UNIVERSIDADE FEDERAL DE SANTA CATARINA
ENGENHARIA DE AUTOMAÇÃO E SISTEMAS

Fernando de Lucca Siqueira

INCORPORANDO ELEMENTOS DE CONTEXTO NA NAVEGAÇÃO INTELIGENTE DE ROBÔS MÓVEIS USANDO TRAJETÓRIAS SEMÂNTICAS, ÁRVORE DE COMPORTAMENTO E MAPA SEMÂNTICO

Florianópolis

2017

Fernando de Lucca Siqueira

INCORPORANDO ELEMENTOS DE CONTEXTO NA NAVEGAÇÃO INTELIGENTE DE ROBÔS MÓVEIS USANDO TRAJETÓRIAS SEMÂNTICAS, ÁRVORE DE COMPORTAMENTO E MAPA SEMÂNTICO

Tese submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas para a obtenção do Grau de Doutor em Engenharia de Automação e Sistemas.
Orientador: Prof. Dr. Edson Roberto De Pieri
Coorientadora: Prof. Dra. Vania Bogorny

Florianópolis

2017

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Siqueira, Fernando de Lucca
INCORPORANDO ELEMENTOS DE CONTEXTO NA NAVEGAÇÃO INTELIGENTE DE ROBÔS MÓVEIS USANDO TRAJETÓRIAS SEMÂNTICAS, ÁRVORE DE COMPORTAMENTO E MAPA SEMÂNTICO / Fernando de Lucca Siqueira ; orientador, Edson Roberto De Pieri, coorientadora, Vania Bogorny, 2017.
166 p.

Tese (doutorado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós Graduação em Engenharia de Automação e Sistemas, Florianópolis, 2017.

Inclui referências.

1. Engenharia de Automação e Sistemas. 2. Robótica Móvel. 3. Navegação Inteligente. 4. Trajetória Semântica. 5. Sistemas Sensíveis ao Contexto. I. De Pieri, Edson Roberto. II. Bogorny, Vania. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. IV. Título.

Fernando de Lucca Siqueira

INCORPORANDO ELEMENTOS DE CONTEXTO NA NAVEGAÇÃO INTELIGENTE DE ROBÔS MÓVEIS USANDO TRAJETÓRIAS SEMÂNTICAS, ÁRVORE DE COMPORTAMENTO E MAPA SEMÂNTICO

Esta Tese foi julgada aprovada para a obtenção do Título de "Doutor em Engenharia de Automação e Sistemas", e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Florianópolis, 03 de Abril 2017.

Prof. Dr. Daniel Ferreira Coutinho
Coordenador do Curso

Prof. Dr. Edson Roberto De Pieri
Orientador

Prof. Dra. Vania Bogorny
Coorientadora

Banca Examinadora:

Prof. Dr. João Maurício Rosário
Presidente

Prof. Dr. André Bittencourt Leal

Prof. Dr. Marcelo Ricardo Stemmer

Prof. Dr. Renato Fileto

Prof. Dr. Ricardo Alexandre Reinaldo de Moraes

Dedico esta tese à(ao) minha(meu) filha(filho).
Que a ciência ilumine suas dúvidas sobre
as questões da vida, do universo e tudo
mais.

AGRADECIMENTOS

Agradeço inicialmente meu orientador Edson Roberto De Pieri que me aceitou como seu aluno. Sua orientação transformou e deu rumo para este trabalho. Muito obrigado por acreditar no meu potencial. Também agradeço a minha co-orientadora Vania Bogorny que me ensinou o que é pesquisa científica e tornou esse doutorado possível ao me indicar para o Edson. Um abraço especial para Patrícia Della Mea Plentz, colega de pesquisa que me deu oportunidades para crescer ao me convidar para trabalharmos juntos. O apoio destes professores foi minha base científica onde construí minha pesquisa.

Um agradecimento especial à minha família. Meus pais, Amaro e Marilene, que sempre me ajudaram quando eu precisava, concordando ou não com o caminho que decidi trilhar. Devo toda a educação que tive a eles e serei sempre grato. Agradeço também a minha irmã Marina, ao meu cunhado Alessandro e minha linda sobrinha Alice, que juntos de meus pais fazem de Florianópolis o lar da minha família.

Existe a família de sangue e a família que a gente escolhe. E minha família de amigos, que iniciou em 2005, sempre me acolheram em meio de risadas, conversas e alegria. Karina (Kah), Fernando (Tom), Daniel (Samurai), Tiago e Bárbara (Bah) foram indispensáveis para que o período deste doutorado fosse muito mais do que a pesquisa. E boas vindas ao Victor, o novo integrante.

Gostaria de agradecer a Universidade Federal de Santa Catarina, especialmente ao Departamento de Engenharia de Automação e Sistemas por ter sido meu lar de trabalho e estudo durante estes anos e à Capes e ao CNPQ por apoiar minha pesquisa durante meu doutorado.

Por fim, e mais importante, todo agradecimento e amor do mundo para minha melhor amiga, amante e companheira Lívia. Nada disso seria possível sem seu apoio todos os dias. Você nunca duvidou de mim e ajudou de todas as maneiras possíveis. Sou o melhor de mim com você. Eu te amo.

The best solution to a problem is usually the easiest one...

GLaDOS

RESUMO

O desenvolvimento e popularização de novas tecnologias resultaram em uma produção em massa de informações por diversos dispositivos. Aplicar estas informações em sistemas computadorizados permite dar sentido e uso para estes dados, construindo um arcabouço de informações úteis para o funcionamento de diversos dispositivos. Uma área que pode se beneficiar de tais informações é a robótica móvel. Um robô móvel deve navegar em um ambiente a partir de informações que permitem ao robô entender e visualizar o ambiente ao seu redor. Para um robô se comportar de maneira similar a um ser humano, é preciso que ele saiba compreender e estruturar tais informações. Esta tese estuda a aplicação de informações de contexto em aplicação de robôs móveis, resultando em uma navegação inteligente baseada em contexto. A aplicação destas informações dá-se através do uso de trajetórias semânticas, árvore de comportamento e mapa semântico. Todas as informações são armazenadas para o refinamento do comportamento do robô através de análise do histórico de execução. Além disto, a tese explora a problemática da recarga autônoma de robôs móveis utilizando lógica fuzzy e informações de contexto. Experimentos e resultados de simulação considerando diferentes tarefas típicas de robótica ilustram a aplicabilidade dos métodos propostos.

Palavras-chave: Robótica Móvel. Navegação Inteligente. Trajetória Semântica. Árvore de Comportamento. Mapa Semântico. Sistemas Sensíveis ao Contexto

ABSTRACT

New and popular techologies produce a lot of information through several devices. Applying these information in computerized systems give meaning and use to this data, building a framework of useful information for various devices. One area that can benefit from such information is mobile robotics. A mobile robot must navigate through an environment using information that allows the robot to understand the environment around it. For a robot to behave similar to a human being, it must be able to understand and organize such information. This thesis studies the application of context information in the application of mobile robots, resulting in intelligent navigation based on context. The intelligent navigation is achieved through the use of semantic trajectories, behavior tree and semantic map. All information is stored for refinement of the behavior of the robot through analysis of the execution history. In addition, the thesis explores the problem of autonomous recharging of mobile robots using fuzzy logic and context information. Experiments and simulation with different robotic tasks illustrate the applicability of the proposed methods.

Keywords: Mobile Robot. Intelligent Navigation. Semantic Trajectory. Behavior Tree. Semantic Map. Context-Aware Systems

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AFD	Autômato Finito Determinístico
ARP	Autonomous Recharging Problem
ASCENS	Autonomic Services-Component ENSembles
BART	Behavior Architecture for Robotic Tasks
BT	Behavior Tree
CB-SMoT	Clustering-Based Stops and Moves of Trajectories
ConBINaMoR	Context-Based Inteligent Navigation for Mobile Robots
CONSTAnT	CONceptual model of Semantic TrAjecTories
CONSTAnT-MR	CONceptual model of Semantic TrAjecTories for Mobile Robots
CSD	Current Source Distance
CTD	Current Task Distance
DB-SMoT	Direction-Based Stops and Moves of Trajectories
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
GPS	Global Positioning System
IB-SMoT	Intersection Based Stops and Moves of Trajectories
ISD	Initial Source Distance
ITD	Initial Task Distance
JSON	JavaScript Object Notation
OWL	Web Ontology Language
ROS	Robot Operating System
RUBICON	Robotics UBIquitous COgnitive Network
SEEK	SEmantic Enrichment of trajectory Knowledge discovery
SGBD	Sistema Gerenciador de Banco de Dados
SLAM	Simultaneously Localization and Mapping
SLEEP	Staying-aLive and Energy-Efficient Path planning
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
WWW	What, Where and When

SUMÁRIO

1 INTRODUÇÃO

Seres humanos usam seus sentidos para obter informações de contexto sobre o ambiente e planejar um caminho até certo destino (HUANG; LIU, 2004). Este processo de um agente utilizar informações de contexto para adaptar seu movimento ao cenário encontrado é chamado de navegação inteligente. O próprio comportamento humano também depende do ambiente e da situação atual em que está inserido, podendo o próprio comportamento ser modificado quando o ambiente também muda. A capacidade de um ser humano modificar seu comportamento de acordo com o contexto atual também pode ser transportado para um sistema computadorizado. Os sistemas computadorizados que modificam seu comportamento de acordo com o contexto são chamados de sistemas sensíveis ao contexto (DEY, 2001). Segundo Dey (2001), contexto é qualquer informação que pode ser utilizada para caracterizar a situação de uma entidade, sendo esta entidade uma pessoa, um lugar ou um objeto relevante para a aplicação. Sistemas computadorizados podem fazer uso destes dispositivos e informações para gerar e utilizar informações de contexto, inclusive na robótica móvel.

A robótica já se estabeleceu no setor industrial como alternativa e complemento ao trabalho humano. Segundo Craig (2005), desde a década de 90 a adoção de robôs para a automatização de processos industriais tem tido custo cada vez menor. Robôs são não somente mais baratos, mas mais rápidos e exatos também, conseguindo realizar tarefas que exigem cada vez mais precisão. A evolução da tecnologia contribui para que robôs possam executar tarefas cada vez mais complexas e que seriam impossíveis para trabalhadores humanos.

Secchi (2008) define robôs móveis como dispositivos de transporte automático, em outras palavras, robôs móveis são plataformas mecânicas que se locomovem através de um sistema de navegação em um ambiente conhecido ou desconhecido, onde a locomoção possui um certo nível de autonomia. Robôs móveis podem realizar tarefas em ambientes hostis ou de difícil acesso aos seres humanos. Por exemplo, o robô Sojourner, ilustrado na Figura 1, foi usado na missão Pathfinder na NASA coletando e analisando amostras de rochas em Marte. Apesar de ser operado da Terra, o robô possuía sensores para detectar e desviar de obstáculos. Robôs móveis também podem realizar outras tarefas, tais como: transporte de cargas perigosas, manutenção de dispositivos nucleares, exploração subterrânea, etc.

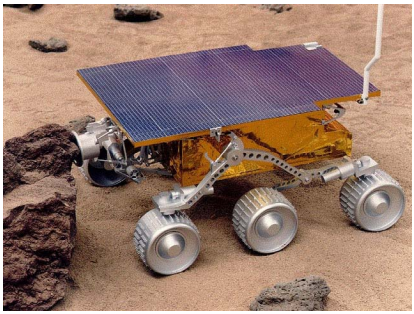


Figura 1 – Robô Sojourner usado na missão Pathfinder em Marte

Para um robô móvel possuir uma navegação inteligente, ele deve, assim como um ser humano, perceber o ambiente e utilizar informações de contexto para adaptar seu comportamento com o cenário atual, tornando-o mais independente possível de intervenção humana (BACCIU et al., 2014; FOUKARAKIS et al., 2014; GALINDO et al., 2008). A autonomia de um robô móvel pode ser melhorada a partir de informações sobre o ambiente, entidades, usuários, tarefas e da própria aplicação que ele está inserido. Além destas informações de contexto, é necessário uma forma estruturada de apresentá-las e utilizá-las. Isso é possível através de tecnologias como trajetória semântica, árvore de comportamento e mapa semântico.

Uma trajetória semântica é uma trajetória enriquecida com informações de contexto (PARENT et al., 2013). Uma trajetória, descrita como uma sequência de pontos no tempo e no espaço, captura o movimento de um objeto, sendo possível descobrir padrões de movimentos em grandes conjuntos de trajetórias. Para uma trajetória semântica, existem iniciativas para representar, além dos pontos, as informações de contexto associadas com a trajetória, como o modelo CONSTAnT (BOGORNY et al., 2014). Este modelo permite estruturar as informações de espaço e tempo dos pontos juntamente com as informações de contexto que enriquecem a trajetória. O campo de pesquisa de análise de trajetórias de objetos móveis tem como objetivo identificar padrões e comportamentos a partir da análise de movimento de objetos. Trabalhos da literatura já identificaram padrões de comportamento como desvios (ALVARES et al., 2011), grupos de objetos (WACHOWICZ et al., 2011), perseguição (SIQUEIRA; BOGORNY, 2011), entre outros, mas são usualmente focados em humanos, carros ou animais.

O conceito de trajetória na robótica móvel está usualmente associado ao planejamento de movimento do robô, foco de diversos tra-

balhos da área (KALAKRISHNAN et al., 2011; WEI et al., 2012; SOUISSI et al., 2013) e tratam basicamente da posição, velocidade e aceleração do robô móvel para cada instante de tempo considerado. Entretanto, não foram encontrados trabalhos que analisam a trajetória executada pelo robô, resultado da navegação autônoma do dispositivo, levando-se em conta as restrições dinâmicas e estáticas e também de informações suplementares obtidas do ambiente. As tarefas de um robô móvel são usualmente baseadas na locomoção, sendo possível capturar sua trajetória, uma vez que seu deslocamento pode ser rastreado. Assim como os seres humanos exibem um comportamento em seu movimento, a trajetória de um robô móvel também pode ser analisada e correlacionada com os eventos que ocorreram no meio.

Um robô móvel deve ser autônomo o suficiente para tomar decisões com a mínima ajuda possível de um operador, definindo o comportamento necessário para adaptar-se em cenários diferentes. A tomada de decisão na robótica móvel é usualmente modelada através de máquinas de estados finitos (KURT; ÖZGüNER, 2013; MARINO et al., 2009; FOUKARAKIS et al., 2014) ou modelos de Markov (SEYHAN; ALPASLAN; YAVAŞ, 2013). Estudos recentes apostam em uma nova abordagem para o controle de comportamento e decisão em robôs móveis: a árvore de comportamento (COLLEDANCHISE; OGREN, 2014; MARZINOTTO et al., 2014). Uma árvore de comportamento permite descrever o processo de tomada de decisão de um robô através de condições e ações de maneira robusta e modular (COLLEDANCHISE; OGREN, 2014). A estrutura em árvore permite adaptar o comportamento do robô durante a execução da tarefa de acordo com o contexto atual, definindo as ações apropriadas para cada cenário e para as mudanças que podem ocorrer.

Apesar de existir estratégias para a navegação de um robô móvel em ambientes desconhecidos, o planejamento e execução de trajetórias a partir de um mapa do ambiente permitem uma navegação mais otimizada e estruturada do robô. Um grande problema com trabalhos clássicos da robótica é o uso de um mapa estático ou somente com representação métrica do ambiente, ignorando outras informações relevantes. Segundo Kostavelis e Gasteratos (2015), para um robô compreender o ambiente de maneira similar a um humano, por exemplo, diferenciando um corredor de um quarto ou banheiro, é preciso de um mapa que represente mais informações além da distribuição espacial dos elementos do ambiente. Para isso surgiu o conceito de mapa semântico. Um mapa semântico é uma representação aprimorada do ambiente, unindo informações geométricas com características qualitativas de alto nível (KOSTAVELIS; GASTERATOS, 2015). Através de um mapa semântico

é possível descrever o significado de um local assim como dos objetos pertencentes ao ambiente. Apesar de diversos trabalhos na literatura proporem modelos para criação de mapas semânticos (GALINDO et al., 2008; LIM et al., 2013b; PRONOBIS, 2011), pouca atenção tem sido dada para o uso do mapa semântico na navegação e tomada de decisão de robôs, além de não terem sido encontrados trabalhos na robótica móvel em que o mapa semântico também represente o estado atual do ambiente, que pode variar com o tempo e a execução da aplicação.

Outro fator pouco explorado na literatura em robótica móvel é quanto à capacidade do robô de se manter ativo, evitando ficar sem energia durante a execução de uma tarefa. O gerenciamento de energia em um robô móvel é essencial para seu funcionamento e independência, evitando a necessidade de interferência humana para resgatar um robô sem energia. A propriedade de um robô autogerenciar sua bateria é chamada de Autonomous Recharging Problem (ARP) ou Problema da Recarga Autônoma (KANNAN et al., 2013). Estratégias da literatura se baseiam em como um robô deve recarregar (OH; ZELINSKY; TAYLOR, 2000) (SILVERMAN et al., 2002) ou quando ele deve recarregar (WEI et al., 2012) (KANNAN et al., 2013). Entretanto, essas estratégias são normalmente muito restritivas e não levam em consideração informações de contexto importantes para a caracterização do estado atual do robô como, por exemplo, sua localização atual em relação à tarefa e à fonte de energia. Uma forma de integrar o uso destas informações com a informação sobre o estado atual de energia do robô para gerenciar a recarga de bateria é através de lógica fuzzy. A lógica fuzzy permite tomar decisões a partir de um conjunto de entradas e de regras lógicas de inferência. Entretanto, não foram identificados trabalhos da literatura que utilizam uma estratégia fuzzy para o problema de recarga autônoma.

Uma navegação inteligente é importante para o agente planejar e executar tarefas, adaptando seu comportamento de acordo com as mudanças no ambiente. Para um robô móvel ter uma navegação inteligente, é preciso ter suporte de ferramentas e métodos para ter um comportamento racional como um humano. Esta tese propõe a incorporação de informações de contexto para uma navegação inteligente de robôs móveis. As informações de contexto são utilizadas para descrever o ambiente através de um mapa semântico, definir o comportamento apropriado através de uma árvore de comportamento e enriquecer a trajetória do robô através da trajetória semântica, armazenando dados de execução das tarefas. Esses dados são utilizados para compreender o efeito destas informações de contexto na execução das tarefas do

robô e refinar seu comportamento de maneira incremental, otimizando a execução das tarefas.

1.1 OBJETIVO

O objetivo deste trabalho é demonstrar que informações de contexto associadas a estratégias de árvore de comportamento, mapa semântico, lógica fuzzy e trajetória semântica são alternativas eficientes para navegação inteligente e adaptativa de um robô móvel. A solução proposta aborda um modelo de incorporação de contexto em aplicações de robótica móvel utilizando árvore de comportamento para tomada de decisões, mapa semântico para planejamento adaptativo de caminho, lógica fuzzy para controle de recarga de bateria, trajetória semântica para organizar as informações de contexto de maneira estruturada e refinamento de comportamento através de mineração de dados espaço-temporais.

1.1.1 Objetivos Específicos

Parte deste trabalho se apropria de técnicas e modelos existentes na literatura, porém adaptando-os e modificando-os para incorporar as informações de contexto e se adequar na área de robótica móvel. O conjunto destas técnicas modificadas aliado a novas proposições de modelos e procedimentos resultam no trabalho final e podem ser discriminados nos seguintes objetivos específicos:

- Definição de um modelo para categorização de informações de contexto, facilitando o entendimento e impacto destas informações na aplicação;
- Adaptação e avaliação do modelo de representação de trajetória semântica CONSTAnT (BOGORNY et al., 2014) para robótica móvel;
- Incorporação de informações de contexto na criação de árvores de comportamento;
- Criação de um controlador fuzzy para identificar necessidade de recarga de bateria de um robô móvel;
- Construção de um framework para descrever a metodologia de incorporação de contexto em aplicações de robótica móvel;

- Aplicação do framework em cenários para demonstrar potencial uso.

1.1.2 Contribuições da Tese

A problemática da navegação inteligente de robôs móveis pode ser abordada de diversas maneiras diferentes. É possível tratar o problema de planejamento de caminho (STENTZ; MELLON, 1993; OŚMIAŁOWSKI, 2009; WEI et al., 2012), fusão de sensores (LYNEN et al., 2013; SCARAMUZZA et al., 2014), aprendizado de máquina (VASQUEZ; OKAL; ARRAS, 2014; KRETZSCHMAR et al., 2016), mapa semântico (GALINDO et al., 2008; PRONOBIS, 2011; LIM et al., 2013b), etc. A Figura 2 enumera algumas das áreas da literatura relacionadas com navegação inteligente de robôs móveis.

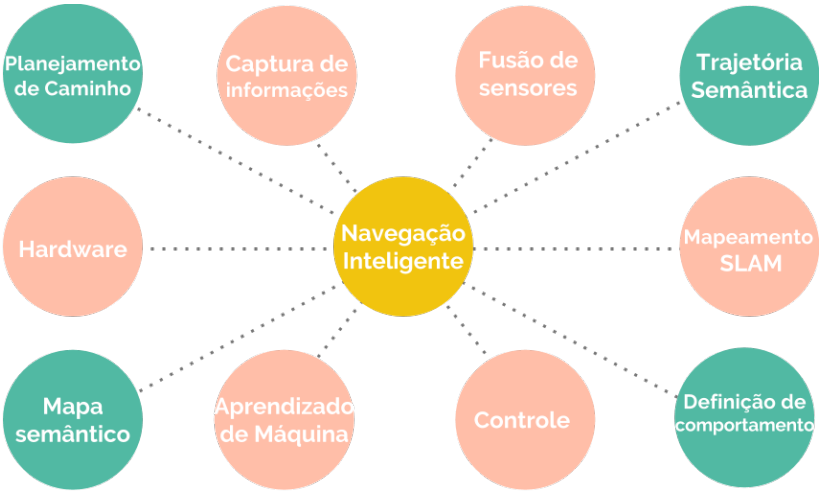


Figura 2 – Áreas relacionadas à navegação inteligente e áreas abrangidas na tese

Nesta tese, a navegação inteligente é definida como a capacidade de um robô móvel navegar de maneira autônoma e eficiente por um ambiente se adaptando a mudanças e novas informações. Para tratar deste problema, o trabalho proposto se foca em quatro áreas principais, destacadas na Figura 2 : planejamento de caminho, trajetória semântica, mapa semântico e definição de comportamento. Apesar das outras áreas relacionadas contribuírem para resolver o problema da navegação

inteligente, é necessário limitar a área de atuação a pesquisa.

Nas áreas de atuação desta tese, entre as contribuições propostas, podemos destacar as seguintes:

- Planejamento de Caminho: Algoritmo de planejamento de caminho considerando como os elementos de contexto modificam o custo de deslocamento entre as regiões;
- Trajetória Semântica: Modelo de representação de trajetória semântica de um robô móvel para armazenamento de execuções de tarefas e extração de conhecimento de histórico de execução através de análise da trajetória semântica;
- Mapa Semântico: Representação do ambiente através de um mapa semântico que enumera não somente as características de definição do ambiente, mas seu estado atual de acordo com o contexto identificado;
- Definição de Comportamento: Modelagem de comportamento do robô móvel considerando as tarefas a serem executadas na aplicação e os elementos de contexto que modificam o comportamento do robô.

Todas as contribuições da tese são compiladas em um framework que define a metodologia e procedimentos necessários para a navegação inteligente de robôs móveis incorporando elementos de contexto.

1.1.3 Organização da Tese

A tese está organizada da seguinte forma: O capítulo 2 aborda trabalhos da literatura que propõem sistemas sensíveis ao contexto, ou seja, sistemas que modificam seu comportamento de acordo com o contexto atual identificado. São abordados trabalhos na área da robótica e de sistemas de navegação.

O capítulo 3 enumera os diversos conceitos utilizados neste trabalho. A natureza da proposta desta tese integra diversas estratégias e tecnologias diferentes para tratar a navegação inteligente de robôs móveis. Cada uma delas é descrita e formalizada neste capítulo.

A contribuição da tese é apresentada no capítulo 4, que inclui as definições formais dos conceitos da abordagem proposta e um framework que integra todos estes conceitos e estratégias.

Para validar o trabalho, o capítulo 5 apresenta os experimentos realizados em um ambiente simulado utilizando o framework proposto

para uma aplicação de patrulha. Outra aplicação é modelada seguindo os passos definidos pelo framework.

Por fim, o capítulo 6 finaliza o trabalho compilando as contribuições da tese e sugere direções para trabalhos futuros.

2 SISTEMAS SENSÍVEIS AO CONTEXTO

Um sistema sensível ao contexto é um sistema que modifica seu comportamento baseado em informações sobre o estado, cenário, usuário, aplicação, etc (DEY, 2001). Sistemas e aplicações que fazem uso de contexto podem resolver problemas distintos de maneiras distintas, cada uma considerando as informações específicas de cada problema. Existem diversos trabalhos na literatura que incorporam contexto em sistemas distintos, conforme será visto a seguir.

2.1 INFORMAÇÕES DE CONTEXTO APLICADO À ROBÓTICA

Aplicações robóticas usualmente requerem um certo nível de autonomia para funcionarem apropriadamente (SPONG; HUTCHINSON; VIDYASAGAR, 2006). Tal autonomia pode ser alcançada através da percepção do robô sobre o ambiente em sua volta. Essa percepção ocorre utilizando sensores para captar informações sobre o ambiente e assim tomar decisões a partir do ambiente em sua volta. Logo, é natural que sistemas robóticos utilizem informações de contexto para aprimorar suas capacidades e desempenho nas mais variadas tarefas.

Um dos primeiros trabalhos de sistemas sensíveis ao contexto aplicado em robótica foi proposto por Turner (1998). O autor apresenta como um agente inteligente deve adaptar seu comportamento de acordo com o contexto observado, dividindo suas informações de contexto em três classes: contexto do ambiente, contexto da tarefa e contexto do agente. O contexto do ambiente se refere às informações que descrevem o ambiente e não dependem das ações do robô. O robô captura essas informações através de seus sensores. Contexto da tarefa são informações específicas da tarefa proposta como local da tarefa, *deadline* para a execução da tarefa e sua prioridade. Por fim, o contexto do robô representa o seu estado atual como mau funcionamento ou nível de bateria.

O robô manipulador de Witzig et al. (2013), por exemplo, utiliza informações de contexto fornecidas pelo usuário para manipular corretamente um recipiente. Apesar do robô poder agarrar um recipiente de diversas maneiras, o tipo de recipiente, seu conteúdo e o propósito da manipulação influenciam diretamente na maneira correta de segurar. Caso o robô deva colocar leite em uma tigela, deve segurar o recipiente do leite pelas laterais. Caso o recipiente do leite esteja vazio e que-

ria retirar de uma mesa, ele pode segurar por qualquer lado, sem se preocupar em derramar o conteúdo.

Esta tese foca em aplicações de robótica móvel. Robôs móveis são dispositivos que se locomovem de maneira autônoma para diversos objetivos diferentes, por exemplo, para transporte, exploração, inspeção, patrulha, montagem, etc (SECCHI, 2008). Pode-se dizer, outras palavras, que robôs móveis são plataformas mecânicas que se locomovem através de um sistema de navegação em um ambiente conhecido ou desconhecido, onde a locomoção possui um certo nível de autonomia, com a finalidade de executar uma tarefa. Robôs móveis são usualmente utilizados em tarefas que apresentam riscos a seres humanos como, por exemplo, transporte de cargas perigosas, manutenção de dispositivos nucleares, exploração subterrânea, etc. Robôs móveis também podem ser utilizados no transporte de medicamentos em um hospital ou em tarefas domésticas.

A navegação de um robô móvel é a sua capacidade de se mover com base em um planejamento de trajetória, dados de sensores e conhecimento do robô para alcançar um objetivo da maneira mais eficiente e confiável quanto possível (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). Assim como seres humanos precisam de informações de contexto para se locomoverem de maneira eficiente e inteligente (HUANG; LIU, 2004), os robôs móveis também fazem uso destas informações para aprimorar sua capacidade de locomoção.

Existem diversas iniciativas na literatura para incorporar contexto na robótica móvel de maneira a melhorar o comportamento do robô. Rocco, Pecora e Saffiotti (2013) propõem um sistema de replanejamento de tarefas para robôs móveis onde exite uma dependência entre tarefas e ocorre um atraso não previsto. O sistema então deve replanejar a ordem que irá executar as tarefas para não ficar ocioso enquanto aguarda as tarefas dependentes. Os autores definem os conceitos de configuração de planos, objetivos e tarefas através de informações de contexto sobre a representação do ambiente atual. Este ambiente engloba o estado do robô e recursos necessários para realização de uma tarefa. O sistema alterna entre planejamento e monitoramento da execução do plano. No primeiro, o algoritmo cria um plano para o robô seguir. No segundo este plano é monitorado através de sensores do robô e é feita a atualização do plano anterior. Na atualização do plano, é possível identificar atrasos ou inconsistências geradas por dependências entre processos, sendo necessário replanejar os planos dos robôs. Este sistema suporta tarefas dependentes entre robôs.

Amato et al. (2015) apresentam uma proposta de planejamento

adaptativo em robótica móvel considerando elementos de contexto. A princípio é apresentada a estrutura do projeto RUBICON (Robotics UBIquitous COgnitive Network) (SEVEN, 2011), que consiste em quatro camadas de software que interagem entre si para controle e planejamento do robô. As camadas são: Camada de Comunicação, Camada de Aprendizagem, Camada de Controle e Camada Cognitiva. O artigo se concentra em mostrar como as camadas de aprendizagem e controle cooperam para a navegação adaptativa do robô. A camada de aprendizagem utiliza dados dos sensores para predizer um estado futuro baseado em histórico. A partir desta predição, a camada de controle planeja o caminho do robô. A camada de controle adapta o plano do robô a partir de novas informações e ao contexto atual. A camada de controle também mede desempenho da execução com o contexto atual para servir de *feedback* na consulta da camada de aprendizagem nos dados passados.

Uma outra proposta de navegação inteligente baseada em contexto é apresentada por Foukarakis et al. (2014). A proposta utiliza AFD (Autômato Finito Determinístico) para selecionar o comportamento adequado do robô segundo o contexto atual. Neste trabalho, os autores dividem os elementos de contexto em três grupos: contexto do usuário, contexto do ambiente e contexto do robô. Os estados do AFD devem ser programados por um especialista, que também observa os elementos de contexto para modelar as transições entre os estados.

O projeto europeu ASCENS (Autonomic Services-Component ENSembles) (WIRSING et al., 2015) reúne diversos trabalhos para a criação de um modelo de sistemas autoconscientes, autoadaptativos e autoexpressivos de maneira paralela e distribuída, com vários comportamentos e interações. O trabalho faz uso de árvores de comportamento modificadas (HÖLZL; GABOR, 2015) e uma ontologia para representação do conhecimento (VASSEV; HINCHEY, 2015) para organizar um enxame de robôs em uma aplicação de busca e resgate de pessoas (PINCIROLI et al., 2015).

Uma outra maneira de incorporar elementos de contexto na navegação de um robô móvel é através de mapas semânticos. Trabalhos na literatura se dividem entre a criação de um mapa semântico (PRONOBIS; JENSFELT, 2012; LIM et al., 2013a) e no seu uso para a navegação inteligente (GALINDO et al., 2008; KO; YI; SUH, 2012). A definição e relação dos trabalhos na literatura sobre mapas semânticos serão apresentados na seção 3.4.

2.2 ELEMENTOS DE CONTEXTO EM APLICAÇÕES DE NAVEGAÇÃO

É possível incorporar elementos de contexto em aplicações de navegação não relacionadas a robôs móveis. Aplicações de navegação, em geral, buscam fornecer ao usuário um caminho entre uma origem e um destino, levando em consideração um mapa. Aplicações de navegação que utilizam elementos de contexto são denominadas de navegação sensível ao contexto.

Um dos exemplos populares mais conhecidos é do aplicativo Waze (MOBILE, 2009). Waze é um sistema de navegação baseado em informações fornecidas por usuários em tempo real (SILVA et al., 2013). A partir destas informações, o Waze calcula rotas de navegação que evitam trânsitos ou acidentes, prevendo um tempo de viagem relativo à atual situação do trânsito. Pode-se considerar que as informações registradas pelo usuário são elementos de contexto sobre o ambiente, uma vez que descreve o estado atual do trânsito da cidade. Por ser uma aplicação específica, o Waze possui poucos tipos de elementos de contexto, tendo pouco poder de expressão sobre o ambiente.

Já o trabalho de Afyouni et al. (2013) foca em modelos espaciais de navegação em ambiente fechado sensíveis ao contexto. Os tipos de contexto são classificados dependendo de sua origem. A Figura 3 sumariza as três dimensões de contexto: contexto do usuário, contexto do ambiente e contexto de execução.

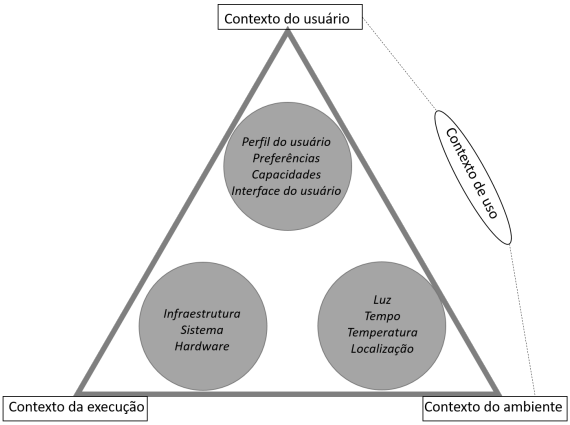


Figura 3 – Dimensões de contexto (Afyouni et al. (2013))

O contexto do usuário abrange fatores referentes a quem usa o sistema como preferências do usuário, capacidades e interface. O contexto do ambiente caracteriza o estado do ambiente com dados como luz, tempo, temperatura e local. O contexto do usuário e do ambiente juntos formam o contexto de uso da aplicação, que descreve o cenário que ela será executada. Já o contexto de execução diz respeito a informações do sistema como infraestrutura e hardware de execução.

A partir destas dimensões de contexto, a navegação é dividida em duas fases: fase estática, onde é gerado um caminho do ponto de origem até o destino levando em consideração o contexto atual; e a fase dinâmica, onde o sistema deve monitorar o progresso do usuário para evitar desvios do caminho planejado, adaptando o caminho caso algum imprevisto aconteça.

Afyouni, Ray e Claramunt (2012) propõem uma arquitetura para consultas espaciais sensíveis ao contexto em ambientes fechados definindo uma arquitetura e linguagem para consultas espaciais sensíveis ao contexto. Nessas consultas, a localização de cada objeto relacionado com a busca influencia na resposta final.

Com o foco em navegação, Afyouni, Ray e Claramunt (2012) utilizam elementos de contexto como localização, restrições de usuários e objetos presentes para indicar qual caminho um usuário humano deve seguir para atingir certo objetivo. Esta arquitetura permite responder perguntas como: *listar os 10 usuários mais próximos de um objeto o1*, *verificar se o usuário u1 consegue alcançar o objeto o2*, *qual o caminho mais curto para a sala s1 passando pelo corredor c2*, etc.

Outra iniciativa de navegação sensível ao contexto é proposta por Wang, Hwang e Ting (2011). A proposta, chamada de UbiPaPaGo, oferece um planejamento de caminho baseado no contexto atual do mapa e preferências do usuário. Os autores dividem o contexto em contexto do usuário, do GPS (Global Positioning System) e do mapa, além de distinguir entre contexto estático e dinâmico em cada uma das classes.

A arquitetura do UbiPaPaGo, ilustrada na Figura 4, é dividida no servidor geral, *web service*, sensores, dispositivos e rede. Uma vez que o sistema é baseado fortemente em fornecer serviços *online*, uma das premissas do planejamento de caminho sensível ao contexto é observar no mapa quais locais têm pontos de acesso à internet. O planejamento do caminho leva em consideração o menor caminho entre dois pontos que tenha um bom sinal de internet.

Saeedi, Moussa e El-Sheimy (2014) explora o uso de *smartphone* para uma navegação sensível ao contexto. Para a autora, aplicações

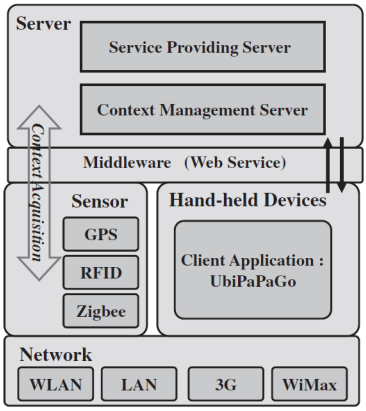


Figura 4 – Camada da arquitetura do UbiPaPaGo (WANG; HWANG; TING, 2011))

sensíveis ao contexto devem perceber o ambiente que estão inseridas e modificar seu comportamento sem precisar de ações do usuário. Apesar de não implementar de fato um planejamento de caminho, a autora consegue, a partir de fusão dos sensores do *smartphone*, detectar o tipo de atividade do usuário.

A figura 5 resume a arquitetura de fusão dos sensores e detecção da atividade. A partir de aplicação de regras fuzzy com os elementos de contextos capturados do sensores, os autores identificam quando o *smartphone* está na mão do usuário, no bolso, em ambiente aberto, em ambiente fechado, etc.

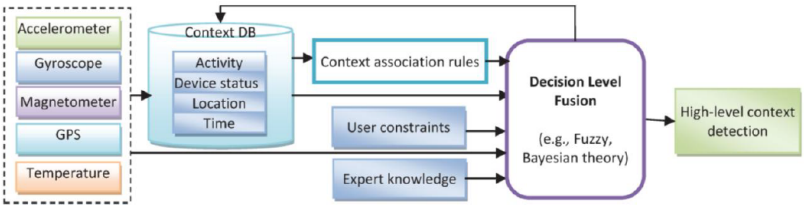


Figura 5 – Arquitetura da detecção de contexto de Saeedi, Moussa e El-Sheimy (2014)

A partir da definição do estado do usuário, é possível modificar o comportamento do algoritmo de navegação. Por exemplo, em na-

vegações com auxílio da câmera do aparelho, saber o posicionamento correto do *smartphone* (se está de pé, deitado, olhando para frente, para o chão) é essencial para informar as direções corretas na navegação ou para estimar velocidade do usuário.

Como os trabalhos citados focam em aplicações para humanos, eles não tratam de individualidades previstas na robótica móvel. Para um robô móvel, além do caminho a ser seguido, é necessário definir os comportamentos necessários para os possíveis cenários encontrados.

2.3 SÍNTESE DO CAPÍTULO

Para aplicar elementos de contexto na navegação inteligente de robôs móveis é necessário entender como essa informações são utilizadas em outros trabalhos da literatura. Este capítulo apresentou o conceito de sistemas sensíveis ao contexto e alguns trabalhos relacionados. Os trabalhos foram divididos em dois grupos: informações de contexto aplicado à robótica e elementos de contexto em aplicações de navegação.

Na robótica móvel, um sistema sensível ao contexto deve modificar o comportamento do robô considerando o contexto que ele está inserido. O primeiro trabalho a utilizar o termo de sistema sensível ao contexto em robôs móveis foi proposto por Turner (1998). Outros trabalhos incorporaram contexto e a modificação do comportamento do robô utilizando tecnologias e metodologias diferentes (AMATO et al., 2015; FOUKARAKIS et al., 2014; TECHNOLOGIES, 2015).

Outros trabalhos explorados focam no uso de contexto para aplicações de navegação. Apesar do foco destes trabalhos ser auxiliar a navegação humana por um ambiente, é possível transpor alguns conceitos e definições de navegação humana para a navegação inteligente de robôs móveis (AFYOUNI et al., 2013; AFYOUNI; RAY; CLARAMUNT, 2012; WANG; HWANG; TING, 2011).

A partir destes estudos foi possível analisar como propostas diferentes lidam com a navegação em sistemas sensíveis ao contexto. Para robôs, o contexto auxilia na definição do comportamento adequado para cada tipo de situação, enquanto em sistemas de navegação o contexto auxilia a definir o melhor caminho para se seguir ou a identificar qual foi o comportamento adotado pelo usuário.

A Tabela 1 sumariza algumas características dos trabalhos apresentados com a proposta desta tese. Esta comparação não tem como objetivo definir a melhor abordagem para tratar o problema de navegação inteligente, mas sim explicitar as diferenças entre as abordagens.

Para cada proposta, a Tabela 1 caracteriza os seguintes atributos: O tipo de navegação, que se refere à navegação de robôs ou de humanos; o contexto, que caracteriza a modelagem de contexto da proposta, podendo ser fixa caso trate sempre dos mesmos contextos; o planejamento que diz se a proposta contempla um algoritmo de planejamento de caminho para a navegação; o comportamento que identifica se a tomada de decisão de comportamento é através da técnica de autômatos finitos determinísticos (AFD) ou árvore de comportamento (BT); mapa que identifica qual tipo de mapa a proposta utiliza; e histórico que identifica quais informações da execução das tarefas são armazenadas. Em alguns casos existem características que não foram definidas no trabalho (ND) ou que a característica não se aplica na proposta (NSA).

Tabela 1 – Diferenças entre abordagens de aplicação de contexto em navegação. NSA: não se aplica. ND: não definido no trabalho

Proposta	Tipo	Contexto	Planejamento	Comportamento	Mapa	Histórico
(TURNER, 1998)	Robô	Semântica	ND	AFD	ND	ND
(AMATO et al., 2015)	Robô	ND	Sim	AFD	Métrico	Estado do ambiente
(FOUKARAKIS et al., 2014)	Robô	Semântica	Métrico	AFD	Métrico	ND
(WIRSING et al., 2015)	Robô	ND	Sim	BT	Métrico	Estado do ambiente
(AFYOUNI et al., 2013)	Humana	Semântica	Sim	NSA	Topológico	ND
(WANG; HWANG; TING, 2011)	Humana	Fixa	Sim	NSA	Semântico	ND
Proposta da Tese	Robô	Semântica, Espacial e Temporal	Sim	BT	Semântico	Trajetória Semântica

3 FUNDAMENTAÇÃO TEÓRICA

Um robô móvel deve se locomover no espaço durante certo período de tempo até atingir seu objetivo que consiste basicamente na realização de uma tarefa planejada. Um robô normalmente se locomove conforme a tarefa que foi planejada, reagindo ao que ocorre no ambiente ou em uma combinação dessas ações. Essas estratégias são descritas na literatura como deliberativa, reativa ou híbrida (SECCHI, 2008).

Secchi (2008) defende que a estratégia deliberativa é baseada em estratégias simbólicas, onde o modelo do ambiente deve ser bem semelhante ao real para atingir os objetivos desejados. Nesta estratégia, a trajetória do robô é calculada antes da execução, mas o robô não reage a imprevistos no caminho. Por exemplo, caso tenha um obstáculo não identificado previamente no caminho da trajetória do robô, haverá uma colisão. Por outro lado, estratégias deliberativas podem garantir o caminho de custo mínimo para um robô alcançar um objetivo. Por sua simplicidade, a estratégia deliberativa foi a primeira das estratégias utilizadas para a locomoção de robôs móveis, sem entretanto, mostrar-se adequada à navegação autônoma.

Já estratégias reativas são estratégias baseadas em estímulos, ou seja, o robô irá decidir o rumo de sua trajetória a partir da análise de seus sensores, sem um conhecimento prévio do ambiente. Secchi (2008) aponta que estratégias reativas são melhores para sistemas que necessitam de uma resposta rápida com baixo custo computacional. Entretanto, estratégias reativas podem gerar trajetórias não-ótimas e que sequer consigam atingir seu objetivo. Sem um conhecimento prévio do ambiente, por exemplo, um robô que se locomove apenas com estímulos de seus sensores pode não conseguir sair de situações em que as reações do robô levem a uma navegação cíclica sem que ele seja capaz de sair do ambiente onde está.

Uma abordagem mais adequada para o problema de planejamento e execução de trajetórias de robôs móveis é utilizar uma estratégia híbrida. Estratégias híbridas buscam resgatar as vantagens das outras duas abordagens, utilizando um conhecimento prévio do ambiente para poder se locomover e fazendo o uso de sensores para identificar objetos e fatores não modelados anteriormente. O uso destes sensores permite que o robô possa reagir a imprevistos, enquanto o conhecimento prévio do ambiente permite identificar rotas alternativas que possam levar ao alcance do objetivo, retomando a trajetória

previamente planejada após evitar um obstáculo não previsto.

Uma estratégia híbrida é a melhor abordagem para modelar e monitorar trajetórias que incorporam informações de contexto que auxiliam a melhorar a capacidade reativa do planejamento híbrido sem perder as vantagens do que foi planejado. A partir desta estratégia pode-se planejar uma trajetória com dados do ambiente e, com o uso de um mapa semântico ou de outra forma de organização das informações coletadas, o robô executa a trajetória modificada com essas novas informações, ajustando seu movimento a fatores não previstos no ambiente. A implementação desta forma de navegação pode ser feita a partir do conceito de autômatos finitos (FOUKARAKIS et al., 2014; KURT; ÖZGüNER, 2013; MARINO et al., 2009) ou de árvores de comportamento (COLLEDANCHISE; OGREN, 2014; MARZINOTTO et al., 2014) ou ainda sob a forma de agentes, tendo por base a inteligência artificial ou outra forma de aprendizado (SEYHAN; ALPASLAN; YAVAŞ, 2013).

Além da robótica móvel, outros conceitos e tecnologias complementam esta tese. Para compreender o trabalho realizado é necessário apresentar alguns conceitos básicos que fazem parte da tese: informações de contexto, análise de trajetórias de objetos móveis, árvore de comportamento, mapa semântico e framework ROS (Robot Operating System). Eles serão explorados nas seções a seguir.

3.1 ELEMENTOS DE CONTEXTO

Humanos são seres inteligentes que conseguem se adaptar a cenários e situações, modificando seu comportamento e suas ações de acordo com uma base construtiva de conhecimento que, para cada situação específica, existe um ou mais comportamentos adequados. Tal processo cognitivo se dá pelo acúmulo de conhecimento, experiência e cultura que um humano adquire durante sua vida. Essa habilidade dá ao ser humano a capacidade de responder ao cenário de acordo com o *contexto* que se apresenta.

Não existe um consenso na definição do termo *contexto* na literatura. Bazire e Brézillon (2005) dizem que a definição de contexto depende do próprio contexto do uso do termo. O autor analisou 66 definições do termo contexto em diversas áreas de pesquisa como, por exemplo, inteligência artificial, documentação, ergonomia cognitiva, psicologia cognitiva, negócios, filosofia, linguística entre outros. Existem vários conceitos na psicologia que se aproximam de uma definição de contexto. Situação, campo, meio social e *background* são exemplos destes concei-

tos que integram uma noção geral de contexto. Os autores criticam outros trabalhos devido à utilização do termo contexto com a suposição de que o leitor já conhece seu significado. Isso leva a uma série de divergências e definições diferentes sobre o mesmo termo. As definições de contexto são apresentadas como um conjunto de restrições que influenciam o comportamento de um sistema (ou de um usuário) em uma tarefa, mas não é possível indicar uma definição geral do termo.

No campo da informática e de desenvolvimento de sistemas, a noção de contexto é usada para definir sistemas que modificam seu comportamento de acordo com o cenário ou situação. Segundo Dey (2001), esses sistemas são denominados sistemas sensíveis ao contexto. A definição de contexto está associada a uma entidade que pode ser uma pessoa, lugar ou objeto que é relevante para a aplicação e usuário, incluindo ambos. Com isso resulta que, para a utilização que se pretende dar neste trabalho, será considerada a seguinte concepção de contexto:

Contexto é qualquer informação que pode ser utilizada para caracterizar a situação de uma entidade.

Um sistema sensível ao contexto utiliza contexto para prover informação e/ou serviços relevantes ao usuário, onde a relevância depende da tarefa do usuário. Com essa definição de contexto, Dey indica que toda e qualquer informação obtida, que seja relevante na aplicação, pode ser considerada como contexto e que aplicações que utilizam essa informação de maneira a modificar seu comportamento são sistemas sensíveis ao contexto.

A problemática de se construir sistemas sensíveis ao contexto resume-se em três pontos principais: como obter as informações de contexto relevantes, como estruturar as informações para serem utilizáveis na aplicação e como essas informações vão impactar no funcionamento do sistema. Seres humanos utilizam seus cinco sentidos para perceber informações de contexto no ambiente. Visão, tato, olfato, paladar e audição permitem ao ser humano perceber essas informações. Na robótica móvel, informações de contexto podem ser obtidas através de sensores instalados no ambiente ou no robô. Sensores são dispositivos que capturam informações sobre os elementos presentes no ambiente, servindo de perceptores dos sentidos do robô.

Existem diversos tipos de sensores na robótica, podendo classificá-los de diversas maneiras. Secchi (2008), por exemplo, define que um sensor pode ser classificado de acordo sua interação com um objeto, sendo um sensor de contato ou sem contato. Os sensores também são classificados de acordo com o que está sendo medido. Caso o sensor

capture informações do robô, é um sensor proprioceptivo. Já sensores que capturam informações do ambiente são sensores exteroceptivos. Os sensores na robótica também são classificados de acordo com seu princípio de funcionamento como, por exemplo, carga elétrica, radiação térmica, resistência, indutância, etc.

No escopo da tese, o contexto do ambiente e do robô serão capturados por sensores ou informados por outro meio de comunicação. Por exemplo, um usuário pode informar ao robô que um evento ocorreu. Essa informação também é retratada como informação de contexto, uma vez que descreve uma situação de uma entidade. Outras formas de obtenção de contexto da tese são informações pré-definidas ou obtidas de consultas a banco de dados ou de outra fonte de informação externa que não seja necessariamente um sensor. Basicamente, no escopo deste trabalho, o que se torna relevante para a autonomia do robô é a comunicação que ele recebe, seja de um sensor ou de outra fonte qualquer.

3.2 TRAJETÓRIA E TRAJETÓRIA SEMÂNTICA

Uma trajetória de objeto móvel no sentido originalmente usado é uma representação do seu movimento. A partir da análise da trajetória é possível identificar padrões de movimentos e comportamentos destes objetos. Braz e Bogorny (2012) classificam estes padrões como padrões geométricos ou semânticos. Um padrão que se limita aos atributos físicos de uma trajetória é chamado de padrão geométrico, enquanto um padrão que utiliza elementos de contexto de uma trajetória semântica para identificar padrões de comportamento é chamado de padrão semântico.

As figuras 6 e 7 ilustram a diferença entre um padrão geométrico e um padrão semântico para um mesmo conjunto de dados. A partir de uma região B, o padrão geométrico, representado na Figura 6, identifica que todas trajetórias cruzaram esta região. A informação relevante, neste caso, é a origem da trajetória em um dado instante, a passagem por diferentes pontos intermediários e a chegada ao destino em um instante final de tempo. Já em uma aplicação específica, representada na Figura 7, onde elementos de contexto identificam a origem e destino das trajetórias, são encontrados dois padrões semânticos: trajetórias saindo de uma origem H e indo para um destino C passando pela região B e trajetórias saindo de outra origem H e indo para um destino R passando pela região B havendo, portanto, uma qualificação das origens

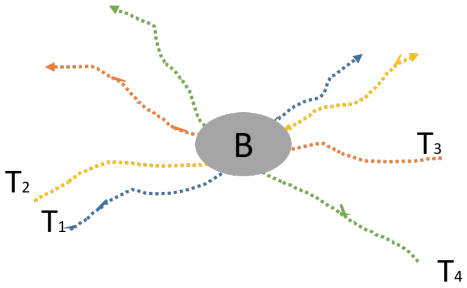


Figura 6 – Padrão geométrico entre trajetórias. Adaptado de (BOGORNY; KUIJPERS; ALVARES, 2009)

e destinos representando informações suplementares ao caso anterior.

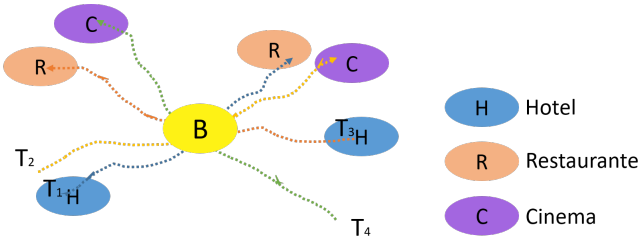


Figura 7 – Padrão semântico entre trajetórias. Adaptado de (BOGORNY; KUIJPERS; ALVARES, 2009)

Analisando as figuras 6 e 7 notam-se diferenças entre as informações representadas pelo padrão geométrico e pelo padrão semântico. As informações são de naturezas distintas. Enquanto a informação geométrica permite analisar as características estruturais das trajetórias, as trajetórias semânticas possuem informações referentes ao objetivo do agente, que foram enriquecidas com informações de contexto.

Uma trajetória que representa apenas o padrão geométrico é denominada de trajetória, enquanto uma trajetória que representa padrões semânticos é chamada de trajetória semântica. Nesta seção será explorado o conceito de ambos os tipos de trajetória.

3.2.1 Trajetória

Uma trajetória é um dado espaço temporal. Dados espaço temporais são dados com uma especificação no espaço e outra no tempo. Esta tese utiliza a formalização de trajetória proposta em (SIQUEIRA; BOGORNY, 2011) e apresentada na Definição 1. A representação espacial normalmente necessita de 3 componentes para a localização no espaço (eixos x, y e z). Para simplificação, neste trabalho a movimentação será considerada no plano (2D) e, portanto, a representação necessita apenas duas componentes espaciais, como ilustra a Figura 8, com os pontos nos eixos x e y.

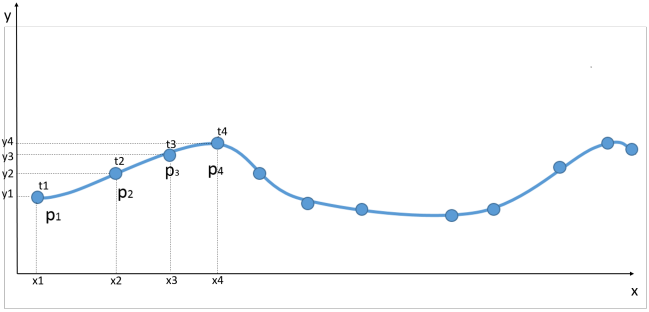


Figura 8 – Exemplo de uma trajetória em um plano

Definição 1. Trajetória
Uma *trajetória* T é uma lista ordenada de pontos que associam para cada instante t uma posição no espaço. Normalmente é representada por uma lista ordenada $\langle tid, p_0, p_1, \ldots, p_n \rangle$, onde tid é o identificador da trajetória, $p_i = (x_i, y_i, t_i)$ é um ponto da trajetória onde (x_i, y_i) são as coordenadas geográficas que representam a localização espacial e t_i é o instante de tempo em que a coordenada foi capturada. O conjunto $x_i, y_i, t_i \in R$ para cada $i = 0, \ldots, n$ e $t_0 < t_1 < \cdots < t_n$.

Uma trajetória possui propriedades a partir de suas dimensões de espaço e tempo. Giannotti, Giannotti e Pedreschi (2008) dividem essas propriedades em duas classes: propriedades gerais (ou propriedades da trajetória) e propriedades momentâneas (ou propriedades de cada ponto da trajetória). Essas propriedades estão representadas na Figura 9. Propriedades momentâneas são referentes às propriedades mutáveis ao longo da trajetória e são próprias de cada ponto. Atributos como instante de tempo de captura do ponto, coordenada geográfica, direção,

velocidade e aceleração fazem parte das propriedades momentâneas. Por exemplo, o ponto p_4 na Figura 9 foi capturado no dia 06 de janeiro de 2012 às 07:49:31 com uma velocidade de 60km/h.

Já as propriedades gerais se referem à trajetória inteira ou a uma subtrajetória. Uma subtrajetória é um subconjunto ordenado de pontos de uma trajetória. O comprimento total, duração total, direção do movimento e a velocidade média são exemplos de atributos gerais. Por exemplo, a trajetória da Figura 9 percorreu ao todo 10km em 25 minutos.

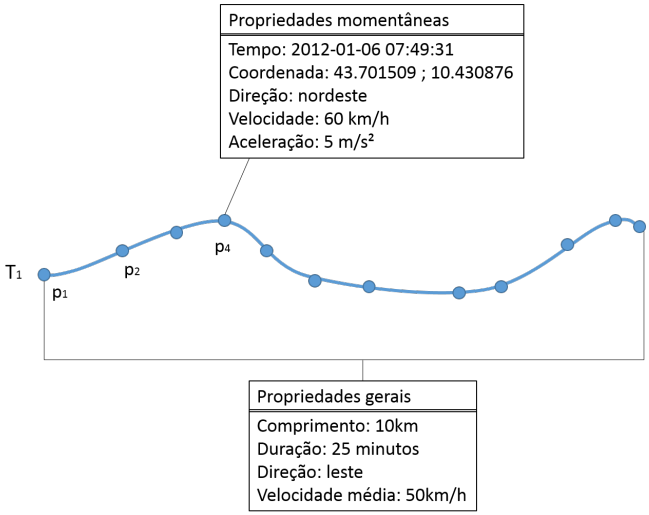


Figura 9 – Propriedades gerais e momentâneas de uma trajetória T_1

É possível analisar o relacionamento entre trajetórias a partir da comparação de suas propriedades momentâneas e gerais. Estas comparações podem ser realizadas através de operadores simples como maior que, menor que ou igual. Um padrão de comportamento de trajetória pode ser identificado através da análise destes relacionamentos. As relações na dimensão espacial são chamadas de relacionamentos espaciais. Trajetórias podem ter relacionamentos espaciais com outras trajetórias, com objetos fixos, áreas, ou qualquer outro elemento de mesma dimensão espacial. Braz e Bogorny (2012) dividem os relacionamentos espaciais em quatro classes: orientados, topológicos, métricos e nebulosos.

Relacionamentos orientados se referem a orientação dos objetos

a partir de um objeto de referência. Assim um objeto pode estar, a partir da referência, localizado ao sul, ao norte, acima, à esquerda, a leste, a oeste, etc. Por exemplo, a Figura 10(a) ilustra um objeto O_1 ao sul de uma trajetória T_1. Relacionamentos topológicos descrevem a posição relativa entre os objetos e o espaço. A Figura 10(b) exemplifica um relacionamento topológico onde a trajetória T_2 intercepta uma região A_1. Relacionamentos topológicos também podem ocorrer entre trajetórias. Por exemplo, a Figura 10(d) ilustra as trajetórias T_1 e T_2 se cruzando. Relacionamentos métricos descrevem tamanhos e distância de objetos. A Figura 10(c) ilustra duas trajetórias próximas, T_1 e T_2, onde a menor distância do ponto p_1 com a trajetória T_2 é com o ponto q_1, enquanto a menor distância do ponto p_2 com a trajetória T_2 é com o ponto q_2. Relacionamentos nebulosos tratam objetos sem limites bem definidos como lagos que mudam de tamanho em períodos de chuvas.

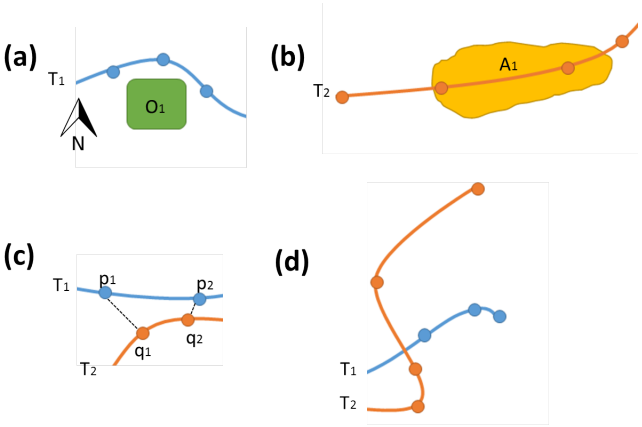


Figura 10 – Tipos de relacionamentos espaciais: (a) trajetória ao norte de ponto de referência (orientado), (b) trajetória dentro de uma área (topológico); (c) distância entre trajetórias (métrico); (d) trajetórias que se intersectam (topológico)

Além de relacionamentos espaciais, também é possível analisar trajetórias a partir de relacionamentos temporais. Relacionamentos temporais são calculados através da componente tempo das trajetórias. Cao, Mamoulis e Cheung (2007) buscam identificar *padrões periódicos*, onde um objeto segue a mesma rota em tempos regulares. Por exemplo, uma pessoa pode, em dias de semana, sair de um local A às 7 horas, ir para o um local B, ficar até as 18 horas e retornar

ao lugar inicial. Os autores definiram *padrões periódicos* como uma sequência de regiões intersectadas pela trajetória, onde estas regiões podem ser informadas pelo usuário ou calculadas pelo método. Quando calculadas, as regiões são identificadas com o algoritmo de clusterização *DBSCAN*(Density-Based Spatial Clustering of Applications with Noise) (ESTER et al., 1996), que identifica regiões densas.

A combinação da dimensão espacial e temporal da trajetória possibilita a análise de relacionamentos espaço temporais. Relacionamentos espaço temporais são aqueles entre trajetórias que analisam juntamente as propriedades de tempo e espaço. Laube, Kreveld e Imfeld (2005) foram pioneiros na identificação de padrões de comportamento de trajetórias. Neste trabalho, os autores definem quatro tipos de comportamento de trajetórias: *convergência, encontro, flock e liderança*. Convergência é um comportamento onde várias trajetórias convergem para um mesmo ponto ou mesma região. Um encontro é um comportamento onde duas ou mais trajetórias se encontram no mesmo local e no mesmo intervalo de tempo. *Flock* são trajetórias que se movimentam próximas por um determinado período de tempo. Liderança é um comportamento semelhante ao *flock*, mas com uma trajetória que lidera à frente das demais.

A Figura 11 ilustra o padrão de *flock* e de liderança definidos por Laube, Kreveld e Imfeld (2005). O gráfico (à esquerda) representa o número de objetos em um determinado período de tempo e a sua direção. Por exemplo, no *flock* temos 4 objetos que, no mesmo período de tempo, estão se movimentando na mesma direção, enquanto que na liderança temos um objeto andando em uma direção por um determinado tempo e, logo após, dois objetos começam a se movimentar na mesma direção. A proximidade entre os objetos está representada no lado direito da Figura 11.

As trajetórias permitem também identificar padrões analisando a velocidade e a direção dos objetos móveis. Carboni e Bogorny (2011) propõem um método para identificar comportamentos anômalos em trajetórias de carros, reconhecendo motoristas com direção perigosa. Um comportamento anômalo é identificado quando o motorista apresenta aceleração brusca, desaceleração brusca ou mudança de direção brusca. Um motorista que apresenta essas características pode perder o controle do veículo, oferecendo risco para si próprio e para outros, sendo classificado pelo método como direção perigosa.

Trajetórias são limitadas a informações espaço temporais, sendo possível fazer apenas análises sobre suas propriedades momentâneas e gerais. Apesar da trajetória ser útil para identificar padrões de mo-

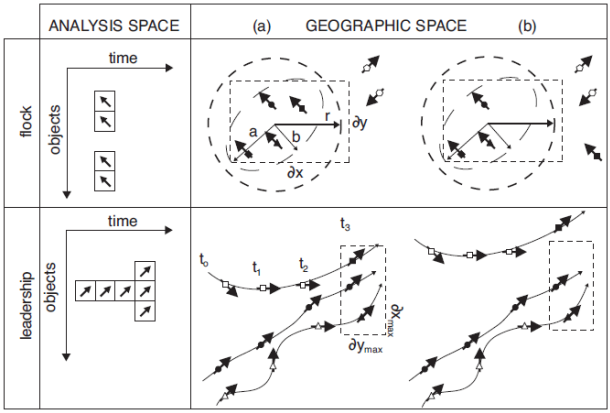


Figura 11 – Exemplo de flock e liderança de (LAUBE; KREVELD; IMFELD, 2005)

vimento, ela não é suficiente para responder perguntas mais complexas. Onde as trajetórias se encontraram? Ou por que uma trajetória está liderando as outras? Se um grupo de trajetórias andaram juntas formando um *flock*, por que elas se separaram? Ou sobre o comportamento anômalo de Carboni e Bogorny (2011), por que o motorista freou bruscamente? Tinha um buraco na pista? Ou um pedestre atravessando? Ou o semáforo fechou? Diversas questões permanecem em aberto quando se observam apenas as trajetórias. Para dar mais significado à trajetória e aos padrões encontrados, surgiu a definição de Trajetória Semântica.

3.2.2 Trajetória Semântica

Uma trajetória semântica é uma trajetória enriquecida com anotações (PARENT et al., 2013). Tais anotações agregam mais informações à trajetória, sendo possível extrair mais conhecimento destes dados. Essas anotações são definidas como elementos de contexto.

A pesquisa na área de trajetórias semânticas, em comparação com outras áreas de conhecimento de computação, é recente. Spaccapietra et al. (2008) criaram um modelo conceitual de trajetórias chamado de *stops and moves*. O modelo propõe uma segmentação da trajetória em partes importantes para aplicação. As partes onde a trajetória permaneceu um período de tempo é denominada de *stops*. Já *moves*

são as partes da trajetória que conectam dois *stops* consecutivos. O segmento de início da trajetória até o primeiro *stop* ou o segmento do último *stop* até o fim da trajetória também são chamados de *moves*.

Spaccapietra et al. (2008) definiram apenas o modelo, mas não os algoritmos para instanciá-lo. O primeiro algoritmo desenvolvido para instanciar o modelo de stops e moves e adicionar semântica em trajetórias foi o IB-SMoT (Intersection Based Stops and Moves of Trajectories), proposto por Alvares et al. (2007). No algoritmo, os autores identificam stops a partir da intersecção da trajetória com regiões geográficas importantes. Estas regiões são consideradas elementos de contexto. Por exemplo, a Figura 12 ilustra a identificação de stops e moves de trajetórias em uma aplicação de turismo. As regiões X, Y, A, B e C são locais como pontos turísticos ou hotéis. O algoritmo então identifica um stop quando uma trajetória intersecta uma destas regiões por um período mínimo de tempo. Logo, um stop identificado pelo IB-SMoT acrescenta semântica, identificando um local importante para aquela trajetória.

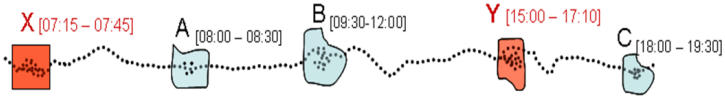


Figura 12 – Identificação de stops e moves proposto por Alvares et al. (2007)

Existem outros algoritmos que instanciam o modelo de stops e moves como o CB-SMoT (Clustering-Based Stops and Moves of Trajectories) (PALMA et al., 2008), onde os stops são partes da trajetória de velocidade baixa, ou o DB-SMoT (Direction-Based Stops and Moves of Trajectories) (ROCHA et al., 2010), onde os stops são partes da trajetória com mudança de direção constante.

O modelo de stops e moves permite visualizar uma trajetória baseando-se em segmentos importantes para a aplicação (stops) e segmentos de ligação (moves), mas limita a quantidade de informação da trajetória semântica. O modelo não cobre pontos importantes como, por exemplo, informações geográficas do local onde a trajetória se localiza, eventos temporais, características do objeto ou de seus objetivos. Visando criar uma modelagem para trajetória semântica mais completa, Bogorny et al. (2014), junto com o proponente da presente tese, propuseram o modelo CONSTAnT (CONceptual model of Semantic TrAjecTories), ilustrado na Figura 13.

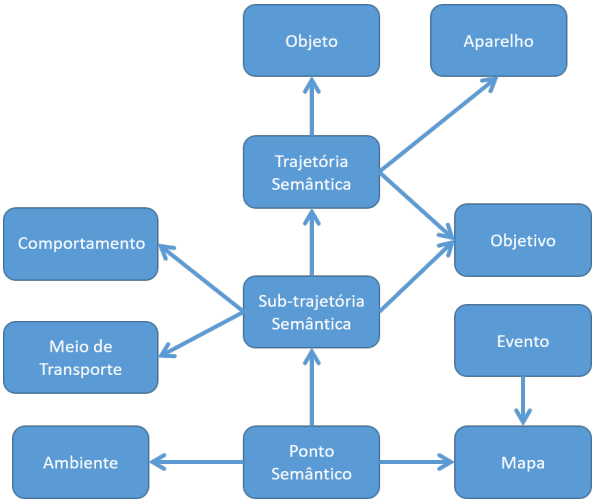


Figura 13 – Modelo conceitual do CONSTAnT

O modelo representa a trajetória de um objeto móvel de maneira hierárquica, onde cada nível de maior granularidade representa um conjunto do nível de menor granularidade. É possível resumir as entidades do modelo CONSTAnT da seguinte maneira:

- **Ponto Semântico:** Menor granularidade da trajetória. Representa cada ponto do movimento e suas informações semânticas;
- **Ambiente:** Representa informações sobre o ambiente onde o ponto está localizado;
- **Mapa:** Representação do mapa do local onde um ponto está localizado. Pode ser uma rua, um quarto, uma cidade, etc;
- **Evento:** Um evento é um acontecimento que tem uma localização e uma duração. Por exemplo um show musical de fim de ano em Florianópolis;
- **Objetivo:** O objetivo é o propósito do movimento. Pode ser um objetivo geral quando é relacionado com a trajetória ou um objetivo específico quando relacionado com a sub-trajetória;
- **Meio de Transporte:** É o modo que o objeto se deslocou, podendo ser a pé, de carro, de avião, etc;

- **Comportamento:** É o comportamento apresentado pelo objeto durante uma sub-trajetória. Um objeto pode apresentar diversos comportamentos diferentes durante uma trajetória completa;
- **Sub-trajetória Semântica:** A sub-trajetória é um conjunto de pontos de mesmo objetivo, mesmo meio de transporte ou mesmo comportamento;
- **Trajetória Semântica:** A trajetória semântica é o conjunto de todas as sub-trajetórias e todos os pontos, representando todo o movimento de um objeto durante a trajetória;
- **Aparelho gerador de trajetória:** Representa o aparelho que capturou os pontos do objeto. Ele pode ter informações sobre a precisão e o modo de captura (por tempo, por distância, etc);
- **Objeto:** É o agente que realiza o movimento capturado. Um objeto pode ter várias trajetórias.

CONSTAnT se baseia em três informações principais para a estrutura de seu modelo: os pontos da trajetória, o comportamento apresentado e elementos de contexto. Os autores definem contexto como informações sobre o objeto móvel que podem ser extraídas do mundo real (temperatura, clima) ou definidas pela aplicação (objetivo, meio de transporte). O comportamento é um conjunto de características que uma trajetória (ou sub-trajetória) apresenta em determinado momento. Na Figura 13 a direção das setas infere a relação de posse da informação onde, por exemplo, um ponto semântico tem associado a ele informações sobre o ambiente e o local onde se encontra. A partir dos três tipos de informação, o modelo pode ser dividido em duas partes: informações simples (objeto, aparelho gerador, trajetória semântica, sub-trajetória semântica, ponto semântico, ambiente, lugar e evento) e informações complexas (comportamento, meio de transporte, objetivo).

Informações simples são informações que já vêm prontas de uma fonte de dados ou são inseridas pelo usuário. A trajetória é capturada pelo aparelho gerador, as variáveis de ambiente (como clima e temperatura) podem ser obtidas de outros dispositivos medidores ou de sites de informações climáticas, eventos podem ser cadastrados pelo usuário, etc.

Já informações complexas são informações que necessitam de algoritmos de mineração de dados para extrair tais informações. Por exemplo, analisando a velocidade e a localização do objeto pode-se identificar o meio de transporte, ou analisando a semântica dos pontos

de saída e de chegada pode-se inferir o objetivo do movimento (por exemplo, se sair de um hotel para um ponto turístico então o objetivo é conhecer a cidade, enquanto sair para um restaurante o objetivo é alimentar-se) ou analisando a trajetória pode-se identificar o comportamento apresentado pelo objeto. Apesar destas informações serem complexas, caso já tenha conhecimento, elas podem ser identificadas pelo próprio usuário.

A Figura 14 ilustra uma trajetória semântica de um turista representada pelo modelo CONSTAnT. O objeto, de nome Artur, carregava um GPS para gravar sua trajetória. O objetivo geral da trajetória é visitar Roma, enquanto o objetivo de uma sub-trajetória específica que se iniciou as 15hr e foi até as 17hr era visitar o Coliseu. No momento em que visitava o Coliseu, Artur estava se locomovendo a pé (meio de transporte) e tinha um comportamento de *flock*, pois andava com um grupo de turistas. O dia estava ensolarado (ambiente) e os pontos dessa sub-trajetória estavam localizados no Coliseu (lugar), durante uma visita turística (evento).

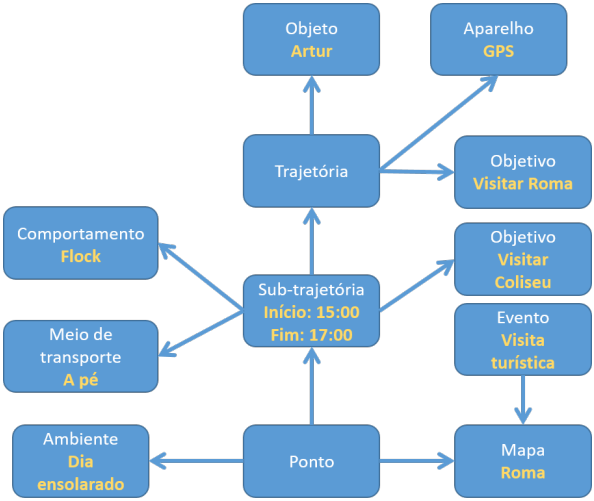


Figura 14 – Instância do modelo CONSTAnT para uma aplicação de turismo

É importante observar que, para instanciar esse modelo, é necessário um conjunto de elementos de contexto sobre diversas informações. Apesar do modelo ser flexível no ponto em que não é necessário instanciar cada informação (por exemplo, pode-se ignorar o ambiente

caso não se tenha informações sobre ele), não é possível representar uma trajetória semântica somente com dados de GPS. Como dito anteriormente, um GPS só fornece a trajetória, sendo necessário buscar elementos de contexto em outras fontes de dados. Portanto, para dados de GPS, é necessário o desenvolvimento de ferramentas que auxiliem o enriquecimento de trajetórias com informações semânticas. Para isso, o proponente desta tese ajudou no desenvolvimento da ferramenta *DayTag* (RINZIVILLO et al., 2013), para anotação semântica de trajetórias. A ferramenta gera stops e moves com o método CB-SMoT (PALMA et al., 2008), identificando locais de partida e chegada de cada trecho da trajetória. A ferramenta transforma trajetórias em um diário do movimento do objeto e, a partir do modelo CONSTAnT, fornece uma interface para preencher as informações de contexto de cada trecho do movimento. Cada viagem (move) ou permanência no local (stop) pode ser anotada com o objetivo, meio de transporte, informações climáticas, etc. *DayTag* armazena as trajetórias, as sub-trajetórias e a anotação semântica. Essa ferramenta permite inserir os elementos de contexto diretamente na trajetória pelo usuário.

Outra abordagem para anotação semântica de trajetórias foi definida por Fileto et al. (2013). Essa abordagem se baseia em uma ontologia, denominada *Baquara*, que utiliza *linked data* de dados abertos na internet. Ontologia é um modelo que representa conceitos e relacionamentos entre estes conceitos dentro de um domínio. Já a *linked data* é uma forma estruturada de organizar informação na internet, organizando os dados a partir de uma ontologia. Esta abordagem permite adquirir elementos de contexto em dados da internet e relacioná-los com uma trajetória. Por exemplo, o termo *Corcovado* é do tipo atração turística e tem relacionamento espacial com a cidade de Rio de Janeiro. Os autores utilizam informações de localização e tempo da trajetória para buscar informações do local ou eventos ocorridos no mesmo horário em bancos de dados de *linked data*. Utilizando fotos georreferenciadas, isto é, fotos que possuem informações sobre o local de captura, descobrem-se todas as fotos publicadas em um raio de 10 km do Corcovado. Baquara foi a primeira proposta de utilização de *linked data* em análise de trajetórias.

Trajetórias semânticas possuem informações para extrair conhecimento usando elementos de contexto. Existem várias vantagens em se utilizar trajetórias semânticas sobre trajetórias. Por ter mais informações, é possível descrever melhor os movimentos e comportamentos dos objetos, levando em consideração outros fatores além das informações espaço temporais. Elementos de contexto permitem analisar o impacto

de certo contexto no comportamento dos objetos. Por ser uma área nova, poucos trabalhos utilizam trajetórias semânticas ou elementos de contexto para identificar e analisar comportamentos de trajetórias. A próxima seção irá exemplificar como contexto auxilia no enriquecimento semântico de comportamento de trajetórias.

3.2.3 Analisando Contexto em Trajetórias de Objetos Móveis

Por ser um conceito novo, trabalhos da literatura têm se focado na modelagem e construção de uma trajetória semântica, mas pouco na identificação de comportamentos de trajetórias utilizando os elementos de contexto. Baglioni et al. (2009) enriquecem trajetórias individuais com informações semânticas obtidas de ontologias, buscando inferir o objetivo do objeto com base na trajetória. Por exemplo, uma trajetória que sempre começa em um mesmo local (ex. casa) e em dias de semana para no mesmo local por certo período de tempo (ex. trabalho) é classificada como trajetória de um trabalhador. Por outro lado, uma trajetória que sai de um hotel e para em vários pontos turísticos é classificada como uma trajetória de turista. Apesar de usar elementos de contexto, o método dos autores não analisa o impacto do contexto no movimento das trajetórias. Por exemplo, o que difere o caminho percorrido por um indivíduo que vai para um ponto turístico e um indivíduo que vai trabalhar? Como um agente classificado como turista se comporta no trânsito em um dia de sol e em um dia chuvoso? Quais os melhores dias para se visitar um ponto turístico?

Para responder essas perguntas é necessário construir uma base de dados de trajetórias com elementos de contexto. Como tais elementos são difíceis de serem capturados automaticamente, um grupo de voluntários do projeto europeu SEEK (SEmantic Enrichment of trajectory Knowledge discovery), que o proponente desta tese fez parte, gerou uma base de dados de trajetórias semânticas na cidade de Pisa - Itália, seguindo o modelo CONSTAnT (BOGORNY et al., 2014). Tal base conta com 66 trajetórias de 7 indivíduos, totalizando 119864 pontos. As trajetórias foram anotadas manualmente quanto ao seu objetivo, clima, temperatura média do percurso, meio de transporte e local.

A Figura 15 ilustra uma trajetória semântica capturada por um dos voluntários. A partir dos elementos de contexto informados pelo usuário, é possível representar a trajetória a partir de seu contexto. Na figura, pode-se observar o objetivo de cada trecho da trajetória, assim como o meio de transporte, informações climáticas e local da trajetória.

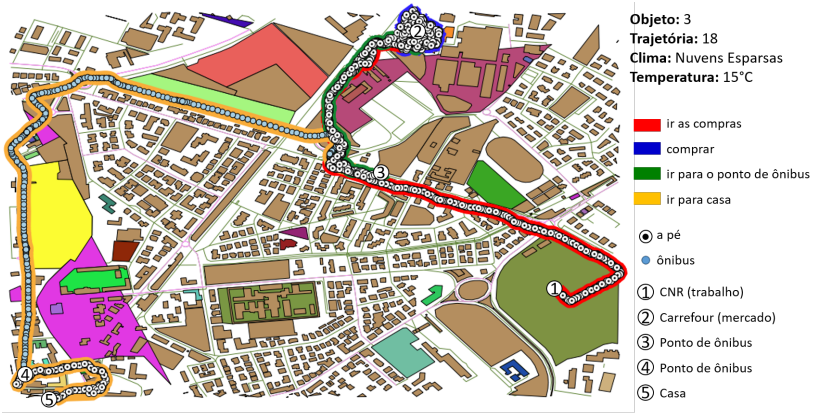


Figura 15 – Trajetória semântica de um indivíduo com sua atividade, clima, temperatura média, local e meio de transporte

Com uma base de dados de trajetórias semânticas, é possível extrair conhecimento baseado em elementos de contexto como, por exemplo, analisar o impacto do clima no movimento dos indivíduos. Observando os pontos dos objetos que andam a pé e que não estão parados, podemos verificar a velocidade média do objeto em cada temperatura e clima apresentado. As Tabelas 2 e 3 ilustram o impacto da temperatura e do clima na velocidade. É possível observar que, quanto mais frio, em geral, mais lentamente os indivíduos andam. Também observa-se que chuva ou climas de temperaturas mais baixas também afetam a velocidade do indivíduo, que anda mais rápido quando tem clima de céu aberto. Os elementos de contexto também podem ser analisados em conjunto para entender o comportamento. Por exemplo, a 4°C a velocidade média é mais elevada que, por exemplo a 15°C. Isso se deu pois nos dias que a temperatura estava em 4°C o clima era de sol ou nublado, sem chuvas. Logo, apesar da temperatura menor influenciar o agente a andar mais lentamente, o clima também tem impacto no movimento. Por outro lado, os dias de indivíduos se locomovendo com menores velocidades, com temperaturas de 2°C e 5°C, eram dias chuvosos.

Apesar destas conclusões serem óbvias para um ser humano, não é possível um sistema computadorizado fazer estas inferências sem elementos de contexto. Um sistema não tem o conceito do impacto da temperatura e do clima no movimento de um agente, mas com elementos de contexto é possível extrair esta relação. Por outro lado, é

Tabela 2 – Comparação da velocidade média do objeto com a temperatura média do ambiente

Velocidade média m/s	**Temperatura média C°**
1.22	2
1.60	3
2.26	4
1.13	5
1.64	6
1.85	7
1.69	8
1.93	9
1.96	10
2.08	11
4.91	12
2.00	15

Tabela 3 – Comparação da velocidade média do objeto com o clima do ambiente

Velocidade média m/s	**Clima**
0.63	Chuvoso
0.85	Nuvens esparsas
1.42	Chuva leve
1.63	Nublado
1.65	Tempo aberto
1.94	Noite aberta
2.08	Neblina
2.61	Ensolarado

impossível para um ser humano analisar uma quantidade massiva de dados para chegar nestas conclusões. A partir de um conjunto de trajetórias semânticas com elementos de contexto, é possível realizar uma série de análises sobre o movimento do objeto a partir do contexto que ele se encontra.

Uma trajetória gerada por GPS não possui informações suficientes para permitir interpretar o comportamento com elementos de contexto. É necessário o uso de fontes de dados externas para complementar e enriquecer a trajetória antes de analisar seu contexto. Entretanto, na robótica é possível extrair, além da trajetória, elementos de contexto do cenário utilizando sensores, informações do robô e do usuário e de um banco de dados associado a tarefas realizadas no passado.

3.3 ÁRVORE DE COMPORTAMENTO

Existem várias maneiras de representar a evolução de estados de um agente. A maneira mais conhecida e comum na robótica é a representação de estados através de um AFD (Autômato Finito Determinístico), ilustrado na Figura 16.

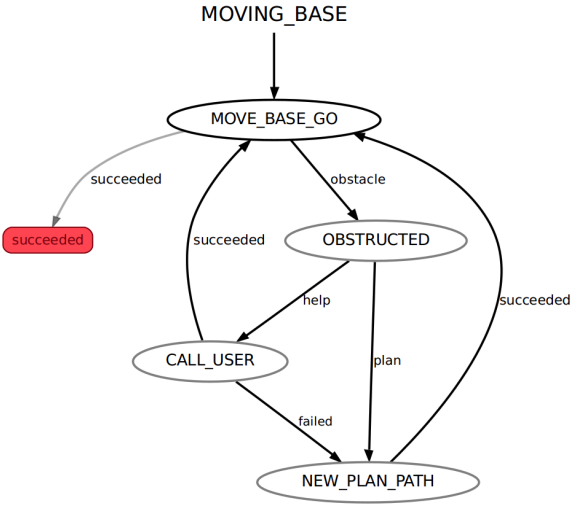


Figura 16 – Exemplo de um AFD para controlar o movimento do robô (FOUKARAKIS et al., 2014)

Nesta representação, cada possível estado do agente é represen-

tado por um nodo em um grafo orientado, onde a transição de estados é mapeada em um conjunto de funções de transição de um estado para outro. Esse conjunto de funções modelam eventos do AFD. Por exemplo, um robô no estado *nível de energia baixo* ficará neste estado até que ocorra o evento *recarregar energia*.

A Figura 16 ilustra os possíveis estados de um robô no comportamento de recarga de energia e quais ações são utilizadas para mudar de estado. Um grande problema com autômatos finitos é que quanto maior a quantidade de informações e estados possíveis, maior é a possibilidade de explosão de estados, uma vez que seria necessário modelar as funções de transição entre cada estados válidos.

Para representar estados mais complexos que modificam o comportamento do agente de acordo com o contexto inserido, surgiu o conceito de Árvore de Comportamento (ISLA, 2005).

Árvore de comportamento, ou *Behavior Tree* (BT), é um modelo matemático de representação do plano de execução baseado na estrutura de árvores, onde o estado atual do agente é obtido percorrendo os nodos da árvore onde certa condição ou ação é executada com sucesso. As primeiras aplicações de BT foram na área de jogos eletrônicos, modelando a inteligência artificial de personagens controlados pelo computador (ISLA, 2005). Neste modelo, cada funcionalidade do sistema é modelada com uma BT a partir de seus requisitos e, ao final, as diferentes BTs são fundidas, definindo o comportamento geral do sistema ou do agente.

Diferente de um AFD, onde, para se determinar como se chegou a um estado é necessário armazenar toda sequência de eventos ocorridos, uma BT permite identificar todo o fluxo de estados passados que resultaram no estado atual apenas percorrendo a árvore. Através da modularidade dos estados, a árvore de comportamento permite representar os vários estados possíveis de um agente de maneira simples e intuitiva, servindo de estrutura base para a tomada de decisão do agente.

Árvore é um tipo de estrutura de dados baseada em grafos, onde os nodos são conectados por vértices direcionados, formando uma estrutura hierárquica. A Figura 17 ilustra a estrutura geral de uma árvore. O nodo que não tem nodo-pai é chamado de raiz e representa o topo da árvore. Os nodos que não tem nodo-filho são chamados de folhas, e representam os possíveis finais da árvore. A execução de uma BT é iniciada a partir de um tique de disparo. O tique é um acionador que percorre a BT, ativando cada nodo de acordo com seu tipo e o resultado de sua execução. O tique é definido de acordo com a aplicação,

podendo ocorrer a uma certa frequência ou a um certo estímulo. O tique percorre a árvore seguindo uma ordem de execução da esquerda para a direita e de cima para baixo.

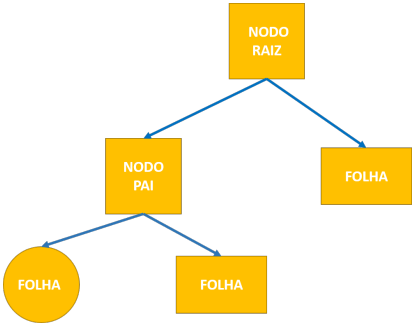


Figura 17 – Estrutura geral de uma árvore de comportamento

Em uma BT, cada nodo folha pode ser de dois tipos: condição ou ação, ambos ilustrados na Figura 18.

Tipo de Nodo	Sucesso	Falha	Executando
Condição	O resultado da condição do nodo é verdadeiro.	O resultado da condição do nodo é falso.	Não se aplica
Ação	A ação foi executada com sucesso.	A ação falhou por algum motive e não foi executada até o final.	A ação ainda está em execução e não está definido se será executada com sucesso ou se irá falhar.

Figura 18 – Possíveis resultados da execução de cada nodo folha em uma árvore de comportamento

Uma folha de condição, usualmente representada por uma elipse ou círculo, retorna sucesso caso a condição representada por ela seja satisfeita. Por exemplo, um nodo folha de condição $x > 5$ irá retornar sucesso para todos os valores de x superiores a 5, e retornará falha para valores de x iguais ou inferiores a 5. Uma folha do tipo ação,

usualmente representada por um retângulo, simboliza uma ação a ser executada pelo agente da BT. Por exemplo, a ação *vá para posição Y* irá retornar sucesso caso o agente consiga caminhar até a direção Y e retornará falha caso o agente falhe em chegar na posição Y. Um nodo de ação ainda pode retornar o valor *executando* caso não tenha completado sua tarefa.

Por exemplo, a ação *vá para a posição Y* retorna executando enquanto o agente está se deslocando até a posição Y. Esses comportamentos das execuções dos nodos folhas estão resumidos na Figura 18. A maior diferença entre esses dois tipos de nodos é que o nodo de condição não produz mudanças no ambiente, apenas consultando as condições enquanto um nodo de ação interage com o ambiente fazendo possíveis alterações nos estados internos e externos do agente. Os retornos de sucesso e falha dos nodos folhas de uma BT é que define o comportamento do agente.

Se um nodo folha pode ser do tipo condição ou ação, um nodo não-folha (nodo pai ou nodo raiz) pode ser do tipo sequência ou seletor. Um nodo do tipo sequência, representado por um quadrado com o símbolo $\rightarrow$, significa que todos seus nodos filhos serão executados em sequência, retornando falha caso um de seus nodos filhos retorne falha. Logo, para um nodo sequência retornar um valor de sucesso, todos seus nodos filhos devem retornar, sequencialmente, um valor de sucesso. A Figura 19 ilustra este processo.

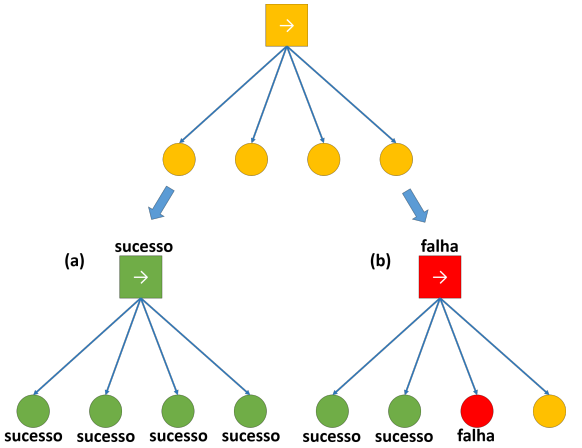


Figura 19 – Nodo de sequência onde (a) retorna sucesso e (b) retorna falha

Na Figura 19(a), todos os nodos filhos, executados em sequência, retornaram sucesso, fazendo com que o nodo também retornasse sucesso. Já na Figura 19(b), um dos nodos filhos retornou falha, o que interrompe o fluxo da sequência e faz com que o nodo retorne falha. O comportamento de um nodo sequência é equivalente ao operador *AND* de linguagens de programação.

Um nodo do tipo seletor, representado por um quadrado com o símbolo ?, ainda executa todos seus nodos filhos em sequência mas, neste caso, retorna sucesso para o primeiro de seus nodos filhos que retorne sucesso, sem executar os nodos filhos seguintes ao nodo que retornou sucesso. Caso um nodo filho retorne falha, o processo continua, executando seu próximo nodo filho, até que um retorne sucesso ou que todos retornem falha, significando neste último caso que o nodo seletor vai retornar falha também. A Figura 20 ilustra um nodo seletor e seus filhos. Quando um filho falha, o nodo continua testando os próximos nodos-filhos até um retornar sucesso (Figura 20(a)) ou até todos retornarem falha (Figura 20(b)). O comportamento de um nodo seletor é semelhante ao operador *EXCLUSIVE OR* de linguagens de programação.

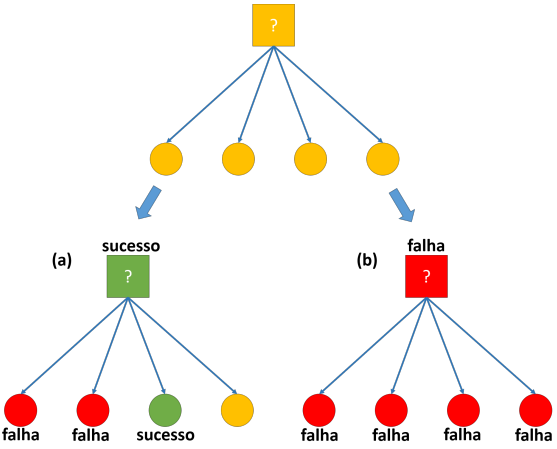


Figura 20 – Nodo seletor onde (a) retorna sucesso e (b) retorna falha

A Figura 21 ilustra um exemplo de uma árvore de comportamento com um nodo seletor, um nodo de sequência, uma condição (C1) e duas ações (A1 e A2). O nodo seletor vai primeiro verificar se o nodo de sequência irá retornar com sucesso e, caso contrário, irá realizar a ação A2. O nodo de sequência, por sua vez, irá verificar a

condição C1 e, caso retorne sucesso, irá executar a ação A1. Caso o nodo de sequência retorne sucesso, a ação A2 não será executada, uma vez que o nodo seletor retorna sucesso a partir do primeiro nodo-filho que retornar sucesso. Usualmente, BT são organizadas de acordo com a prioridade de cada comportamento, ou seja, quanto maior a prioridade de um comportamento, mais à esquerda da árvore vai estar o conjunto de nodos referentes àquele comportamento, uma vez que a execução dos nodos se dá da esquerda para direita e de cima para baixo.

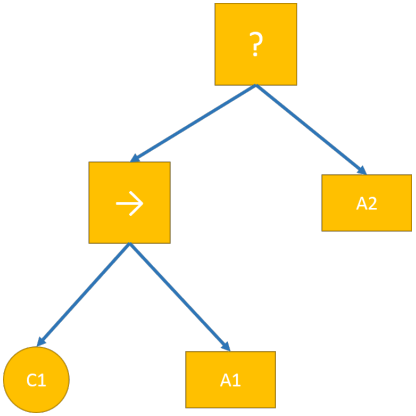


Figura 21 – Exemplo de árvore de comportamento

O trabalho de Bagnell et al. (2012) foi um dos primeiros a utilizar uma BT em robótica. O trabalho propõe uma arquitetura de comportamento para execução de tarefas de robôs chamada de BART (Behavior Architecture for Robotic Tasks). A tomada de decisão das tarefas do robô autônomo é realizada através de uma BT implementada como uma árvore de decisão binária, com apenas dois filhos. Além de nodo de sequência e seletor, o autor também implementa um nodo de paralelismo (onde ambos nodos filhos são disparados em conjunto, cada um em uma *thread*). Os nodos de BART suportam funcionalidades adicionais, chamadas de decoradores de nodos. Os decoradores permitem, por exemplo, adicionar uma funcionalidade de repetição a um comportamento, onde ele irá se repetir um certo número de vezes ou até ele retornar sucesso. O autor defende o uso de BT em robótica como meio de definir o comportamento do robô de acordo com o contexto do estado anterior, podendo reutilizar pedaços da BT em comportamentos diferentes.

Marzinotto et al. (2014) apresentam um framework de árvore de

comportamento para controle de robôs. Além dos nodos clássicos da BT, os autores definem o nodo paralelo (também descrito em (BAGNELL et al., 2012)) e decoradores. O trabalho também propõe uma extensão no conceito de nodo e de decorador. Originalmente, em uma BT, um nodo é acionado a cada tique da árvore, não guardando nenhuma referência ao tique anterior. Isso acarreta que, por exemplo, em um nodo de sequência, o último nodo filho só será acionado após cada nodo filho anterior já ter sido acionado pelo tique. Caso exista um tique muito pequeno, o último nodo filho pode nunca ser alcançado. A extensão proposta guarda o último nodo que retornou o estado de executando para que, em um próximo tique, a BT ignore os nodos já retornados e continue de onde parou. Essa funcionalidade é opcional para casos em que não é necessário garantir todos os nodos anteriores do tique. Já a extensão do decorador permite sincronizar sistemas multiagentes, controlando o estado de cada agente. Quando um número mínimo de agentes chegar no estado de sincronização, o algoritmo aciona o tique em *broadcast* para todas BT de todos os agentes que necessitam de sincronização. Esse framework foi implementado no framework ROS, o qual será detalhado na seção 3.7.

Colledanchise e Ogren (2014) demonstram como uma árvore de comportamento pode representar o comportamento de robôs e auxiliar na robustez e segurança de sistemas híbridos de controle na robótica. Através de uma BT, os autores mostram como o comportamento de um robô pode ser modelado através de nodos de ação, condição, seleção e sequência. A BT influencia no fluxo de execução e na estrutura do código final do robô.

A Figura 22 ilustra uma BT de um carro autônomo. A sequência de possíveis comportamentos do carro é expressa através do nodo raiz como um nodo seletor. A partir do início da execução, da esquerda para direita, o carro autônomo irá verificar se está em um estacionamento e, caso positivo, irá estacionar (comportamento de maior prioridade). Este comportamento de estacionar é modelado como um nodo de sequência com uma condição (*No estacionamento*) e uma ação (*Estacionar carro*) Caso não esteja em um estacionamento, ou seja, caso a condição *No estacionamento* retorne falsa, o robô irá verificar se está em uma intersecção (condição *Em intersecção*) e, caso positivo, irá escolher um caminho (ação *Escolher caminho*). O carro segue adiante com os diversos possíveis comportamentos de acordo com sua prioridade.

Colledanchise e Ogren (2014) definem formalmente diversos conceitos relacionados à árvore de comportamento. O conceito de BT é expresso pela **Definição** 2.

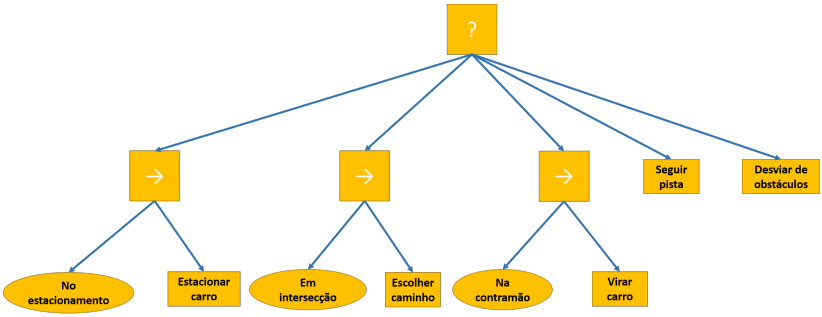


Figura 22 – Árvore de comportamento de um carro autônomo (figura adaptada de (COLLEDANCHISE; OGREN, 2014))

Definição 2. Árvore de Comportamento BT
Uma *Árvore de comportamento* é uma tripla $\tau_i = \{f_i, r_i, \Delta t\}$, onde $f_i : \mathbb{R}^n \to \mathbb{R}^n$ é a função do i-ésimo nodo da Árvore de comportamento, $r_i : \mathbb{R}^n \to \{R, S, F\}$ é uma função que retorna o estado do iésimo nodo, que pode ser executando (conjunto R), sucesso (conjunto S) ou falha (conjunto F) e Δt é o passo de execução.

Segundo os autores, a execução de uma BT pode ser expressa através de chamadas a funções recursivas de nodos da árvore, que retornam o possível estado do nodo (sucesso, falha ou executando) e a dinâmica de sistema de controle. O estado de retorno r_i de uma BT é dependente recursivamente dos estados de retornos de seus nodos filhos. A execução de uma BT é formalizada como uma equação diferencial ordinária com um intervalo de tempo Δt constante. Essa formalização é expressa através das equações (3.1) e (3.2).

$$x_{k+t}(t_{k+1}) = f_i(x_k(t_k)) \tag{3.1}$$

$$t_{k+1} = t_k + \Delta t \tag{3.2}$$

O resultado de cada função é expresso nos possíveis estados da BT (sucesso, falha ou executando). Entretanto, os possíveis resultados de uma árvore de comportamento de Colledanchise e Ogren (2014) dependem do tipo de BT. O autor classifica que uma BT é uma raiz quando não tem pai, um nodo quando tem uma BT pai e BT filhos, e uma folha quando não tem BT filhos. Uma BT folha pode ser do tipo condição ou ação.

Definição 3. Condição
Uma *Condição* é uma BT τ_i sem BT filhas onde $r_i : \mathbb{R}^n \rightarrow \{S, F\}$.

Definição 4. Ação
Uma *Ação* é uma BT τ_i sem BT filhas onde $r_i : \mathbb{R}^n \rightarrow \{R, S, F\}$.

Colledanchise e Ogren (2014) também definem os nodos de sequência e seletor como composição de sequência e composição de seleção. Essas composições conectam duas ou mais BT formalizando as operações. Uma composição de sequência é formalizada pela **Definição** 5.

Definição 5. Composição de Sequência de BT
Uma *Composição de Sequência* de BT τ_0 =Sequencia(τ_1, τ_2) é uma BT onde $\begin{cases} \text{se } x_k \in S_1 \rightarrow r_0(x_k) = r_2(x_k) \text{ e } f_0(x_k) = f_2(x_k) \\ \text{se } x_k \notin S_1 \rightarrow r_0(xk) = r_1(x_k) \text{ e } f_0(x_k) = f_1(x_k) \end{cases}$

Segundo a **Definição** 5, resultado de uma sequência τ_0 = Sequência (τ_1, τ_2) será igual ao resultado da BT τ_2 caso a BT τ_1 tenha retornado sucesso. Caso contrário, o resultado da sequência τ_0 será igual ao resultado da BT τ_1 (falha ou executando). Isso significa que τ_0 irá continuar executando τ_1 a cada intervalo de tempo Δt enquanto não retornar sucesso. A mesma lógica pode ser aplicada na composição de sequência de mais de duas BT. A definição de composição de seleção é formalizada na **Definição** 6.

Definição 6. Composição de Seleção de BT
Uma *Composição de Seleção* de BT τ_0 =Selecao(τ_1, τ_2) é uma BT onde $\begin{cases} \text{se } x_k \in F_1 \rightarrow r_0(x_k) = r_2(x_k) \text{ e } f_0(x_k) = f_2(x_k) \\ \text{se } x_k \notin F_1 \rightarrow r_0(x_k) = r_1(x_k) \text{ e } f_0(x_k) = f_1(x_k) \end{cases}$

A **Definição** 6 mostra que o resultado de uma seleção τ_0 = Seleção(τ_1,τ_2) será igual ao resultado da BT τ_2 caso a τ_1 tenha retornado falha. Caso contrário o resultado da sequência τ_0 será igual ao resultado da BT τ_1 (sucesso ou executando). Por sua vez, isso significa que τ_0 retorna sucesso ou executando a partir da primeira BT que retornar sucesso ou executando, passando para próxima BT filha somente se a anterior retornar falha. τ_0 só irá falhar caso todas BT filhas retornem falha. Novamente, a mesma lógica pode ser aplicada na composição de seleção de mais de duas BTs.

O comportamento de um robô móvel pode ser modelado através de uma árvore de comportamento. Cada sub-árvore representa um comportamento desejado para o robô analisando elementos de contexto

do ambiente. A facilidade de se incluir novos comportamentos e novos contextos complementa ainda mais a utilidade da estrutura de uma BT para navegação adaptativa. Analisando os elementos de contexto, o robô poderá adaptar sua navegação em tempo real ao comportamento mais adequado. A BT será utilizada nesta tese para modelar a tomada de decisão do robô, definindo seu comportamento. Serão utilizadas as definições apresentadas em (COLLEDANCHISE; OGREN, 2014) e o framework implementado em (MARZINOTTO et al., 2014).

3.4 MAPA SEMÂNTICO

Na navegação deliberativa e híbrida, um robô móvel necessita conhecer o mapa do ambiente onde ele irá executar a tarefa. Este mapa pode ser construído e atualizado em tempo real enquanto o robô conhece o ambiente através de técnicas SLAM (Simultaneously Localization and Mapping) (STACHNISS; LEONARD; THRUN, 2016; DAVISON et al., 2007), ou informado pelo sistema ou usuário quando se tratar de um ambiente conhecido. Técnicas clássicas de mapeamento da robótica móvel focam na construção de uma mapa métrico do ambiente, auxiliando o robô a se deslocar a partir de coordenadas espaciais que levem ao objetivo. Entretanto, tal mapa métrico não possui informações suficientes para tratar de problemas mais complexos como encontrar a saída mais próxima ou localizar um objeto em um dado ambiente. Para que um robô possa conseguir abstrair o mundo no nível de complexidade similar a de um ser humano, foi definido o conceito de mapa semântico.

Um mapa semântico é uma representação enriquecida do ambiente, armazenando não somente informações métricas e sim a semântica e significado de seus locais (KOSTAVELIS; GASTERATOS, 2015). A partir de um mapa semântico é possível incrementar as capacidades de navegação de um robô móvel, uma vez que agrega significado às coordenadas métricas, principalmente no que se refere ao uso de linguagem natural para dar ordens a um robô. Por exemplo, ao invés de passar coordenadas específicas referentes a um ponto no mapa, o uso de termo natural, conforme são definidos os ambientes (quarto, biblioteca, prédio público, supermercado etc), pode-se, a partir do mapa semântico, fazer a relação entre o termo usado para definir o ambiente e sua coordenada métrica. Um mapa semântico permite representar, além das coordenadas geométricas, conceitos sobre as entidades, objetos, funcionalidades ou eventos que estão localizados no espaço (NÜCHTER; HERTZBERG,

2008).

Usualmente, trabalhos da literatura organizam um mapa métrico como um arranjo geométrico e um mapa semântico como um mapa topológico (KOSTAVELIS; GASTERATOS, 2015). Um mapa topológico[1] é representado através de um grafo, onde cada nó representa uma entidade geométrica e cada aresta representa uma conexão de proximidade entre as entidades, associada com um custo de travessia. Um mapa topológico pode ter diversos níveis de granularidade espacial, representando desde o mapa por completo, quartos separados a até uma área do mapa.

A união de um mapa métrico com um mapa topológico é chamada de mapa híbrido (BUSCHKA, 2005). A parte topológica de um mapa híbrido apenas representa os ambientes como nós em diversos níveis de abstração diferente, sem considerar a semântica ou as informações referentes ao local.

As Figuras 23 e 24 ilustram um mesmo mapa nas duas representações: métrica e topológica. A forma métrica, Figura 23, expressa o ambiente com seus espaços livres e com obstáculos, auxiliando o robô a planejar uma trajetória livre de obstáculos e que consiga se locomover pelo ambiente.

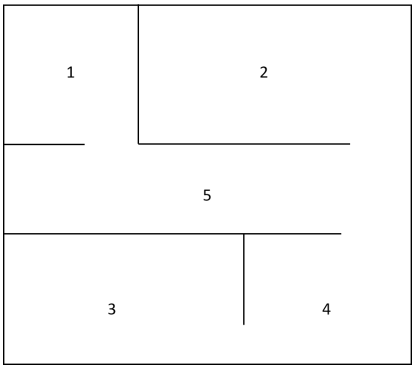


Figura 23 – Representação de um mapa métrico

Já a forma topológica, na Figura 24, expressa o ambiente identificado por um rótulo e as conexões entre os ambientes. Os rótulos podem se traduzir em quartos do mapa ou até mesmo em objetos localizados

[1] Apesar de existir outras terminologias na literatura com esta mesma definição, o termo mapa topológico expressa melhor o conceito trabalhado na área de robótica móvel

em um quarto. Enquanto o mapa métrico permite o robô se locomover para uma coordenada específica, o mapa topológico permite comandar o robô para ir até um nó, que deve ser traduzido na coordenada do local.

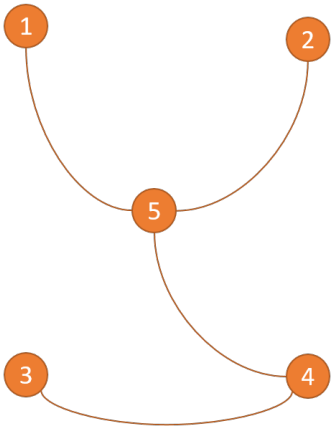


Figura 24 – Representação de um mapa topológico

A representação de um mapa semântico a partir de um mapa topológico se dá adicionando informações semânticas na descrição de cada nó. Por poder abstrair o mapa de maneira mais genérica que um mapa métrico, o mapa topológico fornece uma estrutura mais organizada e intuitiva para a adição destas informações.

Galindo et al. (2008) define mapa semântico como um mapa integrado com um domínio semântico, permitindo habilidades de dedução ao robô móvel para fazer inferências sobre o mundo antes mesmo de observá-lo. Os autores propõem um mapa semântico que integra uma hierarquia espacial dos objetos e lugares (chamada de S-Box) com uma hierarquia semântica de conceitos e relações destes lugares e objetos (chamada T-Box), utilizando este mapa semântico para melhorar o planejamento de tarefas do robô móvel. A Figura 25 ilustra a representação de um mapa semântico com as hierarquias S-Box e T-Box.

A hierarquia espacial, na parte S-Box da Figura 25, conecta objetos e locais próximos a partir de um mapa topológico dos elementos do ambiente enquanto a hierarquia semântica, na parte T-Box da Figura 25, identifica o significado destes objetos e locais, servindo de base para definir que objetos pertencem a um dado tipo de local. A parte S-Box da Figura 25 demonstra o nodo *area-1* conectado ao nodo *obj-2*. Ao

mesmo tempo, a parte T-Box da Figura 25 mostra que o nodo *area-1* é do tipo *Room* (quarto), enquanto o nodo *obj-2* é do tipo *Table* (mesa). Logo, ambas hierarquias (espacial e semântica) indicam que existe uma mesa no quarto. A hierarquia semântica também define o conceito de cada tipo de quarto. A parte T-Box da Figura 25 ilustra a definição de *livingroom* (sala de estar) como um ambiente que contém exatamente um sofá, pelo menos uma TV, pelo menos uma cômoda, não pode ter pia, etc. Essas definições juntamente com o significado de cada objeto permite o robô identificar o tipo de cômodo onde ele se encontra.

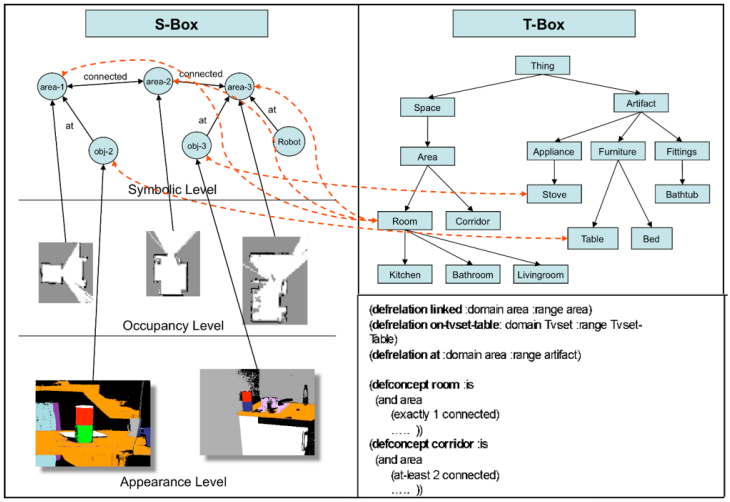


Figura 25 – Exemplo de mapa semântico de (GALINDO et al., 2008)

Uma prática comum na construção de mapas semânticos é o uso de câmeras e vídeos sobre o ambiente para a identificação do local. Meger et al. (2008) definiram uma abordagem onde um robô utiliza um banco de dados de imagens para identificar objetos no ambiente. Enquanto o robô constrói um mapa métrico do local, os objetos identificados são incluídos em um mapa semântico com sua localização e seu significado, permitindo identificar a localização de elementos cada vez mais específicos dentro de um ambiente, por exemplo, uma mesa em uma sala, um copo sobre uma mesa etc.

Ko, Yi e Suh (2012) definem um mapeamento semântico e uma navegação com marcadores visuais. A navegação do robô é realizada a partir de estratégias humanas como integração de caminhos, reconhecimento visual de local, reorientação e busca ativa. O algoritmo identifica

marcadores visuais no cenário como, por exemplo, certa imagem, e relaciona a distância daquele marcador com outros elementos do mapa a partir de uma descrição simbólica (perto, longe, em frente, à direita, etc). A partir da identificação destes marcadores, o mapa semântico é atualizado com a localização do marcador e suas relações espaciais com outros elementos. Essa marcação permite o robô se localizar no mapa ao visualizar uma determinada imagem.

O mapeamento semântico de Pronobis e Jensfelt (2012) utiliza ontologia para descrever as propriedades esperadas de cada localidade do mapa, utilizando sensores como lasers e câmeras do robô para identificar possíveis objetos e propriedades do ambiente. Cada localidade do mapa é caracterizada segundo seu formato, tamanho, aparência e objetos. A partir destas propriedades identificadas, o algoritmo faz inferência sobre a probabilidade do quarto atual ser de um tipo de quarto definido em uma ontologia. A Figura 26 ilustra um mapa semântico criado a partir do algoritmo proposto. A cor de cada nodo indica a probabilidade do tipo de cada quarto. Os corredores, de cor cinza, foram identificados apenas analisando o tamanho e o formato do ambiente, uma vez que na ontologia eles não têm objetos associados. Os demais ambientes foram identificados corretamente pelo algoritmo.

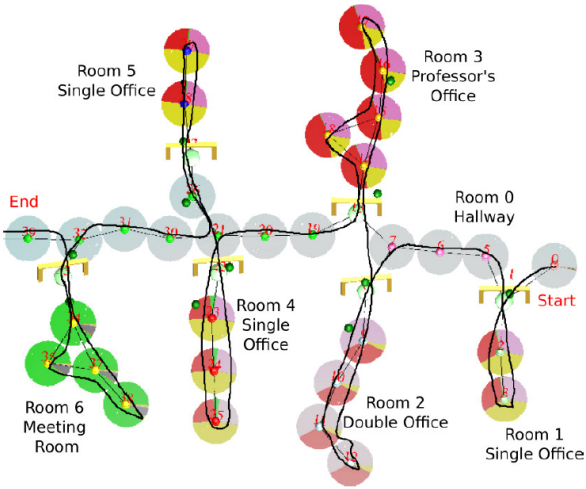


Figura 26 – Mapa semântico de (PRONOBIS; JENSFELT, 2012) com os locais identificados de acordo com a ontologia

Lim et al. (2013a) propõem uma ontologia para representação e

instanciação de mapa semântico para robôs móveis. O autor defende que robôs de serviço são feitos para realizar tarefas automaticamente ou semiautomaticamente, sendo necessário entender as relações semânticas entre os objetos, espaços e contextos para atender aos comandos humanos. Os autores utilizam a ontologia OWL (Web Ontology Language) (WELTY; MCGUINNESS, 2004) para representar o conhecimento do robô. A ontologia OWL permite criar descrições formais de classes com suas propriedades e relacionamentos. Como a ontologia OWL é um padrão para definição de ontologias na internet, esse modelo permite que o robô de Lim interprete ontologias de outras fontes de dado que sigam o mesmo padrão.

A definição de mapa semântico dos trabalhos da literatura se baseiam na adição de informações sobre o ambiente, entidades e aplicação a um mapa métrico de um robô. Entretanto, cada trabalho explora este conceito da maneira que lhe for mais conveniente. Uma iniciativa de Lang e Paulus (2014) propôs formalizar a definição de mapa semântico a partir da definição de mapa híbrido de Buschka (2005). A partir de um ambiente e do domínio da tarefa, os autores estendem a definição de mapa híbrido adicionando uma estrutura que representa o conhecimento sobre o domínio da tarefa, incluindo o relacionamento entre entidades, classes e atributos. A definição de Lang e Paulus porém não relaciona explicitamente estas informações com as localidades do mapa, além de não representar uma possível mudança de estado da informação.

Apesar de a grande maioria dos trabalhos que produzem um mapa semântico depender dos sensores de laser e câmeras do robô para identificar o ambiente, é possível utilizar, além destes dados, informações de contexto para caracterizar o estado de um mapa semântico. Dentro das pesquisas realizadas pelo autor desta tese, não se encontrou propostas de mapas semânticos para robôs móveis com possibilidade de mudança de estado a partir de informações de contexto. Uma informação de contexto pode caracterizar o estado de um ambiente do mapa semântico. Por exemplo, em uma aplicação de trânsito, o estado de um semáforo (aberto ou fechado) simboliza se o caminho entre os dois nós vai estar aberto ou fechado. Um engarrafamento pode aumentar o custo da aresta que liga dois nós. Um mapa semântico que leve em consideração informações de contexto pode alterar seu estado para representar não somente o significado do ambiente, mas seu estado atual e como isso impacta no planejamento e execução da trajetória do agente.

3.5 O PROBLEMA DA RECARGA AUTÔNOMA

O problema da recarga autônoma (Autonomous Recharging Problem - ARP) é um termo cunhado por Kannan et al. (2013) para descrever a capacidade de um robô móvel gerenciar sua bateria sem precisar de intervenção humana. O ARP pode ser dividido em duas categorias: como recarregar e quando recarregar. A primeira abordagem propõe modificação no *hardware* e *software* do robô e da aplicação para solucionar o processo de conectar o robô a uma fonte de energia. Já a segunda abordagem foca em *softwares* e algoritmos para decidir o melhor momento em que um robô deve ir recarregar, seja abortando a tarefa que está sendo executada ou planejando a recarga em termos da execução da tarefa e do conhecimento do ambiente.

Grande parte dos trabalhos da literatura focam em solucionar o ARP através da abordagem de como recarregar. Oh, Zelinsky e Taylor (2000), por exemplo, propõem um método para recarga automática baseado no pouso de aeronaves. A fonte de energia possui fitas refletoras para o robô localizá-las à distância, através de sensores lasers. Além das fitas refletoras, a fonte de energia possui o conector em padrão de grade diferenciado, que o robô identifica e alinha através do sensor laser, permitindo que o acoplamento e a recarga sejam feitas de forma autônoma.

O trabalho de Silverman et al. (2002) propõe modificações nos conectores da fonte de energia e do robô. O conector do robô possui um pino com molas laterais, possibilitando uma flexibilidade no manuseio do conector. Já do lado da fonte, o conector tem formato de cone, para reduzir os erros de alinhamento no procedimento de encaixe, redirecionando o conector flexível do robô no centro da fonte. Todo o procedimento é acionado quando o nível de bateria atinge um limite fixo, abortando qualquer tarefa em estado de execução.

Cassinis et al. (2005) se baseiam em estratégias de navegação do mundo real para conectar o robô com a fonte de energia. Assim como nas navegações antigas onde eram utilizados faróis para atracar no porto, o robô localiza e alinha com a fonte de energia através de duas luzes perpendiculares ao conector da fonte. Assim, caso o robô identifique a luz do lado esquerdo, então a fonte de energia está do lado direito. E caso identifique a luz do lado direito, a fonte está do lado esquerdo. Assim como no trabalho anterior, o robô procura a fonte de energia quando ele identifica que sua bateria está abaixo de um certo nível definido pelo usuário.

Os trabalhos citados funcionam para robôs que se movimentam

no plano. Para drones, que também podem se mover na vertical, Angrisani et al. (2015) propõem uma plataforma de pouso acoplada com uma fonte. Esta fonte recarrega o robô através de indução magnética, evitando a necessidade de conectores entre o robô e a fonte de energia. O drone pode pousar, recarregar e decolar novamente.

As abordagens de como recarregar um robô móvel são baseadas principalmente em modificações de hardware. Já as abordagens de identificar o melhor momento para acionar a tarefa de recarga são baseadas em softwares e algoritmos. Por exemplo, Wei et al. (2012) propõem um algoritmo de planejamento de tarefas chamado SLEEP (Staying-aLive and Energy-Efficient Path planning). A partir de uma lista de tarefas de deslocamento, o algoritmo SLEEP inclui uma tarefa de recarga no planejamento. Os autores representam o planejamento como um grafo, onde cada vértice e aresta tem um custo associado. O algoritmo procura no grafo qual o momento mais otimizado para incluir a tarefa de planejamento entre duas outras tarefas da aplicação, se baseando em um limite fixo do nível de bateria.

Kannan et al. (2013), além de cunhar o termo ARP, definiram uma solução baseada em medidas usando o nível de energia e sua conversão em distância. A proposta converte a energia da bateria do robô em distância que o robô pode percorrer. Essa estimativa é feita através de diversos experimentos de medições sobre o deslocamento do robô e o consumo da bateria. A partir desta estimativa, toda energia do robô é convertida em uma distância equivalente que o robô pode se deslocar. Assim, verificando a distância entre a posição atual do robô e a posição da tarefa, o algoritmo consegue identificar se o robô pode executar a tarefa com sucesso ou se deve recarregar a bateria e retornar a tarefa posteriormente.

Os trabalhos anteriores focam em um robô se deslocando até uma fonte de energia. Um alternativa proposta por Mathew, Smith e Waslander (2015) é ter robôs de recarga. O trabalho divide os robôs da aplicação em dois tipos: robôs trabalhadores e robôs de recarga. Cada robô trabalhador tem uma janela temporal onde deve ser recarregado antes de continuar executando a tarefa. O algoritmo então sincroniza esta janela temporal com um robô de recarga que esteja disponível, evitando que um robô aborte uma tarefa e procure uma fonte de energia.

Trabalhos existentes na literatura tentam resolver o problema da recarga autônoma de robôs móveis de diversos modos. Por serem focadas em mudanças de hardware, as abordagens que resolvem como o robô deve recarregar são usualmente exclusivas de um robô específico, requerendo alguma mudança física no robô e na fonte para pode

se adaptar à estratégia. Por outro lado, as abordagens de quando recarregar são focadas na criação de algoritmos, que podem então ser portados para diversos robôs, independente do hardware disponível.

Uma das abordagens que tem despertado interesse na literatura usa lógica difusa (lógica fuzzy) para o problema de navegação de robôs móveis (DRIANKOV; SAFFIOTTI, 2013). Uma contribuição deste trabalho é usar dessa técnica para melhorar o desempenho do robô no que diz respeito à decisão de quando efetuar a recarga da bateria.

3.6 LÓGICA FUZZY

Sistemas computadorizados recebem informações com valores absolutos como, por exemplo, distância de objetos, temperatura, velocidade ou resultados de equações. Esses valores, capturados por sensores ou computados por algoritmos, são usualmente utilizados em lógica booleana para operações lógicas e tomada de decisões. Uma limitação da lógica booleana é que seu resultado se limita em definir afirmações como verdadeira (usualmente representada por 1) ou falsa (usualmente representada por 0). Essa limitação dificulta sistemas computadorizados a fazerem inferências com incertezas ou abstração dos dados, tornando impossível a modelagem do comportamento de raciocínio humano. A lógica booleana não fornece respostas adequadas para perguntas como "o dia está quente?", "o copo está muito cheio?"ou "estou perto do meu destino?". Para responder perguntas com linguagem natural e conceitos poucos precisos é necessário utilizar a lógica fuzzy.

Lógica fuzzy é uma lógica que trata incertezas e aproximações ao invés de exatidão (NETO et al., 2006). Um sistema fuzzy permite que os dados tenham um grau de pertinência, sendo parcialmente verdadeiro. Diferente da lógica booleana que retorna 0 ou 1, a lógica fuzzy pode retornar qualquer valor contínuo entre 0 e 1, permitindo descrições em linguagem natural como frio, morno ou quente. Tratar valores com incertezas permite que sistemas computadorizados emule expressões de imprecisão, simulando o raciocínio humano.

A Figura 27 ilustra a diferença entre lógica booleana e lógica fuzzy em determinar se uma pessoa é alta.

Na lógica booleana (Figura 27a), deve-se definir um valor limite onde, acima dele, a pessoa é considerada alta, retornando 1, e abaixo é considerada baixa, retornando 0. No exemplo, uma pessoa é considerada alta caso tenha altura acima de 1,75m. Logo, uma pessoa medindo 1,74m não pode ser considerada alta. Já na lógica fuzzy (Figura 27 (b))

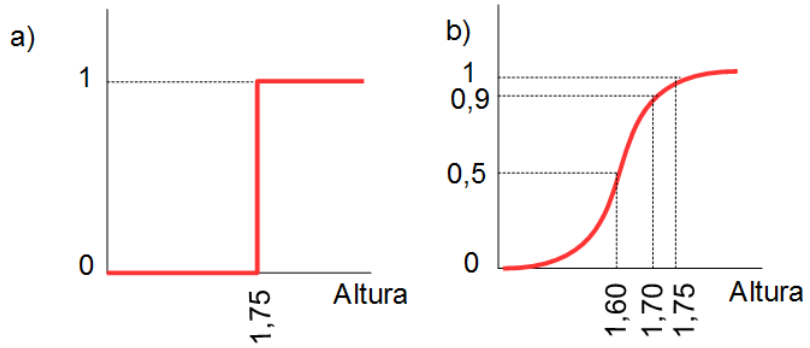


Figura 27 – Diferenças entre lógica booleana (a) e lógica fuzzy (b)

existe uma função que define quanto uma pessoa é considerada alta em um intervalo entre 0 e 1. Por exemplo, conforme mostrado na Figura 27 (b), uma pessoa medindo 1,60m é considerado alta com 0,5 de certeza. Já uma pessoa medindo 1,70 é alta com 0,9 de certeza.

O processo da lógica fuzzy, conforme mostrado na Figura 28, é dividido em três etapas: fuzzyficação, inferência e desfuzzyficação. A etapa de fuzzyficação é responsável por receber os dados de entradas e transformar em linguagem natural.

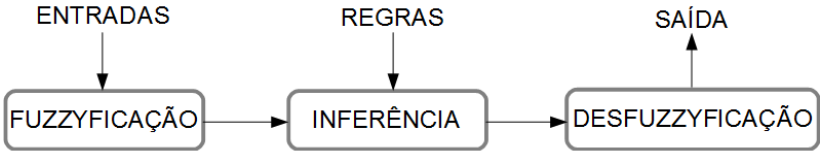


Figura 28 – Processo da lógica fuzzy

Como as informações são capturadas e fornecidas com valores absolutos, é necessário traduzi-las para a lógica fuzzy. Logo, um conjunto fuzzy A em um universo U é composto por um conjunto de pares $(\mu A(u), u)$ onde $\mu A(u)$ é uma função de pertinência que define o grau de pertinência da entrada u (NETO et al., 2006). A função de pertinência mapeia cada elemento de entrada em um termo de linguagem natural com um valor entre 0 e 1.

$$A = \{\mu A(u)/u\} = \{(\mu A(u), u) | u \in U\} \tag{3.3}$$

Apesar de existir várias funções de pertinência para representar

a distribuição do conjunto fuzzy na literatura, três são mais utilizadas: triangular, trapezoidal e gaussiana. A função de pertinência triangular, ilustrada na Figura 29 pode ser expressa a partir do equacionamento:

$$f(x,a,b,c) = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ \frac{c-x}{c-b}, & b < x < c \\ 0, & c \leq x \end{cases} \tag{3.4}$$

$$f(x,a,b,c) = max\left(min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right) \tag{3.5}$$

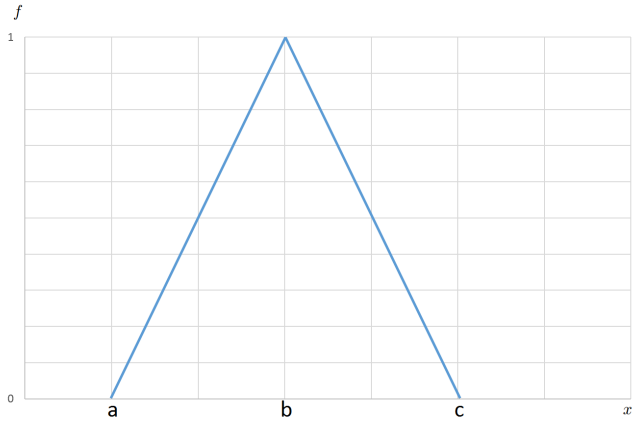


Figura 29 – Função de pertinência triangular

Já a função trapezoidal, ilustrada na Figura 30, tem as equações:

$$f(x,a,b,c,d) = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ 1, & b < x < c \\ \frac{d-x}{d-c}, & c \leq x \leq d \\ 0, & d < x \end{cases} \tag{3.6}$$

$$f(x,a,b,c,d) = max\left(min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right) \tag{3.7}$$

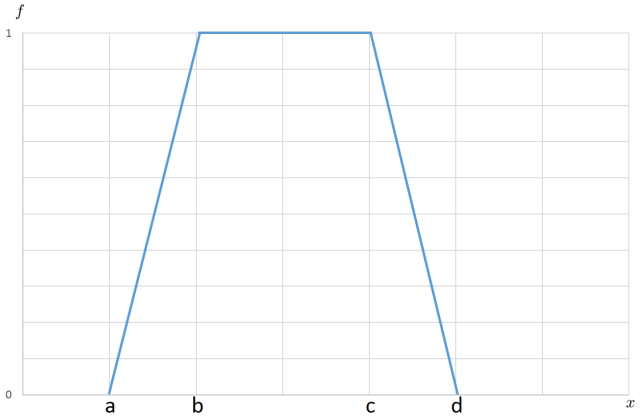


Figura 30 – Função de pertinência trapezoidal

Por fim, a função de pertinência gaussiana, ilustrada na Figura 31, tem o equacionamento:

$$f(x, \sigma, c) = \epsilon^{\frac{-(x-c)^2}{2\sigma^2}} \tag{3.8}$$

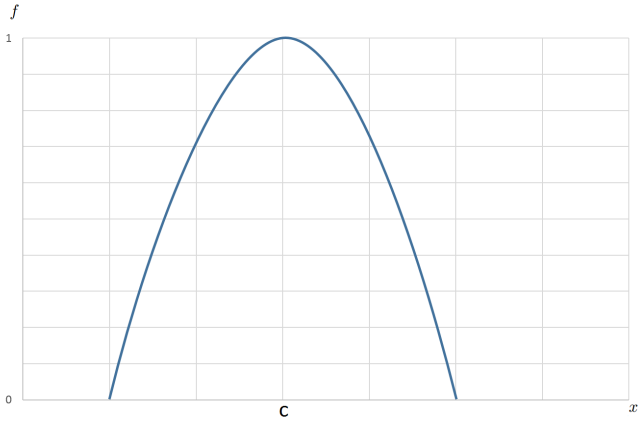


Figura 31 – Função de pertinência gaussiana

As funções de pertinência são utilizadas para transformar as entradas da lógica fuzzy em termos de linguagem natural. Para isso, é necessário que o usuário defina os termos e seus intervalos de valores.

Esses valores podem ser definidos por um especialista ou observando o comportamento do sistema.

Por exemplo, a Figura 32 ilustra um controle de temperatura implementado por lógica fuzzy. A lógica fuzzy permite indicar o grau de pertinência de uma temperatura em relação a cada termo (neste exemplo: muito frio, frio, quente e muito quente). O termo muito frio foi definido por uma função trapezoidal de intervalo -10° até 15°. Em 15° o valor de pertinência de muito frio é 0, pois é o final de seu intervalo. O termo frio foi definido por uma função triangular de intervalo 10° até 25°, onde a temperatura 17,5° tem grau de pertinência 1.

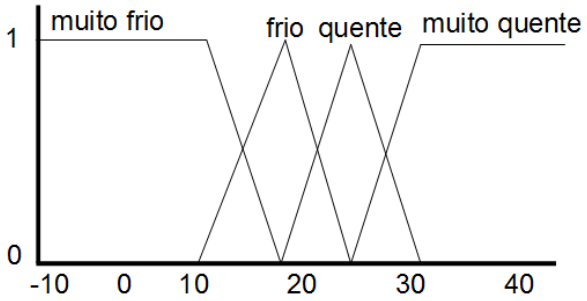


Figura 32 – Lógica fuzzy de controle de temperatura

Neste sistema, a partir de 10° o grau de pertinência do termo muito frio começa a diminuir e o grau de pertinência do termo frio começa a aumentar. Em um determinado momento, o grau de pertinência do termo frio ultrapassa o do termo muito frio, e a temperatura é definida como fria. Já algumas temperaturas podem ser consideradas muito fria e fria ao mesmo tempo, por exemplo, em 12° a temperatura é considerada muito fria com grau de pertinência de 0,7 e fria com grau de pertinência 0,3.

A próxima etapa do processo é a de inferência. Essa etapa recebe o resultado da fuzzyficação das entradas e define o resultado da lógica a partir de regras de inferência. Esse processo é definido como raciocínio fuzzy.

As regras fuzzy são derivadas da lógica proposicional e utilizam os operadores lógicos IF, AND, OR e $THEN$. As regras são separadas em duas partes: agregação e composição. A agregação é a descrição da condição, ou seja, o estado do sistema que se quer tratar, e vem sucedida do operador lógico IF. Os operadores AND e OR são modelados

respectivamente como interseção através da equação 3.9 e união através da equação 3.10.

$$AND \rightarrow \mu_{AandB} = min(\mu_A; \mu_B) \tag{3.9}$$

$$OR \rightarrow \mu_{AorB} = max(\mu_A; \mu_B) \tag{3.10}$$

Já a composição é a descrição do resultado da condição, ou seja, a ação a ser tomada pelo sistema quando encontrar o estado descrito na agregação, e vem sucedida do operador lógico $THEN$.

A etapa de inferência é responsável por definir a ação a ser tomada para que o sistema produza uma resposta adequada para tratar o estado atual.

Por exemplo, utilizando o sistema ilustrado na Figura 32 pode-se construir a seguinte regra de um chuveiro: ***se*** *a temperatura é* ***muito fria*** *então* ***esquente muito a água***. Se o resultado da etapa de fuzzyficação retornar que a temperatura da água é muito fria, com essa regra, a etapa de inferência irá produzir a resposta esquente muito a água. É possível construir regras com os termos das diversas entradas do sistema fuzzy. Como na etapa anterior, as regras podem ser definidas por um especialista ou pela observação do comportamento do sistema.

A última etapa é a defuzzyficação. Apesar da lógica fuzzy tentar simular o raciocínio humano em sistemas computadorizados, o programa ainda trabalha com valores reais. A etapa de defuzzyficação é responsável por pegar o resultado do processo de inferência e transformar as deduções em um valor real referente à resposta do sistema.

Assim como a entrada do sistema é modelada em termos linguísticos através de funções de pertinência, a saída do sistema é modelada da mesma maneira. Para cada possível termo de resposta (definido nas regras de inferência), é necessário definir uma função de pertinência referente à resposta. Existem vários cálculos para determinar o valor real da saída, sendo o cálculo do centroide o mais utilizado. A saída final R do sistema fuzzy é calculada pelo centro de gravidade do conjunto de respostas obtido pelas regras no processo de inferência. Seja n o número de quantização de saída, d_i o valor da variável de saída no intervalo de quantização i e $\mu_A(d_i)$ o grau de pertinência, temos o ponto de equilíbrio da região fuzzy calculando a média ponderada das regiões pela equação:

$$R \leftarrow \frac{\sum_{i=0}^{n} d_i \mu_A(d_i)}{\sum_{i=0}^{n} \mu_A(d_i)} \tag{3.11}$$

A lógica fuzzy permite que sistemas tomem decisões baseadas em raciocínio similar aos seres humanos. Em um sistema inteligente e sensível ao contexto, vários controladores podem utilizar informações de contexto como entrada para o sistema fuzzy, tomando decisões a partir de termos linguísticos conhecidos pelo usuário. Além disso, para alterar o comportamento de um sistema que utiliza lógica fuzzy só é necessário alterar as regras de inferência, podendo modificar as noções de quente, frio, perto, longe, rápido, devagar e etc de acordo com a aplicação e do ambiente.

3.7 ROS

Robot Operating System (ROS) (QUIGLEY et al., 2009) é um framework gratuito *open-source* e cooperativo de desenvolvimento de software para robôs. Ele é composto por uma coleção de ferramentas e bibliotecas que buscam abstrair a visão do *software*, separando os componentes do robô e do cenário em nodos e permitindo o reuso de código sem grandes mudanças.

ROS foi desenvolvido originalmente em 2007 no *Stanford Artificial Intelligence Laboratory*. Na época o framework se chamava *Switchyard* e foi desenvolvido para o controle do robô STAIR (QUIGLEY et al., 2007). No mesmo ano, a Garage (2009) contribuiu significativamente para o projeto, transformando a pequena iniciativa no framework ROS. Por ser de código livre e aberto, ROS foi desenvolvido em várias instituições simultaneamente, adaptando seu código para os diversos *hardwares* e robôs existentes. Em agosto de 2013, ROS passou a ser propriedade da *Open Source Robotics Foundation*, que gerencia seu desenvolvimento e expansão juntamente com contribuidores de todo o mundo.

Segundo Quigley et al. (2009), a arquitetura do ROS é baseada em 5 princípios: *peer-to-peer*, multi-linguagem, baseado em ferramenta, independência e *open-source*.

Peer-to-peer permite abstrair o robô, seus componentes e o cenário como um sistema operacional. Cada nodo, que representa um componente, sensor ou atuador do robô, é encarado como um processo em um sistema operacional conectados por uma rede *peer-to-peer*. A comunicação entre estes processos é feita através de troca de mensagens entre os processos. Cada processo tem seu identificador e pode comunicar ou receber mensagens através tópicos de mensagens. Múltiplos processos podem comunicar ou monitorar o mesmo tópico.

A multi-linguagem prevê a implementação e desenvolvimento do framework em diferentes linguagens de programação. Atualmente ROS tem como linguagens nativas C++, Python, Octave e LISP. Existe ainda uma biblioteca java para ROS (rosjava), permitindo produzir código para robôs na plataforma Android. Outra iniciativa implementa ROS como uma biblioteca javascript, permitindo a comunicação de páginas web com o robô (TORIS, 2015).

O framework baseado em ferramenta permite dividir um código extenso e complexo de controle de robôs em pequenos módulos que controlam e vistoriam cada componente separadamente. Dividindo as tarefas em múltiplos módulos pode diminuir a eficiência mas contribui para a estabilidade, reuso e gerenciamento do código.

A independência entre módulos permite o reuso de componentes desenvolvidos em diferentes robôs, cenários e códigos diferentes. O grupo encoraja que cada módulo desenvolvido seja o mais independente possível, sem depender de outros módulos diretamente. Assim, cada módulo pode ser reutilizado em outro contexto.

Por fim, ROS segue o princípio do *open-source*, sendo livre para uso e modificações em seu código. Isso incentivou a criação de diversos módulos e implementações para o ROS como, por exemplo, integração com o framework de programação de robôs *Player*, o simulador de robôs 2D *Stage*(RESEARCH, 2011) e o simulador de robôs 3D *Gazebo* (FOUNDATION, 2014a).

A arquitetura do ROS se baseia em troca de mensagens entre processos. Para que essa troca de mensagens seja possível, é necessário iniciar o servidor principal do ROS através do comando *roscore*. Roscore é um conjunto de nodos e processos do ROS que são pré-requisitos para o funcionamento do sistema. O roscore inicia dois serviços principais: o *Master* e o *Servidor de parâmetros*.

O ROS Master é um provedor de nomes e serviços para os demais nodos do ROS, ligando os nodos publicadores de mensagens (Publishers) e os nodos assinantes de mensagens (Subscribers) aos tópicos e serviços do ROS, permitindo a troca de mensagens entre os processos. A função principal do ROS Master é auxiliar cada nodo do ROS a encontrar um nodo específico.

A troca de mensagens entre nodos do ROS é feita através de publicação e assinatura de tópicos. Tópicos são canais de fluxo unidirecionais de mensagens entre nodos. Um nodo publicador manda uma mensagem para um tópico específico e um nodo assinante pode capturar mensagens enviadas a um tópico específico. Um tópico pode ter vários publicadores e assinantes, sem manter nenhum controle sobre

quem publica ou assina o tópico. Os tópicos do ROS utilizam a arquitetura TCP/IP e o protocolo de transporte UDP para a troca de mensagens.

A classe Publisher publica mensagens em tópicos no ROS e é instanciada pela classe NodeHandle, informando o tipo de mensagem e o nome do tópico onde ela será publicada. Já a classe Subscriber é a classe assinante, também instanciada pela classe NodeHandle, informando o nome do tópico que está assinando e a função que será invocada quando uma nova mensagem for publicada no tópico. Ambas classes Publisher e Subscriber têm opções para fila de mensagens, que pode ocorrer quando mensagens são publicadas ou recebidas mais rápido do que são consumidas e utilizadas. O ROS fornece tipos primitivos de mensagens (int, float, boolean) e permite a criação de tipos personalizados.

A Figura 33 ilustra o processo de publicação de tópicos na arquitetura do ROS. Uma câmera de vídeo que captura imagens pode ser representada na arquitetura ROS como um nodo publicador *Camera*. Como ele quer publicar as informações capturadas pelo dispositivo, o nodo comunica ao servidor Master que vai publicar mensagens no tópico *images* através da função *Advertise(images)* (Figura 33(a)). O servidor Master cria então um tópico *images* que é conectado ao nodo *Camera* (Figura 33(b)). Logo, toda vez que a câmera de vídeo capturar uma imagem e quiser publicar no framework, o nodo *Camera* irá transmitir a informação pelo tópico *images*.

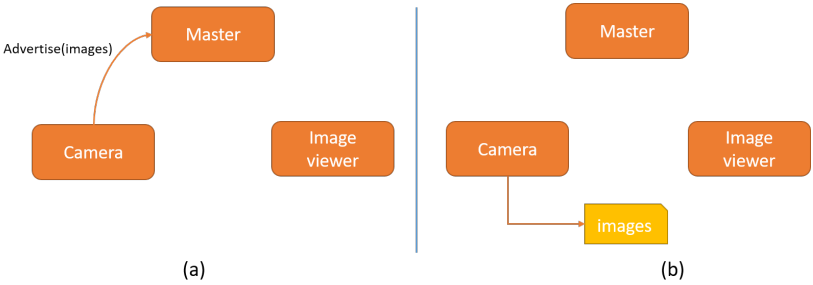


Figura 33 – Processo de registro de publicação no ROS

Já a Figura 34 ilustra o processo de assinatura de um tópico. O programa que irá consumir e interpretar as imagens é representado pelo nodo assinante *Image viewer*. Assim como no processo de publicação, o nodo comunica ao servidor *Master* a intenção de assinar ao tópico

images através da função *Subscribe(images)* (Figura 34(a)). Como já existe um tópico de nome *images*, o servidor responde ao nodo *Image viewer* com o endereço do tópico, conectando ao nodo *Image viewer* (Figura 34(b)). Assim, uma vez que a câmera de vídeo captura uma imagem e publica a informação no tópico *images*, o nodo *Image viewer* irá automaticamente ser notificado e receber esta informação.

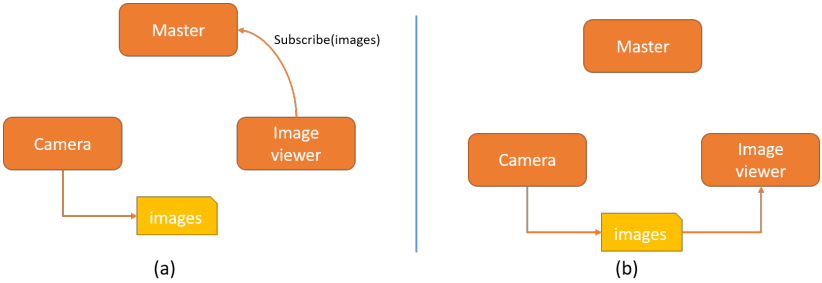


Figura 34 – Processo de registro de assinatura no ROS

O servidor de parâmetros é um provedor de armazenamento de parâmetros utilizados por nodos do ROS. A partir de um servidor de parâmetros, os nodos podem gravar e acessar dados globais em tempo de execução. Por não ser otimizado, é recomendado utilizar o servidor de parâmetros para variáveis estáticas como, por exemplo, parâmetros de configuração.

O framework ROS foi escolhido para a implementação da tese pelas características descritas acima. Por ser um framework modular, qualquer informação de contexto da aplicação pode ser capturada através da comunicação entre os tópicos de mensagens, facilitando a captura destas informações sem a necessidade de alterar toda implementação, bastando o tratamento da publicação e captura da mensagem. Além disso, é feito o uso do framework de BT implementado por Marzinotto et al. (2014), também na plataforma do ROS.

3.8 SÍNTESE DO CAPÍTULO

Este capítulo explorou diversos conceitos relacionados à proposta da tese. Tais conceitos, em conjunto, são essenciais para o entendimento do trabalho como um todo. O conceito de elemento de contexto auxilia a compreender como as informações capturadas caracterizam as entidades da aplicação. Tal conceito será complementado nos próximos

capítulos a partir de uma modelagem de contexto.

As definições de trajetória e de trajetória semântica são essenciais para compreender como as informações serão organizadas e representadas. Além disso, é importante diferenciar o conceito de trajetória do ponto de vista de mineração de dados e do ponto de vista de robótica, uma vez que as áreas utilizam o mesmo termo para definições diferentes.

A formalização da árvore de comportamento e a conceituação de mapa semântico serão utilizados para o controle da execução do robô, tanto na definição do comportamento, planejamento do caminho e representação do ambiente.

Paralelamente à proposta da tese, o problema de recarga autônoma foi identificado e tratado através de uma abordagem baseada em lógica fuzzy. Logo é necessário formalizar ambos conceitos.

Por fim, os experimentos foram implementados utilizando o framework ROS, que possui uma arquitetura diferenciada em comparação com outras abordagens de programação para robôs.

4 INCORPORANDO ELEMENTOS DE CONTEXTO NA NAVEGAÇÃO DE ROBÔS MÓVEIS

4.1 DEFINIÇÕES

A navegação inteligente baseada em contexto utiliza informações sobre o estado atual do cenário para definir o comportamento adequado do robô para realizar uma dada tarefa planejada. Para compreender o processo e a estrutura da navegação baseada em contexto é necessário definir uma série de conceitos que serão utilizados no restante desta tese. Alguns destes conceitos já foram explorados em seções anteriores mas a definição apresentada neste capítulo é a definição formal utilizada a partir deste ponto da tese.

4.1.1 Modelagem de Contexto

A primeira definição importante de ser formalizada é a de Contexto. Como explorado na seção 3.1, não existe uma definição universal de contexto. O próprio significado de contexto depende de onde o termo está inserido. Utilizando a descrição de Dey (2001) sobre contexto, que defende o termo como sendo qualquer informação que possa ser utilizada para caracterizar a situação de uma entidade, a **Definição 7** formaliza este conceito.

Definição 7. Contexto
C é um conjunto de contextos $C = \{C_1, C_2, ..., C_n\}$ onde um contexto C_i é um par (atributo, valor) que descreve uma característica ou estado de uma entidade. Uma entidade é descrita através de um conjunto de contextos C.

No âmbito desta tese, cada elemento presente na aplicação pode ter um conjunto de contextos C descrevendo suas características ou estado atual. Entretanto, para organizar as informações de contexto e categorizar suas características, foi proposta uma modelagem de contexto chamada WWW (WHAT, WHERE and WHEN), que analisa um contexto em três dimensões: seu significado (WHAT), sua localidade (WHERE) e sua linha de tempo (WHEN). Como já dito anteriormente, trabalhos da literatura que exploram sistemas sensíveis ao contexto usualmente já pré-definem os contextos considerados (WANG; HWANG; TING, 2011) ou classificam um contexto apenas pelo seu signi-

ficado (FOUKARAKIS et al., 2014). A proposta da tese propõe que uma informação de contexto pode ser qualquer informação relevante para a aplicação, independente de seu significado. A modelagem de contexto existe para auxiliar na organização e entendimento destes elementos a partir das dimensões definidas.

4.1.1.1 Dimensão WHAT

A primeira dimensão WHAT descreve o significado do contexto. Como um elemento de contexto pode alterar o comportamento do robô, portanto, compreender seu significado é essencial. Esta dimensão é dividida em três categorias: contexto do robô, contexto do usuário e contexto do ambiente:

- **Contexto do robô**: São informações sobre o robô e seu estado interno. Estas informações incluem sua posição, a tarefa atual, nível de energia, aceleração, identificador, direção, etc.
- **Contexto do usuário**: São informações sobre um usuário ou grupo de usuários que podem utilizar serviços fornecidos pelo robô. Estas informações incluem as preferências do usuário, sua localização, nome, relação com o robô, etc.
- **Contexto do ambiente**: São informações sobre o ambiente onde a aplicação e o robô estão inseridos. Estas informações incluem o mapa, obstáculos, estado do mapa, clima, localização das tarefas, localização de fontes de energia, eventos, etc.

A base do entendimento de um elemento do contexto é seu significado. A dimensão WHAT divide o significado em categorias compostas pelos atores envolvidos na aplicação (robô, usuário e ambiente). Por ser uma divisão comum e lógica, ela possui semelhanças, por exemplo, com a divisão em perfil dos parâmetros de decisão de Foukarakis et al. (2014). Entretanto, o modelo completo permite dividir e caracterizar um elemento de contexto em duas dimensões primordiais para aplicações em robótica móvel: o espaço e o tempo.

4.1.1.2 Dimensão WHERE

A dimensão WHERE descreve as propriedades espaciais do contexto, identificando onde no mapa o contexto pertence. Esta dimensão

auxilia no enriquecimento da trajetória semântica, identificando como o elemento de contexto se relaciona com a trajetória do robô. Ela é dividida em duas categorias: contexto local e contexto global:

- **Contexto local**: São contextos que pertencem a um lugar específico ou a um conjunto de lugares do mapa. O lugar pode ser um ambiente específico, uma posição, uma rua, ou uma condição qualquer. Por exemplo, o limite máximo de velocidade pertencente a uma rua específica do mapa.

- **Contexto global**: São contextos que descrevem ou atingem o mapa inteiro da aplicação, e não somente lugares específicos. Por exemplo, o clima do ambiente atinge normalmente o mapa inteiro.

A localidade de um contexto permite definir se o robô deve modificar seu comportamento apenas no local onde o contexto foi identificado ou deve adaptar ao contexto independente de sua localização.

4.1.1.3 Dimensão WHEN

A dimensão WHEN descreve a variável temporal do contexto, caracterizando sua linha do tempo. Saber qual a granularidade temporal de um contexto permite identificar até quando ele tem efeito sobre o comportamento de um robô. A dimensão é dividida em três categorias: contexto temporal, contexto periódico e contexto permanente:

- **Contexto temporal**: São contextos que têm um tempo de vida ou um instante de tempo específico para o valor atual. Isso significa que o valor deste elemento se altera em um curto período de tempo. Por exemplo, o nível de bateria de um robô.

- **Contexto periódico**: São contextos que têm um valor temporal mas que se repete em um período específico. Por exemplo, a luz de um semáforo muda de vermelho para verde a cada período de tempo.

- **Contexto permanente**: São contextos que têm um valor fixo pela duração da aplicação. Por exemplo, as dimensões de um robô, o nome de um usuário, etc.

A partir das três dimensões, é possível categorizar os elementos de contexto previstos para uma aplicação e analisar qual o possível

impacto do contexto no comportamento do robô. Essa análise inicial auxilia na definição dos comportamentos iniciais do robô, que podem sofrer alterações quando for analisar o histórico de execuções.

4.1.2 CONSTAnT-MR

O modelo CONSTAnT (BOGORNY et al., 2014), explorado na seção 3.2.2, foi criado para modelar uma trajetória semântica de uma pessoa, modelando seu comportamento, meio de transporte, ambiente, etc. Para utilizá-lo em aplicações de robótica móvel, é necessário fazer algumas adaptações no modelo, excluindo ou alterando alguns elementos e relações. Nesta tese, o modelo adaptado foi batizado de CONSTAnT-MR (CONceptual model of Semantic TrAjecTories for Mobile Robots), ilustrado na Figura 36. Para efeito de comparação, será reapresentada a Figura 13 que representa a estrutura original do CONSTAnT na Figura 35.

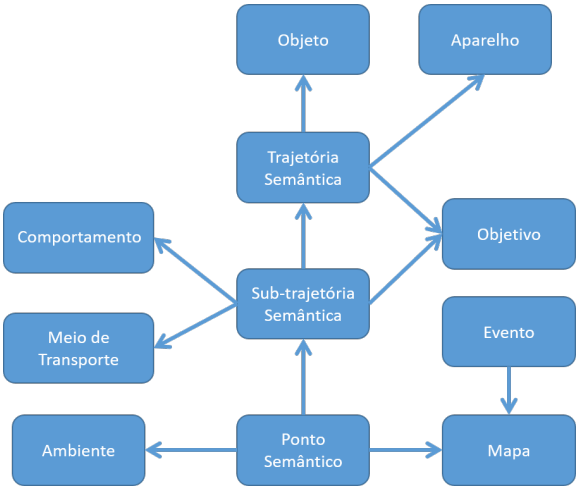


Figura 35 – Modelo CONSTAnT original (BOGORNY et al., 2014)

A Figura 35 ilustra o modelo CONSTAnT original e a Figura 36 ilustra o modelo CONSTAnT-MR para a robótica móvel. As diferenças principais são as seguintes:

- **Meio de transporte**: Essa dimensão foi removida pois o próprio robô é o um equipamento de transporte. O objetivo desta

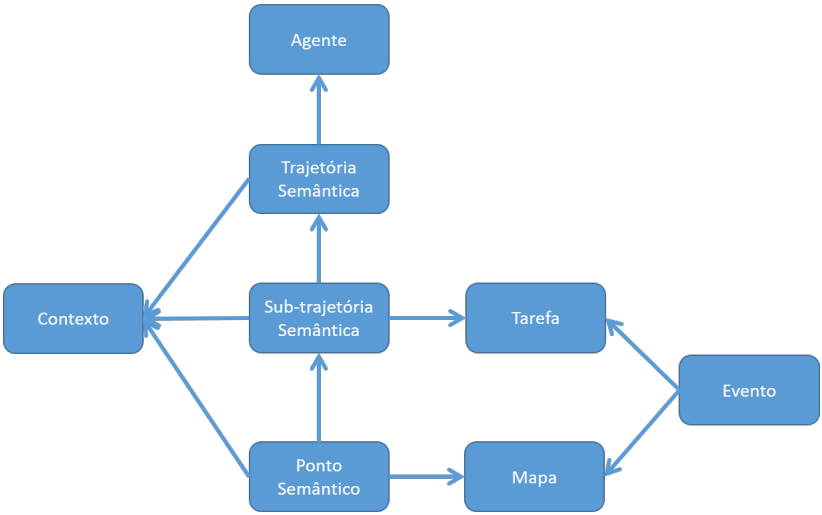


Figura 36 – Modelo CONSTAnT-MR para robótica móvel

entidade era identificar o modo como a pessoa estava se locomovendo (de carro, de ônibus, a pé), o que não se aplica na robótica móvel, onde o robô é seu próprio meio de locomoção.

- **Objetivo**: O movimento de um ser humano pode ter um objetivo claro ou genérico como ir ao supermercado ou visitar pontos turísticos. Na robótica móvel, o deslocamento de um robô está associado com a tarefa que ele deve executar. Logo, ao invés de uma entidade objetivo, se tem uma entidade tarefa.

- **Comportamento**: O comportamento do movimento de uma pessoa está associado ao modo que a pessoa se movimentou. Tal comportamento pode ser de desvio (ALVARES et al., 2011), perseguição (SIQUEIRA; BOGORNY, 2011), encontro (SANTOS et al., 2015), etc. Como o movimento do robô é definido pela tarefa e pelo algoritmo de planejamento de trajetória, a entidade não se aplica para o robô.

- **Ambiente**: Ao invés de descrever as informações de contexto através de uma entidade de ambiente relacionada apenas aos pontos, foi modelado uma entidade contexto que pode se relacionar com todos os níveis de granularidade da trajetória semântica (ponto, sub-trajetória e trajetória).

- **Evento**: Um evento pode estar localizado em um ou vários locais do mapa e também pode acionar uma nova tarefa. Por exemplo, um evento de mal funcionamento em uma máquina pode acionar a tarefa para o robô reparar a máquina defeituosa.

Além de definir o modelo CONSTAnT-MR, também é necessário formalizar cada entidade pertencente ao modelo. Como o modelo foi adaptado do CONSTAnT original, algumas de suas formalizações também foram adaptadas do modelo para poder ser aplicado na robótica móvel. Pode-se definir tarefa e evento da seguinte forma:

Definição 8. Tarefa
Uma tarefa (*task*) é uma sextupla $\langle x,\ y,\ d,\ c,\ q,\ A\rangle$ onde x e y são as coordenadas da localização da tarefa (em uma aplicação para robôs terrestres), d é o deadline (ou tempo máximo de execução) da tarefa, c é o custo para completar a tarefa, q é a prioridade da tarefa e A é o conjunto de ações que o agente deve executar para completar a tarefa.

Definição 9. Evento
Um evento E é uma quádrupla $\langle t, d, L, T\rangle$ que representa um evento ocorrido no ambiente onde t é o instante de tempo inicial que o evento ocorreu, d é a duração do evento, L é o conjunto de locais afetados pelo evento e T é o conjunto de tarefas geradas para tratar o evento. Um evento é considerado global quando L possui todos os locais possíveis da aplicação e local quando L possui um número reduzido de locais. O evento pode ser periódico, quando ocorre de tempos em tempos com horário marcado, ou imprevisível, quando pode ocorrer a qualquer momento.

Apesar desta tese inicialmente restringir as coordenadas da tarefa a um plano, é possível expandir esta definição utilizando a localização da tarefa em três dimensões. A tarefa contém um conjunto de ações que são traduzidas em comandos para o robô executar. Uma tarefa por si só representa um objetivo ou atividade a ser realizada. Porém, no momento que esta tarefa motiva o deslocamento do robô ela passa a ser considerada como um elemento de contexto, pois descreve o estado que o robô se encontra (executando a tarefa). O conjunto de ações de uma tarefa é modelado a partir de uma árvore de comportamento.

O mesmo raciocínio pode ser aplicado ao evento. O evento por si só é uma representação de um acontecimento, já ocorrido ou previsto. Entretanto, no momento que ocorre um evento que modifica o estado atual do cenário, este evento passa a ser interpretado como um

elemento de contexto. Um evento é um acontecimento pontual, logo, possui informações sobre o instante de tempo que ocorreu e sua duração. Um evento acontece em um instante de tempo e tem efeito em certo local do mapa. Como o evento ocorrido é um elemento de contexto, ele também pode ser caracterizado a partir do modelo WWW, por exemplo, o evento pode ser local ou global, dependendo da área afetada por ele. Um evento local pode ser, por exemplo, uma falha em uma máquina. O local de efeito do evento é a própria máquina e as tarefas associadas podem estar relacionadas com fazer o reparo. Já um evento global pode ser fim do expediente, onde todas áreas do mapa são afetadas e as tarefas associadas podem ser o processo de desativação dos agentes.

A partir dos elementos de contexto, é possível definir os elementos que compõem uma trajetória semântica. Discutido anteriormente na seção 3.2.2, uma trajetória semântica é uma trajetória bruta enriquecida com anotações. Para a navegação sensível ao contexto, as anotações são os elementos de contexto presentes na aplicação. Nesse cenário a definição de trajetória semântica, seguindo o modelo CONSTAnT-MR, divide-se em três elementos: a trajetória semântica, com a **Definição 10**, a subtrajetória semântica, com a **Definição 11** e o ponto semântico, com a **Definição 12**.

Definição 10. Trajetória Semântica
Uma *trajetória semântica* Ts é uma quádrupla $\langle oid, tid, Ss, C\rangle$ onde oid é o identificador do agente, tid é o identificador da trajetória, Ss é um conjunto de subtrajetórias semânticas e C é um conjunto de elementos de contexto.

A trajetória semântica é o elemento de nível hierárquico superior, sendo responsável por abrigar os demais elementos. Cada agente pode ter diversas trajetórias semânticas. A trajetória semântica é composta de subtrajetórias semânticas. A subtrajetória semântica é relacionada diretamente com a tarefa que está sendo executada no momento pelo agente. Logo, quando o agente recebe uma nova tarefa, uma nova subtrajetória é gerada, finalizando a anterior. A **Definição 11** formaliza o conceito de subtrajetória semântica.

Definição 11. Subtrajetória Semântica
Uma *subtrajetória semântica* Ss de Ts é uma n-tupla $\langle tid$, sid, P, $task$, $status$, $startTime$, $endTime$, $C\rangle$ onde tid é o identificador da trajetória semântica, sid é o identificador da subtrajetória, P é uma lista ordenada de pontos semânticos, $task$ é o objetivo da subtrajetória, $status$ é o estado final da tarefa, startTime é o tempo de início da sub-

trajetória, endTime é o tempo final da sub-trajetória e C é um conjunto de elementos de contexto.

Além da tarefa, a subtrajetória também tem informações sobre se a tarefa foi executada com sucesso ou se foi abortada e o tempo de início e fim da subtrajetória, calculado com o intervalo de tempo entre o primeiro e último ponto.

Cada subtrajetória semântica é associada com uma tarefa imposta ao agente. Uma subtrajetória semântica é composta de um conjunto de pontos semânticos formalizado na **Definição 12**.

Definição 12. Ponto Semântico
Um *ponto semântico ps* é uma n-tupla $\langle sid, pid, x, y, \theta, t, C \rangle$, onde *sid* é o identificador da sub-trajetória semântica, *pid* é o identificador do ponto, x e y são as coordenadas espaciais que representam um local em um plano, θ é a orientação angular do agente em relação ao eixo x, t é o tempo de captura do ponto e C é um conjunto de contextos informados, estimados ou medidos por sensores sobre o ambiente e/ou o agente.

A principal informação do ponto é a localização espacial do agente. Nesta tese, a localização é representada em um plano. O ponto representa o local onde o agente estava em certo instante de tempo, junto com sua orientação e conjunto de informações de contextos associadas a esse ponto. As informações de contexto dependem da aplicação e representam informações sobre o estado ou características de entidades relacionadas ao cenário, incluindo o próprio agente.

A Figura 37 ilustra um exemplo de dois pontos semânticos. Cada ponto da trajetória tem um conjunto de contexto associado a ele. Informações como a velocidade, aceleração ou energia fazem parte de elementos de contexto que descrevem o estado do agente naquele instante de tempo. É possível observar a evolução dos elementos de contexto durante a trajetória. Por exemplo, o ponto de id 30 tem 80% de energia no tempo de 35 segundos, entretanto, quando o agente continua a se deslocar pelo mapa, ele vai gastando energia e, no ponto de id 145 e tempo de 145 segundos, o nível de energia está em 15%. Outros elementos de contexto podem estar associados com cada ponto semântico da trajetória dependendo da aplicação. Um ponto semântico pode armazenar a temperatura, se está frio ou se está quente, umidade do ar, significado do local onde se encontra, entre outros.

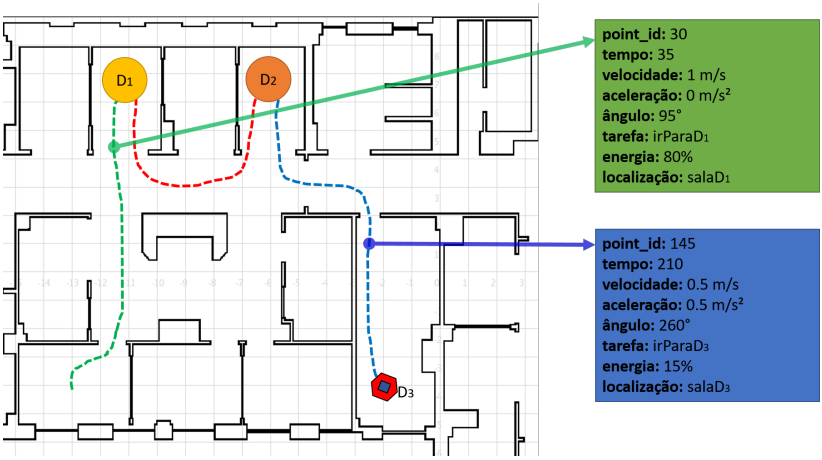


Figura 37 – Exemplo de pontos semânticos e seus atributos

4.1.3 Mapa Semântico e Contexto

Uma navegação híbrida de robôs móveis permite que, a partir de um mapa estático e fixo, o robô navegue por um ambiente, utilizando sensores para identificar e desviar de possíveis obstáculos encontrados pelo caminho e que não estavam presentes no mapa inicial. Esta navegação utiliza um mapa métrico para que o robô consiga se localizar e locomover no ambiente, conhecendo previamente a posição de marcos e locais definidos no ambiente de locomoção. Um mapa métrico permite associar um elemento de contexto a uma posição específica do mapa, entretanto este tipo de representação não é muito útil, uma vez que o elemento de contexto só irá interagir com o robô na posição específica. A incorporação de elementos de contexto em um mapa pode ser facilitada se, ao invés de se utilizar um mapa métrico, utilizar um mapa topológico do ambiente, em formato de grafo, criando um mapa semântico.

A representação topológica do mapa permite abstrair a dimensão espacial do mapa em níveis de granularidade diferente, representando um espaço geométrico através de um vértice. Por exemplo, a Figura 38 ilustra um mapa métrico dividido em 3 partes onde cada ambiente se comunica com seu adjacente e o ambiente B se comunica com A e C. Quaisquer espaços geométricos que se conectam com uma fronteira são ligados por uma aresta. Na Figura 38(a), os espaços A e B comparti-

lham uma fronteira, representado no mapa topológico na Figura 38(b) como o nodo A e o nodo B conectados com uma aresta.

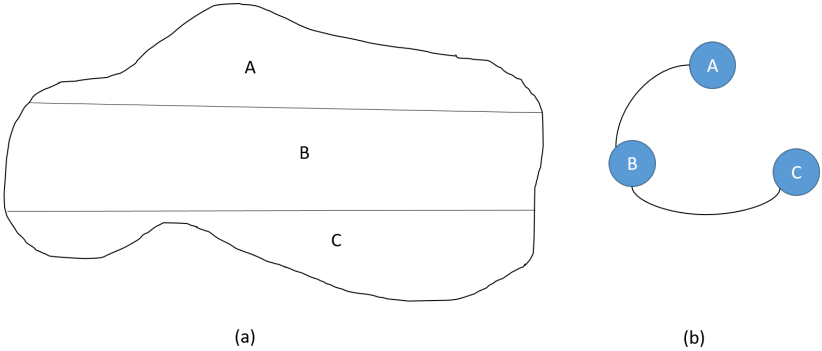


Figura 38 – Um mapa métrico abstraído em um mapa topológico dividido em 3 partes

Já a Figura 39 ilustra o mesmo mapa da Figura 38 mas agora dividido em 7 partes. De maneira similar, o espaço B_1 da Figura 39(a) faz fronteira com os espaços A_1, B_2, B_3 e C_1, sendo representado na Figura 39(b) pelas conexões entre os vértices. Diferentes níveis de granularidade na abstração de um mapa modifica o número de nodos e arestas. Os objetos presentes no mapa e que estão localizados em um espaço geométrico também são representados por um vértice, ligados por uma aresta com a região do mapa. Um objeto O_1 que está dentro do espaço A irá ser representado no mapa topológico como um vértice O_1 conectado por uma aresta ao nodo A.

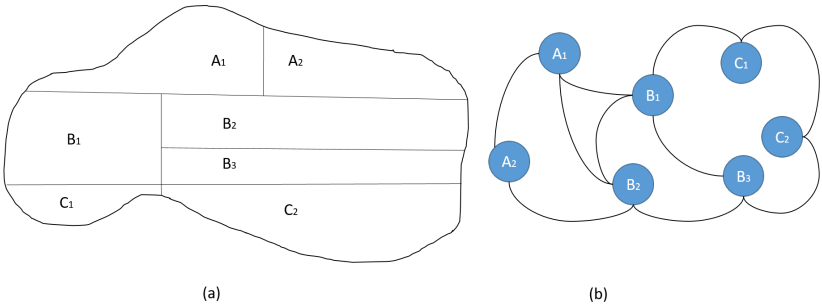


Figura 39 – Um mapa métrico abstraído em um mapa topológico dividido em 7 partes

Desta maneira, um mapa semântico é melhor representado através de um mapa topológico. Para melhor definir o mapa semântico e facilitar o entendimento de seu uso e atributo, é necessário definir o conceito de vértice semântico.

Definição 13. Vértice Semântico
Um *vértice semântico vs* é uma quíntupla $\langle vid, x, y, type, C \rangle$, onde *vid* é o identificador do vértice, x e y são as coordenadas planares do vértice em um plano, *type* é o tipo de entidade que o vértice representa (marcos do ambiente considerado) e C é um conjunto de contextos informados, estimados ou medidos por sensores sobre o ambiente e/ou o agente ou fornecidos pelo usuário.

Uma vez que o mapa topológico é uma representação em grafo de um mapa métrico, o vértice é uma representação deste mapa métrico. A quantidade e formato do espaço é dependente da granularidade de abstração do mapa, podendo representar desde centímetros quadrados até ambientes como um todo. As coordenadas x e y serão úteis na definição do melhor caminho para o planejamento de trajetória do robô. O tipo de entidade caracteriza o espaço de maneira simples, ajudando a identificar e a diferenciar os tipos de vértices. Por fim, o conjunto de contexto caracteriza o atual estado do vértice, que pode impactar no movimento do robô e, consequentemente, no custo de travessia entre os vértices. A partir de um vértice semântico pode-se definir o mapa semântico.

Definição 14. Mapa Semântico
Um *mapa semântico* Ms é um grafo $\langle Vs, A \rangle$, onde Vs é um conjunto de vértices semânticos e A é um conjunto de arestas que conectam vs_i e vs_j onde vs_i e $vs_j \in$Vs.

Diferentemente de Lang e Paulus (2014), que relacionam informações sobre a aplicação ao mapa semântico como um todo, a Definição 14 permite relacionar um conjunto de elementos de contexto a cada vértice do mapa semântico, descrevendo cada ambiente do mapa através deste conjunto. Esta definição está mais próxima do conceito de mapa semântico de Nüchter e Hertzberg (2008) e Kostavelis e Gasteratos (2015), que defendem que um mapa semântico representa informações e conceitos auxiliares mas relacionados diretamente a uma localidade no espaço.

Para navegar em um mapa semântico, é preciso de um algoritmo de busca de grafos. Existem várias abordagens clássicas na literatura para busca em um grafo como o algoritmo de Dijkstra (DIJKSTRA, 1959), A* (HART; NILSSON; RAPHAEL, 1968), D* (STENTZ; MELLON,

1993), Lifelong Planning A* (KOENIG; LIKHACHEV; FURCY, 2004), etc. Nesta tese utilizamos uma implementação do algoritmo A* para a navegação do mapa semântico.

A principal vantagem do A* sobre o algoritmo de Dijkstra é que, enquanto o segundo deve verificar cada possível caminho a partir do custo de ir de um vértice ao outro, ou seja, percorrer uma aresta, o primeiro utiliza estratégia gulosa para definir o melhor caminho através de uma função heurística. Enquanto Dijkstra considera somente o custo do caminho até o vértice atingido, A* incrementa uma função heurística ao custo do caminho, definindo a melhor alternativa, segundo a equação 4.1.

$$f(n) = g(n) + h(n) \tag{4.1}$$

O custo para chegar no vértice n é igual ao custo acumulado $g(n)$, que representa a soma do custo das arestas do caminho inicial até n, mais o resultado de uma função heurística $h(n)$.

Logo, uma boa função heurística permite encontrar o caminho de menor custo entre dois vértices de uma maneira mais rápida que Dijkstra. Na robótica, uma prática comum é implementar a função heurística como a distância euclideana entre os dois nodos.

$$h(n) = \sqrt{(vs_n.x - vs_f.x)^2 + (vs_n.y - vs_f.y)^2} \tag{4.2}$$

A navegação do mapa semântico nesta tese utiliza como função heurística a distância euclidiana entre dois vértices, apresentada na equação 4.2, onde vs_n é o vértice semântico atual de $h(n)$ e vs_f é o vértice semântico do destino final.

Além da função heurística, é preciso definir a função de custo das arestas. O mapa semântico deste trabalho é implementado com o grafo de arestas ponderadas, ou seja, cada aresta tem um peso diferente dependente dos vértices que são conectados. Existem três custos para o deslocamento de um objeto: a distância, o tempo e o gasto de energia. A distância refere-se ao espaço físico que se deve deslocar para sair do ponto de origem até o destino. O tempo está relacionado à duração deste deslocamento. Já o gasto de energia está relacionado a quanto o robô consome da bateria ao se deslocar pelo mapa.

Mei et al. (2005) estimam o gasto de energia com a potência do movimento com a equação (4.3):

$$p_m(m, v, a) = p_l + m(a + g\mu)v \tag{4.3}$$

Nesta equação p_m é a potência da locomoção, m é a massa, μ é o coeficiente de atrito do conto do robô com o solo, supostamente constante para um mesmo tipo de solo, v é a velocidade, a é a aceleração, g a aceleração da gravidade e p_l é a perda de transformação da energia.

Considerando uma aceleração nula, ou seja, uma velocidade constante, o gasto de energia é diretamente proporcional à velocidade que o robô se locomove. Esta velocidade pode sofrer influência de fatores externos como obstáculos, inclinação de piso, tipo de chão, clima, limite máximo de velocidade, etc. Logo, o gasto de energia do robô também pode variar de acordo com estes fatores externos.

Para o mapa semântico deste trabalho, o custo inicial de cada aresta é calculado pela previsão de tempo necessário para deslocar e pela estimativa de gasto de energia, a partir da equação (4.4):

$$a_{i,j} = \alpha \frac{\sqrt{(vs_i.x - vs_j.x)^2 + (vs_i.y - vs_j.y)^2}}{v_{max}} + \beta p_m \tag{4.4}$$

Na equação acima $a_{i,j}$ é a aresta que liga o vértice vs_i com o vértice vs_j, v_{max} é a velocidade máxima do robô, p_m é a potência de locomoção do robô e α e β são valores entre 0 e 1 onde $\alpha + \beta = 1$. Os valores de α e β são definidos pelo usuário, indicando se a preferência do robô deve ser pelo caminho mais rápido, dando prioridade à α, ou o caminho menos custoso, dando prioridade à β. É possível encontrar valores ótimos para α e β através de experimentos, entretanto este problema não será abordado na tese, se limitando apenas a definir valores fixos para os pesos.

Como os vértices representam um espaço físico, a distância entre eles sempre será constante. Já o tempo e o gasto de energia pode variar de acordo com a velocidade do deslocamento ou com o número de obstáculos presente. Isso permite alterar de maneira dinâmica o peso das arestas de acordo com os elementos de contextos observados em cada vértice. Este processo será explorado posteriormente.

A partir destas definições é possível definir e estruturar uma navegação inteligente baseada em contexto, explorada na próxima seção.

4.2 ESTRUTURA DO FRAMEWORK

Uma vez que elementos de contexto descrevem o ambiente, o robô e os usuários da aplicação, suas informações podem auxiliar a navegação inteligente em diversos aspectos. Esta tese explora a in-

corporação de elementos de contexto na tomada de decisão, estado do mapa semântico e planejamento de trajetória. Além disso, todas as informações são estruturadas em uma trajetória semântica e armazenadas em um banco de dados, servindo para a etapa de refinamento dos processos anteriores baseado no histórico de execução. Nesta seção será detalhado como cada passo destes processos funciona através do framework ConBINaMoR (Context-Based Inteligent Navigation for Mobile Robots).

A Figura 40 ilustra o framework dividido em processos. Apesar de, na prática, ser possível executar paralelamente alguns destes processos, será apresentado como um ciclo de passos para facilitar o entendimento.

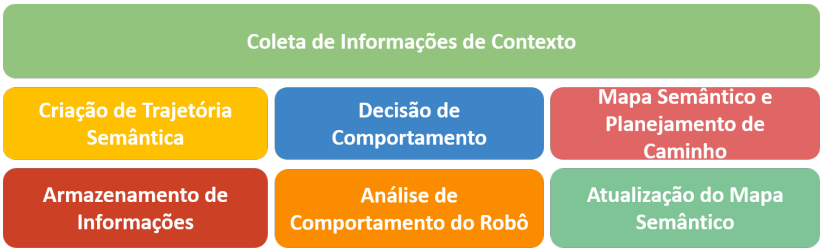


Figura 40 – Sequência de processos do framework ConBINaMoR

Inicialmente é necessário capturar o contexto local através dos sensores do robô e do ambiente. A partir destas informações de contexto, é decidido o comportamento adequado do robô, que irá executar uma tarefa. O caminho da posição atual até a posição da tarefa é auxiliado por um mapa semântico e, durante o deslocamento do robô, a trajetória semântica é gerada a partir das informações coletadas. Por fim, as trajetórias geradas são armazenadas no banco de dados e utilizadas como histórico de execução das tarefas, ajudando no processo de refinamento da árvore de comportamento na análise do comportamento do robô e na atualização do mapa semântico.

Apesar de reconhecer propostas e iniciativas de modelagens de comportamento do robô automatizadas através de algoritmos de inteligência artificial, esta tese está focada na definição manual de cada processo. Isso permite que um especialista defina uma proposta de comportamento do robô e, analisando os resultados, refine a modelagem até chegar ao desempenho desejado. O processo proposto pelo framework poderia também ser automatizado com técnicas de aprendizagem de máquina.

4.2.1 Coleta de Informações de Contexto

A primeira etapa da metodologia de contexto é definir quais são as informações necessárias para sua aplicação. Essas informações serão essenciais para o robô definir seu comportamento durante a execução. Alguns elementos podem estar sempre presentes como, por exemplo, a localização do robô, direção, velocidade e nível de bateria. Outros podem ser dependentes da aplicação, de sensores ou de informações do próprio usuário. Uma aplicação de carros autônomos pode ter como contexto o estado de um semáforo, o clima, a velocidade máxima da via, localização de faixa de pedestre, etc. Já uma aplicação carga e descarga pode ter elementos como local de carga e descarga, período de funcionamento, evento de chegada de carga, armazenamento, questões de logística, etc. A identificação dos elementos de contexto é parte de todo o processo, podendo ser adicionados novos contextos à medida que a aplicação aumenta o foco.

Uma vez identificados os contextos da aplicação (ou parte deles), cada elemento de contexto pode ser descrito segundo as dimensões WWW (What, Where and When), apresentadas na seção 4.1.1. Esta descrição auxilia o responsável por modelar o comportamento do robô a entender os elementos de contexto envolvidos na aplicação e como podem influenciar a execução e a aplicação.

Os elementos de contexto podem ser capturados através de sensores do robô, sensores do ambiente, informações do mapa semântico, informações fornecidas pelo usuário ou qualquer outra fonte de dados disponível. Para todas estas fontes, o framework ROS fornece ferramentas de suporte através das classes *Publisher* e *Subscriber*, que utilizam tópicos identificados para a troca de mensagens. Cada elemento de contexto deve ter um tópico específico para publicar informações relativas ao elemento. Por exemplo, para o estado de energia da bateria do robô, pode-se criar um tópico *energyLevel*. Para cada contexto capturado, um *Publisher* irá publicar seu valor no tópico correspondente, enquanto todos os processos que utilizam essa informação terá um *Subscriber* para capturá-la.

Os processos de publicação e assinatura de tópicos são assíncronos, ou seja, são executados à medida que mensagens são enviadas e recebidas. Uma vez que o contexto do cenário será composto de vários sensores, com tempos diferentes de envio e recebimento, a sincronização dos dados é feita abstraindo o instante de tempo de captura do contexto na casa de segundos. Como a trajetória do robô é expressa em geral em segundos, a abstração do contexto também é feita na mesma

unidade para permitir a compatibilização dos dados.

O algoritmo 4.1 é um exemplo de nodo assinante do ROS. Este nodo assina o tópico *energyLevel*, tópico hipotético em que o robô publica informações sobre seu nível de energia. Assim que uma mensagem chegar nesse tópico, a *thread* do nodo chama automaticamente a função *energyCheck*. A função *energyCheck* verifica o nível de energia do robô e, caso esteja abaixo do valor limite definido por *fixedThresholdValue*, muda o atributo *criticalLevel* para verdadeiro. O robô pode então definir um novo comportamento baseado na nova informação de que seu nível de energia está em nível crítico.

Algoritmo 4.1 – Código de um Subscriber (assinante) de mensagens

```
1  #include "ros/ros.h"
2  #include "std_msgs/String.h"
3
4  double fixedThresholdValue = 10;
5
6  void energyCheck(const std_msgs::String::ConstPtr& msg){
7    if(msg->data < fixedThresholdValue)
8                  criticalLevel = true;
9                  ROS_INFO("Energy at critical level");
10   }
11 }
12
13 int main (int argc, char **argv){
14
15    ros::init(argc, argv, "listener");
16    ros::NodeHandle n;
17    ros::Subscriber sub;
18    sub = n.subscribe("energyLevel", 1000, energyCheck);
19    ros::spin();
20    return 0;
21
22 }
```

O usuário também pode informar manualmente contextos específicos para o algoritmo. A arquitetura do ROS fornece ferramentas para a publicação de mensagens em um tópico específico por comando de console, sendo possível então o usuário interagir com o sistema em tempo real. Através da função *rostopic echo %nome do tópico %formato %mensagem*, o usuário informa o tópico em que se deseja propagar a informação, formato do dado e a informação desejada.

Apesar deste trabalho utilizar o ROS para coleta de informações e troca de mensagens, qualquer estrutura que permita que uma informação seja transmitida para o robô ou para o algoritmo pode ser utilizada como coleta de contexto. Os procedimentos descritos neste capítulo refletem a implementação desta tese.

Os elementos de contextos definidos, caracterizados e coletados são utilizados nos próximos processos do framework.

4.2.2 Decisão de Comportamento

O processo de decisão de comportamento monitora os elementos de contexto que descrevem o cenário para definir o comportamento adequado do robô. O processo de decisão de comportamento é implementado a partir de uma árvore de comportamento, definida na seção 3.3, utilizando a biblioteca *behavior_tree* (MARZINOTTO et al., 2014), implementada no ROS (FOUNDATION, 2014b).

O comportamento do robô pode ser definido em dois grupos: comportamento da tarefa e comportamento do contexto. O comportamento da tarefa se refere as ações que o robô deve executar para concluir suas tarefas. Nesta etapa, os elementos de contextos são abstraídos, focando apenas nas ações propostas. Isso permite modelar inicialmente o comportamento do robô baseando-se nas tarefas da aplicação. As ações são modeladas baseando-se na definição de árvore de comportamento, explorada na seção 3.3, utilizando os nodos de sequência, seleção, condição e ação. A Figura 41 ilustra uma possível árvore de comportamento de uma tarefa com as ações A1, A2 e A3. Esta árvore é construída focando apenas na execução da tarefa.

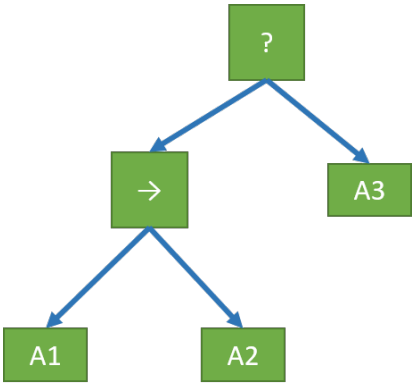


Figura 41 – Exemplo de uma árvore de comportamento para tratar uma tarefa

Após definir os comportamentos necessários para tratar as tarefas, é necessário definir as árvores de comportamento para tratar os elementos de contexto. Cada árvore pode tratar um ou mais elementos de contexto, variando de acordo com as dimensões definidas no processo anterior. Importante notar que os contextos devem ser tratados

de acordo com ações possíveis de serem executadas pelo robô ou outro agente do ambiente.

A Figura 42 ilustra uma possível árvore de comportamento para tratar um elemento de contexto com as condições C1, C2 e C3 e as ações A4 e A5. Finalizando as árvores de comportamento dos elementos do contexto, é necessário fundir as árvores resultantes para gerar o comportamento geral do robô para a aplicação. A Figura 43 ilustra uma composição das árvores das Figuras 41 e 42 a partir de um nodo raiz seletor, resultando na árvore de comportamento da aplicação.

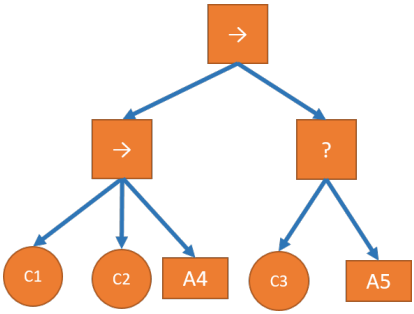


Figura 42 – Exemplo de uma árvore de comportamento para tratar elementos de contexto

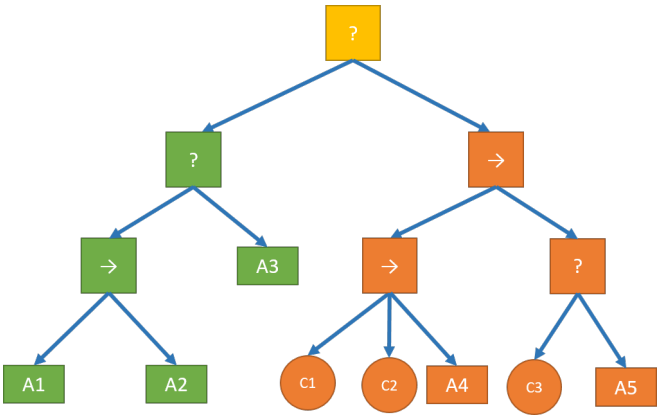


Figura 43 – Árvore de comportamento para a aplicação criada a partir da fusão das árvores da tarefa e do contexto

Por exemplo, considerando uma aplicação de robô de patrulha

com o nível de energia como elemento de contexto, será necessário definir duas árvores: uma para tratar a tarefa do robô (no caso, de patrulha) e outra para tratar o elemento de contexto disponível (o nível de energia). As árvores resultantes de cada processo serão integradas em uma árvore de comportamento da aplicação a partir da composição de um nodo de seletor ou de sequência. Caso a aplicação tenha outras tarefas ou outros elementos de contexto poderá ser necessário criar novas árvores ou modificar as já existentes.

Inicialmente será definida a árvore de comportamento da tarefa de patrulha. O comportamento inicial do agente seria patrulhar as áreas A e B de um mapa hipotético. Para isso é definida uma árvore de comportamento com 2 ações: ir para A e ir para B. Como o agente precisa visitar essas 2 áreas em ordem, a árvore de comportamento será um nodo de sequência com essas duas ações.

A Figura 44 ilustra como é esta árvore de comportamento. O nodo do comportamento 'patrulhar' é o nodo de sequência e o agente irá executar as 2 ações em ordem.

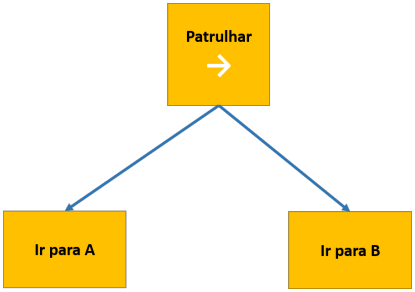


Figura 44 – Árvore de comportamento de patrulha

Uma vez definida a árvore da tarefa, é necessário definir a árvore que irá tratar o contexto. Além de patrulhar o mapa, é necessário manter o agente em funcionamento. O deslocamento de uma área para outra demanda um nível de energia mínimo para o agente funcionar. Logo, é necessário definir um comportamento de 'permanecer vivo' para que o agente possa manter seu comportamento de patrulha. Para esse comportamento é necessário inicialmente verificar o nível de energia do agente. A verificação do nível de energia é traduzida como um nodo de condição. Além de verificar o nível, o comportamento de permanecer vivo tem outras duas ações: ir até a fonte de energia e recarregar a bateria.

A Figura 45 ilustra a árvore de comportamento resultante. O

nodo do comportamento 'permanecer vivo' é um nodo de sequência que inicialmente verifica a condição do nível de energia, se ele for insuficiente ele parte para as ações de deslocar até uma fonte de energia e recarregar a bateria.

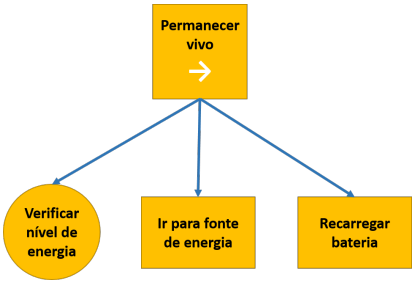


Figura 45 – Árvore de comportamento de permanecer vivo

Com essas duas árvores de comportamento, resta definir a prioridade das árvores e a forma de composição das mesmas. Apesar do comportamento padrão do agente ser o de patrulha, não é possível patrulhar sem um nível mínimo de energia disponível. Logo, o comportamento 'permanecer vivo' deve ter prioridade ao comportamento 'patrulhar'. Assim, ambas árvores são integradas a partir de uma composição de seleção por um nodo raiz seletor. O agente deve, em determinado momento, apresentar o comportamento ou de permanecer vivo ou o de patrulha. A figura 46 ilustra a árvore resultante final da aplicação de patrulha.

Primeiramente a árvore busca definir o comportamento de permanecer vivo ao agente, já que ele está na parte mais à esquerda da árvore. O primeiro nodo do comportamento é a condição do nível de energia. Caso o nível de energia seja menor que um certo valor considerado crítico, esta condição retorna sucesso e a árvore passa para o próximo nodo da sequência. O nodo em questão é a ação de ir até uma fonte de energia. Essa ação é uma ação de deslocamento do agente, logo, enquanto ele se movimenta até a fonte, o nodo irá retornar como executando. Quando chegar à fonte de energia, o nodo retorna sucesso e a árvore executa a última ação do comportamento, de recarregar bateria. Enquanto estiver recarregando, o nodo retorna executando até recarregar por completo, retornando sucesso para o nodo pai (o nodo do comportamento 'permanecer vivo') que, por sua vez, retorna sucesso para o nodo raiz. De outra forma, caso a condição do nível de energia retorne falha, ou seja, a energia do robô não está em estado crítico, o

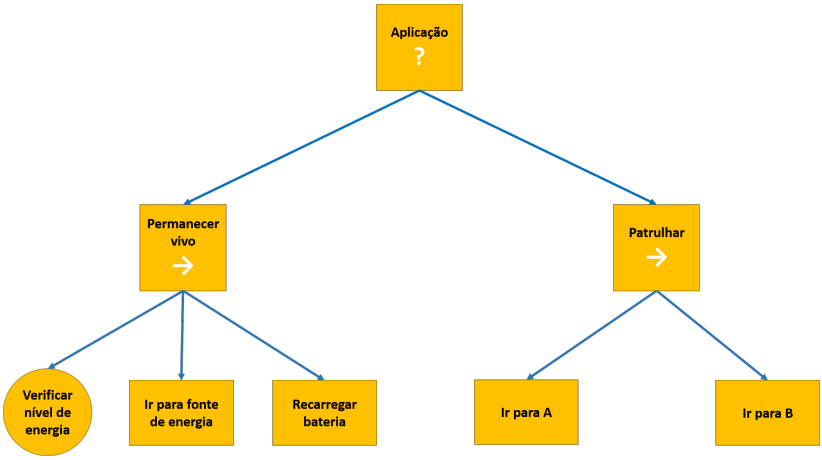


Figura 46 – Árvore de comportamento final da aplicação de patrulha

nodo do comportamento 'permanecer vivo' retorna falha, já que é um nodo de sequência. Por sua vez, o nodo raiz tenta executar seu próximo nodo (nodo de comportamento 'patrulhar'), pois é um nodo seletor. O comportamento 'patrulhar', por sua vez, irá executar duas ações de deslocamento em sequência. Enquanto não chegar ao ponto A, o nodo retorna como executando. Assim que chegar ao ponto A, retorna como sucesso e passa para a próxima ação: ir ao ponto B que, por sua vez, retorna sucesso ao nodo raiz.

É possível modelar diversos comportamentos para o agente, dependendo de quais são os elementos de contexto que devem ativar tal comportamento. A estrutura da árvore de comportamento permite adicionar sub-árvores, sem precisar descartar toda árvore já produzida. No caso de uma nova tarefa ou um novo elemento de contexto, basta produzir uma sub-árvore referente ao novo comportamento e incorporá-la na árvore de comportamento original, decidindo sua posição de acordo com sua prioridade em relação aos demais comportamentos.

É possível, por exemplo, expandir a árvore da Figura 46 acrescentando um contexto de detecção de intruso e o comportamento de tratamento para tal evento. A partir de um mapa semântico disponível com todas as localizações de fontes de energia e seu estado, também é possível adicionar a ação 'identificar fonte mais próxima' no comportamento de 'permanecer vivo'. Estes novos elementos de contexto podem ser incorporados em uma nova árvore de comportamento, ilustrada na

Figura 47. O novo comportamento 'detectar intruso' deve ser integrado na árvore através da composição de seleção em uma ordem relativa a sua prioridade. Como detectar intruso e acionar alarme é mais importante que patrulhar mas menos importante que permanecer vivo, ele é posicionado no meio da árvore da aplicação.

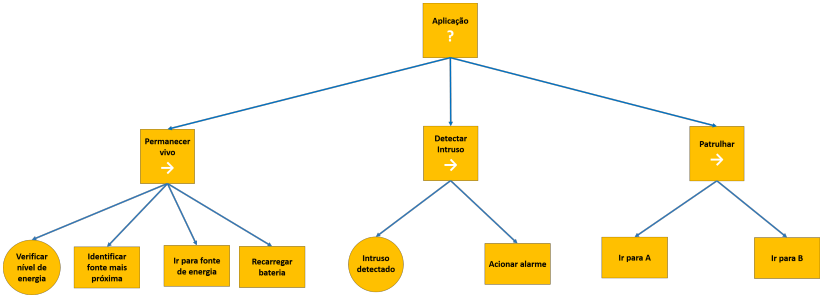


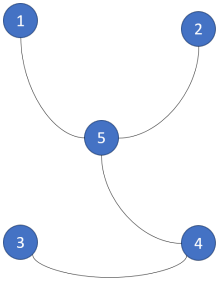
Figura 47 – Árvore de comportamento expandida com elemento de contexto de detecção de intruso

Quanto mais elementos de contexto estiverem disponíveis na aplicação, mais completos serão os possíveis comportamentos. A partir de uma trajetória semântica, a execução de uma árvore de comportamento monitora os elementos de contexto que descrevem o cenário, definindo o comportamento apropriado. Esse processo é a parte inteligente do robô, que analisa os dados de contexto capturados para adaptar a navegação do robô de acordo com a necessidade.

4.2.3 Mapa Semântico e Planejamento de Caminho

O mapa semântico é responsável por manter informações métricas e semânticas sobre o ambiente. O diferencial do mapa semântico proposto neste trabalho está no fato de ter, além do significado dos locais, a descrição do estado atual de acordo com o elemento de contexto. Quando um elemento de contexto de uma localidade no mapa é capturado, ele é incorporado ao vértice do local e atualizado no mapa semântico, que agora conta com um novo estado do elemento. Por exemplo, o vértice 4 da Figura 48 representa uma ponte levadiça com um sensor que indica se ela está levantada ou abaixada. Caso ela esteja levantada, o caminho que liga o vértice 3 e 4 não estará disponível, mudando o estado do mapa. Assim que ela for abaixada, o caminho volta a existir.

Os elementos de contexto também servem para moldar o comportamento do robô de acordo com o estado atual do mapa. Por exemplo, os vértices 4 e 5 do mapa da Figura 48 são localizados em um ambiente aberto. Como o clima é de chuva, os demais vértices de ambiente fechado não foram afetados. Com a informação da chuva nos vértices 4 e 5, o robô pode precisar limitar sua velocidade para evitar acidentes ao atravessar estes vértices. Mas é preciso definir quais locais do mapa são afetados pela chuva, uma vez que partes cobertas não são afetadas. Portanto, apesar de ser um elemento de contexto do mapa inteiro, apenas algumas partes são afetadas. O mapa semântico permite relacionar estas informações e, quando o robô estiver em uma área afetada, ele limita sua velocidade.



vid	x	y	type	Contexto
1	20	50	Quarto	
2	55	47	Quarto	
3	18	5	Quarto	
4	50	7	Ponte	Clima: Chuva; Estado: Levantada
5	30	20	Corredor	Clima: Chuva

Figura 48 – Exemplo de um mapa semântico e as informações de cada vértice semântico

A partir de um mapa semântico do ambiente, que identifica os locais, objetos e seus relacionamentos, o deslocamento do robô utiliza o mapa semântico para navegar no ambiente. Como mencionado na seção 4.1.3, o caminho global é planejado a partir do mapa semântico através do algoritmo A*, que retorna o conjunto de vértices para ir do ponto inicial até o destino da tarefa. Já o caminho local é executado através da biblioteca *dwa_local_planner* do ROS que, a partir de um mapa métrico e de sensores lasers do robô, planeja um caminho livre de obstáculos entre dois pontos.

Quando um novo elemento de contexto é incorporado em um vértice do mapa, os custos das arestas referentes ao vértice podem ser recalculados, uma vez que o estado do ambiente foi alterado. O custo das arestas de vértice é uma combinação ponderada do custo de tempo de travessia mais o custo de gasto de energia, logo, caso um elemento de contexto influencie em alguma dessas variáveis do robô, as arestas podem sofrer alteração no custo. Isso pode ser automatizado através

da consulta do histórico de execução, explorado nas próximas seções.

4.3 CRIAÇÃO DE TRAJETÓRIA SEMÂNTICA

Uma vez definida a tarefa e o caminho a ser percorrido, o próximo processo foca na construção da trajetória semântica do robô. A trajetória semântica é essencial para descrever e armazenar as execuções de tarefas do robô juntamente com o contexto observado, que irá servir de base para o refinamento dos comportamentos. A trajetória semântica será construída seguindo o modelo CONSTAnT-MR definido na seção 4.1.2.

Durante o deslocamento do robô, os pontos de sua localização espacial são capturados juntamente com o instante de tempo, formando a trajetória do robô. A transformação da trajetória em trajetória semântica se dá incorporando os elementos de contexto observados com os pontos da trajetória do robô a partir da dimensão do espaço, do tempo ou do conjunto das duas. A caracterização do elemento de contexto através das dimensões WWW auxilia na definição de como ele vai se relacionar com as variáveis de tempo e espaço dos pontos da trajetória.

Como o ponto da trajetória semântica tem o menor grau de granularidade da trajetória, um elemento de contexto pode ser facilmente associado com ele, utilizando o instante de tempo de captura ou a posição espacial do ponto e do contexto. Logo, se o ponto da trajetória tem o mesmo instante de tempo que o elemento de contexto capturado, eles são associados. Por exemplo, o nível de bateria do robô pode ser capturado a cada segundo, associando o ponto da trajetória ao nível de bateria daquele tempo específico. Logo, o nível de bateria do robô no instante de tempo t_i será associado ao ponto da trajetória do robô de tempo t_i. O mesmo raciocínio pode ser aplicado na dimensão espacial. Quando um ponto da trajetória está perto ou dentro da posição do elemento de contexto, ambos podem ser associados. Por exemplo, em uma avenida onde a velocidade máxima é 60 km/h, os pontos da trajetória que estão contidos nesta avenida também terão esse contexto associado.

Como apontado na **Definição 11** e no modelo CONSTAnT-MR, cada sub-trajetória semântica possui uma tarefa específica. Logo, as sub-trajetórias do robô são divididas de acordo com a tarefa em execução do robô. Uma vez que uma sub-trajetória possui uma tarefa, seus elementos de contexto descrevem a execução da tarefa. Quando um robô finaliza ou aborta uma tarefa, uma nova sub-trajetória é criada.

A seção 4.2.1 define que, uma vez que os pontos são capturados na casa dos segundos, os elementos de contexto também seriam abstraídos na mesma unidade. Mas uma sub-trajetória representa um conjunto de pontos e, portanto, tem um período de duração acima da unidade temporal do contexto. Para incorporar esses elementos de contexto na sub-trajetória é necessário criar funções de agregação para o contexto. Uma função de agregação permite abstrair múltiplos valores em um só. Como cada elemento de contexto é diferente, pode-se definir uma regra geral de abstração de contexto de sub-trajetória a partir da equação (4.5) onde Ss é uma sub-trajetória semântica, $Ss.C_X$ é o elemento de contexto de identificador X, $Ss.P$ é o conjunto de pontos da sub-trajetória Ss, p é um ponto de $Ss.P$ e $p.C_x$ é o elemento de contexto de identificador x do ponto p. Em outras palavras, a equação (4.5) define que um novo elemento de contexto $Ss.C_X$ da sub-trajetória é formado por uma função de agregação do elemento de contexto $p.C_x$ de todos os pontos de Ss.

$$\begin{gathered} Ss.C_X = funcao(p.C_x) \\ \forall p \in Ss.P \end{gathered} \tag{4.5}$$

Por exemplo, a velocidade do robô é um contexto do robô (descreve o movimento do robô), um contexto local (a velocidade pertence a um ponto específico) e um contexto temporal (o valor pode mudar com o tempo). Como a velocidade está associada com o ponto semântico, é possível abstrair o contexto de velocidade média para a sub-trajetória semântica a partir da equação (4.6), onde Ss é a sub-trajetória semântica, $\overline{C_{vel}}$ é o novo elemento de contexto de velocidade média da sub-trajetória Ss, p_i é o iésimo ponto da sub-trajetória Ss, C_{vel} é o elemento de contexto da velocidade do ponto p_i e n é o número total de pontos da sub-trajetória.

$$Ss.\overline{C_{vel}} = \frac{\sum_{i=1}^{n} p_i.C_{vel}}{n} \tag{4.6}$$

4.3.1 Armazenamento de Informações

Os dados de execução das tarefas são essenciais para construir o histórico sobre como o comportamento do robô é afetado pelos elementos de contexto. Como o CONSTAnT-MR tem uma arquitetura hierárquica e normalizada, o modelo é traduzido em tabelas em um banco de dados relacional. Um SGBD (Sistema Gerenciador de Banco

de Dados) permite armazenar as informações de maneira estruturada, fornecendo ferramentas de consultas através de linguagem SQL (*Structured Query Language*), linguagem padrão de consulta de dados em bancos relacionais.

Cada tabela do banco de dados será uma implementação direta de cada entidade do CONSTAnT-MR. Como uma trajetória é constituída de dados espaciais com uma dimensão de tempo, é possível utilizar banco de dados geográficos para armazenar as informações espaciais, que fornece suporte para dados como pontos, linhas e geometria, além de funções como distância, cumprimento e interseção. Neste trabalho foi utilizado o SGBD Postgres (GROUP, 2014) juntamente com a extensão para dados geográficos Postgis (Refractions Research, 2011). O Postgres é um banco de dados objeto relacional *open-source* e grátis, além de robusto e seguro.

Os elementos de contextos são todos armazenados em uma única coluna, chamada contexto. A coluna recebe os dados de contexto no formato JSON (*JavaScript Object Notation*) que permite expressar diversas informações em uma estrutura só, sendo identificadas através de um rótulo. O formato JSON é composto de duas estruturas: uma coleção de pares de nomes/valores e uma lista ordenada destes pares. Este formato é fácil para seres humanos lerem e escreverem e fácil para máquinas interpretarem e gerarem (CROCKFORD, 2006). O próprio SGBD fornece ferramentas para manipular dados deste tipo, permitindo buscas pelas informações dos elementos de contextos através de seus rótulos.

A cada finalização de uma tarefa (seja ela completada com sucesso ou abortada), os pontos da trajetória são armazenados no servidor do SGBD, junto com sua sub-trajetória e todas as informações capturadas. Como o vértice do mapa também tem elementos de contexto associado, as informações do mapa também são armazenadas, gerando novos registros a cada mudança de estado capturada.

Essas informações servem de base para o próximo processo que, a partir deste conjunto de informações do banco de dados, gera conhecimento para o usuário remodelar os comportamentos do robô.

4.3.2 Análise de Dados

Após algumas execuções da aplicação, é possível analisar o histórico através das trajetórias semânticas armazenadas no banco de dados para extrair informações sobre a aplicação através de análise de da-

dos. O processo de análise de dados utiliza algoritmos de mineração nas trajetórias e dados armazenados no SGBD, determinando a partir dos dados como os elementos de contexto podem afetar a execução das tarefas. Este processo é um dos mais importantes para o refinamento e realimentação do framework, melhorando a navegação inteligente do robô. Podemos dividir este processo em duas partes: análise de comportamento do robô e atualização do mapa semântico.

4.3.2.1 Análise de Comportamento do Robô

A análise do comportamento do robô se propõe a verificar como os elementos de contexto se comportam ao longo da execução, tentando identificar possíveis relações entre seus valores e tarefas que falharam ou que tiveram um custo elevado. Se, por exemplo, uma tarefa sempre falha quando ocorre um evento ou passa por uma certa região do mapa, pode-se reformular a execução da tarefa para que ela possa ser executada com sucesso. Identificando o comportamento do robô e o resultado das ações tomadas, novos comportamentos podem ser derivados para tratar os problemas identificados de maneira mais eficiente. A análise de histórico de trajetórias do robô permite refinar todos os outros processos anteriores, resultando na melhoria do processo final.

Como cada aplicação possui dados específicos, o framework não define quais os algoritmos de mineração de dados devem ser aplicados. Para isso é necessário um analista de dados que conhece a aplicação. Entretanto, pode-se listar alguns algoritmos e técnicas gerais que podem ser utilizadas para a extração de conhecimento destes dados.

Uma forma relevante de associar quais elementos de contexto influenciam na execução da tarefa é através do coeficiente de correlação de Pearson, calculado através da equação (4.7). A correlação de Pearson mede o grau em que duas variáveis métricas são correlatas, ou seja, o relacionamento entre duas variáveis. Esta correlação é medida em uma escala entre -1 e 1 onde -1 indica uma correlação negativa (quando uma variável aumenta de valor, a outra sempre diminui) e 1 indica uma correlação positiva (quando uma variável aumenta de valor, a outra sempre aumenta). Apesar do coeficiente de Pearson não indicar causalidade, ele pode ser usado para analisar os elementos de contexto e seus relacionamentos.

$$p(x,y) = \sum_{i=1}^{n} \frac{(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}} \quad onde \quad \overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \text{ e } \overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{4.7}$$

Existem outros algoritmos de mineração de dados que podem auxiliar na compreensão do comportamento do robô. Por exemplo, a partir da aplicação do algoritmo *k-means* (HARTIGAN; WONG, 1979), é possível agrupar dados que tenham similaridade em *clusters*. O algoritmo utiliza a distância euclideana entre os pontos juntamente com o número desejado de *clusters* para identificar proximidade entre os valores dos dados.

A Figura 49 ilustra a aplicação do algoritmo *k-means* com k = 3. Cada dado é classificado em um dos três grupos identificados (pontos azuis, verdes e vermelhos) de acordo com a proximidade do dado com o centro do *cluster*. Aplicando o algoritmo nos dados do histórico e analisando as características dos elementos de contextos das tarefas executadas com sucesso e das tarefas abortadas, é possível identificar possíveis padrões de similaridade nos elementos de contexto. Por exemplo, se o histórico de execução de um robô resultar nos dados da Figura 49 e o *cluster* azul representar as tarefas abortadas, os valores dos demais elementos de contexto podem auxiliar na identificação da causa da falha de execução.

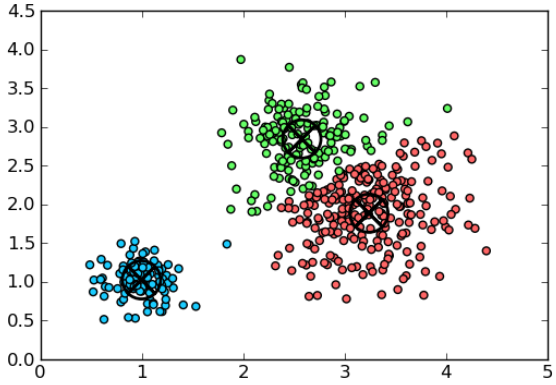


Figura 49 – Aplicação do algoritmo k-means agrupando dados em 3 clusters (PACULA, 2011)

Tabela 4 – Histórico de execução de tarefas para algoritmo Apriori

Item
Tarefa Abortada, Clima Chuva, Velocidade Baixa
Tarefa Executada, Clima Sol, Velocidade Baixa
Tarefa Executada, Clima Sol, Velocidade Alta
Tarefa Abortada, Clima Chuva, Velocidade Baixa
Tarefa Executada, Clima Nublado, Velocidade Média
Tarefa Abortada, Clima Chuva, Velocidade Baixa
Tarefa Abortada, Clima Chuva, Velocidade Alta

Outra possibilidade é, a partir de dados qualitativos, ou seja, dados não numéricos, é possível analisar os dados do histórico a partir do algoritmo Apriori (AGRAWAL; SRIKANT et al., 1994). O algoritmo Apriori se baseia em itens frequentes em um conjunto de dados, ou seja, em itens que aparecem frequentemente juntos em vários registros.

A Tabela 4 ilustra um exemplo de como identificar causas de falha das tarefas. Todas as vezes que o item *Clima Chuva* aparece nos dados, também aparece o item *Tarefa Abortada*, o que pode indicar uma relação de causa entre esses elementos. De maneira similar, 75% das vezes que aparece o item *Velocidade Baixa* também aparece o item *Tarefa Abortada*, outro fato que também pode implicar a relação entre os valores destes elementos. Uma vez identificado os possíveis problemas, é necessário criar ou modificar a árvore de comportamento para tratar destes elementos de contexto.

Importante notar que a análise do comportamento auxilia na identificação de possíveis problemas na modelagem do comportamento, cabendo então ao projetista interpretar as informações e remodelar a árvore de comportamento para que consiga tratar apropriadamente os elementos de contexto que influenciam negativamente na execução das tarefas.

4.3.2.2 Atualização do Mapa Semântico

Diferentemente da análise de comportamento, a atualização do mapa semântico pode ser automatizada mais facilmente. O custo das arestas do mapa é definido pela equação 4.4 que analisa o tempo de

Tabela 5 – Exemplo de pontos semânticos armazenados no histórico de execução

pid	x	y	t	Contexto
101	4.5	5.6	10	Local: A, Energia: 40, Clima: Chuva
102	4.7	5.0	15	Local: A, Energia: 35, Clima: Chuva
103	5.0	4.8	20	Local: A, Energia: 28, Clima: Chuva
104	5.1	4.8	25	Local: A, Energia: 24, Clima: Chuva
105	5.5	4.0	30	Local: B, Energia: 20, Clima: Chuva
...	...	...	...	...
301	4.5	5.6	210	Local: A, Energia: 65, Clima: Sol
302	4.7	5.0	215	Local: A, Energia: 62, Clima: Sol
303	5.5	4.0	220	Local: B, Energia: 60, Clima: Sol
304	5.1	4.8	225	Local: B, Energia: 57, Clima: Sol
305	5.5	4.0	230	Local: C, Energia: 56, Clima: Sol

travessia e o gasto de energia, duas informações que podem ser extraídas do histórico de execução através das trajetórias semânticas.

O CONSTAnT-MR define que cada ponto semântico está associado a um local do mapa. Por exemplo, a Tabela 5 ilustra o histórico de pontos semânticos capturados em um intervalo de 5 segundos com seu identificador *pid*, suas coordenadas x e y, seu instante de tempo t e seus elementos de contextos associados. A partir deste histórico, é possível extrair que, para o contexto *clima* de chuva, ele levou 20 segundos para atravessar do vértice A para o vértice B. Entretanto, para o contexto *clima* de sol, ele levou 10 segundos para atravessar do vértice A para o vértice B.

Com essa informação, é possível atualizar o mapa semântico para que, quando identificar o elemento de contexto do clima para chuva, o custo de tempo da aresta entre os vértices A e B seja 20 segundos, enquanto o custo de tempo para a mesma aresta com o clima sol seja 10 segundos. Isso permite que o mapa semântico se adapte aos elementos de contextos identificados de acordo com o histórico de execução.

Este mesmo processo pode ser aplicado para o custo de energia da aresta. No mesmo exemplo da Tabela 5, o custo de energia para o robô atravessar do vértice A para o vértice B no clima de chuva foi de 20 unidades de energia. Já o custo de energia para o clima de sol foi

de 5 unidades de energia.

Este processo de atualização de custo do mapa pode ser feito automaticamente após algumas execuções da aplicação. Entretanto é importante notar que, quanto maior o histórico de execução da aplicação, melhor o resultado da atualização de custo, uma vez que o framework terá dados o suficiente para expressar o custo de maneira mais fiel a realidade.

4.4 CONTROLE DE BATERIA

Além da definição de novos comportamentos e de atualização do mapa semântico, elementos de contexto também podem auxiliar a resolver o problema da recarga autônoma (ARP) em robôs móveis. Este trabalho propõe uma abordagem fuzzy para o problema da recarga autônoma usando elementos de contexto.

Como mencionado na seção 3.5, diversos trabalhos da literatura usam um limite fixo para decidir quando um robô deve abortar a tarefa atual e recarregar. Essa abordagem é muito restritiva, levando a dois problemas principais. No primeiro, ilustrado na Figura 50, o robô está longe da tarefa e próximo da fonte, com baixa bateria. Porém, como não atingiu o limite fixo definido, ele irá tentar executar a tarefa da mesma maneira, mesmo sem ter energia suficiente para tal feito.

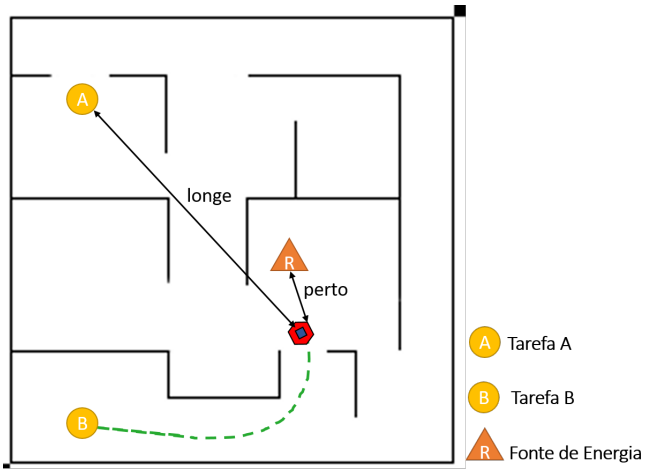


Figura 50 – Problema da abordagem de limite fixo tentando executar tarefa longe

Já no segundo problema, ilustrado na Figura 51, o robô está próximo de completar a tarefa, entretanto, ele atingiu o limite fixo da bateria. Logo, independente a distância para a tarefa, ele irá abortar e tentar recarregar. Isso leva a um desperdício de energia do robô, já que toda energia gasta com uma tarefa abortada poderia ser usada de maneira mais útil. Logo, o nível de bateria atual sozinho não é o suficiente para determinar quando o robô deve recarregar a bateria.

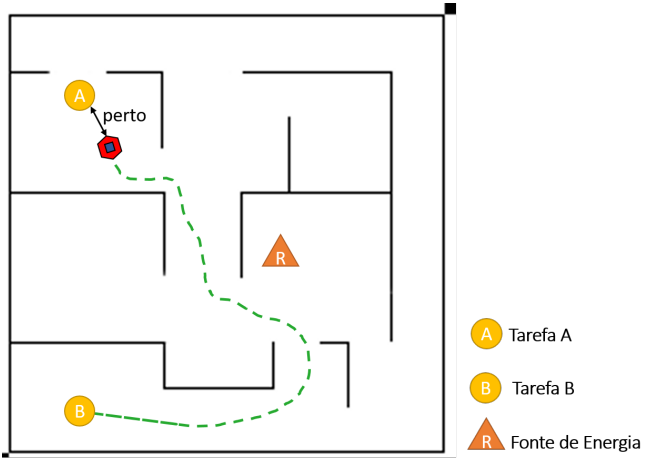


Figura 51 – Problemas da abordagem de limite fixo abortando tarefas próximas

Para ter um controle mais inteligente da bateria, foi proposta uma abordagem fuzzy que considera inicialmente, além do nível atual de bateria, a distância atual do robô para a tarefa atual e para a fonte de energia mais próxima. O nível de bateria é a entrada mais importante, visto que é o fator principal a ser observado para decidir se o robô deve recarregar. A distância do robô para a tarefa indica o quanto de energia o robô deve considerar que irá consumir, visto que, quanto mais longe a tarefa, mais energia o robô consome. Por fim, como o robô recarrega em uma fonte de energia, a distância atual permite monitorar se o robô está muito longe para poder recarregar.

Em um sistema fuzzy, cada entrada precisa ser mapeada em um termo de linguagem natural através de uma função de pertinência. Esta tese se limita a definir os termos e regras do sistema fuzzy, dando um exemplo de função de pertinência que pode ser utilizada em uma aplicação. Porém, uma vez que o consumo de bateria e a distância

entre as tarefas podem variar de acordo com a aplicação, as funções de pertinência e seus limites podem precisar sofrer ajustes para se adequar a cada cenário.

Para o processo de fuzzyficação foram definidos os termos relativos a cada conjunto de entradas. Por exemplo, a Figura 52 ilustra a função de pertinência para o nível da bateria com quatro termos: completo, operacional, cuidado e crítico. O termo completo significa que o nível da bateria está cheio ou quase completo. Operacional significa que não está completo mas tem energia suficiente para o robô continuar executando suas tarefas sem precisar recarregar. O termo crítico significa que a bateria está quase acabando e que o robô deve recarregar. Já o termo cuidado significa que o nível de bateria está em uma área de perigo, onde pode ter energia suficiente para executar algumas tarefas mas talvez seja melhor recarregar primeiro.

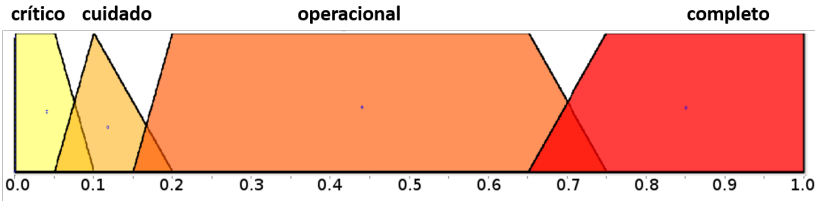


Figura 52 – Função de pertinência para a energia

A distância da tarefa e da fonte de energia compartilham a mesma função de pertinência, uma vez que são duas informações sobre a distância atual do robô para um destino. A Figura 53 ilustra a função de pertinência com três termos: longe, médio e perto. Como um robô pode ter tarefas localizadas pelo mapa inteiro, a definição do que é longe, médio e perto é relativa. Portanto, as entradas da distância são normalizadas em um intervalo entre 0 e 1 a partir das equações (4.8) e (4.9) onde CTD é a distância atual do robô até a tarefa, CSD é a distância atual do robô até a fonte de energia, ITD é a distância inicial do robô até a tarefa e ISD é a distância inicial do robô até a fonte de energia. Aplicando estas fórmulas nas entradas, é possível modelar a distância de maneira dinâmica dependendo da tarefa atual.

$$CTD' = min\left\{\frac{CTD}{max(ITD,ISD)}, 1\right\} \tag{4.8}$$

$$CSD' = min\left\{\frac{CSD}{max(ITD,ISD)}, 1\right\} \tag{4.9}$$

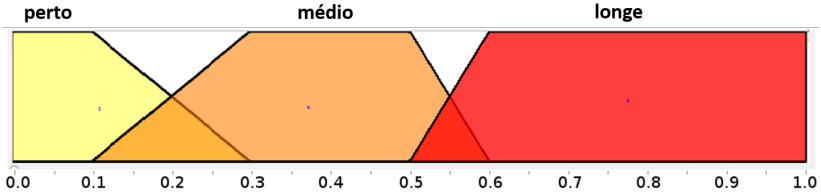


Figura 53 – Função de pertinência para a distância

Os termos CTD' e CSD' apresentam a distância final normalizada, dependendo da maior distância inicial identificada. Logo, os termos de longe, médio e perto podem mudar para cada tarefa. Entretanto, para isso, é necessário que nenhuma distância seja maior do que a capacidade máxima de deslocamento do robô a partir da bateria.

As próximas etapas do processo fuzzy necessitam do conjunto de regras para inferência e o conjunto de saídas do sistema. A saída pode assumir duas possíveis: ou o robô vai continuar executando a tarefa ou ele vai abortar e ir recarregar. Apesar de um sistema fuzzy permitir que as regras sejam geradas a partir de um treinamento com redes neurais, neste trabalho foi proposto o seguinte conjunto de regras:

1. SE NívelDeEnergia É **crítico** ENTÃO Saída É **Recarregar**
2. SE NívelDeEnergia É **cuidado** E DistânciaDaTarefa É **perto** E DistânciaDaFonte É **longe** ENTÃO Saída É **Recarregar**
3. SE NívelDeEnergia É **cuidado** E DistânciaDaTarefa É **perto** E DistânciaDaFonte NÃO É **longe** ENTÃO Saída É **TerminarTarefa**
4. SE NívelDeEnergia É **cuidado** E DistânciaDaTarefa É **médio** OU **longe** ENTÃO Saída É **Recarregar**
5. SE NívelDeEnergia É **operacional** OU **completo** ENTÃO Saída É **TerminarTarefa**

Uma vez que sem energia o robô não consegue mais executar tarefas, a prioridade deve ser sempre evitar ficar sem energia. Logo, sempre que o nível de energia estiver no **crítico**, na regra 1, o robô deve abortar a tarefa atual e ir recarregar. Para isso, os limites da função de pertinência da energia no termo **crítico** deve sempre reservar um nível mínimo de energia para o robô chegar até a fonte. Por outro lado,

se o nível de energia é **operacional** ou **completo**, na regra 5, o robô não precisa se preocupar com ficar sem energia, podendo continuar sua tarefa.

O maior problema se encontra no nível **cuidado**. A bateria do robô no nível **cuidado** representa uma faixa de execução do robô onde ele pode terminar mais algumas tarefas, mas deve ter atenção para não ficar sem energia. Para saber se é possível executar mais tarefas ou se ele deve recarregar, o sistema fuzzy verifica a distância atual do robô para a tarefa e para a fonte. Considerando tarefas de deslocamento, ou seja, tarefas que só gastam energia na locomoção, o robô deve verificar a distância atual para seu destino e decidir se consegue terminar a tarefa e ir para a fonte de energia ou se é melhor já abortar e recarregar. Se a distância para a tarefa é **médio** ou **longe**, na regra 4, então é melhor não arriscar tentando terminar a tarefa e ir recarregar, uma vez que uma distância **média** ou **longe** pode gastar muita bateria. Por outro lado, se a distância da tarefa é **perto** e a distância da fonte não é **longe** (ou seja, a distância da fonte é **médio** ou **perto**), na regra 3, então o robô pode arriscar terminar a tarefa, uma vez que se encontra próximo dela. E por fim, se a distância da tarefa é **perto** mas a distância da fonte é **longe**, na regra 2, é preferível que o robô aborte a tarefa e vá recarregar, dando prioridade para a recarga.

Apesar deste trabalho resolver o problema da recarga autônoma com este sistema fuzzy com três entradas, ele é flexível o suficiente para incorporar outras entradas que possam auxiliar na melhor decisão quanto a recarga ou não da bateria. Caso seja interessante considerar algum elemento de contexto que influencia no consumo da bateria como, por exemplo, número de objetos carregados pelo robô, é só modelar a entrada em uma função de pertinência do sistema fuzzy e alterar as regras de inferência para considerar este novo dado. Já para tarefas que não sejam só de locomoção, onde o robô deve gastar energia para finalizá-la (por exemplo, reparar uma máquina), é possível definir funções de pertinência de energia específica para cada tarefa, considerando, além da energia consumida na locomoção, a energia necessária para finalizar a tarefa.

O sistema fuzzy de controle de recarga autônoma permite evitar os problemas do limite fixo onde um robô pode abortar uma tarefa quando ele estiver quase finalizando ou quando um robô vai tentar executar uma tarefa, mesmo que ela esteja a uma grande distância e ele não teria energia o suficiente para alcançar o destino.

4.5 SÍNTESE DO CAPÍTULO

Este capítulo apresentou as contribuições desta tese separadas em três partes: definições, estrutura framework e controle de bateria.

Para compreender a contribuição da tese, foram propostas uma série de definições formais que servem de base para a construção do framework. As definições propostas adaptam as estratégias da literatura para o problema da navegação inteligente. Primeiramente foi proposto a modelagem de contexto WWW (WHAT, WHERE, WHEN), que caracteriza um elemento de contexto em três dimensões: significado, espaço e tempo. Diferente de propostas de modelagem de contexto anteriores que focam apenas no significado (TURNER, 1998; FOUKARAKIS et al., 2014; AFYOUNI et al., 2013), a proposta desta tese explora características espaciais e temporais do contexto, essenciais para a integração com a trajetória semântica.

Para a trajetória semântica foi proposto o modelo CONSTAnT-MR, uma adaptação do modelo CONSTAnT (BOGORNY et al., 2014) para trajetória semântica de humanos. O modelo proposto incorpora características referentes à robótica móvel, formalizando diversas definições a partir deste novo modelo, incorporando os elementos de contexto nos diversos níveis de uma trajetória.

O mapa semântico proposto se diferencia de outras abordagens adicionando elementos de contexto através dos vértices semânticos. Além do significado de cada vértice do mapa, o contexto permite caracterizar o estado atual do mapa, que infere diretamente no custo de travessia do vértice. A proposta também define uma fórmula de custo de travessia do mapa baseado no histórico de execução da aplicação.

Todas as definições foram integradas na proposta através do framework ConBINaMoR. O framework é composto de sete processos principais divididos em vários passos. A Figura 54 resume os passos de cada processo em etapas de pré-definição, definição, execução e pós-execução. A etapa de pré-definição incorpora os passos do processo de coleta de informações de contexto. Com essas informações, a etapa de definição modela o cenário da aplicação, incluindo passos dos processos de criação de trajetória semântica, decisão de comportamento e mapa semântico. A próxima etapa contempla a execução da aplicação, de acordo com o modelo definido no processo anterior. Essa etapa inclui passos dos processos de decisão de comportamento, planejamento de caminho, armazenamento de informações, coleta de informações de contexto e criação de trajetória semântica. Por fim, a etapa de pós-execução utiliza as informações capturadas e armazenadas na etapa de

execução para a melhoria da modelagem, incluindo passos dos processos de análise de comportamento do robô e atualização do mapa semântico.

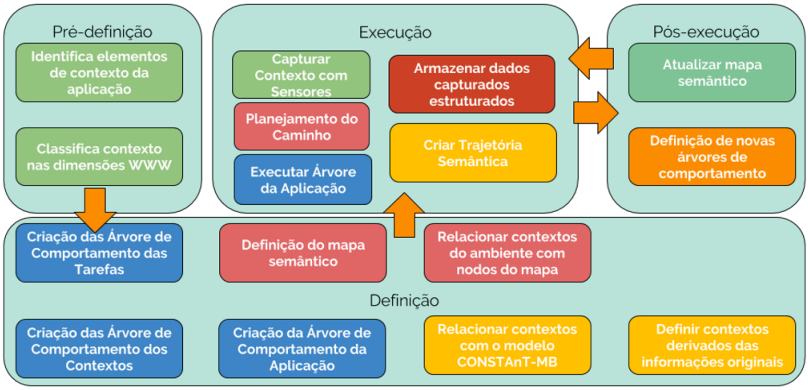


Figura 54 – Resumo dos passos de cada processo do framework ConBINaMoR

A última contribuição desta tese foi uma abordagem fuzzy para o problema de recarga autônomo. A proposta, diferente de trabalhos da literatura, utiliza a lógica fuzzy, juntamente com elementos de contexto de nível de energia do robô, localização do robô, localização da tarefa e localização da fonte para, a partir de uma série de regras de inferência, decidir se o robô deve finalizar a tarefa ou abortar e ir recarregar. O grande diferencial da abordagem foi utilizar elementos de contexto além do nível de bateria para gerenciar o controle de recarga do robô.

O conjunto destas contribuições compõem a estratégia de navegação inteligente de robôs móveis desta tese.

5 EXPERIMENTOS

A aplicação do framework será ilustrada com dois estudos de caso. Todas as simulações foram implementadas em C++ e no framework ROS (FOUNDATION, 2014b), utilizando o simulador Stage. Foi simulado uma bateria ideal, que nunca fica viciada e sempre recarrega 100% da carga em todas as recargas. Para fins de experimentos, cada unidade de energia equivale a 1% da bateria.

5.1 EXPERIMENTO 1: O PROBLEMA DA RECARGA AUTÔNOMA

O problema da recarga autônoma se resume na capacidade de um robô móvel gerenciar a recarga de sua bateria de maneira autônoma. Trabalhos da literatura focam em duas categorias de soluções: como recarregar e quando recarregar. Este primeiro experimento aplica o framework proposto em uma aplicação de patrulha para resolver o problema da recarga autônoma focando em quando recarregar.

Nesta aplicação, com o mapa ilustrado na Figura 55, o robô deve patrulhar dois locais do ambiente: A e B. No meio do caminho da patrulha, existe a fonte de energia R, onde o robô recarrega a bateria. O mapa métrico foi transformado em um mapa semântico abstraindo os vértices como cada ambiente, corredor e passagem de entrada.

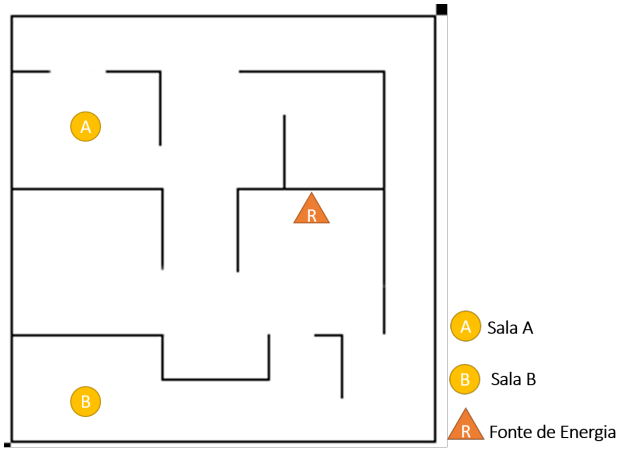


Figura 55 – Mapa métrico da aplicação de patrulha

A Figura 56 ilustra como cada espaço do mapa foi abstraído em um vértice. Cada vértice representado por um círculo é do tipo quarto ou corredor, quadrado é do tipo passagem e triângulo do tipo objeto.

O objeto R é a fonte de energia. Por fim, o mapa topológico resultante está ilustrado na Figura 57. O custo de cada aresta é calculado inicialmente através da equação 4.4 considerando $v_{max} = 5.0$ m/s, $\alpha = 1$ e $\beta = 0$.

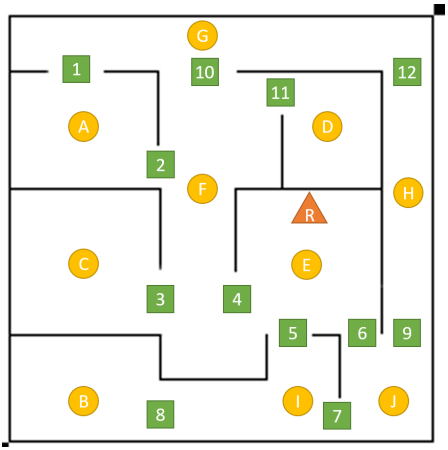


Figura 56 – Mapa métrico da aplicação

O primeiro passo é definir quais os elementos de contexto estão presentes na aplicação e categorizá-los nas dimensões WWW (What, Where, When).

A Tabela 6 resume os elementos selecionados. O nível de energia será observado para verificar quando o robô deve abortar a tarefa e ir recarregar. Este é o elemento de contexto principal para o problema da recarga autônoma. Como o nível de energia descreve o estado da bateria, que pertence ao robô, ele é um contexto do robô. O nível de energia está relacionado com o movimento do robô, não com o mapa como um todo, sendo assim um contexto local. E, como o nível de energia muda com o tempo e o gasto de energia, ele é um contexto temporal. Outro contexto importante é a velocidade de movimento do robô, uma vez que ela impacta diretamente no consumo de energia. As dimensões da velocidade são iguais à do nível de energia, pelos mesmos motivos. O estado da fonte, se está livre ou se está ocupada, também auxilia o robô a decidir o momento ideal de ir recarregar. O estado da fonte é característica de um objeto do mapa, sendo então da

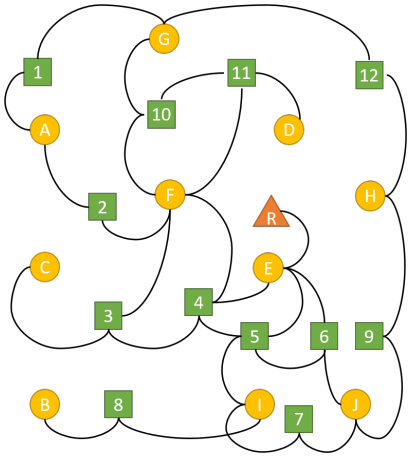


Figura 57 – Mapa topológico da aplicação de patrulha

categoria ambiente. Ele é local, pois afeta somente a fonte de energia e é temporário, pois o estado pode mudar de livre para ocupado no momento que estiver em uso.

Tabela 6 – Elementos de contexto analisados na aplicação

Contexto	What	Where	When
Nível de Energia	Robô	Local	Temporal
Velocidade	Robô	Local	Temporal
Estado da Fonte	Ambiente	Local	Temporal

A partir dos elementos de contexto foi definida a árvore de comportamento ilustrada na Figura 58.

Como a aplicação é similar ao exemplo apresentado na seção 4.2.2, a árvore também tem descrição similar. O nodo raiz, aplicação, é um nodo de seleção que vai retornar sucesso caso um de seus nodos filhos retorne sucesso. O primeiro filho do nodo da aplicação é a árvore de comportamento do contexto de energia do robô, a 'Permanecer vivo'. Esta árvore é um nodo de sequência que irá retornar sucesso caso todos seus filhos retornem sucesso. Seu primeiro nodo filho é um nodo folha de condição 'energia baixa?'. Esse nodo retorna sucesso caso identifique que a energia do robô está baixa. Logo, como o nodo 'Permanecer vivo' é um nodo de sequência, caso a energia esteja baixa

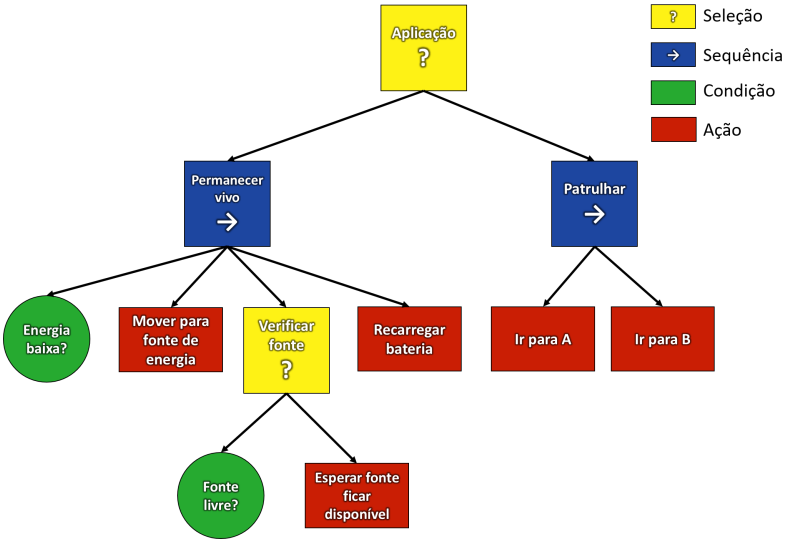


Figura 58 – Árvore de comportamento da aplicação de patrulha

ele vai executar o próximo filho e, caso a condição não identificou que o nível de energia atual do robô seja baixo, retorna falha. Como o comportamento 'Permanecer vivo' é implementado como um nodo de sequência, caso um dos filhos retornar falha ele retorna falha também. Logo, não identificando uma energia baixa, o nodo raiz da aplicação irá tentar executar seu próximo filho, da árvore de comportamento da tarefa de patrulhar. Neste primeiro momento foi implementado o controle de bateria de limite fixo onde, ao chegar a um certo nível de bateria, o robô aborta a tarefa atual, retornando falha para a árvore de patrulha. Como o nodo 'energia baixa?' é um nodo de condição, é possível implementar qualquer outro tipo de algoritmo de verificação de energia, somente retornando o estado no final. No comportamento 'Permanecer vivo', caso identifique uma energia baixa, ou seja, a condição 'energia baixa?' retorne verdade, a árvore executa o próximo nodo filho, a ação 'mover para fonte de energia'. Essa ação comanda o robô para se locomover até a fonte de energia do mapa. Enquanto ele estiver se locomovendo, o nodo retorna 'executando', e ao chegar na fonte, retorna 'sucesso'. Chegando na fonte, o robô deve identificar se ela está livre ou ocupada. Isso foi implementado através do nodo de seleção 'verificar fonte' onde primeiramente verifica a condição 'fonte livre?'. Caso a fonte de energia esteja livre, o robô segue para o próximo filho

do comportamento de 'Permanecer vivo'. Caso a fonte esteja ocupada, ou seja, outro robô esteja usando, o robô executa a ação 'esperar fonte ficar disponível', que retorna 'executando' enquanto a fonte estiver ocupada e 'sucesso' quando a fonte ficar livre. Nesta aplicação só existe uma fonte de energia, portanto o robô deve aguardar que ela fique livre para então recarregar. Com o nodo 'verificar fonte' retornando sucesso, o comportamento executa sua última ação que é 'recarregar bateria'. Enquanto o robô estiver recarregando na fonte a ação retorna 'executando' e, assim que completar a carga, retorna 'sucesso'. Uma vez executado o comportamento 'Permanecer vivo', na próxima execução da árvore da aplicação, o comportamento de patrulha será executado.

Os elementos de contexto da aplicação também são incorporados na trajetória semântica. A partir do modelo CONSTAnT-MR, as dimensões espaço-temporais de cada elemento de contexto são analisadas para identificar em qual granularidade da trajetória o contexto deve ser incorporado. Como o nível de energia e a velocidade são contextos que podem ser capturados a cada intervalo de tempo especificado, eles são associados com os pontos da trajetória. Assim é possível representar a evolução destas informações ao longo da vida da trajetória. Já o estado da fonte, por ser uma informação esporádica, é modelada como um evento.

A partir das informações da trajetória, é possível gerar elementos de contexto para as sub-trajetórias com funções de agregação. Um novo elemento de contexto gerado a partir do nível de bateria é o custo total de uma sub-trajetória, calculado a partir da equação (5.1):

$$s_{custoTotal} = \sum_{i=1}^{n-1} p_i.nivel\_energia - p_{i+1}.nivel\_energia \quad onde \quad p \in Ss.P \tag{5.1}$$

O custo total é calculado a partir da soma das diferenças do nível de energia entre dois pontos consecutivos p_i e p_{i+1} para os n pontos da sub-trajetória Ss. Como cada sub-trajetória representa a execução (com sucesso ou abortada) de uma tarefa, o custo total irá representar o custo de cada tarefa executada durante a aplicação. Esta informação será importante mais a frente.

A velocidade do robô, associada com cada ponto da trajetória, é utilizada para calcular a velocidade média da sub-trajetória, calculada a partir da equação (4.6) definida na seção 4.1.3. A velocidade média permite avaliar como a velocidade do robô foi alterada em cada

execução das tarefas. A sub-trajetória também irá armazenar as informações referentes à tarefa, como seu identificador e o estado final (se foi executada com sucesso ou abortada).

Com os elementos de contextos, árvore de comportamento, mapa semântico e trajetória semântica definidos, a aplicação foi executada em um ambiente de simulação para gerar o histórico necessário para os próximos passos.

5.1.1 Análise dos Resultados

O experimento realizado colocou como valor fixo do nível de bateria em 15%. Em um total de 611 tarefas executadas, o robô executou com sucesso 486 tarefas, falhou 60 e recarregou 65 vezes. As 60 falhas gastaram 370 unidades de energia do robô.

Para resolver o problema da recarga autônoma, a análise dos resultados é focada no desperdício de energia. Considera-se desperdício de energia toda a energia gasta em uma tarefa que foi abortada, uma vez que ela não cumpriu seu papel. Para isso é necessário analisar como os elementos de contexto se comportam e sua relação com o estado final da tarefa. Uma das maneiras de realizar esta análise é através da correlação de Pearson, citada na seção 4.3.2. A Tabela 7 ilustra as correlações entre as variáveis.

Tabela 7 – Correlação de Pearson entre os elementos de contexto e a execução da tarefa

Variável	Velocidade	Energia	Estado da Fonte	Execução da tarefa
Velocidade	1.0	-0.01	0.001	-0.06
Energia	-0.01	1	0.002	0.41
Estado da Fonte	0.001	0.002	1	0.001
Execução da tarefa	-0.06	0.41	0.001	1

De acordo com a Tabela 7, a variável que está mais associada à execução da tarefa, que determina se foi executada com sucesso ou abortada, é o nível de energia. Uma vez que o nível de energia determina quando uma tarefa deve ser abortada, tal associação é óbvia para humanos. Entretanto, caso algum outro elemento de contexto também influenciasse na execução final da tarefa, ele teria uma correlação elevada com a variável de execução da tarefa.

A partir do histórico de execução, é possível extrair mais elementos de contexto. Por exemplo, analisando o custo de cada tarefa executada com sucesso, ou seja, o elemento de contexto de custo total de cada sub-trajetória executada com sucesso, é possível estimar o custo

Tabela 8 – Exemplo de informações semânticas sobre sub-trajetórias

Tid	Sid	Ponto Inicial	Tarefa	Status	Custo
79	4662	Nodo B	Ir para A	Sucesso	11.70
79	4663	Nodo A	Ir para B	Sucesso	10.75
79	4664	Nodo B	Ir para A	Sucesso	11.32
79	4665	Nodo A	Ir para B	Falha	13.03
79	4666	Nodo I	Ir Recarregar	Sucesso	10.33
79	4667	Nodo E	Ir para A	Sucesso	6.72
79	4668	Nodo A	Ir para B	Sucesso	10.61
79	4669	Nodo B	Ir para A	Sucesso	10.65
79	4670	Nodo A	Ir para B	Sucesso	8.93
79	4671	Nodo B	Ir para A	Sucesso	11.80

de cada tarefa. Em um ambiente estático, onde não há mudanças no mapa como obstáculos desconhecidos ou passagens obstruídas, o custo de cada tarefa tem tendência a permanecer estável. A Tabela 8 mostra algumas das informações obtidas pela trajetória semântica. Essas informações são então utilizadas para refatorar o sistema, modificando as árvores de comportamento do robô.

O custo estimado de cada tarefa é um novo elemento de contexto que analisa a média das últimas execuções de sucesso do robô, analisando o ponto de partida e o ponto de chegada. Esse custo pode ser recalculado a cada n interações, definidas pelo usuário. O maior problema identificado na árvore de comportamento é que o robô começa a realizar uma tarefa de deslocamento sem saber se possui energia suficiente para completá-la. Isso resulta em tarefas que são abortadas quando atingem o limite estabelecido pelo algoritmo. O histórico de execução permite reestruturar a árvore de comportamento, resultando na Figura 59.

Essa nova árvore de comportamento incorpora os elementos de contexto de previsão de custo de cada tarefa. A nova árvore da 'aplicação' agora é um nodo de sequência com dois filhos: 'PatrulharA' e 'patrulharB', que executam o comportamento de patrulhar o quarto A e B respectivamente. Cada árvore de patrulha agora é responsável por executar a checagem do nível de energia para aquela tarefa específica.

O comportamento 'PatrulharA' é implementado como um nodo

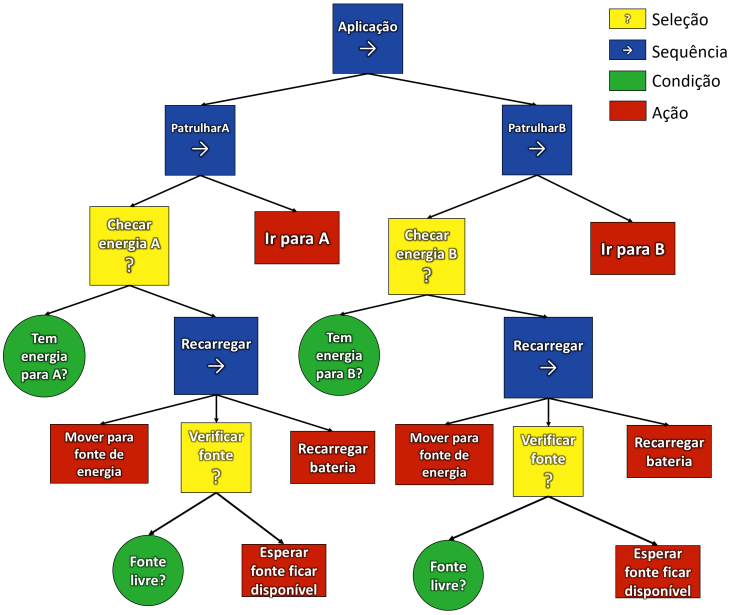


Figura 59 – Nova árvore de comportamento após refatoração da árvore

de sequência com dois filhos. O primeiro filho é o nodo de seleção 'Checar energia A'. Esse nodo vai retornar sucesso caso um dos seus filhos retornem sucesso. O primeiro filho é o nodo de condição 'Tem energia para A?'. Esse nodo de condição checa se o nível de energia atual do robô é o suficiente para executar a tarefa A e retornar para a fonte de energia. Esse valor é baseado no histórico de execução do robô, onde o algoritmo identifica a média do valor gasto para executar a tarefa A nas mesmas condições de contexto atuais do robô. Caso retorne sucesso, como o nodo 'Checar energia A' é um nodo de seleção, ele também retorna sucesso, e então o nodo de sequência de 'PatrulharA' irá executar seu próximo filho, a ação 'Ir para A'. Caso a condição 'Tem energia para A?' retorne falha, então o nodo de seleção 'Checar energia A' vai executar o nodo de sequência 'Recarregar'. Esse nodo tem o comportamento semelhante à arvore anterior, ilustrada na Figura 58. O comportamento 'PatrulharB' tem o mesmo funcionamento, diferenciando apenas na condição 'Tem energia para B?' onde ele checa se o robô tem energia o suficiente para executar a tarefa B.

Com a nova árvore de comportamento que incorpora a previsão de custo de cada tarefa, o robô irá sempre checar se tem energia o

suficiente para executar cada tarefa antes de executá-la de fato. Caso negativo, a árvore já encaminha o robô para o comportamento de recarga. Caso positivo, o comportamento segue como planejado, indo para o próximo ponto. Desta maneira, o robô só executa uma tarefa caso ele tenha energia suficiente para terminá-la, segundo o histórico de execução. O valor de previsão de custo de cada tarefa pode ser continuamente atualizado com as novas informações armazenadas no histórico de execução.

Esta tese propõe duas abordagens diferentes para o gerenciamento de energia de robôs móveis a partir de elementos de contexto: aplicando o framework para definir novos comportamentos através da árvore de comportamento e a abordagem fuzzy. Para destacar as vantagens e desvantagens de cada abordagem, foram realizados novos experimentos comparando a nova árvore de comportamento, que faz a previsão de custo antes da execução, com a abordagem tradicional (limite fixo) e a abordagem fuzzy. Os experimentos foram realizados considerando o mapa da Figura 55 em ambiente estático (sem mudanças no ambiente) e dinâmico (com mudanças no ambiente).

5.1.2 Problema da recarga autônoma com mapa estático

Este experimento compara o desempenho das três abordagens de gerenciamento de energia em um mapa estático. A abordagem de limite fixo e de fuzzy utilizam a mesma árvore de comportamento inicial ilustrada na Figura 58 com uma pequena modificação na condição *checar energia*. Enquanto o limite fixo verifica apenas se o valor atual da energia está acima do limite definido, o fuzzy aplica a abordagem definida na seção 4.4.

Utilizando o histórico da execução das tarefas, o limite fixo foi definido em 15% do nível da bateria. Como cada tarefa consumia aproximadamente 10% da bateria, um limite de 15% é o suficiente para garantir que o robô retorne à fonte de energia durante a execução de cada tarefa. Um valor menor poderia resultar no robô ficar sem energia antes de chegar na fonte. Já um valor maior poderia abortar tarefas que o robô tem energia para executar.

A abordagem fuzzy utiliza as regras e funções de pertinência definidas na seção 4.4. Como esta aplicação possui somente uma fonte de energia, a distância será sempre calculada considerando a posição do robô e a localização da fonte R. A função de pertinência da figura 52 já foi definida considerando os dados de histórico desta aplicação de

patrulha.

Por fim, a nova árvore de comportamento já considera o histórico de execução do experimento anterior para prever os valores de custo de cada tarefa.

Este experimento analisa o custo das tarefas que foram abortadas em cada abordagem diferente. O objetivo desta análise é comparar como cada algoritmo gerencia a energia consumida e como isso impacta nas tarefas abortadas. A energia gasta em tarefa abortada é considerada energia desperdiçada, uma vez que o robô poderia utilizar esta energia em tarefas executadas com sucesso. Os dados analisados são o percentual de tarefas abortadas, a quantidade de energia desperdiçada nas tarefas abortadas e a distância entre o robô o destino quando a tarefa foi abortada. Como a nova árvore de comportamento só executa uma tarefa quando possui energia suficiente, essa abordagem não teve nenhuma tarefa abortada.

O limite fixo e o fuzzy apresentaram quase a mesma porcentagem de tarefas abortadas (aproximadamente 10%), ilustrada na Figura 60. Isso indica que, para as configurações do experimento, o gerenciamento de energia de ambos algoritmos não apresentam diferenças marcantes no número de tarefas abortadas.

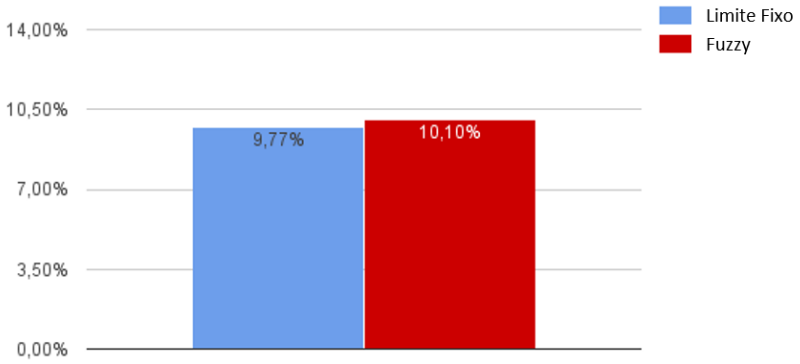


Figura 60 – Porcentagem das tarefas abortadas nas abordagens fuzzy e limite fixo no mapa estático

A maior diferença entre os algoritmos está na média de energia desperdiçada em cada um. Apesar de abortar uma quantidade semelhante de tarefas, o limite fixo desperdiçou cerca de 95% a mais de energia que o fuzzy (média de 5,2% no limite fixo contra 2,7% no fuzzy), como ilustra a Figura 61.

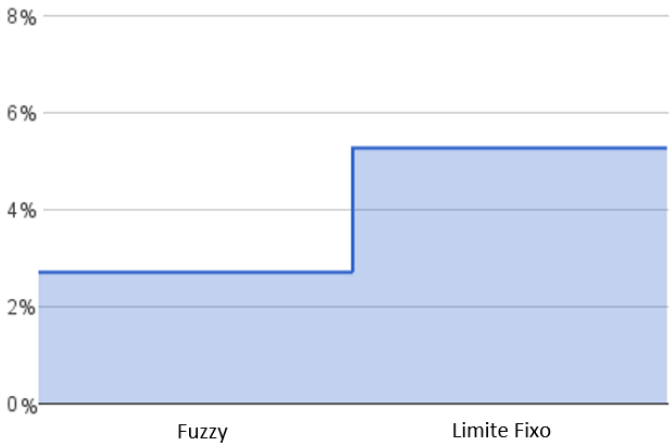


Figura 61 – Porcentagem da energia desperdiçada nas abordagens fuzzy e limite fixo no mapa estático

Esta diferença pode ser explicada analisando a distância do robô para o destino quando abortou uma tarefa, ilustrada na Figura 62. Enquanto a abordagem do limite fixo observa apenas o nível de energia para decidir se deve abortar a tarefa, a abordagem fuzzy verifica também a distância do robô para o destino final. Quanto mais o robô se desloca, mais ele gasta energia. Logo, as tarefas que foram abortadas a menos de 2 metros de distância do destino desperdiçam muito mais energia do que tarefas que foram abortadas a mais de 10 metros de distância do destino.

Segundo a Figura 62, mais de 28% das tarefas no limite fixo foram abortadas a menos de 2 metros do destino, enquanto na abordagem fuzzy nenhuma tarefa foi abortada a menos de 2 metros. Por outro lado, enquanto apenas 5% das tarefas no limite fixo foram abortadas a mais de 10 metros do destino, 25% das tarefas foram abortadas na abordagem fuzzy nesta mesma distância. Esse comportamento na abordagem fuzzy pode ser explicado a partir das regras 3 e 4 da seção 4.4. A regra 3 determina que se a energia está no nível cuidado e a distância para a fonte não é longe, o robô deve terminar a tarefa, impedindo que tarefas próximas de serem completadas sejam abortadas. Já a regra 4 determina que, se o nível da energia é cuidado e a distância para a tarefa é longe, é mais vantajoso para o robô abortar a tarefa e ir recarregar, resultando nas 25% de tarefas abortadas acima de 10 me-

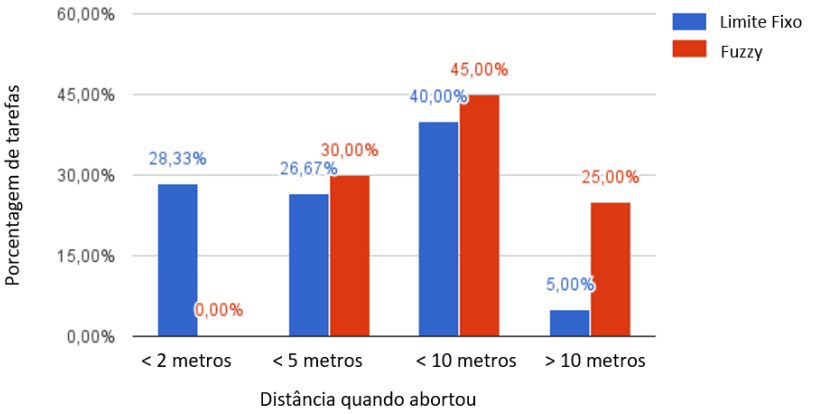


Figura 62 – Porcentagem das tarefas abortadas por distância do objetivo (menos de 2 metros, entre 2 e 5 metros, entre 5 e 10 metros e mais de 10 metros) no mapa estático

tros de distância do destino. Como o limite fixo não avalia a distância, ele aborta tarefas próximas ou distantes do destino, apenas verificando o nível de energia.

O experimento no mapa estático demonstrou a nova árvore de comportamento resultante da análise do histórico de execução não aborta nenhuma tarefa por verificar previamente o nível de energia atual com a previsão de gasto de cada tarefa, apenas executando as tarefas que tem energia suficiente para terminar. Este experimento também mostrou que, no cenário estático, a abordagem fuzzy tem um gasto de energia mais inteligente que a abordagem de limite fixo, desperdiçando menos energia nas tarefas abortadas. Entretanto, poucas aplicações no mundo real são executadas em um mapa estático, podendo sofrer mudanças a partir de obstáculos não previstos ou caminhos bloqueados. O próximo experimento irá comparar as abordagens em um mapa dinâmico, onde o custo de execução das tarefas pode ser alterado de acordo com o novo estado do mapa.

5.1.3 Problema da recarga autônoma com mapa dinâmico

Este experimento utiliza as mesmas configurações do experimento anterior mas com um mapa levemente diferente. Primeiramente,

não existe mais a passagem 6 da Figura 56. Outra diferença é que agora existem dois obstáculos que podem bloquear duas passagens, forçando o robô a fazer um caminho mais longo para executar cada tarefa. Logo, também serão comparados os mesmos dados entre as abordagens.

O comportamento do mapa dinâmico está ilustrado nas Figuras 63 e 64. Cada um dos quartos das tarefas (A e B) tem duas entradas, representadas na Figura 56 pelos nodos 2, 1, 5 e 7. Para facilitar a compreensão, estes nodos serão identificados como A_{e1}, A_{e2}, B_{e1}, B_{e2} respectivamente. O caminho mais curto e, por consequência, de menor custo, utiliza as entradas A_{e1} e B_{e1}, ilustrado na Figura 63. Os obstáculos O_1 e O_2 bloqueiam as entradas A_{e1} e B_{e1}, respectivamente, em momentos randômicos.

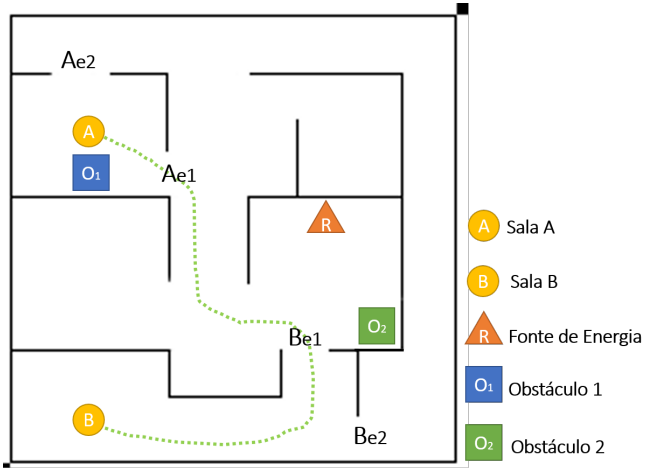


Figura 63 – Caminho do robô no mapa dinâmico sem passagens bloqueadas.

Quando as entradas são bloqueadas, o robô é obrigado a tomar um caminho mais longo, ilustrado na Figura 64. O objetivo do experimento é analisar o comportamento do gerenciamento de energia quando as tarefas aumentam de custo.

Ao contrário do experimento anterior, desta vez a nova árvore de comportamento obteve o pior resultado entre as três abordagens. Como cada tarefa pode ter custo diferente durante as execuções, a previsão de custo acaba falhando, resultando no robô ficar sem energia no meio do caminho. Apesar de ser possível recalcular a previsão de custo com a nova execução, o robô não tem informação prévia dos obstáculos no

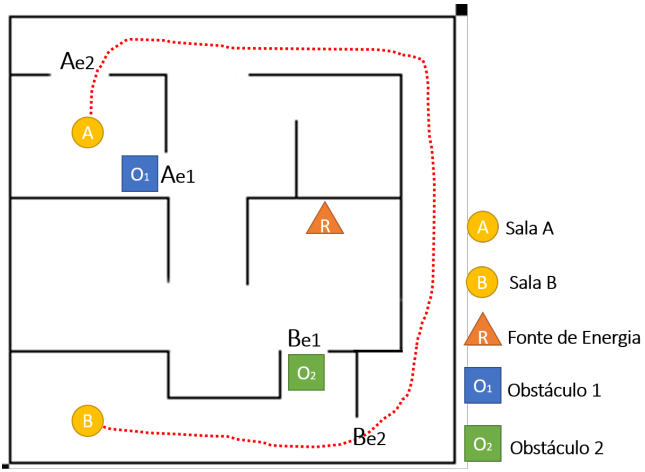


Figura 64 – Caminho do robô no mapa dinâmico com as passagens A_{e1} e B_{e1} bloqueadas.

caminho, podendo falhar na previsão do novo custo. Logo, enquanto em um ambiente estático a nova árvore de comportamento tem o melhor resultado, ela sozinha não é apropriada para um ambiente dinâmico, uma vez que ela não se adapta bem ao custo de tarefa variável.

Comparando as abordagens de limite fixo e de fuzzy, agora para o ambiente dinâmico, ambas novamente obtiveram a mesma média de tarefas abortadas (cerca de 12%), como ilustra a Figura 65.

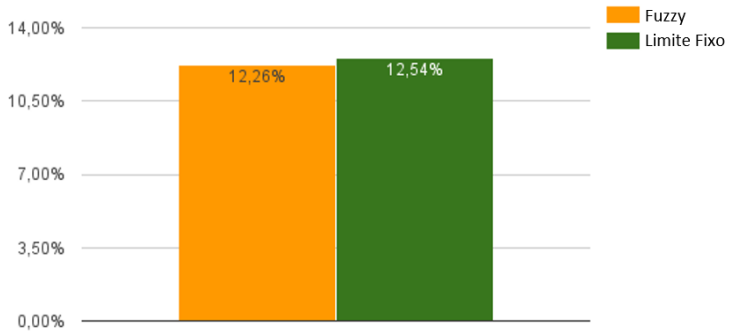


Figura 65 – Porcentagem das tarefas abortadas nas abordagens fuzzy e limite fixo no mapa dinâmico

A energia desperdiçada nas tarefas abortadas, ilustrada na Figura 66, manteve o mesmo comportamento do mapa estático, mas desta vez com a abordagem de limite fixo desperdiçando 40% a mais de energia que a fuzzy (média de 6,7% para o limite fixo e 4,7% para o fuzzy).

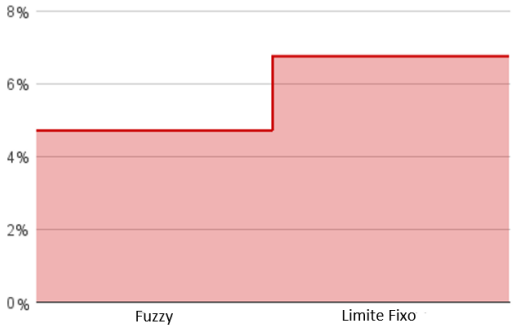


Figura 66 – Porcentagem da energia desperdiçada nas abordagens fuzzy e limite fixo no mapa dinâmico

Já para a distância do robô para com o destino quando abortou as tarefas, ilustrada na Figura 67, também teve um comportamento similar com o mapa estático.

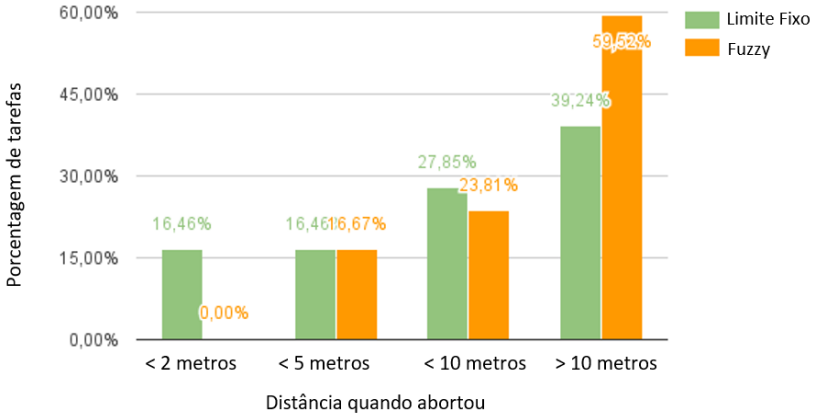


Figura 67 – Porcentagem das tarefas abortadas por distância do objetivo (menos de 2 metros, entre 2 e 5 metros, entre 5 e 10 metros e mais de 10 metros) no mapa dinâmico

Novamente, enquanto o limite fixo abortou 16% das tarefas a menos de 2 metros do destino, o fuzzy não abortou nenhuma tarefa nesta distância. Como a distância para a tarefa aumenta consideravelmente quando os caminhos estão bloqueados, 60% das tarefas abortadas com o gerenciamento fuzzy estavam a mais de 10 metros de distância do destino, enquanto 40% das tarefas abortadas do limite fixo também estiveram a esta mesma faixa de distância. Enquanto o motivo para o gerenciamento fuzzy continua o mesmo (a aplicação das regras de inferência), para o limite fixo este valor está relacionado com o robô gastando mais energia enquanto percorre o maior caminho, atingindo o limite mais rápido.

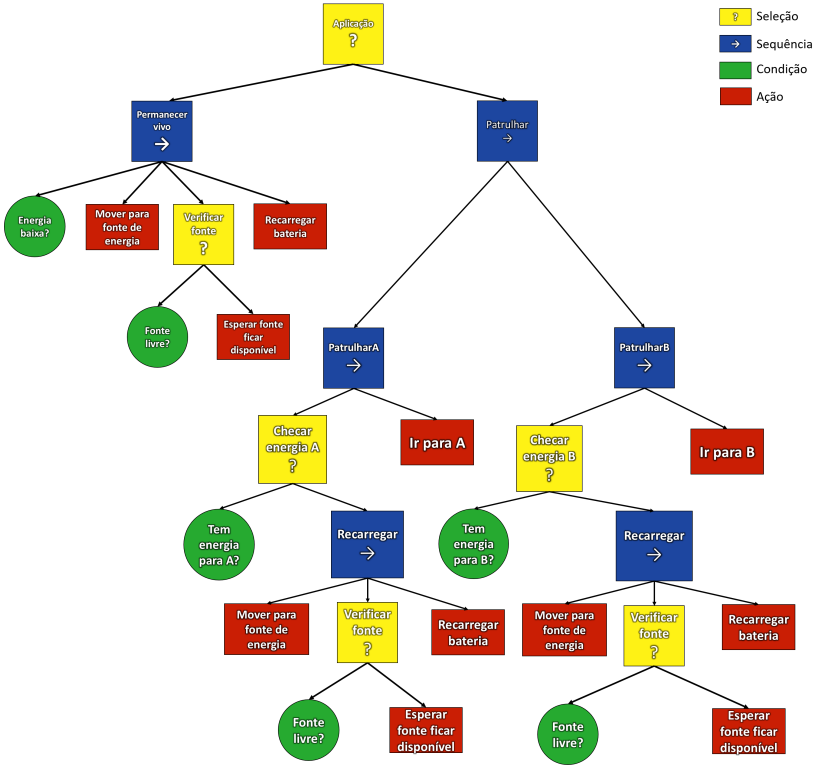


Figura 68 – Árvore de comportamento unificando a abordagem fuzzy e de previsão de custo

Um ambiente dinâmico é mais parecido com o mundo real, uma vez que podem surgir obstáculos desconhecidos ou passagens obstruí-

das durante o trajeto do robô. Este experimento demonstra que a abordagem fuzzy se adapta melhor em ambientes dinâmicos do que o limite fixo e a nova árvore de comportamento. Para solucionar ambos os cenários (estático e dinâmico), é possível integrar as árvores de comportamento a partir de uma composição de seleção e utilizar as abordagens de previsão de custo e fuzzy ao mesmo tempo, ilustrada na Figura 68.

Enquanto o robô utiliza dados históricos para prever se tem energia suficiente para executar a tarefa, no comportamento de patrulha, o gerenciamento de bateria fuzzy funciona em paralelo, sendo ativado quando a bateria atingir o nível cuidado. Para a previsão de custo também acompanhar as mudanças do mapa, é possível calcular o custo de cada tarefa levando em consideração o estado atual do mapa. Este calculo é realizado nos nodos de condição 'Energia A OK' e 'Energia B OK'.

5.2 EXPERIMENTO 2: TRANSPORTE DE CARGA

Enquanto o primeiro experimento explorou o problema da recarga autônoma, o segundo experimento usa um mapa semântico em conjunto com a árvore de comportamento em uma aplicação de transporte de carga. Nesse experimento será apenas modelado o problema, sem a execução. O mapa da aplicação é ilustrado na Figura 69.

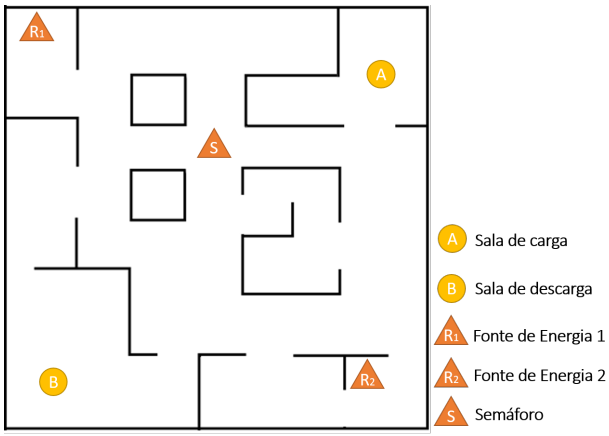


Figura 69 – Mapa da aplicação de transporte de carga

O robô deve levar as cargas do ponto A até o ponto B da Figura 69. Além disso, existem duas fontes de energia R_1 e R_2. O mapa também conta com um semáforo S que controla o fluxo do cruzamento, possibilitando a passagem no sentido horizontal ou vertical por vez. Além disso, existe um usuário que se locomove pelo mapa e pode, a qualquer momento, solicitar a presença do robô em sua localização.

Novamente, o mapa métrico foi transformado em um mapa semântico onde cada vértice tem um dos três tipos: quarto, corredor, passagem de entrada e objeto. A Figura 70 ilustra a abstração do mapa métrico. Os vértices representados por um círculo são do tipo quarto ou corredor, quadrado são do tipo passagem e triângulo do tipo objeto. O objeto R é a fonte de energia.

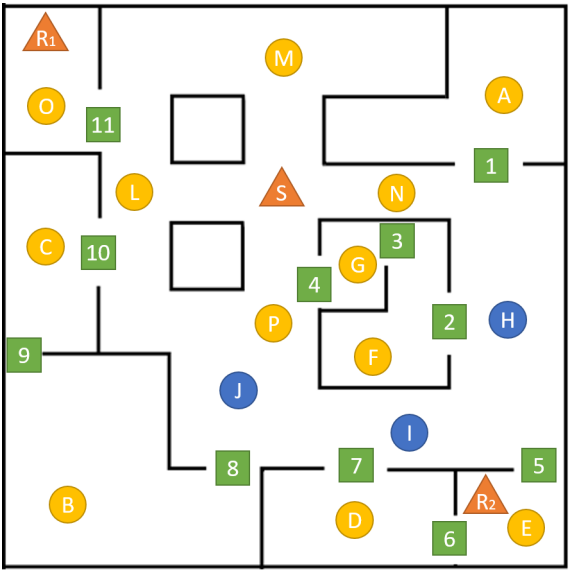


Figura 70 – Mapa métrico da aplicação de transporte de carga

Por fim, o mapa topológico resultante está ilustrado na Figura 71. O custo inicial das arestas é calculado através da equação (4.4) considerando $v_{max} = 5.0$ m/s, $\alpha = 1$ e $\beta = 0$.

Os elementos de contexto presentes na aplicação estão resumidos na Tabela 9.

Alguns elementos de contexto observados alteram o estado do mapa. Por exemplo, uma vez que os nodos em destaque azul (nodos J, I e H) representam partes do mapa em área aberta, ou seja, região

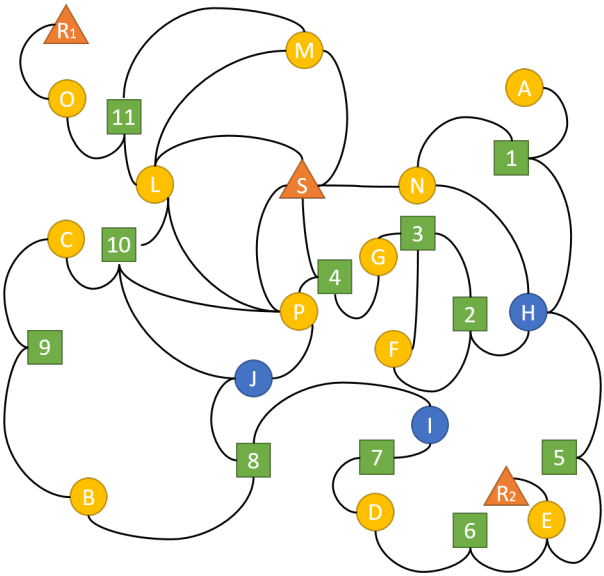


Figura 71 – Mapa topológico da aplicação de transporte de carga

Tabela 9 – Elementos de contexto da aplicação de transporte de carga e sua classificação nas dimensões WWW

Contexto	What	Where	When
Nível de Energia	Robô	Local	Temporal
Estado da fonte	Ambiente	Local	Temporal
Velocidade do robô	Robô	Local	Temporal
Clima atual	Ambiente	Global	Temporal
Requisição do usuário	Usuário	Local	Temporal
Semáforo	Ambiente	Local	Periódico
Posição do usuário	Usuário	Local	Temporal
Posição do robô	Robô	Local	Temporal
Tarefa Atual	Robô	Global	Temporal
Número de cargas	Ambiente	Local	Temporal
Posição da fonte	Ambiente	Local	Permanente
Tipo de ambiente	Ambiente	Local	Permanente

sem cobertura no teto, o elemento de contexto de clima altera o estado destes nodos. O estado do semáforo determina se as transições entre os vértices L e N estão ativas ou se as transições entre os vértices M e P estão ativas. O número atual de cargas descreve quantas cargas estão no nodo A e precisam ser levadas ao nodo B.

Definido o mapa semântico, é necessário criar a árvore de comportamento do robô. Para executar a tarefa de transporte de carga o robô deve primeiro verificar se existe alguma carga para ser transportada no quarto A. Caso positivo, ele deve pegar a carga e levar até o quarto B. Chegando no quarto B, ele deve depositar a carga. Esse comportamento é resumido na Figura 72, onde o comportamento 'Transportar carga' é implementado através de um nodo de sequência, logo, cada nodo filho deve retornar sucesso para que o próximo seja executado.

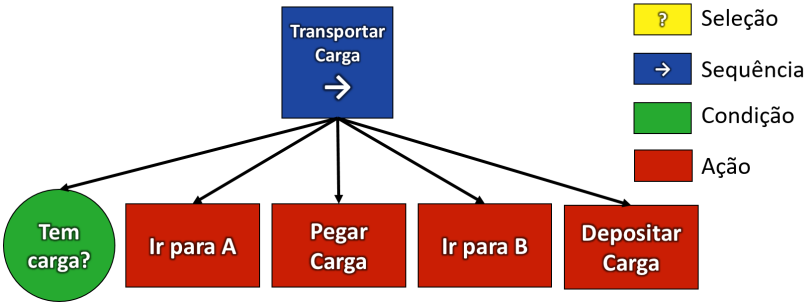


Figura 72 – Árvore de Comportamento para transporte da carga

Depois de definir a árvore de comportamento da tarefa é necessário criar as árvores de comportamento para tratar os elementos de contexto. Para o nível de energia será utilizada a mesma árvore de comportamento 'Permanecer vivo' ilustrada na Figura 58. Uma outra árvore de comportamento é necessária para tratar das requisições do usuário.

Para atender a requisição do usuário, primeiramente o robô deve verificar se existe alguma tarefa atual sendo executada ou se ele está livre. Caso ele esteja executando o comportamento de transporte de carga, ele não estará livre. Nessa situação, se o robô estiver carregando alguma carga, ele deve descarregar antes de atender ao usuário e abortar a tarefa atual. Finalmente, o robô se desloca até a posição do usuário e realiza a tarefa requisitada para, depois, retornar à tarefa anterior. Esse comportamento é ilustrado na Figura 73.

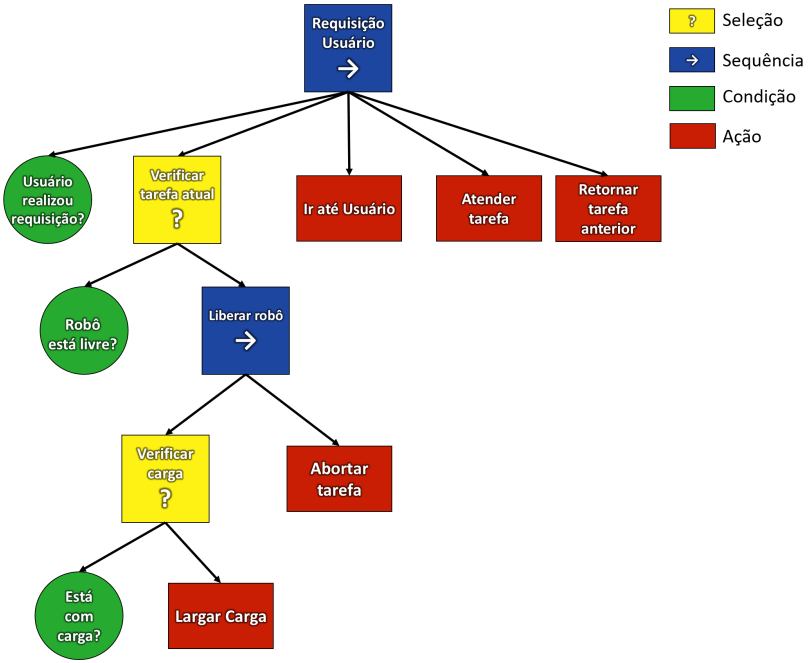


Figura 73 – Árvore de Comportamento para requisição do usuário

O último passo é integrar as árvores de comportamento a partir de uma composição. De maneira similar ao experimento anterior, o comportamento de permanecer vivo continua tendo prioridade maior, uma vez que sem energia o robô fica impossibilitado de realizar qualquer tarefa. As requisições do usuário também têm prioridade em relação à tarefa de transporte de carga. Portanto, a primeira sub-árvore é de *permanecer vivo*, seguida pela *requisição usuário* e finalizando com *transportar carga*. A composição de seleção das árvores é ilustrada na Figura 74.

A partir do histórico de navegação é possível identificar o comportamento do robô com cada elemento de contexto. Por exemplo, o robô gasta mais energia ao navegar nos nodos I, J e H em clima de chuva? Qual o caminho mais rápido entre A e B quando o semáforo está fechado para os nodos N e L? Qual caminho é mais eficiente quando o robô está transportando uma carga? E quando não está?

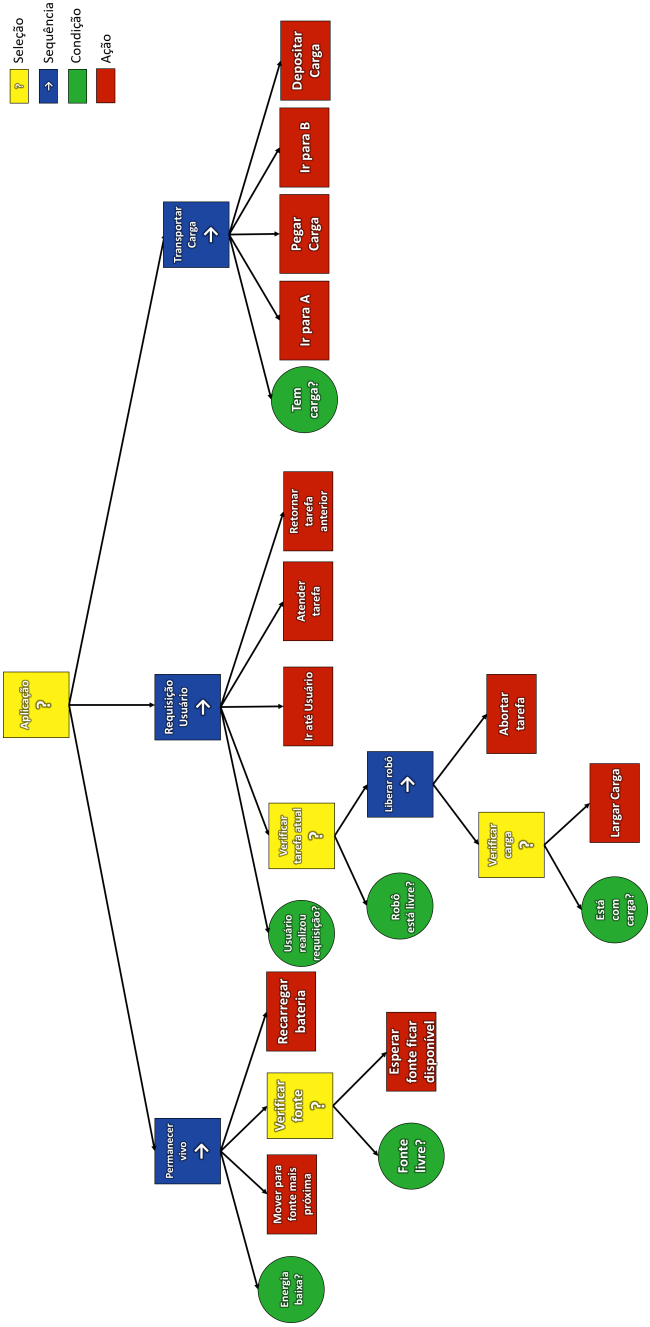


Figura 74 – Árvore de Comportamento da aplicação de transporte de carga

5.3 SÍNTESE DO CAPÍTULO

Este capítulo apresentou o uso do framework ConBINaMoR em duas aplicações diferentes. No primeiro experimento, de patrulha, o framework foi aplicado em todos processos e etapas definidos nas figuras 40 e 54. Depois da primeira execução e da análise de dados, a árvore de comportamento foi modificada, incluindo novos comportamentos para o robô.

Um segundo passo foi comparar a abordagem de limite fixo, fuzzy e da nova árvore de comportamento para o problema de recarga autônoma em um mapa estático e um mapa dinâmico. O resultado demonstrou que a nova árvore foi a melhor abordagem para o mapa estático e a abordagem fuzzy foi a melhor abordagem para o mapa dinâmico, resultando em uma árvore que integra ambas abordagens.

Já a segunda aplicação foi aplicado apenas os procedimentos de pré-definição e definição em uma aplicação de transporte de carga. A árvore de comportamento resultante da aplicação do framework considera mais elementos de contexto do que a aplicação anterior como clima, requisição do usuário, semáforo, número de cargas, etc.

6 CONCLUSÃO

Seres humanos navegam por um ambiente a partir do uso de seus sentidos para capturar informações de contexto sobre o ambiente (HUANG; LIU, 2004). É possível aprimorar a navegação humana usando sistemas computadorizados de navegação que faça uso de diversas informações disponíveis sobre o ambiente para criar rotas de navegação mais apropriadas ao ambiente atual (AFYOUNI et al., 2013). Esta mesma estratégia pode ser aplicada na robótica móvel.

Esta tese tratou o problema da navegação inteligente de robôs móveis através da incorporação de elementos de contexto utilizando árvore de comportamento, mapa semântico, lógica fuzzy, trajetória semântica e técnicas de mineração de dados. O conjunto destas estratégias foi implementado no framework ConBINaMoR.

Para modelar os elementos de contexto, foram definidas as dimensões WWW (What, Where and When) que caracterizam cada contexto em diversos aspectos. Além disto, foi proposto o modelo CONSTAnT-MR, uma adaptação do modelo CONSTAnT (BOGORNY et al., 2014) para robótica móvel. A modelagem de contexto através das dimensões WWW facilitou a criação da árvore de comportamento para a tomada de decisão, uma vez que é possível descrever as características de um contexto de maneira mais detalhada em comparação a outras alternativas da literatura.

A partir do histórico de navegação, o framework permite extrair informações importantes sobre o comportamento do robô, auxiliando a identificar possíveis falhas e quais elementos de contexto impactam na execução da tarefa. O histórico também auxilia na atualização do mapa semântico, refletindo os custos reais de deslocamento entre os vértices do mapa dependendo do seu estado atual, aprimorando o planejamento de trajetória.

Em paralelo, foi tratado o problema da recarga autônoma através de um sistema de gerenciamento de energia fuzzy, que se mostrou mais eficiente em evitar o desperdício de energia em comparação com propostas clássicas. A aplicação do framework ConBINaMoR produziu um gerenciamento de energia melhor para ambientes estáticos enquanto o gerenciamento fuzzy comprovou ser melhor em ambientes dinâmicos.

Experimentos demonstraram que a aplicação do framework com simples elementos de contexto aprimoram a execução inicial do robô através do refinamento da árvore de comportamento utilizando o histórico de navegação. A estrutura hierárquica do CONSTAnT-MR per-

mite uma maior facilidade de compreender os dados e extrair informações relevantes do banco de dados.

A estrutura proposta abre uma série de possibilidades mais intuitivas para incorporar informações que melhoram a navegação de robôs móveis e aproximam-se da movimentação do seres humanos em ambientes conhecidos ou não. O framework ConBINaMoR abrange desde a definição e caracterização dos elementos de contexto, ao planejamento do caminho com mapa semântico, tomada de decisão com árvore de comportamento, organização de informações com trajetória semântica e o modelo CONSTAnT-MR, armazenamento do histórico de execução e extração de informações, resultando na melhoria da navegação inteligente.

O trabalho desta tese possui limitações em sua aplicação. Problemas comuns da robótica móvel, como a captura de informações e fusão de sensores, não foi tratado nesta tese. Assumiu-se a premissa de que as informações trabalhadas já sofreram um processo de filtragem dos dados. Não foi definida uma arquitetura de hardware do robô, assumindo que qualquer robô com qualquer arquitetura teria a capacidade de executar o framework. O trabalho também não abordou a questão de otimização de memória e processamento, fatores muito importante para aplicações em robótica mas que ficaram fora do escopo da proposta. Esta tese se focou em propor uma nova estratégia para a navegação inteligente de robôs móveis incorporando elementos de contexto juntamente com trajetória semântica, árvore de comportamento, mapa semântico e lógica fuzzy, abstraindo as demais áreas relacionadas à problemática de robótica móvel.

Os estudos deste trabalho resultaram nas seguintes publicações:

- Rinzivillo, Salvatore ; de Lucca Siqueira, Fernando ; Gabrielli, Lorenzo ; Renso, Chiara; Bogorny, Vania . Where Have You Been Today? Annotating Trajectories with DayTag. Lecture Notes in Computer Science. 13ed.: Springer Berlin Heidelberg, 2013, v. , p. 467-471 (RINZIVILLO et al., 2013)

- Siqueira, F. L.; De Pieri, E. R. . A Context-Aware Approach to the Navigation of Mobile Robots. In: Simpósio Brasileiro de Automação Inteligente, 2015, Natal. Anais do Simpósio Brasileiro de Automação Inteligente, 2015 (SIQUEIRA; PIERI, 2015)

- Raia, H. F. ; Siqueira, F. L. ; Plentz, P. D. M. . Using a Deadline Missing Prediction Mechanism and Recovery Actions to Avoid Deadline Losses. In: International Conference on Computers and

Their Applications, 2016, Las Vegas. Proceedings of the 31st International Conference on Computers and Their Applications, 2016. p. 305-310 (RAIA; SIQUEIRA; PLENTZ, 2016)

- Siqueira, F. L.; Plentz, P. D. M.;De Pieri, E. R. . A fuzzy approach to the autonomous recharging problem for mobile robots. In: 2016 12th International Conference on Natural Computation and 13th Fuzzy Systems and Knowledge Discovery (ICNCFSKD), 2016, Changsha. 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD). p. 1065 (SIQUEIRA; PLENTZ; PIERI, 2016a)

- Siqueira, F. L.; Plentz, P. D. M.;De Pieri, E. R. . Semantic Information Applied to Autonomous Navigation of Mobile Robots. In: International Conference on Computer Systems and Applications AICCSA 2016, Agadir, Marrocos. 2016 13th ACS/IEEE International Conference on Computer Systems and Applications AICCSA 2016 (SIQUEIRA; PLENTZ; PIERI, 2016b)

6.1 TRABALHOS FUTUROS

O trabalho desta tese unificou os campos de robótica móvel com análise de trajetórias, abrindo perspectivas de diversos trabalhos futuros. Além disto, campos que não foram explorados na tese, como aprendizagem de máquina, podem se beneficiar do framework proposto. Dentre os trabalhos futuros podem se destacar:

- Criação de algoritmo de aprendizagem de máquina para sugestão de modificações na árvore de comportamento a partir do histórico de execução;

- Desenvolvimento de algoritmo de planejamento de caminho a partir do estado futuro do mapa semântico;

- Modificação da árvore de comportamento em tempo real;

- Utilização de restrições de deadline para planejamento de caminho e atualização do mapa semântico;

- Aplicar o framework para tratar aplicações de multi-robôs.

REFERÊNCIAS

AFYOUNI, I. et al. Context-aware modelling of continuous location-dependent queries in indoor environments. *Journal of Ambient Intelligence and Smart Environments*, IOS Press, v. 5, n. 1, p. 65–88, 2013.

AFYOUNI, I.; RAY, C.; CLARAMUNT, C. Spatial models for context-aware indoor navigation systems: A survey. *Journal of Spatial Information Science*, n. 4, p. 85–123, 2012.

AGRAWAL, R.; SRIKANT, R. et al. Fast algorithms for mining association rules. In: *Proc. 20th int. conf. very large data bases, VLDB*. [S.l.: s.n.], 1994. v. 1215, p. 487–499.

ALVARES, L. O. et al. A model for enriching trajectories with semantic geographical information. In: *ACM-GIS*. New York, NY, USA: ACM Press, 2007. p. 162–169.

ALVARES, L. O. et al. An algorithm to identify avoidance behavior in moving object trajectories. *J. Braz. Comp. Soc.*, v. 17, n. 3, p. 193–203, 2011.

AMATO, G. et al. Robotic ubiquitous cognitive ecology for smart homes. *Journal of Intelligent & Robotic Systems*, Springer Science & Business Media, v. 80, p. 57, 2015.

ANGRISANI, L. et al. Autonomous recharge of drones through an induction based power transfer system. In: IEEE. *2015 IEEE International Workshop on Measurements & Networking (M&N)*. [S.l.], 2015. p. 1–6.

BACCIU, D. et al. Learning context-aware mobile robot navigation in home environments. In: IEEE. *The 5th International Conference on Information, Intelligence, Systems and Applications (IISA 2014)*. [S.l.], 2014. p. 57–62.

BAGLIONI, M. et al. Towards semantic interpretation of movement behavior. In: SESTER, M.; BERNARD, L.; PAELKE, V. (Ed.). *AGILE Conf.* [S.l.]: Springer, 2009. (Lecture Notes in Geoinformation and Cartography), p. 271–288. ISBN 978-3-642-00317-2.

BAGNELL, J. A. et al. An integrated system for autonomous robotics manipulation. In: IEEE. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.], 2012. p. 2955–2962.

BAZIRE, M.; BRÉZILLON, P. Understanding context before using it. In: *Modeling and using context*. [S.l.]: Springer, 2005. p. 29–40.

BOGORNY, V.; KUIJPERS, B.; ALVARES, L. O. St-dmql: A semantic trajectory data mining query language. *International Journal of Geographical Information Science*, v. 23, p. 1245–1276, 2009.

BOGORNY, V. et al. Constant–a conceptual data model for semantic trajectories of moving objects. *Transactions in GIS*, Wiley Online Library, v. 18, n. 1, p. 66–88, 2014.

BRAZ, J. F.; BOGORNY, V. *Introdução a Trajetórias de Objetos Móveis: conceitos, armazenamento e análise de dados*. [S.l.]: Univille, 2012.

BUSCHKA, P. *An investigation of hybrid maps for mobile robots*. Tese (Doutorado) — Örebro University, Langhuset, Fakultetsgatan 1, 702 81 Örebro, Suécia, 2005.

CAO, H.; MAMOULIS, N.; CHEUNG, D. W. Discovery of periodic patterns in spatiotemporal sequences. *IEEE Trans. Knowl. Data Eng.*, v. 19, n. 4, p. 453–467, 2007.

CARBONI, E. M.; BOGORNY, V. Identificando comportamentos anômalos em trajetórias de objetos móveis. In: *Proceedings of the XII Simposio Brasileiro de Geoinformatica (GEOINFO)*. [S.l.: s.n.], 2011. p. 141–146.

CASSINIS, R. et al. Docking and charging system for autonomous mobile robots. *Technical Report. Department of Electronics for Automation, University of Brescia, Italy*, 2005.

COLLEDANCHISE, M.; OGREN, P. How behavior trees modularize robustness and safety in hybrid systems. In: *Intelligent Robots and Systems (IROS)*. [S.l.: s.n.], 2014.

CRAIG, J. J. *Introduction to robotics: mechanics and control*. [S.l.]: Pearson/Prentice Hall Upper Saddle River, NJ, USA:, 2005.

CROCKFORD, D. *The application/json media type for javascript object notation (json)*. [S.l.], 2006.

DAVISON, A. J. et al. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 29, n. 6, p. 1052–1067, 2007.

DEY, A. K. Understanding and using context. *Personal and ubiquitous computing*, Springer-Verlag, v. 5, n. 1, p. 4–7, 2001.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. *Numerische mathematik*, Springer, v. 1, n. 1, p. 269–271, 1959.

DRIANKOV, D.; SAFFIOTTI, A. *Fuzzy logic techniques for autonomous vehicle navigation*. [S.l.]: Physica, 2013.

ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: SIMOUDIS, E.; HAN, J.; FAYYAD, U. M. (Ed.). *Second International Conference on Knowledge Discovery and Data Mining*. [S.l.]: AAAI Press, 1996. p. 226–231.

FILETO, R. et al. Baquara: A holistic ontological framework for movement analysis using linked data. In: *Conceptual Modeling*. [S.l.]: Springer, 2013. p. 342–355.

FOUKARAKIS, M. et al. Combining finite state machine and decision-making tools for adaptable robot behavior. In: *Universal Access in Human-Computer Interaction. Aging and Assistive Environments*. [S.l.]: Springer, 2014. p. 625–635.

FOUNDATION, O. S. R. *Gazebo*. 2014. Acessado em 01/07/2014. <http://gazebosim.org/>.

FOUNDATION, O. S. R. *ROS.org | Powering the world's robots*. 2014. Acessado em 11/06/2014. <http://www.ros.org/>.

GALINDO, C. et al. Robot task planning using semantic maps. *Robotics and Autonomous Systems*, Elsevier, v. 56, n. 11, p. 955–966, 2008.

GARAGE, W. *Willow Garage*. 2009. `http://www.willowgarage.com/`. Acessado em 20/12/2016.

GIANNOTTI, G.; GIANNOTTI, F.; PEDRESCHI, D. *Mobility, data mining and privacy: Geographic knowledge discovery*. [S.l.]: Springer, 2008.

GROUP, T. P. G. D. *PostgresSQL: The world's most advanced open source database*. 2014. Acessado em 11/06/2014. <http://www.postgresql.org/>.

HART, P. E.; NILSSON, N. J.; RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, IEEE, v. 4, n. 2, p. 100–107, 1968.

HARTIGAN, J. A.; WONG, M. A. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, [Wiley, Royal Statistical Society], v. 28, n. 1, p. 100–108, 1979. ISSN 00359254, 14679876. <http://www.jstor.org/stable/2346830>.

HÖLZL, M.; GABOR, T. Reasoning and learning for awareness and adaptation. In: *Software Engineering for Collective Autonomic Systems*. [S.l.]: Springer, 2015. p. 249–290.

HUANG, B.; LIU, N. Mobile navigation guide for the visually disabled. *Transportation Research Record: Journal of the Transportation Research Board*, Trans Res Board, v. 1885, n. 1, p. 28–34, 2004.

ISLA, D. Handling complexity in the halo 2 ai. In: *Game Developers Conference*. [S.l.: s.n.], 2005. v. 12.

KALAKRISHNAN, M. et al. Stomp: Stochastic trajectory optimization for motion planning. In: IEEE. *2011 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2011. p. 4569–4574.

KANNAN, B. et al. The autonomous recharging problem: Formulation and a market-based solution. In: IEEE. *IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2013. p. 3503–3510.

KO, D. W.; YI, C.; SUH, I. H. Semantic mapping and navigation with visual planar landmarks. In: *Ubiquitous Robots and Ambient Intelligence (URAI), 2012 9th International Conference on*. [S.l.: s.n.], 2012. p. 255–258.

KOENIG, S.; LIKHACHEV, M.; FURCY, D. Lifelong planning aâ^—. *Artificial Intelligence*, v. 155, n. 1, p. 93 – 146, 2004. ISSN 0004-3702. <http://www.sciencedirect.com/science/article/pii/S000437020300225X>.

KOSTAVELIS, I.; GASTERATOS, A. Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems*, v. 66, p. 86 – 103, 2015. ISSN 0921-8890. <http://www.sciencedirect.com/science/article/pii/S0921889014003030>.

KRETZSCHMAR, H. et al. Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, SAGE Publications Sage UK: London, England, v. 35, n. 11, p. 1289–1307, 2016.

KURT, A.; ÖZGüNER Ümit. Hierarchical finite state machines for autonomous mobile systems. *Control Engineering Practice*, v. 21, n. 2, p. 184 – 194, 2013. ISSN 0967-0661. <http://www.sciencedirect.com/science/article/pii/S0967066112002031>.

LANG, D.; PAULUS, D. Semantic maps for robotics. In: *Proc. of the Workshop" Workshop on AI Robotics" at ICRA*. [S.l.: s.n.], 2014.

LAUBE, P.; KREVELD, M. van; IMFELD, S. Finding remo—detecting relative motion patterns in geospatial lifelines. In: *Developments in spatial data handling*. [S.l.]: Springer, 2005. p. 201–215.

LIM, G. et al. Ontology representation and instantiation for semantic map building by a mobile robot. In: LEE, S. et al. (Ed.). *Intelligent Autonomous Systems 12*. Springer Berlin Heidelberg, 2013, (Advances in Intelligent Systems and Computing, v. 193). p. 387–395. ISBN 978-3-642-33925-7. <http://dx.doi.org/10.1007/978-3-642-33926-4$_3$6>.

LIM, G. H. et al. Ontology representation and instantiation for semantic map building by a mobile robot. In: *Intelligent Autonomous Systems 12*. [S.l.]: Springer, 2013. p. 387–395.

LYNEN, S. et al. A robust and modular multi-sensor fusion approach applied to mav navigation. In: IEEE. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.], 2013. p. 3923–3929.

MARINO, A. et al. Behavioral control for multi-robot perimeter patrol: A finite state automata approach. In: IEEE. *IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2009. p. 831–836.

MARZINOTTO, A. et al. Towards a unified behavior trees framework for robot control. In: *IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.: s.n.], 2014.

MATHEW, N.; SMITH, S. L.; WASLANDER, S. L. Multirobot rendezvous planning for recharging in persistent tasks. *Robotics, IEEE Transactions on*, IEEE, v. 31, n. 1, p. 128–142, 2015.

MEGER, D. et al. Curious george: An attentive semantic robot. *Robotics and Autonomous Systems*, Elsevier, v. 56, n. 6, p. 503–511, 2008.

MEI, Y. et al. A case study of mobile robot's energy consumption and conservation techniques. In: IEEE. *12th International Conference on Advanced Robotics (ICAR'05)*. [S.l.], 2005. p. 492–497.

MOBILE, W. *Free Community-based Mapping, Traffic and Navigation App*. 2009. Acessado em 13/10/2014. <https://www.waze.com/>.

NETO, L. B. et al. Minicurso de sistema especialista nebuloso. *XXXVIII Simpósio Brasileiro de Pesquisa Operacional, Goiânia-GO*, 2006.

NÜCHTER, A.; HERTZBERG, J. Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, Elsevier, v. 56, n. 11, p. 915–926, 2008.

OH, S.; ZELINSKY, A.; TAYLOR, K. Autonomous battery recharging for indoor mobile robots. In: *Proceedings of the australian conference on robotics and automation*. [S.l.: s.n.], 2000.

OŚMIAŁOWSKI, B. On path planning for mobile robots: Introducing the mereological potential field method in the framework of mereological spatial reasoning. *Journal of Automation Mobile Robotics and Intelligent Systems*, v. 3, p. 24–33, 2009.

PACULA, M. *K-means clustering example (Python)*. 2011. Acessado em 20/12/2016. <http://blog.mpacula.com/2011/04/27/k-means-clustering-example-python/>.

PALMA, A. T. et al. A clustering-based approach for discovering interesting places in trajectories. In: WAINWRIGHT, R. L.; HADDAD, H. (Ed.). *SAC*. [S.l.]: ACM, 2008. p. 863–868. ISBN 978-1-59593-753-7.

PARENT, C. et al. Semantic trajectories modeling and analysis. *ACM Computing Surveys (CSUR)*, ACM, v. 45, n. 4, p. 42, 2013.

PINCIROLI, C. et al. Adaptation and awareness in robot ensembles: Scenarios and algorithms. In: *Software Engineering for Collective Autonomic Systems*. [S.l.]: Springer, 2015. p. 471–494.

PRONOBIS, A. *Semantic Mapping with Mobile Robots*. Tese (Doutorado) — Royal Institute of Technology, Stockholm, Sweden, 2011.

PRONOBIS, A.; JENSFELT, P. Large-scale semantic mapping and reasoning with heterogeneous modalities. In: IEEE. *2012 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2012. p. 3515–3522.

QUIGLEY, M. et al. Stair: Hardware and software architecture. In: *AAAI 2007 robotics workshop*. [S.l.: s.n.], 2007. v. 3, p. 14.

QUIGLEY, M. et al. Ros: an open-source robot operating system. In: *ICRA Workshop on Open Source Software*. [S.l.: s.n.], 2009. v. 3, n. 3.2, p. 5.

RAIA, H. F.; SIQUEIRA, F. d. L.; PLENTZ, P. D. M. Using a deadline missing prediction mechanism and recovery actions to avoid deadline losses. In: INTERNATIONAL SOCIETY FOR COMPUTERS AND THEIR APPLICATIONS. *Proceedings of the 31st International Conference on Computers and Their Applications*. [S.l.], 2016. p. 305–310.

Refractions Research. *PostGIS*. 2011. Acessado em 20/12/2016. <http://postgis.refractions.net/>.

RESEARCH, R. *Player Stage*. 2011. Acessado em 01/07/2014. <http://playerstage.sourceforge.net/>.

RINZIVILLO, S. et al. Where have you been today? annotating trajectories with daytag. In: NASCIMENTO, M. et al. (Ed.). *Advances in Spatial and Temporal Databases*. Springer Berlin Heidelberg, 2013, (Lecture Notes in Computer Science, v. 8098). p. 467–471. ISBN 978-3-642-40234-0. <http://dx.doi.org/10.1007/978-3-642-40235-7$_3$0>.

ROCCO, M. D.; PECORA, F.; SAFFIOTTI, A. When robots are late: Configuration planning for multiple robots with dynamic goals. In: IEEE. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.], 2013. p. 9515–5922.

ROCHA, J. A. M. R. et al. Db-smot: A direction-based spatio-temporal clustering method. In: *IEEE Conf. of Intelligent Systems*. [S.l.]: IEEE, 2010. p. 114–119. ISBN 978-1-4244-5164-7.

SAEEDI, S.; MOUSSA, A.; EL-SHEIMY, N. Context-aware personal navigation using embedded sensor fusion in smartphones. *Sensors*, v. 14, n. 4, p. 5742–5767, 2014. ISSN 1424-8220. <http://www.mdpi.com/1424-8220/14/4/5742>.

SANTOS, A. A. dos et al. Inferring relationships from trajectory data. In: *XVI Brazilian Symposium on GeoInformatics (GeoInfo)*. [S.l.: s.n.], 2015. p. 68–79.

SCARAMUZZA, D. et al. Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in gps-denied environments. *IEEE Robotics & Automation Magazine*, IEEE, v. 21, n. 3, p. 26–40, 2014.

SECCHI, H. A. Una introducción a los robots móviles. *Instituto de Automática–INAUT. Universidade Nacional de San Juan–UNSJ–Argentina. Edição: Agosto de*, 2008.

SEVEN, E. C. F. P. *EU FP7 Robotics UBIquitous COgnitive Network Network, RUBICON*. 2011. Acessado em 20/12/2016. <http://fp7rubicon.eu/>.

SEYHAN, S. S.; ALPASLAN, F. N.; YAVAŞ, M. Simple and complex behavior learning using behavior hidden markov model and cobart. *Neurocomputing*, Elsevier, v. 103, p. 121–131, 2013.

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. *Introduction to autonomous mobile robots*. [S.l.]: MIT press, 2011.

SILVA, T. H. et al. Traffic condition is more than colored lines on a map: Characterization of waze alerts. In: _____. *Social Informatics: 5th International Conference, SocInfo 2013, Kyoto, Japan, November 25-27, 2013, Proceedings*. Cham: Springer International Publishing, 2013. p. 309–318. ISBN 978-3-319-03260-3. <http://dx.doi.org/10.1007/978-3-319-03260-3$_2$7>.

SILVERMAN, M. C. et al. Staying alive: A docking station for autonomous robot recharging. In: IEEE. *IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2002. v. 1, p. 1050–1055.

SIQUEIRA, F. d. L.; PIERI, E. R. D. A context-aware approach to the navigation of mobile robots. In: SOCIEDADE BRASILEIRA DE AUTOMATICA. *Proceedings on Simposio Brasileiro de Automacao Inteligente. XII SBAI.* [S.l.], 2015. p. 1–6.

SIQUEIRA, F. de L.; BOGORNY, V. Discovering chasing behavior in moving object trajectories. *Transactions in GIS*, v. 15, n. 5, p. 667–688, 2011.

SIQUEIRA, F. de L.; PLENTZ, P. D. M.; PIERI, E. R. D. A fuzzy approach to the autonomous recharging problem for mobile robots. In: IEEE. *12th International Conference on Natural Computation Fuzzy Systems and Knowledge Discovery (ICNC-FSKD).* [S.l.], 2016. p. 1065–1070.

SIQUEIRA, F. de L.; PLENTZ, P. D. M.; PIERI, E. R. D. Semantic information applied to autonomous navigation of mobile robots. In: INTERNATIONAL SOCIETY FOR COMPUTERS AND THEIR APPLICATIONS. *Proceedings of the 13th ACS/IEEE International Conference on Computer Systems and Applications AICCSA 2016.* [S.l.], 2016.

SOUISSI, O. et al. Path planning: A 2013 survey. In: IEEE. *Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (IESM).* [S.l.], 2013. p. 1–8.

SPACCAPIETRA, S. et al. A conceptual view on trajectories. *Data and Knowledge Engineering*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 65, n. 1, p. 126–146, 2008. ISSN 0169-023X.

SPONG, M. W.; HUTCHINSON, S.; VIDYASAGAR, M. *Robot modeling and control.* [S.l.]: Wiley New York, 2006.

STACHNISS, C.; LEONARD, J. J.; THRUN, S. Simultaneous localization and mapping. In: *Springer Handbook of Robotics.* [S.l.]: Springer, 2016. p. 1153–1176.

STENTZ, A.; MELLON, I. C. Optimal and efficient path planning for unknown and dynamic environments. *International Journal of Robotics and Automation*, v. 10, p. 89–100, 1993.

TECHNOLOGIES, I. S. *ASCENS Project.* 2015. Acessado em 20/12/2016. <http://www.ascens-ist.eu/>.

TORIS, R. *The Standard ROS JavaScript Library*. 2015. `http://wiki.ros.org/roslibjs`. Acessado em 20/12/2016.

TURNER, R. M. Context-mediated behavior for intelligent agents. *International Journal of Human-Computer Studies*, Elsevier, v. 48, n. 3, p. 307–330, 1998.

VASQUEZ, D.; OKAL, B.; ARRAS, K. O. Inverse reinforcement learning algorithms and features for robot navigation in crowds: an experimental comparison. In: IEEE. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.], 2014. p. 1341–1346.

VASSEV, E.; HINCHEY, M. Knowledge representation for adaptive and self-aware systems. In: *Software Engineering for Collective Autonomic Systems*. [S.l.]: Springer, 2015. p. 221–247.

WACHOWICZ, M. et al. Finding moving flock patterns among pedestrians through collective coherence. *International Journal of Geographical Information Science*, Taylor & Francis, v. 25, n. 11, p. 1849–1864, 2011.

WANG, C.-Y.; HWANG, R.-H.; TING, C.-K. Ubipapago: Context-aware path planning. *Expert Systems with Applications*, Elsevier, v. 38, n. 4, p. 4150–4161, 2011.

WEI, H. et al. Staying-alive path planning with energy optimization for mobile robots. *Expert Systems with Applications*, Elsevier, v. 39, n. 3, p. 3559–3571, 2012.

WELTY, C.; MCGUINNESS, D. L. Owl web ontology language guide. *W3C recommendation, W3C (February 2004) http://www. w3. org/TR/2004/REC-owl-guide-20040210*, 2004.

WIRSING, M. et al. *Software Engineering for Collective Autonomic Systems: The ASCENS Approach*. [S.l.]: Springer International Publishing, 2015.

WITZIG, T. et al. Context aware shared autonomy for robotic manipulation tasks. In: IEEE. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.], 2013. p. 5686–5693.