial da câmera, em coordenadas métricas. Entretanto, qualquer análise sobre a imagem depende da posição do ponto (em *pixels*) dado por p' (x',y').

3.3 - Reconstrução Tridimensional

Nessa secção é apresentada a geometria de reconstrução de um ponto P no espaço, baseado em suas projeções em um par de câmeras.

Duas imagens de uma mesma cena estacionária com centros de projeção não coincidentes podem ser obtidas por duas câmeras ou mesmo por uma translação de uma única câmera. Essas duas perspectivas diferentes de uma mesma imagem estática são chamadas de visão estéreo. As relações geométricas entre ambas as imagens descrevendo os pontos do espaço são estabelecidas pela geometria epipolar.

Sejam $\mathbf{p_1}$ e $\mathbf{p_2}$ os vetores das coordenadas de projeção do ponto P no referencial da câmera 1 e 2, respectivamente. A determinação dessa correspondência é o primeiro passo para se derivar informações do espaço tridimensional a partir de duas imagens. A relação das coordenadas de projeção na câmera 2 em função das coordenadas na câmera 1 é dada por:

$$\mathbf{p}_2 = \mathbf{R}\,\mathbf{p}_1 + \mathbf{t} \tag{3.13}$$

onde os vetores $\mathbf{p_1}=[x_1,\ y_1,\ z_1]^T$ e $\mathbf{p_2}=[x_2,\ y_2,\ z_2]^T$ são conhecidos e $z_1 = f_1$ e $z_2 = f_2$. Neste estudo, utiliza-se a mesma distância focal para as duas câmeras. Assim, $f_1=f_2=f$. **R** é a matriz de rotação e **t** é o vetor de translação entre os referenciais das duas câmeras.

A relação entre o ponto da imagem no referencial da imagem e o ponto P, visto do referencial de uma câmera, é dado por:

$$\mathbf{p}_i^{'} = \mathbf{K}_i \, \mathbf{P}_i \tag{3.14}$$

onde $\mathbf{p_i}'$ são as coordenadas em *pixels* na imagem, $\mathbf{K}_i$ é a matriz de parâmetros intrínsecos, $\mathbf{P}_i$ são os vetores que representam as coordenadas espaciais do ponto P visto da câmera, isto é, $\mathbf{P}_i=[X_i, Y_i, Z_i]^T$.

A matriz de parâmetros intrínsecos é dada por:

$$\mathbf{K_i} = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.15}$$

onde o_x e o_y são as coordenadas em *pixel* do centro do plano da imagem, e s_x e s_y representam o tamanho do *pixel* em milímetros.

Se a câmera é dita calibrada, temos que a equação (3.14) é proporcional a uma matriz identidade:

$$\lambda \, \mathbf{p}_i = \Pi_0 \, \mathbf{P}_i \tag{3.16}$$

onde λ é escala de profundidade desconhecida que diferencia $\mathbf{p_i}$ de $\mathbf{P_i}$, já que os dois pontos estão sobre uma mesma linha. Π_0 é a matriz identidade de parâmetros intrínsecos $[I,0]$.

Considere a figura (3.5). O plano definido pelo ponto P e pelos centros óticos $\mathbf{O_1}$ e $\mathbf{O_2}$ é designado plano epipolar. Esse plano intercepta cada imagem formando um linha chamada linha epipolar. O ponto projetado $\mathbf{p}_1$ está sobre a linha l_1 e o ponto projetado $\mathbf{p}_2$ está sobre a linha l_2. A linha que une $\mathbf{O_1}$ e $\mathbf{O_2}$ também intercepta a imagem, formando os pontos $\mathbf{e}_1$ e $\mathbf{e}_2$, chamados de epipolos. O epipolo $\mathbf{e}_2$ é a projeção do centro ótico $\mathbf{O_1}$ da primeira câmera, na imagem observada pela segunda câmera. O epipolo $\mathbf{e}_1$ é a projeção do centro ótico $\mathbf{O_2}$ da segunda câmera, na imagem observada pela primeira câmera. Se $\overline{\mathbf{O_1p}}_1, \overline{\mathbf{O_2p}}_2$ e $\overline{\mathbf{O_1O_2}}$, temos:

$$\overline{\mathbf{O_1p}}_1 \bullet \left[\overline{\mathbf{O_1O}}_2 \times \overline{\mathbf{O_2p}}_2\right] = 0 \tag{3.17}$$

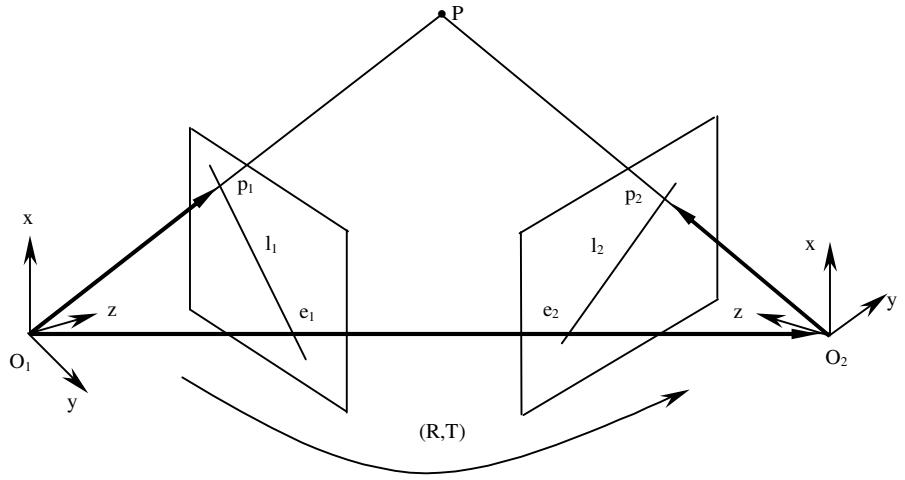


Figura 3.5: Representação da geometria epipolar.

A coplanaridade também pode ser expressa como:

$$\mathbf{p}_1 \cdot [\mathbf{t} \times (\mathbf{R}\,\mathbf{p}_2)] = \mathbf{0} \tag{3.18}$$

Da álgebra, sabemos que o produto vetorial entre dois vetores quaisquer **u** e **v** é igual ao produto da matriz anti-simétrica do primeiro pelo segundo:

$$\mathbf{v} \wedge \mathbf{u} = \tilde{\mathbf{v}}\mathbf{u} \tag{3.19}$$

A matriz anti-simétrica de **v** é da forma:

$$\tilde{\mathbf{v}} = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix} \tag{3.20}$$

A matriz anti-simétrica do vetor de translação **t** é dado por:

$$\hat{\mathbf{T}} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \tag{3.21}$$

onde, t_x, t_y e t_z são as coordenadas do vetor de translação *t*. A matriz $\hat{\mathbf{T}}$ possui as seguintes propriedades:

$$\hat{\mathbf{T}}\mathbf{t} = 0 \qquad \text{e} \qquad \hat{\mathbf{T}}\mathbf{x} = \mathbf{T} \times \mathbf{x} \tag{3.22}$$

Aplicando-se (3.22) em (3.18), temos:

$$\mathbf{p}_1 \cdot [\hat{\mathbf{T}}\mathbf{R}\mathbf{p}_2] = \mathbf{0} \tag{3.23}$$

A matriz essencial é definida como:

$$\mathbf{E} = \hat{\mathbf{T}}\mathbf{R}\,, \tag{3.24}$$

contém os parâmetros de rotação e translação no sistema de referência da câmera, também conhecidos como parâmetros de *pose*. Assim, a equação (3.23) pode ser reescrita como:

$$\mathbf{p}^T{}_1\mathbf{E}\,\mathbf{p}_2 = \mathbf{0} \tag{3.25}$$

Para se achar os parâmetros de *pose* **R** e $\hat{\mathbf{T}}$ usa-se o algoritmo de 8 pontos. Pode-se assumir que a matriz essencial tem a seguinte forma:

$$\mathbf{E} = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \tag{3.26}$$

ou, na forma de coluna:

$$(\mathbf{E}^s)^T = \begin{bmatrix} e_{11}\, e_{21}\, e_{31}\, e_{12}\, e_{22}\, e_{32}\, e_{13}\, e_{23}\, e_{33} \end{bmatrix} \tag{3.27}$$

A idéia básica do algoritmo é estabelecer inicialmente *k* pontos correspondentes entre duas imagens. Cada correspondência de pontos forma uma equação linear homogênea. Assim, o seguinte sistema homogêneo de equações lineares é formado:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{19} \\ a_{21} & a_{22} & \cdots & a_{29} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{k9} \end{bmatrix} \mathbf{E}^s = 0 \tag{3.28}$$

onde, a_{kj} é o *j*-ésimo elemento do produto de Kronecker, do par "k" de dois vetores de projeção $\mathbf{p}_1$ e $\mathbf{p}_2$:

$$\mathbf{a}_k = \lfloor a_{k1}\ a_{k2} \cdots a_{kj} \cdots a_{k9} \rfloor = \mathbf{p}_1 \otimes \mathbf{p}_2 \tag{3.29}$$

Os elementos $\mathbf{a}_k$ estão associados às coordenadas de cada par correspondente do total de nove pares. Por isso, os valores de $\mathbf{a}_k$ são conhecidos e os elementos e_{ij} da matriz essencial são obtidos pela resolução do sistema linear, contendo nove equações e nove incógnitas. Sendo o sistema singular, este pode ser resolvido no mínimo com oito equações, com solução conhecida. A matriz **a** é descrita como:

$$\mathbf{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_8 \end{bmatrix} \tag{3.30}$$

e pode ser decomposta em valores singulares da forma:

$$\mathbf{a}^T = \mathbf{U}_{a^T} \exists_{a^T} \mathbf{V}_{a^T}^T \tag{3.31}$$

onde $\mathbf{U}_{a^T}$ é uma matriz ortogonal cujas colunas são os autovetores de $\mathbf{a^T}.\mathbf{a}$, $\exists_{a^T}$ é uma matriz ortogonal cujas colunas são os autovetores de $\mathbf{a}.\mathbf{a^T}$, $\mathbf{V}_{a^T}$ é a matriz dos autovalores não nulos de $\mathbf{a^T}.\mathbf{a}^T$ e $\mathbf{a}.\mathbf{a^T}$.

E finalmente, com **U** e ***V*** encontrados pode-se extrair ***R*** e ***T*** da matriz essencial, conforme mostrado nas equações seguintes:

$$\mathbf{R} = \mathbf{U}\mathbf{R}_Z^T\left(\pm\frac{\pi}{2}\right)\mathbf{V}^T \qquad \text{e} \qquad \hat{\mathbf{T}} = \mathbf{U}\mathbf{R}_z\left(\pm\frac{\pi}{2}\right)\Sigma\mathbf{U}^T \tag{3.32}$$

$$\mathbf{R}_Z^T\left(\pm\frac{\pi}{2}\right) = \begin{bmatrix} 0 & \pm 1 & 0 \\ \mp 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.33}$$

Com os parâmetros intrínsecos $\mathbf{K}_i$ de cada câmera, podem ser relacionados os pontos de projeção **p** e o ponto **P**, ambos em coordenadas de câmera.

Considere a situação não calibrada em que as coordenadas em *pixel* na imagem $\mathbf{p}'_i$ estão relacionadas com as coordenadas $\mathbf{p_i}$ de acordo com a equação (3.34):

$$\mathbf{p}_\mathrm{i} = \mathbf{K}_i^{-1}\mathbf{p}'_i \tag{3.34}$$

Substituindo-se a equação (3.34) em (3.25) temos:

$$\mathbf{p}_2^{'T}\left(\mathbf{K}_2^{-1}\right)\mathbf{E}\mathbf{K}_1^{-1}\mathbf{p}_1^{'} = \mathbf{0} \tag{3.35}$$

onde

$$\mathbf{F} = \left(\mathbf{K}_2^{-1}\right)\mathbf{E}\mathbf{K}_1^{-1} \tag{3.36}$$

Então, podemos escrever:

$$\mathbf{p}_2^{'T}\mathbf{F}\mathbf{p}_1^{'} = \mathbf{0} \tag{3.37}$$

A matriz **F** é designada por matriz fundamental.

Depois dos pontos correspondentes terem sido identificados, resta finalmente recuperar a informação tridimensional, o que pode ser feito por simples triangulação. Entretanto, mesmo esta

fase não é tão trivial, pois este cálculo possui certo grau de incerteza devido a erros de estabelecimento da geometria, orientação do plano de imagem e distância focal.

3.4 - Reconstrução do Mapa de Profundidade a Partir das Intensidades

No presente trabalho considera-se apenas os casos de projeção de objetos em um plano bidimensional, em um par de câmeras unidimensionais. Dessa forma, a imagem de cada câmera consiste dos mapas de intensidade luminosa de projeção de objetos bidimensionais nas telas de cada câmera, que são segmentos lineares. Na secção (3.1) mostra-se a imagem como uma matriz, e neste trabalho, ela é representada por um vetor. Cada elemento desse vetor engloba o valor da intensidade em níveis de cinza da imagem. Para efeito de simulação e validação dos métodos de navegação por câmeras introduzidos neste trabalho, outra simplificação é introduzida, no que diz respeito às projeções dos objetos do plano bidimensional. Considera-se que os objetos presentes no plano possuem intensidades constantes. Também, as intensidades de objetos distintos possuem valores diferentes entre si. Assim, os mapas de intensidade unidimensionais possuem a representação na forma de degraus de intensidade. Tais considerações são utilizadas nas simulações descritas no capítulo 7.

De uma maneira mais simples, para se encontrar os degraus, que representam diferentes objetos em uma cena, trabalha-se com as diferenças de intensidade de objetos adjacentes. Assim, percorre-se o vetor de intensidades à procura de variações de intensidade para localizar a mudança de objetos e existência de um possível portal de passagem. Encontram-se, para as imagens de uma câmera, os locais de mudança de intensidade, guardando-se os índices correspondentes a essas variações. Procuram-se os locais dessas mesmas variações de intensidade na imagem da segunda câmera. Pares de pontos com a mesma mudança de intensidades nas duas câmeras representam um mesmo objeto. A figura 3.6 mostra os mapas de intensidade das câmeras a e b, indicando os pares com as correspondentes mudanças de intensidade.

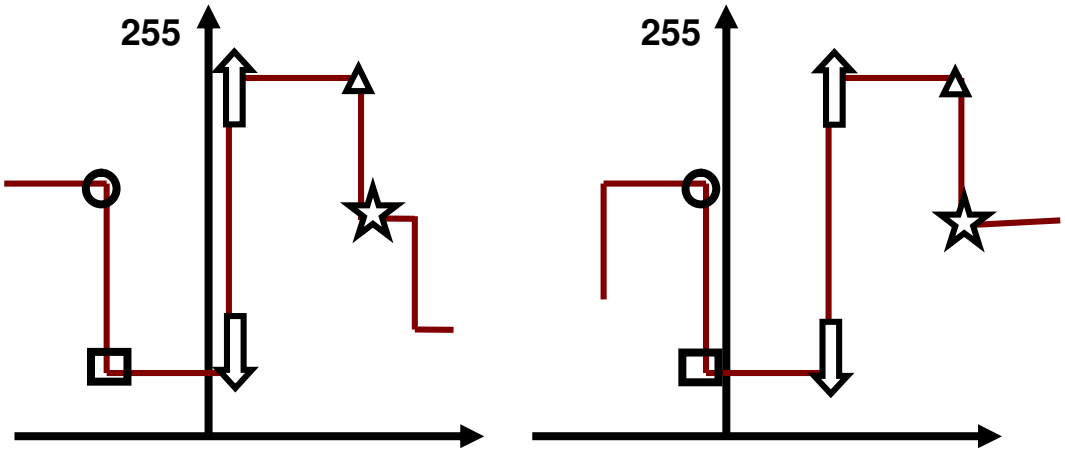


Figura 3.6: Mapa de intensidade das câmeras *a* e *b*.

Os índices que caracterizam os pares de intensidades diferentes representam pontos de objetos que são projetados no referencial da câmera *a*. Para encontrar os pontos correspondentes no referencial câmera *a*, considere um caso particular que mostra o esquema de coordenadas das câmeras *a* e *b* (figura (3.7)). Nessa figura, é mostrado o esquema do sistema de coordenadas de duas câmeras *a* e *b*, respectivamente (x_{ca},z_{ca}) e (x_{cb},z_{cb}), que estão perfeitamente alinhados, diferenciando-se apenas na posição de suas origens. Considere um caso bidimensional da secção (3.2), onde se exclui o eixo y. Uma vez que as câmeras estejam alinhadas, os valores da variável z do ponto P para cada câmera, z_{ca} e z_{cb}, são iguais.

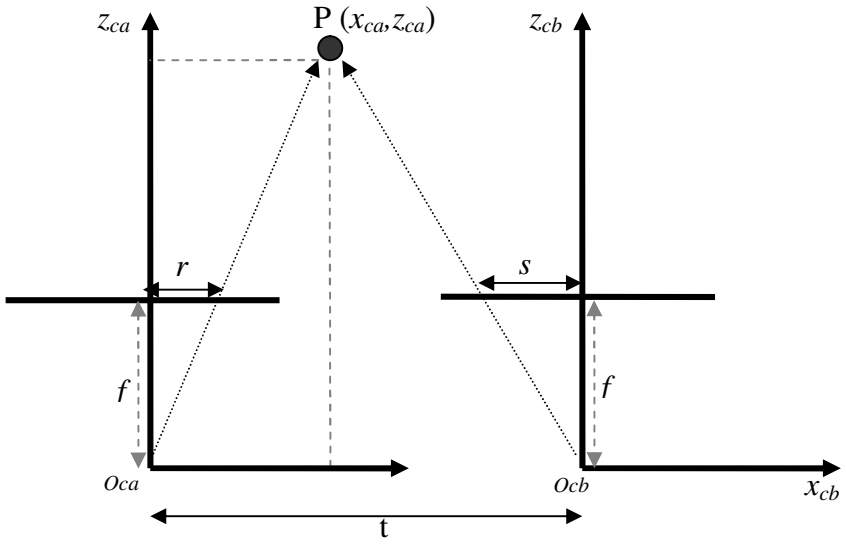


Figura 3.7: Visão estéreo com deslocamento paralelo ao eixo óptico.

Considere *r* e *s* como sendo as coordenadas da imagem, respectivamente, das câmeras a e b.

Por semelhança de triângulos tem-se:

$$\frac{x_{ca}}{z_{ca}} = \frac{r}{f} \tag{3.38}$$

$$\frac{x_{cb}}{z_{ca}} = \frac{s}{f} \tag{3.39}$$

Sabendo que *t* é a distância entre as origens das câmeras tem-se:

$$\mathbf{t} = \mathbf{O}_{ca} - \mathbf{O}_{cb} \tag{3.40}$$

sendo $\mathbf{O}_{ca}$ o vetor que indica o centro da câmera *a* no referencial de tal câmera e $\mathbf{O}_{cb}$ indicando o centro da câmera *b* em coordenadas da câmera *a*.

Subtraindo-se as equações (3.38) e (3.39) com o uso da equação (3.40) chega-se a:

$$z_{ca} = \frac{tf}{r - s} \tag{3.41}$$

Substituindo a equação (3.41) em (3.38) têm-se:

$$x_{ca} = \frac{tr}{r - s} \tag{3.42}$$

Sendo assim, a posição do ponto P_i visto da câmera *a* é:

$$\mathbf{x}_{cai} = \begin{bmatrix} x_{cai} \\ z_{cai} \end{bmatrix} = \frac{t}{(r_i - s_i)} \begin{bmatrix} r_i \\ f \end{bmatrix} \tag{3.43}$$

O mapa de profundidade, que corresponde aos pontos P_r visualizados pelo robô em seu referencial, é construído levando-se em consideração o vetor de separação entre o robô e a origem do referencial da câmera *a,* $\mathbf{x}_a$, isto é,

$$\mathbf{P}_r = \mathbf{x}_{cai} + \mathbf{x}_a \tag{3.44}$$

Neste capítulo foi descrito uma teoria sobre como reconstruir objetos no espaço ou num ambiente 2D usando um par de câmeras. Portanto, precisa ser explicado posteriormente como esses objetos reconstruídos são usados para cálculo de posição do robô. O próximo capítulo detalha a influência dos pontos reconstruídos sobre a localização do robô em um ambiente bidimensional.

Capítulo 4

Navegação de Robôs Através de Imagens Reconstruídas

A localização do robô pode ser encontrada através de um par de câmeras, usando as informações das imagens de uma cena. Se os objetos possuem suas posições conhecidas no mundo, é necessário saber como as posições desses mesmos objetos são vistos pelo robô, sendo agora observados pelas câmeras, através da reconstrução tridimensional descrita no capítulo anterior. Considerando-se um caso mais realístico, são introduzidos erros de projeção nas telas das câmeras, fazendo com que a posição dos objetos no mundo sejam diferentes do que o encontrado pelas câmeras. A informação da localização dos objetos no referencial do robô, juntamente com suas posições absolutas, resulta na informação de uma possível localização do robô.

4.1 - Posicionamento do Robô Baseado em Dois Pontos da Imagem

A determinação da posição espacial de um robô pode ser feita a partir da projeção conhecida de pelo menos dois pontos em duas câmeras, presentes no referencial próprio do robô.

Consideram-se os referenciais usados no modelo de um robô navegando num ambiente bidimensional, como ilustrado na figura (4.1). Na figura (4.1 é mostrada as referências do mundo (X,Z), do robô (x,z), de duas câmeras, *a* (x_{ca},z_{ca}) e *b* (x_{cb},z_{cb}) bem como os referenciais *r* e *s* usados para obtenção de imagens. As câmeras estão afastadas de uma distância *t* e seus referenciais encontram-se rotacionados de um ângulo θ em relação ao mundo, assim como o referencial do robô. A rotação do robô θ (valor do ângulo da posição real) é definida como o ângulo entre o eixo *z* do robô e o eixo *X* do mundo.

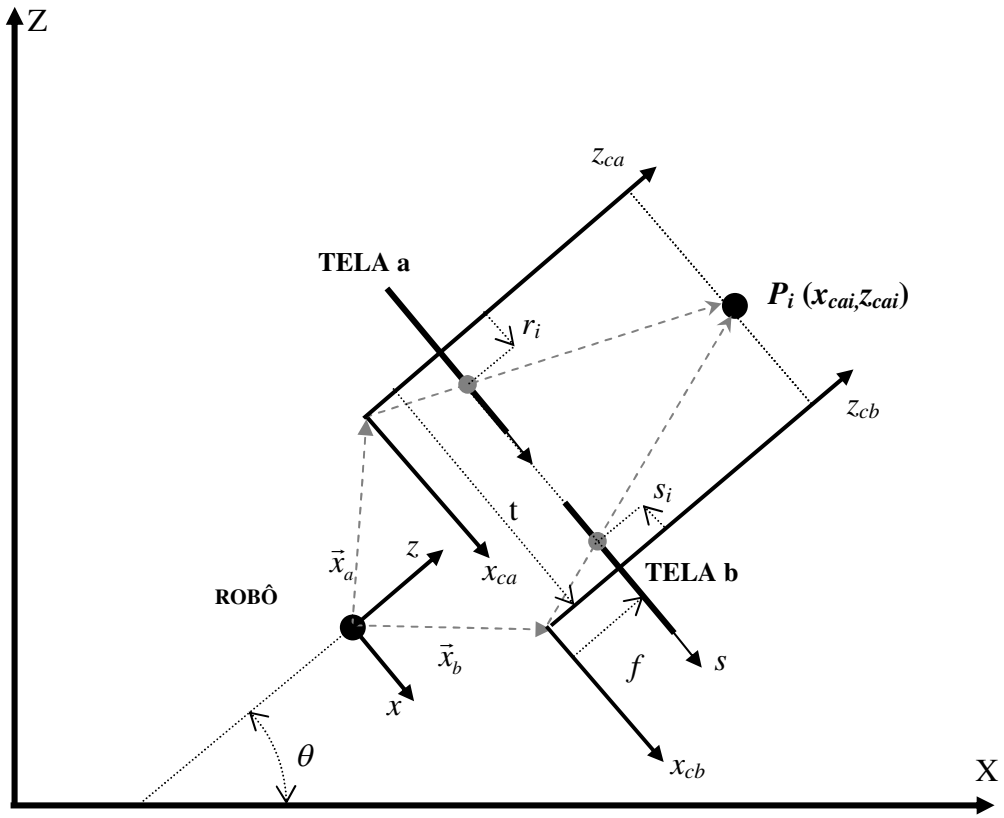


Figura 4.1: Referenciais das câmeras, do robô e do mundo.

O vetor de coordenadas (x_{cai},z_{cai}) descreve a posição do ponto P_i em relação aos eixos de referências da câmera *a*. Os elementos de $\mathbf{x}_{cai}$ são calculados a partir da equação (3.43), repetida a seguir por conveniência:

$$\mathbf{x}_{cai} = \begin{bmatrix} x_{cai} \\ z_{cai} \end{bmatrix} = \frac{t}{(r_i - s_i)} \begin{bmatrix} r_i \\ f \end{bmatrix} \tag{3.43}$$

A incerteza associada a P_i a partir de suas projeções na tela é dada por:

$$\mathbf{dx}_{cai} = \begin{bmatrix} dx_{cai} \\ dz_{cai} \end{bmatrix} \tag{4.1}$$

Derivando a equação (3.43), tem-se :

$$\mathbf{dx}_{cai} = \begin{bmatrix} \dfrac{t}{(r_i - s_i)} dr_i - \dfrac{r_i t}{(r_i - s_i)^2} dr_i + \dfrac{r_i t}{(r_i - s_i)^2} ds_i \\ -\dfrac{tf}{(r_i - s_i)^2} dr_i + \dfrac{tf}{(r_i - s_i)^2} ds_i \end{bmatrix} \tag{4.2}$$

Então:

$$\mathbf{dx}_{cai} = \begin{bmatrix} \dfrac{t(r_i - s_i) - r_i t}{(r_i - s_i)^2} & \dfrac{r_i t}{(r_i - s_i)^2} \\ -\dfrac{tf}{(r_i - s_i)^2} & \dfrac{tf}{(r_i - s_i)^2} \end{bmatrix} \begin{bmatrix} dr_i \\ ds_i \end{bmatrix} \tag{4.3}$$

Dessa forma, o erro de estimação do vetor $\mathbf{x}_{cai}$ é:

$$\mathbf{dx}_{cai} = \mathbf{B}_i \mathbf{dr}_i \tag{4.4}$$

onde a matriz $\mathbf{B}_i$ e o vetor $\mathbf{dr}_i$ são definidos como:

$$\mathbf{B}_i = \frac{z_{cai}^2}{ft} \begin{bmatrix} -\dfrac{s_i}{f} & \dfrac{r_i}{f} \\ -1 & 1 \end{bmatrix} \tag{4.5}$$

$$\mathbf{dr}_i = \begin{bmatrix} dr_i \\ ds_i \end{bmatrix} \tag{4.6}$$

O vetor de estado **Z**, ou *pose* do robô, obtida por imagens, pode ser definido a partir de dois pontos fixos *P* e *A* no mundo, conforme mostrado na figura (4.2). Nessa figura também se utiliza a nomenclatura de Z_X, Z_Z e Z_θ para indicar as coordenadas do robô no mundo, bem como a sua

posição angular Z_θ. Tal nomenclatura é introduzida, pois, conforme será mostrado, os erros associados às medidas das câmeras fazem com que as coordenadas estimadas de pose do robô sejam diferentes do seu estado real.

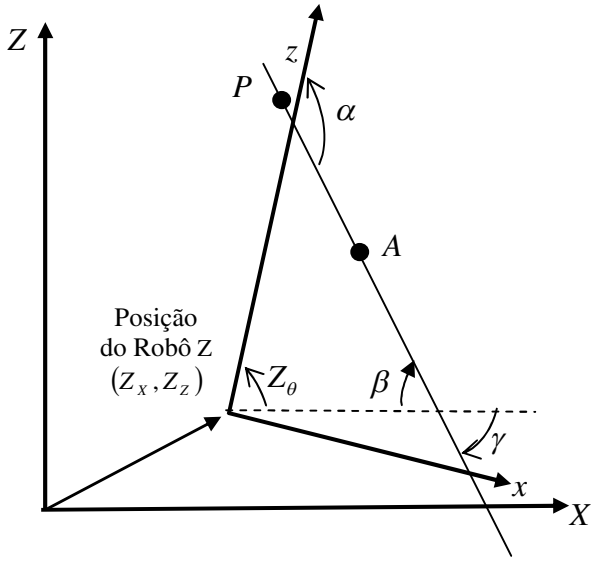


Figura.4.2: Localização do robô através dos pontos P e A.

Na figura (4.2) é mostrada a geometria de dois pontos *P* e *A* na região planar de movimento do robô. A equação (4.7) mostra como a pose do robô **Z** está relacionada com as coordenadas dos pontos *P* e *A* no mundo e com as coordenadas desses pontos vistos das câmeras:

$$\mathbf{Z} = \begin{bmatrix} Z_X \\ Z_Z \\ Z_\theta \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\left[\mathbf{P} + \mathbf{A} - \mathbf{R}^T\left(\mathbf{x}_{caP} + \mathbf{x}_{caA} + 2 * \mathbf{x}_a\right)\right] \\ \alpha + \gamma \end{bmatrix} \tag{4.7}$$

onde a matriz de rotação **R** é definida pelo ângulo Z_θ estimado pelas câmeras:

$$\mathbf{R} = \begin{bmatrix} Sin(Z_\theta) & -Cos(Z_\theta) \\ Cos(Z_\theta) & Sin(Z_\theta) \end{bmatrix} \tag{4.8}$$

A posição angular Z_θ do robô é calculada a partir dos ângulos α e γ, como mostra a figura (4.2). O ângulo entre a direção X e $\overline{PA}$ é dado por γ. O ângulo α é a inclinação entre o eixo z do robô segmento $\overline{PA}$ dado por:

$$\alpha = \frac{\pi}{2} + \beta \tag{4.9}$$

onde

$$\beta = \arctan\left(-\frac{z_{rP} - z_{rA}}{x_{rP} - x_{rA}}\right) = \arctan\left(\frac{z_{rP} - z_{rA}}{x_{rA} - x_{rP}}\right) \tag{4.10}$$

onde $\mathbf{P}_r=(x_{rP},z_{rP})$ e $\mathbf{A}_r=(x_{rA},z_{rA})$ são as coordenadas dos pontos P e A respectivamente no referencial do robô, assim sendo:

$$\mathbf{P}_r = \mathbf{x}_{caP} + \mathbf{x}_a \tag{4.11}$$

$$\mathbf{A}_r = \mathbf{x}_{caA} + \mathbf{x}_a \tag{4.12}$$

4.2 - Modelo de Erro de Posicionamento

O tratamento do erro da posição estimado pelas câmeras é muito importante, uma vez que ele é usado no filtro de Kalman para estimar a posição do robô num ambiente bidimensional.

As incertezas associadas ao vetor de estado do robô dado pelas câmeras são dadas pelas derivadas de cada um dos termos da equação (4.7):

$$d\mathbf{Z} = \begin{bmatrix} dZ_X \\ dZ_Z \\ dZ_\theta \end{bmatrix} \tag{4.13}$$

A derivada da equação (4.7) é:

$$d\mathbf{Z} = \left[\frac{1}{2}\left(-\frac{\partial \mathbf{R}^T}{\partial Z_\theta}\left(\mathbf{x}_{caP} + \mathbf{x}_{caA} + 2 * \mathbf{x}_a\right) dZ_\phi - \mathbf{R}^T\left(d\mathbf{x}_{caP} + d\mathbf{x}_{caA} + 2 * d\mathbf{x}_a\right) + d\mathbf{P} + d\mathbf{A} \right) \right] \quad (4.14)$$
$$dZ_\theta$$

e os vetores $d\mathbf{P}$ e $d\mathbf{A}$ estão relacionados aos erros de variação de posição dos pontos P e A, respectivamente, no mundo. Por serem valores absolutos, seus erros são nulos. O vetor $\mathbf{x}_a$ também é absoluto e, portanto seu erro $d\mathbf{x}_a$ é nulo.

$$dZ_\theta = d\alpha + d\gamma = d\beta + d\gamma \quad (4.15)$$

sendo $d\gamma = 0$ pois γ é um ângulo definido como absoluto. Para o cálculo de dZ_θ basta saber a derivada de β conforme a dedução abaixo:

$$dZ_\theta = d\beta \rightarrow d\left(\tan\left(\beta\right)\right) = \frac{1}{\cos^2\left(\beta\right)} d\beta \quad (4.16)$$

Derivando-se a equação (4.10), tem-se:

$$\frac{1}{\cos^2\left(\beta\right)} d\beta = \frac{1}{x_{rA} - x_{rP}} dz_{rP} - \frac{1}{x_{rA} - x_{rP}} dz_{rA} + \frac{z_{rP} - z_{rA}}{\left(x_{rA} - x_{rP}\right)^2} dz_{rP} - \frac{z_{rP} - z_{rA}}{\left(x_{rA} - x_{rP}\right)^2} dx_{rA} \quad (4.17)$$

Isolando-se $d\beta$:

$$d\beta = \frac{\cos^2\left(\beta\right)}{x_{rA} - x_{rP}}\left(\tan\left(\beta\right) dx_{rP} + dz_{rP} - \tan\left(\beta\right) dx_{rA} - dz_{rA}\right)$$

$$d\beta = \frac{\cos^2\left(\beta\right)}{x_{rA} - x_{rP}}\begin{bmatrix} \tan\left(\beta\right) & 1 & -\tan\left(\beta\right) & -1 \end{bmatrix}\begin{bmatrix} dx_{rP} \\ dz_{rP} \\ dx_{rA} \\ dz_{rA} \end{bmatrix} = \mathbf{C}\mathbf{d}\mathbf{r}_{PA} \quad (4.18)$$

onde a matriz $\mathbf{C}$ e o vetor $\mathbf{dr}_{PA}$ são definidos como:

$$\mathbf{C} = \frac{\cos^2(\beta)}{x_{rA} - x_{rP}} \begin{bmatrix} \tan(\beta) & 1 & -\tan(\beta) & -1 \end{bmatrix} \tag{4.19}$$

$$\mathbf{dr}_{PA} = \begin{bmatrix} \mathbf{dr}_P \\ \mathbf{dr}_A \end{bmatrix} \tag{4.20}$$

e sendo $\mathbf{dx}_a$ =0:

$$\begin{bmatrix} dx_{rP} \\ dz_{rP} \\ dx_{rA} \\ dz_{rA} \end{bmatrix} = \begin{bmatrix} dx_{caP} \\ dz_{caP} \\ dx_{caA} \\ dz_{caA} \end{bmatrix} \tag{4.21}$$

Substituindo dZ_θ em (4.14):

$$d\mathbf{Z} = \begin{bmatrix} -\frac{1}{2}\left(\frac{\partial \mathbf{R}^T}{\partial Z_\theta} \left(\mathbf{x}_{caP} + \mathbf{x}_{caA} + 2*\mathbf{x}_a \right) \mathbf{C}\mathbf{dr}_{PA} + \mathbf{R}^T \left(\mathbf{B}_P \mathbf{dr}_P + \mathbf{B}_A \mathbf{dr}_A \right) \right) \\ \mathbf{C}\mathbf{dr}_{PA} \end{bmatrix} \tag{4.22}$$

Finalmente $d\mathbf{Z}$ é escrito como:

$$d\mathbf{Z} = \begin{bmatrix} -\frac{1}{2}\left(\frac{\partial \mathbf{R}^T}{\partial Z_\theta} \mathbf{C}\left(\mathbf{x}_{caP} + \mathbf{x}_{caA} + 2*\mathbf{x}_a \right) + \mathbf{R}^T \begin{bmatrix} \mathbf{B}_P & \mathbf{B}_A \end{bmatrix} \right) \\ \mathbf{C} \end{bmatrix} \mathbf{dr}_{PA} \tag{4.23}$$

Sendo:

$$\mathbf{D} = -\frac{1}{2}\left[\frac{\partial \mathbf{R}^T}{\partial Z_\theta} \left(\mathbf{x}_{caP} + \mathbf{x}_{caA} + 2*\mathbf{x}_a \right) \mathbf{C} + \mathbf{R}^T \begin{bmatrix} \mathbf{B}_P & \mathbf{B}_A \end{bmatrix} \right] \tag{4.24}$$

com:

$$d\mathbf{R} = \begin{bmatrix} Cos(Z_\theta) & Sin(Z_\theta) \\ -Sin(Z_\theta) & Cos(Z_\theta) \end{bmatrix} \tag{4.25}$$

Reescrevendo $d\mathbf{Z}$:

$$d\mathbf{Z} = \begin{bmatrix} \mathbf{D} \\ \mathbf{C} \end{bmatrix} d\mathbf{r}_{PA} \tag{4.26}$$

Assumido que as incertezas das projeções dos pontos nas câmeras são associadas a um erro de discretização ν dado por:

$$\nu = \frac{1}{2}\frac{l}{N_{pix}} \tag{4.27}$$

onde l é o comprimento da tela e N_{pix} é o número de *pixels* igual para ambas as câmeras.

A matriz de covariância $\overline{\mathbf{R}}$ é dada pela incerteza $d\mathbf{Z}$:

$$\overline{\mathbf{R}} = \begin{bmatrix} \mathbf{D} \\ \mathbf{C} \end{bmatrix} \mathbf{E} \begin{bmatrix} \mathbf{D}^T & \mathbf{C}^T \end{bmatrix} \tag{4.28}$$

onde a matriz **E** é dada por:

$$\mathbf{E} = \overline{\sigma}^2 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.29}$$

sendo $\overline{\sigma}$ o desvio padrão do erro da câmera.

O próximo capítulo descreve o processo de se obter a localização ótima do robô baseado nas informações de posição e matrizes de covariância descritas nos capítulos 2 e 4.

Capítulo 5

Navegação Otimizada

Os sensores de odometria apresentam erros que vão se acumulando com a navegação. Já a câmera não apresenta erros cumulativos, mas possui o erro de projeção dos objetos na tela inerente ao processo de discretização espacial de seus *pixels*. Assim, de forma a minimizar os erros oriundos da odometria e visão através de câmeras, utiliza-se o algoritmo de obtenção de estimativas ponderadas conhecido como filtro de Kalman

5.1 – O Filtro de Kalman

O filtro de Kalman (Welch, 2001; Negenborn, 2003; Brown e Hwang, 1997; Santana, 2005) é baseado na satisfação de três condições: o sistema deve ser linear, os ruídos associados ao processo e às medidas devem ser brancos e gaussianos. Assim, aplicado no caso específico do robô móvel, o filtro fornece uma estimativa ótima de seu estado **X**, dada uma medida independente do mesmo estado, definida como **Z**.

O filtro tem seu algoritmo divido em duas partes, uma referente à predição e a outra à correção. Seja $\mathbf{X}_k$ o estado ou postura (ou *pose* em inglês) real do robô no instante k e $\mathbf{u}_k$ o sinal de entrada de leitura do sistema de odometria. No presente caso, $\mathbf{X}_k$ é composto das coordenadas de translação e rotação, definidas como (X,Z,θ).

A equação geral de um modelo que se pode aplicar no filtro é representada pelo sistema linear descrita pela equação (5.1):

$$\mathbf{X}_k = \mathbf{F}_k \mathbf{X}_{k-1} + \mathbf{G}_{k-1}\mathbf{u}_k + \boldsymbol{\mu}_k \tag{5.1}$$

onde $\mathbf{F}_k$ é a matriz de transição que projeta o estado atual $\mathbf{X}_k$ no tempo k+1 (para o modelo do capítulo 2, ela é a matriz identidade), $\mathbf{G}_{k-1}$ é a matriz de controlabilidade, associada ao vetor de controle do sistema. Assume-se no presente modelo que o vetor de ruído do processo, $\boldsymbol{\mu}_k$ seja igual a $\mathbf{G}_{k-1}\beta_{\mathbf{k}}$,. Os elementos do vetor β_k possuem distribuição de probabilidade gaussiana com desvio padrão σ, ou seja,

$$p(\beta_{\mathbf{k}}(i)) \sim N(0,\sigma) \tag{2.17}$$

sendo i=1,2.

A distribuição gaussiana é conhecida como distribuição normal, sendo da forma:

$$p(\tilde{X}) = N(m,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(\tilde{X}-m)^2}{\sigma^2}} \tag{5.2}$$

onde o valor médio é dado por *m* e a variância por σ^2.

Uma medida de observação independente do estado do sistema, definida pelo vetor $\mathbf{Z}_k$, relaciona-se com o estado estimado pela equação (5.3) como:

$$\mathbf{Z}_k = \mathbf{H}\,\mathbf{X}_k + \Sigma_k \tag{5.3}$$

onde $\mathbf{H}$ é a matriz de observabilidade e Σ_k é um vetor de ruído, associado à incerteza da medida de observação do sistema. Aqui também o vetor de ruído Σ_k é assumido como tendo distribuição de probabilidade normal, com matriz de covariância $\mathbf{R_k}$, ou seja,

$$p(\Sigma_k) \sim N(0,\mathbf{R}_k) \tag{5.4}$$

A medida de observação do estado do sistema,neste trabalho, é oriunda do processo de visão computacional.

Na fase inicial da estimação, o estado "a priori" do sistema, $\hat{\mathbf{X}}_k^-$, é calculado à partir do estado ótimo anterior, $\hat{\mathbf{X}}_{k-1}$ e dos valores de entrada do sistema de odometria, $\mathbf{u}_k$, segundo a equação:

$$\hat{\mathbf{X}}_k^- = \mathbf{F}_k \hat{\mathbf{X}}_{k-1} + \mathbf{G}_k \mathbf{u}_k \tag{5.5}$$

Sabe-se que tal estimativa é acompanhada de uma incerteza, inferida pela perturbação aleatória contida no vetor $\boldsymbol{\mu}_k$ (equação (5.1)). Ao mesmo tempo, tem-se a leitura de posição $\mathbf{Z}_k$, oriunda da técnica de visão computacional, que por sua vez é acompanhada da incerteza de observação do sistema, Σ_k.

A estimativa *a posteriori* do estado do sistema otimizada, $\hat{\mathbf{X}}_k$, é dada por:

$$\hat{\mathbf{X}}_k = \hat{\mathbf{X}}_k^- + \mathbf{K}_k \left(\mathbf{Z}_k - \mathbf{H}\hat{\mathbf{X}}_k^- \right) \tag{5.6}$$

onde $\mathbf{K}_k$ é a matriz de ganho, que minimiza o erro de covariância *a posteriori*. O valor entre parênteses é chamado de inovação. Os detalhes para a obtenção da matriz $\mathbf{K}_\mathrm{k}$ podem ser vistos no apêndice B, e sua expressão é dada por

$$\mathbf{K}_\mathrm{k} = \mathbf{P}_k^- \mathbf{H}^T \left(\mathbf{H}\mathbf{P}_k^- \mathbf{H}^T + \overline{\mathbf{R}}_k \right)^{-1} \tag{5.7}$$

onde o valor entre parênteses é número chamado de covariância da inovação, S.

A matriz $\mathbf{P}_k^-$, da expressão (5.6), é a matriz de covariância do erro de estimação *a priori*, e que pode ser calculada por

$$\mathbf{P}_k^- = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q}_k \tag{5.8}$$

O termo $\mathbf{P}_{k-1}$ que aparece na expressão (5.8) é a matriz de covariância do erro total de estimação, calculado na iteração anterior (*k-1*). A matriz de covariância do erro total de estimação

atual, ou erro de estado *a posteriori*, pode ser calculada para uso no passo seguinte através da expressão:

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H})\mathbf{P}_k^- \tag{5.9}$$

Ainda, a matriz de covariância do erro total de estimação ao início do laço interativo, pode ser tomada com o mesmo valor de covariância de ruído do processo, ou seja,

$$\mathbf{P}_0 = \mathbf{Q}_0 \tag{5.10}$$

Para ilustrar a aplicação do filtro de Kalman considere um veículo se movendo em uma estrada retilínea, onde a aceleração que deveria ser constante, se altera por causa da queima de combustível que é aleatória. Considere que a cada T segundos a posição possa ser medida para cada variação da aceleração. O espaço percorrido s_{k+1} a partir do inicial s_k e a velocidade do veículo v_k são dados matricialmente por:

$$\mathbf{X}_k = \begin{bmatrix} s_k \\ v_k \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \mathbf{X}_{k-1} + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} u_k + \boldsymbol{\mu}_k \tag{5.11}$$

sendo:

$$\boldsymbol{\mu}_k = \begin{bmatrix} T^2/2 \\ T \end{bmatrix} * i \tag{5.12}$$

onde i é um número aleatório que segue uma distribuição gaussiana representando o ruído do processo introduzido no sistema. O desvio padrão do ruído do sistema é dado por σ_u.

A medida de posição pode ser obtida por algum sensor que a meça diretamente, acrescida de um erro, de acordo com a seguinte equação:

$$Z_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{X}_k + \bar{i} \tag{5.13}$$

onde $\bar{i}$ é um número aleatório que segue uma distribuição gaussiana representando o ruído de medida do sistema com desvio padrão σ_p.

Suponha que o desvio padrão da posição σ_p seja 10 m, e que o desvio padrão da aceleração σ_u seja 0,5 m/s^2, u_k=0 m/s^2 e T= 0,5 s. Então, $\mathbf{X}_k$, a posição real que não se consegue medir, é dada pela equação (5.11), substituindo-se valores para T e u_k. Para efeito de comparação, mede-se a posição absoluta para se averiguar se há uma convergência entre a posição estimada via filtro de Kalman e a posição real, ou seja, para observar se o filtro fornece uma estimativa confiável.

Para calcular a posição estimada $\hat{\mathbf{X}}_k$, de acordo com a equação (5.6), precisa-se conhecer o ganho (equação (5.7)), sendo que este é dependente das matrizes de covariância $\mathbf{Q_k}$ e $\overline{\mathbf{R}}_\mathbf{k}$. A matriz de covariância do ruído do processo é dada pelo valor esperado da parte da equação (5.11) que contem o ruído, isto é, o vetor $\boldsymbol{\mu}_k$:

$$\mathbf{Q}_k = E\left[i\begin{bmatrix} T^2/2 \\ T \end{bmatrix} i\begin{bmatrix} T^2/2 & T \end{bmatrix}\right] = \sigma_u^2 * \begin{bmatrix} T^4/4 & T^3/2 \\ T^3/2 & T^2 \end{bmatrix} \tag{5.14}$$

A matriz $\mathbf{Q}_k$ é constante em cada iteração. Em nossa simulação, no caso do robô móvel, essa matriz varia em cada iteração, devido à dependência de um valor angular variante para cada k (equação 2.24).

A matriz covariância de ruído $\overline{\mathbf{R}}_\mathbf{k}$, é encontrada calculando o valor esperado da parte da equação (5.13) que contem o valor de ruído:

$$\overline{\mathbf{R}}_k = E\left[\bar{i}\,\bar{i}\right] = \sigma_p^2 \tag{5.15}$$

A matriz $\overline{\mathbf{R}}_\mathbf{k}$, assim como $\mathbf{Q}_k$, é constante aqui. Na simulação deste trabalho, esta varia em cada iteração por depender do valor de um ângulo, cujo valor muda a cada iteração (equação 4.28).

Uma vez determinadas as matrizes $\mathbf{Q}_k$ e $\overline{\mathbf{R}}_\mathbf{k}$, assim como os valores iniciais do vetor $\hat{\mathbf{X}}_k^-$ e da matriz $\mathbf{P}_k$, a posição estimada $\hat{\mathbf{X}}_k$ finalmente pode ser obtida.

O método exemplificado anteriormente é uma demonstração simplificada do método de obtenção de posição que é apresentando neste trabalho. Isso porque, no exemplo, o veículo está andando em uma única direção. Neste trabalho, a posição é medida em duas coordenadas lineares e uma angular. O exemplo simula um erro muito simples na medida de posição Z_k. O valor da medida de posição usado aqui vem da visão, e várias mudanças de referencial são usadas para se obter a posição do robô via imagens.

Um esquema do processo de estimação do estado através do filtro de Kalman é apresentado à figura (5.1).

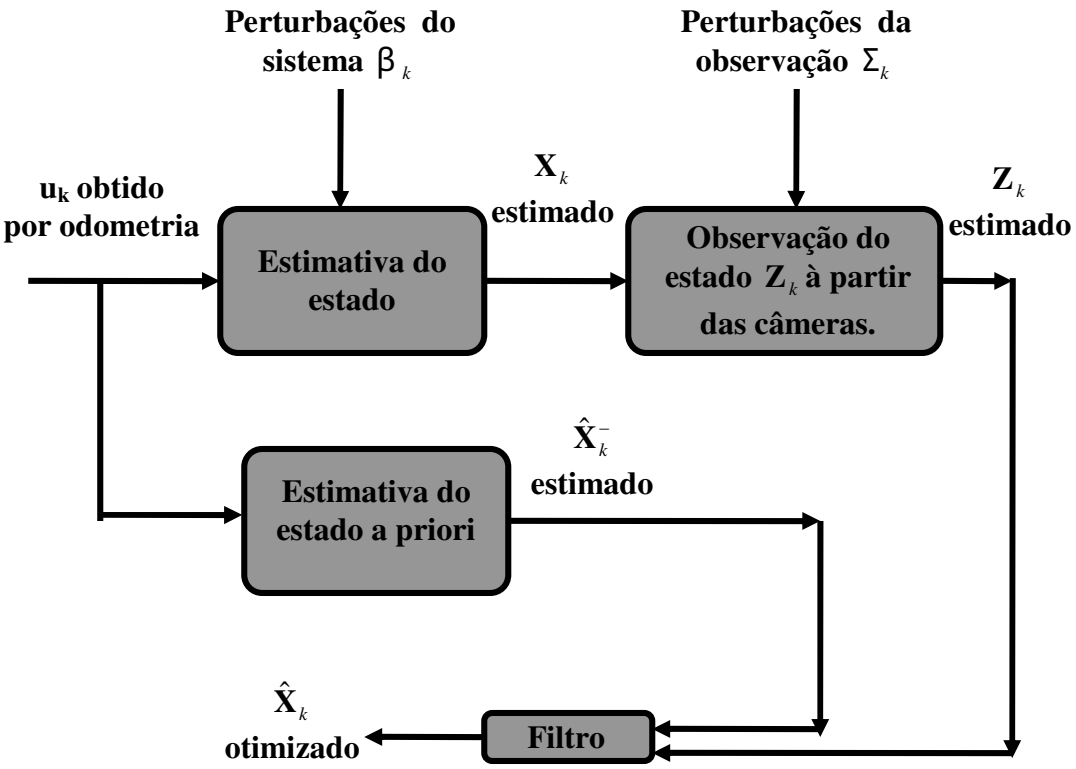


Figura 5.1: Esquema de implementação do filtro de Kalman.

A figura (5.1) é explicada de forma simplificada, pois maiores detalhes sobre a navegação do robô encontram-se no capítulo 7. Nessa figura, o valor de entrada $\mathbf{u_k}$ (equação (2.15)) é o vetor que contém as informações do número de voltas das rodas da esquerda e da direita, num trecho genérico da trajetória ideal do robô, que é uma reta traçada de sua posição estimada $\hat{\mathbf{X}}_k$ até um portal de passagem. A medida real $\mathbf{X}_k$ **,** de acordo com as descrições do capítulo 2, é obtida através

do vetor $\mathbf{u_k}$ e do vetor de ruído β_k, que introduz perturbações aleatórias independentes para $[N_d \quad N_e]$. Para observar a posição do robô, o sensor usado é um par de câmeras, cujos valores de erros estão associados ao vetor Σ_k. A estimação da posição através da medida, $\mathbf{Z}_k$, portanto, inclui a incerteza associada ao erro de observação das câmeras.

A estimação da posição $\hat{\mathbf{X}}_k$ depende tanto de $\mathbf{Z}_k$ como do valor *a priori* $\hat{\mathbf{X}}_k^-$, sendo esta posição dependente do vetor $\mathbf{u_k}$**.** Com a fusão dos valores de ambos os estados, e mais as matrizes de covariância $\mathbf{Q_k}$ e $\overline{\mathbf{R}}_\mathbf{k}$, obtém-se finalmente a posição estimada.

O capítulo a seguir mostra uma visão de controle híbrido, onde o vetor de controle $\mathbf{u_k}$ usado para o robô seguir uma trajetória pré-determinada, pode ser substituído por um outro tipo controle, modificando a trajetória do robô. Esse tipo de controle, por exemplo, pode ser aplicado para que robô se desvie de um obstáculo.

Capítulo 6

Utilização do Controle Híbrido

O comportamento geral de navegação robótica deve possuir um caráter híbrido onde o robô tem como objetivo principal navegar por uma trajetória pré-definida, cessando esse padrão de navegação quando encontra alguma situação imprevista ou adversa, como a necessidade de realizar um desvio para evitar uma possível colisão. Dessa maneira, uma estratégia de controle adicional é necessária, a fim do retirar o robô do modo de padrão de navegação, até certo ponto de sua trajetória, onde possa retomar ao cumprimento de seu principal objetivo. As considerações sobre aplicação de controle híbrido são apresentadas aqui, como ferramenta para aperfeiçoamento futuro dos procedimentos de controle. O conceito de aplicação de uma estratégia híbrida para desvio de obstáculos do robô do projeto Pilgrim é proposto ao final do capítulo.

6.1 - Sistemas Híbridos

Um sistema híbrido é descrito como um sistema dinâmico que envolve dois diferentes tipos de estado: um estado dinâmico contínuo e um estado dinâmico discreto. Os estados variáveis no tempo são considerados discretos se ele toma finitos valores e contínuo se ele toma valores no espaço Euclidiano $\Re^n$ onde $n \geq 1$. Pela sua natureza, a alteração dos valores dos estados discretos só ocorrem em saltos, e os estados contínuos têm seus valores alterados seguindo uma equação diferencial, e também nos saltos. Ambos os sistemas dinâmicos são aplicados nos sistemas híbridos. A análise dos modelos de sistemas híbridos, em geral, são mais complicados que os sistemas

puramente discretos ou contínuos, uma vez que o sistema discreto influencia no contínuo e vice-versa.

Modelos usando dinâmica híbrida são muito aplicados na área de engenharia:

- Em sistemas mecânicos, onde o movimento contínuo pode ser interrompido por colisões.
- Em circuitos elétricos, onde o carregamento de um capacitor pode ser interrompido por uma chave on-off.
- Um sistema químico com reação contínua pode ser controlado por válvulas.

Nesses modelos citados os componentes discretos são os obstáculos, chaves e válvulas, respectivamente, que introduzem mudanças na velocidade de um veículo, no carregamento de um capacitor e na reação química.

O estudo do comportamento híbrido pode ser exemplificado pelos modelos a seguir:

1) Salto de uma Bola

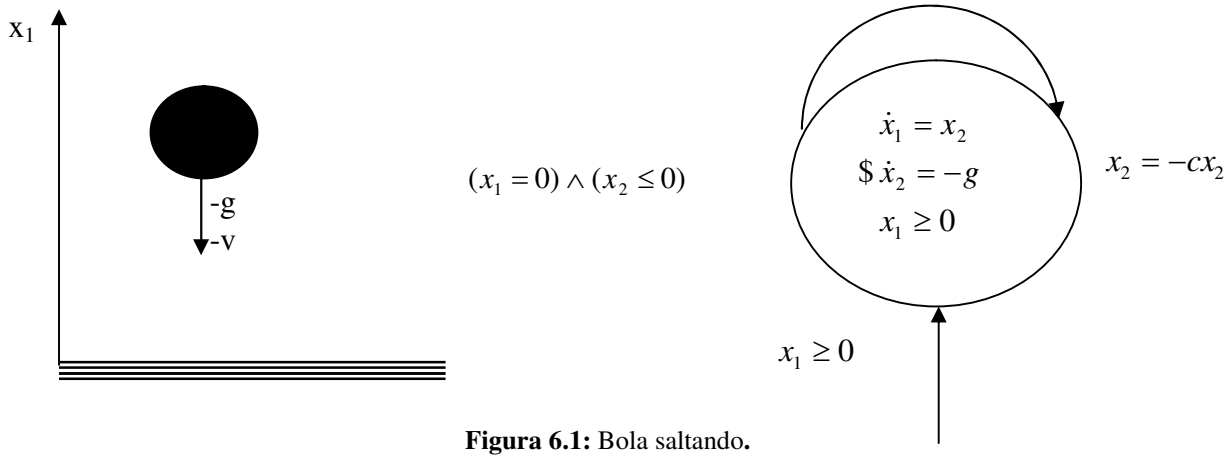


Figura 6.1: Bola saltando**.**

Uma bola saltando sobre o chão pode der representada como um sistema híbrido simples (figura (6.1)). Nesse caso, temos um único estado discreto x e dois estados contínuos x_1 e x_2, onde x_1 é a posição vertical da bola e x_2 a sua velocidade:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{6.1}$$

A equação diferencial que aparece dentro do círculo é baseada nas leis de Newton e descreve o movimento contínuo da bola, onde g é aceleração da gravidade. Essa equação é válida somente quando a bola está acima do chão, isto é, $x_1 \geq 0$, indicando queda livre e isto está sendo indicado abaixo pelas equações. Em queda livre, tem-se:

$$\begin{aligned} x_1 &= x_{10} + v_0 t + \frac{at^2}{2} = x_{10} - \frac{gt^2}{2} \\ \dot{x}_1 &= -gt = x_2 \\ \dot{x}_2 &= -g \end{aligned} \tag{6.2}$$

onde x_{10} é a altura máxima, v_0 é a velocidade inicial, no caso é nula, g é a aceleração da gravidade ≈ 10 m/s^2.

A bola muda sua direção quando $x_1=0$ e $x_2 \leq 0$ (no instante em que toca o chão). A partir daí, uma fração de energia c é perdida, e a velocidade inicial de subida é em sentido contrário (positiva) é menor. A constante de dissipação c $\in$ [0,1], se c <1 tem-se colisão inelástica e c=1, é elástica.

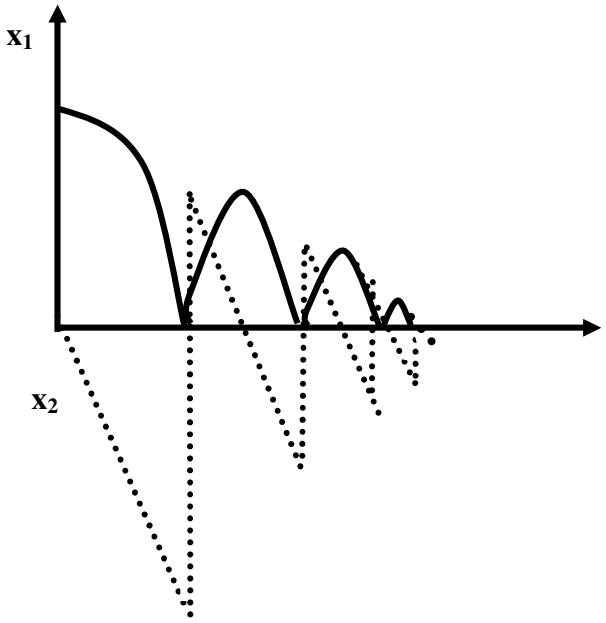


Figura 6.2: Representação da trajetória x_1 (linha preenchida) e da velocidade x_2 (linha pontilhada) de uma bola saltando em função do tempo.

2) Termostato

Podemos supor um ambiente sendo aquecido por um radiador e controlado por um termostato. A temperatura x decai exponencialmente até 0 quando o termostato é desligado (off) de acordo com a equação diferencial:

$$\dot{x} = -ax \tag{6.3}$$

para qualquer a > 0. A solução, é portanto:

$$x = e^{-at} + Const. \tag{6.4}$$

Quando o termostato é operado para aquecer (em ON), a temperatura aumenta exponencialmente até 30 graus, seguindo a equação:

$$\dot{x} = -a(x - 30) \tag{6.5}$$

A solução é:

$$x(t) = 30(1 - e^{-at}) + Const. \tag{6.6}$$

Desse modo, quando $t \to \infty$ quando o aquecedor é ligado, a temperatura fica em torno dos 30 graus.

Suponha que o termostato tente manter a temperatura em torno de uns 20 graus. Desse modo, o sistema funcionará on/off para as temperaturas limites 19 e 21 graus, o aquecedor será ligado a temperaturas menores que 19 graus e desligado se maior que 21. Devido a incertezas na dinâmica no radiador, a temperatura ainda pode descer mais até uns 18 graus ou subir até os 22 graus.

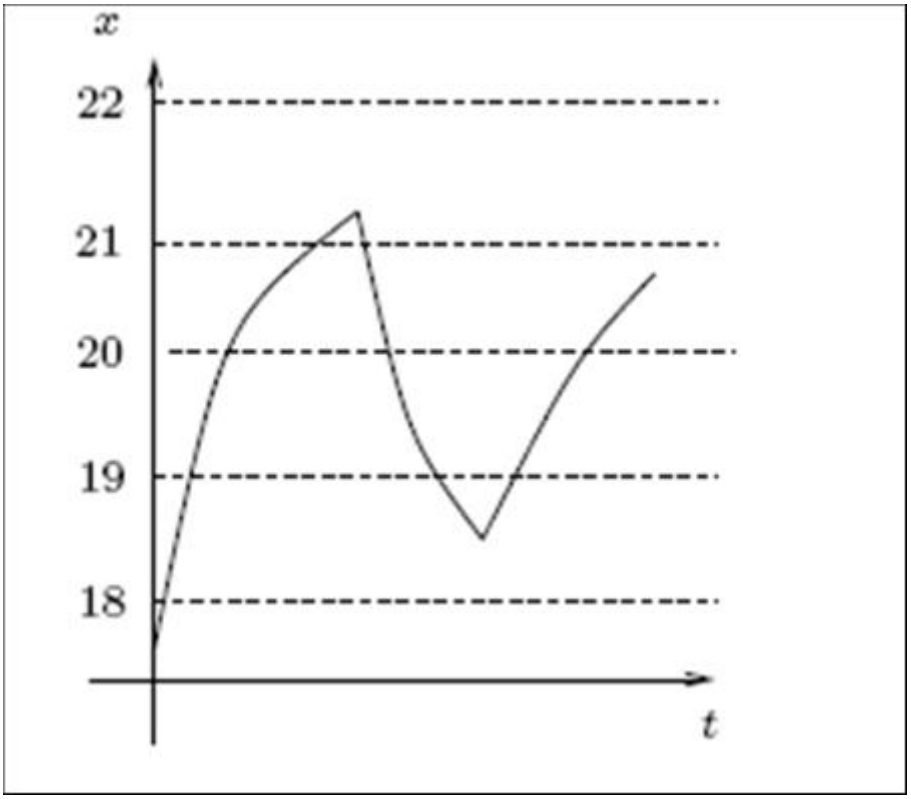


Figura 6.3: Termostato operando em ON/OFF (Lygeros, 2004).

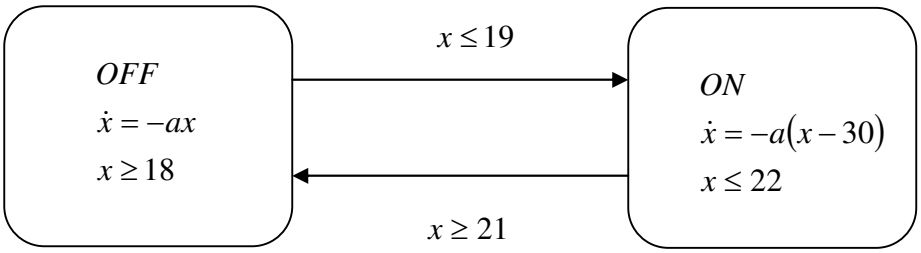


Figura 6.4: Notação usada para um sistema com termostato.

Esse sistema também tem estados contínuos e discretos. O estado contínuo é a temperatura do ambiente, e o estado discreto está relacionando se o termostato está ligado ou desligado, q = {ON, OFF}. Há um acoplamento desses dois estados: quando q = ON, a temperatura aumenta de acordo com a equação diferencial (6.5), e quando q = OFF, x decai de acordo com a equação diferencial (6.3).

6.2 - Autômatos Híbridos

Um modelo para um sistema cujo comportamento é governado por componentes discretas e contínuas constitui um autômato híbrido. Dessa forma, um autômato híbrido é a união de um autômato finito com variáveis que evoluem continuamente no tempo. No autômato finito é assumido a variável temporal evoluir em forma discreta, isto é, $0, \pm 1, \pm 2, \ldots$.

Matematicamente, um autômato híbrido H é uma coleção H= (Q, X, f, Init, D, E, G, R), onde:

- $Q = \{q_1, q_2, ...\}$ é um conjunto de estados discretos;
- $X = \Re^n$ é um conjunto de estados contínuos;
- f (.,.): $Q \times X \rightarrow \Re^n$ é um campo vetorial;
- Init $\subseteq$ Q x X é um conjunto de estados iniciais;
- Dom (.): $Q \rightarrow P(X)$ é um domínio;
- $E \subseteq Q \times Q$ é um conjunto de bordas;
- G (.): $E \rightarrow P(X)$ é uma condição de guarda;
- R (.,.): E x X $\rightarrow$ é um mapa "ressetado".

Um autômato híbrido define as evoluções temporais para seus estados (q,x), onde um estado $(q,x) \in Q \times X$ como um estado de H. Começando de um estado inicial $(q_0, x_0) \in$ Init, esse estado contínuo x flui de acordo com a equação diferencial:

$$\dot{x} = f(q_0, x) \tag{6.7}$$

$$x(0) = x_0 \tag{6.8}$$

enquanto o estado discreto *q* permanece constante:

$$q(t) = q_0 \tag{6.9}$$

A evolução contínua nesse estado acontece até quando *x* permanecer em $Dom(q_0)$. Em certo momento, o estado contínuo *x* atinge a condição de guarda $G(q_0, q_1) \subseteq \Re^n$ de algum limite (q_0, q_1)

$\in$ E, e o estado discreto muda para um valor q_1. As transições entre estados ocorrem quando as condições de guarda são atingidas. Então, os valores do estado contínuo $R(q_0,q_1,x) \subseteq \Re^n$ são "ressetados". Depois desta transição discreta, a evolução contínua recomeça e este processo é repetido novamente.

Seja uma simplificação supondo que o número de estados discretos é finito, e que para todo q $\in$ Q, o campo vetorial f (q,.) é uma função contínua de Lipschitz. Isso garante que para todo e $\in$ E, $G(e) \neq 0$, e para todo x $\in$ G(e), $R(e,x) \neq 0$. Isso elimina alguns casos indesejáveis.

Seja uma função $f: \Re^n \to \Re^n$. Se ela é uma função contínua de Lipschitz então existe certo $\lambda > 0$, tal que para todo x, $\hat{x} \in \Re^n$:

$$\left\| f(x) - f(\hat{x}) \right\| < \lambda \left\| x - \hat{x} \right\| \tag{6.10}$$

onde λ: constante de Lipschitz.

Um exemplo é a função:

$$f(x) = x^2 \tag{6.11}$$

que é uma função contínua de Lipschitz em [-3,7] para $\lambda = 14$.

O uso de grafos (Q, E) é empregado para representar autômatos híbridos com vértices Q e bordas E. Para cada vértice q $\in$ Q, é associado um conjunto de estados $\{x \in \mathbf{X} \mid (q,x) \in \text{Init}\}$, um campo vetorial $f(q,.): \Re^n \to \Re^n$ e um domínio $\text{Dom}(q) \subseteq \Re^n$. Uma borda $(q,q') \in$ E, começa em q $\in$ Q e termina em q' $\in$ Q. Cada $(q,q') \in$ E, é associado $G(q,q') \subseteq \Re^n$ e a uma função "reset" $R(q,q'): \Re^n \to P(\Re^n)$.

Considere a figura (6.5) como um exemplo de um autômato híbrido:

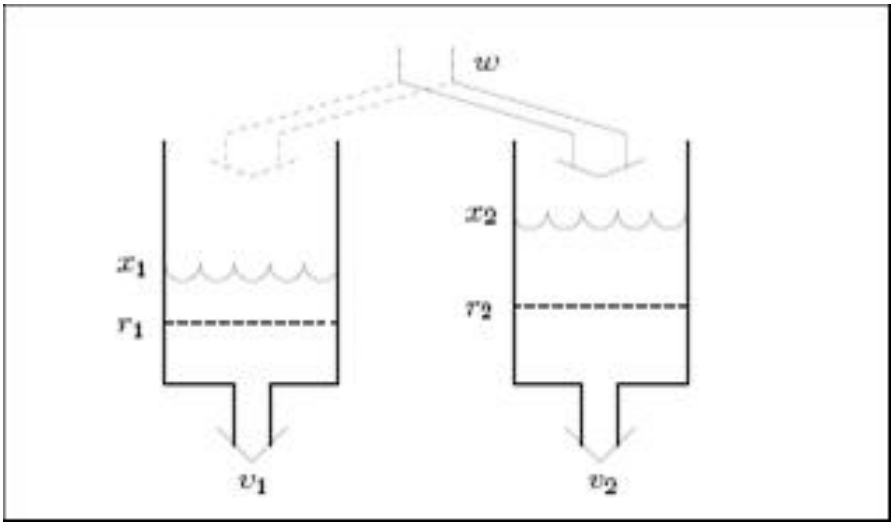


Figura 6.5: Um sistema de um tanque com água.

Neste exemplo (figura (6.5)) existem dois tanques inicialmente com água em que ambos estão vazando a uma taxa constante. Mais água é introduzida no sistema por uma mangueira a um fluxo constante. Considere que a mangueira pode chavear os dois tanques instantaneamente. O objetivo é manter a água acima dos volumes r_1 e r_2 assumindo que inicialmente os volumes de água em ambos os tanques estão acima de r_1 e r_2.

Seja i=1,2, e considere o volume de água do tanque i como sendo x_i e v_i como sendo o fluxo de água para fora. O fluxo constante de água que entra no sistema é denotado por ω. O controle de volume é feito por um controlador que liga o fluxo de entrada em um tanque i sempre que $x_i \leq r_i$.

Esse autômato híbrido é definido mais claramente como:

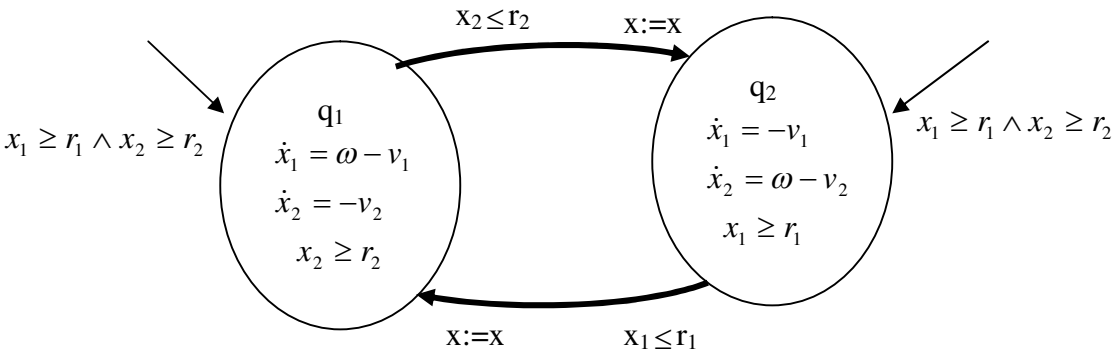


Figura 6.6: Representação gráfica do autômato híbrido representado por um sistema de tanques.

- Q = $\{q_1,q_2\}$ são dois estados discretos que representa entrada de água no tanque 1 ou no tanque 2;
- X = $\Re^n$ representa dois estados contínuos, ou seja, o nível de água de cada tanque;
- Quando a água entra no tanque 1, o nível de água de 2 apenas diminui cada vez mais e vice-versa, isto é:

$$f(q_1,x)=\begin{bmatrix}\omega-v_1\\ -v_2\end{bmatrix},\ f(q_2,x)=\begin{bmatrix}-v_1\\ \omega-v_2\end{bmatrix} \tag{6.12}$$

- Init = $\{q_1,q_2\} \times \{x \in \Re^2 \mid x_1 \geq r_1 \wedge x_2 \geq r_2\}$, ambos os níveis de água começam acima do limite r_1 e r_2;
- Dom $(q_1) = \{x \in \Re^2 \mid x_2 \geq r_2\}$ e Dom $(q_2) = \{x \in \Re^2 \mid x_1 \geq r_1\}$. Essa expressão significa que é colocado água num tanque enquanto o outro estiver acima do nível mais baixo, r_i;
- E = $\{(q_1,q_2), (q_2,q_1)\}$, são chaveamentos possíveis do fluxo de entrada, da esquerda para a direita, ou vice-versa;
- $G(q_1,q_2) = \{x \in \Re^2 \mid x_2 \leq r_2\}$ e $G((q_2,q_1)) = \{x \in \Re^2 \mid x1 \leq r_1\}$, chaveamento para um outro tanque quando a água atinge o nível mais baixo,
- $R(q_1,q_2,x) = R(q_2,q_1,x) = \{x\}$, o estado contínuo não se altera quando há interrupção.

6.3 - Conjunto Híbrido Temporal

Um conjunto híbrido é uma sequência de intervalos $\tau = \{I_0, I_1, ..., I_N\} = \{I_i\}_{i=0}^{N}$, finito ou infinito ($N=\infty$) tal que:

- $I_i = [\tau_i, \tau_i']$ para todo $i < N$;
- Se $N < \infty$ então também $I_N = [\tau_N, \tau_N']$ ou $I_N = [\tau_N, \tau_N')$ e
- $\tau_i < \tau_i' = \tau_{i+1}$ para todo i.

Esses critérios são analisados como o exemplo representado pela figura (6.7):

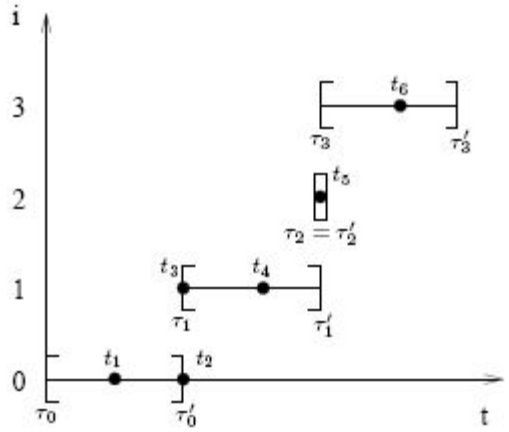


Figura 6.7: Um exemplo de conjuntos híbridos temporais $\tau = \{[\tau_i, \tau_i']\}_{i=0}^{3}$.

Observando a figura (6.7) é visto que o ponto final τ_i' do intervalo I_i coincide com o ponto inicial τ_{i+1} do intervalo adjacente I_{i+1}, como t_2 e t_3 em τ_0' e τ_1, respectivamente. Isso indica que para transições discretas ocorrerem tem que existir algum tempo específico. O tempo antes da transição discreta é dado por τ_i' e o tempo justamente no instante após a transição é dado por τ_{i+1}. Desse modo, as transições discretas são assumidas instantâneas quando $\tau_i' = \tau_{i+1}$. Modelos onde ocorrem múltiplas transições discretas uma após a outra podem ser criados de modo que $\tau_{i-1}' = \tau_i$ $= \tau_i' = \tau_{i+1}$ como no intervalo $I_2 = [\tau_2, \tau_2']$.

Apesar da análise de um sistema híbrido ser mais complicada do que um caso discreto ou contínuo puro, o conjunto híbrido temporal τ é bem comportado matematicamente, podendo ser ordenado. Por exemplo, se $t_1 \in [\tau_i, \tau_i'] \in \tau$ e $t_2 \in [\tau_j, \tau_j'] \in \tau$ dizemos que t_1 precede t_2 ($t_1 \prec t_2$) se $t_1 < t_2$ ou se $i < j$, ou seja, pelo intervalo sabemos quem é o tempo que precede o outro. Pela figura (6.7), temos que $t_1 \prec t_2 \prec t_3 \prec t_4 \prec t_5 \prec t_6$. Usando a linguagem matemática, se é dito que o cada conjunto híbrido temporal τ é linearmente ordenado em relação a $\prec$.

Dados dois conjuntos híbridos temporais τ e $\hat{\tau}$, pode-se analisar qual deles é o menor (τ é prefixo de $\hat{\tau}$ se ele for menor). De uma maneira mais formal, a forma escrita é $\tau = \{I_i\}_{i=0}^{N}$ é um

prefixo de $\hat{\tau} = \{\hat{I}_i\}_{i=0}^{M}$ se eles são idênticos ou τ (referente a N) é uma sequência finita de modo que N$\leq$ M, sendo que M pode ser infinito. Pode ser dito que τ é um prefixo absoluto de $\hat{\tau}$ se $\tau \subseteq \hat{\tau}$ e $\tau \neq \hat{\tau}$, ou seja, uma sequência está dentro de uma outra que é ainda maior que aquela. Na figura (6.8) τ é um prefixo absoluto de $\hat{\tau}$ e $\tilde{\tau}$, já que τ é uma sequência que está tanto contida em $\hat{\tau}$ como em $\tilde{\tau}$. O conjunto $\hat{\tau}$ não é prefixo de $\tilde{\tau}$ e nem vice-versa. Desse modo, dados dois conjuntos τ e $\hat{\tau}$, pode-se ter $\tau \not\subset \hat{\tau}$ e $\hat{\tau} \not\subset \tau$. Como por exemplo, se o conjunto A e B sejam definidos como A=$\{1,2\} \in \Re$ e B=$\{4,5\} \in \Re$. Mas nem A$\subseteq B$ e nem B$\subseteq$A. Usando uma linguagem matemática é dito que os conjuntos híbridos temporais são parcialmente ordenados em relação a $\subseteq$. Apenas $\tau \subset \hat{\tau}$ e $\tau \subset \tilde{\tau}$.

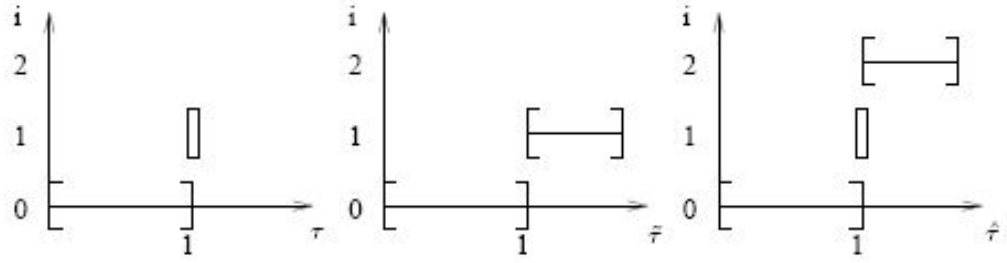


Figura 6.8: Verificação para analisar se os conjuntos híbridos temporais são prefixos absolutos.

6.4 - Trajetórias Híbridas

Um conjunto (τ, q, x) é definido como trajetória híbrida, onde, $\tau = \{I_i\}_0^N$, q=$\{q_i(.)\}_0^N$ e x=$\{x_i(.)\}_0^N$ onde $q_i(.):I_i \rightarrow Q$ e $x_i(.):I_i \rightarrow \Re^n$.

Executar uma trajetória híbrida (τ, q, x) de um autônomo híbrido H significa satisfazer as seguintes condições:

- Condição inicial: $(q_0(\tau_0), x_0(\tau_0)) \in$ Init.

  Essa condição indica que as execuções devem começar num tempo inicial em Init. Podemos usar ainda a seguinte notação $(q_0,x_0)= (q_0(\tau_0), x_0(\tau_0))$.

- Ter uma evolução discreta: para todo i, $\left(q_i\left(\tau_i^{'}\right), q_{i+1}\left(\tau_{i+1}\right)\right) \in E$, $x_i\left(\tau_i^{'}\right) \in G\left(\left(q_i\left(\tau_i^{'}\right), q_{i+1}\left(\tau_{i+1}\right)\right)\right)$ e $x_{i+1}\left(\tau_{i+1}\right) \in R\left(q_i\left(\tau_i^{'}\right), q_{i+1}\left(\tau_{i+1}\right), x_i\left(\tau_i^{'}\right)\right)$.

Isso mostra quando as transições discretas ocorrem e como serão os estados depois da transição discreta. Essa relação relaciona o estado antes da transição discreta $\left(q_i\left(\tau_i^{'}\right), x_i\left(\tau_i^{'}\right)\right)$ com os após a transição $\left(q_{i+1}\left(\tau_{i+1}\right), x_{i+1}\left(\tau_{i+1}\right)\right)$.

- Ter uma evolução contínua: para todo i

1) $q_i(.)$: $I_i \rightarrow Q$ é constante em $t \in I_i$, isto é, $q_i(t)=q_i(\tau_i)$ para todo $t \in I_i$.

2) $x_i(.)$:$I_i \rightarrow X$ é a solução da equação diferencial

$$\frac{dx_i}{dt} = f\left(q_i(t), x_i(t)\right) \tag{6.13}$$

no intervalo I_i começando em $x_i(\tau_i)$.

3) Para todo $t \in \left[\tau_i, \tau_i^{'}\right)$, a solução da equação diferencial $x_i(t) \in Dom(q_i(t))$.

Essa restrição mostra o que acontece ao longo de uma evolução contínua e quando a evolução contínua deve admitir um valor constante para o estado discreto.

Uma execução (τ, q, x) é dita:

- Finita: se $\tau = \left\{\left[\tau_i, \tau_i^{'}\right]\right\}_{i=0}^{N}$. é uma sequência finita e o último intervalo $I_{i=N}$ é um intervalo fechado.
- Infinita: se $N \rightarrow \infty$ e a soma dos intervalos em τ é infinita, ou seja,

$$\sum_{i=0}^{N}\left(\tau_i^{'} - \tau_i\right) = \sum_{i=0}^{N} I_i = \infty \tag{6.14}$$

- Zeno: se $N \rightarrow \infty$ mas

$$\sum_{i=0}^{N}\left(\tau_i^{'} - \tau_i\right) = \sum_{i=0}^{N} I_i < \infty \tag{6.15}$$

Neste presente trabalho tem-se uma execução finita, já que o robô para cada estado tem um tempo finito de execução, inclusive o último estado que é quando atinge a porta.

A figura (6.9) mostra alguns exemplos de conjuntos híbridos temporais com execuções finitas, infinitas e Zeno:

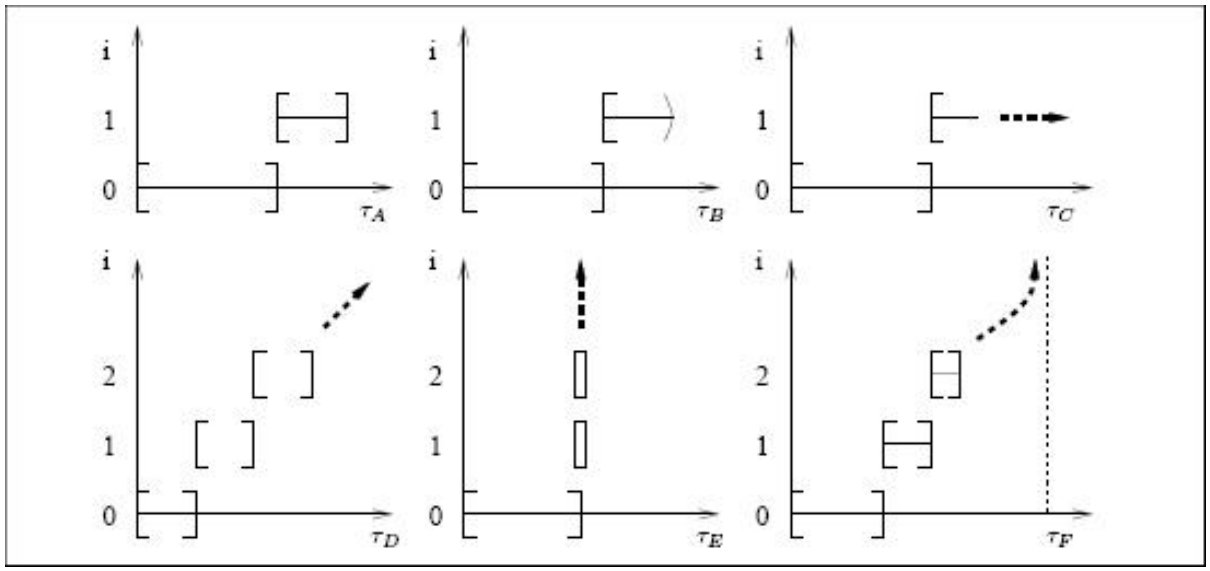


Figura 6.9: A sequência τ_A é uma finita, $\tau_C \ e \ \tau_D$ são infinitas e $\tau_E \ e \ \tau_F$ são do tipo Zeno.

O esquema do tanque da figura (6.5) é uma execução do tipo Zeno, onde cada descontinuidade da curva representa um intervalo de τ_i:

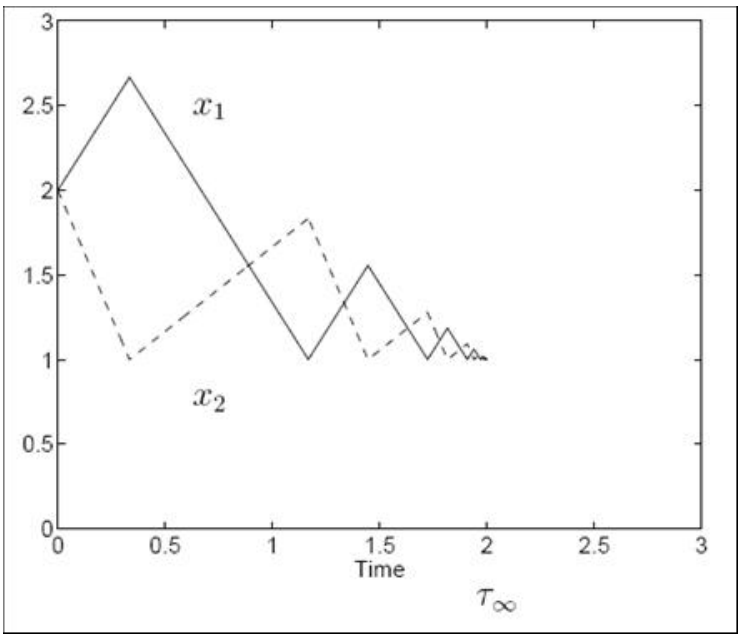


Figura 6.10: Execução Zeno para o sistema híbrido do tanque de água (Johansson, et. al., 1999).

6.5 - Possibilidades de Execução

Em uma grande variedade de fenômenos físicos é aplicada a linguagem dos autômatos híbridos, mas isso pode se tornar perigoso quando não se tem a existência de algumas soluções. Os sistemas que não admitem soluções para alguns estados iniciais, são chamados de sistemas híbridos bloqueadores. Isso é uma propriedade indesejável já que a matemática dá uma visão incompleta da realidade física. As vezes o sistema aceita execuções para todos os estados iniciais, porém não aceita para execuções com tempo infinito. A execução Zeno, por exemplo, tem $N \to \infty$, mas o tempo que limita essas execuções é finito (veja a figura (6.9)). Apesar de na natureza não existirem execuções do tipo Zeno, pode-se modelar sistemas que se aproximam desse tipo de execução.

Outro problema além da ausência de solução, é o fato de poderem existir múltiplas soluções. Sistemas híbridos que aceitam várias execuções para um único estado inicial são conhecidos como não determinísticos.

"Reachability" (Alur, et al., 2000) é um conceito fundamental nos estudos de sistemas híbridos. O estado $(\hat{q}, \hat{x}) \in Q \times X$ de um autômato híbrido H é chamado de "reachable" se o autômato híbrido é definido ao longo de uma execução. Formalmente dizendo: um estado $(\hat{q}, \hat{x}) \in Q \times X$ de um autômato híbrido H é chamado "reachable" se existe uma execução (τ, q, x) do tipo finita, terminando em $(\hat{q}, \hat{x})$, isto é, $\tau = \{[\tau_i, \tau_i']\}_{i=0}^{N}$ com $N < \infty$ e $(q_N(\tau_N'), x_N(\tau_N')) = (\hat{q}, \hat{x})$. Então, pode-se escrever Reach $\subseteq Q \times X$ para nomear todos os estados "reachable" de H. E assim, Init $\subseteq$ Reach.

Outro problema que se deve ter em mente é que nem sempre as execuções são possíveis, pois a evolução pode não ser contínua para algum tempo específico. Isso é chamado de estados de transição. Para $(\hat{q}, \hat{x}) \in Q \times X$ e para algum $\varepsilon > 0$, considere $x(.): [0, \varepsilon) \to \Re^n$ como a solução da equação diferencial:

$$\frac{dx}{dt} = f(\hat{q}, \hat{x}) \; com \; x(0) = \hat{x} \tag{6.16}$$

Se f é uma função contínua de Lipschitz, a solução da equação (6.16) é contínua e única. Os estados $\text{Trans} \subseteq \text{Q} \times \text{X}$ nos quais a evolução contínua é impossível são:

$$\text{Trans}=\left\{(\hat{q},\hat{x}) \in Q \times X \,\middle|\, \forall \varepsilon > 0 \, \exists \, t \in [0,\varepsilon) \; tal\ que \; (\hat{q}, x(t)) \notin Dom(\hat{q})\right\}$$

Isto é, os estados que evoluem continuamente e que são forçados a sair do domínio do sistema são definidos como os estados Trans.

Considere novamente o exemplo dos tanques de água para descrever quais são os estados Reach e os Trans.

Os estados Reach devem conter todos os estados iniciais, desse modo:

$$\{q_1, q_2\} \times \left\{x \in \Re^2 \,\middle|\, (x_1 \geq r_1) \wedge (x_2 \geq r_2)\right\} \subseteq \text{Re}\, ach$$

6.6 – Verificação do Modelo

Trabalhar com sistemas que tenham número de estados finitos é mais fácil para uma investigação, pois se pode analisar todas as possíveis transições. Em sistemas híbridos, falando em geral, essa possibilidade é impossível pois o número de estados é infinito. Porém, há sistemas híbridos que podem ser reduzidos a um número finito de estados. Isso pode ser feito fazendo uma separação do espaço de estados em um número finito de conjuntos. Esse processo de analisar os estados é chamado de verificação do modelo.

A verificação de modelos (*model checking*) é uma técnica que permite a verificação de propriedades de sistemas de modo automático. Esta é uma técnica aplicável aos sistemas computacionais que podem ser modelados por autômatos finitos ou variações deste estilo de representação, e consiste em três passos: a representação de uma aplicação através de um modelo A representado como um sistema de transição de estados finitos (autômatos); a representação de uma propriedade em uma linguagem lógica de especificação de propriedades e, finalmente, a execução de um algoritmo de verificação que responda a questão: "o modelo A satisfaz a propriedade ϕ, ou seja, $\mathsf{A} = \phi$?".

Um sistema de transição $T = (S, \delta, S_0, S_F)$ é definido como:

- Ter um conjunto de estados S, finito ou infinito.
- Ter uma relação de transição $\delta : S \rightarrow P(S)$.
- Ter um conjunto de estados iniciais $S_0 \subseteq S$.
- Ter um conjunto de estados finais $S_F \subseteq S$.

Uma sequência finita ou infinita de estados $\{s_i\}_{i=0}^{N}$ definem uma trajetória, de modo que:

- $s_0 \in S_0$
- $s_{i+1} \in \delta(s_i)$ para i=0,1,...,(N-1).

Veja um exemplo de um sistema de transição com finitos estados mostrado na figura (6.11):

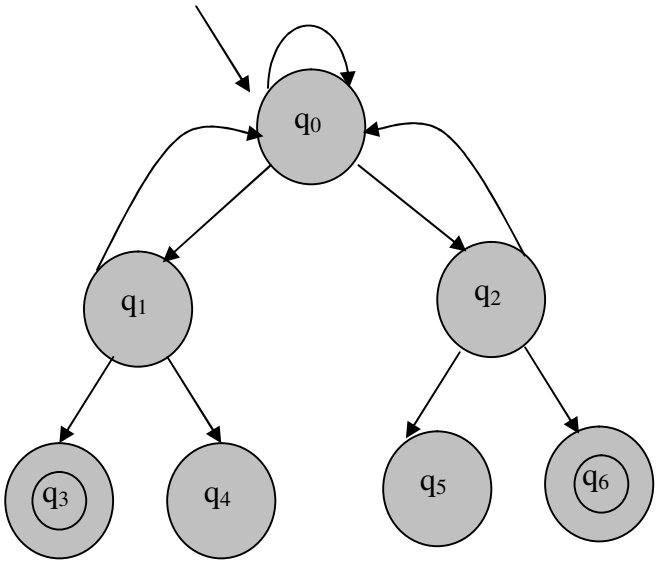


Figura 6.11: Exemplo de um sistema de transição com número finito de estados.

Neste presente estudo é considerado o seguinte sistema de transição de finitos estados (figura (6.12)):

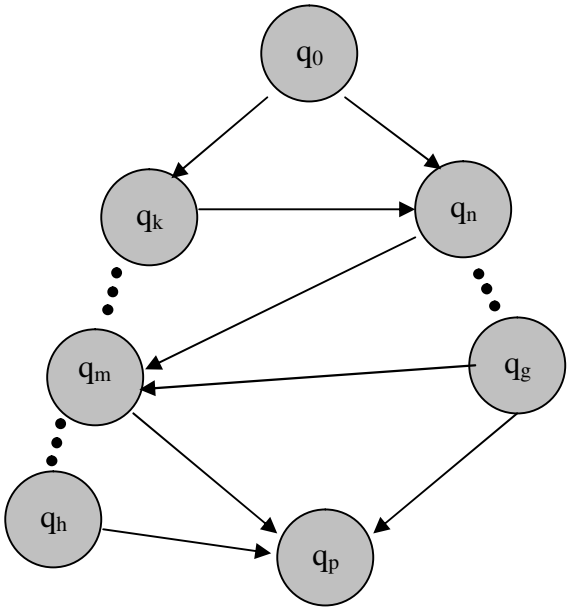


Figura 6.12: Sistema de transição aplicado neste trabalho.

Na figura (6.12), q_0 é o estado inicial da posição do robô, considerado na origem e direcionado para o objetivo. Do lado esquerdo da figura estão representados os estados, ou a pose estimada $\hat{\mathbf{X}}_k$, que são obtidos dado um controle linear. No lado direito, os estados esquematizados referem-se a pose obtida num controle circular.

Ao começar a navegação, o robô atinge um estado q_k ou q_n. O estado q_k está relacionado com a navegação do robô utilizando o controle para seguir uma trajetória retilínea. Já q_n, é o estado que caracteriza o controle que fará o robô navegar por uma trajetória circular, contornando o objeto. Na primeira iteração k, o robô pode visualizar um objeto estando muito a sua frente. Se o robô estiver próximo ao objeto, ou seja, a uma distância dada pela margem de segurança, então, o veículo é forçado a circundar o objeto para evitar uma possível colisão. O robô sai da navegação com controle de trajetória circular quando atinge um ponto na região da intersecção da trajetória circular ideal entre uma reta traçada entre o portal e a origem, posteriormente, passando para o controle retilíneo no estado q_m. A trajetória circular ideal é uma circunferência cujo raio é a soma da margem de segurança com o raio do objeto. E assim o processo se repete até atingir a porta no estado q_p.

A descrição formal do sistema da figura (6.11) é escrita como:

- $S = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$
- $\delta(q_0) = \{q_0, q_1, q_2\}, \delta(q_1) = \{q_0, q_3, q_4\}, \delta(q_2) = \{q_0, q_5, q_6\}, \delta(q_3) = \delta(q_4) = \delta(q_4) = \delta(q_5) = \emptyset$
- $S_0 = \{q_0\}$
- $S_F = \{q_3, q_6\}$ (mostrado em duplo circulo na figura (6.11))

É usado um operador predecessor que toma um conjunto de estados $\hat{S}$ e retorna os estados que atingem $\hat{S}$ em uma transição.

$$\text{Pre} : P(S) \rightarrow P(S)$$

onde cada conjunto de estados $\hat{S} \subseteq S$ é definido como:

$$\text{Pre}(\hat{S}) = \left\{ s \in S \mid \exists\ \hat{s} \in \hat{S}\ \textit{sendo}\ \hat{s} \in \delta(s) \right\}$$

O algoritmo para "Reachability" é assim descrito:

Início: $W_0 = S_F$

Repita

If $W_i \cap S_0 \neq \emptyset$

escreva S_F é "reachable"

fim

$$W_{i+1} = \text{Pre}(W_i) \cup W_i$$

i = i+1

até $W_i = W_{i-1}$

escreva S_F não “reachable”

O algoritmos de “Reachability” pode ser aplicado no exemplo da figura (6.11):

$$Pre(W_0) = \{q_1,q_2\} \quad W_0 = S_F = \{q_3,q_6\} \quad W_1 = Pre(W_0) \cup W_0 = \{q_1, q_2, q_3, q_6\}$$

$$Pre(W_1) = \{q_0, q_1, q_2\} \quad W_1 = \{q_1, q_2, q_3, q_6\} \quad W_2 = Pre(W_1) \cup W_1 = \{q_0, q_1, q_2, q_3, q_6\}$$

O algoritmo pára em W_2, pois $W_2 \cap S_0 = \{q_0\} \neq \emptyset$, e então, é dada a informação S_F é “reachable”.

6.7 – Bissimulação

Considerando que trabalhar com sistemas finitos seja muito mais fácil, então, para tentar fazer uma associação de sistemas infinitos com sistemas finitos, fazendo um agrupamento de estados (conjunto) que exibem comportamento que exibem certas propriedades definidas. Tal agrupamento total é chamado de partição do espaço de estados. Formalmente: a partição é um conjunto de estados, $\{S_i\}_{i \in I}$ com $S_i \subseteq S$ (todos os estados) tal que:

- Se dois conjuntos diferentes são tomados: S_i e Sj, $S_i \cap S_j = \emptyset$.
- A união de todos os conjuntos da partição irá resultar em todo o espaço S, isto é,

$$\bigcup_{i \in I} S_i = S$$

onde I é o índice da partição, que pode ser finito (I=1,2,3,...,M) ou infinito. Se for finita, ou seja, $M < \infty$, a partição é dita finita.

Admita um exemplo e um contra-exemplo de casos sobre partição. Novamente, considere a figura (6.11) e a coleção de conjuntos como:

$$\{q_0\},\{q_1,q_2\},\{q_3,q_6\},\{q_4,q_5\} \text{ e}$$

$$\{q_0\},\{q_1, q_3,q_4\},\{q_2,q_5,q_6\}$$

são considerados partições. Mas,

$$\{q_1, q_3,q_4\},\{q_2,q_5,q_6\} \text{ e}$$

$$\{q_0, q_1, q_3,q_4\},\{q_0, q_2,q_5,q_6\}$$

não são partições. Embora no primeiro caso a intersecção seja nula entre ambos os conjuntos, q_0 não está dentro dessa coleção. E no segundo caso, a intersecção não é nula pois q_0 aparece duas vezes.

A bissimulação de um sistema de transição T = (S,δ,S_0,S_F) é uma partição $\{S_i\}_{i\in I}$ do espaço S, tal que:

- S_0 é um dos conjuntos na partição.
- S_F é um dos conjuntos na partição.
- Se um estado s dentro de um conjunto S_i da partição fizer a transição para outro estado dentro de um conjunto S_j, então todos os estados $\hat{s}$ de S_i devem fazer a transição a todos de S_j. De uma maneira mais formal, para todo i, j $\in$ I e para todos os estados s, ou seja $\hat{s}$, $\hat{s} \in S_i$, se $\delta(s) \cap S_j \neq \emptyset$ então, $\delta(\hat{s}) \cap S_j \neq \emptyset$.

Novamente, voltando a figura (6.11) e considere a partição:

$$\{q_0\},\{q_1,q_2\},\{q_3,q_6\},\{q_4,q_5\}$$

Ela pode ser considerada uma bissimulação do sistema finito por 3 motivos:

- $S_0 = \{q_0\}$ é um dos conjuntos da partição
- $S_F = \{q_3,q_6\}$ é um dos conjuntos da partição
- q_1 e q_2 podem saltar para qualquer um dos elementos de outros conjuntos.

Considerando um contra-exemplo admitindo a partição:

$$\{q_0\},\{q_1,q_3,q_4\},\{q_2,q_5,q_6\}$$

ela não é uma bissimulação pois o conjunto S_F não está presente e o elemento q_1 pode fazer a transição para q_0, enquanto que os outros elementos do conjunto que q_1 pertence, q_3 e q_4 não podem ir até q_0.

6.8 - Proposta de Aplicação do Controle Híbrido ao Projeto Pilgrim

Um trabalho proposto é realizar um segundo nível de controle na trajetória do robô. A estratégia de simulação e controle proposta no capítulo 7, consiste em definir trechos de navegação retilíneos, onde o robô se desloca de uma posição atual para um portal de passagem, em um ambiente composto de vários obstáculos. Na definição dessa trajetória, baseada na análise de câmeras, é possível que o robô encontre obstáculos em seu caminho. Assim, contruindo-se uma região virtual circular em torno de um obstáculo, com uma medida apropriada de raio de segurança, define-se uma estratégia secundária de navegação com trajetória circular, desde o primeiro ponto de interseção da trajetória retilínea do robô com a região virtual de segurança. O segundo ponto de interseção da trajetória do robô com a trajetória circular é o ponto de abandono do nível de controle secundário.

A figura 6.13 ilustra o estabelecimento de duas regiões circulares virtuais, em torno de obstáculos, que se interpõe na trajetória do robô entre a sua posição atual e um portal de passagem.

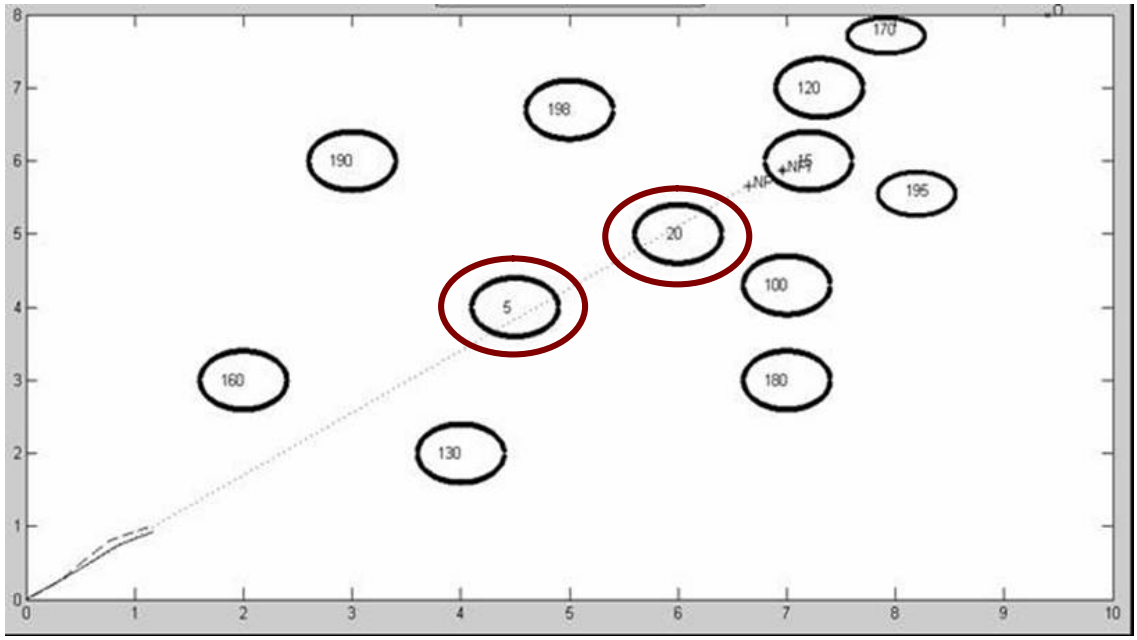


Figura 6.13: Ambiente obstáculos e trajetórias virtuais para controle híbrido.

Capítulo 7

Proposta de um Sistema de Navegação Autônomo

Os dados de odometria e medidas de câmeras formam uma base mínima de sensores que permitem a navegação de um robô de forma satisfatória, em ambientes conhecidos. Nesse capítulo são implementadas simulações numéricas de estratégias de navegação de um robô móvel, utilizando as técnicas descritas anteriormente. A combinação das técnicas de processamento de imagem e de odometria permitem a estimativa da posição ótima do robô. A trajetória com valores ótimos obtidos pelo uso do filtro de Kalman é comparada com o deslocamento real do robô.

7.1 - Simulação 1: Navegação com Dois Pontos de Referência

1. *Controle da trajetória*

As simulações mostradas nessa secção consistem em controlar uma trajetória retilínea do robô que navega em um ambiente onde dois pontos de referência P e A são visíveis pelas câmeras e têm posições conhecidas. A trajetória de referência para o robô seguir é retilínea, partindo da origem (0mm,0mm) até o ponto alvo A. A orientação angular inicial do robô, θ, é a mesma da reta de trajetória.. O percurso do robô é divido em várias iterações k até a chegada ao objetivo final. Para cada iteração é estimada uma nova trajetória retilínea determinada pelo segmento que vai da posição estimada do robô, $\hat{\mathbf{X}}_k$, até o alvo, A.

O controle iterativo é realizado nas rodas do robô, para que ele se desloque de um pequeno trecho retilíneo que parte da posição estimada, em direção do ponto A. A nova posição iterativa,

$\mathbf{X}^*_{k+1}$, dista da posição estimada o equivalente á distância reta percorrida com um número constante *N* de revoluções para cada roda. Isto é, é aplicado um controle cuja representação vetorial ideal **u**=[N;N], faz o robô chegar até o ponto de iteração $\mathbf{X}^*_{k+1}$. O modelo cinemático (capítulo 2) permite determinar o vetor **u** real que fornece o número de revoluções necessárias para o robô partir da posição $\hat{\mathbf{X}}_k$ e chegar na posição $\mathbf{X}^*_{k+1}$. Devido aos erros de odometria, entretanto, o robô não atinge $\mathbf{X}^*_{k+1}$, escapando da trajetória ideal naquela iteração. Assim, o processo se repete no passo *(k+1),* buscando a partir de $\hat{\mathbf{X}}_{k+1}$ uma nova trajetória ideal e um ponto $\mathbf{X}^*_{k+2}$ de caminhada instantânea para o robô.

O procedimento de navegação descrito nessa seção apresenta-se também na forma de um artigo submetido para publicação, no apêndice C.

2. *Processo de Navegação*

O processo de navegação proposto combina informações das leituras de odometria com o processamento das imagens de câmeras, para fornecer estimativas sobre a posição ou estado do robô. As imagens de câmera são utilizadas também para realizar um mapeamento básico do ambiente de navegação. A obtenção de estimativas de posição, baseado em dados de odometria, é um procedimento direto e de baixo esforço computacional, embora esteja associado a imprecisões pela acumulação de erros de leitura. Estimativas de posição, baseado nas imagens de câmeras, por sua vez, possuem um nível mais elevado de esforço computacional, sem o mesmo potencial de acumulação de erros do que no caso das leituras de odometria. Assim, a estratégia de navegação do robô utiliza diferentes sensores para a estimativa de posição via odometria e imagem.

O programa usado para fazer a simulação de navegação do robô utiliza a plataforma Matlab e suas diferentes partes são descritas a seguir.

1 - programa principal *(pilgrim)*

Esse programa contém as chamadas das principais rotinas. Nele são dadas a posição absoluta pontos *P* e *A*, o número básico de rotações das rodas do robô no processo iterativo de caminhada, as variáveis de estado e os desvio padrão assumidos para as incertezas das leituras de odometria e

visão. Contém ainda as variáveis de posição absoluta do robô, $\mathbf{X}_k$, a sua posição a priori definida pela odometria, $\hat{\mathbf{X}}_k^-$, a posição estimada pela visão das câmeras, $\mathbf{Z}_k$ e a posição ótima estimada, $\hat{\mathbf{X}}_k$.

2 – função *Grobo*

Essa rotina possui como entrada o parâmetro de posição angular do robô, e tem na sua saída a matriz $\mathbf{G_{k-1}}$ do modelo cinemático (capítulo 2)

3 – função *comandarobo*

A função *comandarobo* é usada para calcular o vetor de controle **u** necessário para que o robô atinja o ponto intermediário $\mathbf{X}_{k+1}^*$ da trajetória ideal que vai de sua posição atual $\hat{\mathbf{X}}_k$ ao ponto *A*. Possui como entrada o vetor de posição atual do robô.

4 – função *olharobo*

Em *olharobo* são calculadas a posição e orientação do robô baseado no processamento de imagens de câmera de dois pontos da imagem, conforme explicado no capítulo 4. Tem como entrada o vetor de posicionamento atual do robô e como saída o vetor $\mathbf{Z}_k$ de estimativa de posicionamento via câmeras.

5 – função *Kalf*

A função *Kalf* calcula a matriz de covariância do erro de estimação *a priori* $\mathbf{P}_k^-$ e o ganho de Kalman **K**. Esses dois valores de saídas são usados no programa principal "pilgrim" para estimar $\hat{\mathbf{X}}_k$ e a matriz de covariância do erro total de estimação atual, $\mathbf{P}_k$.

7.2 - Simulação 1: Gráficos

A primeira simulação consiste em um ambiente contendo dois pontos *P* e *A*, dispostos no espaço bidimensional com **P=**(6,0 m; 4,0 mm) e **A=**(6,5 m;3 m). O ponto *A* é considerado o alvo

final, isto é, o objetivo. O robô termina sua trajetória quando a distância entre sua posição estimada e o alvo é inferior a 0,1 m. O robô possui a variável de estado inicial descrita por $\mathbf{X}_0$=(0m;0m;0.4324rad). A posição angular inicial é o ângulo entre a origem e o ponto *A*. Os valores iniciais de $\hat{\mathbf{X}}^-$ e $\hat{\mathbf{X}}$ podem ser considerados como $\mathbf{X}_0$, ou como $\mathbf{Z}_0$. Isto equivale a dizer que a posição inicial do robô pode ser estabelecida a partir de seu valor real previamente conhecido, ou a partir de um valor estimado por uma primeira observação das câmeras.

As trajetórias retilíneas iterativas do robô consideram um número básico de *N=18* revoluções em cada roda, isto é, o robô desloca-se o equivalente a metade do perímetro da circunferência da roda, uma vez que N_{res}=36. Pequenas variações aleatórias são introduzidos nos elementos do vetor de controle, de forma a simular os erros não sistemático de odometria existentes no processo de navegação. O filtro de Kalman é usado com a finalidade de regularizar as variações do controle/leitura de odometria e dados de observação das câmeras.

Os valores numéricos usados na simulação são encontrados na tabela 7.1.

Tabela 7.1: Pontos de referência, parâmetros de geometria do robô e valores de erros dos sensores usados na simulação.

Ponto inicial – m	(0, 0)
Ponto de Referência **P** – m	(6;4)
Ponto de Referência **A** (alvo) – m	(6,5 ;3)
Vetor de Estado Inicial ($\mathbf{X}_0$)	$\mathbf{X}_0^T = [0 \quad 0 \quad 0.4324]$
Raio das Rodas (r_d e r_e) – m	0,1
Distância entre as Rodas (b) – m	0,2
Número de Passos do *Encoder* por Revolução (N_{res})	36
Número de Passos do *Encoder* Considerados no Movimento (N)	18
Desvio Padrão para o Erro de Odometria (σ) - m	0.5
Desvio Padrão para o Erro de Câmera ($\overline{\sigma}$) – m	$5*10^{-7}$

Um resumo dos valores de desvio padrão usados nas simulações assim como os valores inicias de posição, são mostradas na tabela 7.2:

Tabela 7.2: Condições da simulação descrita na secção (7.1).

Cond.	Estado Inicial	Erro da Câmera	Erro de Odometria
1	$\mathbf{X}_0$	Erro (5e-7 m)	Erro (0.5) m
2	Estimado pela Câmera: $\mathbf{X}_0=\mathbf{Z}_0$	Erro (5e-7 m)	Erro (0.5) m
3	$\mathbf{X}_0$	2 x Erro (1e-6 m)	2 x Erro (1.0) m
4	Estimado pela Câmera: $\mathbf{X}_0=\mathbf{Z}_0$	2 x Erro (1e-6 m)	2 x Erro (1.0) m
5	$\mathbf{X}_0$	4 x Erro (2e-6 m)	Erro (0.5) m
6	Estimado pela Câmera: $\mathbf{X}_0=\mathbf{Z}_0$	4 x Erro (2e-6 m)	Erro (0.5) m
7	$\mathbf{X}_0$	Erro (5e-7 m)	4 x Erro (2.0) m
8	Estimado pela Câmera: $\mathbf{X}_0=\mathbf{Z}_0$	Erro (5e-7 m)	4 x Erro (2.0) m

- Resultado da simulação da condição 1: O robô tende a seguir o caminho ideal sem grandes desvios, como mostra a figura 7.1.

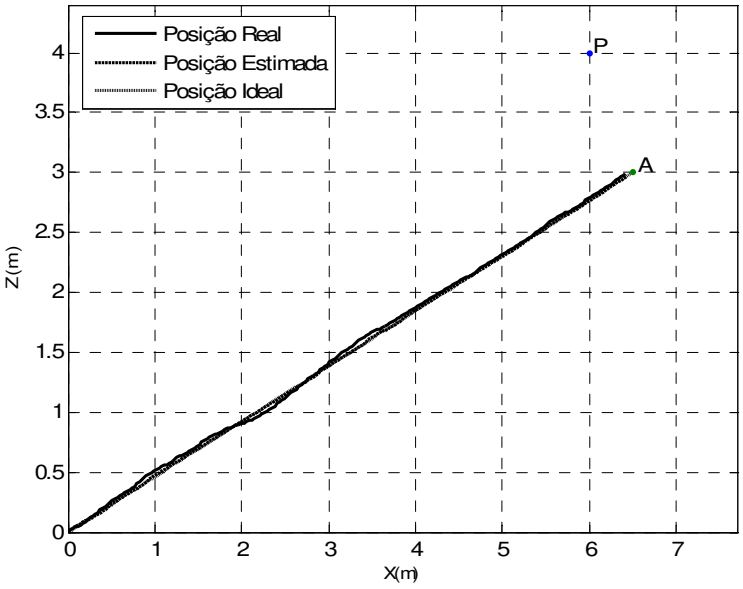


Figura 7.1: Ponto inicial X_0, 1x erro da câmera, 1 x erro de odometria.

- Resultado da simulação da condição 2: o caminho real do robô é influenciado pela estimativa do ponto inicial a partir das câmeras. Observa-se aqui um maior erro de navegação devido a estimativa da posição inicial do robô, feita exclusivamente a partir das imagens das câmeras. O ponto inicial calculado a partir das câmeras, desloca-se da origem.

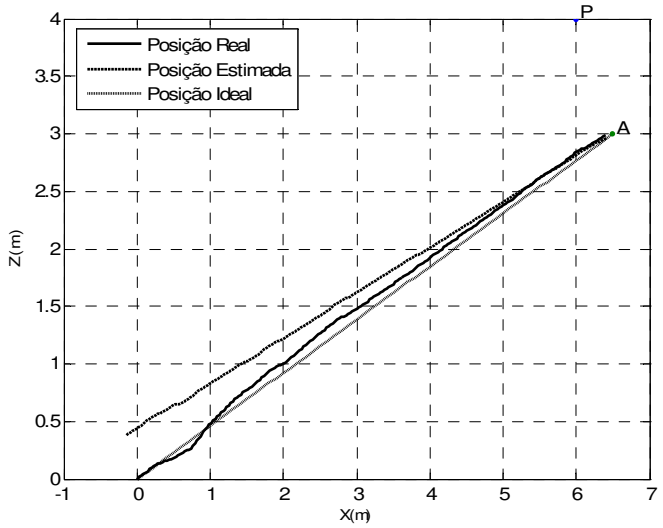


Figura 7.2: Ponto inicial X_0=Z_0, 1x erro da câmera, 1 x erro de odometria.

- Resultado da simulação da condição 3: com um aumento de 100% nos erros de discretização das câmeras e de odometria, pode-se observar pela figura (7.3) que esse aumento não afeta de forma significativa as trajetórias reais e estimadas do robô em relação a ideal.

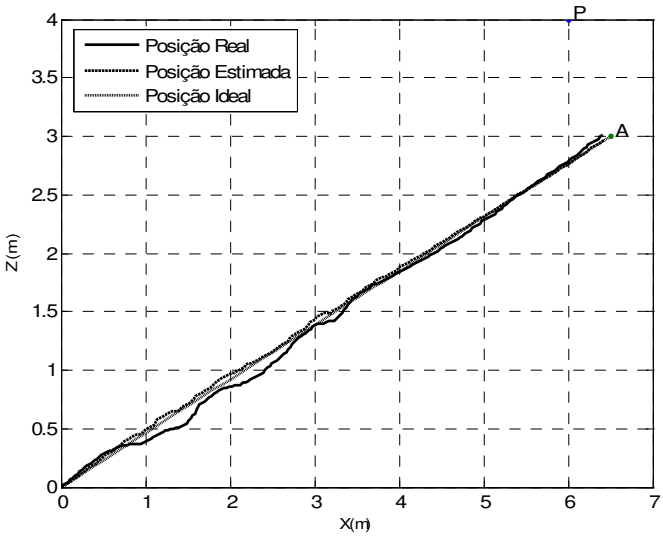


Figura 7.3: Ponto inicial X_0, 2 x erro da câmera, 2 x erro de odometria.

- Resultado da simulação da condição 4: aqui é mostrado o mesmo desvio do caminho real do robô a partir do ideal como visto na condição 2 (figura 7.4).

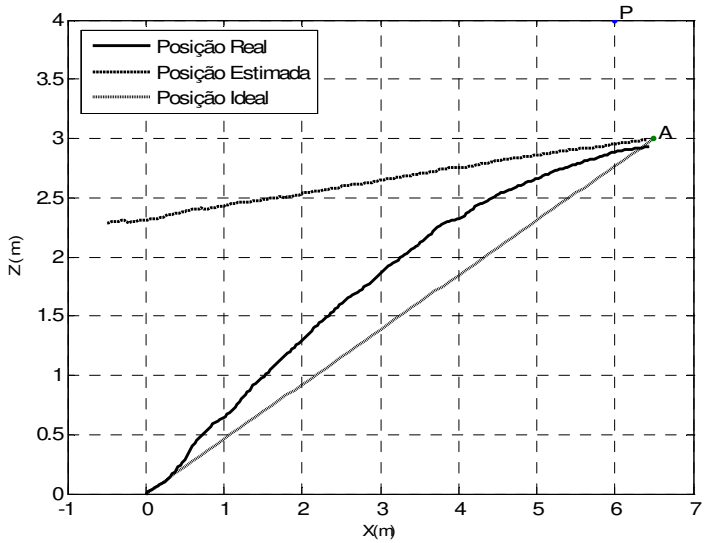


Figura 7.4: Ponto inicial X_0=Z_0, 2 x erro da câmera, 2 x erro de odometria.

• Resultado da simulação da condição 5: esta simulação testa a influência de um erro de câmera cuja magnitude seja muito alta contra um erro de magnitude menor de odometria. O caminho real do robô se desvia suavemente da trajetória, tendendo a real convergir para a ideal quando aquela se aproxima cada vez mais do ponto de chegada, como mostra a figura (7.5).

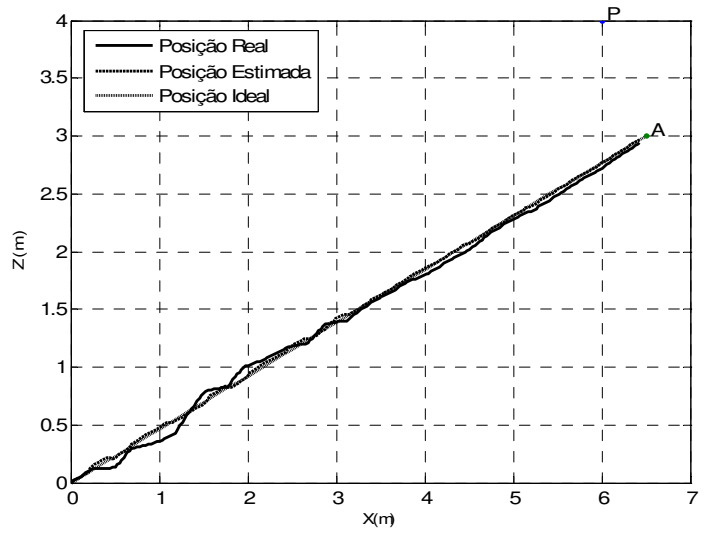


Figura 7.5: Ponto inicial X_0, 4 x erro da câmera, 1 x erro de odometria.

• Resultado da simulação da condição 6: o desvio inicial da trajetória estimada em relação a ideal é devido ao valor inicial da posição estimada ser igual a pose obtida via imagens, isto é,

$\hat{\mathbf{X}}_0 = \mathbf{Z}_0$. Devido ao erro da câmera ser quatro vezes maior, então, o ponto inicial se desvia mais do ponto de partida do que nas condições 2 e 4. Nessa condição 6, o ponto inicial da posição estimada no eixo Z é de aproximadamente 3 m. A figura (7.6) mostra em gráfico esse tipo de comportamento.

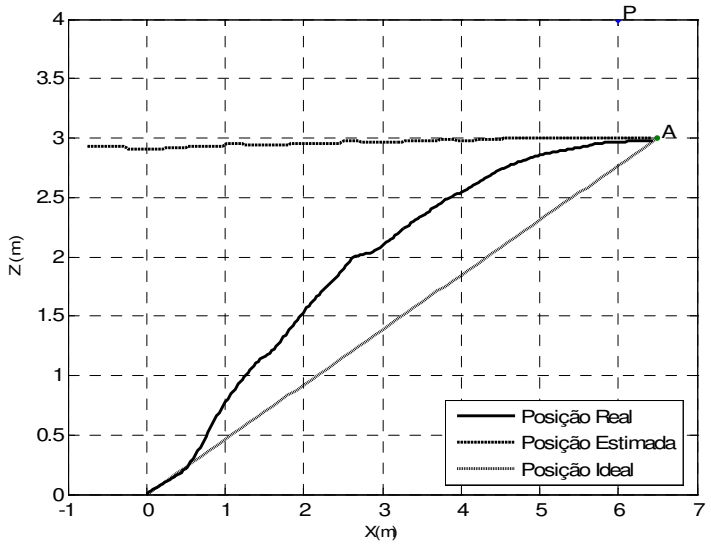


Figura 7.6: Ponto inicial X_0=Z_0, 4 x erro da câmera, 1 x erro de odometria.

Cada vez que o programa se inicia, números aleatórios são inseridos na equação da *pose* real ou em r_i e s_i da equação (3.43). Portanto, para se entender o porquê de um desvio de posição inicial ser consideravelmente grande comparado a origem, toma-se um vetor de erro aleatório ξ_1 que para o caso em que desvio padrão de erros de odometria e câmeras equivalem respectivamente a $\sigma = 0{,}5$ m e $\overline{\sigma}$ = 4*10^{-7}m:

$$\xi_1 = 10^{-5}\ [0{,}1 \quad 0{,}1 \quad 0{,}1 \quad -0{,}2] \tag{7.1}$$

onde os elementos de ξ_1 correspondem respectivamente a um exemplo de erros inseridos nas coordenadas de imagem do ponto A, r_A na câmera *a* e s_A na câmera *b*, e nas coordenadas de imagem do ponto P, r_P da câmera *a* e s_P da câmera *b*.

O robô encontra-se na origem e mirando para o objetivo final.

Sem erros inseridos nas coordenadas de imagem, r_A, s_A, r_P e s_P podem ser representados, nessa ordem, pelos elementos do vetor ψ_1:

$$\psi_1 = 10^{-5}\,[1{,}9 \quad -1{,}9 \quad -40{,}7 \quad -44{,}6] \tag{7.2}$$

Para o caso onde os desvio padrões de erros de odometria e câmeras equivalem respectivamente a $\sigma = 0{,}5$ m e $\overline{\sigma} = 1*10^{-7}$m, o vetor de erro ξ_2, calculado da mesma forma que ξ_1, pode ser representado pelos seguintes números aleatórios provenientes de uma distribuição normal:

$$\xi_2 = 10^{-6}\,[-0{,}4 \quad 0{,}9 \quad -0{,}5 \quad 0{,}00] \tag{7.3}$$

Sem erros inseridos nas coordenadas de imagem, o vetor ψ_2 cujos elementos são representados por r_A, s_A, r_P e s_P, é igual ao vetor ψ_1.

Conclui-se que um erro de quatro vezes no desvio padrão das câmeras, isto é, $\overline{\sigma} = 4*10^{-7}$m, faz com que os números aleatórios do vetor ξ_1 sejam dez vezes maiores do que os erros encontrados no vetor ξ_2, onde se usou $\overline{\sigma} = 1*10^{-7}$m. As vezes, o valor de Z_x se desvia menos da origem do que Z_z, já que na primeira equação de (3.43) a grandeza do erro se distribui entre o numerador e o dominador, praticamente se cancelando, e em Z_z a grandeza do erro está só no denominador, alterando mais o resultado, ou seja, afastando mais Z_z da origem.

- Resultado da simulação da condição 7: Nesta simulação é testado a influência de erro de odometria com alta magnitude, contra um erro de câmera de baixa magnitude. A trajetória do robô quase não se desvia do ideal, como mostra a figura (7.7).

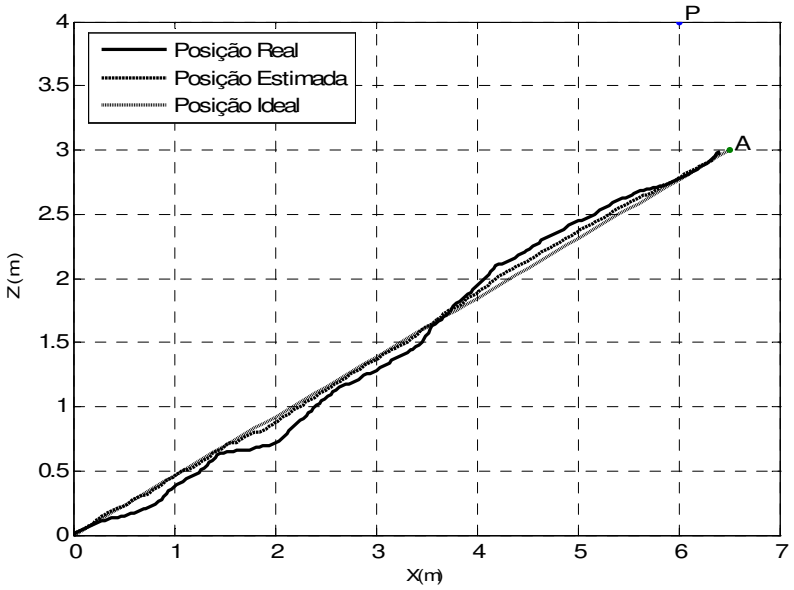


Figura 7.7: Ponto inicial X_0, 1 x erro da câmera, 4 x erro de odometria.

- Resultado da simulação da condição 8: neste caso, o ponto de partida do robô no caso real se desvia do início (0,0) m, devido ao erro da câmera. A partir disso, a trajetória estimada caminha em direção ao ponto de chegada, e quanto mais ela estiver próxima desse ponto, pode-se dizer que as trajetórias estimada e real convergem entre si. Isso ocorre devido a matriz covariância de ruído de medida tender a zero. (figura 7.8).

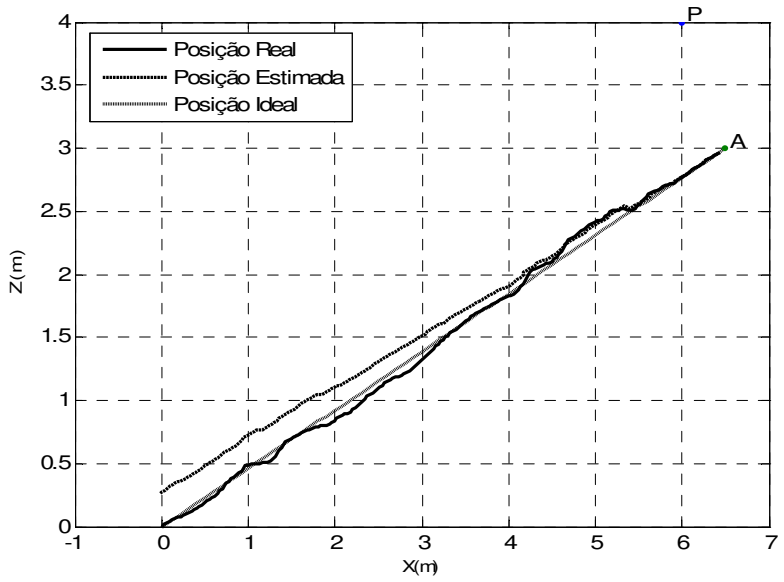


Figura 7.8: Ponto inicial X_0=Z_0, 1 x erro da câmera, 4 x erro de odometria.

7.3 - Simulação 2: Ambiente com circunferências

A segunda simulação envolve um ambiente contendo diversos objetos. Considera-se circunferências com diferentes intensidades luminosas, sendo o objetivo de o robô aproximar-se da circunferência com intensidade mapeada com o valor 250. O detalhe da simulação de projeção dos objetos do ambiente de simulação nas câmeras do robô é descrito no apêndice A.

Diferente da primeira simulação, o objetivo final de navegação do robô pode não estar imediatamente visível, sendo necessário que este se desloque entre os diferentes obstáculos do trajeto. Antes de atingir o objetivo final, o robô se direciona aos diferentes portais identificados nas imagens de câmera. Os portais são os pontos de passagem para onde o robô se direciona através de um percurso retilíneo. Um novo portal é determinado a cada vez que o robô atinge o portal previamente almejado. O objetivo final, ou ponto *O*, é o último portal que o robô deve se aproximar, finalizando toda sua navegação.

A trajetória global do robô, portanto, é composta de diferentes trechos de trajetórias retilíneas nas quais o robô caminha até atingir um ponto final (portal).

Considere a situação do robô no estado inicial $\mathbf{X}_0 = (0,0)$ m e $\hat{\mathbf{X}}_0^- = \mathbf{X}_0$ e $\hat{\mathbf{X}}_0 = \mathbf{X}_0$. O ângulo inicial do estado estimado $\hat{\mathbf{X}}_0(3)$ é o ângulo definido pela reta virtual que vai da origem ao objetivo final *O*. Imagens do ambiente localizado em um ângulo de 180^0 a frente do robô são processadas pelas câmeras *a* e *b*. As intensidades luminosas dos objetos do ambiente bidimensional produzem imagens unidimensionais, caracterizadas aqui por gráficos do tipo degrau, uma vez que os diferentes objetos do ambiente simulado possuem intensidades constantes e cores diferentes. A presença de degraus na imagem unidimensional indica uma interface geométrica entre dois objetos diferentes. Dessa forma, buscam-se as posições onde ocorrem os degraus da câmera *a*, que são comparados com os valores obtidos pela câmera *b*.

Os pontos projetados no mundo são aqueles que possuem a mesma variação de intensidade em ambas as câmeras,. Os objetos, nesse caso, ficam representados por esses pontos que denotam pontos extremos ou de separação dos objetos, conforme vistos pelas câmeras.

Para calcular a posição do robô através das imagens é usado o modelo apresentado no capítulo 4, ou seja, há a necessidade de se encontrar dois pontos na imagem que sirvam de referência ao cálculo da posição estimada por imagem. A cada iteração, busca-se dois novos pontos de referência que devem respeitar duas regras: estar próximos do robô e devem guardar certa distância entre eles.

No primeiro caso tem-se que, quanto mais próximos os pontos das câmeras, menores são os erros de localização. No segundo caso, se os pontos observados pelo robô estiverem muito próximos entre si, o erro de estimativa de posicionamento pode tornar-se igualmente grande.

Quando o robô atinge um portal de passagem ele posiciona-se novamente com a orientação angular de uma reta que vai da sua posição atual até a posição estimada do objetivo final. Nessa condição, as câmeras mapeiam o ambiente a 180^o a sua frente, na busca de um novo portal de passagem ou do objetivo final.

O programa possui as funções adicionais em relação às aquelas anteriormente apresentadas descritas, resumidamente , a seguir.

1 - pilgrim

Esse é o programa principal que contém todas as outras chamadas. Nele estão contidas as informações do ambiente: raios dos círculos, suas respectivas intensidades, os limites do ambiente (localização das paredes), a posição do objetivo final O, as variáveis e os erros de odometria e visão, as chamadas para outras funções e as posições iniciais de $\mathbf{X}_k$, $\hat{\mathbf{X}}_k^-$ e $\hat{\mathbf{X}}_k$, sendo que as posições iniciais a priori e estimada são dadas por $\mathbf{X}_0$.

O ambiente é aqui representado por doze circunferências e toda a trajetória do robô é mostrada em gráficos.

2 – intercir

Nesse programa há uma busca em pontos das circunferências que seriam candidatos a serem visualizados pelas câmeras. É feito a intersecção de todas as circunferências com retas virtuais que saem de cada câmera, as quais varrem a tela do lado esquerdo até o lado direito, no total de 320 vezes, já que neste trabalho é considerado, 320 *pixels.* As circunferências possuem valores de intensidade menores que 225. Valores variando entre 225 e 255 são a intensidades dadas às paredes.

3 – paredeintesection

Assim como o intercir, esse programa busca achar a intersecção das retas virtuais com as paredes. As intensidades das quatro paredes são: 225, 235, 245, 255.

4 – Intensidade

Essa função encontra para cada câmera individualmente, os degraus de intensidade, que correspondem aos pontos de separação entre os objetos e seus respectivos *pixels*.

5 – portal

Na função portal, o robô faz uma varredura de 180^{o} das imagens a sua frente, buscando obter um par de imagens correspondentes as câmeras *a* e *b* para cada ângulo de rotação. São usadas as funções *intercir*, *paredeintersection* e *Intensidade* para encontrar os pontos com intensidade menores que 225 (que não sejam paredes) e projetá-los no referencial do robô. Os pontos de intersecção da reta virtual que se originam nas câmeras que estão mais próximos ao robô são os pontos projetados. Calcula-se então os pontos distantes a certa margem de segurança de cada dupla de objeto no referencial do robô, e em cada combinação possível, toma-se como portal aquele ponto com menor distância entre tal ponto, o robô e O. Se caso a distância entre esse portal e "O" encontrados seja menor que 1,3 m (ambos próximos) ou se não for encontrado pelo menos dois pontos, ou o portal esteja avante no eixo x do objetivo, então, o portal é o objetivo e o programa termina com o robô caminhando em direção ao objetivo. O ponto O no robô é encontrado usando a intensidade de 250 nos mapas de intensidade de ambas as câmeras. O portal aqui definido é chamado de Nova Passagem (NP) e o robô tem sua trajetória ideal baseado nesse ponto, que deverá ser encontrado instantaneamente em cada iteração k. O portal encontrado via imagens em cada instante k é chamado de "Nova Passagem Instantânea", abreviadamente escrito como NPins. Os valores de NP e NPins fazem o papel do ponto A do modelo anterior, e todo o controle do vetor **u** é encontrado entre NPins e $\hat{\mathbf{X}}_k$.

Na função portal há como saída a posição NP, a intensidade entre os dois pontos que originaram o NP e a distância entre NP e O (Teste).

6- portalinstantaneo

A função *portalinstantaneo* busca encontrar, via visão, o ponto NP já encontrado na função portal, através da correspondência de imagens no ângulo real $\mathbf{X}(3)$. São usadas as funções *intercir*, *paredeintersection* e *Intensidade* para encontrar os pontos com intensidade menores que 225 (que não sejam paredes) e projetá-los no referencial do robô.

Já que em *portal* tem-se a intensidade dos dois pontos que formam o NP, então, essa mesma intensidade é buscada nos pontos visualizados por ambas as câmeras. Se não encontrar pelo menos dois pontos projetados ou não se encontrar pontos com intensidades correspondentes a de NP, para ambos os casos, usa-se o NPins anterior. Pode também ocorrer de se encontrar pontos com a mesma intensidade e não serem os pontos procurados, por isso é empregado o uso de "Teste", pois calculando a distância entre os dois pontos do objeto que originaram o portal, pode-se comparar com a distância entre cada dupla de pontos no portal instantâneo, e tomar a distância mais próxima. Por exemplo, se na função "portal", o ponto que define o portal foi obtido por dois objetos de intensidades diferentes I_1 e I_2. Na função "portalinstantaneo", busca-se essas intensidades. Ocorre que cada objeto com uma intensidade I é representado no mapa de intensidade por no máximo dois pontos, onde se formam os degraus. Desse modo, não se sabe qual extremidade do degrau que corresponde a intensidade do objeto que foi usado. Faz-se então uma combinação das possibilidades e busca-se a distância entre dois pontos mais próxima do valor encontrado em "Teste".

7 – função DoisPontos

A posição do robô $\mathbf{Z}_k$ só pode ser encontrada se houver dois pontos visualizados pelas câmeras que podem ser encontrados usando as funções *intercir*, *paredeintersection* e *Intensidade* que dão os pontos com intensidade menores que 225 (que não sejam paredes) e são projetados no referencial do robô. Os dois pontos que vão servir de referência, similar a P e A do modelo anterior, são os pontos mais próximos do robô e que tenham uma dada separação entre eles. Quanto mais próximos os pontos das câmeras, menores são os erros, pois os erros dependem de z_{ca} e isso influencia na matriz $\overline{\mathbf{R}}$ e, dessa forma, a medida de visão se torna mais confiável para o filtro.

O fluxograma da figura 7.9 descreve resumidamente o comportamento do robô durante a navegação num ambiente bidimensional.

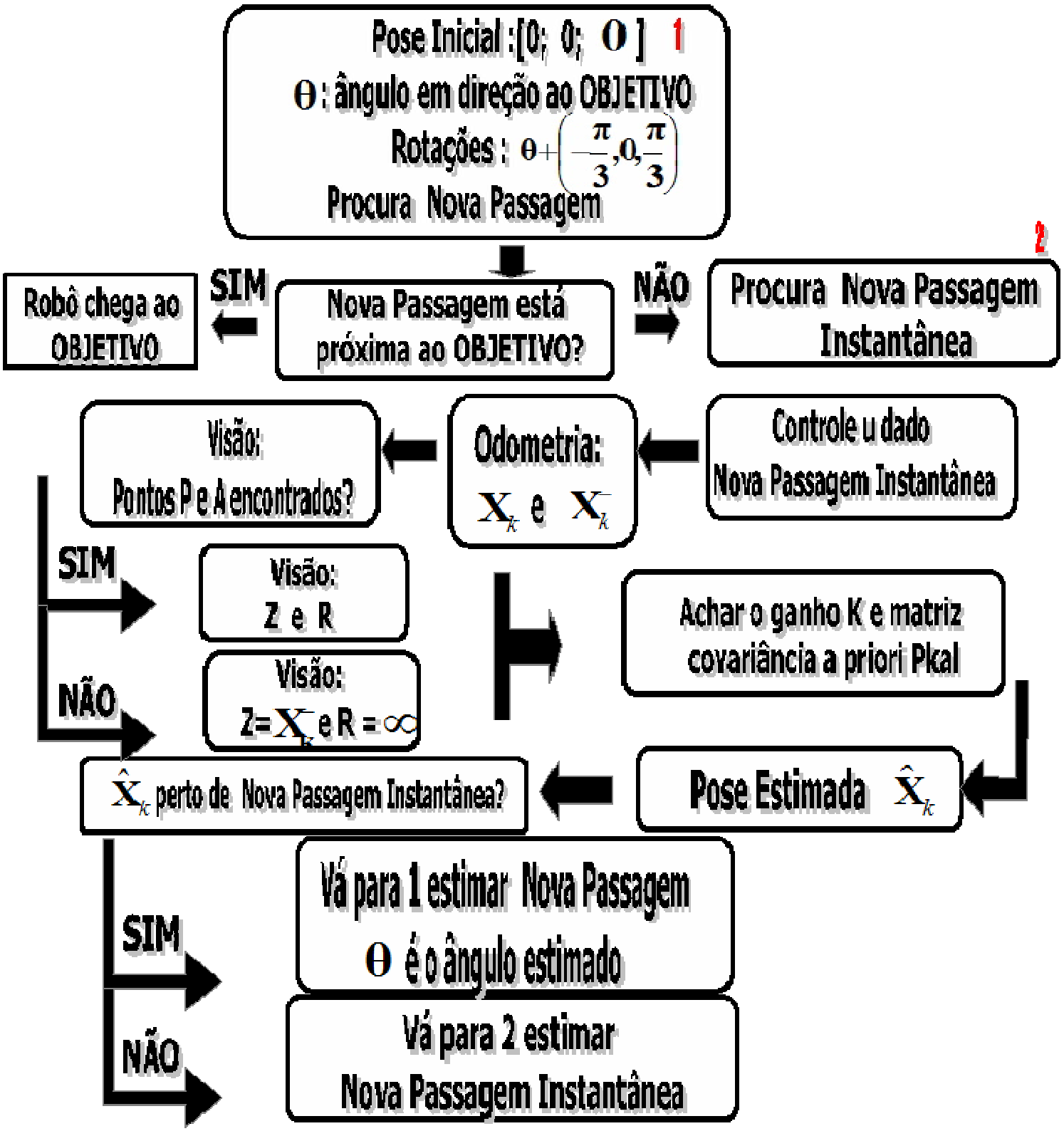


Figura 7.9: Fluxograma de controle da trajetória de um robô móvel, descrito em 7.3

7.4 – Simulação 2: Gráficos

O procedimento descrito na secção 7.3 é apresentado aqui através de simulação do robô percorrendo um ambiente que contem doze circunferências em seu interior. O ambiente de navegação é retangular, limitado por quatro paredes. O processo de navegação consiste em determinar os portais de passagem e dois pontos da imagem que servem de referência para localização do robô. Os valores usados nessa simulação estão apresentados nas tabelas 7.3 e 7.4.

Tabela 7.3: Pontos de referência, parâmetros de geometria do robô e valores de erros dos sensores usados na simulação da secção (7.3).

Ponto inicial – m	(0, 0)
Objetivo – m	(8,5 ;8)
Vetor de Estado Inicial ($\mathbf{X}_0$)	$\mathbf{X}_0^T = [0 \quad 0 \quad 0.4324]$
Raio das Rodas (r_d e r_e) – m	0,1
Distância entre as Rodas (b) – m	0,2
Número de Passos do *Encoder* por Revolução (N_{res})	36
Número de Passos do *Encoder* Considerados no Movimento (N)	100
Desvio Padrão para o Erro de Odometria (σ) - m	0,5
Desvio Padrão para o Erro de Câmera ($\overline{\sigma}$) – m	$5*10^{-7}$

Tabela 7.4: Condições da simulação descrita na secção (7.3).

Condições de Simulação	**Estado Inicial**	**Erro da Câmera**	**Erro de Odometria**
1 - figuras (7.10 - 7.12)	$\mathbf{X}_0$	Erro= $5*10^{-7}$ m	Erro = 0,5 m
2 - figuras (7.13 - 7.16)	$\mathbf{X}_0$	Erro= $5*10^{-7}$ m	20 x Erro=10 m
3 - figuras (7.17 - 7.20)	$\mathbf{X}_0$	12 x Erro = $6*10^{-6}$ m	Erro = 0,5 m
4 - figura (7.21)	$\mathbf{X}_0$	20 x Erro= $1*10^{-5}$ m	20 x Erro=10 m

As figuras (7.10-7.21) mostram o comportamento do robô durante a navegação num ambiente como o descrito anteriormente. Gráficos da trajetória ideal, da posição real e da posição estimada são traçados, para efeito de comparação. A navegação é considerada adequada quando os três gráficos.são próximos entre si. As trajetórias ideais são dadas em linhas pontilhadas finas, a posição real se encontra em linha preenchida e as linha tracejadas representa a trajetória estimada. A linha pontilhada indica a trajetória ideal, a sólida indica a real e a tracejada indica a trajetória estimada via filtro de Kalman.

Simulação na Condição 1

Na figura (7.10) é mostrado a primeira trajetória retilínea ideal, construída a partir da origem e do primeiro portal de passagem encontrado NP. Na medida em que o robô se aproxima do portal, a posição atualizada do portal, NPins, é calculada com uma precisão cada vez maior. Ao atingir o portal uma nova trajetória ideal é estabelecida para o robô e o processo se repete até a chegada do robô ao objetivo.

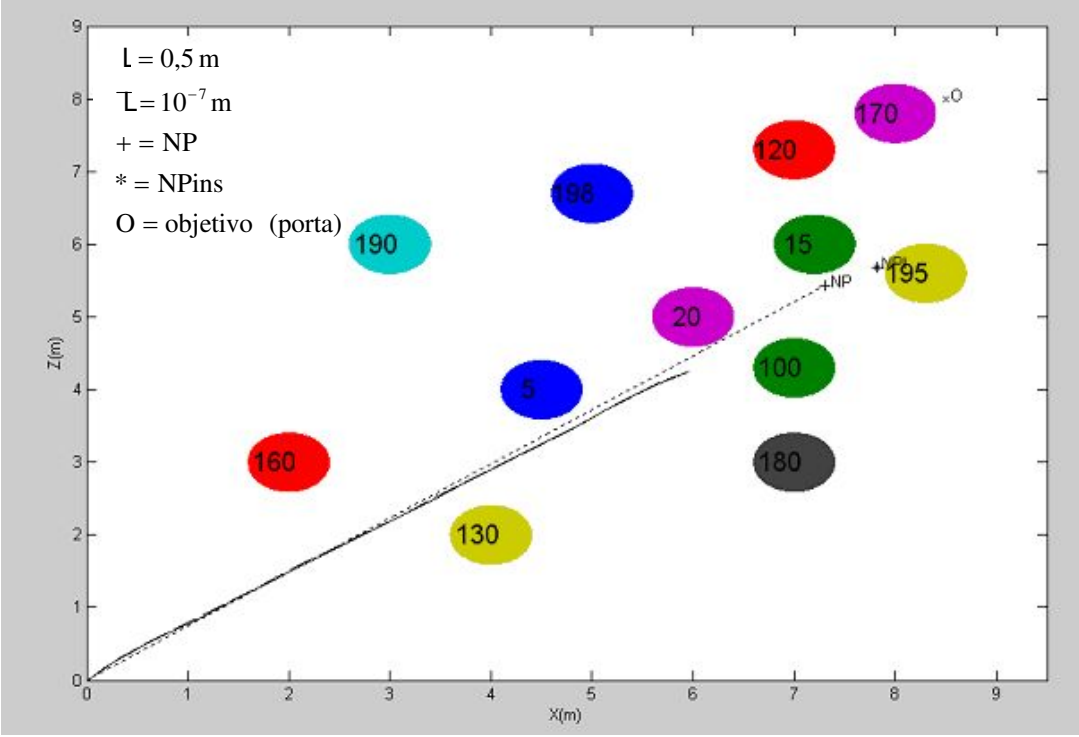


Figura 7.10: Robô navegando em direção a NPins, obtido através do primeiro portal NP.

O robô por passar dentro de um limite entre sua posição estimada e NPins, ou seja, ao se aproximar de um limite estipulado e passar próximo a NPins da, ele realiza novamente as três rotações. O ângulo para iniciar essa rotação é o valor que está em sua *pose* estimada, e não o ângulo em direção a porta. Na fig. 7.11, a nova passagem e nova passagem instantânea coincidem com o objetivo. Provavelmente a nova passagem é o objetivo porque o robô ou deve ter encontrado um novo portal próximo ao objetivo, ou não encontrou pelo menos dois pontos para calcular o novo portal, e nessas condições o programa manda o robô ir para o objetivo.

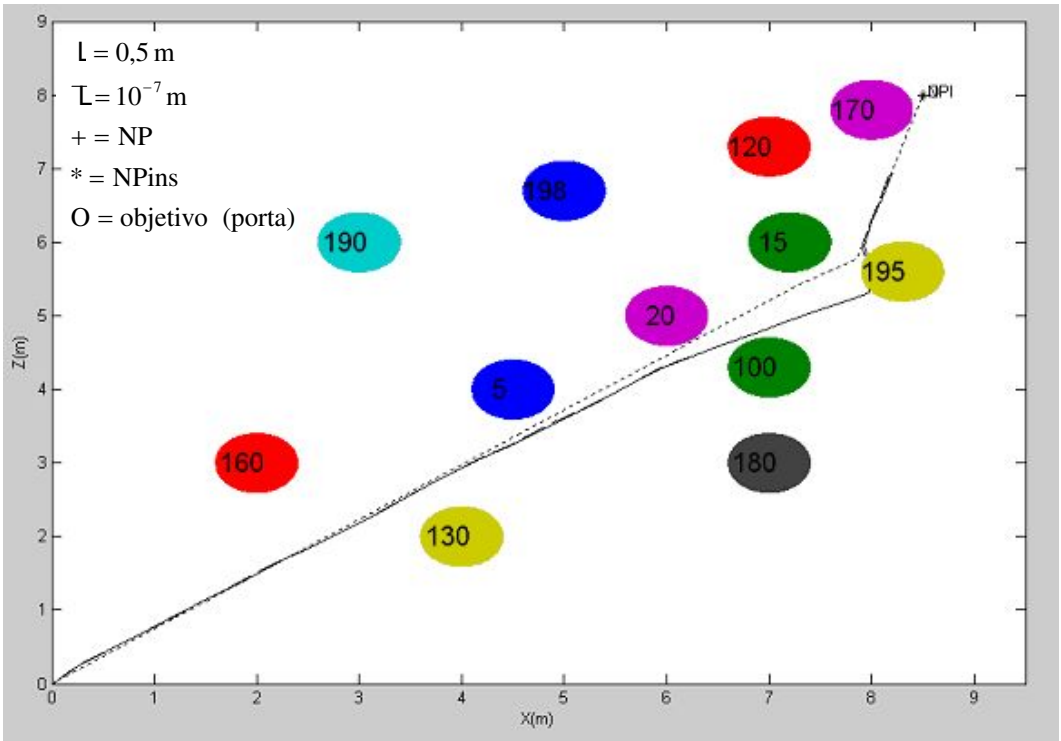


Figura 7.11: Robô navegando em direção a NPins, obtido através do segundo portal NP.

.

A figura 7.12 mostra todo o percurso de navegação do robô, desde a origem (0; 0) m até a porta, o seu objetivo final. O percurso é dividido em duas trajetórias, onde se calculou o portal duas vezes, sendo que no último percurso, o portal é o objetivo. Pode-se ver que durante toda a navegação as trajetórias reais e estimadas são bem próximas, validando o método apresentado (fusão sensorial aplicada filtro de Kalman).

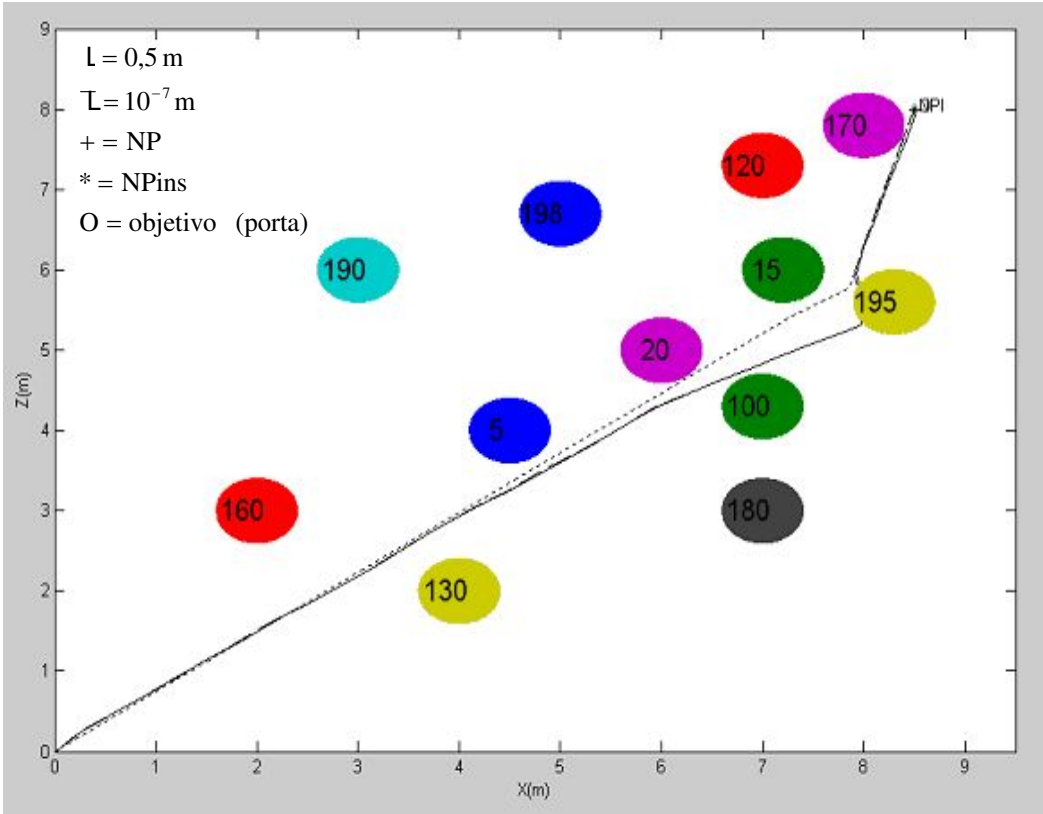


Figura 7.12: O robô chega no objetivo final, uma porta.

Simulação na Condição 2

Nas figuras (7.13-7.16), o valor do desvio padrão de odometria sofre um acréscimo, sendo σ igual a 20*0.5 m. A determinação dos portais NP e NPins é feita da mesma maneira descrita anteriormente. Os pequenos desvios da trajetórias real em relação a estimada se dão pelo aumento no valor do erro de odometria. Isso faz a posição estimada se desviar da real, pois a posição estimada depende do erro de odometria.

Na figura 7.13 é mostrado a primeira trajetória de navegação do robô, onde o NP mostrado foi calculado na origem, e o NPins no gráfico refere-se ao calor calculado na posição estimada na iteração atual. O robô vai se deslocando para NPins conforme vai navegando no ambiente.

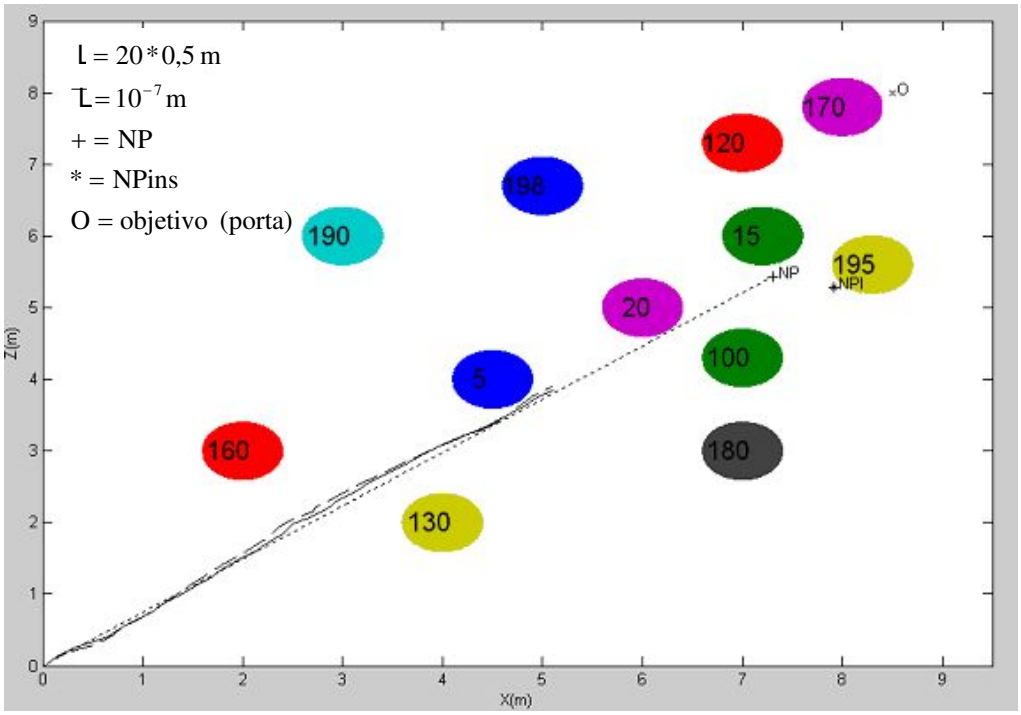


Figura 7.13: Robô navegando em direção a NPins, obtido através do primeiro portal NP.

Na figura 7.14 o robô vai chegando próximo a NPins, e quando se aproximar mais um pouco, ele realizará três rotações para buscar informação do ambiente via câmeras e encontrar o seu novo portal.

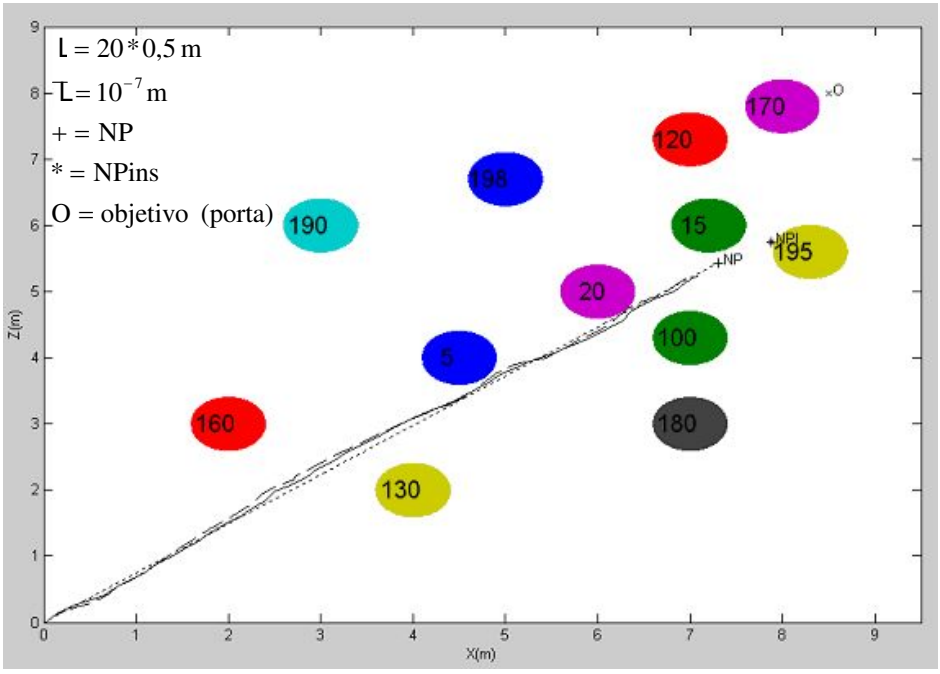


Figura 7.14: Robô próximo ao seu primeiro portal NPins.

As figuras 7.15 e 7.16 mostram o mesmo processo descritos nas figuras 7.11 e 7.12, respectivamente.

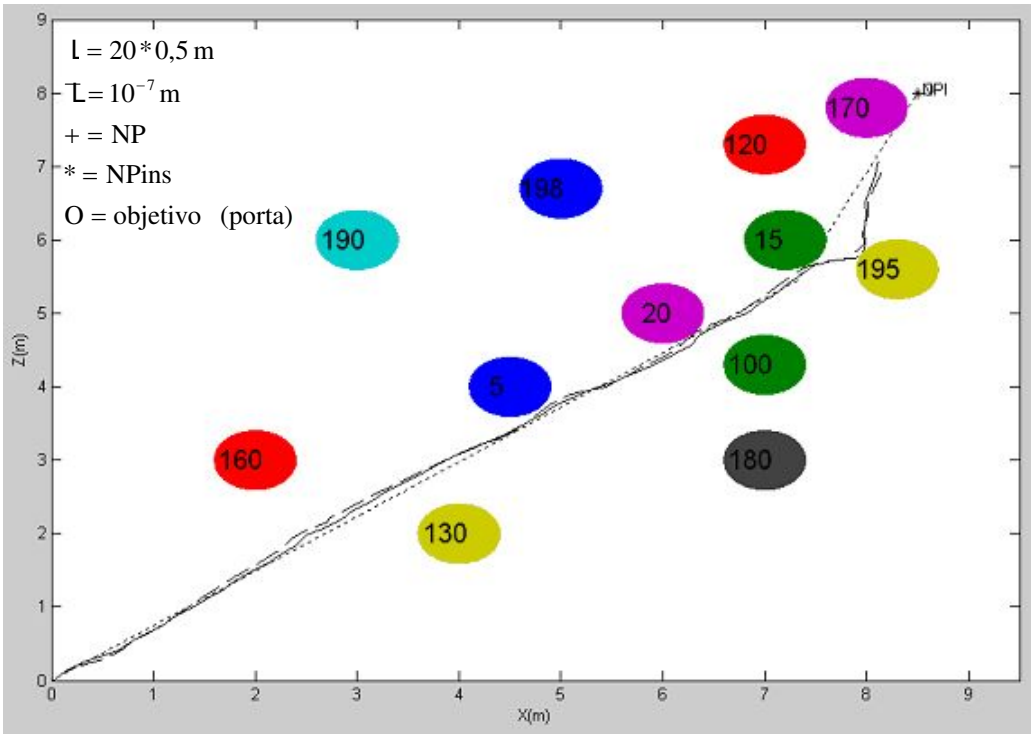


Figura 7.15: Robô navegando em direção a NPins, obtido através do segundo portal NP.

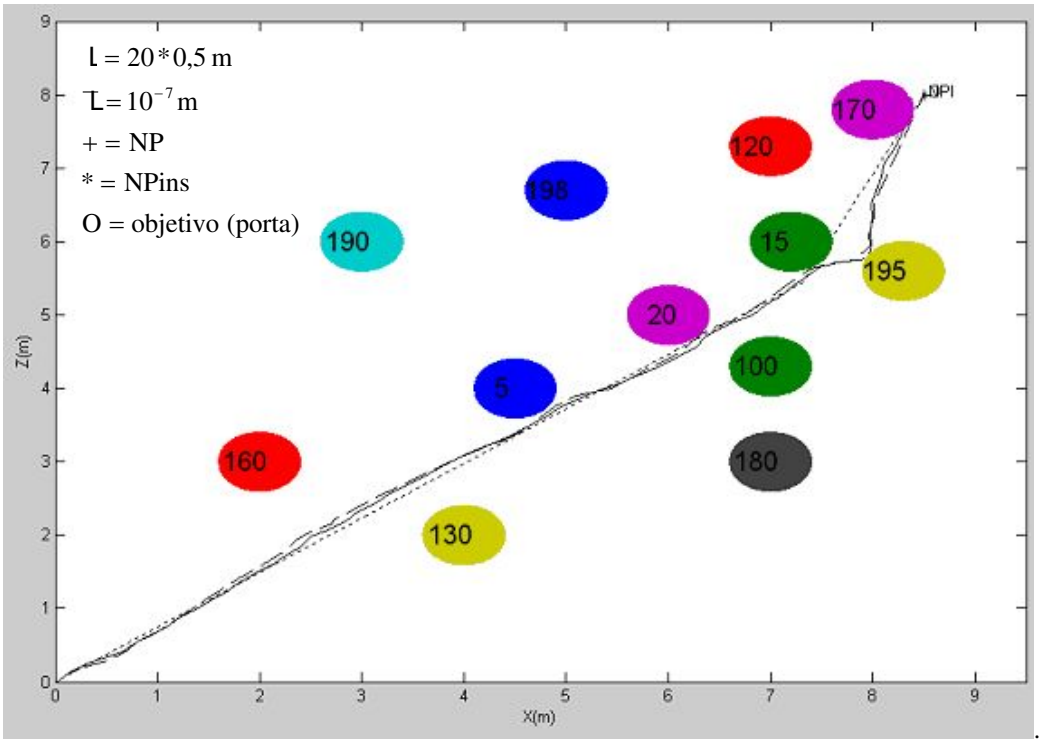


Figura 7.16: O robô chega no objetivo final, uma porta.

Simulação na Condição 3

Nas figuras (7.17-7.20) é introduzido um desvio padrão do erro de câmera $\overline{\sigma}$ igual a $12*5*10^{-7}$ m. Os desvios entre as trajetórias real e estimada são devido aos maiores erro da câmera. Dessa maneira, o robô se desvia da trajetória ideal, principalmente ao início de cada trecho percorrido.

A figura 7.17 segue o mesmo princípio da figura 7.13. O desvio entre as trajetórias reais e estimadas são devido ao aumento de desvio padrão.

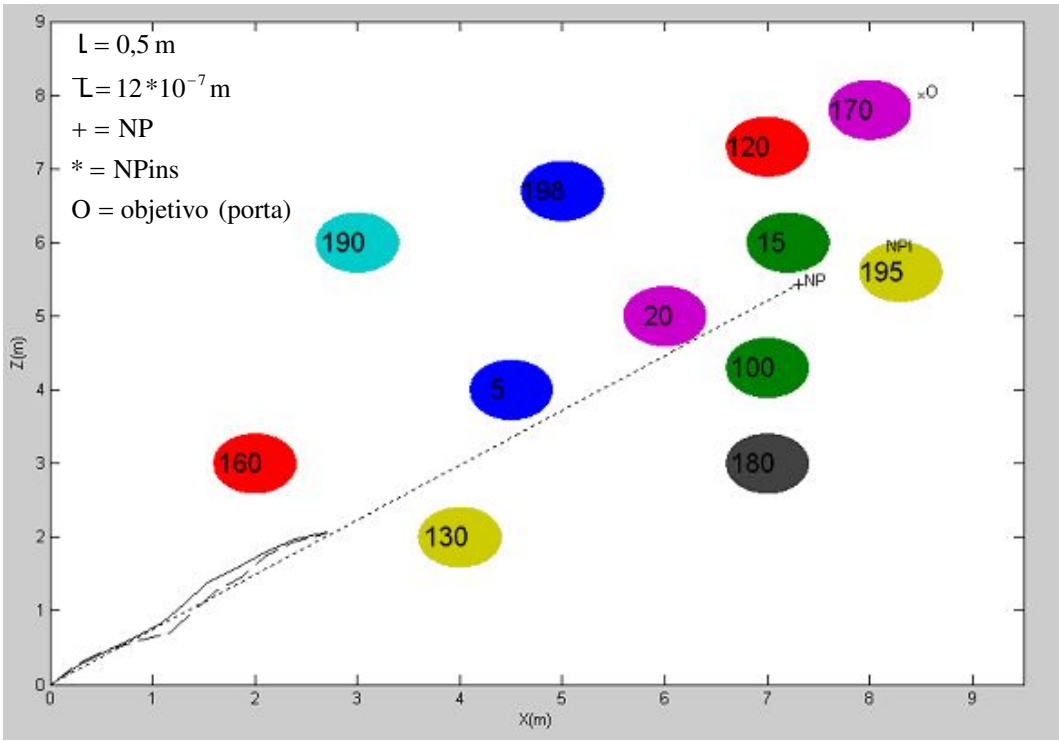


Figura 7.17: Robô navegando em direção a NPins, obtido através do primeiro portal NP.

As figuras 7.18, 7.19 e 7.20, seguem respectivamente a explicação das figuras 7.14, 7.11 e 7.12.

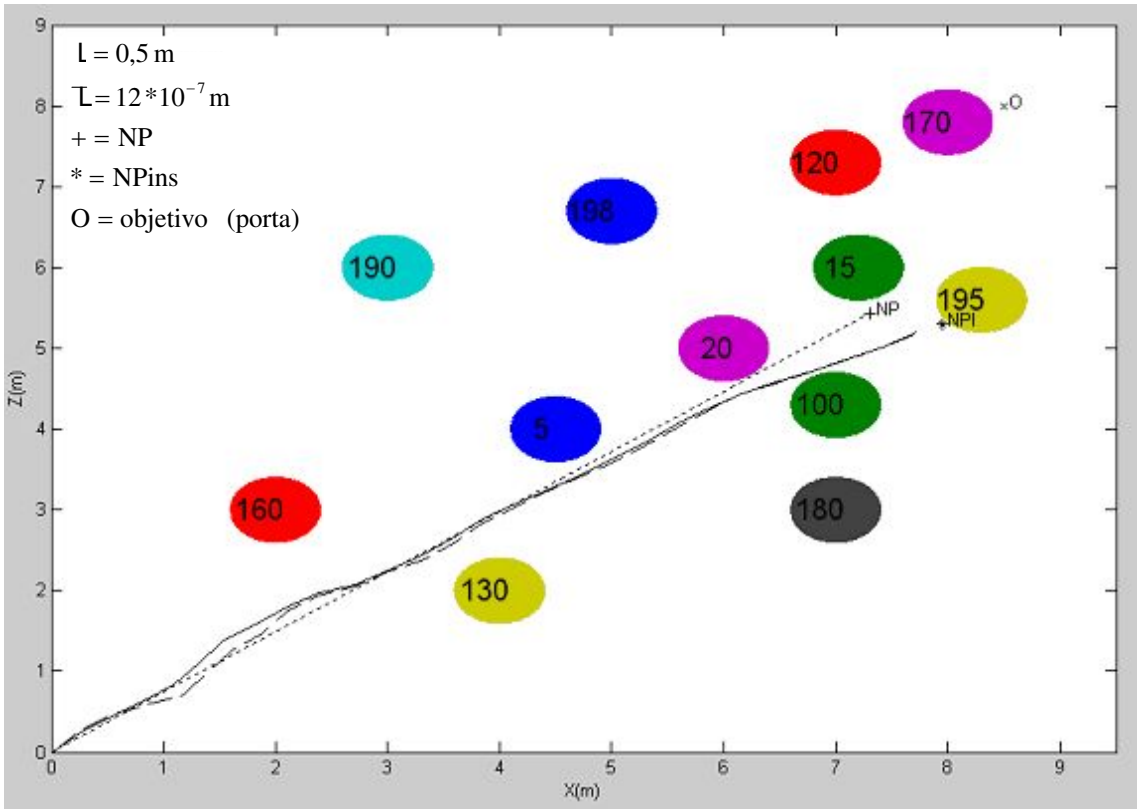


Figura 7.18: Robô próximo ao seu primeiro portal NPins.

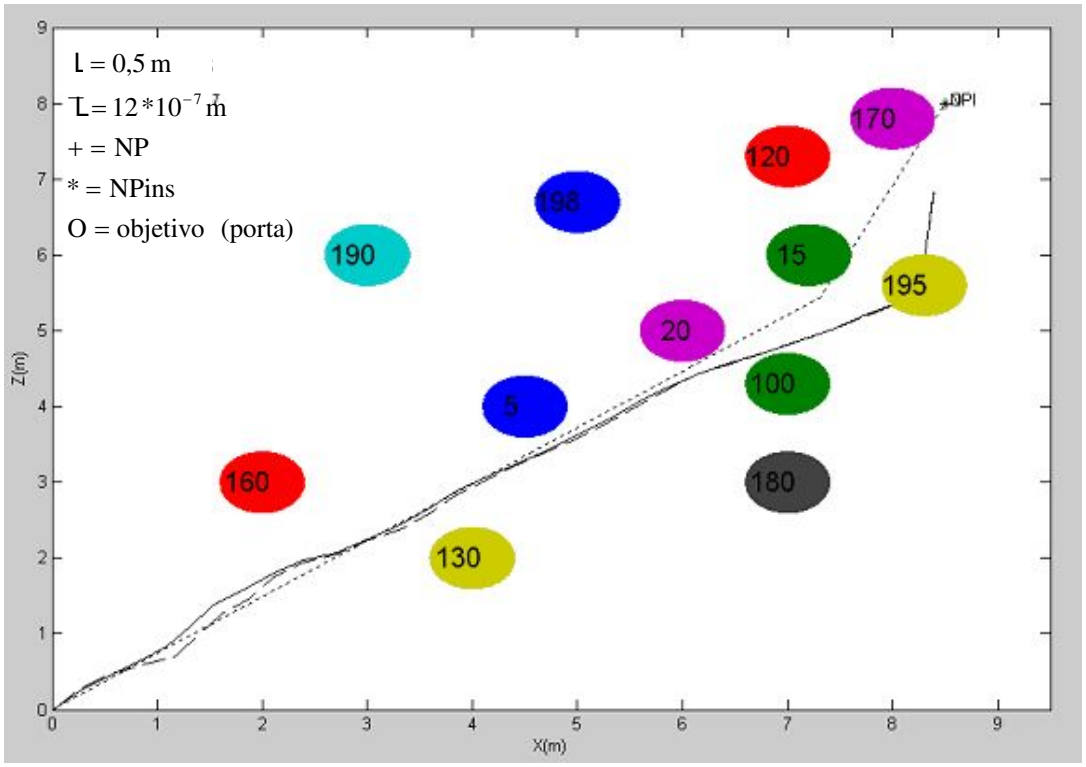


Figura 7.19: Robô navegando em direção a NPins, obtido através do segundo portal NP.

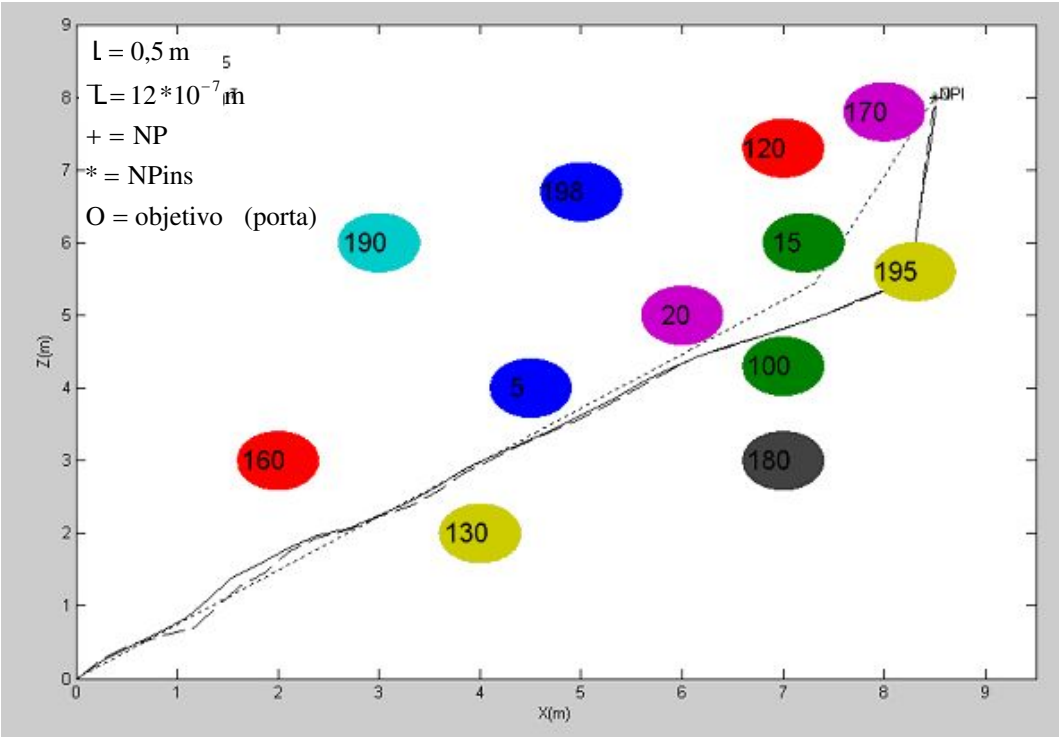


Figura 7.20: O robô chega no objetivo final, uma porta.

.

Simulação na Condição 4

Um caso particular está apresentado na figura (7.21), onde é mostrado toda a trajetória do robô, passando por um sistema de duas trajetórias pré-determinadas. Próximo do objeto com intensidade 195 deve estar localizado o primeiro NPins (considere que seja o último portal instantâneo na primeira trajetória retilínea) e portanto, ao atingir esse ponto, ele realiza três rotações para encontrar o seu novo portal NP, que é o objetivo. Para essa simulação é usado um desvio padrão de erro de odometria σ igual a 20*0.5 m e desvio padrão do erro de câmera $\overline{\sigma}$ igual a $20*5*10^{-7}$ m. Pode-se observar a ocorrência de maiores desvios entre as trajetórias real e estimada. Quando o robô aproxima-se do primeiro ou do segundo NPins (último portal instantâneo da segunda trajetória retilínea), as trajetórias convergem, justamente porque a matriz de covariância $\overline{\mathbf{R}}$ tem seus elementos próximos a 0 e a posição medida $\mathbf{Z_k}$ tende ao valor estimado.

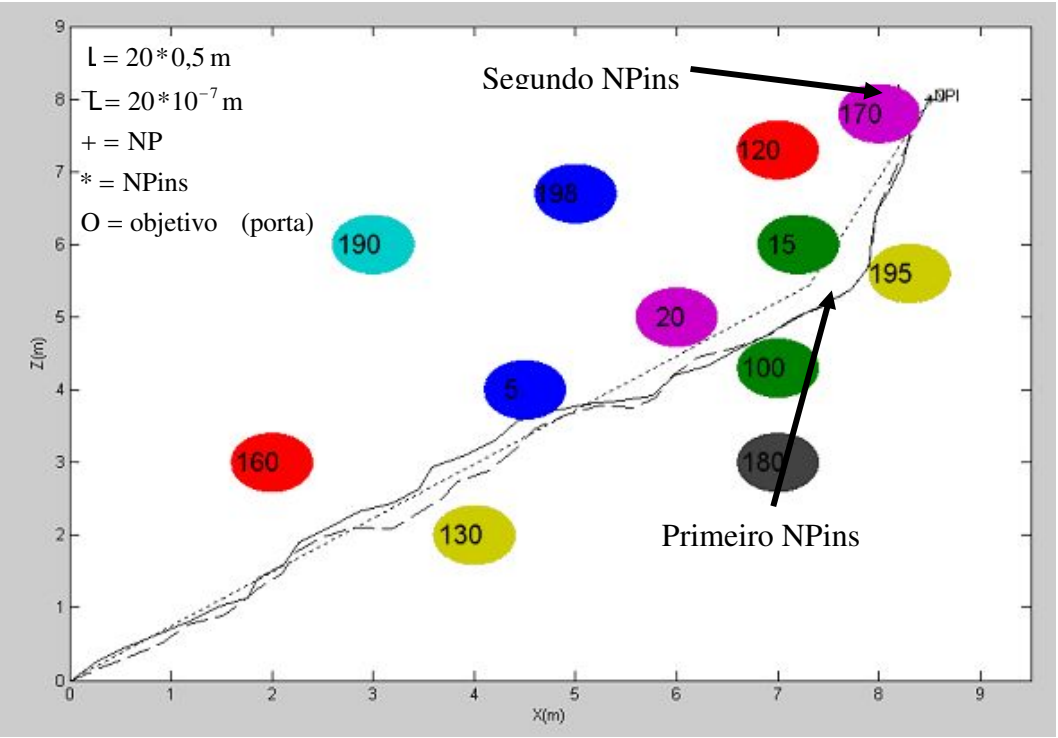


Figura 7.21: O robô chega no objetivo final, uma porta.

.

7.5 - Problemas de Navegação

Durante a navegação, alguns problemas poderiam ocorrer por algoritmos não implementados ao programa principal *pilgrim,* já apresentado nesse trabalho. O problema principal que será discutido nessa secção é a ocorrência de *looping*, ocasionado por um fator: o objetivo sempre ser superior ao limite estipulado.

O objetivo final em todas as simulações da secção 7.4 foi colocado em uma posição estratégica, (8,5 ;8) m. Para outras posições o robô pode entrar em *looping,* isto é, o robô pode estar sempre em processo de navegação. Considere a figura 7.21 e suponha um objetivo que esteja entre os objetos de intensidade 20 e 100, como por exemplo a posição (6,5; 4,5) m. O robô tentará ir para o objetivo sempre, mas ele pode sempre observar pelo menos dois pontos para encontrar o seu novo portal de passagem, mesmo esse portal estando a frente do objetivo, já que haveria objetos depois dele que seriam visualizados, como os objetos de intensidade 15, 195, 120 e 170. Dessa maneira, se a nova passagem e o objetivo sempre tiverem uma distância maior que o limite estipulado, então o robô permanecerá em navegação e entrará em processo de *looping*.

Capítulo 8

Conclusões e Sugestões para Trabalhos Futuros

8.1- Conclusões

Uma estratégia de movimento de robô autônomo foi descrita neste trabalho usando uma combinação de visão computacional e odometria. O veículo navega de um ponto inicial, caminha em uma ambiente contendo obstáculos, até chegar a um alvo final.

O ambiente é descrito em termos de um espaço bidimensional e o processamento de imagens é feito de forma simplificada.

Foi apresentado o funcionamento do filtro de Kalman que é utilizado como ferramenta para estimar a posição do robô. Esse tipo de filtro combina a informação disponível nos sensores (*encoders* e câmeras) para a realização de estimativas do estado de movimento do robô.

As simulações apresentadas demonstram a adequação do emprego das técnicas de navegação selecionadas.

Os desenvolvimentos teóricos mostrados, bem como resultados de simulação apresentados nesse trabalho, constituem uma plataforma básica para o controle de navegação de robôs autônomos que utilizam imagem de câmeras. Tal plataforma deverá ser aperfeiçoada e expandida, de acordo com as sugestões apresentadas a seguir.

8.2 - Sugestões para Trabalhos Futuros

Um primeiro aperfeiçoamento do processo de navegação consiste na introdução do controle híbrido, descrito no capítulo 6, com a finalidade de contornar obstáculos encontrados pelo robô em seu processo de caminhada. A estratégia atual de utilização de um percurso retilíneo do robô, de sua posição atual até um portal de passagem, pode induzi-lo a uma rota de colisão ou aproximação extrema de objetos no entorno de sua trajetória. Nesses casos, adicionalmente ao uso das câmeras para o posicionamento correto do robô, as mesmas podem ser usadas para a localização de obstáculos imediatos e acionamento de um algoritmo com um adequado procedimento de contorno de obstáculos.

Uma segunda aplicação imediata consiste na implementação prática das técnicas aqui descritas. Um modelo de robô de pequeno porte pode ser construído, ou aproveitado de pesquisas anteriores, onde se possa instalar um par de câmeras de baixo custo, com alinhamento paralelo. As imagens bidimensionais das câmeras podem ser processadas ao longo de uma única linha, formando o mesmo tipo linear de mapa de intensidades descrito nesse trabalho, e possibilitando a aplicação dos procedimentos aqui simulados.

Uma aplicação futura consiste em aperfeiçoar o procedimento de mapeamento 3-D a partir dos mapas de intensidade fornecidos pelas câmeras. Nesse processo é possível também utilizar-se câmeras não paralelas, ou mesmo uma única câmera, levando-se em conta o deslocamento do próprio robô na aquisição de imagens do ambiente sob diferentes ângulos.

Outra aplicação futura consiste no desenvolvimento e utilização de diferentes métodos de regularização da trajetória de navegação a partir da integração dos sinais oriundos da odometria e da imagem das câmeras.

Referências Bibliográficas

Alur, R., Henzinger, T. A., Lafferriere, G. e Pappas, G. J. Discrete abstraction of hybrid systems, Proceedings of the IEEE, vol. 88, nº.7, pp. 971-984, 2000.

Banerjee, S. Camera models and affine multiple views geometry, Dept. Computer Science and Engineering, ITT Delhi, 2001.

Bezerra, C. G. Localização de um robô móvel usando odometria e marcos naturais, Tese de Mestrado, Depto. Eng. Elétrica, Universidade Federal do Rio Grande do Norte, Brasil 2004.

Bianchi, R. A. C., Simões, A. S. e Costa, A. H. R. Comportamentos reativos para seguir pistas em um robô móvel guiado por visão, Anais SBAI'2001 – Simpósio Brasileiro de Automação Inteligente, Canela, RS, 2001.

Borenstein, J., Everett, H. R., Feng, L. e Wehe, D. Mobile robot positioning: sensors and techniques, Journal of Robotic Systems 4 (14), pp. 231 -249, 1996.

Brown, R. G. e Hwang, P. Y. C. Introduction to Random Signals and Applied Kalman, Filtering. Hohn Wiley & Sons, 1997.

Castro, A. P. A., Silva, J. D. S. e Simoni, P. O. Image Based Autonomous Navigation with Logic Fuzzy Control, IJCNN International Joint Conference on Neural Networks, vol.1, pp. 2200-2205, 2001.

Cazangi, R. R. Uma proposta evolutiva para controle inteligente em navegação autônoma de robôs, Master´s Thesis, Faculdade de Engenharia Elétrica e de Computação, Unicamp, Campinas, Brasil, 2004.

Chen, J., Dixon, W. E., Dawson, D. M., Mcintyre, M. Homography-based visual servo tracking control of a wheeled mobile robot, IEEE Transactions on Robotics, vol.2, nº.2, Abr. 2006.

Choomuang, R. e Afzulpurkar N. Hybrid Kalman Filter/Fuzzy logic basead position control of autonomous mobile robot, International Journal of Advanced Robotic System, vol. 2, no. 3, pp. 197-208, 2005.

Costa, E. R., Gomes, M. L. e Bianchi, R. A. C. Um mini robô móvel seguidor de pistas guiado por visão local, VI Simpósio Brasileiro de Automação Inteligente, Bauru. Anais do VI Simpósio Brasileiro de Automação Inteligente. São Paulo: Sociedade Brasileira de Automática, vol. 1, pp. 710-715, 2003.

Deccó, C. C. G. Construção de Mapas de Ambiente para Navegação de Robôs Móveis com Visão Ominidirecional Estéreo. Tese de doutorado. Faculdade de Engenharia Mecânica, USP, São Paulo, 2004.

Duda R.O. e Hart P.E. Use of the Hough transformation to detect lines and curves in pictures, Commun. Ass. Comput. Mach., vol. 15, no. 1, pp. 11-15, Jan. 1972.

Fang, Y., Dixon, W. E., Dawson, D. M. Homography-based visual servo regulation of mobile robots, IEEE Transactions on Systems and Cybernetics – Parte B: Cybernetics, vol. 35, no. 5, Out. 2005.

Forsberg, J.; Larsson, U.; Wernersson. Mobile robot navigation using the range-weighted Hough transform Robotics & Automation Magazine, IEEE, vol. 2, Issue 1, Mar 1995, pp. 18 – 26.

Forster, C. H. Q. Alinhamento imagem-modelo baseado na visão estéreo de regiões planares arbitrárias, tese de doutorado, Faculdade de Engenharia Elétrica, Unicamp, Brasil, 2004.

Forsyth, D. A, Ponce, J. Computer Vision: A Modern Approach, Prentice Hall, 2003.

Gonzalez, R. C., Woods, R. E., Processamento de imagens digitais, Editora Edgard Blucher Ltda, 1992.

Hebert, M. e Kanade, T. 3-D vision for outdoor navigation by an autonomous vehicle, Proc. Image Understanding Workshop, San Mateo, CA, pp.593-601, Abr., 1998.

Heinen, F. J. Robótica Autônoma: A integração entre Planificação e Comportamento Reativo. Editora Unisinos, 2000

Jain, R., Kasturi, R., Brian, B. G. Machine Vision, McGraw-Hill Series in Computer Science, 1995.

Johansson, K. H., Lygeros, J., Sastry, S. e Egerstedt M. Simulation of zeno hybrid automata, Proceedings of the 38*Conference on Decision & Control Phoenix, Arizona USA December, pp.3538-3543, Dez., 1999.

Josias, C.S. Radiation Instrumentation Electronics for the Pioneers III and IV Space Probes, Proceedings of the IRE Publication, April 1960, Vol. 48, Issue: 4, pp.: 735-743.

Karlsson, N., di Bernado, E., Ostrowski, Gonçalves, J. L., Pirjanian, P., Munich, M. E. The vSLAM algorithm for robust localization and mapping, IEEE International Conference on Robotics and Automation, Barcelona, Espanha, pp. 24-29, 2005.

Kriegman, D. J., Triendl, E. e Binford, T. O. Stereo vision and navigation in building for mobile robots. IEEE Transaction on Robotics and Automation", vol 5, no. 6, pp. 792-803, 1989.

Kurka, P. R. G. e Rudek, M. 3-D volume and position recovering using a virtual reference box, IEEE Transactions on Image Processing, Estados Unidos, vol. 16, nº 2, pp. 573-576, 2006.

Lygeros, J. Lecture Notes on Hybrid Systems", Department of Electrical and Computer Engineering University of Patras Rio, Patras, GR-26500, Greece, ENSIETA 2-6/2/2004.

Ma, Y., Kosecká, J. e Sastry, S.S. Vision guided navigation for a nonholonomic mobile robot, IEEE Transactions on Robotics and Automation, vol.15, no. 3, Jun. 1999.

Maia, R. R. Aprendizado Autônomo Aplicado a Solução de Conflitos em Gerenciamento de Tráfego Aéreo, Tese de Mestrado, ITA, São José dos Campos, 2004.

Moballegh, H. R., Amini, P., Pakzad Y., Hashemi, M. e Narmiami, M., An improvement of self-localization for omndirirectional mobile robots using a new odometry sensor and omnidirectional vision, IEEE, Electrical and Computer Engineering, vol. 4, pp. 2337- 2340, 2004.

Negenborn, R. Robot localization and Kalman Filters, Tese de Mestrado, Institute of Information and Computing Sciences, Thesis no. INF/SCN-03-09, Set., 2003.

Plácido, M. E. B. Sistemas robotizados de inspeção interna de dutos, Tese de Mestrado, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil, 2005.

Rudek, M., Método de posicionamento e dimensionamento 3D baseado em imagens digitais. PhD´s thesis, Depto. Engenharia Mecânica, Unicamp, Brazil, 2006.

Santana, D. D. S. Estimação de trajetórias terrestres utilizando unidades de medição inercial de baixo custo e fusão sensorial, Tese de Mestrado, Politécnica da USP, São Paulo, Brasil, 2005.

Santos, F. M., Silva, V. F., Almeida, L. Auto-localização em pequenos robôs móveis e autônomos: o caso do robô Bulldozer IV, Actas do Encontro Científico do ROBOTICA 2002 – Festival Nacional de Robótica, publicadas na revista Electrónica e Telecomunicações 3, (6), Abr., 2002.

Schmidt, R. M. e Lages, W. F. Identificação visual para navegação de robôs móveis utilizando controlador fuzzy sintonizado por otimização numérica, Anais do XIII Congresso Brasileiro de Automática, Florianópolis-SC, Brasil, pp. 2245-2250, 2000.

Shak, M. Fundamentals of computer vision, Computer Science Department, University of Central Florida, Orlando, FL32816, Dez., 1997. http://www.cs.ucf.edu/courses/cap6411/book.pdf

Siciliano, A. V., Determinação de trajetória ótima em navegação robótica móvel, utilizando algoritmo genético, Tese de Mestrado, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil, 2006.

Sousa, A. J., Costa, P. J., Moreira A. P. e Carvalho A. S. Self localization of an autonomous robot: using an EKF to merge odometry and vision based landmarks, Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation - ETFA 2005, Catania, Itália, pp. 227-234, 2005.

Surrécio A., Nunes, U. e Araújo R. Fusion of odometry with magnetic sensors using kalman filters and augmented system models for mobile robot navigation, IEEE ISIE 2005, Dubrovnik, Croácia, pp.1551-1556, 2005.

Thorpe, C., Hebert, M., Kanade, T., Shafer, S. Vision and navigation for the Carnegie-Mellon Navlab, IEEE Trans. Pattern Anal. Mach. Intell., vol.10, no. 3, pp. 362-373, Mar. 1988.

Tomatis, N. Hybrid Metric-Topological Mobile Robot Navigation, tese nº 2444, École Politechinique Fédérale de Lausanne, 2001.

Turk, M. A., Morgenthaler, D. G., Gremban, K. D., Marra M. VITS-a vision system for autonomous land vehicle navigation, IEEE Trans. Pattern Anal. Mach. Intell.,vol. 3, pp. 342-361, Mar. 1988.

Valgas, J. H. Uma metodologia de correção dinâmica de erros de odometria em robôs móveis, Tese de Mestrado, Universidade Federal de Minas Gerais, 2002.

Victorino, A. C. Controle de trajetória e estabilização de robôs móveis não-holonômicos: um Caso experimental, Tese de Mestrado, Faculdade de Engenharia Mecânica, Unicamp, Brasil, 1998.

Xia, T. K., Yang, M. e Yang R. Q. Vision based global localization for intelligent vehicles, Intelligent Vehicles Symposium, Tóquio, Japãp, pp. 571-576, 2006.

Waxman, A. M. A visual navigation system for autonomous land vehicles, IEEE J. Robot. Autom., vol. RA-3, nº. 2, pp. 124-141, Abr. 1987.

Welch, G. e Bishop, G. An introduction to the Kalman Filter. http://www.cs.unc.edu/~{welch,gb}@cs.unc.edu, Dept. of Computer Science Chapel Hill, NC 27599-3175, University of North Carolina, 2001.

Wilt, D.M.; Jenkins, P.P.; Scheiman, D.A. Photodetector development for the Wheel Abrasion Experiment on the Sojourner microrover of the Mars Pathfinder mission. Energy Conversion Engineering Conference, 1997. IECEC-97, Proceedings of the 32nd Intersociety, vol. 1, Issue 27, Jul-1 Aug 1997 , pp. 738 – 742.

Apêndices

Apêndice A – Simulação da Projeção de Objetos nas Câmeras do Robô

Suponha um ambiente contendo doze objetos, definidos como doze circunferências de mesmo raio, $\bar{r}$, em diferentes regiões do plano, além de quatro paredes limite. O centro dessas circunferências são dados nas coordenadas do mundo $\mathbf{C}_w$. Sua projeção em termos das coordenadas $\mathbf{C}_a$ e $\mathbf{C}_b$ das câmeras *a* e *b* é dada por:

$$\mathbf{C}_w = \mathbf{X}(1:2) + \mathbf{R}(\mathbf{x}_a + \mathbf{C}_a) \tag{A1}$$

$$\mathbf{C}_w = \mathbf{X}(1:2) + \mathbf{R}(\mathbf{x}_b + \mathbf{C}_b) \tag{A2}$$

Inversamente, $\mathbf{C}_a$ e $\mathbf{C}_b$ podem ser deduzidos das equações (A1) e (A2):

$$\mathbf{C}_a = \mathbf{R}'(\mathbf{C}_w - \mathbf{X}(1:2)) - \mathbf{x}_a \tag{A3}$$

$$\mathbf{C}_b = \mathbf{R}'(\mathbf{C}_w - \mathbf{X}(1:2)) - \mathbf{x}_b \tag{A4}$$

As equações das circunferências em ambas as câmeras são descritas como:

$$(x_{ca} + C_a(1))^2 + (z_{ca} + C_a(2))^2 = \bar{r}^2 \tag{A5}$$

$$(x_{cb} + C_b(1))^2 + (z_{cb} + C_b(2))^2 = \bar{r}^2 \tag{A6}$$

onde $C_a(1)$ e $C_b(1)$ são as coordenadas do centro da câmera em x_{ca} e x_{cb}, respectivamente e $C_a(2)$ e $C_b(2)$ são as coordenadas do centro da câmera em z_{ca} e z_{cb},

Os pontos projetados em cada câmera, ou da circunferência são obtidos a partir de uma reta virtual que passa pela origem das câmeras e que percorre suas respectivas telas um total de 320 pontos, ou 320 pixels. Simula-se ainda a existência de erros de projeção dos objetos no referencial do robô acrescentando-se uma medida de incerteza na localização da projeção de cada ponto na tela das câmeras. Tal incerteza é modelada por valores aleatórios com uma distribuição gaussiana.

A reta virtual numa dada câmera tem coeficiente angular $\bar{a}$ dado por:

$$\bar{a} = \frac{f}{l\left(\frac{k}{319} - \frac{1}{2}\right)}, \tag{A7}$$

sendo *l* o tamanho da tela em mm.

A reta que passa pela origem tem sua equação é da forma:

$$z_{ca} = \bar{a} x_{ca} \tag{A8}$$

O ponto de intersecção da reta virtual com cada uma das circunferências é encontrada pela busca das soluções $(x_{ca}.z_{ca})$ do seguinte polinômio de segundo grau:

$$x_{ca}^2 (1 + \bar{a})^2 + x_{ca}\left(-2 * \left(C_a(1) + C_a(2) * \bar{a}\right)\right) + C_a(1)^2 + C_a(2)^2 - \left(\bar{r}\right)^2 \tag{A9}$$

As intersecções da reta virtual não são apenas com as circunferências, há também as paredes no ambiente retangular. Considere cada parede, 1,2, 3 e 4 como sendo a parede da direita, a da vista superior, da esquerda e a inferior. Seja a posição de uma câmera *a* no mundo. Considere quatro segmentos virtuais, partindo da câmera no mundo aos cantos das paredes. Tem-se quatro regiões separadas por ângulos $\alpha_1, \alpha_2, \alpha_3$ e α_4, sendo *Lim*, o vetor que define o limite das paredes no mundo.

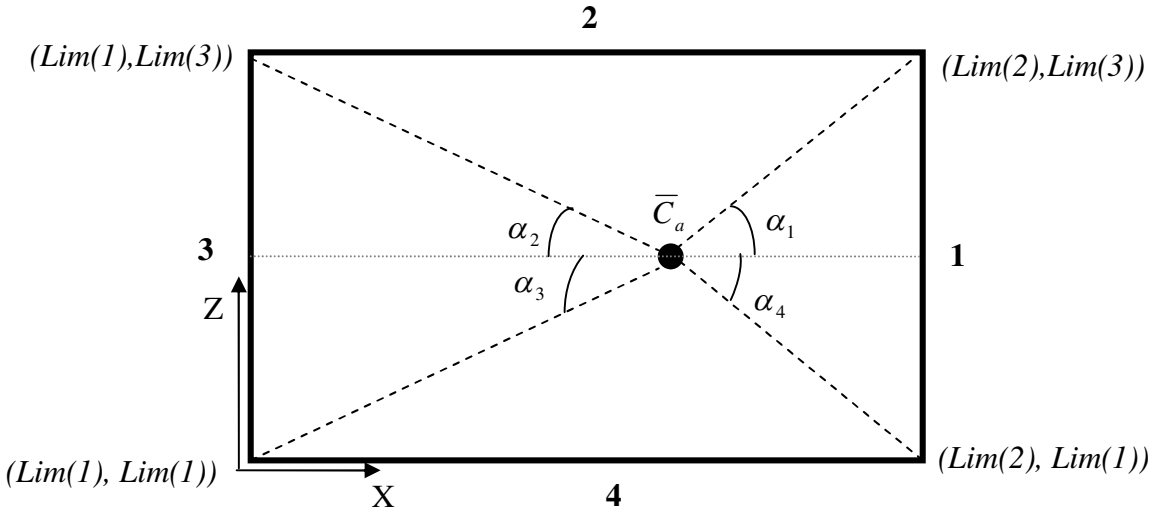


Figura A1: Para encontrar a intersecção nas paredes, basta dividir o ambiente em quatro regiões.

Os ângulos de separação do ambiente definidos na figura (A1), são:

$$\alpha_1 = atan\left(\frac{Lim(3)-\overline{C}_a(2)}{Lim(2)-\overline{C}_a(1)}\right) \quad \text{(A10)}$$

$$\alpha_2 = atan\left(\frac{Lim(3)-\overline{C}_a(2)}{\overline{C}_a(1)}\right) \quad \text{(A11)}$$

$$\alpha_3 = atan\left(\frac{\overline{C}_a(2)}{\overline{C}_a(1)}\right) \quad \text{(A12)}$$

$$\alpha_4 = atan\left(\frac{\overline{C}_a(2)}{Lim(2)-\overline{C}_a(1)}\right) \quad \text{(A13)}$$

onde $\overline{C}_a$ é a posição da câmera *a* no mundo.

As equações (A10 – A13) devem ser aplicadas também a câmera *b,* substituindo-se $\overline{C}_a$ por $\overline{C}_b$, onde $\overline{C}_b$ é a posição da câmera *b* no mundo.

Sendo o ângulo da reta virtual com x_{ca} definido como:

$$\alpha_5 = \frac{\pi}{2} - \left|atan(\overline{a})\right| \tag{A14}$$

e α_6 como sendo o ângulo dessa mesma reta com o eixo X do mundo:

$$\alpha_6 = \text{sinal}\left(319/2 - \text{k}\right) * \alpha_5 + \theta \tag{A15}$$

onde $\theta = \mathbf{X}(3)$.

Pelo valor de α_6, tendo-se os valores de α_1 a α_4, é encontrado em qual parede a reta virtual faz a intersecção. O ponto de intersecção (X,Z) no mundo, é dado pelos valores de um dos limites, substituindo-se na equação da reta virtual para se encontrar a outra coordenada. O coeficiente linear da reta virtual no mundo é dado por:

$$\overline{b} = \overline{C}_a(2) - \tan(\alpha_6)\overline{C}_a(1) \tag{A16}$$

e o coeficiente angular é dado por tan(α_6).

O ponto de intersecção do mundo é projetado nas câmeras pela equação (A17):

$$\mathbf{P}_{ca} = \mathbf{R}\left(\mathbf{P} - \mathbf{X}\right) - \mathbf{x}_a \tag{A17}$$

onde $\mathbf{P}_{ca}$ é o ponto **P** na câmera *a*. Para projetar na câmera *b* basta substituir $\mathbf{x}_a$ por $\mathbf{x}_b$.

Os pontos que são projetados pela câmera são os que possuem a menor distância entre eles e a câmera. Dessa forma, pode-se projetá-los como descrito na secção 4.2.

B – Formulação do Filtro de Kalman

Considere $\mathbf{X}_k$ o verdadeiro estado atual no tempo k, $\hat{\mathbf{X}}_k^-$ como a representação do estado a priori (sem algum tipo de ruído) em um determinado instante k e $\hat{\mathbf{X}}_k$ o estado a posteriori, estimado pelo filtro, também no tempo k.

$$\mathbf{X}_k = \mathbf{F}\mathbf{X}_{k-1} + \mathbf{G}\mathbf{u}_k + \boldsymbol{\mu}_k \tag{2.16}$$

$$\hat{\mathbf{X}}_k^- = \mathbf{F}\hat{\mathbf{X}}_{k-1}^- + \mathbf{G}\mathbf{u}_k \tag{5.4}$$

Com essas informações é possível calcular o erro relativo a priori $\mathbf{e}_k^-$ e o posteriori $\mathbf{e}_k$, ou seja, o erro do estado real com ruído comparado com o sem, e o estado real comparado com o estimado:

$$\mathbf{e}_k^- = \mathbf{X}_k - \hat{\mathbf{X}}_k^- \tag{B1}$$

$$\mathbf{e}_k = \mathbf{X}_k - \hat{\mathbf{X}}_k \tag{B2}$$

Subtraindo-se as equações (A1) e (A2), temos:

$$\mathbf{e}_k^- = \mathbf{X}_k - \hat{\mathbf{X}}_k^- = \mathbf{F}\left(\mathbf{X}_{k-1} - \hat{\mathbf{X}}_{k-1}\right) + \beta_k = \mathbf{F}\mathbf{e}_{k-1} + \beta_k \tag{B3}$$

O erro de uma medida é dita desvio padrão. O desvio padrão ao quadrado, nos dá a covariância. A matriz de covariância do erro de estimação *a priori*, a partir do erro já definido anteriormente é dada por:

$$\mathbf{P}_k^- = E\left[\mathbf{e}_k^-(\mathbf{e}_k^-)^T\right] = E\left[\left(\mathbf{F}\mathbf{e}_{k-1} + \beta_k\right)\left((\mathbf{e}_{k-1})^T\mathbf{F}^T + (\beta_k)^T\right)\right] = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q} \tag{B4}$$

sendo os vetores de estado e de ruído independentes e **Q** uma medida de covariância de ruído do processo:

$$\mathbf{Q} = E\left[\beta_k (\beta_k)^T\right] \tag{B5}$$

No tempo k, a observação feita no estado verdadeiro é dada por:

$$\mathbf{Z}_k = \mathbf{H}\mathbf{X}_k + \Sigma_k \tag{5.2}$$

Definindo $\tilde{\mathbf{y}}_k$ como erro residual ou inovação que dá a discrepância entre a medida atual $\mathbf{Z}_k$ e a medida predita $\mathbf{H}\,\mathbf{X}_k^-$, temos:

$$\tilde{\mathbf{y}}_k = \mathbf{Z}_k - \mathbf{H}_k\hat{\mathbf{X}}_k^- = H\mathbf{X}_k + \Sigma_k - \mathbf{H}\hat{\mathbf{X}}_k^- = \mathbf{H}\left(\hat{\mathbf{X}}_k - \hat{\mathbf{X}}_k^-\right) + \Sigma_k = \mathbf{H}\mathbf{e}_k^- + \Sigma_k \tag{B6}$$

Calculando a covariância de $\tilde{\mathbf{y}}_k$, e chamando essa variável de **S**:

$$\mathbf{S} = E\left[\tilde{\mathbf{y}}\tilde{\mathbf{y}}^T\right] = E\left[\mathbf{H}\mathbf{e}_k^-\mathbf{e}_k^{-T}\mathbf{H}^T\right] + E\left[\Sigma_k(\Sigma_k)^T\right] = \mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R} \tag{B7}$$

onde $\overline{\mathbf{R}}$ é a medida de covariância do ruído:

$$\overline{\mathbf{R}} = E\left[\Sigma_k(\Sigma_k)^T\right] \tag{B8 }$$

A matriz de covariância do erro de estimação *a posteriori* é dado por:

$$\mathbf{P}_k = E\left[\mathbf{e}_k(\mathbf{e}_k)^T\right] = \mathrm{cov}\left(\mathbf{X}_k - \hat{\mathbf{X}}_k\right) = \mathrm{cov}\left(\mathbf{X}_k - (\hat{\mathbf{X}}_k^- + \mathbf{K}_k\tilde{\mathbf{y}}_k)\right) \tag{B9}$$

Substituindo-se $\hat{y}_k$:

$$\mathbf{P}_k = \mathrm{cov}\left(\mathbf{X}_k - \left(\hat{\mathbf{X}}_k^- + \mathbf{K}_k(Z_k - \mathbf{H}_k\hat{\mathbf{X}}_k^-)\right)\right) \tag{B10}$$

Se z_k é agora substituído, temos:

$$\begin{aligned}\mathbf{P}_k &= \mathrm{cov}\left(\mathbf{X}_k - \left(\hat{\mathbf{X}}_k^- + \mathbf{K}_k(\mathbf{H}_k\mathbf{X}_k + \Sigma_k - \mathbf{H}_k\hat{\mathbf{X}}_k^-)\right)\right) \\ &= \mathrm{cov}\left((I - \mathbf{K}_k\mathbf{H}_k)\left(\mathbf{X}_k - \hat{\mathbf{X}}_k^-\right) - \left(\mathbf{K}_k\Sigma_k\right)\right)\end{aligned} \tag{B11}$$

Então:

$$\mathbf{P}_k = \mathrm{cov}\left((I - \mathbf{K}_k\mathbf{H}_k)\left(\mathbf{X}_k - \hat{\mathbf{X}}_k^-\right)\right) + \mathrm{cov}\left(\mathbf{K}_k\Sigma_k\right) \tag{B12}$$

Reescrevendo $\mathbf{P}_k$:

$$\mathbf{P}_k = (I - \mathbf{K}_k\mathbf{H}_k)\,\mathrm{cov}\left(\mathbf{X}_k - \hat{\mathbf{X}}_k^-\right)(I - \mathbf{K}_k\mathbf{H}_k)^T + \mathbf{K}_k\,\mathrm{cov}\left(\Sigma_k\right)\mathbf{K}_k^T$$

$$\mathbf{P}_k = (I - \mathbf{K}_k\mathbf{H}_k)\;\mathbf{P}_k^-(I - \mathbf{K}_k\mathbf{H}_k)^T + \mathbf{K}_k\overline{\mathbf{R}}_k\mathbf{K}_k^T \tag{B13}$$

Podemos abrir essa expressão, considerando as seguintes propriedades de matriz transposta, para duas matrizes **A** e **B:**

$$\left(\mathbf{A} + \mathbf{B}\right)^T = \mathbf{A}^T + \mathbf{B}^T \tag{B14}$$

$$(\mathbf{AB})^{\mathrm{T}} = \mathbf{B}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}} \tag{B15}$$

Aplicando-se essas propriedades em (A13), temos:

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k\mathbf{H}_k\,\mathbf{P}_k^- - \mathbf{P}_k^-\mathbf{H}_k^T\mathbf{K}_k^T + \mathbf{K}_k\left(\mathbf{H}_k\,\mathbf{P}_k^-\mathbf{H}_k^T + \overline{\mathbf{R}}_k\right)\mathbf{K}_k^T \tag{B16}$$

O termo entre parênteses, é $\mathbf{S}_k$, onde já foi definido anteriormente:

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k\mathbf{H}_k\,\mathbf{P}_k^- - \mathbf{P}_k^-\mathbf{H}_k^T\mathbf{K}_k^T + \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T \tag{B17}$$

Para minimizar o erro, derivamos $\mathbf{P}_k$ e o igualamos a zero:

$$\frac{\partial tr\left(\mathbf{P}_k\right)}{\partial K_k} = -2\left(\mathbf{H}_k\,\mathbf{P}_k^-\right)^T + 2\mathbf{K}_k\mathbf{S}_k = 0 \tag{B18}$$

sendo

$$\mathbf{H}_k\mathbf{K}_k\,\mathbf{P}_k^- = \mathbf{P}_k^-\mathbf{H}_k^T\mathbf{K}_k^T \tag{B19}$$

$$\frac{d\left[trace(AB)\right]}{dA} = B^T \;\rightarrow\; \text{A e B devem ser quadradas} \tag{B20}$$

$$\frac{d\left[trace\left(ACA^T\right)\right]}{dA} = 2AC \;\rightarrow\; \text{C deve ser simétrica} \tag{B21}$$

Fazendo-se as igualdades:

$$\mathbf{K}_k\mathbf{S}_k = -2\left(\mathbf{H}_k\,\mathbf{P}_k^-\right)^T = \mathbf{P}_{k-1}\mathbf{H}_k^T \tag{B22}$$

Isolando-se K_k, temos o valor que dá o ganho de Kalman:

$$\mathbf{K}_k = \mathbf{P}_{k-1}\mathbf{H}_k^T\mathbf{S}_k^{-1} \tag{B23}$$

Multiplicando a equação (A22) por $\mathbf{K}_k^T$, temos:

$$\mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T = \mathbf{P}_k^-\mathbf{H}_k^T\mathbf{K}_k^T \tag{B24}$$

Dessa forma, retornando P_k na equação (B16), temos que seus dois últimos termos são nulos:

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{H}_k\mathbf{K}_k\,\mathbf{P}_k^- = \left(\mathrm{I} - \mathbf{H}_k\mathbf{K}_k\right)\mathbf{P}_k^- \tag{5.8}$$

C – Artigo Submetido a RBCM

Assessment of Uncertainties in the Control of Trajectory of a Robot Using Image Information

Paulo R. G. Kurka, Luciana Diogenes, Fernando L. Pereira

Abstract—**The paper proposes a trajectory control strategy for a roving robot, using Kalman filtered estimates of position from odometry and stereoscopic vision measurements. Position estimate uncertainties are determined from the constitutive robot's kinematic model and the geometry of spatial reconstruction of image projected points. Different levels of uncertainties from odometry and vision estimates of position are input in a simulation of the robot's movement. The simulations evaluate the influence of sensor noise to the robot's attitude in an autonomous roving stretch..**

Index Terms—Mobile robot motion-planning, Robot vision systems, Kalman Filtering**.**

INTRODUCTION

The proposal of the work is the integration of control and navigation techniques, where a "line of sight" roving scheme is adopted in order to guide a robotic vehicle to the desired direction of motion. The only sensors used by the robot are odometry encoders and a pair of parallel cameras. The combination of odometry and computer vision technique is a simple and inexpensive way to control the trajectory of the vehicle through an unknown environment.

Several works have been written on the fusion of various sensors, including vision, to build a map of the environment for the navigation of wheeled mobile robots [1]-[5]. The main purpose of such works is to achieve precision in the determination of the mobile robot's position, for a variety of practical applications. A few other works are specifically dedicated to investigate the problem of using camera images for the stabilization of the trajectories of mobile robots.

In [6], one camera is used to control the trajectory of a mobile robot, moving about piecewise analytic curves. In the work of [7] and [8] the visual trajectory stabilization is done using light emitting diodes (LEDs) as fixed spatial references. Pre-recorded images of the reference LEDs, seen from different angles, comprise the stationary referential database. The mentioned works use also stabilization models, based on Butterworth or Extended Kalman filters. The present work has the same goal to establish a trajectory control for a mobile robot, based in computational vision. In the present study, the referential geometric system is based on cameras

The authors wish to thank CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nivel Superior – Brazil) and GRICES (Gabinete de Relações Internacionais – Portugal) for the financial support of the research.

P. R. G. Kurka is with the Faculty of Mechanical Engineering, Universidade Estadual de Campinas, Brazil, Caixa Postal 6122, CEP 13086-860, Campinas, São Paulo, Brazil (phone: +55 19-3788-3175; fax: +55 19 3289-3722; e-mail: kurka@ fem.unicamp.br).

L. Diogenes is a Ph. D student at the Faculty of Mechanical Engineering, Universidade Estadual de Campinas, Brazil, Caixa Postal 6122, CEP 13086-860, Campinas, São Paulo, Brazil (e-mail: luciana@ fem.unicamp.br).

F. L. Pereira is with the Dept. of Electrotechnical Engineering of Faculdade de Engenharia da Universidade do Porto, Portugal. (phone+351 22 508 1857; fax.:+ 351 22 508 1443; e-mail: flp@fe.up.pt).

observations. It works with the hypothesis of an unstructured environment. In this sense, the information of spatial orientation depends, on depth maps that are obtained from the image analysis of a pair of cameras.

Another source for spatial orientation is provided from odometry measurements in the robot. Each kind of sensor (camera or encoder) has an intrinsic error in the spatial orientation estimation. The quantification of the position error obtained by analyses of the stereoscopic images, and the position error obtained by odometry measurement, can be used in calculating a stabilized trajectory path. To this purpose, a digital Kalman filter is also integrated in the navigating process.

The proposed control structure is modular in the sense that it becomes active with the appropriate parameter values whenever a motion segment is considered pertinent. A motion segment is that specified by a starting point and a vision landmark defined by the overall mission planner. In particular, it can be easily seen that obstacle detection and collision avoidance features can be naturally articulated into a control architecture in which the control system whereby discussed has a critical role. The present approach allows a step further in regarding the controllability, in the context of management of the consumption of onboard resources, as the control of an autonomous vehicle should be. This can be easily seen from the following two important features: a) Feedback from image information, which is a very expensive resource from the computational point of view, and may be used at a lower frequency than that of odometry. In fact, an event-driven control can be defined that occurs whenever the covariance estimate of odometry reaches some given level. b) The consumption of onboard resources is allocated to two large classes of activities: observing the environment (sensing and processing sensing data), and acting on the environment (processing control signals and powering the actuators). A desirable feature of the control architecture consists in allocating the onboard resources to the two classes so that the mission goals are achieved with the best performance. Since the covariance matrix of the state (position) variable depends on the control effort at each time, the control synthesis may be formulated as a performance optimization problem with (among others) constraints on the covariance of the state (position) estimate.

In the present development, vision through a pair of cameras is employed to yield the vehicle's estimated position and aid the calculation of the best roving trajectory. Data from the two different position sensors (cameras and encoder) is fused by using Kalman filter techniques ([9]-[15]), in order to produce a minimum variance estimate of the state variable of a mobile robot that navigates in an unknown environment.

A contribution of the present work is the analysis and discussion of a simulated application of such a technique to a mobile robot navigating in an environment with a minimum number of fixed references. The complete roving strategy for the movement of such a robot, as mentioned before, can be regarded as the concatenation of a number of intermediary stretch paths with known starting and arrival points. Since the navigation in each stretch path is a basic and repetitive pattern of the robot's roving strategy, the study concentrates on the detailed aspect of such a movement.

The work is organized as follows. Section 2 presents the stereoscopic techniques of 3-D map reconstruction used in the estimation of the robot's position, as referred to fixed objects. The kinematical model of a robot with differential traction is presented in section 3. Section 4 displays the Kalman filter strategy that yields state estimates based on odometry and camera positioning information. The basic roving control strategy is described in section 5. Simulation results and their analysis are presented in section 5 helping to assess the use of the combined sensing techniques.

USE OF VISION FOR POSITIONING MAPPING

The position and dimensioning of solid objects can be determined from a pair of images of two calibrated cameras through epipolar geometry measurements ([16]-[18]). The image (or intensity map) of solids that are projected in the 2-D vision field of the cameras are used to form a simple space map of the 3-D environment ahead of it.

In the case of a mobile robot that moves in a flat surface, the relevant environment information from the camera images can be reduced to the horizon line of the robot, giving 1-D intensity maps that are used to yield a 2-D space map.

Fig. 1 shows the geometry of projection of a point Pi of a 2-D space onto the linear coordinates of the screens of two parallel pinhole cameras located on the robot. Coordinates x and z of camera a are the reference axis of the robot vision system. Coordinates r_i and s_i are the corresponding positions of the image projections of point Pi into the screens of cameras a and b. The optical centers of the cameras are separated from their screens by the focal distance f. The optical centers of the two parallel cameras are also separated between themselves by distance t.

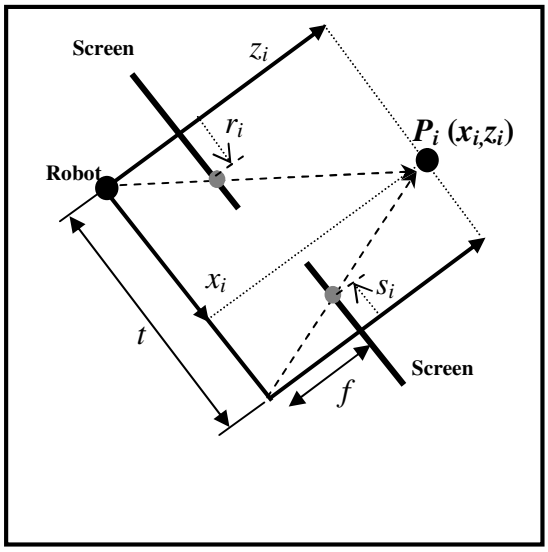


Fig. 1. Projection of planar cameras.

Vector xi of coordinates (xi, zi) describes the position of point Pi with respect to the reference camera axis. The elements of xi are calculated from the correspondent image positions of point Pi:

$$\mathbf{x}_i = \begin{bmatrix} x_i \\ z_i \end{bmatrix} = \frac{t}{(r_i - s_i)} \begin{bmatrix} r_i \\ f \end{bmatrix}. \qquad (1)$$

The uncertainty associated with calculation of the coordinates of Pi from its screen projections is given by:

$$d\mathbf{x}_i = \begin{bmatrix} dx_i \\ dz_i \end{bmatrix}, \qquad (2)$$

or

$$d\mathbf{x}_i = \mathbf{B}_i d\mathbf{r}_i \qquad (3)$$

where

$$\mathbf{B}_i = \frac{z_i^2}{ft} \begin{bmatrix} -\frac{s_i}{f} & \frac{r_i}{f} \\ -1 & 1 \end{bmatrix} \qquad (4)$$

and,

$$d\mathbf{r}_i = \begin{bmatrix} dr_i \\ ds_i \end{bmatrix}. \qquad (5)$$

The robot's state vector Z, as observed from the camera, can be obtained by defining its location with respect to a pair of fixed points.

Fig. 2 shows the geometry of the robot's observed position with respect to fixed points Pi and Pj. Equation (6) below shows the relation between the absolute coordinates of points Pi and Pj and their visualization as seen from the robot:

$$\mathbf{Z} = \begin{bmatrix} Z_X \\ Z_Z \\ Z_\phi \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\left[\mathbf{P}_i + \mathbf{P}_j - \mathbf{R}^T(\mathbf{x}_i + \mathbf{x}_j)\right] \\ \alpha + \gamma \end{bmatrix}, \qquad (6)$$

(1)

where Pi and Pj are vectors that define the absolute position of each point. The same points seen from the robot's reference frame are written as xi and xj respectively. The robot's rotation matrix R is defined as

$$\mathbf{R} = \begin{bmatrix} \sin(Z_\phi) & -\cos(Z_\phi) \\ \cos(Z_\phi) & \sin(Z_\phi) \end{bmatrix}. \quad (7)$$

Vj g"tqdqv="u'cpi wrct"r qukkqp"\ N'ku'ecrewrcvgf "htqo "cpi rgu'I'cpf "K"*ugg"

Fig. 2. Position of two points in space.+0'C pi rg" I'ku' the estimated inclination between the robot's heading position and segment $\overline{P_i P_j}$ and is defined as

$$\alpha = \frac{\pi}{2} + \beta \quad (8)$$

where

$$\beta = \arctan\left(\frac{z_i - z_j}{x_j - x_i}\right) \quad (9)$$

with xi, zi, xj and zj being the x and z coordinates of points Pi and Pj, as seen from the robot's reference frame.

C pi rg"K"ku'vj g"kpenkpcvkqp"qH'ugi o gpv $\overline{P_i P_j}$ with respect to the absolute direction X.

The uncertainties associated with the calculation of the robot's state vector are given by the derivative of the terms of (6), which leads to:

$$d\mathbf{Z} = \begin{bmatrix} dZ_X \\ dZ_Z \\ dZ_\theta \end{bmatrix} = \begin{bmatrix} \mathbf{D} \\ \mathbf{C} \end{bmatrix} d\mathbf{r}_{ij} \quad (10)$$

where

$$\mathbf{D} = -\frac{1}{2}\left[\frac{\partial \mathbf{R}^T}{\partial Z_\phi}(\mathbf{x}_i + \mathbf{x}_j)\mathbf{C} + \mathbf{R}^T \begin{bmatrix} \mathbf{B}_i & \mathbf{B}_j \end{bmatrix}\right], \quad (11)$$

$$\mathbf{C} = \frac{\cos^2(\beta)}{x_j - x_i}\begin{bmatrix} \tan(\beta) & 1 & -\tan(\beta) & -1 \end{bmatrix} \quad (12)$$

and

$$d\mathbf{r}_{ij} = \begin{bmatrix} d\mathbf{r}_i \\ d\mathbf{r}_j \end{bmatrix}. \quad (13)$$

It is assumed that the cameras projection uncertainties are associated with the f kuetgvk|cvkqp"gttqt"Σ."f gHkpgf "cu

$$\nu = \frac{1}{2}\frac{\ell}{N_{Pix}}, \quad (14)$$

where ℓ is the length of the camera's screen and NPix is the number of pixels of the camera. It is also assumed that the cameras projection uncertainties are statistically independent variables, which finally leads to the expression of the camera's error covariance matrix S:

$$\mathbf{S} = \begin{bmatrix} \mathbf{D} \\ \mathbf{C} \end{bmatrix} \mathbf{E}\begin{bmatrix} \mathbf{D}^T & \mathbf{C}^T \end{bmatrix}, \quad (25)$$

where

$$\mathbf{E} = \nu^2 \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}. \quad (16)$$

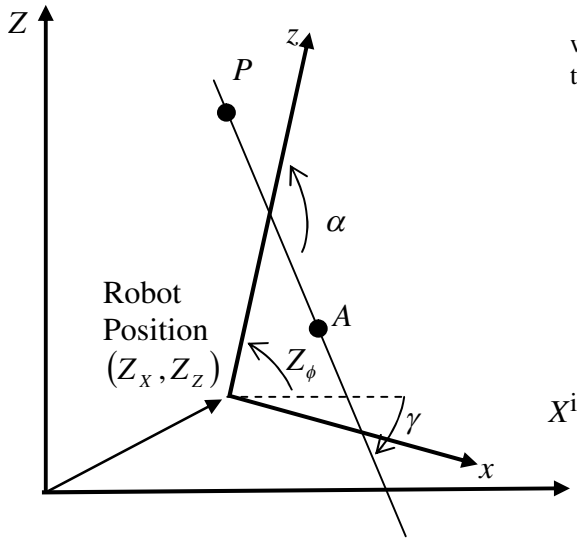


Fig. 2. Position of two points in space.

ODOMETRY MEASUREMENTS AND KINEMATIC MODEL OF THE ROBOT

The robot model used consists of two lateral wheels; set in motion by two independent motors as shown in
F

ig. 3. The robot's position and orientation, is described by the pose vector

$$\mathbf{X}_k = \begin{bmatrix} X_{Xk} \\ X_{Zk} \\ X_{\phi k} \end{bmatrix}, \quad (17)$$

whose estimate can be calculated by the rotation of the wheels, through a method know as odometry ([19],[20]). In this process, the right and left wheels revolve an angle equivalent to Nd and Ne encoder step counts, yielding the a-priori or predicted state $\mathbf{X}_k^-$ given by:

$$\mathbf{X}_k^- = \hat{\mathbf{X}}_{k-1} + \mathbf{G}_{k-1}\mathbf{u}_k \quad (18)$$

where $\hat{\mathbf{X}}_{k-1}$ is the previously estimated state of the robot,

$$\mathbf{G}_{k-1} = \begin{bmatrix} \frac{\pi r_d}{N_{res}} CosX_{\phi k-1} & \frac{\pi r_e}{N_{res}} CosX_{\phi k-1} \\ \frac{\pi r_d}{N_{res}} SinX_{\phi k-1} & \frac{\pi r_e}{N_{res}} SinX_{\phi k-1} \\ \frac{2\pi r_d}{bN_{res}} & -\frac{2\pi r_e}{bN_{res}} \end{bmatrix} \quad (19)$$

$X$is the robot's state transition matrix and

$$\mathbf{u}_k = \begin{bmatrix} N_{dk} \\ N_{ek} \end{bmatrix}. \quad (20)$$

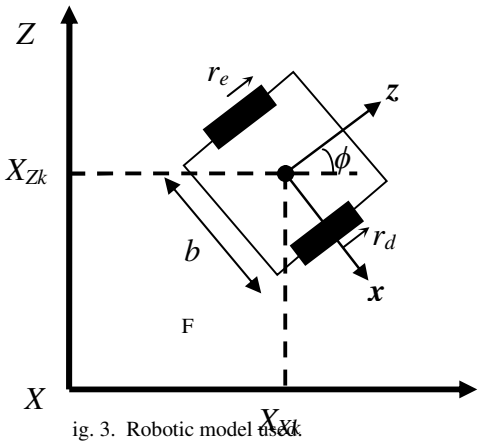
is the robot's odometry control vector at instant k. The other robot parameters are:

re and rd: left and right wheels radius,

b: distance between wheels,

$X_{\phi k-1}$: angle of robot,

Nres: number of encoder steps per revolution.



F
ig. 3. Robotic model used.

The uncertainties associated with the

estimation of the robot's state through odometry are given by the derivative of (21) above, yielding

$$d\mathbf{X}_k^- = \mathbf{L}_k d\mathbf{u}_k, \qquad (21)$$

where

$$\mathbf{L}_k = \frac{\partial \mathbf{G}_k}{\partial X_{\phi k}} \mathbf{u}_k \begin{bmatrix} \frac{2\pi r_d}{bN_{res}} & \frac{-2\pi r_e}{bN_{res}} \end{bmatrix} + \mathbf{G}_k \qquad (22)$$

Errors in the calculation of the predicted state of the robot are a measure of uncertainties that are introduced in the odometry input control information. Such an error can come from a number of different disturbing factors such as imprecise information on the dimensions of robot elements, misalignment and slipping of the wheels. On top of all those errors, a systematic odometry discretization uncertainty is also present, which has a dimension of half of a step of the controlling motor. It is assumed that the total odometry error has a magnitude of β."kp"wpkw'qh' rotor steps, and that this error is an independent stochastic variable for each motor controlled wheel. In this case, the odometry covariance error matrix is defined as

$$\mathbf{Q} = \mathbf{L}\mathbf{d}\mathbf{L}^T, \qquad (23)$$

with

$$\mathbf{d} = \omega^2 \begin{bmatrix} 1 & \\ & 1 \end{bmatrix}. \qquad (3)$$

THE CONTROL STRATEGY

The Kalman filter ([21]-[23]) uses information obtained by odometry and camera measurements to predict the future state of the robot when subject to a given path (position) control strategy. The algorithm is divided in two parts: prediction and correction. In such an approach, Xk is the real pose of the robot at instant k and uk the input rotation or odometry measurement. The discrete dynamic equation for such a linear system can be written as:

$$\mathbf{X}_k = \mathbf{X}_{k-1} + \mathbf{G}_{k=1}\mathbf{u}_k + \mathbf{q}_k \qquad (44)$$

where $\mathbf{q}_k$ is the system error positioning vector, which is assumed to have zero mean normal Gaussian distribution and covariance matrix Qk, that is,

$$p(\mathbf{q}_k) \sim N(0, \mathbf{Q}_k). \qquad (25)$$

The system error positioning vector is the same as described in equation (21), comprised of the variations that occur to the robot's position when the commanded odometry is not fully translated into the expected change of state.
The camera observation of the system state, shown in section 2, is defined by vector Zk. Such a position estimate relates to the real state through the observation error vector $\mathbf{s}_k$, given by equation:

$$\mathbf{Z}_k = \mathbf{H}\mathbf{X}_k + \mathbf{s}_k, \qquad (56)$$

where H is the observation matrix, assumed to be identity in the present work. The observation error is assumed to have zero mean normal Gaussian distribution and covariance matrix Sk, that is,

$$p(\mathbf{s}_k) \sim N(0, \mathbf{S}_k) \qquad (67)$$

The a-priori state estimate of the system, $\mathbf{X}_k^-$, is calculated from the previous optimal state estimate, $\hat{\mathbf{X}}_{k-1}$ and from the values given by odometry at the initial phase of the estimation process, as shown below:

$$\mathbf{X}_k^- = \hat{\mathbf{X}}_{k-1} + \mathbf{G}_{k-1}\mathbf{u}_k \qquad (78)$$

An estimate of the robot's position, $\mathbf{Z}_k$, is, independently and in parallel, obtained from the vision sensor. The a-posteriori (or optimum) estimate of the system's state $\hat{\mathbf{X}}_k$, is given by:

$$\hat{\mathbf{X}}_k = \mathbf{X}_k^- + \mathbf{K}_k\left(\mathbf{Z}_k - \mathbf{H}\mathbf{X}_k^-\right) \quad (29)$$

where K_k is the optimal Kalman gain matrix.

The value between parentheses in (29) is known as the innovation or measurement residual. Matrix Kk is given by:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}^T\left(\mathbf{H}\mathbf{P}_k^- \mathbf{H}^T + \mathbf{S}_k\right)^{-1} \quad (30)$$

Term $\mathbf{P}_k^-$ of (31) is the estimated a-priori error covariance matrix and is calculated as:

$$\mathbf{P}_k^- = \hat{\mathbf{P}}_{k-1} + \mathbf{Q}_k . \quad (31)$$

Matrix, $\hat{\mathbf{P}}_{k-1}$ of (32) is the a-posteriori estimate error covariance calculated in the previous interaction. The updated error covariance matrix, is calculated by:

$$\hat{\mathbf{P}}_k = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}\right)\mathbf{P}_k^- \quad (32)$$

A scheme for the state estimation process is shown in Fig.4

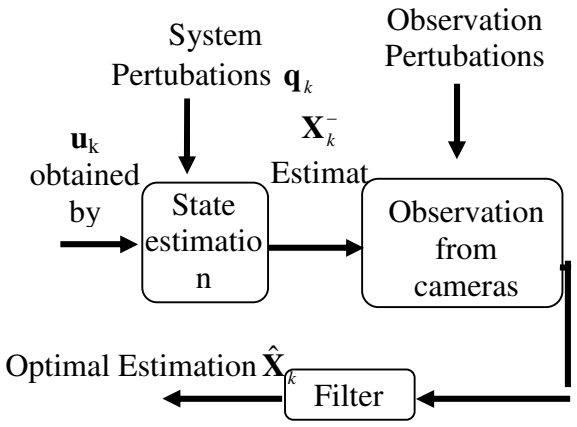


Fig. 4. Kalman filter scheme

ROVING STRATEGY

The robot's roving strategy is defined in the following sequence:

The robot calculates its approximate position via the optimal Kalman scheme described in section 4, using odometry and visual estimates of its state. In the process of estimating its position, the robot also estimates the position of the aimed point of arrival.

The robot estimates the ideal intermediate position where it should move, $\overline{\mathbf{X}}_{k+1}$, in order to approach the point of arrival.

The ideal intermediate position is one where the robot is oriented towards the estimated point of arrival, and travels an equal, pre defined, number of encoder steps N. After establishing the ideal intermediate position, a true number of moving steps is calculated for each wheel, based on a proportional control, in order to move the robot form the present to the ideal position. Equation (34) shows the scheme for calculation of the proportional control:

$$\mathbf{u}_k = \mathbf{G}_k^+\left(\overline{\mathbf{X}}_{k+1} - \hat{\mathbf{X}}_k\right) \quad (83)$$

where $\mathbf{G}_k^+$ is the pseudo inverse of Gk and

$$\overline{\mathbf{X}}_{k+1} = \hat{\mathbf{X}}_k + \overline{\mathbf{G}}_k\begin{bmatrix} N \\ N \end{bmatrix}. \quad (94)$$

$\overline{\mathbf{G}}_k$ is the robot's state transition matrix with the ideal angular orientation towards the estimated point of arrival.

The robot is controlled to move to the ideal intermediate position, but in fact it will move to a new and distinct real position, due to the system odometry error.

Steps a) to c) are repeated until the robot's estimated position is within a tolerated distance error from the aimed POA.

SIMULATION

A numerical simulation is presented in order to observe the trajectory of the robot undergoing a basic movement from a staring point to an aimed position. A second point is also given, together with the aimed position, which forms the minimum set of points that can serve as a reference to the robot's camera positioning system. Geometrical information concerning the robot's characteristics and reference points is given in Table I:

TABLE 1
GEOMETRIC PARAMETERS AND REFERENCE POINTS

Starting Point – mm	(0, 0)
Reference Point P1 – mm	(6000,4000)
Reference Point P2 (aimed position) – mm	(6500,4000)
Initial State vector (X0)	$\mathbf{X}_0^T = [0 \quad 0 \quad 0.55$
Radius of wheels (rdand re) – mm	100
Distance between wheels (b) – mm	200
Number of encoder steps per revolution (Nres)	36
Number of encoder steps counts for ideal movement (N)	18
Standard deviation for odometry discretization gttqt"*β+	0.5
Standard deviation for camera discretization error (ν) –mm	5e-07

The simulations show the effect of different odometry and camera errors, as well as different assumptions for the initial condition, as shown in Table II.

TABLE II
SUMMARY OF SIMULATION CONDITIONS

Cond.	Initial state	Camera Error	Odometry error
1	X0	Discretization error (5e-07 mm)	Discretization error (0.5)
2	Camera estimated : X0=Z0	Discretization error (5e-07 mm)	Discretization error (0.5)
3	X0	2 x Discretization error (1e-6 mm)	2 x Discretization error (1.0)
4	Camera estimated : X0=Z0	2 x Discretization error (1e-6 mm)	2 x Discretization error (1.0)
5	X0	4 x Discretization error (2e-6 mm)	Discretization error (0.5)
6	Camera estimated : X0=Z0	4 x Discretization error (2e-6 mm)	Discretization error (0.5)
7	X0	Discretization error (5e-07 mm)	4 x Discretization error (2.0)
8	Camera estimated : X0=Z0	Discretization error (5e-07 mm)	4 x Discretization error (2.0)

RESULTS OF SIMULATION CONDITION 1: The robot tends to follow the exact path without great deviations, as shown in Figura.

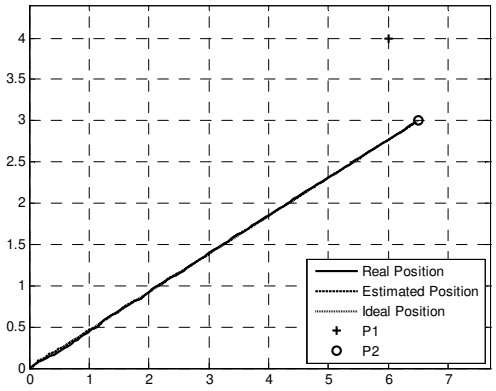


Fig. 5. Start X0 (m), 1 x camera error, 1 x odometry error.

RESULTS OF SIMULATION CONDITION 2: The robot's real path is influenced by the camera estimated starting point. The error in the camera estimation of the starting position is high, since the visual reference points are at a relatively long distance from the robot. As a result of such a high error estimation of the starting position, the real path of the robot is drawn away from the estimated ideal position. The moment the estimated and real paths become close together, at the proximity of the reference points, the robot tends to complete its movement in a correct approach to the point of arrival, as shown in Fig. 6.

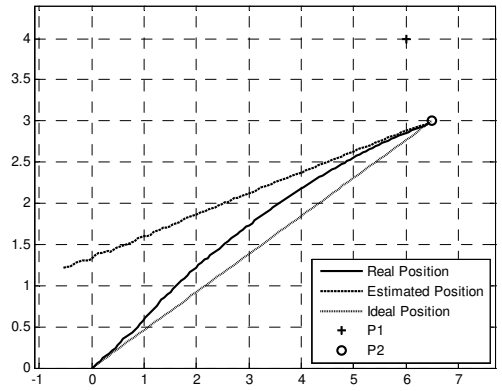


Fig. 6. Start X0=Z0 (m), 1 x camera Error, 1 x odometry error.

RESULTS OF SIMULATION CONDITION 3: An increase of 100% in the camera and odometry and discretization errors doesn't affect in a substantive way the ability of the robot to follow the ideal path on both real and estimated positions, as shown in Fig.7:

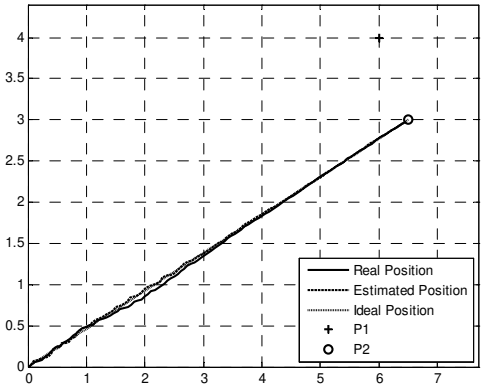


Fig. 7. Start X0 (m), 2 x camera error, 2 x odometry error.

RESULTS OF SIMULATION CONDITION 4: The same deviation of the real path of the robot from the ideal position, as observed in simulation condition 2, is perceived here. The estimated starting position in this case is much further from the real point of departure of the robot, due to the high magnitude of errors involved in image estimation of state, as seen in Fig. 8.

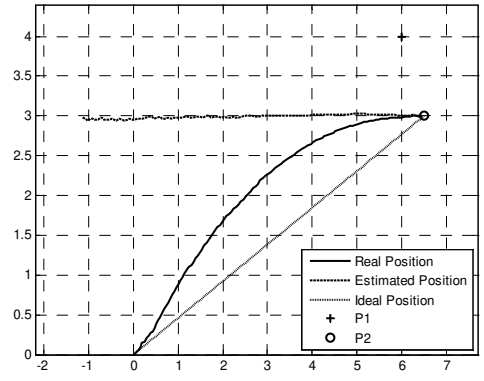


Fig. 8. Start X0=Z0 (m), 2 x odometry error, 2 x camera error.

RESULTS OF SIMULATION CONDITION 5: This simulation tests the influence of a high magnitude error in camera estimates of position, against a low magnitude error in odometry measurements. The robot's real path in this case has a slight deviation from the ideal position, away from the reference points, tending to stabilize into the predicted path as it approaches the point of arrival as seen in Fig. 9.

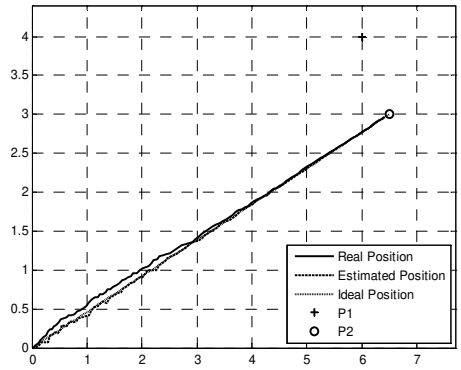


Fig. 9. Start X0 (m), 4 x camera error, 1 x odometry error.

RESULTS OF SIMULATION CONDITION 6: The initial deviation of the estimated starting point is again responsible for the real path to depart from the ideal position. The deviation magnitude is also is proportional to amount of error in the camera position estimate as seen in Fig. 10.

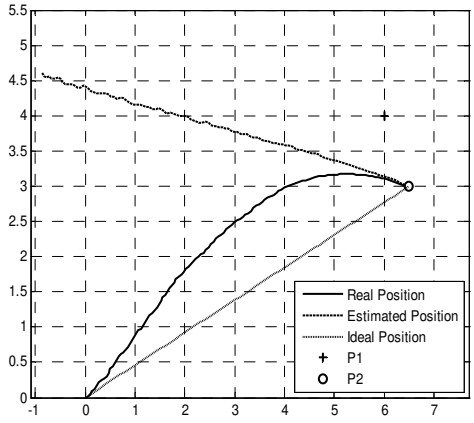


Fig. 10. Start X0=Z0 (m), 4 x camera error, 1 x odometry error.

RESULTS OF SIMULATION CONDITION 7: This simulation tests the influence of a high magnitude error in odometry measurements, against a low magnitude error in camera estimates of position. The robot's real path in this case has almost no deviation from the ideal position as seen in Fig. 11.

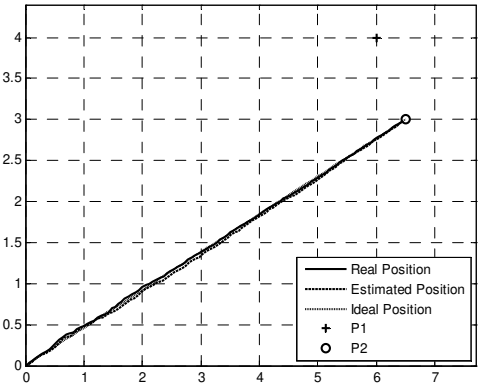


Fig. 11. Start X0 (m), 1 x camera error, 4 x odometry error.

RESULTS OF SIMULATION CONDITION 8: In this simulation again, departure of the robot from the ideal path is only due to an erroneous estimation of its initial position. Convergence to the aimed position however is achieved in spite of a high magnitude odometry error, as seen in Fig.12.

Fig. 12. Start X0=Z0 (m), 1 x camera error, 4 x odometry error.

CONCLUSION

The simulations show in all cases that the control strategy is able to guide the robot to the desired objective. The estimated initial condition is important in the way the robot keeps to the ideal path during its roving trajectory. The tendency to follow the ideal path happens each time the exact initial condition is given as the robot's staring state. Deviations from the ideal path are mainly due to incorrect prediction of the initial position of the robot.

The simulations have also shown the importance of the camera estimated positions in the path following algorithm, as well as the smaller influence of odometry errors in stabilizing of the trajectory.

REFERENCES

[1] M. Hebert and T. Kanade,, "3-D vision for outdoor navigation by an autonomous vehicle", Proc. Image Understanding Workshop, San Mateo, CA, april, pp.593-601, 1998.

[2] D. J. Kriegman, E. Triendl and T. O.Binford, "Stereo Vision and Navigation in Building for Mobile Robots. IEEE Transaction on Robotics and Automation", vol 5, no. 6, pp. 792-803, 1989.

[3] C. E.,Thorpe, M.,Hebert, T. Kanade, S. Shafer, "Vision and navigation for the Carnegie-Mellon Navlab", IEEE Trans. Pattern Anal. Mach. Intell., vol.10, no. 3, pp. 362-373, Mar. 1988.

[4] M.A. Turk, D. G. Morgenthaler, K. D. Gremban, M. Marra, "VITS-a vision system for autonomous land vehicle navigation", IEEE Trans. Pattern Anal. Mach. Intell., vol. 3, pp. 342-361, Mar. 1988.

[5] A. M. Waxman, "A visual navigation system for autonomous land vehicles", IEEE J. Robot. Autom., vol. RA-3, no. 2, pp. 124-141, Apr. 1987.

[6] Y. Ma, J. Kosecká and S.S Sastry, "Vision Guided Navigation for a Nonholonomic Mobile Robot", IEEE Transactions on Robotics and Automation, vol.15, no. 3, Jun. 1999.

[7] W. Fang, W. E. Dixon, D. M. Dawson, "Homography-Based Visual Servo Regulation of Mobile Robots", IEEE Transactions on Systems and Cybernetics – Parte B: Cybernetics, vol. 35, no. 5, Oct. 2005.

[8] J. Chen, W. E. Dixon, D. M. Dawson, M. Mcintyre,"Homography-Based Visual Servo Tracking Control of a Wheeled Mobile Robot", IEEE Transactions on Robotics, vol. 2, no. 2, Apr. 2006.

[9] R. Choomuang and N. Afzulpurkar, "Hybrid Kalman Filter/Fuzzy Logic Basead Position Control of Autonomous Mobile Robot", International Journal of Advanced Robotic System, vol. 2, no. 3, pp. 197-208, 2005.

[10] T. K. Xia, M. Yang and R. Q. Yang, "Vision Basead Global Localization for Intelligent Vehicles", Intelligent Vehicles Symposium, pp. 571-576, 2006.

[11] H. R. Moballegh, P. Amini, Y. Pakzad, M. Hashemi and M. Narmiami. “An Improvement of Self-Localization for Omndirirectional Mobile Robots Using a New Odometry Sensor and OmnidireCtional Vision”. IEEE, Electrical and Computer Engineering, vol. 4, pp. 2337- 2340, 2004.

[12] A. J. Sousa, P. J. Costa, A. P. Moreira and A. S. Carvalho, “Self Localization of an Autonomous Robot: Using an EKF to merge Odometry and Vision based Landmarks”, Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation - ETFA 2005, Catania, Italy, pp. 227-234, 2005.

[13] A. Surrécio, U. Nunes and R. Araújo, “Fusion of Odometry with Magnetic Sensors Using Kalman Filters and Augmented System Models for Mobile Robot Navigation”, IEEE ISIE 2005, Dubrovnik, Croatia, pp.1551-1556, 2005.

[14] D. J. Kriegman, E. Triendl and T. O.Binford. “Stereo Vision and Navigation in Building for Mobile Robots”. IEEE Transaction on Robotics and Automation, vol.5, no. 6, pp. 792-803, 1989.

[15] N. Karlsson, E. Di Bernado, J. Ostrowski, L. Gonçalves, P. Pirjanian, M. E. Munich, The vSLAM Algorithm for Robust Localization and Mapping. IEEE, International Conference on Robotics and Automation, Barcelona, Spain, pp. 24-29, 2005.

[16] P. R. G. Kurka and M. Rudek. 3-D Volume and Position Recovering Using a Virtual Reference Box. IEEE Transactions on Image Processing, Estados Unidos, vol. 16, no. 2, pp. 573-576, 2006.

[17] M. Rudek, Método de Posicionamento e Dimensionamento 3D Baseado em Imagens Digitais. PhD´s thesis, Depto. Engenharia Mecânica, Unicamp, Brazil, 2006.

[18] C. H. Q. Forster, (2004). Alinhamento Imagem-Modelo Baseado na Visão Estéreo de Regiões Planares Arbitrárias. PhD´s thesis, Faculdade de Engenharia Elétrica, Unicamp, Brazil.

[19] G.C. Bezerra, Localização de um Robô Móvel Usando Odometria e Marcos Naturais. Master´s Thesis, Depto. Eng. Elétrica, Universidade Federal do Rio Grande do Norte, Brazil, 2004.

[20] A. C. Victorino, Controle de Trajetória e Estabilização de Robôs Móveis Não-Holonômicos: Um Caso Experimental. Master´s thesis, Faculdade de Engenharia Mecânica, Unicamp, Brazil, 1998.

[21] G. Welch and G. Bishop. An Introduction to the Kalman Filter. http://www.cs.unc.edu/~{welch,gb}@cs.unc.edu, Dept. Of Computer Science Chapel Hill, NC 27599-3175, University of North Carolina, 2001.

[22] R. Negenborn, “Robot Localization and Kalman Filters”. Master´s Thesis, Institute of Information and Computing Sciences, Thesis no. INF/SCN-03-09, 2003.

Paulo R. G. Kurka was born in Brazil in September 1958. The author’s first degree and MsC in Mechanical Enginnering were granted in 1982 and 1985 respectively, by Pontificia Universidade Católica do Rio de Janeiro, Brazil. His PhD in Mechanical Engineering was granted by the University of Manchester (former UMIST) in 1989. He is presently an associated professor with the Faculty of Mechanical Engineering at Universidade Estadual de Campinas (UNICAMP) in São Paulo, Brazil. His main interests are image and signal processing and system dynamics.

Luciana C. M. F. Diogenes received the B. S. degree in 2001 and M. S. degree in 2004 both in Physics from Unicamp, Campinas, Brazil. She does Ph. D. in mechanics engineering from Unicamp, Campinas, Brazil . Her Ph. D. project is about the study a trajectory control strategy for a roving robot, using Kalman filtered estimates of pose from odometry and stereoscopic vision measurements.

Fernando Lobo Pereira graduated as an engineer and obtained the M.S. and Ph.D. degrees in electrical and computer engineering from Instituto Superior Técnico (IST), Technical University of Lisbon, Portugal, in 1985, 1988, and 1991, respectively. He is currently a Professor at the Electrical and Computer Engineering Department of IST. He is Member of the Editorial Board and Area Editor on Image/Video Compression of Signal Processing: Image Communication Journal, a Member of the IEEE Press Board, an Associate Editor of IEEE Transactions on Circuits and Systems for Video Technology, IEEE Transactions on Image Processing, and IEEE Transactions on Multimedia. He is an IEEE Distinguished Lecturer and a member of the scientific and program committees of tens of international conferences. He contributed more than 150 papers to journals and international conferences. He won the 1990 Portuguese IBM Award and an ISO Award for Outstanding Technical Contribution for his participation in the MPEG-4 Visual standard. For many years, he participates in the ISO/MPEG work, notably as the Portuguese Delegation Head, MPEG Requirements Group Chairman, and many MPEG-4 and MPEG-7 related ad hoc groups Chairman. Current areas of interest are video analysis, processing, coding and description, and multimedia interactive services.