UNIVERSIDADE DO VALE DO RIO DOS SINOS

EDUARDO MAESTRI RIGHES

Processamento de Imagens para Navegação de Robôs Autônomos

Monografia apresentada como requisito parcial
para a obtenção do grau de
Bacharel em Ciência da Computação

Prof. Dr. Fernando Santos Osório
Orientador

São Leopoldo, Junho de 2004

AGRADECIMENTOS

Primeiramente, gostaria de agrader ao Prof. Dr. Fernando Santos Osório pela dedicação na orientação e ajuda na elaboração deste trabalho. Além do bom orientador que é, tornou-se um amigo. Meu muito obrigado.

Também gostaria de agradecer aos meu pais, Eloi e Neusa, pela ajuda que puderam me dar ao longo caminho que percorri durante a minha vida.

Um agradecimento especial a minha noiva, Milene, pela ajuda, companherismo, compreensão e afeto nestes últimos anos da minha vida. Espero que todos os bons momentos que passamos juntos continuem por muitos e muitos anos.

Agradeço aos meus colegas de curso que, ao final desta longa caminhada, tornaram-se bons amigos, proporcionando momentos memoráveis que eu me lembrarei com carinho para sempre.

Agradeço também ao Marcelo Willadino e a T&T por compreender este momento da minha vida, permitindo que eu me ausentasse temporariamente para a conclusão deste trabalho.

Por último, mas não menos importante, agradeço a minha avó, dona Rosa, por toda ajuda que me deu durante todo curso. Sem esta ajuda, tudo seria mais difícil. Muito, mas muito obrigado mesmo.

Mais uma vez, agradeço a todos que me ajudaram a concluir este trabalho.

RESUMO

Atualmente a robótica autônoma móvel vem assumindo um importante papel no desenvolvimento de pesquisas em Inteligência Artificial. Nesta área existem tópicos que são de fundamental importância, a saber: manutenção da localização do robô, planejamento de trajetórias e navegação. Um dos dispositivos sensores que vem sendo cada vez mais usados em robótica móvel são os sistemas de aquisição de imagens (câmeras de vídeo). Este trabalho se propõe a apresentar uma nova técnica para navegação visual baseada na correlação entre imagens coloridas. São discutidos os resultados de experimentos realizados com imagens reais e a sua aplicação no controle de um robô móvel.

Palavras-chave: Robótica, processamento de imagens, visão computacional.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AGV	Automated Guided Vehicle
BI	Banco de Imagens
CCD	Charge Coupled Device
GPS	Global Positioning System
ICR	Imagem Capturada pelo Robô
IR	Imagem de Referência
HSI	Hue-Saturation-Intensity
HSV	Hue-Saturation-Value
NCC	Normalized Cross-Correlation
PI	Processamento de Imagens
RGB	Red-Green-Blue
SMPA	Sense-Model-Plan-Act

1 INTRODUÇÃO

Atualmente a robótica vem assumindo um importante papel junto a sociedade moderna, onde podemos vislumbrar em um futuro próximo o crescimento da importância de uma área particular da robótica: os robôs móveis autônomos (Dudek and Jenkin, 2000). Estes dispositivos móveis, a fim de se tornar autônomos, precisam ser dotados de sensores capazes de "sentir" e "perceber" o estado e as informações provenientes do ambiente no qual estão inseridos. Esta percepção é essencial para que possam planejar e agir levando em consideração o contexto no qual se encontram inseridos.

Dentro das diversas áreas de pesquisa em robótica autônoma, existem tópicos que são de fundamental importância, a saber: manutenção da localização do robô, planejamento de trajetórias e navegação (Heinen, 2002). Em relação a estas áreas, destaca-se a dificuldade em se obter informações provenientes dos sensores que permitam uma leitura precisa de dados relativos a posição atual do robô e sobre a rota a ser realizada por este robô. Um dos dispositivos sensores que vem sendo cada vez mais usados em robótica móvel, dado seu baixo custo e a riqueza das informações que ele fornece, são os sistemas de aquisição de imagens (câmeras de vídeo).

Este trabalho se propõe a utilizar uma técnica similar à apresentada em (Jones et al., 1997), que utiliza *normalized cross-correlation* (NCC) (Martin and Crowley, 1995) para fazer o *matching* entre duas imagens de modo a obter uma referência relativa à posição atual do robô que possui uma câmera acoplada. Entretanto, em (Jones et al., 1997) somente é abordada a correlação baseada em imagens monocromáticas do tipo *grayscale*.

Imagens coloridas proporcionam um volume de informação e uma riqueza de detalhes maior do que imagens em *grayscale* (Gonzalez and Woods, 2002). Quando uma imagem é capturada diretamente ou convertida para *grayscale*, toda essa riqueza de informações é perdida, pois cores diferentes podem ser mapeadas para um mesmo tom de cinza.

Deste modo, buscamos adaptar ou criar novas técnicas de processamento de imagens para aproveitar a informação que as cores podem proporcionar. As técnicas descritas neste trabalho fazem uma adaptação do modo de uso da NCC para imagens coloridas.

O objetivo deste trabalho é estudar as técnicas de processamento de imagens disponíveis atualmente para se desenvolver um sistema de navegação visual (Matsumoto et al., 1996; Jones et al., 1997) para controle de robôs móveis.

Este trabalho está organizado da seguinte forma: no capítulo 2 serão descritos os principais conceitos envolvidos neste trabalho, a fim de proporcionar um embasamento necessário para a sua compreensão, além de uma visão geral dos trabalhos

relacionados à navegação visual disponíveis na literatura. No capítulo 3 serão vistas as abordagens de como se utilizar o *matching* por correlação empregadas neste trabalho, para buscar o objetivo de se denvolver um sistema de navegação visual. No capítulo 4, serão vistos os resultados dos experimentos realizados durante a elaboração deste trabalho, além do resultado de uma simulação de navegação visual utilizando imagens coloridas. Por fim, no capítulo 5 serão vistas as conclusões obtidas a partir dos resultados deste trabalho e uma pequena discussão sobre trabalhos futuros relacionados à navegação visual.

2 ESTADO DA ARTE

Neste capítulo, serão vistos os conceitos relacionados à robótica móvel autônoma, relativos à navegação, passando pelos principais tipos de sensores e sistemas de controle utilizados atualmente. Também serão vistos os conceitos relativos aos diferentes tipos de *matching* em imagens e sobre o espaço de cores RGB. Por último, uma revisão dos trabalhos relacionados à navegação visual, disponíveis na literatura, que foram estudados.

2.1 Robótica Móvel

Atualmente a robótica móvel (Dudek and Jenkin, 2000) é empregada em diversos campos, desde a automação industrial até a exploração espacial. Em alguns casos, desejamos que o robô execute alguma tarefa autonomamente e, para que isso seja obtido, devemos de alguma forma informar a ele como é o ambiente em que está inserido. A complexidade desta tarefa diminui em ambientes controlados, ou seja, estáticos ou com muito poucas alterações. Em ambientes dinâmicos, onde móveis ou pessoas podem mudar de posição, esta tarefa pode se tornar muito complexa. Na robótica móvel autônoma, a navegação do robô (Dudek and Jenkin, 2000; Matsumoto et al., 1996; Heinen, 2002) é um conceito muito importante, pois através dela o robô deverá planejar sua trajetória em um dado ambiente para que possa se deslocar de um ponto a outro e garantir que a execução desta tarefa seja feita de uma forma segura.

Para que o robô possa planejar e executar sua navegação, ele precisa receber informações sobre o ambiente onde está inserido. Estas informações são recebidas através de sensores (Dudek and Jenkin, 2000; Heinen, 2002) instalados no corpo do robô. Os principais tipos de sensores utilizados serão mostrados na Seção 2.1.1.

Outro aspecto importante na robótica móvel autônoma, é a manutenção da posição atual do robô. Para que o robô possa planejar a trajetória que deverá seguir, ele precisa ser capaz de responder a pergunta "onde estou?". A manutenção da posição pode ser dada de duas formas. Na primeira, o robô deve estimar sobre sua localização atual sobre alguma representação global do espaço onde está (*strong localization*). Na segunda, o robô deve ser capaz de determinar se já esteve em um determinado local (*weak localization*). Baseado na *weak localization*, o robô pode contruir mapas para que depois possa manter sua posição de uma forma global, usando *strong localization* (Dudek and Jenkin, 2000).

Se um robô possuir sensores suficientes, utilizar um mapa e fazer uso de odometria e *dead reckoning*, ele deverá ser capaz de estimar sua posição. Possivelmente, esta é a forma mais simples de se manter a posição do robô. Por *dead reckoning*

ou odometria entendemos que robô estima quanto se deslocou observando apenas parâmetros internos, sem fazer consultas ao mundo exterior.

2.1.1 Sensores

Na robótica, são empregados diversos tipos de sensores (Dudek and Jenkin, 2000; Heinen, 2000) para dar ao robô um indicativo do ambiente onde está, sua distância em relação a outros objetos, o quanto se deslocou, etc. Os sensores podem ser ativos ou passivos. Sensores ativos são aqueles que emitem energia ou modificam o ambiente sendo observado. Por outro lado, os sensores passivos são aqueles que apenas fazem a leitura de energia para fazer observações sobre o ambiente onde estão operando. Os sensores mais empregados são descritos a seguir.

2.1.1.1 Sensores de choque (bumpers)

Basicamente, os *bumpers* ou sensores de choque/colisão são usados para determinar se um robô tocou em algum obstáculo. Os *bumpers* mais simples funcionam com pequenas chaves com dois estados: aberto e fechado.

2.1.1.2 Sensores infravermelhos

Os sensores infravermelhos (IR – *Infraread*) são rápidos e baratos. Utilizados para medir a proximidade entre um robô e um objeto. Esta proximidade é medida através da intensidade do sinal de retorno emitida pelo sensor e de suposições sobre o tipo de superfície que refletiu o sinal. Para não sofrer influência de outras fontes de IR, o sinal pode ser codificado e, durante a leitura de sinais, ignorar aqueles com tamanho de onda diferentes do sinal codificado. O sensor IR é um sensor ativo.

2.1.1.3 Sensores do tipo sonar

O sensor do tipo sonar é um sensor de proximidade do tipo ativo, onde a distância é medida através de sinais acústicos. Assim como os sensores IR, o sonar emite um pulso ou sinal sonoro e faz a leitura do sinal refletido, calculando o *delay* entre a emissão e o recebimento do eco.

2.1.1.4 Sensores do tipo radar

O funcionamento de um sensor do tipo radar é similar ao de um sonar. Ondas de rádio de alta freqüência são emitidas e suas reflexões observadas para se determinar uma distância. Os radares são interessantes para a robótica móvel por que são rápidos e fornecem informações sobre a superfície e sua geometria. Por ser baseado em ondas de rádio, não depende de atmosfera, sendo próprio para a exploração espacial, explorando superfícies de outros planetas. O radar é classificado como um sensor ativo.

2.1.1.5 Sensores Laser

Os sensores laser são do tipo ativo e se destacam pela sua precisão. A medição de distâncias através de um sensor laser pode ser feita das seguintes formas:

- Triangulação;
- *Time-of-flight*;

- Baseado em fase.

Na triangulação, as relações geometricas são observadas entre a posição atual do robô, o raio sendo emitido e o raio sendo recebido. No *time-of-flight*, é medido o atraso entre o raio ser emitido, atingir um objeto e retornar. No baseado em fase, é medida a diferença de fase do sinal emitido e recebido. O laser é uma alternativa muito boa para a robótica móvel, dado sua eficiência e precisão para a manutenção da posição do robô.

2.1.1.6 Sensores baseados em satélites (GPS)

Surgiu a partir de um projeto do Departamento de Defesa dos Estados Unidos, onde o objetivo era criar um sistema para se estimar uma posição baseado em satélites orbitando a Terra. Este sistema permite calcular a posição absoluta e velocidade com uma precisão muito alta. Funciona baseado no *time-of-flight*, ou seja, no atraso do sinal, é possível estimar a longitude e latitude, elevação, condição atmosférica e a hora atual em qualquer ponto na superfície da Terra. Existem dois sistemas: o SPS (*Standard Positioning System*), que é voltado para o uso civil, possuindo um certo nível de erro. Existe também o PPS (*Precise Positioning System*), voltado para o uso militar. Por ser criado para propósitos militadores, não há a emissão de qualquer tipo de energia, sendo o receptor completamente passivo.

2.1.1.7 Bússola Eletrônica

A bússola eletrônica é um sensor passivo que utiliza o campo magnético da Terra para orientar o robô em relação ao norte. Por fazer leituras do campo magnético, é suscetível a interferências externas, como grandes estruturas metálicas (podendo incluir o corpo do robô). Para que este problema seja amenizado, é preciso um passo de calibração antes do uso da bússola.

2.1.1.8 Odômetros

Os odômetros basicamente fazem a medida de qual a distância o robô percorreu em um determinado espaço de tempo. Odômetros podem apresentar problemas no que diz respeito à exatidão desse tipo de sensor, por não haver um *feedback* preciso do mundo exterior indicando se o robô realmente se deslocou. Por exemplo, o robô pode avançar e derrapar um pouco, mas o odômetro fará a leitura absoluta de quanto as rodas do robô giraram.

2.1.1.9 Visão

Os sensores de visão obtém uma grande quantidade de dados. Para o processamento desses dados, normalmente é necessário um *hardware* dedicado para um determinado tipo de análise e processamento destes dados. Com um sensor CCD (*charge-coupled device*), o robô pode amostrar o ambiente onde está inserido cerca de 30 vezes por segundo, obtendo imagens com uma resolução de 640x480 pixels (*picture elements*). Normalmente estas imagens obtidas são do tipo *grayscale* (sistemas de cores vide Seção 2.2).

Com as imagens obtidas, podem ser aplicadas técnicas de processamento de imagens para se extrair informações. Estas informações podem ser obtidas por reconhecimento de padrões, detecção de bordas, análise de cores, etc. Usando-se duas câmeras, pode ser feita a reconstrução do ambiente em três dimensões. Isso

permite que, baseado na visão, a distância do robô a um objeto possa ser calculada. Entretanto, a reconstrução de ambientes acarreta um alto custo computacional e é sujeita à problemas de oclusão e imprecisão. Com o uso de visão, também é possível estimar a posição atual do robô baseando-se em padrões, que seriam imagens previamente armazenadas, a imagem que o robô está "vendo". Mais informações sobre o uso de visão na navegação podem ser encontradas na Seção 2.5.

2.1.2 Modos de Operação

Na robótica móvel, existem três tipos de modos de operação relativos à autonomia. Estes três tipos são:

- Não-Autônomo;
- Semi-Autônomo;
- Autônomo.

Na maioria dos sistemas robóticos móveis, é esperado que o robô execute tarefas com um certo grau de controle humano ou por uma programação previamente feita (Dudek and Jenkin, 2000). No modo não-autônomo, o robô é totalmente controlado por um operador humano em tempo integral, não possuindo nenhum tipo de autonomia. No modo semi-autônomo, ainda existe a presença de um operador humano, mas não em tempo integral. Neste modo de operação, o robô possui um certo nível de autonomia, onde é possível avaliar uma tarefa requisitada pelo operador humano para corrigir eventuais enganos/limitações. Por exemplo, o operador envia um comando de avançar mas o robô detecta que há um obstáculo a sua frente. Os AGV (*automated guided vehicles*) podem ser classificados como semi-autônomos. Quando o robô é classificado como completamente autônomo, isto é, sem a presença de um controlador humano, ele pode operar durante longos períodos sem receber comandos externos, baseado somente em uma programação anterior.

2.1.3 Sistemas de Controle

Nesta seção, serão descritos os principais tipos de sistemas de controle para robôs móveis autônomos.

2.1.3.1 Sistemas de Controle Deliberativo

Este sistema de controle usa o modelo SMPA (*sense-model-plan-act*). Os robôs recebem informações sobre o ambiente onde estão inseridos através de seus sensores. Baseado nos dados dos sensores, o robô construirá um modelo mais completo possível do ambiente onde está para que possa gerar um plano de ação para conseguir cumprir seu objetivo e por último, executa este plano.

Estes sistemas são frágeis, incapazes de operar fora de um ambiente controlado. Isso ocorre por que é necessário um modelo muito exato do ambiente onde o robô está operando.

2.1.3.2 Sistemas de Controle Reativo

Em geral, não possuem uma representação do ambiente onde estão inseridos e funcionam sob o princípio dos sistemas sensoriais-motores, ou seja, sentem e agem.

É baseado no princípio da reatividade, isto é, na suposição que comportamentos inteligentes podem emergir a partir da interação de comportamentos simples em sistemas dinâmicos complexos.

Existem vários processos que concorrem entre si. Cada processo recebe dados de seus sensores. Alguns desses processos podem inibir outros, dependendo das prioridades de cada um.

2.1.3.3 Sistemas de Controle Híbrido

O objetivo destes sistemas é integrar os sistemas deliberativos e reativos. Basicamente existirão dois módulos: um será responsável por planejar e executar tarefas de longo prazo enquanto o outro será responsável por controlar o robô em situações de reação imediata, como desviar de um obstáculo. O grande problema do sistema híbrido é fazer a integração entre a parte deliberativa e a reativa. Para se resolver estes conflitos, usualmente é criado um terceiro módulo para fazer a comunicação entre o módulo deliberativo e o reativo. Estes tipos de sistema também são conhecidos como sistemas de três camadas.

2.1.4 Navegação

Para que um robô seja capaz de ir de um ponto a outro, ele precisa saber sua posição em um dado ambiente e o caminho que ele precisa percorrer até o objetivo. Com estas informações, o robô poderá planejar e executar a tarefa de se deslocar até um local específico. Este conjunto de condições e ações é chamado de navegação (Heinen, 2000). A navegação robótica pode ser dividida em três grupos: global, local e híbrida. Em relação a este trabalho, a navegação pode ser acrescida de mais uma forma de navegação: a baseada em visão.

2.1.4.1 Navegação Global

Na navegação global, o robô planeja o caminho que irá percorrer baseado no posicionamento absoluto ou relativo a um mapa de um ambiente. Com estas informações, o robô pode então se movimentar.

2.1.4.2 Navegação Local

Na navegação local, a posição é sempre relativa a objetos (estáticos ou em movimento) do ambiente. O robô então avança no seu caminho evitando tais obstáculos.

A navegação visual pode ser enquadrada como uma navegação local, pois ela será relativa a pontos de referência contidos em uma sequência de imagens. Esta seqüência de imagens que descreverá uma determinada área do ambiente onde o robô está inserido. Exemplos de trabalhos relacionados à navegação visual podem ser vistos na Seção 2.5.

2.1.4.3 Navegação Híbrida

Neste tipo de navegação, existe a integração da navegação global e local. Assim, o robô é capaz de planejar tarefas (seu deslocamento, por exemplo) e também detectar e desviar de obstáculos encontrados durante o caminho (Heinen, 2002).

2.1.4.4 Discussão sobre Navegação

Para que o robô possa executar uma navegação indo de um ponto a outro, existem basicamente duas abordagens: com ou sem mapas do ambiente onde o robô está inserido. Outro ponto importante a ser ressaltado é o tipo de controle utilizado pelo robô, que pode ser deliberativo ou reativo.

Na navegação visual não é utilizado um mapa convencional, isto é, um mapa geométrico ou grade de ocupação do ambiente onde está o robô, mas uma representação da trajetória que deverá ser percorrida através de uma seqüência de imagens. Utilizando-se esta seqüência de imagens, é possível executar uma tarefa de alto nível, que é a navegação do robô de um ponto a outro. Ao mesmo tempo que a navegação visual pode proporcionar um comportamento deliberativo, pode proporcionar um comportamento reativo quando o robô ajusta sua posição por *feedback* visual, além de poder detectar a oclusão do caminho e reagir adequadamente a esta situação.

De uma forma geral, a navegação visual possui elementos de navegação global e local, além de apresentar também mais de uma forma de comportamento (deliberativo e reativo).

2.2 O Espaço de Cores RGB

Segundo a definição do modelo de cor RGB encontrado em (Gonzalez and Woods, 2002), cada cor (componente) está na sua forma primária Red/Green/Blue (vermelho, verde, azul). Este modelo é baseado em um sistema de coordenadas cartesianas. Nesse modelo, os tons de cinza (pontos com os mesmos valores em R, G e B) vão desde a origem (preto) até o ponto mais distante (branco). As cores neste modelo são pontos dentro do cubo e são definidos por vetores se estendendo desde a origem. Um exemplo de cubo RGB pode ser visto na Figura 2.1. Todas faces do cubo RGB podem ser vistas na Figura 2.2.

Imagens representadas no modelo de cores RGB consistem de três imagens componentes, uma para cada cor primária. Quandos estas imagens componentes são dadas como entrada em um monitor, essas três imagens são combinadas na tela de fósforo para produzir uma imagem colorida. O número de bits usados para representar cada pixel no espaço RGB é chamado de *pixel depth*. Vamos considerar uma imagem que possui 8-bit em cada componente RGB. Nessas condições, cada pixel RGB possui uma profundidade de 24-bit (três planos de imagens vezes o número de bits por plano). Uma imagem é dita *full-color* quando possuir pelo menos 24-bit de profundidade no espaço RGB.

2.3 Outros Espaços de Cores

O modelo de cores RGB se encaixa bem para implementação em *hardware*, mas não para o olho humano. Desse forma, um observador humano não tem como definir com precisão uma cor a partir do modelo RGB visto que a cor é composta por três componentes. Assim, uma forma de escolha de cores mais intuitiva é preferível. A seguir, serão descritos os modelos de cores HSI e HSV (Couleur.org, 2004; Gonzalez and Woods, 2002).

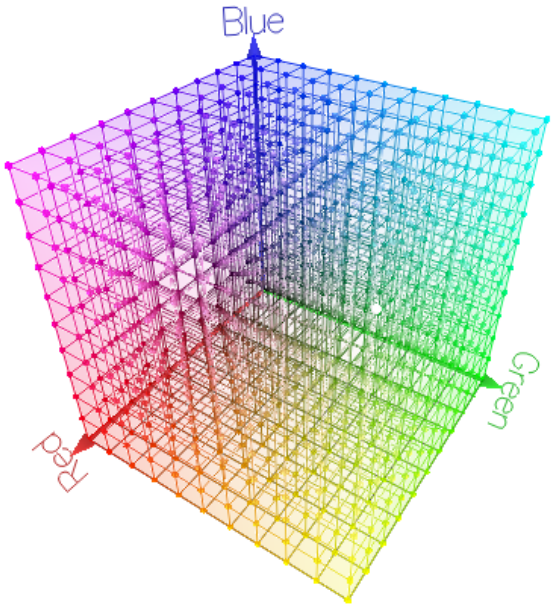


Figura 2.1: Cubo RGB, gerado pelo programa RGB Cube (Couleur.org, 2004).

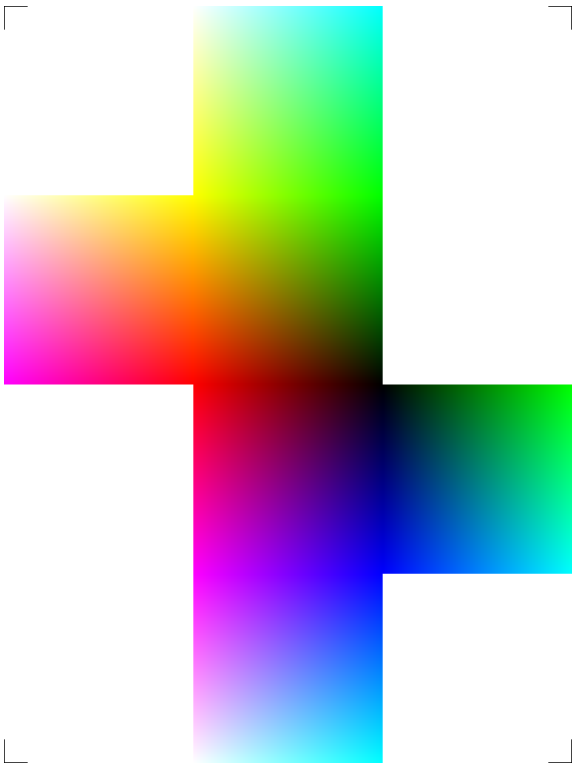


Figura 2.2: Faces do Cubo RGB.

2.3.1 O modelo de cores HSI

O modelo HSI (*Hue-Saturation-Intensity*) possui três componentes assim como o modelo RGB. O modelo HSI pode ser pensado como o cubo RGB "pendurado" pelo vértice branco. O componente *Hue* descreve a cor. O componente *Saturation* informa o quão próximo o componente *Hue* está próximo do eixo da intensidade. Quanto mais distante do eixo da intensidade, mais pura é a cor. A componente *Intensity* informa se a cor está mais próxima do branco ou preto. A forma polar do modelo HSI é adequado para o uso em uma interface com um usuário para se fazer a escolha de uma cor. As Figuras 2.3(a) e 2.3(b) mostram o espaço de cores HSI e sua forma polar. A conversão de RGB para HSI pode ser vista na Equação 2.1 (Couleur.org, 2004).

$$\begin{cases} H &= \arctan\frac{\beta}{\alpha} \\ S &= \sqrt{\alpha^2+\beta^2} \\ I &= \frac{R+G+B}{3} \end{cases} \tag{2.1}$$

$$\begin{cases} \alpha &= R - \frac{1}{2}(G+B) \\ \beta &= \frac{\sqrt{3}}{2}(G-B) \end{cases} \tag{2.2}$$

2.3.2 O modelo de cores HSV

O modelo HSV é muito similar ao modelo HSI, possuindo também três componentes (*Hue-Saturation-Value*). O modelo HSV pode ser pensado como o vértice branco sendo achatado até o plano que corta o cubo RGB e passa pelo vértice azul e amarelo. Assim como no modelo HSI, o componente *Hue* descreve a cor. O componente *Saturation* descreve a pureza de uma cor em relação a cor branca. O componente *Value* funciona como se fosse a luminosidade. Quanto menor o valor, mais escura será a cor. Como no modelo HSI, a forma polar do HSV é adequada para interfaces com o usuário para se fazer a escolha de uma cor. Figuras 2.4(a) e 2.4(b) mostram o espaço de cores HSV e sua forma polar. A conversão de RGB para HSV pode ser vista no Algoritmo 1 (Couleur.org, 2004).

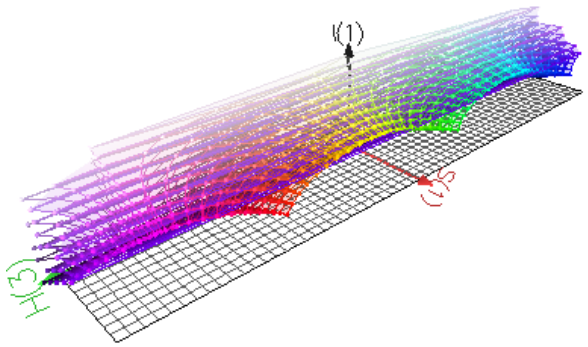
2.4 Matching em Imagens

Nesta seção serão vistas algumas alternativas para se realizar o *matching* entre imagens, destacando o *matching* através de correlação, que foi utilizado neste trabalho.

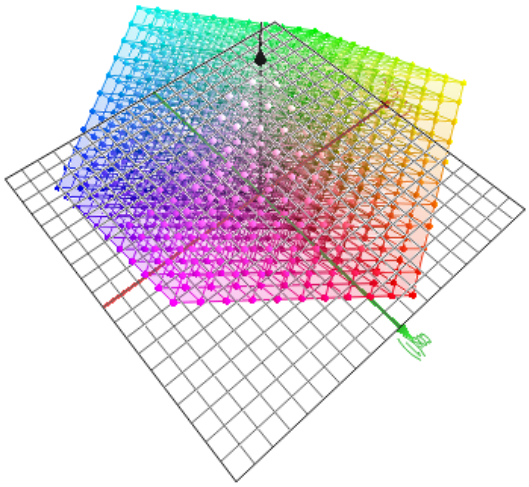
2.4.1 Correlação

Na visão computacional (Jain, 1995), o objetivo da correlação é determinar a posição que uma imagem de exemplo é encontrada em uma outra imagem. Uma técnica que permite se obter tal resultado é a correlação cruzada. As definições a seguir são encontradas em (Martin and Crowley, 1995).

Vamos definir que usaremos uma imagem de exemplo S de tamanho M por N pixels. Esta imagem S deve ser detectada em uma imagem de referência R. A imagem S tem tamanho igual ou inferior ao tamanho da imagem R. Quando não houver ruído, S é detectada em R na posição (i,j) se a soma dos quadrados das



(a) Espaço HSI

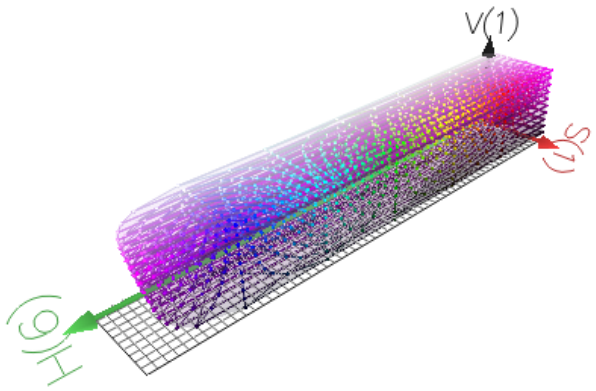


(b) Espaço HSI Polar

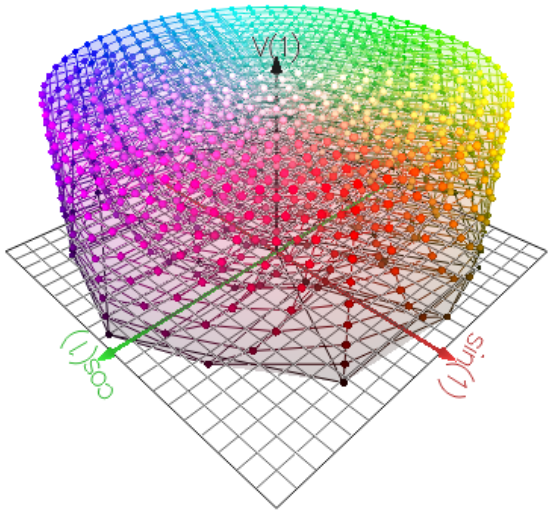
Figura 2.3: O modelo de cores HSI

se $R > G$ **então**
  $Max = R$; $Min = G$; $Position = 0$;
senão
  $Max = \mathrm{G}$; $Min = R$; $Position = 1$;
fim
se $Max < B$ **então**
  $Max = B$; $Position = 2$;
fim
se $Min > B$ **então**
  $Min = B$;
fim
$V := Max$;
se $Max \neq 0$ **então**
  $S = \frac{Max-Min}{Max}$;
senão
  $S = 0$;
fim
se $S \neq 0$ **então**
  se $Position = 0$ **então**
    $H = \frac{1+G-B}{Max-Min}$;
  senão se $Position = 1$ **então**
    $H = \frac{3+B-R}{Max-Min}$;
  senão
    $H = \frac{5+R-G}{Max-Min}$;
  fim
fim

Algoritmo 1: Conversão de RGB para HSV



(a) Espaço HSV



(b) Espaço HSV Polar

Figura 2.4: O modelo de cores HSV

diferenças (SSD) é um valor abaixo de um certo limite. Assim, o *matching* por SSD é medido na posição (i,j), como mostrado na Equação 2.3.

$$SSD(i,j) = \sum_{m=0}^{M}\sum_{n=0}^{N}(R(i+m,j+n) - S(m,n))^2 \tag{2.3}$$

Assim, a SSD(i,j) deve ser computada sobre a área da imagem que possa conter a sua localização.

Com um R ótimo (sem ruído), calculando-se a SSD, o resultado é igual a duas vezes o produto interno de R(i,j) e S, subtraída da soma dos quadrados de S e da vizinhança de R, como visto na Equação 2.4. A correlação cruzada é um produto interno de cada pixel. A raiz quadrada da soma dos quadrados é uma medida da energia do sinal. Assim podemos escrever:

$$SSD(i,j) = {E^2}_R(i,j) + {E^2}_S - 2.SR(i,j) \tag{2.4}$$

$$onde \tag{2.5}$$

$$SR(i,j) = \sum_{m=0}^{M}\sum_{n=0}^{N} S(m,n).R(i+m,j+n) \tag{2.6}$$

$${E^2}_R = \sum_{m=0}^{M}\sum_{n=0}^{N} R(i+m,j+n)^2 \tag{2.7}$$

$${E^2}_S = \sum_{m=0}^{M}\sum_{n=0}^{N} S(m,n)^2 \tag{2.8}$$

O termo ${E^2}_S$ é a energia do sinal do exemplo S e o termo ${E^2}_R(i,j)$ é a energia do sinal da vizinhança M por N de R, localizada na posição (i,j). Como mostrado acima, a SSD de uma imagem de exemplo e uma vizinhança na imagem de referência pode ser substituída por correlação cruzada, provendo para as imagens de exemplo e referência uma nomalização adequada. A normalização mais direta é dividir cada produto interno pela raiz quadrada da energia do exemplo e da vizinhança. Isso é expresso por:

$$NCC(i,j) = \frac{SR(i,j)}{\sqrt{{E^2}_R(i,j).{E^2}_S}} \tag{2.9}$$

Com a Equação 2.9, é possível localizar uma imagem exemplo S em uma imagem de referência R. A existência de *hardware* de baixo custo para correlação faz com que a correlação cruzada seja uma operação atrativa.

2.4.2 Comparação de Histogramas

No *matching* por histogramas (Moschetta et al., 2002, 2001), os histogramas dos componentes RGB para uma imagem exemplo são gerados. A seguir, uma cabeça de leitura com dimensões idênticas à imagem de exemplo é deslizada sobre as imagens de referência. Para cada posição da cabeça de leitura, são gerados histogramas para cada componente RGB e estes comparados contra os histogramas da imagem de

exemplo. As imagens de referência que possuírem os histogramas mais similares ao da imagem de exemplo, isto é, com erro inferior a um determinado limite, serão escolhidas.

A comparação de histogramas faz uma avaliação da distribuição das cores em uma determinada região, não se importando com a disposição espacial dos pixels. Isto pode comprometer a qualidade do *matching* caso a ordem das cores influa no resultado.

2.4.3 Comparação Ponto a Ponto

A idéia por trás do *matching* através de comparações ponto a ponto (Moschetta et al., 2002, 2001) é bem simples. Este tipo de comparação consiste de uma janela (imagem exemplo) que desliza sobre uma ou mais imagens de referência, calculando-se a diferença entre os pixels do exemplo e da referência. Esta diferença é cumulativa para cada posição da janela e as imagens de referência que possuírem o valor de erro abaixo de um limite são escolhidas. A imagem de exemplo é uma imagem menor ou de igual tamanho ao das imagens de referência.

Esta abordagem possui problemas no que diz respeito a qualidade do *matching*. A imagem de exemplo e parte da imagem de referência precisam ser muito similares para se obter um bom resultado, não sendo tolerantes a ruído, distorções e mesmo pequenas variações de tonalidade.

2.4.4 Invariant Features

Invariant Features (Lowe, 2004) é um método capaz de extrair *features* de uma imagem, as quais podem ser usadas para se fazer um *matching* confiável entre imagens de um objeto ou de uma cena em uma outra imagem. As *features* são invariantes a escala e rotação da imagem, mostrando robustez no *matching* em uma gama muito grande de situações, como distorções, adição de ruído, mudança no ponto de vista e mudanças de iluminação. As *features* são distintas entre si, no sentido que uma única *feature* tem uma alta probabilidade de ser casada corretamente em uma base de imagens grande com muitas imagens.

2.5 Trabalhos na Área de Navegação Visual

Várias propostas são encontradas na literatura (Matsumoto et al., 1996; Jones et al., 1997; Hu and Uchimura, 2002; Matsumoto et al., 1999; Gross et al., 2003), descrevendo sistemas que permitem uma navegação do robô através do uso de visão computacional, onde consideram as imagens adquiridas por uma câmera de vídeo para realizar o controle do robô. Estes sistemas são usualmente denominados de *sistemas de navegação visual*. O objetivo de alguns destes sistemas é imitar o comportamento humano, onde uma pessoa é capaz de ser conduzida por um caminho, armazenar em sua memória o caminho percorrido, e em um momento posterior, realizar de modo autônomo novamente este mesmo caminho, baseando-se em suas lembranças.

Sistemas de navegação visual, baseados na memória de um caminho percorrido previamente, se propõem a permitir que um robô possa executar uma tarefa pré-determinada, como levar um material de uma sala à outra, considerando esta memória de imagens. Nesta seção, serão descritos alguns trabalhos estudados durante a realização deste trabalho.

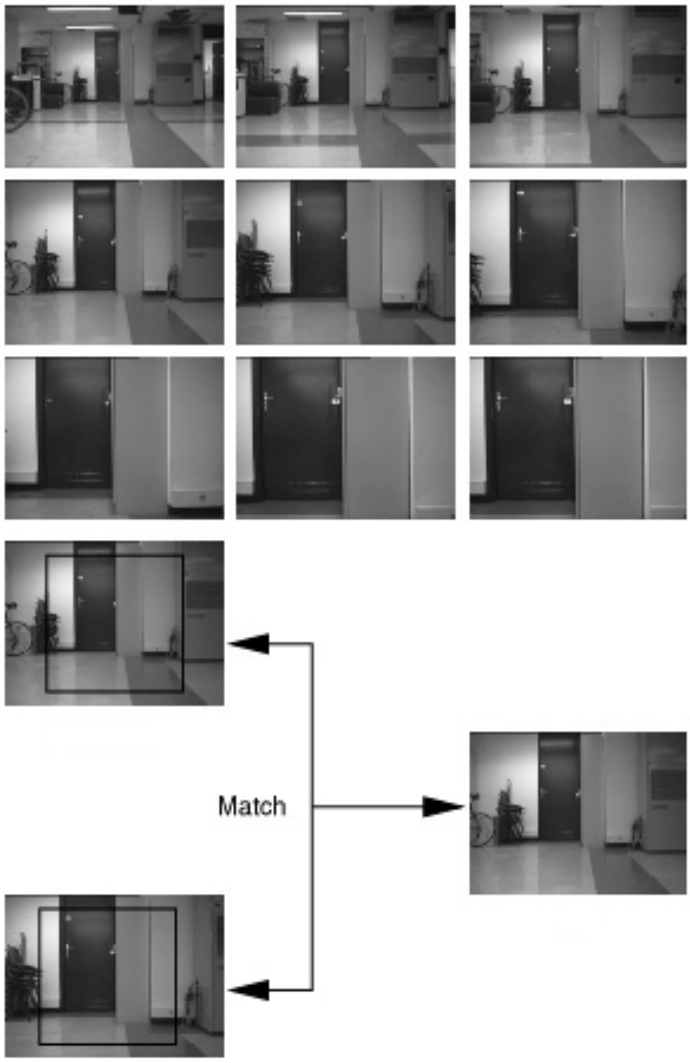


Figura 2.5: Exemplo de trajetória que um robô deve seguir (Jones et al., 1997).

Dentre os trabalhos relacionados pesquisados, pode ser destacado o de (Jones et al., 1997), que serviu de modelo para a proposta apresentada na Seção 3.4.1. Este trabalho optou em seguir um modelo similar ao apresentado por ele, sendo descrito brevemente a seguir. A principal diferença entre este trabalho e o trabalho de (Jones et al., 1997) é o uso de cores no processo de *matching*.

Segundo Jones, Andersen e Crowley (Jones et al., 1997), as técnicas baseadas em visão fornecem uma abordagem alternativa para o uso de visão computacional na navegação. Tais técnicas provem meios de definir processos simples que transformem imagens em comandos de deslocamento e orientação para se estimar a posição e orientação de um robô. Estas técnicas também são facilmente usadas para se obter informações sobre a navegação.

O experimento que (Jones et al., 1997) realizou consistia em fazer um robô móvel navegar de sua posição inicial até um escritório. O escritório se encontra no mesmo

andar em que o robô está. A navegação inclui chegar até uma porta aberta, cruzar esta porta e chegar em um corredor. O robô utilizado usa odometria e câmeras para obter informações do ambiente. O ambiente é estruturado e estável, porém não é estático.

Em (Jones et al., 1997), a correlação de imagens é utilizada para se determinar o posicionamento atual de um robô. O robô captura uma imagem, e faz o cálculo de correlação contra imagens de referência previamente armazenadas. Estas imagens de referência compõem um caminho ou trajetória que o robô deverá seguir, como visto na Figura 2.5. Conforme o robô avança na sua trajetória, o campo de visão irá se alterar, logo o valor da correlação vai diminuindo. Quando não satisfizer mais um determinado limite, uma nova imagem de referência é adotada, sendo usualmente esta imagem a seguinte na seqüência que descreve o caminho, e o processo então se repete.

O trabalho de Matsumoto, Inaba e Inoue (Matsumoto et al., 1996) serviu de base para o trabalho de (Jones et al., 1997). Em (Matsumoto et al., 1996), é apresentada uma nova representação visual de um caminho, chamado de *View-Sequenced Route Representation* (VSRR). A VSRR é um modelo de um caminho que contém uma sequência de imagens obtidas ao longo do mesmo. Uma representação deste modelo pode ser vista na Figura 2.6. Em uma execução autônoma, a localização, determinação de ângulo e detecção de obstáculos são obtidas em tempo real pelo *matching* entre a imagem atual vista pelo robô e uma imagem da sequência memorizada, através do uso de correlação.

As imagens utilizadas para compor a VSRR são obtidas em uma "trajeto de aprendizagem". O robô percorre o caminho obtendo as imagens, armazenando-as para posterior consulta. Devido ao limite de memória do robô, as imagens são reduzidas antes de serem armazenadas na memória do robô. Uma nova imagem é armazenada sempre que a nova imagem possuir um diferênça visual muito grande em relação a anterior ou um limite de distância percorrida é atingido. As imagens de referência (VSRR) podem ter ações pré-determinadas associadas para que, quando o robô use como referência uma imagem com uma ação associada, a execute.

O processo de comparação de duas imagens e cálculo de suas similaridades é chamado de *matching*. Este *matching* é feito através de correlação, implementada em *hardware*, em uma placa de processamento de visão. O processo de correlação é feito entre uma imagem de exemplo e a região central de uma imagem contida na VSRR.

A correlação entre uma imagem obtida pelo robô e uma imagem da VSRR será maior quando o robô estiver perto de onde a imagem da VSRR foi obtida. Durante o percurso do robô, é calculada a correlação entre a imagem atual e imagens da VSRR. A imagem para a qual uma maior correlação foi obtida é identificada como posição atual do robô. Os resultados da correlação também são utilizados para determinar o deslocamento lateral do robô. Quando um obstáculo é encontrado, o robô pára até que o obstáculo seja removido.

Em outro trabalho, Matsumoto, Ikeda, Inaba e Inoue (Matsumoto et al., 1999), Utilizou um câmera equipada com um espelho hiperboloidal. A imagem capturada com este tipo de equipamento é muito distorcida. Após um processamento, é gerado um panorama da região ao redor do robô. Este processamento consiste em se fazer uma projeção cilíndrica para "abrir"a imagem e se ter a noção real de panorama em 360 graus.

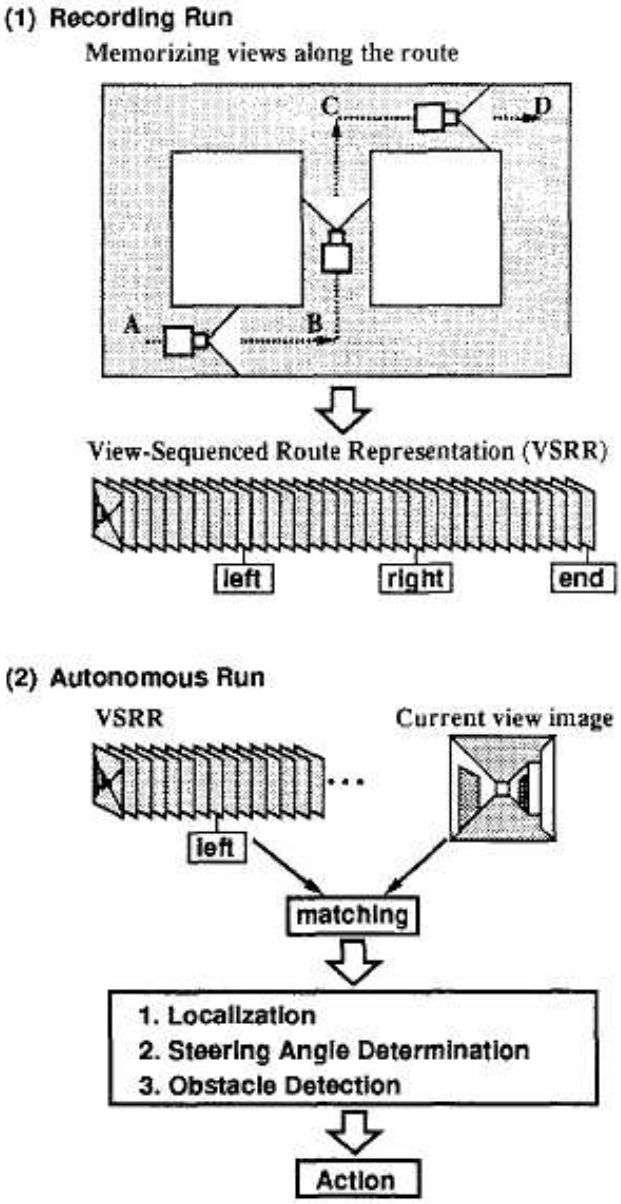


Figura 2.6: Estratégia descrita em (Matsumoto et al., 1996).

Com esta abordagem, alguns problemas que existiam em (Matsumoto et al., 1996) foram solucionados. O primeiro problema solucionado foi a questão da navegação visual *one-way*, ou seja, o robô sabia apenas o caminho em uma direção. No trabalho anterior, o robô memorizava o caminho em apenas uma direção e, para realizá-lo no sentido oposto, uma nova memorização era necessária. Usando *omni-views*, isso é possível apenas invertendo a seqüência das imagens. Uma segunda melhoria é sobre a precisão ao se seguir uma trajetória. Tendo uma *omni-view* (imagem de 360 graus), o robô tem dois pontos de referência distintos para se guiar: a visão frontal e traseira do caminho. Se o robô estiver perfeitamente alinhado na trajetória sendo seguida, o ângulo entre a imagem da frente e a de trás é de 180 graus. Este ângulo diminui caso o robô esteja deslocado lateralmente em relação a sua trajetória. A última melhoria obtida foi a robustez da localização de uma imagem em outra. Tendo um ângulo de visão muito maior, o *matching* é melhorado em muito, especialmente em locais próximos à paredes. Memorizando imagens maiores, a tarefa do robô de achar um ponto de localização é facilitada.

No trabalho (Gross et al., 2003), também foi utilizada uma câmera com um espelho hiperboloidal. Diferentemente dos outros trabalhos, as imagens sendo processadas no trabalho de (Gross et al., 2003) são coloridas. As imagens coloridas são muito suscetíveis a mudanças na iluminação. Para atacar este problema, foi acoplada à câmera um dispositivo para calibrar a câmera em relação a quantidade da cor branca sendo capturada. A calibragem ocorria sempre antes de se obter uma nova imagem. Com isso, as imagens possuiam cores mais uniformes. Os caminhos que o robô pode percorrer são descritos por um grafo e, em cada nó desse grafo, informações sobre odometria e referências visuais são capturadas. Para se avaliar as cores de uma cena, a imagem de referência é dividida em partes que não se sobrepõem e obtida a cor média. A imagem servia para corrigir eventuais desvios do controle feito via odometria.

O estudo que (Jones et al., 1997) realizou utilizava imagens *grayscale*. Neste trabalho, visamos o estudo de correlação em imagens coloridas, para isso precisamos definir qual espaço de cores que iremos empregar no estudo. O espaço de cores utilizado nas imagens coloridas foi o RGB (Red-Green-Blue) (Gonzalez and Woods, 2002). A escolha deste espaço de cores foi baseada principalmente pela sua simplicidade. Pretendemos em trabalhos futuros realizar estudos em outros espaços de cores, tais como o HSV/HSI.

Para fazer a correlação em imagens coloridas, adotamos uma técnica que avaliou individualmente cada componente de uma imagem RGB. Os componentes em uma imagem RGB não são independentes, mas mesmo assim podemos tratar cada componente como se ele fosse uma imagem *grayscale*, desde que depois possamos "integrar" (juntar) os resultados obtidos para cada componente. Isto nos permitiu aplicar a técnica NCC (*normalized cross-correlation*) disponível atualmente.

Uma vez calculada a correlação em cada componente do espaço RGB, teremos 3 indicações do posicionamento do robô. Isto nos levou a propor um método baseado em vetores e correlação, como descrito na Seção 3.4.1.

3 METODOLOGIA

Neste capítulo, serão discutidos e abordados alguns pontos-chave deste trabalho. O problema proposto neste trabalho é utilizar técnicas existentes na área de Processamento de Imagens para auxiliar no controle e navegação de robôs móveis autônomos. Como visto no Capítulo 2, os trabalhos desenvolvidos até agora empregam somente imagens do tipo *grayscale*. Neste trabalho, pretendemos adaptar e desenvolver técnicas de processamento de imagens para a utilização de imagens coloridas, no intuito de aproveitarmos toda a informação que as cores podem proporcionar.

Aqui serão vistas a metodologia e as estratégias utilizadas neste trabalho para a resolução do problema proposto. Primeiramente, serão vistas algumas abordagens de como pode ser feito o processamento de imagens, para que se possa atingir o objetivo de se desenvolver uma proposta de algoritmo de *matching*. O algoritmo fará o *matching* através do uso de NCC (*normalized cross-correlation*) entre imagens coloridas. Este algoritmo será utilizado para a prototipação de um sistema de navegação visual. Neste capítulo discutiremos a metodologia e os processos a fim de implementar este protótipo.

3.1 Definição do Problema

O problema básico que este trabalho aborda é como desenvolver um sistema de navegação visual baseado em imagens coloridas para robôs móveis autônomos.

Neste tipo de sistema, um robô móvel autônomo com uma câmera acoplada deverá ser capaz de seguir referências visuais. Estas referências irão compor uma lista de imagens, definindo assim uma trajetória virtual descrita através de um banco de imagens (BI).

Tais sistemas já existem (Jones et al., 1997; Matsumoto et al., 1996, 1999; Gross et al., 2003). Entretanto, os sistemas existentes hoje fazem uso apenas de imagens do tipo *grayscale*.

O uso de cores pode proporcionar uma grande quantidade de informações e uma riqueza de detalhes ainda muito pouco explorada em outros trabalhos. Desta forma, pretendemos desenvolver um sistema de navegação visual baseado em imagens coloridas. Para se atingir este objetivo, técnicas de processamento de imagens existentes devem ser adaptadas para o uso de imagens coloridas.

As técnicas de processamento de imagens que este trabalho propõe serão baseadas principalmente nos trabalhos de (Jones et al., 1997) e (Matsumoto et al., 1996). Elas deverão ser capazes de determinar se uma imagem capturada pelo robô (ICR) pertence ao trajeto, localizá-la dentro do banco de imagens (BI) e verificar o posi-

cionamento atual do robô em relação a uma imagem de referência (IR) do BI. Esse deslocamento em relação ao posicionamento consiste em o robô estar mais para direita ou esquerda, para frente ou para trás em relação a uma dada IR.

Com o resultado do processamento das imagens (ICR e IR), podemos analisá-lo e definir qual ação será tomada para ser gerado um comando para o robô. Após definido se uma ICR é relativa a uma dada IR, analisamos o deslocamento da IR em relação a ICR referente a direção (esquerda ou direita) e isto resultará em comandos para o robô para este se alinhar voltado para a nova direção em relação a uma direção de referência indicada pela IR . Isto também é válido para a proximidade entre imagens (*zoom*).

Um ponto importante a ser ressaltado é que a imagem capturada pelo robô (ICR) não é necessariamente uma imagem idêntica a alguma imagem contida em um BI.

Conforme o processo de *matching* vai ocorrendo, conseguimos determinar e manter um controle sobre a posição atual do robô. Esta informação sobre o posicionamento é parte fundamental do processo de navegação.

3.2 Limitações

Uma das principais limitações deste trabalho é não possuir o *hardware* ideal para a validação do sistema de navegação visual. Sendo assim, a validação dos resultados será de forma off-line, ou seja, sem a utilização de um robô. Os dados para validação serão basicamente bancos de imagens coloridas de ambientes internos e uma câmera digital que será deslocada simulando o que seria a movimentação do robô.

O grau de eficiência do protótipo (informações corretas) será baseado unicamente em experimentos, considerando o deslocamento (erro) da imagem atual (ICR) em relação a uma imagem de referência (IR) contida no banco de imagens.

O sistema de navegação visual proposto também não prevê qualquer tipo de ação caso apareça algum tipo de obstáculo imprevisto na trajetória a ser percorrida. Basicamente, o robô apenas deve ser capaz de seguir uma trajetória previamente armazenada, não havendo nenhum tipo de adaptação na trajetória, sendo apenas um "AGV guiado de forma visual".

Existem também limitações referentes às técnicas de processamento de imagem utilizadas. A principal limitação é o problema da iluminação no momento de aquisição das fotos. Quando ocorre uma variação de luminosidade um pouco mais acentuada, os valores de correlação e a posição do pico de correlação obtidos podem variar muito. Nessas condições, o algoritmo NCC pode fornecer falsos-positivos, ou seja, indicação indevida da localização de um ponto de referência, podendo até mesmo guiar de forma incorreta o robô.

3.3 Estratégias

Abaixo, são descritas algumas estratégias utilizadas neste trabalho. Com estas estratégias, foram obtidos alguns resultados e, baseando-se nestes resultados, foi proposta a abordagem descrita na Seção 3.4.

3.3.1 Criação do Banco de Imagens

Uma câmera digital foi utilizada para a captura das imagens que, posteriormente, foram organizadas em uma seqüência que representa um trajeto, constituindo assim

um banco de imagens (BI).

A criação dos Bancos de Imagens (BI) baseou-se em imagens não controladas, ou seja, imagens de ambientes comuns. Esses ambientes são interiores, como apartamentos, empresas ou laboratórios. Exemplos de BIs podem ser vistos no Apêndice A.

Basicamente, foram utilizados dois BI, um para testes da técnica de correlação e outro para o experimento de navegação. Estes BI podem ser vistos nas Seções A.1 e A.2. O BI para o teste da técnica de correlação (BI Teste) possui 8 fotografias. O BI para o teste da navegação (BI Navegação) possui 30 fotografias.

Os BIs foram gerados utilizando-se uma câmera digital sobre um tripé para a aquisição das imagens. O sensor CCD da câmera utilizada tem uma resolução de 3.2 Megapixels. O modelo utilizado foi uma Canon Powershot A300 (Canon, 2003). Cada imagem capturada tem 640x480 pixels de dimensão com a qualidade máxima permitida pela câmera.

O caminho para montagem das Imagens de Referência (IR) era percorrido e ao, longo deste, fotografado em intervalos de aproximadamente um metro. Para a montagem do caminho das Imagens Capturadas pelo Robô (ICR) o mesmo é percorrido e o intervalo das fotografias são mais irregulares, variando entre 30 e 50 centímetros.

3.3.2 Ferramentas Utilizadas

Para as prototipações do algoritmo de *matching* e o sistema de navegação visual, foi utilizado o Matlab (Mathworks, 2002; Hanselman and Littlefield, 2003). O ambiente do Matlab possui uma curva de aprendizado suave e integra computação matemática, visualização e linguagem de programação. Além disso, fornece a possibilidade de incorporar rotinas externas escritas em outras linguagens, como C++ (Stroustrup, 1997) por exemplo. Uma rotina específica do *Toolbox* (Mathworks, 2002) de processamento de imagens é usada intensamente, a rotina `normxcorr2`, que faz o cálculo da NCC. O código fonte da rotina `normxcorr2` está disponível, o que torna possível a migração do código escrito para Matlab para um linguagem de programação de propósito geral. Outro diferencial é a facilidade de se manipular diferentes formatos de arquivos gráficos, como JPEG e TIFF (Murray, 1996).

Abaixo está um exemplo de script do Matlab para se calcular a correlação para uma componente RGB e gerar o gráfico resultante (visto de cima).

```
>> img1 = imread('ir\b02ir08.jpg');
>> img2 = imread('icr\b02icr08.jpg');
>> ncc = normxcorr2( img2(:,:,1), img1(:,:,1) );
>> figure,surf(ncc),shading interp,axis ij,view(3);
>> title('Ex. Correlacao'),ylabel('Altura'),xlabel('Largura');
```

O gráfico resultante do script acima pode ser visto na Figura 3.1. Mais exemplos de scripts do Matlab podem ser vistos no Apêndice B.

3.3.3 Correlação de Imagens

Este trabalho visa aplicar o algoritmo similar ao de (Jones et al., 1997), que utiliza NCC. Entretanto isso não se dá de forma direta para imagens coloridas. A técnica de NCC que está disponível na bibliografia e na implementação do Matlab apenas trata de matrizes bidimensionais, ou seja, imagens do tipo *grayscale*, formadas por um único componente. Para tratar uma imagem colorida, precisamos

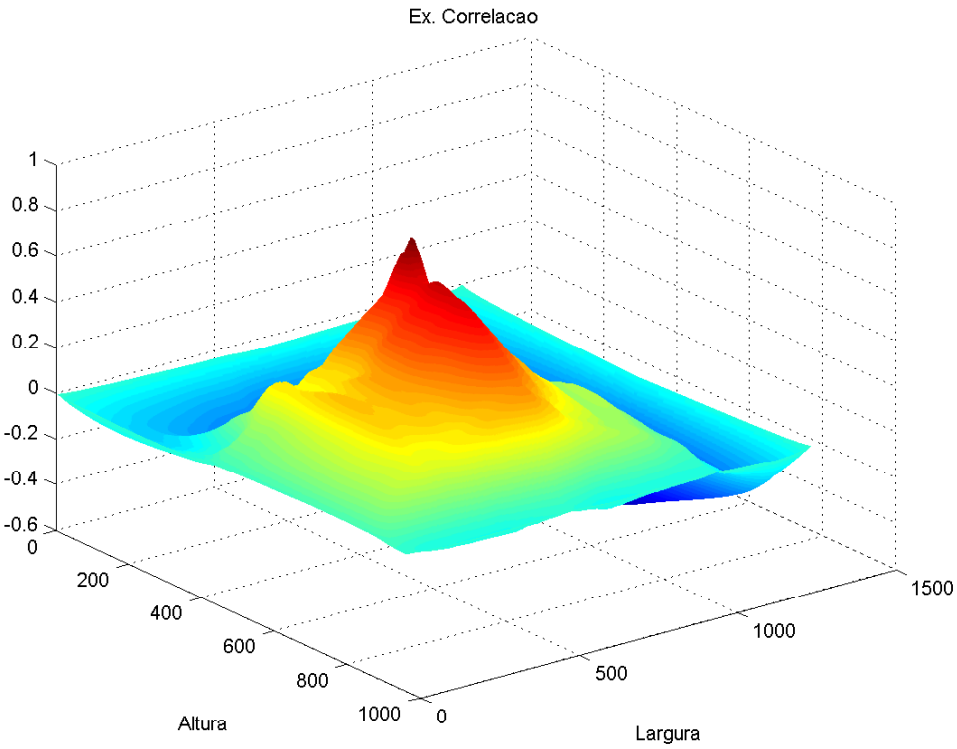


Figura 3.1: Exemplo de gráfico gerado pelo Matlab.

de uma técnica de NCC que trabalhe com matrizes bidimensionais com múltiplos planos, onde cada plano da matriz representa uma das componentes do espaço de cores RGB.

A seguir serão discutidas quatro abordagens de como a correlação pode ser feita a fim de facilitar a identificação da posição de onde ocorreu um *matching*. As quatro abordagens de correlação são:

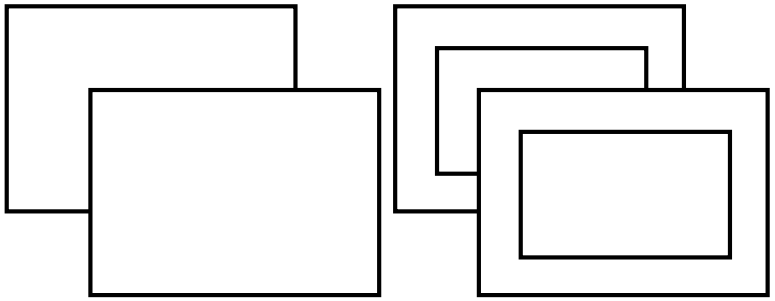
- Por força bruta;
- Entre janelas de imagens;
- Usando um fragmento de imagem (subimagem);
- Usando fragmentos de imagem dispostos em forma de *grid*.

Uma das primeiras tentativas de aplição da NCC foi através de força bruta (Figura 3.2(a)), isto é, fazer a correlação (um componente RGB por vez) entre duas imagens 640x480 pixels, sendo uma a ICR (Imagem Capturada pelo Robô) e a outra a IR (Imagem de Referência), respectivamente. Nesta abordagem a imagem inteira é usada no cálculo da correlação.

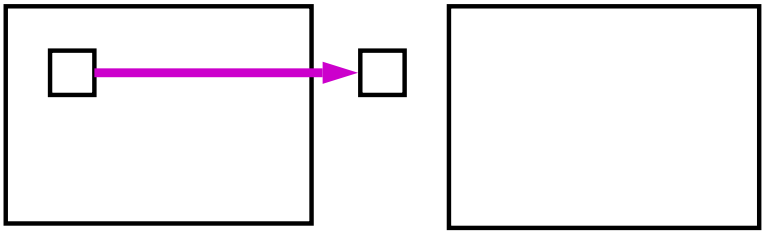
A abordagem seguinte, entre janelas, é similar ao trabalho de (Jones et al., 1997) no que diz respeito a área das imagens que serão submetidas ao algoritmo de NCC. Inicialmente, a correlação será feita entre duas janelas (Figura 3.2(b)), uma na imagem de exemplo (ICR) e outra na imagem de referência. Cada janela tem tamanho inferior ao tamanho original e está localizada de forma centralizada na imagem original.

Outra forma de utilzação da NCC foi utilizando-se um fragmento da imagem ou subimagem (Figura 3.2(c)) de exemplo e aplicar o algoritmo sobre este fragmento e uma imagem de referência. Este fragmento é deslizado sobre a imagem de referência, calculando-se a NCC em cada posição da janela para se encontrar o maior pico de correlação. É importante que este fragmento possua algumas características que possam contribuir na sua localização na imagem de referência. Atualmente o processo de escolha do fragmento é feito de forma manual. O objetivo desse experimento é tentar isolar as áreas de valor de correlação mais significativo dentro da imagem de referência.

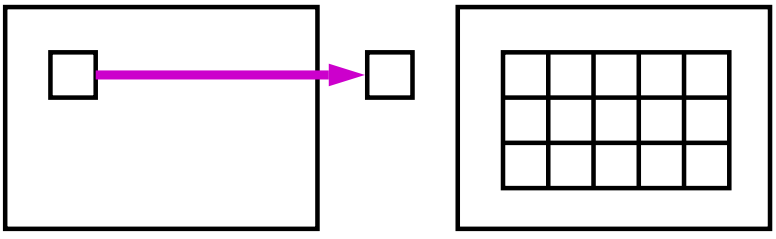
Um último experimento utilizou uma combinação das abordagens anteriores. Neste teste, foi utilizada uma janela na imagem de referência e esta foi subdividida em um formato de *grid* (Figura 3.2(d)). Cada uma das subdivisões do *grid* tem as dimensões do fragmento da imagem de exemplo. Este teste tem o objetivo de acelerar a realização dos cálculos da NCC e de tentar discretizar a área que será processada, tornando mais simples a decisão do posicionamento da imagem de exemplo sobre a imagem de referência.



(a) Força Bruta (b) Janelas



(c) Fragmento de Imagem



(d) *Grid*

Figura 3.2: Abordagens do uso de correlação.

A seguir, será mostrada uma visão geral de como foram elaboradas as abordagens descritas nos parágrafos anteriores. A correlação por força bruta tinha o propósito de encontrar o ponto de maior similaridade entre duas imagens inteiras. A partir da abordagem de força bruta, foi tentada uma abordagem utilizando-se janelas. Por ser uma área menor, o tempo de cálculo de correlação é menor e o local de melhor

correlação fica melhor delimitado. Tomando por base que a janela delimita melhor a posição de um pico de correlação, foi tentado o uso de um fragmento de imagem (subimagem). Com este tipo de abordagem, é possível determinar com uma precisão muito boa onde determinada parte de uma imagem está contida na outra (caso sejam iguais) ou é muito bem correlacionada (em imagem com algumas distorções). Na abordagem em que foi utilizado o *grid*, além de se tentar determinar a posição do ponto de maior correlação em regiões de tamanhos reduzidos e localizações fixas, também foi tentada a implementação de uma forma de mapear a posição de maior correlação para um possível comando a ser enviado para o robô.

Entre todas abordagens, a que se mostrou mais efetiva em testes preliminares foi a abordagem do fragmento de imagem (Figura 3.2(c)), com um desempenho muito bom, tanto no que diz respeito aos valores de correlação obtidos, quanto na localização da subimagem dentro de uma imagem de referência.

Os resultados dos experimentos utilizando as abordagens descritas acima podem ser vistos no Capítulo 4.

3.3.4 Quantização do Espaço RGB

Uma vez que não é possível aplicar diretamente a NCC em uma imagem colorida, foi analisada a possibilidade de se alterar a quantização do espaço de cores RGB de modo a se obter uma imagem que reduza o número de cores de modo a codificar em uma única componente as informações contidas no cubo RGB.

Esta imagem não é necessariamente "agradável" aos olhos, dado que queremos marcas bem características como, por exemplo, troca de tons mais acentuados, melhorando assim o desenho das bordas. Este processo também deveria tornar uniforme áreas da imagem onde exista uma grande quantidade de pequenas variações. Um ponto importante neste processo é a **existência da perda de informações**. A quantização pode tanto ajudar, quanto prejudicar, pois pode remover características importantes da imagem.

A redução da quantização do espaço de cores RGB foi feita de forma simples, onde o cubo RGB é subdivido em cubos menores, tendo dimensões de 3x3x3 subcubos, por exemplo. O objetivo disso é tentar agrupar cores próximas no espaço RGB. Após a subdivisão do cubo RGB, uma imagem é analisada e observamos em qual subcubo o ponto (cor) pertence. Para cada subcubo foi atribuído um intervalo de valores em uma escala de $[0, 255]$. Quando um ponto pertence a este subcubo, ele receberá um valor que representa uma faixa de valores dentro da escala de $[0, 255]$.

É importante ressaltar que a imagem não foi convertida para *grayscale*, e sim utilizada uma escala similar a de tons de cinza (*grayscale*). Esta codificação foi utilizada para gerarmos uma nova imagem, que é uma representação em uma única componente do espaço RGB.

Tendo imagens quantizadas em 8 bits, os experimentos descritos na Seção 3.3.3 puderam ser aplicados e os resultados averiguados. Entretanto, testes iniciais mostraram que esta abordagem forneceu resultados inferiores as demais abordagens. O que posteriormente foi atribuído ao tipo de algoritmo de quantização adotado que não favoreceu a uma adequada codificação da imagem. Assim, esta abordagem foi abandonada, podendo voltar a ser pesquisada em estudos futuros. O trabalho de (Sinclair, 1997) mostra uma alternativa à redução do número de cores em uma imagem, que poderia vir a ser considerada em trabalhos futuros.

3.4 Proposta de um Sistema de Navegação Visual

A partir dos resultados obtidos com os testes descritos na Seção 3.3.3, optou-se em desenvolver a abordagem em que foi usado um fragmento de imagem (subimagem) correlacionado contra uma imagem inteira. Além disso, para se averiguar e avaliar a posição atual do robô (ICR) contra as imagens de referência (IR) que compõem o banco de imagens (BI), foi feita uma análise vetorial. Esta análise vetorial consiste em mensurar o deslocamento dos vetores obtidos a partir da posição atual do robô (ICR), gerados a partir de uma determinada IR, e de como estes vetores deveriam estar orientados na IR. Além disto, deve-se destacar que são gerados sempre três vetores, representando o deslocamento relativo a cada um dos três componentes das imagens. O desenvolvimento dessa idéia resultou a proposta mostrada na Seção 3.4.1.

3.4.1 Vetores de Correlação em Imagens Coloridas

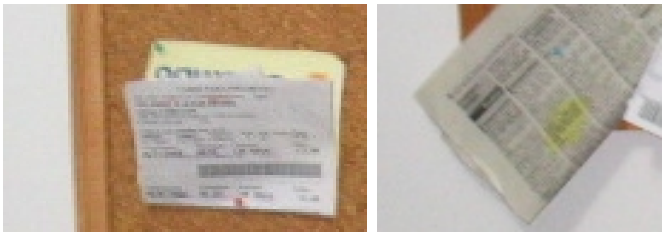
Para determinar uma possível mudança na trajetória do robô, precisamos definir uma forma de informar ao robô sobre o seu desvio em relação as imagens de referência para que este corrija sua trajetória. Desta forma, precisamos definir um mecanismo para que o robô possa determinar seu deslocamento relativo a um ou mais pontos de referência sendo utilizados em um dado momento.

Para que possamos detalhar o método proposto neste artigo, precisamos definir alguns termos e conceitos utilizados neste trabalho. O conjunto de imagens que o robô utilizará como referência para seguir uma determinada trajetória é chamado de banco de imagens (BI). Cada entrada do BI contém uma imagem de referência (IR_i) e os pontos de referência ($P_{ref_{ij}}$), que são subimagens de IR_i. O BI pode possuir n entradas, onde i varia de $1..n$. Uma entrada i do BI pode possuir m subimagens de IR_i ($P_{ref_{ij}}$), onde j varia de $1..m$. As imagens que o robô captura são denominadas ICR – Imagem Capturada pelo Robô. O referencial usado para determinar os deslocamentos, tanto nas ICR quanto nas IR_i será o ponto central de cada imagem (C_x, C_y). Este ponto também é chamado de P_{alvo} quando estamos nos referenciando a uma ICR. O ponto (C_x, C_y) é utilizado como referência por que a correlação entre duas imagens idênticas tem seu valor máximo neste ponto. Para determinar (C_x, C_y), assumimos que a imagem tem x colunas por y linhas, sendo o ponto central dado por $(\frac{x}{2}, \frac{y}{2})$. Quando nos referirmos a um componente de uma determinada imagem, ele será indexado pelo índice c. Por exemplo, ICR_c está referenciando o componente c da ICR sendo analisada. De posse das definições básicas, podemos definir como é atualizada a posição do robô em relação a uma IR_i do BI. Exemplos de ICR, IR_i e subimagens $P_{ref_{ij}}$ podem ser vistos na Figura 3.3.

O robô corrige sua trajetória seguindo o processo descrito a seguir. Primeiramente, precisamos de uma referência para guiar o robô. Assim, recuperamos do BI uma determinada entrada i. Essa entrada consiste da IR_i e uma ou mais subimagens $P_{ref_{ij}}$. Com estes dados, podemos calcular, para cada componente do espaço RGB, a posição de cada $P_{ref_{ij}}$ na IR_i. Ou seja, o quão deslocados estão os pontos de referência do ponto (C_x, C_y) da IR. Isso pode ser visto no algoritmo 2.

O algoritmo NCC (Martin and Crowley, 1995) obtém a partir de duas imagens (contendo apenas um componente) uma matriz M bidimensional com valores normalizados entre [-1,1] que indicam o grau de correlação entre as imagens. Esta matriz pode ser visualizada na Figura 3.4, onde percebe- se a existência de um pico

(a) ICR (b) IR_i



(c) $P_{ref_{i1}}$ (d) $P_{ref_{i2}}$

Figura 3.3: Exemplo de ICR, IR_i e $P_{ref_{ij}}$.

Entrada: Uma imagem de referência IR_i e um conjunto de subimagens $P_{ref_{ij}}$
Saída: Um vetor D_{ref} composto por x, y e valor máximo de correlação

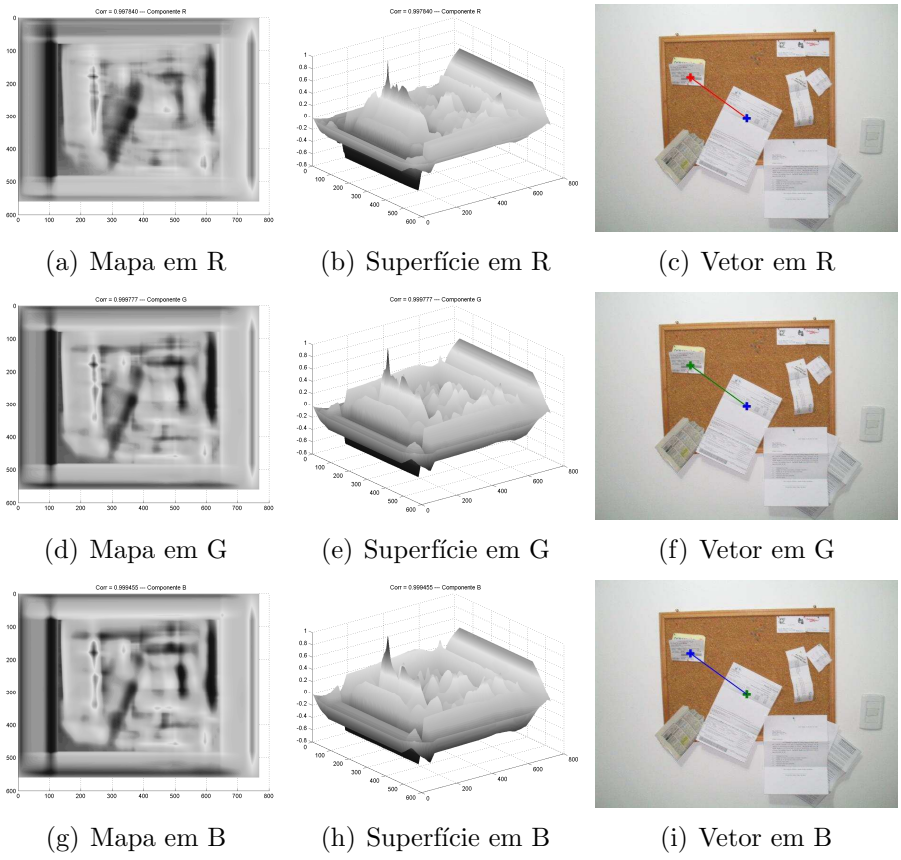
$i \leftarrow$ Entrada Atual do BI ;
para todo $j \leftarrow 1$ ***até*** m **faça**
  para cada *Componente c de RGB* **faça**
    $M \leftarrow$ `NCC(`$P_{ref_{ij_c}}$, IR_{ic} `);`
    $D_{ref_j} \leftarrow$ `CalculaDeslocamento(`M `);`
  fim
fim

Algoritmo 2: Cálculo dos deslocamentos das subimagens $P_{ref_{ij}}$ em relação ao centro da imagem de referência IR_i

de correlação (máximo de correlação). Os valores da correlação são obtidos através do deslizamento de uma janela ($P_{ref_{ij_c}}$) sobre uma imagem de referência (IR_{ic}) e, para cada posição da janela, é aplicado o algoritmo NCC.

O vetor D_{ref} é composto da posição (par de coordenadas X,Y) onde a correlação alcançou seu valor máximo e do valor deste pico de correlação, isso para cada componente do espaço RGB. As posições dos picos de correlação são os deslocamentos das subimagens $P_{ref_{ij}}$ relativos ao ponto (C_x, C_y) da IR. As posições dos picos de correlação encontrados na matriz M devem ser mapeadas para a IR_i. Isso é feito através da rotina `calculaDeslocamento()`, onde as dimensões da matriz M são mapeadas para as dimensões de IR_i. Feito isto, deve ser então realizado o procedimento equivalente para a ICR, conforme visto no algoritmo 3.

Assim como D_{ref}, D_{alvo} possui os deslocamentos das subimagens $P_{ref_{ij}}$ relativos a P_{alvo} e o valor do pico de correlação. Os valores obtidos podem ser vistos na Figura 3.5. Com estes valores, D_{ref} e D_{alvo}, podemos gerar vetores para avaliarmos



(a) Mapa em R (b) Superfície em R (c) Vetor em R

(d) Mapa em G (e) Superfície em G (f) Vetor em G

(g) Mapa em B (h) Superfície em B (i) Vetor em B

Figura 3.4: Mapa de correlação, superfície de correlação e localização da subimagem $P_{ref_{i1}}$ em IR_i (Figuras 3.3a e 3.3b).

o deslocamento relativo entre a ICR e a IR_i para cada componente da imagem.

Os vetores obtidos são traçados na IR e na ICR a partir de P_{alvo}, conforme pode ser visto nas Figuras 3.4 e 3.5 respectivamente. A partir dos resultados obtidos, são traçados os vetores referentes ao processamento da subimagem $P_{ref_{i1}}$, conforme é mostrado na Figura 3.6. Com isso, podemos fazer uma análise para determinar se os valores de cada um dos três componentes (RGB) da imagem obtidos estão convergindo, ou seja, apontam para a mesma direção. Para ilustrar o conceito de convergência dos vetores, é mostrado na Figura 3.7 como seria a convergência de três vetores. Para determinar se existe a convergência ou não, foi utilizada a heurística descrita a seguir.

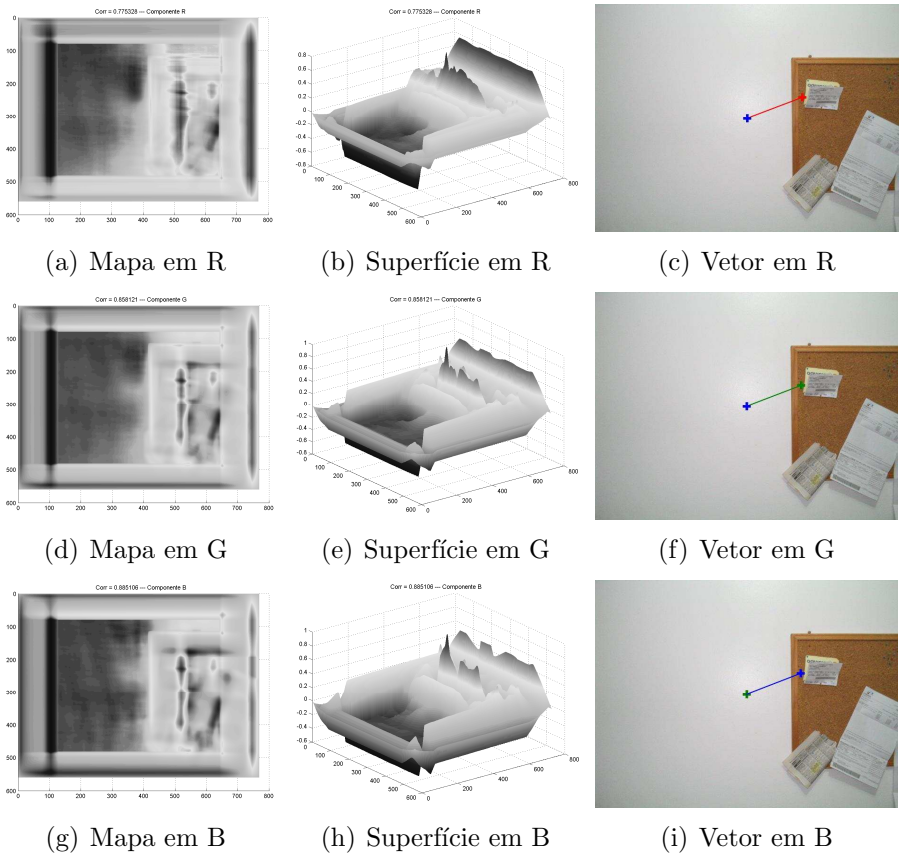
Definimos uma distância limite $Limite_V$ entre os vetores V_r, V_g e V_b, gerados a partir de uma entrada j de D_{ref_j} ou D_{alvoj} e estes não podem violar tal limite. É calculada a distância euclidiana entre os vetores V_r, V_g e V_b e caso algum não satisfaça o $Limite_V$, o *matching* é rejeitado. Isto pode ser visto nas Equações 3.1, 3.2 e 3.3.

$$B_1 = \sqrt{(V_{rx} - V_{g_x})^2 + (V_{ry} - V_{g_y})^2} \leq 2.Limite_V \tag{3.1}$$

$$B_2 = \sqrt{(V_{rx} - V_{bx})^2 + (V_{ry} - V_{by})^2} \leq 2.Limite_V \tag{3.2}$$

$$B_3 = \sqrt{(V_{g_x} - V_{bx})^2 + (V_{g_y} - V_{by})^2} \leq 2.Limite_V \tag{3.3}$$

Se todos valores booleanos B_1, B_2 e B_3 forem verdadeiros, o *matching* é aceito,



(a) Mapa em R (b) Superfície em R (c) Vetor em R

(d) Mapa em G (e) Superfície em G (f) Vetor em G

(g) Mapa em B (h) Superfície em B (i) Vetor em B

Figura 3.5: Mapa de correlação, superfície de correlação e localização da subimagem $P_{ref_{i1}}$ em ICR_i.

caso contrário é rejeitado. Um exemplo de rejeição pode ser visto na Figura 3.8, onde os vetores obtidos a partir da subimagem $P_{ref_{i2}}$ começam a divergir quando a ICR está muito diferente da IR_i sendo processada.

Acima foi discutido o limite $Limite_V$, relacionado a posição. Existe também um segundo limite, o $Limite_C$, que determina a qualidade da correlação. O $Limite_C$ é um limiar aplicado sobre o valor de correlação obtido, pois cada ponto da imagem terá um valor associado e este é normalizado entre $[-1, 1]$. Se o valor de correlação de um determinado vetor estiver abaixo de $Limite_C$, o *matching* também é rejeitado.

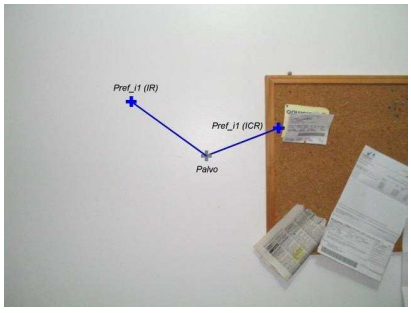


Figura 3.6: Vetores resultantes do processamento de $P_{ref_{i1}}$.

Entrada: Uma imagem de referência ICR e um conjunto de subimagens $P_{ref_{ij}}$

Saída: Um vetor D_{alvo} composto por x, y e valor máximo de correlação

$i \leftarrow$ Entrada Atual do BI ;
para todo $j \leftarrow 1$ ***até*** m **faça**
  para cada *Componente c de RGB* **faça**
    $M \leftarrow$ `NCC(`$P_{ref_{ijc}}$, ICR_{ic} `);`
    $D_{alvoj} \leftarrow$ `CalculaDeslocamento(`M `);`
  fim
fim

Algoritmo 3: Cálculo dos deslocamentos das subimagens $P_{ref_{ij}}$ em relação ao P_{alvo}

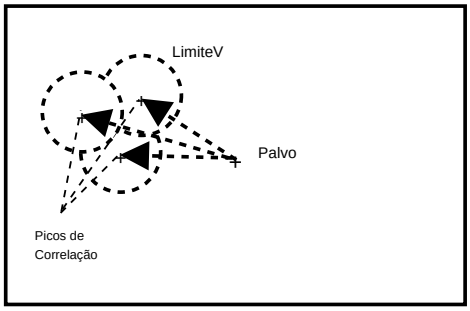


Figura 3.7: Exemplo hipotético de convergência de vetores.

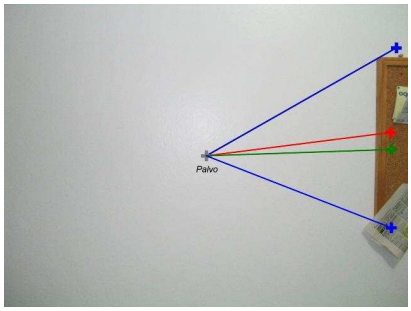


Figura 3.8: Exemplo de divergência de vetores.

4 EXPERIMENTOS

Neste capítulo serão vistos os resultados dos experimentos realizados durante o desenvolvimento deste trabalho. Todos experimentos descritos aqui são o resultado da aplicação das abordagens vistas no capítulo 3. Serão vistos os resultados dos testes de como se aplicar a correlação em imagens coloridas, além do resultado do experimento de navegação baseando unicamente em uma seqüência de imagens.

Recapitulando o que foi visto no capítulo 3, os resultados que serão descritos nas seções seguintes utilizaram o ambiente descrito a seguir. Todas imagens possuem 640x480 pixels de resolução e qualidade máxima permitida pela câmera utilizada. A rotina de correlação utilizada foi fornecida pelo *Image Processing toolbox* do Matlab. A versão do Matlab utilizada foi a 6.1 *Release* 12. O resultado da função de correlação (`normxcorr2`) tem como retorno uma matriz bidimensional, como visto na Figura 4.1. Os testes de correlação com imagens coloridas realizados na Seção 4.1 utilizou as imagens apresentadas nas Figuras 4.2(a), 4.2(b) e 4.2(c).

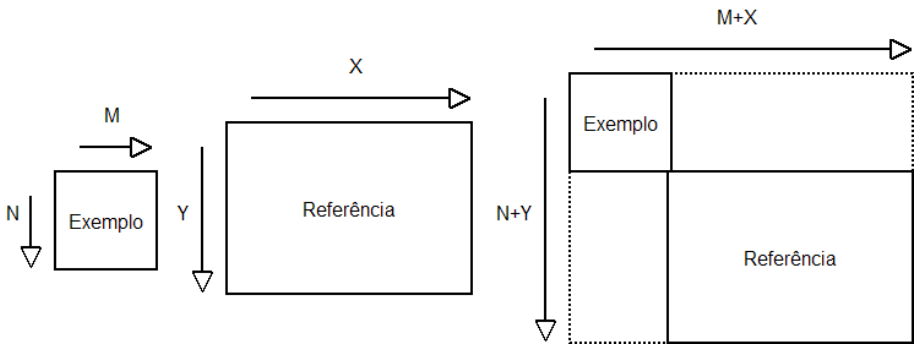


Figura 4.1: Dimensão da matriz retornada pela função `normxcorr2`.

4.1 Testes de Aplicação da Correlação

A seguir serão apresentados os resultados obtidos com as abordagens discutidas anteriormente no capítulo 3. A sequência de todas operações foi calcular a correlação dos componentes RGB entre as imagens ICR e IR_1 e ICR e IR_2. O conjunto ICR-IR que apresentar o valor mais elevado (pico) de correlação será a imagem *current*, e a qual o robô tomará como base da tomada de suas decisões. Os gráficos de correlação exibidos nos testes representam a correlação do componente vermelho (R) do espaço de cores RGB.

(a) IR_1

(b) IR_2

(c) ICR

Figura 4.2: IRs pertencentes ao BI. A imagem IR_1 é anterior à imagem IR_2 dentro da sequência do BI. A imagem ICR está localizada entre as imagens IR_1 e IR_2.

4.1.1 Correlação por Força Bruta

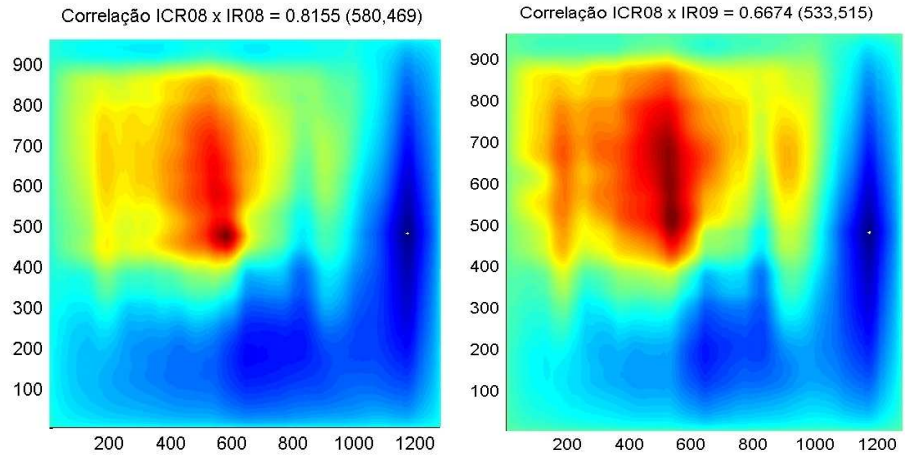
A aplicação da correlação através de força bruta obteve o segundo melhor resultado em termos de valor (pico) de correlação entre as abordagens apresentadas. Os gráficos apresentados nas Figuras 4.3(a) e 4.3(b) mostram os valores de obtidos. A região mais escura significa maior valor de correlação.

Ambos gráficos apresentam regiões de máximo similares. Entretando, o gráfico da Figura 4.3(a) apresenta um valor de correlação melhor do que apresentado no gráfico da Figura 4.3(b). O valor de correlação está indicado acima do gráfico, junto com as coordenadas do pico de correlação. Desta forma, a imagem IR_1 (Figura 4.2(a)) foi determinada como imagem *current*. Fazendo o mapeamento das dimensões da matriz de correlação retornado pela função `normxcorr2` para o tamanho da imagem *current*, onde pode ser vista na Figura 4.3(c) assinalada com um círculo a região que possui o pico de correlação obtido.

4.1.2 Correlação entre Janelas

A correlação entre janelas apresentou o pior resultado dentre todas abordagens. O tamanho das janelas foi definido como 400x400 pixels, centralizadas em cada imagem. Os gráficos resultantes dos cálculos da correlação são mostrados nas Figuras 4.4(a), 4.4(b) e 4.4(c).

podemos observar novamente que os gráficos resultantes são similares. Neste caso, a melhor correlação entre a ICR e as imagens de teste ocorreu na imagem IR_2 (Figura 4.2(b)).



(a) IR_1 (b) IR_2

(c) Região de maior correlação

Figura 4.3: Resultado da correlação por Força Bruta.

4.1.3 Correlação Usando um Fragmento de Imagem

De todas abordagens utilizadas, esta foi a que apresentou os melhores resultados de posicionamento e valores de correlação.

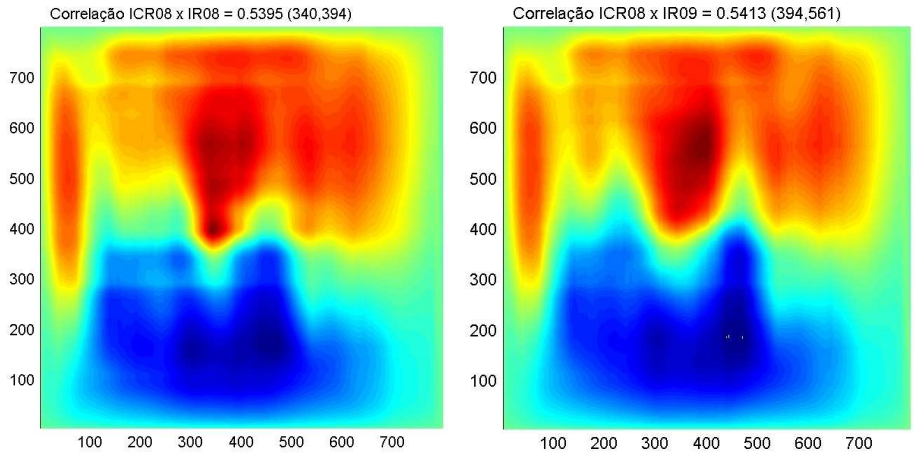
O fragmento de imagem foi escolhido de forma manual. Isso possivelmente influenciou o resultado obtido nesta abordagem. O fragmento escolhido é mostrado na Figura 4.5(c), assim como os gráficos de correlação.

Neste exemplo, podemos ver que no gráfico da Figura 4.5(a) foi obtido quase que o valor máximo da correlação (0.9818). Apesar do gráfico da Figura 4.5(b) não ter sido escolhido, também pode ser visto que ele foi bem correlacionado, dado as distorções entre a ICR (inteira) e a IR_2.

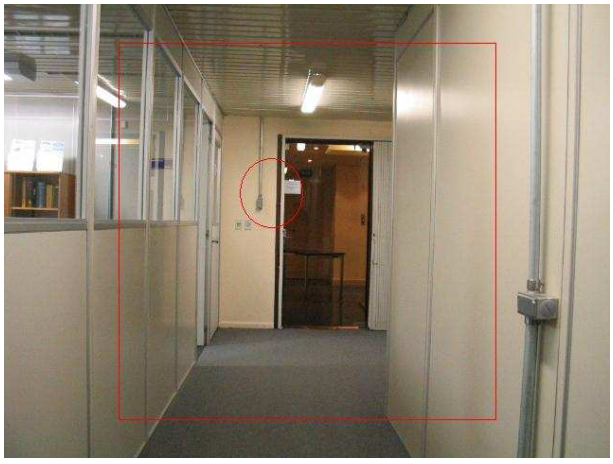
4.1.4 Correlação em um *grid*

A abordagem em forma de *grid* apresentou o segundo pior desempenho. Aqui, as IRs foram divididas em um *grid* 3x3. O mesmo fragmento de imagem da Seção 4.1.3 foi utilizado neste experimento. O *grid* pode ser ilustrado na Figura 4.6.

Cada elemento do *grid* possui o mesmo tamanho do fragmento da imagem. Após aplicarmos a NCC entre o fragmento e cada um dos itens do *grid*, obtemos as seguintes matrizes com os valores da correlação:



(a) IR_1 (b) IR_2



(c) Região de maior correlação, com Janela limitante

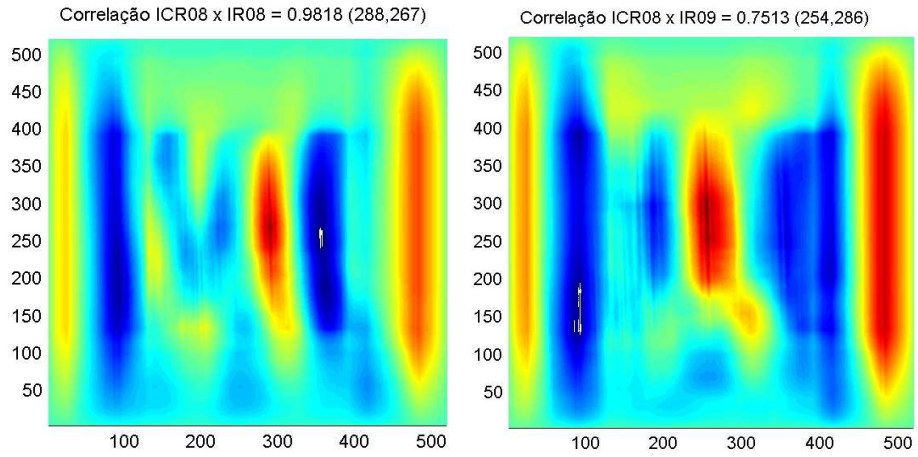
Figura 4.4: Resultado da correlação por Janelas.

$$\begin{pmatrix} 0.6168 & 0.5979 & 0.6273 \\ 0.6391 & 0.6134 & \mathbf{0.6436} \\ 0.6323 & 0.6278 & 0.6428 \end{pmatrix} \begin{pmatrix} 0.5979 & 0.5949 & 0.6445 \\ 0.6439 & \mathbf{0.6474} & 0.6445 \\ 0.6232 & 0.6019 & 0.6451 \end{pmatrix}$$

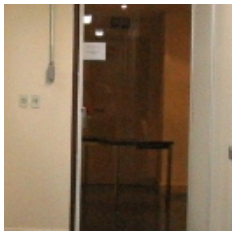
A matriz da esquerda é referente à imagem IR_1 e a da direita é referente à imagem IR_2, Figuras 4.2(a) e 4.2(b) respectivamente. Os itens em negrito mostram os maiores valores de cada tabela. Neste caso escolheríamos a matriz da direita, visto que o maior valor de todos é 0.6474. O *grid* deve ser interpretado da seguinte forma: se o maior valor aparecer na coluna da direita ou esquerda, o robô deve virar para a respectiva direção. Se for na coluna central, de prosseguir a trajetória atual. No exemplo acima, o robô deveria avançar, já que não foi indicada nenhuma correção na direção.

4.1.5 Visão Geral das Abordagens

De todas as abordagens pesquisadas, a que melhor se destacou foi a que utilizou o *matching* entre um fragmento de imagem e uma imagem inteira. O resultado desta abordagem mostrou bons resultados em termos de valor de correlação e localização do *matching*. Desta forma, a abordagem do uso de fragmentos de imagens foi



(a) IR_1 (b) IR_2



(c) Fragmento da ICR escolhida

Figura 4.5: Resultado da correlação usando um fragmento de imagem.

adotada, sendo utilizada posteriormente no algoritmo de navegação visual proposto.

As demais abordagens, força bruta, entre janelas e em *grid* mostraram-se inferiores, sendo assim abandonadas. A correlação por força bruta mostrou-se cara demais computacionalmente. A correlação entre janelas forneceu resultados de valor de correlação baixos. A abordagem em *grid* poderia fornecer resultados que poderiam colocar em dúvida qual ação deveria ser tomada.

4.2 Vetores de Correlação em Imagens Coloridas

A técnica mostrada na Seção 3.4.1 foi validada através de simulações de um robô móvel, com a aquisição de imagens reais. O algoritmo empregado visa avaliar a técnica desenvolvida nesse trabalho de uma forma qualitativa.

Para realizar a simulação, o seguinte critério foi adotado: deve-se gerar um banco de imagens (BI) em ambientes internos (salas, escritórios, laboratórios, etc) em que todas as fotos possuam uma iluminação semelhante. Outro ponto importante a ressaltar é que este algoritmo não se propõe a desviar de obstáculos imprevistos. O algoritmo também assume que o robô está numa posição que gere uma ICR semelhante a primeira entrada do BI.

O algoritmo utilizado para a simulação realiza a seguinte seqüência de passos, conforme descrito a seguir.

Inicialmente, o robô captura uma imagem (ICR) e, de posse dela, realizará o cálculo de correlação das subimagens $P_{ref_{ij}}$ (fragmentos) contra a IR_i e também contra a ICR. Com os valores dos vetores D_{ref} e D_{alvo} preenchidos, devemos verificar a qualidade da correlação obtida através do limite $Limite_C$. Caso algum valor de

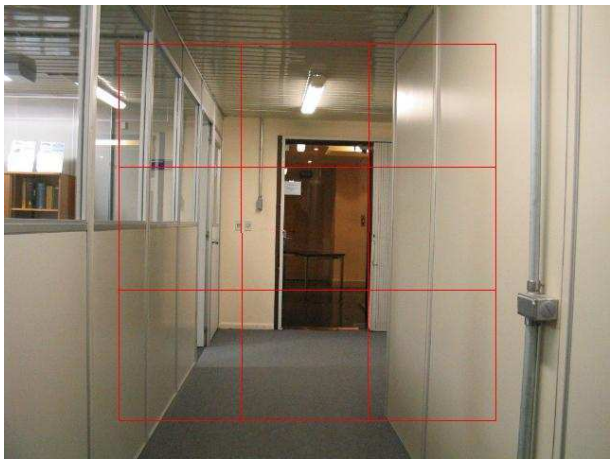


Figura 4.6: Exemplo de *grid*.

correlação esteja abaixo de $Limite_C$, a entrada i do BI é incrementada em 1. Esse procedimento localiza dentro do BI o que o robô "está vendo". Se o critério $Limite_C$ for satisfeito, deve ser feita a análise dos vetores obtidos para determinar se existe convergência entre eles.

A verificação do critério de convergência é feita através da análise dos vetores V_r, V_g e V_b obtidos de uma subimagem $P_{ref_{ij}}$. Se os vetores de cada componente RGB satisfizerem as Equações 3.1, 3.2 e 3.3, existe então a convergência. Isso deve ser repetido para todos os vetores obtidos (armazenados em D_{ref} e D_{alvo}). Caso o conjunto de vetores V_r, V_g ou V_b não satisfaça as equações, a entrada i do BI também é incrementada, voltando ao passo descrito no parágrafo anterior. Se o incremento do índice i ocorrer múltiplas vezes, sem no entanto o robô se deslocar, então podemos considerar que o robô não consegue mais manter o controle de sua posição e então a execução do algoritmo é terminada.

Para a geração do comando enviado ao robô, é feita a análise da diferença entre o vetor de uma $P_{ref_{ij}}$ obtido na IR e do vetor obtido com a mesma $P_{ref_{ij}}$ na ICR. Os comandos são enviados ao robô para que ele tente se posicionar de modo que os vetores das subimagens $P_{ref_{ij}}$ encontrados na ICR se movimentem até que se aproximem dos vetores obtidos com as mesmas subimagens na IR, ou seja, aproximando deste modo P_{alvo} da ICR do (C_x, C_y) da imagem IR_i. O robô terá conseguido se posicionar se os vetores obtidos na IR e na ICR satifizerem o limite $Limite_V$. Conseguindo isso, o robô pode ser considerado posicionado com uma orientação adequada, pois está na trajetória correta e portanto está apto a avançar, mudando para a próxima imagem de referência.

Foi feito um experimento onde foram obtidas imagens através de uma câmera digital, onde foi contruído um banco de imagens definindo um trajetória. Este BI pode ser visto no Apêndice A e é composto de 30 imagens de referência e, para cada referência, foram escolhidas duas $P_{ref_{ij}}$ que serviram de referência para a navegação. Após a criação do BI de navegação, a câmera foi colocada em uma posição similar ao início do BI. Uma ICR foi obtida e então repassada ao algoritmo. A implementação do algoritmo, feita em Matlab, pode ser vista no Apêndice B. Após o processamento da ICR, o algoritmo forneceu uma indicação de qual movimento o robô deveria executar. Com este resultado, a câmera foi reposicionada manualmente no local indicado e então uma nova ICR foi adquirida, repetindo este processo outras vezes.

Dado a dificuldade de posicionar adequadamente o tripé da câmera em certos trechos do caminho percorrido devido a limitações de espaço, possivelmente foi introduzido um ruído nas ICRs processadas. Devido a forma como a simulação da navegação foi feita, um passo extra que foi incluído no algoritmo, onde foi feito o incremento da entrada i do BI após dois comandos de "avançar" enviados ao robô. O objetivo deste passo extra foi minimizar as distorções entre a ICR e as imagens de referência. Possivelmente este passo extra no algoritmo não seria necessário caso o *hardware* adequado estivesse disponível. Apesar das limitações, com este experimento foi possível simular a navegação até o destino, demonstrando o funcionamento adequado do método proposto. A seqüência completa das imagens que o robô capturou durante a navegação podem ser vistas na Figura 4.7.

Nas Figuras 4.8, 4.9 e 4.10, podemos ver alguns exemplos das referências utilizadas, a imagem que o robô capturava em um dado momento e a indicação que o algoritmo forneceu.

Como pode ser visto, o método proposto na Seção 3.4.1 obteve resultados positivos. Isto demonstra que é possível manter a posição do robô através do uso de uma seqüência de imagens coloridas. Esta seqüência de imagens, juntamente com o *matching* através de correlação, torna possível a localização de pontos de referência, e assim conseguir estimar a posição atual do robô.

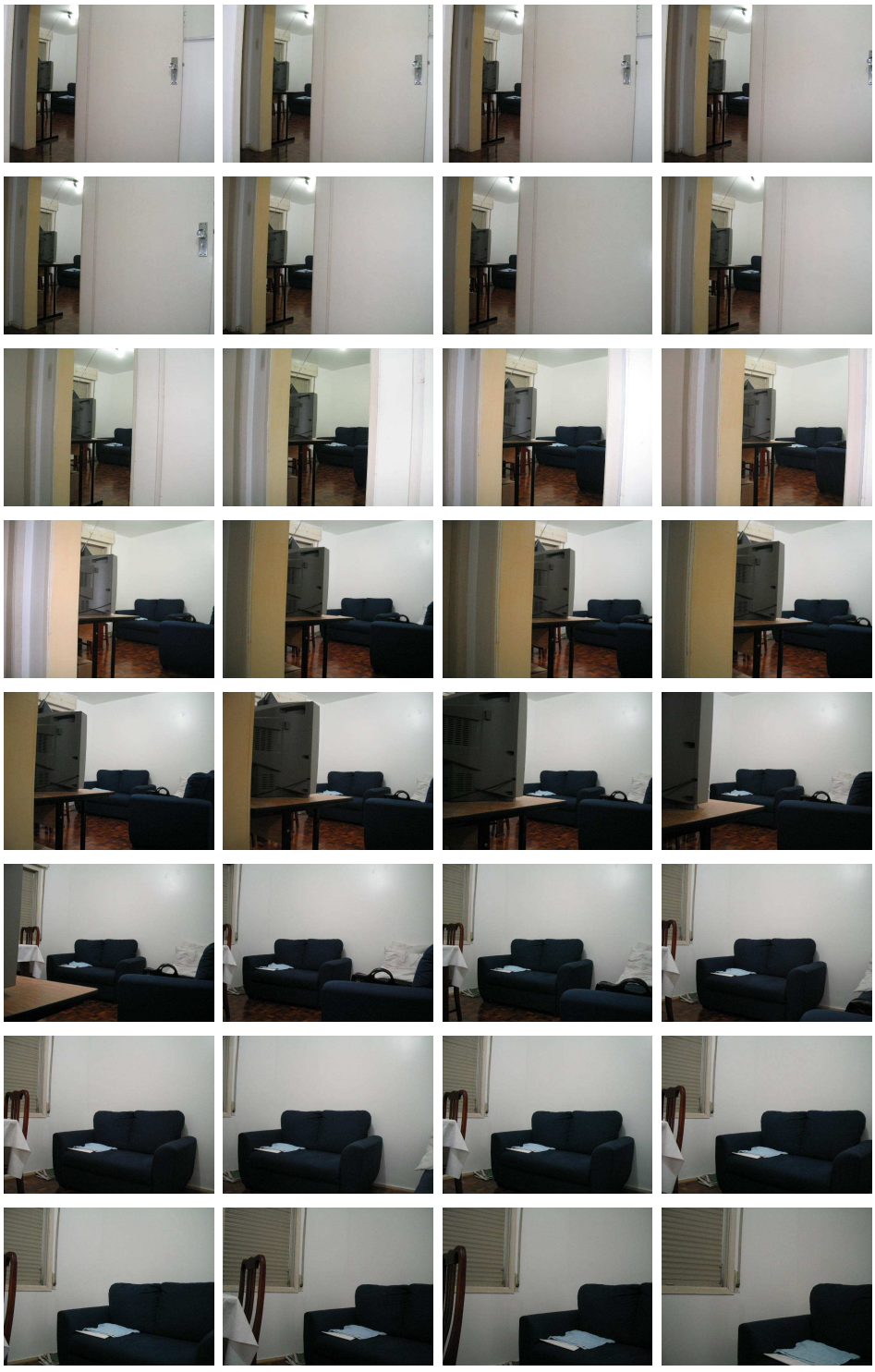


Figura 4.7: Seqüência completa das imagens da navegação.

(a) IR (b) ICR (c) $P_{ref_{i1}}$ (d) $P_{ref_{i2}}$

Figura 4.8: Nestas imagens, o algoritmo de navegação indicou que o robô deveria virar à direta.

(a) IR (b) ICR (c) $P_{ref_{i1}}$ (d) $P_{ref_{i2}}$

Figura 4.9: Nestas imagens, o algoritmo de navegação indicou que o robô deveria avançar.

(a) IR (b) ICR (c) $P_{ref_{i1}}$ (d) $P_{ref_{i2}}$

Figura 4.10: Nestas imagens, o algoritmo de navegação indicou que o robô deveria virar à esquerda.

5 CONCLUSÃO

Neste trabalho foram vistos os principais pontos relacionados no desenvolvimento de um sistema de navegação visual para controle de robôs autônomos.

Foram vistos os principais conceitos envolvidos, relacionados à robótica e a processamento de imagens pertinentes ao *matching* entre imagens. Também foram vistos alguns trabalhos relacionados ao controle de robôs através de *feedback* visual, usando-se uma câmera acoplada no robô.

Também foram vistas algumas abordagens de como se empregar o *matching* entre imagens coloridas utilizando *normalized cross-correlation*, o que resultou num método de *matching* capaz de fornecer informações suficientes para se realizar a navegação de um robô unicamente por uma seqüência de imagens que descrevem um caminho que o robô deverá percorrer.

Os experimentos realizados neste trabalho (Righes and Osório, 2004) apresentaram bons resultados. Isso demonstra que a navegação visual pode ter aplicações no mundo real. Entre as principais contribuições deste trabalho podemos destacar:

- O uso de cores, até então uma fonte valiosa de informação ainda pouco explorada;
- O uso de vetores e subimagens como pontos de referência;
- A proposta de um algoritmo completo para a navegação visual, adotando critérios de limiar ($Limite_V$ e $Limite_C$).

A análise da correlação através de vetores e pontos de referência pré-selecionados pode tornar a navegação de um robô autônomo mais robusta, no que diz respeito inclusive à modificações no ambiente e oclusões.

Entre os problemas enfrentados, o maior deles foi a dificuldade de se propor uma solução de navegação visual, já que não existe uma abordagem de concenso na área de visão computacional e de como utilizar tais técnicas na robótica móvel. Outro grande problema enfrentado foi em relação à iluminação das fotos, o que influi diretamente nos resultados obtidos.

5.1 Trabalhos Futuros

As perspectivas para trabalhos futuros apontam para o tratamento de problemas relativos a disparidades muito grandes na iluminação das fotos. Também há a perspectiva da análise de outros picos de correlação distintos do pico máximo obtido. Existe a possibilidade de realização de testes em outros espaços de cores, como o HSI

ou HSV. E, por fim, a realização de estudos a fim de selecionar de forma automática subimagens $P_{ref_{ij}}$ que sejam representativas para a tarefa de navegação.

Também está sendo analisada a possibilidade de se associar mais informações ao processo de navegação. Por exemplo, manter um registro de movimentos feitos pelo robô entre uma foto e outra durante a construção do BI através do uso de odômetros implementados com sensores do tipo *encoder*. O uso de um odômetro seria aplicado tanto na geração do BI quanto durante a navegação do robô. Com isto, podemos ter uma melhor estimativa da localização atual do robô e de quanto ele precisa se deslocar. Esta última proposta permitiria criar um processo de localização e navegação misto usando diferentes sensores para estimar a posição do robô ao mesmo tempo que a visão contribuiria para corrigir erros de posicionamento e deslocamento.

APÊNDICE A BANCOS DE IMAGENS

Na Figura A.1, podemos ver um exemplo de banco de imagens gerado. Nas demais seções deste capítulo, serão vistos os bancos de imagens utilizados nos testes e experimentos deste trabalho.

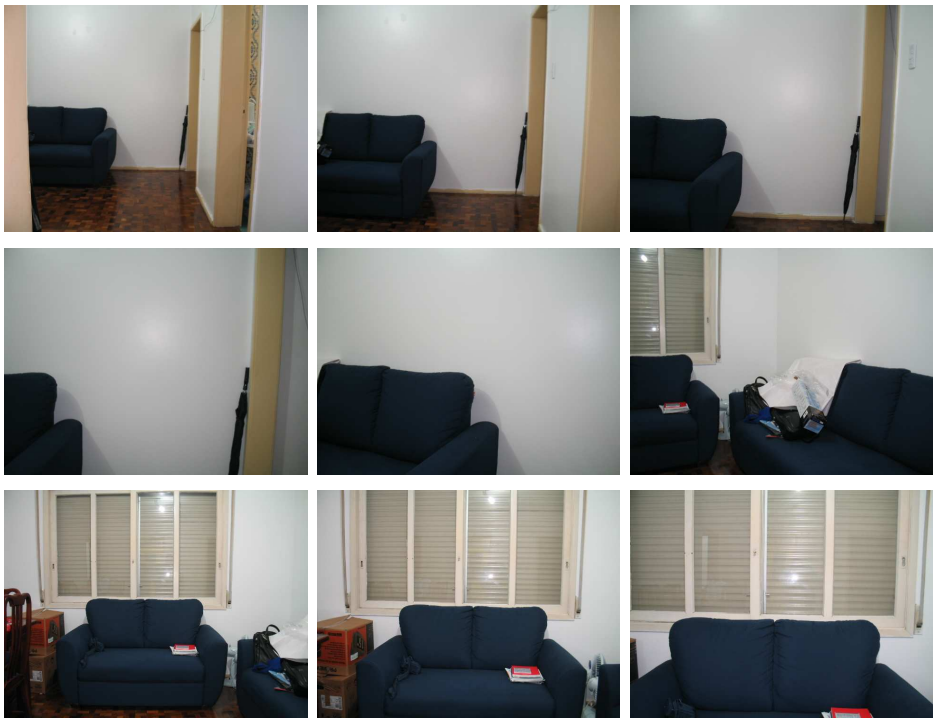


Figura A.1: Exemplo de BI gerado.

A.1 BI para Testes de Correlação

O BI mostrado na Figura A.2 foi utilizado nos testes iniciais deste trabalho para se avaliar como seria utilizado a correlação para se fazer o *matching* em imagens coloridas.

A.2 BI para Testes de Navegação

Para o experimento de navegação, utilizando a abordagem descrita na Seção 3.4.1, foi utilizado o BI mostrado na Figura A.3.

Figura A.2: BI utilizado para os testes iniciais de correlação.

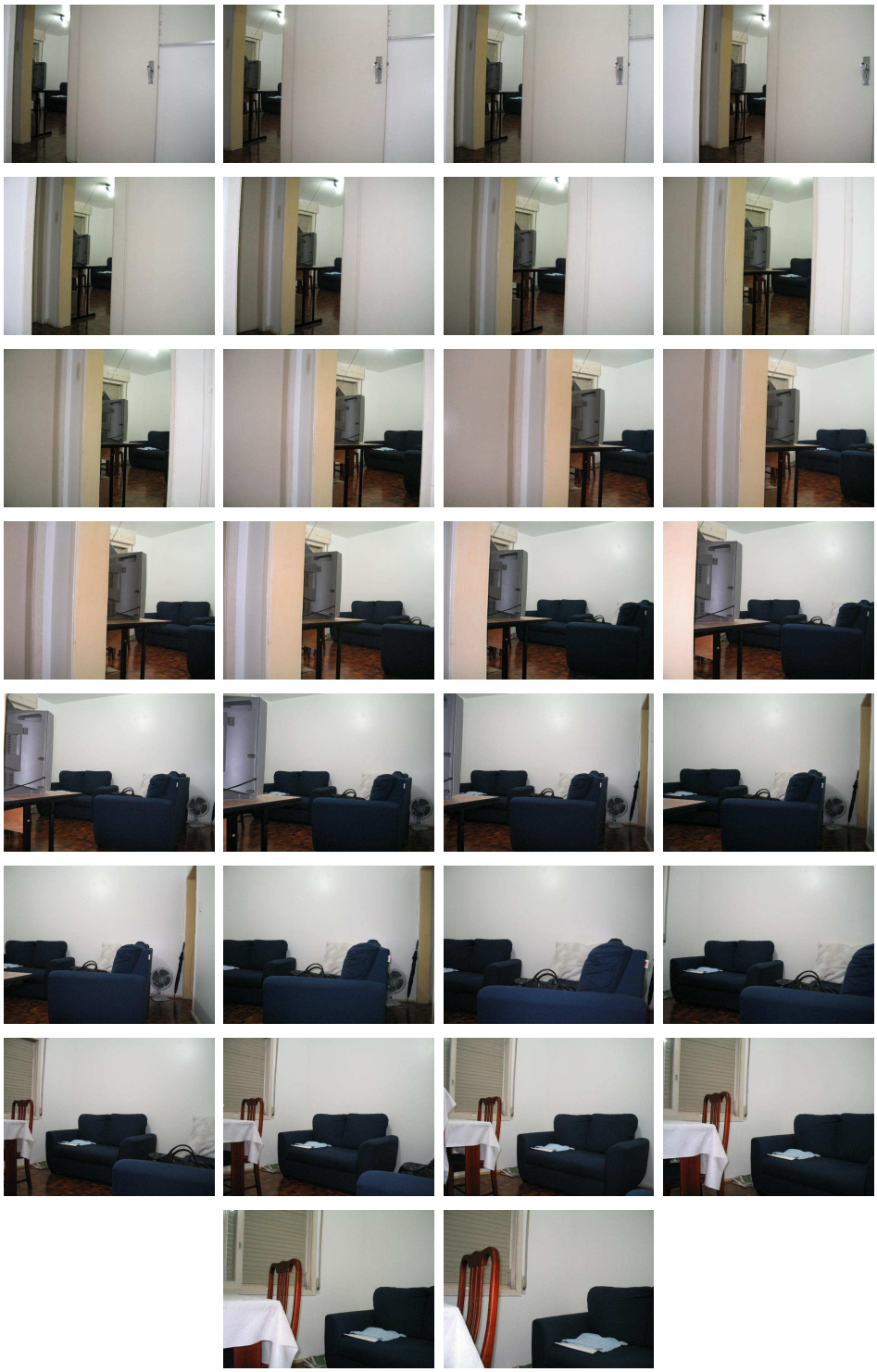


Figura A.3: BI utilizado para os testes iniciais de correlação.

APÊNDICE B SCRIPTS MATLAB

B.1 Scripts Principais

B.1.1 Script robotNavigation

```
function [ robotCmd, lastIndex ] = robotNavigation( imageBaseDir, ...
        imageBaseStartIndex, currentRobotImageFile )
%
% Parametros da Funcao
% --------------------
%
% imageBaseDir:
%       Caminho que o robo deve seguir. diretorio que contem
%       as imagens no formato imgrefxxxx.jpg
%
% imageBaseStartIndex:
%       Indice inicial das imagens de referencia.
%
% currentRobotImageFile:
%       Imagem que o robo capturou. baseado nela sera gerado um
%       comando.
%
% Retorno da Funcao
% -----------------
%
% robotCmd:
%       Comando que o robo deve executar.
%
% lastIndex:
%       Ultimo indice do banco de imagens utilizado.
%

  %
  % Parametros do Algoritmo
  %
  IMG_ROWS = 480; % altura das imagens, em pixels
  IMG_COLS = 640; % largura das imagens, em pixels
  % Valor minimo necessario para caracterizar um match.
  % Varia de [0,1].
  THRESHOLD_CORRELATION = 0.7;
  % Distancia minima entre os picos de correlacao para
  % que "concordem". Em pixels.
  THRESHOLD_VECTOR_DISTANCE = 50;
  % Valor para definir a regiao central da imagem de
  % referencia. A regiao vai de
  % (IMG_COLS/2)-THRESHOLD_DELTA_IMG_CENTER ate
  % (IMG_COLS/2)+THRESHOLD_DELTA_IMG_CENTER. Em pixels.
  THRESHOLD_DELTA_IMG_CENTER = 40;
  % Indice inicial das imagens de referencia.
  CURRENT_BASE_INDEX = imageBaseStartIndex;

  %
  % o algoritmo...
  %
  try
    lastIndex = CURRENT_BASE_INDEX;
```

```
robotCmd = -1; % Comando invalido.
% Para ter certeza que o diretorio existe
if ~ exist( imageBaseDir )
  formatedMsg = sprintf( 'Diretorio %s nao existe', ...
   imageBaseDir );
  disp( formatedMsg );
  return;
end
% Para ter certeza que o arquivo existe
if ~ exist( currentRobotImageFile )
  formatedMsg = sprintf( 'Arquivo %s nao existe',
   currentRobotImageFile );
  disp( formatedMsg );
  return;
end
currentRobotImage = imread( currentRobotImageFile );
% Para ter certeza que os arquivos existem
referenceImageFile  = sprintf( '%s\\imgref%04d.jpg', imageBaseDir, ...
 lastIndex );
referenceImageFile1 = sprintf( '%s\\imgref%04d_1.jpg', imageBaseDir, ...
 lastIndex );
referenceImageFile2 = sprintf( '%s\\imgref%04d_2.jpg', imageBaseDir, ...
 lastIndex );
hasAllReferences = ...
  ~exist( referenceImageFile)  & ...
  ~exist( referenceImageFile1) & ...
  ~exist( referenceImageFile2) ;
if hasAllReferences
  disp( 'Uma das imagens de referencia esta faltando' );
  return;
end
% Fazemos o loop ate executar um comando ou chegarmos ao limite
% de tentativas.
corrResults = zeros(3,3,2);
corrReference = zeros(3,3,2);
tryNo = 0;
while tryNo < 5
  % Nao e possivel fazer um 'for' aqui pq o tamanho das
  % referencias pode ser diferente.
  referenceImage  = imread( referenceImageFile );
  referenceImage1 = imread( referenceImageFile1 );
  referenceImage2 = imread( referenceImageFile2 );
  corrReference(:,:,1) = doCorrelation( referenceImage, ...
    referenceImage1 )
  corrReference(:,:,2) = doCorrelation( referenceImage, ...
    referenceImage2 )
  corrResults(:,:,1) = doCorrelation( currentRobotImage, ...
    referenceImage1 )
  corrResults(:,:,2) = doCorrelation( currentRobotImage, ...
    referenceImage2 )
  hasValidThreshold = evaluateThresholds( corrResults, ...
    THRESHOLD_CORRELATION );
  if all( hasValidThreshold )
          hasValidVectors = evaluateVectors( corrResults, ...
            THRESHOLD_VECTOR_DISTANCE );
          if all( hasValidVectors )
            disp( 'Gerando comando' );
            robotCmd = robotCommand( corrResults, corrReference, ...
              THRESHOLD_DELTA_IMG_CENTER, [ IMG_ROWS IMG_COLS ] );
            if robotCmd(1) == 0 % posicao
              if robotCmd(2) == -1 % recuo
                disp('Ops... recuo. Recuar e ir para a ...
                  proxima referencia');
                lastIndex = lastIndex + 1;
                break;
              else
                disp('Comando gerado... saindo do laco');
                break;
              end
            else
              disp('Comando gerado... saindo do laco');
              break;
            end
```

```
            else
              disp( 'A verificacao dos vetores de uma das referencias ...
                nao satisfez o criterio threshold' );
              lastIndex = lastIndex + 1;
            end
      else
        disp( 'Uma das referencias nao satisfez o criterio threshold' );
        lastIndex = lastIndex + 1;
      end
      referenceImageFile  = sprintf( '%s\\imgref%04d.jpg', imageBaseDir, ...
        lastIndex );
      referenceImageFile1 = sprintf( '%s\\imgref%04d_1.jpg', imageBaseDir, ...
        lastIndex );
      referenceImageFile2 = sprintf( '%s\\imgref%04d_2.jpg', imageBaseDir, ...
        lastIndex );
      hasAllReferences = ...
        ~exist( referenceImageFile)  & ...
        ~exist( referenceImageFile1) & ...
        ~exist( referenceImageFile2) ;
      if hasAllReferences
        disp( 'Uma das imagens de referencia esta faltando' );
        return;
      end
      tryNo = tryNo + 1;
      if tryNo == 5
        disp('Nao sei minha posicao atual');
      end
    end
  catch
          %
          % algum erro ocorreu...
          %
          [ msg, id ] = lasterr;
          formatedMsg = sprintf( '%s (%d)\n', msg, id );
          disp( formatedMsg );
  end
```

B.1.2 Script robotCommand

```
function robotCmd = robotCommand( corrValues, corrReference, threshold, ...
  imageDims )
%
% Baseado na posicao do pico de correlacao, gera um codigo indicando o
% comando a ser executado pelo robo.
%
% Parametros da Funcao
% --------------------
%
% corrValues:
%       Matriz 3 x 3 x 2 que contem em cada linha a (corr, r, c).
%
% corrReference:
%       Matriz 3 x 3 x 2 que contem em cada linha a (corr, r, c).
%
% threshold:
%       Define a area central da imagem (centro-threshold) ate (centro+threshold).
%       Medido em pixels.
%
% imageDims:
%       tamanho da imagem de referencia usada para calcular os valores contidos em
%       corrValues.
%
% Retorndo da Funcao
% ------------------
%
% Retorna um vetor de duas posicoes (tipo de comando,comando, quando girar)
%
% Codigo do Comando a ser Executado:
%  0 : Posicao
%  1 : Direcao
%
% Codigo dos Comandos de Direcao:
%
```

```
%  1 : virar a direita
% -1 : virar a esquerda
%
% Codigo dos Comandos de Posicao:
%  1 : avancar
% -1 : recuar
%
  try
    resPoints = zeros(2);
    refPoints = zeros(2);
    directionCmd = zeros(1,2);
    %forwardCmd = zeros(1,2);
    resThreshold = zeros(1,2);
    for i = 1:2
      resPoints(i,:) = doMeanPoint(corrValues(:,:,i));
      refPoints(i,:) = doMeanPoint(corrReference(:,:,i));
      resThreshold(i) = sqrt((resPoints(i,1)-refPoints(i,1))^2 + ...
        (resPoints(i,2)-refPoints(i,2))^2) <= (2 * threshold);
    end

    cmd = 0;
    cmdType = 0;
    angle = 0;

    if all(resThreshold)
      robotCmd = [ cmdType cmd angle ];
      return
    end

    for i = 1:2
      % verifica as colunas...
      if resPoints(i,2) < refPoints(i,2)
              directionCmd(i) = -1;
      else
              directionCmd(i) = 1;
      end
    end

    if directionCmd(1) == directionCmd(2)
      cmd = directionCmd(1);
      cmdType = 1;
      angle = 5;
    elseif (directionCmd(1) > 0 & directionCmd(2) < 0) | ...
           (directionCmd(2) > 0 & directionCmd(1) < 0)
      posCmd = 1;
      angle = 0;
    else
      posCmd = -1;
      angle = 0;
    end
    % cmdType == 0 => posicao (avanca, recua)
    % cmdType == 1 => direcao (direita, esquerda)
    robotCmd = [ cmdType cmd 5 ];
  catch
    %
    % algum erro ocorreu...
    %
    [ msg, id ] = lasterr;
    formatedMsg = sprintf( '%s (%d)\n', msg, id );
    disp( formatedMsg );
  end
```

B.2 Scripts Auxiliares

B.2.1 Script doCorrelation

```
function corrResults = doCorrelation( robotImage, referenceImage )
%
% Calcula os valores de correlacao das componentes RGB entre as imagens
% robotImage e referenceImage. Tambem determina a posicao de cada pico
% de correlacao.
```

```
%
% Parametros da Funcao
% --------------------
%
% robotImage:
%       Imagem que o robo capturou.
%
% referenceImage:
%       Imagem que pertence ao caminho que o robo deve seguir.
%
% Retorno da Funcao
% -----------------
%
% corrResults:
%       Matriz 3 x 3 que contem em cada linha a (corr, r, c).
%
  try
    %
    % Definimos tudo como zero...
    %
    corrResults = zeros(3,3);
    %imageDims = size( referenceImage );
    imageDims = size( robotImage );
    imageDims = [ imageDims(1) imageDims(2) ];
    %
    % Para cada um dos canais (R,G,B)...
    %
    for i = 1 : 3
      % Procura a referencia dentro da imagem que o robo esta
      % "olhando".
      corrMatrix = normxcorr2( referenceImage(:,:,i), robotImage(:,:,i) );
      [ corrResults(i,1), idx ] = max( corrMatrix(:) );
      [ corrResults(i,2), corrResults(i,3) ] = ...
        toMaxCoord( corrMatrix, imageDims );
    end
  catch
    %
    % algum erro ocorreu...
    %
    [ msg, id ] = lasterr;
    formatedMsg = sprintf( '%s (%d)\n', msg, id );
    disp( formatedMsg );
  end
```

B.2.2 Script toMaxCoord

```
function [ row, col ] = toMaxCoord( corrMatrix, imageSize )
%
% Converte as coordenadas da matriz de correlacao para coordenadas da
% imagem
%
% Parametros da Funcao
% --------------------
%
% corrMatrix:
%       Matriz de correcao entre duas imagens.
%
% imageSize:
%       Tamanho da imagem [linhas colunas] da imagem de referencia
%   usada para gerar corrMatrix.
%       pixels.
%
% Retorno da Funcao
% -----------------
%
% Retorna a linha e coluna [ r c ] onde ocorre o pico de correlacao.
%
  [ maxValue, idx ] = max( corrMatrix(:) );
  [ r, c ] = ind2sub( size(corrMatrix), idx );
  corrMatrixDims = size(corrMatrix);
  row = round( (r * imageSize(1)) / corrMatrixDims(1) );
  col = round( (c * imageSize(2)) / corrMatrixDims(2) );
```

B.2.3 Script evaluateVectors

```
function validVectors = evaluateVectors( corrValues, threshold )
%
% Verifica se os picos de correlacao de cada componente estao
% proximos.
%
% Parametros da Funcao
% --------------------
%
% corrValues:
%       Matriz 3 x 3 x 2 que contem em cada linha a (corr, r, c).
%
% threshold:
%       Raio de influencia de um pico sobre o outro. Medido em
%       pixels.
%
% Retorno da Funcao
% -----------------
%
% Retorna um valor booleano indicando se os vetores estao proximos
% uns dos outros.
%
%
  try
    %
    % Verifica se todos os picos de correlacao estao proximos uns dos
    % outros.
    %
    d1 = zeros(1,2);
    d2 = zeros(1,2);
    d3 = zeros(1,2);
    validVectors = zeros(1,2);
    for i = 1 : 2
      d1(i) = sqrt( ( corrValues(1,2,i)-corrValues(2,2,i) ) ^ 2 + ...
        ( corrValues(1,3,i)-corrValues(2,3,i) ) ^ 2 ) <= (threshold * 2);
      d2(i) = sqrt( ( corrValues(1,2,i)-corrValues(3,2,i) ) ^ 2 + ...
        ( corrValues(1,3,i)-corrValues(3,3,i) ) ^ 2 ) <= (threshold * 2);
      d3(i) = sqrt( ( corrValues(2,2,i)-corrValues(3,2,i) ) ^ 2 + ...
        ( corrValues(2,3,i)-corrValues(3,3,i) ) ^ 2 ) <= (threshold * 2);
      validVectors(i) = d1(i) & d2(i) & d3(i);
    end
  catch
    %
    % algum erro ocorreu...
    %
    [ msg, id ] = lasterr;
    formatedMsg = sprintf( '%s (%d)\n', msg, id );
    disp( formatedMsg );
  end
```

B.2.4 Script evaluateThresholds

```
function validThresholds = evaluateThresholds( corrValues, threshold )
%
% Verifica se todos valores de correlacao estao iguais ou maiores que um
% determinado threshold.
%
% Parametros da Funcao
% --------------------
%
% corrValues:
%       Matriz que contem os valores de correlacao para cada
%       compomponente RGB.
%
% threshold:
%       Valor minimo exigido de correlacao.
%
% Retorno da Funcao
% -----------------
%
% Retorna um valor booleano indicando se os valores de correlacao
% estao acima de threshold.
%
```

```
try
        %
        % Faz a verificacao
        %
        validThresholds1 = ...
                (corrValues(1,1,1) >= threshold) & ...
                (corrValues(2,1,1) >= threshold) & ...
                (corrValues(3,1,1) >= threshold);
        validThresholds2 = ...
                (corrValues(1,1,2) >= threshold) & ...
                (corrValues(2,1,2) >= threshold) & ...
                (corrValues(3,1,2) >= threshold);
        validThresholds = [ validThresholds1 validThresholds2 ];
catch
        %
        % algum erro ocorreu...
        %
        [ msg, id ] = lasterr;
        formatedMsg = sprintf( '%s (%d)\n', msg, id );
        disp( formatedMsg );
end
```

REFERÊNCIAS

Canon (2003). Acessado em `http://www.powershot.com`.

Couleur.org (2004). Acessado em `http://www.couleur.org`.

Dudek, G. and Jenkin, M. (2000). *Computational Principles of Mobile Robotics*. Cambridge University Press.

Gonzalez, R. C. and Woods, R. E. (2002). *Digital Image Processing* 2^{nd} *Edition*. Prentice Hall.

Gross, H.-M., Koenig, A., Schroeter, C., and Boehme, H.-J. (2003). Omnivision-based probabilistic self-localization for a mobile shopping assistant continued. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, Nevada – USA.

Hanselman, D. and Littlefield, B. (2003). *Matlab 6 Curso Completo*. Prentice Hall.

Heinen, F. J. (2000). *Robótica Autônoma: A Integração entre Planificação e Comportamento Reativo*. Editora Unisinos.

Heinen, F. J. (2002). *Sistema de Controle Híbrido para Robôs Móveis Autônomos*. Unisinos / PIPCA – Mestrado em Computação Aplicada, São Leopoldo – RS, Brasil. Dissertação de Mestrado.

Hu, Z. and Uchimura, K. (2002). Dynamical road modeling and matching for direct visual navigation. In *Proceedings of the* 6^{th} *IEEE Workshop on Applications of Computer Vision (WACV'02)*.

Jain, R. (1995). *Machine Vision*. McGraw-Hill.

Jones, S. D., Andersen, C. S., and Crowley, J. L. (1997). Appearance based processes for visual navigation. In *Proceedings of the* 5^{th} *International Symposium on Intelligent Robotic Systems, SIRS'97*, pages 227–236, Stockholm, Sweden.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. volume 60, pages 91–110.

Martin, J. and Crowley, J. L. (1995). Experimental comparison of correlation techniques. In *Proceedings of the International Conference on Intelligent Autonomous Systems*, Karlsruhe, Germany.

Mathworks (2002). Acessado em `http://www.mathworks.com`.

Matsumoto, Y., Ikeda, K., Inaba, M., and Inoue, H. (1999). Visual navigation using omnidirectional view sequence. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Matsumoto, Y., Inaba, M., and Inoue, H. (1996). Visual navigation using view-sequenced route representation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 83–88, Minneapolis, Minnesota.

Moschetta, E., Osório, F. S., and Cavalheiro, G. G. H. (2001). Reconhecedor de imagens concorrente. *Scientia UNISINOS*, 12(1):11–25.

Moschetta, E., Osório, F. S., and Cavalheiro, G. G. H. (2002). Reconhecimento de imagens em aplicações críticas. In *WSCAD – III Workshop em sistemas computacionais de alto desempenho*, Vitória – ES.

Murray, J. D. (1996). *Encyclopedia of Graphics File Formats* 2^{nd} *Edition*. OReilly & Associates.

Righes, E. M. and Osório, F. S. (2004). Correlação de imagens coloridas visando auxiliar na navegação e controle de robôs autônomos. In *I EnRI – I Encontro Nacional de Robótica Inteligente*, Salvador – BA. Submetido ao I EnRI.

Sinclair, D. (1997). Image parsing for image retrieval from large image data bases: from coloured image to coloured regions. Technical report, ORL – Olivetti Research Labs Ltd UK.

Stroustrup, B. (1997). *The C++ Programming Language* 3^{rd} *Edition*. Addison-Wesley.