Sistema de Visão para Robôs Móveis: Uma Aplicação ao Reconhecimento de Referências Geométricas

Roger Alex de Castro Freitas

Dissertação de Mestrado em Engenharia Elétrica (Automação)

Mestrado em Engenharia Elétrica (Automação)
Universidade Federal do Espírito Santo
Vitória, Fevereiro de 1999

Sistema de Visão para Robôs Móveis: Uma Aplicação ao Reconhecimento de Referências Geométricas

Roger Alex de Castro Freitas

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

Aprovada em 26/02/1999 por:

_____

Prof. Dr. Teodiano Freire Bastos Filho, UFES – Orientador

_____

Prof. Dr. Mário Sarcinelli Filho, UFES – Co-Orientador

_____

Prof. Dr. Ailson Rosetti de Almeida, UFES

_____

Prof. Dr. Márcio Rillo, USP

Universidade Federal do Espírito Santo

Vitória, Fevereiro de 1999

Freitas, Roger Alex de Castro, 1972

Sistema de Visão para Robôs Móveis: Uma Aplicação ao Reconhecimento de Referências Geométricas.
[Vitória] 1999

XIII, 85p., 29.7 cm (UFES, M. Sc., Engenharia Elétrica, 1999)

Dissertação de Mestrado, Universidade Federal do Espírito Santo

Dedicatória

Dedico este trabalho à minha
família e à Anastácia,
minha noiva.

Agradecimentos

Esta é, na minha opinião, a melhor parte de qualquer trabalho. É hora de agradecer e homenagear a todos aqueles que direta ou indiretamente me ajudaram a finalizar essa etapa. É muito provável que não me recorde de todos neste momento de euforia, mas quero que saibam que sou muito grato.

O primeiro agradecimento é para Deus, nosso criador, início e fim de tudo. Porto seguro nas horas felizes e nas atribulações.

Agradeço a minha família. Meus pais, Expedito e Geralda Augusta, pela oportunidade de estudar e pelo apoio e incentivo durante toda a minha vida. Meus irmãos, Geber Alan e Chander Rian, pela compreensão nas vezes em que eu tive que "retirá-los" da frente do computador para que pudesse trabalhar. Agradeço também o apoio e as palavras de incentivo. Ao meu padrinho José Augusto, que já não está entre nós, pelos sábios conselhos. Meus avós, tios, primos, enfim toda minha família, entes presentes e aqueles que já se foram, muito obrigado pela compreensão em relação à minha ausência nas ocasiões onde a família se reunia.

Agradeço a Anastácia, por ser compreensiva nos vários finais de semana em que não estivemos juntos, e pelo amor e apoio que sempre me deu. Agradeço ao Sr. Odimar e à D^{na}. Edinê, meus futuros sogros, pela orientação correta e pela confiança que depositaram em mim.

Aos professores Teodiano Freire Bastos Filho e Mário Sarcinelli Filho, orientador e co-orientador, respectivamente, pelo voto de confiança, por todo o apoio e boa vontade, pela compreensão e pelas oportunidades de intercâmbio no exterior, que foram de grande importância para minha vida profissional e pessoal. Muito obrigado.

Aos professores Rogério Silveira de Queiroz e Benedito Miglio Pinto, por guiarem os primeiros passos na pesquisa científica e pelo aval para que eu pudesse

ingressar no mestrado. E a todos os professores que contribuíram para a minha formação, muito obrigado.

A todos os pesquisadores do Instituto de Automática da Universidade Nacional de San Juan, Argentina, especialmente ao Dr. Benjamin Kuchen, ao Dr. Ricardo Carelli, aos Eng^{os}. Alfredo Castro e Adrián Orellana, por todo apoio que deram em minha estadia na Argentina e pela cessão do *software* DECVISION, desenvolvido naquele instituto e utilizado para processar as imagens encontradas neste trabalho.

À nossa secretária Marlene Patrício da Silva, sempre presente quando mais precisávamos e a todos os funcionários do departamento que, com muito zelo, cuidaram de sua manutenção.

A todos os colegas do mestrado. À minha "sócia" Raquel, por termos trilhado juntos os primeiros passos no estudo da visão computacional, ao meu amigo Fransérgio por ter aceitado o desafio de migrar para outra área, e pela perseverança que sempre nos deu força. Ao Alessandro Frizzera, pelo empréstimo da câmara digital que gerou as imagens contidas neste trabalho. Ao Carlos, pelo otimismo de sempre, traduzido pelo seu sorriso. Ao Jan, pelo senso prático e pelas estórias que animavam o dia-a-dia. Ao meu amigo Rodrigo José Batista, pela amizade, lealdade e os "altos papos" que temos desde os tempos de escola técnica. E a todos os outros que, se citados, ocupariam o volume de uma dissertação de mestrado, o meu muito obrigado. Seguramente, aprendi um pouco com cada um deles.

Ao CNPq, pelo suporte financeiro concedido.

À Gerência de Manutenção de Vagões da Companhia Vale do Rio Doce, pelas preciosas horas cedidas para que eu pudesse finalizar este trabalho.

Muito Obrigado a Todos!

Roger Alex de Castro Freitas

SUMÁRIO

Índice de Figuras

Resumo

A disponibilidade de dados sensoriais é muito importante para os sistemas de navegação de robôs móveis autônomos. O robô necessita de informação acerca do ambiente para que possa decidir que ação que deve ser tomada quando ele se depara com um obstáculo. Dependendo do tipo de ação que o robô deverá tomar, não bastará somente a detecção do obstáculo, mas a sua identificação, o que torna o sistema de sensoriamento à bordo do robô mais especializado. Para um robô móvel autônomo com controle baseado em agentes, um sistema de sensoriamento ultra-sônico combinado com visão artificial é aqui abordado. Esse sistema de sensoriamento disponibiliza informações sobre o tipo de obstáculo detectado bem como sua distância ao robô. Tal sistema de sensoriamento externo é capaz de evitar a geração de informação redundante ou desnecessária, assim como não demanda excessivo esforço computacional para processar as imagens a bordo do robô.

Abstract

The availability of sensorial data is an important issue for the navigation system of autonomous mobile robots. The robot needs information about its surrounding environment in order to make a decision on what to do when it faces an obstacle. In some cases, it not only does need to detect the presence of the obstacle but also to recognize it. This way, it is necessary that the onboard sensing system be much more specialized. For an agent-based controlled autonomous mobile robot, a sensing system that combines ultrasonic sensors and artificial vision is here addressed. It makes available information about the type of obstacle detected in the trajectory of the robot, as well as the distance from the detected object to the robot. This external sensing system is also able to avoid the generation of redundant and unnecessary information, besides not demanding too much computational effort to process the acquired images onboard the robot.

Capítulo 1

Robôs Móveis e sua Percepção Sensorial

1.1 - Robôs Móveis

As pesquisas relacionadas à Robótica têm proporcionado cada vez mais autonomia aos robôs móveis devido ao rápido desenvolvimento de microcontroladores com alta capacidade de processamento, sistemas de armazenamento de energia de longa duração e técnicas de controle de robôs de baixa demanda computacional. Além disto, nos últimos anos, particularmente, pesquisadores têm se empenhado em dotar tais robôs de sistemas sensoriais capazes de obter diferentes tipos de informação relativa ao ambiente.

Este trabalho propõe a utilização de um sistema de sensoriamento externo para robôs móveis composto de uma única câmara de vídeo CCD monocromática e sensores ultra-sônicos para auxiliar sua navegação em um ambiente semi-estruturado, tal como um laboratório.

Para ilustrar a utilização de diferentes tipos de sensores na coleta de informações sobre o ambiente de trabalho de robôs móveis, alguns exemplos podem ser citados.

HERMIES-IIB [1] é um robô móvel equipado com 24 transdutores ultra-sônicos e 3 câmaras de vídeo CCD. Ele utiliza o sensoriamento ultra-sônico para navegar entre obstáculos até um local previamente determinado, e então utiliza o sistema de visão artificial para localizar um painel de controle. Como última tarefa, através de manipuladores montados sobre si, o robô móvel deve acionar uma série de botões e contatos deslizantes com base em leituras de medidores analógicos presentes no painel de controle. Essas leituras são feitas através do sistema de visão artificial. Outro exemplo é o robô móvel MARS (*Mobile Autonomous Robot Stanford*) [2]. Ele possui quatro modalidades de sensoriamento: visão artificial, sensoriamento ultra-sônico, sensores de contato e odômetros. O sistema de visão é composto por duas câmaras formando um par estereoscópico binocular (os sistemas de visão estereoscópica binocular serão comentados mais adiante). O sistema de sensoriamento ultra-sônico é composto por 12 transdutores ultra-sônicos igualmente espaçados na circunferência do robô. O sensoriamento de contato é formado por 12 batentes que detectam colisões. Os odômetros têm a finalidade de proporcionar informação sobre a posição e velocidade do robô.

Quando se projeta um robô móvel deve-se levar em consideração o tipo de ambiente em que ele atuará. Normalmente os tipos de ambientes nos quais um robô móvel pode operar são classificados em estruturados, semi-estruturados e não-estruturados. O nível de estruturação de um ambiente depende principalmente do nível de conhecimento que o robô tem acerca do mesmo e da sua dinâmica [3]. Em um ambiente estruturado, o robô conhece *a priori* dados como o número, tipo e a localização dos objetos. Esse ambiente geralmente não é alterado, o que permite a construção de um mapa fixo das rotas possíveis para o robô. Existe uma certa variação na definição de ambiente semi-estruturado, mas nesta dissertação, este será considerado simplesmente um ambiente cujos limites não são previamente conhecidos, mas que é fechado e possui terreno não acidentado. São permitidas variações repentinas na configuração do ambiente, mas deve existir um número finito de objetos conhecidos (no nosso caso, paredes, portas, cantos, quinas, mesas e cadeiras). Objetos não pertencentes a tal conjunto de objetos conhecidos, no caso de aparecerem no caminho do robô, serão detectados e classificados como objetos desconhecidos [4]. Em um ambiente não-estruturado, o robô não possui nenhum conhecimento prévio a respeito dos objetos

presentes ou do tipo de terreno. Esse ambiente pode ser constantemente modificado e o robô deve ser capaz de se adequar à dinâmica do ambiente.

1.2 - Tipos de Sensoriamento

Existem dois tipos de sensoriamento que são utilizados em robôs móveis: sensoriamento interno e sensoriamento externo.

1.2.1 - Sensoriamento Interno

Um sensoriamento interno é executado quando se quer medir as condições internas do robô. Por exemplo, sensores de nível de bateria podem informar ao robô quando é hora de recarregá-las, e sensores de temperatura podem ser usados para avisar sobre algum aquecimento excessivo dos motores [3]. Também se utilizam codificadores óticos (*encoders*), acoplados às rodas do robô, para informar sua velocidade ao sistema de controle e para estimar a sua posição em relação a um sistema de referência pré-definido. No caso de robôs móveis, não é seguro basear a navegação apenas no sensoriamento interno, pois existe um erro cumulativo associado à medição de posição do robô.

1.2.2 - Sensoriamento Externo

No caso de robôs móveis, o sensoriamento externo pode, por exemplo, ser usado para corrigir o erro cumulativo de posição inerente à utilização dos *encoders*, pela localização de algum ponto de referência que permita calcular a posição real do robô e assim recalibrar o sistema. Ademais, o sensoriamento externo torna possível a obtenção de diversas informações sobre o ambiente de operação do robô, como, por exemplo, a detecção de obstáculos presentes em sua trajetória e a sua distância até eles [3].

Dois dos sistemas de sensoriamento externo mais utilizados são os baseados em sensores ultra-sônicos e os baseados em visão artificial.

1.2.2.1 - Sensoriamento Ultra-Sônico: Sensor de Ultra-Som

Um sensor pode ser definido como um dispositivo projetado para quantificar ou detectar parâmetros específicos por meio de elementos transdutores. Os transdutores são os elementos que realizam a função de transformação de uma grandeza física em outra. No caso dos transdutores ultra-sônicos ocorre a conversão de energia elétrica em energia mecânica e vice-versa [5].

Os dois tipos mais comuns de transdutores ultra-sônicos para operação no ar encontrados no mercado são os transdutores eletrostáticos e os transdutores piezoeléctricos.

- **Transdutores Eletrostáticos**

Os transdutores eletrostáticos são formados basicamente por um capacitor de placas paralelas. Uma das placas é composta por uma lâmina flexível de celofane, metalizada com alumínio em uma das faces. A outra placa é um disco metálico que contém ranhuras circulares formando cavidades em forma de anéis concêntricos (Figura 1.1). Portanto, as placas do capacitor formado são separadas pela lâmina e pelo ar que se encontra nas ranhuras [5].

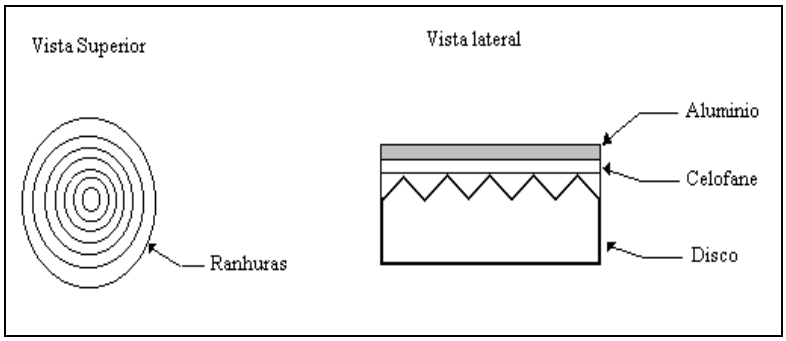


Figura 1.1 – Esquema de um transdutor eletrostático [5].

Os transdutores eletrostáticos podem operar em dois modos: emissão e recepção.

Para carregar as placas do capacitor, aplica-se a elas uma tensão contínua de polarização *E*, criando uma força de atração. Quando operando em modo emissor (Figura 1.2) adiciona-se uma tensão alternada *V* à tensão de polarização, modulando-se assim o campo elétrico existente entre as placas e, consequentemente, a força eletrostática de atração. Isto ocasiona a vibração de uma das faces do transdutor e este movimento se transmite ao ar produzindo uma onda acústica [5].

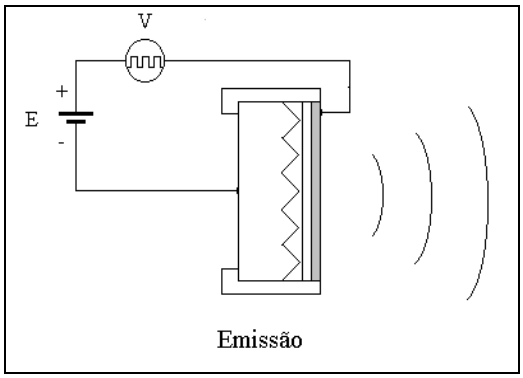


Figura 1.2 – Transdutor eletrostático operando como emissor [5].

Quando operando em modo receptor (Figura 1.3), a tensão de polarização se mantém, através da resistência *R*, e acontece o processo inverso: a vibração do ar produz uma variação da distância entre as placas do capacitor, provocando assim uma variação da capacitância e gerando uma tensão alternada *u* no transdutor.

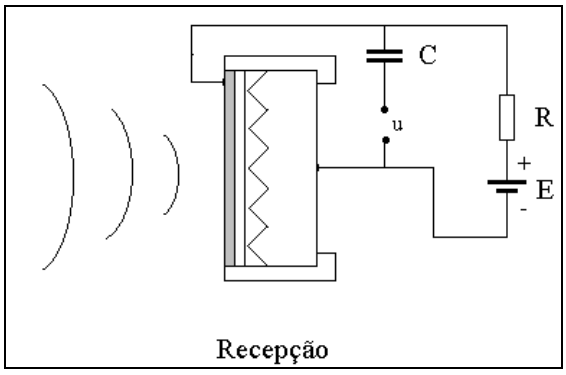


Figura 1.3 – Transdutor eletrostático operando como receptor [5].

- **Transdutores Piezoeléctricos**

Os transdutores piezoeléctricos baseiam seu funcionamento na propriedade de certos cristais naturais (quartzo) e de algumas cerâmicas sintéticas de se polarizarem eletricamente quando submetidas a esforços mecânicos. A prática revela que, para uma larga faixa de valores, a quantidade de cargas produzidas é proporcional aos esforços aplicados. Porém, o efeito inverso também é observado, ou seja, o material sofre uma deformação quando lhe é aplicado um campo elétrico. Devido à sua grande sensibilidade e facilidade de industrialização, as cerâmicas piezoeléctricas são os materiais mais utilizados na conformação de transdutores piezoeléctricos [5].

A Figura 1.4 mostra um esquema da montagem mais típica de transdutores piezoeléctricos. Pode-se observar que um disco cerâmico é acoplado a um diafragma metálico, e estes a um cone radial, para aumentar a sensibilidade [5].

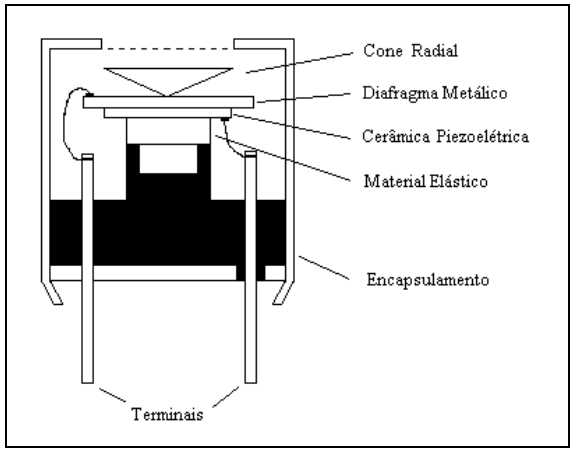


Figura 1.4 – Esquema de um transdutor piezoeléctrico [5].

Os modos de operação do transdutor piezoeléctrico podem ser vistos nas Figuras 1.5 e 1.6. No modo emissor (Figura 1.5), aplica-se uma tensão alternada *V* às duas faces opostas da cerâmica, fazendo com que ocorra uma vibração de mesma freqüência da tensão aplicada. Se a freqüência da tensão alternada coincide com a freqüência de vibração da cerâmica, ela entra em ressonância e as vibrações alcançam um máximo. Estas vibrações se transmitem ao meio produzindo uma onda acústica. Este fenômeno é recíproco, de tal forma que na recepção (Figura 1.6) a pressão da onda acústica faz aparecer cargas elétricas na cerâmica, produzindo desta forma um sinal elétrico *u* [5].

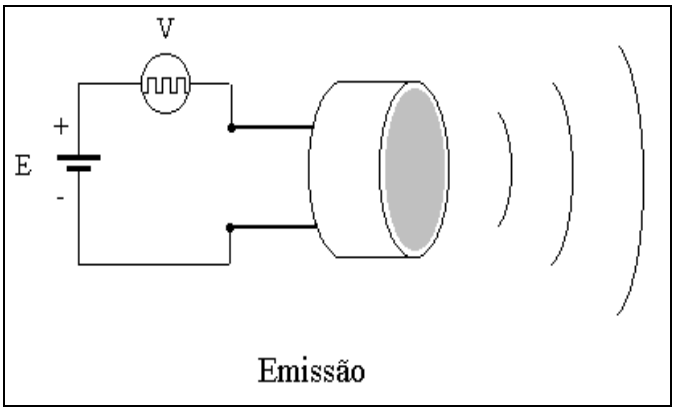


Figura 1.5 – Transdutor piezoeléctrico no modo emissor [5].

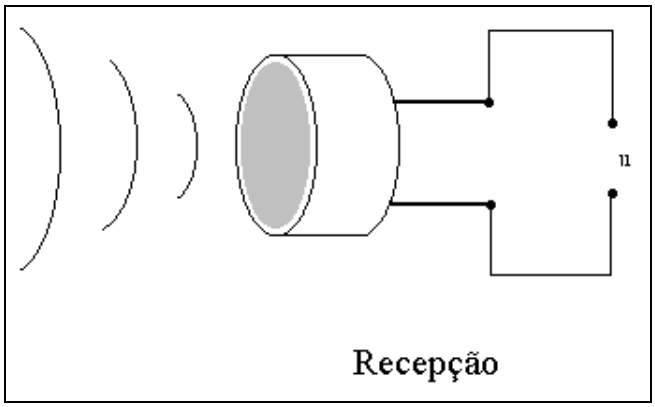


Figura 1.6 – Transdutor piezoeléctrico no modo receptor [5].

- **Características dos sensores ultra-sônicos**

Duas características principais que devem ser consideradas na escolha do sensor ultra-sônico para uma determinada aplicação são a *diretividade* e a *zona morta* [5].

Os transdutores ultra-sônicos emitem energia sonora concentrada em lóbulos, ou cones, de radiação. A *diretividade* está associada com o ângulo de abertura deste cone de radiação, que está relacionado, por sua vez, com a região frontal na qual um objeto pode ser detectado: uma baixa diretividade permite que um objeto localizado lateralmente possa gerar um eco detectável; um transdutor altamente diretivo será capaz de detectar unicamente superfícies localizadas perpendicularmente ao seu lóbulo de radiação [5].

Por outro lado, quando se excita um transdutor, sua vibração não se limita, no tempo, à duração do sinal de excitação. Ao cessar o sinal de excitação, a vibração persiste, com um amortecimento progressivo, o que impossibilita sua utilização como receptor até que o sinal emitido não tenha se atenuado suficientemente. Este fenômeno faz com que se produza uma região à frente do transdutor onde não se pode efetuar qualquer medida, região essa comumente denominada *zona morta* ou *zona cega* [5].

Os sistemas de sensoriamento baseados unicamente em sensores ultra-sônicos têm baixo custo de implementação, além de serem leves e produzirem dados sensoriais de fácil processamento. Porém, tais sistemas podem, em várias ocasiões e devido a diversos fatores, produzir informações falsas, ou seja, detectar a presença de obstáculos que não existem, não detectar obstáculos reais ou simplesmente executar medições de distância completamente equivocadas [4].

1.2.2.2 - Sensoriamento Utilizando Visão Artificial

A visão é, sem dúvida, o sentido mais poderoso e complexo do ser humano. É através dela que ele recebe uma grande quantidade de informações a respeito do mundo que o cerca, o que facilita interagir inteligentemente com o ambiente dinâmico. Utilizando informações visuais, o homem é capaz de estimar a posição de objetos (relativa a ele mesmo ou relativa a outros objetos), identificar formas, perceber movimentos e estimar dimensões. Tudo isso sem contato direto com os objetos.

Com o intuito de proporcionar cada vez mais autonomia aos robôs (notadamente no caso de robôs móveis), pesquisadores têm se empenhado em dotá-los de um sistema sensorial que de certo modo possa desempenhar as mesmas funções do sistema de percepção visual biológico humano [6]. Tais sistemas, denominados sistemas de visão artificial, ainda estão longe de processarem as informações tão rápida e precisamente quanto o sistema de visão biológico humano, mas grandes avanços têm surgido rapidamente.

A visão artificial trata da obtenção automática de informação a respeito do ambiente disponibilizando-a ao sistema de controle, que tomará uma decisão sobre qual o comportamento a adotar. Ela utiliza dispositivos de aquisição de imagens (câmaras) e plataformas onde essas imagens possam ser processadas, para obtenção de informações relevantes sobre o ambiente. Os problemas inerentes ao projeto de um sistema de visão artificial dependerão em grande parte da aplicação prevista para este sistema, e basicamente, deve ser considerado que tipo de informação relevante deve ser extraída das imagens, qual a metodologia a ser utilizada para extrair tal informação e quais representações devem ser empregadas para permitir ao robô realizar as tarefas exigidas [7].

Uma das características mais marcantes associadas à captura de informações do ambiente através de câmaras é a perda da componente tridimensional, ou seja, as imagens obtidas são bidimensionais, enquanto o mundo real é tridimensional. Existem várias metodologias que tratam da recuperação da informação tridimensional, e a técnica mais comumente utilizada é a triangulação [8]. Esta técnica pode ser aplicada quando estão disponíveis duas imagens de um mesmo objeto (pela utilização de um par de câmaras ou pela utilização de uma câmara que se desloca entre dois pontos conhecidos), quando estão disponíveis uma câmara e uma fonte de iluminação (estimativa das propriedades de reflexão difusa de superfícies Lambertianas), ou quando estão disponíveis uma única câmara e uma fonte de iluminação especialmente controlada, chamada luz estruturada. Adicionalmente, a informação 3D também pode ser obtida utilizando uma câmara e um sensor auxiliar que proporcione a informação de distância do objeto à câmara (profundidade) [5].

Grande parte dos sistemas de visão artificial desenvolvidos atualmente utilizam duas câmaras como dispositivos de aquisição de imagens. Neste caso específico o sistema é classificado como sistema de *Visão Estereoscópica Binocular*. Neste sistema são utilizadas duas imagens da cena, obtidas de diferentes posições, para determinar a estrutura tridimensional dos objetos que aparecem nas duas imagens simultaneamente. Segundo Vicente [7], pode-se caracterizar o processo da visão estereoscópica pelas seguintes etapas: localização de um elemento característico em uma imagem, busca do elemento homólogo, que corresponde à projeção do mesmo elemento do espaço na outra

imagem, e reconstrução tridimensional da cena a partir das coordenadas dos pontos homólogos, utilizando o princípio da triangulação.

O método da triangulação é fundamentado em princípios relativamente simples e a maior dificuldade a ser suplantada no projeto do sistema de visão estereoscópico é saber como identificar os elementos homólogos nas duas imagens, ou seja, dado um elemento em uma imagem, deve-se localizar qual elemento corresponde à sua projeção na outra imagem. Este problema é clássico e é conhecido como *o problema da correspondência estereoscópica* [9]**.**

A técnica de obtenção de informação 3D utilizando uma câmara e um sensor auxiliar que proporcione a informação de profundiddade obviamente diminui o tempo de processamento, já que se deve processar apenas uma imagem, o que é de suma importância no caso de um robô móvel, o qual deve executar o processamento a bordo. Esta será a técnica utilizada neste trabalho. Dois tipos de sensores são os mais utilizados para proporcionar a informação de distância: sensores ultra-sônicos e laser. Neste trabalho, serão usados sensores ultra-sônicos.

Para que o robô possa desempenhar as suas tarefas de maneira autônoma, é preciso que seu sistema de controle possa gerenciar e interpretar as informações que chegam dos diversos sensores. A organização do sistema de controle é determinada pelo tipo de arquitetura de controle adotada.

1.3 - Controle Baseado em Comportamentos

Arquiteturas tradicionais utilizadas para o controle de robôs móveis o fazem de forma centralizada, utilizando-se de modelos do ambiente. Os dados provenientes dos diversos tipos de sensores são tratados através da *fusão sensorial*, o que acarreta o armazenamento de grandes quantidades de dados necessários para a construção do modelo do ambiente de operação do robô, tornando o processo computacionalmente intensivo [3].

Um outro tipo de arquitetura de controle de robôs, denominada “baseada em comportamentos”, realiza a decomposição da tarefa de controle do robô móvel em comportamentos propriamente ditos. Comportamentos são camadas de um sistema de controle que trabalham em paralelo sempre que são disparadas pelos sensores apropriados [10]. A Figura 1.7 ilustra exemplos de alguns comportamentos que são disparados por determinados sensores. Observe-se que o sistema é decomposto em elementos relativamente desacoplados que possuem seus próprios objetivos, especialidades e interfaces, e são responsáveis por comportamentos específicos do robô. Esses elementos são executados paralelamente entre si e apresentam recursos próprios de processamento, do sensoriamento à ação dos atuadores, permitindo uma resposta rápida. A relativa independência entre esses elementos resulta em modularidade e flexibilidade para o sistema [4]. Finalmente, note-se que a fusão de dados sensoriais inerente às arquiteturas tradicionais dá lugar à *fusão de comportamentos*, onde um sistema de arbitração com prioridades é usado para determinar qual o comportamento dominante em cada situação [4]. Este tipo de arquitetura é a que será utilizada neste trabalho, e suas características específicas e implementação computacional serão discutidas de forma mais detalhada no Capítulo 3.

Figura 1.7 – Arquitetura de controle baseada em comportamentos [4].

1.4 - O Robô Móvel Brutus

No **LAI/UFES** (Laboratório de Automação Inteligente da Universidade Federal do Espírito Santo) encontra-se em desenvolvimento o robô móvel a rodas chamado "Brutus**"** (Figura 1.8). Tal robô é do tipo tração diferencial, e utiliza motores CC para acionamento das rodas tracionadas. O controle dos vários processos (sensoriamento interno e geração de trajetória) é feito por um microcontrolador MC68332. O sensoriamento interno é realizado através de *encoders* e o sensoriamento externo é feito unicamente através de sensores ultra-sônicos, os quais são controlados através de um microcontrolador 68HC11. Estes sensores ultra-sônicos são capazes de proporcionar informação sobre a distância aos obstáculos presentes na trajetória do robô, assim como reconhecer um número limitado de obstáculos presentes no ambiente [4]. Entretanto, este sistema está sujeito a falhas pelas limitações dos sensores ultra-sônicos, além de não prover informações mais detalhadas sobre o ambiente de operação do robô.

Como forma de melhorar o sensoriamento já implementado no robô móvel Brutus, este trabalho propõe a utilização de um sistema de sensoriamento externo composto de uma única câmara de vídeo CCD monocromática e sensores ultra-sônicos para auxiliar sua navegação. Este sistema será descrito em detalhes nos próximos capítulos, tanto no que tange à sua operação quanto no que tange à sua inserção no sistema de controle do robô.

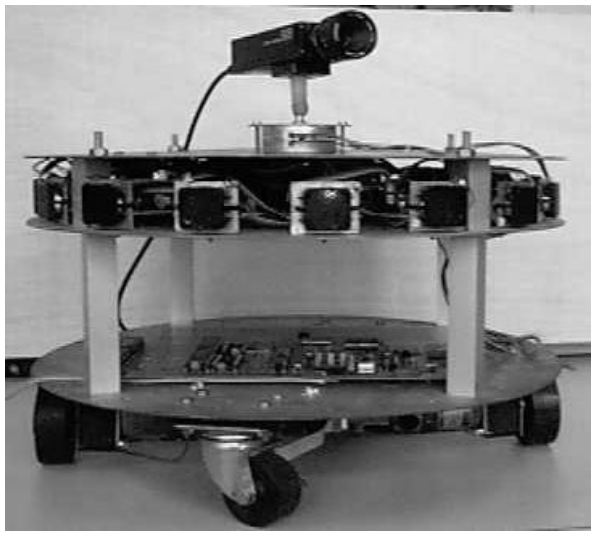


Figura 1.8 - Robô Móvel a Rodas "Brutus".

Capítulo 2

Visão Artificial: Conceitos Básicos

2.1 - Imagens e suas propriedades

2.1.1 - Representação

O termo *imagem monocromática*, ou simplesmente *imagem*, refere-se a uma função bidimensional da intensidade de luz *f(x,y)*, onde *x* e *y* denotam coordenadas espaciais, sendo o valor da função *f*, em qualquer ponto *(x,y)*, proporcional à intensidade luminosa (ou nível de cinza) da imagem naquele ponto [11].

A luz é uma forma de energia, e portanto *f(x, y)* deve assumir valores finitos e não nulos. As imagens que o homem percebe são formadas normalmente a partir da luz refletida pelos objetos à sua volta. A função *f(x, y)* é influenciada pela quantidade de luz que incide sobre os objetos (*iluminância*) e pela quantidade de luz que é refletida pelos objetos (*reflectância*) (Figura 2.1). Deste modo, pode-se escrever:

$$f(x, y) = i(x, y)r(x, y)$$

onde *i(x, y)* é a iluminância, sendo finita e não nula, e *r(x, y)* é a reflectância, que assume valores do intervalo aberto (0, 1). O valor 0 significa total absorção e o valor 1 significa total reflexão.

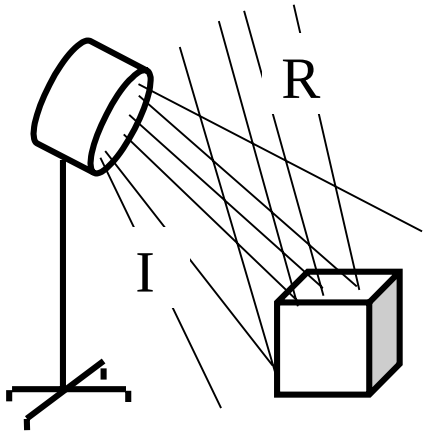


Figura 2.1 - Componentes de *f(x, y)*: (I) Iluminância, (R) Reflectância [12].

Outra forma gráfica de representar uma imagem seria associar os valores de intensidade da função *f* com um terceiro eixo, perpendicular ao plano da imagem, como mostra a Figura 2.2.

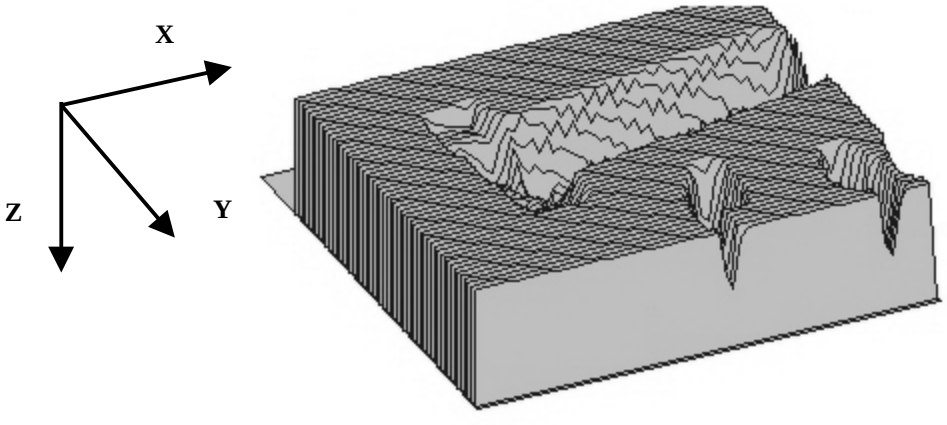


Figura 2.2 – Representação gráfica de uma imagem em três eixos.

O termo *imagem digital* refere-se a uma imagem gerada a partir da discretização da função *f(x,y)* tanto espacialmente, ou seja, discretização das coordenadas *x* e *y*, quanto ao próprio valor que a função assume, ou seja, discretização do nível de intensidade.

No presente trabalho somente serão tratadas imagens monocromáticas, ou seja, imagens em preto e branco. Neste tipo de imagem, os níveis de intensidade de cada ponto são chamados *níveis de cinza*, pois os valores intermediários entre o preto e o branco são tonalidades de cinza. A faixa total de valores de intensidade possíveis na imagem é chamada *escala de cinza*.

Ao trabalhar com imagens digitais, pode-se também representá-las através de matrizes. Cada elemento da matriz, chamado de *pixel*, pode ser localizado pela interseção de uma determinada linha com uma determinada coluna. O valor que este *pixel* armazena é o seu nível (ou tom) de cinza (Figura 2.3). O termo *pixel* é uma abreviação da expressão inglesa *picture elements* [11].

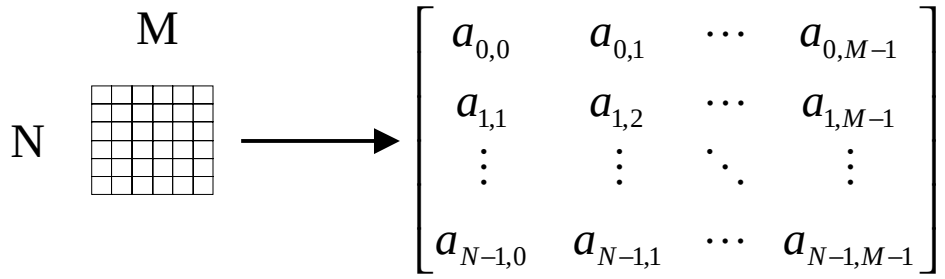


Figura 2.3 - Representação matricial de uma imagem [12].

É prática comum em processamento de imagens associar um sistema de coordenadas à imagem, tendo como origem o *pixel* superior esquerdo. A Figura 2.4 mostra a convenção de orientação dos eixos que será utilizada ao longo de toda esta dissertação. A representação dessa imagem em três eixos é mostrada na Figura 2.2.

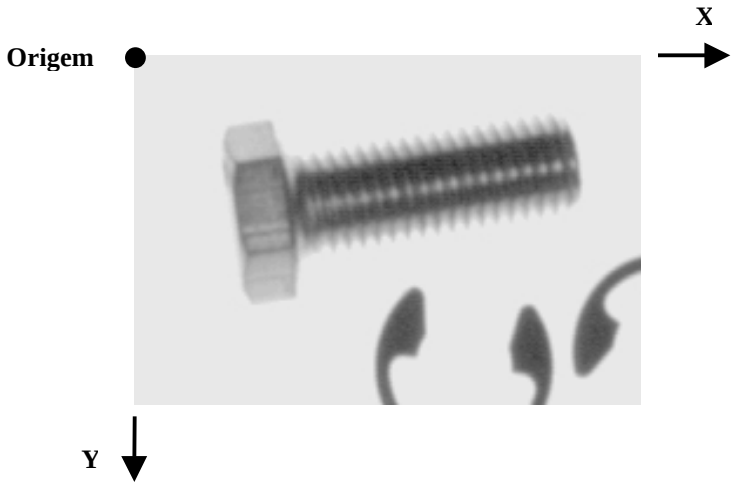


Figura 2.4 - Convenção para orientação dos eixos. Representação em dois eixos.

2.1.2 - Amostragem e Quantização

O termo *amostragem* significa a discretização da função *f(x,y)* espacialmente, ou seja, é a definição das coordenadas *x* e *y* onde a função vai assumir algum valor. Isto é feito para que os sinais bidimensionais (imagens) que chegam ao sensor responsável pela aquisição possam ser tratados digitalmente em um computador. O resultado desta operação é uma matriz de pontos (*pixels*), onde o valor de *f(x,y)* está definido.

O termo *quantização* significa a discretização dos valores que a função *f(x,y)* poderá assumir. Quando se quantiza o sinal bidimensional (imagem), na verdade, define-se um conjunto finito de valores que ele poderá assumir.

Exemplificando o que acaba de ser exposto, quando se tem uma imagem digital com 512 x 512 *pixels* e 256 tons de cinza, significa que a matriz utilizada na amostragem do sinal tem dimensões M=512 e N=512, ou seja, são M x N = 262.144 pontos onde a função está definida, e existem 256 valores possíveis para a função em cada ponto. Na prática, quando se utiliza 256 níveis de cinza, no intervalo fechado [0,255], faz-se corresponder o valor 0 ao preto e o valor 255 ao branco, ou vice-versa.

É necessário frisar que se trata de amostragem e quantização *uniformes*, o que significa que os intervalos entre os valores amostrados são constantes. Porém, existem também amostragem e quantização *não-uniformes*. O que se faz na *amostragem não-uniforme* é diminuir o espaçamento entre as amostras em regiões da imagem que mereceriam maior atenção por conterem mais informações, como é o caso dos contornos dos objetos. Já nas regiões caracterizadas por níveis mais ou menos constantes de cinza, a amostragem poderia ser feita em intervalos espaciais maiores, sem um grande risco de perda de informação. A *quantização não-uniforme* é feita analisando-se os níveis de cinza presentes na imagem. Se grande parte da informação contida na imagem está numa determinada faixa de valores de cinza, então essa faixa é refinada, apresentando intervalos de quantização menores. Isso permite que se possa representar com mais detalhes algumas regiões de interesse da imagem.

2.1.3 - Relações entre *Pixels*

2.1.3.1 - Vizinhança de um *Pixel*

Usualmente se definem alguns tipos de vizinhança associada aos *pixels*. Algumas delas são:

a) *Vizinhança 4*: um *pixel p* localizado nas coordenadas (x, y) possui como vizinhança 4 os *pixels* localizados em (x+1,y), (x-1,y), (x,y+1) e (x,y-1). Esses *pixels* são denominados vizinhos 4 e são representados por $N_4(p)$ (Figura 2.5).

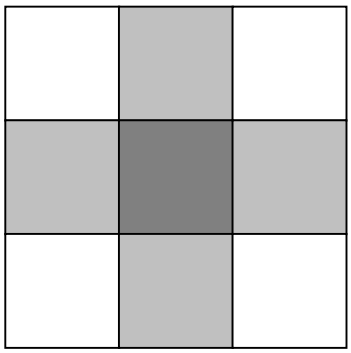


Figura 2.5 - Vizinhança 4 (cinza claro) do *pixel* central (cinza escuro).

b) *Vizinhança D*: um *pixel* localizado nas coordenadas (x,y) possui como vizinhança D (diagonal) os *pixels* localizados em (x+1,y+1), (x+1,y-1), (x-1,y+1) e (x-1,y-1). Esses *pixels* são denominados vizinhos D e são representados por $N_D(p)$ (Figura 2.6).

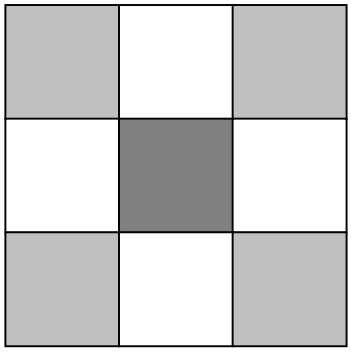


Figura 2.6 - Vizinhança D (cinza claro) do *pixel* central (cinza escuro).

b) *Vizinhança 8*: um *pixel* localizado nas coordenadas (x,y) possui como vizinhança 8 o conjunto de seus vizinhos 4 mais o conjunto de seus vizinhos D. Esses *pixels* são denominados vizinhos 8 e são representados por $N_8(p)$ (Figura 2.7).

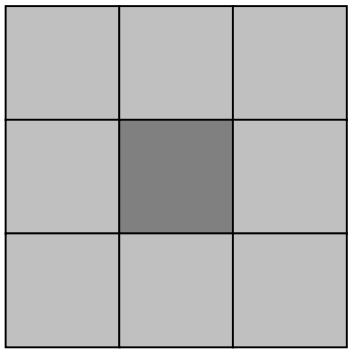


Figura 2.7 - Vizinhança 8 (cinza claro) do *pixel* central (cinza escuro).

2.1.3.2 - Conectividade

O conceito de conectividade é importante para que se possa definir em que situações dois *pixels* estão conectados. Isto é de grande valia no momento de estabelecer os limites das regiões em uma imagem. Para que dois *pixels* sejam considerados conectados, é necessário que eles sejam adjacentes, ou seja, que exista uma relação de vizinhança entre eles e que seus níveis de cinza obedeçam a algum critério de similaridade. Por exemplo, seja uma imagem binária (em que $f(x,y)$ só pode assumir os valores 0 ou 1) e dois *pixels* a ela pertencentes que sejam vizinhos. Então, eles só estarão conectados se ambos tiverem valor 0 ou ambos tiverem valor 1. A seguir se formaliza um pouco mais a definição de conectividade.

Assuma-se que V representa o conjunto dos níveis de cinza usados para definir conectividade. Por exemplo, em uma imagem binária, $V = \{1\}$ para a conectividade de *pixels* com valor 1. Para uma imagem com vários tons de cinza, se $V = \{32, 33, \ldots, 63, 64\}$, significa considerar a conectividade de *pixels* que tiverem níveis de cinza dentro do conjunto especificado. Isto posto, definem-se três tipos de conectividade:

a) *conectividade 4*: dois *pixels* p e q com valores contidos no conjunto V apresentam *conectividade 4* se q está contido no conjunto $N_4(p)$;

b) *conectividade 8*: dois *pixels* p e q com valores contidos no conjunto V apresentam *conectividade 8* se q está contido no conjunto $N_8(p)$;

c) *conectividade m* (conectividade mista): dois *pixels* p e q com valores contidos no conjunto V apresentam *conectividade m* se:

i) q está contido no conjunto $N_4(p)$, ou

ii) q está contido no conjunto $N_D(p)$ e o conjunto $N_4(p) \cap N_4(q)$ é vazio. (Esses são os *pixels* que são *vizinhos 4* simultaneamente de p e q e cujos valores estejam em V).

2.1.3.3 - Distância

A distância entre dois *pixels* pode ser medida de diferentes maneiras. Os tipos de distâncias mais comumente usados são a *distância Euclidiana*, a *distância D_4* e a *distância D_8* [13].

a) *distância Euclidiana* (D_E): a geometria clássica define a distância Euclidiana entre dois pontos, cujas coordenadas são (*i, j*) e (*h, k*), através da relação

$$D_E((i,\ j),(h,k)) = \sqrt{(i-h)^2 + (j-k)^2}$$

b) *distância D_4*: a distância D_4 é definida como o número mínimo de passos verticais ou horizontais, executados sobre um *grid* digital, necessários para se mover desde um *pixel* inicial (*i, j*) até um *pixel* final (*h, k*).

$$D_4((i,\ j),(h,k)) = |i-h| + |j-k|$$

c) *distância D_8*: se além de movimentos verticais e horizontais for possível executar movimentos diagonais, então a distância D_8 é definida como o valor máximo entre os módulos das diferenças entre as coordenadas dos *pixels*, ou seja:

$$D_8((i,\ j),(h,k)) = max\ \{|i\text{-}h|, |j\text{-}k|\}$$

Antes de estudar um sistema de visão artificial, é interessante se fazer uma breve explanação sobre os principais elementos que compõem o mais perfeito e complexo sistema de detecção, formação e análise de imagens conhecido até hoje, o humano. Muitos sistemas implementados na atualidade tentam reproduzir artificialmente as soluções que a evolução humana encontrou para o problema de extrair informações de imagens.

2.2 - Sistema de Percepção Visual Humano

Os olhos humanos são aproximadamente esféricos, com um diâmetro médio de aproximadamente 20 mm [11]. Esse par de sensores permite enxergar tanto os componentes monocromáticos quanto os componentes multiespectrais da luz (cores). O espectro visível é a porção do espectro eletromagnético que os humanos conseguem perceber. Ambos são mostrados na Figura 2.8. O espectro visível compreende valores de comprimento de onda desde aproximadamente 400 nm até 700 nm [12].

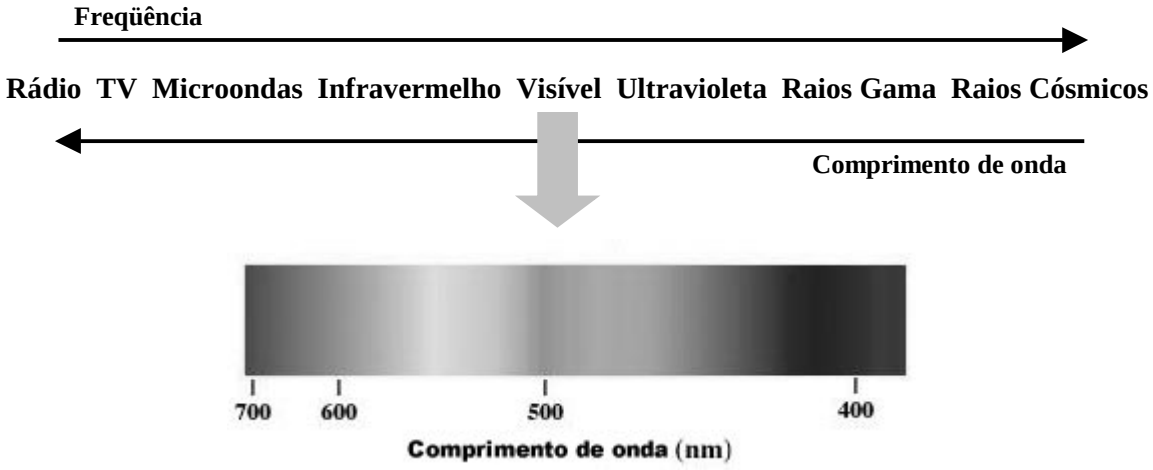


Figura 2.8 - Espectro eletromagnético e espectro visível.

As imagens que se formam no interior do olho humano são transformadas em impulsos elétricos, os quais são enviados ao cérebro para a sua interpretação. Os principais elementos que compõem o olho humano são córnea, pupila, lentes, retina e nervo óptico (Figura 2.9).

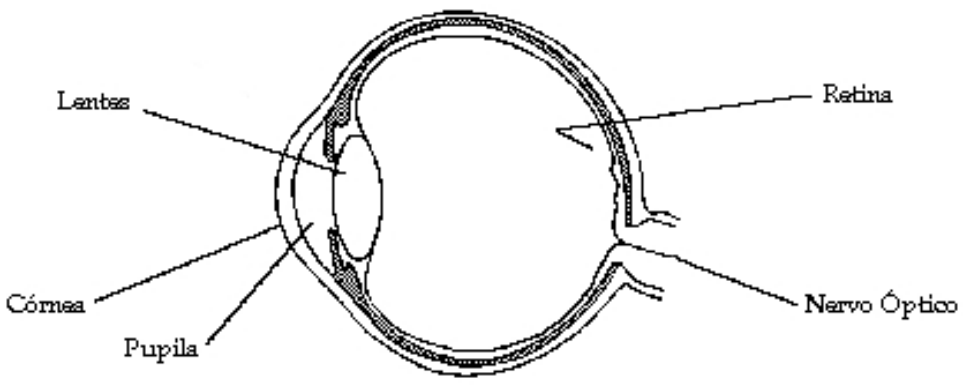


Figura 2.9 – Componentes principais do olho humano.

A córnea é a superfície externa do olho e é formada por um tecido claro que tem como função os primeiros ajustes focais dos raios de luz que chegam ao olho. A pupila está localizada na base da córnea e é um músculo circular tipo esfíncter, que através da sua abertura e fechamento regula a quantidade de luz que penetra no olho. Protegidas pela pupila estão as lentes, que são constituídas de uma massa semicristalina de proteínas conectadas a fibras musculares nas laterais. Os músculos têm a função de comprimir e tracionar as lentes, variando sua espessura e permitindo, portanto, focalizar objetos próximos e distantes, automaticamente. Mais interiormente, existe uma membrana chamada retina que possui quase 200 milhões de terminações nervosas sensíveis à luz. Esses receptores de luz estão divididos em dois tipos: *cones* e *bastonetes*. Os cones estão mais concentrados na porção central da retina e são mais sensíveis à cor, proporcionando uma visão de alta resolução. Os bastonetes estão distribuídos pela retina e são sensíveis a baixos índices de iluminação. Eles produzem uma visão global da cena com baixa resolução e estão envolvidos com a visão monocromática. Ambos os tipos de células nervosas apresentam uma distribuição radial decrescente a partir da área central da retina (Figura 2.10). É por essa razão que o homem consegue distinguir mais detalhes dos objetos que se encontram na região central do seu campo de visão. Esta é uma forma de reduzir a quantidade de informações que serão enviadas ao cérebro. Esse modelo vem inspirando pesquisadores a desenvolverem sistemas de visão que possuam resolução decrescente ao se afastar da região central, proporcionando assim uma economia no trabalho computacional [14, 15].

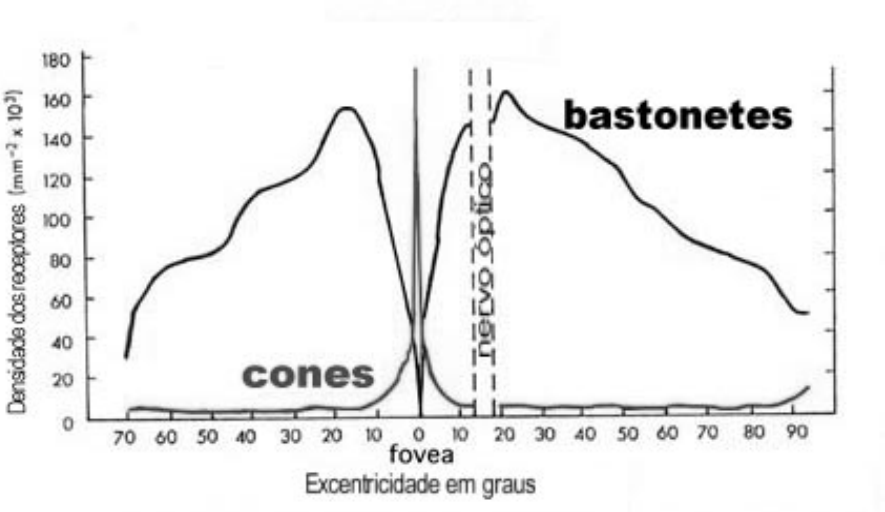


Figura 2.10 - Distribuição de clones e bastonetes na retina [11].

2.3 - Sistemas de Visão Artificial

2.3.1 - Sistema de Visão Artificial Genérico

Esta seção tenta resumir as etapas de um sistema de processamento de imagens desde a formulação do problema até a obtenção de um resultado. A Figura 2.11 apresenta o diagrama das diversas etapas de um sistema de processamento de imagens, como sugerido por Gonzalez e Woods [11].

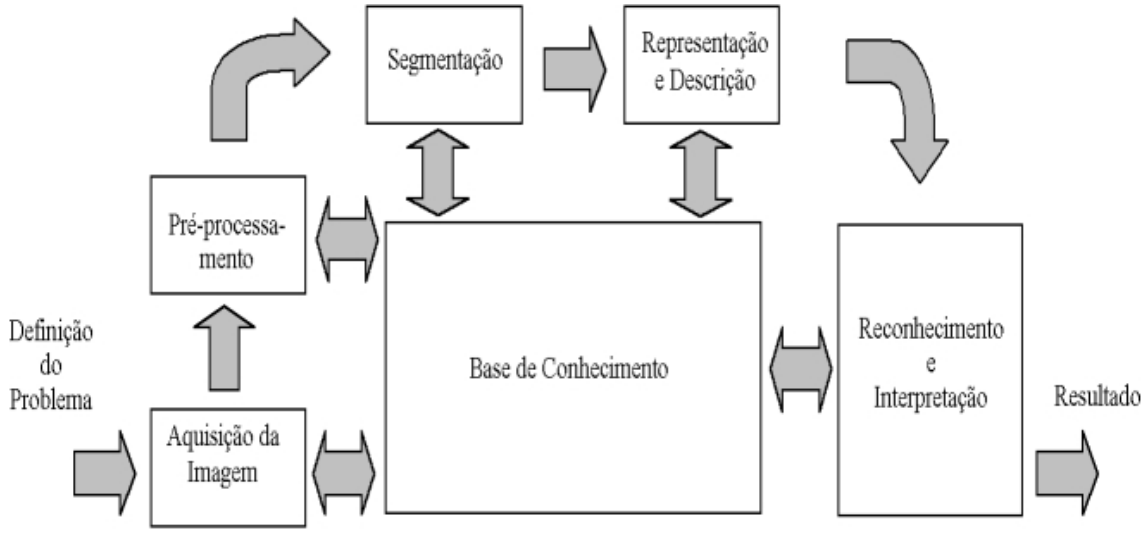


Figura 2.11 - Etapas principais de um sistema de visão artificial [11].

- **Definição do problema**

A primeira etapa do processo é a *definição do problema*, ou seja, qual é o objetivo que o sistema deve cumprir. Pode ser a identificação de peças numa esteira rolante, o reconhecimento de caracteres numa folha de papel, a identificação de impressões digitais ou o reconhecimento de obstáculos na trajetória de um robô móvel. Se o sistema tiver sido bem planejado, chega-se à última etapa do processo, que é o *resultado*, ou seja, qual é a peça, qual é o caractere, a quem pertence a impressão digital ou qual é o obstáculo à frente do robô.

- **Aquisição da imagem**

O segundo passo trata da *aquisição da imagem*. Necessita-se, então, de um dispositivo que seja sensível à energia na faixa do espectro eletromagnético em que estamos trabalhando, e que produza em sua saída um sinal elétrico proporcional à quantidade de energia captada. Esse sinal elétrico, então, precisa passar por um dispositivo conversor analógico-digital para que possa posteriormente ser processado em um computador. Ao ser completada esta etapa, já se tem a imagem digitalizada da cena que se quer analisar.

- **Pré-processamento da imagem**

O terceiro passo trata do *pré-processamento* da imagem obtida. Nesta etapa são realizadas operações, sobre a imagem, que geralmente visam realçar as informações importantes para os próximos passos. Essas operações podem ser realizadas tanto no domínio espacial quanto no domínio da freqüência. O domínio espacial refere-se ao próprio plano da imagem e é caracterizado pela manipulação direta dos *pixels*. As técnicas de processamento no domínio da freqüência são baseadas na manipulação da Transformada de Fourier da imagem. Neste trabalho serão utilizadas apenas técnicas de processamento no domínio espacial. Entre as operações comuns a esta etapa estão o realçamento de contornos e a filtragem de ruídos.

Funções de processamento de imagens no domínio espacial podem ser definidas como

$$g(x,y) = T[f(x,y)]$$

onde *f(x,y)* é a imagem inicial, *g(x,y)* é a imagem processada, e *T* é um operador em *f*, definido em alguma vizinhança de *(x,y)*. Com isso, define-se uma subimagem, ou região dentro da imagem inicial, onde a função é definida. Por exemplo, pode-se utilizar a vizinhança 3x3 do *pixel* localizado em *(x,y)*. A região de definição da função estará centrada no *pixel (x,y)* (Figura 2.12). A operação sobre toda a imagem é realizada

movendo-se a subimagem sobre cada *pixel* da imagem inicial, gerando os valores da imagem processada.

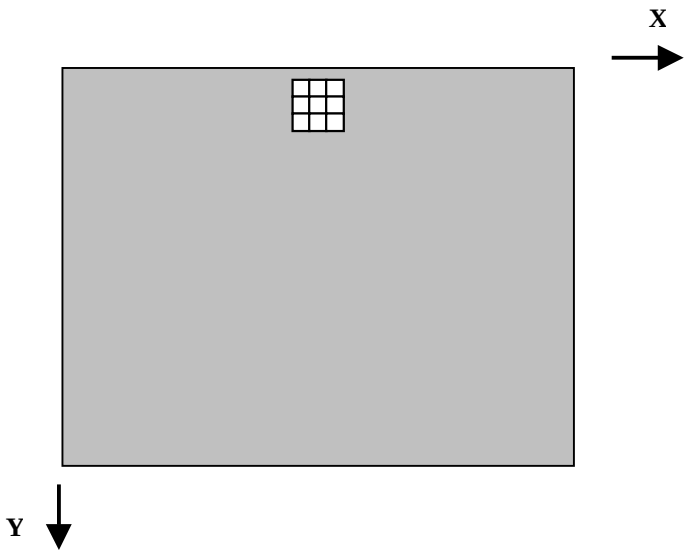


Figura 2.12 – Região 3x3 em uma imagem [11].

Funções de processamento tais como *máscara espacial*, *máscara de convolução* ou *filtro espacial* são, basicamente, uma matriz bidimensional cujos coeficientes são escolhidos para detectar determinada propriedade em uma imagem. Para exemplificar, seja o seguinte problema: dada uma imagem que é composta por um fundo cuja tonalidade de cinza é constante, e, sobre ele, vários pontos isolados de tonalidade diferente do fundo. O objetivo é detectar esses pontos. Uma máscara comumente utilizada neste tipo de aplicação é aquela mostrada na Figura 2.13.

-1	-1	-1
-1	8	-1
-1	-1	-1

Figura 2.13 – Máscara 3x3 para detecção de pontos isolados sobre fundo constante [16].

O processo de detecção se dá da seguinte maneira: o centro da máscara (cujo valor é 8) é deslocado sobre todos os *pixels* da imagem inicial. Para cada posição da

máscara sobre a imagem, multiplica-se o valor (nível de cinza) de cada *pixel* encoberto por ela, pelo coeficiente correspondente na máscara, ou seja, o *pixel* da imagem que está sob o *pixel* central da máscara deve ser multiplicado por 8, enquanto a sua vizinhança 8 deve ser multiplicada por –1. Os resultados dessas 9 multiplicações são então somadas. Se todos os *pixels* sob a máscara têm o mesmo valor (fundo constante), a soma será zero. Se, por outro lado, o centro da máscara estiver localizado sobre um ponto distinto do fundo, a soma será diferente de zero. Pode-se, assim, perceber que é possível detectar os pontos, pois estes estarão destacados sobre um fundo negro na imagem resultante [16].

Pode-se generalizar este processo para uma máscara 3x3 considerando a Figura 2.14. Sejam os coeficientes da máscara espacial representados por w_1, w_2, ..., w_9 e sejam os vizinhos N_8 de (x, y). Então:

$$\begin{aligned} T[f(x,y)] = {} & w_1 f(x-1, y-1) + w_2 f(x-1, y) + w_3 f(x-1, y+1) + \\ & + w_4 f(x, y-1) + w_5 f(x, y) + w_6 f(x, y+1) + \\ & + w_7 f(x+1, y-1) + w_8 f(x+1, y) + w_9 f(x+1, y+1) \end{aligned}$$

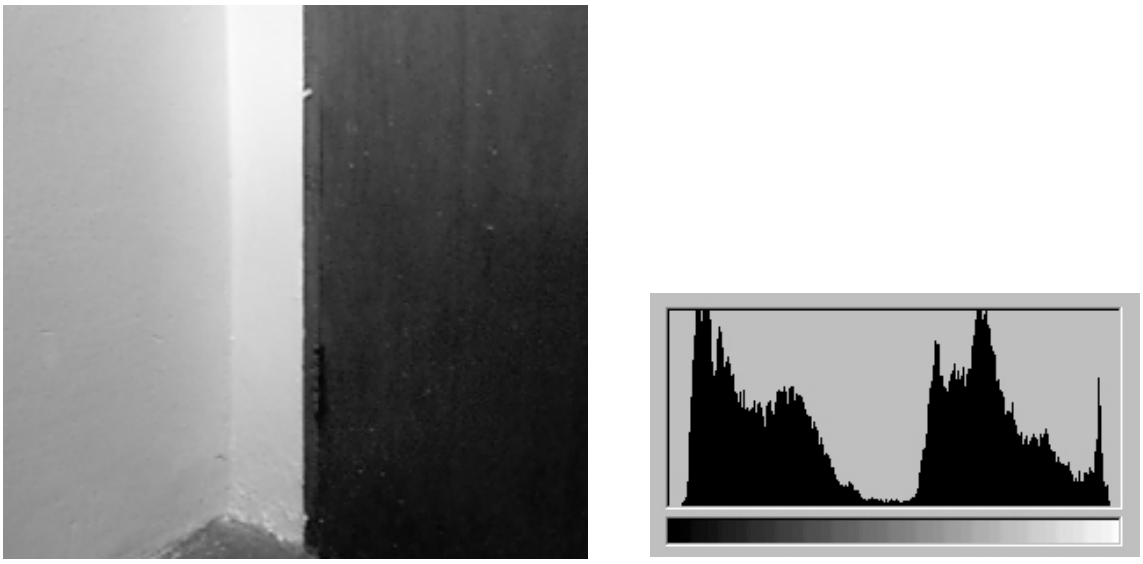
w_1 (x-1, y-1)	w_2 (x-1, y)	w_3 (x-1, y+1)
w_4 (x, y-1)	w_5 (x, y)	w_6 (x, y+1)
w_7 (x+1, y-1)	w_8 (x+1, y)	w_9 (x+1, y+1)

Figura 2.14 – Máscara geral 3x3 mostrando os coeficientes e as posições correspondentes dos *pixels* [16].

- **Segmentação da imagem**

O quarto passo trata da *segmentação* da imagem. Nesta etapa, a imagem pré-processada é subdividida em seus componentes principais. O que se faz é tentar separar as regiões que possuam as mesmas características ou os contornos mais efetivos da imagem. Essa é uma etapa crítica do processo. Se for bem planejada, as etapas subsequentes têm grande possibilidade de serem bem sucedidas.

Uma ferramenta largamente utilizada nesta etapa é a operação conhecida como *threshold* [13]. Freqüentemente, essa técnica é utilizada após uma análise do *histograma* da imagem, para efetuar a segmentação. O histograma de uma imagem digital com níveis de cinza na faixa [0, L-1], onde L é o número de núveis de cinza, é uma função discreta $p(r_k) = n_k/n$, onde r_k é o k-ésimo nível de cinza, n_k é o número de *pixels* da imagem que apresentam esse nível de cinza, e k = 0, 1, 2, ..., L-1. Uma imagem exemplo (mostrando a fronteira entre uma porta fechada e uma parede),e seu respectivo histograma, são mostrados na Figura 2.15.



(a) (b)

Figura 2.15 – (a) Fronteira entre parede e porta fechada, (b) histograma correspondente.

Como se pode observar na Figura 2.15(b), esse histograma possui uma característica aproximadamente bimodal, ou seja, os *pixels* estão separados em dois grupos distintos, o que nos permitiria selecionar um nível de *threshold T* que separasse

os dois grupos, realizando a segmentação. Neste exemplo, o agrupamento da esquerda representa a porta (tonalidade mais escura) enquanto o da direita representa a parede (tonalidade mais clara). Uma vez selecionado o nível de *threshold*, a imagem segmentada resultante *g(x, y)* é definida como

$$g(x,y)=\begin{cases}1 & se \quad f(x,y)>T \\ 0 & se \quad f(x,y)\leq T.\end{cases}$$

A Figura 2.16 mostra a imagem após a operação de *threshold*.

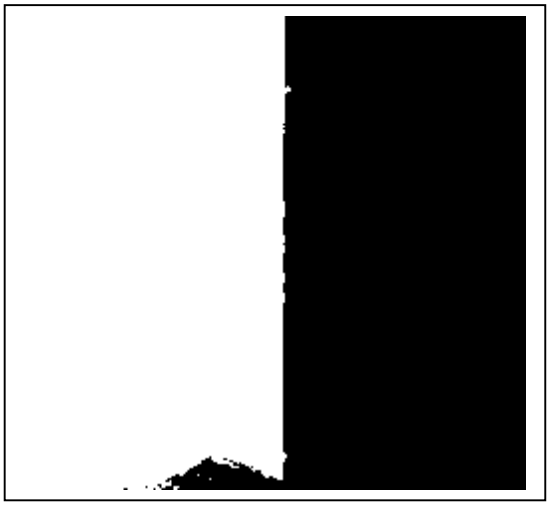


Figura 2.16 –Imagem da Figura 2.15(a) após a operação de *threshold*, permitindo a diferenciação entre parede e porta (T=128).

- **Representação e descrição das informações**

A quinta etapa do processo trata da *representação* e *descrição* das informações obtidas do processo de segmentação. Uma vez que os elementos de interesse da imagem tenham sido individualizados, deve-se escolher uma maneira adequada de representar esses elementos. Isto significa que se deve decidir se o elemento de interesse na imagem fica melhor representado usando apenas seu contorno ou se é necessário representar todos os *pixels* da região que o elemento ocupa. Exemplificando, se houver interesse na textura da superfície do objeto, tem-se que representar toda a região que ele ocupa, ao passo que se o objetivo é identificar a forma do objeto, por exemplo, somente aqueles

pixels pertencentes aos contornos seriam suficientes para a sua representação. Uma vez escolhida a representação adequada, deve-se fazer a descrição dos elementos. A descrição é também chamada de *seleção de características.* Nesta fase tem-se que eleger características dos objetos que possam futuramente contribuir para a sua identificação. Essas características são também conhecidas como *elementos descritores*. Um exemplo de característica descritora de um objeto seria o número de buracos ou vazios que ele contém.

- **Reconhecimento e interpretação**

A última etapa do processo refere-se ao *reconhecimento* e à *interpretação*. A fase de reconhecimento associa uma etiqueta ou "*label*" aos objetos, baseada nas informações providas pelos descritores. A fase de interpretação visa o entendimento ou tradução do conjunto de objetos etiquetados.

Cabe agora falar um pouco sobre a *base de conhecimento*. A base de conhecimento refere-se ao conjunto de informações *a priori* que se tem sobre o problema. Como se pode ver na Figura 2.11, o canal de comunicação entre a base de conhecimento e as diversas etapas é de duplo sentido, e o canal de comunicação entre os processos é de sentido único. Isto significa que a base de dados guia e troca informações com os diversos processos, ao passo que a saída de cada etapa representa a entrada da etapa posterior. Exemplificando, suponha que o problema é o reconhecimento automático do número de telefone em uma ficha de cadastro de clientes de uma loja. Então, na base de conhecimento constará a informação do número de dígitos que devem ser reconhecidos, por exemplo sete dígitos. Se durante o processo foram distinguidos somente seis elementos distintos, então a base de dados vai ordenar que seja feita uma nova segmentação, porque provavelmente dois dígitos foram identificados como um só elemento. Finalmente, vale destacar que o esquema exposto refere-se a um sistema genérico. Várias aplicações em processamento de imagens podem ser realizadas com apenas algumas das etapas anteriormente descritas, como é o caso, por exemplo, da eliminação de ruído e realçamento de bordas, operações muito utilizadas quando as imagens são destinadas à análise humana.

Capítulo 3

Integração Multisensorial e Estratégias de Controle

3.1 - Introdução

Conforme comentado no Capítulo 1, os robôs móveis utilizam diversos sensores para a realização de uma determinada tarefa. Esta utilização de sensores múltiplos permite que um certo grau de inteligência seja incorporado aos robôs, fazendo com que eles possam desempenhar suas tarefas sem a intervenção direta do homem. Tal utilização de sensores é uma forma de solucionar o problema de impossibilidade ou inviabilidade de que o robô disponha, *a priori*, do grau de conhecimento necessário sobre as condições do ambiente de trabalho, de modo a permitir sua operação autônoma [17].

O sensoriamento torna-se imprescindível, ao menos quando o robô tem que operar em um ambiente completamente desconhecido, visto que não há informações prévias a respeito dos objetos presentes no ambiente [18-20]. O mesmo ocorre quando se dispõe de informações *a priori* sobre o ambiente, se o processamento e armazenamento se tornam inviáveis com os recursos computacionais disponíveis. Além disso, deve-se considerar as possíveis mudanças na configuração do ambiente, o que praticamente obriga a fazer atualizações sucessivas das informações armazenadas, novamente usando as informações provenientes de tais sensores.

Nas últimas décadas, pesquisadores têm estudado a utilização de sensores múltiplos como forma de aumentar a autonomia de máquinas em geral e de robôs. Os campos de aplicação são vários e dentre eles pode-se citar o reconhecimento automático de alvos, a navegação de robôs móveis, as montagens industriais, o seguimento de alvos e a navegação aérea, áreas em que já há resultados significativos [17].

Existem vários métodos utilizados para fazer a integração da informação proveniente de sistemas baseados em sensores múltiplos. O método mais direto é aquele que utiliza a informação proveniente de cada sensor como uma entrada diferente para o sistema de controle. Este método é mais apropriado para sistemas cujos sensores provêem informações relativas a diferentes aspectos do ambiente. Por exemplo, sensores infravermelhos podem captar a presença de corpos emitindo radiação térmica enquanto as câmaras de vídeo captam imagens de objetos que estejam sob boas condições de iluminação. A maior vantagem deste método é o aumento na complexidade do ambiente que poderia ser detectado, em contraste com a limitação existente se for utilizado um único tipo de sensor. Por outro lado, é possível obter-se um efeito cooperativo da integração de sensores múltiplos utilizando a informação proveniente de um sensor para guiar a operação de outros sensores (por exemplo, um sensor de proximidade acusaria a presença de um objeto à curta distância de um robô, para que uma câmara de vídeo capture uma imagem do objeto detectado), ou então pela combinação ou fusão das informações provenientes dos vários sensores (um exemplo que pode ser considerado é o da cobra cascavel, que realiza a fusão das informações provenientes dos sensores visuais e dos sensores de infravermelho para fazer a distinção entre um rato e um sapo) [17]. As próximas seções distinguirão os processos de integração e fusão sensorial.

3.2 – Integração Multisensorial

A integração multisensorial pode ser implementada de diversas maneiras, porém algumas funções básicas são comuns à maioria dos sistemas existentes. Como exemplo, considere-se o sistema mostrado na Figura 3.1, no qual são mostradas as funções básicas da integração multisensorial.

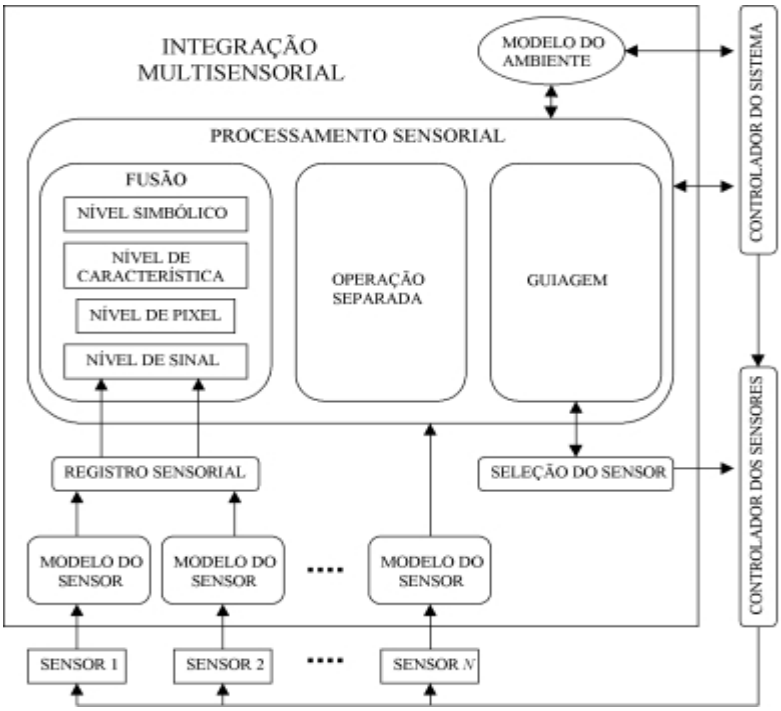


Figura 3.1 – Diagrama funcional da integração e fusão multisensorial na operação de um sistema [17].

Um grupo de *N* sensores provê a entrada para o processo de integração. Antes, porém, de sofrer o processo de integração em si, os dados provenientes dos sensores precisam ser modelados. Um *modelo do sensor* representa a incerteza e o erro associado aos dados provenientes de cada sensor e representa uma medida da sua qualidade que será utilizada posteriormente pelas funções de integração. Uma suposição comumente empregada é a de que a incerteza associada aos dados provenientes dos sensores pode ser modelada por uma distribuição Gaussiana. Após o modelamento, cada sensor pode

ser integrado ao sistema de acordo com três tipos de processamento sensorial: *fusão*, *operação separada* e *guiagem*.

No exemplo da Figura 3.1, os dados provenientes dos sensores 1 e 2 são fusionados. Antes da fusão, os dados devem sofrer um processo de correspondência. O módulo responsável por essa correspondência chama-se *registro sensorial* e refere-se aos meios utilizados para relacionar os dados provenientes dos sensores nas dimensões espacial e temporal, ou seja, fazer com que os dados se refiram ao mesmo ponto do ambiente no mesmo período de tempo. As modalidades de fusão sensorial mostradas na Figura 3.1 serão comentadas na próxima seção.

Caso os dados provenientes de um sensor sejam muito diferentes dos dados obtidos pelos demais sensores, a integração sensorial é realizada na modalidade *operação separada*, o que significa que existe apenas uma influência indireta sobre a operação dos demais sensores, baseada nos efeitos que este sensor em especial exerce sobre o sistema de controle e sobre o modelo do ambiente. Isto pode ocorrer, por exemplo, quando um sensor de luminosidade informa as condições de iluminação do ambiente ao sistema de controle de um avião bombardeiro, para que este, então, ative o tipo de sensor adequado à identificação do alvo a ser atingido, ou seja, sob boas condições de luminosidade, uma câmara de vídeo pode ser utilizada para localizar um tanque inimigo em terra, ao passo que, à noite, a identificação poderia ser feita com sensores infravermelhos.

A *guiagem* é caracterizada quando os dados provenientes de um sensor são utilizados para guiar a operação de outros sensores. Um exemplo clássico deste tipo de integração é a utilização da informação visual para guiar sensores de tato montados em uma garra de um robô manipulador. Ainda como exemplo, neste trabalho, será usada a informação proveniente de um anel de sensores ultra-sônicos para guiar a operação de uma câmara CCD de vídeo monocular, que é responsável pela captura das imagens usadas no reconhecimento de um conjunto de objetos presentes no laboratório onde o robô opera.

Após deixarem o módulo de *processamento sensorial*, os dados são usados para a construção do modelo do ambiente. O modelo do ambiente reúne informações a respeito do local de operação do robô e pode conter tanto informações previamente conhecidas como informações atualizadas que chegam através do sistema sensorial. O modelo do ambiente pode variar de acordo com o propósito do sistema, ou seja, quando a finalidade do sistema é apenas o reconhecimento de alguns objetos, o modelo do ambiente pode apresentar somente os modelos dos objetos a serem identificados. Quando o propósito é a navegação de um robô móvel, o modelo poderá conter uma descrição detalhada do local, tanto dos objetos presentes quanto das condições do terreno.

O último módulo é o da *seleção do sensor*, e refere-se aos meios utilizados para fazer a escolha de um sensor em particular ou do grupo de sensores que serão utilizados pelo sistema [17].

3.3 – Fusão Multisensorial

A fusão das informações provenientes de vários sensores, ou mesmo de um único sensor, em um determinado intervalo de tempo pode se dar em diferentes níveis de representação. No exemplo da Figura 3.1, pode-se observar os seguintes níveis de representação: *nível de sinal*, *nível de pixel*, *nível de característica* e *nível simbólico* de representação. Os dados provenientes dos sensores comumente usados na prática podem ser fusionados em um ou mais níveis de representação.

A fusão sensorial realizada no *nível de sinal* refere-se à combinação dos sinais provenientes de um grupo de sensores e geralmente produz um sinal da mesma forma que os sinais originais, porém de qualidade superior. Este é o nível que requer uma maior sincronização dos dados, tanto espacial quanto temporal.

A fusão sensorial realizada no *nível de pixel* é utilizada para aumentar a quantidade de informação associada a cada *pixel* de uma imagem que foi formada pela

combinação de outras imagens. Por exemplo, a fusão de uma imagem que representa um mapa de profundidades com uma imagem bidimensional de intensidade (imagem representada, por exemplo, em tonalidades de cinza), associará a cada *pixel* da imagem de intensidade uma informação da profundidade associada àquele ponto.

A fusão sensorial realizada no *nível de característica* é utilizada para aumentar a probabilidade de que uma característica extraída da informação proveniente de um sensor corresponda a um importante aspecto do ambiente, bem como para criar características compostas adicionais que possam ser usadas pelo sistema. Características compostas são geradas a partir da combinação de características existentes. Características típicas extraídas de imagens e utilizadas na fusão são os contornos e regiões de semelhante intensidade e/ou profundidade.

A fusão sensorial realizada no *nível simbólico* permite que a informação proveniente de vários sensores seja utilizada conjuntamente no nível mais alto de abstração. A fusão no *nível simbólico* talvez seja o único meio no qual a informação sensorial, proveniente de sensores dissimilares ou referente a diferentes aspectos do ambiente, pode ser fusionada.

Como será visto adiante, neste trabalho não será usada a fusão sensorial como parte do processo de integração multisensorial, e sim o que se chamou de *fusão de comportamentos*[21].

3.4 – Estratégias de Controle

Uma vez definida a forma como será realizada a integração multisensorial, é necessário especificar um sistema de controle que realize a tarefa de coordenação das diversas atividades desenvolvidas pelo robô, tais como a integração dos dados sensoriais, a construção e atualização do mapa do ambiente de operação e o planejamento da trajetória [21].

Os sistemas de controle geralmente empregados em robôs móveis podem ser divididos em dois grupos, de acordo com a distribuição dos níveis que compõe o sistema: decomposição horizontal (caracterizado pelas estruturas tradicionais de controle) e decomposição vertical (caracterizado pelas estruturas baseadas em comportamentos) [4].

3.4.1 – Decomposição Horizontal

Este tipo de arquitetura baseia-se na construção de um modelo do ambiente e no planejamento subsequente das ações do robô. Este tipo de abordagem resulta em um processo serial, onde cada módulo é executado seqüencialmente, para a realização das tarefas. A Figura 3.2 mostra um esquema representativo de uma arquitetura horizontal.

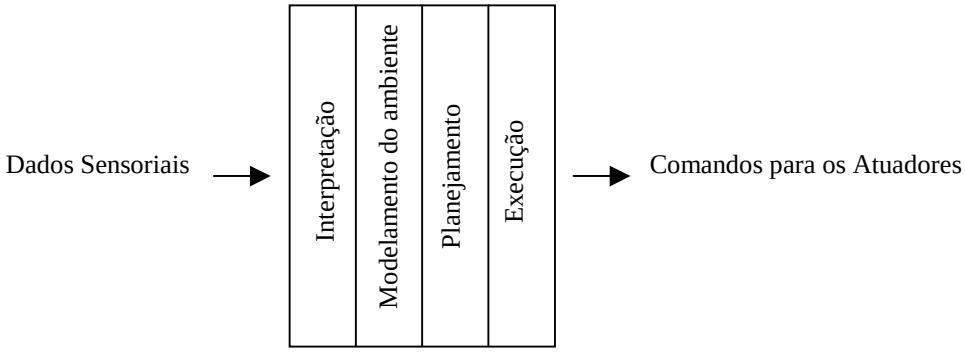


Figura 3.2 – Arquitetura com distribuição horizontal para controle de robôs móveis [22].

Após os dados provenientes de todos os sensores terem sido coletados, o ruído inerente às medições e os dados conflitantes são fusionados em um ou mais níveis descritos na Seção 3.3. Então, um modelo do ambiente contendo as descrições geométricas dos objetos presentes, bem como de sua posição e orientação, é construído. Para executar uma determinada tarefa, o modelo previamente construído é utilizado para planejar a seqüência de operações que alcançará o resultado. As operações planejadas são executadas pelo envio de comandos aos atuadores [22].

Uma vantagem em relação a este tipo de arquitetura é a utilização de uma informação global a respeito do ambiente. Entretanto, a informação global é baseada no modelo do ambiente, que, para ser confiável, deve contar com sensores de alta precisão e uma minuciosa calibração, tornando o sistema dispendioso.

Uma desvantagem associada a este tipo de arquitetura é que se deve armazenar e processar grande quantidade de dados. No caso de robôs móveis autônomos, este é um fator preponderante, pois geralmente eles devem carregar todo o *hardware* necessário à sua operação.

As arquiteturas com decomposição horizontal caracterizam-se pela seqüência: leitura dos sensores/ processamento dos dados/ ação. Logo, se houver uma mudança no ambiente entre a fase de leitura dos sensores e a execução da ação, o planejamento feito pelo sistema poderá falhar. Isto ocorre porque o sistema deve resolver os dados sensoriais conflitantes, atualizar o modelo do ambiente e otimizar o planejamento antes de executar qualquer ação, e isto acaba comprometendo o seu desempenho.

3.4.2 – Decomposição Vertical

Neste tipo de arquitetura, as camadas do sistema de controle têm uma distribuição paralela, onde cada camada é geralmente associada a um comportamento [21]. Como conseqüência, a fusão sensorial característica das arquiteturas tradicionais dá lugar à fusão de comportamentos (Figura 3.3).

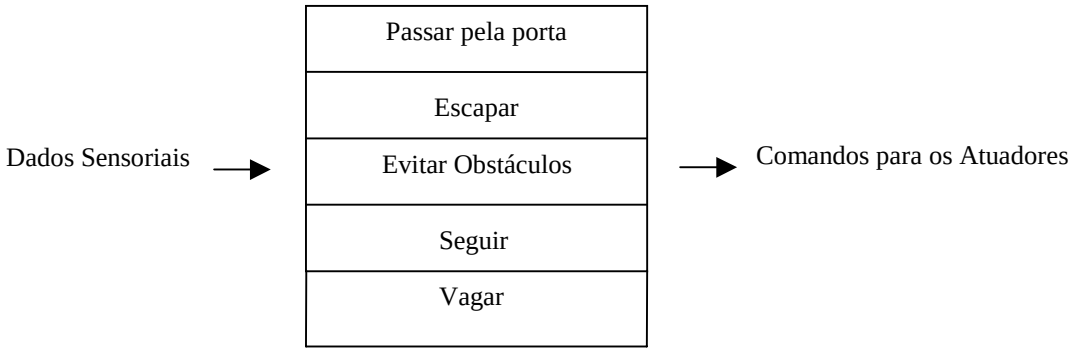


Figura 3.3 – Arquitetura com distribuição vertical para o controle de robôs móveis [22].

O sistema é decomposto em elementos (camadas) relativamente desacoplados, que possuem seus próprios objetivos, especialidades e interfaces, e são responsáveis por comportamentos específicos do robô. Esses elementos são executados paralelamente entre si e apresentam percursos próprios de processamento, do sensoriamento à ação dos atuadores, o que permite uma resposta rápida. A relativa independência entre esses elementos resulta em modularidade e flexibilidade para o sistema [4].

A arquitetura de subsunção (*subsumption architecture* na literatura inglesa) proposta por Brooks [10,23-25] incorpora as características de um sistema com decomposição vertical, combinando controle distribuído em tempo real com sensores ativando comportamentos [22].

Os comportamentos são as camadas do sistema de controle e atuam paralelamente, sempre que disparados pelos sensores. A fusão é feita, então, à saída dos comportamentos, e não à saída dos sensores, como nas arquiteturas clássicas. Um esquema de arbitração de prioridades é utilizado para resolver os conflitos entre comportamentos. Não existe, portanto, uma estrutura unificada de dados ou um modelo geométrico do ambiente.

Os comportamentos estão agrupados no sistema em níveis de competência. Os níveis mais altos sobrepõem os níveis mais baixos na tomada de decisões (Figura 3.4). Novos comportamentos podem ser adicionados ao sistema, sobrepondo-se aos que já existem, sem haver necessidade de realizar modificações nos mesmos. Essa inserção de novos comportamentos, no entanto, não é uma tarefa óbvia, pois é necessário conhecer perfeitamente as especificações dos níveis inferiores, pois tanto as ações de controle realizadas por eles como as saídas que produzem sobre os atuadores devem ser levadas em conta [26].

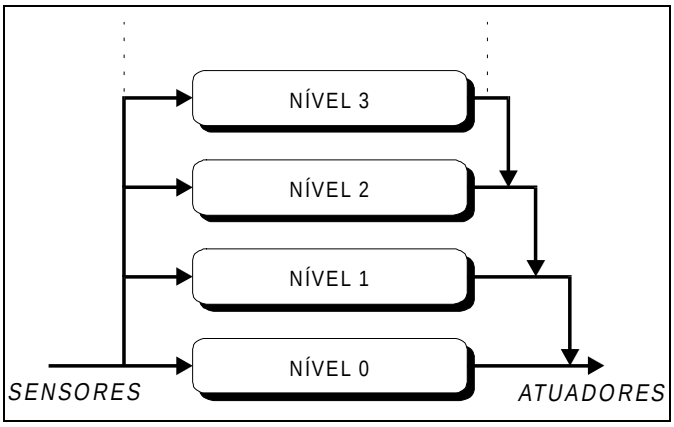


Figura 3.4 - Decomposição vertical do sistema de controle em níveis de competência [21].

3.5 – Agente Sensor, Agente Comportamento e Agente Atuador

Uma das formas de controlar um sistema contendo sensores e atuadores para a realização de comportamentos é utilizar controle baseado em agentes [4]. Os agentes são entidades computacionais que funcionam concorrentemente e que podem se comunicar [27]. Estes agentes podem ser representados por módulos concorrentes e intercomunicantes distribuídos em três categorias (Figura 3.5): agente *sensor*, agente *comportamento* e agente *atuador* [28].

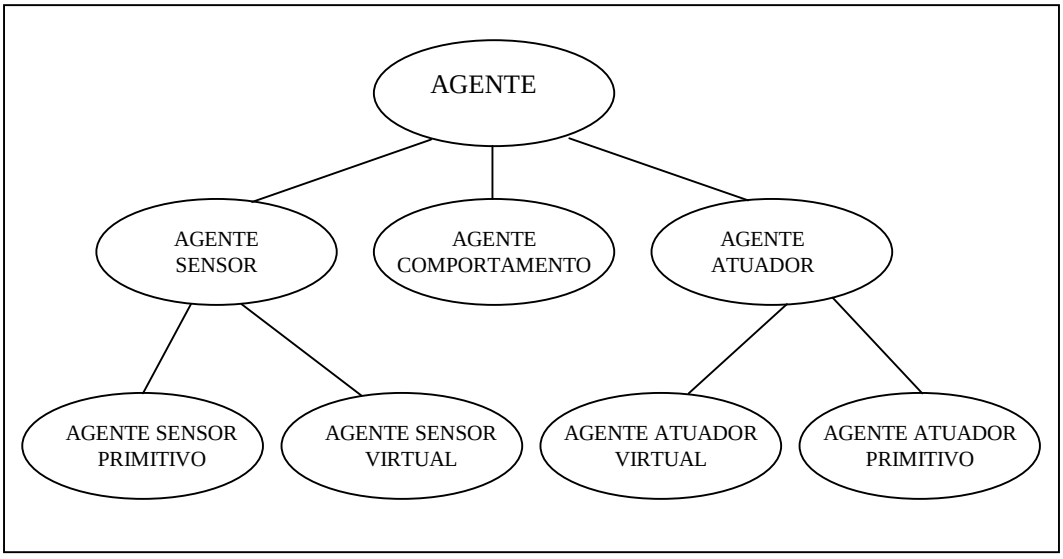


Figura 3.5 – Categorias de agentes [21].

Nesta abordagem, utilizada por Brooks [10], [23-25] e Schneebeli [27], [28], um módulo ou grupo de módulos é responsável por uma atividade (comportamento) a ser executado pelo robô. Cada atividade representa um objetivo a ser atingido por uma parte do sistema. Mecanismos de interação entre os módulos são responsáveis pela mediação e definição do objetivo prioritário [27].

Os agentes *sensor* têm basicamente duas atribuições: monitorar periodicamente e coletar os sinais dos sensores do robô, e realizar tratamentos nesses sinais e enviar os sinais tratados aos agentes que deles necessitarem. Os agentes *sensor* são divididos em dois grupos: agente *sensor primitivo* e agente *sensor virtual*. Os agentes *sensor primitivo* apresentam a função de interagir diretamente com os sensores do robô. Os agentes *sensor virtual* têm a função de processar e realizar tratamentos sobre as informações provenientes de um ou mais agentes *sensor primitivo*. Os agentes *comportamento* têm como funções a tomada de decisões a respeito das atitudes do robô e todo o processamento relacionado com os comportamentos produzidos por ele. Eles são ativados ao receberem mensagens provenientes dos agentes *sensor primitivo*, *sensor virtual* ou de outros agentes *comportamento*. Os agentes *atuador* irão, por meio dos comandos enviados aos atuadores, agir sobre o meio em que se encontram. Esses agentes são elementos análogos às camadas de execução e atuação das arquiteturas tradicionais de controle. Assim como os *agentes sensor*, os *agentes atuador* são divididos em dois grupos: agentes *atuador virtual* e agentes *atuador primitivo*. Os agentes *atuador virtual* irão receber as informações de um ou mais agentes *comportamento*, e processá-las para depois transmiti-la aos agentes *atuador primitivo*, que têm a tarefa de interagir diretamente com os atuadores do robô [21],[27].

Para realizar a construção deste sistema de controle, utiliza-se a ferramenta desenvolvida por Xavier [21],[27], a qual usa a linguagem C++ e bibliotecas de concorrência. Com isso, pode-se construir uma estrutura concorrente orientada a objetos agrupando características como modularidade, encapsulamento, herança, concorrência e a eficiência da linguagem C++.

Nessa estrutura, os módulos de controle (agentes), são construídos como instâncias de classes definidas pelo usuário, que por sua vez são derivadas das classes

sensor primitivo, *sensor virtual*, *comportamento*, *atuador virtual* e *atuador primitivo*. Desse modo, múltiplos agentes podem ser criados a partir de uma única definição de classe, por instanciação, mesmo em tempo de execução. Neste caso, tais agentes seriam diferenciados uns dos outros através dos argumentos passados ao construtor da nova classe e pelas portas de comunicação estabelecidas. A instanciação e a desinstanciação das classes que originam os módulos de controle, mais os mecanismos provenientes do envio de mensagens expressas, proporcionam reconfigurabilidade dinâmica ao sistema [29]. A expansão do sistema de controle é possível pelo acréscimo de novas classes derivadas das classes mostradas na Figura 3.5. O encapsulamento dos dados e funções em classes, por sua vez, assegura a modularidade do sistema [21],[27].

3.6 – Uma Proposta de Sistema de Controle

Para mostrar como o sistema de sensoriamento externo desenvolvido para o robô Brutus (composto pelo sistema ultra-sônico [4] associado à visão artificial) pode ser integrado à unidade de controle central, um exemplo de sistema de controle baseado em agentes que é capaz de usar a informação fornecida pelo sistema de sensoriamento é proposto na Figura 3.6 [30].

O sistema de controle é composto por 16 agentes sensores primitivos chamados S_i, i=1,...,16, dois agentes sensores primitivos chamados *Encoder1* e *Encoder2*, três agentes sensores virtuais, um chamado *Ultrasonic_Sensor*, um chamado *Camera_Sensor* e o terceiro, chamado *Recognizer*, dois agentes comportamento, chamados *Collision_avoidance* e *Behavior-Changer*, três agentes atuadores virtuais, chamados *Go_ahead*, *Rotate* e *Activate_motor*, e, finalmente, dois agentes atuadores primitivos, chamados *Cam_Motor* e *PID*.

Esses agentes estão conectados para poderem implementar três comportamentos ou níveis de controle, denominados *Motor_Control*, *Wandering* e *Obstacle-Based_Path_Definition*.

Cada agente S_i, i=1,...,16, é responsável por disparar um dos transdutores ultra-sônicos e processar o sinal de eco (medir a distância, digitalizar a envoltória, e medir o seu valor de pico). Essas informações, adicionadas à prioridade absoluta associada a cada agente S_i [4], [30], são passadas como uma mensagem ao agente *Ultrasonic_sensor*, que mantém duas listas, uma com a distância medida e outra com a prioridade absoluta associada a cada agente S_i. Quando o agente *Ultrasonic_sensor* recebe uma mensagem que contém uma distância menor que 1 m (indicando que um obstáculo a esta distância foi detectado), ele envia uma mensagem ativando os agentes *Collision_avoidance* e *Camera_Sensor*.

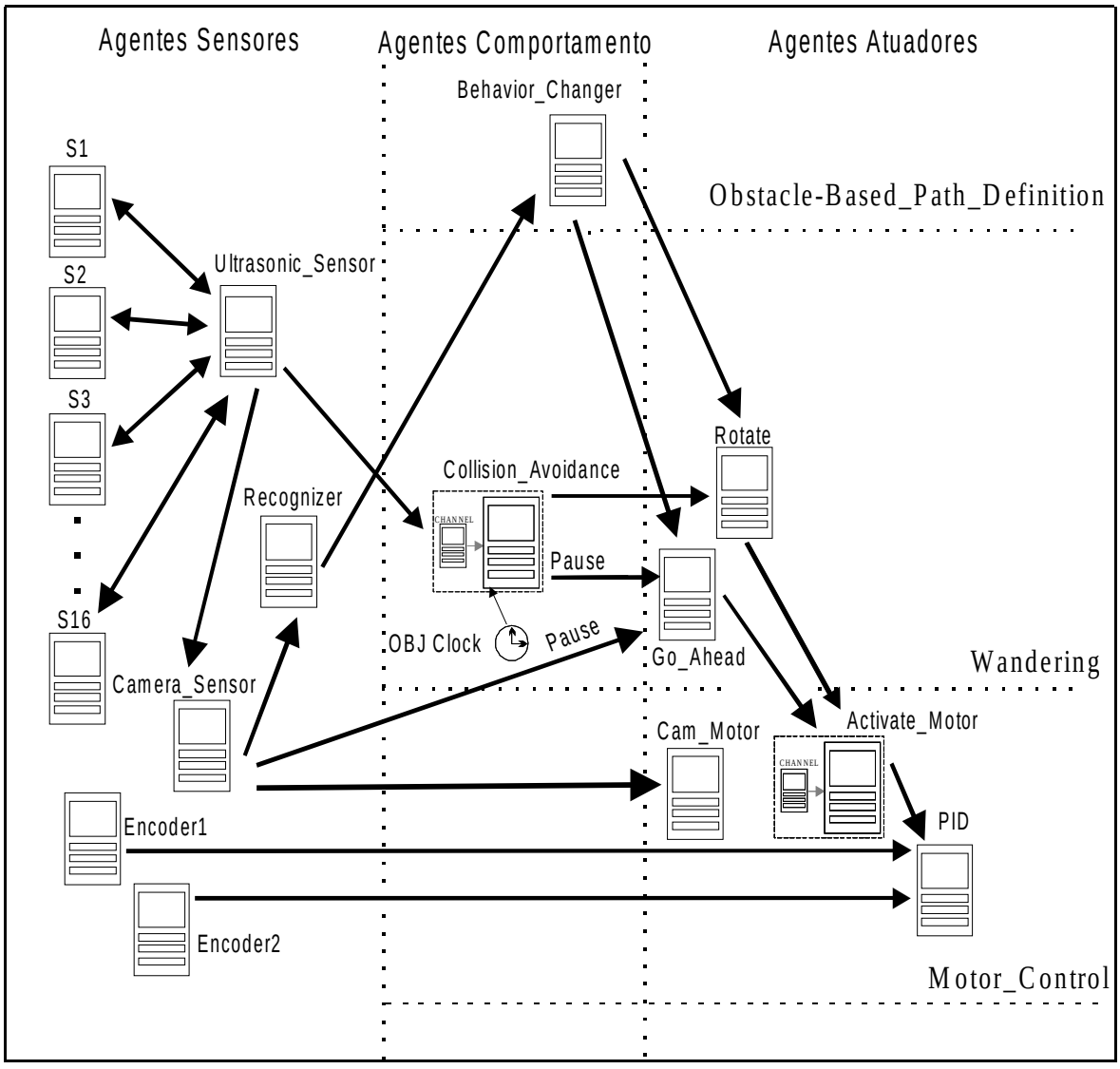


Figura 3.6 – Um sistema de controle baseado em agentes.

Quando um agente S_i envia uma mensagem ao *Ultrasonic_sensor* ele recebe como resposta outra mensagem informando a soma das prioridades absolutas associadas aos agentes *S*. Desta forma, cada agente S_i é capaz de calcular sua prioridade relativa, pela divisão da sua prioridade absoluta pelo valor da soma das prioridades totais informadas pelo agente *Ultrasonic_sensor*. O inverso desta prioridade relativa é utilizado para programar seu relógio interno (objeto *Clock*), mudando o intervalo de tempo em que o agente é ativado [4]. Uma vez que o hardware é compartilhado por todos os sensores ultra-sônicos, quando um agente S_i é ativado ele passa a uma fila e fica esperando a sua vez. Quando a prioridade associada ao agente S_i aumenta, o intervalo de tempo entre cada ativação se reduz e o agente S_i vai à fila mais freqüentemente.

Se o agente *Ultrasonic_sensor* recebe uma mensagem que contém uma distância menor que 40 cm, o agente *Camera_Sensor* se desativa e o agente *Collision_avoidance* começa o procedimento para evitar obstáculos. Inicialmente, ele envia uma mensagem de pausa ao agente *Go_ahead*, que pára o robô enquanto ele gira sobre seu eixo central, devido a uma mensagem também enviada pelo agente *Collision_avoidance*, agora ao agente *Rotate*. A direção de rotação é determinada de forma a afastar o robô do obstáculo. Após algum tempo, medido pelo objeto *Clock* associado ao agente *Collision_avoidance*, esse agente envia uma mensagem desativando o agente *Rotate* e, então, a mensagem de pausa enviada ao agente *Go_ahead* é suspensa. A partir deste instante o robô volta a explorar o ambiente.

Se a rotação efetuada não for suficiente para evitar a colisão com o obstáculo, certamente outro agente *S* o detectará a uma distância menor que 40cm e o processo novamente se repetirá. O agente *Collision_avoidance* tem uma entrada chamada *canal* que pode receber mensagens de todos os agentes *S* e realizar algumas operações sobre elas, tais como calcular o valor médio, o maior valor e o menor valor. Nesse caso, o *canal* informa ao agente *Collision_avoidance* a menor distância recebida (direção mais provável para a colisão).

Porém, se o agente *Ultrasonic_sensor* recebe uma mensagem que contém uma medida de distância na faixa de 40 cm a 1 m, o agente *Collision_avoidance* se desativa, e o agente *Camera_Sensor* começa o procedimento de reconhecimento do obstáculo. A primeira etapa é o envio de uma mensagem de pausa ao agente *Go_ahead*, que então pára os motores. Em seguida, o agente *Camera_Sensor* ativa o agente *Cam_Motor*, que gira a câmara em direção ao agente S_i que detectou o obstáculo. Então, o agente *Camera_Sensor* captura uma imagem do obstáculo detectado enviando o resultado ao agente *Recognizer*. Este agente executa o reconhecimento do obstáculo. Uma vez reconhecido o obstáculo, um novo comportamento deve ser definido para o robô, o qual contemplará o desvio do obstáculo ou outra atitude, dependendo de qual é o obstáculo (por exemplo, seguir através da porta, se ela estiver aberta). Esse novo comportamento é definido por um agente denominado *Behavior-Changer*, o qual é ativado pelo agente *Recognizer*, quando este identifica o obstáculo (mesmo que o obstáculo não seja identificado, o agente *Recognizer* ativa o agente *Behavior-Changer*, que define uma atitude padrão de desvio do obstáculo). Finalmente, uma mensagem enviada pelo agente *Behavior-Changer* ativa os agentes *Go_ahead* e *Rotate*, permitindo que o robô volte a se mover.

O agente *Activate_motor* também possui uma entrada *canal* que informa ao agente a soma das componentes de velocidade enviados pelos agentes *Go_ahead* e *Rotate*. O agente *Activate_motor* calcula a velocidade que deve ser aplicada a cada motor e envia a informação ao agente *PID*, que é responsável pelo controle de ambos os motores.

Neste trabalho, é discutido e implementado um procedimento para efetuar o reconhecimento de alguns objetos presentes no ambiente de trabalho do robô, assim como é tratada a integração de tal estrutura no sistema sensorial e no sistema de controle do robô. O resultado é a proposição dos agentes *Camera_Sensor*, *Cam_Motor* e *Recognizer* apresentados na Figura 3.6, assim como sua integração no sistema de controle como um todo. O sistema ultra-sônico, inicialmente concebido para prover toda a informação sensorial, foi desenvolvido em um trabalho prévio [4]. Assim, a combinação das informações sensoriais, na forma descrita neste capítulo, permite ao robô Brutus navegar sem um conhecimento prévio detalhado do seu entorno, limitando-

se tal conhecimento à caracterização dos objetos que podem estar presentes no ambiente (incluído no agente *Recognizer*).

Finalmente, vale mencionar que os detalhes acerca da tarefa de reconhecimento dos obstáculos, embutida no agente *Recognizer*, serão discutidos no próximo capítulo.

Capítulo 4

O Sistema de Visão Artificial Proposto

O Capítulo 2 cobriu alguns tópicos básicos relacionados aos sistemas de visão artificial e ao processamento de imagens. O presente capítulo descreverá o trabalho específico aqui desenvolvido, do ponto de vista do sistema de visão que será instalado a bordo do robô móvel. Serão abordados os algoritmos de processamento das imagens, a estratégia de reconhecimento dos objetos e os resultados alcançados. Como será visto neste capítulo, a estratégia de reconhecimento baseia-se no número de contornos verticais, na distância relativa entre os contornos e na tonalidade média de cinza das regiões formadas entre os contornos.

4.1 – Captura das Imagens

Para realizar os testes dos algoritmos desenvolvidos, foram capturadas várias imagens do laboratório em que o robô móvel Brutus deve operar. Essas imagens contêm os objetos que ele deverá ser capaz de reconhecer. Os objetos que podem ser reconhecidos são quinas/cantos, pés-de-cadeiras, pés-de-mesas, paredes e portas (abertas ou fechadas). Estes objetos são os mais comuns no ambiente de operação do robô.

O dispositivo utilizado para capturar as imagens foi uma câmara fotográfica digital Kodak DC20. Este modelo de câmara apresenta foco automático, tem comprimento focal de 47 mm e permite capturar imagens em duas resoluções: 493 x 373 e 320 x 240 *pixels*, ambas no modo *true color* (cada cor é representada utilizando-se 24 bits). Este dispositivo foi escolhido pela facilidade que se tem em transferir as imagens, já digitalizadas, para um computador tipo PC (plataforma escolhida para testar a estratégia de reconhecimento dos objetos). Em sua versão final, porém, os algoritmos aqui mostrados serão instalados no computador de bordo do robô, e será utilizada uma câmara de vídeo CCD, também a bordo do robô, como dispositivo de captura de imagens. Então, o objetivo deste trabalho se resume em desenvolver e testar os algoritmos de processamento das imagens e a estratégia de reconhecimento dos objetos presentes no ambiente do robô móvel.

As imagens dos objetos foram capturadas sob duas condições de iluminação: a iluminação artificial utilizada no laboratório, composta por lâmpadas fluorescentes, e a iluminação natural proveniente das janelas do laboratório. Nas próximas seções serão mostradas as imagens dos objetos que foram utilizadas nos testes dos algoritmos, bem como as imagens intermediárias de cada fase do processamento.

4.2 – Pré-Processamento das Imagens Capturadas

4.2.1 – O *Software DECVISION*

Como parte do desenvolvimento desta dissertação, foi realizado um estágio de um mês no Instituto de Automática (INAUT) da Universidade Nacional de San Juan, Argentina. Esta instituição participa de um projeto conjunto com a UFES, financiado pela FUNDACIÓN ANTORCHAS (Argentina) e CAPES (Brasil), denominado "Sistema de Controle de Robôs Móveis em Ambientes Parcialmente Estruturados". Neste estágio foi possível utilizar o *software DECVISION* (*Development Enviroment for Computer Vision*), que é um ambiente de desenvolvimento para visão computacional criado pelos pesquisadores do INAUT. Este ambiente utiliza a linguagem de programação *Microsoft Visual C++*, sob plataforma *Microsoft Windows*, e possui várias ferramentas comumente usadas no processamento de imagens, permitindo, também, que o usuário acrescente suas próprias ferramentas na forma de *DLL's (Dynamic Link Libraries)*. Todos os algoritmos desenvolvidos neste trabalho foram implementados na forma de *DLL's* que podem ser acrescentadas ao *software DECVISION*. Assim é que, neste trabalho, as imagens foram trabalhadas usando o referido *software DECVISION*.

4.2.2 – Fases do Pré-Processamento

Após a captura das imagens, algumas operações preliminares se tornaram necessárias para que elas pudessem ser utilizadas no *software DECVISION*. O primeiro passo foi transformar as imagens coloridas capturadas em imagens contendo apenas tonalidades de cinza. Isto deve ser feito porque a câmara CCD a ser utilizada pelo robô só trabalha com imagens descritas por tons de cinza. O segundo passo foi reduzir o tamanho das imagens capturadas, já que as fotos digitais obtidas têm um tamanho maior que aquele gerado pela câmara CCD a ser usada, além do que o *software* utilizado, por estar em fase de desenvolvimento, tem limitação quanto ao tamanho das imagens que podem ser

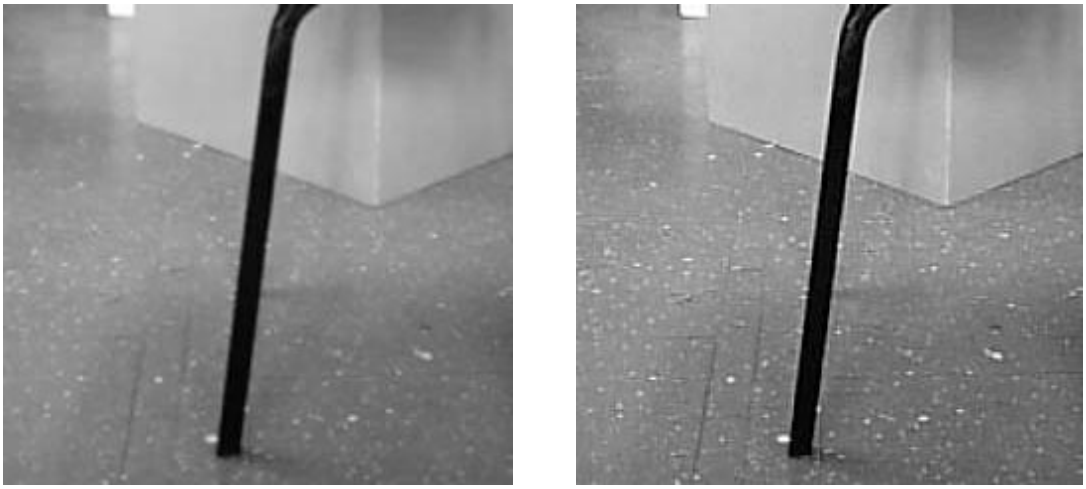
trabalhadas. As imagens, após estas operações iniciais, apresentavam tamanho de 256 x 242 *pixels* e 256 tonalidades de cinza. Essas operações foram realizadas utilizando o *software Corel Photo-Paint 7* da *Corel Corporation*. Vale ressaltar que essas operações iniciais não serão realizadas quando o sistema for implementado à bordo do robô, onde a visualização das etapas de processamento não é requerida. As próximas subseções tratarão dos filtros utilizados em cada fase do pré-processamento.

4.2.2.1 – Agudização (*Sharpening*)

Uma vez no formato adequado, as imagens sofreram uma filtragem chamada agudização (*sharpening*). O filtro espacial utilizado é mostrado na Figura 4.1 e tem a propriedade de tornar mais nítidas as transições entre regiões diferentes da imagem. Essa operação mostrou-se necessária, especialmente, para que pudesse ser realizado o reconhecimento dos pés-de-mesa, que possuem uma tonalidade de cinza muito próxima à tonalidade do fundo da cena (*background*). Uma desvantagem associada à utilização deste tipo de filtro é a amplificação do ruído existente na imagem. A aplicação dos filtros espaciais foi tratada no Capítulo 2, Seção 2.3.1. As Figuras 4.2(a) e 4.2(b) mostram a imagem de um pé-de-cadeira antes e após a aplicação do filtro representado na Figura 4.1, respectivamente.

-1	1	-1
1	1	1
-1	1	-1

Figura 4.1 – Filtro espacial 3 x 3 utilizado para tornar mais agudas as formas na imagem.



(a) (b)

Figura 4.2 – (a) Imagem de um pé-de-cadeira; (b) aplicação do filtro da Figura 4.1.

4.2.2.2 – Detecção dos Contornos Verticais

Quando se trata de imagens, um contorno pode ser definido como o limite entre duas regiões que possuem níveis de cinza relativamente diferentes [11]. Para a detecção de um contorno, é comum a utilização de um operador derivativo local. Para explicar o processo de detecção de contornos pela aplicação de operadores derivativos locais, considere-se a Figura 4.3. A Figura 4.3(a) mostra uma imagem que apresenta uma faixa clara sobre um fundo escuro, o perfil de tons de cinza de uma seção transversal da imagem, e o gráfico da primeira derivada do perfil de tons de cinza. A Figura 4.3(b) apresenta as mesmas informações, porém, para o caso de uma faixa escura sobre um fundo claro. Analisando a Figura 4.3, percebe-se que a primeira derivada do perfil de tons de cinza é positiva nos trechos em que a transição ocorre de tons mais escuros para tons mais claros, negativa nos trechos em que a transição ocorre de tons mais claros para tons mais escuros e igual a zero em regiões de tonalidade constante. Sendo assim, torna-se possível, através da magnitude da primeira derivada do perfil de tonalidades, detectar contornos em uma imagem.

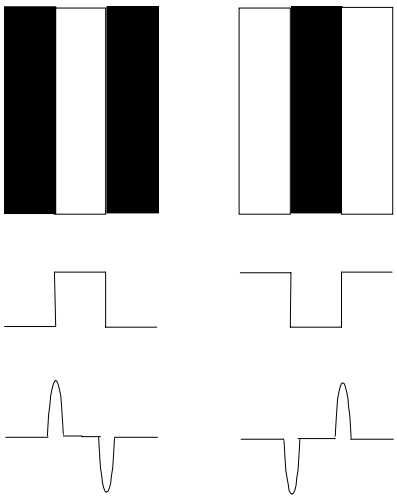


Figura 4.3 – (a) Faixa clara sobre fundo escuro; (b) Faixa escura sobre fundo claro.

Embora a discussão precedente tenha se limitado ao caso de um perfil de tonalidades unidimensional, pode-se utilizar o mesmo raciocínio para perfis com qualquer orientação na imagem. Basta que se defina o perfil de tonalidades perpendicularmente à direção do contorno, e, então, se repita o processo acima descrito. Tratando-se de imagens bidimensionais, a primeira derivada em qualquer ponto é obtida calculando-se a magnitude do gradiente naquele ponto [11]. Na prática, é comum fazer uso de operadores que aproximam a primeira derivada da imagem. Entre eles, pode-se citar os operadores de Sobel, Prewitt, Robinson e Kirsch [13]. Neste trabalho, utilizou-se uma das formas do operador de Kirsch para detectar os contornos verticais das imagens, pois esse operador se mostrou mais eficaz para o tratamento das imagens capturadas. A máscara utilizada é mostrada na Figura 4.4. A Figura 4.5 mostra o resultado da aplicação do operador de Kirsch mostrado na Figura 4.4 sobre a imagem da Figura 4.2(b).

-5	3	3
-5	0	3
-5	3	3

Figura 4.4 – Operador de Kirsch utilizado para detectar os contornos verticais.

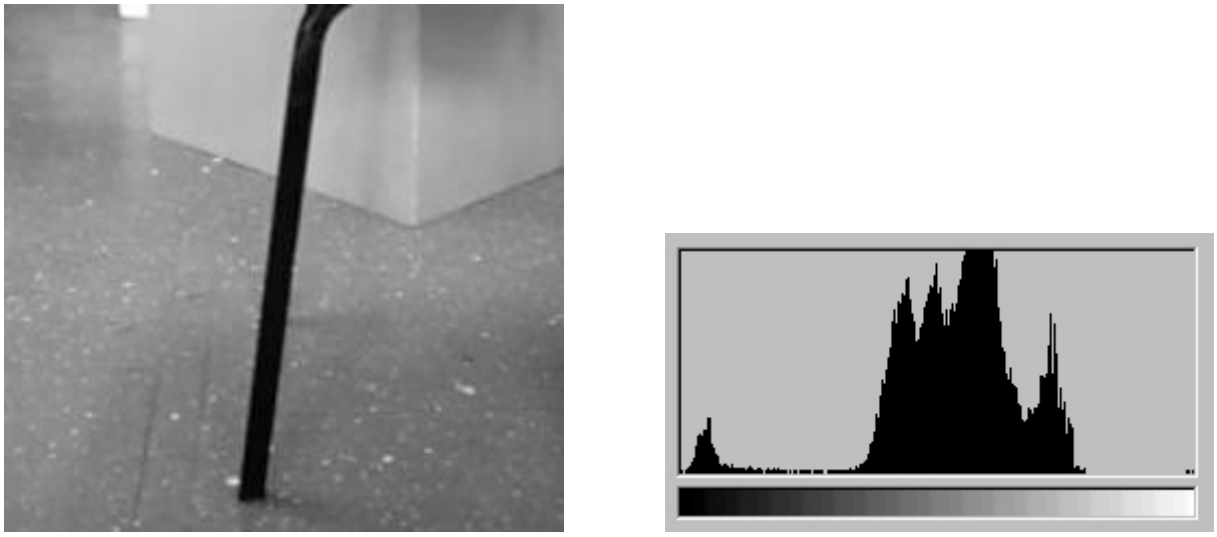
Figura 4.5 – Aplicação do operador de Kirsch sobre a imagem da Figura 4.2(b).

4.3 – Segmentação das Imagens

Como visto no Capítulo 2, a segmentação das imagens é um dos passos mais importantes de um sistema de visão artificial. Seu principal objetivo é dividir a imagem em partes que correspondam aos objetos ou elementos do mundo real contidos na imagem. Neste trabalho, a segmentação visa separar os contornos verticais mais significativos (em termos de tamanho) do restante da imagem. Para isto, foram necessárias duas etapas: *threshold* da imagem e fechamento de contornos abertos durante as fases de pré-processamento e *threshold*.

4.3.1 – *Threshold*

Como foi discutido no Capítulo 2 (Seção 2.3.1), em certos casos é possível realizar a segmentação de uma imagem utilizando somente o processo de *threshold*, notadamente quando o histograma dessa imagem apresente uma distribuição tal que possibilite a separação do objeto de interesse na imagem e o fundo (*background*) da cena. Neste trabalho, devido às características das imagens capturadas, torna-se difícil a separação dos objetos de interesse (pés-de-cadeira, pés-de-mesa, portas, cantos ou quinas) utilizando-se somente o processo de *threshold*. A Figura 4.6(a) representa um pé-de-cadeira, que é um objeto típico utilizado para testar os algoritmos de reconhecimento de objetos



(a) (b)

Figura 4.6 – (a) Pé-de-cadeira; (b) histograma correspondente.

desenvolvidos neste trabalho. A Figura 4.6(b) mostra o histograma correspondente à imagem do pé-de-cadeira. É possível verificar que se torna difícil a separação do pé-de-cadeira do restante da cena somente pela aplicação de um nível de *threshold* (Figura 4.7). Todavia, o *threshold* foi utilizado como uma fase inicial do processo de segmentação, visando diminuir a quantidade de informação presente nas imagens após a utilização do operador de Kirsch. Em uma escala de tonalidades que vai do valor 0 (correspondendo ao preto) até o valor 255 (correspondendo ao branco), para todas as imagens capturadas, foi

adotado, empiricamente, um nível de threshold de 200, ou seja, todos os *pixels* com tonalidades inferiores a 200 recebem o valor 0 e todos os *pixels* com tonalidades iguais ou superiores a 200 recebem o valor 255, como mostra a Figura 4.7.

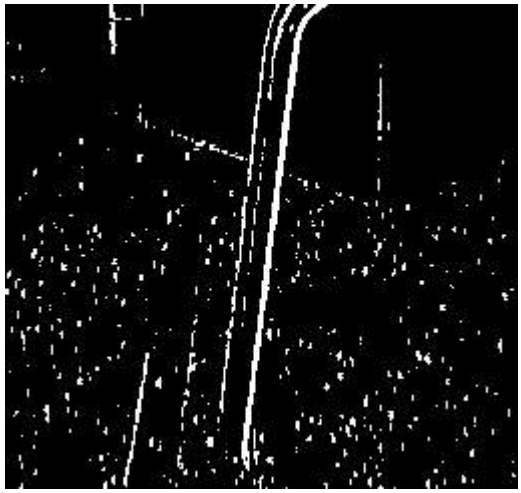


Figura 4.7 – *Threshold* de nível 200 aplicado à imagem da Figura 4.5.

4.3.2 – Função *VerticalLines*

O fluxograma que representa o algoritmo da função *VerticalLines* é mostrado na Figura 4.8. O objetivo da função *VerticalLines* é identificar os contornos verticais mais significativos, em termos de comprimento, presentes em uma imagem monocromática (contornos brancos sobre fundo preto, no caso presente). Essa imagem monocromática é o resultado da etapa em que foi realizada a operação de *threshold*. Além disso, durante a execução da função, ocorre uma redução do nível de ruído presente na imagem, restando, ao final do processo, somente os contornos verticais mais significativos. Simultaneamente, a função também fecha alguns contornos que tenham sido abertos durante a etapa de pré-processamento.

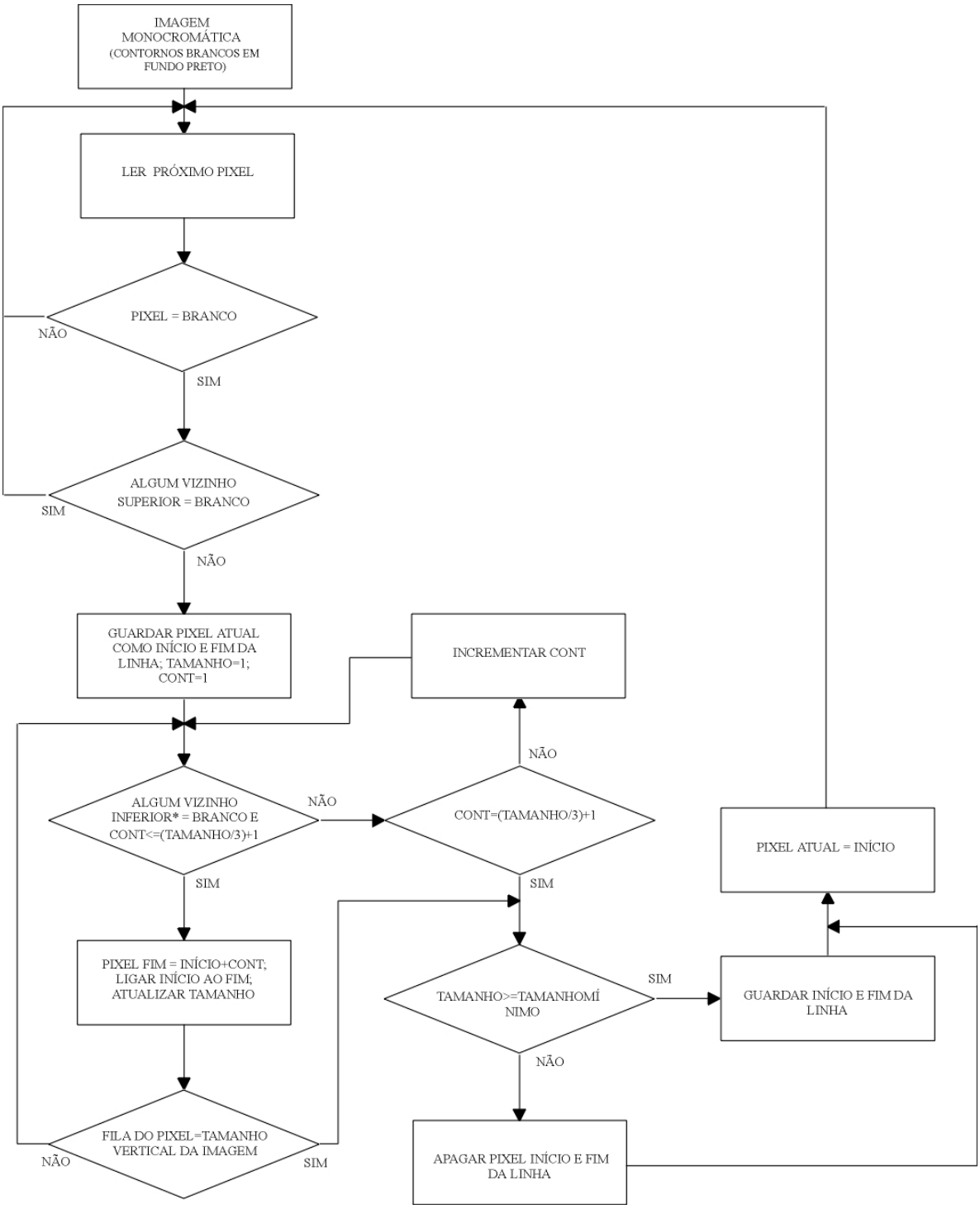


Figura 4.8 – Algoritmo da função *VerticalLines*.

A entrada da função é uma imagem monocromática gerada pela fase de pré-processamento. Essa imagem caracteriza-se por contornos brancos distribuídos sobre fundo

preto (Figura 4.7). Os *pixels* da imagem são lidos sucessivamente, linha por linha, até que seja encontrado um *pixel* branco. Quando um *pixel* branco é encontrado, realiza-se um teste para verificar se algum vizinho superior (ver Figura 4.9) deste *pixel* também é branco.

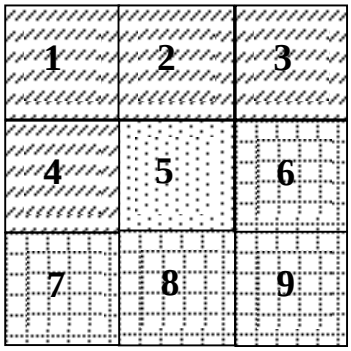


Figura 4.9 – Vizinhos superiores: *pixels* 1, 2 e 3; *pixels* já testados: 1, 2, 3 e 4; *pixel* atual (*pixel* que está sendo lido pela função): *pixel* 5; *pixels* não-testados: 6, 7, 8 e 9.

Caso se encontre algum vizinho superior que seja branco, o *pixel* atual (*pixel* que está sendo lido e testado pela função) não é armazenado e passa-se à leitura do próximo *pixel*. Isto acontece porque os vizinhos superiores de um *pixel* já foram testados devido à natureza seqüencial de leitura. Ora, se o *pixel* atual é branco, e algum de seus vizinhos superiores também o é, significa que fazem parte do mesmo contorno, e portanto, o *pixel* atual não necessita ser armazenado. Ou seja, o contorno ao qual ele pertence já foi identificado em etapas anteriores. Como os obstáculos que o robô deve reconhecer são formados basicamente por retas verticais, e como para definir uma reta são necessários apenas dois pontos, todos os *pixels* pertencentes a uma mesma reta (no caso presente, reta confunde-se com contorno) não precisam ser testados a cada passada do algoritmo; basta que se conheça o primeiro e último *pixel* da reta. Este teste tem por objetivo aumentar a velocidade do algoritmo.

Caso nenhum vizinho superior seja branco, significa que se está diante de um novo potencial contorno. Então, a função armazena o *pixel* atual como sendo o início e também o fim de uma nova reta, cujo comprimento, por enquanto, é de 1 (um) *pixel*. Neste ponto, a função buscará *pixels* que possam ser a continuação da nova reta encontrada. A região onde

será feita a busca é mostrada na Figura 4.10. Esta figura representa uma região qualquer em uma imagem. O *pixel* branco (A) é o *pixel* atual, isto é, o *pixel* que está sendo lido e testado. Como nenhum vizinho superior deste *pixel* é branco, significa que possivelmente um novo contorno foi encontrado, e, então, a busca pelo prolongamento do contorno é iniciada na região mostrada em cinza escuro.

				A					
			1a	1b	1c				
			2a	2b	2c				
			3a	3b	3c				
			4a	4b	4c				
			5a	5b	5c				
			6a	6b	6c				
			7a	7b	7c				
			8a	8b	8c				
			9a	9b	9c				

Figura 4.10 – Região de busca dos contornos (cinza escuro) e *pixel* atual (branco).

A seqüência de busca pelo prolongamento do contorno é a seguinte (Figura 4.10): *pixel* 1a, 1b, 1c, 2a, 2b, 2c, 3a, 3b, 3c, etc. O alcance vertical desta busca, ou seja, a quantas linhas localizadas abaixo do *pixel* atual (*pixel* A da Figura 4.10) vai se restringir a busca, é determinado por uma variável chamada *Alcance*, que é definida como

$$Alcance = \left(\frac{Tamanho}{3} \right) + 1$$

onde *Tamanho* representa o tamanho atual do contorno, em *pixels*. Esse foi um artifício utilizado para privilegiar os contornos maiores, ou seja, contornos que se encontram mais desenvolvidos terão um alcance maior na busca de seu prolongamento. Isto faz com que

contornos menores, que não são relevantes para a identificação do obstáculo, tenham uma possibilidade restrita de crescer.

Se nenhum *pixel* branco é encontrado dentro da região de busca definida pela variável *Alcance*, significa que não foi encontrado um prolongamento do contorno atual. Neste ponto, se a variável *Tamanho* for maior ou igual a 60% da altura da imagem (esse valor é empírico e pode ser ajustado na função) os *pixels* inicial e final são armazenados em um vetor para posterior reconhecimento do obstáculo. Se a variável *Tamanho* for menor que 60% da altura da imagem, significa que não é um contorno significativo para o reconhecimento do obstáculo e, então, os *pixels* inicial e final são apagados, ou seja, tornam-se pretos. Esse processo faz com que após analisado o último *pixel*, só restem na imagem os contornos significativos para o processo de reconhecimento, os quais, neste trabalho, foram definidos como sendo aqueles cujo comprimento em *pixels* se iguala ou excede 60% da altura da imagem. Isto permite isolar apenas o plano do obstáculo mais próximo, numa cena.

Se algum *pixel* branco é encontrado dentro da região de busca definida pela variável *Alcance*, significa que provavelmente este *pixel* pertence ao contorno. Então, a função liga através de uma reta, o *pixel* que originou a busca pelo prolongamento (o último *pixel* do contorno) e o *pixel* que foi encontrado dentro da região de busca, o que acarreta o aumento do comprimento do contorno e consequentemente a atualização da variável *Tamanho* e das variáveis que armazenam os *pixels* inicial e final do contorno. Novamente, a partir do final atualizado do contorno, é iniciada uma nova busca de um possível prolongamento e o processo se repete até que não se encontre mais nenhum *pixel* dentro do alcance, ou que se tenha atingido a última linha da imagem.

Um exemplo ilustrando a imagem que resulta quando a função *VerticalLines* é aplicada à imagem da Figura 4.7 é mostrado na Figura 4.11.

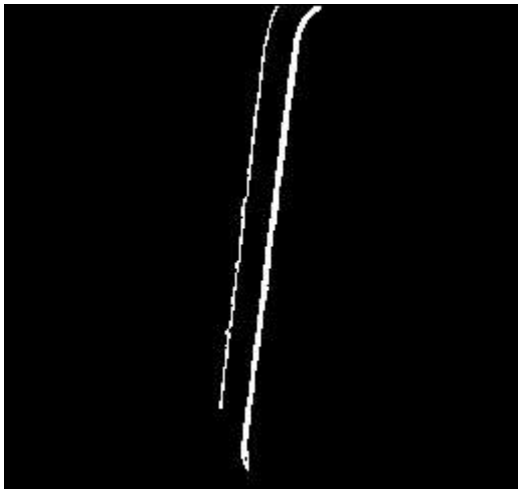


Figura 4.11 – Resultado da aplicação da função *VerticalLines* à imagem da Figura 4.7.

4.4 – Reconhecimento dos Obstáculos

Como foi mencionado anteriormente, os objetos que podem ser reconhecidos são paredes, portas, quinas ou cantos, pés-de-mesa e pés-de-cadeira presentes no Laboratório de Automação Inteligente da UFES. A estratégia de reconhecimento baseia-se no fato de que os objetos citados acima compõem-se, basicamente, de contornos verticais. Por essa razão, como explicado nas seções anteriores, as imagens são processadas até que somente restem os contornos verticais mais importantes da imagem, que, no caso, são os que apresentem um tamanho mínimo medido em *pixels*. Para se efetuar o reconhecimento são considerados o número de contornos presentes na imagem processada, a distância relativa entre esses contornos e a tonalidade média de cinza das regiões entre os contornos.

A primeira característica levada em consideração para o reconhecimento dos objetos é o número de contornos presentes na imagem processada. Se o número de contornos detectados for igual a 0 (zero), então o obstáculo é reconhecido como um plano infinito.

Se o número de contornos detectados for igual 1 (um), a função calcula a diferença de tonalidade média de cinza das regiões à esquerda e à direita do contorno. Se essa

diferença for maior que 110 ou ambas as tonalidades forem menores que 150, o obstáculo é reconhecido como sendo uma porta. Se nenhuma dessas condições for satisfeita, então o obstáculo é reconhecido como sendo um canto ou quina. Se o número de contornos detectados for igual a 2, a função calcula a diferença entre a tonalidade média da região à esquerda do primeiro contorno (os contornos são enumerados da esquerda para a direita) e a tonalidade média da região à direita do segundo contorno. A função calcula também a distância entre os dois contornos. Se essa distância for menor que 12 (doze) *pixels*, o obstáculo é encarado como desconhecido. Se a distância calculada estiver entre 12 e 20 *pixels* e a diferença entre as tonalidades das regiões for menor que 110, o obstáculo é reconhecido como um pé-de-cadeira; se maior que 110, como uma porta. Se a distância calculada estiver entre 20 e 70 *pixels* e a diferença entre as tonalidades das regiões for menor que 110, o obstáculo é reconhecido como um pé-de-mesa; se maior que 110, como uma porta. Se a distância calculada for maior que 70 *pixels*, o obstáculo é reconhecido como uma quina.

Se o número de contornos detectados estiver no intervalo fechado de 3 a 10, os contornos são testados em duplas, seguindo o mesmo procedimento para o caso de haver somente 2 contornos. A função não reconhecerá o obstáculo que gerar mais que 10 contornos. Todos os valores de distância, tonalidades e diferenças de tonalidades de cinza utilizadas para identificar os obstáculos foram obtidos através de experimentação e se mostraram eficazes para as variações de iluminação e de distância da câmara aos obstáculos permitidos neste trabalho. Também se deve ter em conta que tais parâmetros seriam, necessariamente, modificados para a detecção de outros tipos de obstáculos ou mesmo para outro ambiente. A seguir, serão mostradas as imagens que foram usadas na fase de testes da função, com os resultados obtidos e alguns comentários que facilitarão o entendimento da estratégia de reconhecimento explanada nesta seção.

Um esquema que representa o processo de reconhecimento dos obstáculos pode ser visto na Figura 4.12.

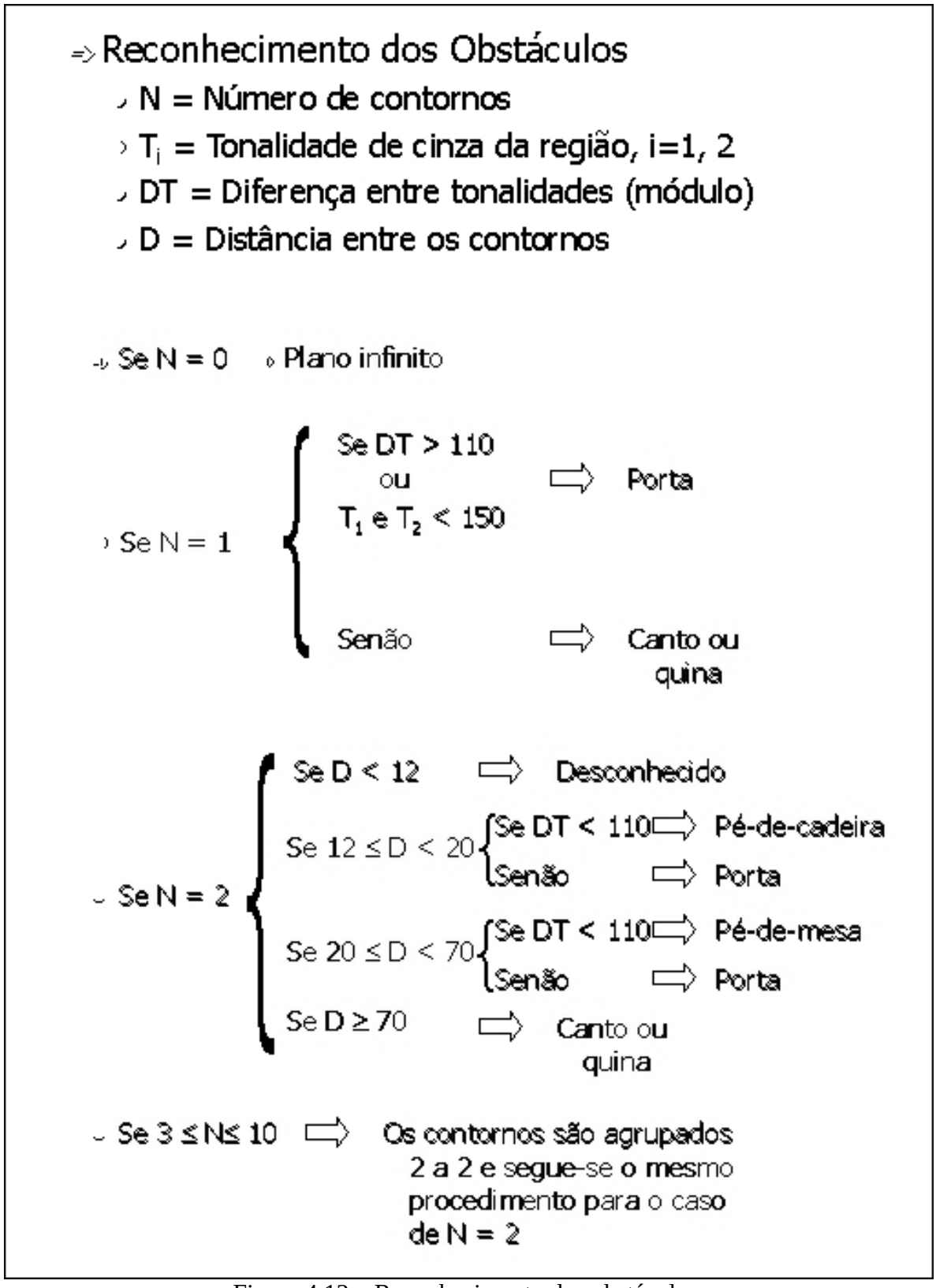
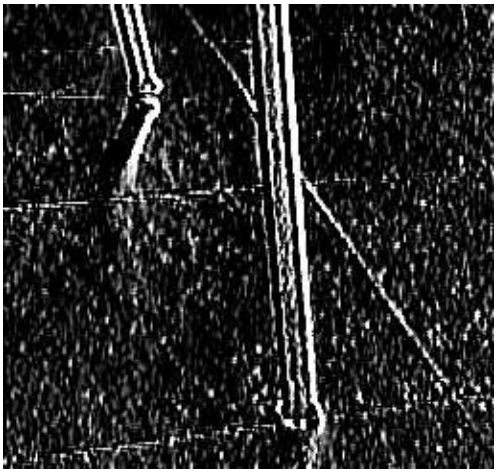


Figura 4.12 – Reconhecimento dos obstáculos.

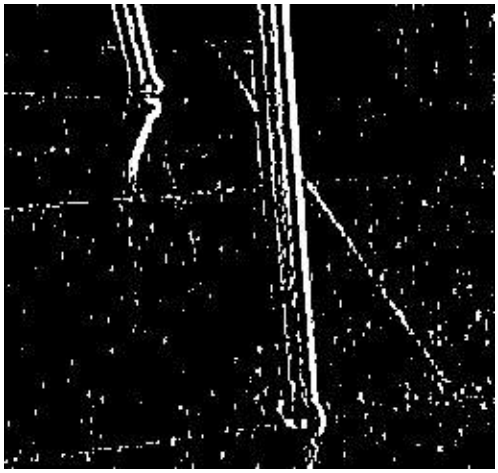
4.5 – Resultados Alcançados

(a)

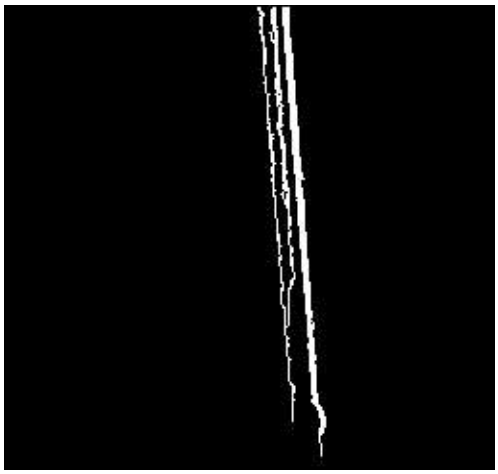
(b)



(c)

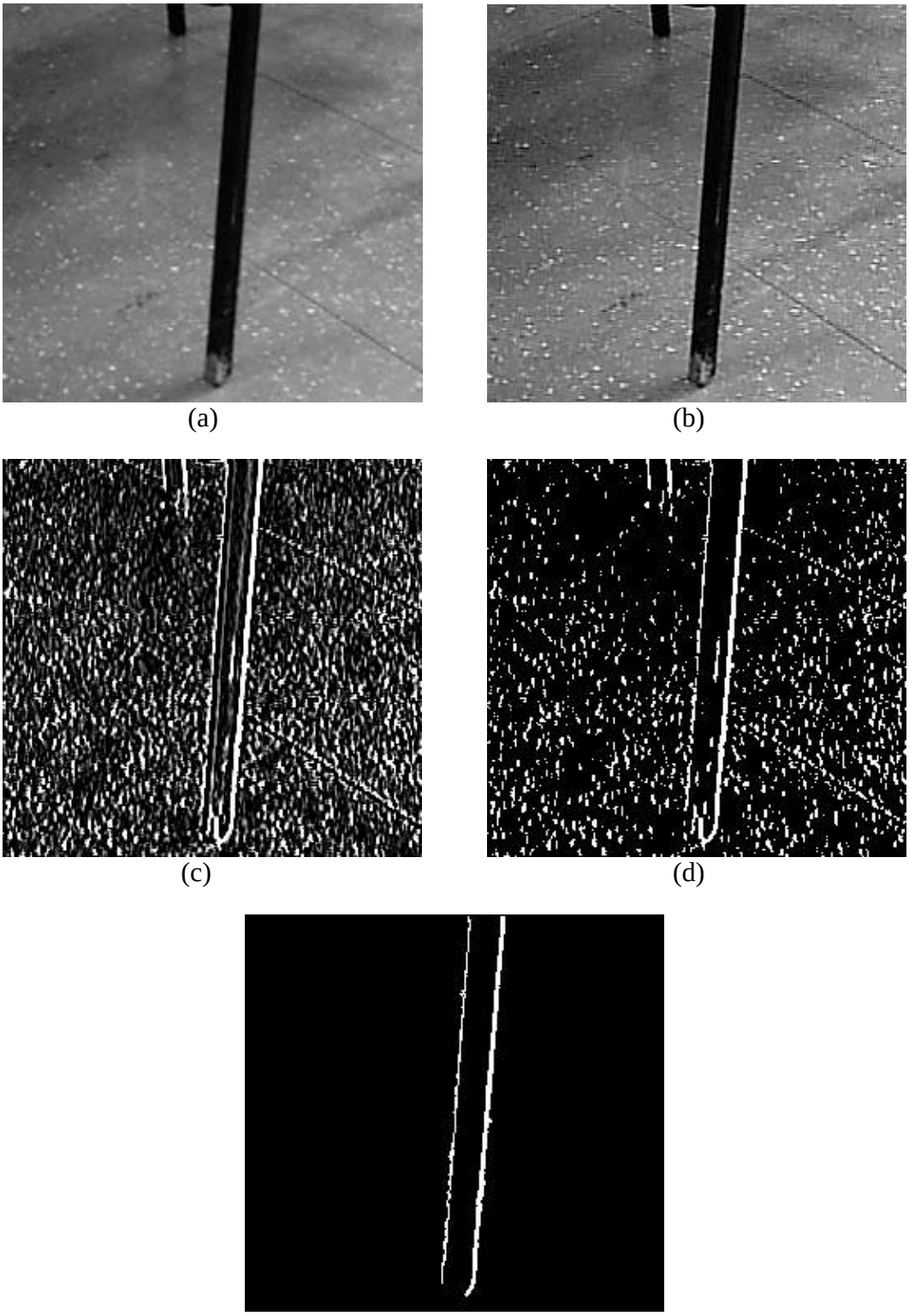


(d)



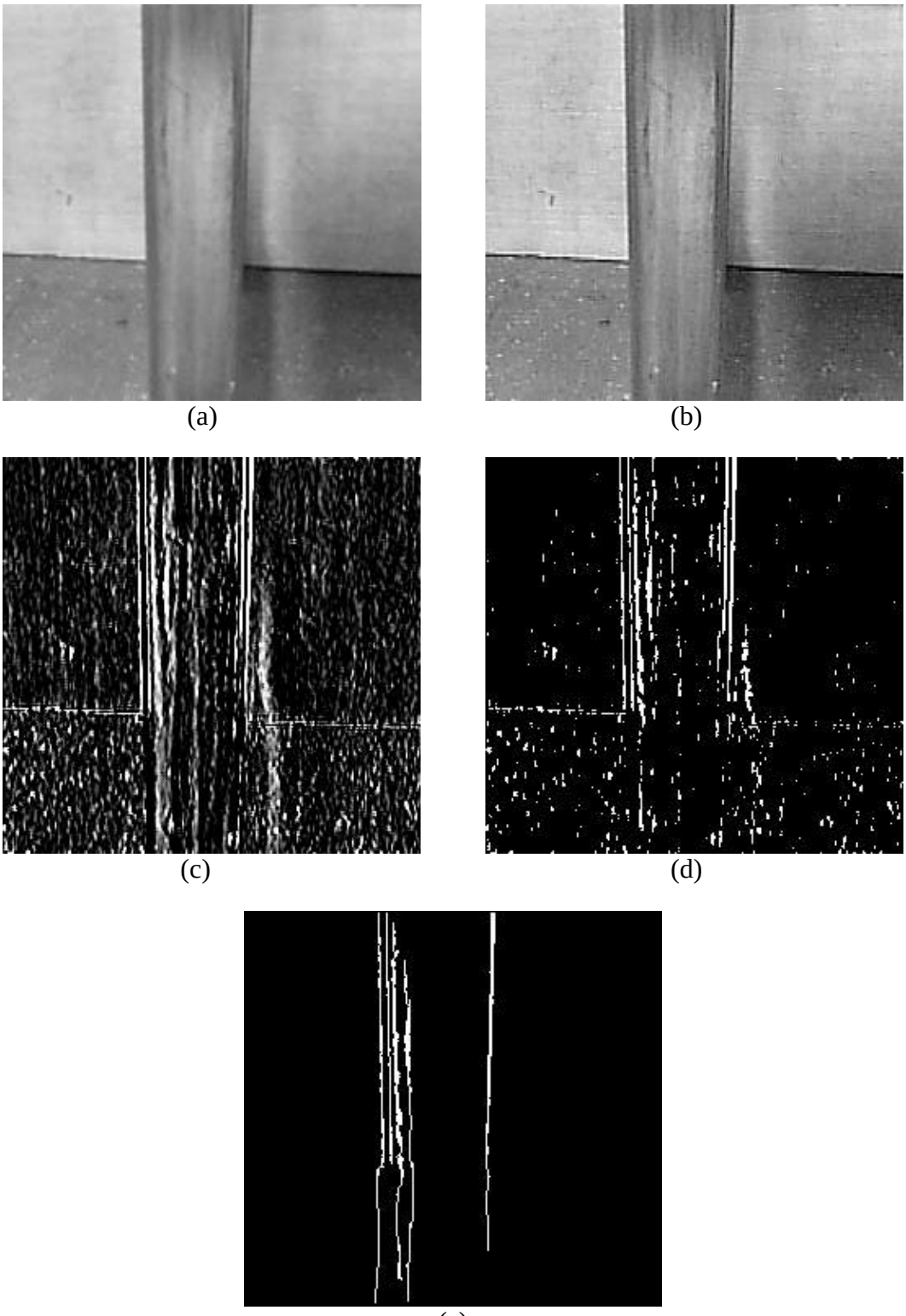
(e)

Figura 4.13 – (a) Pé-de-cadeira; (b) Após realçamento dos contornos; (c) Após aplicação do operador de *Kirsch*; (d) Após *threshold*; (e) Após aplicação da função *VerticalLines*.



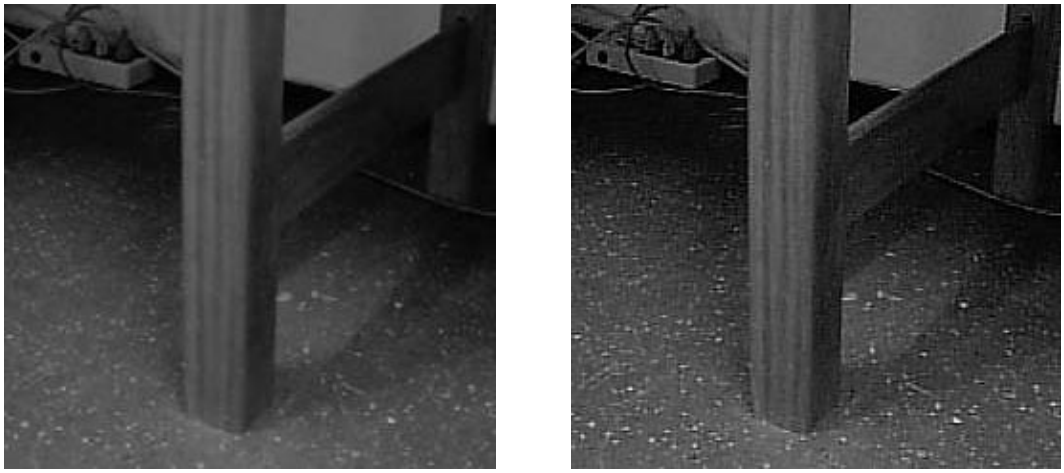
(a) (b) (c) (d) (e)

Figura 4.14 – (a) Pé-de-cadeira; (b) Após realçamento dos contornos; (c) Após aplicação do operador de *Kirsch*; (d) Após *threshold*; (e) Após aplicação da função *VerticalLines*.

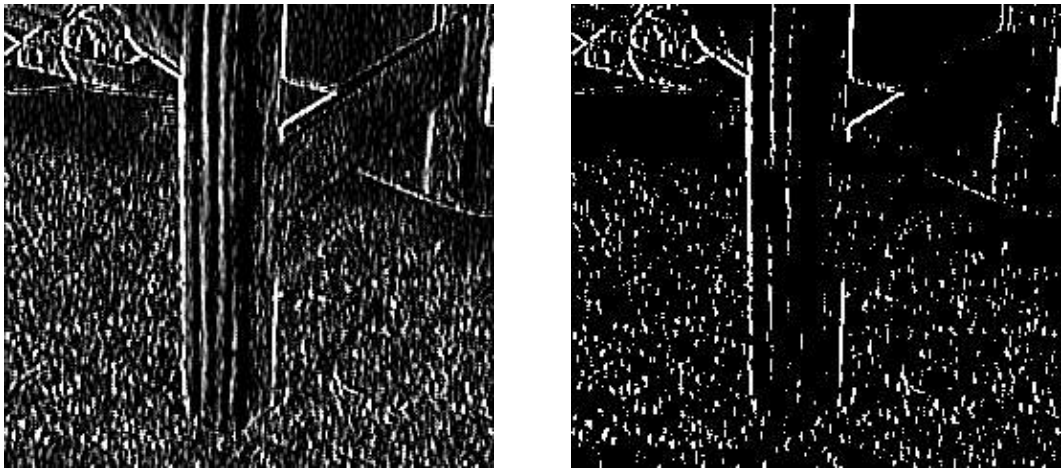


(a) (b) (c) (d) (e)

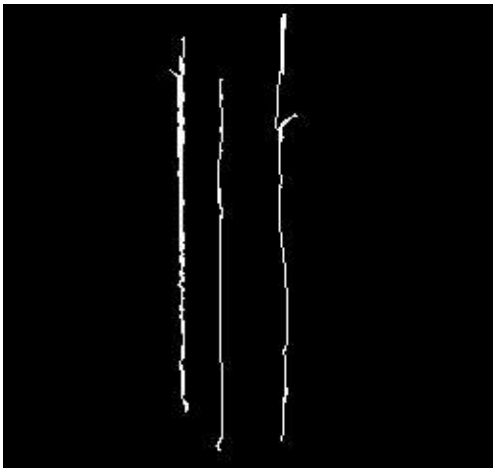
Figura 4.15 – (a) Pé-de-mesa; (b) Após realçamento dos contornos; (c) Após aplicação do operador de *Kirsch*; (d) Após *threshold*; (e) Após aplicação da função *VerticalLines*.



(a) (b)

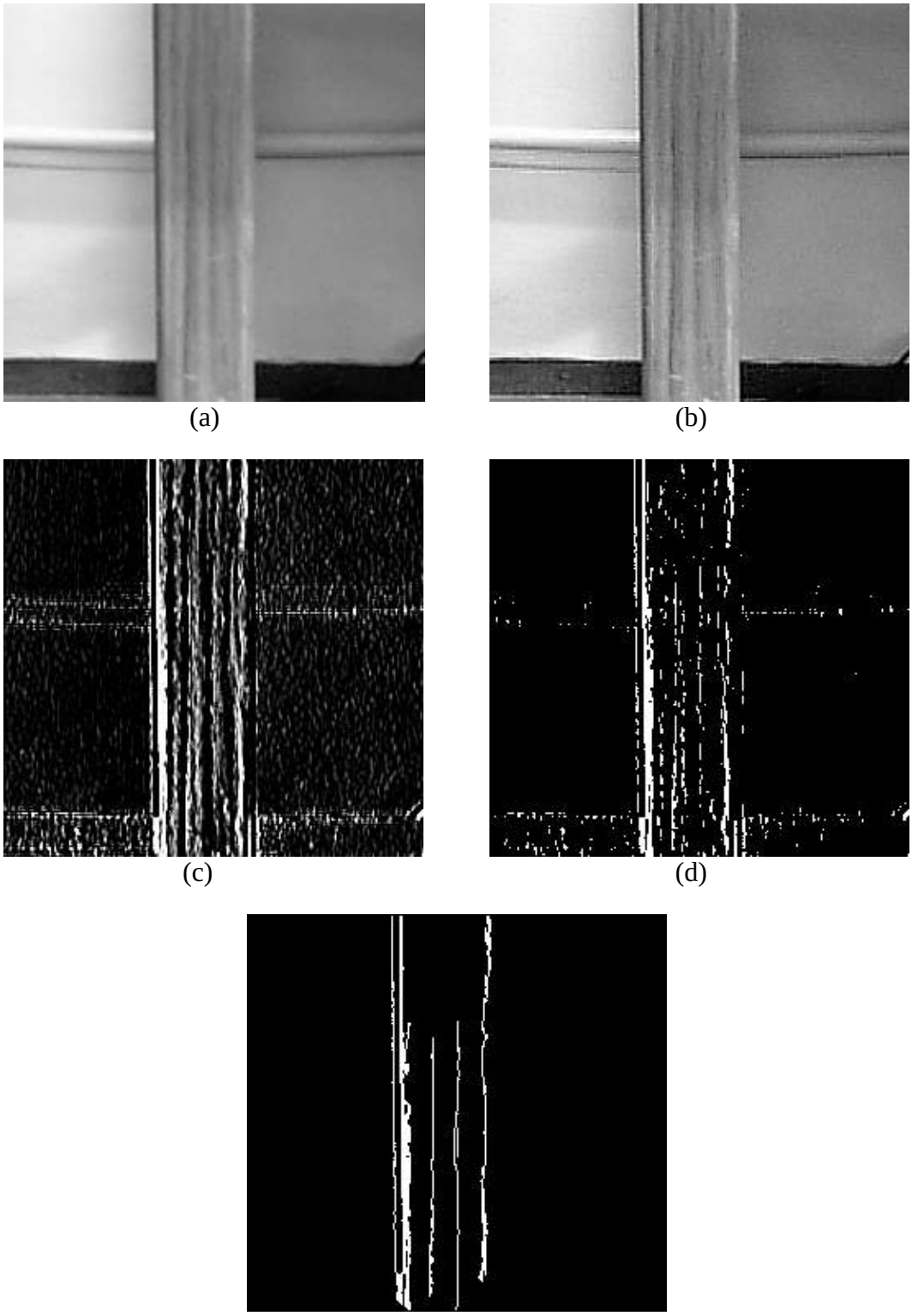


(c) (d)



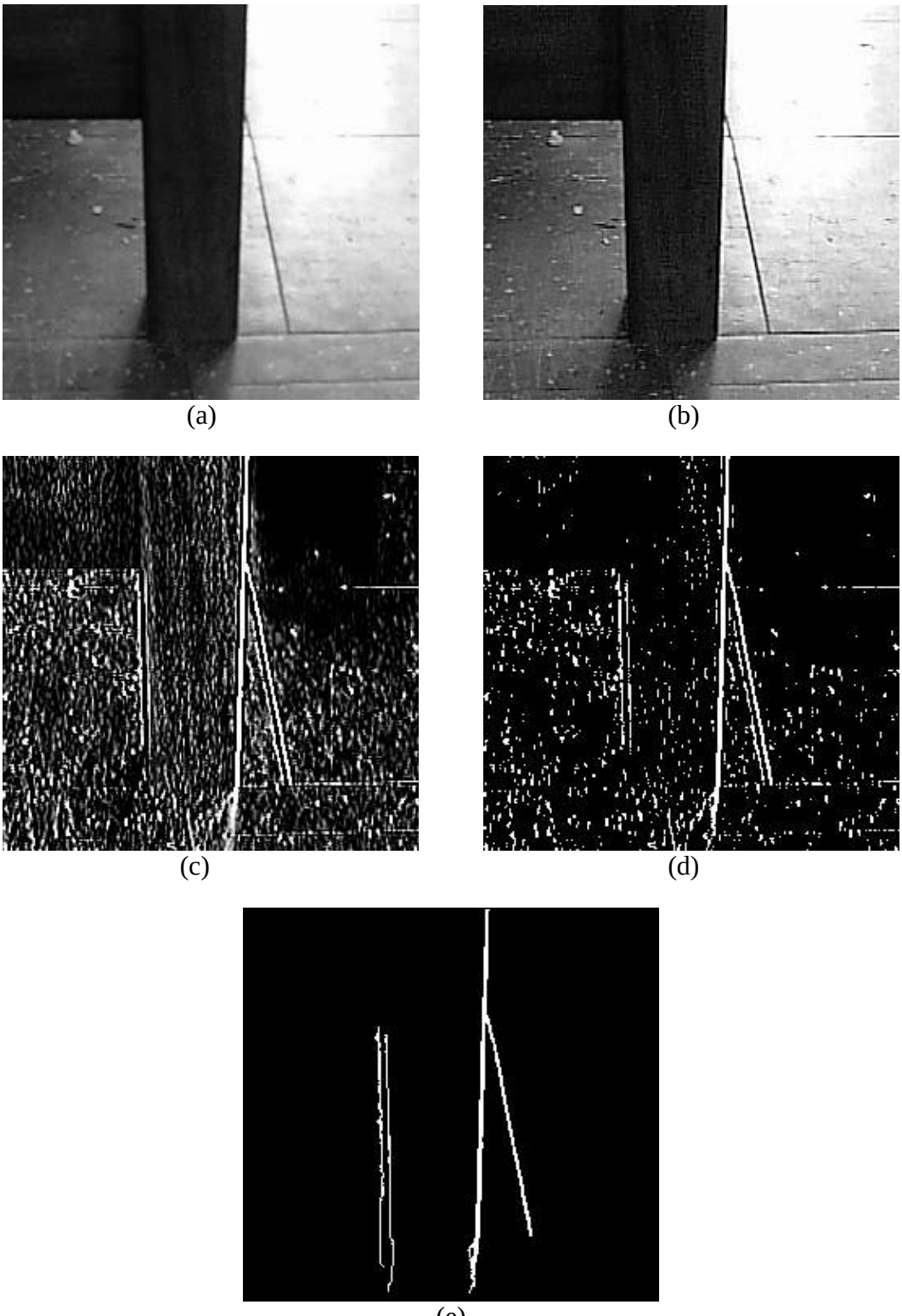
(e)

Figura 4.16 – (a) Pé-de-mesa; (b) Após realçamento dos contornos; (c) Após aplicação do operador de *Kirsch*; (d) Após *threshold*; (e) Após aplicação da função *VerticalLines*.



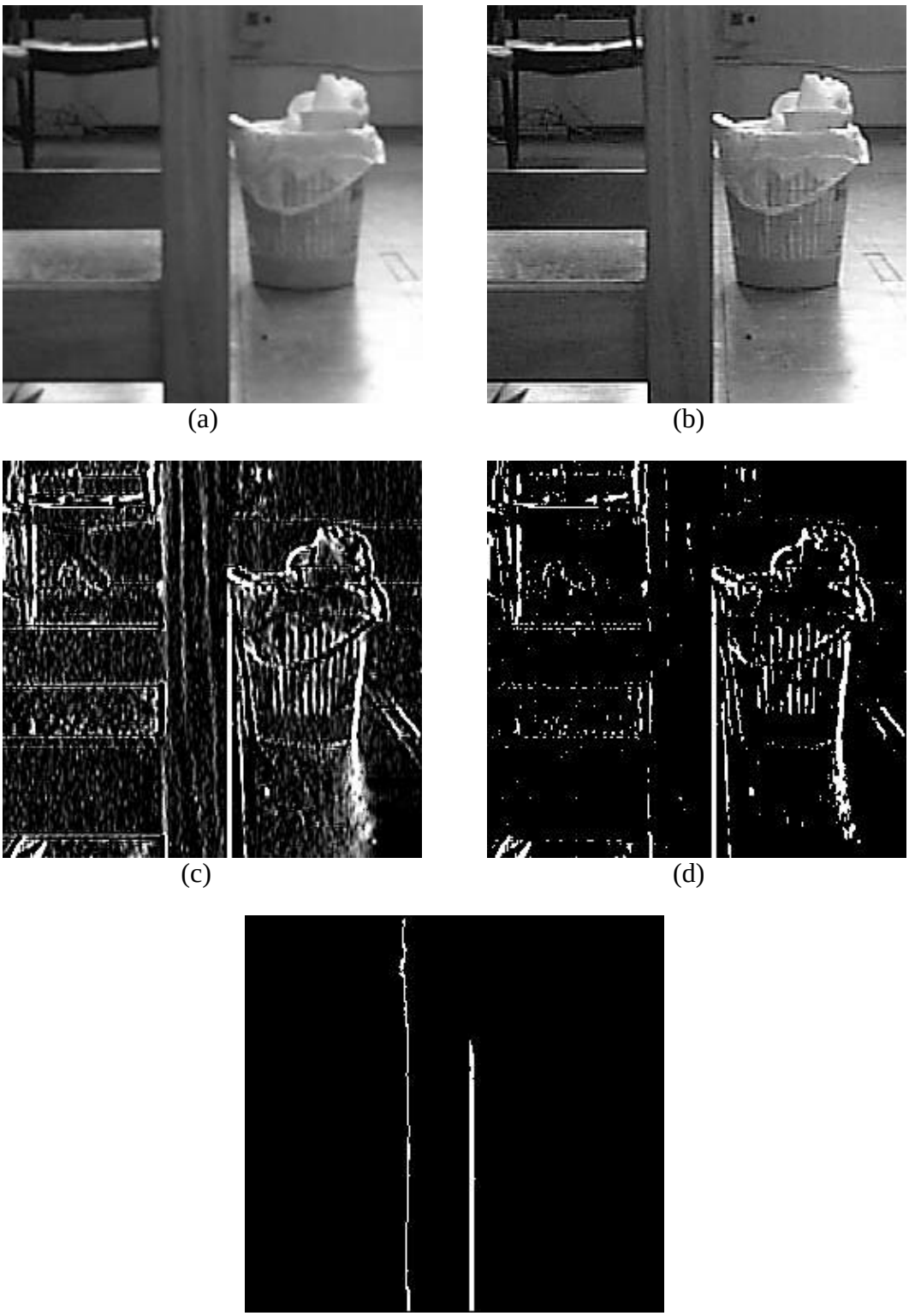
(a) (b) (c) (d) (e)

Figura 4.17 – (a) Pé-de-mesa; (b) Após realçamento dos contornos; (c) Após aplicação do operador de *Kirsch*; (d) Após *threshold*; (e) Após aplicação da função *VerticalLines*.



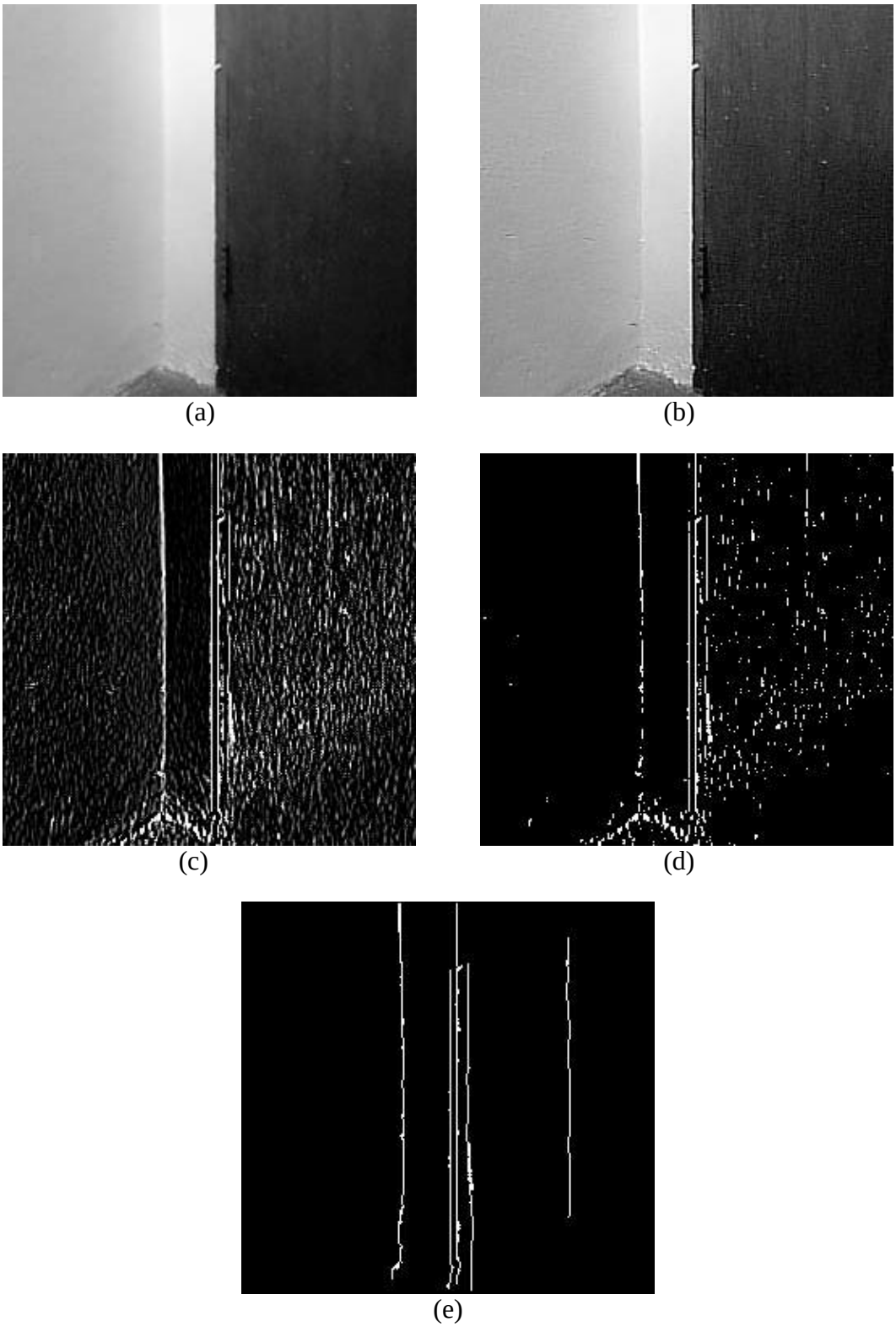
(a) (b) (c) (d) (e)

Figura 4.18 – (a) Pé-de-mesa; (b) Após realçamento dos contornos; (c) Após aplicação do operador de *Kirsch*; (d) Após *threshold*; (e) Após aplicação da função *VerticalLines*.



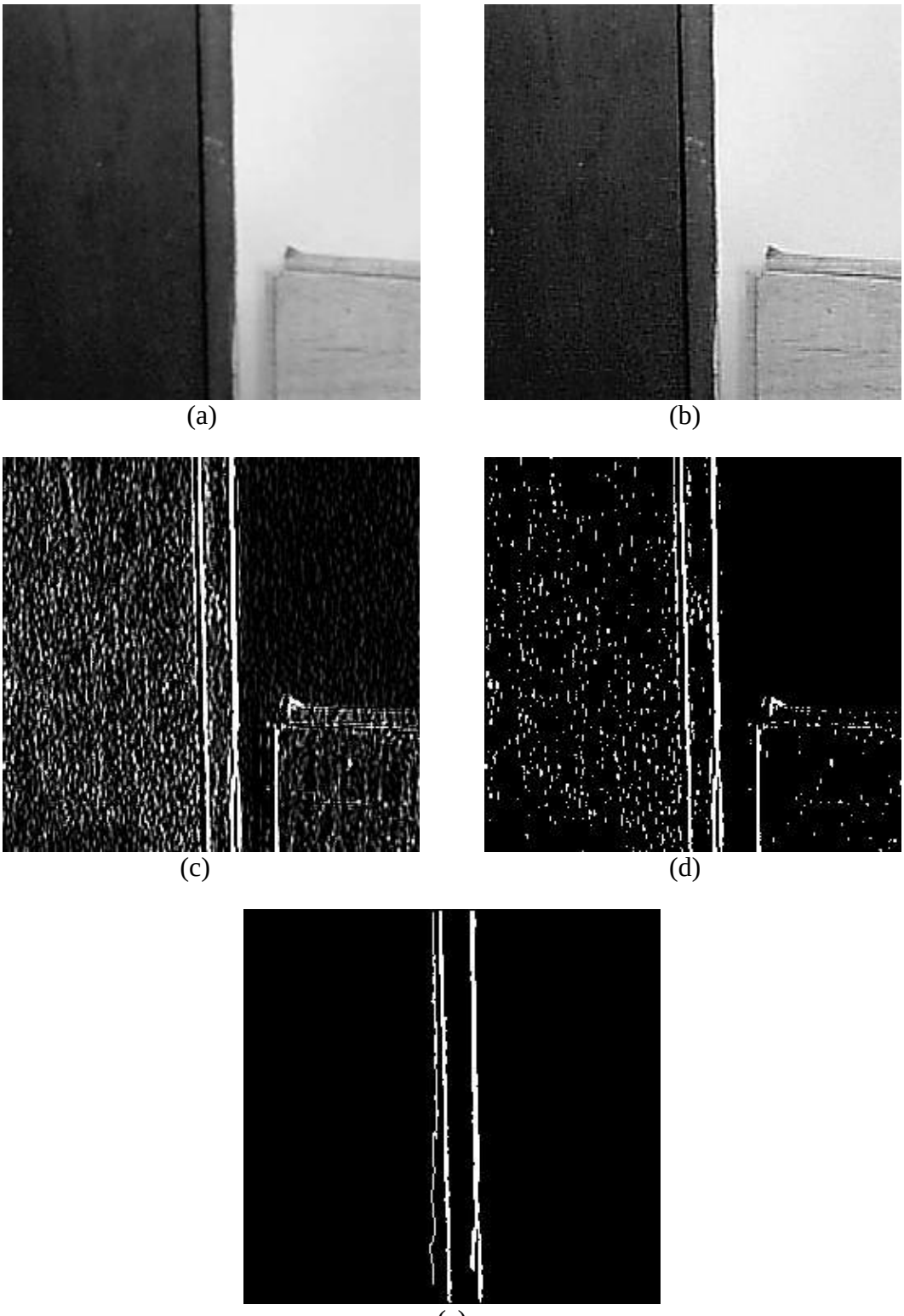
(a) (b) (c) (d) (e)

Figura 4.19 – (a) Pé-de-mesa; (b) Após realçamento dos contornos; (c) Após aplicação do operador de *Kirsch*; (d) Após *threshold*; (e) Após aplicação da função *VerticalLines*.



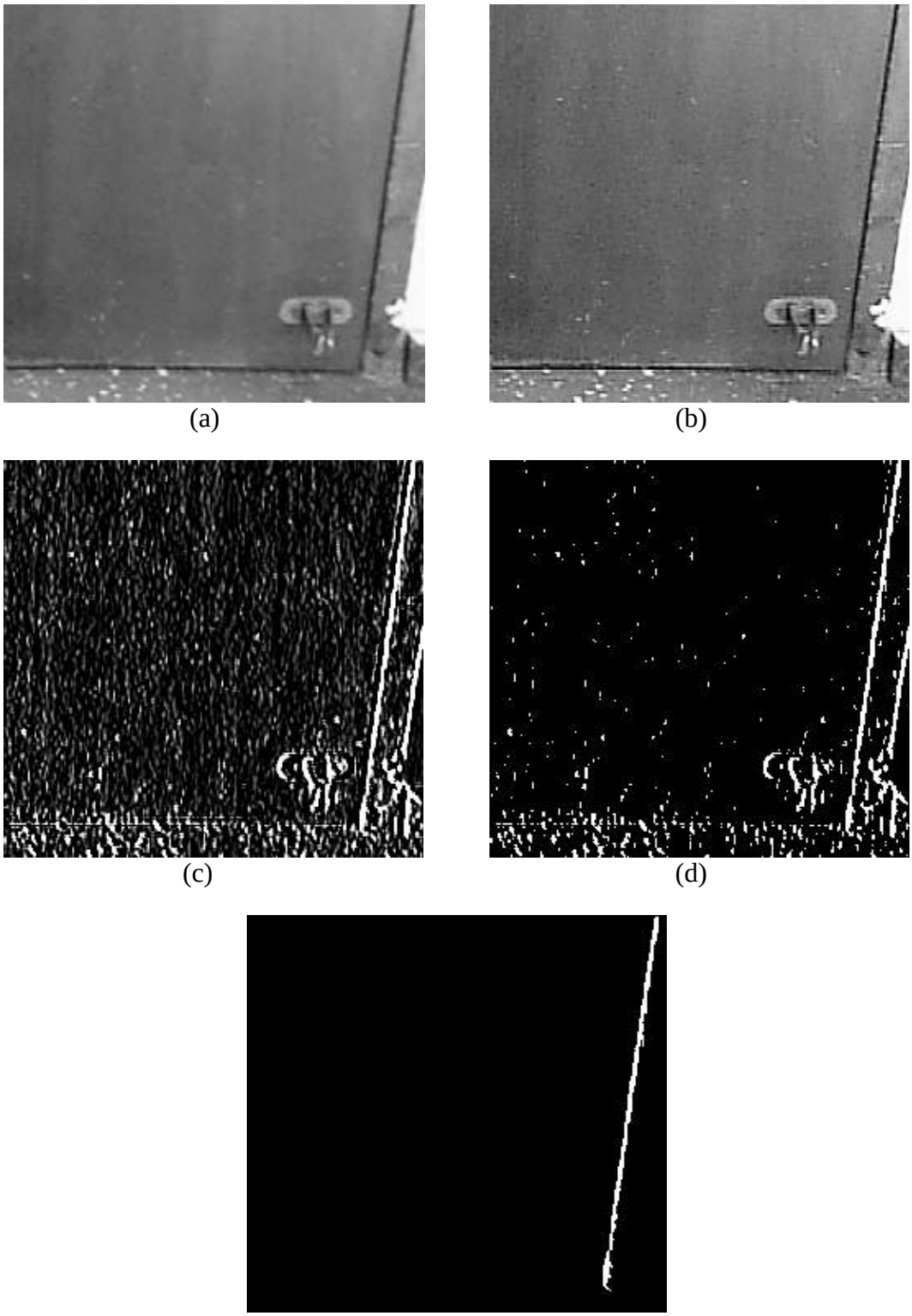
(a) (b) (c) (d) (e)

Figura 4.20 – (a) Porta; (b) Após realçamento dos contornos; (c) Após aplicação do operador de *Kirsch*; (d) Após *threshold*; (e) Após aplicação da função *VerticalLines*.



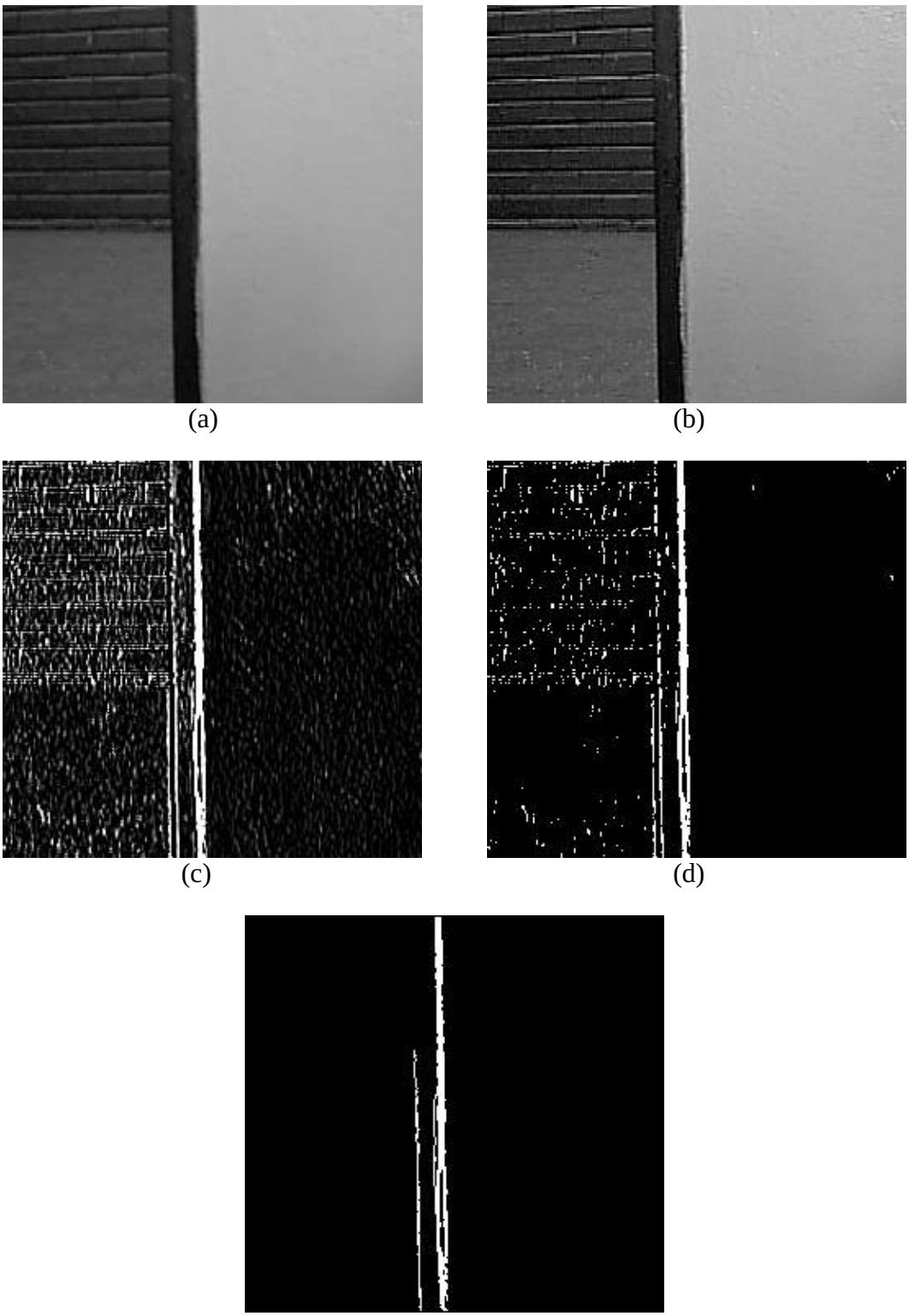
(a) (b) (c) (d) (e)

Figura 4.21 – (a) Porta; (b) Após realçamento dos contornos; (c) Após aplicação do operador de *Kirsch*; (d) Após *threshold*; (e) Após aplicação da função *VerticalLines*.



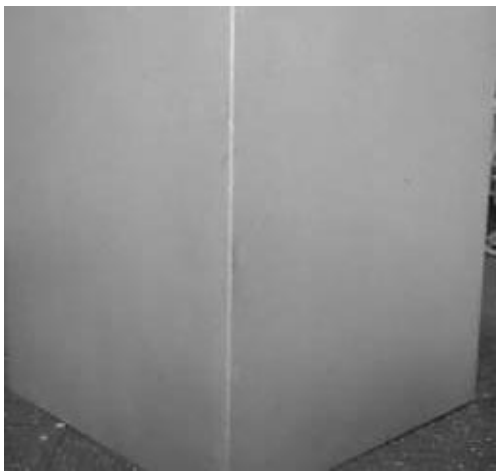
(a) (b) (c) (d) (e)

Figura 4.22 – (a) Porta; (b) Após realçamento dos contornos; (c) Após aplicação do operador de *Kirsch*; (d) Após *threshold*; (e) Após aplicação da função *VerticalLines*.

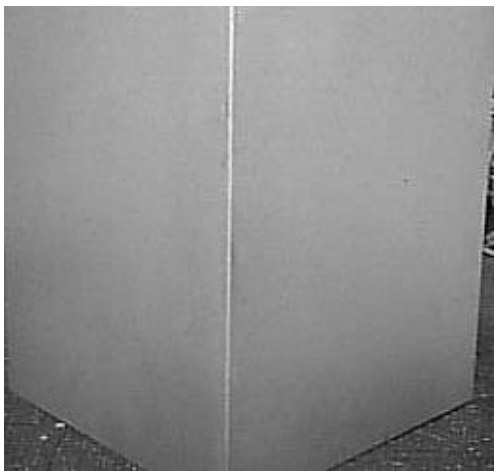


(a) (b) (c) (d) (e)

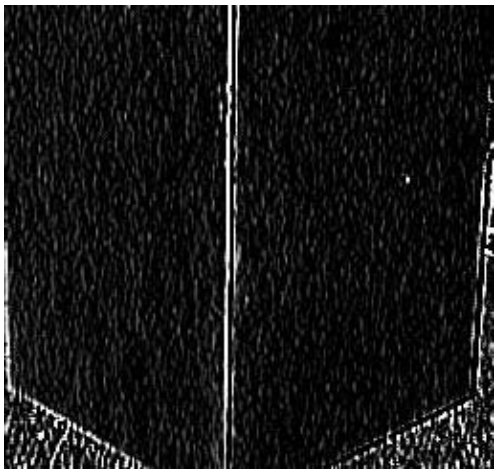
Figura 4.23 – (a) Porta; (b) Após realçamento dos contornos; (c) Após aplicação do operador de *Kirsch*; (d) Após *threshold*; (e) Após aplicação da função *VerticalLines*.



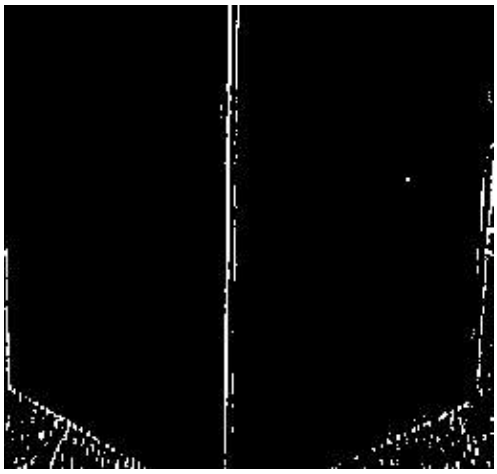
(a)



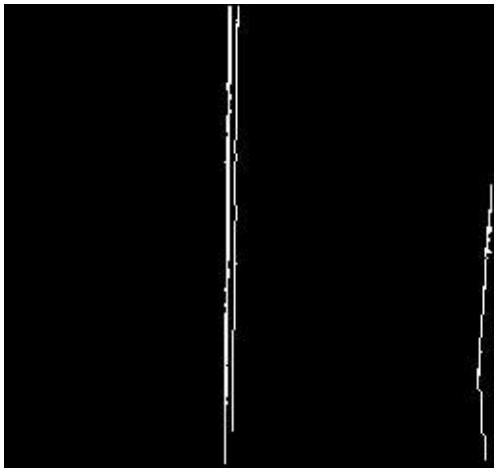
(b)



(c)



(d)

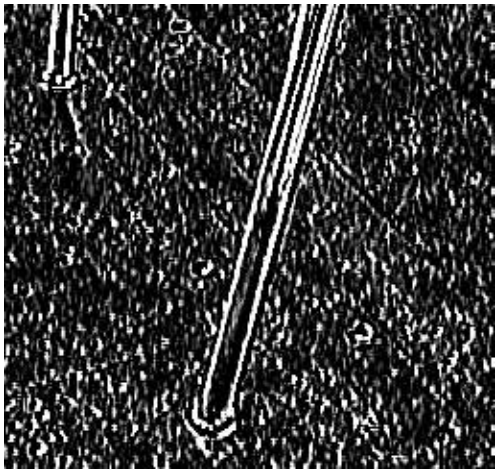


(e)

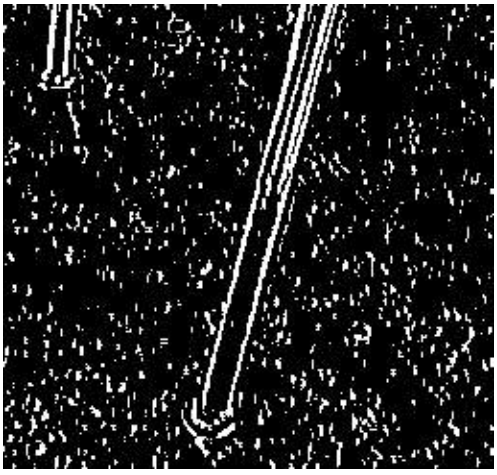
Figura 4.24 – (a) Quina; (b) Após realçamento dos contornos; (c) Após aplicação do operador de *Kirsch*; (d) Após *threshold*; (e) Após aplicação da função *VerticalLines*.

(a)

(b)



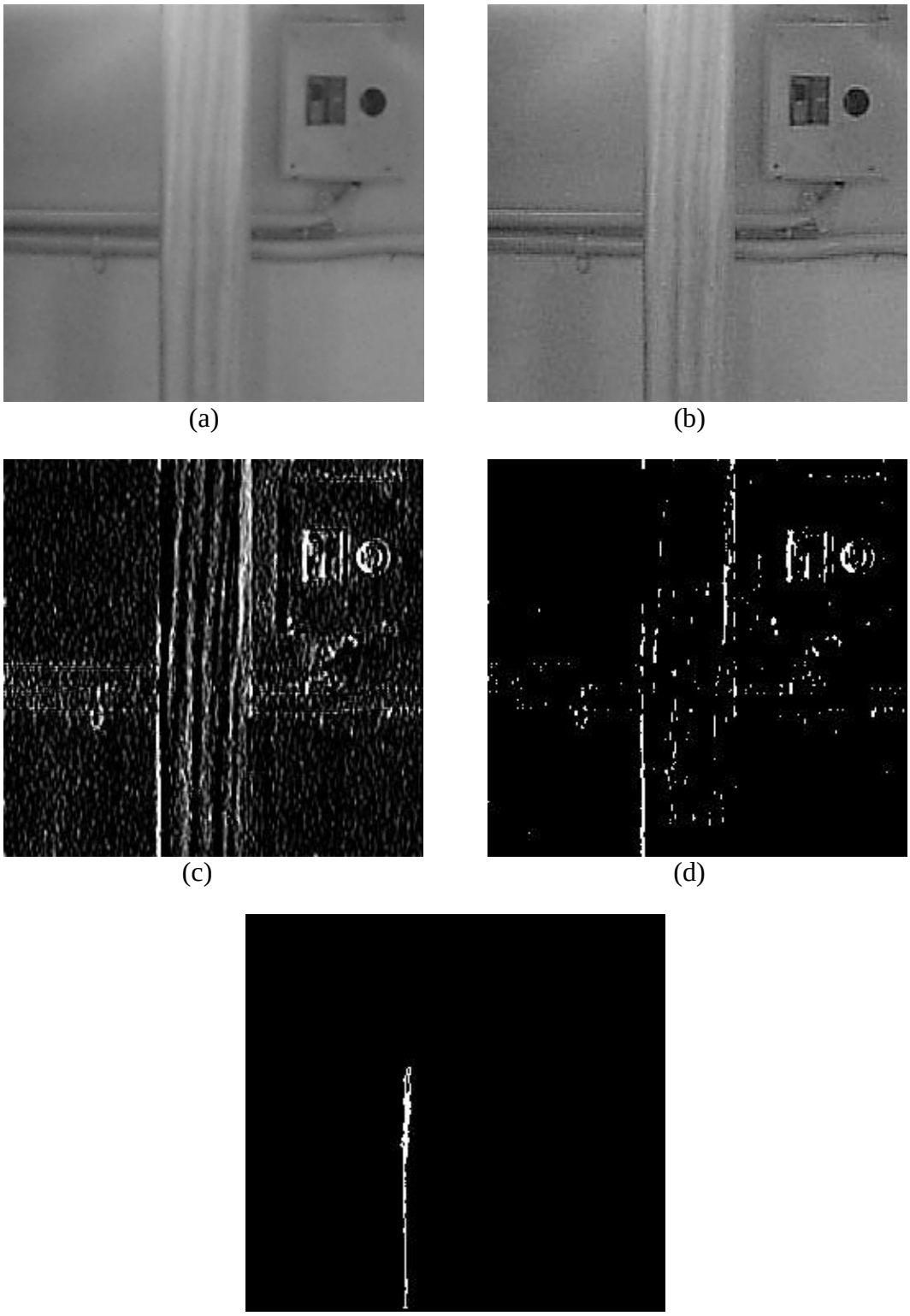
(c)



(d)

(e)

Figura 4.25 – (a) Pé-de-cadeira (reconhecido como quina); (b) Após realçamento dos contornos; (c) Após aplicação do operador de *Kirsch*; (d) Após *threshold*; (e) Após aplicação da função *VerticalLines*.



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

Figura 4.26 – (a) Pé-de-mesa (reconhecido como porta); (b) Após realçamento dos contornos; (c) Após aplicação do operador de *Kirsch*; (d) Após *threshold*; (e) Após aplicação da função *VerticalLines*.

A Figura 4.13 mostra as fases do processamento da imagem de um pé-de-cadeira. O reconhecimento foi realizado com sucesso pelo algoritmo de processamento. Pode-se observar que na imagem inicial (Figura 4.13(a)) aparecem dois pés da mesma cadeira, porém, aquele mais à esquerda não aparece na imagem final (Figura 4.13(e)). Isto deve-se ao fato de o contorno gerado por ele não ter apresentado um tamanho suficiente para ser considerado como um contorno significativo da imagem. Na Figura 4.13(e) pode-se observar a existência de um contorno relativo à reflexão da luz ambiente no pé-de-cadeira (metálico). Mesmo assim, o reconhecimento foi feito corretamente, pois o algoritmo desconsidera contornos que possuam uma distância inferior a 12 *pixels* entre si, medida na horizontal, o que foi o caso do contorno gerado pela reflexão da luz. A mesma análise é válida para a Figura 4.14.

A Figura 4.15 mostra as fases do processamento da imagem de um pé-de-mesa. O reconhecimento foi realizado com sucesso pelo algoritmo de processamento. Na Figura 4.15(e) pode-se observar alguns contornos que representam a superfície fibrosa da madeira. Estes contornos não interferiram no reconhecimento do obstáculo, pois os mesmos não apresentam a distância mínima entre si (12 *pixels*) necessária para que fossem considerados contornos significativos da imagem.

A Figura 4.16 mostra outro exemplo do processamento da imagem de um pé-de-mesa. O reconhecimento foi realizado com sucesso pelo algoritmo de processamento. Na Figura 4.16(e) pode-se observar a presença de um contorno gerado por uma mancha da madeira e dos dois contornos referentes aos limites do pé-de-mesa. Como explicado na Seção 4.4, na presença de três contornos, eles são agrupados dois a dois na tentativa de reconhecer o obstáculo. No presente exemplo, o reconhecimento baseou-se nos dois primeiros contornos (contados da esquerda para a direita na imagem), cujas características de distância relativa e tonalidade de cinza foram suficientes para identificar o pé-de-mesa. A Figura 4.17 mostra outro exemplo do processamento da imagem de um pé-de-mesa. O reconhecimento foi realizado com sucesso pelo algoritmo de processamento. A análise é a mesma feita para a Figura 4.16, porém, foram quatro o número de contornos identificados na imagem.

A Figura 4.18 mostra outro exemplo do processamento da imagem de um pé-de-mesa. O reconhecimento foi realizado com sucesso pelo algoritmo de processamento. Na Figura 4.18(e) pode-se observar uma ramificação do contorno mais à direita na imagem. Essa ramificação não foi considerada pelo algoritmo de reconhecimento por estar conectada a um contorno já identificado anteriormente. A Figura 4.19 mostra outro exemplo do processamento da imagem de um pé-de-mesa. O reconhecimento foi realizado com sucesso pelo algoritmo de processamento. Neste caso, na imagem final (Figura 4.19(e)) só restaram os contornos relativos aos limites do pé-de-mesa.

A Figura 4.20 mostra um exemplo do processamento da imagem de uma porta. O reconhecimento foi realizado com sucesso pelo algoritmo de processamento, apesar de ter surgido um falso contorno, localizado mais à direita na imagem final (Figura 4.20(e)), resultante da conexão de pequenos pontos gerados pela superfície fibrosa da madeira.

A Figura 4.21 mostra outro exemplo do processamento da imagem de uma porta. O reconhecimento foi realizado com sucesso pelo algoritmo de processamento. Na etapa final do processamento (Figura 4.21(e)) pode-se observar os contornos referentes ao marco da porta. A mesma análise é válida para as Figuras 4.22 e 4.23.

A Figura 4.24 mostra um exemplo do processamento da imagem de uma quina. O reconhecimento foi realizado com sucesso pelo algoritmo de processamento. Na etapa final do processamento (Figura 4.24(e)) pode-se observar a presença de dois contornos, que devido às características de distância entre si e tonalidade de cinza, não foram confundidos com um pé-de-mesa ou um pé-de-cadeira.

As Figuras 4.25 e 4.26 mostram exemplos de imagens onde o reconhecimento dos obstáculos não foi realizado corretamente. A Figura 4.25 mostra a imagem de um pé-de-cadeira metálico. Nesta imagem foram gerados falsos contornos, provenientes do outro pé da mesma cadeira e do reflexo da luz no pé que aparece em primeiro plano. Como a distância entre os extremos inferiores desses contornos é maior que 70 *pixels*, o objeto foi reconhecido erroneamente como uma quina (Seção 4.4). Por outro lado, a Figura 4.26

mostra a imagem de um pé-de mesa. A diferença entre as tonalidades de cinza do pé-de-mesa e da parede foram suficientes apenas para gerar um contorno. Desta maneira, e levando-se em consideração as tonalidades de cinza das regiões à direita e à esquerda do contorno, o obstáculo foi erroneamente reconhecido como uma porta.

Alguns desses resultados podem ser encontrados também em [30],[31],[32],[33].

Capítulo 5

Conclusão

Neste trabalho, foi desenvolvido um sistema de reconhecimento de obstáculos para ser incorporado a um robô móvel cujo controle seja feito através de uma arquitetura baseada em comportamentos. A estrutura implementada contempla a detecção do obstáculo por um anel de sensores ultra-sônicos, o direcionamento de uma câmara na direção detectada, a captura de uma imagem e o seu processamento com vistas ao reconhecimento do obstáculo. A base do algoritmo de reconhecimento está na suposição de que os maiores contornos verticais presentes na imagem pertencem ao obstáculo detectado pelos sensores ultra-sônicos, visto que é este o objeto mais próximo do robô.

Tais resultados foram alcançados utilizando-se um sistema simples e de baixo custo (uma câmara CCD e sensores ultra-sônicos), assim como um reduzido esforço computacional, pela simplicidade dos algoritmos e a baixa taxa de captura das imagens (a câmara somente é acionada depois que os sensores ultra-sônicos detectam a presença de um obstáculo na direção de movimentação do robô, a uma distância pré-definida). Além disso, não houve qualquer estruturação do ambiente. Este fato, em alguns casos, gerou dificuldades para o correto reconhecimento do obstáculo, principalmente quando o chão do laboratório (fundo azul coberto por pequenas manchas brancas) aparecia na

imagem. Nestes casos, cada mancha do chão representava uma potencial fonte de geração de falsos contornos (Figura 4.24). Porém, os experimentos realizados mostraram que, na grande maioria dos casos, o algoritmo proposto foi capaz de reconhecer corretamente os obstáculos presentes na trajetória do robô. Mesmo em casos onde na imagem apareciam outros objetos compondo o fundo da cena, o algoritmo obteve sucesso na tarefa de reconhecimento.

A sensibilidade do algoritmo pode ser modificada alterando-se a constante que define o tamanho mínimo que o algoritmo considera para definir quando um contorno é significativo ou não. Desse modo, se a fase final do processamento estiver apresentando muitos contornos além daqueles necessários para identificar o obstáculo, pode-se aumentar o valor da referida constante para eliminar os contornos menores. Esta possibilidade é interessante porque permite que se adeqüe o algoritmo de reconhecimento para a operação em diferentes ambientes. Outra maneira de modificar a sensibilidade do algoritmo é mudar o nível de *threshold*. Quanto mais elevado este nível, menos contornos estarão presentes na imagem binária final gerada pelo processo de segmentação (ver Capítulo 4).

Certamente, um melhor desempenho do sistema poderia ser conseguido através de alguma estruturação do ambiente para a aplicação em si, ou seja, pintar o chão com uma cor homogênea e pintar paredes e portas com cores que permitissem acentuar o contraste ajudaria a melhorar o desempenho do algoritmo de reconhecimento dos obstáculos. Todavia, esta não é a proposta deste trabalho, que prevê a operação em ambientes apenas semi-estruturados.

Como sugestões de caminhos para dar continuidade ao presente trabalho, pode-se citar a sua implementação e teste à bordo do robô (os experimentos aqui realizados usaram imagens obtidas e processadas "*off-line*"), a comparação do seu desempenho com o de outras estratégias de navegação, tais como o método da impedância, outros métodos visuais de servo-controle, etc., ou ainda a implementação de comportamentos adequados, de forma a especializar ainda mais o sistema de controle, sempre em função dos obstáculos detectados. Um exemplo que se poderia citar, neste caso, seria a

implementação de um comportamento que permitisse ao robô vaguear até encontrar a porta aberta, e então sair do laboratório.

Bibliografia

[1] Beckerman, M., D. L. Barnett, M. Dickens and C. R. Weisbin, “Performance of Multiple Tasks by an Autonomous Robot Using Visual and Ultrasound Sensing”, *Proc. of the 3º International Symposium on Robotics and Manufacturing: Research, Education and Applications (ISRAM ’90)*, pp. 389-395, Canadá, 1990.

[2] Kriegman, E. T., Binford, T. O., “Stereo Vision and Navigation in Buildings for Mobile Robots”, *IEEE Transactions on Robotics and Automation*, Vol. 5, N° 6, pp. 792-803, 1989.

[3] Bastos, T. F., H. Schneebeli e V. Dynnikov, “Tópicos Avançados en el Control de Robots: Robots Móviles”, *I Curso Iberoamericano de Automática*, Santa Cruz de La Sierra, Bolívia, dezembro de 1995.

[4] Freire, E. O., “Desenvolvimento de um Sistema de Sensoriamento Ultra-Sônico para um Robô Móvel com Controle Baseado em Agentes”, *Dissertação de Mestrado*, UFES, Vitória - ES, 1997.

[5] Bastos, T. F., “Seguimiento y Análisis de Entornos de Soldadura por Arco Automatizada Mediante Ultrasonidos”, *Tesis Doctoral*, Universidad Complutense de Madrid, Espanha, 1994.

[6] Marr, D., *Vision - A Computational Investigation into the Human Representation and Processing of Visual Information*, W. H. Freeman and Company, San Francisco, 1982.

[7] Vicente, L. A., “Estrategias de correspondencia jerárquica y métodos directos de autocalibración para un sistema estereoscópico binocular”, *Tesis Doctoral*, Universidad Complutense de Madrid, Espanha, 1996.

[8] Iocchi, L., Konolige, K., *A Multiresolution Stereo Vision System for Mobile Robots*.

[9] Faugeras, O., *Three-Dimensional Computer Vision*, MIT Press, 1996.

[10] Brooks, R. A., *Achieving Artificial Intelligence Through Building Robots*, A. I. Memo 899, MIT, Massachussets, 1986.

[11] Gonzalez, R. C., Woods, R. E., *Digital Image Processing*, Addison-Wesley Publishing Company, 1993.

[12] Myler, H. R., Weeks, A. R., *Computer Imaging Recipes in C*, Prentice Hall, 1993.

[13] Sonka, M., Hlavac, V., Boyle, R., *Image Processing, Analysis and Machine Vision*, Chapman & Hall, 1993.

[14] Bernardino, A., Santos-Victor, J., "Correlation Based Vergence Control using Log-polar Images", International Symposium on Intelligent Robotic Systems - SIRS96, Lisbon, Portugal, 1996.

[15] Santos-Victor, J., Silva, C., "Egomotion Estimation Using Log-Polar Images", International Conference of Computer Vision ICCV, Bombay, India, 1998

[16] Fu, K. S., Gonzalez, R. C., Lee, C. S. G., *Robotics: Control, Sensing, Vision, and Intelligence*, McGraw-Hill, 1987.

[17] Luo, R. C., Kay, M. G., "Multisensor Integration and Fusion in Intelligent Systems", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-19, N° 5, pp. 901-931, 1989.

[18] Lorigo, L. M., "Visually-Guided Obstacle Avoidance in Unstructured Environments", Master Thesis, MIT, 1996.

[19] Stone, H. W., “Mars Pathfinder Microrover: A Low-Cost, Low-Power Spacecraft”, *Proceedings of the 1996 AIAA Forum on Advanced Developments in Space Robotics*, Madison, WI, 1996.

[20] Matijevic, J., “Mars Pathfinder Microrover - Implementing a Low Cost Planetary Mission Experiment”, *Proceedings of the Second IAA International Conference on Low-Cost Planetary Missions*, John Hopkins University Applied Physics Laboratory, Laurel, Maryland, 1996.

[21] Xavier, J. E. M., “Uma Estrutura para a Construção de Sistemas de Controle Baseados em Agentes para Robôs Móveis”, *Dissertação de Mestrado*, UFES, Vitória - ES, 1996.

[22] Jones, J. L., Flynn, A. M., *Mobile Robots: from Inspiration to Implementation*, capítulo 9, pp. 243-269, Addison-Wesley Publishing Company, 1993.

[23] Brooks, R. A. “A Robust Layered Control System for a Mobile Robot”, *IEEE Journal of Robotics and Automation*, Vol. 2, No. 1, pp. 14-23, March 1986.

[24] Brooks, R. A., “A Hardware Retargetable Distributed Layered Architecture for Mobile Robot Control”, *Proceedings of 1987 International Conference on Robotics and Automation*, Raleigh, NC, pp. 106-110, March 1987.

[25] Brooks, R. A., “A Robot that Walks: Emergent Behavior from a Carefully Evolved Network”, *IEEE International Conference on Robotics and Automation*, Scottsdale, AZ, pp. 292-296, May 1989.

[26] Fontán, M. S., “Planificación Basada en Percepción Activa para la Navegación de un Robot Móvil”, *Tesis Doctoral*, Universidad Complutense de Madrid, 1996.

[27] Xavier, J. E. M., Schneebeli, H., “A Framework Based on Concurrent Object-oriented Programming for Building Behavior-based Control Systems for Mobile Robots”, *Journal of the Brazilian Computer Society*, Vol. 4, No. 3, 1998.

[28] Schneebeli, H., “Die steuerung von mehrfinger-greifersystemen”, Dr. rer-nat Thesis, Karlsruhe University, 1992.

[29] Yonezawa, A., Modeling and programming in an object-oriented concurrent language. In: Yonezawa, A., Tokoro, M. Object Oriented Concurrent Programming. Massachussetts: MIT Press, pp. 55-90, 1988.

[30] Freire, E. O., Bastos, T. F., Freitas, R. A. C., Schneebeli, H. A., Sarcinelli, M., “Um Sistema de Sensoriamento Externo para Robôs Móveis com Controle Baseado em Agentes”, Anais do XII Congresso Brasileiro de Automática (XII CBA), Uberlândia/MG, setembro de 1998, vol. 2, pp. 581-586.

[31] Freire, E. O., Freitas, R. A. C., Bastos, T. F., Schneebeli, H. A., Sarcinelli, M., "Object Recognition for an Agent-Based Controlled Mobile Robot", Proceedings of the 5th IFAC Workshop on Intelligent Manufacturing Systems - IMS'98, Gramado/RS, Brasil, novembro de 1998, vol. 1, pp. 323-328.

[32] Bastos, T. F., Freitas, R. A. C., Sarcinelli, M., Schneebeli, H. A., "An Agent-Based Structure for Mobile Robots Using Vision and Ultrasonic Sensors", Proceedings of the 1998 IEEE International Symposium on Circuits and Systems (ISCAS'98), Monterey, California, USA, junho de 1998, volume VI, pp. 602-605.

[33] Freire, E. O., Bastos, T. F., Freitas, R. A. C., Sarcinelli, M., "Development of an External Sensing System for an Agent-Based Controlled Mobile Robot", Proceedings of the 3rd IFAC Symposium on Intelligent Autonomous Vehicles (IAV'98), Madrid, Espanha, março de 1998, pp. 243-248.