UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ESTIMAÇÃO DE CURVAS POLINOMIAIS EM SEQÜÊNCIAS DE IMAGENS PARA NAVEGAÇÃO DE ROBÔS MÓVEIS

DEBORAH SILVA ALVES

ORIENTADOR: PROF. DR. GEOVANY ARAÚJO BORGES

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA

PUBLICAÇÃO:PPGENE.DM-278A/06

BRASÍLIA/DF: OUTUBRO - 2006

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ESTIMAÇÃO DE CURVAS POLINOMIAIS EM SEQÜÊNCIAS DE IMAGENS PARA NAVEGAÇÃO DE ROBÔS MÓVEIS

DEBORAH SILVA ALVES

DISSERTAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE.

APROVADA POR:

_____

Prof^{o}. Dr. Geovany Araújo Borges (ENE-UnB)
(Orientador)

_____

Prof^{o}. Dr. Prof. Adolfo Bauchspiess (ENE-UnB)
(Examinador Interno)

_____

Prof^{a}. Dr. Carla Cavalcante Koike, (CIC-UnB)
(Examinador Externo)

BRASÍLIA/DF, 24 DE OUTUBRO DE 2006.

FICHA CATALOGRÁFICA

ALVES, DEBORAH SILVA
Estimação de Curvas Polinomiais em Seqüências de Imagens para Navegação de Robôs Móveis[Distrito Federal]2006.
xvii, 92p., 297mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2006).
Dissertação de Mestrado- universidade de Brasília. Faculdade de Tecnologia.
Departamento de Engenharia Elétrica.
1.Estimação de Parâmetros de curvas. 2. Visão Computacional
3. Rastreamento de trajetória
I. ENE/FT/UnB II. Título(série)

REFERÊNCIA BILIOGRÁFICA

ALVES, D. S. (2006). Estimação de Curvas Polinomiais em Seqüências de Imagens para Navegação de Robôs Móveis. Dissertação de Mestrado em Engenharia Elétrica, Publicação PPGENE.DM-278A/06, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 92p.

CESSÃO DE DIREITOS

AUTOR: DEBORAH SILVA ALVES.
TÍTULO: Estimação de Curvas Polinomiais em Seqüências de Imagens para Navegação de Robôs Móveis.

GRAU: Mestre ANO: 2006

Deborah Silva Alves
Rua T-38 n^{o} 912 Ed. Cardeal, Setor Bueno.
74223-040 Goiânia - GO - Brasil.

AGRADECIMENTOS

Agradeço primeiramente a Deus, que renova minhas forças a cada manhã, o que foi fundamental para a conclusão deste trabalho.

Ao meu esposo que acreditando em meu sonho, contribuiu de todas as formas possíveis para sua realização com amor, apoio, dedicação, paciência e compreensão. Inclusive na minha mudança para Brasília, que fez com que separássemos fisicamente por um tempo.

Aos meus pais, pelo amor, apoio, paciência e compreensão demonstrados em todo esses anos de convivência. Por terem sempre acreditado em mim e com entusiasmo me incentivarem a seguir em frente. Esse trabalho é fruto de sua dedicação como pais.

Ao meu orientador, professor Geovany Araújo Borges, por aceitar me orientar nos "45 minutos do segundo tempo", pela oportunidade da realização desse trabalho, orientação, apoio, encorajamento e dedicação. Além da carga profissional adquirida por mim durante esse período, guardo profunda gratidão e admiração por seu exemplo de professor amigo, dedicado e presente.

Ao professor Adolfo Bauchspiess, por ser parte fundamental na realização desse sonho. Pela aceitação, incentivo, por ter lutado por minha bolsa de mestrado, amizade e confiança. Deixo minha profunda gratidão, admiração e respeito.

À toda minha família, irmãos, sogros, avós, tios e primos pelo amor, apoio e incentivo depositados em mim. Em especial, aos meus tios João Dantas e Fátima Dantas, por tudo o que já fizeram por mim e fazem até hoje. Por terem acompanhado e vibrado com todas as minhas conquistas. Essa, é de vocês também.

Agradeço também aos meus amigos e à todos que conheci na UnB durante esse período. Ao pessoal do Laboratório de Visão, Automação e Sistemas Inteligentes (bloco SG-11) pela disponibilidade, pela ajuda prestada e pela simpatia. Em especial obrigada Ronaldo, Flávio, Salvador, Daniela e Iwens pelo companheirismo e amizade durante o primeiro ano em Brasília. À todos os professores, sou grata pelos ensinamentos que serão tão úteis em minha vida acadêmica.

Ao meu querido esposo Márcio Giovane, pelo amor, companheirismo e dedicação.

RESUMO

ESTIMAÇÃO DE CURVAS POLINOMIAIS EM SEQÜÊNCIAS DE IMAGENS PARA NAVEGAÇÃO DE ROBÔS MÓVEIS

Autor: Deborah silva Alves.
Orientador: Geovany Araújo Borges.
Programa de Pós-graduação em Engenharia Elétrica
Brasília, outubro de 2006.

Resumo: Neste trabalho, foi implementado um sistema de visão computacional para navegação de robôs móveis baseado em rastreamento de trajetórias. Essas trajetórias são formadas por marcas artificiais adicionadas ao teto do ambiente na forma de curvas e retas. Para marcar o teto foram utilizadas cordas e fita adesiva escuras fixadas ao teto formando retas e curvas contínuas que indicam a trajetória a ser seguida pelo robô. A finalidade desse trabalho é o rastreamento de tais marcas. Para tanto, foi implementado um sistema de rastreamento que trabalha concorrentemente com o sistema de controle do robô.

O sistema implementado foi desenvolvido em um robô móvel equipado com uma câmera com foco direcionado para o teto. Esse sistema foi subdivido em quatro etapas que são executadas seqüencialmente a cada passo de tempo. A primeira das etapas adquire e processa uma imagem com técnicas para extração de bordas e eliminação de ruído. Em seguida, se o sistema não possui nenhuma informação *a priori* ele salta duas outras etapas e realiza a parametrização de uma curva baseado nos pixeis da imagem processada na primeira etapa. Essa é a quarta etapa do sistema para qual foram implementados cinco algoritmos estimadores, três baseados em Mínimos Quadrados e dois em filtragem estocástica. Caso contrário, se o sistema possuir informação a priori, após a primeira etapa ele realiza a segunda que é a de predição. Esta recebe os parâmetros da curva estimada no passo de tempo anterior e os dados da odometria, referentes ao movimento do robô entre o passo de tempo anterior e o atual, para atualizar a informação a priori sobre a trajetória. A próxima etapa recebe a imagem processada na etapa 1 e a informação de predição da etapa 2 e realiza uma pré-seleção de pixeis da imagem processada que poderão participar da estimação de parâmetros na quarta e última etapa do sistema.

Para a avaliação de desempenho do sistema implementado, foram utilizados bancos de imagens adquiridos por uma câmera com foco direcionado para o teto e acoplada a um robô móvel Omnidirecional guiado por *joystick*. O sistema foi avaliado para várias trajetórias, sendo analisados os resultados de todos os processos intermediários do rastreamento e o desempenho global do sistema. Os resultados observados mostraram que o sistema implementado permite o rastreamento de trajetórias a partir apenas da informação extraída, pelos algoritmos de visão, das imagens adquiridas durante a movimentação do robô e da odometria. Cada um dos algoritmos estimadores implementados foram testados e analisados quanto as suas características e tempo de execução, dentre os quais, os de filtragem estocástica obtiveram melhores resultados no rastreamento da trajetória.

ABSTRACT

POLINOMIAL CURVES ESTIMATION IN IMAGE SEQUENCES TO MOBILE ROBOT NAVIGATION

Author: Deborah silva Alves.
Supervisor: Geovany Araújo Borges.
Programa de Pós-graduação em Engenharia Elétrica
Brasília, outubro, 2006.

Abstract: In this work, a computer vision system was implemented for curve tracking-based navigation of mobile robots. These trajectories are formed by artificial landmarks added to the ceiling of the environment in the form of curves and straight lines. In order to mark the ceiling, ropes and adhesive tapes were fixed fixed to the ceiling forming continuous straight lines and curves that indicate the trajectory to be followed by the robot. The purpose of this work is to propose tracking algorithms for such marks. Therefore , it was implemented a tracking system that works concurrently with the control system of the robot.

The implemented system was developed in a mobile robot equipped with a video camera looking at the ceiling. This system is composed of four stages that are executed sequentially. The first one of the stages acquires and processes an image with techniques for edges extraction and of noise elimination. After this, if the system does not possess no *a priori* information about the trajectory, it jumps two stages and carries through the parametrization of a curve based on pixels of the processed image at the first stage. This is the fourth stage of the system, for which five estimators were implemented. Three of them are based on Least Squares and two in stochastic filtering. Otherwise, if the system has *a priori* information, after the first stage it carries through the prediction stage. This stage receives the curve estimated parameters computed in the previous time and robot's odometric data, referring to the robot's movement between the step of previous time and the current one. It is used to update the *a priori* information about the trajectory. The next stage receives the processed image (first stage) and information from prediction (second stage) and carries through an previous election of pixels of the processed image.

Several images sequences had been acquired for performance evaluation of the implemented system off line. In such experiments, the mobile robot was guided using joystick. The system was evaluated for several trajectories. The results were analyzed for all the intermediate processes. The observed results had shown that the implemented system allows the trajectory tracking only with the extracted information of the images (through vision algorithms) acquired during the robot's movement and the odometry. Each one of the estimators implemented algorithms had been tested, and their characteristics and execution time were analyzed. From such analysis, the stochastic ones had gotten better results in the tracking trajectory process.

Sumário

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SíMBOLOS

Símbolos Latinos

t	tempo	
v	Coordenada afim de um pixel (linha) na imagem	pixeis
u	Coordenada afim de um pixel (coluna) na imagem	pixeis
f	Distância focal	
D	Coeficientes de mudança de escala	
v_0	Coordenada do centro da imagem	pixeis
u_0	Coordenada do centro da imagem	pixeis
$\mathcal{I}$	Matriz da Imagem	
d	Distância da câmera ao teto do ambiente	metros
r	Raio	metros
e	Distância entre o eixo de orientação do suporte da roda do robô e seu centro de rotação	
(l_i, α)	Coordenadas polares de um ponto	
a	Coeficientes da equação da curva	

Símbolos Gregos

Δ	Variação entre duas grandezas similares
θ	Ângulo formado entre dois eixos quaisquer
ξ	Postura
β	Ângulo de direção de uma roda do robô
ϕ	Posição angular de uma roda do robô Omni
ϑ	Parâmetros geométricos do modelo cinemático do robô Omni
λ	Vetor de parâmetros da equação curva
σ	Vetor direção dos pixeis da curva
φ	Vetor de coeficientes da equação da curva

ε	Resíduo
ψ	Função de influência
ω	Função Peso
ρ	Função que minimiza o incremento da soma dos quadrados dos resíduos

Grupos Adimensionais

$\{M\}$	Sistema de Coordenadas do Mundo
$\{R\}$	Sistema de Coordenadas do Robô
$\{C\}$	Sistema de Coordenadas da Câmera
O	Origem de um sistema de coordenadas
(x, y, z)	Coordenadas de um ponto
$\mathcal{T}$	Matriz de transformação homogênea
k	Passo de tempo
l, c	Escalares
N	Quantidade de amostras selecionadas
M	Quantidade total de amostras disponíveis
N_{iter}	Quantidade total de iterações
n	Número da Iteração
h	Valor de tolerância do algoritmo RANSAC
w	Medida
s	
P	Ponto observado
R	Matriz de Rotação
t	Vetor de Translação
p	Coordenadas de um ponto P
K	Matriz de parâmetros intrínsecos da câmera
q	Variáveis de configuração dos parâmetros do modelo cinemático do robô Omni

J	Matriz Jacobiana
P e **S**	Matrizes de Covariâncias
I	Matriz Identidade
G	Ganho de Kalman

Subscritos

k	Passo de tempo
$k\|k-1$	Passo de tempo k dada a informação do passo $k-1$
a	Indicação de coordenadas das rodas do robô
M, R, C	Sistemas de coordenadas do mundo, robô ou câmera

Sobrescritos

-1	Inversa de uma matriz
†	Matriz pseudoinversa de Moore_Penrose
$\cdot$	Valor Absoluto
$\hat{}$	Estimação
$\vec{}$	Vetor unitário
$'$	Instâncias
T	Transposta de uma matriz
M, R, C	Sistemas de coordenadas do mundo, robô ou câmera
A	Sistema de coordenadas genérico
B	Sistema de coordenadas genérico

Siglas

SNR,	Razão sinal ruído
MAD,	Mediana dos desvios absolutos dos resíduos

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Sistemas de visão computacional vêm sendo amplamente utilizados para diversas tarefas relacionadas a robôs móveis, dentre elas podem ser citadas a navegação e a localização. Um exemplo é a identificação da trajetória a partir de imagens capturas por câmeras acopladas na estrutura do robô, denominado de rastreamento da trajetória.

O rastreamento de trajetórias é uma aplicação comum da visão computacional para o controle de robôs móveis. Tal procedimento parte de uma informação visual para que o robô siga uma seqüência de posições especificada durante o processo, seja por um alvo móvel rastreado ponto a ponto, seja por uma marca natural ou artificialmente adicionada ao ambiente representando um alvo de referência para o sistema de rastreamento.

Grande parte das abordagens para navegação e localização de robôs móveis é baseada no rastreamento de marcas visuais [18] que podem ser reconhecidas mesmo sob condições adversas. Muitas das aplicações desenvolvidas para ambientes fechados como indústria e laboratórios com uso de marcas artificiais têm obtido significativo avanço, tendo em vista que o controle das condições do ambiente e tarefas realizadas é bem definido.

1.2 DEFINIÇÃO DO PROBLEMA

A pesquisa a ser relatada nessa dissertação aborda o problema de rastreamento de trajetórias formadas a partir de marcas artificiais fixadas ao teto do ambiente. Têm-se um ambiente fechado no qual fixam-se marcas artificiais contínuas no teto, formando retas e curvas que representam uma trajetória. Um robô móvel equipado com uma câmera com foco direcionado para o teto captura imagens. Tais imagens contém, além de informações sobre o ambiente, informações sobre a trajetória a ser percorrida pelo robô. O problema a ser solucionado consiste no rastreamento dessa trajetória.

1.3 OBJETIVOS DO PROJETO

O trabalho aqui apresentado, tem como objetivo propor uma arquitetura de rastreamento de trajetórias para a navegação de robôs móveis. A solução implementada é divida em etapas para a identificação da trajetória.

Cada uma das etapas da arquitetura proposta teve objetivos específicos. Tais como:

- Conhecer a literatura a respeito de rastreamento de marcas e dentre elas as marcas artificiais para navegação de robôs móveis;
- Estudar e implementar técnicas de processamento de imagens para extrair da imagem pixeis que representem a trajetória;
- Estudar e implementar um mecanismo para atualizar dados de imagens baseados no movimento do robô;
- Estudar e implementar uma forma de realizar correspondência e seleção de pixeis candidatos para estimar a trajetória;
- O rastreamento da trajetória foi realizado através de estimação de parâmetros de curvas formadas pela trajetória. Para tal, foram estudados, implementados e testados cinco algoritmos estimadores: mínimos quadrados, M-estimadores, RANSAC, Filtro de Kalman e Filtro de Kalman Estendido Iterativo Robusto.

A arquitetura proposta nesse trabalho fundamenta-se nos pixeis da imagem que pertencem à curva que representa a trajetória a ser seguida pelo robô. Nesse sentido, os algoritmos estimadores implementados têm importância fundamental nesta pesquisa, pois são responsáveis por gerar um modelo que mais se aproxima da trajetória desejada através dos pixeis selecionados nas imagens capturadas.

1.4 ESTRUTURA DA DISSERTAÇÃO

A dissertação está organizada em seis capítulos. No capítulo 2, é apresentada uma revisão bibliográfica sobre o assunto abordado nessa dissertação com destaque para o rastreamento de marcas para navegação de robôs móveis.

No capítulo 3 é apresentado o robô móvel utilizado para a realização dos experimentos. É abordado de forma resumida sua arquitetura e configuração, bem como a configuração adotada para os experimentos. São mostradas também as transformações dos sistemas de coordenadas adotados necessários ao processo de rastreamento da trajetória do robô.

O capítulo 4 trata da arquitetura do sistema desenvolvido para o problema de rastreamento da trajetória formada através de marcas artificiais que formam curvas e retas. Tal capítulo descreve as quatro etapas envolvidas e as técnicas empregadas como solução. A etapa de aquisição e processamento de imagens apresenta as técnicas utilizadas para a remoção de ruído nas imagens que interfere grandemente na estimação dos parâmetros da curva da trajetória. A etapa de predição dos parâmetros da curva realiza uma atualização temporal da estimativa obtida no passo de tempo anterior e que será utilizada como informação para a estimação da curva do próximo passo de tempo, tal predição é realizada pelo uso dos dados da odometria passados pelo sistema de controle do robô Na etapa de correspondência de pixeis é apresentada uma técnica para seleção de pixeis da imagem candidatos à estimação na próxima etapa. A quarta e última etapa do processo é descrita em detalhes na última seção desse capítulo. Nessa etapa é proposta a realização da estimação dos parâmetros do modelo que representa a trajetória do robô por algoritmos separados em duas abordagens: a não-seqüencial ou em lote e a seqüencial. Na primeira são apresentados três algoritmos baseados no estimador de mínimos quadrados e na segunda abordagem, dois algoritmos baseados em Filtro de Kalman.

Finalmente, o capítulo 5 apresenta os resultados dos testes realizados. São apresentadas duas das trajetórias testadas, bem como os resultados obtidos em cada etapa do processo. Para a etapa de aquisição e processamento de imagens foram apresentados os resultados para imagens adquiridas em ambientes com características diferentes comprovando assim o funcionamento do sistema em casos opostos. Para as etapas de predição de parâmetros da curva, correspondência de pixeis e estimação dos parâmetros da curva, são apresentados e analisados os resultados de

seqüencias de imagens de uma mesma trajetória.

As conclusões sobre o trabalho realizado e sugestões para trabalhos futuros são apresentadas no capítulo 6.

2 REVISÃO BIBLIOGRÁFICA

2.1 INTRODUÇÃO

Visão é o sentido mais poderoso e complexo [26] que os seres vivos possuem. Ele provê uma extraordinária quantidade de informação sobre o que está em volta e habilita os seres que o possuem a interagir de maneira inteligente como o ambiente. Através da visão pode-se aprender as posições, identificar objetos e a relação entre eles. A aplicação de visão em robôs móveis para navegação autônoma é objeto desse estudo.

Desde os anos 60 [1], robôs móveis autônomos são assunto de pesquisa por cientistas de todo mundo. Durante esse período foram desenvolvidas várias técnicas que habilitam robôs a navegarem de forma satisfatória dentro de seus respectivos ambientes, usando diferentes tipos de sensores [67]. Esses robôs móveis são conhecidos por atuarem em ambientes que mudam continuamente de configuração. Quando autônomos, devem ter condições de realizar atividades sem intervenção humana.

A criação de veículos autônomos capazes de tomarem decisões com o conhecimento do ambiente no qual se encontram é tarefa da robótica móvel. Isso é possível quando a plataforma de um robô possui capacidade operacional proporcionada pela estrutura eletromecânica e pelos sistemas de controle e tomada de decisão que formam sua arquitetura.

A arquitetura do robô móvel depende da tarefa que será por ele executada e do tipo de ambiente no qual está imerso, seja estruturado ou não, interno ou externo. Em ambientes estruturados, existem restrições geométricas, como retas e planos, que podem ser parametrizadas ou utilizadas na construção de modelos ou mapas. Ambientes internos são limitados por paredes e geralmente podem ser modelados como salas, corredores, galpões, já os externos são difíceis de modelar como o fundo do mar e áreas ao ar livre. A definição do tipo de ambiente é importante na elaboração de algoritmos para a construção de mapas e no projeto da estrutura mecânica do robô que deve ser flexível no caso de ambientes inicialmente desconhecidos.

[1] Relatório técnico on-line sobre o robô Shakey no Artificial Intelligence Center (www.ai.sri.com/shakey).

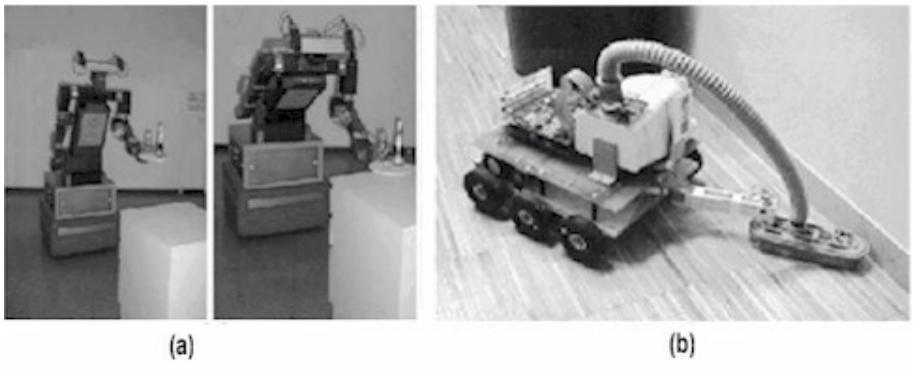


Figura 2.1: (a) Seqüencia de movimento em que HERMES serve uma pessoa. (b) *Vaccum Clear*.

Robôs móveis podem ser utilizados para muitas aplicações na área de transporte [2], limpeza [64], manutenção [44] e exploração [63]. Embora muitos não sejam totalmente autônomos devido à necessidade de intervenção humana para configuração e ajuste de sensores, construção de mapas de navegação e modelos de ambiente, grande parte desses sistemas requer que o robô se ajuste automaticamente [2], isto é, que tenha habilidade para realizar missões navegacionais, explorações de ambiente e alcance de objetivos sem colisões [18], [62].

Alguns robôs móveis podem ser visualizados nas Figuras 2.1 e 2.2. Na Figura 2.1 (a) é apresentado o robô HERMES [5], um humanóide desenvolvido para serviços domésticos, que possui dois braços com dois dedos cada e duas câmeras, é autônomo ou pode ser guiado. Em (b), *Vaccum Clear*, um pequeno robô que aspira a casa [64], ambos atuando em ambientes internos. Na Figura 2.2 são mostrados robôs móveis para ambientes externos, em (a) um veículo submarino autônomo para inspecionar oleodutos e cabos submarinos, realiza serviços de manutenção [44]. Esses robôs que navegam em ambientes externos utilizam geralmente sensores típicos para localização e mapeamento local do ambiente como GPS *(Global Positioning System)*, câmeras de vídeo e sensores de ultra-som.

Robôs móveis também são utilizados no meio científico, como por exemplo na exploração espacial onde o ambiente é não estruturado e externo. Na Figura 2.2 (a) é mostrado o robô *Sojourner*, desenvolvido para exploração de superfície planetária pelo *Jet Propulsion Laboratory Mars Rover* e que foi enviado em missão ao planeta Marte em 1996 [45]. Em (b) o *Spirit* que está

[2] http://www.robosoft.fr.

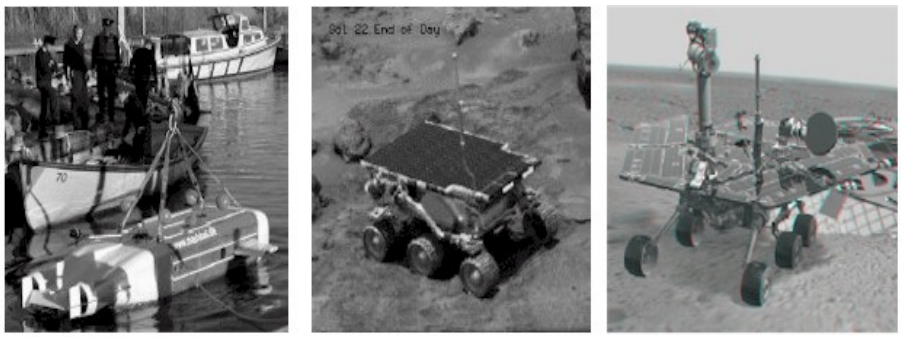


Figura 2.2: (a) *Martin*, desenvolvido pela *Maridan ApS* destinado a inspeção de oleodutos e cabos submarinos. Veículos de exploração enviados ao planeta Marte: (b) *Sojourner* e (c) *Spirit*.

em Marte desde janeiro de 2004 [63] e que interage diariamente com cientistas enviando fotos de regiões desconhecidas do planeta .

Em aplicações industriais, onde o ambiente é parcialmente conhecido, controlado e onde a tarefa executada pelo robô é relativamente simples, o progresso é maior. Já em atividades que envolvem movimento autônomo de robôs principalmente em ambiente desconhecidos [26] o avanço é mais lento.

Como mostrado anteriormente, existem projetos utilizando robôs móveis para solucionar vários tipos de problemas. Muitos deles utilizam sensores simples para locomoção, como os de medição de distância por ultra-som [32] [59], outros utilizam-se de câmeras para navegação visual, mas são tipicamente restritivos devido à complexidade das cenas visualizadas [67].

2.2 SISTEMA DE VISÃO PARA ROBÔS MÓVEIS

Sistemas de visão para robôs móveis analisam imagens e produzem descrições sobre o que está sendo visualizado [26]. Essas descrições devem capturar os aspectos dos objetos ou marcas que serão úteis na realização de alguma tarefa. Entretanto, considera-se que o sistema de visão é parte de uma entidade maior que interage com o ambiente por estar relacionado com percepção, enquanto outros elementos são dedicados a tomada de decisão e a implementação dessas decisões.

A cena visualizada pela câmera traz informações importantes para as tarefas a serem executadas, em se tratando do uso de visão para navegação de robôs móveis, elas devem indicar o movimento a ser realizado pelo robô de forma a seguir sua trajetória.

Uma das abordagens mais importantes para navegação de robôs móveis com visão computacional é a baseada em marcas visuais [18]. Tais marcas podem ser artificiais ou naturais e devem ser rapidamente e confiavelmente detectadas, mesmo sob condições de visão adversas como baixa luminosidade, posicionamento e obstrução parcial.

Nesse trabalho será apresentada uma solução para navegação de robô móvel usando visão computacional. A navegação será baseada no rastreamento de marcas artificiais, que formam curvas polinomiais fixadas ao teto do ambiente interno.

2.3 RASTREAMENTO DE MARCAS PARA NAVEGAÇÃO DE ROBÔS COM VISÃO

O rastreamento de marcas para localização e navegação de robôs móveis pode ser realizado por diversas formas, algumas delas são através do uso de marcas artificiais, naturais ou objetos que se movimentam.

Marcas artificiais são adicionadas ao ambiente e podem ser entre outros, códigos de barras [12], figuras tridimensionais planas e multicoloridas [33] e cilindros multi-coloridos [27]. Marcas naturais são as características que podem ser extraídas do próprio ambiente [62] [3].

Vários trabalhos consultados trazem diferentes soluções para o problema de rastreamento de marcas e objetos que se movimentam, tanto para implementação na localização e navegação de robôs móveis, quanto para o uso em outras áreas. Nas próximas subseções serão apresentadas algumas dessas soluções.

2.3.1 Rastreamento de Marcas Naturais

Algoritmos para rastreamento de marcas artificiais mostram-se interessantes como solução para o problema de navegação e localização de robôs móveis mas, abordagens que utilizem mar-

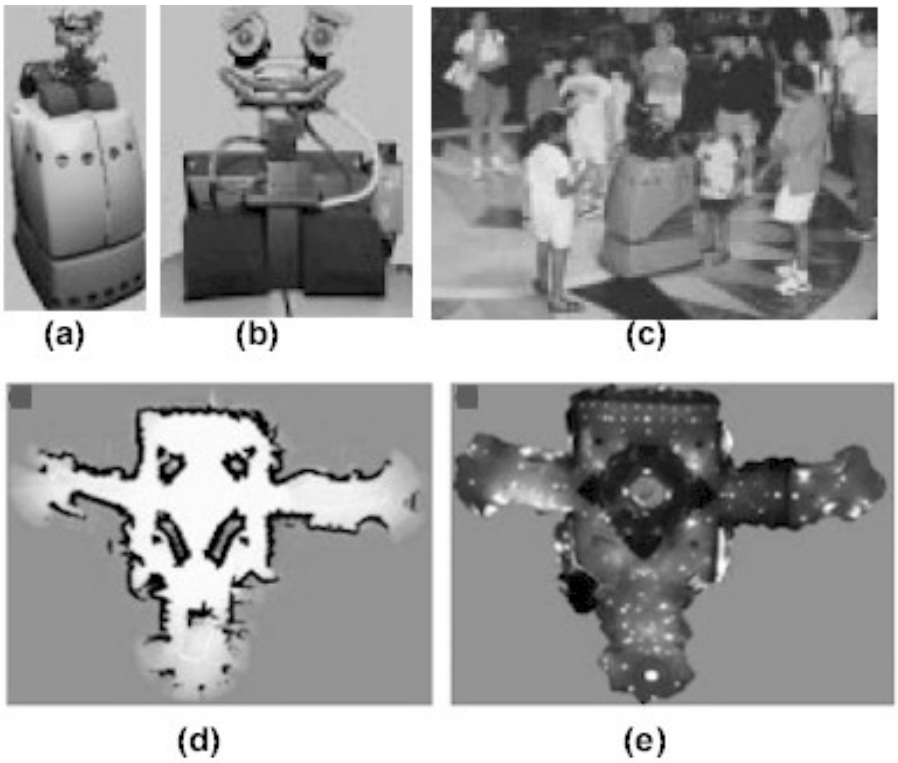


Figura 2.3: Imagens retiradas de [62]:(a) Robô Minerva, (b) Face motorizada , (c) Minerva em um *tour* pelo *Smithsonian* - Museu Nacional da História Americana, (d) Mapa de ocupação da parte central do Museu e (e) Mosaico do teto do Museu.

cas naturais do ambiente são desejáveis para vários tipos de aplicações [55].

Características extraídas do teto, chão ou paredes do ambiente, podem ser utilizadas como marcas naturais para localização e confecção de mapas. O robô MINERVA [62] [13], realiza a tarefa de guia turístico para um museu, Fig. 2.3 (a), (b) e (c), seu sistema utiliza imagens de mosaicos do teto, Figura 2.3 (e), para localização em adição ao mapa construído pelo rastreamento a laser. Para aprender o mapa de ocupação, MINERVA utiliza um algoritmo que procura por um mapa mais provável considerando simultaneamente as localizações anteriores fornecidas pelo sonar.

Um outro algoritmo, apresentado em [56], aprende características visuais naturais para estimação de posição. O sistema rastreia as marcas e verifica se a mesma é parte de um conjunto de mini-imagens detectadas na fase de aprendizado. O casamento de características se dá através de análise de componentes principais.

Marcas naturais do ambiente também podem ser usadas para rastreamento de posição baseado em Filtro de Kalman e características geométricas [3]. Tal solução é uma técnica poderosa de localização com várias propriedades desejáveis, pois trabalha com representações minimalistas do ambiente. Tal solução mostra-se robusta quanto a dinâmica do ambiente, e combina precisão na localização com implementações de processamento mais leve. Esse sistema reage diretamente no ambiente, uma vez que as características contribuem para a formação de hipóteses de localização do robô. Uma característica, nesse caso, é uma primitiva geométrica contendo no mínimo uma medida como ângulo, posição *(x,y)* ou postura (x, y, θ). As hipóteses são geradas a partir de métodos de busca baseada em restrições, nos quais os nós da árvore de busca são possíveis pares de localização local-global. Para o rastreamento é utilizada a mesma técnica de busca baseada em restrições.

2.3.2 Rastreamento de Objetos, silhuetas e curvas em Movimento

Existem métodos, desenvolvidos pela comunidade de visão computacional, que realizam o rastreamento de objetos ou formas geométricas que se movimentam. Esses métodos demonstram grande potencial para aplicações gráficas de tempo real [6] e permitem que entidades gráficas, tais como curvas, sejam superpostas sobre o frame de vídeo para marcar certos objetos e perseguir sua movimentação.

Uma solução para estimação de posição de um robô equipado com uma câmera CCD é apresentada em [37]. A estimação é construída a partir do reconhecimento de características topográficas ou de um objeto e da comparação dessa característica ou objeto com uma imagem modelo armazenada anteriormente. Sua desvantagem é a velocidade de processamento baixa para realizar casamentos de objetos [35].

Estimação de posição é uma das mais importantes funções para navegação de robôs móveis em ambientes não estruturados [35]. A maioria dos sistemas de localização absoluta estima a posição atual do robô móvel aplicando algoritmos de localização, que utilizam informações obtidas a partir de sensores ou através do reconhecimento de uma marca artificial anexada ao ambiente ou de objetos do próprio ambiente. Para amenizar várias desvantagens desse modelo, [35] propõe, além da câmera fixada ao robô, anexar mais uma câmera CCD no teto do ambiente-

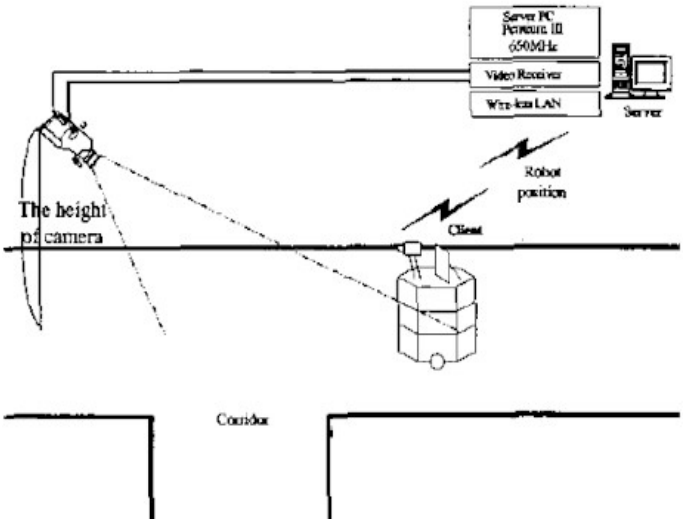


Figura 2.4: Modelo utilizado por [35] para estimação de posição do robô em um ambiente com câmera.

corredor, Figura 2.4. Essa câmera do teto auxiliará na localização do robô em movimento no ambiente. O sistema reconhecerá se o objeto capturado na imagem da câmera fixada ao teto é ou não o robô, se for, o sistema obtém sua posição e a transmite a ele.

Algumas soluções para rastreamento de curvas, silhuetas ou formas geométricas em movimento são aplicadas em outras áreas. Tais soluções serão citadas pelo fato de tratarem sobre rastreamento de curvas, objeto de estudo desse trabalho. Um exemplo é caso de rastreamento visual de contorno, no qual se persegue o movimento de silhuetas [7], [41] e características de superfícies. Tal rastreamento é aplicado com uso de vários tipos de objetos e formas em movimento como: lábios, mãos, veículos e outros em inúmeras situações como análise de imagens biomédicas [48] [4], segurança [68] e outros.

Nessa área, métodos baseados em modelos probabilísticos são tópicos de pesquisa ativos [29], e muitos desses utilizam filtragem Kalman [50]. Como no mundo real a maioria dos modelos são não-lineares, o que é problemático para o filtro de Kalman linear, desenvolveu-se um algoritmo de amostragem seqüencial Monte Carlo chamado *CONDENSATION* [31] que lida com rastreamento de contorno não-linear e não gaussiano de forma unificada, mas que demanda um número grande de amostras discretas. Na Figura 2.5 pode-se observar o rastreamento do contorno de uma mão através da aplicação desse algoritmo.

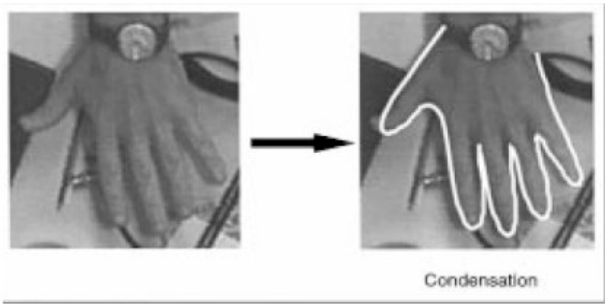


Figura 2.5: Rastreamento de contorno de mão [31].

Figura 2.6: Rastreamento de contorno de cabeça em movimento pelo algoritmo de rastreamento baseado em filtro de Kalman descentralizado [41].

O rastreamento visual de contorno em imagens que possuem fundo complexo é uma tarefa difícil [41]. Devido à não-linearidade do modelo de medição causada pela grande quantidade de ruído na imagem, o filtro de Kalman linear entra em colapso [31]. Em [41] é apresentado um algoritmo de rastreamento de curvas baseado no filtro de Kalman descentralizado que aplica um modelo de medida não-linear. Durante cada passo de tempo, o rastreador realiza múltiplas medições, em termos do conjunto de pontos apropriadamente escolhidos, obtendo assim a melhor observação de acordo com a densidade medida. O algoritmo resultante é apto a obter uma estimação mais exata do estado do sistema. Na figura 2.6 pode-se observar o rastreamento de cabeça em movimento, realizado pelo algoritmo baseado em filtro de Kalman descentralizado.

2.3.3 Rastreamento de Marcas Artificiais

Marcas artificiais são muito úteis como ferramenta para auto-localização em ambientes fechados [33] e para rastreamento. Por serem simples e possuírem uma forma determinada, são efi-

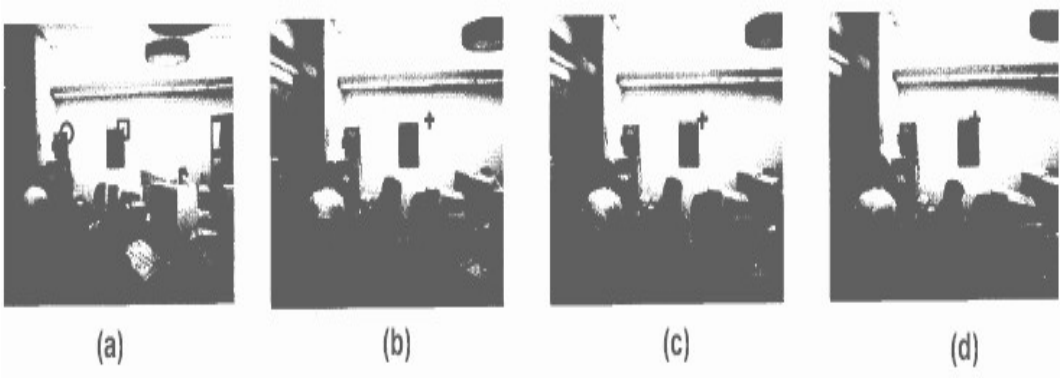


Figura 2.7: Seqüencia de imagens para o reconhecimento de uma das marcas (bola ou quadrado em (a) e o reconhecimento indicado por uma cruz em (d)[14].

cientes e rapidamente capturadas pelo sistema quando bem projetadas [12]. Sendo assim, são uma outra maneira de estimar localização e navegação para robôs móveis.

As marcas artificiais podem ter infinitas formas, em [23] são propostas marcas baseadas em códigos de barras ou padrões de forma específica para navegação de robô móvel. Nesse, o robô observa uma grande área com a finalidade de encontrar marcas úteis para definir o caminho a ser percorrido. Uma característica importante dessa arquitetura é a habilidade de prever e adquirir marcas à medida que as mesmas aparecem no campo de visão do robô. A predição é feita baseada em informações adquiridas através de imagens disponíveis. Na Figura 2.7, é apresentada parte de uma seqüência de imagens que mostram o reconhecimento de uma das marcas, bola ou quadrado 2.7(a). A marca reconhecida é indicada por uma cruz na imagem 2.7(d).

Códigos de barras únicos anexados às paredes do ambiente de navegação não modelado do robô, também são utilizados por [12]. As marcas auxiliam o sistema de navegação na construção de um mapa topológico o qual é construído à medida que o robô explora o ambiente. Como o código de barras é uma marcação simples e fácil de ser detectada pelo sistema de visão, o algoritmo torna-se confiável e pode ser utilizado em tempo real.

Outra solução simples para o problema de rastreamento de marcas artificiais para navegação de robôs móveis em ambientes internos, trata-se de um algoritmo robusto e estocástico [33] baseado no algoritmo *CONDENSATION* [31]. Esse algoritmo rastreia a marca usando a distribuição de cores do padrão. A marca artificial projetada, mostrada na Figura 2.8 (e), tem es-

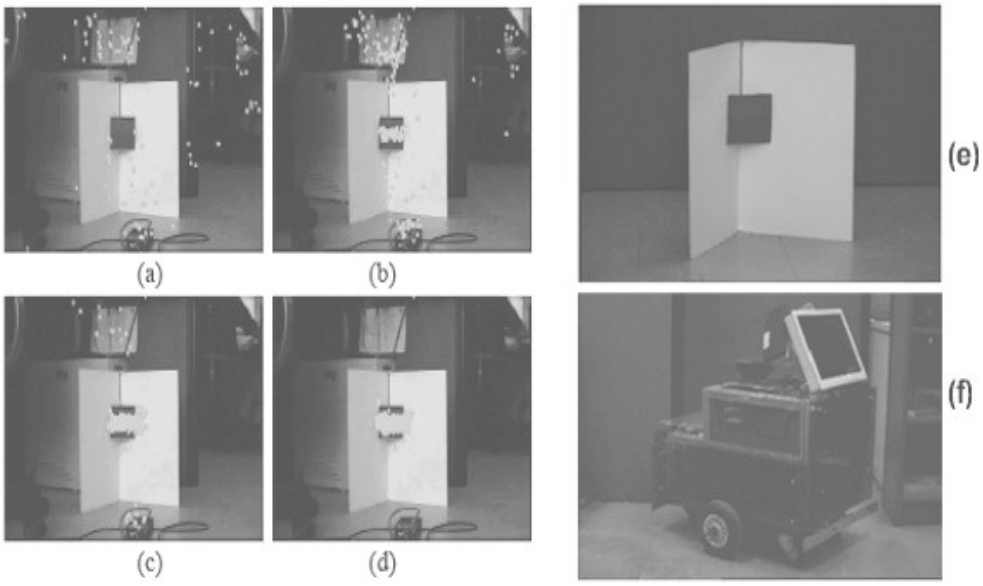


Figura 2.8: Rastreamento de marca pelo robô [33]. (a-d) Seqüencia de detecção da marca, (e) marca confeccionada e (f) robô utilizado no experimento.

trutura tridimensional que consiste em um padrão plano e multicolorido. Tal modelo mostrou-se muito eficiente em ambiente interno e em tempo real nos testes realizados pelos pesquisadores, Figura 2.8. O algoritmo inicialmente espalha as amostras de pontos na imagem (a), cada amostra tem uma probabilidade de representar similaridade ao modelo da marca. Em (d), o resultado, mostrando que todas as amostras se concentraram na região da marca. O robô utilizado nessa solução é mostrado em (f).

2.4 RASTREAMENTO DE CURVAS

Curvas são utilizadas para extração de características de imagens aplicadas a processamento de imagens médicas, segmentação e rastreamento [8]. Um exemplo é o uso de modelos como contorno ativo, originalmente proposto por Kass, Witkin e Terzopoulos [36]. Tal modelo utiliza-se de uma curva de minimização de energia para extrair características importantes de uma imagem. A energia associada à curva é definida de forma que seja mínima quando a curva se encontrar sobre uma região com as características que se deseja extrair. O contorno ativo foi proposto

em 1988 e desde então é aplicado em uma variedade de problemas em visão computacional e análise de imagem, como detecção de borda [36], modelagem de formas [47], segmentação [17] e rastreamento de movimento [40].

Uma das abordagens de contorno ativo adota parametrização de curvas. Curvas paramétricas são atrativas porque são capazes de representar eficientemente conjuntos de curvas limitadas na imagem [8]. Tanto formas simples como formas mais complexas podem ser representadas por curvas bastando alterar a ordem do polinômio que representa a curva. Geralmente a ordem do polinômio é fixada em quadrática ou cúbica. A manutenção de uma ordem baixa para o polinômio, mesmo face a complexidade geométrica, conduz à estabilidade e simplicidade computacional.

Nesse trabalho será adotada a parametrização de curvas aplicada ao rastreamento de trajetória para navegação de robôs móveis. Marcas artificiais fixadas ao teto do ambiente formando curvas serão rastreadas compondo assim a trajetória do robô.

O rastreamento de curvas fixas ao teto pode ser utilizado para navegação de robôs empregados em várias situações como por exemplo, guias turísticos, bibliotecas, hospitais, indústrias, ou até mesmo para realização de transporte de mercadorias em distribuidoras, indústrias e outros. Esse rastreamento é tratado por vários autores na literatura, mas em sua maioria, sua aplicação não está ligada diretamente a navegação de robôs móveis. Alguns desses trabalhos relacionados à parametrização e rastreamento de curvas serão citados.

Tarefas como reconhecimento de objetos usando curvas e silhuetas e estimação de posição em visão computacional, possuem uma literatura considerável. Existem várias soluções que podem ser consultadas como baseadas em momentos, *B-splines*, superquadráticas, cônicas, invariantes diferenciais e descritores de Fourier [60],[28], [57], [43], [49], [15].

Soluções baseadas em formas que são representadas parametricamente também são amplamente estudadas, e seu uso permanece dominante em computação gráfica, visão computacional e modelagem geométrica [69]. Uma representação paramétrica denominada Descritores Paramétricos de Fourier(EFD), é amplamente usada para representação de curvas 2D e 3D. Embora possam representar uma ampla quantidade de curvas, é conveniente ter uma descrição algébrica implícita da forma $f(x, y) = 0$, por várias razões discutidas em [69], que apresenta uma técnica de im-

plicitação não simbólica chamada de método de aniquilação de matriz(*matrix annihilation*) para converter representações EFD em algébricas (polinômio implícito). Outro trabalho que trata sobre parametrização de curvas, [52], propõe um método de parametrização quase-invariante usado especialmente em correspondência de curvas em movimento, tal método mostrou-se menos sensível à ruído do que métodos de parametrização totalmente invariante.

Além de representações paramétricas, modelos algébricos que possuem vantagens matemáticas e computacionais têm recebido atenção crescente, pois superfícies e curvas algébricas provam ser muito úteis na representação de formas [61], [9]. Invariantes associadas a modelos algébricos tem sido utilizados em várias aplicações com modelos baseados em visão e reconhecimento de padrões [65].

Vários outros artigos/trabalhos tratam de solucionar problemas referentes à parametrização de curvas [52], geometria e correspondência de curvas [53], classificação de silhuetas [21] e outros em seqüência de imagens. Mas, poucos utilizam técnicas de rastreamento de curvas para obtenção de trajetória para navegação de robôs móveis, desses podem ser citados [20], que analisa o problema de seguir um contorno desconhecido com um veículo não-holonômico e [51] que descreve um método para o uso de curvas cúbicas para determinar uma única trajetória para uma postura alvo arbitrária.

3 ROBÔ MÓVEL OMNI

3.1 INTRODUÇÃO

O sistema de rastreamento de curvas para navegação de robôs móveis proposto nesse trabalho, foi validado no robô móvel Omni, Fig. 3.1. Nesse capítulo será apresentada de forma sucinta um pouco da arquitetura do robô Omni. Maiores detalhes podem ser encontrados em [10] [38] [1].

O robô móvel Omni foi desenvolvido no LIRMM (*Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier*), em Montpellier na França, para ser utilizado em pesquisas de robótica móvel [38]. Sua plataforma sofreu modificações na estrutura mecânica e eletrônica até alcançar sua configuração atual apresentada na Figura 3.1. As dimensões físicas do robô Omni são: 860mm de comprimento, 725mm de largura e 760mm de altura distribuídos em 350Kg. Possui três rodas orientáveis, não centradas e tracionáveis de forma independente, o que resulta em seis eixos articulados, sendo então classificado como omnidirecional. Possui três atuadores e eixos articulares controlados por um motor. A redundância de atuadores e a configuração mecânica de suas rodas (descentralizadas e orientáveis ou do tipo "Castor") fazem com que menos restrições sejam impostas na planificação de trajetórias, permitindo plena mobilidade em um plano [1].

O controle do robô é feito por um micro computador IBM-PC Pentium TM II 300 MHz e placas *transputer* que servem de interface entre o computador e certos sensores. Sua arquitetura lógica foi desenvolvida no *Windows 2000*© [1] com extensão tempo real *RTX*© [2]. A interação entre o usuário e o Omni se dá através de periféricos, tais como: monitor LCD, teclado, mouse e joystick.

A plataforma do robô Omni é dotada de vários sensores. As subseções seguintes apresentam alguns dos componentes que equipam o robô Omni.

[1] Windows 2000 é uma marca registrada da Microsoft, Inc

[2] RTX é uma marca registrada da Venturcom, Inc

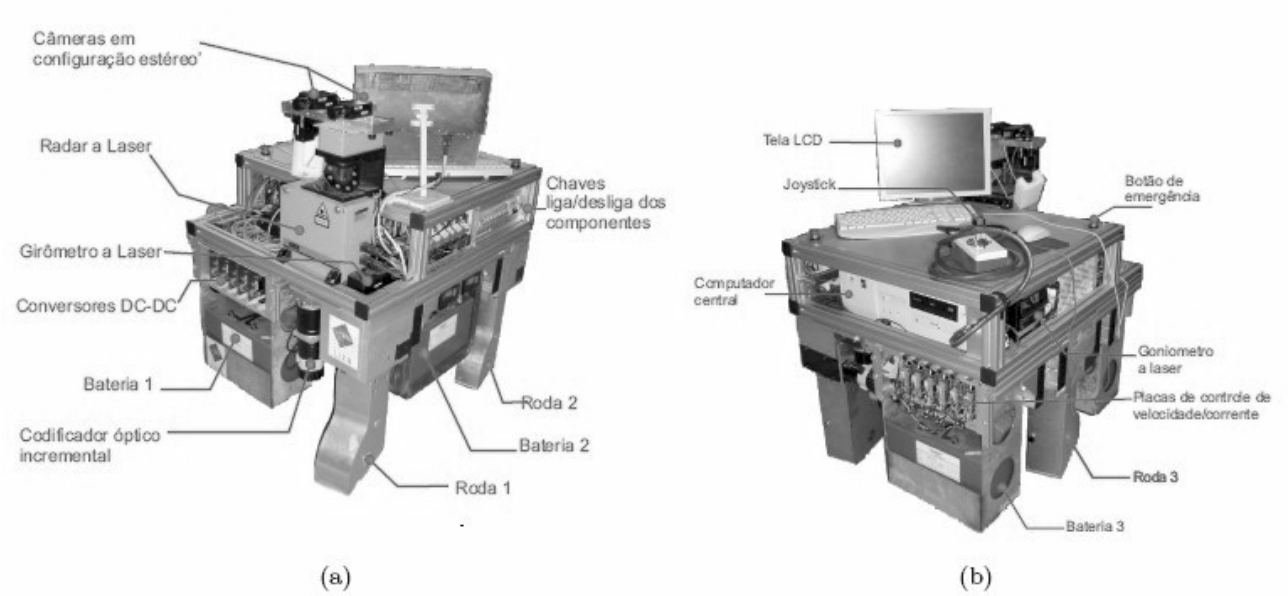


Figura 3.1: Plataforma móvel do robô Omni e seus componentes.

3.2 ARQUITETURA DO ROBÔ

A plataforma móvel é equipada com vários sensores os quais são utilizados para estimação relativa do veículo por Odometria (denominados sensores proprioceptivos) ou para localização absoluta do veículo no ambiente (denominados sensores exteroceptivos). Todos os sensores, atuadores, circuitos e computador central são alimentados por quatro baterias de 12V conectadas em série, fornecendo 48V ao sistema. Como nem todos os equipamentos são alimentados por esta tensão, conversores CC - CC são necessários para fornecer a tensão necessária para cada equipamento.

Cada eixo articulado do robô Omni é acionado por um motor de corrente contínua, isto é, cada roda é controlada por dois motores: um para tração e outro para direção. Tais motores são alimentados por 24V providos da saída de um conversor CC - CC que fornece um torque de até 6.5Kg - cm. O acionamento dos motores é controlado pelo computador central.

Os sensores próprioceptivos do robô Omni fornecem informações a respeito da movimentação relativa do veículo. Tais sensores são seis codificadores ópticos incrementais, que medem a variação angular relativa das rodas e a velocidade de rotação de cada eixo articulado, três codifi-

cadores ópticos absolutos e o girômetro a laser para medição.

Os sensores externoceptivos fornecem informações absolutas em suas medidas sobre o ambiente no qual o sistema se encontra e são sensíveis a mudanças do ambiente e às variáveis referentes ao próprio sistema. São eles: radar a laser, que fornece uma imagem de profundidade 2-D que representa o contorno dos obstáculos ao seu redor, dentro de um plano de dispersão, câmera e outros. Maiores detalhes sobre a arquitetura do robô podem ser obtidos em [1][11].

Em robótica móvel, é comum o emprego de técnicas de fusão de dados. No caso do robô Omni, os sensores proprioceptivos são usados juntamente com os exteroceptivos para obtenção de uma melhor estimativa do posicionamento do veículo.

3.3 SISTEMAS DE COORDENADAS

Da arquitetura adotada, fazem parte três sistemas de coordenadas (Figura 3.2):

- $\{M\}$ é o sistema de coordenadas do mundo, que representa a referência absoluta adotada;
- $\{R\}$ é o é o sistema de coordenadas local, ou referente ao robô, fixo no centro geométrico dos eixos de direção das rodas;
- $\{C\}$ é o sistema de coordenadas relativo à câmera, fixa à plataforma do robô Omni e direcionada para o teto.

3.3.1 Transformações de Coordenadas

Sejam dois sistemas de coordenadas A e B, no espaço Euclidiano 3D, com origens O^A e O^B respectivamente, a relação entre as coordenadas (x^A, y^A, z^A) no espaço A e (x^B, y^B, z^B) no espaço B é apresentada na Figura 3.3 e é dada por

$$\mathcal{T}_A^B = [\mathbf{R}_A^B \mid \mathbf{t}_A^B] \tag{3.1}$$

$$\mathcal{T}_B^A = [\mathbf{R}_B^A \mid \mathbf{t}_B^A], \tag{3.2}$$

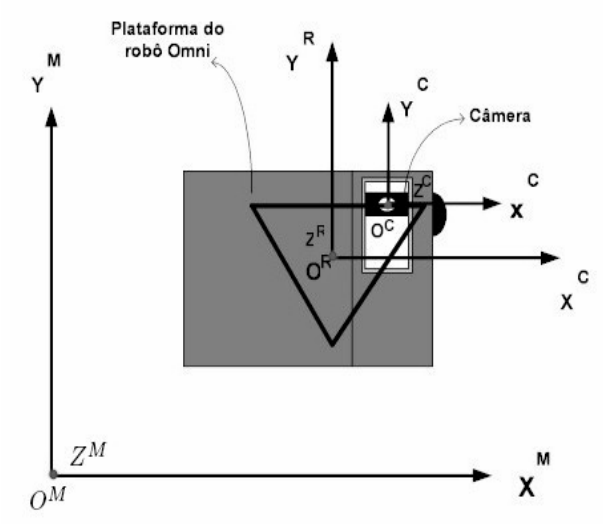


Figura 3.2: Principais sistemas de referência do robô Omni, visão superior.

em que $\mathcal{T}_A^B$ é a transformação direta do sistema de coordenadas $\{A\}$ para o $\{B\}$ e $\mathcal{T}_B^A$ a transformação inversa. $\mathbf{R}_A^B$ é ortogonal e representa a matriz de rotação entre os sistemas $\{A\}$ e $\{B\}$. $\mathbf{t}_A^B$ é o vetor de translação. A matriz de rotação e o vetor de translação de $\{B\}$ para $\{A\}$ são dados por

$$\mathbf{R}_B^A \;=\; (\mathbf{R}_A^B)^T \tag{3.3}$$

$$\mathbf{t}_B^A = -(\mathbf{R}_A^B)^T\, \mathbf{t}_A^B. \tag{3.4}$$

Dessa forma, se existe um ponto P^A no sistema de coordenadas $\{A\}$ e deseja-se obtê-lo no sistema de coordenadas $\{B\}$ aplica-se a transformação

$$\mathbf{p}^B = \mathbf{R}_A^B\, \mathbf{p}^A \;+\; \mathbf{t}_A^B \tag{3.5}$$

na qual $\mathbf{p}^A = (x^A, y^A, z^A)$ e $\mathbf{p}^B = (x^B, y^B, z^B)$ são as coordenadas 3D dos pontos P^A e P^B nos seus respectivos sistemas [16]. Assim,

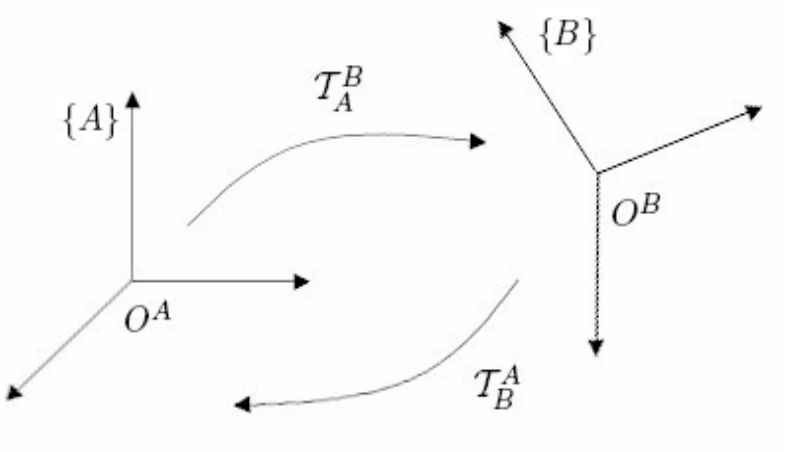


Figura 3.3: Transformações de coordenadas direta e inversa entre os sistemas A e B .

$$\begin{pmatrix} x^A \\ y^A \\ z^A \end{pmatrix} = \mathbf{R}_B^A \left\{ \begin{pmatrix} x^B \\ y^B \\ z^B \end{pmatrix} + \mathbf{t}_B^A \right\}. \tag{3.6}$$

A Figura 3.2 mostra os três principais sistemas de referências do robô Omni, tais sistemas com suas origens $O^M,\ O^R,\ O^C$ e suas respectivas transformações podem ser visualizados na Figura 3.4. Essas transformações, de acordo com as equações (3.3) - (3.5), podem ser descritas da seguinte forma:

$$\mathcal{T}_M^R \ : \ \mathbf{p}^R \ = \ \mathbf{R}_M^R \, \mathbf{p}^M \ + \mathbf{t}_M^R, \tag{3.7}$$

sendo $\mathcal{T}_M^R$ a transformação entre os sistemas de referências do mundo e do robô e, nesse caso um ponto $\mathbf{p}^M$ no sistema de coordenadas do robô sendo transformado em $\mathbf{p}^R$, ou seja, um ponto no sistema de coordenadas do robô. Para transformar um ponto do sistema de referência do robô para a câmera, usa-se

$$\mathcal{T}_R^C \ : \ \mathbf{p}^C \ = \ \mathbf{R}_R^C \, \mathbf{p}^R \ + \mathbf{t}_R^C. \tag{3.8}$$

A transformação direta do sistema de referências do mundo para a câmera é dada por

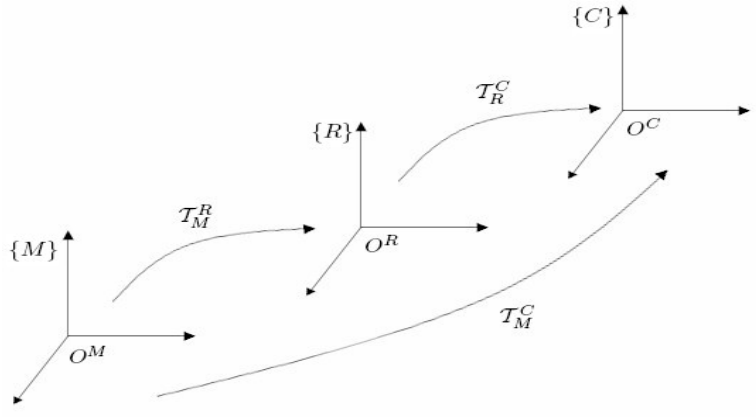


Figura 3.4: Referências do sistema de rastreamento - Mundo, Robô e Câmera, suas origens e transformações que serão usadas pelo sistema de rastreamento.

$$\mathcal{T}_M^C \; : \; \mathbf{p}^C \; = \; \mathbf{R}_R^C \, (\mathbf{R}_M^R \, \mathbf{p}^M \; + \; \mathbf{t}_M^R) + \; \mathbf{t}_R^C \; = \; \mathbf{R}_R^C \, \mathbf{R}_M^R \, \mathbf{p}^M \; + \; \mathbf{R}_R^C \, \mathbf{t}_M^R \; + \; \mathbf{t}_R^C \, , \tag{3.9}$$

na qual $\mathbf{R}_R^C \, \mathbf{R}_M^R \; = \; \mathbf{R}_M^C$ e $\mathbf{R}_R^C \, \mathbf{t}_M^R \; + \; \mathbf{t}_R^C \; = \mathbf{t}_M^C$, assim

$$\mathcal{T}_M^C \; : \; \mathbf{p}^C \; = \; \mathbf{R}_M^C \; \mathbf{p}^M \; + \; \mathbf{t}_M^C . \tag{3.10}$$

Dessa forma, as transformações $\mathcal{T}_M^R$, $\mathcal{T}_R^C$ e $\mathcal{T}_M^C$ realizam respectivamente as transformações do sistema de referência do mundo para o robô, do robô para a câmera e do mundo para a câmera.

3.3.2 Posição de referência da câmera em relação ao sistema de referência do robô

A plataforma móvel do robô Omni é preparada para fixação de três câmeras em sua estrutura: duas em configuração estéreo e uma terceira com foco direcionado para o teto (Figura 3.5). Para o sistema de rastreamento da trajetória é utilizada apenas a configuração (b) da Figura 3.5.

A referência local do robô $\{R\}$ e o sistema de referência da câmera $\{C\}$ estão ligados aos seguintes valores:

$x_R^C = 0,1714m \quad y_R^C = 0,0730m \;\; {}_R^C = 0,6250m \quad (\theta_R^C)_x = 0^o \;\; (\theta_R^C)_y = 0^o \;\; (\theta_R^C)_z = 0^o .$

Dessa forma, tem-se a seguinte transformação

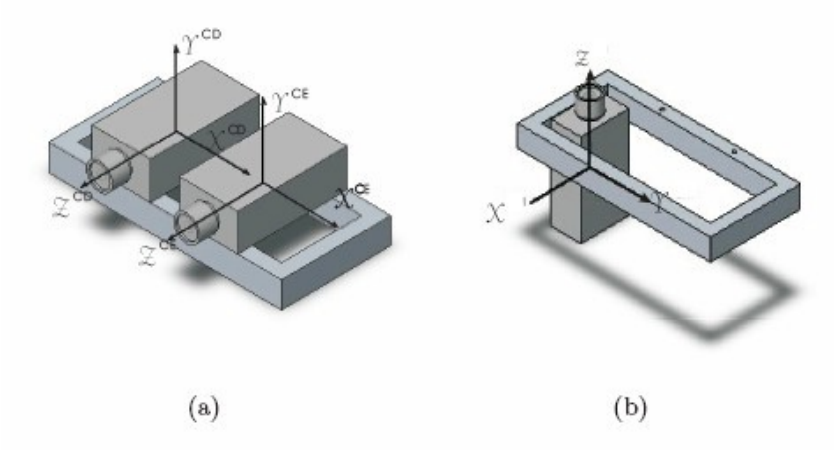


Figura 3.5: Configurações possíveis para câmeras fixadas à plataforma do robô Omni.

$$\begin{pmatrix} x^R \\ y^R \\ z^R \end{pmatrix} = \mathbf{R}_R^C \left\{ \begin{pmatrix} x^C \\ y^C \\ z^C \end{pmatrix} + \begin{pmatrix} x_C^R \\ y_C^R \\ z_C^R \end{pmatrix} \right\}, \tag{3.11}$$

na qual, pela Figura 3.4 e pelos valores de $(\theta_R^C)_x$, $(\theta_R^C)_y$,. Como a câmera está fixada na plataforma do robô, $(\theta_R^C)_z$, $\mathbf{R}_R^C = \mathbf{I}$, ou seja, a matriz identidade.

3.4 SISTEMA DE VISÃO

Para esse trabalho foi utilizada apenas uma câmera modelo *Unibrain Fire-i400* na configuração da Figura 3.5 (b). A câmera fornece imagens

$$\mathcal{I}_k = \{\, p(u,v) \mid u = 1, ..., M; v = 1, ..., N\}, \tag{3.12}$$

onde $\mathcal{I}_k$ é a imagem capturada no instante t_k e $p(u,v)$ a intensidade de luz captada pelo pixel no plano da imagem (a intensidade está em níveis de cinza) e M e N o número de colunas e linhas, respectivamente. A Figura 3.6 apresenta uma seqüência de imagens adquiridas pela câmera fixada ao robô e com foco direcionado para o teto do ambiente.

A interface entre a câmera e o computador central é realizada por um barramento *IEEE 1394*,

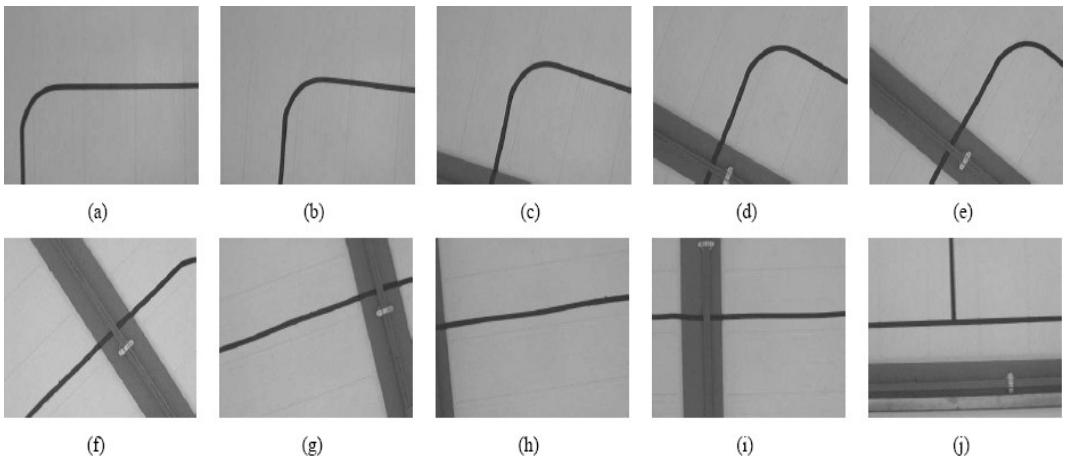


Figura 3.6: Seqüência de imagens capturadas pela câmera fixada à plataforma do robô Omni.

também conhecido como *FireWire*. Tal barramento é serial, de alto desempenho e suporta uma taxa de transmissão máxima de 400 Mbps (com expansão prevista para até 1 Gbps) [1].

O espaço de trabalho 2D observado pela câmera em seu sistema de coordenadas depende exclusivamente das características internas da câmera, representadas pelos parâmetros intrínsecos que são:

- Distância focal multiplicada pelo coeficiente de mudança de escala de metros para pixel na vertical $f.D_u$ e na horizontal $f.D_v$;
- Coordenada em pixel do centro da imagem (u_0, v_0);
- Coeficiente de distorção radial *k*.

Parâmetros	Valor Estimado	Erro Associado a Estimativa
$f.D_u$	1450,93191	6,08074
$f.D_v$	1453,43086	6,15063
u_0	273,77180	8,17728
v_0	157,90635	9,03598
k	-0,32323	0,03862

Tabela 3.1: Parâmetros intrínsecos estimados através de calibração da câmera Fire-i400 [1].

Os parâmetros intrínsecos da câmera são obtidos pela calibração da câmera, que foi realizada por [1], através do uso da toolbox de Calibração de Câmeras no ambiente Matlab [3], desenvolvido por Jean-Yves Bouguet[4]. Os valores de calibração estão disponíveis na Tabela 3.1.

Uma vez obtidos os parâmetros intrínsecos da câmera, pode-se estimar o espaço 2D observado pela câmera. Como a câmera está com foco direcionado para o teto e o solo do ambiente adotado é paralelo ao teto, assim, a distância do chão ao teto permanece fixa e, conseqüentemente, a distância observada do *target* - da câmera ao teto - também é fixa. Essa hipótese não é requerida no sistema proposto, o qual aceita mudanças abruptas nessa variável, até determinado grau. Portanto, a relação entre os parâmetros intrínsecos e o espaço de trabalho 2D observado pela câmera é mostrado na eq. (3.13). Para cada par (u, v), ou seja, para cada pixel da imagem tem-se a coordenada 3D do ponto observado (x^C, y^C, z^C), em relação ao sistema de coordenadas fixo na câmera C:

$$\begin{pmatrix} x^C \\ y^C \end{pmatrix} = \begin{pmatrix} -f.D_u & 0 \\ 0 & f.D_v \end{pmatrix}^{-1} \begin{pmatrix} u - u_0.z^C \\ v - v_0.z^C \end{pmatrix} \tag{3.13}$$

O processo inverso ao que está apresentado na eq. (3.13) [58],

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{K} \begin{pmatrix} x^C/z^C \\ y^C/z^C \\ 1 \end{pmatrix}, \tag{3.14}$$

na qual (x^C, y^C, z^C) são as coordenadas 3D de um ponto P qualquer observado pela câmera, (u, v) é a coordenada da projeção deste ponto na imagem, **K** é a matriz de parâmetros intrínsecos da câmera

$$\mathbf{K} = \begin{pmatrix} f.D_u & 0 & u_0 \\ 0 & f.D_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{3.15}$$

[3] $http://www.mathworks.com/$

[4] $http://www.vision.caltech.edu/bouguetj/calib\_doc/$

e z^C a distância fixa da câmera ao teto que passará a ser chamada de d. Dessa forma, reescrevendo a eq. (3.14),

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \frac{\mathbf{K}}{d} \begin{pmatrix} x^C \\ y^C \\ d \end{pmatrix}. \tag{3.16}$$

3.5 MODELO CINEMÁTICO

O robô Omni possui em seu modelo cinemático os parâmetros apresentados na Figura 3.8. Tal modelo permite variações das variáveis de configuração **q** no espaço articular do robô e de sua postura $\xi_k^R = (x, y, \theta)$ no instante k dentro do espaço cartesiano de trabalho. Na posição absoluta ξ_k^R, (x, y) são as coordenadas do centro geométrico G do suporte das rodas no sistema de coordenadas absoluta e θ é o ângulo entre o eixo *X'* e o eixo *X* que representa a orientação do robô. Na modelagem das equações cinemáticas do robô é desconsiderado o deslizamento na região de contato entre a roda e o solo, além de outros efeitos [1].

Na Figura 3.8 estão dispostas duas ilustrações dos parâmetros do modelo cinemático do robô Omni e na Figura 3.7, a configuração de sua roda descentralizada e orientável. Na primeira, os eixos XY e $X`Y`$ representam os sistemas de coordenadas absoluto e do robô respectivamente. Para cada uma das três rodas $(i = 1, 2, 3)$ não-centradas têm-se os seguintes parâmetros geométricos:

- r_i: o raio (em metros);
- e_i: a distância entre o eixo de orientação do suporte da roda (que é o ponto A_i - Figura 3.7, no sistema de coordenadas $X'Y'$) e seu centro de rotação. Pela Figura 3.7, o eixo de orientação passa pelo ponto de contato do suporte da roda e a estrutura do robô;
- x_{ai} e y_{ai} coordenadas do ponto A_i (Fig. 3.7) do eixo de orientação do suporte da roda no sistema de coordenadas do robô;
- (l_i, α) são as coordenadas polares do ponto A_i no sistema de coordenadas $X'Y'$.

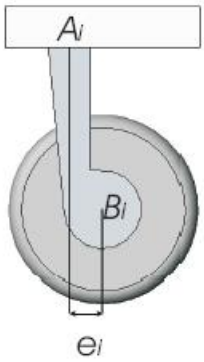


Figura 3.7: Configuração da roda do Omni, descentralizada e orientável [1].

Os parâmetros geométricos $\mathbf{r} = (r_1, r_2, r_3)^T$, $\mathbf{e} = (e_1, e_2, e_3)^T$, $\mathbf{x}_a = (x_{a1}, x_{a2}, x_{a3})^T$ e $\mathbf{y}_a = (y_{a1}, y_{a2}, y_{a3})^T$ são representados por $\vartheta = (\mathbf{r}^T, \mathbf{e}^T, \mathbf{x}_a^T, \mathbf{y}_a^T)$. O vetor

$$\mathbf{q} = (\beta_1 \;\; \beta_2 \;\; \beta_3 \;\; \phi_1 \;\; \phi_2 \;\; \phi_3)^T, \tag{3.17}$$

em que β_i é o é o ângulo de direção da roda i e ϕ_i sua posição angular, representa a configuração do robô.

Para o robô Omni, composto por três rodas diretrizes e motrizes não-centradas, o modelo cinemático inverso é dado por [10]:

$$\dot{q} = \mathbf{J}(\mathbf{q}, \vartheta, \theta)\dot{\xi}, \tag{3.18}$$

no qual $\dot{\xi}$ é a derivada temporal da posição do robô e $\mathbf{J}(\mathbf{q}, \vartheta, \theta)$ a matriz Jacobiana. O modelo cinemático direto dá a variação de posição do robô em função das variáveis cinemáticas. Tal modelo é obtido a partir do modelo cinemático inverso da eq. (3.18) e dado por:

$$\dot{\xi} = \mathbf{J}^{\dagger}(q, \vartheta, \theta)\dot{q}, \tag{3.19}$$

no qual $\mathbf{J}^{\dagger}(q, \vartheta, \theta)$ é pseudo-inversa de Moore$-$Penrose de $\mathbf{J}(q, \vartheta, \theta)$:

$$\mathbf{J}^{\dagger}(q, \vartheta, \theta) = (\mathbf{J}^T(q, \vartheta, \theta)\ \mathbf{J}(q, \vartheta, \theta))^{-1}\ \mathbf{J}^T(q, \vartheta, \theta). \tag{3.20}$$

Maiores detalhes de equações para esse modelo podem ser adquiridos em [1][10].

3.6 ODOMETRIA

O modelo cinemático descrito pela equação (3.19) é empregado sob a forma discretizada para atualizar a posição do veículo em função da variação das variáveis de comando das rodas dentro do período de amostragem. Dessa forma, discretizando tal equação através da aproximação de Euler de primeira ordem,

$$\hat{\xi}(k) = \hat{\xi}(k-1) + \Delta\hat{\xi}(k), \tag{3.21}$$

$$\Delta\hat{\xi}(k) = \mathbf{J}^{\dagger}\left((q(k-1), \vartheta, \hat{\theta}(k-1)\right)\Delta q(k), \tag{3.22}$$

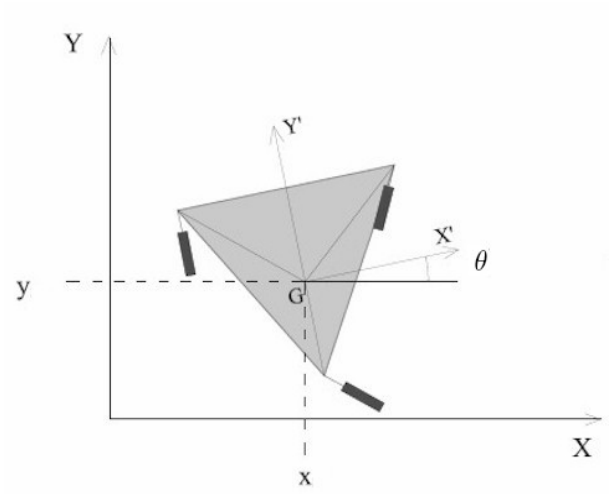
sendo $\Delta q(k) = q(k) - q(k-1)$,

$$\hat{\xi}(k) = \hat{\xi}(k-1) + \mathbf{J}^{\dagger}\left(q(k-1), \vartheta, \hat{\theta}(k-1)\right)\Delta q(k) \tag{3.23}$$

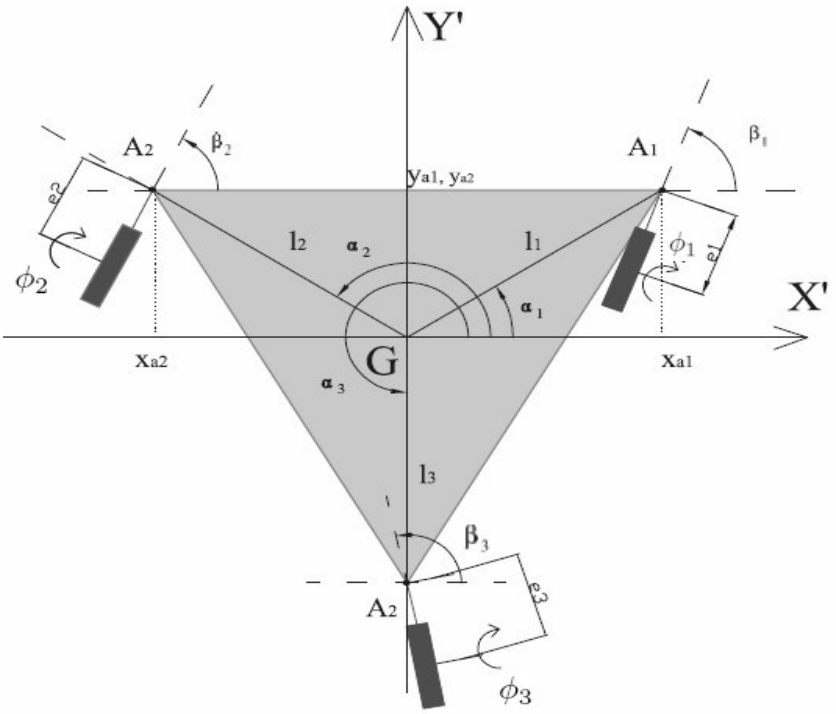
As equações (3.21)-(3.23) mostram como a estimação de posição do robô é realizada no instante discreto k, a partir do seu estado no instante $k-1$ e da leitura dos encoders nos instantes k e $k-1$. Essas equações mostram que a estimação de deslocamento do robô entre os instantes k e $k-1$ é obtida por uma aproximação de primeira ordem, e os termos de ordem superior são desconsiderados, isso porque espera-se que a movimentação do robô respeite as hipóteses seguintes [10]:

1. A configuração geométrica do robô é rígida como mostrado na Figura 3.8;
2. A discretização desconsidera as componentes de ordem superior da aproximação do modelo cinemático;
3. As rodas não derrapam durante o deslocamento do robô, sendo o seu contato considerado pontual;
4. O solo no qual o robô navega é perfeitamente plano;
5. Os valores dos parâmetros são exatamente conhecidos.

Se as hipóteses enumeradas não forem cumpridas, haverá uma contribuição para um aumento sem limite do erro de posicionamento por odometria. Em casos de sistemas reais nenhuma dessas hipóteses é plenamente satisfeita. Isso faz com que a odometria não seja um método confiável para localização. Na verdade, ela é uma técnica bastante interessante para sistemas com pequenos deslocamentos, ou para prover estimativas *a priori* sobre o posicionamento do robô. Para tanto, faz-se necessária uma estimativa inicial, que é atualizada por odometria.



(a)



(b)

Figura 3.8: Parâmetros do modelo cinemático do robô Omni: Variáveis (a) da posição absoluta e (b) da configuração local do modelo [1].

4 ARQUITETURA PROPOSTA PARA RASTREAMENTO DE CURVAS

4.1 INTRODUÇÃO

Esse capítulo apresenta a arquitetura proposta para solucionar o problema de navegação de robôs móveis baseada no rastreamento de marcas artificiais. No caso desse trabalho foram adicionadas cordas e fita adesiva preta ao teto do ambiente formando segmentos polinomiais que devem ser rastreados pelo sistema para a realização da tarefa de navegação do robô.

A Figura 4.1 mostra um robô equipado com uma câmera, com foco direcionado para o teto, nos instantes t_{k-1} e t_k. Nela é apresentado o processo de rastreamento o qual define que, a cada passo de tempo, uma estimação dos parâmetros da curva visualizada estará disponível pelo sistema. Os parâmetros da curva podem ser descritos na forma de coordenadas 3D (robô, teto ou câmera) ou no plano da imagem projetada. Os parâmetros estimados são representados na Figura por λ que são os parâmetros de um polinômio que representa a trajetória e que será descrito na seção 5.6.

Simplificando, o interesse desse trabalho está em rastrear a curva projetada no plano da imagem. Os sensores disponíveis para tal rastreamento são: câmera e odometria (*dead-reckoning*), usada para refinar a estimação do movimento do robô.

Além da trajetória e do robô apresentados (Figura 4.1), há também uma subdivisão do processo de rastreamento da trajetória composto por quatro etapas que são executadas concorrentemente com o processo de controle do robô. Essas etapas, responsáveis pelo rastreamento de curvas para a identificação da trajetória, serão descritas nas seções desse capítulo em maiores detalhes.

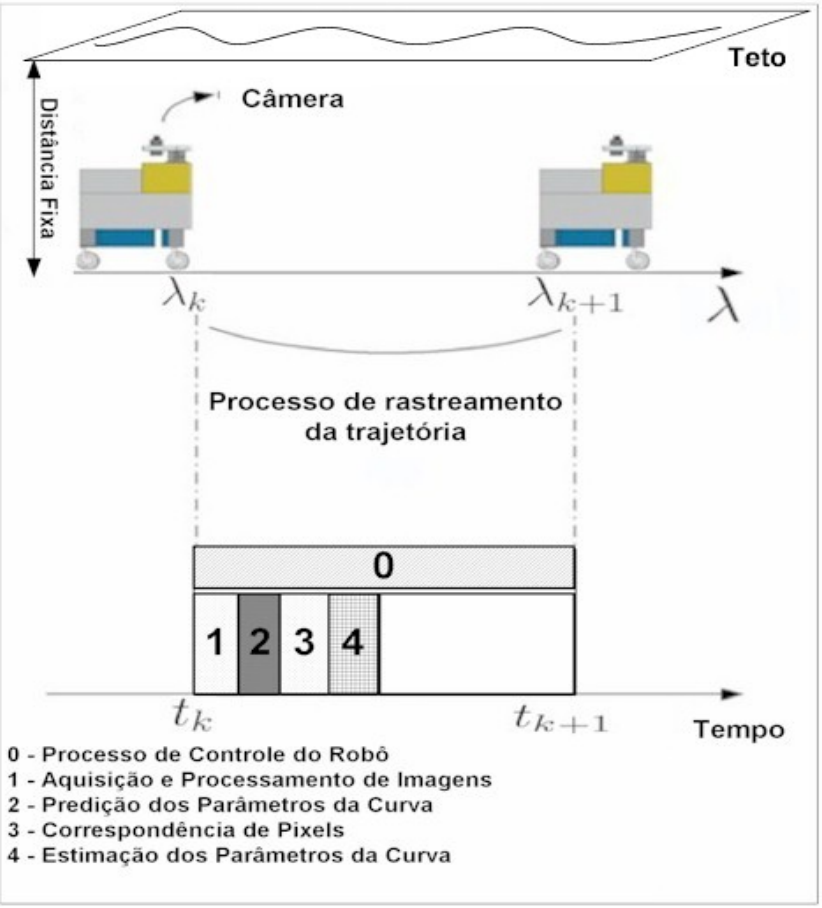


Figura 4.1: Processo de rastreamento da trajetória.

4.2 ARQUITETURA DE RASTREAMENTO

A arquitetura proposta como solução para o rastreamento de curvas que representam a trajetória é composta por um robô equipado com seus sensores e o sistema de controle e rastreamento. O sistema de rastreamento realiza a identificação da trajetória a partir de imagens adquiridas pela câmera acoplada à plataforma do robô e de informações de odometria obtidas através do sistema de controle para atualização temporal de parâmetros estimados que serão descritos na Seção 4.4.

Na Figura 4.1, os tempos t_{k-1} e t_k correspondem aos instantes nos quais as imagens são adquiridas. Para tais instantes, os parâmetros correspondentes da curva fixada ao teto são $\hat{\lambda}_{k-1}$ e $\hat{\lambda}_k$.

No intervalo de tempo $\Delta t = (t_k - t_{k-1})$, dois processos rodam concorrentes: um responsável pelo controle do robô e outro pelo rastreamento da curva. A tarefa de controle do robô também realiza estimação de movimento baseada em odometria. No instante t_k, o sistema de rastreamento realiza aquisição da imagem $\mathcal{I}_k$, predição de $\hat{\lambda}_{k|k-1}$ (que são os parâmetros da curva observada no teto e que serão descritos na próxima subseção) baseada na odometria, correspondência de pixeis e a estimação de $\hat{\lambda}_k$. Nesse momento, $\hat{\lambda}_k$ é enviado para o processo de controle do robô que atualiza a estimativa anterior. Um novo ciclo é iniciado no instante de tempo t_{k+1}.

4.2.1 Parâmetros da curva

As curvas a serem parametrizadas pelo sistema de rastreamento são as curvas algébricas planas que podem ser descritas por *f(u,v)=0*, em que *f* é polinomial de terceira ordem. Essas curvas possuem uma grande variedade de formas [39], entretanto, nesse trabalho será tratado o caso mais comum:

$$v = a_0 + a_1 u + a_2 u^2 + a_3 u^3 \ , \ casos\ especiais\ com\ a_3 \ = \ 0. \tag{4.1}$$

Para obter uma curva a partir dos pixeis selecionados da imagem gerada na etapa de correspondência de pixeis, deve-se estimar os coeficientes $a_0,\ a_1,\ a_2,\ a_3$ da Eq.(4.1). O conjunto de termos da equação formarão

$$\lambda = \left[\begin{array}{cccc} a_0 & a_1 & a_2 & a_3 \end{array} \right]^T , \tag{4.2}$$

sendo λ o conjunto de parâmetros a ser encontrado por um algoritmo estimador.

$$\varphi^T = \left[\begin{array}{cccc} 1 & u & u^2 & u^3 \end{array} \right], \tag{4.3}$$

é o vetor de regressores. Sendo assim,

$$v = \varphi^T \lambda. \tag{4.4}$$

A Figura 4.2 apresenta a imagem de uma curva, as coordenadas (u, v) representando os pixeis na

horizontal e vertical, respectivamente, e o vetor λ que representa os coeficientes da Eq. (4.1) para essa curva.

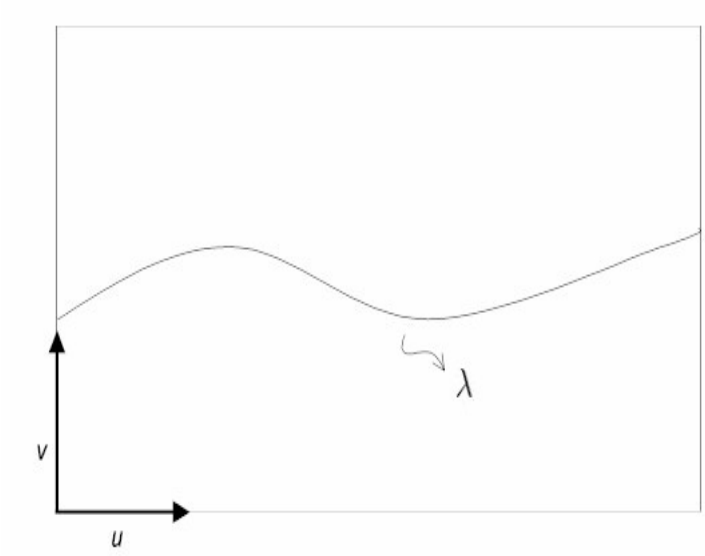


Figura 4.2: Curva formada a partir de λ com (u, v) no espaço imagem tendo suas unidades em pixeis.

Em situações reais, nem sempre os pixeis candidatos da imagem formarão uma curva, portanto, a restrição $a_3 \neq 0$ da Eq. (4.1) não será adotada, tendo em vista que retas também poderão fazer parte da trajetória e deverão ser identificadas. Para realizar a parametrização da curva representada pela marca artificial fixada ao teto foram implementados cinco algoritmos estimadores a serem descritos.

4.3 O PROCESSO DE RASTREAMENTO DE TRAJETÓRIA

Como mostrado na Figura 4.1, o processo de rastreamento de curvas consiste de quatro procedimentos que ocorrem paralelamente ao processo de controle do robô:

1. Aquisição e processamento de imagem: uma imagem é adquirida pela câmera e processada com técnicas de limiarização e conectividade. Após esse processamento inicial, as bordas da imagem são extraídas com Filtro de Canny [14];

2. Predição dos parâmetros da curva: nesse procedimento, o movimento do robô, $\Delta\xi^R$ entre t_{k-1} e t_k é usado para atualizar $\hat{\lambda}_{k-1}$(estimação obtida em t_{k-1}), obtendo assim a predição $\hat{\lambda}_{k|k-1}$ que será utilizado nas próximas etapas;

3. Correspondência de pixeis: ocorre um processo seletivo no qual são selecionados das bordas da imagem apenas os pixeis que estão numa região próxima à curva formada a partir de $\hat{\lambda}_{k|k-1}$. Esse procedimento é essencial para reduzir a quantidade de pixeis da imagem que provavelmente não fazem parte da trajetória;

4. Estimação dos parâmetros da curva: Esta etapa é realizada por um dos algoritmos estimadores implementados que serão descritos posteriormente. Esses algoritmos fornecem uma estimativa de $\hat{\lambda}_k$ baseado somente nos pixeis pré-selecionados no processo anterior.

O algoritmo que realiza o rastreamento da trajetória é simples e comporta as etapas de 1 a 4 apresentadas na Figura 4.1. Todas as etapas do sistema são executadas a cada nova imagem adquirida até que o fim da trajetória seja alcançado. No caso desse trabalho, as imagens da trajetória são adquiridas *off-line*, portanto o fim da trajetória é determinado pelo fim das marcas artificiais do teto.

Inicialmente (instante t_0 ou instante inicial), o sistema não possui informação *a priori* portanto, ao obter uma imagem $\mathcal{I}_0$, ele a processa (etapa 1) e em seguida estima os parâmetros de $\hat{\lambda}_0$ (etapa 4) com pixeis da imagem processada na etapa 1, ou seja, as etapas 2 e 3 são suprimidas. A partir do instante t_1, todas as etapas do sistema são executadas, pois o sistema passará a ter informações *a priori*.

No instante t_k a imagem $\mathcal{I}_k$ é adquirida e processada pela etapa 1, em seguida é realizada a etapa de predição que incorpora à estimativa $\hat{\lambda}_k$ os dados da odometria relativos ao movimento do robô entre os instantes t_{k-1} e t_k. Essa etapa provoca a evolução dos parâmetros $\hat{\lambda}_k$ que, após essa etapa são representados por $\hat{\lambda}_{k|k-1}$, isto é, λ predito no instante t_k baseado no instante t_{k-1}.

A etapa de correspondência de pixeis, executada após a etapa 2, seleciona pixeis da imagem resultante da etapa 1, a partir da curva formada com os parâmetros $\hat{\lambda}_{k|k-1}$. Esses pixeis pré-selecionados servirão para estimar $\hat{\lambda}_k$ na etapa 4 que empregará um algoritmo estimador para tal encerrando assim o processamento para o instante t_k. O processo de rastreamento é repetido,

até que o fim da trajetória seja alcançado. Cada uma dessas etapas será descrita nas próximas subseções.

4.3.1 Aquisição e Processamento de Imagens

O processo de aquisição de imagens é realizado por uma tarefa do *Windows 2000* que utiliza funções de tempo real. A cada período de tempo de aproximadamente 200*ms*, é executada uma função responsável por algumas atividades de interface com o usuário, dentre elas o armazenamento das imagens capturadas pela câmera. Nesse período de tempo são lidos da memória compartilhada os dados providos pelos sensores, as estimativas de posicionamento fornecidas pela odometria e a última imagem capturada pela câmera. A imagem e os dados são armazenados na memória, até que sejam salvos no final da execução do processo.

Devido ao fato de a biblioteca utilizada não permitir a aquisição de imagens em tempo real, as imagens são salvas em arquivo pela função mencionada no parágrafo anterior em formato compatível com a biblioteca *OpenCv*[1], cujas funções serão utilizadas para tratamento das imagens.

As imagens são adquiridas seqüencialmente, uma a cada 35*ms* aproximadamente. Para realizar a aquisição e conversão de cada imagem para o formato compatível com o *OpenCV* são necessários cerca de 75*ms* [1]. Por ser um processo controlado pelo *Windows 2000*, o tempo de aquisição de uma imagem é não determinístico e pode variar bastante em torno desses 75*ms*. Para tentar amenizar, ou seja, minimizar o efeito do não determinismo de tal tarefa, é associada prioridade máxima a este processo [1]. Um mecanismo de comunicação entre as tarefas, permite que o sistema de rastreamento receba as imagens em formato compatível com o *OpenCV* para a realização de suas etapas.

Características geométricas e físicas de objetos são passadas à imagem pelo fato de ocasionarem variações em seus tons de cinza. No caso das imagens adquiridas para o experimento, deseja-se encontrar as marcas artificiais adicionadas ao ambiente. Existem várias técnicas para detectar e extrair informações de objetos, dentre elas a detecção de bordas. Uma borda ou aresta corresponde a uma descontinuidade significativa (alta freqüência) na intensidade dos tons de cinza dos pixeis que formam a imagem [22]. Um operador que detecta essa variação é chamado de de-

[1] http://sourceforge.net/projects/opencvlibrary/

tector de bordas [66]. Para detectar as bordas de objetos dispostos em imagens existem vários tipos de filtros e dentre eles foi escolhido o Filtro de Canny [14], por se tratar de um filtro ótimo.

O detector de bordas desenvolvido por Canny é baseado em critérios de quantificação de desempenho de operadores de bordas conhecidos como critérios de detecção e de localização. Canny definiu três condições básicas que um filtro deveria atender:

- Taxa de Erro ou Detecção consistindo na maximização da razão sinal/ruído *SNR*, quanto maior for o SNR, maior a probabilidade de se detectar as bordas verdadeiras da imagem;
- Distâncias entre os pontos extraídos pelo detector e as respectivas posições verdadeiras devem ser minimizadas, ou seja, os pontos marcados como bordas devem estar o mais próximo possível do centro da verdadeira borda;
- Critério da localização, que é definido pelo inverso da distância entre um ponto detectado e a respectiva posição verdadeira, quanto maior for o valor desse critério, mais próximos estarão os pontos detectados pelo filtro das posições verdadeiras.

Tais critérios garantem ao algoritmo eficiência no processamento de imagens com ruídos ou com bordas difusas. Para a realização do rastreamento de marcas, a princípio, não há como quantificar o ruído do ambiente adotado, portanto, o filtro de Canny foi utilizado por produzir bons resultados de detecção em situações adversas. Como não faz parte do trabalho a avaliação desse filtro, cuja complexidade está muito longe da trivialidade dos operadores convencionais de borda, maiores detalhes podem ser obtidos em [14].

A Figura 4.3 apresenta os passos de processamento executados na etapa de aquisição e processamento de imagens do sistema de rastreamento. Em (a) a imagem original do teto do ambiente marcando artificialmente a trajetória com uma fita isolante preta na horizontal, além dessa marca existe outra marca adicionada artificialmente na vertical e a laje do teto (marca natural) também na horizontal que podem ser confundidas com a trajetória. Em (b) a imagem (a) após a aplicação de Filtro de Canny, além das bordas desejadas, são encontradas várias outras espúrias, provenientes de ruído, texturas e características do próprio ambiente. A eliminação de grande parte desse ruído é efetivada ao aplicar técnicas de conectividade-8 de pixeis e limiarização [22], nesse caso da Figura (b) obteve-se a (c).

(a) Imagem Original (b) Filtro de Canny

(c) Conectividade-8 e limiarização

Figura 4.3: Passos de processamento da etapa 1 do sistema de rastreamento de trajetória.

4.3.2 Predição de parâmetros da curva

Após a realização da etapa de aquisição processamento da imagem para o t_k, o movimento $\Delta\xi^R$ realizado pelo robô entre os instantes t_{k-1} e t_k é usado para atualizar os parâmetros de $\hat{\lambda}_{k-1}$ estimados no instante t_{k-1}. Essa atualização causa uma evolução nos parâmetros de $\hat{\lambda}_{k-1}$ que após tal etapa passa a ser denominado $\hat{\lambda}_{k|k-1}$. A evolução de $\hat{\lambda}_{k-1}$ depende dos dados da odometria a respeito do movimento do robô no mundo. Esses dados são transformados para que venham refletir em pixeis da imagem o movimento realizado pelo robô.

Seja a posição do robô no sistema de coordenadas do mundo nos instantes t_{k-1} e t_k dada por

$(\xi_M^R)_{k-1}$ e $(\xi_M^R)_k$, respectivamente, o movimento do robô pode ser descrito por

$$(\Delta\xi_M^R)_k = (\xi_M^R)_k \ - \ (\xi_M^R)_{k-1}. \tag{4.5}$$

Pela Equação (3.10) apresentada na subseção 3.3.1 do Capítulo 3, tem-se a transformação de um ponto do sistema de referência do mundo para o da câmera. Entretanto, quando o robô realiza um movimento a câmera que está acoplada a ele realiza o mesmo movimento, ou seja, a câmera realiza uma translação entre os instantes t_{k-1} e t_k. Dessa forma, a transformação da câmera é dada por

$$(\mathbf{t}^C)_k = (\mathbf{p}^C)_k \ - \ (\mathbf{p}^C)_{k-1} \tag{4.6}$$

sendo $\mathbf{p}^C$ dado pela Eq.(3.10). Dessa forma, aplicando os dados da odometria $\Delta\xi_M^R = (x, y, \theta)$,

$$(\mathbf{t}^C)_k = (\mathbf{R}_M^C)_k(\mathbf{t}_R^C) + (\mathbf{t}_R^M)_k \ - \ (\mathbf{R}_R^M)_{k-1}(\mathbf{t}_R^C) + (\mathbf{t}_R^M)_{k-1} \tag{4.7}$$

com $(\mathbf{t}_R^C) = [0,1714 \quad 0,0730 \quad 0,6250]^T$ dado na subseção 3.3.2 do Capítulo 3, $(\mathbf{t}_R^M) = (x, y, z)$ para cada instante t_k e t_{k-1} com x e y obtidos de $\Delta\xi_M^R$ e, como z é a altura fixa receberá 1. A matriz de rotação é dada por

$$\mathbf{R}_R^M = \begin{pmatrix} cos(\Delta\theta) & cos(\Delta\theta) & 0 \\ -sen(\Delta\theta) & cos(\Delta\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{4.8}$$

para θ recebido da odometria nos instantes t_k e t_{k-1}.

O processo de evolução dos parâmetros de $\hat{\lambda}_k$ para a obter $\hat{\lambda}_{k|k-1}$, em que

$$\hat{\lambda}_{k|k-1} \ = \ f(\hat{\lambda}_k, \Delta u_k, \Delta v_k, \Delta\theta_k), \tag{4.9}$$

utiliza de transformações de coordenadas dos sistemas de referência do mundo, robô e câmera (Subseções 3.3.1 e 3.3.2 do Capítulo 4), 3.3.2) para que o movimento do robô no mundo entre t_k

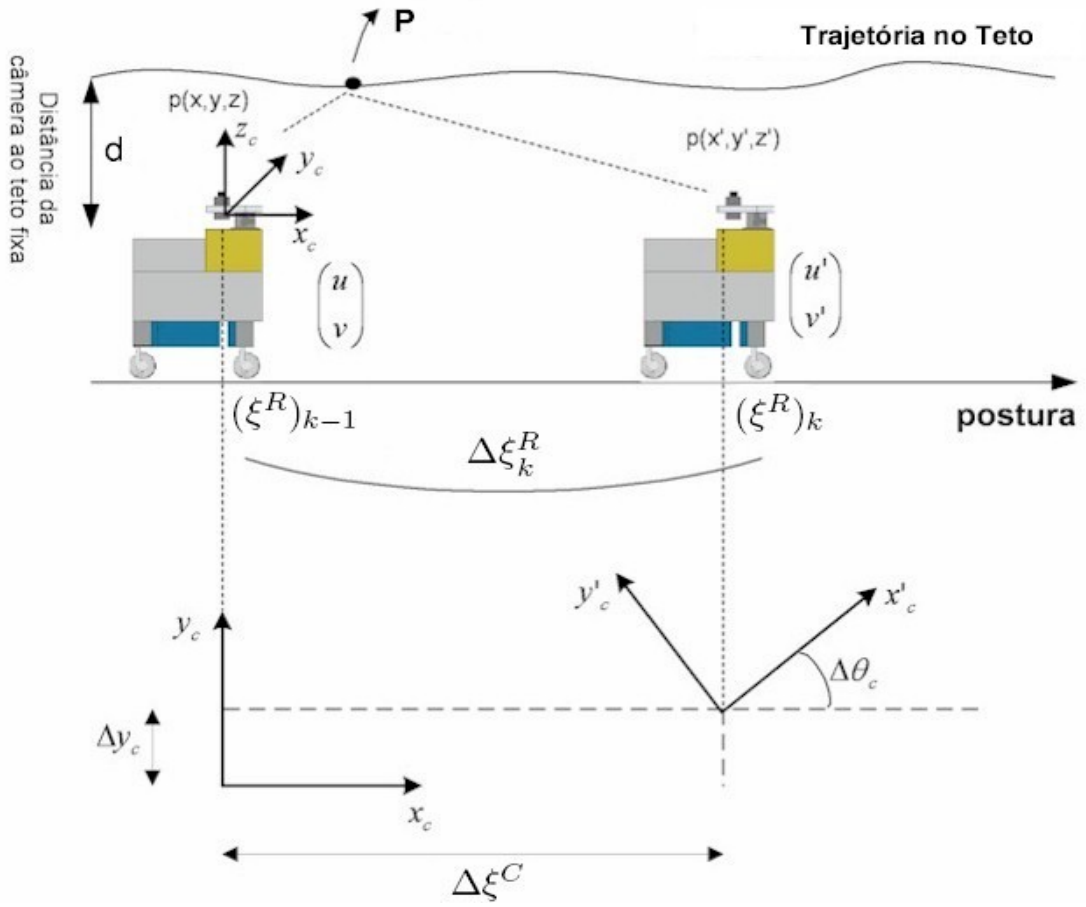


Figura 4.4: Valores de um mesmo ponto p em imagens adquiridas em posições do robô em instantes diferentes.

e t_{k-1}, $((\Delta x_R^M)_k, (\Delta y_R^M)_k, (\Delta \theta_R^M)_k)$ seja transformado em pixeis $(\Delta u_k, \Delta v_k, \Delta \theta_k)$ -(Seção 3.4) do Capítulo 4.

Entre os instantes t_{k-1} e t_k há o movimento 2D da câmera. Em conseqüência, ocorre uma translação do ponto P que na imagem adquirida em t_{k-1} é p_{k-1} e em t_k é p_k (Figura 4.4). A transformação dos pontos observados pela câmera em coordenadas de câmera (pixeis), é dada por

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{K}/d \begin{pmatrix} x^C \\ y^C \\ d \end{pmatrix}, \tag{4.10}$$

em que (u, v) são as coordenadas afins da projeção do ponto $p = (x^C, y^C, z^C)$, que são as coor-

denadas 3D de um ponto com relação à câmera. $d = 1,78m$ é a distância fixa da câmera ao teto e

$$\mathbf{K} = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{4.11}$$

a matriz de parâmetros intrínsecos da câmera com $\alpha_u = fD_u,\ \alpha_v = fD_v$, u_0 e v_0 obtidos da tabela 3.1.

As transformações de coordenadas entre posições da câmera nos instantes t_{k-1} e t_k apresentados na Figura 4.5, e dados por

$$\mathcal{T}_k^C \ = \ [\mathbf{R}(\Delta\theta^C)_k \mid (\Delta\mathbf{t}^C)_k] \tag{4.12}$$

com a matriz de rotação e o vetor de translação dados respectivamente por

$$\mathbf{R}(\Delta\theta^C)_k = \begin{pmatrix} cos(\Delta\theta^C)_k & sen(\Delta\theta^C)_k & 0 \\ -sen(\Delta\theta^C)_k & cos(\Delta\theta^C)_k & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{4.13}$$

$$(\Delta t^C)_k = \begin{pmatrix} (\Delta x^C)_k \\ (\Delta y^C)_k \\ 0 \end{pmatrix} = \begin{pmatrix} (x^C)_k - (x^C)_{k-1} \\ (y^C)_k - (y^C)_{k-1} \\ 0 \end{pmatrix}. \tag{4.14}$$

Um ponto p^C expresso no sistema de coordenadas da câmera, para ser expresso no instante t_k, sofre uma translação **t** e uma rotação **R** (Figura 4.5) [58]. Dessa forma,

$$p_k^C \ = \ \begin{pmatrix} x_k^C \\ y_k^C \\ z_k^C \end{pmatrix} \ = \ \mathbf{R}(p_{k-1}^C - \mathbf{t}). \tag{4.15}$$

Esse mesmo ponto p^C pode ser expresso em coordenadas homogêneas da imagem I (Eq. 4.10), de forma que

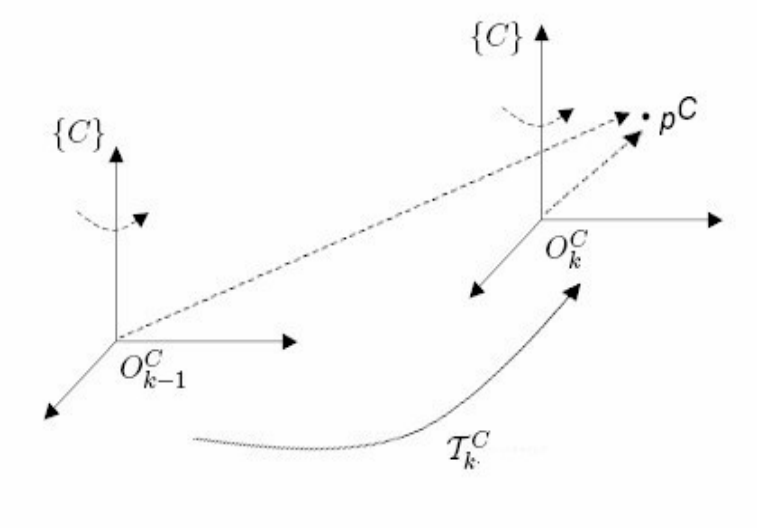


Figura 4.5: Transformação do sistema de coordenadas da câmera entre os instantes t_{k-1} e t_k.

$$p_k^I = \frac{\mathbf{K}}{d}\, p_k^C \;=\; d\mathbf{K}^{-1} p_k^C \tag{4.16}$$

.

Então, realizando algumas substituições de p_k e p_{k-1} temos:

$$p_k = \tfrac{\mathbf{K}}{d}\mathbf{R}(p_{k-1} - \mathbf{t})$$

$$p_k = \tfrac{\mathbf{K}}{d}\mathbf{R}\,(d\,\mathbf{K}^{-1} p_{k-1} - \mathbf{t})$$

assim,

$$p_k = \mathbf{K}\mathbf{R}\mathbf{K}^{-1}(p_{k-1} - \frac{\mathbf{K}}{d}\, t), \tag{4.17}$$

com $\mathbf{K}^{-1}$ sendo a inversa de **K**. Para obter $\hat{\lambda}_{k|k-1}$ (Eq. 4.9), substitui-se p_k e p_{k-1} por $\begin{pmatrix} u_k \\ u_k \\ 1 \end{pmatrix}$ e $\begin{pmatrix} u_{k-1} \\ u_{k-1} \\ 1 \end{pmatrix}$, **K**, **R** e **t** pelas respectivas equações (4.11), a (4.13) e (4.14). Assim,

$$\begin{pmatrix} u_k \\ v_k \\ 1 \end{pmatrix} = \mathbf{KRK}^{-1} \left\{ \begin{pmatrix} u_{k-1} \\ v_{k-1} \\ 1 \end{pmatrix} - \frac{\mathbf{K}}{d} \begin{pmatrix} (\Delta x^C)_k \\ (\Delta y^C)_k \\ 0 \end{pmatrix} \right\}. \tag{4.18}$$

Realizando a substituição de v por $\varphi^T(u)\hat{\lambda}k|k-1$ e $(\Delta x^C)_k,\ (\Delta y^C)_k$ por Δu_k e Δv_k na equação acima temos

$$\begin{pmatrix} u_k \\ \varphi^T(u_k)\hat{\lambda}_{k|k-1} \\ 1 \end{pmatrix} = \underbrace{\mathbf{KRK}^{-1}}_{M} \underbrace{\left\{ \begin{pmatrix} u_{k-1} \\ \varphi^T(u_{k-1})\hat{\lambda}_{k-1} \\ 1 \end{pmatrix} - \begin{pmatrix} \Delta u_k \\ \Delta v_k \\ 0 \end{pmatrix} \right\}}_{N}, \tag{4.19}$$

com

$$M \ : \ \mathbf{KRK}^{-1} = \begin{pmatrix} cos\Delta\theta_k & \frac{\alpha u}{\alpha v} sen\Delta\theta_k & u_0(cos\Delta\theta_k - 1) + v_0(\frac{\alpha u}{\alpha v} sen\Delta\theta_k) \\ -\frac{\alpha v}{\alpha u} sen\Delta\theta_k & cos\Delta\theta_k & v_0(cos\Delta\theta_k + 1) - u_0(\frac{\alpha v}{\alpha u} sen\Delta\theta_k) \\ 0 & 0 & 1 \end{pmatrix} \tag{4.20}$$

e

$$N \ : \ \left\{ \begin{pmatrix} u_{k-1} \\ \varphi^T(u_{k-1})\hat{\lambda}_{k-1} \\ 1 \end{pmatrix} - \begin{pmatrix} \Delta u_k \\ \Delta v_k \\ 0 \end{pmatrix} \right\} = \begin{pmatrix} u_{k-1} - \Delta u_k \\ \varphi^T(u_{k-1})\hat{\lambda}_{k-1} - \Delta v_k \\ 1 \end{pmatrix}. \tag{4.21}$$

Desenvolvendo os termos u_k e $\varphi^T(u_k)\hat{\lambda}_k$ da Eq. (4.19) tem-se que

$$u_k = cos\Delta\theta_k(u_{k-1} - \Delta u_{k-1} + u_0) + sen\Delta\theta_k \frac{\alpha u}{\alpha v}(\varphi^T(u_{k-1})\hat{\lambda}_{k-1} - \Delta v_{k-1} + v_0) + u_0, \tag{4.22}$$

$$\varphi^T(u_k)\hat{\lambda}_{k|k-1} = cos\Delta\theta_k(\varphi^T(u_{k-1})\hat{\lambda}_{k-1} - \Delta v_k + v_0) - sen\Delta\theta_k[\frac{\alpha v}{\alpha u}(u_{k-1} - \Delta u_{k-1} + u_0)] + v_0. \tag{4.23}$$

Como $\hat{\lambda}_k$ estimado depende de todos os pixeis (u, v) selecionados para estimação, assim, com o movimento da câmera, todos os pixeis da imagem obtidos em t_{k-1} sofrerão deslocamento na imagem obtida em t_k. Portanto, reescrevendo a Eq. (4.22) na forma matricial

$$\underbrace{\begin{pmatrix} \varphi^T(u^1)_k \\ \varphi^T(u^2)_k \\ \vdots \\ \varphi^T(u^{Nc})_k \end{pmatrix}}_{\mathbf{C}} \hat{\lambda}_{k|k-1} = \underbrace{\begin{pmatrix} cos\Delta\theta_{k-1}\varphi^T(u^1)_{k-1} \\ cos\Delta\theta_{k-1}\varphi^T(u^2)_{k-1} \\ \vdots \\ cos\Delta\theta_{k-1}\varphi^T(u^{Nc})_{k-1} \end{pmatrix}}_{\mathbf{A}} \hat{\lambda}_{k-1} +$$

$$\underbrace{\begin{pmatrix} sen\Delta\theta_{k-1}[\frac{\alpha u}{\alpha v}(-u^1_{k-1} + \Delta u_{k-1} - u_0)] + cos\Delta\theta_{k-1}(-\Delta v_{k-1} + v_0) + v_0 \\ sen\Delta\theta_{k-1}[\frac{\alpha u}{\alpha v}(-u^2_{k-1} + \Delta u_{k-1} - u_0)] + cos\Delta\theta_{k-1}(-\Delta v_{k-1} + v_0) + v_0 \\ \vdots \\ sen\Delta\theta_{k-1}[\frac{\alpha u}{\alpha v}(-u^{Nc}_{k-1} + \Delta u_{k-1} - u_0)] + cos\Delta\theta_{k-1}(-\Delta v_{k-1} + v_0) + v_0 \end{pmatrix}}_{s}. \tag{4.24}$$

Para obter a predição $\hat{\lambda}_{k|k-1}$, as matrizes **C** e **A** e o vetor **s** da equação acima, podem ser escritos como

$$\mathbf{C}\hat{\lambda}_{k|k-1} = \mathbf{A}\lambda_{k-1} + \mathbf{s}. \tag{4.25}$$

Multiplicando ambos os lados dessa equação por $\mathbf{C}^T$, em seguida obtém-se multiplicando por $(\mathbf{C}^T\mathbf{C})^{-1}$,

$$\hat{\lambda}_{k|k-1} = (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T(\mathbf{A}\hat{\lambda}_{k-1} + \mathbf{s}), \tag{4.26}$$

que é a equação de predição de $\hat{\lambda}_k$.

4.3.3 Correspondência de Pixeis

A etapa de correspondência de pixeis realiza uma pré-seleção de pixeis baseada na informação dos parâmetros estimados no instante de tempo anterior e atualizados na etapa 2. Da imagem resultante da etapa de aquisição e processamento de imagens, são selecionados os pixeis que participarão da estimativa dos parâmetros λ do instante t_k, ou seja $\hat{\lambda}_k$ da etapa 4 no sistema de

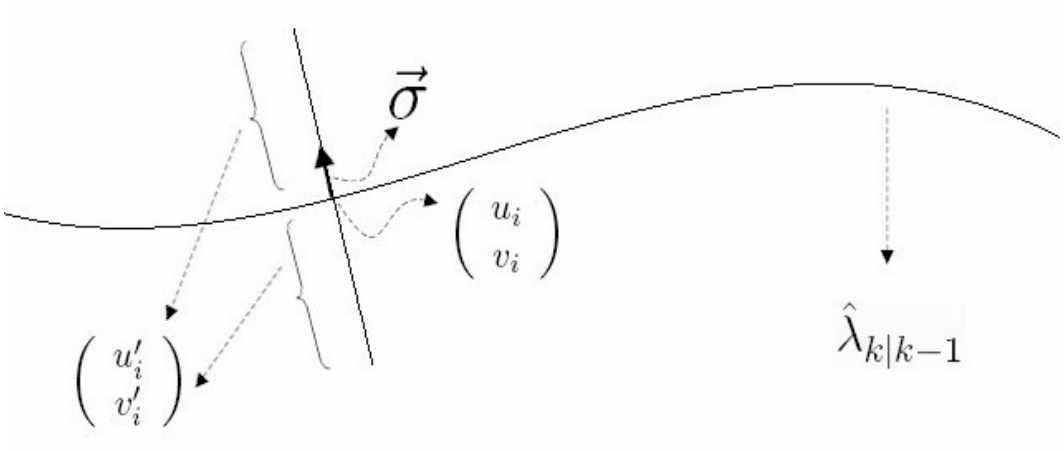


Figura 4.6: Curva gerada com $\hat{\lambda}_{k|k-1}$, o vetor direção $\vec{\sigma}$ de (u_i, v_i) e os pixeis obtidos (Eq. 4.27) acima e abaixo de (u_i, v_i).

rastreamento de trajetórias. Essa seleção é feita tendo como informação a curva gerada com $\hat{\lambda}_{k|k-1}$, obtido da etapa de predição (Figura 4.1).

O processo de correspondência de pixeis acontece pela obtenção do vetor direção unitário de cada pixel componente da curva gerada através de $\hat{\lambda}_{k|k-1}$. Tal vetor pode ser visualizado na Figura 4.6. O vetor direção é representado na curva por $\vec{\sigma}$.

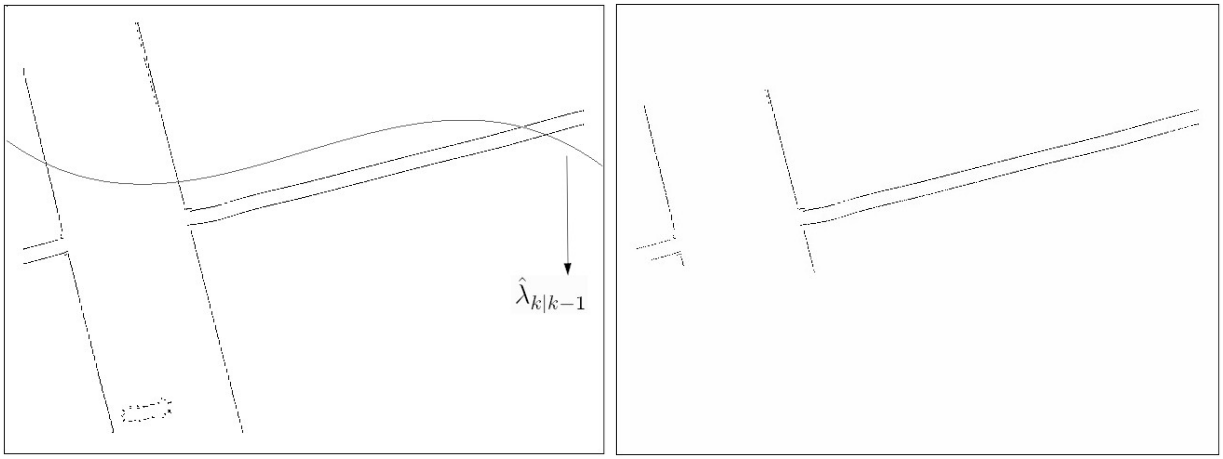
A partir do vetor direção unitário obtido para cada ponto (u_i, v_i), sendo i o pixel em questão, calculam-se 100 pontos (u'_i, v'_i) acima e abaixo de (u_i, v_i) na direção do vetor direção. Sendo

$$\begin{pmatrix} u'_i \\ v'_i \end{pmatrix}_l = \begin{pmatrix} u_i \\ v_i \end{pmatrix} + \vec{\sigma}.l, \; para \; l = -100, \cdots, 100, \tag{4.27}$$

para cada ponto p_i são gerados outros $(p'_i)_l$, ou seja, a curva não será formada apenas por um pixel como na Figura 4.6 para cada (u, v), e sim por 100 pixeis acima e 100 abaixo do pixel (u, v) da curva. Todos esses pixeis serão usados na correspondência.

A correspondência de pixeis entre a imagem resultante da etapa 1 do processo de rastreamento de trajetória e a curva com todos os pontos (u'_i, v'_i) gerados, se dá gerando uma nova imagem que conterá apenas pixeis da imagem obtida da etapa 1 que estiverem nas mesmas coordenadas dos pixeis da curva gerada com $\hat{\lambda}_{k|k-1}$ e dos (u'_i, v'_i) geradas pelo processo descrito acima.

A Figura 4.7, ilustra o funcionamento desta etapa. Esse procedimento é essencial para reduzir a quantidade de *outliers* na imagem resultante da etapa 1 do sistema. A imagem gerada servirá para a etapa seguinte do processo que é a estimação de $\hat{\lambda}_k$. No caso desse projeto, foram utilizadas imagens com tamanho $480x640$ pixeis.



(a) (b)

Figura 4.7: A etapa de correspondência de pixeis recebe a imagem (a) da etapa 1 e produz a imagem (b) baseada em $\hat{\lambda}_{k|k-1}$.

4.4 ABORDAGENS PARA ESTIMAÇÃO DA TRAJETÓRIA

Para estimar os parâmetros da curva que representa a trajetória do robô, $\hat{\lambda}_k$, que é o vetor de parâmetros λ da eq.(4.4) estimado no instante t_k, foram implementados cinco algoritmos estimadores. Tais algoritmos foram classificados em não-seqüenciais e seqüenciais. Os não-seqüenciais caracterizam-se por necessitarem de todo o conjunto de dados antes do início do processo de estimação, pois o vetor de parâmetros, $\hat{\lambda}_k$, é calculado em um único conjunto de operações. Já os seqüenciais utilizam os dados seqüencialmente, ou seja, à medida que são disponibilizados.

Os algoritmos não-seqüenciais e seqüenciais serão detalhados nas próximas seções. Todos foram implementados com a finalidade de verificar a melhor resposta para o problema do rastreamento de curvas.

4.4.1 Estimadores Não-seqüenciais

Os algoritmos estimadores não-seqüenciais (ou em lote) são caracterizados pelo uso de todos os pixeis (selecionados na etapa 3 do rastreamento) disponíveis para a estimação dos parâmetros de um modelo, no caso uma curva ou reta, de uma só vez.

Para verificar o resultado da estimação pelos algoritmos desta abordagem, foram implementados três métodos: Mínimos Quadrados, Mínimos Quadrados Robusto (M-Estimador) e RANSAC-*Random Sample Consensus*.

4.4.1.1 Mínimos Quadrados

O método dos mínimos quadrados é uma técnica de otimização matemática que determina parâmetros desconhecidos de uma equação através da minimização da soma dos quadrados dos resíduos. Dado o modelo

$$(\hat{v}_i)_k = \varphi_k^T(u_i)\hat{\lambda}_k \tag{4.28}$$

a partir da eq. (4.4), e um conjunto de pontos $p_i(u, v)$, o método dos mínimos quadrados tem como princípio a minimização da métrica

$$\sum_{i=1}^{N}(\varepsilon_i)^2, \tag{4.29}$$

em que $\varepsilon_i = (v_i)_k - (\hat{v}_i)_k$, é o i-ésimo resíduo do instante t_k com $i = 1, ..., N$ e N é a quantidade de amostras ou valores observados, $(v_i)_k$ e $(\hat{v}_i)_k$ valores de v observados e calculados no instante t_k, respectivamente. O objetivo é encontrar os parâmetros $\hat{\lambda}_k$ que mais aproximam $\hat{v}_i$ de v_i. Dessa forma, tenta-se encontrar uma função f, tal que $f(u_i) \approx v_i$.

No caso do sistema de rastreamento de curvas, tem-se um problema de regressão linear. Portanto, troca-se a relação $f(u_i) \approx v_i$ por $f(u_i) \approx v_i + \varepsilon_i$, na qual o termo de ruído ε_i é uma variável randômica de média zero. Pela eq. (4.28), a regressão linear emprega

$$\hat{v}_k(u_i, \hat{\lambda}_k) = \varphi_k^T(u_i)\hat{\lambda}_k + (\varepsilon_i)_k, \tag{4.30}$$

que é linear em $\hat{\lambda}_k$, e o erro se torna

$$(\varepsilon_i)_k \;=\; \varepsilon(u_i, \hat{\lambda}_k) = \;(v_i)_k \;-\; \varphi_k^T(u_i)\hat{\lambda}_k, \tag{4.31}$$

que é portanto, os resíduos a serem minimizados no instante t_k.

O critério dos mínimos quadrados para a regressão linear aplicada à eq. (4.30) é

$$\frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\left[(v_i)_k - \varphi_k^T(u_i)\hat{\lambda}_k\right]^{-2}, \tag{4.32}$$

no qual a partir da parametrização linear e do critério quadrático [42], tal equação é uma função quadrática em $\hat{\lambda}_k$. Portanto, pode ser minimizada analiticamente, de forma que

$$\hat{\lambda}_k \;=\; arg_{\lambda_k}\, min\left(\frac{1}{N}\sum_{i=1}^{N}\;\frac{1}{2}[(v_i)_k - \varphi_k^T\lambda_k]^{-2}\right), \tag{4.33}$$

$$\hat{\lambda}_k \;=\; \left[\frac{1}{N}\sum_{i=1}^{N}\varphi_k(u_i)\varphi_k^T(u_i)\right]^{-1}\frac{1}{N}\sum_{i=1}^{N}\varphi_k(u_i)v_k(u_i), \tag{4.34}$$

é a estimação de $\hat{\lambda}_k$ no sentido dos mínimos quadrados.

4.4.1.2 Mínimos Quadrados Robusto - M-Estimador

Seja a soma dos quadrados dos resíduos dada pela eq.(4.29), o algoritmo dos mínimos quadrados a minimiza. Entretanto ele se torna impreciso caso exista uma grande quantidade de *outliers* (pontos que não condizem com o modelo) presentes nos dados de amostra. A presença dos *outliers* causa um grande efeito na minimização, provocando distorção dos parâmetros estimados. E essa é uma limitação do algoritmo dos mínimos quadrados, sua função objetivo dada por eq. (4.29) sofre grande influência de pontos *outliers* extremos com resíduos arbitrários acarretando uma estimação ruim.

O M-Estimador, é uma implementação robusta do algoritmo de mínimos quadrados [70], pois tenta limitar a influência dos *outliers* substituindo a eq. (4.29) por outra

$$\sum_{i=1}^{N} \rho(\varepsilon_i), \tag{4.35}$$

em que ρ é uma função positiva com mínimo único em zero, e é escolhida por garantir menor incremento à função objetivo.

Seja o vetor $\hat{\lambda}_k = [a_0 \quad a_1 \quad a_2 \quad a_3]$ (eq. 4.2), o conjunto de parâmetros a ser estimado, o M-Estimador de λ_k baseado na função $\rho(\varepsilon_i)$ é o vetor $\hat{\lambda}_k$ que é a solução para as seguintes equações:

$$\sum_{i=1}^{N} \psi(\varepsilon_i) \frac{\partial\, \varepsilon_i}{\partial\, a_j} \;=\; 0,\; para\; j = 0, 1, 2, 3, \tag{4.36}$$

na qual a função derivativa $\psi(\varepsilon_i) \;=\; \frac{d\, \rho(\varepsilon_i)}{d\, \varepsilon_i}$ é denominada função de influência [70]. Se uma função peso for definida

$$\omega(\varepsilon_i) = \frac{\psi(\varepsilon_i)}{\varepsilon_i} \tag{4.37}$$

então, a equação eq. (4.38) se torna

$$\sum_{i=1}^{N} \omega(\varepsilon_i)\varepsilon_i \frac{\partial\, \varepsilon_i}{\partial\, a_j} \;=\; 0,\; para\; j = 0, 1, 2, 3. \tag{4.38}$$

Este é exatamente o sistema de equações obtido se o seguinte problema de mínimos quadrados com cálculo iterativo de pesos for resolvido

$$min \sum_{i=1}^{N} \omega_i(\varepsilon_i^{(n-1)})(\varepsilon_i^{(n-1)})^2, \tag{4.39}$$

na qual $n = 1, ..., N_{iter}$, sendo N_{iter} a quantidade de iterações determinadas pelo usuário e o peso $\omega_i(\varepsilon_i^{(Ninter-1)})$ deve ser recomputado após cada iteração para que possa ser utilizado na próxima execução.

Na literatura, encontram-se várias funções ρ [70]. Para esse trabalho, foram implementadas apenas as métrica de Huber [25][30], que é recomendada para a maioria das situações, e a de Tukey [24] que reprime a influência dos *outliers*.

Utilizando a métrica de Huber, a função peso ω para cada n iteração é dada por

$$\omega(\varepsilon_i)^{(n)} = \ min \ \left(1, \ \frac{MAD \ . \ 1,345}{|\varepsilon_i|} \ \right), \tag{4.40}$$

sendo que MAD é uma estimativa de escala (mediana dos desvios absolutos dos resíduos):

$$MAD = median(||\varepsilon_1| - \bar{\varepsilon}| \, , ||\varepsilon_2| - \bar{\varepsilon}| \, , \cdots \, , ||\varepsilon_N| - \bar{\varepsilon}|) \tag{4.41}$$

com $\bar{\varepsilon} = median(|\varepsilon_1|, |\varepsilon_2|, \cdots \, , |\varepsilon_N|)$.

No caso do uso da métrica de Tukey, a função peso ω para cada n iteração é dada por

$$\omega(\varepsilon_i)^{(n)} = \begin{cases} \left(1 - \left(\frac{|\varepsilon_i|}{MAD \ \cdot \ c}\right)^2\right)^2 & , \text{ se } |\varepsilon_i| \leq \ MAD \ \cdot c \\ 0 & , \text{ caso contrário} \end{cases} \tag{4.42}$$

com c variando linearmente partindo de 12 (primeira iteração em $n = 1$) e terminando com 9 (última iteração em $n = N_{iter}$).

4.4.1.3 RANSAC - Random Sample Consensus

O RANSAC é um algoritmo robusto para ajuste de modelos que foi publicado inicialmente por Fischler e Bolles em 1981 [19]. É robusto no sentido de boa tolerância aos *outliers* existentes nos dados experimentais. Este algoritmo é também capaz de interpretar e suavizar dados que contenham uma porcentagem significante de erros.

O algoritmo RANSAC é aplicado a uma grande variedade de problemas de estimação de parâmetros em visão computacional, tais como combinação de características, registro ou detecção de primitivas geométricas. No caso do sistema de rastreamento de curvas, o RANSAC é outro algoritmo da classe de não-seqüenciais utilizado para estimar parâmetros $\hat{\lambda}_k$ da eq. (4.28), tendo como dados os pixeis selecionados da imagem processada no instante t_k, na etapa de correspondência de pixeis do sistema.

Para um dado problema de estimação de parâmetros, o RANSAC assume que tais parâmetros podem ser estimados a partir de um subconjunto de M amostras do total de N existentes. A quan-

tidade de amostras do subconjunto M é denominada consenso por ser as amostras responsáveis pela estimação dos parâmetros do modelo.

No caso da estimação dos parâmetros $\hat{\lambda}_k$, o algoritmo pode ser descrito:

1. São dadas: uma quantidade de iterações N_{iter} e o valor inicial de amostras N';
2. Seleciona-se M amostras do conjunto N randomicamente, sem reposição;
3. Estima-se $\hat{\lambda}_k$ a partir das M amostras por Mínimos Quadrados;
4. Calcula-se os resíduos das M amostras disponíveis e verifica-se quais das M amostras possuem módulo do resíduo inferior a uma certa tolerância (nesse trabalho a tolerância é $h = 0,05$). N' passa a ser o conjunto das amostras cujo resíduo é menor que h;
5. Se N_{iter}, aceita-se o ajuste e calcula-se $\hat{\lambda}_k$ através do M-Estimador de Tukey [24] aplicada ao conjunto consenso associado a N';
6. Repete-se os passos II a V, N_{iter} vezes;

A quantidade de amostras N é incrementada durante a execução, pelo algoritmo, de modo a se obter o máximo consenso. Tal valor depende da quantidade de amostras que se julga pertencer ao modelo sendo ajustado.

4.4.2 Abordagem Seqüencial - Filtragem Estocástica

Como dito anteriormente, na estimação de parâmetros da curva por abordagem seqüencial, os dados medidos são processados seqüencialmente, ou seja, à medida que são disponibilizados, atualiza-se o vetor $\hat{\lambda}_k$ de parâmetros do modelo.

Na abordagem seqüencial serão adotados os procedimentos seguintes que apresentam algoritmos de correção considerando que as medidas são independentes. Para tanto, será lançado mão de técnicas de filtragem estocástica para o seguinte modelo:

$$\lambda_k = \mathbf{f}(\lambda_{k-1}, \mathbf{u}_k) + \mathbf{w}_k \tag{4.43}$$

$$(v_i)_k = \varphi_k^T(u_i)\lambda_k + b_k \tag{4.44}$$

A Eq. (4.43) representa os procedimentos de evolução dos parâmetros λ_k entre as imagens $k-1$ e k, consecutivas. Esta relação permite prever no instante k os parâmetros da curva a partir do que ocorreu no intervalo de tempo t_{k-1} e t_k. A Eq. (4.44) determina a forma como as medições na imagem são relacionadas com λ_k. A medição $\hat{v}_i$, resulta em um modelo de medição linear em que se percebe a dependência deste modelo com u_i. As medidas são consideradas independentes, de forma que tem-se um modelo (4.44) para cada medida.

Por fim, toda incerteza de modelo é representada pelos processos $\mathbf{w}_k$ (vetorial) e b_k (escalar), descorrelacionados, de distribuição Gaussiana com média nula e

$$E\{\mathbf{w}_k\mathbf{w}_k^T\} = \mathbf{Q}_k \tag{4.45}$$

$$E\{b_k^2\} = r_k. \tag{4.46}$$

Os procedimentos a seguir são relacionados à etapa de correção dos parâmetros da curva $\hat{\lambda}_{k|k-1}$ (etapa de predição - Seção 5.4), preditos a partir da imagem anterior e do movimento do robô. Associada à predição $\hat{\lambda}_{k|k-1}$ temos sua matriz de covariâncias, dada por

$$\mathbf{P}_{k|k-1} = \mathbf{W}\mathbf{P}_k\mathbf{W}^T \tag{4.47}$$

em que $\mathbf{W} = (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T(\mathbf{A})$ $(\mathbf{P}_k)$, **C** e **A** matrizes obtidas na Eq. (4.24)e $\mathbf{P}_{k-1}$ a matriz de covariâncias obtida no instante t_{k-1}.

Sendo as medidas independentes, por filtro de Kalman obtém-se a integração delas sob forma seqüencial, conforme mostrado pelo algoritmo abaixo:

1. Inicie $\lambda_k^{(0)} = \lambda_{k|k-1}$ e $\mathbf{P}_k^{(0)} = \mathbf{P}_{k|k-1}$.

2. Para $i = 1$ até N aplique o procedimento de correção devido à *i-ésima* medição:

$$\begin{aligned}\hat{\lambda}_k^{(i)} &= \lambda_k^{(i-1)} + \mathbf{G}_i\left(v_i - \varphi_k^T(u_i)\lambda_k^{(i-1)}\right)\\ \mathbf{P}_k^{(i)} &= (\mathbf{I} - \mathbf{G}_i\varphi_k^T(u_i))\mathbf{P}_k^{(i-1)}\end{aligned}$$

   com

$$\mathbf{G}_i = \mathbf{P}_k^{(i-1)}\varphi_k(u_i)\left(\varphi_k^T(u_i)\mathbf{P}_k^{(i-1)}\varphi_k(u_i) + r_k\right)^{-1}$$

3. Use $\hat{\lambda}_k = \hat{\lambda}_k^{(N)}$ e $\mathbf{P}_k = \mathbf{P}_k^{(N)}$.

4.4.2.1 Discussão acerca do filtro de Kalman

O Filtro de Kalman é um estimador recursivo de máximo *a posteriori* da função da densidade

$$p(\lambda_k|\mathbf{v}_1,\cdots,\mathbf{v}_k) = \frac{p(\mathbf{v}_k|\lambda_k,\mathbf{v}_1,\cdots,\mathbf{v}_{k-1})p(\lambda_k|\mathbf{v}_1,\cdots,\mathbf{v}_{k-1})}{p(\mathbf{v}_k|\mathbf{v}_1,\cdots,\mathbf{v}_{k-1})}$$

.

Este estimador, para o caso linear, resulta na estimativa $\hat{\lambda}_k$ que maximiza $p(\lambda_k|\mathbf{y}_1,\cdots,\mathbf{y}_k)$ apenas para o caso em que (4.43) e (4.44) são modelos estocásticos lineares. Assim, um único procedimento de correção permite maximizar $p(\lambda_k|\mathbf{y}_1,\cdots,\mathbf{y}_k)$. Entretanto, um dos problemas associados ao uso do filtro de Kalman é que uma predição ruim $\hat{\lambda}_{k|k-1}$ pode fazer com que a etapa de correção não seja suficiente para maximizar $p(\lambda_k|\mathbf{y}_1,\cdots,\mathbf{y}_k)$, podendo divergir se $\hat{\lambda}_{k|k-1}$ fizer com que o algoritmo siga uma direção contrária à de maximização de $p(\lambda_k|\mathbf{y}_1,\cdots,\mathbf{y}_k)$, se aproximando assim de um máximo local.

Então, tem-se dois problemas que podem ocorrer devido a uma predição ruim $\hat{\lambda}_{k|k-1}$, que pode ser conseqüência de erro de modelamento em $\mathbf{f}(\lambda_{k-1},\mathbf{u}_k)$ [34][11]:

1. O filtro de Kalman somente chega à estimativa de máximo *a posteriori* de $p(\lambda_k|\mathbf{y}_1,\cdots,\mathbf{y}_k)$ se o modelo (4.43)-(4.44) for estocástico linear, o que não é para este problema devido a (4.43). Na verdade, sendo o modelo não linear, este filtro é denominado Filtro de Kalman Estendido;

2. Se $\hat{\lambda}_{k|k-1}$ estiver próximo o suficiente de um modo secundário de $p(\lambda_k|\mathbf{y}_1,\cdots,\mathbf{y}_k)$, i.e., de um máximo local de $p(\lambda_k|\mathbf{y}_1,\cdots,\mathbf{y}_k)$, então o passo de correção do Filtro de Kalman o fará seguir na direção deste modo secundário.

O problema 1 pode ser minimizado se a etapa de correção for iterativa, fazendo com que $\hat{\lambda}_k$ se aproxime do máximo de $p(\lambda_k|\mathbf{y}_1,\cdots,\mathbf{y}_k)$. Para tanto, pode-se fazer uso do Filtro de Kalman Estendido Iterativo (FKEI). Entretanto, em decorrência do problema 2, o procedimento iterativo do FKEI faz com que, no melhor dos casos, $\hat{\lambda}_k$ alcance algum modo secundário. Uma forma de resolver este problema é procurar eliminar o modo secundário que atrai $\hat{\lambda}_k$. Como modos secundários aparecem devido à existência de medições incompatíveis com o modelo (*outliers*), a

minimização de sua influência para a criação de um modo secundário pode ser realizada com o auxílio de um estimador robusto, que é apresentado a seguir.

4.4.2.2 Algoritmo de correção por filtro de Kalman Robusto

O filtro de Kalman robusto é uma versão robusta do filtro de Kalman Estendido Iterativo (FKEI) [34], trata-se do FKEIR [11]. O algoritmo do FKEI é dado por:

1. Inicie $\hat{\lambda}_k^{(0)} = \hat{\lambda}_{k|k-1}$ e $\mathbf{P}_k^{(0)} = \mathbf{P}_{k|k-1}$.

2. Para $n = 1$ até N_{iter}, realize a *n-ésima* iteração de integração dos dados:

   (a) $\hat{\eta}_k^{(0)} = \hat{\lambda}_k^{(n-1)}$ e $\mathbf{S}_k^{(0)} = \mathbf{P}_k^{(n-1)}$.

   (b) Para $i = 1$ até N aplique o procedimento de correção devido à *i-ésima* medição:

$$\begin{aligned} \hat{\eta}_k^{(i)} &= \hat{\lambda}_{k|k-1} + \mathbf{G}_i \left(v_i - \varphi_k^T(u_i)\hat{\eta}_k^{(i-1)} - \varphi_k^T(u_i) \left(\hat{\lambda}_{k|k-1} - \hat{\eta}_k^{(i-1)} \right) \right) \\ \mathbf{S}_k^{(i)} &= (\mathbf{I} - \mathbf{G}_i \varphi_k^T(u_i))\mathbf{S}_k^{(i-1)} \end{aligned}$$

   com

$$\mathbf{G}_i = \mathbf{S}_k^{(i-1)} \varphi_k(u_i) \left(\varphi_k^T(u_i) \mathbf{S}_k^{(i-1)} \varphi_k(u_i) + r_k \right)^{-1}$$

   (c) Atualize as estimativas decorrentes da integração: $\hat{\lambda}_k^{(n)} = \hat{\eta}_k^{(N)} =$ e $\mathbf{P}_k^{(n)} = \mathbf{S}_k^{(N)}$.

3. Use $\hat{\lambda}_k = \hat{\lambda}_k^{(N_{iter})}$ e $\mathbf{P}_k = \mathbf{P}_k^{(N_{iter})}$.

N_{iter} é o número de iterações a serem realizadas. Tal quantidade é definida pelo usuário.

O FKEIR é dado por:

1. Inicie $\hat{\lambda}_k^{(0)} = \hat{\lambda}_{k|k-1}$ e $\mathbf{P}_k^{(0)} = \mathbf{P}_{k|k-1}$.

2. Sem conhecimento prévio, considera-se que todas as medições têm a mesma importância no processo de medição. Portanto, faça $\omega_i^{(0)} = 1$ para $i = 1, \cdots, N$.

3. Para $n = 1$ até N_{iter}, realize a *n-ésima* iteração de integração dos dados:

   (a) $\hat{\eta}_k^{(0)} = \hat{\lambda}_k^{(n-1)}$ e $\mathbf{S}_k^{(0)} = \mathbf{P}_k^{(n-1)}$.

(b) Para $i = 1$ até N aplique o procedimento de correção devido à *i-ésima* medição apenas para as medições para as quais $\omega_i^{(n-1)} > 0$:

$$\begin{aligned}\hat{\eta}_k^{(i)} &= \hat{\lambda}_{k|k-1} + \mathbf{G}_i \left(v_i - \varphi_k^T(u_i)\hat{\eta}_k^{(i-1)} - \varphi_k^T(u_i) \left(\hat{\lambda}_{k|k-1} - \hat{\eta}_k^{(i-1)} \right) \right) \\ \mathbf{S}_k^{(i)} &= (\mathbf{I} - \mathbf{G}_i \varphi_k^T(u_i))\mathbf{S}_k^{(i-1)}\end{aligned}$$

com

$$\mathbf{G}_i = \mathbf{S}_k^{(i-1)}\varphi_k(u_i) \left(\varphi_k^T(u_i)\mathbf{S}_k^{(i-1)}\varphi_k(u_i) + \frac{r_k}{\omega_i^{(n-1)}} \right)^{-1}$$

(c) Atualize as estimativas decorrentes da integração: $\hat{\lambda}_k^{(n)} = \hat{\eta}_k^{(N)} =$ e $\mathbf{P}_k^{(n)} = \mathbf{S}_k^{(N)}$.

(d) Atualize os pesos $\omega_i^{(n)}$ de acordo com uma métrica robusta, que pode ser Huber[25][30] ou Tukey[24] - cujas equações podem ser encontradas na seção 4.4.1.2- sobre os resíduos atuais:

$$(\varepsilon_i)_k^{(n)} = v_i - \varphi_k^T(u_i)\hat{\lambda}_k^{(n)}.$$

4. Use $\hat{\lambda}_k = \hat{\lambda}_k^{(N_{iter})}$ e $\mathbf{P}_k = \mathbf{P}_k^{(N_{iter})}$.

5 RESULTADOS EXPERIMENTAIS

5.1 INTRODUÇÃO

O sistema proposto nos capítulos anteriores foi submetido ao rastreamento de diferentes trajetórias. O desempenho dos cinco algoritmos implementados e apresentados no capítulo anterior, bem como a resposta do sistema em cada uma das etapas apresentadas no capítulo 4 foram analisadas. Para obter os resultados que serão apresentados, foram realizados testes *off-line* em um computador com processador Pentium III 1700GHZ, 256MB de memória RAM, com sistema operacional Windows XP e compilador Visual C++ 6.0.

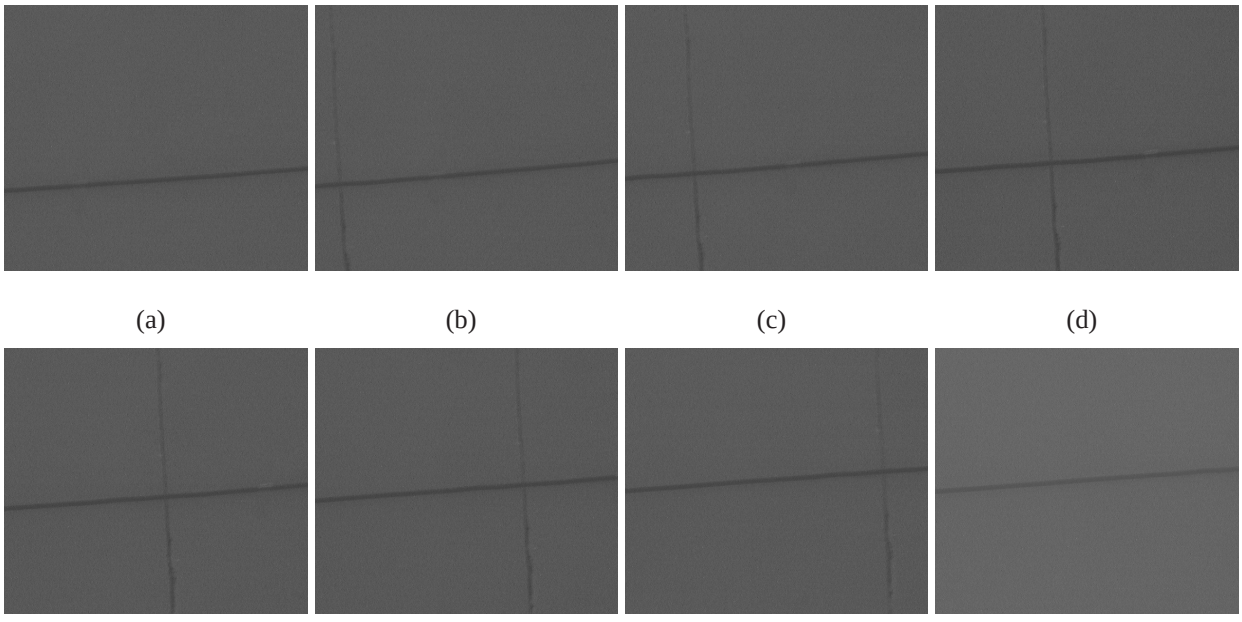
Este capítulo trata dos resultados experimentais obtidos para as trajetórias apresentadas a cada um dos algoritmos estimadores implementados. Para imagens da trajetória considerada, são ilustrados os resultados das etapas de aquisição e processamento de imagens, predição dos parâmetros da curva e correspondência de pixeis e estimação dos parâmetros da curva (Figura 4.1). As Seções 5.3 - 5.5 tratam das etapas de 1 a 3 do sistema e finalmente, a Seção 5.6 trata dos resultados obtidos por cada um dos algoritmos estimadores para as seqüências de imagens apresentadas.

5.2 APRESENTAÇÃO DAS TRAJETÓRIAS EMPREGADAS

Os resultados de rastreamento aqui apresentados referem-se a seqüências de imagens obtidas em ambientes de teste. O sistema de rastreamento foi implementado para situações reais nas quais o ambiente pode ter características próprias diversas.

Para se ter uma noção do tipo de trajetória a ser rastreada pelo sistema, foram escolhidas duas seqüências de imagens capturadas pela câmera do robô em momento de navegação guiada por usuário através de *joystick*. Uma das trajetórias é apresentada na Figura 5.1. Outra com características de ambiente completamente diferentes na Figura 5.2.

A seqüência observada na Figura 5.1 apresenta um ambiente de teste com pouca luminosi-



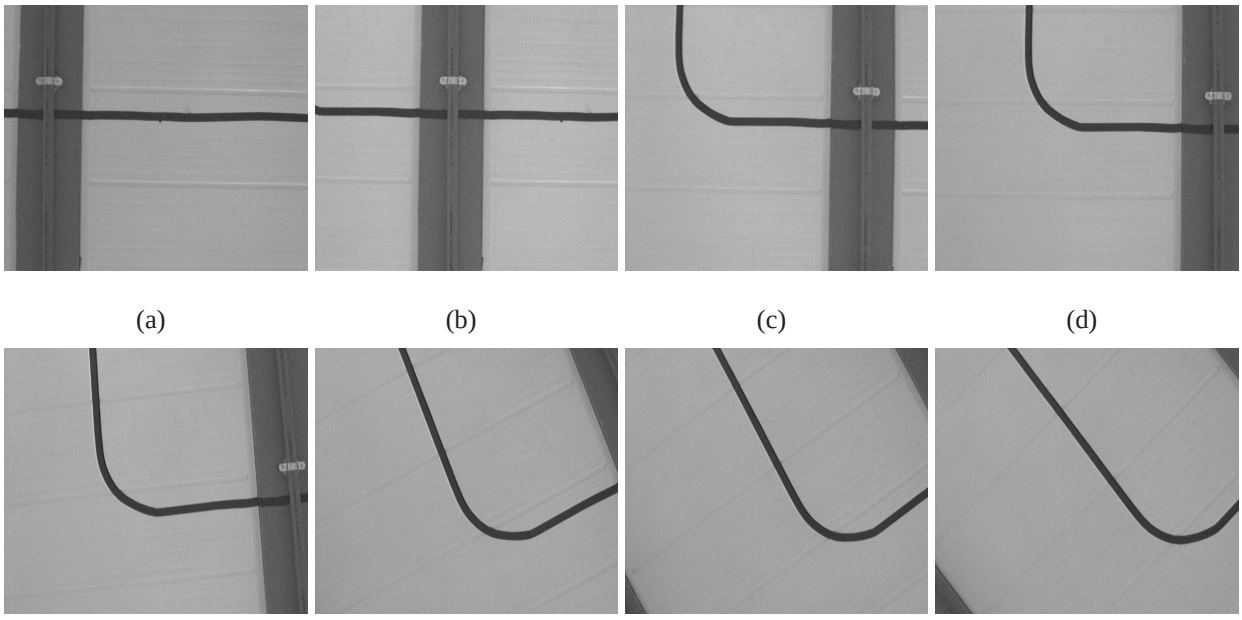
(a) (b) (c) (d)

(e) (f) (g) (h)

Figura 5.1: Seqüência de imagens adquiridas pela câmera do robô em ambiente com pouca luminosidade e com uma corda como marca artificial.

dade, o que contribuiu para que as imagens adquiridas ficassem escuras e com ruído do tipo *salt and pepper*. Nesta imagens, a marca artificial observada é uma corda de nylon escura e de espessura média. Devido à dificuldade em fixar a corda ao teto, a trajetória formada é composta em sua maioria por retas e curvas muito suaves. Como pode ser observado, durante o percurso aparecem outras retas próprias do teto (Figuras 5.1 (b)-(g)) que podem atrapalhar e confundir o reconhecimento da trajetória.

A Figura 5.2 apresenta outra seqüencia de imagens obtidas para teste. Nesse o ambiente possui características diferentes das da Figura 5.1, maior luminosidade e um teto com revestimento formando listras contínuas e paralelas. Para esse teste foi utilizada uma fita adesiva preta como marca artificial, o que facilitou a incorporação de curvas mais acentuadas à trajetória. Além das retas do revestimento do teto, existem outras que podem confundir o sistema como, a laje, canos de alumínio, luminária. Além das marcas que demarcam a trajetória, foram adicionadas outras com a finalidade de testar o funcionamento do sistema. Esses artifícios tendem a dificultar o processo de rastreamento, pois podem ser confundidos com as trajetórias reais. No mais, eles são detectados no processo de filtragem de bordas.



(a) (b) (c) (d)

(e) (f) (g) (h)

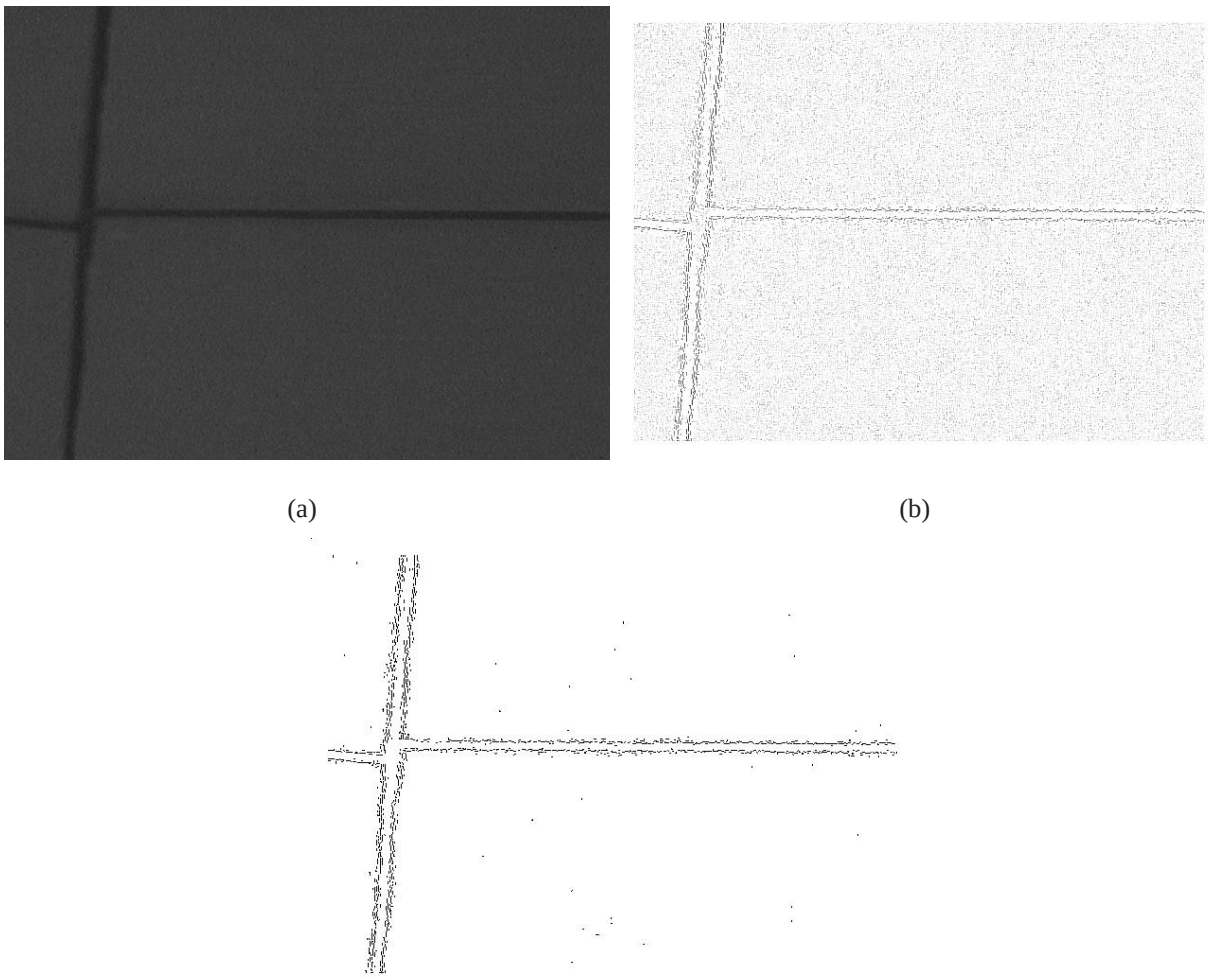
Figura 5.2: Seqüência de imagens adquiridas pela câmera do robô em ambiente claro e com fita adesiva preta como marca artificial.

5.3 PROCESSAMENTO DAS IMAGENS

Conforme descrito nos capítulos anteriores, a primeira etapa do sistema de rastreamento de trajetória é a aquisição da imagem do instante t_k e o processamento dessa imagem. O objetivo dessa etapa é obter, a partir da imagem adquirida, uma grande quantidade de pixeis candidatos para a estimação e conseqüentemente uma pequena quantidade de *outliers*.

Para apresentar a eficácia dessa etapa, bem como sua importância para as demais etapas, são apresentados os resultados obtidos com imagens retiradas das seqüências das Figuras 5.1 e 5.2. Da primeira seqüência, a Figura 5.3 (a) mostra a imagem original adquirida. Em (b) é mostrado o resultado do primeiro processamento com filtro de Canny para extração de bordas e em (c) o emprego de técnicas de limiarização juntamente com conectividade8 na imagem (b). O resultado do processamento de uma imagem retirada da seqüência da Figura 5.2 é apresentado na figura 5.4.

Uma característica importante das imagens processadas com filtro de Canny nas situações apresentadas (Figuras 5.3 e 5.4 (b)) é a grande quantidade de *outliers* provenientes das bordas



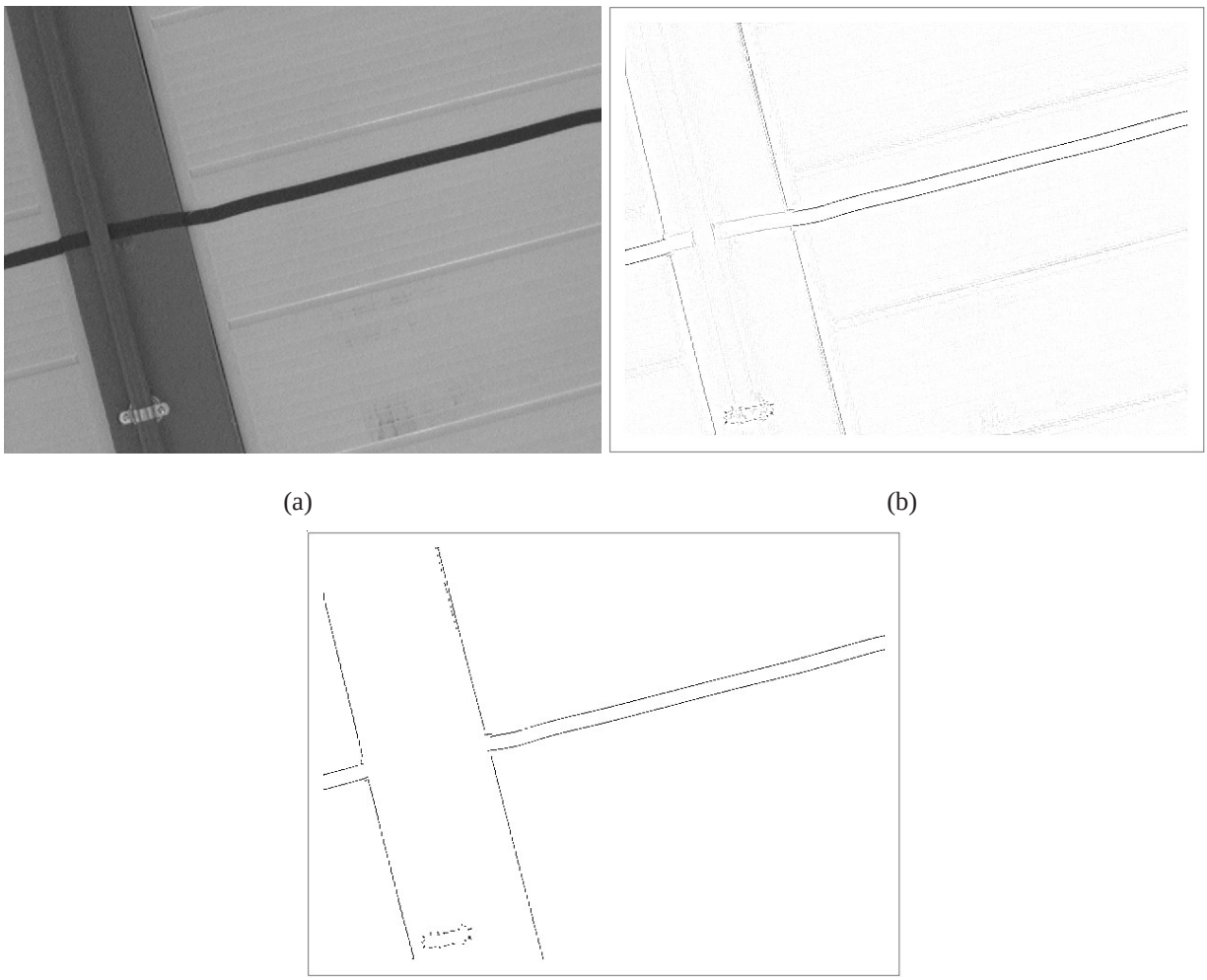
(a) (b)

(c)

Figura 5.3: Resultados da etapa 1 obtidos em ambiente escuro. A imagem (a) é obtida, processada com filtro de Canny (b) e finalmente com técnicas de limiarização e conectividade-8.

extraídas a partir das características de cada ambiente. Na Figura 5.3 (b), os *outliers* são conseqüência da baixa luminosidade do ambiente, pois o teto é branco e sem manchas. Para amenizar o efeito da baixa luminosidade foram testadas técnicas de equalização de histograma e filtro passa-baixas, mas não foram eficazes, tornando-se desnecessárias. Entretanto, em (b) da Figura 5.4 tais pontos são provenientes da textura do revestimento do teto, laje e cano de alumínio que podem ser visualizados na imagem original (a).

Os *outliers* observados em (b)(Figuras 5.3 5.4) representam um problema na estimação de $\hat{\lambda}_k$, pois, em grande quantidade, influenciarão a estimação ao desviarem do sentido da trajetória. Boa



(a) (b)

(c)

Figura 5.4: Resultados da etapa 1 obtidos em ambiente claro. A imagem (a) é obtida, processada com filtro de Canny (b) e finalmente com técnicas de limiarização e conectividade-8.

parte desses *outliers* foram removidos com o uso de técnicas de limiarização e conectividade-8 aplicadas logo após o filtro de Canny. Tal resultado pode ser observado para ambas Figuras 5.3 5.4 em (c). Pode-se observar que ambas figuras encontram-se aptas ao próximo passo do sistema de rastreamento, sem a necessidade do uso de outras técnicas como por exemplo, amenização de baixa luminosidade.

As imagens obtidas após o emprego de limiarização e conectividade-8, permitirão a determinação de quais pixeis representam possíveis pontos da trajetória na etapa 3 do sistema, a qual trata da pré-seleção realizada pela correspondência de pixeis descrita no capítulo 4 e mostrada na

Seção 5.5.

5.4 PREDIÇÃO DOS PARÂMETROS DA CURVA

Após a execução e obtenção da imagem processada na etapa 1 do sistema, é executada a etapa de predição dos parâmetros da curva. Nessa etapa ocorre a evolução dos parâmetros $\hat{\lambda}_{k-1}$ estimados em t_{k-1} e a partir do movimento $\Delta\xi^R$ do robô entre os instantes t_{k-1} e t_k (descrita no Capítulo 4). Tal evolução gerará $\hat{\lambda}_{k-1|k}$.

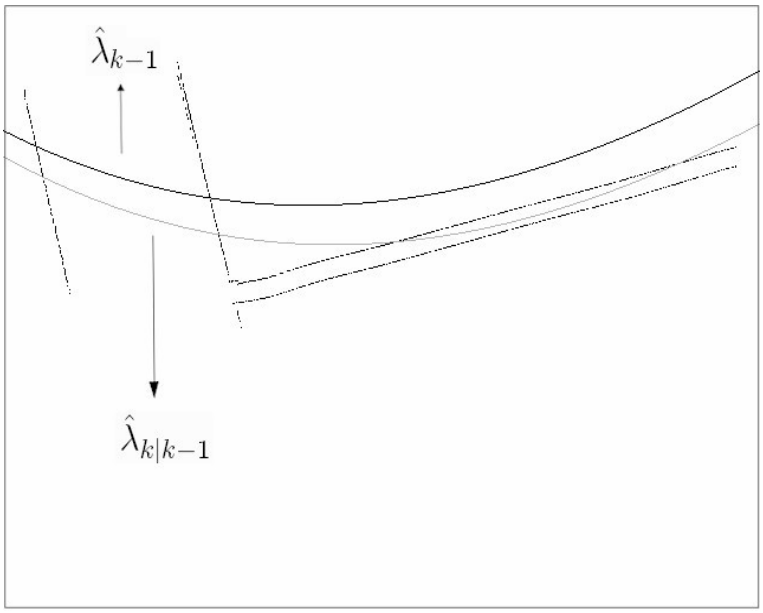


Figura 5.5: A etapa de predição recebe $\hat{\lambda}_{k-1}$, $\Delta\xi^R$ e calcula $\hat{\lambda}_{k|k-1}$. Nesse caso $\Delta u = 31,16$ e $\Delta v = 21,41$.

A Figura 5.5 apresenta os pixeis da imagem do instante t_{k-1} utilizados para estimar $\hat{\lambda}_{k-1}$. Sobre tais pixeis da imagem estão projetadas as curvas geradas a partir de $\hat{\lambda}_{k-1}$ e de $\hat{\lambda}_{k|k-1}$. Para obter $\hat{\lambda}_{k|k-1}$, a informação recebida da odometria foi transformada em coordenadas afim, refletindo assim tal moviemtento na imagem. No caso da Figura 5.5, o movimento do robô obtido da odometria e transformado para ser refletido na imagem foi de $\Delta u = 31,16$ pixeis (na vertical) e $\Delta v = 21,41$ pixeis (na horizontal) e a rotação $\Delta\theta = -0,16\ rad$. Esse movimento aplicado em $\hat{\lambda}_{k-1}$ gerou $\hat{\lambda}_{k|k-1}$, ambos sinalizados na figura.

A predição de $\hat{\lambda}_{k-1}$, como pôde ser observadO (Figura 5.6), É uma importante etapa do sistema de rastreamento, principalmente por gerar uma atualização dos parâmetros no tempo. Esses parâmetros atualizados participarão da etapa 3 que pré-selecionará pixeis da imagem obtida em t_k. Mesmo com a realização de testes sem a predição o sistema conseguiu rastrear a trajetória, entretanto, quanto maior a quantidade de informações utilizaddas, mais confiável e robusto será o sistema.

5.5 CORRESPONDÊNCIA DE PIXEIS

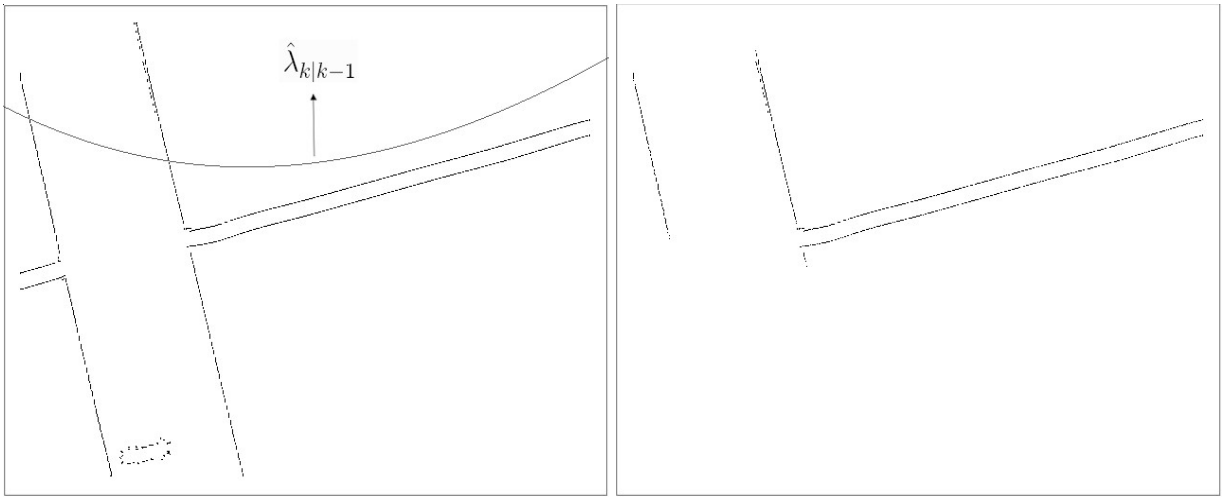
Responsável por receber a imagem processada na etapa de aquisição e processamento de imagens e os parâmetros evoluídos $\hat{\lambda}_{k|k-1}$ da etapa de predição, a etapa de correspondência de pixeis pré-seleciona pixeis da imagem que serão utilizados pela próxima etapa do sistema como parte do conjunto de amostras para a estimação de $\hat{\lambda}_k$.

A Figura 5.6 apresenta o resultado obtido para essa etapa. Em (a) estão dispostos os pixeis pré-selecionados da imagem Figura 5.4(a) pela etapa 1 do sistema, bem como a projeção da curva gerada com os parâmetros $\hat{\lambda}_{k|k-1}$ obtida na etapa 2 e apresentada na Figura 5.4. Em Figura 5.6 (b) pode-se observar o resultado obtido pela etapa de correspondência de pixeis. Apenas os pixeis da imagem Figura 5.6(a) que estão na região de 100 pixeis acima e abaixo de $\hat{\lambda}_{k|k-1}$ são selecionados.

A etapa de correspondência de pixeis é responsável por eliminar uma grande quantidade de *outliers*, reduzindo assim o efeito dos mesmos na obtenção de $\hat{\lambda}_k$ na próxima etapa do processo. A contribuição dessa etapa é muito importante para os algoritmos estimadores, que dependendo da situação e das características do algoritmo, são muito influenciados por *outliers*. Tal procedimento além de reduzir ainda mais a quantidade de *outliers*, define a área da imagem na qual a probabilidade de existirem pixeis relativos à trajetória é maior, pois baseia-se em um conhecimento *a priori* obtido da etapa anterior do sistema.

Nem sempre todos os pixeis da trajetória estarão na área de correspondência, entretanto os resultados mostram que tal procedimento funcionou no caso apresentado e nos demais testados. Outro fato importante é que, o sucesso dessa etapa está ligado ao sucesso do algoritmo estimador ao estimar $\hat{\lambda}_{k-1}$, pois esse é o conhecimento *a priori* que após a evolução será usado para

pré-seleção de pixeis. Conseqüentemente, a robustez da correspondência de pixeis depende da robustez do algoritmo estimador da etapa 4 do ciclo t_{k-1} na estimação de $\hat{\lambda}_{k-1}$, e da predição em t_k na obtenção de $\hat{\lambda}_{k|k-1}$.



(a) (b)

Figura 5.6: Correspondência de pixeis: (a) imagem resultante da etapa 1 com a projeção da curva gerada a partir de $\hat{\lambda}_{k|k-1}$ e (b)imagem resultante da etapa de correspondência.

5.6 ETAPA DE ESTIMAÇÃO DOS PARÂMETROS DA CURVA

A etapa de estimação de parâmetros λ_k pode ser realizada por um dos cinco algoritmos estimadores implementados. Portanto, para demonstrar o resultado obtido por cada um para um mesmo problema, será adotada apenas uma imagem capturada em t_k, que é a mesma imagem apresentada e trabalhada pelas etapas 1-3 nas Figuras 5.4-5.6.

A Figura 5.7 apresenta o resultado obtido por cada algoritmo estimador para um mesmo caso, a imagem original Fig. 5.7 (a). Tal imagem foi escolhida por apresentar além da marca artificial, uma marca natural altamente expressiva no teto (laje com canos fixados), e portanto, pode ser confundida com a trajetória. A Figura 5.6 (b) apresenta o resultado do processamento da etapa 3 sobre a imagem da Figura 5.7 (a). Os pixeis dessa imagem são os candidatos para a estimação. Pode-se observar pela imagem que boa parte deles não condizem com a trajetória. Portanto,

Estimadores	Tempo de execução em ms para uma imagem
Mínimos Quadrados	836.517432
M-Estimadores	1429.311059
RANSAC	1570.825702
Filtro de Kalman	839.343122
Filtro de Kalman Robusto	1640.251489

Tabela 5.1: Tempo de execução para cada algoritmo da etapa de estimação da curva.

verificou-se a resposta de cada algoritmo estimador para esse caso.

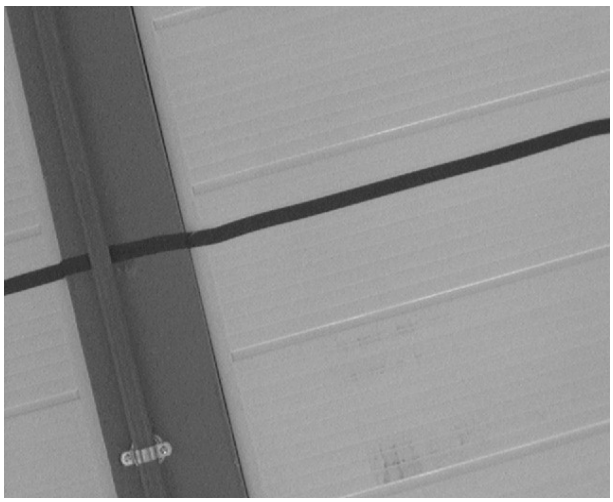
Ambas classificações (não-seqüencial e seqüencial) apresentaram bons resultados na estimação de parâmetros, embora cada uma com suas particularidades. Todos esses resultados são mostrados e analisados nas próximas subseções.

Para cada algoritmo estimador, foi medido o tempo de processamento gasto para execução de todas as etapas do sistema para uma única imagem adquirida. Tais tempos estão disponíveis na Tabela 5.1. Vale lembrar que tais tempos são referentes à execução do sistema em máquina com configuração descrita na Seção 5.1 deste capítulo. Pela tabela observa-se que os algoritmos com maior tempo de execução são RANSAC, o Kalman Robusto e os de menor tempo Mínimos Quadrados e Kalman. Tais tempo dependem das características de cada algoritmo que serão analisadas nas seguintes subseções.

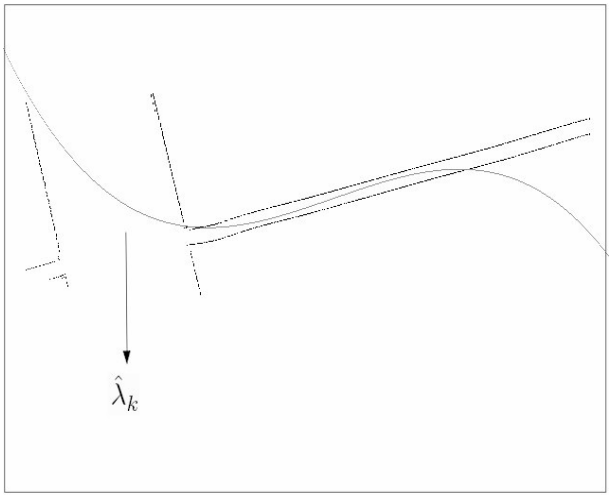
5.6.1 Algoritmo Mínimos Quadrados

O algoritmo de Mínimos Quadrados faz parte da classe dos não-seqüenciais ou em lote, na qual todos os pixeis candidatos são utilizados para a estimação. O resultado obtido com Mínimos Quadrados para a estimação de λ_k dados os pixeis da imagem de t_k (Fig. 5.7 (a)), obtidos na etapa 3, pode ser observado na Figura 5.7 (b). Nela são mostradas os pixeis pré-selecionados pela etapa de correspondência de pixeis e sobre eles é projetada a curva formada a partir de $\hat{\lambda}_k$ estimado.

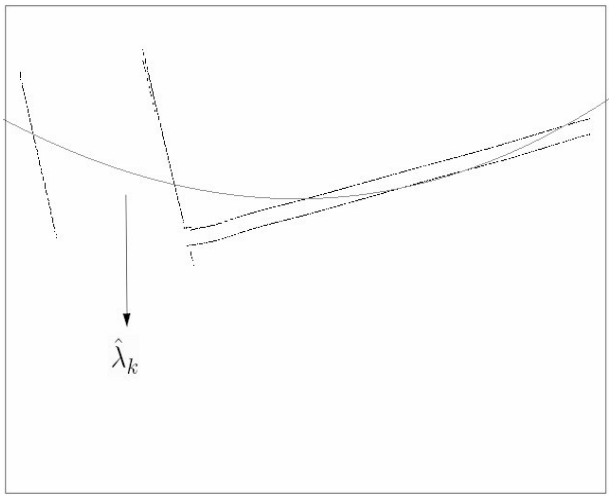
A busca de uma função que mais aproxime dados observados dos estimados através da minimização da soma dos quadrados dos resíduos é o objetivo do algoritmo. Entretanto, a estimação



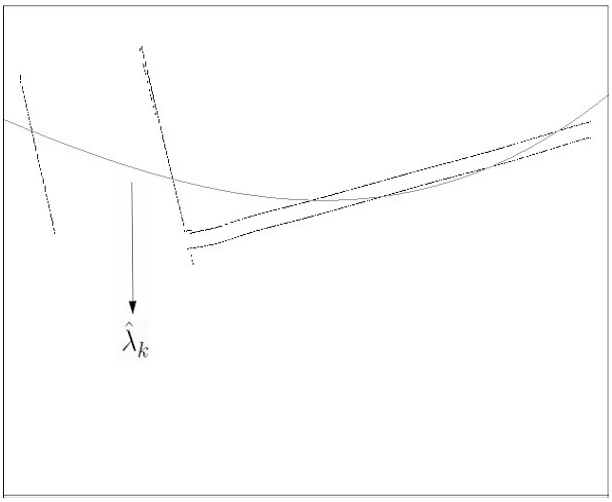
(a) Imagem original



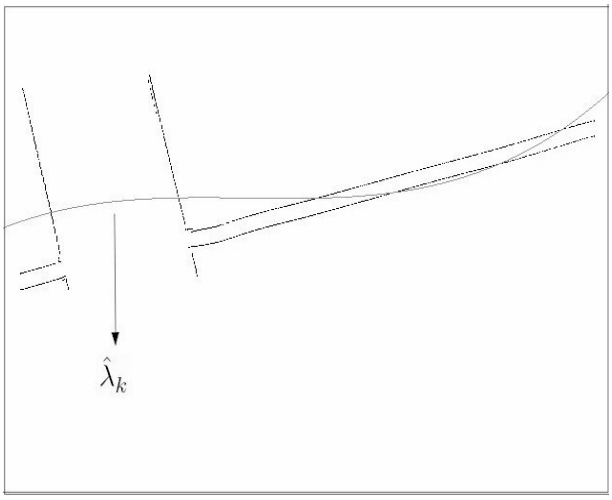
(b) Mínimos Quadrados



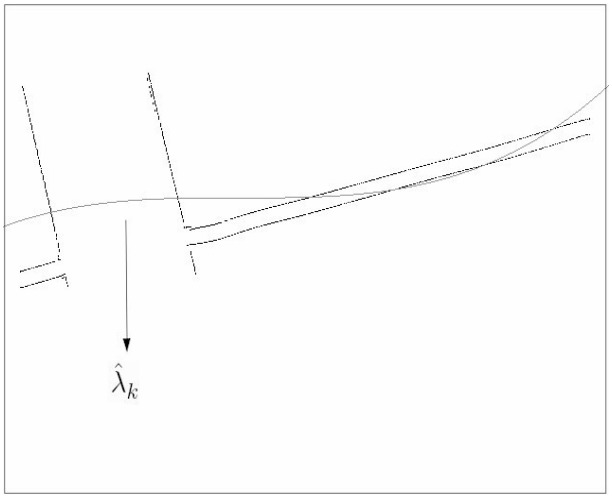
(c) M-estimador



(d) RANSAC



(e) Filtro de Kalman



(f) FKEIR

Figura 5.7: Estimação de λ_k. Em (a), a imagem original, as demais imagens (b,c,d,e) apresentam os pixeis pré-selecionados de (a) ao final da etapa 3 do sistema e o $\hat{\lambda}_k$ obtido por cada um dos algoritmos estimadores implementados.

sofre grande influência dos pontos da imagem que não fazem parte da trajetória, como pode ser observados pela (Figura 5.7 (b)). Como mencionado na Seção 4.4.1.2 do capítulo 4, comprova-se que a influência dos *outliers* causa um desvio na estimação prejudicando a obtenção de $\hat{\lambda}_k$ que conseqüentemente reflete numa estimação ruim na detecção da trajetória. No caso da estimação por mínimos quadrados, Figura 5.7 (b), a trajetória foi detectada. Entretanto, a curva gerada através de $\hat{\lambda}_k$ sofreu perturbações nas extremidades causadas pela influência de pixeis que não fazem parte da trajetória.

Justamente por não realizar uma seleção dos pixeis que mais se adequam ao modelo ou por garantir a mesma importância para $outliers$ e $inliers$ apenas por fazerem parte do conjunto de amostras, a estimação por mínimos quadrados é prejudicada. Tal fato pode ser observado pela Figura 5.7 (a), a qual mostra que a laje detectada na vertical não faz parte da trajetória, e sim do ambiente.

Pela Tabela 5.1 observa-se que o estimador de Mínimos Quadrados é o que gasta menos tempo para execução, isso pelo fato de calcular $\hat{\lambda}_k$ de uma vez a partir dos pixeis disponíveis tentando minimizar a soma dos quadrados dos resíduos. Portanto, não existem mecanismos de seleção de pixeis da amostra e nem cálculo iterativo de $\hat{\lambda}_k$.

5.6.2 Algoritmo M-Estimador

O algoritmo de M-estimadores caracteriza-se por adicionar ao algoritmo de Mínimos Quadrados robustez. No capítulo 4 foi citado que pontos extremos com resíduos arbitrários e que não condizem com o modelo podem ter uma influência grande no resultado da estimação por mínimos quadrados, como é o caso da figura 5.7 (b).

A robustez do M-Estimador está na tentativa de limitação da influência dos *outliers* dos dados de entrada. Os M-Estimadores, diferente dos Mínimos Quadrados, aplica pesos aos pixeis candidatos à estimação. Esse peso é determinado através do uso de algumas funções que aumentam "menos rapidamente"a soma dos quadrados dos resíduos, ao contrário dos estimadores de mínimos quadrados que crescem rapidamente. Dessa forma, pela Figura 5.7 (c) em comparação com a Figura 5.7 (b) percebe-se uma pequena redução da influência dos *outliers*, comprovando que a robustez do M-estimador frente aos Mínimos Quadrados.

O M-estimador realiza um cálculo iterativo de pesos para cada um dos pixeis candidatos. Dessa forma garante menor influência aos pixeis que não condizem com o modelo e maior influência aos demais. Para obter o $\hat{\lambda}_k$ a partir do qual foi gerada a curva exposta na Figura 5.7 (c), foram realizadas duzentas iterações.

O M-Estimador por ser um Mínimos Quadrados robusto garante melhores resultados do que o Mínimos Quadrados. O fato de agregar pesos às amostras baseado em métricas robustas causa um efeito de melhora nos resultados. Entretanto, todos os pixeis continuam atuando no cálculo de $\hat{\lambda}_k$, embora uns com menor importância que outros, os outliers da laje do teto continuam a influenciar na estimação 5.7 (c).

5.6.3 Resultados do Algoritmo RANSAC

O algoritmo estimador RANSAC seleciona iterativamente, a partir dos pixeis candidatos, um conjunto de pixeis cujo resíduo seja inferior à um determinado valor. Uma função custo que computa a qualidade do modelo encontrado e se tal modelo for aceitável, esse será o conjunto consenso utilizado para estimar λ_k por M-estimador. Dessa forma, apenas os pixeis com resíduo aceitável pelo algoritmo serão utilizados para a estimação. Essa seleção do conjunto consenso garante ao RANSAC boa tolerância aos *outliers* existentes nos dados de entrada.

A Figura 5.7 (d) apresenta a curva gerada a partir de $\hat{\lambda}_k$ estimado por RANSAC. Pelo algoritmo, o conjunto consenso foi selecionado em 100 iterações. Após a seleção desse conjunto executou-se o algoritmo M-Estimador que estimou $\hat{\lambda}_k$ a partir do conjunto consenso.

Pela Figura 5.7 (d), em comparação com os demais algoritmos adotados na classificação não-seqüencial, percebe-se que tanto o RANSAC quanto o M-Estimador conseguem interpretar e suavizar dados que contenham uma porcentagem significativa de erros. No caso apresentado, tanto o M-estimador quanto o RANSAC obtiveram resultados semelhantes. Entretanto, pelas características do algoritmo RANSAC, percebe-se que ele é mais robusto que o M-Estimador e o Mínimos Quadrados. A seleção realizada por ele dos pixeis reduz a quantidade de erros no modelo e conseqüentemente elimina boa parte dos *outliers* contidos nos dados de entrada. Além da seleção de um conjunto consenso de pixeis, ele utiliza o M-Estimador que agregará pesos aos pixeis tornando a estimação ainda mais robusta.

Quanto ao tempo necessário para estimar parâmetros, por ser um algoritmo iterativo, o RANSAC, dos três algoritmos apresentados para a classificação não-seqüencial, é que mais necessita de tempo para ser executado.

5.6.4 Filtro de Kalman

Como parte da abordagem seqüencial para a estimação de parâmetros da curva, o filtro de Kalman caracteriza-se por ser um algoritmo processador de dados recursivo e ótimo. Seu caráter ótimo está ligado a vários aspectos do algoritmo, dentre eles o fato de poder incorporar à estimação toda informação à medida em que essa é disponibilizada. Mesmo tratando-se de um filtro de medição linear, tal filtro apresentou bons resultados na estimação dos parâmetros no caso não linear apresentado.

A Figura 5.7 (e) demonstra o resultado obtido para a estimação de λ_k. Em comparação com os algoritmos não-seqüenciais, o Filtro de Kalman mostrou-se menos sensível aos *outliers*, estimando λ_k mais próximo da trajetória no instante t_k.

O filtro de Kalman processa toda a informação disponível, ou seja todos os pixeis candidatos obtidos na etapa 3 do sistema, mesmo que existam muitos dados imprecisos, para estimar os parâmetros de interesse $\hat{\lambda}_k$, com descrição estatística de ruídos, medição de erro (matriz de covariâncias) e incerteza da dinâmica do modelo e qualquer informação *a priori* do sistema, que nesse caso é $\hat{\lambda}_{k|k-1}$. Todas essas informações agregadas pelo filtro contribuem para a obtenção de uma estimação ótima [46].

Por ser recursivo, o filtro de Kalman não necessita imediatamente de toda a informação disponível do sistema para a estimação. Ou seja, à medida que a informação é apresentada ela é incorporada à estimação. Por isso, a cada processamento do filtro, é incorporada em sua nova estimação toda a informação calculada anteriormente através do uso de $\hat{\lambda}_{k|k-1}$ e $\mathbf{P}_{k|k-1}$ para a obtenção de $\hat{\lambda}_k$. Se a informação *a priori* $\hat{\lambda}_{k|k-1}$, que é utilizada pelo Filtro, estiver próxima de uma solução secundária, a correção fará com que haja um desvio, conduzindo a estimação para essa solução secundária. Ou seja, mesmo que exista uma grande quantidade de *inliers* no conjunto de amostras, o filtro confiará mais na informação $\hat{\lambda}_{k|k-1}$. Além da informação *a priori*, o filtro também possui fator que influencia na estimação. Trata-se do ganho que é influenciado por

um valor empírico, determinado pelo usuário que se for próximo de zero aumentará a influência dos resíduos no modelo, caso contrário alivia a influencia dos resíduos. Dessa forma, prova-se que não é uma tarefa fácil implementar e experimentar estimações estocásticas.

Quanto ao tempo de execução, o Filtro de Kalman gasta praticamente a mesma quantidade de tempo necessária ao estimador de Mínimos Quadrados (Tabela 5.1). Por se tratar de um algoritmo cujas iterações não ultrapassam a quantidade de pixeis candidatos da amostras, é mais rápido que os demais e de tempo semelhante ao Mínimos Quadrados.

O fato de o Filtro de Kalman ser um estimador para o caso linear, pode resultar em estimações não tão precisas em alguns momentos da trajetória. Embora o filtro de Kalman tenha realizado uma boa estimação em todos os casos apresentados durante a fase de experimentação, esse tipo de problema pode ocorrer pois as medidas nesse caso são não lineares. Portanto, para verificar o resultado obtido através de um filtro de Kalman para casos não lineares, foram testadas as mesmas seqüências de imagens com o Filtro de Kalman Estendido Iterativo Robusto - FKEIR. Esse, é o filtro de Kalman para o caso não-linear (Filtro de Kalman Estendido Iterativo) acrescido de métrica robusta e cujos resultados serão apresentados na próxima subseção.

5.6.5 Algoritmo Filtro de Kalman Estendido Iterativo Robusto

Como mencionado na subseção anterior, em alguns casos o Filtro de Kalman pode não ser suficiente para estimar os parâmetros da curva. O sistema linearizado é apenas uma aproximação do sistema real, o que invalida todas as propriedades ótimas e de convergência do Filtro de Kalman [54]. Entretanto, a convergência do filtro de Kalman depende de uma série de fatores: estimação inicial ($\hat{\lambda}_{k|k-1}$), a não linearidade das equações, a ordem na qual as medidas são processadas e as próprias medidas. O Filtro de Kalman Estendido Robusto é um Filtro de Kalman para o caso não linear. Ele ameniza os problemas citados do filtro Kalman reduzindo a influência dos *outliers* com auxílio de um estimador robusto, um M-estimador.

A Figura 5.7 (f) mostra o resultado obtido com FKEIR para o caso apresentado. Como pode ser observado, o λ estimado pelo FKEIR, devido às suas características apresentadas no capítulo anterior, sofre menor influencia de *outliers* do que os estimadores não-seqüenciais. Mas, para o caso apresentado, ao observar as Figuras 5.7 (e) e (f) percebe-se que ambos filtros (Kalman e

FKEIR) obtiveram resultados semelhantes.

O Filtro de Kalman Estendido Robusto calcula para uma quantidade de iterações determinando o valor de $\hat{\lambda}_k$ para cada amostra. No caso do resultado apresentado na Figura 5.7 (f), foram executadas 150 iterações. A cada iteração utiliza-se uma métrica robusta para balancear a influência de *outliers*. Tal métrica agrega maior robustez ao algoritmo reduzindo ou aumentando a influência dos resíduos, e nesse caso serve para propagar corretamente apenas os componentes das amostras que são relevantes. Caso não ocorra um mapeamento entre o valor medido e o estado $\hat{\lambda}_k$, o ganho de Kalman é influenciado de tal forma que ele fica bastante sensível à porção do resíduo que afeta o estado, e se não houver esse mapeamento o filtro rapidamente divergirá. Entretanto, com a adição da métrica robusta ao algoritmo, o algoritmo fará com que o estimador gere uma solução próxima da global e não uma secundária. Porém, pode acontecer de não haver esse mapeamento entre medida e estado estimado, e conseqüentemente a estimação de um $\hat{\lambda}_k$ ruim, o que pode ser causado por fatores citados inicialmente como a estimação inicial ($\hat{\lambda}_{k|k-1}$), a não linearidade das equações, a ordem na qual as medidas são processadas e as próprias medidas.

O tempo de execução gasto pelo FKEIR é próximo ao tempo gasto pelo RANSAC e M-Estimador, dependendo da quantidade de iterações. Porém com os resultados observados (Figura 5.7), é preferível optar por um filtro ótimo na estimação de parâmetros. Pois, os melhores resultados de estimação foram obtidos através dos algoritmos seqüenciais.

6 CONCLUSÕES

Neste trabalho foi apresentado um sistema de visão computacional implementado para o problema de rastreamento de trajetórias formadas por curvas para a navegação de robôs móveis. Em tal solução, a ênfase foi dada na arquitetura do sistema de rastreamento, bem como nos algoritmos estimadores dos parâmetros das curvas implementados. As entradas do sistema são as imagens adquiridas e dados provenientes da odometria para predição dos parâmetros estimados.

Na etapa de aquisição e processamento de imagens, cada imagem adquirida é inicialmente processada com técnicas de extração de bordas por Filtro de Canny e posteriormente por limiarização e conectividade-8. As técnicas adotadas no processamento de imagens foram suficientes para eliminar uma grande quantidade de *outliers* que poderiam causar divergências na estimação dos parâmetros da curva da trajetória. E mesmo em ambiente sob condições adversas, pequenos ajustes no valor da limiarização permitiram que a aplicação de equalização de histograma e outros filtros para amenização da baixa luminosidade fosse desnecessária.

Na etapa de predição dos parâmetros da curva, os parâmetros $\hat{\lambda}_{k-1}$ da curva estimados no instante de tempo t_{k-1} são atualizados de acordo com os dados recebidos da odometria. Isto é, o movimento realizado pelo robô entre os instantes t_{k-1} e t_k é refletido na estimação $\hat{\lambda}_{k-1}$ gerando $\hat{\lambda}_{k|k-1}$. O uso dos dados da odometria garantem à predição mais confiabilidade pois, mesmo possuindo erros, esses dados reduzirão ainda mais a quantidade de *outliers* nos novos dados a serem utilizados para a estimação de parâmetros de $\hat{\lambda}_k$. Refletindo o movimento do robô, a curva formada com $\hat{\lambda}_{k|k-1}$ ficará mais próxima dos pixeis na imagem do instante t_k que fazem parte trajetória, contribuindo assim para uma boa estimação de $\hat{\lambda}_k$.

A imagem gerada na etapa de aquisição e processamento juntamente com o $\hat{\lambda}_{k|k-1}$, formarão a entrada da etapa de correspondência de pixeis. Nessa etapa os pixeis que restaram da imagem carregada e processada no instante t_k passaram por uma seleção que, através da ampliação da curva formada por $\hat{\lambda}_{k|k-1}$, pixeis da imagem que corresponderam aos pixeis da curva ampliada foram selecionados, os demais eliminados. Assim, a etapa de correspondência de pixeis elimina mais *outliers* da imagem adquirida em t_k. Finalmente, os pixeis selecionados foram passados para

um estimador que responsabiliza-se por calcular $\hat{\lambda}_k$. Para essa etapa, foram implementados cinco algoritmos, os não-seqüenciais Mínimos Quadrados, M-Estimador e RANSAC, e os seqüenciais filtro de Kalman e Filtro de Kalman Estendido Iterativo Robusto.

Os algoritmos estimadores implementados foram analisados quanto à capacidade de reconhecimento da trajetória obtida através de imagens capturadas pela câmera do robô. Para tal foram apresentados no capítulo anterior os resultados obtidos em cada etapa executada pelo sistema para o processo de rastreamento. Os cinco algoritmos estimadores implementados para a etapa 4 foram analisados quanto ao desempenho e robustez ante os *outliers* presentes nas imagens. Embora cada um possua características particulares quanto à tolerância a *outliers* e tempo de execução, todos conseguiram, em boa parte da trajetória, obter uma curva aceitável para a predição. Para as situações apresentadas no capítulo 5 e pelas características próprias dos algoritmos RANSAC e Filtros de Kalman e Kalman Estendido Iterativo Robusto, esses se comportaram melhor na tarefa de rastreamento. Mesmo utilizando o teto como ambiente, o qual não possui grande quantidade de objetos, o revestimento do mesmo pode provocar a existência de uma grande quantidade de *outliers*, bem como lajes, luminárias, canos e outras marcas naturais. Esses pontos, não pertencentes ao modelo, influenciaram menos a estimação pelos algoritmos.

Já os algoritmos Mínimos Quadrados e M-Estimador, para as situações apresentadas no capítulo 5, foram os mais influenciados por características próprias do ambiente. O primeiro, por dar a mesma prioridade para *inliers* e *outliers* e possuir uma função objetivo que cresce grandemente com o aumento dos resíduos. O M-estimador ameniza os problemas dos Mínimos Quadrados ao adicionar métricas robustas para reduzir o crescimento da função objetivo, entretanto, utiliza de todo o conjunto de amostras para a estimação. Mas, mesmo com curvas geradas por uma estimação ruim de ambos os algoritmos, para boa parte das imagens adquiridas pelo sistema, eles conseguiram estimar a trajetória.

Nas situações apresentadas no capítulo 5, os algoritmos seqüenciais mostraram-se melhores que os não-seqüenciais. Entretanto, durante uma seqüência grande de imagens, as quais representam uma trajetória completa percorrida pelo robô, os estimadores não-seqüências apresentaram melhores resultados, é o caso do experimento a ser apresentado no anexo.

Quanto ao desempenho, os algoritmos mais rápidos para a execução de um ciclo de proces-

samento (incluindo todas as etapas) foram Mínimos Quadrados e Filtro de Kalman, os outros por incluírem uma maior quantidade de iterações para a obtenção de uma estimação mais robusta, gastaram mais tempo. Entretanto, o Filtro de Kalman com menor número de iterações que o M-Estimador e RANSAC e conseqüentemente menor tempo de execução, obteve melhores resultados que os dois primeiros, comprovando que o Kalman é um conjunto de equações que proporcionam um eficiente método computacional recursivo para estimar o estado de um processo [46].

Como o modelo a ser estimado não é linear e o Filtro de Kalman é ótimo para o caso linear, um outro estimador implementado foi o Filtro de Kalman Estendido Iterativo Robusto o qual é o filtro ótimo para o caso não linear com métrica robusta. Para as trajetórias apresentadas, tal filtro também realizou uma boa estimação. A única dificuldade encontrada é a existência de fatores como estimação inicial ($\hat{\lambda}_{k|k-1}$), quantidade de iterações e as próprias medidas que influenciam na estimação.

O estimador de Mínimos Quadrados e o Filtro de Kalman sofreram maior influência dos *outliers*. Quanto ao desempenho, na execução da mesma trajetória para todos os algoritmos, foi comprovado que os estimadores mais rápidos são mínimos quadrados, filtro de Kalman e filtro de Kalman Robusto.

Os resultados obtidos e observados no capítulo anterior mostraram que o sistema implementado permite o rastreamento da trajetória a partir das informações adquiridas através das imagens capturadas do ambiente pela câmera do robô e dos dados da odometria.

Uma sugestão para pesquisas futuras é o aprimoramento do sistema apresentado no que diz respeito à ampliação dos modelos de curva parametrizados, ou seja, aceitação da parametrização de modelos de curva com polinômios superiores a terceira ordem pelo sistema implementado, pois os estimadores implementados já possuem essa capacidade.

Naturalmente, a próxima etapa a ser desenvolvida no sistema consiste na derivação de algoritmos de controle de trajetória que é o acoplamento dos sistemas de controle do robô móvel Omni ao sistema de rastreamento de trajetórias aqui apresentado, causando assim uma fusão de informações pertinentes à trajetória observada no teto e as informações obtidas de outros sensores para evitar colisão.

Além das sugestões já mencionadas, o sistema de controle juntamente com o sistema de rastreamento implementado poderiam ser utilizados para que o robô realize seus movimentos automaticamente, baseado nas informações de saída dos dois sistemas citados.

Finalmente, uma restrição do sistema implementado e que corresponde a um importante problema em rastreamento de trajetórias para navegação de robôs móveis, é a determinação do ponto de partida da trajetória. Sem informação *a priori*, ao iniciar o sistema, a primeira imagem capturada passa apenas pelas etapas de processamento e estimação dos parâmetros da curva. Tendo em vista que a curva estimada através da primeira imagem está correta, o operador humano permite a execução do sistema. Esse restrição consiste em um empecilho para a total automatização da tarefa de navegação do robô móvel. Dessa forma, um aprimoramento do sistema proposto seria o desenvolvimento de um algoritmo que permitisse identificar o ponto de partida da trajetória.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] C. S. R. Aguiar, *Estudo da plataforma móvel omni e calibração de câmera para localização 3-d*, Monografia de graduação apresentada à Engenharia Mecatrônica - Faculdade de Tecnologia - Universidade de Brasília, Brasília - DF, Março 2006.

[2] O. A. Aider, P. Hoppenot, and E. Colle, *A model-based method for indoor mobile robot localization using monocular vision and straight-line correspondences*, Robotics and Autonomous Systems, vol. 52, 2005, pp. 229–246.

[3] K. O. Arras, J. A. Castellanos, R. Siegwart, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, *Feature-based multi-hypothesis localization and tracking for mobile robots using geometric constraints*, IEEE International Conference on Robotics and Automation, 2002.

[4] N. Ayache, I. Cohen, and I. Herlin, *Medical image tracking*, In A. Blake and A. Yuille, editors, Active Vision, MIT, 1992, pp. 285–302.

[5] R. Bischoff, *Recent advances in development of the humanoid service robot hermes*, 3rd EUREL Workshop and Masterclass - European Advanced Robotics Systems Development (Salford, U.K.), vol. I, 2000, pp. 125–134.

[6] A. Blake, *Applying visual curve tracking to graphics*, Computer Animation, 1995.

[7] A. Blake, M. Isard, and D. Reynard, *Learning to track curves in motion*, 33rd Conference on Decision and Control, 1994.

[8] A. Blake and Isard M., *Active contours*, Springer, 2000.

[9] J. Bloomenthal, *Introduction to implicit surfaces*, Kaufmann Publishers, Inc., 2001.

[10] G. A. BORGES, *Cartographie de l'environnement et localisation robuste pour la navigation de robots mobiles.*, Ph.D. thesis, U. Montpellier 2, Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier, 2002.

[11] G. A. Borges and M. J. Aldon, *Robustified estimation algorithms for mobile robot localization based on geometrical environment maps*, Robotics and Autonomous Systems, vol. 45, Elsevier, February 2003, pp. 131–159.

[12] A. J. Briggs, D. Scharstein, D. Braziunas, C. Dima, and P. Wall, *Mobile robot navigation using self-similar landmarks*, IEEE International Conference on Robotics and Automation, 2000.

[13] W. Burgard, A. Cremers, D. Fox, D. Hahnel, G. Lakemeyer, W. Schulz, D. Steiner, and S. Thrun, *Experiences with an interactive museum tour-guide robot*, Artificial Intelligence, vol. 114, 2000.

[14] J. Canny, *A Computational Approach to Edge Detection*, IEEE. Trans Pattern Analysis. and Machine Intelligence, vol. 8, Novembro 1986, pp. 679–698.

[15] E. Clabi, P.J. Olver, C. Shakiban, A. Tannenbaum, and S. Haker, *Differential and numerically invariant signature curves applied to object recognition*, International Journal Ccomputer Vision **26** (1998), no. 2, 107–135.

[16] J. J. Craig, *Introduction to robotics - mechanics and control*, second edition ed., Addison Wesley Longman, 1989.

[17] R. Durikovic, K. Kaneda, and H. Yamashita, *Dynamic contour: a texture approach and contour operations*, The Visual Computer, vol. 11, 1995, pp. 277–289.

[18] T. Edlinger and G. Wei, R. Dillmann, and T. LuK, Autonome mobile system - , Informatik Aktuell, 1995, pp. 142–151.

[19] M. A. Fischler and Bolles R. C., *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*, Comm. of the ACM, vol. 24, 1981, pp. 381–395.

[20] R. Frezza, *How much does one need to know about a curve in order to follow it with a nonholonomic vehicle?*, Proceedings of the Fourteenth International Symposium of Mathematical Theory of Networks and Systems, 2000.

[21] Y. Gdalyahu and D. Weinshall, *Flexible syntactic matching of curves and its application to automatic hierarchical classification of silhouettes*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, 1999, pp. 1312–1328.

[22] R. C. Gonzalez and R. E. Woods, *Processamento de imagens digitais*, Editora Edgard Blucher. Second Edition, 1992.

[23] G.D. Hager, D. Kriegman, E. Yeh, and C. Rasmussen, *Prediction of landmark features for mobile robot navigation*, IEEE International Conference on Robotics and Automation (Albuquerque, New Mexico), 1997.

[24] R. M. Haralick, H. Joo, C. N. Lee, X. Zhuang, V. G. Vaidya, and M. B. Kim, *Pose Estimation from Corresponding Point Data*, IEEE Transactions on Systems, Man, and Cybernetics, vol. 19, NovemberDecember 1989, pp. 1426–1446.

[25] D. Hermey and G. A. Watson, *Fitting Data with Errors in All Variables Using the Huber M-estimator*, SIAM Journal on Scientific Computing, vol. 20, 1999, pp. 1276–1298.

[26] B. K. P. Horn, *Robot vision*, The MIT Electrical Engineering and Computer Science Series. MIT Press, 1986.

[27] W. Hu, T. Downs, G. Wyeth, M. J. Milford, and D. Prasser, *A modified particle filter for simultaneous robot localization and landmark tracking in an indoor environment*, Australasian Conference on Robotics and Automation (Canberra, Australia), 2004.

[28] Z. H. Huang and F. S. Cohen, *Affine invariant b-spline moments for curve matching*, Image Processing, vol. 5, 1996, pp. 1473–1480.

[29] Yunqiang Chen Huangand and T. Yong Rui, *Parametric contour tracking using unscented Kalman filter*, Proceedings International Conference on Image Processing, vol. 3, 2002, pp. 613–616.

[30] P. J. Huber, *Robust statistics*, John Wiley & Sons, 1981.

[31] M. Isard and A. Blake, *Condensation-conditional density propagation for visual tracking*, Int. J. Computer Vision, vol. 29, 1998, pp. 5–28.

[32] H. Ishiguro, R. Sato, and T. Ishida, *Robot oriented state space construction*, In Proceedings of the Conference on Intelligent Robots and Systems, 1996, pp. 1496–1501.

[33] G. Jang, S. Kim, W. Lee, and I. Kweon, *Color landmark based self-localization for indoor mobile robots*, IEEE International Conference on Robotics and Automation, 2002.

[34] A. H. Jazwinski, *Stochastic processes and filtering theory*, Academic Press, 1970.

[35] T. Jin, S. Park, and J. Lee, *A study on position determination for mobile robot navigation in an indoor environment*, IEEE International Symposium on Computational Intelligence in Robotics and Automation, 2003.

[36] A. Kass, M. Witkin and D. Terzopoulos, *Active contour models*, International Journal of Computer Vision, vol. 1, 1988, pp. 321–331.

[37] M.H. Kim, S. C. Lee, and K. H. Lee, *Self-Localization of Mobile Robot with Single Camera in Corridor Environment*, Proc. of ISIE, vol. 3, junho 2001, pp. 1619–1623.

[38] Y. Le Corre, *Conception et commande d'un robot omnidirectionnel*, Ph.D. thesis, U. Montpellier 2, Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier, 1998.

[39] S. Levy, *Geometry formulas and facts - special plane curves*, 30th Edition of the CRC Standard Mathematical Tables and Formulas, CRC press, 1995.

[40] F. Leymarie and M. D. Levine, *Tracking deformable objects in the plane using and active contour model*, IEEE Trans. Pattern Anal. Mach. Intell, vol. 15, 1993, pp. 617–634.

[41] P. Li, T. Zhang, and B. Ma, *Unscented Kalman filter for visual curve tracking*, vol. 22, Julho 2003, pp. 157–164.

[42] L. Ljung, *System identification- theory for the user*, second edition, ed., Prentice-Hall PTR, 1999.

[43] S. Ma, *Conics-based stereo, motion estimation and pose determination*, International Journal Computer Vision, vol. 10, n.1, 1993.

[44] H. Madsen, P. Christensen, and K. Lauridsen, *Securing the operational reliability of an autonomous mini-submarine*, Reliability Engineering and System Safety, vol. 68, 2000, pp. 7–16.

[45] J. Matijevic, *Sojourner- the mars pathfinder microrover flight experiment,*, Technical Report 2315, Jet Propulsion laboratory California Institute of Technology, 1996.

[46] P. S. Maybeck, *The Kalman Filter: Anintroduction to Concepts*, Stochastic models, estimation and Control. Academic Press, 1979, pp. 3–16.

[47] T. McInerney and D. Terzopoulos, *A dynamic finite element surface model for segmentation and tracking in multidimensional medical images with application to cardiac 4d image analysis*, Computerized Medical Imaging and Graphics, vol. 19, 1995, pp. 69–83.

[48] I. Mikic, S. Krucinski, and J. D. Thomas, *Segmentation and tracking in echocardiographic sequences: active contours guided by optical flow estimates*, IEEE Transactions on Medical Imaging, vol. 17, 1998, pp. 274–284.

[49] J. L. Mundy and A. Zisserman, *Geometric invariance in computer vision*, MIT Press, 1992.

[50] Peterfreud N., *Robust tracking of position and velocity with Kalman snakes*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, 1999, pp. 564–569.

[51] B. Nagy and A. Kelly, *Trajectory generation for car-like robots using cubic curvature polynomials*, In Field and Service Robots 2001, 2001.

[52] J. Sato and Cipolla R., *Quasi-Invariant Parameterisations and Matching of Curves in Images*, International Journal of computer Vision, vol. 28, 1998, pp. 117–136.

[53] C. Schmid and Zisserman A., *The Geometry and Matching of Lines and Curves Over Multiple Views*, International Journal of computer Vision, vol. 40, 2000, pp. 199–233.

[54] J. Schutter, Geeter J., Lefebvre T., and Bruyninckx H., *Kalman filers: a tutorial*, Benelux Quaterly Journal on Automatic Control, vol. 40, December 1999, pp. 538–546.

[55] S. Se, Lowe D., and Little J., *Vision-based mobile robot localization and mapping using scale-invariant features*, Proceedings of the 2001 IEEE International Conference on Robotics and Automation (Seoul, Korea), 2001.

[56] R. Sim and G. Dudek, *Learning and evaluating visual features for pose estimation*, Proceedings of the Seventh International Conference on Computer Vision, 1999.

[57] F. Solina and R. Bajcsy, *Recovery of parametric models from range images: The case for superquadratics with global deformations*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, 1990, pp. 505–519.

[58] M. Sonka, V. Hlavac, and R. Boyle, *Imagem processing analysis, and machine vision.*, Second Edition, 1999.

[59] J. Tani and N. Fukumura, *Learning goal-directed sensory-based navigation of a mobile robot*, 1995, pp. 553–563.

[60] G. Taubin, *Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations, with applications to edge and range image segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 9, 1991, pp. 405–416.

[61] G. Taubin, F. Cukierman, S. Sullivan, J. Ponce, and D.J. Kriegman, *Parameterized families of polynomials for bounded algebric curve and surfaze fitting*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16, 1994, pp. 287–303.

[62] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, *Minerva: A second generation museum tour-guide robot*, IEEE International Conference on Robotics and Automation, 1999.

[63] E. Tunstel, M. Maimone, A. Trebi-Ollennu, J. Yen, R. Petras, and R. Willson, *Mars exploration rover mobility and robotic arm operational performance*, IEEE International Conference Systems, Man and Cybernetics, vol. 2, 2005, pp. 1807–1814.

[64] I. Ulrich, F. Mondada, and J. D. Nicoud, *Autonomous Vacuum Cleaner*, Robotics and Autonomous Systems, vol. 19, 1997, pp. 233–245.

[65] M. Unel and W. A. Wolovich, *On the contruction of complete sets of geometric invariants for algebric curves*, Advances in Aplied Mathematics, vol. 24, 2000, pp. 65–87.

[66] G. M. Vale and A. P. Dal Poz, *O processo de detecção de bordas de canny: fundamentos, algoritmos e avaliação experimental*, Simpósio Brasileiro de Geomática, Presidente Prudente (Presidente Prudente), 2002, pp. 292–303.

[67] G. V. Wichert, *Can Robots learn to see?*, IFAC Journal Control Engineering Practice, vol. 7, 1999, pp. 783–795.

[68] N. Xu, R. Bansal, and N. Ahuja, *Object segmentation using graph cuts based active contours*, IEEE International Conference on Computer Vision and Pattern Recognition (Madison, WI), 2003.

[69] H. Yalcin, Unel M., and Wolovich W., *Implicitation of Parametric Curves by Matrix Annihilation*, International Journal of Computer Vision, vol. 54, 2003, pp. 105–115.

[70] Z. Zhang, *Parameter Estimation Techniques: A Tutorial with Application to Conic Fitting*, vol. 15, 1997, pp. 59–76.

ANEXOS

A. RESULTADOS PARA O SISTEMA DE RASTREAMENTO COM SEQÜÊNCIAS DE IMAGENS

A Figura A.1 apresenta um desenho do teto do ambiente o qual foram adquiridas imagens para a análise de resultados descrita no capítulo 5. O teto possui marcas naturais e marcas artificiais contínuas adicionadas para a realização de testes. Nesta Figura, é apresentada a trajetória percorrida pelo robô guiado por *joystick* sinalizada por uma seta. Essa trajetória é composta por retas e curvas acentuadas.

Para apresentar o resultado do sistema de rastreamento para uma seqüência de imagens foram selecionadas as imagens da primeira curva da trajetória. Tal curva está marcada na Figura A.1 com um círculo. As imagens originais adquiridas (uma após a outra) pela câmera no momento em que o robô estava embaixo das marcas sinalizadas pelo círculo da Figura A.1, são mostradas na Figura A.2.

As Figuras A.3, A.4, A.5, A.6 e A.7 apresentam o resultado do sistema para a seqüência de imagens da Figura A.2. O sistema é executado continuamente. O cálculo de λ_k com o uso de Mínimos Quadrados, M-Estimadores e RANSAC é baseado nos pixeis da imagem adquirida e processada em t_k, pré-selecionados na etapa de correspondência de pixeis. O resultado para tais estimadores é apresentado nas Figuras A.3, A.4 e A.5, respectivamente.

O cálculo de λ_k com Filtro de Kalman e FKEIR é baseado nos pixeis da imagem adquirida e processada em t_k, selecionados pré-selecionados na etapa de correspondência de pixeis e na estimação atualizada $\lambda_{k|k-1}$. O resultado para tais estimadores é apresentado nas Figuras A.6, A.7, respectivamente.

Pelo resultado apresentado nas Figuras A.3, A.4, A.5, A.6 e A.7, observa-se que os estimadores não-seqüenciais obtiveram melhor resultado na estimação dos parâmetros da curva que representa a trajetória. Embora os seqüenciais alcançassem êxito em algumas estimações, como as apresentadas no capítulo 5, em outras sua estimação confiava mais no $\lambda_{k|k-1}$ do que nos pixeis pré-selecionados pela etapa de correspondência de pixeis. como é o caso das imagens das Figuras A.6 e A.7. Isso, devido à variáveis do Filtro de Kalman que possuem valores empíricos.

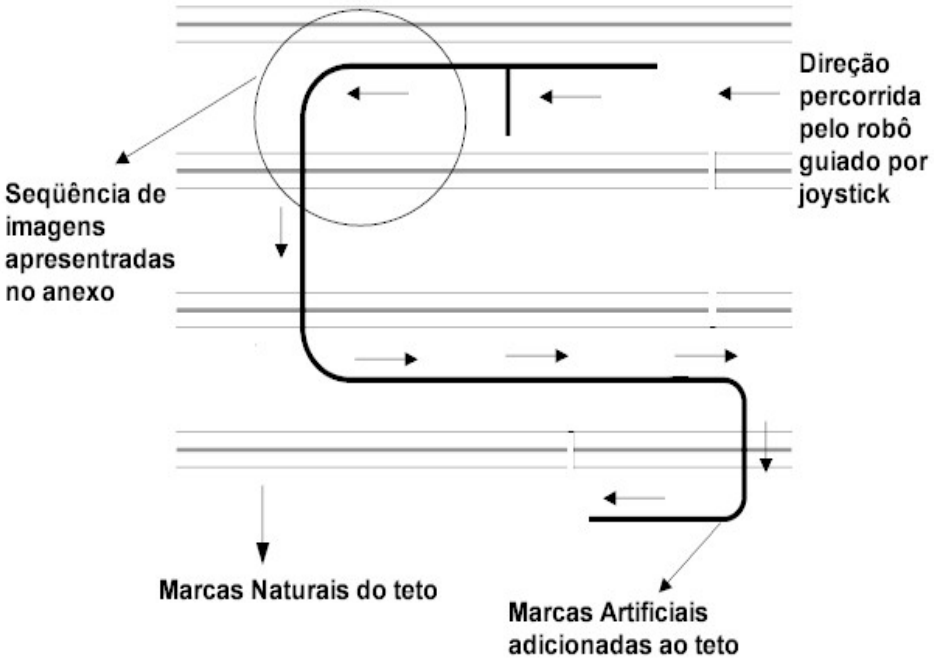
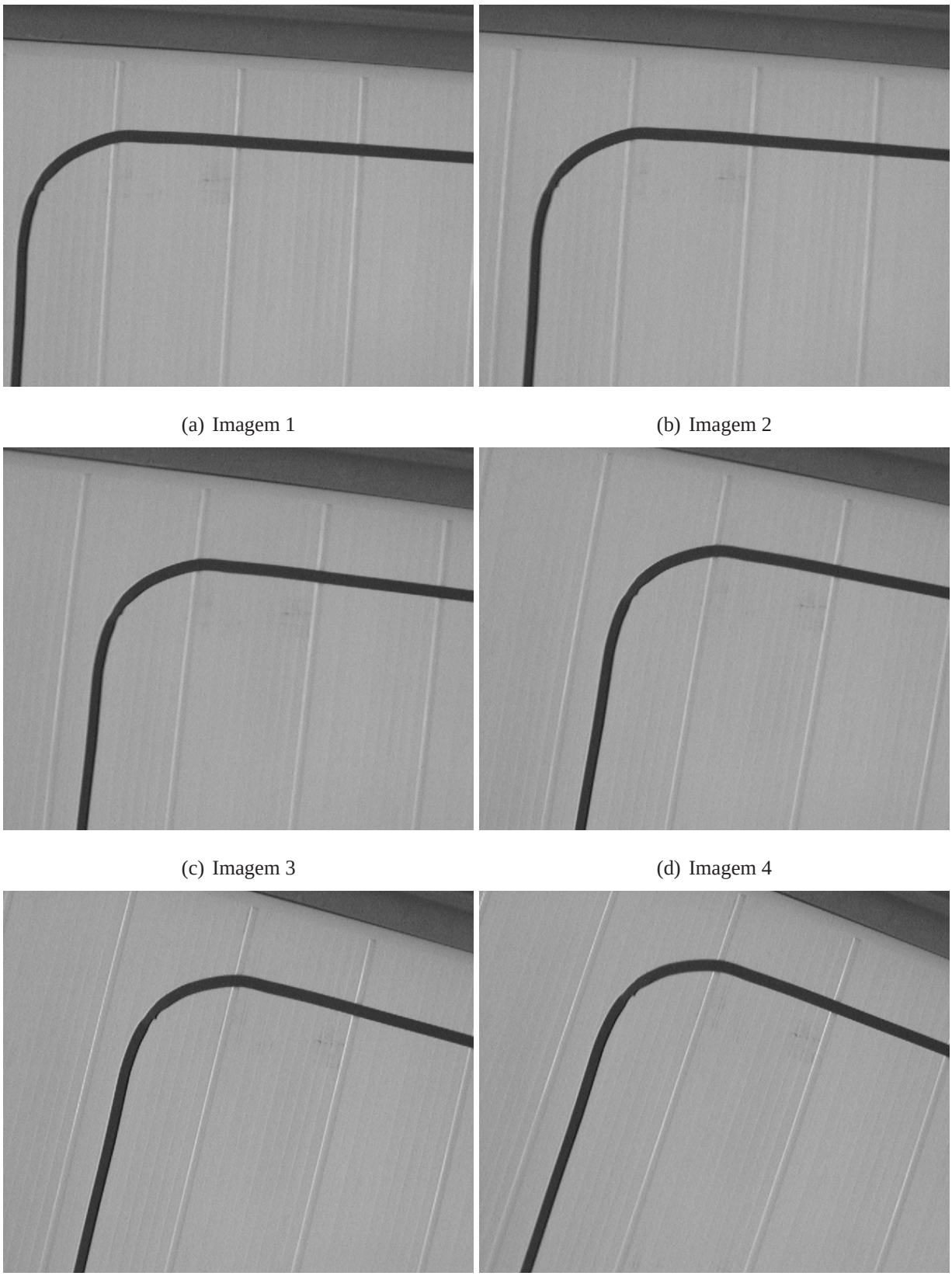


Figura A.1: Desenho do teto com marcas artificiais e naturais. O círculo indica o local de onde a seqüência de imagens da Figura A.2 do anexo foram adquiridas.

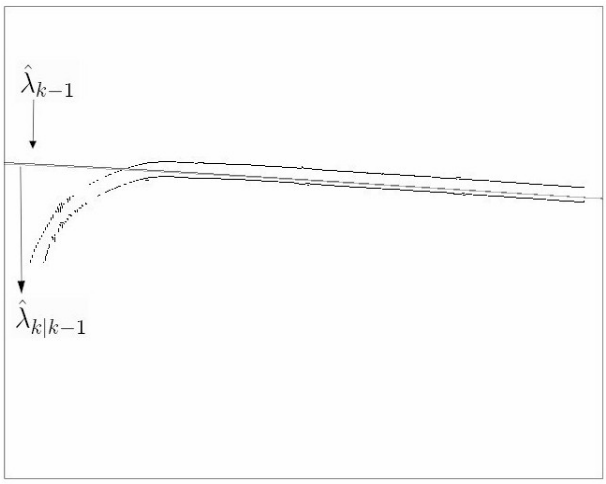


(a) Imagem 1 (b) Imagem 2

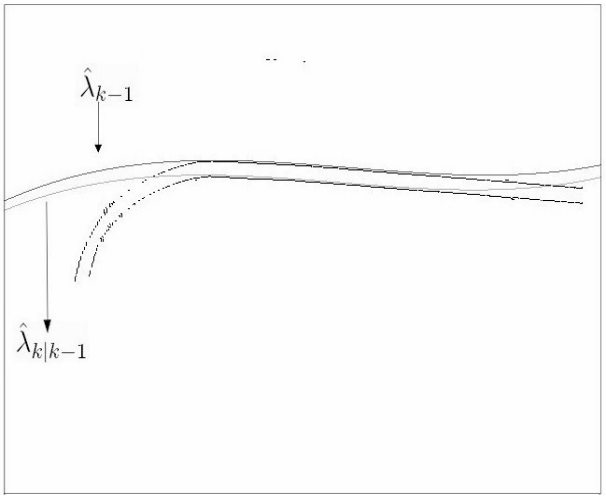
(c) Imagem 3 (d) Imagem 4

(e) Imagem 5 (f) Imagem 6

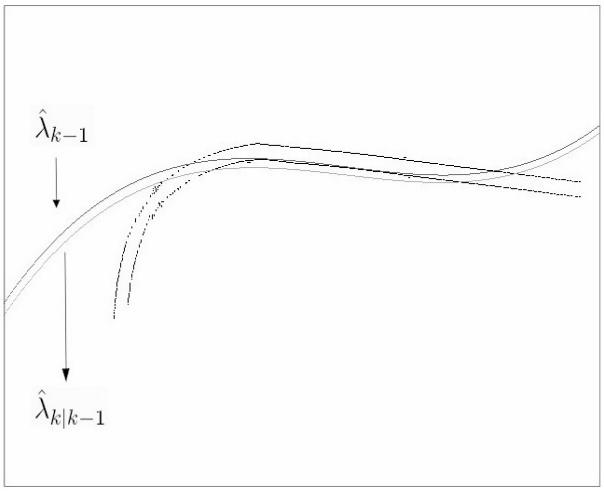
Figura A.2: Seqüência de imagens originais obtidas através da câmera do robô guiado por *joy-stick*.



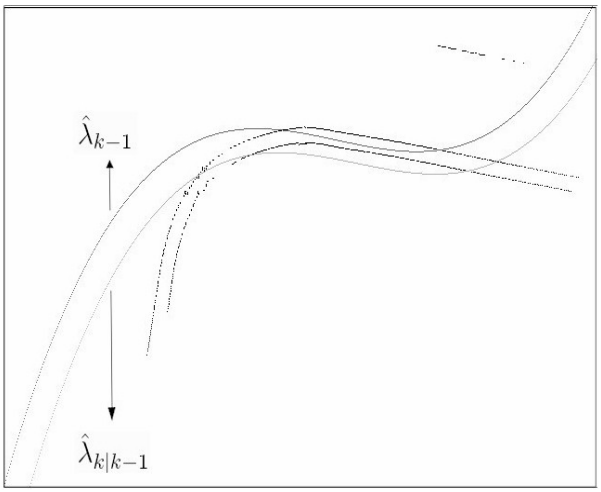
(a) Imagem 1



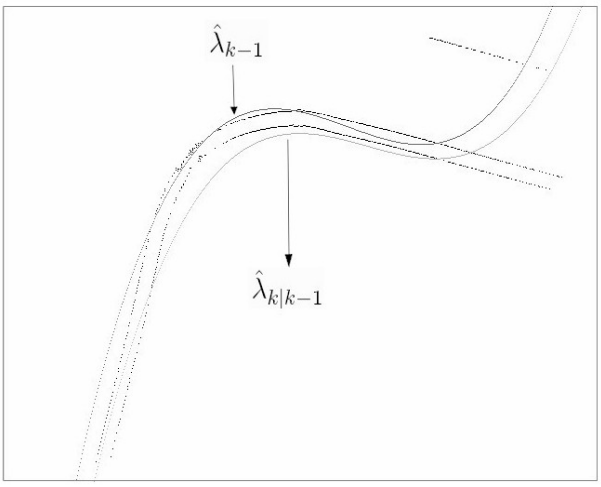
(b) Imagem 2



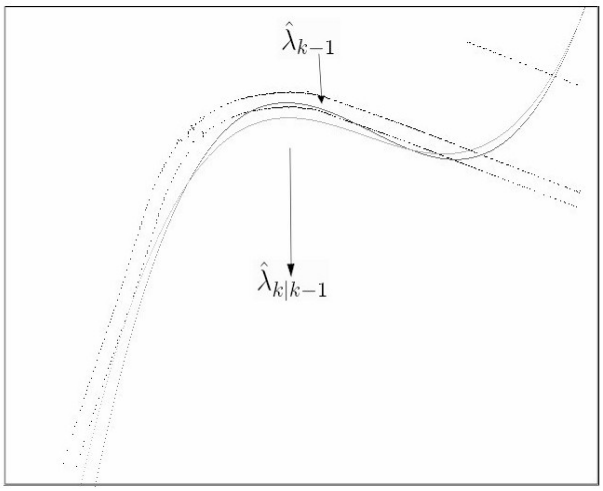
(c) Imagem 3



(d) Imagem 4

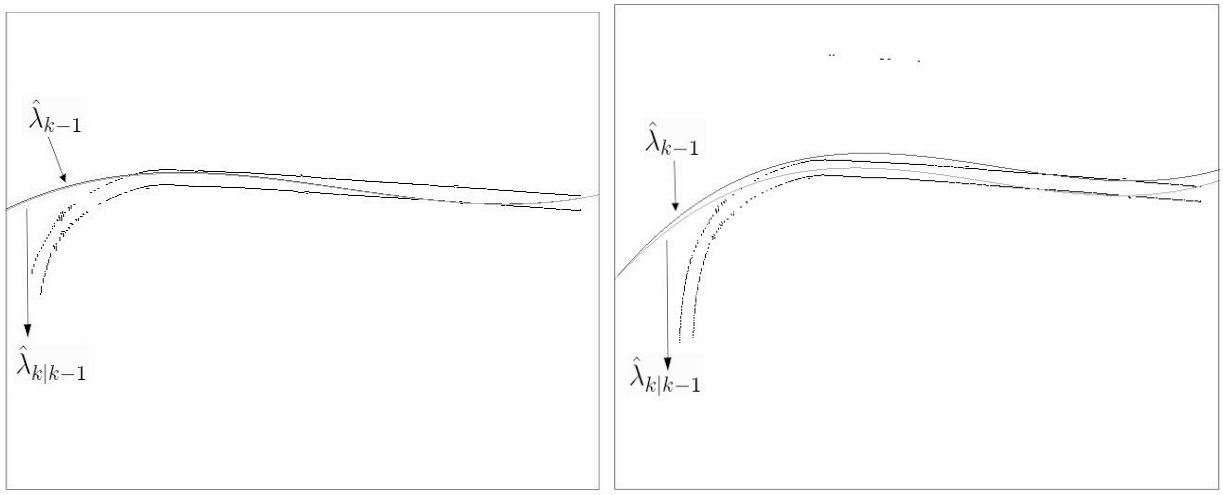


(e) Imagem 5

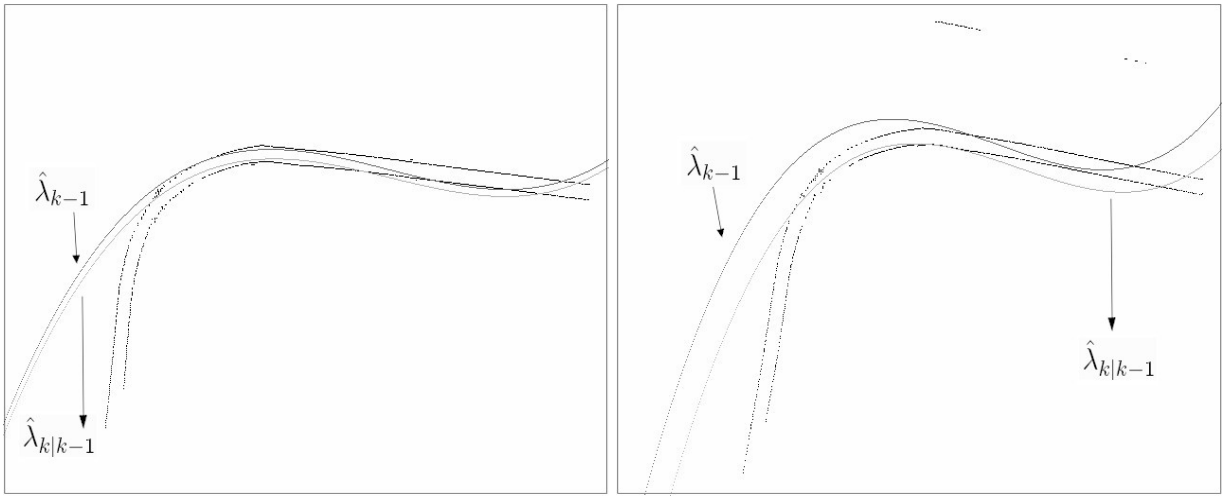


(f) Imagem 6

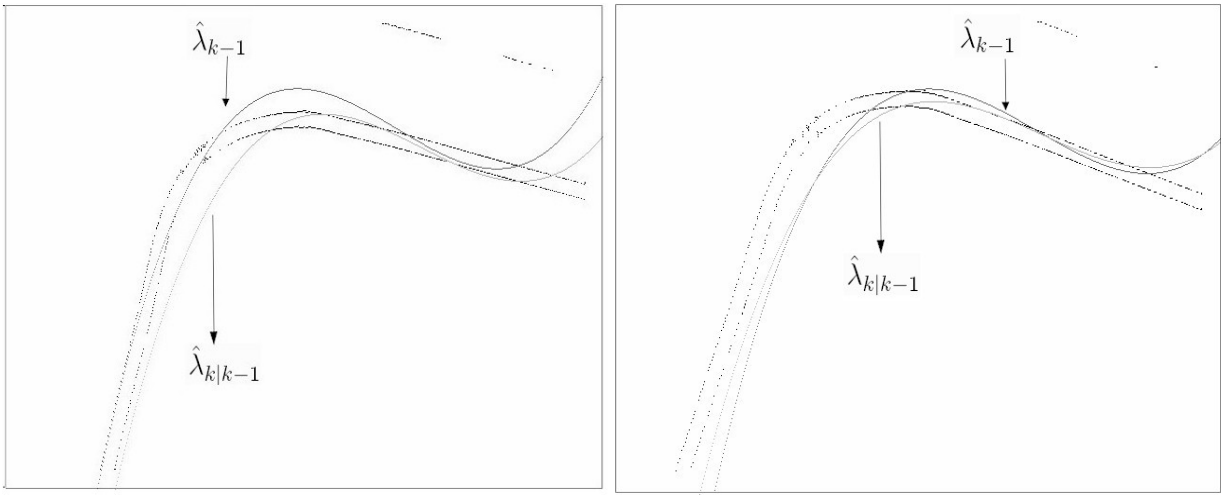
Figura A.3: Seqüência de imagens obtidas após o processamento do das etapas do sistema de rastreamento utilizando Mínimos Quadrados.



(a) Imagem 1 (b) Imagem 2

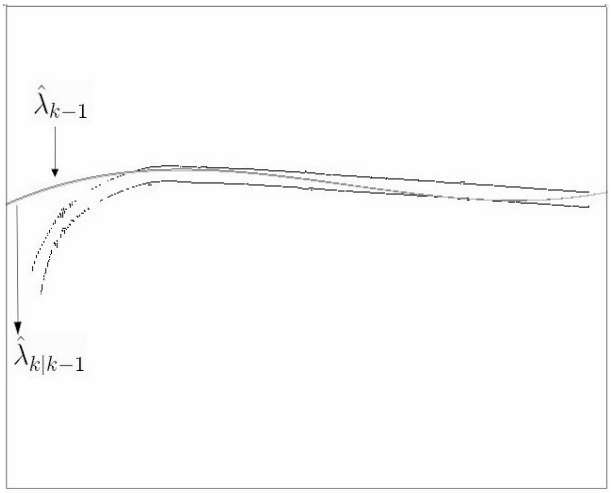


(c) Imagem 3 (d) Imagem 4

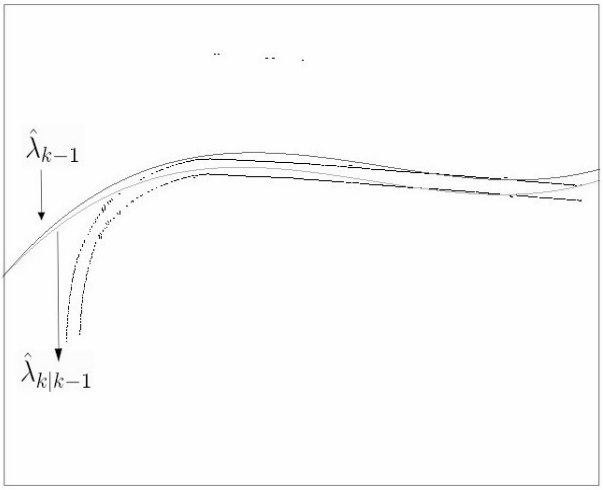


(e) Imagem 5 (f) Imagem 6

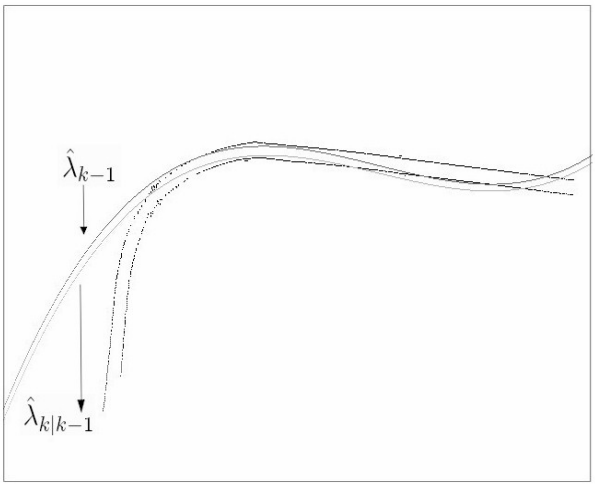
Figura A.4: Seqüência de imagens obtidas após o processamento do das etapas do sistema de rastreamento utilizando M-Estimador.



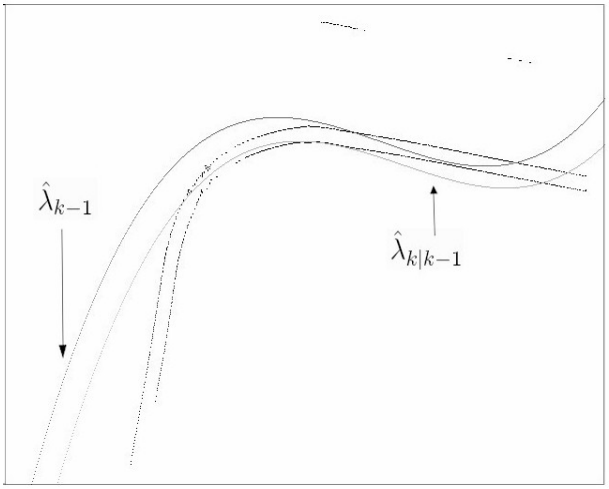
(a) Imagem 1



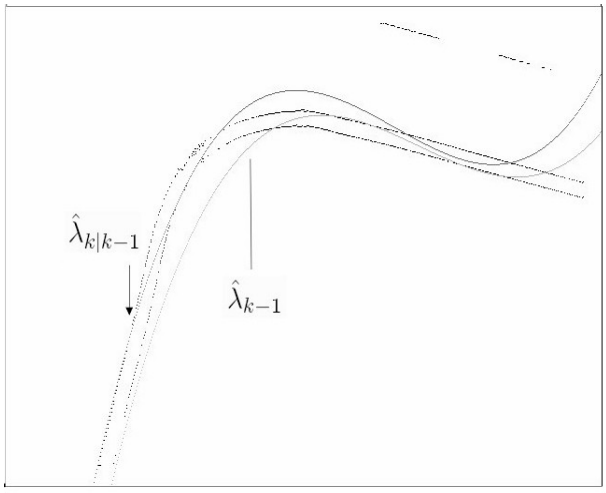
(b) Imagem 2



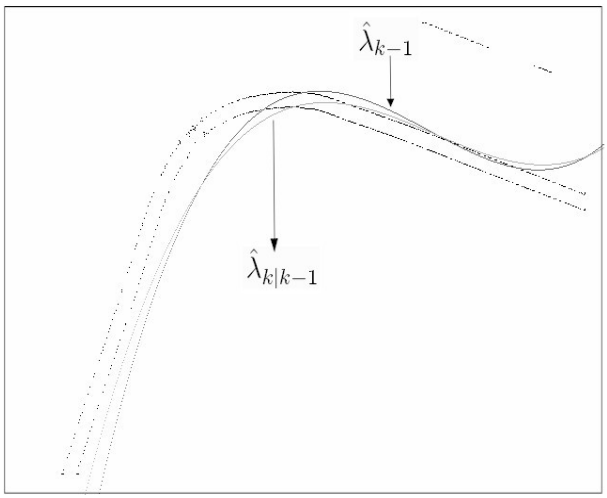
(c) Imagem 3



(d) Imagem 4

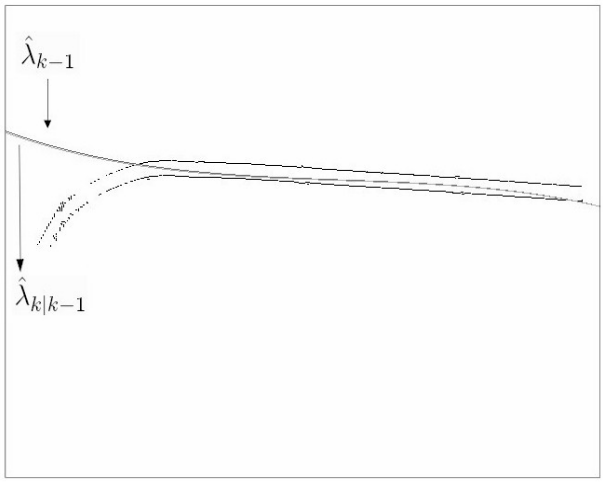


(e) Imagem 5

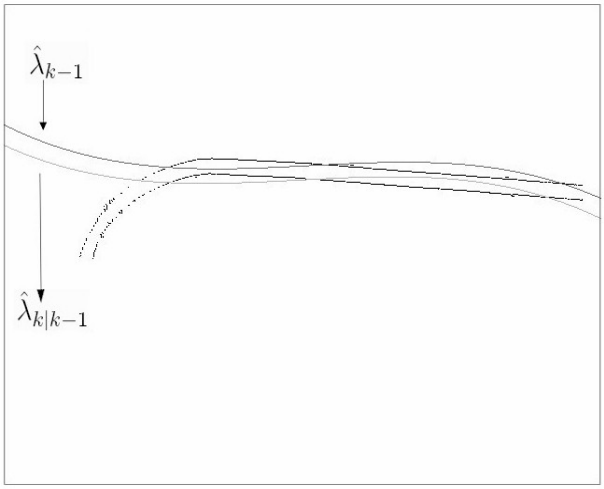


(f) Imagem 6

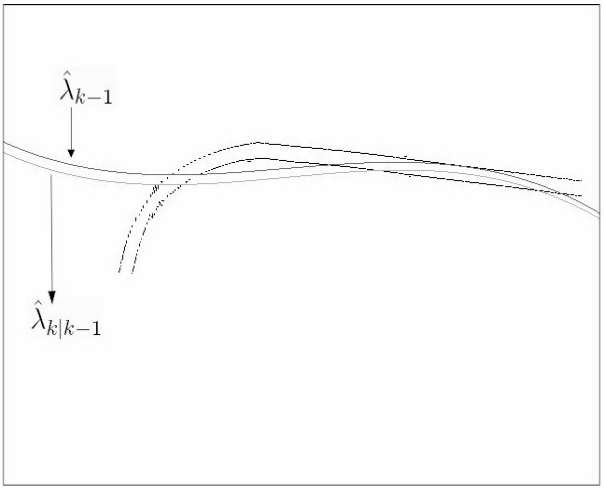
Figura A.5: Seqüência de imagens obtidas após o processamento do das etapas do sistema de rastreamento utilizando RANSAC.



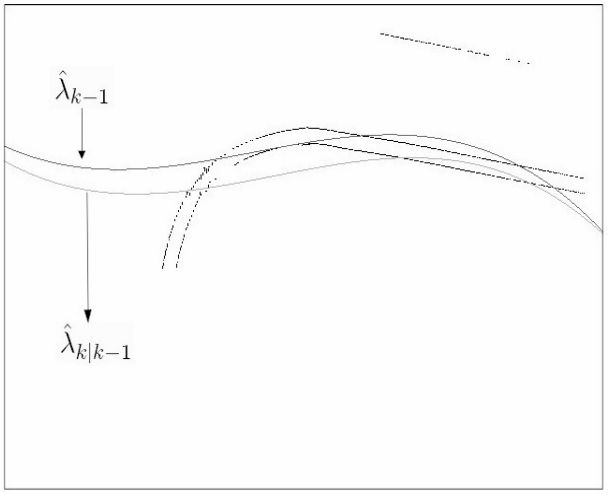
(a) Imagem 1



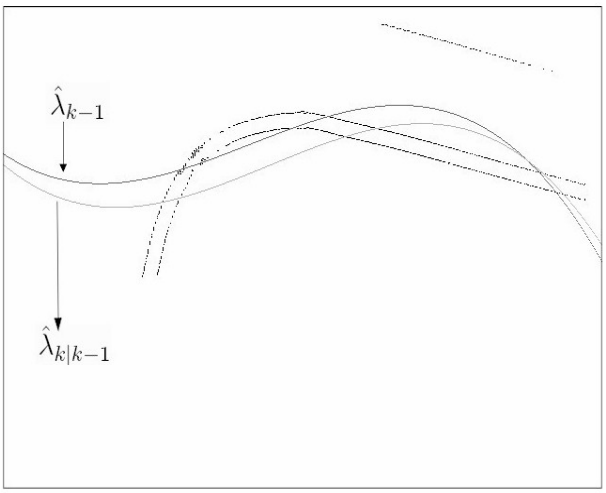
(b) Imagem 2



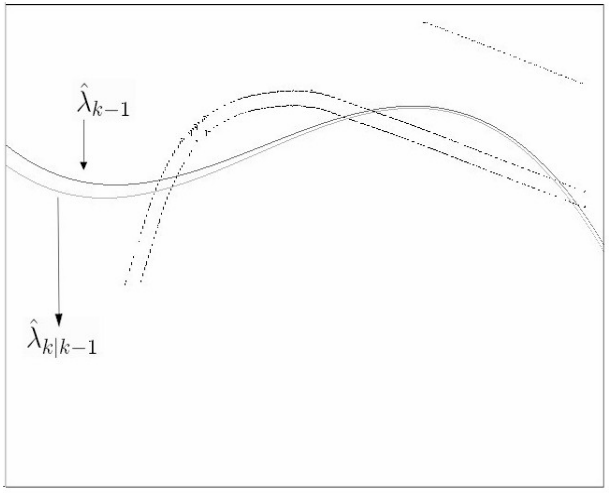
(c) Imagem 3



(d) Imagem 4

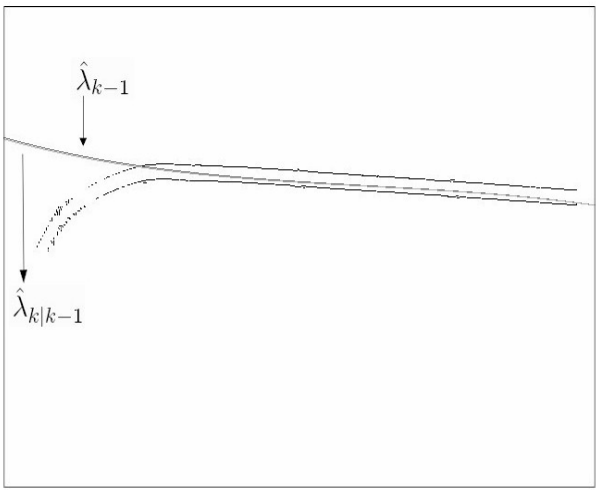


(e) Imagem 5

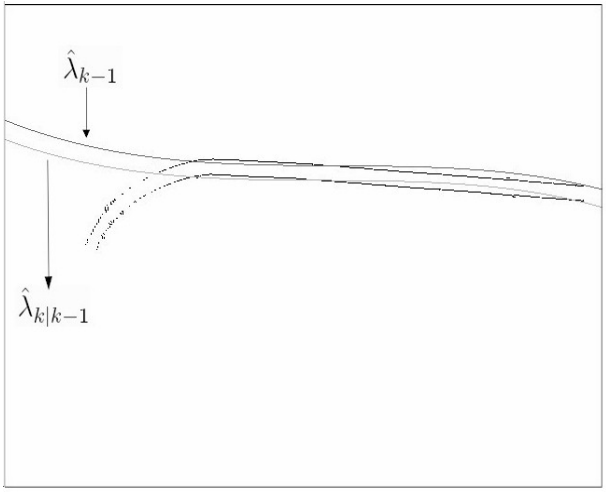


(f) Imagem 6

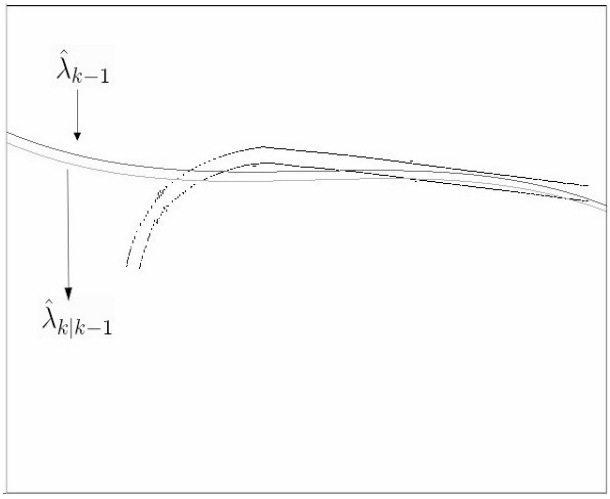
Figura A.6: Seqüência de imagens obtidas após o processamento do das etapas do sistema de rastreamento utilizando Filtro de Kalman.



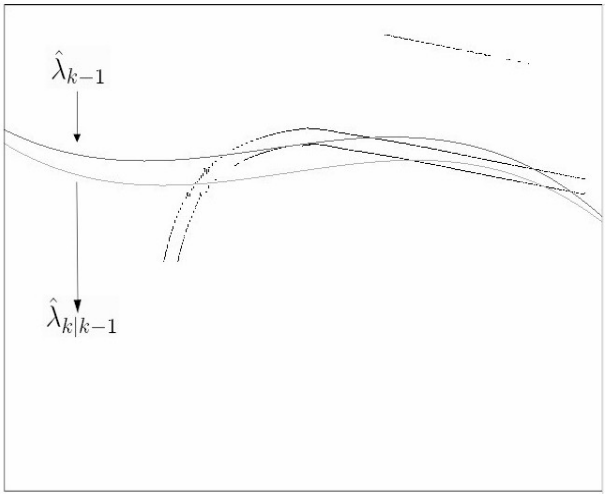
(a) Imagem 1



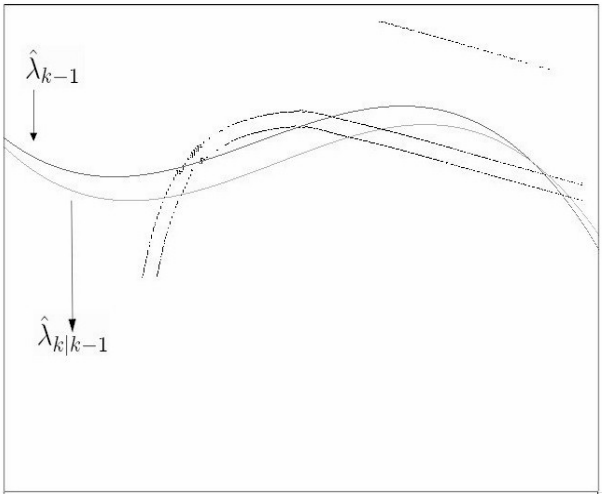
(b) Imagem 2



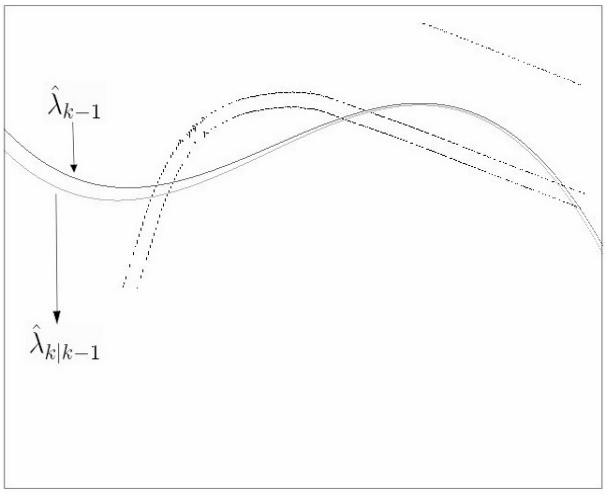
(c) Imagem 3



(d) Imagem 4



(e) Imagem 5



(f) Imagem 6

Figura A.7: Seqüência de imagens obtidas após o processamento do das etapas do sistema de rastreamento utilizando FKEIR.