Universidade Federal de Santa Catarina
Centro de Blumenau
Departamento de Engenharia de
Controle e Automação e Computação

Rômulo Pacher Júnior

Localização de Robôs por Reconhecimento Ótico de Caracteres de Placas

Blumenau
2018

Rômulo Pacher Júnior

Localização de Robôs por Reconhecimento Ótico de Caracteres de Placas

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Santa Catarina como parte dos requisitos necessários para a obtenção do Título de Engenheiro de Controle e Automação.
Orientador: Prof. Dr. Marcelo Roberto Petry

Universidade Federal de Santa Catarina
Centro de Blumenau
Departamento de Engenharia de
Controle e Automação e Computação

Blumenau
2018

Rômulo Pacher Júnior

Localização de Robôs por Reconhecimento Ótico de Caracteres de Placas

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Santa Catarina como requisito parcial para a obtenção do título de Engenheiro de Controle e Automação.

Comissão Examinadora

Prof. Dr. Marcelo Roberto Petry
Universidade Federal de Santa Catarina
Orientador

Prof. Dr. Marcos Vinicius Matsuo
Universidade Federal de Santa Catarina

Prof. Dr. Leonardo Mejia Rincon
Universidade Federal de Santa Catarina

Blumenau, 31 de janeiro de 2019

Dedico este trabalho a todos aqueles que, de alguma forma, auxiliaram para a concretização desta etapa em minha jornada.

Agradecimentos

Agradeço aos meus pais pela compreensão e apoio que me ofereceram durante a realização deste trabalho. Agradeço também ao meu orientador que me guiou e motivou até a conclusão deste trabalho.

"Nobody exists on purpose, nobody belongs anywhere, everybody's gonna die. Come watch TV."

(Morty Smith)

Resumo

Reconhecimento ótico de caracteres é o processo pelo qual o conteúdo textual presente em uma imagem é convertido em strings. Localização é o problema de descobrir onde se está em um determinado ambiente. Neste trabalho abordamos a aplicação de reconhecimento ótico de caracteres (OCR) para localização de robôs. Nós propomos, desenvolvemos e testamos um sistema de localização baseado em visão capaz de detectar placas de identificação de salas presentes no ambiente, reconhecer seu conteúdo textual e aplicá-lo para determinar sua posição referente a um mapa topológico do ambiente. Desenvolvemos um método para detecção das placas baseado em segmentação de imagem por cor e detecção de cantos pela análise de seu contorno. O reconhecimento de caracteres é realizado com a aplicação de um motor de OCR de código aberto. A localização é realizada pela comparação das leituras das placas com as informações textuais registradas no mapa topológico do ambiente. O algoritmo desenvolvido foi testado em um dataset de imagens adquiridas em um corredor. Os resultados experimentais mostram que o sistema proposto é capaz de determinar sua localização com sucesso em 54,54% dos casos testados. Com este trabalho verificamos a possibilidade da aplicação de reconhecimento ótico de caracteres para localização topológica de robôs.

Palavras-Chave: 1. Localização de robôs. 2. Visão computacional. 3. Reconhecimento ótico de caracteres. 4. Robótica. 5. Localização topológica.

Abstract

Optical character recognition is the process by which the textual content of an image is converted into strings. Localization is the problem of figuring out where one is in a given environment. In this work we approach the application of optical character recognition (OCR) in robot location. We propose, develop and test a vision based localization system that is capable of detecting room identification signs present in the environment, recognise their textual contents and apply them to determine its position referent to a topological map of the environment. We developed a method to detect the signs based on image segmentation by color and corners detection by the analysis of its contour. The recognition of characters is performed with the application of an open-source OCR engine. Localization is realized through the comparison of signs' readings with the textual information embedded in the topological representation of the environment. The developed algorithm was tested in a dataset of images acquired in a corridor. The experimental results show that the developed system is capable of successfully determining its location in 54,54% of tested cases. With this work we verify the possibility of employing character recognition for robot topological localization.

Keywords: 1. Robot localization. 2. Computer vision. 3. Optical character recognition. 4. Robotics. 5. Topological localization.

Lista de figuras

Lista de tabelas

Lista de Siglas e Abreviaturas

AGV	*Autonomous Ground Vehicles*
API	*Application Programming Interface*
AUV	*Autonomous Underwater Vehicles*
GPS	*Global Positioning System*
ICDAR'13	*International Conference on Document Analysis and Recognition*
INS	*Inertial Navigation System*
LTSM	*Long Short Term Memory*
MSER	*Maximally Stable Extremal Region*
OCR	*Optical Character Revognition*
ROI	*Region Of Interest*
SLAM	*Simultaneous Location And Mapping*
UAV	*Unmanned Aerial Vehicle*
UFSC	*Universidade Federal de Santa Catarina*

Sumário

1 Introdução

A capacidade de navegação é uma característica essencial para robôs autônomos que se deslocam livremente dentro de um ambiente de trabalho, lhes permitindo alcançar objetivos e evitar obstáculos. Esta característica é atribuída ao sistema de navegação do robô, cuja função é desempenhar os diversos processos envolvidos na navegação. Esta função pode ser realizada por um sistema único ou pela integração de subsistemas responsáveis respectivamente pela localização, detecção de obstáculos, planejamento de trajetória e demais processos.

O conceito de navegação é muito amplo e podem-se encontrar diversas definições para este conceito na literatura que se diferenciam quanto a sua abrangência. Siegwart *et al.* [2] definem o processo de navegação como sendo composto por quatro subprocessos: i) percepção, em que o robô deve interpretar as leituras de seus sensores e extrair informações importantes; ii) localização, processos pelo qual o robô deve determinar sua posição no ambiente; iii) cognição, em que o robô deve decidir como agir para atingir seus objetivos; e iv) controle de movimento, processo no qual o robô realiza o controle de seus motores para seguir uma trajetória determinada. Outras definições incluem processo como a construção, atualização e interpretação de um modelo do ambiente ou realizam a subdivisão dos processos de forma diferente. De qualquer forma, todas estas tarefas são interligadas e cada uma afeta de forma direta ou indireta o desempenho das demais.

Devido a ampla gama de sensores disponíveis e a grande importância deste sistema, inúmeras pesquisas foram e vêm sendo desenvolvidas com a navegação de robôs como seu foco. Dos quatro subprocessos usados por Siegwart *et al.* [2] para definir a navegação de robôs, o processo de localização é um dos que apresenta os problemas mais desafiadores e mais recebe atenção em pesquisas. Os resultados de suas contribuições são um variado espectro de métodos de navegação e localização, tanto para aplicações *indoor* quanto *outdoor*, e seu constante aprimoramento, que provocam a expansão das possíveis aplicações de robôs autônomos. Soluções baseadas em visão computacional, em particular, contribuíram para esta expansão. Sua versatilidade possibilita aplicações em veículos aéreos não tripulados (UAV) e em veículos submarinos autônomos (AUV), além das aplicações clássicas em veículos terrestres autônomos (AGV) [5].

Os sistemas de navegação baseados em visão podem ser classificados em relação a algumas características como o uso (ou não) de mapas para representar o ambiente, os tipos de mapas que utilizam, e se são capazes de construir os mapas por conta própria [6]. Existem dois tipos básicos de mapas que são usados por sistemas de navegação, mapas métricos e mapas topológicos. Sistemas que não fazem o uso de mapas percebem seu ambiente conforme se deslocam por ele, sem construir representações do mesmo. Se

orientam por meio de pontos de referência visuais obtidos por segmentação de imagens, *optical flow*, ou rastreando *features*. A maioria destes sistemas apresenta comportamento reativo durante sua navegação. Sistemas que utilizam mapas métricos usam representações completas de seu ambiente baseadas em informações quantitativas e sistemas de coordenadas. Os sistemas com capacidade de mapeamento determinam estas informações durante uma fase de exploração do ambiente, antes do início de suas tarefas de navegação. Os sistemas sem essa capacidade precisam se basear em um mapa previamente fornecido. O desempenho destes sistema os torna mais apropriados para aplicações em ambientes de menor escala e quando é necessária uma maior precisão no posicionamento do robô. Sistemas de navegação que usam mapas topológicos empregam representações qualitativas do ambiente na forma de grafos. Os sistemas que realizam mapeamento adicionam nós ao grafo e definem as ligações entre os nós e seu significado. Os sistemas sem essa capacidade usam uma rede de nós que lhes é fornecida antes do início da navegação. Estes sistemas apresentam melhor escalabilidade e robustez à erro numérico acumulado quando comparados aos sistemas métricos. Há ainda sistemas que empregam representações híbridas, buscando combinar as vantagens e amenizar as desvantagens de cada representação. Em alguns casos os sistemas são capazes de se localizar e mapear o ambiente simultaneamente, independente do tipo de representação usada. Estes sistemas recebem a classificação de *Simultaneous Location And Mapping* (SLAM).

No caso de sistemas de navegação baseados em visão computacional, os métodos de localização aplicados variam dependendo da classificação do sistema. Isto está relacionado à necessidade dos métodos de localização e a representação do ambiente usada pelo sistema serem compatíveis. Apesar disto implicar na restrição de opções viáveis, a ampla gama de métodos de localização que existem permitem certo grau de liberdade para escolha da alternativa mais adequado para a aplicação. São alguns exemplos a Odometria Visual, que busca estimar a localização do robô analisando o efeito do movimento nas imagens obtidas pela câmera acoplada ao robô. Técnicas de SLAM, que além de estimar a posição do robô a partir do movimento, usam o conceito de *Structure from Motion* para desenvolver uma representação do ambiente e melhorar os resultados da estimação da posição. Métodos baseados em *features* locais que extraem informações características de elementos do ambiente presentes na imagem e os associam com uma respectiva localização. E sistemas que utilizam marcadores fiduciais[1] como pontos de referência, seja para caracterizar uma região do ambiente ou para estimar a posição do robô em relação ao marcador.

Entre as diversas pesquisas realizadas, pode-se observar que o reconhecimento ótico de caracteres não teve sua aplicação em localização de robôs tão amplamente explorada. Especialmente quando comparado com os métodos baseados em Odometria Visual, *Bag of Words*, *features* locais, descritores globais ou marcadores fiduciais. Isso está relacionado

[1] Padrões bidimensionais artificiais posicionados no ambiente para extração de medidas usando câmeras.

com a dificuldade de se realizar o reconhecimento ótico de caracteres.

Embora já se disponha de soluções capazes de reconhecer corretamente mais de 99% dos caracteres em documentos impressos, o reconhecimento de texto em cenas naturais ainda é considerado um problema aberto para pesquisas. Há uma grande distância a ser percorrida para se atingir a performance requerida, mesmo com os promissores resultados apresentados por trabalhos recentes. Mesmo assim, já é possível se aproveitar destes resultados para explorar sua aplicação na localização de robôs.

Neste projeto apresentamos um estudo da viabilidade e desenvolvimento de um sistema de localização de robôs móveis autônomos baseado em marcadores textuais (placas de identificação de salas) destinados ao uso humano. Este sistema será destinado a aplicações em ambientes estruturados onde não há a ocorrência de marcadores iguais. O sistema integra um algoritmo de detecção de marcadores que desenvolvemos, um motor de reconhecimento ótico de caracteres (OCR) de código aberto para o reconhecimento dos marcadores e um algoritmo para a localização referente à um mapa topológico do ambiente.

Não faz parte do escopo deste trabalho o tratamento de oclusão parcial e condições de iluminação desfavoráveis durante a detecção de marcadores. O teste do sistema aplicado em um robô também não faz parte do escopo. Consequentemente, não se pretende detalhar a natureza das interligações entre nós do mapa topológico além de simples relações de adjacência.

1.1 Identificação dos Objetivos

1.1.1 Objetivo Geral

O objetivo geral deste trabalho foi desenvolver um sistema capaz de determinar sua localização referente à um mapa topológico do seu ambiente de trabalho. Este sistema utiliza placas de identificação de salas como pontos de referência. Para isso ele às identifica por meio do reconhecimento do seu conteúdo textual.

1.1.2 Objetivos Específicos

- Desenvolver um método para detectar e isolar as placas textuais nas imagens obtidas por uma câmera;
- Realizar o reconhecimento do conteúdo das placas usando uma interface com um motor de OCR;
- Elaborar uma estratégia para o descarte de falsos positivos;
- Formular um mapa topológico básico do ambiente;

- Realizar a localização dentro do mapa topológico baseando-se nos marcadores visíveis;

1.2 Motivação

A principal motivação para o uso de reconhecimento de caracteres na localização de robôs autônomos encontra-se em aplicações de navegação semântica. Nestas aplicações o robô recebe comandos textuais ou de voz para definir sua tarefa. Aplicando reconhecimento de caracteres para realizar sua localização, o robô pode utilizar a mesma sintaxe para localização e compreensão de comandos. A motivação por trás da localização topológica vem da sua aplicação em navegação de alto nível. Por fim, a motivação para o uso de marcadores intrínsecos ao ambiente vem de situações em que a instalação de marcadores artificiais seja indesejada ou impossível.

1.3 Estrutura do Documento

Este documento encontra-se estruturado da seguinte maneira. No Capítulo 2 é realizada uma revisão da literatura relacionada aos temas e subtemas abordados por este trabalho como, por exemplo, o problema da localização, métodos de localização baseados em visão, tipos de mapas empregados por sistemas de localização, reconhecimento ótico de texto e trabalhos relacionados. No Capítulo 3 o sistema de localização proposto é elaborado detalhadamente. Em seguida, no Capítulo 4 são descritos os testes realizados com o sistema e apresentados os resultados obtidos. Por fim no Capítulo 5 concluímos este trabalho com uma discussão sobre os resultados apresentados e possibilidades para trabalhos futuros.

2 Revisão bibliográfica

O sistema proposto neste trabalho envolve diversos tópicos que já foram abordados por trabalhos anteriores. Neste capítulo faremos uma relação destes diferentes assuntos, apresentando suas definições e pesquisas relacionadas com seu desenvolvimento ou aplicação. Começaremos com uma breve definição do problema da localização, seguindo adiante com uma discussão geral de métodos de localização baseados em visão. Abordaremos então marcadores fiduciais, em particular devido às suas semelhanças com o método proposto. Em seguida discutiremos sobre diferentes tipos de representações do ambiente empregadas em localização e navegação. Na sequência será discutido sobre reconhecimento de texto e concluiremos o capítulo revisando alguns trabalhos semelhantes a este.

2.1 O Problema da Localização

Conforme Werner *et al.* [7], a navegação é um tópico de pesquisa interdisciplinar pois os problemas da navegação em sua forma geral são aplicáveis tanto para organismos vivos quanto agentes artificiais. O mesmo pode ser dito da localização por ser uma parte fundamental deste processo. Navegação, conforme Leonard [8], pode ser resumida a responder 3 perguntas: "Onde estou?", "Onde vou?" e "Como devo chegar lá?". A primeira destas perguntas é nosso foco, pois representa o processo de localização. Ela diz respeito ao problema de descobrir onde se está em um determinado ambiente, baseando-se no que pode ser percebido e no que já se conhece.

A percepção do ambiente define uma ação de aquisição de informação. No caso de seres vivos, isto é realizado por meio de seus sentidos e respectivos órgãos sensoriais. Já em agentes artificiais, esta ação é desempenhada pelos sensores com os quais foram equipados. Estes sensores são provenientes de uma ampla gama de modelos com funções distintas. De sensores ultra-sônicos e lasers que medem proximidade à câmeras que geram imagens, cada um extraindo um tipo diferente de informação do ambiente.

O conhecimento previamente adquirido sobre o ambiente pode ser interpretado como uma representação ou um mapa do mesmo. Este mapa estabelece relações espaciais quantitativas e qualitativas entre elementos e regiões do ambiente. Quando as informações recentemente percebidas são armazenadas e integradas às informações previamente conhecidas, o agente pode atualizar ou expandir a representação do ambiente.

2.2 Métodos de Localização Baseados em Visão

Nas últimas décadas podê-se observar um significante aumento no número de soluções de localização, mapeamento e navegação baseadas em visão. Sua premissa básica é o uso de conceitos de visão computacional ao processar imagens capturadas por uma ou mais câmeras para obter informações úteis na solução do problema da localização. O baixo custo das câmeras, quando comparadas com outros sensores, e a riqueza de informações que podem ser extraídas das imagens foram alguns dos fatores que motivaram os diversos trabalhos realizados na área.

As principais contribuições e progresso desenvolvidos até 1998 foram amplamente estudadas e documentadas por DeSouza e Kak [6] e posteriormente, até 2008, por Bonin-Font *et al.* [5]. Ambos autores classificam os sistemas desenvolvidos até o final dos anos 90 como localização em ambientes internos (*indoor*) ou ambientes externos (*outdoor*). Os sistemas de localização *indoor* foram subdivididos conforme o uso de mapas em: sistemas que não empregavam representações do ambiente [9][10]; sistemas que utilizam representações prontas do ambiente [11][12][13]; e sistema capazes de construir a própria representação do ambiente [14]. Os sistemas de localização *outdoor* foram subdivididos conforme seu ambiente de operação em sistemas destinados a ambientes estruturados [15] e sistemas destinados a ambientes não estruturados [16].

Para a classificação das contribuições de publicações entre o final dos anos 90 e 2008, Bonin-Font *et al.* [5] adotaram o paradigma de classificação baseado no tipo de representação do ambiente empregada pelo sistema de localização. Assim, os sistemas foram classificados em sistemas que usam mapas métricos [17][18], sistemas que usam mapas topológicos [19][20] e sistemas que não empregavam mapas [21][22]. As classes de sistemas que usam mapas foram subdivididas entre sistemas que usam mapas prontos e sistemas capazes de mapeamento. Na seção 2.3 falaremos mais detalhadamente sobre mapas. A mudança no método de classificação deve-se a maior maturidade das técnicas deste período, que implicou em sistemas que poderiam ser adaptados para operar tanto em ambientes *indoor* quanto *outdoor*.

A classificação em sistemas de localização topológica ou métrica continua sendo usada. Um exemplo disso é o trabalho de Garcia-Fidalgo e Ortiz [23]. Neste trabalho os autores realizaram um estudo dos métodos recentes de localização e mapeamento topológico baseados em visão. Eles subdividem estas técnicas de acordo com a forma de representação da imagem adotada, seja ela feita por descritores globais [24][25][26][27], *features* locais [28][29][30], esquema *Bag-Of-Words* [31][32][33], ou suas combinações [34][35]. Enquanto isso, Ben-Afia *et al.* [36] apresentaram um sistema de classificação que busca abranger tanto técnicas de localização que usam exclusivamente visão, quanto técnicas que combinam visão com outras informações por meio de fusão de sensores.

O tipo de câmera utilizada pelas diferentes estratégias de localização propostas na lite-

ratura também representam um diferencial entre elas. As configurações mais comumente encontradas são câmeras monoculares [34][33] e binoculares (estéreo) [37], embora câmeras omnidirecionais [35][31] também sejam bastante usadas. Existem ainda estratégias que utilizam estruturas trinoculares [38], porém estas são menos frequentes. Estas configurações distintas apresentam características particulares. Câmeras estereoscópicas, por exemplo, possibilitam a recuperação de informação de distância à curto, médio ou longo alcance, dependendo da distânica entre os dois sensores de imagem (*baseline*), quando a calibração da câmera é conhecida. Câmeras omnidirecionais são compostas por uma câmera normal direcionada para uma superfície reflexiva cônica ou esférica. Elas oferecem uma vista de 360º em torno do robô, assim a permanência de *features* no campo de visão do robô é prolongada, o que facilita seu rastreamento.

Outro aspecto que diferencia os métodos de localização são o tipo de agente móvel no qual são aplicados. Originalmente visão era usada exclusivamente por técnicas destinadas a aplicação em AGVs. Suposições podem ser feitas sobre o movimento deste tipo de veículo, reduzindo o número de graus de liberdade de seu movimento, o que facilita o processo de localização. Porém, com o passar do tempo e desenvolvimento de novos trabalhos, a versatilidade do uso de visão para localização foi explorada. Isto, em conjunto com o aumento da capacidade de carga e poder computacional de outros tipos de veículo, contribuiu para o surgimento de aplicações capazes de lidar com os 6 graus de liberdade em UAVs [39][40] e AUVs [41][42].

O espectro de soluções de localização baseadas em visão disponível atualmente é bastante amplo. Sua abrangência inclui diferentes representações do ambiente, configurações de sensores distintas e aplicação em vários modelos de agentes móveis. Porém entre todas estas estratégias, as que usam marcadores fiduciais são particularmente de interesse deste trabalho pela semelhanças que compartilha com o método proposto.

2.2.1 Marcadores Artificiais

Embora marcadores fiduciais sejam semelhantes a outros sistemas como QR code ou Maxixode, suas aplicações possuem objetivos completamente diferentes. QR code (Figura 1) e Maxicode (Figura 2) têm o objetivo de transmitir uma grande quantidade de informações e são projetados para aplicações em condições favoráveis para sua detecção e reconhecimento, como imagens ocupadas quase exclusivamente pelos códigos sem apresentar distorção perspectiva ou iluminação irregular.

Marcadores fiduciais são elementos artificiais posicionados em uma cena para servirem como identificadores, pontos de referência ou medidas, cuja detecção é realizada por algoritmos que são normalmente desenvolvidos juntamente com os marcadores. Esses elementos apresentam modelos visuais conhecidos, os quais são projetados para facilitar seu reconhecimento em meio a cena e sua distinção uns dos outros. Isto significa que

Figura 1 – Exemplo de QR code.

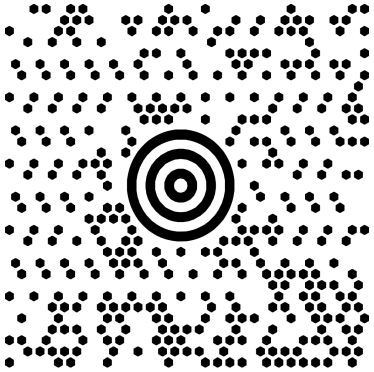


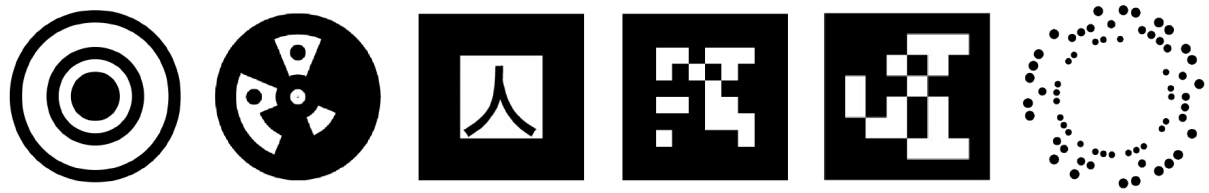
Figura 2 – Exemplo de Maxicode.

seus projetos precisam levar em consideração condições adversas como baixa resolução da imagem, iluminação desigual, oclusão, posição e rotação arbitrária do marcador. Tais modelos costumam tirar proveito de aspectos como contraste e características geométricas para facilitar sua identificação. Círculos, por exemplo, são um formato comumente usados pois seu centro apresenta características como coincidência com seu centróide, invariância quanto ao ângulo e direção de visualização, possibilidade de correção exata de *offsets* causados por distorção perspectiva e sua determinação pode apresentar precisão *sub-pixel* [43]. Quadrados também são amplamente usados pois retas não são distorcidas por transformações perspectivas. Contanto que a distorção causada pela lente da câmera seja pequena ou seja corrigida, os marcadores podem ser detectados buscando-se por 4 retas que se interceptam [43]. Além disso, quadrados fornecem informação necessária para se calcular a postura relativa entre câmera e o marcador. As informações contidas nos marcadores pertinentes para fins de identificação podem ser apresentadas de diversas formas. De códigos binários a padrões, alguns modelos apresentam recursos para detecção e correção de erro, enquanto outros são particularmente robustos contra oclusão.

Na Figura 3 podemos observar alguns exemplos de marcadores fiduciais. Da esquerda para a direita, começamos em (a) com os círculos concêntricos contrastantes (CCC) [44], um dos primeiros modelos de marcadores fiduciais desenvolvidos. Este modelo herda as características geométricas positivas dos círculos, e determina se um elemento é um marcador checando se os centróides dos anéis encontrados na imagem coincidem. Para distinção do marcador, são usados círculos de diferentes tamanhos, cores [45] ou retrorrefletividade.

Na Figura 3(b) temos os marcadores Intersense, apresentados em [43] como parte do desenvolvimento de um sistema de *self-tracking*. Estes marcadores foram projetados tomando os círculos concêntricos como base e buscando uma forma de aumentar a quantidade de códigos de identificação distintos. Sua estrutura é composta por dois anéis pretos e um círculo branco concêntricos. O espaço entre os anéis pretos é dividido em 8 seções, das quais 3 são usadas para determinação da orientação do marcador. As demais 5, que são compostas por 3 campos cada uma, contém um código binário de identificação.

Mais à direita, na Figura 3(c), é apresentado o exemplo de um marcador ARToolkit



(a) Concentric Circles (b) Intersense (c) ARToolkit (d) ARTag (e) AprilTag (f) RUNE-129

Figura 3 – Exemplos de marcadores fiduciais. Imagem adaptada de [1]

[46, 47]. Seu processo de detecção é baseado em uma simples binarização da imagem usando um limiar estipulado pelo usuário. A partir da imagem binarizada, tira-se proveito das características geométricas do quadrado para se detectar marcadores em meio a cena. Dentro do quadrado é posicionado um padrão, o qual é usado para verificar a detecção e identificar os marcadores. Embora isso teoricamente permita um grande número de marcadores distintos, o uso de padrões implica em uma série de problemas. Alguns deles são a reduzida robustez à iluminação irregular, interferência da orientação do marcador na detecção e redução da singularidade de cada marcador conforme o crescimento da biblioteca de padrões, ambos problemas que aumentam o tempo de processamento. Estes problemas implicam na possibilidade do sistema de detecção confundir marcadores diferentes, especialmente em imagens de baixa resolução.

Isso motivou a criação dos marcadores ARTag [48], um exemplo do qual pode-se observar logo à direita na Figura 3(d). Este modelo de marcador substitui o padrão por uma matriz binária. Esta matriz é composta por 32 bits (6x6), dos quais 10 carregam o código de identificação. Os demais 26 são redundâncias usadas por técnicas de decodificação digital robustas para correção de erros. Além disso, o seu processo de detecção é baseado no gradiente da imagem. Estas alterações resultaram em marcadores com melhor desempenho, tanto na taxa de identificação dos marcadores quanto na performance do algoritmo de detecção. Estes marcadores também apresentam robustez à oclusão parcial dos marcadores e à condições de iluminação problemáticas para marcadores ARToolkit.

Seguindo adiante, em (e) temos as AprilTags, uma das opções mais recentes de marcadores fiduciais. Originalmente apresentadas por Olson [49], as AprilTags apresentam semelhanças com as ARTags, tanto em sua aparência, por empregarem uma matriz binária, quanto em sua detecção, que também é baseada na análise do gradiente da imagem. Porém uma diferença importante entre os dois marcadores é a geração da matriz binária. AprilTags são baseadas em um *lexicode* modificado [50]. O algoritmo de geração deste *lexicode* gera *codewords* que respeitam uma distância mínima de Hemming mesmo quando rotacionadas e que apresentem uma certa complexidade geométrica. Esta complexidade geométrica é definida pelo número mínimo de quadriláteros necessários para se desenhar a matriz binária. Com isso, embora o número de códigos distintos gerados para um tamanho específico de marcador seja reduzido, pode-se gerar códigos para ta-

manhos arbitrários de marcadores (3x3, 4x4, 5x5, 6x6). Estas diferenças proporcionaram um maior número de códigos distintos e um desempenho melhor que ARTags e ARToolkits. Posteriormente Olson e Wang [51] apresentaram um novo sistema de detecção que troca a capacidade de detecção de marcadores sob oclusão parcial por maior velocidade de detecção e sensitividade sem prejudicar a precisão das detecções. Este novo método de detecção baseia-se na análise de uma imagem binária obtida com a aplicação de um limiar adaptativo (*adaptative threshold*) à imagem de entrada.

Por fim, na Figura 3(f) pode-se observar uma RUNE-Tag, outro exemplo de um modelo recente. Bergamasco et. al. [1] apresentam estes marcadores com a proposta de oferecer elevada robustez à oclusão. A estrutura do marcador é composta por um disco que é dividido em anéis concêntricos denominados níveis. O disco dividido em anéis é então subdivididos em setores angulares uniformes. Os pares integrados por um nível e um setor formam campos que podem ser ocupados por pontos. Estes pontos são *features* circulares de tamanho proporcional ao raio do nível em que se encontram e a forma como são distribuídos pelos campos permite a inserção de informação no marcador. Seu processo de detecção inicialmente encontra elipses presentes na imagem e então faz uma análise das estruturas cônicas associadas a estas elipses para determinar quais delas formam um marcador. A identificação dos marcadores é então realizada através dos códigos cíclicos que definem a distribuição dos pontos em cada nível do marcador.

Originalmente desenvolvidos para aplicações de realidade aumentada, como inserção de elementos virtualmente gerados em imagens, marcadores fiduciais tiveram seu uso progressivamente difundido para diversas outras áreas que utilizam visão. Isto ocorreu por serem a melhor opção em questão de precisão e repetibilidade para realização de medições baseadas em imagem [1]. Comunicação, *tracking*, *structure from motion*, detecção de objetos, *ground truthing* e definição de postura de câmeras são apenas algumas das várias aplicações de fiduciais.

Em localização de robôs, marcadores fiduciais são utilizados principalmente para estimação da posição relativa do robô referente ao marcador. Esta informação apresenta caráter métrico e, quando a localização global do marcador é conhecida, permite estimar a localização global do robô. A velocidade, a precisão e o baixo custo computacional quando comparado com algumas das alternativas são fatores que motivam sua aplicação em UAVs [39][52][40][53][54], robôs terrestre [55][56][57], e até veículos autônomos submarinos [42]. Lee *et al.* [58] desenvolvem um sistema de vigilância para aplicações indoor baseado em drones que realizam patrulhas. Monajjemi *et al.* [59] apresentam uma interface gestual para controlar o comportamento de grupos de UAVs. Ling *et al.* [60] elaboram um sistema para aterrissagem de UAVs em embarcações marinhas. Em todos estes casos marcadores fiduciais são utilizados para realizar a localização do drone. Muñoz-Salinas *et al.* [61] desenvolvem um sistema de localização e mapeamento baseado em fiduciais. Sweatt *et al.* [62] utiliza WiFi em conjunto com GPS, INS e marcadores fiduciais para localização

de um robô autônomo destinado a inspeção de refinarias de óleo e gás. Olson *et al.* [63] apresentam um sistema de reconhecimento multi robô em que os robôs determinam suas posições referentes uns aos outros por meio de marcadores fiduciais.

2.3 Mapas

Mapas são representações do ambiente pelo qual o robô se desloca, que podem tanto ser fornecidas antes do começo da navegação quanto construídas pelo próprio robô. Embora nem todos os sistemas de localização e navegação façam o uso de mapas, como mencionado, estes auxiliam no processo de planejamento de trajetória e influenciam na definição da posição do robô. Existe uma série de aspectos do sistema de localização que induzem a forma como o mapa deve ser estruturado. Entre eles podemos citar as informações fornecidas pelos sensores do robô, a forma de representar a posição do robô e os métodos usados para estimar a localização. Existem diversas formas, compatíveis com estes aspectos, de se estruturar o mapa, cada uma com suas vantagens e limitações.

Mapas com maior complexidade, que representam o ambiente com maior fidelidade, possibilitam representações mais precisas de posicionamento. Porém podem ocupar muito espaço em memória e aumentar o custo computacional dos processos de mapeamento, localização e navegação. Por outro lado, mapas mais simplificados podem resolver estes problemas de memória e custo computacional, porém, em alguns casos, não permitem a definição precisa da posição do robô, apenas destacando uma região na qual ele se encontra. De forma geral, os diferentes mapas podem ser classificados conforme seu princípio básico (baseados em coordenadas ou informações relacionais) em mapas métricos ou mapas topológicos [3].

2.3.1 Mapas Métricos

Mapas métricos são caracterizados por desenvolverem uma representação quantitativa do ambiente. Normalmente fazendo o uso de sistemas de coordenadas para estabelecer as relações entre espaço livre e ocupado [3]. Estas representações são elaboradas com certo grau de seletividade e abstração, ao concentrarem-se em características de objetos que podem ser percebidas pelos sensores do robô, gerando informações relevantes para descrição espacial [2].

Um exemplo desta classe de mapas são os mapas geométricos, que utilizam elementos geométricos básicos como pontos, retas ou curvas para demarcar os limites entre espaço livre e ocupado. O uso de uma representação contínua do espaço permite o posicionamento destes elementos de forma arbitrária, possibilitando representações precisas do ambiente. Na Figura 4 pode-se observar um exemplo de uma representação geométrica. A simplicidade destes elementos básicos, junto com a possibilidade de aproximar objetos

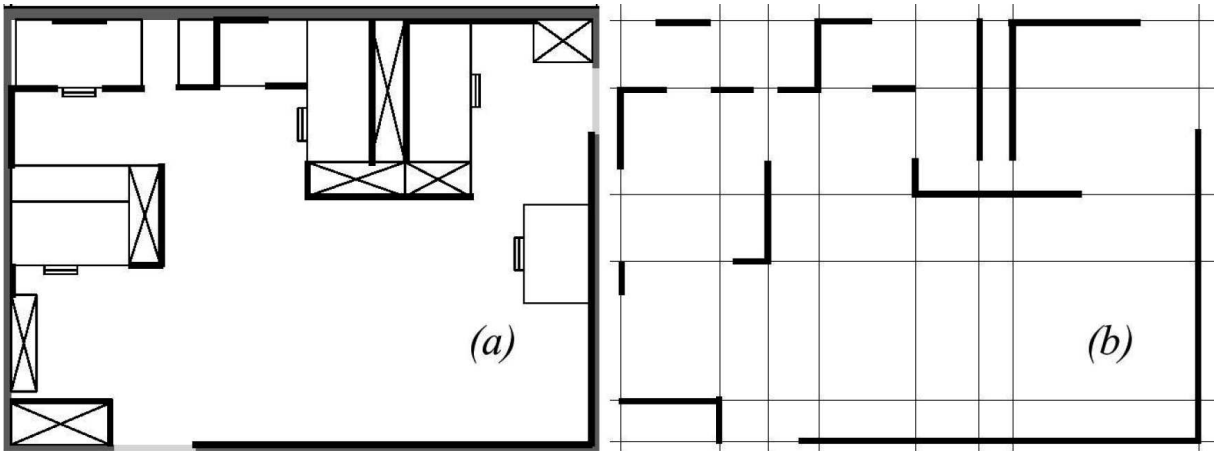


Figura 4 – Em (a) é apresentado o mapa real do ambiente de trabalho do robô e em (b) pode se observar uma representação geométrica do ambiente construída com retas infinitas [2].

por polígonos simples, permite sacrificar a fidelidade do mapa em troca de velocidade de processamento e reduzir espaço em memória. Estes mapas são adequados para processos de localização, porém sua natureza contínua pode resultar em maior custo computacional para certos métodos de planejamento de trajetória. Isto se deve a necessidade de segmentar a representação para fornecer um espaço discreto de procura aos métodos de planejamento de trajetória.

Uma alternativa às representações contínuas se dá pela aplicação de técnicas de decomposição. Estas técnicas buscam decompor o ambiente de trabalho do robô em uma coleção de regiões não-sobrepostas chamadas de células [64]. Dependendo do formato das células usadas para a decomposição é possível que a representação sofra perda de fidelidade em relação ao ambiente real. A decomposição por células exatas (*exact cell decomposition*), apresentada por Latombe [64], é uma forma de evitar esta perda de fidelidade. Esta decomposição tira proveito de criticidades geométricas para segmentar o ambiente em áreas geometricamente simples de espaço livre, como triângulos ou trapezóides. A possibilidade de armazenar cada célula individualmente significa que esta representação pode ser bastante compacta. Na Figura 5 é apresentada a decomposição exata de uma área de trabalho bidimensional.

Contanto que as células apresentem as duas características definidas por Latombe [64], pode-se assumir que o importante não é a posição do robô dentro das áreas de espaço livre, mas a sua capacidade de transitar de uma área para outra adjacente [2]. Isto implica que esta representação é capaz de capturar a conectividade do ambiente em conjunto com uma descrição fiel do mesmo. Esta conectividade é muito útil para processos de navegação. Porém a irregularidade das células implica em maior complexidade para algoritmos de planejamento de trajetória.

Seja por conta disto ou pela dificuldade de se obter as informações necessárias para

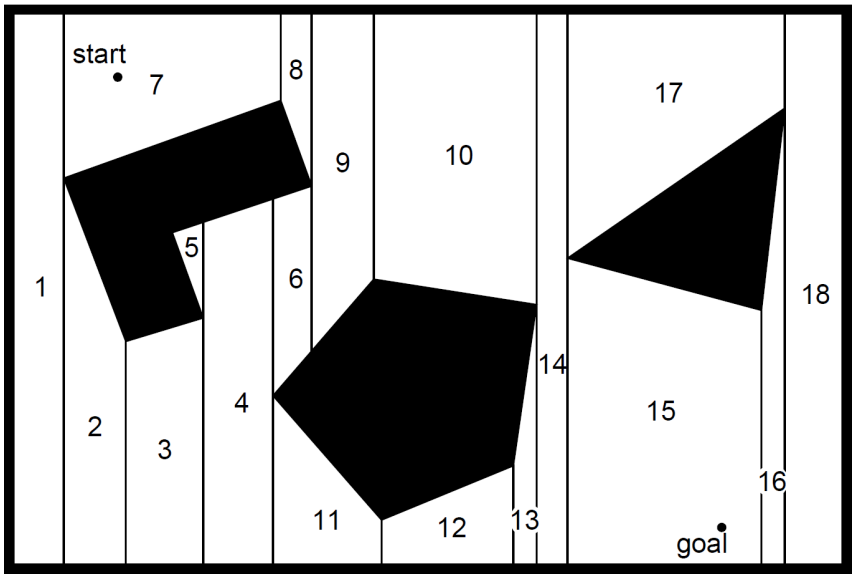


Figura 5 – Representação espacial do ambiente construída pela abordagem de decomposição exata [2].

construção de uma representação exata, a decomposição exata nem sempre é apropriada. Em vez disso, pode se realizar uma representação baseada em ocupação ao se decompor a área do ambiente em células e atribuir a cada uma um valor que represente se estão, pelo menos parcialmente, ocupadas ou livres. As células usadas para a decomposição são uniformes, apresentando tamanho e formato predefinido. Sua distribuição é independente da configuração dos elementos presentes no ambiente, normalmente assumindo a forma de uma grade composta por células quadradas. Em função destas características este tipo de representação recebe a nomenclatura de decomposição por células fixas (*fixed cell decomposition*) ou grade de ocupação (*occupancy grid*) [37][18]. As Figuras 6 e 7 representam o processo de discretização realizado pela decomposição por células fixas. A Figura 8 apresenta uma grade de ocupação.

Grades de ocupação oferecem descrições detalhadas do ambiente sem a necessidade de extração de informações complexas das leituras de sensores, o que acaba facilitando a modelagem da propagação de incertezas. O desenvolvimento de métodos de construção e manutenção do mapa também é simplificado, uma vez que estes processos precisam simplesmente atualizar o valor de ocupação de cada célula [3]. Também é possível representar obstáculos transientes com esta abordagem, o que possibilita seu uso em ambientes que apresentem certo comportamento dinâmico [3, 2]. Porém a forma como a representação é discretizada implica em problemas relacionados à sua resolução. Se as células possuem área muito grande é possível que a perda de informação faça com que caminhos presentes no ambiente sejam perdidos na representação. Em contrapartida, o uso de células com área pequena ou aplicações em ambientes muito grandes podem elevar drasticamente o consumo de memória da representação.

Outra abordagem de decomposição, denominada decomposição por células aproxima-

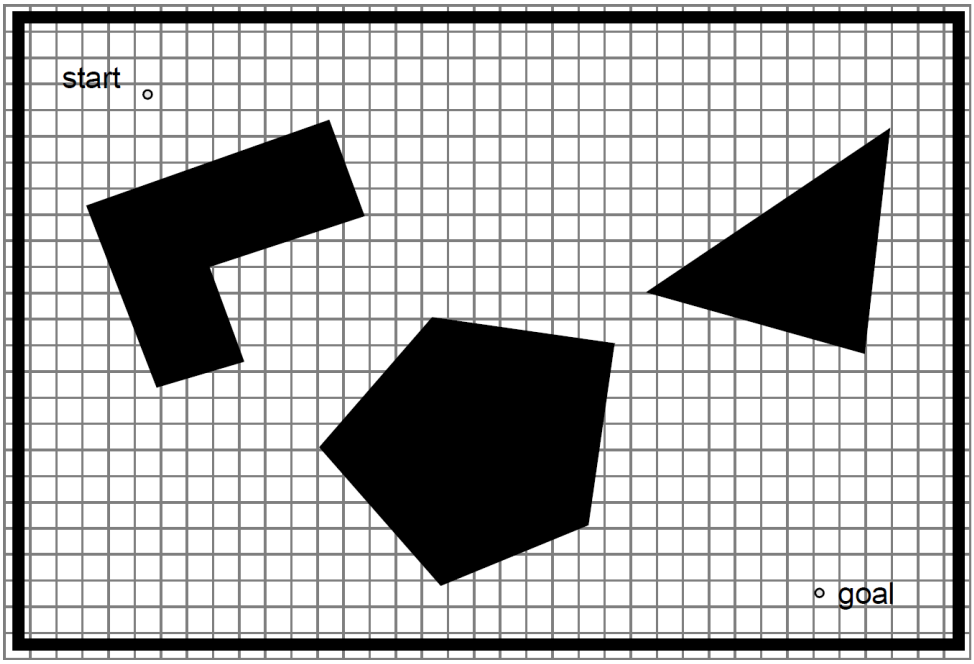


Figura 6 – Distribuição de células pela área do ambiente [2].

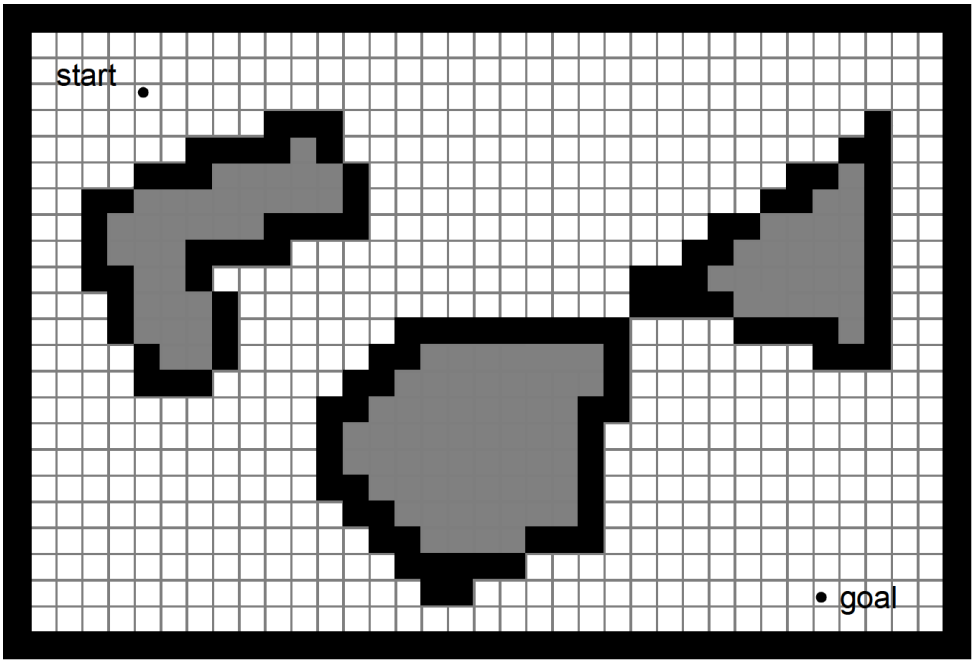


Figura 7 – Decomposição fixa do ambiente. Células brancas estão desocupadas, pretas parcialmente ocupadas e cinzas completamente ocupadas [2].

das (*approximate cell decomposition*) [64], lida melhor com a questão da resolução. Assim como a decomposição por células fixas, o formato das células é predefinido por esta abordagem, sendo normalmente retangulares ou quadradas, porém seu tamanho é variável. Com isso células parcialmente ocupadas são recursivamente decompostas em células menores, até se atingir um limite mínimo estabelecido para a área das células, gerando uma aproximação conservadora da representação exata do ambiente. Assim é possível geren-

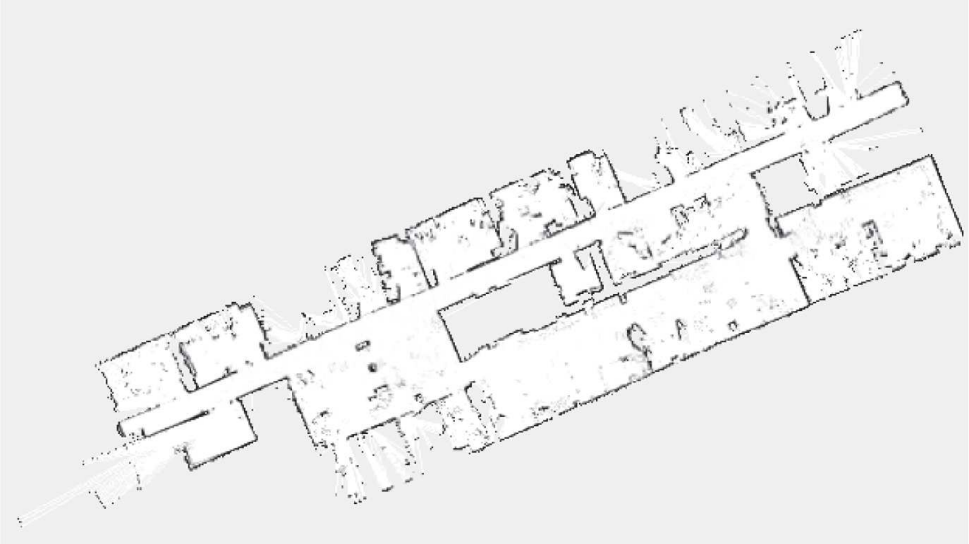


Figura 8 – Representação de *occupancy grid*. A probabilidade da célula estar ocupada é representada por seu tom de cinza, variando de branco (livre) até preto (ocupado) [3].

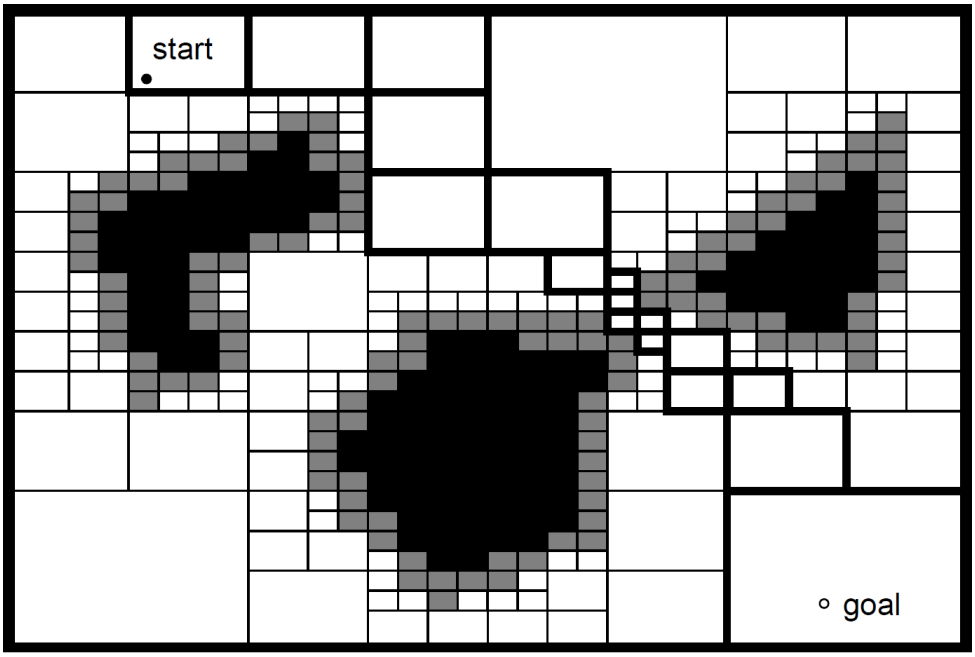


Figura 9 – Exemplo de decomposição aproximada [2].

ciar melhor o uso de espaço de armazenamento, economizando na descrição de grandes áreas totalmente livres ou ocupadas e obtendo-se maior resolução nas bordas dos objetos presentes no ambiente. Mesmo assim esta abordagem pode apresentar os mesmo problemas da decomposição por células fixas em ambientes muito complexos e caso o tamanho mínimo estabelecido para as células seja muito grande. Na Figura 9 é apresentada uma representação por decomposição aproximada.

2.3.2 Mapas Topológicos

Mapas topológicos são representações espaciais mais abstratas. Eles modelam o ambiente de forma qualitativa, evitando medidas diretas de propriedades geométricas do ambiente e focando em aspectos que caracterizam suas diferentes regiões, que são mais relevantes para localização. São baseados em modelos relacionais que buscam descrever a conectividade entre objetos ou regiões. Na maioria de suas aplicações, mapas topológicos têm sua implementação baseada em grafos. Estes grafos são compostos por nós e suas interligações. Nós são associados a regiões e interligações entre pares de nós representam alguma relação entre as regiões, como conectividade e adjacência. As interligações também representam alguma instrução relevante para a navegação do robô entre os nós.

Assim como nas representações por decomposição apresentadas anteriormente, a informação de adjacência é fundamental para mapas topológicos. Porém, diferentemente das células, os nós de um mapa topológico não possuem tamanhos e formatos específicos ou se quer descrevem regiões de espaço livre. Em vez disso, os nós registram áreas baseando-se em características que as distinguem umas das outras, do ponto de vista dos sensores utilizados pelo robô [2]. Isto implica que a representação pode ser adaptada conforme os sensores que estão a disposição do robô. Estas características podem ser intrínsecas do ambiente, como *features* visuais, ou podem ser artificiais, como *beacons*[1] com identificadores únicos distribuídos pelo ambiente. Tal representação implica em perda de clareza e expressividade quanto a posição atual do robô, dificultando ou impossibilitando uma localização mais precisa que a determinação da presença do robô na região de um nó. Contudo, a escassez de informações específicas e explícitas constitui uma de suas vantagens em relação a representações métricas, ao permitir a conservação da consistência de representações de ambientes cuja escala causaria problemas para mapas métricos.

Os mapas topológicos podem ser subdivididos em duas categorias em função de sua estrutura e como os nós são associados às regiões do ambiente. Uma delas são os mapas denominados *view graphs* [19][65][35]. Nestas representações é construída uma rede com alta densidade de nós que são distribuídos mais ou menos uniformemente por todo o ambiente de trabalho do robô. Os nós desta rede são associados com vistas (*views*), que se tratam das informações dos sensores do robô em uma localização particular. As conexões entre nós representam consecutividade na percepção das vistas, o que acarreta em relações de adjacência espacial. Elas também armazenam as informações necessárias para o deslocamento do robô entre os nós que interligam [3]. A alta densidade de nós da representação é necessária para permitir o deslocamento confiável do robô entre nós. Este movimento é realizado por técnicas como *visual homing* [66], que requerem a sobreposição das áreas de captação individuais dos nós [3]. A Figura 10 apresenta uma sobreposição de um *view graph* sobre o ambiente representado, exibindo os nós em suas respectivas

[1] Dispositivo transmissor de rádio cujo sinal é utilizado para localização de veículos.

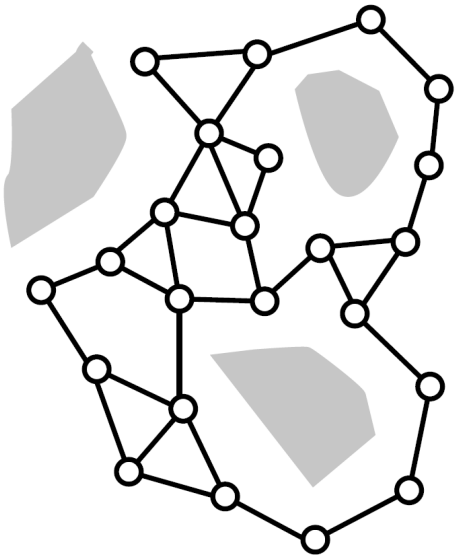


Figura 10 – Representação *view graph* sobreposta ao ambiente representado [3].

posições no ambiente.

Esta representação é universalmente aplicável, contanto que os dados dos sensores contenham informações suficientes para evitar problemas de *aliasing*. Sua estrutura proporciona os requisitos para localização e navegação com êxito, porém estes mapas possuem algumas desvantagens. A primeira está relacionada com os trajetos encontrados para navegação do robô, que não são ótimos devido a disposição dos nós. Este problema é amplificado nos casos em que técnicas como *exact homing* são usadas em cada passo do deslocamento do robô [3]. Outra desvantagem é a possibilidade de problemas de escalabilidade causada pela densidade mínima de nós necessários. A representação também falha em capturar informações estruturais do ambiente e o modelo varia conforme os sensores e capacidades motoras do robô. Isso faz com que esta representação não seja adequada para exploração sistemática e comunicação [3].

A outra categoria são os mapas denominados *route graphs* [31], cujo conceito foi apresentado por Werner *et al.* [7] como um modelo genérico e interdisciplinar para a estruturação de conhecimento de navegação baseado em rotas. Diferentemente dos *view graphs*, que associam nós a qualquer posição no ambiente que ofereça uma vista singular, *route graphs* representam o ambiente construindo uma rede de rotas distintas que o percorrem. Nestas representações o posicionamento dos nós é induzido por locais ou pontos de referência significativos do ambiente encontrados ao longo das rotas. Os nós atuam como subobjetivos para navegação e pontos de tomada de decisão referente a próxima ação do robô. Enquanto isso, as conexões entre nós destacam os diferentes caminhos que os conectam. Elas também realizam a caracterização destes caminhos por meio de sequências de ações ou comportamentos adotados pelo robô para percorrê-los. Esta diferença em sua estrutura implica nos *route graphs* refletindo diretamente a topologia do ambi-

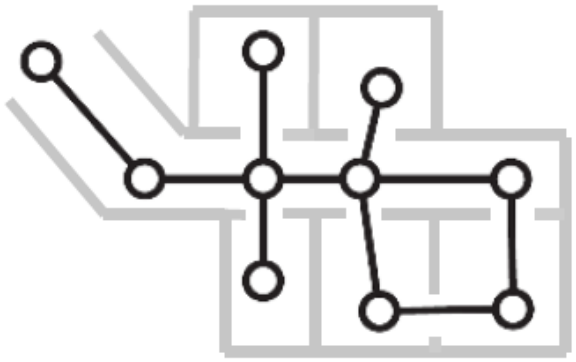


Figura 11 – Representação *route graph* sobreposta ao ambiente representado [3].

ente. A Figura 11 demonstra isso ao sobrepor um *route graph* ao ambiente *indoor* que ele representa.

Route graphs constituem representações compactas do ambiente, quando comparados com *view graphs*. Elas são apropriadas para navegação, como esperado considerando a origem de seu conceito, fornecendo um espaço de pesquisa de tamanho reduzido para métodos de planejamento de trajetória. Além disso, representar as rotas com pontos de tomada de decisão explícitos é conveniente para tarefas de comunicação baseadas em rotas [67] [3]. Além disso, nos casos em que o robô tem a capacidade de distinguir as diferentes interconexões que estão ligadas ao mesmo nó, a exploração sistemática do ambiente é simplificada, pois basta se observar quais faltam ser percorridas para abranger o ambiente por completo [3]. Em contrapartida, alguns processos podem ser problemáticos para estes mapas. Localização, por exemplo, é um processo que se torna problemático em ambientes que apresentam pobreza de características úteis para distinção de nós e interconexões, conforme apresentado por Dudek [68] e Rekleitis [69]. A aplicação deste tipo de representação à ambientes pouco estruturados ou não estruturados, como um campo aberto por exemplo, também não é apropriada. O ambiente precisa apresentar certo grau de segmentação e estruturação para que a elaboração de rotas seja possível.

2.4 Reconhecimento Ótico de Texto

Reconhecimento ótico de caracteres é um problema definido pela identificação e rotulação de caracteres presentes em uma imagem. Ye e Doermann [70] afirmam que os objetivos de detecção, localização e reconhecimento de caracteres são essenciais para sistemas de ponta a ponta que buscam realizar o reconhecimento de texto em cenas naturais. A detecção de texto caracteriza a determinação da presença ou ausência de elementos textuais em uma imagem. A localização analisa a imagem para isolar componentes textuais precisamente e agrupá-los em candidatos à regiões textuais, minimizando o plano de fundo presente [70]. O reconhecimento propriamente dito trata-se da conversão das regiões da

imagem isoladas pela localização em *strings*.

Entre os métodos mais popularmente usados em pesquisas para realizar a localização de texto, temos a análise de componentes conectados (*connected components analysis*). Este método é adotado por Koo e Kim [71] e combinado com um classificador AdaBoost para gerar candidatos a regiões textuais. Os componentes conectados são extraídos usando o algoritmo de *Maximally Stable Extremal Regions* (MSERs) e então são agrupados pelo classificador AdaBoost, diferentemente de métodos convencionais que usam heurística para o agrupamento. Os candidatos à regiões textuais são normalizados e então filtrados por um classificador baseado em uma rede neural que diferencia regiões textuais de regiões não textuais. Outro método bastante usado é a classificação por janela deslizante (*sliding window classification*). Wang *et al.* [72] aplicam este método em seu sistema de ponta a ponta para encontrar caracteres na imagem e então os agrupam usando um algoritmo de *beam search*.

Além do método, as *features* empregados para localização de texto variam entre diferentes trabalhos. *Features* de cor, borda e textura foram utilizados por sistemas convencionais, porém uma grande variedade de *features* diferentes foram explorados [70]. Entre elas as *features* baseadas em MSER particularmente apresentam bons resultados. O fato dos caracteres normalmente apresentarem cores contrastantes com seu plano de fundo e os algoritmos de MSER adaptativamente detectarem regiões de cores estáveis os torna uma boa alternativa para encontrar possíveis caracteres. A abordagem que atingiu melhor performance durante a competição ICDAR'13 utilizou estas *features* [73].

Entre os desafios listado por Ye e Doermann [70] para a detecção e reconhecimento de texto, estão a orientação e curvatura do texto. Estas características apresentam dificuldades por causa das representações limitadas normalmente adotadas para o texto detectado que assumem a forma de retângulos ou quadrângulos. Buscando lidar com isso, Long *et al.* [74] desenvolveram uma nova representação capaz de delimitar corretamente regiões textuais em uma imagem mesmo quando apresentam curvatura.

O reconhecimento do texto em si pode ser realizado reconhecendo caracteres individualmente ou reconhecendo palavras. Shi *et al.* [75] realizam a localização e reconhecimento de caracteres usando classificação por janela deslizante e modelos deformáveis. Durante o reconhecimento, os modelos podem ser deformados para aumentar sua similaridade com os caracteres, porém isto implica em penalidades na função de custo empregada na comparação. Weinman *et al.* [76] usaram a integração de informações de similaridade, antecedentes linguísticos e decisões lexicais para compor um modelo de inferência probabilística para reconhecimento de texto. Seu trabalho inspirou Feild [77] no desenvolvimento de um sistema de ponta a ponta para detecção e reconhecimento de texto. Este sistema realiza o reconhecimento de texto com o motor de OCR Tesseract [78] e um método próprio desenvolvido.

2.5 Trabalhos Relacionados

A aplicação de reconhecimento de caracteres para localização de robôs implica na integração de informações textuais presentes no ambiente percebidas pelo robô no processo de localização. Embora autores que abordam detecção e reconhecimento de texto, como [71], sugerem a possibilidade desta aplicação, trabalhos que abordam esta aplicação em particular são escassos na literatura quando comparados com outras metodologias de localização. Ainda assim, diferentes estratégias para a integração destas informações textuais em localização foram apresentadas.

Tomono e Yuta [79] propõe um sistema de navegação que utiliza o reconhecimento de portas e caracteres de suas placas de identificação para navegar até um destino. A navegação é realizada percorrendo um corredor em busca de portas. Quando uma porta é encontrada o robô busca se posicionar para ler a placas de identificação. Se a leitura corresponder com o destino o robô se posiciona em frente a porta, caso contrário ele continua percorrendo o corredor.

Mata *et al.* [80, 81, 82] desenvolvem um sistema de reconhecimento de marcadores visuais para aplicação em navegação topológica de robôs. O sistema detecta marcadores com um algoritmo genético e os identifica com um motor de OCR próprio baseado em uma rede neural *feed forward*. Posteriormente o sistema é adaptado para que possa aprender novos marcadores por meio de um processo de treinamento. O sistema foi então aplicado para realização de tarefas de navegação topológica como localização e navegação até um destino específico.

Case *et al.* [83] propõe um sistema de navegação que utiliza o reconhecimento de texto em imagens do ambiente para integrar anotações semânticas em um mapa métrico. Estas anotações podem ser usadas para o direcionamento da navegação do robô por comandos textuais. A detecção de texto é feita por meio de *features* de variância local, densidade de bordas local e intensidade de bordas verticais e horizontais. O reconhecimento de caracteres é realizado pelo motor de OCR open source Tesseract [78].

Um sistema de navegação simbólica capaz de exploração direcionada ao objetivo é apresentada por Schulz *et al.* [84]. Este sistema utiliza o reconhecimento de caracteres de placas de identificação de salas para relacioná-las com um modelo de sua distribuição por um ambiente não explorado. Esta informação é então usada para conduzir a exploração do ambiente mais diretamente para o objetivo quando comparada com uma exploração aleatória.

3 Sistema de Localização

Neste capítulo descrevemos um sistema de localização topológica de robôs baseado no reconhecimento ótico de marcadores textuais intrínsecos ao ambiente. Com o desenvolvimento deste sistema buscamos explorar a aplicação dos avanços no campo de reconhecimento ótico de caracteres (OCR) e reconhecimento de texto em cenas (*scene text recognition*) à localização de robôs, avaliando o desempenho e viabilidade do sistema. A aplicação deste sistema é destinada à ambientes indoor estruturados. Os marcadores utilizados para localização são placas de identificação padronizadas, possuindo formato retangular, compostas por cores contrastantes e a fonte utilizada não apresenta estilização, como observado na Figura 12. O sistema é composto por um único software integrado desenvolvido em linguagem C++. Sua arquitetura é composta por: i) um processo de detecção de possíveis marcadores, ii) uma interface com um motor de OCR de código aberto para identificação dos marcadores e iii) um processo que combina as informações obtidas dos marcadores com um mapa do ambiente de trabalho do robô para determinar a sua localização. Um fluxograma representando esta arquitetura é apresentado na Figura 13.

Cada uma das partes do sistema foi desenvolvida para iterar sobre um vetor de entradas e retornar um vetor de resultados. Desta forma, se alguma situação, como nenhuma possível placa ser detectada na imagem, levar algum dos processos a gerar um vetor de resultados vazio, o funcionamento do sistema não é comprometido. Além disso, caso um processo gere um vetor de resultados vazio, os processos seguintes não desperdiçam tempo e o sistema pode rapidamente retornar para a etapa de detecção.

Figura 12 – As placas de identificação da sala na imagem são usadas como marcadores pelo sistema para realizar sua localização.

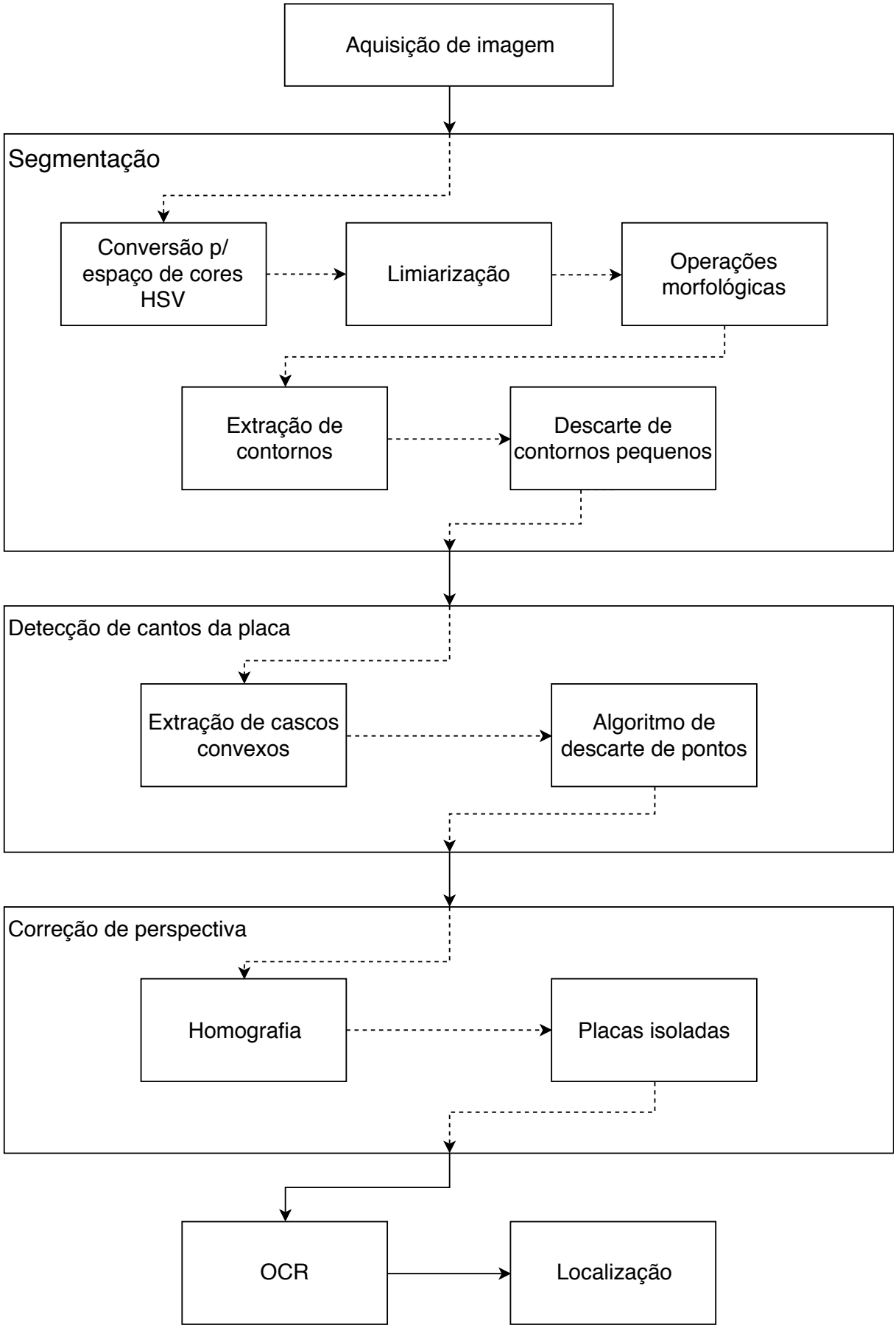


Figura 13 – Fluxograma do sistema de localização desenvolvido.

3.1 Detecção de Marcadores

O processo de detecção procura por elementos presentes nas imagens obtidas pela câmera do robô que apresentem características semelhantes às placas de identificação usadas como marcadores. Caso encontre algum, este elemento é isolado e tratado por técnicas de processamento de imagem antes de ser passado para a interface com o motor de OCR. Isso é realizado para proporcionar condições mais favoráveis para a atuação do motor de OCR, com o intuito de aprimorar os resultados de suas leituras. Tendo em vista a complexidade de tarefas básicas realizadas pelo sistema, como processamento de imagens, optou-se pelo uso de bibliotecas e sistemas com desempenho bem estabelecido e que são amplamente utilizados pela comunidade científica.

OpenCV [85] é uma biblioteca de visão computacional e *machine learning* amplamente utilizada pela comunidade científica, desenvolvida como código aberto tendo o intuito de oferecer uma infraestrutura comum para aplicações de visão computacional. Esta biblioteca possui uma ampla gama de aplicações, incluindo reconhecimento de faces, classificação de ações de pessoas em vídeos, seguimento do movimento dos olhos, geração de nuvens de pontos 3D, extração de modelos 3D de objetos, identificação de objetos, *tracking* de objetos em movimento, união de imagens para montagem de vistas panorâmicas e estabelecimento de marcadores para realidade aumentada entre outras. Isto se deve aos mais de 2500 algoritmos otimizados que compõe a biblioteca, incluindo tanto métodos clássicos quanto métodos do estado-da-arte de visão computacional e *machine learning*.

Neste projeto, a biblioteca OpenCV será utilizada para processamento de imagem, segmentação e cálculo de homografia. Isso inclui a detecção, isolamento e correção de perspectiva de possíveis marcadores na etapa de detecção além de auxiliar na etapa de rejeição de falsos positivos. A versão da biblioteca OpenCV a ser usada é a 3.4.1.

3.1.1 Segmentação

A primeira etapa da detecção de marcadores é a segmentação da imagem, isolando os elementos presentes nela que apresentem características de interesse. Neste caso optou-se por usar a cor como base da segmentação inicial, considerando que a cor das placas às destaca do ambiente e poucos objetos diferentes presentes no ambiente compartilham a mesma cor. Para facilitar a segmentação, a imagem é convertida para o espaço de cores HSV que modela a cor de cada pixel com valores de matiz (*Hue*) que define a cor, saturação (*Saturation*) que define a saturação da cor e intensidade (*Value*) que define o brilho. Uma representação visual deste espaço de cores pode ser observado na Figura 14. Feita a conversão, as regiões de interesse (ROI) podem ser encontradas checando quais pixels se encontram dentro de uma faixa de matiz e saturação específicas, o que resulta em uma imagem binária como a da Figura 15. Para reduzir o ruído observado na Figura 15 são realizadas operações morfológicas de abertura e fechamento, resultando em uma imagem

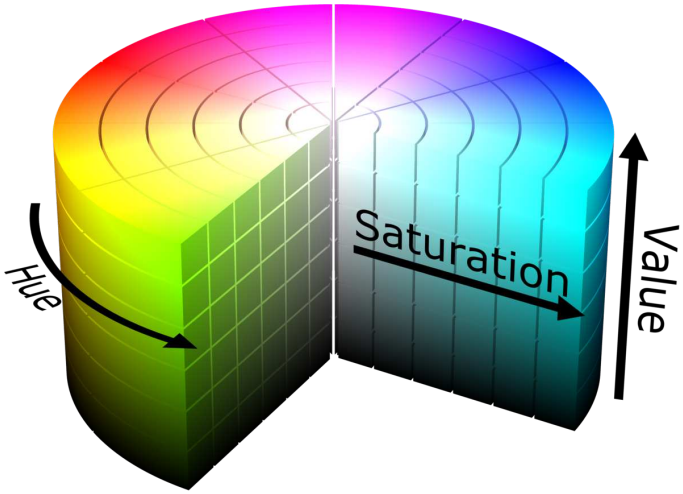


Figura 14 – Representação gráfica do espaço de cores HSV [4].

como a da Figura 16 que apresenta um aspecto mais limpo, apesar de ainda apresentar regiões que não estão associadas à placas de identificação.

Como o objetivo final do processo de detecção é fornecer para a interface com o motor de OCR uma imagem da placa de identificação isolada do ambiente e com sua perspectiva corrigida, é necessário encontrar seus cantos com precisão para calcular a matriz de homografia a ser aplicada. A forma de detecção dos cantos que empregamos baseia-se na análise do contorno da placa. Sendo assim, a etapa final da segmentação é encontrar os contornos das placas presentes na imagem a partir das ROI da Figura 16. Estes contornos são estruturados na forma de um vetor C_n, definido na Eq. (3.1), contendo as coordenadas P_m, definidas em (3.2), de cada ponto da imagem que compõem o contorno

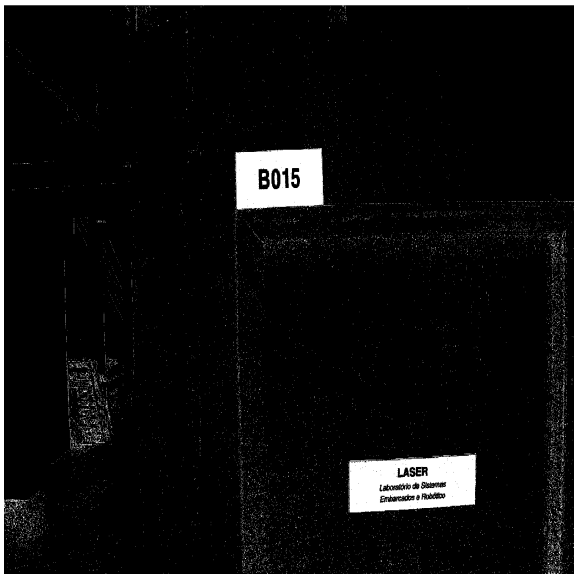


Figura 15 – Resultado da limiarização de matiz e saturação.

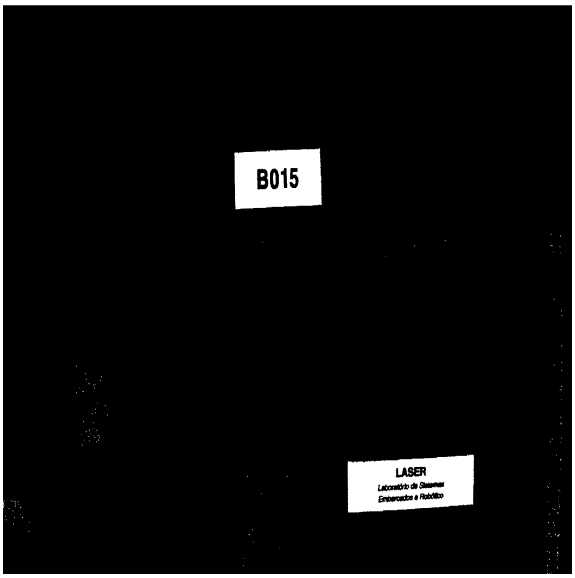


Figura 16 – Imagem binária após operações morfológicas.

Figura 17 – Contornos extraidos das ROI, em vermelho, sobrepostos a imagem original.

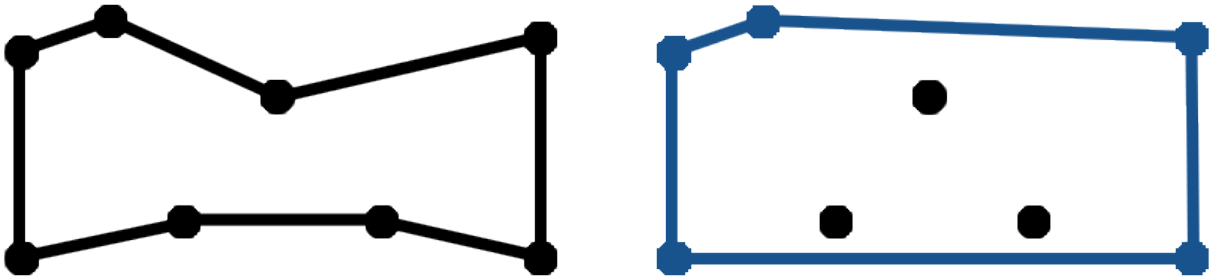
da respectiva ROI. Porém, se simplesmente extrairmos os contornos das ROI, como observado na Figura 17, teremos vários contornos que não estão relacionados com placas. Estes em sua maioria podem ser filtrados ao se descartar contornos que não apresentem uma área mínima. Assim também são descartadas placas de identificação que estão muito longe para se realizar uma boa leitura. Os contornos restantes prosseguem para a etapa de detecção de cantos.

$$C_n = \{P_1, P_2, ..., P_m, ..., P_M\} \tag{3.1}$$

$$P_m = \{X_m, Y_m\} \tag{3.2}$$

3.1.2 Detecção de Cantos

As operações morfológicas realizadas na etapa de segmentação costumam gerar concavidades nas ROI. Estas concavidades são transmitidas para os contornos e podem interferir no processo de detecção de cantos. Para corrigir isto é extraído o casco convexo de cada contorno usando a função *convexHull()* da biblioteca OpenCV. Dado um conjunto de pontos C_n que forma um contorno genérico como na Figura 18 (a), o casco convexo H_n deste contorno é o menor polígono convexo que cerca todos os pontos de C_n. Como pode-se observar na Figura 18 (b), H_n, que respeita (3.3), é composto apenas pelos pontos P_m do



(a) (b)

Figura 18 – Um conjunto de pontos que forma um contorno genérico é apresentado em (a). O casco convexo extraído do contorno é apresentado em (b).

Figura 19 – Cascos convexos obtidos a partir dos contornos filtrados, em azul, sobrepostos a imagem original.

respectivo contorno de forma a omitir as concavidades. Na Figura 19 pode-se observar os cascos convexos extraídos dos contornos obtidos pela segmentação. Em geral, os cascos convexos contém pontos que correspondem aos cantos da placa ou que são suficientemente próximos. O problema é determinar quais são estes pontos.

$$H_n = \{P_1, P_2, ..., P_m, ..., P_M\} \mid H_n \subset C_n \tag{3.3}$$

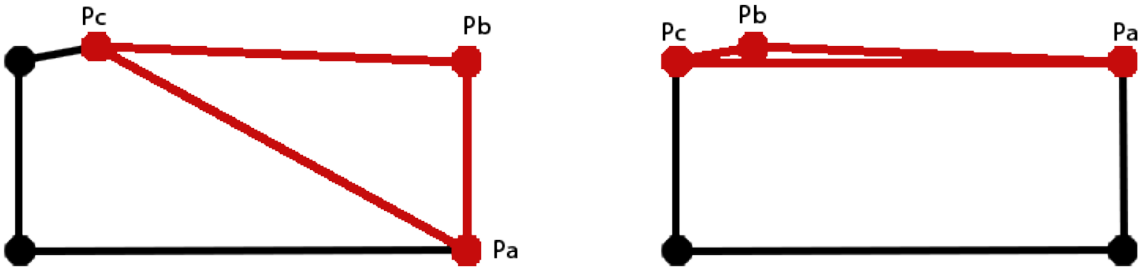
Para isso foi implementada uma adaptação do algoritmo de Visvalingam-Whyatt [86]. O Algoritmo 1 descarta iterativamente os pontos do casco que exercem menos influência

sobre seu formato até restar apenas uma quantidade especificada de pontos. O algoritmo realiza isso selecionando um grupo de três pontos consecutivos contidos em H_n, que chamaremos de P_a (anterior), P_b (atual) e P_c (próximo). Estes três pontos formam um triângulo, como visto na Figura 20 (a), no qual o lado $\overline{P_aP_c}$ é considerado a base e a distância entre esta e o ponto P_b define a altura do triângulo. O algoritmo então calcula a altura h do triângulo. Esta altura h é comparada com um valor h_{min}. Se h for menor que h_{min}, o valor de h é atribuído a h_{min} e o índice do ponto P_b atual é armazenado. Em seguida o algoritmo passa para o próximo grupo de pontos, como visto na Figura 20 (b), em que P_b se torna P_a, P_c se torna P_b e o próximo ponto é atribuído a P_c. Quando o algoritmo conclui o ciclo passando por todos os pontos de H_n, o ponto que gerou a menor altura, representada por h_{min}, cujo índice permaneceu armazenado é descartado e o ciclo é reiniciado. Para encontrar os cantos da placa, o algoritmo repete este processo até que restem 4 pontos no casco, os quais são passados para a etapa de correção de perspectiva. A Figura 21 mostra os resultados da detecção de pontos.

Algorithm 1 Algoritmo de Visvalingam-Whyatt adaptado.

Input: Vetor H_n de pontos conssecutivos que formam um casco convexo.
Output: Vetor H_n contendo apenas 4 pontos que correspondem aos cantos.

Inicialização :
$l \leftarrow$ tamanho de H_n
Processo em LOOP
while $l > 4$ **do**
  $h_{min} \leftarrow \infty$
  $i_{min} \leftarrow -1$
  for $i = 0$ to l **do**
    if $(i = 0)$ **then**
      $P_a \leftarrow H_n[l-1]$
    else
      $P_a \leftarrow H_n[i-1]$
    end if
    if $(i = l-1)$ **then**
      $P_c \leftarrow H_n[0]$
    else
      $P_c \leftarrow H_n[i+1]$
    end if
    $P_b \leftarrow H_n[i]$
    computar h para o triângulo P_a, P_b, P_c
    if $(h < h_{min})$ **then**
      $h_{min} \leftarrow h$
      $i_{min} \leftarrow i$
    end if
  end for
  remover $H_n[i_{min}]$ de H_n
  $l \leftarrow l - 1$
end while



(a) (b)

Figura 20 – O triângulo formado pelos pontos selecionados em uma iteração do algoritmo de detecção de cantos é apresentado em (a). O triângulo formado pelos pontos selecionados na iteração seguinte é apresentado em (b).

Figura 21 – Cantos detectados, em vermelho, e contorno formado por eles, em verde, sobrepostos a imagem original.

Foram levadas em consideração algumas alternativas para a detecção dos cantos. Entre elas estavam a detecção das retas que compõem as laterais das placas com a transformada de Hough, seguida da detecção das interseções destas retas. Outro opção testada foi a aproximação poligonal pelo algoritmo de Ramer-Douglas-Peucker. A aproximação de segmentos do casco convexo por retas também foi testada. Porém quando comparadas com os resultados do algoritmo aplicado, os ganhos em precisão não eram suficientemente significativos para justificar o maior custo computacional. Ou, como no caso do algoritmo de Ramer-Douglas-Peucker, os cantos encontrados eram incorretos em várias situações.

3.1.3 Correção de Perspectiva

Esta é a etapa final do processo de detecção de marcadores. Dependendo da postura relativa entre a câmera do robô e o marcador, o texto contido no marcador pode apresentar severa deformação causada pela perspectiva. Como mencionado anteriormente, o objetivo desta etapa é a correção desta deformação buscando facilitar o reconhecimento dos caracteres pelo motor de OCR.

Inicialmente é usada a função *findHomography()*, da biblioteca OpenCV, para se encontrar a matriz de homografia H. Os argumentos usados pela função são o vetor de cantos encontrados na etapa de detecção de cantos e um vetor contendo coordenadas dos cantos de um retângulo padronizado. É importante que ambos vetores de pontos estejam ordenados de forma que um canto da placa e seu respectivo canto do retângulo padronizado estejam na mesma posição em seus respectivos vetores. Caso a ordenação não esteja correta, a correção de perspectiva pode gerar uma imagem da placa torcida, invertida ou rotacionada. Para garantir a ordenação correta dos vetores, a posição de cada canto em relação ao centróide da placa e sua ordem no vetor de cantos é levada em consideração na geração do vetor de coordenadas padronizadas. Assim, mesmo que a placa esteja rotacionada em torno do eixo ortogonal ao seu plano, a correção pode ser realizada. Rotações com ângulo próximo ou superior a 90º ainda representam problemas para correção de perspectiva, porém estas situações não são esperadas durante a operação normal do robô.

O valor retornado pela função *findHomography()* é uma matriz de homografia H de tamanho 3x3. Esta matriz possui 8 graus de liberdade (3 de rotação, 3 de translação e 2 de escala) e descreve a transformação de pontos entre dois planos, conforme demonstrado pela equação (3.4) [87]. No caso da correção de perspectiva os planos são o plano da imagem e o plano da placa. Esta matriz e a imagem original adquirida pela câmera são passadas como argumentos para a função *warpPerspective()* da biblioteca OpenCV. Ela aplica a transformação descrita pela matriz de homografia à imagem original. Isso mapeia os pontos da placa na imagem original para o sistema de coordenadas do retângulo padronizado, distorcendo a imagem original de forma a corrigindo a distorção causada pela perspectiva na placa. O retorno desta função é uma imagem da placa isolada do resto da imagem original com sua perspectiva corrigida como visto na Figura 22.

$$p_n = \begin{pmatrix} x_n \\ y_n \\ 1 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x'_n \\ y'_n \\ 1 \end{pmatrix} = Hp'_n \tag{3.4}$$

Figura 22 – Placa após correção de perspectiva.

3.2 Interface com Motor de OCR

A interface com o motor de OCR constitui a parte do sistema que realiza a leitura e identificação das placas. Ela recebe as imagens das placas isoladas geradas pela detecção de marcadores e realiza uma última bateria de tratamentos antes de fornecer-las para o motor de OCR. As leituras geradas são então avaliadas e classificadas como boas ou ruins. Os elementos detectados pelo sistema como possíveis placas de identificação que não são realmente placas (falsos positivos) e leituras ruins são descartados enquanto as leituras boas são passadas para o processo de localização.

As imagens das placas isoladas passam por um processo de limiarização para aumentar o contraste entre os caracteres e o fundo da placa. São geradas duas imagens binárias, uma com uma limiarização global e outra com uma limiarização adaptativa. Isto é feito pois dependendo das condições de iluminação da imagem, uma das técnicas apresenta resultados melhores que a outra. Ambas imagens binárias são então alimentadas ao motor de OCR Tesseract.

Tesseract [78] é um motor de reconhecimento óptico de caracteres, originalmente desenvolvida pela Hewlett-Packard de forma fechada. Em 2005 seu código foi aberto e desde 2006 vem sendo desenvolvido pela Google. O motor implementa uma rede neural para realizar o reconhecimento de caracteres, contando com suporte para unicode (UTF-8), sendo capaz de reconhecer mais de 100 linguagens, incluindo equações matemáticas. A rede neural ainda apresenta a possibilidade de ser treinada para reconhecer linguagens que não façam parte de seu repertório inicial. Para sua integração em projetos de terceiros, Tesseract possui uma API compatível com as linguagens C++ e Python. Tesseract pode ser compilado para Windows, Linux, macOS, Android ou iOS.

Neste projeto foi empregada a versão 3.05.02 do motor de OCR. A versão 4.0 implementa uma rede neural do tipo *Long Short Term Memory* (LTSM). Conforme seus desenvolvedores, esta versão apresenta melhores resultados quando comparada com versões anteriores. Porém, como esta versão se encontra em desenvolvimento ativo, optou-se pelo uso da versão anterior por preocupações com a estabilidade do sistema.

Inicialmente a API do Tesseract é usada para detectar e separar as linhas de texto presentes nas placas. As placas que contêm os códigos das salas possuem apenas uma linha de texto, mas as placas com os nomes das salas podem apresentar até três linhas de texto. Cada linha é então lida separadamente. Os resultados das leituras são uma *string*

contendo os caracteres reconhecidos e um valor de confiança associado a leitura que varia de 0 a 100. O valor de confiança é usado pela interface para determinar a leitura com menor chance de apresentar erros entre as leituras das duas imagens binárias.

Como a correção de perspectiva é realizada levando em consideração o formato das placas, outros elementos falsamente detectados como placas acabam sendo severamente distorcidos. Mesmo que contenham conteúdo textual, este é degradado a ponto do Tesseract não conseguir realizar seu reconhecimento. Ainda que consiga, o valor de confiança associado a leitura tende a ser pequeno. Assim, ao realizar a classificação das leituras baseando-se na confiança, a interface descarta tanto leituras ruins quanto falsos positivos. Após a identificação de todas as placas detectadas e o descarte de leituras ruins e falsos positivos os resultados são passados para a etapa final do sistema para realizar a localização.

3.3 Localização

Constituindo a etapa final do sistema, o processo de localização busca determinar o posicionamento do robô referente a uma representação topológica do ambiente de trabalho. Esta representação é composta por nós que são associados a regiões do ambiente. Cada nó contém um código de identificação e as *strings* de texto das placas que o identificam. Uma placa identifica um nó quando ela pode ser percebida pelo robô que se encontra dentro da região associada ao respectivo nó. As interconexões entre nós não foram especificadas. Elas não são usadas pelo processo de localização desenvolvido, além disso, suas estruturas devem ser elaboradas levando em consideração a estratégia de navegação empregada e a arquitetura do sistema de operação do robô. Como não abordamos o desenvolvimento de uma estratégia de navegação e não realizamos testes com um robô, não seria útil elaborar estas interconexões.

A localização busca identificar qual dos nós do mapa representa a área em que o robô se encontra. Isto é efetuado com a comparação das *strings* das leituras das placas observadas pela câmera com as *strings* das placas associadas a cada nó. Inicialmente pretendia-se buscar por *strings* idênticas para estabelecer a localização. Porém, como na maioria dos casos em que um zero estava presente na placa ele era reconhecido pelo Tesseract como um O, grande parte das leituras não resultaria em uma localização. Para contornar isso optou-se por realizar a comparação pela distância de Levenshtein [88] e condições heurísticas.

A distância de Levenshtein caracteriza a quantidade de caracteres que precisam ser alterados, adicionados ou removidos para que as duas strings comparadas sejam idênticas. As condições heurísticas tentam determinar se a leitura representa um caso em que um zero foi reconhecido como um O. Elas levam em consideração características como o

comprimento das strings comparadas, a distância de Levenshtein entre elas e a presença de caracteres específicos em posições específicas.

O algoritmo recebe um vetor de leituras da interface com o motor de OCR e compara cada leitura com todas as *strings* de placas de todos os nós buscando pela menor distância de Levenshtein que puder encontrar. Objetos do tipo *match* são usados para relacionar as leituras com placas e nós. Eles podem ser interpretados como a afirmação "Esta leitura corresponde a esta placa e indica que o robô está localizado neste nó.". Suas estruturas são compostas por variáveis que armazenam o índice das respectivas leituras, o índice das placas com as quais correspondem e o código de identificação dos nós do mapa em que se localizam.

Distâncias maiores que 1 não são aceitas para estabelecer uma localização. Quando uma comparação resulta em uma distância de 1, a leitura e a placa são relacionadas pelo respectivo objeto *match*. Caso uma comparação da mesma leitura com outra placa resultar em uma distância de 0, que indica que as *strings* são idênticas, ou resultar em uma distância de 1 e satisfazer as condições heurísticas estabelecidas, a leitura é relacionada com a placa da comparação mais recente atualizando os valores do respectivo *match*.

4 Testes e Resultados

Para analisar o desempenho do sistema, foi realizada uma bateria de testes que consistiu da execução do sistema utilizando um *dataset* de 66 imagens. A Figura 23 apresenta alguns exemplos destas imagens. No *dataset* estavam presentes imagens com a intenção de testar os efeitos da distância entre a câmera e a placa, rotações dos tipos *roll*, *pitch* e *yaw* da câmera em relação a placa, e presença de objetos semelhantes a placas. A aquisição das imagens do *dataset* foi feita usando uma câmera Basler acA2040-120uc [89] com resolução de 2048x1536 e taxa de 120 *frames* por segundo. A câmera foi equipada com uma lente TAMRON M118FM06 [90] com distância focal de 6mm e abertura variável de 1.4 a 16. A câmera estava conectada a um notebook por meio de um cabo USB 3.0. O hardware usado para os testes do sistema foi um computador desktop com sistema operacional Windows 10 Pro 1809 64 bits, memória RAM de 8,00 GB e processador Intel Core i5-3330 3.00 GHz. É importante ressaltar que o desempenho de todos estes testes são afetados por condições como a câmera usada, as proporções dos marcadores, condições de iluminação e demais condições que afetem a qualidade das imagens da câmera.

4.1 Teste do Processo de Detecção

A análise do desempenho da detecção foi realizada pela observação da quantidade de placas presentes em cada imagem, quantas detecções foram realizadas, quantas destas realmente eram placas, quantas eram falsos positivos e quantas placas presentes na imagem não foram detectadas. Foram considerados falsos positivos quaisquer elementos que não fossem placas de identificação das salas e placas que estavam apenas parcialmente visíveis. Os resultados deste teste podem ser observados na Tabela 1. As taxas de detecção correta e de falsos negativos foram calculadas com relação ao número de placas presentes nas imagens, e são descritas respectivamente pelas Equações (4.1) e (4.2). A taxa de falsos positivos é descrita pela Equação (4.3), e leva em consideração o número total de detecções realizadas. As definições utilizadas para falsos positivos, falsos negativos e suas taxas foram baseados nas definições apresentadas em [48].

$$\text{taxa de acerto} = \frac{\text{detecções corretas}}{\text{placas presentes}} \tag{4.1}$$

$$\text{taxa de falsos negativos} = \frac{\text{falsos negativos}}{\text{placas presentes}} \tag{4.2}$$

$$\text{taxa de falsos positivos} = \frac{\text{falsos positivos}}{\text{detecções totais}} \tag{4.3}$$

(a) (b) (c) (d) (e) (f) (g) (h)

Figura 23 – Exemplos de imgens que compõem o *dataset* usado nos testes.

Tabela 1 – Resultados das detecções.

Placas presentes	Detecções totais	Taxa de acerto	Taxa falso positivo	Taxa falso negativo
139	151	91,37%	15,89%	8,63%

Ao longo dos testes observou-se que a iluminação foi o principal fator que afetou a detecção de placas. A distância também afetou os resultados da detecção, porém apenas em casos mais extremos em que a câmera estava a mais de 7 metros das placas. O posicionamento da câmera em relação a placa não teve efeito perceptível na detecção, com exceção dos casos em que a posição da câmera implicava em reflexos visíveis na placa.

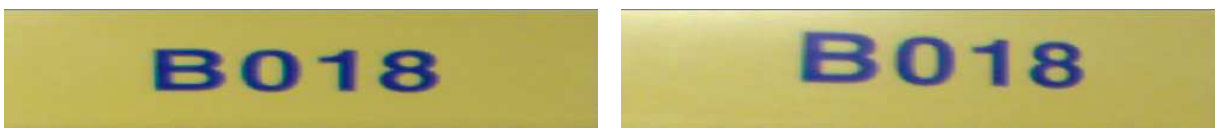
4.2 Teste do Processo de Correção de Perspectiva

O processo de correção de perspectiva foi analisado de forma qualitativa por meio da observação das imagens das placas após a correção. A qualidade da correção foi avaliada com conceito bom ou ruim. O conceito bom foi atribuído a imagens que apresentavam uma correção de perspectiva correta ou que apresentavam distorções sutís. O conceito ruim foi atribuído a qualquer imagem que apresentasse distorções facilmente perceptíveis a olho nu após a correção. A Figura 24 apresenta um exemplo de uma correção boa e uma ruim. A qualidade da correção de perspectiva de falsos positivos do processo de detecção não foi considerada. A baixa definição do texto de placas que se encontravam distantes da câmera não foi levada em consideração pois isto não é algo que a correção de perspectiva busca resolver. Os resultados deste teste são apresentados na Tabela 2. As taxas de correção boa e ruim foram calculadas com as equações (4.4) e (4.5) respectivamente.

$$\text{taxa de correções boas} = \frac{\text{num. de correções boas}}{\text{total de correções realizadas}} \tag{4.4}$$

$$\text{taxa de correções ruins} = \frac{\text{num. de correções ruins}}{\text{total de correções realizadas}} \tag{4.5}$$

Assim como no processo de detecção, a iluminação foi o fator de maior influência sobre a qualidade da correção de perspectiva. A distância também causou efeitos perceptíveis, degradando a qualidade da correção ao ultrapassar a marca de 6 metros. Enquanto



(a) (b)

Figura 24 – Em (a) temos um exemplo de uma correção boa e em (b) uma correção ruim.

Tabela 2 – Resultados das correções de perspectiva.

Conceito	Quantidade	Taxa
Bom	100	78,74%
Ruim	27	21,26%

isso nenhuma rotação afetou o desempenho da correção de perspectiva. Foram testadas rotações de cerca de 45º em torno do eixo ortogonal ao plano da placa, aproximadamente 60º em torno do eixo vertical paralelo ao plano da placa e aproximadamente 60º em torno do eixo horizontal paralelo ao plano da placa.

4.3 Teste do Processo de Identificação

Durante o teste do processo de identificação foram observadas as quantidades de leituras boas e leituras ruins realizadas pelo motor de OCR. Uma leitura é considerada boa quando pelo menos um de seus resultado apresenta no máximo 1 caractere incorreto em relação ao texto real da placa. Caso contrário a leitura é considerada ruim. Em seguida observou-se a quantidade de leituras aceitas, que apresentavam confiança maior ou igual a um limiar especificado, e leituras descartadas, que apresentavam confiança inferior a este limiar. Estes valores foram analisados em conjunto com a quantidade de placas corretamente detectadas e a qualidade da correção de perspectiva. As observações realizadas são apresentadas nas Tabela 3. A taxa de leituras boas e leituras ruins foram calculadas usando as equações (4.6) e (4.7) respectivamente. A taxa de leituras aceitas e a taxa de leituras descartadas foram calculadas com as equações (4.8) e (4.9) respectivamente. O valor usado para o limiar de confiança foi 80. Para selecionar o valor do limiar foram observados os resultados de 275 leituras realizadas pelo Tesseract cuja confiança dos resultados variava entre 18 e 93. Observou-se que 93,15% das leituras com confiança maior ou igual a 80 apresentaram no máximo 1 caractere incorreto. Assim, decidiu-se que este seria um bom limiar que rejeitaria a maioria das leitura inadequadas para localização sem restringir excessivamente a quantidade de leituras aceitas para localização.

$$\text{taxa de leituras boas} = \frac{\text{num. de leituras boas}}{\text{total de leituras realizadas}} \tag{4.6}$$

$$\text{taxa de leituras ruins} = \frac{\text{num. de leituras ruins}}{\text{total de leituras realizadas}} \tag{4.7}$$

$$\text{taxa de aceitação} = \frac{\text{num. de leituras aceitas}}{\text{total de leituras realizadas}} \tag{4.8}$$

$$\text{taxa de descarte} = \frac{\text{num. de leituras descartadas}}{\text{total de leituras realizadas}} \tag{4.9}$$

Tabela 3 – Resultados das identificações.

Leituras totais	Taxa leituras boas	Taxa leituras ruins	Taxa aceitação	Taxa descarte
151	51,66%	48,34%	37,09%	62,91%

O processo de identificação apresentou sensibilidade a fatores como iluminação, distância e qualidade da homografia. Os efeitos da iluminação são transmitidos a cada etapa do processo até a identificação. Uma detecção afetada pela iluminação implica em uma detecção incorreta dos cantos da placa e uma correção de perspectiva de qualidade ruim. Assim, além da iluminação reduzir a visibilidade dos caracteres da placa, os mesmos apresentam distorções, o que dificulta sua leitura. A distância também causou efeitos perceptíveis devido a baixa definição dos caracteres. Também pode-se observar durante os testes que esta parte do sistema causou o maior impacto no tempo de execução do sistema como um todo.

4.4 Teste do Processo de Localização

Para analisar o processo de localização em si, foi observada a quantidade leituras aceitas repassadas para este processo e quantas resultaram em localizações corretas ou incorretas. Cada localização correta significa que o sistema foi capaz de determinar corretamente o nó do mapa que representa a região em que a câmera se encontra a partir de uma leitura aceita de uma placa detectada na imagem. Uma localização incorreta caracteriza o evento do sistema relacionar uma leitura de uma placa detectada com o nó incorreto do mapa. Outro possível resultado é a não localização que ocorre quando a leitura não é suficientemente semelhante às placas registradas no mapa. Também foi observado a quantidade de imagens do *dataset* que geraram localizações corretas. Ao comparar este valor com o número de imagens que proporcionam leituras aceitas para a localização, podemos analisar o desempenho do processo de localização. Já quando comparamos a quantidade de imagens do *dataset* que geraram localizações corretas com o número total de imagens no *dataset* podemos ter uma noção do desempenho de início a fim do sistema como um todo.

Por questão de conveniência, o mapa topológico usado pelo sistema foi estruturado como uma lista de nós, já que a ligação entre os nós não é levada em consideração para a localização. Esta lista foi populada com nós contendo suas respectivas placas presentes nas imagens do *dataset*. Um nó adicional foi incluído no mapa. Nele foram introduzidas placas existentes no ambiente que não estavam presentes nas imagens. Estas placas foram selecionadas por sua semelhança com as demais para testar possíveis localizações incorretas.

Nas Tabelas 4 e 5 são apresentados os dados obtidos com este teste. A taxa de localização em imagens aceitas demonstra a parcela das imagens que geraram leituras

aceitas com as quais o sistema conseguiu realizar localizações corretas, demonstrando assim o desempenho do processo de localização. Ela é descrita pela Equação (4.10). A taxa de localização total demonstra a parcela de todas as imagens do *dataset* com as quais o sistema conseguiu realizar localizações corretas, demonstrando assim o desempenho do sistema como um todo. Ela é descrita pela Equação (4.11). A diferença observada entre o total de localizações e o número de leituras aceitas deve-se ao fato de algumas leituras aceitas apresentarem mais de um caractere errado. Assim, o sistema não conseguiu realizar localizações com estas leituras.

$$\text{taxa de loc. i.a.} = \frac{\text{num. de imagens que geraram localizações}}{\text{num. de imagens que geraram leituras aceitas}} \tag{4.10}$$

$$\text{taxa de loc. total} = \frac{\text{num. de imagens que geraram localizações}}{\text{num. total de imagens no } \textit{dataset}} \tag{4.11}$$

Tabela 4 – Resultados da localização.

Imagens totais no *dataset*	Imagens aceitas	Leituras aceitas
66	39	56

Tabela 5 – Resultados da localização (continuação).

Localizações corretas	Localizações incorretas	Taxa de loc. i.a.	Taxa de loc. total
52	0	92,30%	54,54%

5 Conclusões

Neste trabalho foi estudado o uso de reconhecimento ótico de caracteres para localização topológica de robôs. Esta estratégia de localização possui oportunidades de aplicação em tarefas de navegação semântica por comandos de voz ou texto, por empregar a mesma sintaxe na compreensão de comandos e na localização. Foi proposto, desenvolvido e testado um sistema que é capaz de detectar as placas de identificação de salas presentes no ambiente, reconhecer seu conteúdo textual e aplicá-lo para determinar sua posição referente a uma representação topológica do ambiente. A detecção das placas é realizada por um método desenvolvido que baseia-se nos contornos das placas. Um motor de OCR open-source foi empregado para realizar o reconhecimento de caracteres. A localização foi realizada pela análise da semelhança das leituras das placas com as informações textuais contidas no mapa.

Os testes realizados confirmaram a capacidade de localização do sistema. O desempenho demonstrado pelo processo de detecção condiz com as expectativas, com uma baixa taxa de falsos negativos e uma taxa considerável de falsos positivos. Os casos de falso negativos ocorreram devido a distância entre a câmera e a placa ou devido às condições de iluminação. Isto não representa algo negativo para o sistema considerando que sob estas condições seria difícil realizar um bom reconhecimento de caracteres. Já os falsos positivos não representam problema pois são descartados por uma etapa posterior no sistema.

A correção de perspectiva também apresentou bons resultados, embora haja muita margem para melhoria. As condições de iluminação tiveram um impacto negativo relevante no desempenho deste processo. Em algumas das imagens do dataset usado para o teste, reflexos nas placas causaram uma detecção incorreta do contorno das mesmas. Esta detecção errônea implicou em parâmetros incorretos para o cálculo da homografia e em uma correção de perspectiva incorreta. O método usado para detecção dos cantos da placa também é responsável por problemas na correção de perspectiva devido a sua imprecisão observada em algumas imagens. Apesar disso, este processo apresentou robustez as condições testadas de posicionamento da câmera relativo a placa.

O processo de identificação foi o gargalo de desempenho do sistema como um todo. Tanto em tempo de processamento como em precisão. Mesmo com o uso dos dados de treinamento para a língua portuguesa, o Tesseract apresentou dificuldade em reconhecer corretamente caracteres com acentos. Como podemos garantir que na primeira linha de texto das placas não há a presença de caracteres com acentos, apenas estas leituras foram utilizadas. Observou-se que uma confiança igual ou superior a 80 implicava em no máximo um caractere reconhecido incorretamente na maioria das leituras. Assim o valor de 80 foi selecionado para o limiar da classificação entre leituras boas e ruins. Apenas 51,66%

das detecções resultaram em reconhecimentos utilizáveis e somente 37,74% de todas as leituras realizadas atingiram o valor mínimo de confiança para serem aceitas. As condições de iluminação e a distância entre câmera e placa surtiram grande impacto negativo no desempenho do reconhecimento de caracteres. A qualidade da correção de perspectiva também influenciou o desempenho do motor de OCR.

Por fim a localização apresentou bons resultados. De todas as 57 leituras aceitas para o processo de localização, 52 delas resultaram e localizações corretas. Porém, como apenas uma parcela de todas as leituras foi aceita, o sistema só foi capaz de determinar sua localização em 54,54% das imagens do dataset. Além disso, as condições heurísticas usadas implicam na possibilidade de localizações incorretas. Este método também não é viável para mapeamento do ambiente, visto que as condições heurísticas dependem de conhecimento prévio dos marcadores que podem ser encontrados pelo robô.

Em geral o sistema cumpriu com os objetivos do trabalho. Porém existe muito que pode ser melhorado. Como trabalhos futuros podem ser estudados o desenvolvimento ou aplicação de métodos mais sofisticados e robustos para detecção das placas e seus cantos. Uma opção em particular que despertou o interesse do autor deste trabalho foram as *features* MSER. O uso de outros sistemas de reconhecimento de texto buscando melhorar o desempenho do sistema de localização também é um possível tema para trabalhos futuros. A estruturação completa do mapa do ambiente e a integração do sistema de localização com estratégias de navegação para aplicação em um robô real permitiria o teste do comportamento do sistema em uma aplicação real. Isto também representa um tema interessante para ser abordado por trabalhos futuros.

Referências Bibliográficas

1 BERGAMASCO, F. et al. Rune-tag: A high accuracy fiducial marker with strong occlusion resilience. In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. [S.l.: s.n.], 2011. p. 113–120. ISSN 1063-6919.

2 SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. *Introduction to Autonomous Mobile Robots*. 2nd. ed. [S.l.]: The MIT Press, 2011. ISBN 0262015358, 9780262015356.

3 WALLGRN, J. O. *Hierarchical Voronoi Graphs: Spatial Representation and Reasoning for Mobile Robots*. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2009. ISBN 3642103022, 9783642103025.

4 Wikimedia Commons. *File:HSV color solid cylinder.png*. 2015. Disponível em: <https://commons.wikimedia.org/wiki/File:HSV_color_solid_cylinder.png>.

5 BONIN-FONT, F.; ORTIZ, A.; OLIVER, G. Visual navigation for mobile robots: A survey. *J. Intell. Robotics Syst.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 53, n. 3, p. 263–296, nov. 2008. ISSN 0921-0296. Disponível em: <http://dx.doi.org/10.1007/s10846-008-9235-4>.

6 DESOUZA, G. N.; KAK, A. C. Vision for mobile robot navigation: A survey. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 24, n. 2, p. 237–267, 2002.

7 WERNER, S.; KRIEG-BRÜCKNER, B.; HERRMANN, T. Modelling navigational knowledge by route graphs. In: *Spatial cognition II*. [S.l.]: Springer, 2000. p. 295–316.

8 LEONARD, J. J.; DURRANT-WHYTE, H. F. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on robotics and Automation*, IEEE, v. 7, n. 3, p. 376–382, 1991.

9 OHNO, T.; OHYA, A.; YUTA, S. Autonomous navigation for mobile robots referring pre-recorded image sequence. In: IEEE. *Intelligent Robots and Systems' 96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*. [S.l.], 1996. v. 2, p. 672–679.

10 JONES, S. D.; ANDRESEN, C.; CROWLEY, J. L. Appearance based process for visual navigation. In: IEEE. *Intelligent Robots and Systems, 1997. IROS'97., Proceedings of the 1997 IEEE/RSJ International Conference on*. [S.l.], 1997. v. 2, p. 551–557.

11 CHATILA, R.; LAUMOND, J.-P. Position referencing and consistent world modeling for mobile robots. In: IEEE. *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*. [S.l.], 1985. v. 2, p. 138–145.

12 SUGIHARA, K. Some location problems for robot navigation using a single camera. *Computer vision, graphics, and image processing*, Elsevier, v. 42, n. 1, p. 112–129, 1988.

13 PAN, J. et al. Fuzzy-nav: a vision-based robot navigation architecture using fuzzy inference for uncertainty-reasoning. In: *Procs. of the World Congress on Neural Networks*. [S.l.: s.n.], 1995. p. 602–607.

14 BORENSTEIN, J.; KOREN, Y. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on systems, Man, and Cybernetics*, IEEE, v. 19, n. 5, p. 1179–1187, 1989.

15 TURK, M. A. et al. Vits-a vision system for autonomous land vehicle navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 10, n. 3, p. 342–361, 1988.

16 MATTHIES, L. et al. Mars microrover navigation: Performance evaluation and enhancement. *Autonomous robots*, Springer, v. 2, n. 4, p. 291–311, 1995.

17 DAVISON, A. J.; CID, Y. G.; KITA, N. Real-time 3d slam with wide-angle vision. *IFAC Proceedings Volumes*, Elsevier, v. 37, n. 8, p. 868–873, 2004.

18 GARTSHORE, R.; AGUADO, A.; GALAMBOS, C. Incremental map building using an occupancy grid for an autonomous monocular robot. In: IEEE. *Control, Automation, Robotics and Vision, 2002. ICARCV 2002. 7th International Conference on*. [S.l.], 2002. v. 2, p. 613–618.

19 GASPAR, J.; WINTERS, N.; SANTOS-VICTOR, J. Vision-based navigation and environmental representations with an omnidirectional camera. *IEEE Transactions on robotics and automation*, IEEE, v. 16, n. 6, p. 890–898, 2000.

20 KOSECKA, J. et al. Qualitative image based localization in indoors environments. In: IEEE. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. [S.l.], 2003. v. 2, p. II–II.

21 SARIPALLI, S. et al. Detection and tracking of external features in an urban environment using an autonomous helicopter. In: IEEE. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. [S.l.], 2005. p. 3972–3977.

22 MAJA, J. M. et al. Real-time obstacle avoidance algorithm for visual navigation. In: IEEE. *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*. [S.l.], 2000. v. 2, p. 925–930.

23 GARCIA-FIDALGO, E.; ORTIZ, A. Vision-based topological mapping and localization methods: A survey. *Robotics and Autonomous Systems*, Elsevier, v. 64, p. 1–20, 2015.

24 WERNER, F.; MAIRE, F.; SITTE, J. Topological slam using fast vision techniques. In: SPRINGER. *FIRA RoboWorld Congress*. [S.l.], 2009. p. 187–196.

25 WEISS, C. et al. Fast outdoor robot localization using integral invariants. In: *Proceedings of the 5th International Conference on Computer Vision Systems (ICVS). Bielefeld, Germany*. [S.l.: s.n.], 2007.

26 SINGH, G.; KOSECKA, J. Visual loop closing using gist descriptors in manhattan world. In: *ICRA Omnidirectional Vision Workshop*. [S.l.: s.n.], 2010.

27 ARROYO, R. et al. Bidirectional loop closure detection on panoramas for visual navigation. In: IEEE. *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*. [S.l.], 2014. p. 1378–1383.

28 RAMISA, A. et al. Robust vision-based robot localization using combinations of local feature region detectors. *Autonomous Robots*, Springer, v. 27, n. 4, p. 373, 2009.

29 BACCA, B.; SALVI, J.; CUFÍ, X. Long-term mapping and localization using feature stability histograms. *Robotics and Autonomous Systems*, Elsevier, v. 61, n. 12, p. 1539–1558, 2013.

30 ROMERO, A.; CAZORLA, M. Topological slam using omnidirectional images: Merging feature detectors and graph-matching. In: SPRINGER. *International Conference on Advanced Concepts for Intelligent Vision Systems*. [S.l.], 2010. p. 464–475.

31 MADDERN, W.; MILFORD, M.; WYETH, G. Towards persistent indoor appearance-based localization, mapping and navigation using cat-graph. In: IEEE. *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. [S.l.], 2012. p. 4224–4230.

32 PAUL, R.; NEWMAN, P. Fab-map 3d: Topological mapping with spatial and visual appearance. In: IEEE. *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. [S.l.], 2010. p. 2649–2656.

33 MUR-ARTAL, R.; TARDÓS, J. D. Fast relocalisation and loop closing in keyframe-based slam. In: IEEE. *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. [S.l.], 2014. p. 846–853.

34 WANG, J.; YAGI, Y. Efficient topological localization using global and local feature matching. *International Journal of Advanced Robotic Systems*, SAGE Publications Sage UK: London, England, v. 10, n. 3, p. 153, 2013.

35 LIN, H.-Y.; LIN, Y.-H.; YAO, J.-W. Scene change detection and topological map construction using omnidirectional image sequences. In: *MVA*. [S.l.: s.n.], 2013. p. 57–60.

36 BEN-AFIA, A. et al. Review and classification of vision-based localisation techniques in unknown environments. *IET Radar, Sonar & Navigation*, IET, v. 8, n. 9, p. 1059–1072, 2014.

37 GOLDBERG, S. B.; MAIMONE, M. W.; MATTHIES, L. Stereo vision and rover navigation software for planetary exploration. In: IEEE. *Aerospace Conference Proceedings, 2002. IEEE*. [S.l.], 2002. v. 5, p. 5–5.

38 SE, S.; LOWE, D.; LITTLE, J. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *The international Journal of robotics Research*, SAGE Publications Sage UK: London, England, v. 21, n. 8, p. 735–758, 2002.

39 LEE, D.; RYAN, T.; KIM, H. J. Autonomous landing of a vtol uav on a moving platform using image-based visual servoing. In: IEEE. *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. [S.l.], 2012. p. 971–976.

40 SANCHEZ-LOPEZ, J. L. et al. A multi-layered component-based approach for the development of aerial robotic systems: the aerostack framework. *Journal of Intelligent & Robotic Systems*, Springer, v. 88, n. 2-4, p. 683–709, 2017.

41 KIM, A.; EUSTICE, R. M. Real-time visual slam for autonomous underwater hull inspection using visual saliency. *IEEE Transactions on Robotics*, IEEE, v. 29, n. 3, p. 719–733, 2013.

42 JASIOBEDZKI, P. et al. Underwater 3d mapping and pose estimation for rov operations. In: IEEE. *OCEANS 2008*. [S.l.], 2008. p. 1–6.

43 NAIMARK, L.; FOXLIN, E. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In: *Proceedings of the 1st International Symposium on Mixed and Augmented Reality*. Washington, DC, USA: IEEE Computer Society, 2002. (ISMAR '02), p. 27–. ISBN 0-7695-1781-1. Disponível em: <http://dl.acm.org/citation.cfm?id=850976.854961>.

44 GATRELL, L. B.; HOFF, W. A.; SKLAIR, C. W. Robust image features: Concentric contrasting circles and their image extraction. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. *Cooperative Intelligent Robotics in Space II*. [S.l.], 1992. v. 1612, p. 235–245.

45 CHO, Y.; LEE, J.; NEUMANN, U. A multi-ring color fiducial system and an intensity-invariant detection method for scalable fiducial-tracking augmented reality. In: CITESEER. *In IWAR*. [S.l.], 1998.

46 KATO, I. P. H.; BILLINGHURST, M.; POUPYREV, I. Artoolkit user manual, version 2.33. *Human Interface Technology Lab, University of Washington*, v. 2, 2000.

47 KATO, H.; BILLINGHURST, M. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In: IEEE. *Augmented Reality, 1999.(IWAR'99) Proceedings. 2nd IEEE and ACM International Workshop on*. [S.l.], 1999. p. 85–94.

48 FIALA, M. Artag, a fiducial marker system using digital techniques. In: IEEE. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. [S.l.], 2005. v. 2, p. 590–596.

49 OLSON, E. AprilTag: A robust and flexible visual fiducial system. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.]: IEEE, 2011. p. 3400–3407.

50 TRACHTENBERT, A. *Computational methods in coding theory*. Dissertação (Mestrado) — University of Illinois at Urbana-Champaign, 1996.

51 WANG, J.; OLSON, E. AprilTag 2: Efficient and robust fiducial detection. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.: s.n.], 2016.

52 SANCHEZ-LOPEZ, J. L. et al. A reliable open-source system architecture for the fast designing and prototyping of autonomous multi-uav systems: Simulation and experimentation. *Journal of Intelligent & Robotic Systems*, Springer, v. 84, n. 1-4, p. 779–797, 2016.

53 BOŠNAK, M.; MATKO, D.; BLAŽIČ, S. Quadrocopter hovering using position-estimation information from inertial sensors and a high-delay video system. *Journal of Intelligent & Robotic Systems*, Springer, v. 67, n. 1, p. 43–60, 2012.

54 CHAVES, S. M.; WOLCOTT, R. W.; EUSTICE, R. M. Neec research: Toward gps-denied landing of unmanned aerial vehicles on ships at sea. *Naval Engineers Journal*, American Society of Naval Engineers, v. 127, n. 1, p. 23–35, 2015.

55 FENG, C. et al. Vision guided autonomous robotic assembly and as-built scanning on unstructured construction sites. *Automation in Construction*, Elsevier, v. 59, p. 128–138, 2015.

56 HAMNER, B. et al. An autonomous mobile manipulator for assembly tasks. *Autonomous Robots*, Springer, v. 28, n. 1, p. 131, 2010.

57 STEELE, J. P. et al. Development of an oil and gas refinery inspection robot. In: AMERICAN SOCIETY OF MECHANICAL ENGINEERS. *ASME 2014 International Mechanical Engineering Congress and Exposition*. [S.l.], 2014. p. V04AT04A016–V04AT04A016.

58 LEE, K. S. et al. Autonomous patrol and surveillance system using unmanned aerial vehicles. In: IEEE. *Environment and Electrical Engineering (EEEIC), 2015 IEEE 15th International Conference on*. [S.l.], 2015. p. 1291–1297.

59 MONAJJEMI, V. M. et al. Hri in the sky: Creating and commanding teams of uavs with a vision-mediated gestural interface. In: IEEE. *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. [S.l.], 2013. p. 617–623.

60 LING, K. et al. Autonomous maritime landings for low-cost vtol aerial vehicles. In: IEEE. *Computer and Robot Vision (CRV), 2014 Canadian Conference on*. [S.l.], 2014. p. 32–39.

61 MUÑOZ-SALINAS, R. et al. Mapping and localization from planar markers. *Pattern Recognition*, Elsevier, v. 73, p. 158–171, 2018.

62 SWEATT, M. et al. Wifi based communication and localization of an autonomous mobile robot for refinery inspection. In: IEEE. *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. [S.l.], 2015. p. 4490–4495.

63 OLSON, E. et al. Progress toward multi-robot reconnaissance and the magic 2010 competition. *Journal of Field Robotics*, Wiley Online Library, v. 29, n. 5, p. 762–792, 2012.

64 LATOMBE, J.-C. *Robot Motion Planning*. Norwell, MA, USA: Kluwer Academic Publishers, 1991. ISBN 079239206X.

65 FRANZ, M. O. et al. Learning view graphs for robot navigation. *Autonomous robots*, Springer, v. 5, n. 1, p. 111–125, 1998.

66 NELSON, R. C. Visual homing using an associative memory. *Biological Cybernetics*, Springer, v. 65, n. 4, p. 281–291, 1991.

67 MORALES, Y. et al. Building a model of the environment from a route perspective for human–robot interaction. *International Journal of Social Robotics*, v. 7, n. 2, p. 165–181, Apr 2015. ISSN 1875-4805. Disponível em: <https://doi.org/10.1007/s12369-014-0265-8>.

68 DUDEK, G. et al. Robotic exploration as graph construction. *IEEE transactions on robotics and automation*, IEEE, v. 7, n. 6, p. 859–865, 1991.

69 REKLEITIS, I. M.; DUJMOVIC, V.; DUDEK, G. Efficient topological exploration. In: *ICRA*. [S.l.: s.n.], 1999. p. 676–681.

70 YE, Q.; DOERMANN, D. Text detection and recognition in imagery: A survey. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 37, n. 7, p. 1480–1500, 2015.

71 KOO, H. I.; KIM, D. H. Scene text detection via connected component clustering and nontext filtering. *IEEE transactions on image processing*, IEEE, v. 22, n. 6, p. 2296–2305, 2013.

72 WANG, T. et al. End-to-end text recognition with convolutional neural networks. In: IEEE. *Pattern Recognition (ICPR), 2012 21st International Conference on*. [S.l.], 2012. p. 3304–3308.

73 YIN, X.-C. et al. Robust text detection in natural scene images. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 36, n. 5, p. 970–983, 2014.

74 LONG, S. et al. Textsnake: A flexible representation for detecting text of arbitrary shapes. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2018. p. 19–35.

75 SHI, C. et al. Scene text recognition using part-based tree-structured character detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2013. p. 2961–2968.

76 WEINMAN, J. J.; LEARNED-MILLER, E.; HANSON, A. R. Scene text recognition using similarity and a lexicon with sparse belief propagation. *IEEE Transactions on pattern analysis and machine intelligence*, IEEE, v. 31, n. 10, p. 1733–1746, 2009.

77 FEILD, J. Improving text recognition in images of natural scenes. 2014.

78 SMITH, R. An overview of the tesseract ocr engine. In: *Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 02*. Washington, DC, USA: IEEE Computer Society, 2007. (ICDAR '07), p. 629–633. ISBN 0-7695-2822-8. Disponível em: <http://dl.acm.org/citation.cfm?id=1304596.1304846>.

79 TOMONO, M.; YUTA, S. Mobile robot navigation in indoor environments using object and character recognition. In: IEEE. *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*. [S.l.], 2000. v. 1, p. 313–320.

80 MATA, M. et al. A visual landmark recognition system for topological navigation of mobile robots. In: *ICRA*. [S.l.: s.n.], 2001. p. 1124–1129.

81 MATA, M. et al. Learning visual landmarks for mobile robot navigation. *IFAC Proceedings Volumes*, Citeseer, v. 35, n. 1, p. 445–450, 2002.

82 MATA, M. et al. Using learned visual landmarks for intelligent topological navigation of mobile robots. In: IEEE; 1999. *IEEE International Conference on Robotics and Automation*. [S.l.], 2003. v. 1, p. 1324–1329.

83 CASE, C. et al. Autonomous sign reading for semantic mapping. In: IEEE. *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. [S.l.], 2011. p. 3297–3303.

84 SCHULZ, R. et al. Robot navigation using human cues: A robot navigation system for symbolic goal-directed exploration. In: IEEE. *Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA 2015)*. [S.l.], 2015. p. 1100–1105.

85 BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

86 VISVALINGAM, M.; WHYATT, J. D. Line generalisation by repeated elimination of points. *The cartographic journal*, Taylor & Francis, v. 30, n. 1, p. 46–51, 1993.

87 CORKE, P. *Robotics, Vision and Control: Fundamental Algorithms In MATLAB® Second, Completely Revised*. [S.l.]: Springer, 2017. v. 118.

88 LEVENSHTEIN, V. I. Binary codes capable of correcting deletions, insertions, and reversals. In: *Soviet physics doklady*. [S.l.: s.n.], 1966. v. 10, n. 8, p. 707–710.

89 BASLER. *Product Documentation: acA2040-120uc*. 2018. Disponível em: <https://docs.baslerweb.com/index.htm#t=en\%2Faca2040-120uc.htm>.

90 TAMRON. *CCTV Lenses/Model : M118FM06*. 2018. Disponível em: <https://www.tamron.biz/en/data/ipcctv/cctv_mg/m118fm06.html>.