Modelo distribuído bio-inspirado para planejamento de caminho e navegação de times de robôs

Samuel Cleto Soares Nametala

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2021

Samuel Cleto Soares Nametala

Modelo distribuído bio-inspirado para planejamento de caminho e navegação de times de robôs

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Dra. Gina Maira Barbosa de Oliveira
Coorientador: Dr. Luiz Gustavo Almeida Martins

Uberlândia
2021

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

N174 Nametala, Samuel Cleto Soares, 1991-
2021 Modelo distribuído bio-inspirado para planejamento de caminho e navegação de times de robôs [recurso eletrônico] / Samuel Cleto Soares Nametala. - 2021.

Orientadora: Gina Maira Barbosa de Oliveira.
Coorientador: Luiz Gustavo Almeida Martins.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2021.254
Inclui bibliografia.

1. Computação. I. Oliveira, Gina Maira Barbosa de, 1967-, (Orient.). II. Martins, Luiz Gustavo Almeida, 1974-, (Coorient.). III. Universidade Federal de Uberlândia. Pós-graduação em Ciência da Computação. IV. Título.

CDU: 681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:

Gizele Cristine Nunes do Couto - CRB6/2091

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da Computação
Av. João Naves de Ávila, nº 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Mestrado Acadêmico, 11/2021, PPGCO				
Data:	17 de junho de 2021	Hora de início:	09:05	Hora de encerramento:	12:15
Matrícula do Discente:	11822CCP011				
Nome do Discente	Samuel Cleto Soares Nametala				
Título do Trabalho:	Modelo distribuído bio-inspirado para planejamento de caminho e navegação de times de robôs				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Inteligência Artificial				
Projeto de Pesquisa de vinculação:	-				

Reuniu-se, por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Carlos Roberto Lopes - FACOM/UFU; Roseli Aparecida Francelin Romero - ICMC/USP; Luiz Gustavo Almeida Martins - FACOM/UFU (coorientador) e Gina Maira Barbosa de Oliveira - FACOM/UFU, orientadora do candidato.

Os examinadores participaram desde as seguintes localidades: Roseli Aparecida Francelin Romero - São Carlos/SP; Carlos Roberto Lopes, Luiz Gustavo Almeida Martins e Gina Maira Barbosa de Oliveira - Uberlândia/MG. O discente participou da cidade de Uberlândia/MG.

Iniciando os trabalhos a presidente da mesa, Prof.ª Dr.ª Gina Maira Barbosa de Oliveira, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir a senhora presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado.

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Luiz Gustavo Almeida Martins**, **Professor(a) do Magistério Superior**, em 22/06/2021, às 11:18, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Roseli Aparecida Francelin Romero**, **Usuário Externo**, em 22/06/2021, às 15:05, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Gina Maira Barbosa de Oliveira**, **Professor(a) do Magistério Superior**, em 22/06/2021, às 15:32, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Carlos Roberto Lopes**, **Professor(a) do Magistério Superior**, em 15/07/2021, às 15:47, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2839469** e o código CRC **3F79916E**.

Referência: Processo nº 23117.038695/2021-23 SEI nº 2839469

Este trabalho é dedicado a todos que de alguma forma contribuem para evolução da ciência.

Agradecimentos

Agradeço primeiramente a Deus, pela força e coragem nos momentos difíceis.

Agradeço aos meus pais Cleto Nametala Ibraim e Abadia Soares da Silveira Ibraim, ao meu irmão Matheus Soares Nametala e a todos familiares, amigos e colegas pelo incentivo e apoio.

Agradeço à minha orientadora, Gina Maira Barbosa de Oliveira e ao meu coorientador Luiz Gustavo Almeida Martins pela confiança, amizade, dedicação e orientação.

Agradeço a todos meus outros professores, que fizeram com que eu chegasse até aqui, em especial ao meu orientador de projeto de iniciação científica e de trabalho de conclusão de curso Gabriel da Silva, que sempre me deu força e incentivo.

Agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela concessão da bolsa de mestrado que me auxiliou durante a realização deste trabalho.

E, finalmente, a todos que participaram, direta ou indiretamente do desenvolvimento deste trabalho de pesquisa.

"Não há nada nobre em ser superior ao seu semelhante. A verdadeira nobreza é ser superior ao seu antigo eu."

(Ernest Hemingway)

Resumo

Os autômatos celulares (AC) são abordagens bioinspiradas que foram recentemente investigadas para várias aplicações, incluindo a robótica. Um modelo aprimorado baseado em regras de AC e fluxo de decisões locais é proposto e avaliado para a navegação distribuída e descentralizada de times de robôs autônomos. O modelo proposto foi denominado BioTeam: modelo distribuído bio-inspirado para planejamento de caminho e navegação de times de robôs. Na primeira etapa, o planejamento de caminho é realizado por cada robô de forma independente. O objetivo é que cada robô do time consiga construir um caminho curto e livre de colisão da sua posição atual até sua meta, tentando evitar desvios desnecessários tanto quanto possível. Durante o deslocamento pelo caminho definido, o sistema de controle descentralizado presente em cada robô deve ser capaz de evitar qualquer situação de possível colisão e principalmente superar possíveis conflitos de trajetórias envolvendo outros robôs, sem prejudicar a sua própria eficiência e a dos outros membros do time. Assim, cada robô atinge seu objetivo de forma independente, sem comunicação com outros robôs. Experimentos foram realizados para confirmar a eficiência das novas técnicas empregadas. Simulações utilizando a plataforma Webots apresentaram resultados promissores na navegação de times com 10 a 50 robôs. Esses resultados confirmam que o modelo Bioteam é capaz de planejar rotas suaves e curtas e na navegação os robôs conseguem superar diferentes situações de conflitos de forma independente e descentralizada em ambientes grandes e complexos, com vários robôs se movimentando ao mesmo tempo.

Palavras-chave: Autômatos celulares, computação bioinspirada, controle de navegação descentralizada, difusão à distância, planejamento de caminhos, robótica autônoma.

Abstract

Cellular automata (AC) are bioinspired approaches that have recently been investigated for various applications, including robotics. An improved model based on CA rules and local decision flow is proposed and evaluated for distributed and decentralized navigation of teams of autonomous robots. The proposed model was called BioTeam: a bio-inspired distributed system for path planning and navigation of robot teams. In the first stage, the path planning is carried out by each robot independently. The goal is for each robot on the team to be able to build a short, collision-free path from its current position to its goal, trying to avoid unnecessary detours as much as possible. During displacement along the defined path, the decentralized control system present in each robot must be able to avoid any situation of possible collision and, above all, to overcome possible trajectory conflicts involving other robots, without jeopardizing its own efficiency and that of the other team members. Thus, each robot achieves its goal independently, without communication with other robots. Experiments were carried out to confirm the efficiency of the new techniques employed. Simulations using the Webots platform showed promising results in the navigation of teams with 10 to 50 robots. These results confirm that the Bioteam model is capable of planning smooth and short routes and in navigation the robots can overcome different conflict situations independently and decentrally in large and complex environments, with several robots moving at the same time.

Keywords: Cellular automata, bioinspired computing, decentralized navigation control, distance diffusion, path planning, autonomous robotics.

Lista de ilustrações

Lista de tabelas

Lista de Siglas

AC Autômatos celulares

ARC Regra do alargamento de regiões côncava

CP Célula promissora

CPI Módulo de captura e processamento deimagem

DM Distância até a meta

GPS *Global Positioning System*

ND Módulo de navegação distribuída

OT Objetivo temporário

PC Módulo de planejamento de caminhos

RAO Regra do alargamento de obstáculos

RAP Regra do alargamento de parede

RPD Regra de propagação diagona

RPL Regra de propagação latera

Sumário

Capítulo 1

Introdução

A robótica autônoma tem o objetivo de desenvolver robôs capazes de perceber seu ambiente, tomar decisões com base no que percebem ou foram programados para reconhecer e, em seguida, acionar um movimento ou manipulação nesse ambiente. Uma das principais características de um robô autônomo é sua capacidade de percorrer um ambiente, sem colidir com obstáculos ou outros robôs. Para isso, um dos problemas mais investigados é a tarefa de planejamento de caminhos (ARKIN, 1998), que visa construir o melhor caminho livre de colisões, da posição atual do robô até a sua meta.

Um dos desafios da robótica autônoma, relativo ao problema de planejamento de caminho, é a movimentação distribuída e descentralizada de uma equipe de robôs para que o objetivo principal do sistema seja alcançado. Esses sistemas apresentam diversas vantagens em relação aos sistemas que usam apenas um robô, pois demandam maior flexibilidade e adaptabilidade para resolver tarefas complexas de forma mais eficiente e robusta. Sua característica descentralizada garante que qualquer robô consiga tomar suas próprias decisões independentemente dos outros membros do time (SILVA et al., 2015). Por exemplo, o uso de múltiplos robôs em um centro de distribuição, que é o local onde ficam armazenadas as mercadorias, quando chegam das fábricas e antes de serem entregues aos pontos de venda. A tarefa que envolve o transporte dessas mercadorias é dividida entre vários agentes para agilizar o processo. Um modelo descentralizado garantiria que se algum robô falhasse, o comportamento dos demais não seria comprometido, e a tarefa daquele que falhou poderia ser assumida por outro membro do time. Ou seja, com o uso de uma solução distribuída, pretende-se que mesmo que a eficiência seja prejudicada, o objetivo final ainda é alcançado independentemente de eventuais problemas. Isso não aconteceria em modelos centralizados, pois, se algum robô falhasse, todos os outros robôs poderiam ter seu comportamento comprometido e o objetivo final não seria alcançado. Dessa forma, não apenas a eficiência do modelo é prejudicada, mas também sua eficácia.

Existem diferentes abordagens que tratam o problema de planejamento de caminho. Além de métodos mais convencionais como o uso de mapa de rotas (KAVRAKI et al., 1996), (ZHANG; FATTAHI; LI, 2013), decomposição celular (LINGELBACH, 2004),

(RAMER; REITELSHÖFER; FRANKE, 2013) e campo potencial (BARRAQUAND; LANGLOIS; LATOMBE, 1992), (JIANJUN et al., 2013), várias pesquisas têm adotado heurísticas de busca inteligentes, tais como A* e D*-lite (SETIAWAN et al., 2014) na definição das rotas para os robôs. Contudo, técnicas bioinspiradas, como autômatos celulares (AC), também estão sendo aplicados com sucesso nessa tarefa, em sistemas com um (MARTINS et al., 2018) ou vários agentes (OLIVEIRA et al., 2019). ACs são sistemas computacionais dinâmicos, compostos por um espaço celular de N células idênticas, dispostas em um arranjo d-dimensional, que são totalmente discretos em estados, espaço e tempo. Eles também possuem regras de transição, que definem o estado de uma célula no próximo passo de tempo, baseado na configuração de sua vizinhança atual. Dentre as abordagens que utilizam ACs, este trabalho se baseia nas técnicas de difusão da distância e decisão local. Essas duas abordagens utilizam informações prévias do ambiente, podendo ser aplicadas, por exemplo, no transporte de produtos em centros de distribuição e logística (VIVALDINI et al., 2010), vigilância de ambientes (TINOCO; OLIVEIRA, 2019) e mapeamento cooperativo (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011), (OLIVEIRA et al., 2019).

O presentre trabalho propõe um novo modelo de navegação distribuída e descentralizada para um time de robôs, mas para isso é fundamental que a rota planejada para esses robôs seja eficaz e eficiente. Nesse contexto, novas estratégias foram introduzidas tanto no planejamento do caminho de cada robô, como no controle de navegação de todo o time. Contudo ainda vale destacar que essa descentralização envolve apenas o time de robôs, e não o sistema como um todo, uma vez que para o seu funcionamento é necessário a requisição de uma imagem do ambiente a partir de um dispositivo externo, como será explicado nas próximas seções.

Inicialmente, aprimoramos o modelo de planejamento de caminhos usado por cada robô, o qual é baseado nas regras locais de autômatos celulares. O modelo utilizado como base foi originalmente proposto em (BEHRING et al., 2001) e refinado em trabalhos posteriores (OLIVEIRA; VARGAS; FERREIRA, 2015a) e (MARTINS et al., 2018). Nesses refinamentos, o foco foi melhorar a navegação, enquanto o robô atravessa o ambiente. Porém, foi observado que as rotas geradas nesses modelos exigiam uma grande quantidade de rotações do robô durante a navegação. Em rotas mais longas, o acúmulo de erros de odometria provocado pelo excesso de giros é tão elevado que o robô se perde durante o percurso, levando à colisões ou à impossibilidade de alcançar seu objetivo. A odometria é uma técnica utilizada para determinar a posição do robô dentro do ambiente onde ele está movimentando. Durante a navegação, dois tipos de erros costumam dificultar o posicionamento preciso do robô: erros sistemáticos, relacionados às características inerentes ao robô, como desgaste desproporcional das rodas; e erros não sistemáticos, relacionados à características do meio ambiente, como pisos escorregadios. Os erros sistemáticos são reduzidos de forma satisfatória com o uso da odometria, enquanto os erros não sistemá-

ticos são mais difíceis de lidar. As mudanças propostas nos modelos anteriores visavam corrigir as imprecisões relacionadas ao emprego da odometria durante a navegação. Neste contexto, nossa proposta busca minimizar os erros de odometria por meio da redução da distância e do número de rotações exigidas na rota planejada. Inicialmente, o modelo de difusão de distância foi modificado, de modo a atribuir um custo maior para os movimentos diagonais em relação aos laterais, o que não ocorre nos modelos anteriores (BEHRING et al., 2001), (OLIVEIRA; VARGAS; FERREIRA, 2015a), (MARTINS et al., 2018). Além disso, novas regras de ACs foram incorporadas no modelo a fim de identificar e restringir o acesso às regiões côncavas do ambiente, possibilitando o planejamento de rotas mais curtas e com trajetória mais refinada.

Com o objetivo de propiciar a navegação de vários robôs pelo ambiente sem a ocorrência de colisões, o modelo de planejamento de caminhos proposto deve ser integrado a um sistema de controle de navegação capaz de lidar com um time de robôs. Neste trabalho é desenvolvido um modelo de navegação com inspiração no proposto em (OLIVEIRA et al., 2019) que, por sua vez, é uma extensão do modelo investigado originalmente em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2008) e (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011). Esses trabalhos anteriores focam no controle de navegação de um time formado por três robôs, os quais devem se manter em formação o maior tempo possível enquanto atravessam o ambiente em linha reta. Além disso, apesar do próximo movimento ser decidido em cada robô (decisão descentralizada), o sistema exige a comunicação entre os robôs da formação, o que torna o processo mais complexo e demanda um maior custo computacional. Enquanto os trabalhos anteriores têm como objetivo o controle de formação de no máximo três robôs, o modelo proposto no presente trabalho foca no deslocamento de vários robôs por ambientes mais complexos, sem que eles colidam entre si e adotando políticas de resolução de conflitos durante suas trajetórias. A nova abordagem incorpora um fluxo de decisões locais, que possibilita um controle de navegação distribuído, descentralizado e sem comunicação entre os robôs. Este modelo de fluxo de decisões locais busca capacitar os robôs a se movimentarem para seus objetivos em ambientes maiores e mais complexos, evitando qualquer obstáculo no seu caminho e principalmente superando qualquer situação de conflito que envolva um ou mais robôs no ambiente, sem prejudicar a sua eficiência ou a do time.

Resultados experimentais mostraram que nossa abordagem propiciou uma redução média de até 5% no tamanho das rotas e de até 25% no número de rotações em relação aos modelos anteriores ((OLIVEIRA; VARGAS; FERREIRA, 2015a), (MARTINS et al., 2018)), bem como proporcionou uma navegação eficiente de até 40 robôs, por diferentes ambientes complexos.

1.1 Motivação

Um sistema multi-agente composto por robôs pequenos e de baixo custo apresenta vantagens em relação aos sistemas que utilizam um único robô complexo e caro (OLIVEIRA et al., 2018). Vários robôs simples, quando precisamente coordenados, podem cumprir tarefas mais complexas, permitindo maior flexibilidade e adaptabilidade, assim como paralelismo e redundância ao sistema (ROUMELIOTIS; BEKEY, 2002) (PROROK; BAHR; MARTINOLI, 2012). Esse tipo de sistema também propicia robustez e confiabilidade, considerando que, em uma abordagem descentralizada, se algum robô vier a falhar, os demais membros do time ainda são capazes de executar a tarefa desejada, o que não ocorre em sistemas com um único robô ou com controle centralizado.

A robótica coletiva envolve a coordenação de um grupo de robôs (time) para que realizem certas ações apresentando comportamento cooperativo, enquanto se movem pelo ambiente, evitando eventuais colisões com obstáculos e outros robôs. No geral, as estratégias de coordenação podem ser classificadas em controle de formação ou flocagem (NAVARRO; MATÍA, 2013). No controle de formação, os robôs do time devem se manter em uma formação específica (por exemplo, linear ou triangular) ao percorrer um caminho (OLIVEIRA et al., 2019). Essa estratégia pode ser usada em tarefas de busca (ex: operações de resgate e monitoramento de linhas de transmissão). A flocagem, por sua vez, consiste em mover um time de robôs sem considerar a forma e as posições relativas entre os robôs (LIMA et al., 2017). Esse tipo de coordenação é útil em sistemas de entrega ou transporte de produtos (ex: operação de centros de distribuição e logística). Independentemente do tipo de controle, o planejamento de trajetórias sem obstáculos e o movimento coordenado do time de robôs estão entre as áreas mais estudadas na robótica autônoma (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011) (OLIVEIRA et al., 2019).

Dentre os desafios da robótica cooperativa pode-se destacar: navegar pelo ambiente evitando obstáculos estáticos e dinâmicos, adaptar-se dinamicamente às mudanças no número de robôs e minimizar o gargalo de comunicação (NAVARRO; MATÍA, 2013). Várias pesquisas estão sendo realizadas para lidar com esses problemas: métodos baseados em comportamento (BALCH; ARKIN, 1998), lógica fuzzy (FU; LI; MA, 2006), redes neurais artificiais evolutivas (CAPI; MOHAMED, 2012), rede reguladora de genes (MENG; GUO; JIN, 2013), campos potenciais (REZAEE; ABDOLLAHI, 2013) e aprendizagem profunda por reforço (ZHU et al., 2020). Entretanto, a navegação de robôs em tempo real requer um sistema de planejamento e controle rápido e eficiente, enquanto muitos dos métodos propostos da literatura ou têm um alto custo computacional, ou apresentam uma implementação restritiva (OLIVEIRA et al., 2018).

Autômatos celulares (ACs) são sistemas dinâmicos, discretos, distribuídos e de baixo custo computacional, pois são capazes de representar fenômenos complexos a partir de computações simples baseadas em regras locais (vizinhança) (OLIVEIRA et al., 2019). Essas características favorecem seu uso na robótica autônoma coletiva, uma vez que mé-

todos descentralizados que utilizam informações locais são mais eficientes para lidar com cenários reais, apresentando maior tolerância a falhas de comunicação e mudanças no ambiente (GUANGHUA et al., 2013).

Modelos baseados em ACs têm sido propostos para a coordenação coletiva de times de robôs simples, ou seja, equipados com processadores menos potentes, pouca memória e sensores de baixo custo (ex: sensores de proximidade) (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011) (OLIVEIRA; VARGAS; FERREIRA, 2015b) (TARIQ; KUMARAVEL, 2016) (LIMA; OLIVEIRA, 2017) (OLIVEIRA et al., 2019). Embora essas abordagens apresentem resultados promissores, elas costumam adotar um agente responsável por gerenciar o movimento de todo o time (controle centralizado) ou exigem constantes comunicações entre os robôs para coordenar suas ações, aumentando assim o custo e a complexidade do processo.

Como nas abordagens anteriores, o modelo aqui apresentado também é baseado em autômatos celulares e usa regras de decisões locais para definir o próximo movimento de cada robô. Entretanto, tal decisão é feita de forma descentralizada e distribuída, baseada apenas em informações do ambiente e da vizinhança do robô, capturada a partir de sensores de proximidade, sem a necessidade de comunicação direta entre os robôs do time. Assim, o sistema proposto é capaz de trabalhar com uma grande quantidade de robôs, cada qual tomando suas próprias decisões de forma independente; e resolver qualquer tipo de conflito envolvendo um ou mais robôs em uma variedade de ambientes complexos. Esses conflitos são resolvidos através de uma cooperação indireta entre os membros do time por meio da escolha e realização de diferentes manobras, propiciando que o objetivo coletivo do time seja alcançado. Além disso, o método de planejamento de caminhos baseado na difusão de distâncias foi aprimorado a fim de criar uma trajetória mais eficiente, eliminando rotações desnecessárias e reduzindo a distância percorrida. A redução nessas variáveis diminuiu significativamente a quantidade de erros acumulados pela odometria.

1.2 Objetivos e Desafios da Pesquisa

O objetivo principal deste trabalho é conceber e integrar novas regras de autômatos celulares e um fluxo de decisões locais a modelos prévios da literatura (OLIVEIRA; VARGAS; FERREIRA, 2015a) (MARTINS et al., 2018), a fim de aprimorar o planejamento dos caminhos, gerando rotas livres de colisões que sejam mais curtas e que exijam menos rotações do robô, bem como proporcionar um controle de navegação distribuído, descentralizado e eficiente para um time de robôs. Para isso, nossa pesquisa busca alcançar os seguintes objetivos específicos:

a) Desenvolver um sistema que propicie a independência dos robôs de um time quanto a escolhas de sua trajetória e os controles de sua navegação;

b) Desenvolver um modelo de planejamento capaz de determinar, para cada robô, o caminho mais curto e com a menor quantidade de giros até seu objetivo;

c) Desenvolver uma estratégia de alargamento dos obstáculos capaz de restringir o acesso do robô às regiões côncavas que podem comprometer o planejamento eficiente das rotas;

d) Integrar as informações da imagem do ambiente navegado e de sensores de proximidade na movimentação de cada robô do time, de modo a prover uma rota livre de colisões e a tomada de decisões locais no controle de navegação descentralizado;

e) Prover um controle de navegação descentralizado baseado em fluxo de decisão local e sem a necessidade de comunicação entre os robôs, tornando o sistema mais robusto a falhas individuais nos robôs do time;

f) Integrar o uso de sensores de proximidade ao controle de navegação usado em (MARTINS et al., 2018) de modo que o robô seja capaz de detectar e identificar algum obstáculo na sua trajetória;

g) Conceber estratégias de desvio de obstáculos que não comprometam significativamente a eficiência da rota planejada;

h) Proporcionar ao robô a capacidade de escolher e realizar manobras de desvio com o objetivo de resolver algum conflito de trajetória com outros robôs sem comprometer a sua eficácia e a do time;

i) Prover um sistema de controle de navegação descentralizado e robusto, de modo que cada robô seja capaz de continuar com sua tarefa mesmo com a perda de outros membros da equipe.

1.3 Hipótese

Este trabalho busca tratar as seguintes hipóteses:

a) Através da aplicação de metodologias bioinspiradas e fluxo de decisões locais, é possível construir um sistema de planejamento de caminho para a navegação distribuída e descentralizada de um time de robôs?

b) Essa abordagem é escalável o suficiente para possibilitar a navegação de um número considerável de robôs pelo ambiente de forma eficiente e sem colisões?

1.4 Contribuições

O presente trabalho atua em dois problemas de pesquisa da robótica autônoma: planejamento do caminho do robô e controle de navegação de um time de robôs.

No planejamento de caminho, nossas principais contribuições são:

- ❏ Mudança na forma de propagação da distância de cada célula do AC, priorizando os movimentos cardeais em relação aos diagonais.
- ❏ Criação e aplicação de novas regras de autômato celular no alargamento de obstáculos, a fim de identificar e restringir o acesso às regiões côncavas do ambiente, possibilitando o planejamento de rotas mais curtas e com trajetória mais refinada.

Com o desenvolvimento deste trabalho, primeiramente propomos um novo modelo de algoritmo para a tarefa de planejamento de caminho para um time de robôs, baseado em características bioinspiradas. Sua proposição foi motivada pela identificação de que a quantidade de giros executada pelos robôs em modelos anteriores poderia ser reduzida e, consequentemente, reduzir a quantidade de erros de odometria ao longo do trajeto do robô. Assim, foi proposto um novo modelo para o planejamento de caminho, onde as principais contribuições estão na atualização das regras de autômatos celulares que calculam a distância de cada célula até o objetivo (presente nos modelos anteriores), e na criação de um novo conjunto de regras de autônomos celulares que identifica e bloqueia as regiões côncavas do ambiente, possibilitando o planejamento de rotas mais curtas e com trajetória mais refinada.

Com o modelo de planejamento de caminho para um robô refinado e avaliado através de simulações, partimos para a proposição de um novo modelo de navegação que possibilitasse a utilização de vários robôs no mesmo ambiente, uma vez que diferentes aplicações exigem a utilização de um time de robôs. As principais contribuições desse novo sistema de controle de navegação são:

- ❏ Integração da navegação baseada em sensores de proximidade apresentada em (OLIVEIRA et al., 2019) ao modelo proposto em (MARTINS et al., 2018) a fim de evitar colisões em aplicações com time de robôs.
- ❏ Concepção de estratégias de desvio na rota baseados em autômato celular e decisões locais que seja capaz resolver conflitos na trajetória dos robôs do time de forma descentralizada.

Assim, foi proposto um novo algoritmo distribuído e descentralizado, através de um sistema de navegação por fluxo de decisões locais, possibilitando aos robôs a capacidade de resolver seus conflitos de trajetória de forma independente, sem comprometer eficácia do time. Esse comportamento também foi observado através de experimentos realizados em ambientes simulados na plataforma Webots.

1.5 Organização da Dissertação

O restante do documento está organizado da seguinte forma:

- **Capítulo 2 (fundamentação teórica):** apresenta um breve embasamento teórico sobre autômatos celulares e o problema de planejamento de caminhos, introduzindo uma visão geral e os principais conceitos relacionados a esses temas.
- **Capítulo 3 (trabalhos correlatos):** apresenta uma revisão da literatura, descrevendo os principais trabalhos relacionados ao planejamento de caminhos para um ou mais robôs.
- **Capítulo 4 (modelo proposto):** descreve a presente abordagem para o planejamento de caminhos e controle de navegação para times de robô, detalhando as estratégias empregadas em cada módulo.
- **Capítulo 5 (experimentos e resultados):** descreve os experimentos realizados a fim de validar a eficiência e efetividade de nossa abordagem, tanto no planejamento das rotas do robô, como na navegação com diferentes tamanhos do time de robôs em ambientes complexos.
- **Capítulo 6 (conclusões):** apresenta nossas conclusões e contribuições, bem como sugere possíveis evoluções que poderão ser desenvolvidas no futuro.

Capítulo 2

Fundamentação Teórica

Neste capítulo, são definidos os principais conceitos sobre autômatos celulares e robótica autônoma necessários para o entendimento e desenvolvimento do trabalho proposto.

2.1 Autômatos celulares

Autômatos celulares (ACs) são ferramentas de modelagem computacional capazes de simular e prever uma grande diversidade de fenômenos complexos. Algumas características como natureza intrinsecamente contínua no espaço e tempo, que na maioria dos casos, ocorrem em ambientes com mais de uma dimensão, tornam extremamente difíceis de serem simuladas em sistemas digitais. Dessa forma, os ACs buscam uma maneira de simular esses fenômenos em sistemas digitais através da discretização das variáveis contínuas apresentadas (TINOCO et al., 2019).

Os ACs foram propostos originalmente por von Neumann e Ulam, como uma possível idealização de sistemas biológicos, com a proposta de modelar a auto reprodução biológica. Eles são sistemas computacionais dinâmicos, distribuídos espacialmente e totalmente discretos em estados, espaço e tempo, compostos por componentes simples e idênticos com interações locais (FERREIRA et al., 2014). Basicamente, um AC consiste em duas partes: o espaço celular e a regra de transição. O espaço celular é um reticulado de N células idênticas que podem formar um arranjo unidimensional (1D), bidimensional (2D) ou tridimensional (3D), como ilustrado na Figura 1, e cada célula deste arranjo é associada a um estado a cada instante de tempo (de um conjunto de estados possíveis).

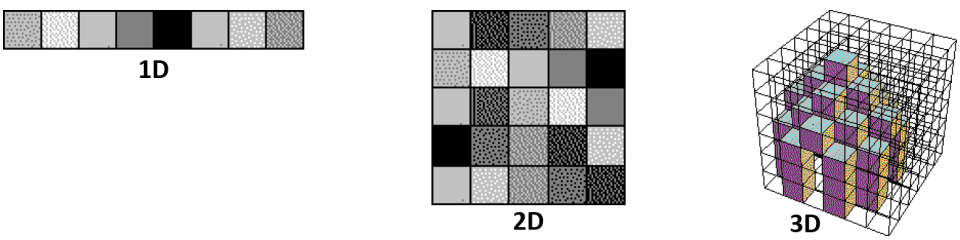


Figura 1 – Representação dos arranjos celulares.

Os autômatos celulares são caracterizados por sua regra de transição, que determina o estado de cada célula do reticulado no próximo instante de tempo a partir da configuração atual de estados de sua vizinhança. Isto é, a cada passo de tempo, a regra é aplicada em todas as células do reticulado, sendo que o estado de uma célula pode ou não ser alterado dependendo de seu valor atual e da configuração de sua vizinhança. As interações locais geram um comportamento global que altera a configuração do reticulado de células no próximo passo de tempo (FERREIRA et al., 2014). Portanto, ACs são sistemas espacialmente descentralizados, com um grande número de componentes conectados localmente, de modo que esses sistemas são capazes de desempenhar funções complexas, tal como o comportamento de um modelo ou sistema natural complexo (CÂNDIDO; OLIVEIRA; MARTINS, 2018).

Dentre as características dos ACs, aquelas que tornam o seu uso promissor para tratar problemas de planejamento de caminhos e controle de navegação em times de robôs autônomos são (SILVA et al., 2015): *(i)* seu baixo custo computacional; *(ii)* sua natureza discreta e paralela que facilita a portabilidade entre cenários de simulação e cenários reais; *(iii)* possuem comportamento simples baseado em regras e não precisam da informação completa do ambiente.

2.1.1 Autômatos celulares unidimensionais

Esse é o arranjo mais simples de um autômato celular e, apesar de não ser o utilizado em nosso trabalho, sua explicação é importante para facilitar o entendimento de como um AC trabalha. O reticulado desse tipo de autômato normalmente é representado por um vetor, no qual cada posição equivale a uma célula. Nos ACs unidimensionais, sua vizinhança é representada pela célula central e suas vizinhas dentro de um raio r, sendo o tamanho da vizinhança (m) normalmente definida como:

$$m = 2r + 1 \tag{1}$$

A Figura 2 apresenta um exemplo de reticulado binário, onde o valor de cada célula é 0 ou 1, e que utiliza uma vizinhança de raio 1 ($m = 3$). Um exemplo dessa vizinhança é destacada nesta figura, onde nota-se a célula central em verde (x) e suas vizinhas imediatas à esquerda (x-1) e à direita (x+1), ambas em azul.

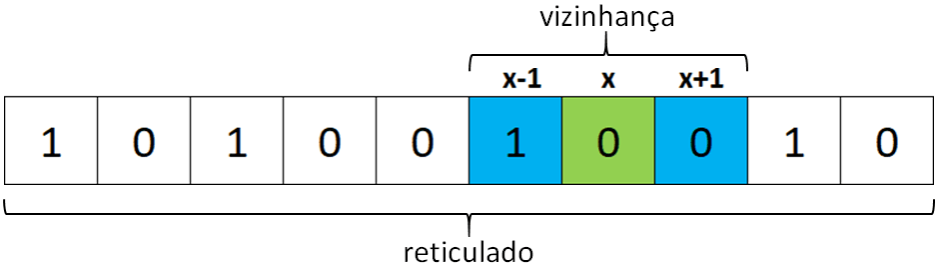


Figura 2 – Reticulado unidimensional e vizinhança de raio 1.

Antes de aplicar uma regra de transição, é definida uma condição de contorno para o AC, a qual é aplicada na transição das células pertencentes à borda esquerda e direita desse reticulado unidimensional. Uma forma usual de se definir a condição de contorno é considerar que o reticulado possui o formato de um anel, de modo que as extremidades do reticulado são conectadas. Nessa definição, mostrada na Figura 3, a primeira célula do reticulado em amarelo, tem como vizinha à esquerda a última célula do reticulado em cinza. De forma similar, a primeira célula do reticulado torna-se a vizinha à direita da última célula. Essa condição de contorno é chamada de periódica (MARTINS; FYNN; OLIVEIRA, 2011).

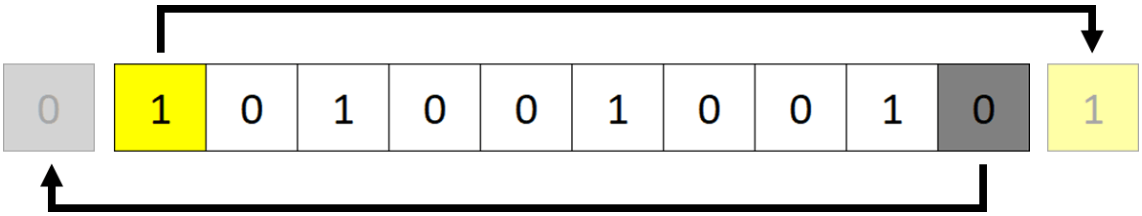
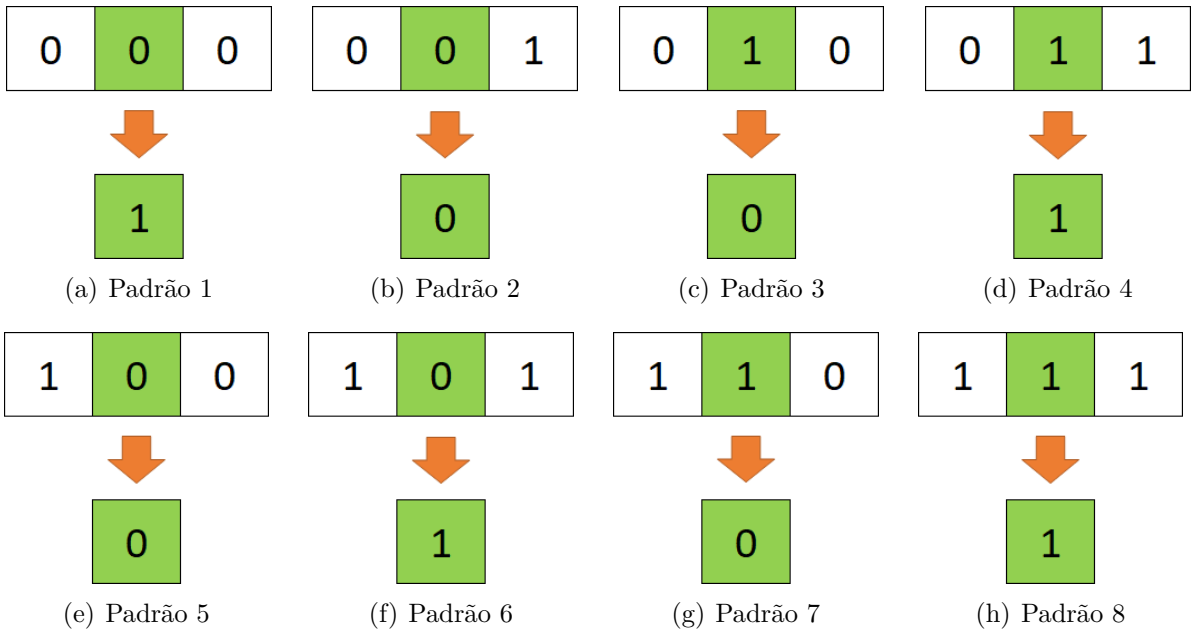


Figura 3 – Exemplo de vizinhança periódica.

A regra de transição do AC define o novo estado da célula central de acordo com o padrão de cada uma das vizinhanças possíveis. A Figura 4 ilustra uma possível regra de transição, com cada um dos padrões de vizinhança e o novo estado da célula central no próximo passo de tempo. Esses padrões correspondem ao mapeamento para todas as vizinhas possíveis, dado por k^m, sendo k a quantidade de possíveis estados de cada célula e m o tamanho da vizinhança. No nosso exemplo, temos $k = 2$ e $m = 3$, então o número de padrões é $2^3 = 8$.

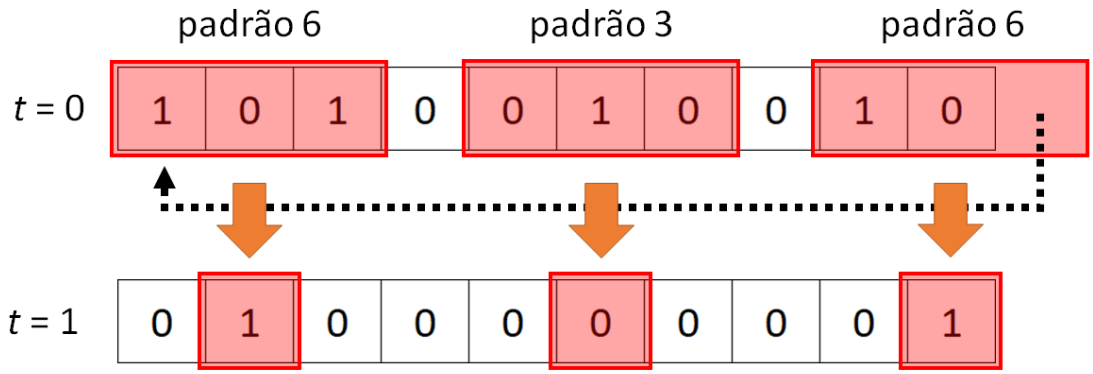


(a) Padrão 1 (b) Padrão 2 (c) Padrão 3 (d) Padrão 4

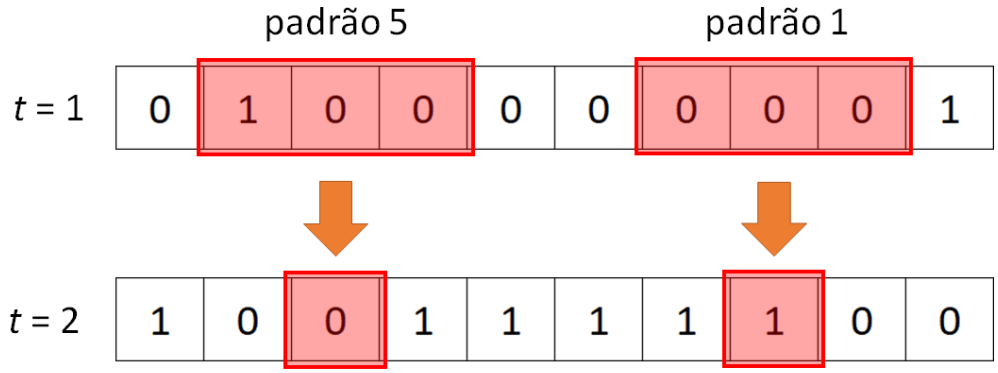
(e) Padrão 5 (f) Padrão 6 (g) Padrão 7 (h) Padrão 8

Figura 4 – Exemplo de regras de transição.

A Figura 5 apresenta o processamento de um AC unidimensional em 2 passos de tempo. Inicialmente (Figura 5(a)), o AC começa no passo de tempo 0 ($t = 0$) com a configuração do reticulado "1010010010'. Após cada célula desse reticulado ser atualizada, aplicando-se a regra mostrada na Figura 4, com base na configuração de cada vizinhança, o passo de tempo é concluído obtendo-se o reticulado do próximo passo de tempo 1 ($t = 1$) com uma nova configuração "0100000001". O passo de tempo 2 ($t = 2$) (Figura 5(b)) é obtido aplicando-se novamente a regra de transição na configuração do reticulado em $t = 1$, obtendo-se a configuração "1001111100". A regra de transição é aplicada sucessivamente pela quantidade de passos necessários para resolver um problema. Vale ressaltar que a principal vantagem de um AC é sua capacidade de trabalhar de forma paralela, de modo que cada uma das células pode ser processada de forma independente. Nesse contexto, a Figura 5 destaca em vermelho o processamento simultâneo de 3 vizinhanças, com o novo valor de cada célula central no próximo passo de tempo, considerando os padrões de vizinhança da regra de transição representada na Figura 4.



(a) Evolução temporal de $t = 0$ para $t = 1$



(b) Evolução temporal de $t = 1$ para $t = 2$

Figura 5 – Evolução de um AC unidimensional em 2 passos de tempo.

A evolução do reticulado do AC por diversos passos de tempo pode ser visualizada por um diagrama de espaço temporal, como mostrado na Figura 6. Para facilitar a visualização, para cada passo de tempo, pode-se observar duas imagens que correspondem ao mesmo espaço celular. Na imagem à esquerda, as células são representadas por estados binários, enquanto na imagem à direita, os estados das células são representados por cores, sendo células em branco equivalente a 0 e células em preto a 1. Esse processo pode continuar por vários passos de tempo de acordo com a necessidade do problema.

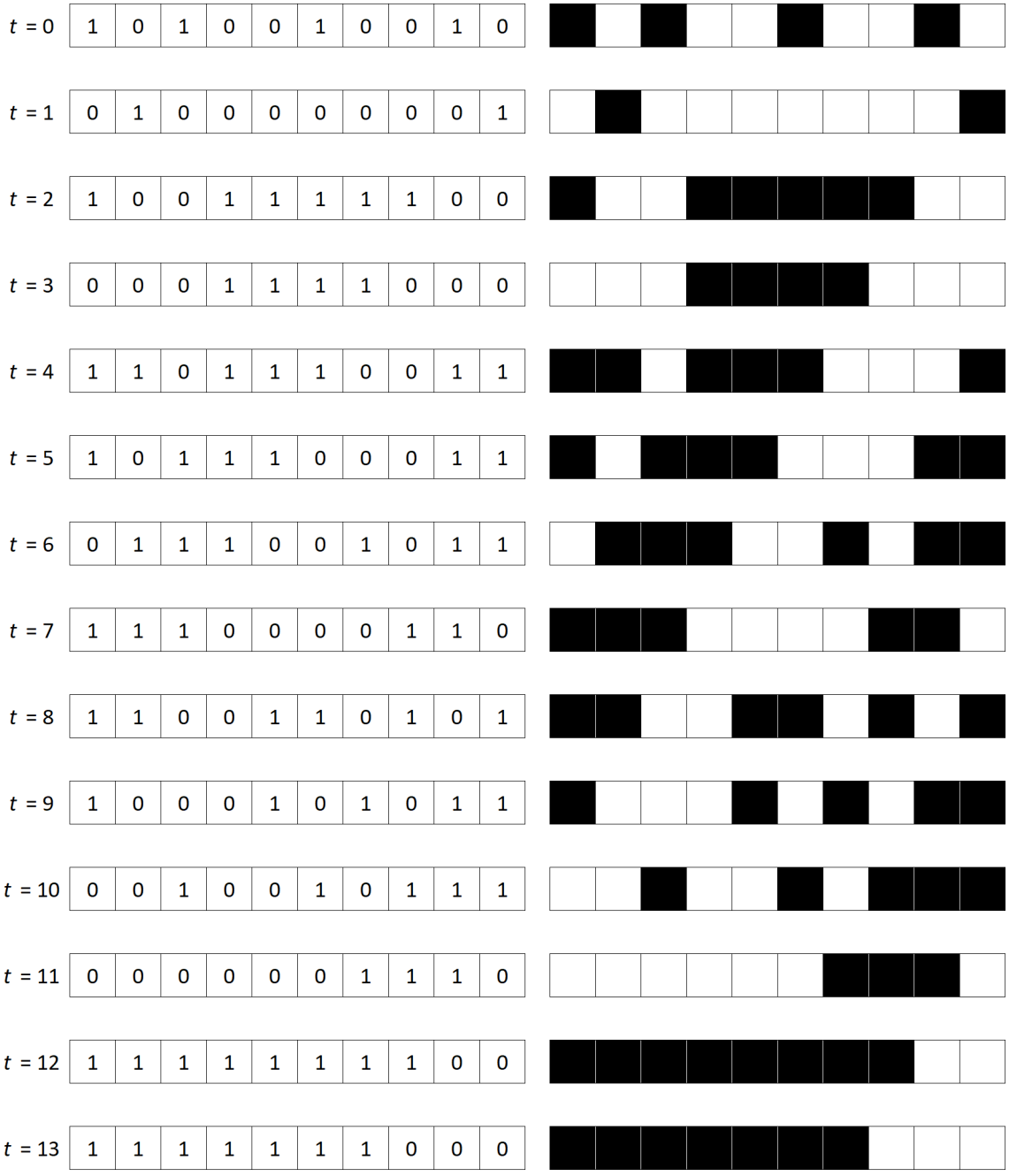
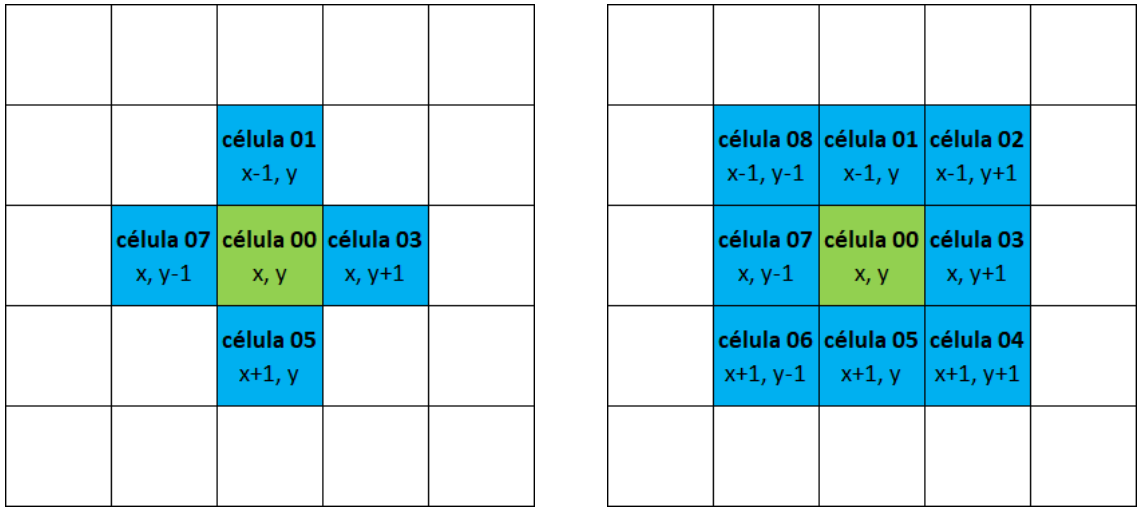


Figura 6 – Diagrama de espaço temporal para 13 passos de tempo.

2.1.2 Autômatos celulares bidimensionais

Os ACs bidimensionais, são geralmente representados por uma matriz, na qual cada posição representa uma célula. A configuração da vizinhança pode variar de acordo com o problema, não restringido apenas às células vizinhas da direita ou da esquerda. Na Figura 7 podemos ver dois exemplos de vizinhança mais utilizados, sendo em verde a célula central e em azul suas vizinhas. Cada uma delas possui raio igual a 1, que determina o alcance da vizinhança em relação à célula central. Elas são chamadas de vizinhança de von Neumann (Figura 7(a)), com suas células vizinhas nas posições cardeais, norte (x-1, y), sul (x+1, y), leste (x, y+1) e oeste (x, y-1), e vizinhança de Moore (Figura 7(b)), que inclui as células nas posições colaterais, nordeste (x-1, y-1), noroeste (x-1, y+1), sudeste (x+1, y-1), e sudoeste (x+1, y+1). Nesse trabalho, os ambientes de navegação dos robôs serão modelados como reticulados de ACs bidimensionais e a vizinhança de Moore será utilizada para representar a vizinhança de uma célula.



(a) Vizinhança de von Neumann (b) vizinhança de Moore

Figura 7 – Tipos de vizinhança em Autômatos Celulares Bidmensionais.

Um exemplo muito conhecido de AC chamado *Game of Life*, ou simplesmente *Life*, utiliza regras bem simples, e altera o estado de cada célula central, com base na sua vizinhança. Esse modelo foi proposto por John Conway em 1970 e foi primeiramente publicado em (GARDNER, 1970). Contudo, apesar das regras simples, o *Life* consegue mostrar um comportamento complexo no espaço celular e padrões interessantes emergem com frequência após uma configuração inicial aleatória. Consiste em um autômato celular bidimensional, de raio 1 e com vizinhança de Moore. Esse AC possui dois estados: célula viva ou célula morta, muitas vezes representados por valores binários. A regra de transição é definida por:

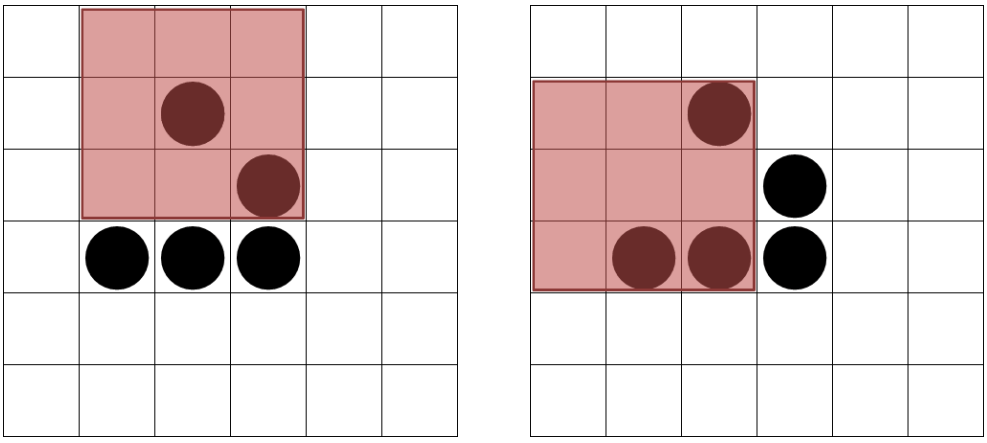
1. Qualquer célula viva com menos de dois vizinhos vivos morre, como se fosse por subpopulação.

2. Qualquer célula viva com dois ou três vizinhos vivos continua viva para a próxima geração.

3. Qualquer célula viva com mais de três vizinhos vivos morre, como se por superpopulação.

4. Qualquer célula morta com exatamente três vizinhos vivos torna-se uma célula viva, como se por reprodução.

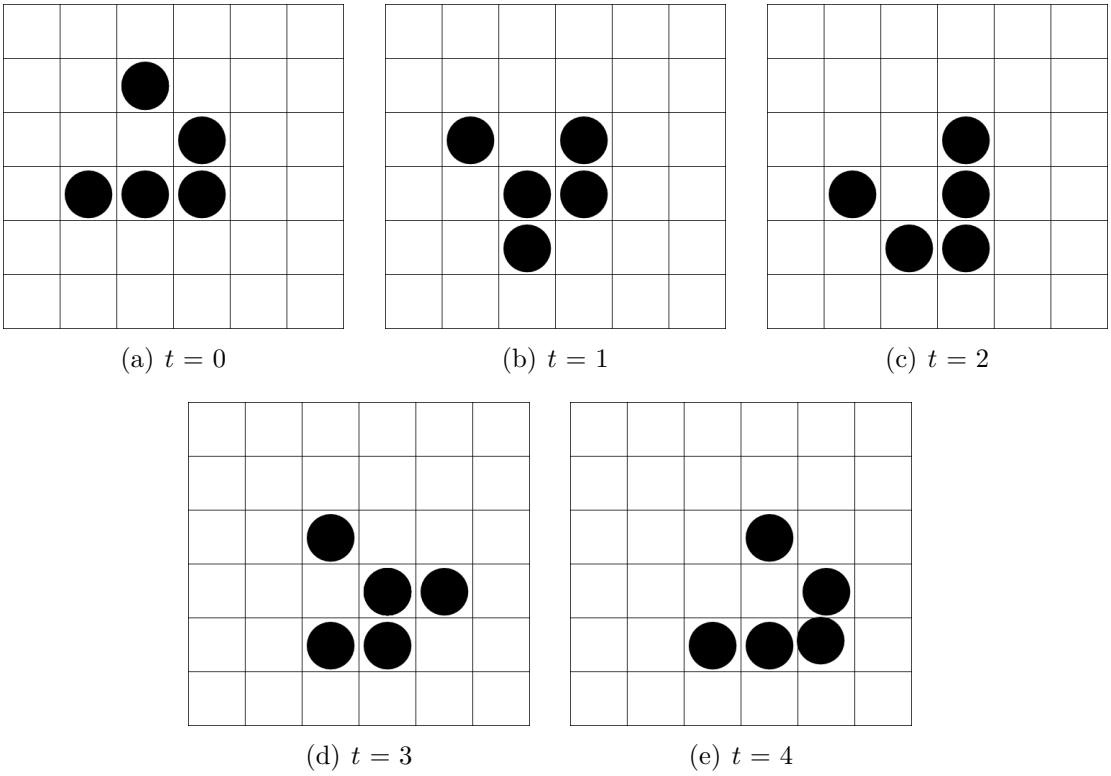
A partir de uma grade celular definida pelo usuário, que corresponde à semente do sistema, a cada passo de tempo, temos um novo reticulado, que será construído com base na regra de transição. A atualização do reticulado será feita pela aplicação da regra a cada célula dessa grade, verificando qual a vizinhança correspondente e o novo estado estabelecido pela regra acima. Assim, a grade celular original é atualizada e um passo de tempo é concluído obtendo-se um novo reticulado. Podemos observar na Figura 8, dois exemplos de atualização de célula, onde a vizinhança da célula central é destacada. Neste exemplo nós temos a configuração inicial ($t = 0$) da grade celular, sendo as células em branco células vivas, e células com o ponto preto, células mortas. Na Figura 8(a), pode-se ver o caso de uma célula central que está viva, mas possui menos de duas vizinhas vivas e, de acordo com a regra do *Life*, ela morrerá no próximo instante de tempo. Na Figura 8(b) temos uma célula central morta, mas que possui três vizinhas vivas, então ela passará a ser viva.



(a) Célula central viva (situação 1) (b) Célula central morta (situação 4)

Figura 8 – Grade inicial e vizinhanças destacadas no *Life*.

Quando o AC concluir essa verificação com cada célula da grade, ela é atualizada, o primeiro passo de tempo ($t = 1$) é concluído, e temos um novo reticulado (Figura 9(b)). A Figura 9 ilustra a evolução de um reticulado bidimensional por cinco passos de tempo, utilizando-se a regra do *Life* na atualização das células. É possível observar que, após 5 passos de tempo, o padrão observado no reticulado inicial volta a se repetir, porém com um deslocamento de 1 célula na diagonal. Se a regra for aplicada por diversos passos de tempo, esse padrão irá se repetir a cada 5 passos, como se fosse uma estrutura deslizante. Esse tipo de padrão é chamado de *Glider*.



(a) $t = 0$ (b) $t = 1$ (c) $t = 2$

(d) $t = 3$ (e) $t = 4$

Figura 9 – Evolução de um *glider*.

2.2 Navegação Autônoma de Robôs

A capacidade de executar determinadas tarefas de forma independente e adaptativa é uma das características desejadas nos robôs que atraem mais pesquisas em robótica. Nesse contexto, robôs autônomos são definidos como sistemas que existem no mundo físico, capazes de perceber seu ambiente, tomar decisões com base no que percebem ou foram programados para reconhecer, e em seguida, acionar um movimento ou manipulação nesse ambiente, a fim de alcançar alguns objetivos (MATARIĆ et al., 2007). Diversas tarefas são investigadas na robótica autônoma, como vigilância de ambiente, armazenamento e busca de produtos em centros de distribuição, busca e resgate e sistemas de entrega de mercadorias. Todos elas têm algo em comum que é a necessidade de criar uma rota

até determinado local, e por esse motivo, uma das tarefas mais investigadas em robótica autônoma é o planejamento de caminhos.

O planejamento de caminhos para robôs autônomos busca encontrar uma sequência de passos a serem aplicados para se obter o melhor caminho, ou a rota ótima, que parte da posição inicial do robô até o seu objetivo, evitando qualquer obstáculo que possa encontrar. O conceito de melhor caminho pode estar associado a diferentes critérios, por exemplo, ao caminho mais rápido, mais curto, ou aquele que minimiza a quantidade de giros e/ou frenagens. Nesse contexto, os algoritmos de planejamento normalmente precisam conhecer previamente o ambiente a ser navegado (CÂNDIDO; OLIVEIRA; MARTINS, 2018).

Esse problema também é conhecido como o problema de movimentação do piano (*Piano Mover's Problem*), pois o problema é similar ao de se movimentar um piano entre duas salas sem que ele colida com as paredes e obstáculos das salas. Uma definição mais formal é dado por um subconjunto U em um espaço n-dimensional e dois subconjuntos C_0 e C_1 de U, onde C_1 é derivado de C_0 por movimentação contínua, é possível mover C_0 até C_1 se mantendo completamente em U? (WEISSTEIN, 2019). Este problema é bastante difícil de ser resolvido, e fortes evidências indicam que a sua resolução é de complexidade exponencial em relação aos graus de liberdade do robô (BARRAQUAND et al., 1997).

A fim de estimar o posicionamento do robô no ambiente a partir de seus movimentos, uma das técnicas utilizadas em diversos trabalhos, incluindo o nosso, é a odometria (SANTANA et al., 2008). Ela é uma técnica amplamente utilizada para determinar o deslocamento de um robô pelo ambiente durante sua navegação, pois fornece informações de maneira simples e acessível (BORENSTEIN; FENG, 1994). A ideia fundamental da odometria é a integração da informação incremental do movimento das rodas ao longo do tempo. Entretanto, esse processo envolve um inevitável acúmulo de erros provenientes do ambiente e da base robótica. Esses acúmulos causam distorções na estimativa da posição, as quais aumentam proporcionalmente ao longo do percurso do robô. Apesar dessas limitações, acredita-se que a odometria é uma parte importante do sistema de navegação de um robô e que deve ser utilizada em conjunto com métodos de posicionamento absoluto (como o GPS) para proporcionar uma estimativa de posição mais confiável (CÂNDIDO; OLIVEIRA; MARTINS, 2018). Além disso, é essencial a utilização de métodos eficientes para a calibração do sistema de odometria. Esse sistema deve ser capaz de determinar valores aceitáveis para as principais fontes de incerteza nos cálculos de odometria. Neste trabalho, foi implementado o mesmo sistema de calibração adotado em (MARTINS et al., 2018), o qual é baseado no método University of Michigan Benchmark test (UMBMark) (BORENSTEIN; FENG, 1994).

A navegação de um robô autônomo, tem o objetivo de diminuir o congestionamento no ambiente, melhorar a segurança eliminando erros humanos e excluir a necessidade de operador, permitindo vários benefícios como aumento na produtividade (PENDLETON et al., 2017).

Nesse contexto, quando se trabalha com um time de agentes trabalhando em paralelo, o potencial para finalizar uma tarefa é maior em comparação com apenas um agente. Outra característica importante é que times de agentes autônomos introduzem redundância no sistema que faz dele mais robusto a falhas. Por exemplo, aqueles agentes que vierem a falhar, podem ser substituídos por outros agentes e o modelo continua funcionando corretamente, sendo essa uma das principais vantagens de se considerar mais de um agente para realizar uma mesma tarefa em conjunto.

Em ambientes com agentes autônomos, seu comportamento pode ser competitivo ou cooperativo. Situações nas quais agentes disputam uns contra os outros para melhor atender seus objetivos particulares, são caracterizadas por comportamento competitivo. Já situações onde agentes precisam atuar juntos a fim de completar o mesmo objetivo, são caracterizadas por comportamento cooperativo (SILVA et al., 2020).

Um sistema de navegação é composto por alguns componentes de hardware e um software. Os hardwares podem ser sensores para captura de dados de percepção do ambiente, uma câmera para captura de imagens do ambiente e atuadores para permitir a direção, aceleração e frenagem no ambiente. O software é composto por três categorias que possibilitam a coleta de dados e tomada de decisões do sistema: (*i*) percepção; (*ii*) planejamento; e (*iii*) controle. A Figura 10 apresenta uma visão geral do relacionamento entre os componentes de um sistema de navegação.

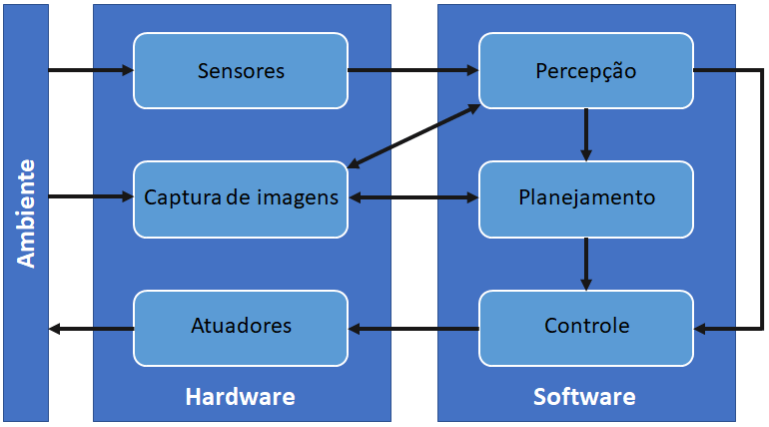


Figura 10 – Visão geral típica de uma arquitetura de um agente autônomo. Imagem adaptada de (ALVES et al., 2017).

- **Percepção:** capacidade de reunir e extrair conhecimento do ambiente;
- **Planejamento:** processo de tomada de decisões para atingir um objetivo, tal como movimentar o robô de sua posição inicial até a posição de seu objetivo, pelo melhor caminho;
- **Controle:** Capacidade do agente autônomo executar as ações planejadas que foram geradas pelos outros processos e eventos inesperados, tal como uma possível colisão ao se aproximar de um obstáculo.

Em (ALVES et al., 2017) foi elaborada uma taxonomia que distingue diferentes abordagens para o controle de agentes autônomos, como mostrado na Figura 11. Destas abordagens as principais para o relacionamento entre os membros do time são: (*i*) centralizada: quando existe um tomador de decisão que possui informações gerais do ambiente e de todos os outros agentes. Tal tomador de decisão pode ser um dispositivo a parte ou até mesmo algum dos próprios agentes com capacidade de comunicação; (*ii*) distribuída: quando não existe um tomador de decisão, uma vez que cada agente toma suas próprias decisões.

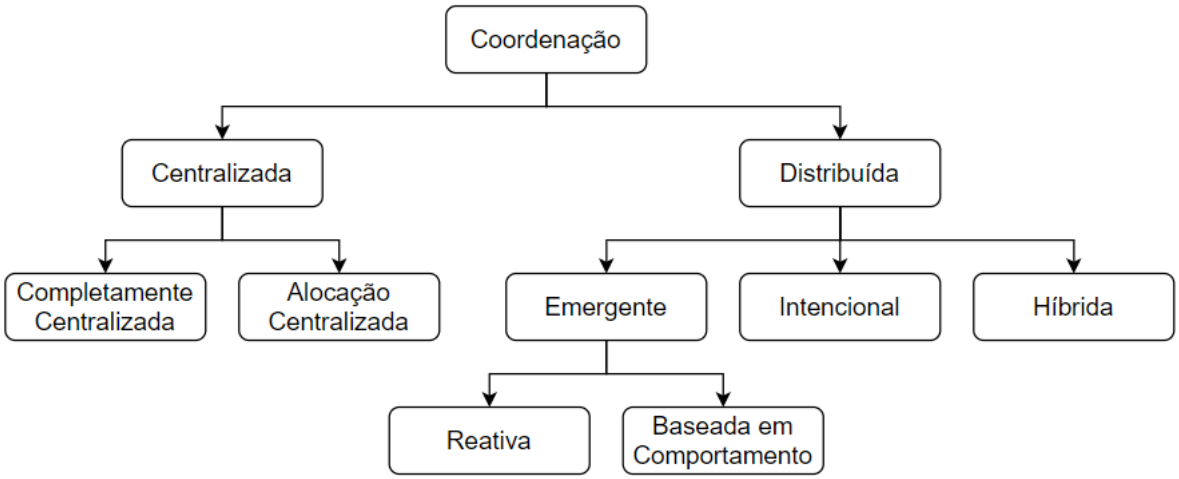


Figura 11 – Taxonomia. Abordagem de controle de navegação para agentes autônomos. Imagem adaptada de (ALVES et al., 2017).

A Tabela 1, realiza um comparativo entre as principais vantagens e desvantagens de cada abordagem apresentada nesta taxonomia.

Abordagem	**Descrição**	**Vantagens**	**Desvantagens**
Completamente Centralizada	único agente plano para o tempo inteiro;	potencial para ser ótimo; coordenação implícita;	geralmente intratável computacionalmente; único ponto de falha; resposta à mudanças;
Alocação Centralizada	único agente aloca tarefas para membros do time; membros do time completam tarefas individuais;	execução distribuída; alocação pode ser ótima;	computacional ainda é cara; ainda possui um ponto central de falha;
Reativa	agentes possuem um pequeno laço de perceber e agir;	extremamente rápida e simples;	não consegue lidar com tarefas complexas;
Baseada em Comportamento	usa informações sobre o estado atual para tomada de ações;	rápida e simples; agentes podem contribuir com múltiplas tarefas;	mais cara que a reativa; agentes ainda não conseguem planejar;
Intencional	comunicação com intenção de coordenação;	facilita escalonamento e planejamento;	lenta em situações de tempo crítico; muito dependente de comunicação;
Híbrida	maior intenção de coordenação;	permite uma melhor distribuição e planejamento de recursos; possui uma pequena coordenação;	não consegue realizar iterações complexas;

Tabela 1 – Comparativo. Abordagens Centralizadas vs. Distribuídas. Tabela adaptada de (ALVES et al., 2017).

A abordagem que melhor se adequa ao problema de planejamento de caminho para a navegação distribuída e descentralizada de um time de robôs tratado nesta pesquisa é a baseada em comportamento, na qual um sistema distribuído dá ao robô a capacidade de perceber e interpretar informações próximas a ele e utilizá-las para tomar suas próprias decisões sem troca de comunicação com outro robô do time.

Por último, um sistema de navegação autônoma composto por vários robôs pode ser classificado em 3 categorias:

- **Focados no controle de formação:** onde os robôs do time devem manter o máximo possível em uma determinada posição/forma em relação aos demais membros do time (ex: linha reta ou triangular);
- **Navegação em nuvem ou em bando (*flocking*):** onde os robôs devem se manter agrupados durante o deslocamento pelo ambiente;
- **Navegação aleatória:** onde a movimentação do robô considera apenas o cumprimento da sua missão, mas sem se preocupar com a sua posição em relação aos demais membros do time.

Nosso trabalho se encaixa nessa última categoria. Apesar de mais simples, a navegação aleatória ainda impõe desafios para o time, como a movimentação sem colisão entre os membros do time, os quais podem ser tratados como obstáculos móveis ou dinâmicos, e a necessidade de resolução de conflitos quando dois ou mais robôs têm suas trajetórias obstruídas por outros robôs.

2.2.1 Plataforma Webots

O WebotsTM (CYBERBOTICS, 2021a) é uma plataforma de simulação robótica de código aberto (licença Apache 2.0), que oferece prototipagem rápida de robôs móveis, flexibilidade na simulação de ambientes e modelagens realistas (DASGUPTA; WHIPPLE; CHENG, 2011). Seu projeto teve início em 1996, sendo desenvolvido por Olivier Michel do Swiss Federal Institute of Technology (EPFL) em Lausanne, Suíça. Em nosso trabalho, o simulador Webots foi utilizado na execução de experimentos com o modelo proposto, pois é uma plataforma de simulação que permite modelagens realistas de robôs e ambientes, incluindo parâmetros de diferentes sensores e variáveis físicas do mundo real. Portanto, aspectos da física são levados em consideração, tanto na arquitetura quanto no ambiente, permitindo, assim, uma análise mais realista de desempenho e adaptabilidade do modelo (GHAFFARI et al., 2004).

No geral, o simulador oferece uma maneira rápida e prática de criar ambientes confiáveis, que levam em conta comportamentos físicos, e permite a construção de controladores que podem ser transferidos diretamente para robôs reais. Além disso, a plataforma fornece alguns modelos de robôs pré-especificados, possibilitando a redução no tempo de

desenvolvimento (SILVA et al., 2015). Em nosso trabalho foi utilizada a versão 7.4.3 do Webots, cujo o ambiente de desenvolvimento pode ser visualizado na Figura 12. Do lado esquerdo da figura pode-se ver a árvore com cada componente do ambiente, no centro a janela com o ambiente de simulação com alguns obstáculos e um robô e-puck, ao lado direito à codificação do controlador do robô e abaixo o console de saída dos resultados.

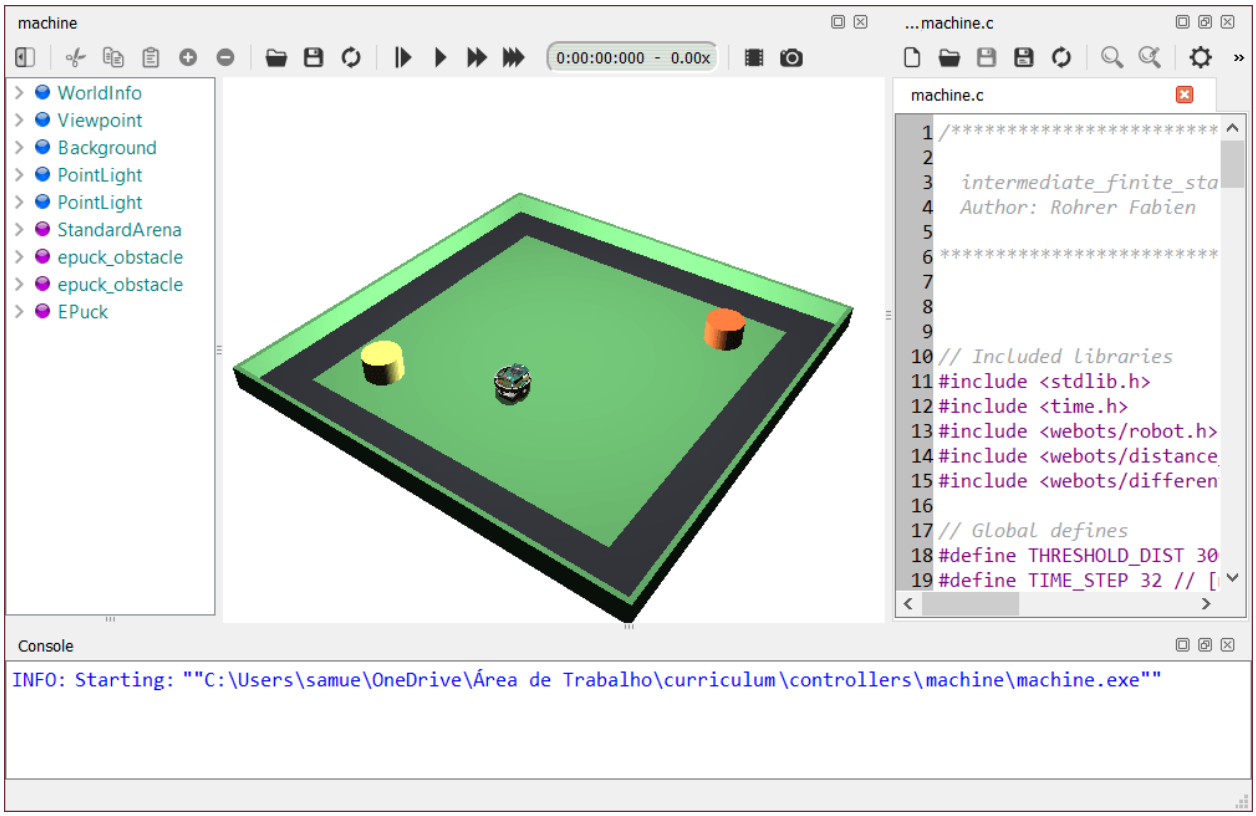


Figura 12 – Tela principal do simulador Webots.

2.2.2 Robô e-puck

Em nossos experimentos foi utilizada a plataforma robótica e-puck (MONDADA et al., 2009), projetada por Francesco Mondada e Michael Bonani, em 2006, no Instituto Federal Suíço de Tecnologia. Projetada inicialmente para ser uma ferramenta de auxílio ao ensino universitário, tornou-se um grande aliado em pesquisas. O projeto é baseado em um conceito de hardware aberto, no qual todos os documentos são distribuídos e submetidos a uma licença que permite que qualquer pessoa possa utilizá-lo ou participar de seu desenvolvimento. Da mesma forma, o software e-puck é de código aberto, fornecendo acesso de baixo nível a todos os dispositivos eletrônicos e oferecendo possibilidades ilimitadas de extensão (CYBERBOTICS, 2021a).

Os robôs e-puck possuem diversos sensores e atuadores, como pode ser observado na Figura 13. Desses, os principais utilizados em nosso trabalho foram: o *Bluetooth* para conexão e comunicação com algum dispositivo externo; o acelerômetro para medir a aceleração do robô; a roda e motor de passo para os cálculos de odometria e movimentação; e os sensores de distância infravermelhos. Os sensores de distância são responsáveis por identificar a proximidade entre o robô e algum obstáculo, em uma distância aproximada de 4 cm (CYBERBOTICS, 2021b). A Figura 13 também mostra a vista superior do modelo e-puck. A seta verde indica a frente do robô, as linhas vermelhas representam as direções dos sensores infravermelhos de distância e os rótulos correspondem aos nomes dos sensores de distância. Como pode-se observar, dos oito sensores dispostos no robô, quatro são posicionados na sua parte frontal (sensores ps0, ps1, ps7 e ps6), dois nas suas laterais, sendo um a esquerda (ps5) e outro a direita (ps2), e os dois últimos na parte traseira do robô (ps3 e ps4). A Figura 14 mostra a imagem real do robô próximo a outros objetos, para permitir a visualização de seu tamanho real.

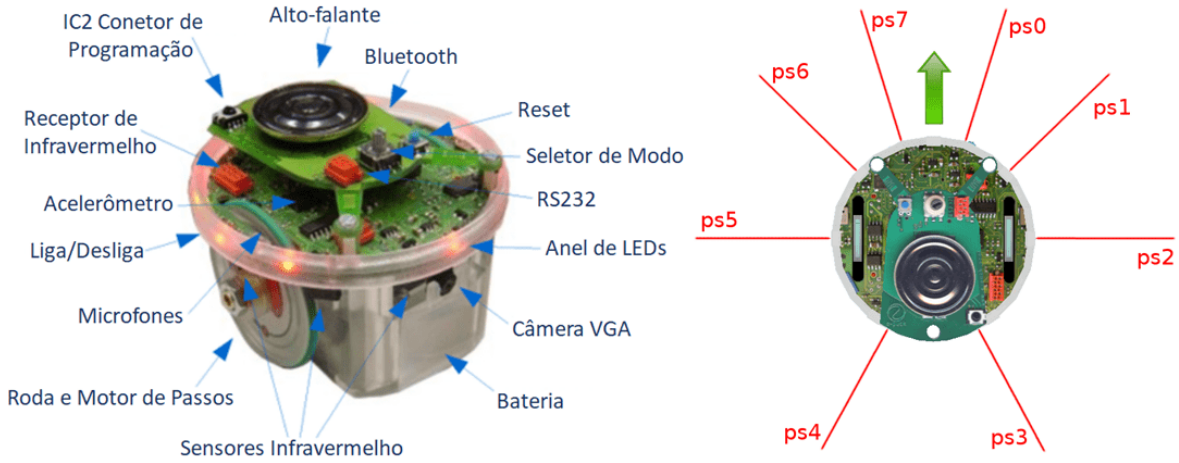


Figura 13 – Atuadores e sensores do robô e-puck. Imagem adaptada de (CYBERBOTICS, 2021b).

Figura 14 – Microrobô e-puck utilizado nos experimentos. (EPFL, 2021)

Capítulo 3

Trabalhos Correlatos

Diversas técnicas têm sido investigadas para a tarefa de planejamento de caminhos para robôs autônomos, dentre elas destacam-se na literatura: o uso de mapa de rotas (KAVRAKI et al., 1996), (ZHANG; FATTAHI; LI, 2013), decomposição celular (LINGELBACH, 2004), (RAMER; REITELSHÖFER; FRANKE, 2013) e campo potencial (BARRAQUAND; LANGLOIS; LATOMBE, 1992), (JIANJUN et al., 2013). No entanto, técnicas bioinspiradas como redes neurais (LUO et al., 2014) e autômatos celulares (AC) (MARTINS et al., 2018), (OLIVEIRA et al., 2019), também estão sendo aplicadas com sucesso ao planejamento de caminhos devido a sua estrutura descentralizada e baixo custo computacional. Neste trabalho são utilizados modelos baseados em AC e, desta forma, foi realizado um levantamento de diversos trabalhos disponíveis na literatura que também adotam essa técnica para o planejamento de caminho em aplicações para um ou vários robôs. Cada um desses trabalhos serão apresentados a seguir, organizados em seis categorias, como definido em (FERREIRA et al., 2014).

3.1 Difusão de Força

A primeira categoria é formada por abordagens baseadas na difusão de força. Em (SHU; BUXTON, 1995) é apresentado um modelo de planejamento de caminho livre de colisões para robôs móveis em que é utilizada uma representação distribuída do espaço de trabalho onde são apresentadas técnicas de processamento paralelo, com captura natural da geometria do mundo real. Dessa maneira, o ambiente onde o robô se movimenta foi discretizado em um array binário, onde 0 são células livres e 1 as células ocupadas por um obstáculo. O planejamento de caminho é baseado em um autômato celular que difunde as forças de atração para encontrar um caminho mais curto, entre a posição inicial até a final percorrendo espaços livres, podendo envolver movimentos translacionais em ambientes bidimensionais e de rotação em ambientes tridimensionais. O conceito de força de atração pode ser definido como a tendência de ir em determinada direção na vizinhança de von Neumann, ou seja, se uma célula tem força para ir ao norte, o robô pode se movimentar

naquela direção. A difusão pode terminar em duas situações. Na primeira situação, a difusão termina quando uma célula com obstáculo não adquire força para nenhuma direção e, então, o robô dá um passo em alguma direção que existe força. Já na segunda situação, a difusão é interrompida quando não existe mais espaço para difundir, nesse caso, concluise que não existe caminho até o objetivo. Não foram realizados experimentos reais, mas simulações mostraram que o método é efetivo e eficiente para um ambiente estático e como pode ser estendido para ambientes dinâmicos. Em (PÉREZ, 2020) também é proposto um modelo baseado em autômatos celulares para o problema de planejamento de caminho. Os obstáculos poligonais foram descritos através de conjuntos simples de vértices e armazenados em um reticulado unidimensional celular. Essa estratégia levou a minimizar a tesselação dos polígonos e também facilitou a elaboração e a modificação de mapas novos para os ambientes de teste. Cada uma das células do reticulado corresponde a uma unidade de informação por meio de um identificador numérico que corresponde a célula ocupada ou livre bem como o custo de transição em função da distância aos respectivos vizinhos. Cada célula também é associada ao respectivo valor de atração e repulsão em direção à posição do alvo. Esses valores são utilizados em metas-heurísticas criadas para a obtenção do melhor caminho. O algoritmo foi avaliado de acordo com três métricas que consideraram custos operacionais, desempenho de tempo e otimização das soluções. Os resultados retornados por este algoritmo foram satisfatórios e mostraram soluções ótimas com caminhos planejados a uma distância segura dos obstáculos do ambiente.

3.2 Modelagem em Camadas e Atração para o Objetivo

A segunda categoria engloba as abordagens baseadas em camadas e atração para o objetivo. A característica principal dessas abordagens é a utilização de um autômato celular de múltiplas camadas e movimentação restrita dos robôs (FERREIRA et al., 2014). Marchese (1996) publicou vários artigos relacionados a planejamento de caminhos utilizando autômatos celulares. Em (MARCHESE, 1996) é apresentado um algoritmo baseado em autômatos celulares para o planejamento do caminho para um único robô simples em um espaço de trabalho plano com obstáculos conhecidos *a priori*. O autômato celular é baseado na arquitetura de múltiplas camadas, ou seja, existe um reticulado para cada variável necessária para definir uma célula. No modelo proposto, o robô tem uma orientação em relação a sua direção original e a limitação de se mover para frente, com pequenos raios de curvatura para os lados, sem a possibilidade de parar ou realizar mudanças bruscas em sua trajetória. O processo de planejamento em si consiste em cinco fases. Na primeira, as células com obstáculos são aumentadas para suas vizinhas para evitar uma eventual colisão do robô com os obstáculos devido aos erros de odometria. Na segunda e terceira fase, são feitas as atualizações das direções nos estados inicial e

final, respectivamente. A quarta fase é a principal e consiste em definir a atração das células desde o objetivo até a célula inicial do robô, considerando as vizinhanças, a fim de explorar o ambiente para que seja possível construir um caminho da posição inicial até o objetivo. A ideia é decidir qual a célula e a direção em que o robô precisa estar para chegar na célula atual e na posição desejada (FERREIRA et al., 2014). Por exemplo, considerando a restrição de que o robô só pode mudar de direção em 45 graus, chegar na célula objetivo com o robô rotacionado para o norte só é possível a partir da célula ao sul, com o robô na direção norte; da célula ao sudeste, com o robô na direção noroeste; ou da célula ao sudoeste, com o robô a direção nordeste. Nesse caso, os valores relativos dessas direções, nas respectivas células da vizinhança são incrementados em 1 unidade, enquanto as demais direções permanecem inalteradas. A quinta e última fase é responsável pela construção da rota. A partir da posição e orientação inicial do robô e a restrição de movimento, verifica-se quais são as possíveis direções a serem seguidas e, caso exista mais de uma possibilidade, escolhe-se a de menor atração para o objetivo. Isso ocorre até que a célula objetivo seja encontrada. Diferentes versões do algoritmo foram implementadas, mostrando diferentes propriedades do modelo. Os resultados são apresentados apenas em simulações, mas o método é ilustrado em exemplos com características diferentes para demonstrar a possibilidade de uso em cenários reais. Esse modelo foi estendido em trabalhos posteriores. Em (MARCHESE, 2002), são introduzidos pesos para cada direção na camada de atração, o qual determina o custo de ações como: ir para frente, mudança de direção, voltar para trás, entre outros. Dessa forma, o algoritmo permite que o robô se movimente para a direção escolhida e não só para frente. Uma nova camada foi incorporada ao modelo apresentado em (MARCHESE, 2005). Essa camada representa o terreno de navegação, ou seja, consiste em um "mapa de elevação"discretizado para cada célula do reticulado. Dessa forma, o custo do movimento passa a depender da distância tridimensional entre as células, considerando o gradiente da superfície no seu cálculo. Nesse modelo também foi incluído um novo tipo de obstáculo, chamado obstáculo direcional, o qual permite a passagem do robô, mas apenas em uma determinada direção. Além de retornar rotas livres de colisões, o modelo se mostrou flexível e reativo, uma vez que pode ser usado para robôs com diferentes tipos de movimentos e formatos, bem como se adapta às modificações no ambiente (FERREIRA et al., 2014). Em (MARCHESE, 2008) e (MARCHESE, 2011), o modelo foi adaptado para lidar com um time de robôs heterogêneos. Apesar de lidar com robôs de diferentes formas, todos eles devem ter a mesma arquitetura interna, a qual é composta por módulos de localização, navegação, comunicação e planejamento. Nesses modelos, existe a figura do robô coordenador, o qual é responsável por planejar os movimentos de todos os robôs a fim de evitar colisões. Durante o planejamento da rota de um robô, os demais são tratados como obstáculos móveis pelo coordenador. Outra modificação é a inclusão da dimensão do tempo na camada de obstáculo, possibilitando que o modelo crie dinamicamente um campo de

repulsão para os obstáculos no ambiente. Por ser um planejamento centralizado, é definida uma ordem de prioridade entre os robôs, a qual é usada para a construção dos caminhos relativos. Os robôs de maior prioridade têm suas rotas planejadas primeiro e passam a ser vistos como obstáculos móveis para os demais. Isto é, a cada instante de tempo, a posição do robô com a rota planejada é atualizada na camada de obstáculos, de modo que seja considerada no próximos planejamentos. A principal contribuição desse modelo foi demonstrar a dualidade do problema de planejamento, ou seja, que ele pode ser resolvido da célula inicial até o objetivo ou vice-versa. Dessa forma, é possível reduzir o tempo de planejamento, realizando uma busca bidirecional, sem afetar a solução encontrada. O modelo também não foi avaliado com robôs reais.

3.3 Diagrama de Voronoi

A abordagem por diagrama de Voronoi a partir dos reticulados de autômatos celulares bidimensionais (TZIONAS; THANAILAKIS; TSALIDES, 1997). O foco desse tipo de abordagem não é encontrar o caminho mais curto, mas aquele que esteja o mais distante possível dos obstáculos do ambiente. Em (TZIONAS; THANAILAKIS; TSALIDES, 1997) é apresentado um novo algoritmo paralelo para o planejamento do caminho livre de colisão utilizando um autômato celular bidimensional, com vizinhança de von Neumann, para representar o ambiente e um diagrama de Voronoi construído a partir do reticulado do AC. O modelo foi avaliado considerando imagens discretizadas de ambientes formados por robôs na forma de diamante entre obstáculos com diferentes formas. O algoritmo proposto é baseado na retração do espaço livre sobre o diagrama de Voronoi, que é construído através da evolução temporal dos autômatos celulares. Inicialmente, ocorre a detecção das bordas dos obstáculos e a codificação de acordo com a sua orientação. Para isso, aplica-se uma iteração do autômato celular na imagem do ambiente adquirida de forma binária, onde são aplicadas um conjunto com 12 regras de transição sobre o reticulado, resultando em células que representam as regiões da imagem. Por exemplo, células com valores 1 e 2 representam regiões homogêneas, enquanto regiões com ruídos são definidas pelos valores 3 e 4. Essas regiões são desconsideradas na próxima etapa do algoritmo. Em seguida, é construído o diagrama de Voronoi, partindo-se do reticulado gerado. As fronteiras do diagrama determinam os caminhos mais distantes das bordas dos obstáculos, os quais devem ser utilizados pelo robô desde que ele seja menor que a largura existente entre os obstáculos. Os autores implementaram o algoritmo em um chip paralelo, demonstrando sua eficiência tanto no espaço quanto no tempo. Porém, neste trabalho também não houve experimentos com robôs reais.

3.4 Difusão da Distância à Meta

A quarta categoria envolve métodos baseados na difusão da distância à meta. A principal característica dessa abordagem é sua simplicidade, pois o planejamento consiste no cálculo da distância entre a posição do robô (ponto inicial) e a meta, utilizando um autômato celular. Em (BEHRING et al., 2001) é apresentada uma aplicação de autômato celular para o problema de planejamento de caminhos para robôs. Neste trabalho, pressupõe-se uma navegação sem erros, uma vez que o robô é considerado um único ponto sem orientação e sem aplicação das leis da dinâmica e da cinemática (FERREIRA et al., 2014). O ambiente é definido como um espaço bidimensional discretizado em células quadradas do tamanho do robô. O algoritmo recebe como entrada uma imagem contendo a posição do robô e de cada obstáculo no ambiente e o discretiza no reticulado do AC, onde cada célula pode assumir um dos quatro possíveis estados: *livre*, *obstáculo*, *posição_inicial* ou *objetivo*. Inicialmente, o autômato celular amplia as bordas dos obstáculos, visando reduzir o risco de colisão durante a navegação do robô. Em seguida, outras regras do AC são aplicadas com o objetivo de associar valores de distância em relação à célula objetivo a todas as células do reticulado. O cálculo do caminho é realizado pela aplicação sucessiva de duas funções simples de transição local, que parte da meta até a posição inicial do robô. O método foi testado experimentalmente com um pequeno robô em um ambiente real. Em todos os casos, foi possível obter bons caminhos com pouco esforço computacional. Esses resultados indicaram que a abordagem de autômatos celulares é um método promissor para o planejamento de caminho de robôs em tempo real. Em (TAVAKOLI; JAVADI; ADABI, 2008) duas contribuições foram feitas no trabalho de (BEHRING et al., 2001). A primeira delas foi a utilização de vários robôs no mesmo ambiente e compartilhando um mesmo objetivo. A segunda foi a inserção de custos na movimentação, uma vez que o movimento diagonal exige mais custo que um movimento em linha reta. Não foram realizados experimentos com robôs reais, porém, esse novo método foi comprovado através de simulações com vários agentes. Em (SOOFIYANI; RAHMANI; MOHSENZADEH, 2010), foi apresentado uma melhoria do trabalho de (TAVAKOLI; JAVADI; ADABI, 2008). A contribuição está em um modelo que dá preferência para movimentos em linha reta em relação a movimentos de zigue-zague. Esse comportamento é realizado ao não permitir movimentos na diagonal caso exista um obstáculo acima e ao lado da célula. Assim, o movimento do robô não é alterado enquanto não atingir o maior raio possível em linha reta. Foi mostrado através de experimentos computacionais que o custo de trabalho é melhor ou igual aos apresentados em (BEHRING et al., 2001). Outro trabalho que utiliza difusão da distância à meta foi proposto em (KOSTAVELIS et al., 2012). Nele, os autores combinam técnicas de visão 3D bem estabelecidas com operações de AC, a fim de traçar um caminho até o horizonte da imagem adquirida onde está a meta conhecida. Primeiramente, é obtido o mapa de profundidade do ambiente e, a partir dele, uma etapa de processamento da imagem é realizada para distinguir os pixels pertencentes ao chão e

aos obstáculos. Em seguida, aplica-se uma transformação polar onde tem-se uma distribuição espacial dos obstáculos em volta do robô. Por fim, é gerado um reticulado celular que representa as distâncias das regiões que são percorríveis pelo robô nesse ambiente. A célula para onde o robô deve se mover é obtida através da aplicação de um conjunto de regras adicionais sobre o reticulado, revelando a rota que o robô deve seguir para chegar ao seu objetivo. O método apresentado foi avaliado em um conjunto de dados externos e provou ser capaz de produzir caminhos livres de colisão entre a localização do robô e a meta selecionada. No trabalho de (PEI; LOU; YE, 2018) foi verificado que um AC com vizinhança de Moore pode evoluir para posições diagonais, permitindo um caminho entre obstáculos, de forma que é proposto um AC com vizinhança mista para o problema de planejamento de caminho de um robô, para evitar a evolução do AC em posições diagonais, que possibilita caminhos que passam entre obstáculos. A teoria de von Neumann é adotada para células livres em locais que envolvem passagem entre obstáculos. A vizinhança de Moore é adotada para células livres nas demais regiões. Existem apenas 2 estados de célula, sendo 0 para célula livre e 1 para célula obstáculo. Através desses dois valores, o reticulado é classificado de acordo com 16 padrões de arranjos de 4 células, que representam todas possibilidades de obstáculos nesse espaço celular. Quando o padrão que representa o problema é encontrado no reticulado, as células livres nele serão alterados do valor 0 para -1. O valor -1 significa que a célula vai adotar a vizinhança de von Neumann. Para as células onde não houve problema, seu estado permanece livre e mantém o valor 0. O valor 0 significa que a célula vai adotar a vizinhança de Moore. Porém, a célula -1 continua sendo livre assim como a célula 0. A evolução do AC ocorre da célula objetivo até a célula inicial através de uma regra de transição para cada tipo de vizinhança a ser aplicada. Como o método proposto neste artigo lida apenas com as áreas onde o problema existe, e não há restrições nas áreas sem problema, foi possível observar através de experimentos em ambiente computacional, que é possível obter caminhos mais curtos e livres de colisão. Os resultados da simulação mostram que o método proposto é eficaz.

3.5 Regras de Atualização Local

O uso de sensores e decisões locais caracterizam a quinta categoria. Nesse tipo de abordagem, a cada passo de tempo, os sensores avaliam a vizinhança do robô e, a partir das informações coletadas, o autômato celular determina para qual célula se deve ir no próximo passo de tempo. O método desenvolvido em (AKBARIMAJD; LUCAS, 2006) emprega uma abordagem de abstração que torna a complexidade do problema gerenciável. Ele utiliza um único robô em um ambiente desconhecido. Esse ambiente é representado por um reticulado de células, cada uma podendo assumir um dos possíveis estados: *robô*, *livre* e *obstáculo*. As ações realizadas pelo robô são definidas pelo AC com base na configuração

corrente do reticulado, ou seja, nos estados atuais das células na vizinhança do robô. A cada passo de tempo, o robô avalia sua vizinhança por meio dos seus sensores e se move para alguma de suas células vizinhas, definida de acordo com a regra de transição do AC. Sua arquitetura de execução é baseada em camadas. No centro está a representação geométrica do espaço em torno do robô. O método decide, a partir de ações de alto e baixo nível, o que deve ser feito. As ações de alto nível são orientadas ao objetivo, enquanto as de baixo-nível são relativas ao desvio de obstáculos. No modelo também existe um controlador central, responsável por gerenciar o funcionamento do sistema, envolvendo a operação dos robôs, a temporização e funcionamento dos sensores, entre outros. Apesar da arquitetura ter sido embarcada em um robô móvel, sua viabilidade foi mostrada apenas em simulações. Em (AKBARIMAJD; HASSANZADEH, 2011) é proposto método para o planejamento em tempo real de caminhos para um time de robôs em ambientes dinâmicos. Ele é baseado em autômatos celulares bidimensionais, onde o ambiente é discretizado em um reticulado retangular, cujo as células podem assumir quatro possíveis estados: *robô*, *livre*, *obstáculo* ou *objetivo*. Essa última determina a célula com o melhor objetivo direcionado. As regras de evolução do AC são propostas de tal forma que, a cada passo de tempo a célula do robô é trocada pela célula com a melhor direção. Apenas simulações foram realizadas, mostrando que o algoritmo foi capaz de encontrar o caminho apropriado e o mais curto. Esse trabalho foi estendido em (AKBARIMAJD; HASSANZADEH, 2012), onde o método foi atualizado para aplicações em ambientes com objetos côncavos e não apenas convexos, como no trabalho anterior. Em (RODRÍGUEZ-SEDA; RICO, 2019) é apresentado um algoritmo descentralizado para a navegação cooperativa de uma equipe de robôs móveis e obstáculos estáticos baseada em autômato celular, para garantir a prevenção de colisões e reduzir a ocorrência de deadlocks entre os agentes. O algoritmo assume que os agentes podem detectar a posição de outros agentes e obstáculos dentro de uma vizinhança de Moore de alcance 2, enquanto sua amplitude de movimento é a vizinhança de Moore de alcance 1. Cada célula desse espaço de trabalho pode ser ocupada por um agente, obstáculo ou ser livre. Os agentes se movem em uma célula de cada vez em qualquer direção de acordo com um conjunto de regras de transição predefinidas como base na sua vizinhança. O algoritmo cooperativo foi projetado para garantir a prevenção de colisão e reduzir a ocorrência de deadlock entre os agentes, contudo, o algoritmo não garante a prevenção de deadlock em todos cenários. Várias simulações foram executadas em ambientes computacionais mostrando que agentes são capazes de convergir para o seu destino na maioria das vezes, principalmente quando a densidade de agentes e obstáculos é pequena.

3.6 Envio de Mensagens

A sexta e última categoria engloba as abordagens baseadas no envio de mensagem. Em (ROSENBERG, 2007) é proposto um modelo baseado em autômatos celulares para o planejamento de caminho de um time de robôs, inspirado no comportamento das formigas. Como no mundo real, podem existir várias formigas no ambiente, sendo que cada uma representa um robô capaz de andar pelo ambiente sem colidir com obstáculos ou outros robôs, em busca de alimento e de um local determinado para estacionar. Nessa abordagem é mantida uma configuração padrão para todos os robôs (formigas) e inverte-se a inteligência da busca, ou seja, ela está no ambiente e não no agente. Portanto, as células comunicam-se localmente, com suas vizinhas, por meio de mensagens contendo a existência de obstáculos, formigas, alimento ou de feromônio. Essas mensagens também envolvem a comunicação com as formigas, a fim de executar ações como capturar o alimento ou mover-se para determinada direção. Nesse processo, os robôs são agentes passivos, ou seja, em vez de tomarem decisão sobre o próximo passo a ser executado, eles apenas recebem essa informação do ambiente e a executam. Durante as simulações, os autores identificaram a necessidade de melhorar o envio de mensagens, adicionando um valor de feromônio junto com as mensagens de alimento, pois esse valor diminui a cada retransmissão, dessa forma as formigas tendem a encontrar os alimentos mais próximos. Em (ROSENBERG, 2008) foi publicada uma continuação de (ROSENBERG, 2007). Aqui o problema de encontrar uma saída em um labirinto é incrementada ao problema de busca por comida, ou seja, as formigas também se movimentam em busca de comida ao invés de só esperarem as mensagens de direção do alimento. Outro trabalho baseado no comportamento das formigas foi proposto em (TINOCO et al., 2019), com o objetivo de coordenar um enxame de robôs, com foco nas tarefas de vigilância e exploração de ambientes. A decisão do próximo movimento dos robôs é realizada com o auxílio de feromônio repulsivo, que permite a busca por áreas não visitadas recentemente. Nessa modelagem, cada robô mantém um mapa independente de feromônio que é atualizado continuamente a cada movimento do próprio robô, ou quando ele estiver trafegando em regiões próximas a outros robôs. Dessa forma, não é necessário utilizar um ambiente inteligente como proposto no trabalho anterior (ROSENBERG, 2007), podendo eliminar a utilização do coordenador central, responsável por armazenar um mapa global que precisa ser compartilhado constantemente com todos robôs. Simulações mostraram que o modelo foi capaz de realizar a coordenação descentralizada do time e obteve um bom desempenho em sua tarefa de vigilância.

3.7 Modelos Anteriores Relacionados à Proposta

Nosso trabalho combina duas dessas abordagens. A primeira delas é a abordagem da técnica por difusão da distância à meta, de onde alguns trabalhos serviram como base para o modelo proposto. O primeiro trabalho foi o de (BEHRING et al., 2001), discutido anteriormente. Ele é um modelo de planejamento de caminhos baseado nas regras locais de AC que foram refinadas posteriormente nos modelos de (OLIVEIRA; VARGAS; FERREIRA, 2015a) e (MARTINS et al., 2018).

Em (OLIVEIRA; VARGAS; FERREIRA, 2015a) é proposto e avaliado um modelo de planejamento de caminho para um robô autônomo. O objetivo é construir um caminho livre de colisões desde a posição inicial do robô até seu objetivo, aplicando um modelo de AC melhorado sobre uma imagem pré-processada do ambiente capturada durante a navegação. Inicialmente, o ambiente é discretizado e transformado em um reticulado celular com células de 14 cm. Cada célula pode assumir um dos quatro valores possíveis: *Livre*, *Obstáculo*, *Inicial* e *Objetivo*. O algoritmo de planejamento é dividido em três fases. A primeira fase do planejamento de caminho provoca a ampliação dos obstáculos inicialmente identificados, visando não apenas compensar os efeitos da dinâmica não considerada no modelo, mas evitar colisões de obstáculos durante a navegação do robô. A segunda fase distribui a distância do objetivo para cada célula livre. A última fase constrói o caminho entre a posição atual do robô e a meta, considerando todas as distâncias calculadas na segunda fase para escolher uma rota mais curta e livre de colisão. Como os erros de odometria acumulados ao longo do caminho do robô tendem a tirar o robô de sua rota planejada originalmente, a cada n passos de tempo o planejamento de caminho é recalculado para aproximar a rota planejada daquela efetivamente executada e evitar possíveis colisões. Simulações mostraram resultados promissores no desempenho de um único robô.

Em (MARTINS et al., 2018), os autores refinaram o modelo discutido em (OLIVEIRA; VARGAS; FERREIRA, 2015a). Inicialmente, a resolução do reticulado usado na representação da imagem foi ajustada, alterando as dimensões das células para 7 cm que corresponde ao tamanho do robô, possibilitando que ele navegue entre espaços estreitos e mais complexos. Entretanto, a principal contribuição foi a criação de um novo método alternativo para reduzir a imprecisão da odometria. O desenvolvimento do método proposto foi motivado pela percepção de que, nos modelos anteriores, considera-se que o robô está sempre no centro da célula para determinar a próxima etapa do robô. No entanto, na maioria das vezes, o robô está deslocado em sua célula atual devido aos erros de odometria, principalmente os não sistêmicos. Assim, o movimento do robô em direção a próxima célula é realizado sem considerar tal erro, propiciando o seu acúmulo ao longo da navegação. O método busca corrigir esse problema ao aplicar um cálculo extra durante o recálculo da rota, com a finalidade de reposicionar o robô no centro da célula corrente, antes de retomar seu trajeto. Esse processo é realizado a cada x passos de tempo,

quando um novo planejamento de caminho é requisitado. Simulações e experimentos com robô real (e-puck) foram realizados, mostrando resultados promissores no desempenho do modelo para um único robô.

A segunda abordagem utilizada em nosso modelo é a regra de atualização local. Dentro dessa abordagem nosso trabalho se baseou parcialmente no modelo proposto por (OLIVEIRA et al., 2019). Esse por sua vez é uma melhoria do modelo investigado originalmente em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2008) e (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011) que também usam a abordagem por atualização local.

Em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2008) é apresentado um algoritmo de autômato celular para resolver o problema de planejamento de caminho em um sistema multiagente, visando a navegação por um ambiente desconhecido. Para isso, são usadas as leituras dos sensores do robô a fim de perceber o ambiente em sua volta (vizinhança) e, a partir desses dados, realizar o planejamento. A equipe de robôs tem a capacidade de se mover em determinadas formações, mais especificamente, foi apresentado um estudo de caso que adota uma formação em linha reta. Além disso, com base no método do AC proposto, os robôs podem se comunicar e realizar tarefas cooperativas, como a troca de posições na formação da equipe, com o objetivo de manterem sua formação inicial caso ocorra um evento inesperado no ambiente. O algoritmo foi implementado em um sistema de tempo real composto por 3 robôs móveis autônomos semelhantes. Os resultados das simulações e com o robô real mostram que o método proposto é eficaz, confiável e pode ser usado em sistemas com mais de três robôs. Esse trabalho foi aprimorado (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011), adequando o modelo para realizar, de forma distribuída, o planejamento de caminhos para uma equipe de robôs. Como no trabalho anterior, esse novo modelo também foca na prevenção de colisão e controle de formação. Todos os robôs devem ajustar suas ações para manter-se, o máximo possível, na formação inicial da equipe. Duas formações diferentes foram testadas (triangular e em linha reta) a fim de estudar a eficiência e a flexibilidade do método. O autômato celular apresentado foi implementado e avaliado em um sistema real usando três robôs E-pucks. Resultados experimentais indicaram que caminhos livres de colisão e precisos podem ser criados com baixo custo computacional. Além disso, as tarefas de cooperação podem ser realizadas usando recursos mínimos de hardware, mesmo em sistemas com robôs de baixo custo como os E-pucks. Foi proposto em (OLIVEIRA; VARGAS; FERREIRA, 2015b), uma adaptação de (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2008) para resolver o problema de planejamento de caminho baseado em regras de transição de autômatos celulares. Esse novo modelo foi aplicado em cenários envolvendo apenas um robô, por onde foi possível notar a ocorrência de um problema que impedia um robô e-puck de completar o desvio de um obstáculo, chamado de conflito de esquina, onde o robô ao se aproximar de um obstáculo ficava indefinidamente alternando entre movimentos de giro a 90° e -90°. Usando apenas 3 estados o modelo proposto em (FERREIRA; VARGAS; OLIVEIRA,

2014), consegue superar o obstáculo, porém, com um comportamento ineficiente, que levava a muitas rotações e movimentos em zigue-zague. O novo modelo proposto por (OLIVEIRA; VARGAS; FERREIRA, 2015b) elaborou 8 estados e conjunto de regras mais complexas para resolver o problema. Experimentos realizados usando robôs e-pucks reais mostraram que o novo modelo foi capaz de corrigir o impasse da esquina dos obstáculos apresentando um bom desempenho.

Em (OLIVEIRA et al., 2018) e (OLIVEIRA et al., 2019), os autores apresentam um modelo de planejamento de caminhos autônomo e descentralizado para time de robôs, baseado nos trabalhos anteriores ((IOANNIDIS; SIRAKOULIS; ANDREADIS, 2008), (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011) e (OLIVEIRA; VARGAS; FERREIRA, 2015b)). Neste novo algoritmo, regras de decisões locais são utilizadas para definir o próximo movimento, mas apenas com base na vizinhança dos robôs e na comunicação com outros robôs próximos. Além disso, o controle de formação deixou de ser realizado de forma centralizada e foram aprimoradas algumas das principais deficiências dos modelos anteriores, como o número expressivo de rotações e trajetórias em zigue-zague, os quais comprometem a manutenção da formação da equipe. O método proposto foi implementado no simulador Webots e em um sistema real com 3 robôs e-puck em formação linear. Os resultados mostraram melhorias na eficiência geral da equipe e robustez em diferentes cenários, exigindo pouca comunicação robô-robô.

Capítulo 4

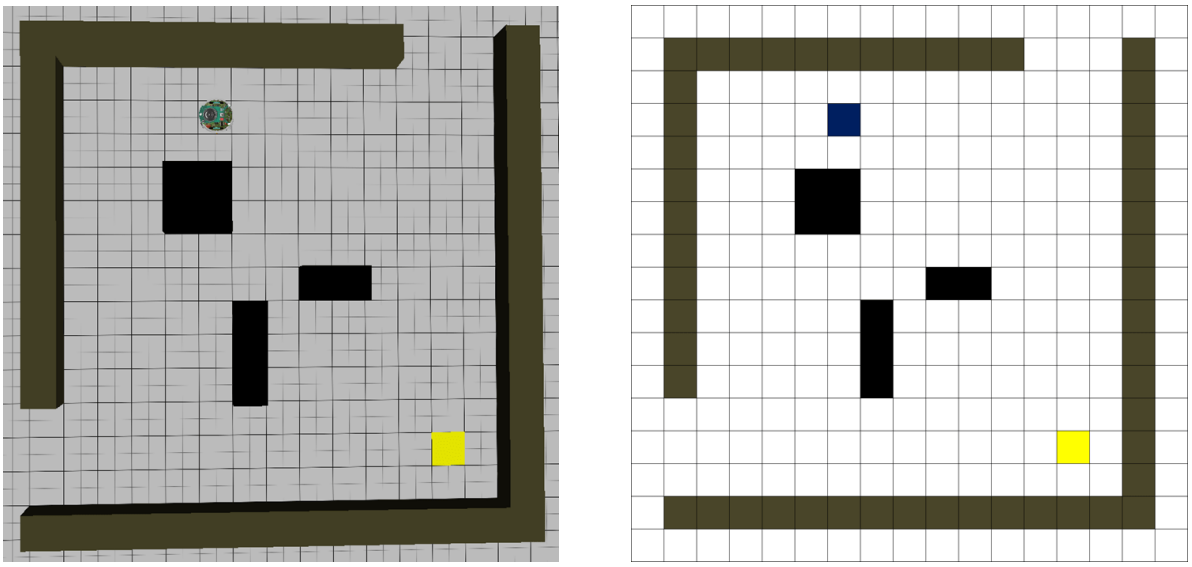
Modelo proposto

Neste trabalho, é proposto um novo modelo de planejamento de caminho e navegação de times de robôs baseado em regras de autômatos celulares (ACs) e fluxo de decisões locais. O modelo é chamado de BioTeam: modelo distribuído bio-inspirado para planejamento de caminho e navegação de times de robôs. O modelo proposto é inspirado nos modelos anteriores apresentados em (OLIVEIRA; VARGAS; FERREIRA, 2015a) e (MARTINS et al., 2018), que foram aplicados apenas à navegação de robôs individuais. Na presente proposta, cada robô é capaz de tomar suas próprias decisões sem a necessidade de um robô controlador e sem comunicação direta entre os robôs do time. Cada robô tem seu próprio objetivo, que deve ser alcançado evitando colisão com qualquer obstáculo ou outros robôs no ambiente, de modo que não prejudique o comportamento de outros membros do time.

O BioTeam é dividido em três módulos: (*i*) Módulo de captura e processamento de imagem (CPI); (*ii*) Módulo de navegação distribuída (ND); e (*iii*) Módulo de planejamento de caminhos (PC). Os módulos ND e PC são embarcados em cada robô do time e o módulo CPI está localizado em um dispositivo externo. O relacionamento entre os módulos do sistema pode ser observado no fluxograma da Figura 16.

O módulo CPI não é detalhado nesta dissertação, pois está fora do escopo deste trabalho. Entretanto, para facilitar o entendimento do modelo, é feita uma breve explicação do funcionamento proposto para esse módulo e sua interação com o restante do sistema. Basicamente, o módulo CPI pode ser acionado a qualquer momento da navegação, pelo módulo ND de cada robô. No início da navegação, o CPI é responsável por capturar uma imagem do ambiente, a partir de um dispositivo externo (por exemplo, uma câmera de celular) e processar essa imagem para construir um mapa de representação do ambiente. Cada robô possui um mapa interno individual, que pode ser acessado pelos módulos ND e PC. Esse mapa é discretizado em um espaço 2D, que representa a grade (ou reticulado) do autômato celular (AC). A grade celular é dividida em células quadradas de forma que o robô ocupe uma única célula. O módulo CPI define, de acordo com as informações processadas da imagem capturada, o estado de todas as células da grade. A Figura 15

exemplifica a discretização de um mapa (Figura 15(b)) a partir de um ambiente simulado na plataforma Webots (Figura 15(a)), que contém as células que representam as paredes em marrom e os obstáculos em preto; o robô em azul representa sua posição inicial e a localização de seu objetivo em amarelo. Inicialmente, apenas os estados *livre*, *posição_robô*, *objetivo*, *parede* e *obstáculo* são possíveis, descrevendo as informações capturadas pela imagem. Ao longo da execução, os estados das células livres podem ser alterados, tanto pela aplicação de regras de atualização de autômatos celulares, quanto pela aquisição de uma nova imagem, que pode vir a ser requisitada pelo módulo ND. No caso de requisição de imagem durante a navegação, o módulo CPI atualiza as informações apenas na vizinhança do robô. As informações das posições das paredes e obstáculos fixos do ambiente não são alteradas durante a navegação. No total, o estado da célula tem onze valores possíveis: *livre*, *posição_robô*, *objetivo*, *parede*, *parede_virtual*, *obstáculo*, *obstáculo_virtual*, *região_concava*, *robô_superior*, *robô_inferior* e *robô_virtual*. Os valores *parede_virtual*, *obstáculo_virtual*, *robô_virtual* e *região_concava*, são criados ao longo do processo de planejamento da rota, pelas regras de ACs que utilizam a vizinhança de Moore. Essas regras são apresentadas na Seção 4.1. Os estados *robô_inferior* e *robô_superior* são criados durante uma requisição de imagem pelo módulo ND para decisão de manobra, e representam a existência de outros robôs na vizinhança do robô em análise. Esses podem ter uma ordem de prioridade superior ou inferior ao robô. Essa classificação depende de uma ordem pré-estabelecida de prioridade entre os robôs, que é fornecida como parâmetro do sistema. As manobras executadas pelo módulo ND são apresentadas na Seção 4.2.



(a) Ambiente simulado (b) Mapa discretizado

Figura 15 – Exemplo de discretização de um ambiente simulado.

Embora não tenha sido efetivamente implementado, o módulo CPI do BioTeam é similar ao sistema de captura e imagem implementado em (MARTINS et al., 2018). Nesse trabalho precursor, um sistema de captura de imagens via celular foi implementado para permitir a aquisição de imagens para a navegação de um único robô. Para a adequação desse sistema prévio ao novo BioTeam, seriam necessárias algumas atualizações, como a identificação de outros robôs na vizinhança do robô que solicitou a imagem. Entretanto, a implementação de experimentos com robôs reais foge ao escopo do presente trabalho. Nos experimentos com a plataforma Webots descritos no Capítulo 5, o módulo CPI foi simulado de uma forma simplista: no início da execução, cada robô recebe um mapa interno previamente preparado do ambiente a ser executado, com a posição inicial do robô, dos obstáculos, das paredes e da meta. Após o início da navegação, sempre que uma nova imagem é requisitada por algum robô, a sua posição é atualizada a partir da leitura do dispositivo GPS, disponível no Webots para a arquitetura simulada. Além disso, caso um obstáculo seja identificado na vizinhança, a partir da leitura dos sensores do robô, a leitura do GPS dos demais robôs é feita para avaliar se um dos membros do time encontra-se próximo ao robô em análise.

O fluxo básico de execução do sistema é descrito a seguir. Primeiramente o módulo ND, responsável por controlar a navegação do robô, solicita o mapa interno que representa o ambiente, ao módulo CPI. O módulo CPI captura e processa a imagem, construindo o mapa interno. Após receber o mapa, o módulo ND requisita para o módulo PC a rota planejada com base nas informações do mapa. Por fim, com a rota planejada, o robô se move pelo ambiente até seu objetivo e aciona seus sensores de proximidade durante seu deslocamento. A estimativa de posicionamento do robô durante a navegação é realizada através da técnica de odometria. Entretanto, dependendo da leitura dos sensores, o módulo ND pode requisitar para o módulo CPI, ao longo da navegação, a captura de novas imagens e a atualização do mapa interno. Com essas informações atualizadas, o módulo ND pode decidir realizar manobras em situações de conflito, por exemplo, na aproximação indevida de obstáculos fixos, devido a erros de navegação/odometria, ou na aproximação de outros robôs do time, devido a conflito de trajetórias. Dependendo da manobra escolhida, o módulo ND também pode solicitar o planejamento de novas rotas para o módulo PC.

A finalização do módulo de navegação depende do fluxo proposto para o experimento. Três tipos de experimentos de navegação foram executados. No primeiro tipo, existe apenas um robô e após chegar na meta planejada, o mesmo encerra sua navegação. No segundo tipo, existem vários robôs no time e cada um recebe sua meta individual no início e uma nova meta após alcançá-la, que é a sua posição inicial. Dessa forma, cada robô realiza um movimento de vai e volta e o experimento é finalizado quando todos conseguirem concluir pelo menos um ciclo de vai e volta. O terceiro tipo é similar ao segundo, a única diferença é que o experimento é finalizado por tempo de execução.

Nas seções seguintes, os módulos PC e ND são descritos em detalhes.

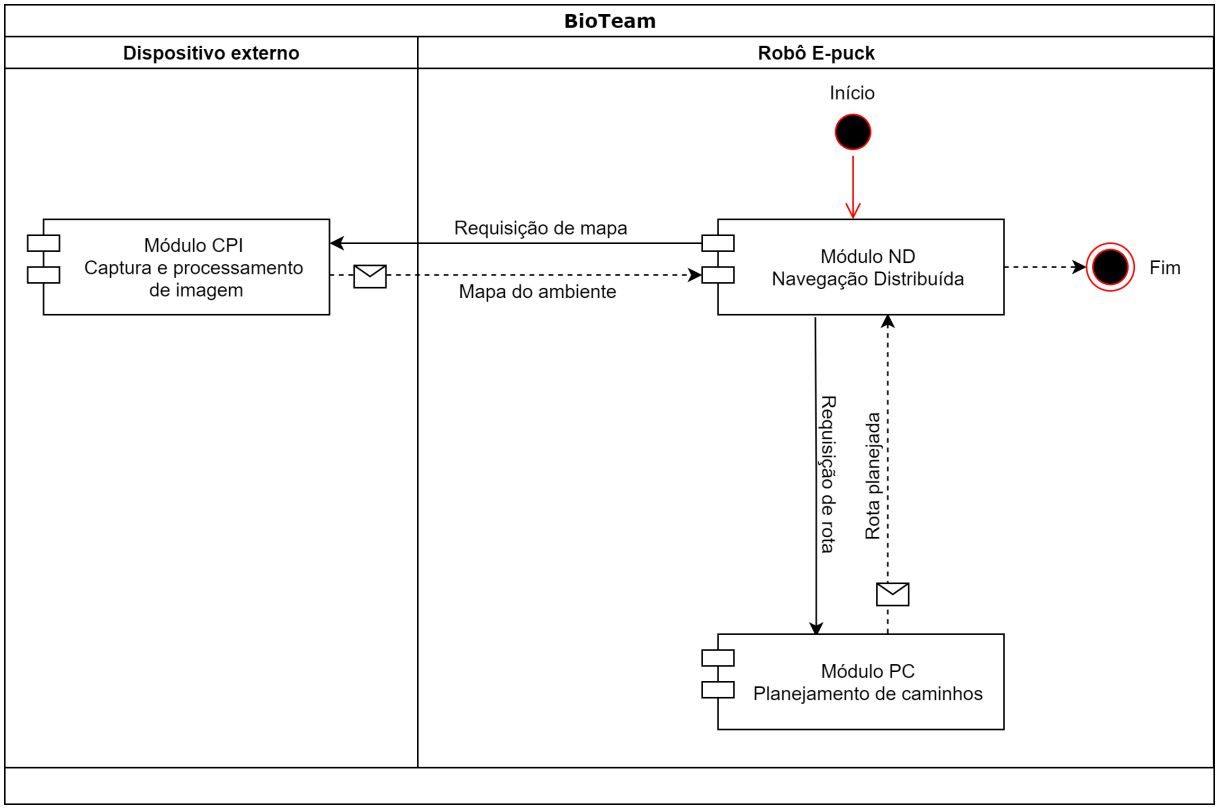


Figura 16 – Fluxograma do modelo distribuído bio-inspirado para planejamento de caminho e navegação de times de robôs (BioTeam).

4.1 Planejamento de caminho

O módulo de planejamento de caminho (PC) é o responsável por preparar o ambiente e planejar uma rota livre de colisão para cada robô, até seu objetivo. Este módulo é dividido em quatro fases: (*i*) Alargamento dos obstáculos e paredes internos; (*ii*) Sombreamento de regiões côncavas de obstáculos; (*iii*) Propagação da distância até o objetivo; e (*iv*) Construção da rota livre de colisão até o objetivo.

Sua execução pode ser observada no fluxograma da Figura 17. Ao receber uma requisição de rota feita pelo módulo ND, o módulo PC deve verificar se a grade celular do mapa interno do ambiente já foi preparada. Essa verificação é necessária pois as etapas de preparação da grade celular (alargamento dos obstáculos e paredes internos e sombreamento de regiões côncavas de obstáculos) devem ser aplicadas antes das etapas que definem a rota (propagação da distância e construção da rota livre de colisão até o objetivo). Uma vez que o ambiente já foi preparado, esse processo não precisa ser refeito. Na primeira requisição de rota, quando o sistema está iniciando, todas as etapas deste módulo são aplicadas.Ao longo do processo de navegação, quando o robô obtém um novo objetivo

e uma nova rota para o objetivo é requisitada, apenas as etapas que definem a rota são aplicadas. Em ambas as situações, o resultado final é a rota planejada que é utilizada pelo módulo ND para movimentar o robô. Cada uma das fases do módulo de planejamento de caminho é explicada a seguir.

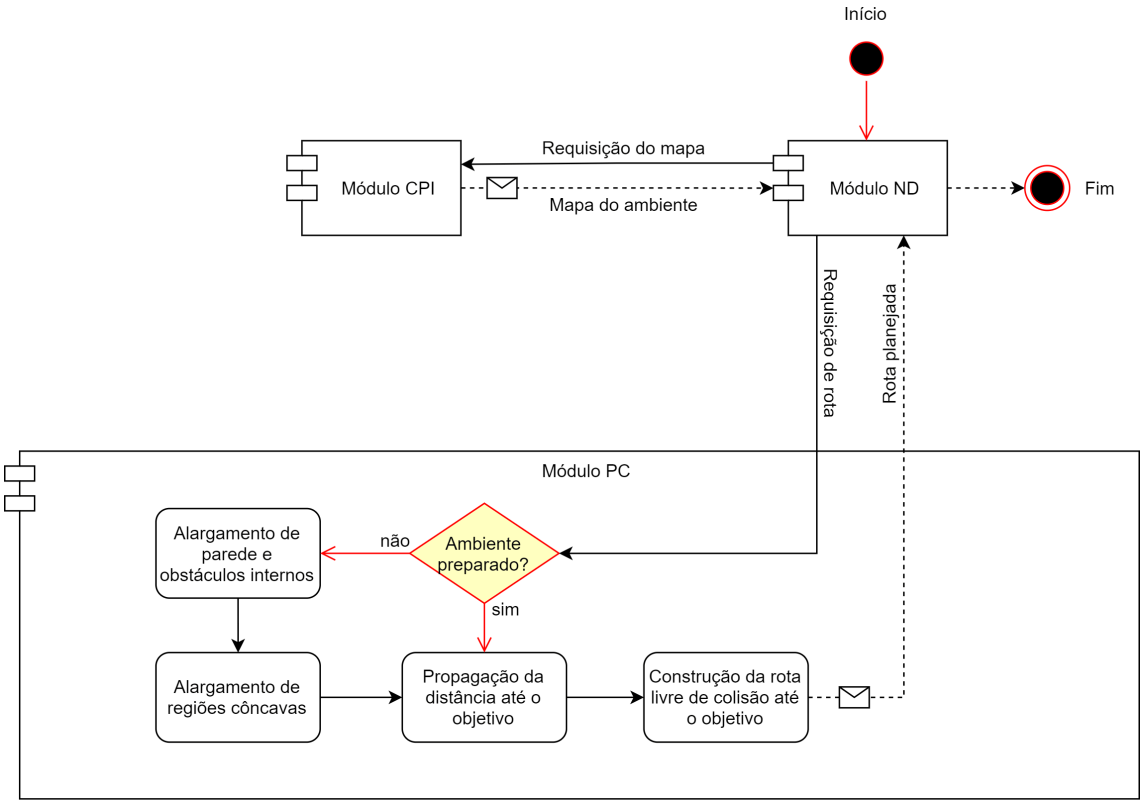


Figura 17 – Fluxograma do módulo de planejamento de caminho.

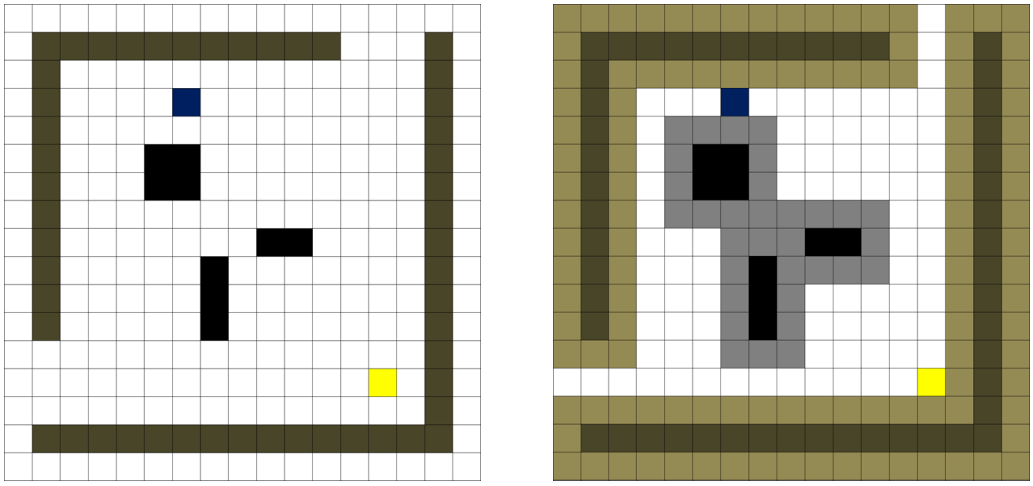
4.1.0.1 Alargamento de paredes e obstáculos internos

A primeira fase provoca o alargamento das paredes e obstáculos internos identificados na imagem pré-processada, visando compensar os efeitos da dinâmica e imprecisões não consideradas no modelo e para evitar colisões durante a navegação do robô. Diferentemente dos trabalhos anteriores, no modelo atual, duas regras distintas são usadas para alargar respectivamente paredes e obstáculos internos, mantendo a diferenciação entre eles. Essa diferenciação entre o alargamento de obstáculos e paredes é necessária, pois na próxima etapa o processo de sombreamento de áreas côncavas será aplicado somente a obstáculos. São elas:

- **Regra do alargamento de obstáculos (RAO):** *SE* a célula central for *livre E* um de seus vizinhos for *obstáculo* ou *obstáculo_virtual*, *ENTÃO* o próximo valor da célula central será *obstáculo_virtual*.

- **Regra do alargamento de paredes (RAP):** *SE* a célula central for *livre E* um de seus vizinhos for *parede* ou *parede_virtual*, *ENTÃO* o próximo valor da célula central será *parede_virtual*.

Ambas as regras são aplicadas por um número fixo de etapas de tempo (x), que dependem da resolução da grade celular e do tamanho do robô. Em nossos experimentos, foi usado $x = 1$, que também foi o valor adotado em (OLIVEIRA; VARGAS; FERREIRA, 2015a), resultando no aumento de uma célula em espessura das paredes e obstáculos internos. Em caso de conflito, a regra RAO tem maior prioridade do que a regra RAP (para ser acionada). A Figura 18(a) apresenta a estrutura do AC inicial ($t = 0$) resultante da discretização de um ambiente, destacando células de estados *livre* (branco), *parede* (marrom), *obstáculo* (preto), *posição_inicial* (azul) e *objetivo* (amarelo). A Figura 18(b) mostra as células após a aplicação das regras do alargamento ($t = 1$), adicionando os estados *parede_virtual* (verde claro) e *obstáculo_virtual* (cinza escuro).



(a) Grade celular inicial ($t = 0$) (b) Grade celular alargada ($t = 1$)

Figura 18 – Evolução das regras do alargamento.

4.1.0.2 Alargamento de regiões côncavas de obstáculos

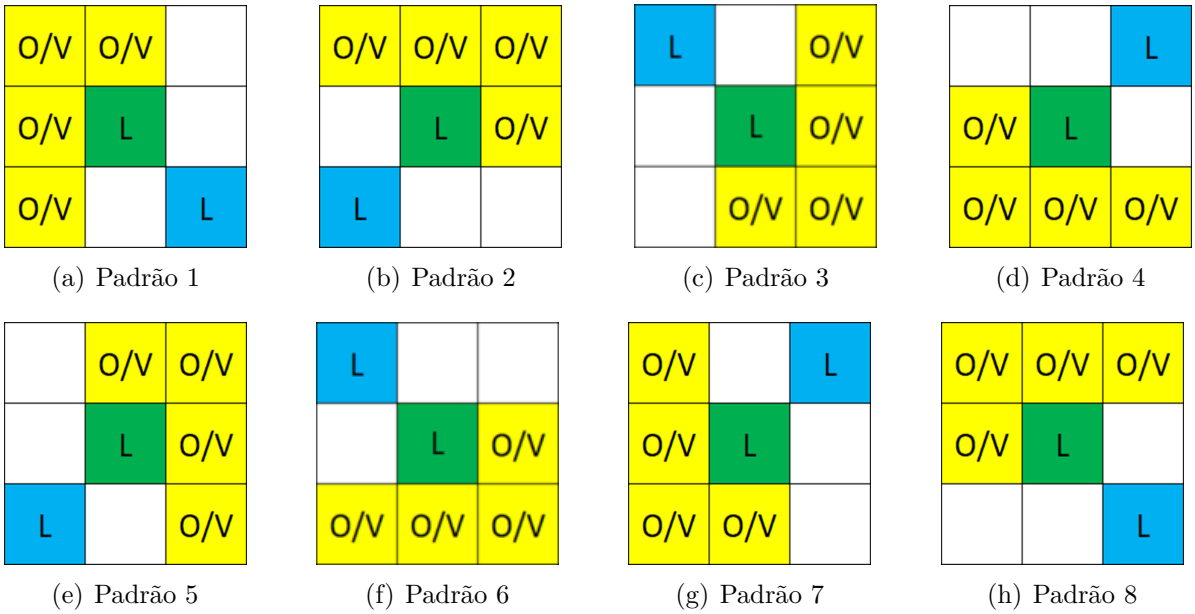
A segunda fase provoca um alargamento adicional quando um obstáculo interno ou um conjunto deles define uma região côncava que não será um caminho viável para o robô atingir seu objetivo ou que caracteriza um risco de colisão se o robô tentar passar por ela. Essas regiões côncavas representam um esforço inútil para os robôs e, portanto, são alargadas por uma regra de AC para que sejam evitadas durante o planejamento do caminho. Uma consequência importante desse sombreamento da região côncava é que o

número resultante de rotações dos robôs na rota planejada tende a diminuir, como apresentado na seção de experimentos. Esta fase é uma contribuição do presente trabalho, e não foi utilizada nos modelos anteriores (BEHRING et al., 2001), (OLIVEIRA; VARGAS; FERREIRA, 2015a) e (MARTINS et al., 2018).

A regra de transição do AC que alarga os obstáculos em regiões côncavas é acionada por oito padrões de vizinhança possíveis para uma célula *livre*. Esses padrões podem ser observados na Figura 19, sendo a célula verde (central) e a azul (vizinha) de estado *livre*, para garantir que o autômato não feche nenhuma passagem e as células em amarelo (vizinhas) de estado *obstáculo* ou *obstáculo_virtual*, indicando a presença de uma concavidade. Os estados das demais células em branco (vizinhas) não são relevantes para o acionamento da regra, ou seja, podem ser de qualquer estado. Uma descrição *totalística* da regra do AC considerando esses 8 padrões é apresentada a seguir:

- **Regra do alargamento de regiões côncavas (ARC):** *SE* a célula central *E* ao menos uma célula do canto da vizinhança forem *livre* (L) *E* a célula do canto oposto da vizinhança junto com pelo menos seus três vizinhos adjacentes forem *obstáculo OU obstáculo_virtual* (O/V), como apresentado na Figura 19, *ENTÃO* o próximo valor da célula central será *região_côncava*.

Assim, a Figura 19 ilustra as 8 situações de vizinhança que atendem às condições da regra ARC, ou seja, os 8 padrões de vizinhança que disparam a regra ARC e alteram o estado da célula central de *livre* para *região_côncava*. Em todas as outras possíveis vizinhanças que não casem com um dos 8 padrões descritos, a célula central permanece no estado atual.

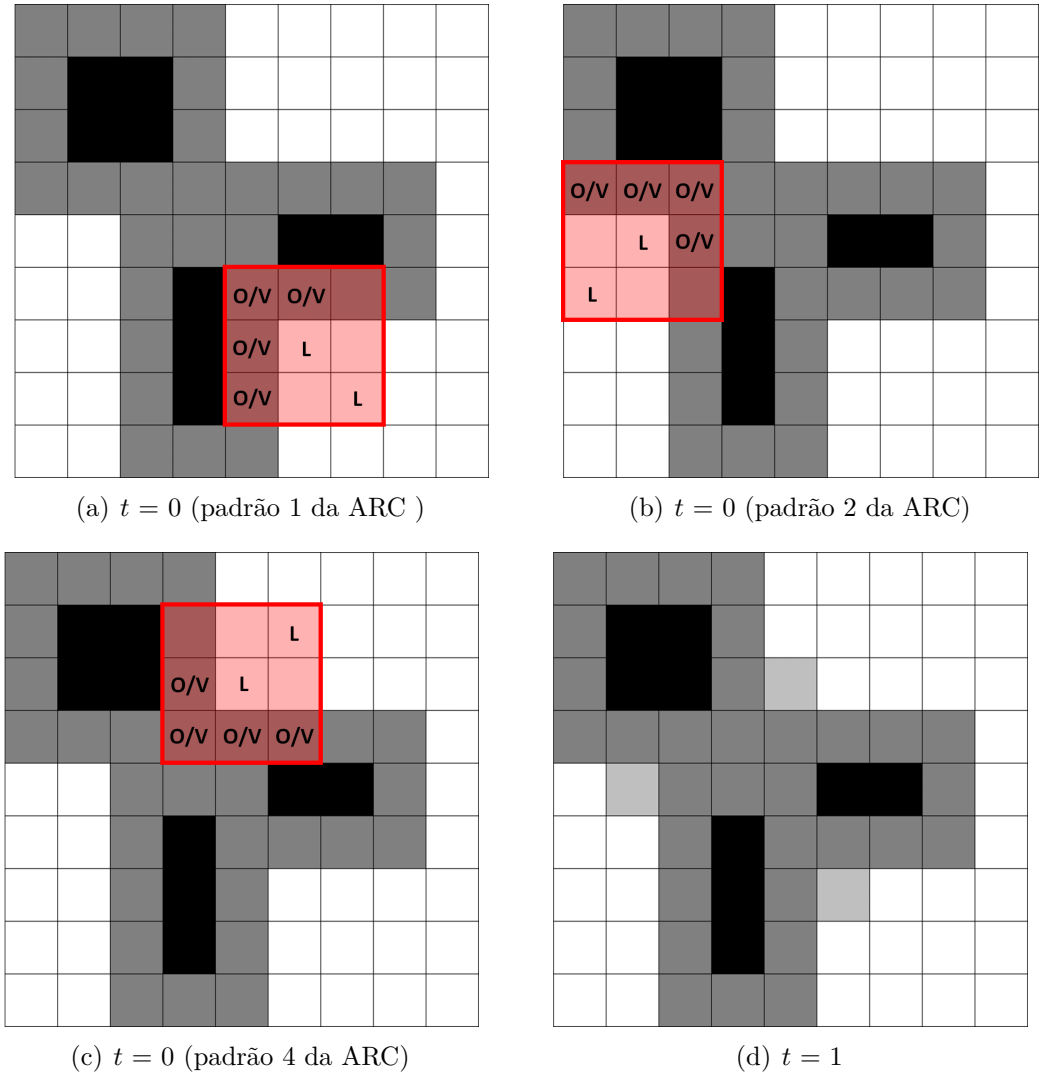


(a) Padrão 1 (b) Padrão 2 (c) Padrão 3 (d) Padrão 4

(e) Padrão 5 (f) Padrão 6 (g) Padrão 7 (h) Padrão 8

Figura 19 – Padrões de disparo da regra ARC.

Para exemplificar o funcionamento da regra do alargamento de regiões côncavas (ARC), a Figura 20 mostra a execução do AC em um passo de tempo completo, onde 3 células do reticulado disparam a regra ARC por casarem com os padrões 1 (Figura 20(a)), 2 (Figura 20(b)) e 4 (Figura 20(c)) da Figura 19.



(a) $t = 0$ (padrão 1 da ARC)

(b) $t = 0$ (padrão 2 da ARC)

(c) $t = 0$ (padrão 4 da ARC)

(d) $t = 1$

Figura 20 – Processo da regra ARC para células acessíveis.

Esse passo de tempo é executado na mesma grade celular da Figura 18(b), onde a fase de alargamento de paredes e obstáculos internos já foi aplicada. A única diferença é que na figura é representada apenas as regiões onde se encontram células de estado *obstáculo*, já que células de estado *parede* não interferem na regra ARC. Sendo assim, pode-se observar que durante a análise realizada no passo de tempo 0 ($t = 0$), o AC consegue encontrar três células de estado *livre* com vizinhança (destacada em vermelho) que correspondem a algum padrão da regra ARC. Cada um desses casos é representado respectivamente nas Figuras 20(a), 20(b) e 20(c). Para facilitar a comparação das áreas destacadas com o respectivo padrão da ARC, suas células foram rotuladas com as mesmas siglas utilizadas pela Figura 19, em sua posição equivalente, sendo L para células de estado *livre*, incluindo

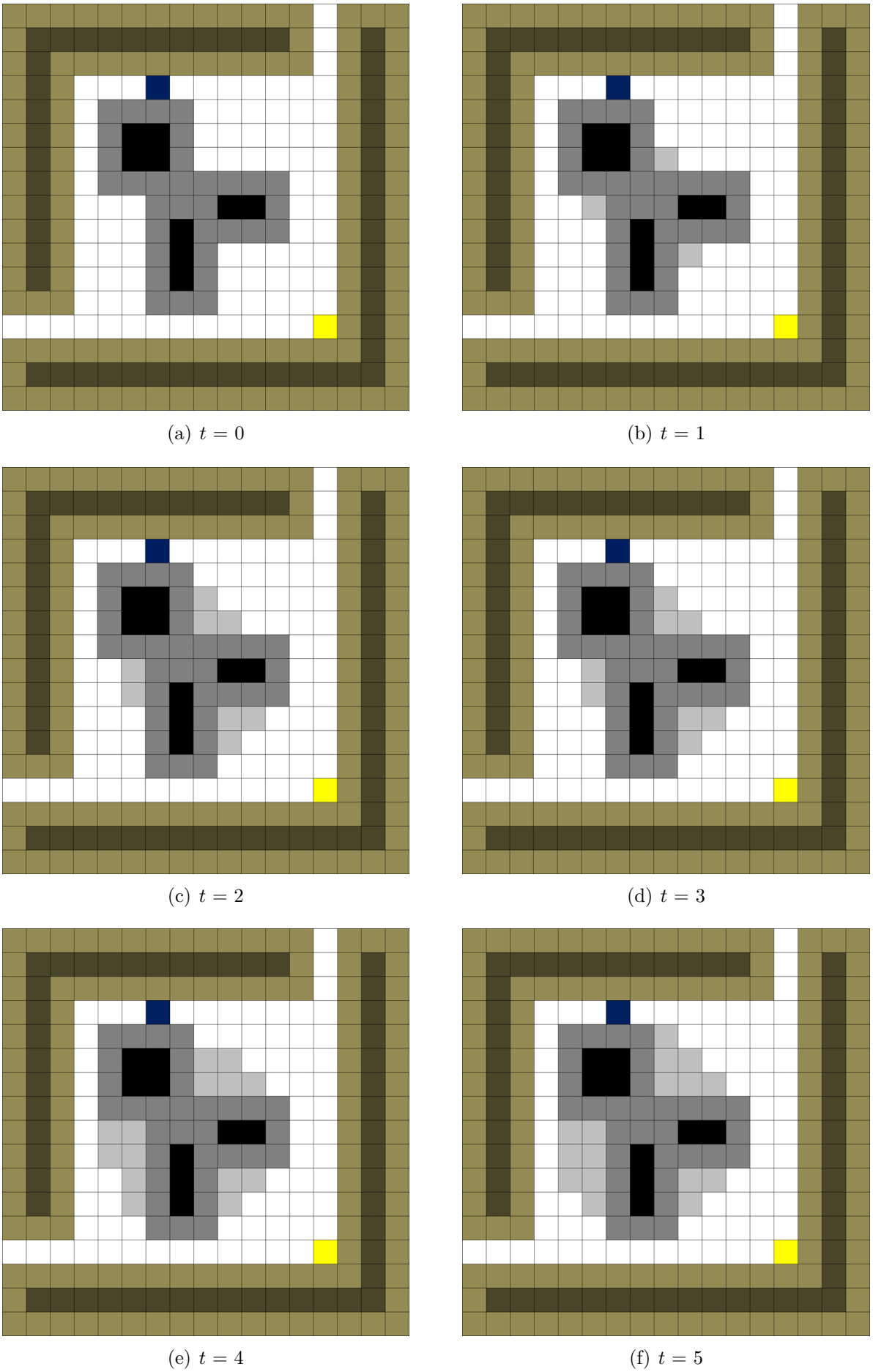
a célula central, e O/V para células de estado *obstáculo* ou *obstáculo_virtual*. Portanto, o padrão encontrado de vizinhança da célula central (*livre*) na Figura 20(a), corresponde ao padrão 1 da ARC (Figura 19(a)). Já o padrão de vizinhança encontrado na Figura 20(b), corresponde com o padrão 2 da ARC (Figura 19(b)). Por último o padrão destacado na Figura 20(c), corresponde com o padrão 4 da ARC (Figura 19(d)). Como não existe outra célula *livre* com vizinhança que corresponda a um dos padrões da ARC (Figura 19), apenas as células centrais das áreas destacadas têm seus estados alterados de *livre* para *região_concava* no próximo passo de tempo. A Figura 20(d) mostra a configuração do reticulado no próximo passo de tempo ($t = 1$) com as células já atualizadas. As células em cinza claro são células de estado *região_concava* provocadas pela regra ARC.

A regra ARC é aplicada sucessivamente, até que nenhuma célula *livre* do reticulado tenha sua vizinhança casada com um dos 8 padrões. Novamente considerando a grade celular completa, a Figura 21 mostra a evolução do AC por 5 passos de tempo, que é a quantidade necessária para que o alargamento de regiões côncavas seja concluído nesse ambiente. Comparando os reticulados inicial (Figura 21(a)) e final (Figura 21(f)), pode-se notar que os três obstáculos originais (células em preto) resultam em uma região compacta, que será evitada no cálculo da rota. As células em azul e amarelo representam respectivamente a posição inicial e a meta do robô.

4.1.0.3 Propagação da distância até o objetivo

Na terceira fase, chamada de difusão de distância até a meta, regras do AC são usadas para determinar a distância de cada célula livre até o objetivo. Durante o processo, as células recebem valores denominados DM (Distância até a meta), que quantificam o custo do deslocamento (distância) até a célula-meta, ou *objetivo*. Essa é a principal fase do modelo de planejamento de caminhos.

Nos modelos apresentados em (OLIVEIRA; VARGAS; FERREIRA, 2015a) e (MARTINS et al., 2018), as distâncias para realizar o movimento de um passo do robô são consideradas equivalentes, sem distinção entre movimentos laterais e diagonais. Em nosso modelo, é feita uma diferenciação entre esses movimentos, considerando que a distância entre os centros de células adjacentes diagonais (nordeste, sudeste, sudoeste e noroeste) é maior que entre os centros de células vizinhas laterais (norte, sul, leste e oeste). Considera-se que a distância é K_L quando um robô faz um movimento lateral (da célula central para um vizinho vertical ou horizontalmente) e K_D quando faz um movimento diagonal (da célula central para um vizinho em uma de suas diagonais), sendo K_L e K_D parâmetros do algoritmo. O processo começa atribuindo zero ao valor DM da célula objetivo. As regras do AC atualizam apenas células de estado *livre* quando pelo menos uma de suas vizinhas tiveram seu valor DM alterado no último passo de tempo. Assim, esse processo ocorre sucessivamente por vários passos de tempo até que nenhuma atualização seja possível. As regras de propagação do DM são definidas por:



(a) $t = 0$ (b) $t = 1$

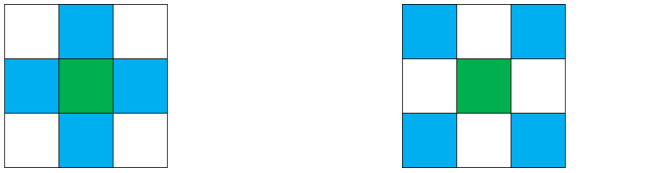
(c) $t = 2$ (d) $t = 3$

(e) $t = 4$ (f) $t = 5$

Figura 21 – Evolução da regra do alargamento de regiões côncavas.

- **Regra de propagação lateral (RPL):** *SE* a célula central (verde) for *livre* *E* seu vizinho com o menor valor DM v estiver em uma posição vertical *OU* horizontal (azul) em relação a ela, conforme mostrado na Figura 22(a), *ENTÃO* o próximo valor da célula central será $v + K_L$.

- **Regra de propagação diagonal (RPD):** *SE* a célula central (verde) for *livre* *E* seu vizinho com o menor valor DM v estiver em uma posição diagonal (azul) em relação a ela, conforme mostrado na Figura 22(b), *ENTÃO* o próximo valor da célula central será $v + K_D$.



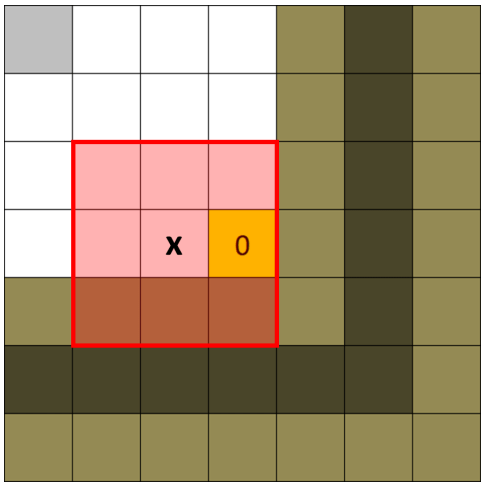
(a) Padrão RPL (lateral) (b) Padrão RPD (diagonal)

Figura 22 – Padrão de disparo para as regras de propagação da distância.

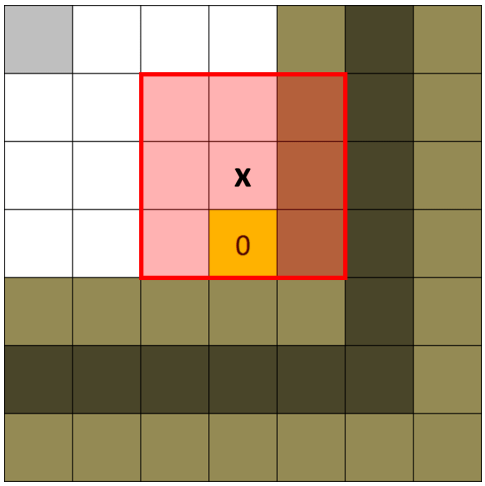
Neste trabalho, adotou-se $K_L = 1$ e $K_D = 1.5$ (para aproximar de $\sqrt{2}$). O valor de DM é calculado para todas as células de estado *livre*, espalhando a distância a partir da célula *objetivo*. As regras RPL e RPD são aplicadas por várias etapas de tempo, enquanto houver pelo menos uma célula *livre* que acione a regra ou até que a célula *posição_robô* seja alcançada. Em caso de conflito, a regra RPL tem maior prioridade do que a RPD. Os vizinhos da posição do robô são sempre alcançados se houver uma rota livre entre a célula *posição_robô* e a célula *objetivo*. Caso contrário, a propagação é interrompida prematuramente, indicando a inexistência de uma rota segura (livre de colisão) para o robô. A diferenciação entre movimentos laterais e diagonais implementada no presente trabalho faz o robô priorizar movimentos laterais em vez de diagonais, os quais são mais custosos, resultando em uma rota final com menor distância e menor número de rotações.

A Figura 23 ilustra o processo de difusão da distância por um passo de tempo, considerando a região onde se encontra a célula objetivo no cenário representado na Figura 21(f). Como já discutido, é atribuído zero para o valor DM da célula objetivo (necessário para iniciar o cálculo). Como o cálculo exige que pelo menos uma vizinha tenha seu valor DM atribuído, no próximo passo de tempo ($t = 1$) as únicas células que serão calculadas são aquelas localizadas ao redor da célula *objetivo*. Observando a vizinhança de cada célula central (x) ao redor da célula *objetivo*, e a relação da orientação entre elas, é possível identificar duas células de estado *livre* que usarão a regra da propagação lateral (Figura 23(a) e Figura 23(b)), e uma célula livre que usará a regra da propagação diagonal (Figura 23(c)). Na Figura 23(a) a célula central (x) de estado *livre* tem apenas uma célula adjacente com valor DM, sendo consequentemente a vizinha de menor valor

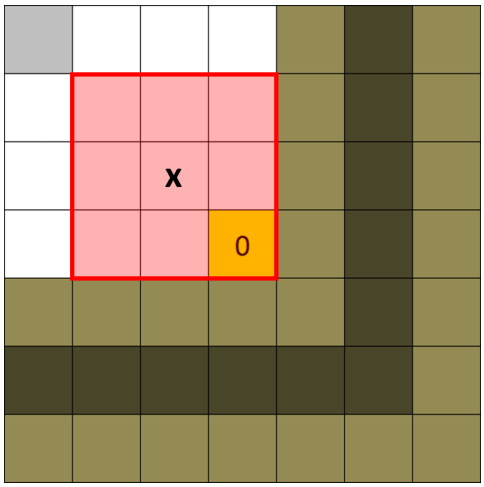
DM (v). Como ela se encontra em uma posição lateral (K_L) em relação a célula central, seu valor DM é igual a 1 (v=0 + K_L=1). Na Figura 23(b), a célula central a ser tratada também terá o valor DM igual a 1, pois sua única vizinha com valor DM (v) é a célula *objetivo* que também se encontra em uma posição lateral (K_L) em relação a célula central. A célula central na Figura 23(c) também utilizará a mesma vizinha com menor valor DM (v) dos exemplos anteriores, mas se encontra em uma posição diagonal (K_D) em relação a célula central, diferentemente das demais. Assim, sua distância é computada por: 0 (v) + 1,5 (K_L), resultando em um valor de DM igual a 1,5. Como não existem mais células livres nesse passo de tempo com pelo menos uma vizinha com valor DM atribuído, apenas as células destacadas têm seus valores atualizados, mantendo seu estado *livre*. A Figura 23(d) mostra o estado das células (valor DM) do reticulado no próximo passo de tempo ($t = 1$), com as células já atualizadas. O valor sobre cada célula *livre* corresponde ao seu valor DM definido pela regra RPD.



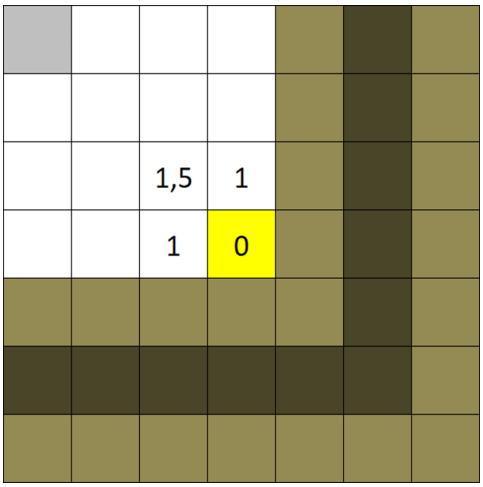
(a) $t = 0$ ($\mathbf{x} = (v = 0) + (K_L = 1{,}0)$)



(b) $t = 0$ ($\mathbf{x} = (v = 0) + (K_L = 1{,}0)$)



(c) $t = 0$ ($\mathbf{x} = (v = 0) + (K_D = 1{,}5)$)

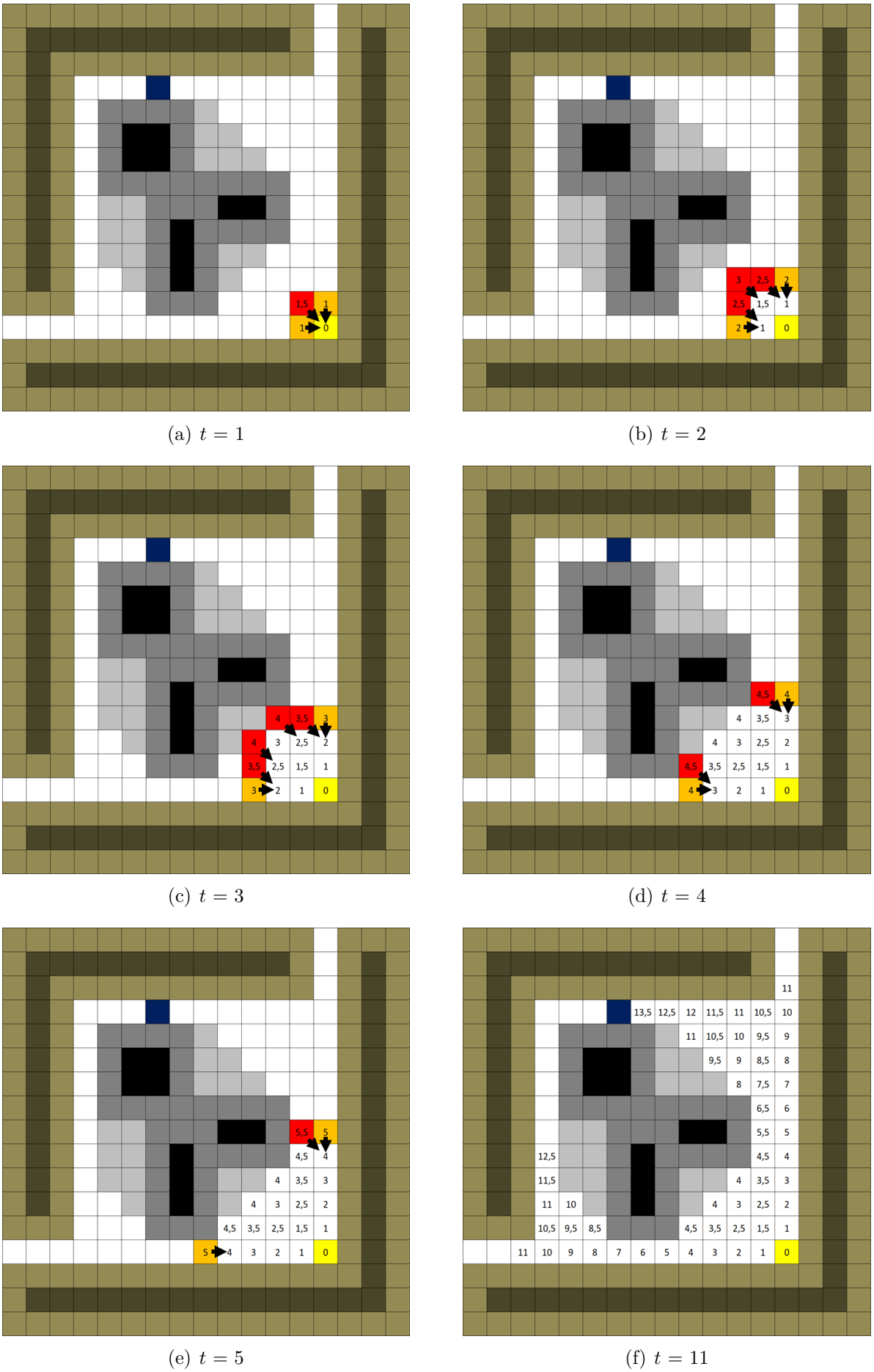


(d) $t = 1$

Figura 23 – Processo das regras de propagação para células acessíveis.

Novamente considerando o reticulado completo, a Figura 24 mostra a evolução do AC por 5 passos consecutivos ($t = 1, ..., 5$) e ao final do processo de propagação da distância à meta ($t = 10$). Observe que o cálculo é encerrado (ou seja, a aplicação das regras RPL e RPD é interrompida) nesse passo de tempo porque a propagação atinge uma célula vizinha da posição atual do robô. As setas relacionam as células que estão sendo atualizadas a seus vizinhos de menor valor DM. Portanto, partindo da célula *objetivo* (em amarelo), com valor DM igual a 0, o cálculo é propagado até que alguma célula vizinha da célula *posição_robô* (em azul).

A Figura 24(a) apresenta os valores propagados para seus vizinhos após o acionamento das regras RPL e RPD ($t = 1$). As células atualizadas com RPD são representadas em vermelho, enquanto as definidas pelo RPL estão em laranja. Por exemplo, como explicado anteriormente, o vizinho com o menor valor DM para a célula vermelha é 0 na posição diagonal. Portanto, RPD foi aplicado a esta célula obtendo 1,5 ($v = 0 + K_D = 1,5$). Para as células em laranja, o vizinho com o menor valor DM também é 0, mas estão nas posições laterais (vertical e horizontal). Assim, o RPL foi aplicado obtendo-se 1 para ambos ($v = 0 + K_L = 1$). No próximo passo de tempo ($t = 2$) (Figura 24(b)), a DM foi propagada usando as células calculadas pelo passo de tempo anterior ($t = 1$). Percebe-se que as células de estado *livre* têm seus valores calculados aplicando as regras de DM: duas células laranjas (RPL) com DM igual a 2 ($v = 1 + K_L = 1$), duas vermelhas (RPD) com DM igual a 3 ($v = 1,5 + K_D = 1,5$) e uma vermelha com MT igual a 2,5 ($v = 1 + K_D = 1,5$). A Figura 24(c) destaca a atualização da DM para seis células adicionais ($t = 3$): duas laranjas (RPL) com valor 3 ($v = 2 + K_L = 1$), duas vermelhas (RPD) com valor 3,5 ($v = 2 + K_D = 1,5$) e duas vermelhas (RPD) com valor 4 ($v = 2,5 + K_D = 1,5$). Da mesma maneira, na Figura 24(d) ($t = 4$) pode-se notar duas células que utilizam RPL e duas que aplicam RPD. Já no instante de tempo $t = 5$ (Figura 24(e)), duas células de estado *livre* têm seus valores DM calculados a partir da RPL, enquanto a RPD foi aplicada em apenas uma. O processo de difusão das distâncias continua até que alguma célula vizinha da célula *posição_robô* (em azul) seja alcançada (no nosso exemplo, a célula com DM = 13,5). O reticulado final, com os valores DM calculados após 10 passos de tempo, é apresentado na Figura 24(f).



(a) $t = 1$ (b) $t = 2$

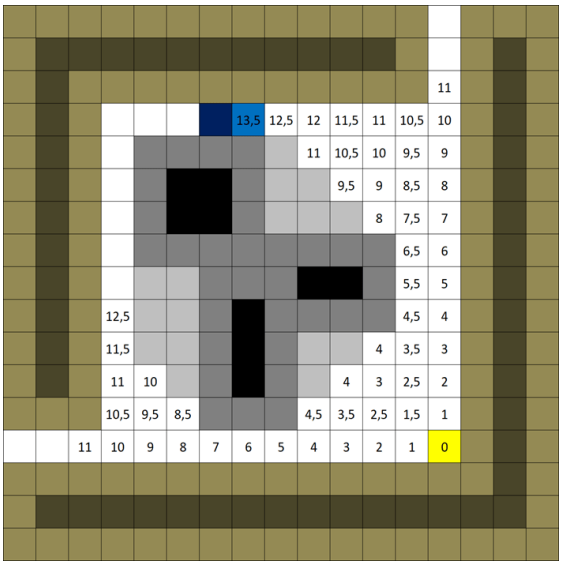
(c) $t = 3$ (d) $t = 4$

(e) $t = 5$ (f) $t = 11$

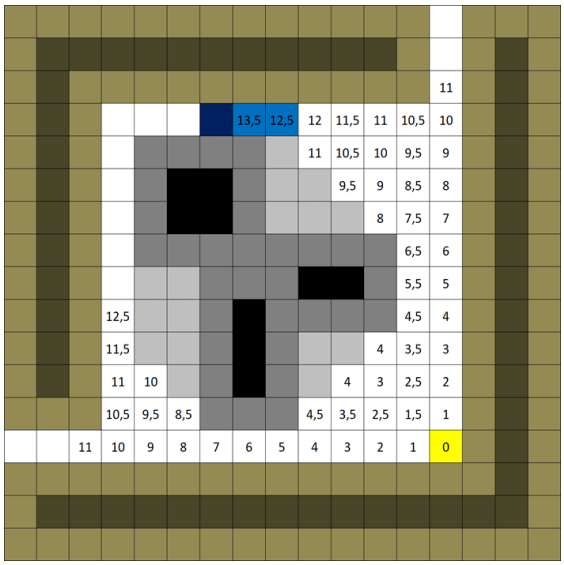
Figura 24 – Evolução da regra de propagação até o objetivo.

4.1.0.4 Construção da rota livre de colisão até o objetivo

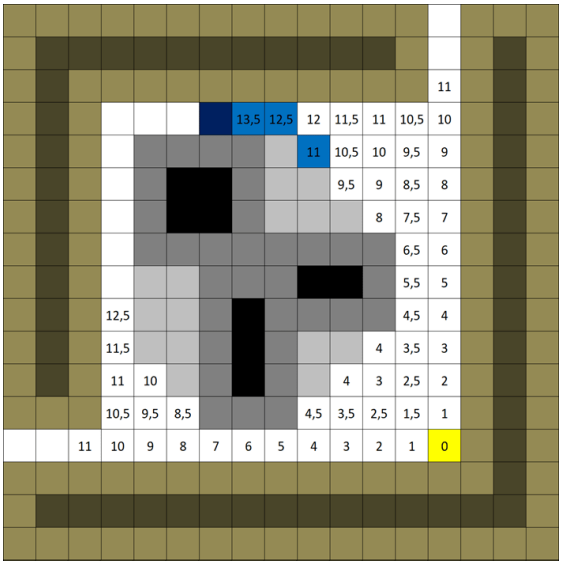
A quarta e última fase é a construção da rota em si. Como observado na Figura 25, esse processo começa da célula *posição_robô* (em azul) em direção a célula *objetivo* (em amarelo), sempre selecionando de forma sequencial a vizinha com menor valor DM (em azul claro), calculada anteriormente. Em caso de conflito, ou seja, duas ou mais células com o menor valor DM, o algoritmo escolhe a célula com a menor distância euclidiana até a célula *objetivo*. As Figuras 25(a), 25(b), 25(c) mostram as três primeiras iterações da construção do caminho, enquanto a etapa final do processo, com a trajetória completa a ser percorrida pelo robô, é apresentada na Figura 25(d).



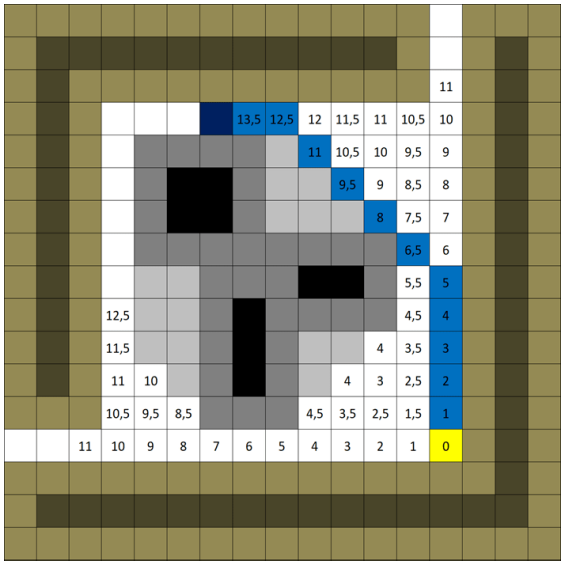
(a) Etapa 1



(b) Etapa 2



(c) Etapa 3



(d) Etapa 12 (final)

Figura 25 – Etapas do processo de construção da rota de um robô.

4.2 Módulo de navegação distribuída (ND)

O algoritmo do módulo de navegação distribuída (ND) é o responsável por controlar a navegação do robô. De todos processos envolvidos neste módulo, três são os principais responsáveis por possibilitar o uso de vários robôs no ambiente e tornar cada um deles independentes. São eles: (*i*) monitoramento da vizinhança; (*ii*) tratamento de obstáculos; e (*iii*) resolução de conflito.

O funcionamento geral deste módulo pode ser observado no diagrama da Figura 26. Com o robô parado, o processo é iniciado com a requisição do mapa do ambiente para o módulo de captura e processo de imagem (CPI), que retorna uma imagem pré-processada do cenário atual. Essa imagem é utilizada para atualizar o reticulado do autômato celular que representa o ambiente, o qual é então utilizado pelo módulo de planejamento de caminho (PC). Com essa imagem disponível, primeiramente, é verificado se existe um objetivo, para prosseguir ou não com a execução. Caso exista um objetivo, é ativado o cálculo da rota, o qual consiste em requisitar ao módulo PC um caminho livre de obstáculos que leve o robô até seu objetivo. Uma vez definida a trajetória a ser percorrida pelo robô, o módulo começa o controle de navegação em si, o qual consiste em um processo iterativo que se inicia pelo monitoramento da vizinhança (processo 1, destacado em azul). Nesse processo, é verificado, com o auxílio dos sensores, se o próximo movimento do robô pode ser realizado ou se seu caminho está sendo obstruído por algum obstáculo ou outro robô do time. Se for detectada alguma obstrução, o robô para e a execução segue para os processos que irão decidir o que fazer nessa situação. Inicialmente, é feito o tratamento dos obstáculos (processo 2, destacado em verde) que estão obstruindo o caminho do robô. Esse tratamento é necessário para que o processo seguinte de resolução de conflito (processo 3, destacado em vermelho) consiga decidir qual manobra o robô deve realizar para superar o obstáculo e continuar a navegação, uma vez que são considerados diferentes tipos de obstáculos. Com a manobra definida, a execução volta para o processo (1) (monitoramento de vizinhança) que reinicia o ciclo de movimento do robô. Por outro lado, quando não existe uma obstrução, o robô realiza um passo do caminho planejado, ou seja, se move na direção e na distância necessárias para alcançar o centro da célula adjacente desejada.

Ao final de cada passo, é verificado se o robô alcançou seu objetivo. Caso tenha alcançado, retorna-se ao início do processo, onde um novo objetivo pode ser definido através da requisição de uma nova imagem para o módulo CPI e a navegação é reiniciada. Caso contrário, o processo iterativo continua até que a célula objetivo seja alcançada. Vale destacar que a definição do objetivo de cada robô do time é ortogonal ao modelo de planejamento e navegação propostos e, portanto, está fora do escopo deste trabalho. No cenário adotado, considera-se que os objetivos, quando existentes, são definidos pelo módulo CPI e repassados ao robô pela imagem pré-processada do ambiente. Diferentes

abordagens podem ser adotadas na definição dos objetivos. Por exemplo, eles podem ser determinados a partir de uma lista preestabelecida, com suas respectivas localizações, ou por meio de interações com o meio (ambientes inteligentes).

Os três processos principais ((1), (2) e (3)), destacados no diagrama, possibilitam a navegação distribuída dos robôs e serão detalhados a seguir.

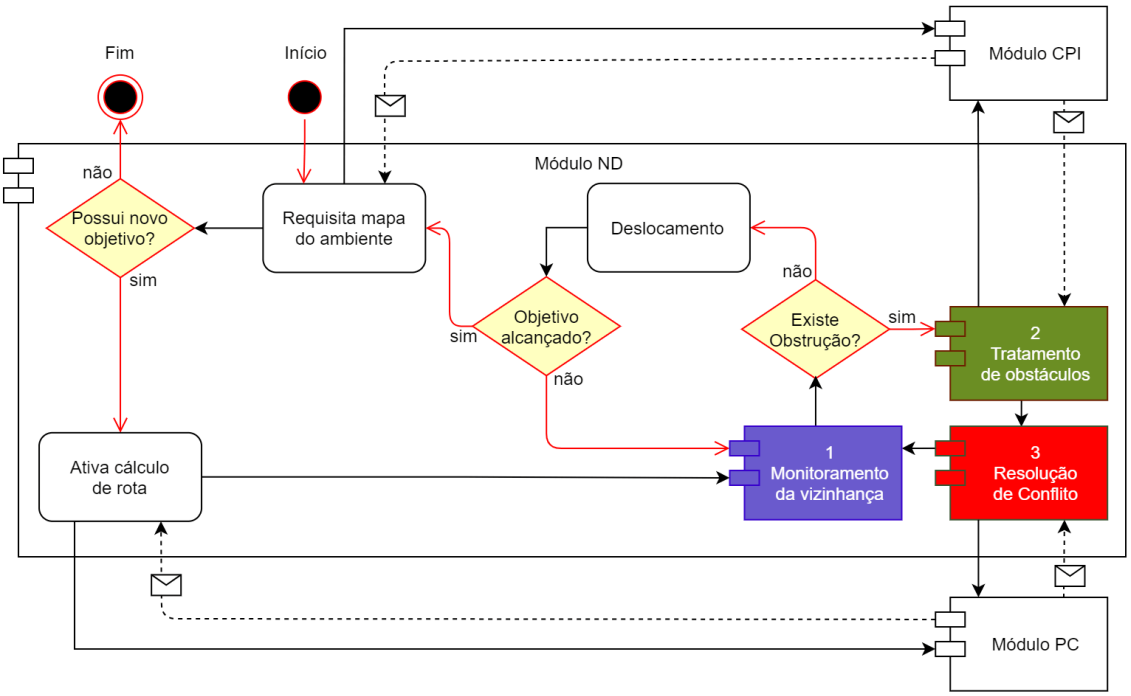


Figura 26 – Fluxograma do módulo de navegação distribuída (ND).

4.2.1 Monitoramento da vizinhança

O algoritmo de monitoramento da vizinhança é dividido em duas fases: (*i*) leitura dos sensores; e (*ii*) classificação da vizinhança.

Sua execução pode ser observada no fluxograma da Figura 27. Como indicado no fluxograma do módulo de navegação distribuída (ND) (Figura 26), esse processo é disparado no início de cada ciclo de movimento do robô, ou seja, ele será executado, a cada passo de movimento, enquanto a célula *objetivo* não for alcançada. Sua função é bem simples e consiste em verificar se existe ou não uma obstrução no caminho do robô e classificar o tipo de obstrução, caso ela exista. Dessa forma, o módulo ND pode decidir se o robô precisa realizar alguma manobra com base na sua situação atual. Nos casos em que existe uma obstrução, o robô para e os processos 2 (tratamento de obstáculos) e 3 (resolução de conflito) no módulo ND determinam a manobra, e o ciclo de movimento é reiniciado, voltando para este processo. Cada uma das fases do monitoramento da vizinhança é explicada a seguir.

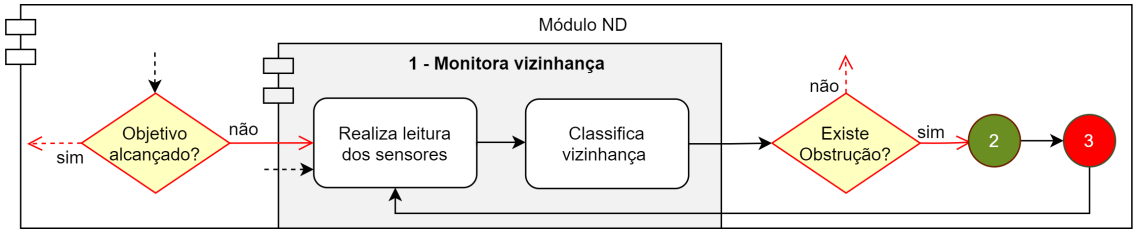


Figura 27 – Fluxograma do processo monitora vizinhança.

4.2.1.1 Leitura dos sensores

A fase de leitura dos sensores é responsável por detectar a presença de obstruções no caminho do robô, sejam obstáculos de fato ou outros robôs navegando pelo ambiente. O processo consiste na leitura do sinal de cada um dos sensores de proximidade infravermelhos localizados ao redor do robô e-puck, como mostrado na Figura 28. Como os robôs em nosso trabalho apenas se movimentam para frente e giram no seu próprio eixo, não há necessidade de realizar a leitura dos sensores localizados na traseira. A Figura 28 mostra o robô E-puck visto por cima, destacando a localização dos seis sensores ativados durante a navegação (cores cinza, verde e azul). A seta azul indica a frente do robô. Dessa forma, pode-se observar que os sensores utilizados são: os dois sensores frontais em cinza ($ps7$ a esquerda e $ps0$ a direita), os dois sensores diagonais em verde ($ps6$ a esquerda e $ps1$ a direita) e os dois sensores laterais em azul ($ps5$ a esquerda e $ps2$ a direita). Apenas os dois sensores posteriores ($ps4$ a direita e $ps3$ a esquerda) não são utilizados.

Os valores retornados pelos sensores ativos são escalados, de forma diretamente proporcional à distância do objeto detectado, entre 0 e 4096 (CYBERBOTICS, 2019). Um valor próximo de 4096 indica que uma grande quantidade de luz é medida, ou seja, um obstáculo está próximo. Já um valor em torno de zero significa que nenhum obstáculo foi identificado pelo sensor. Esses sinais passam por um filtro que determina a presença de um obstáculo, se os valores aferidos atingirem um limiar pré-definido. A descrição desse filtro é apresentada a seguir.

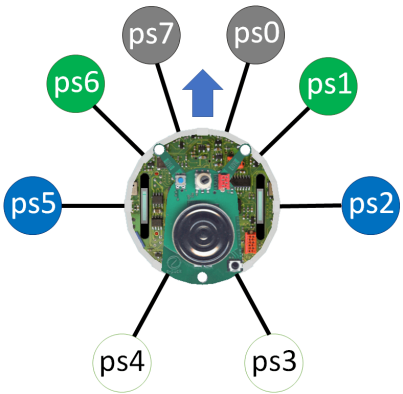


Figura 28 – Sensores utilizados.

4.2.1.2 Classificação da vizinhança

A partir dos sinais coletados dos sensores de proximidade, a fase de classificação da vizinhança determina a existência ou não, de uma obstrução no caminho do robô. Entretanto, durante essa classificação, podem ocorrer falsos positivos, devido à existência de ruído (erro) nos sinais coletados pelos sensores do robô. A fim de reduzir tais ocorrências indesejadas, os sinais coletados são submetidos a um filtro que busca compensar a faixa de erro de cada sensor. O processo de filtragem consiste em atribuir a cada sensor um valor compensador (c), o qual é decrementado do valor detectado por aquele sensor (v). Se, após a compensação, o valor resultante (V_c) ainda for maior ou igual que o limiar definido (l), significa que um obstáculo foi detectado pelo sensor. Caso contrário ($v < l$), significa que nenhum obstáculo foi detectado pelo sensor. Vale ressaltar que o valor compensador pode ser diferente para cada tipo de execução, seja em simulação ou em tempo real. A Figura 29 ilustra um exemplo de classificação, no qual um dos sensores do robô identifica a presença de um obstáculo. As dimensões utilizadas são apenas para facilitar o entendimento e não representam a realidade. Pela figura, pode-se observar que o robô e-puck se movimenta à esquerda de um obstáculo. A faixa cinza é o valor detectado v pelo sensor lateral a direita ($ps2$) e a faixa amarela corresponde ao valor compensador c definido para esse sensor. A linha preta tracejada indica o valor final já compensado (V_c), ou seja, o valor obtido ao subtrair c por v. Considerando que quanto maior um valor na faixa de detecção, mais próximo está o obstáculo, então V_c é maior que o limiar l (linha vermelha tracejada). Portanto, o classificador determina a existência de algum obstáculo à frente do sensor. Esse cálculo é feito para todos os sensores utilizados, e basta um deles detectar algum obstáculo para que seja definida a existência de uma obstrução.

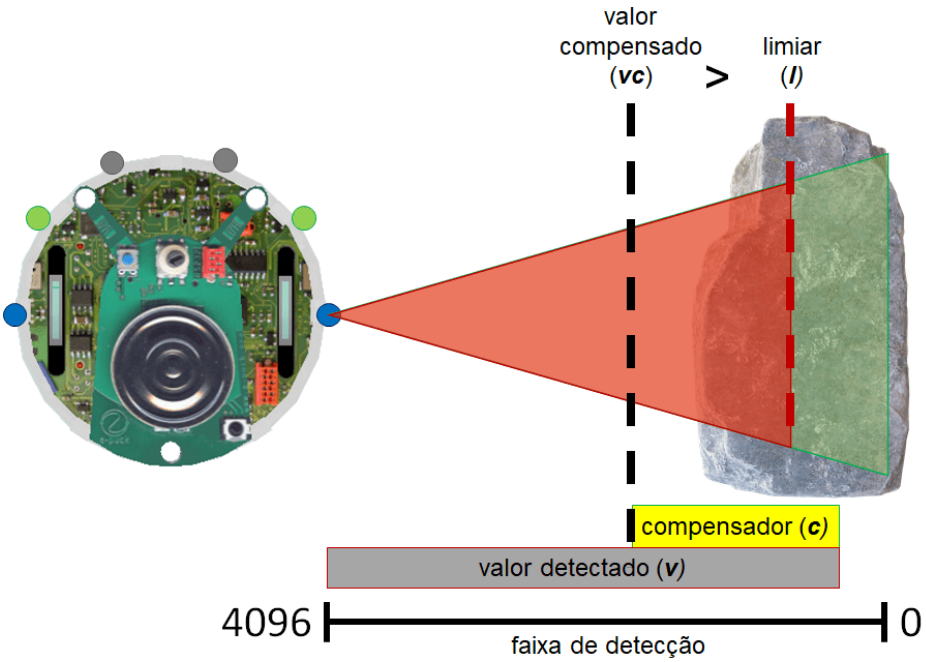


Figura 29 – Filtro de detecção de obstáculos.

Nosso modelo emprega o mesmo algoritmo classificador desenvolvido em (OLIVEIRA; VARGAS; FERREIRA, 2015b), incluindo os valores de limiar ($l = 100$) e de compensação usados para cada sensor ($ps0 = 20$, $ps1 = 120$, $ps2 = 5$, $ps5 = 5$, $ps6 = 120$, $ps7 = 20$). Diferentes configurações para esses valores foram avaliadas a fim de verificar o comportamento do modelo em simulação. Entretanto, os resultados obtidos comprovaram que os valores definidos em (OLIVEIRA; VARGAS; FERREIRA, 2015b), representam um bom compromisso entre detecção de falsos positivos, tempo de navegação e número de colisões. Dessa forma, esses valores foram mantidos no novo modelo.

4.2.2 Tratamento de obstáculos

O algoritmo de tratamento de obstáculos é dividido em três fases: (*i*) requisição do mapa do ambiente; (*ii*) classificação dos robôs na vizinhança; e (*iii*) alargamento dos robôs na vizinhança.

Sua execução pode ser observada no fluxograma da Figura 30. Considerando o diagrama do módulo de navegação distribuída (ND) (Figura 26), pode-se observar que esse processo é executado quando uma obstrução é identificada, ou seja, quando algum obstáculo foi detectado no caminho do robô pelo processo de monitoramento da vizinhança. Portanto, esses obstáculos são tratados nas duas fases classificação e alargamento dos robôs na vizinhança e atualizados no mapa interno com base em uma nova imagem pré-processada requisitada para o módulo CPI. Ao final, o processo 3 (Resolução de conflito) é iniciado, o qual define uma manobra adequada para a situação atual. Cada uma das fases do tratamento de obstáculos é explicada a seguir.

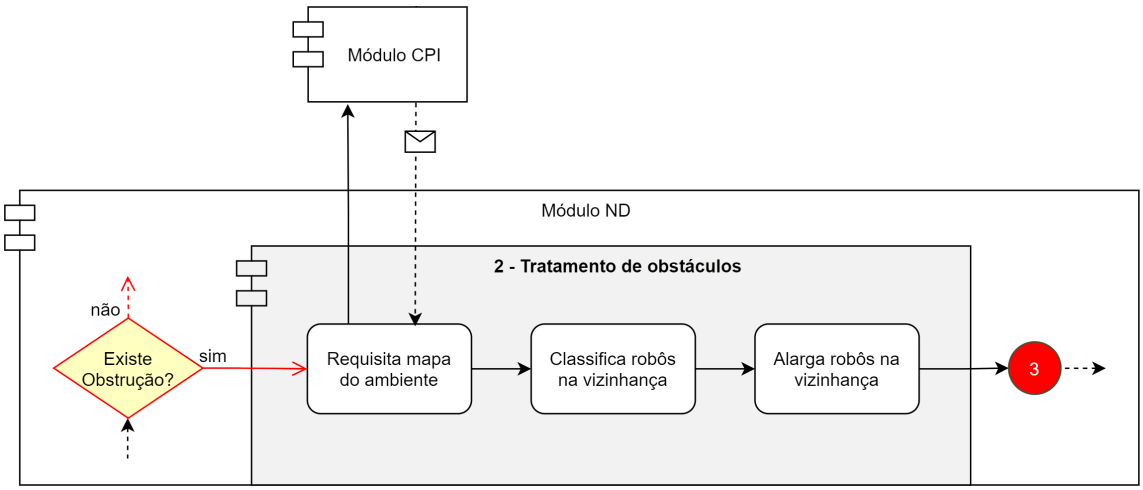


Figura 30 – Fluxograma do processo tratamento de obstáculos.

4.2.2.1 Requisição do mapa do ambiente

A primeira fase apenas realiza a requisição da imagem do ambiente para o módulo de processamento de imagem (CPI) que, por sua vez, retorna uma imagem pré-processada do ambiente com as informações atualizadas sobre o posicionamento dos elementos (robôs e obstáculos) no ambiente. A partir do mapa atualizado, são filtradas as informações referentes à vizinhança do robô, as quais são necessárias para a definição da manobra.

4.2.2.2 Classificação dos robôs na vizinhança

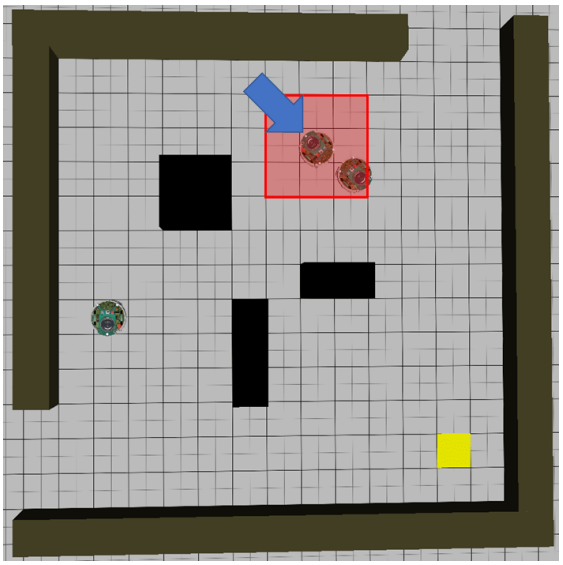
A fase de classificação dos robôs na vizinhança busca identificar os obstáculos que foram detectados pelos sensores do robô, classificando-os de acordo com seu tipo. A necessidade de se identificar o obstáculo vem da proposta de adaptar nosso modelo para trabalhar com vários robôs. Isto é, quando um obstáculo é detectado, além de paredes e obstáculos internos, ele também pode ser um dos robôs que navegam pelo ambiente e, portanto, também devem ser diferenciados. Dessa forma, a vizinhança do robô precisa ser verificada de modo que seja possível fazer a correta classificação do tipo do obstáculo e a identificação de possíveis robôs. Assim o processo 3 (resolução de conflito) do módulo de navegação consegue decidir corretamente qual manobra deve ser executada para evitar o conflito ou eventuais colisões. Ao receber a imagem pré-processada do ambiente, o algoritmo utiliza apenas a parte que contém informações sobre a vizinhança do robô. Assim, é possível analisar quais obstáculos estão a sua volta naquele momento, e atualizar a vizinhança do robô no reticulado do AC, com base nos obstáculos encontrados.

Em nosso trabalho é adotado um método de classificação de robôs bem simples, baseada em níveis de prioridade, na qual cada robô possui um identificador numérico único (inteiro). Assim, quanto maior for esse valor, maior será a prioridade do robô. Portanto, a imagem pré-processada enviada pelo módulo CPI, além de distinguir paredes e obstáculos internos, também consegue distinguir robôs e seus identificadores. A partir dessas informações, as células na vizinhança podem assumir os seguintes estados: *robô_superior*, *robô_inferior* e *robô_virtual*. Dessa maneira, se o robô que está analisando a imagem, encontrar outro robô de prioridade inferior (identificador de menor valor), a célula vizinha que corresponde a posição deste robô recebe o estado *robô_inferior*. Caso contrário, ou seja, quando o robô encontrado tiver maior prioridade (identificador de valor superior), a célula recebe o estado *robô_superior*. Entretanto, se o obstáculo encontrado for parede ou algum obstáculo interno, o estado da célula não é alterado, permanecendo como *parede* ou *obstáculo*. Isso acontece porque não existe a necessidade de atualização para esses tipos de obstáculos, uma vez que eles são estáticos. Em último caso, quando não for identificado obstáculo na imagem, significa que a detecção no processo de monitoramento da vizinhança obteve um falso positivo e, portanto, não há alteração na configuração da vizinhança (estado das células adjacentes). Apesar da maioria dos casos falso positivos

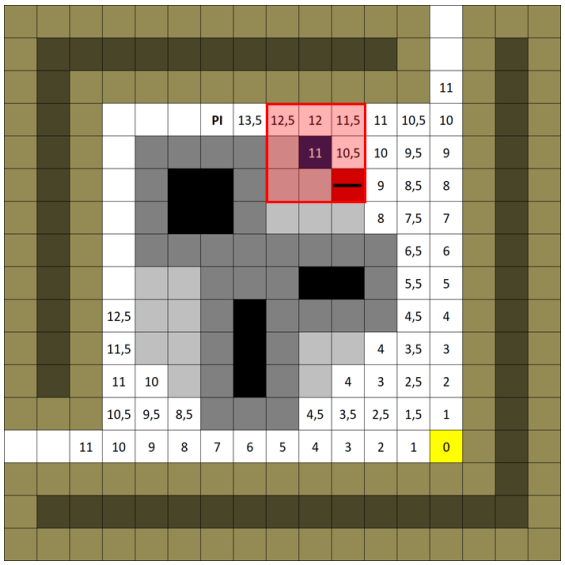
já serem tratados pelo filtro utilizado no monitoramento de vizinhança, eles ainda podem acontecer, quando o erro do sensor é maior que seu compensador.

Na Figura 31 são apresentados 3 cenários diferentes para o mesmo ambiente ilustrado na Figura 24. Nesses exemplos, como o robô não está mais em sua posição inicial, a célula PI mostra qual era sua posição inicial antes de iniciar seu movimento. As Figuras 31(a), 31(c) e 31(e) mostram as representações dos cenários na plataforma de simulação Webots. Cada cenário possui três robôs E-puck e uma seta azul que indica a direção do movimento e o robô sendo analisado, após a detecção de um obstáculo em seu caminho. Nesses cenários é destacado apenas a posição do objetivo (em amarelo) do robô em análise para facilitar a compreensão do exemplo, uma vez que seu mapa interno contém somente a informação do seu objetivo. As Figuras 31(b), 31(d) e 31(f) mostram a representação do espaço celular para os respectivos robôs em análise, destacados em azul, após a atualização de sua vizinhança no mapa interno. As células em vermelho intenso representam a posição de outro robô identificado na vizinhança do robô em análise. Como nesses exemplos o robô identificado possui uma prioridade inferior, a célula em vermelho também possui um sinal negativo indicando a existência de *robô_inferior* na respectiva célula. A relação de vizinhança entre o cenário e o reticulado do robô em análise é destacada por um quadrado vermelho claro, para facilitar a comparação. No cenário 1, ilustrado na Figura 31(a), o robô em análise se aproximou suficientemente de um obstáculo para detectá-lo com os sensores frontais. Ao analisar as informações de sua vizinhança, é identificado que existe apenas um obstáculo em sua volta, o qual é classificado como um robô com menor prioridade (robô inferior). Portanto, a célula vizinha na grade celular, que corresponde com a posição do robô encontrado, tem seu estado alterado para *robô_inferior* (Figura 31(b)). No cenário 2 da Figura 31(c), o robô em análise também detecta um obstáculo. Após a análise da imagem pré-processada, o robô identifica um obstáculo interno (detectado pelo sensor lateral a esquerda). Dessa forma, a célula vizinha correspondente à posição do obstáculo interno encontrado permanece inalterada (Figura 31(d)), uma vez que o reticulado já continha essa informação. No cenário 3 (Figura 31(e)) o robô em análise detecta um obstáculo com seus sensores frontais. Ao analisar as informações de sua vizinhança na imagem pré-processada, é identificado um robô com identificador inferior, então a célula vizinha correspondente tem seu estado alterado para *robô_inferior* (Figura 31(f)).

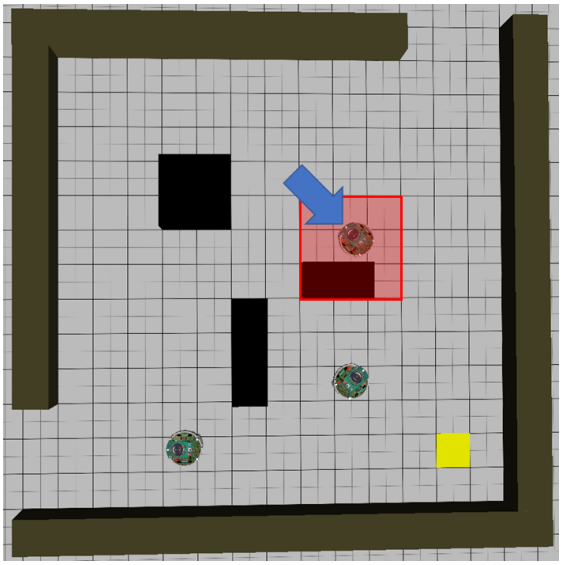
Vale destacar que, mesmo que os sensores posteriores do robô não sejam utilizados, um obstáculo que esteja localizado em sua parte traseira pode ser atualizado em sua vizinhança no reticulado do AC. Isso acontece quando um robô detecta algum obstáculo em seus sensores, mas ao analisar a imagem pré-processada é encontrado um robô que não foi detectado. Nesse caso, ele também é atualizado na vizinhança do robô que o identificou. Essa identificação se faz necessária para representar adequadamente cenários nos quais o robô esteja cercado por outros robôs, possibilitando assim a escolha de uma manobra específica para essa situação.



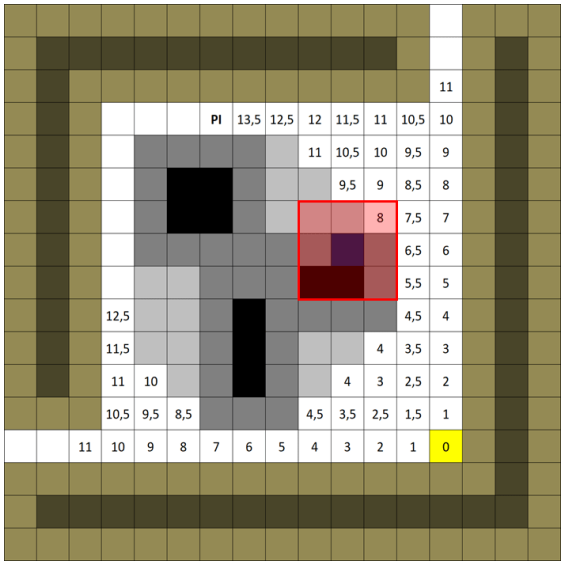
(a) Cenário 1: ambiente simulado



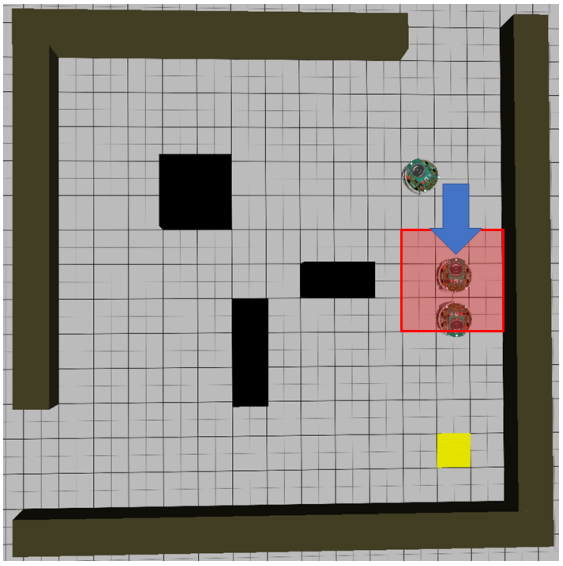
(b) Cenário 1 (reticulado): detecção de um robô



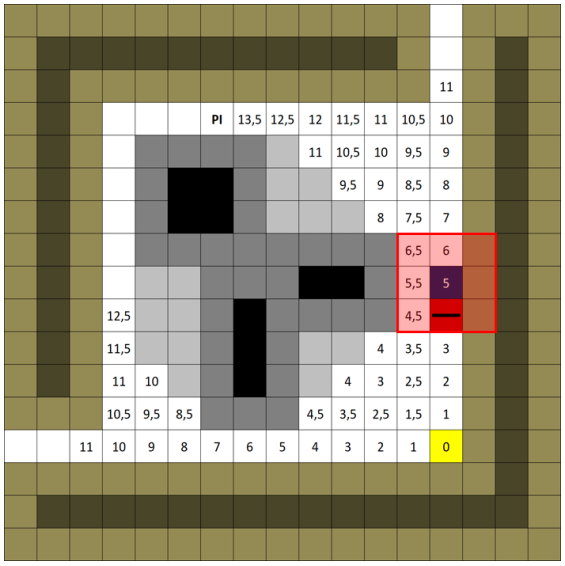
(c) Cenário 2: ambiente simulado



(d) Cenário 2 (reticulado): detecção de um obstáculo



(e) Cenário 3: ambiente simulado



(f) Cenário 3 (reticulado): detecção de um robô

Figura 31 – Identificação e atualização de obstáculos no reticulado celular.

4.2.2.3 Alargamento dos robôs na vizinhança

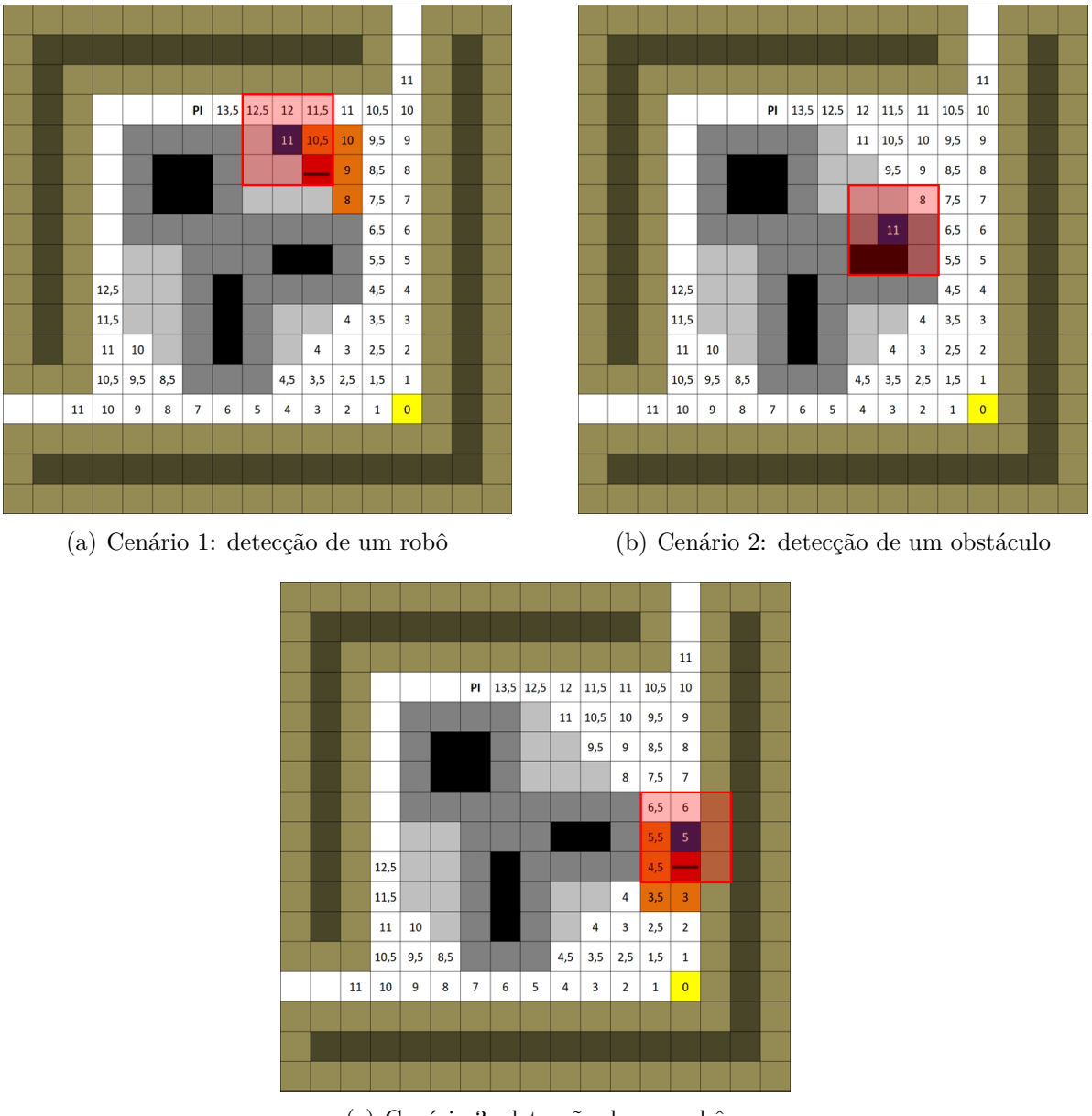
A terceira e última fase consiste no alargamento dos robôs identificados na vizinhança. Diferentemente do alargamento de obstáculos e paredes, nessa fase, a regra é aplicada apenas na vizinhança de cada robô encontrado, e não em todo ambiente. Esse alargamento também é fundamental, dado que após um robô ter parado pela identificação de outros robôs em sua vizinhança, a manobra escolhida pode fazer com que o robô passe próximo aos demais e, se não for considerada uma margem de segurança adequada, podem ocorrer eventuais colisões durante a execução da manobra. Essa margem de segurança é criada pelo alargamento, já que regiões alargadas também são consideradas obstáculos e, portanto, são evitadas pelos robôs. A regra do AC que aplica o alargamento em robôs é bem similar as regras anteriores (RAO e RAP) e é definida por:

- **Regra do alargamento de robôs (RAR):** *SE* a célula central for *livre E* um de seus vizinhos é *robô_superior OU robô_inferior*, *ENTÃO* o próximo valor da célula central será *robô_virtual*.

A regra RAR é aplicada pela mesma quantidade de etapas de tempo (x) utilizada nas regras RAO e RAP ($x = 1$), resultando no aumento de uma célula em espessura dos robôs identificados. Para as paredes e obstáculos internos que também foram identificados, da mesma maneira que não houve a necessidade de atualizar seus estados na vizinhança do robô, também não é feito o seu alargamento, uma vez que esse processo já foi realizado no módulo PC. A Figura 32 mostra os respectivos reticulados dos três cenários da Figura 31 após a aplicação da RAR. As células na cor laranja são células no estado *robô_virtual* que correspondem à região alargada do robô.

4.2.3 Resolução de conflito

A etapa de resolução de conflito é responsável por definir uma manobra para o robô, após ele ter detectado um obstáculo em seu caminho, de modo a evitar uma possível colisão e resolver eventuais conflitos, dando continuidade a sua navegação. Essa manobra consiste em realizar pequenos desvios na rota original do robô para que o mesmo possa superar os obstáculos encontrados, sem comprometer significativamente o seu desempenho e dos outros robôs. Nesse contexto, três manobras podem ser executadas: (*i*) manobra de espera; (*ii*) manobra de recuo; e (*iii*) manobra de desvio. Dependendo do cenário, mais de uma manobra pode ser requerida para resolver um conflito, ou seja, ao final de uma manobra outra pode ser necessária, bem como aquelas que já foram utilizadas podem ser repetidas. Portanto, elas podem ser alternadas até que o conflito seja resolvido. Entretanto, é importante destacar que essas decisões são tomadas por cada robô de forma independente e assíncrona, sem que exista uma comunicação direta entre robôs vizinhos. O robô em análise escolhe a melhor manobra para a situação delineada



(a) Cenário 1: detecção de um robô

(b) Cenário 2: detecção de um obstáculo

(c) Cenário 3: detecção de um robô

Figura 32 – Alargamento de obstáculos detectados.

pela sua vizinhança, mas ele não tem a garantia que o outro robô em conflito irá executar o movimento que se espera, após a escolha dessa manobra. Por isso, é importante que o robô em análise continue verificando a situação da vizinhança após a execução de uma manobra escolhida anteriormente. Dessa forma, cada robô envolvido no conflito realiza suas manobras, decididas a partir de suas próprias percepções acerca de sua vizinhança. Portanto, podemos dizer que essa resolução de conflito é totalmente distribuída, onde cada robô toma sua própria decisão (manobra) para resolver o conflito identificado pelos sensores e imagens. Entretanto, como não existe comunicação direta, nem um agente centralizador que conduza os robôs envolvidos, cada robô deve continuamente monitorar sua vizinhança, até que o conflito seja resolvido. Essa situação será identificada pelo próprio

robô quando sua vizinhança estiver livre de conflitos com outros membros do time.

Outro fator importante é que a escolha da manobra vai depender não apenas do cenário, mas também da prioridade do robô. Normalmente, quando existe um conflito entre dois robôs, aquele de menor prioridade fica parado (manobra de espera), enquanto aquele de maior prioridade executa manobras de desvio. A única exceção a esse comportamento padrão, é quando o robô superior não possui alternativa para resolver o conflito sem afetar consideravelmente a sua trajetória e, consequentemente, o seu desempenho. Nesse caso, o robô inferior precisa agir para desobstruir o caminho (manobra de recuo).

O fluxo de execução dessa etapa é apresentado no diagrama da Figura 33. Ela começa após a conclusão do processo 2 (tratamento de obstáculos), ou seja, os robôs detectados já foram classificados e alargados (se existirem). Caso seja detectado um robô superior na vizinhança daquele que está em análise, a manobra de espera é executada. Essa manobra possui decisores que podem levar a execução de novas manobras (manobra de desvio ou recuo) para resolver o conflito, antes de retomar seu ciclo de movimentação através do processo 1 (monitoramento de vizinhança). Esses decisores serão explicados mais adiante, durante a descrição da manobra de espera. Uma situação diferente acontece quando o robô detectado na vizinhança é um robô inferior. Nessa situação, a manobra de desvio é executada, exigindo que uma nova rota seja requisitada ao módulo PC. Utilizando essa nova rota, o ciclo de movimento é retomado (processo 1), desviando do robô inferior e seguindo seu caminho até o objetivo. Contudo, caso não seja possível executar o desvio, a manobra de recuo é executada, a qual também requisita uma nova rota para o módulo PC. Após a manobra de recuo, o ciclo de movimento é retomado, podendo levar a execução de novas manobras ou o deslocamento em direção ao objetivo. Cada uma das possíveis manobras e o processo da resolução de conflito são explicados a seguir.

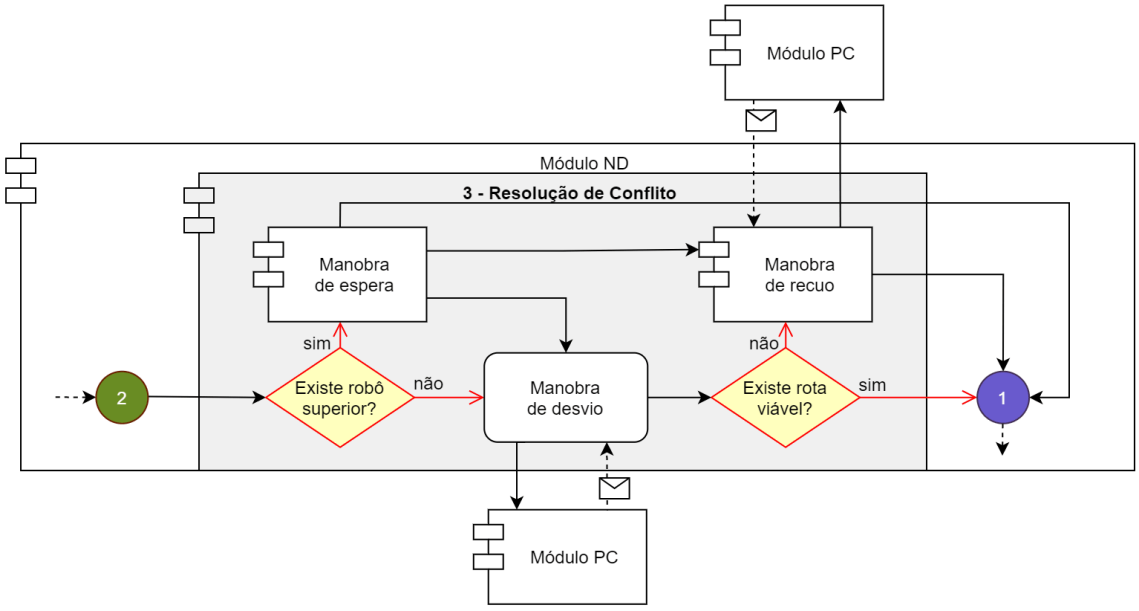


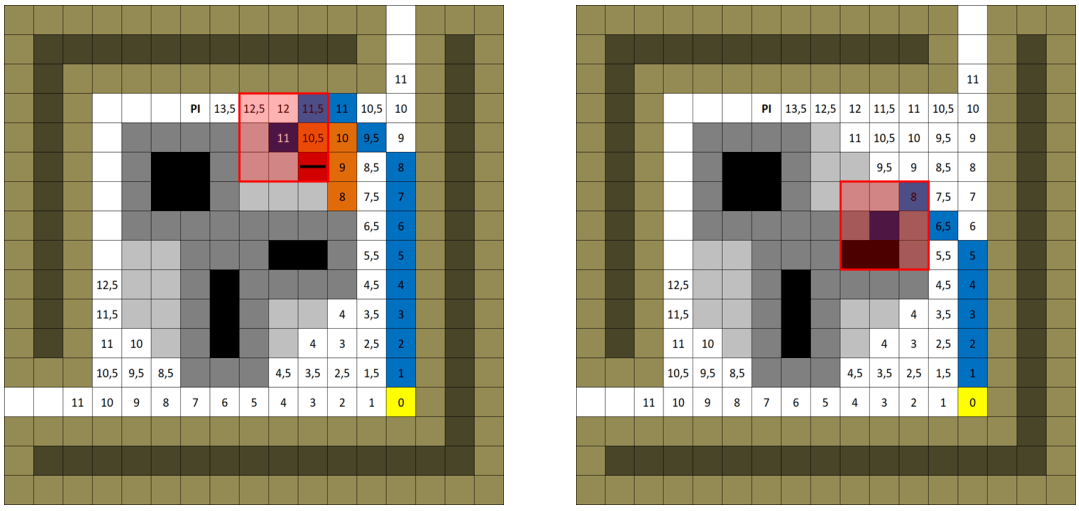
Figura 33 – Fluxograma da resolução de conflito.

4.2.3.1 Manobra de desvio

Conforme pode ser observado no fluxograma de resolução de conflito (Figura 30), essa manobra é escolhida usualmente quando não existe uma célula no estado *robô_superior* na vizinhança, ou seja, em torno do robô analisado existem apenas robôs de menor prioridade ou obstáculos estáticos (células nos estados *parede*, *parede_virtual*, *obstáculo* ou *obstáculo_virtual*). Essa situação ocorre nos três exemplos Figura 32. Entretanto, cabe destacar aqui que, de acordo com o fluxo da Figura 33, essa manobra também pode ser executada após uma manobra de espera. Nesse caso, o robô em análise decide "mudar de estratégia"depois de esperar por um tempo, que o conflito fosse resolvido pela passagem do robô superior. Se, após essa espera, o conflito ainda permanece, o robô inferior opta por fazer o desvio. Entretanto, esse segundo tipo de situação é uma exceção e a maioria das vezes que a manobra de desvio é aplicada trata-se de uma situação típica em que o robô em análise não identifica qualquer robô superior em sua vizinhança.

A manobra de desvio consiste em criar uma nova rota que irá contornar obstáculos que obstruem o caminho do robô possibilitando a retomada do percurso até seu objetivo. Uma rota é requisitada para o módulo de planejamento de caminho (PC), o qual calcula uma nova rota a partir do mapa interno do ambiente. A Figura 34 apresenta dois exemplos de aplicação da manobra de desvio. Na Figura 34(a) uma nova rota é criada a partir do mapa interno do cenário 1 (Figura 32(a)), onde na vizinhança do robô em análise existe um robô inferior que está obstruindo seu caminho, e que deve ser evitado. Como mostrado na Figura 34(a), o robô inferior possui uma região alargada, (células em laranja) que foi gerada pelo processo de tratamento de obstáculos. Assim, quando o módulo PC constrói a nova rota, esse alargamento é evitado e automaticamente a rota é planejada, desviando de forma segura do robô inferior. Isto é, a rota é planejada em direção ao objetivo, desviando de forma segura do robô inferior. O mesmo procedimento é aplicado quando o robô precisa desviar de uma parede ou algum obstáculo interno. Nestes casos, como o robô aproximou desses obstáculos, ele está na região que foi alargada previamente pelo módulo PC, como ilustrado no reticulado do cenário 2 (Figura 32(b)). Assim, ao construir a nova rota, o robô também irá evitar o obstáculo ao sair dessa região alargada em direção ao seu objetivo. A Figura 34(b) mostra a nova rota criada pela manobra de desvio para o exemplo da Figura 32(b), a qual contorna o obstáculo interno detectado no caminho do robô em análise.

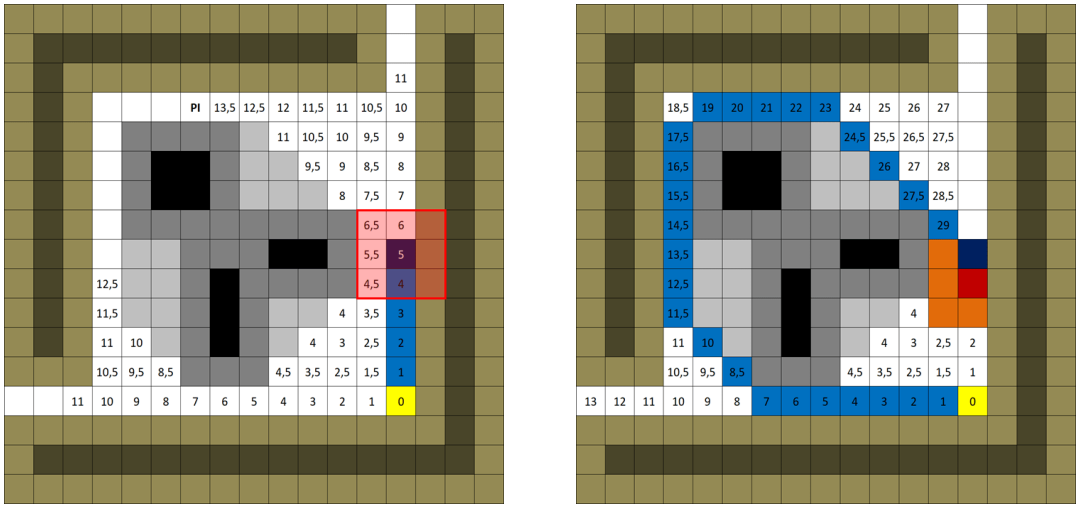
Contudo, em alguns cenários, pode ser que o módulo PC não consiga encontrar uma rota viável. O conceito de viabilidade de uma rota está relacionada com a possibilidade do módulo PC encontrar um caminho até o objetivo, mas sem a realização de grandes desvios que possam comprometer, de forma significativa, a eficiência da trajetória. Em outras palavras, mesmo que possa ser estabelecida uma rota entre o robô e seu objetivo, ela é considerada inviável, se exigir que o robô percorra uma distância bem maior que da sua trajetória original, caso não houvesse a obstrução. Dessa forma, uma rota é viável



(a) Desvio de um robô inferior (cenário 1) (b) Desvio de um obstáculo (cenário 2)

Figura 34 – Rota de desvio construída após identificação de obstáculos.

se ela não exceder consideravelmente a quantidade de passos da posição atual do robô até seu objetivo, mais os passos extras necessários para desviar da obstrução (obstáculo, parede ou robô), caso houvesse espaço para o desvio. Essa situação pode ser observada na Figura 35, a qual representa o reticulado celular do cenário 3 (Figura 32(c)). Enquanto a Figura 35(a) mostra a rota original que seria percorrida pelo robô até seu objetivo, a Figura 35(b) mostra a nova rota planejada pelo módulo PC após a detecção de um robô inferior em seu caminho. Como podemos observar, ao refazer o cálculo de distância no reticulado, o módulo PC atualiza os valores de distância das células livres e consegue planejar uma nova rota entre a posição atual do robô e sua célula objetivo. Entretanto, a nova rota demanda um número de passos substancialmente maior que a original e, portanto, é considerada inviável.

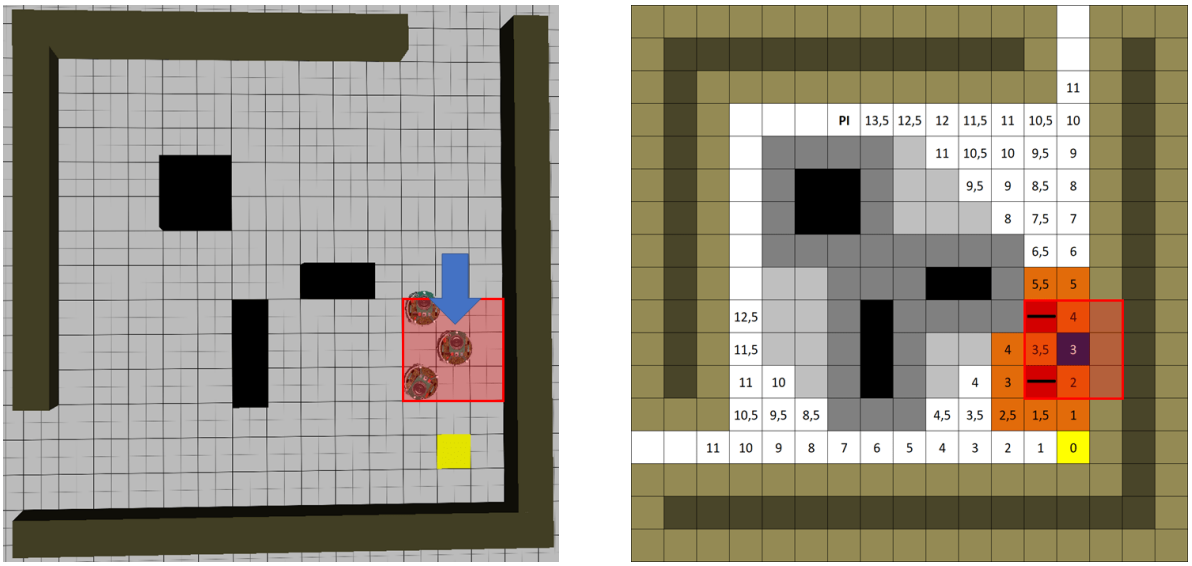


(a) Rota original (b) Rota de desvio inviável

Figura 35 – Exemplo de rota viável e inviável.

Para resolver essa situação, conforme o robô movimenta, ele armazena a quantidade de passos restantes (pr) até seu objetivo. Assim, se a quantidade de passos da nova rota de desvio (rd) for maior que a quantidade de passos restantes mais um valor médio de passos de desvio previamente definido (mpd), a rota de desvio é considerada inviável e, portanto, não será utilizada. Essa condição é implementada no fluxograma de resolução de conflito (Figura 26), de forma que a rota de desvio só é utilizada (rota viável), se $rd \leq pr + mpd$. Por outro lado, se a rota de desvio for considerada inviável ($rd > pr + mpd$), ela não é utilizada e o robô deve executar uma manobra de recuo. Nos experimentos relatados no capítulo 5, foi adotado um valor de mpd igual a 15. Ou seja, a rota de desvio será considerada viável se o acréscimo na nova rota não exceder em 15 passos a distância até o objetivo, em relação à rota previamente calculada.

A verificação de rota viável do fluxograma da resolução de conflito (Figura 33) também pode falhar, quando a rota de desvio não existe, ou seja, quando o módulo PC não consegue planejar um novo caminho desde o objetivo até a posição do robô. Isso acontece porque não existem células livres na vizinhança do robô, ou seja, ele está cercado por outros robôs, obstáculos e suas regiões sombreadas, como ilustrado no cenário 4 apresentado na Figura 36. Nesse caso, conforme o fluxo da Figura 33, o robô irá mudar para a manobra de recuo. A resolução do conflito envolve a tomada de decisões locais e a realização de outras manobras, as quais ocorrem de forma totalmente independente, considerando apenas as informações contidas no mapa interno do robô em análise.



(a) Ambiente simulado do cenário 4

(b) Reticulado do cenário 4

Figura 36 – Exemplo de cenário onde o robô em análise não possui rota de desvio.

4.2.3.2 Manobra de espera

Esta manobra é escolhida quando o robô possui em sua vizinhança ao menos uma célula de estado *robô_superior*, ou seja, ele é um robô inferior e em sua volta existem outros robôs de maior prioridade. Portanto, conforme especificado no fluxograma da Figura 37, a manobra de espera é acionada quando é identificado um robô superior na vizinhança do robô em análise. Esta situação acontece com os robôs identificados nos cenários das Figuras 31(a) e 31(e), já que nos dois cenários, robôs estão de frente uns para os outros, e , assim, os robôs inferiores (representados por uma célula laranja tracejada) também detectam aqueles de maior prioridade (célula preta).

De acordo com o fluxograma desta manobra (Figura 37), o robô inferior fica parado na sua posição por um tempo de espera pré-determinado ($t1$), o qual espera-se ser suficiente para o robô superior possa se movimentar até uma distância segura entre eles. Esse tempo deve ser proporcional a um passo do robô, ou seja, ao tempo necessário para que ele se mova do centro da célula atual até o centro de uma das células adjacentes. Por exemplo, se um passo do robô leva em média x segundos, $t1$ deve ser um múltiplo de x ($t1 = x \times y$), sendo y a quantidade de passos considerada suficiente para o robô superior se distanciar a uma distância segura do inferior. Em nossos experimentos foram utilizados $x = 3$ e $y = 1$, de modo que o robô inferior irá aguardar por 3 segundos ($t1 = 3$). No final desse tempo de espera, é verificado se ainda existe obstrução no caminho do robô de forma similar ao monitoramento da vizinhança. Se nenhuma obstrução for encontrada, significa que o robô superior identificado anteriormente não está mais próximo e, caso nenhum outro obstáculo seja detectado, ele pode continuar sua rota até o objetivo. Caso contrário, ou seja, ainda existe alguma obstrução, o robô deve verificar sua causa. Caso não exista um robô de maior prioridade em seu caminho, ele pode retomar a navegação até seu objetivo, mas deve realizar uma manobra de desvio para contornar a obstrução detectada (parede, obstáculo interno ou algum robô inferior). Entretanto, se foi identificado algum robô superior, pode significar que não houve tempo suficiente para um distanciamento seguro, então ele deve continuar parado e aguardar mais um tempo de espera antes de retomar sua navegação. Dessa forma, todo processo para verificar alguma obstrução em sua vizinhança é repetido. A cada tentativa de sair do modo de espera, o algoritmo incrementa um contador (*cont*). Quando esse contador de tentativas supera um limite máximo previamente definido (*tent*), significa que está ocorrendo um caso especial que exige a ação do robô inferior para a resolução do conflito. Nesse tipo de situação (Figura 36) o robô inferior realiza uma manobra de recuo, uma vez que o robô superior não consegue agir sem que algum dos robôs inferiores abra caminho para ele. Neste caso não necessariamente os dois robôs com menor prioridade utilizarão a manobra de recuo. A definição de qual robô irá se movimentar depende do momento em que eles detectaram o obstáculo, ou seja, quando o contador de cada robô excede o limite de tentativas.

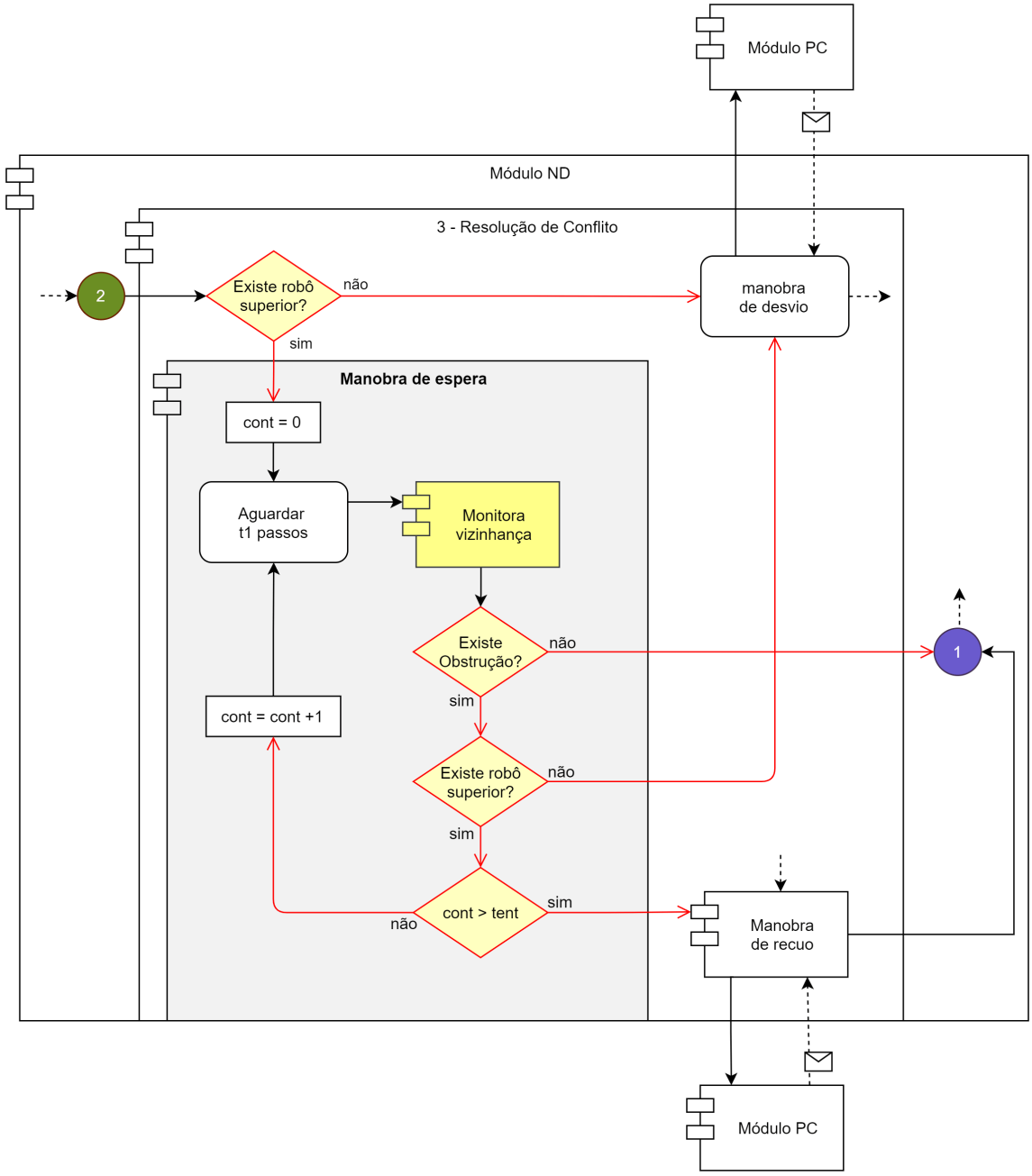
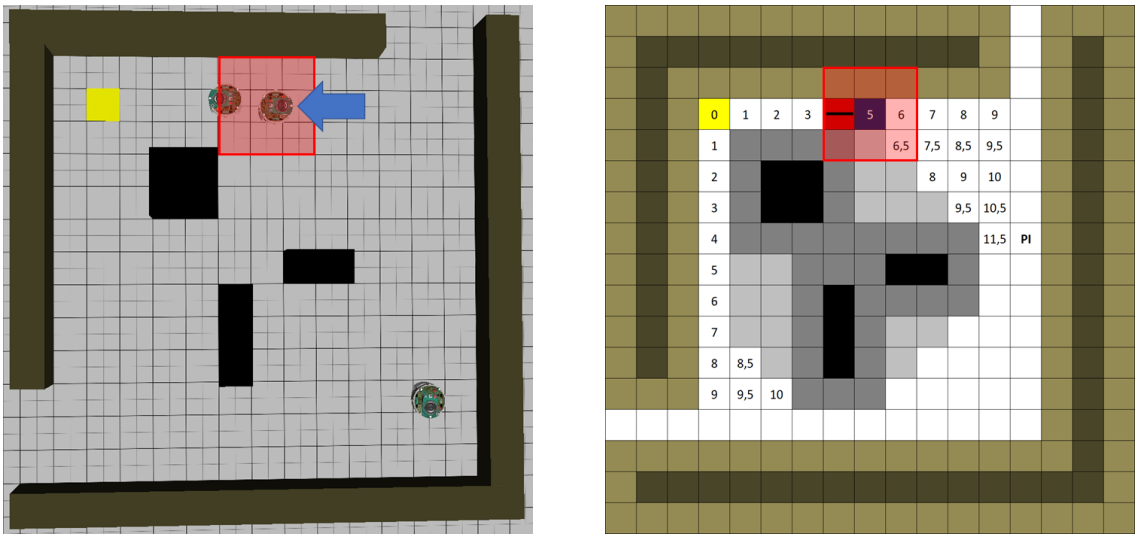


Figura 37 – Fluxograma da manobra de espera.

4.2.3.3 Manobra de recuo

Essa manobra foi criada para evitar situações de *deadlock*, nas quais os robôs inferiores obstruem qualquer possibilidade de rota viável para o robô de maior prioridade, mas ficam aguardando o seu deslocamento para continuar a navegação. Ela é iniciada pelo módulo de resolução de conflito quando não há uma rota viável para o robô superior (Figura 33) ou se a quantidade de manobras de espera do robô inferior exceder o limite pré-estabelecido (Figura 37). A Figura 38 ilustra o ambiente de simulação (Figura 38(a)) e o reticulado resultante (Figura 38(b)) de um cenário onde esse tipo de manobra é exigida (cenário 5). Como pode ser observado na figura, o robô em análise detectou outro robô (inferior) por

seus sensores frontais. Como o robô detectado está bloqueando sua única passagem viável até o objetivo, ele deve executar a manobra de recuo. Os cenários 3 e 4 (Figura 32(c) e Figura 36(b)) também representam situações onde são necessárias manobras de recuo.



(a) Ambiente de simulação do cenário 5 (b) Reticulado do cenário 5

Figura 38 – Exemplo de cenário que exige manobra de recuo.

A Figura 39 apresenta o fluxograma de execução de uma manobra de recuo. O primeiro passo consiste em aplicar as regras de autômato celular com o objetivo de encontrar um objetivo temporário para o robô de modo a desobstruir a passagem do outro robô, e requisitar uma rota para o módulo PC que o leve até este novo objetivo. A mudança temporária no objetivo faz com que o robô realize um movimento de recuo em direção oposta ao robô que bloqueou seu único caminho viável. A posição da nova célula objetivo é definida de forma que quando o robô chegue até ela, a passagem seja liberada e que tenha distância suficiente e segura para qualquer robô na sua vizinhança, além de prevenir novas detecções. O processo de escolha do objetivo temporário é um processo mais complexo e é explicado à frente, após a Figura 39. Se existir uma rota de recuo, o robô irá percorrê-la. O movimento de recuo é realizado da mesma forma que qualquer outro movimento, ou seja, envolve a detecção de novas obstruções e a utilização da manobra ideal para cada caso (processo de resolução de conflito). Por fim, ao chegar no seu objetivo temporário, o robô para e aguarda por um tempo de espera pré-estabelecido ($t2$) , de modo que pelo menos um dos robôs que lhe obstruía consiga seguir o seu caminho. Espera-se que o tempo $t2$ seja suficiente para que os outros robôs saiam do seu raio de detecção, possibilitando a retomada da trajetória em direção ao seu objetivo original. Desta forma, no final do tempo de espera, o objetivo original é restaurado e uma nova rota é requisitada ao módulo PC. Com a nova rota planejada, a execução volta ao monitoramento da vizinhança e o ciclo de movimentação é reiniciado. Nos experimentos relatados no Capítulo 5, foi adotado $t2$ igual a 9.

Entretanto, pode ser que não exista uma manobra de recuo que assegure tais requisitos como no cenário 4 (Figura 36), por exemplo, onde o robô está cercado por outros de menor prioridade. Quando não existe uma rota de recuo, o robô aguarda por um tempo de espera definido ($t3$), que seja suficiente para que algum dos outros robôs em sua volta se movimente até uma distância considerada segura (fora da sua vizinhança). No final deste tempo de espera, o fluxo de execução retorna ao monitoramento da vizinhança (módulo ND), onde o ciclo de movimento é reiniciado. Dessa forma, o robô fica aguardando até que ele consiga movimentar, podendo ou não utilizar alguma manobra de acordo com o estado da sua vizinhança no momento do monitoramento. Nos experimentos relatados no Capítulo 5, foi adotado $t3$ igual a 2.

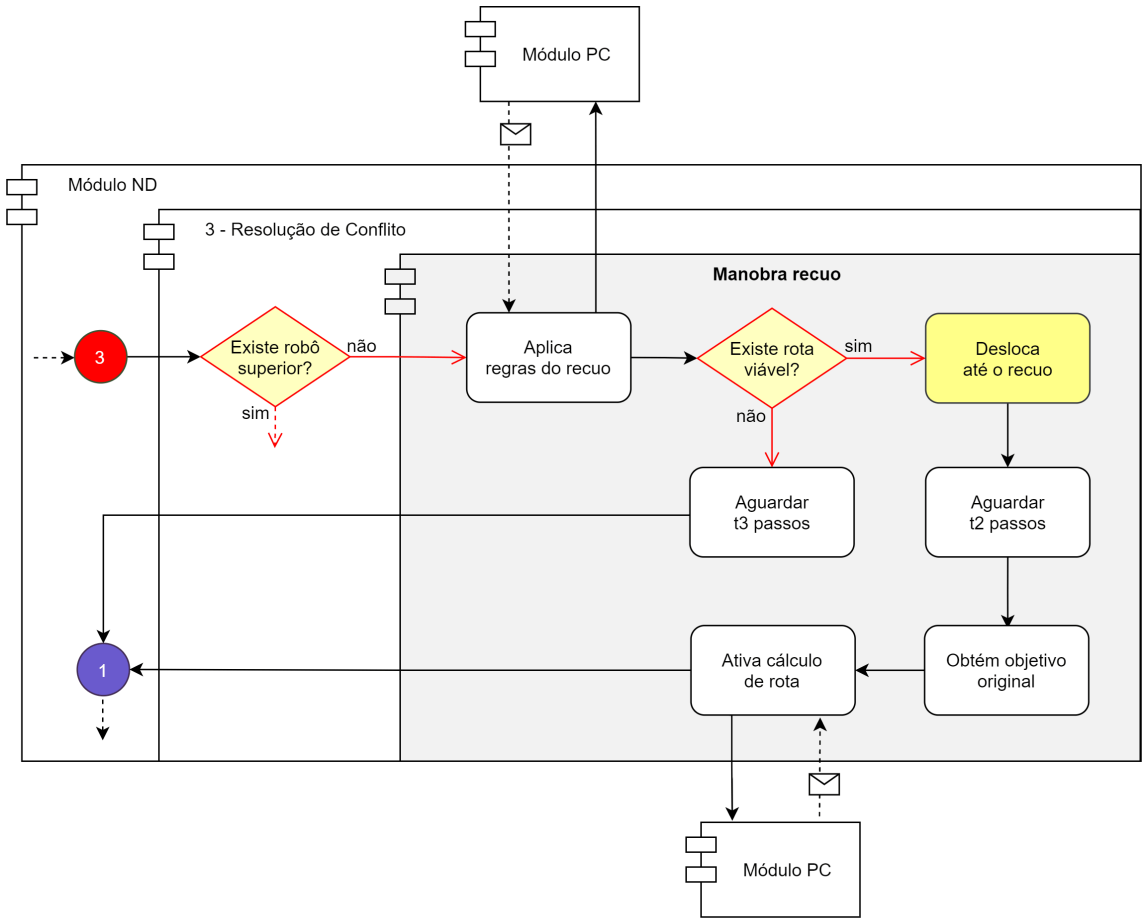


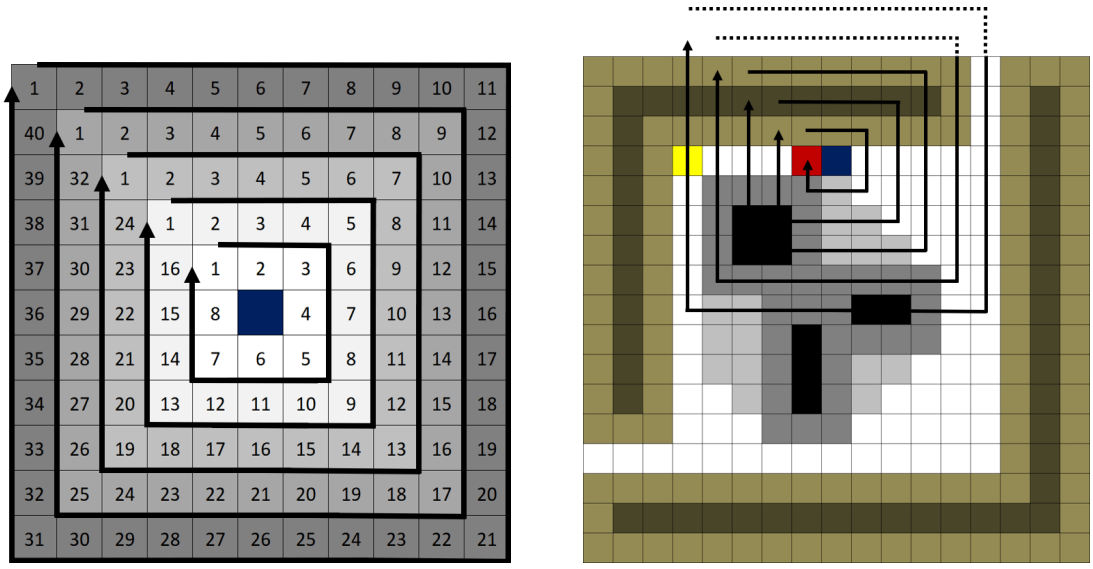
Figura 39 – Fluxograma da manobra de recuo.

Definição do objetivo temporário

O processo de definição de um novo objetivo temporário para o robô começa com uma busca por células promissoras, ou seja, uma posição no ambiente que, ao ser ocupada pelo robô, libere a rota que estava obstruída para os outros robôs. Essas células podem ser encontradas na vizinhança de raio x do robô que está aplicando a manobra de recuo. Inicialmente, a busca avalia as células na vizinhança de raio 1 da posição atual do robô.

Se nenhuma célula for encontrada, aumenta-se o raio de análise (incremento de 1), até que se encontre uma célula promissora para o objetivo temporário ou até que atinja o limite do raio de busca previamente definido. Em nosso trabalho o limite de busca foi definido como 5, mas esse valor pode variar dependendo da aplicação e da configuração do ambiente.

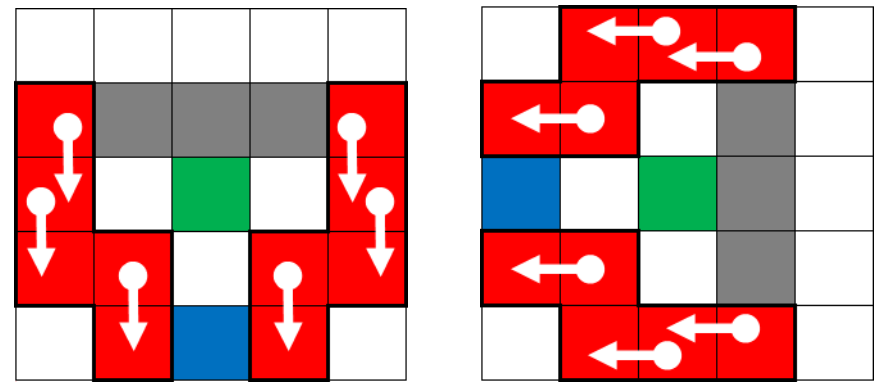
A Figura 40 exemplifica a forma como a busca pelo objetivo temporário é realizada. A ordem da busca é feita em cada raio na sequência estabelecida (1, 2, 3...) e a busca começa pelos raios menores (mais claros) até os raios maiores (mais escuros), se necessário. Na Figura 40(a) pode-se observar que nossa busca começa pela célula no canto superior esquerdo (célula 1) da vizinhança de raio 1. Caso ela não seja adequada, avalia-se a célula 2 e, assim por diante, até encontrar uma célula promissora ou até avaliar a última célula da vizinhança do raio (célula 8). Ou seja, essa busca é realizada no sentido horário, como representado pelas setas em cada raio na figura. Se não houver uma célula adequada na vizinhança de raio 1, a busca parte para as vizinhas de raio 2 e, assim consecutivamente, até encontrar uma posição adequada para ser o novo objetivo ou até analisar todas as células da vizinhança de raio 5 (limite adotado neste trabalho). Quando uma célula promissora é encontrada, o algoritmo tenta criar uma rota até ela. Isso deve ser feito pois o raio de busca pode envolver regiões onde se encontram células promissoras, mas impossíveis de serem alcançadas pelo robô. Se essa rota existir, o processo de busca pelo objetivo temporário termina e o robô inicia seu movimento de recuo. Caso contrário, a busca continua a partir da próxima célula. A Figura 40(b) mostra a possível área de busca no reticulado do cenário 5 (Figura 38), a partir da célula *posição_robô* (em azul) e respeitando os limites do reticulado.



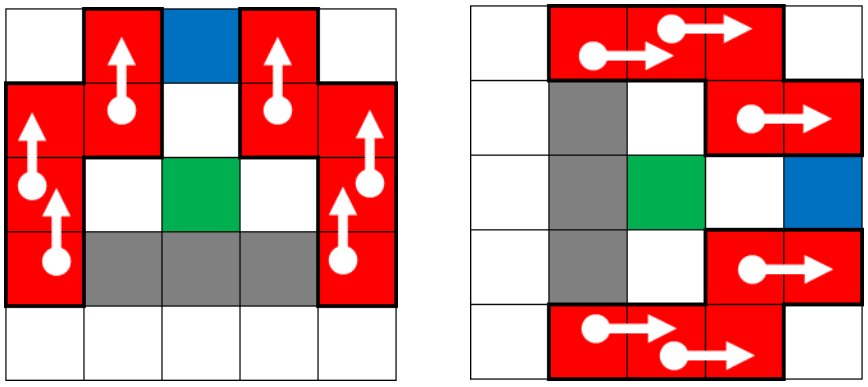
(a) Área de busca de raio 1 ao 5 (b) Área de busca no reticulado do cenário 5

Figura 40 – Área de busca do objetivo temporário.

Quando o robô realiza a manobra de recuo e atinge seu objetivo temporário, o robô que o estava bloqueando não vai mais enxergá-lo, pois eles se distanciaram, e poderá retomar sua navegação. Dessa forma, é necessário que a posição do objetivo temporário considere o alargamento do robô, garantindo uma passagem segura e prevenindo novas detecções entre robôs. Para isso, a validação da célula para o objetivo temporário considera quatro padrões de vizinhança, ou seja, se a vizinhança da célula que está sendo avaliada corresponder com algum desses padrões, ela será uma célula promissora. Esses padrões podem ser observados na Figura 41, onde a célula central da vizinhança que está sendo avaliada, destacada em verde, é aquela que pode se tornar uma célula promissora (CP). O estado dessa célula deve ser *live* ou *concavidade*, para permitir que o robô recue para qualquer uma dessas regiões. As células em cinza devem estar no estado *obstáculo_virtual*, *parede_virtual* ou *concavidade*, enquanto as azuis são células no estado *livre* e as brancas podem assumir qualquer estado. As células vermelhas indicam um conjunto de células que se relacionam, e as setas indicam a relação entre duas células. Essas relações representam opções de configurações (estados das células) que devem ser atendidas para que o conjunto das células vermelhas corresponda ao padrão analisado, indicando assim a existência de uma rota livre. Entretanto, para a célula verde ser considerada promissora (CP), é necessário que todos os subconjuntos estejam corretos.



(a) Padrão 1 (b) Padrão 2



(c) Padrão 3 (d) Padrão 4

Figura 41 – Padrões de células promissoras para o objetivo temporário.

Um subconjunto de duas células vermelhas está correto quando pelo menos uma delas atenda ao estado exigido pelo relacionamento do subconjunto, como especificado na Figura 42. Como pode-se observar no exemplo apresentado na figura, a célula superior representa o início da seta (indicado por um círculo), definida por a, deve ter estado diferente de *livre* e a célula inferior que representa o final da seta (indicado por um triângulo), definida por b, deve estar livre. Esse comportamento é o mesmo para todos subconjuntos. Assim, se pelo menos uma das células (a ou b) possuir o estado solicitado, então o subconjunto está correto. Por outro lado, o subconjunto é considerado incorreto quando as duas células não atendem à configuração especificada pelo subconjunto. Ao realizar a análise de um padrão, ele só é aceito, se possuir a configuração especificada, ou seja, todos os subconjuntos de células vermelhas estiverem corretos e as demais células (verde, azul e cinzas) corresponderem à vizinhança da célula em avaliação (verde). Nesse caso, ela passa a ser uma célula promissora para o objetivo temporário.

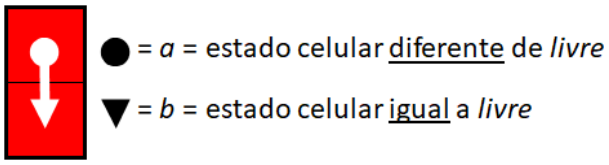
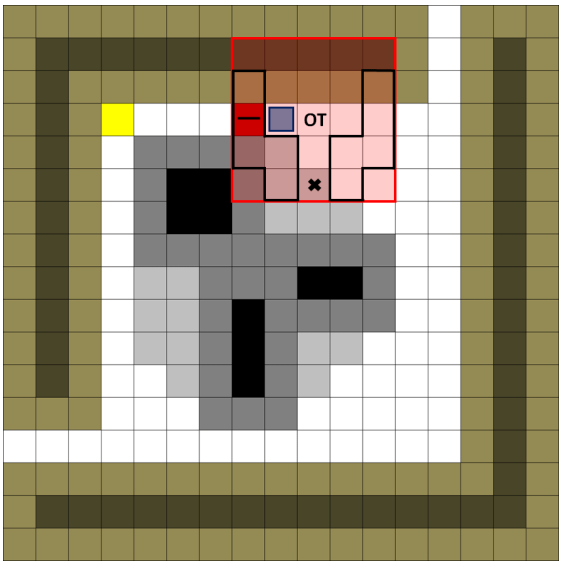
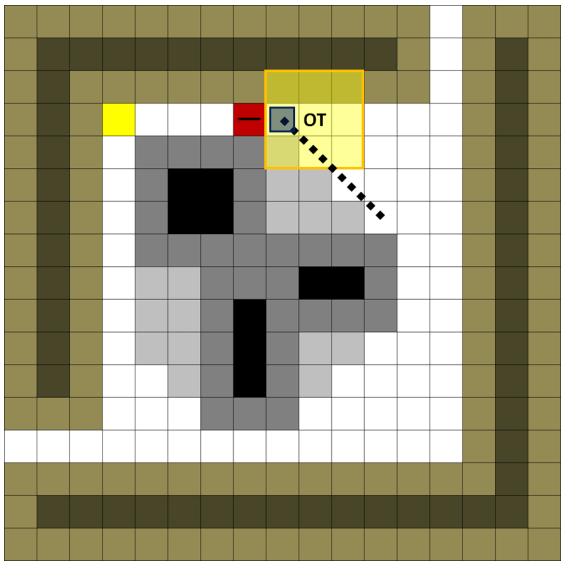


Figura 42 – Padrão do subconjunto de celulas vermelhas.

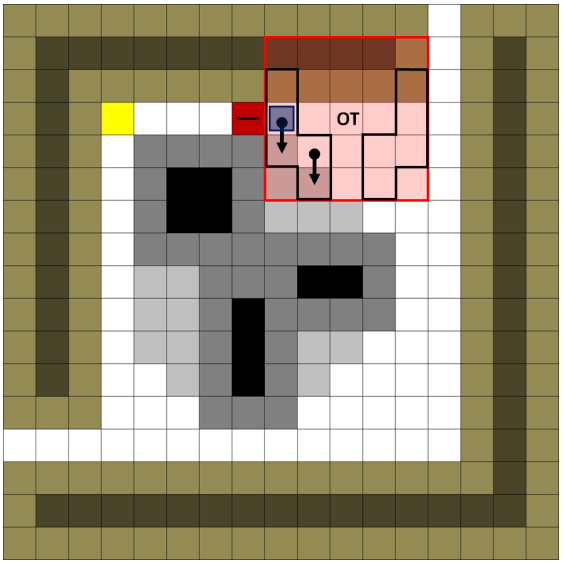
As células vermelhas e azul dos padrões para células promissoras têm o objetivo de garantir uma passagem segura, prevenindo novas detecções, ou seja, elas simulam o alargamento do robô ao posicionar na localização do objetivo temporário. Esse comportamento pode ser observado na Figura 43, onde temos um exemplo de busca e validação para células promissoras para o objetivo temporário. Esta busca é realizada considerando o mesmo reticulado mostrado nas Figuras 38 e 40 (cenário 5). As figuras à esquerda mostram a representação da área considerada na validação do padrão 1 para um possível objetivo temporário (OT), em diferentes raios de busca (raios 1, 2 e 3). Já as figuras à direita destacam em amarelo a área de segurança simulada para as células em avaliação nos respectivos raios. Considerando que a posição do robô que está aplicando a manobra de recuo (*posição_robô*) e as células alargadas dos outros robôs (*robô_virtual*) não são consideradas na busca pela célula promissora, elas não são representadas na Figura 43. A área de busca completa, com esses elementos, pode ser observada na Figura 40(b). Como é possível notar, na vizinhança de raio 1, a primeira célula de estado livre ou concavidade que pode se tornar uma célula promissora (CP) é destacada na Figura 43(a). Por esta imagem também pode-se observar a representação do padrão 1 (Figura 41(a)), sobreposto sobre a possível célula promissora. A célula marcada com o x indica uma incompatibilidade com o padrão, uma vez que a posição equivalente ao x é definida como *livre* no referido padrão, mas no reticulado corresponde ao estado *concavidade*. Portanto, esse padrão não correspondeu com a vizinhança da célula em avaliação e os demais pa-



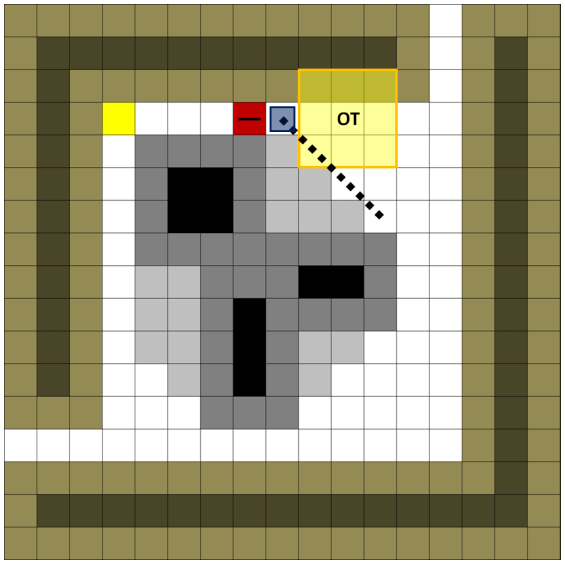
(a) Analise do padrão 1 em uma célula de raio 1



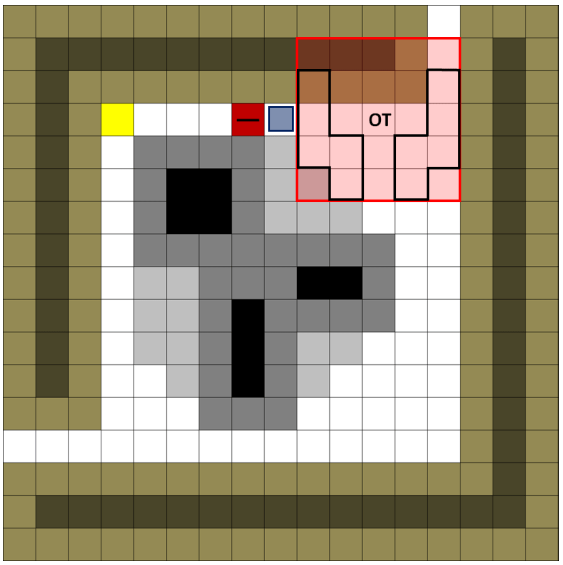
(b) Área de segurança simulada no raio 1



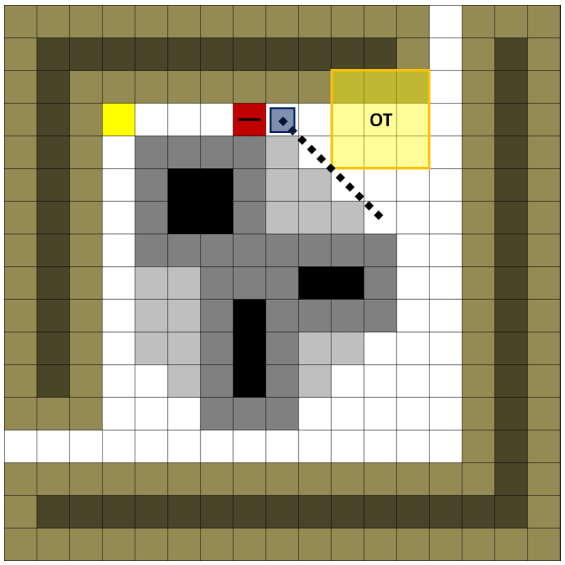
(c) Analise do padrão 1 em uma célula de raio 2



(d) Área de segurança simulada no raio 2



(e) Analise do padrão 1 em uma célula de raio 3

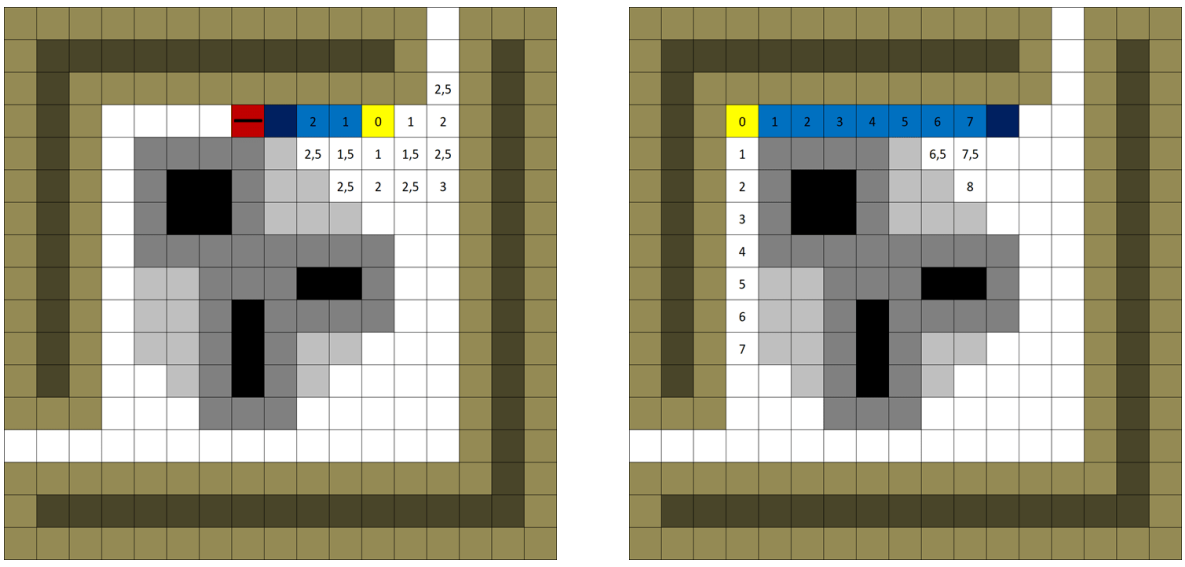


(f) Área de segurança simulada no raio 3

Figura 43 – Processo de busca por uma célula promissora para o objetivo temporário.

drões devem ser testados. Contudo todos os outros padrões também falham e a célula é então descartada. Observando a Figura 43(b), percebe-se que o alargamento simulado para a célula bloquearia a passagem (células de estado *livre* abaixo da linha pontilhada), não deixando uma margem de segurança para evitar novas detecções. Portanto, a célula avaliada realmente não seria uma célula promissora. Então, a busca contínua para as demais células de raio 1 e, como nenhuma delas também atendem a algum dos padrões, a busca no raio 2 é iniciada. A primeira célula avaliada (Figura 43(c)) também não corresponde ao padrão 1. Entretanto, nesse caso, a falha ocorre em dois subconjuntos de células destacadas pelas setas. Observado o padrão dos subconjuntos (Figura 42), nota-se, em ambos os casos, células livres onde não deveriam ser (representadas pelo círculo preto), e células que deveriam ser livres (representadas pelo triângulo) com o estado *concavidade*. Os demais padrões também falham, indicando que essa célula não deve ser usada como objetivo temporário. Isso é confirmado pela Figura 43(d), uma vez que o alargamento simulado ainda bloqueia a passagem e não deixa uma margem de segurança para evitar novas detecções. Após todas as células do raio 2 serem descartadas, a busca começa a avaliar a vizinhança de raio 3. Como pode ser observado na Figura 43(e), a primeira célula avaliada nesse novo raio é uma célula promissora, pois sua vizinhança corresponde ao padrão 1, e o alargamento simulado não bloqueia nenhuma passagem (Figura 43(f)).

Após a identificação de um possível objetivo temporário, o algoritmo tenta criar uma rota até ele. Quando é encontrada uma rota factível, ilustrada na Figura 44(a), o robô a utiliza como manobra de recuo. Feito isso e após aguardar o tempo de espera definido ($t2$), no qual espera-se que o robô em conflito já tenha se deslocado desfazendo a obstrução,o robô retoma seu objetivo original e uma nova rota é planejada, como observado na Figura 44(b).



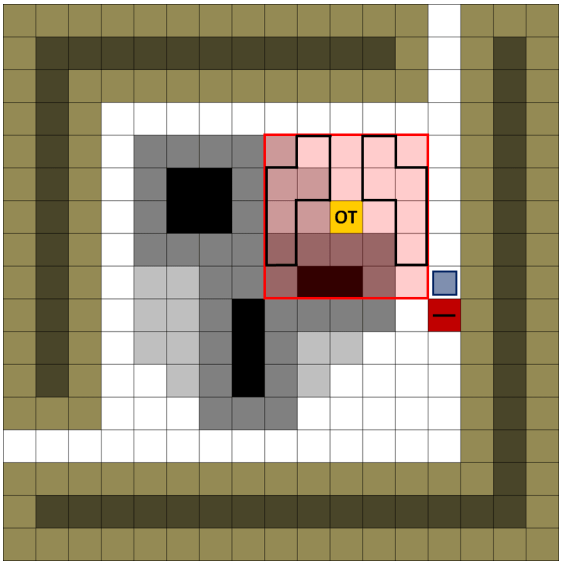
(a) Rota até o objetivo temporário

(b) Nova rota até o objetivo original

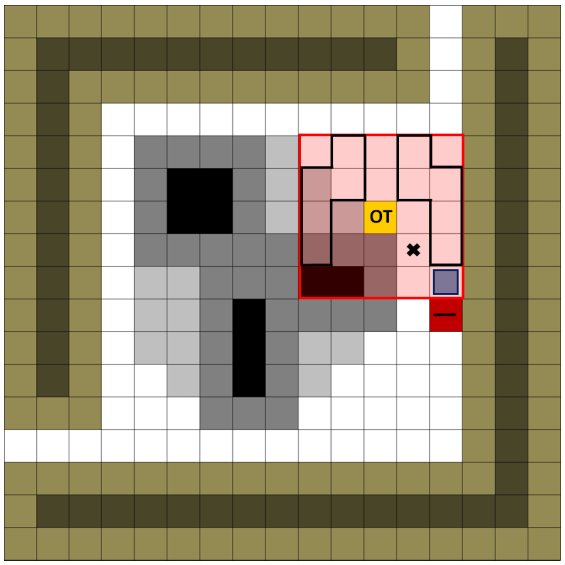
Figura 44 – Exemplo de rotas obtidas para a manobra de recuo.

Nos padrões especificados neste trabalho para a identificação de células promissoras para o objetivo temporário de recuo (Figura 41), as células cinza possuem uma função específica, que é garantir o deslocamento do robô para um local que previna a necessidade de uma nova manobra de recuo. Esse comportamento pode ser observado na Figura 45, a qual exemplifica o processo de busca e validação de possíveis células promissoras para o objetivo temporário, considerando o reticulado do cenário 3 (Figura 31(e)). As figuras da esquerda mostram o processo de validação de uma célula promissora. Como observado na Figura 45(a), a vizinhança da célula em avaliação (OT) corresponde com o padrão 3 e, portanto, é considerada uma célula promissora. Como é possível construir uma rota até essa célula, ela é adotada como o novo objetivo temporário. A Figura 45(c) representa a situação onde o robô superior (destacado em azul) está posicionado no objetivo temporário definido anteriormente, mas ao tentar retomar a trajetória para seu objetivo original, detecta novamente outro robô de menor prioridade (posicionado na célula em vermelho). As células em laranja indicam a área alargada do robô inferior. Esse robô detectado pode ser o mesmo que havia bloqueado o caminho anteriormente ou um diferente. Contudo, a localização do robô em análise (célula azul) garante que este robô inferior se movimente o suficiente para liberar a passagem que estava bloqueada anteriormente, ou seja, como podemos observar na Figura 45(c), o alargamento do robô inferior não bloqueou o melhor caminho até seu objetivo original e assim é possível criar uma rota até ele como observado na Figura 45(e). Esse comportamento não aconteceria caso o robô se movimente para a próxima célula de raio 1, como podemos observar nas figuras a direita. Na Figura 45(b), podemos observar que existe uma falha na comparação da vizinhança da célula em avaliação com o padrão 3, uma vez que a posição equivalente ao x deveria ser *obstáculo_virtual*, *parede_virtual* ou *concavidade*, mas ela é de estado *livre*. Os demais padrões também não correspondem com a vizinhança da célula em avaliação. A Figura 45(d) ilustra um cenário hipotético, no qual essa célula é escolhida como objetivo temporário. Se após a manobra de recuo, algum robô fosse detectado, seu alargamento bloquearia novamente passagem, exigindo que ele faça uma nova manobra de recuo, já que a única rota até seu objetivo original é inviável (Figura 45(f)). Isso ocorre porque o robô não se distanciou o suficiente durante a manobra de recuo.

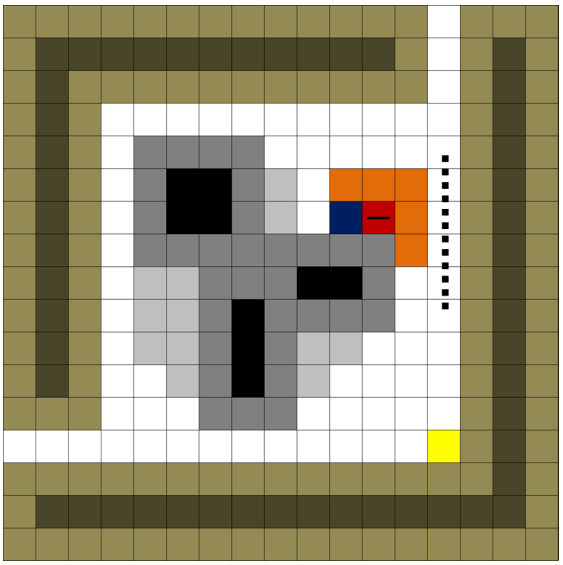
A eficiência das mudanças propostas para o modelo de planejamento de caminhos, bem como a eficácia do nosso sistema de controle de navegação para um time com muitos robôs foram avaliadas através de experimentos, os quais são descritos no próximo capítulo.



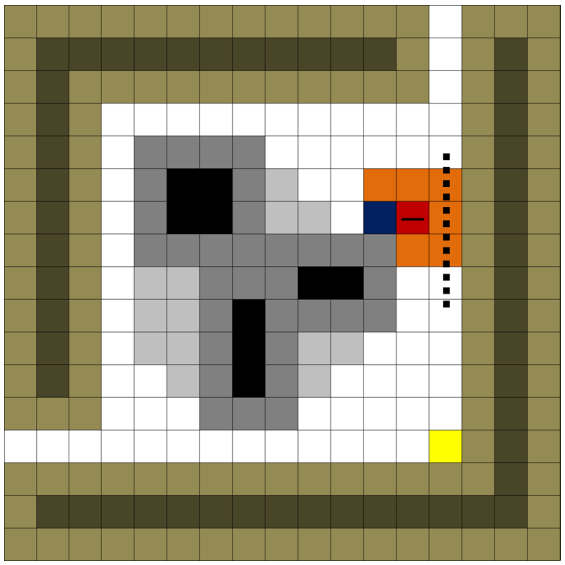
(a) Analise do padrão 1 em uma célula de raio 3



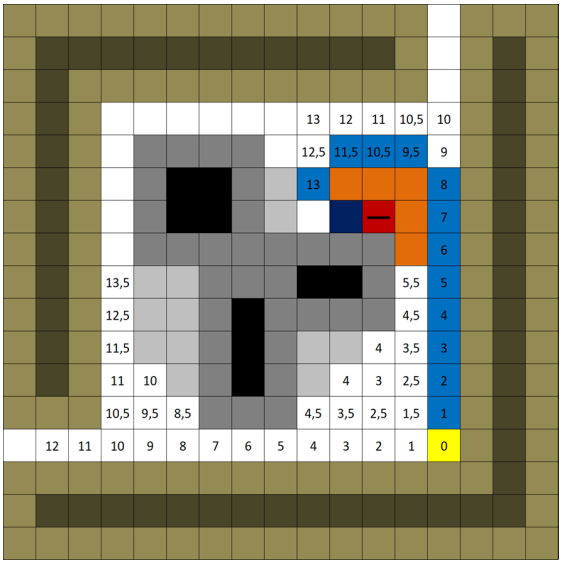
(b) Analise do padrão 3 em uma célula de raio 2



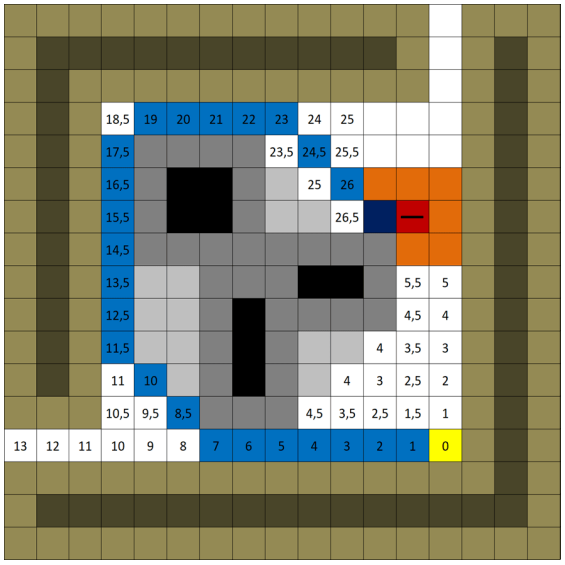
(c) Grade celular após nova detecção



(d) Grade celular após nova detecção



(e) Rota válida



(f) Rota inválida

Figura 45 – Comparação entre uma célula promissora e uma não promissora.

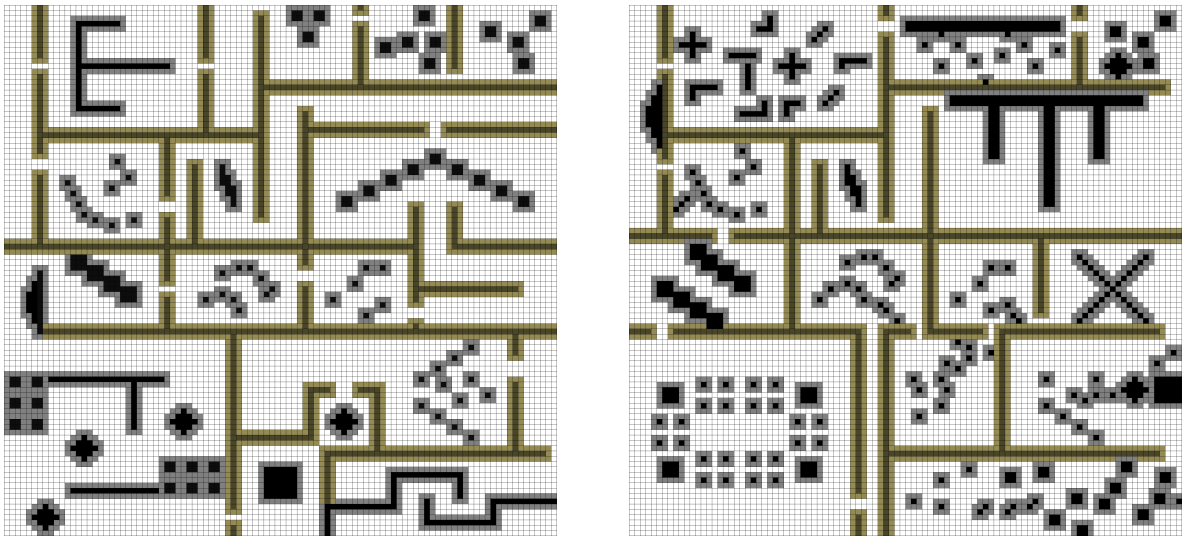
Capítulo 5

Experimentos e Resultados

Neste capítulo são apresentados os experimentos realizados durante a pesquisa a fim de validar o modelo proposto para o sistema de planejamento e navegação de robôs autônomos (BioTeam). Inicialmente, é analisada a eficiência do modelo no planejamento de caminhos para um único robô em cenários complexos, tanto em ambiente computacional, quanto simulado. Em seguida, verifica-se a eficácia da abordagem proposta para o controle de navegação descentralizado de um time de robôs. Todos os experimentos foram executados em um laptop Acer Aspire VX 15 com 16 GB de memória RAM, um processador Intel i7 (7th) de 64 bits e uma placa de vídeo NVIDIA GeForce GTX 1050 TI. A descrição de cada um dos experimentos realizados, incluindo os resultados obtidos e suas respectivas análises, é apresentada a seguir.

5.1 Avaliação do Modelo de Planejamento de Caminho

Esse primeiro experimento consiste na realização de simulações em ambientes computacionais mais simples, implementados na linguagem C, os quais implementam modelos que reproduzem a navegação ideal dos robôs pela rota planejada, sem considerar os erros e problemas inerentes aos aspectos físicos do ambiente e do robô. Esses ambientes simplificados foram desenvolvidos com o objetivo de mostrar o comportamento do modelo de difusão de distância proposto e confrontá-lo com o algoritmo discutido em (MARTINS et al., 2018). Nossa análise considera os resultados da aplicação dos modelos em dois ambientes com 100x100 células, chamados de A1 e A2. A Figura 46 mostra ambos os ambientes após o alargamento dos obstáculos e paredes. Cada ambiente possui várias salas com paredes, portas estreitas e alguns obstáculos internos, o que aumenta a complexidade do planejamento de caminho.



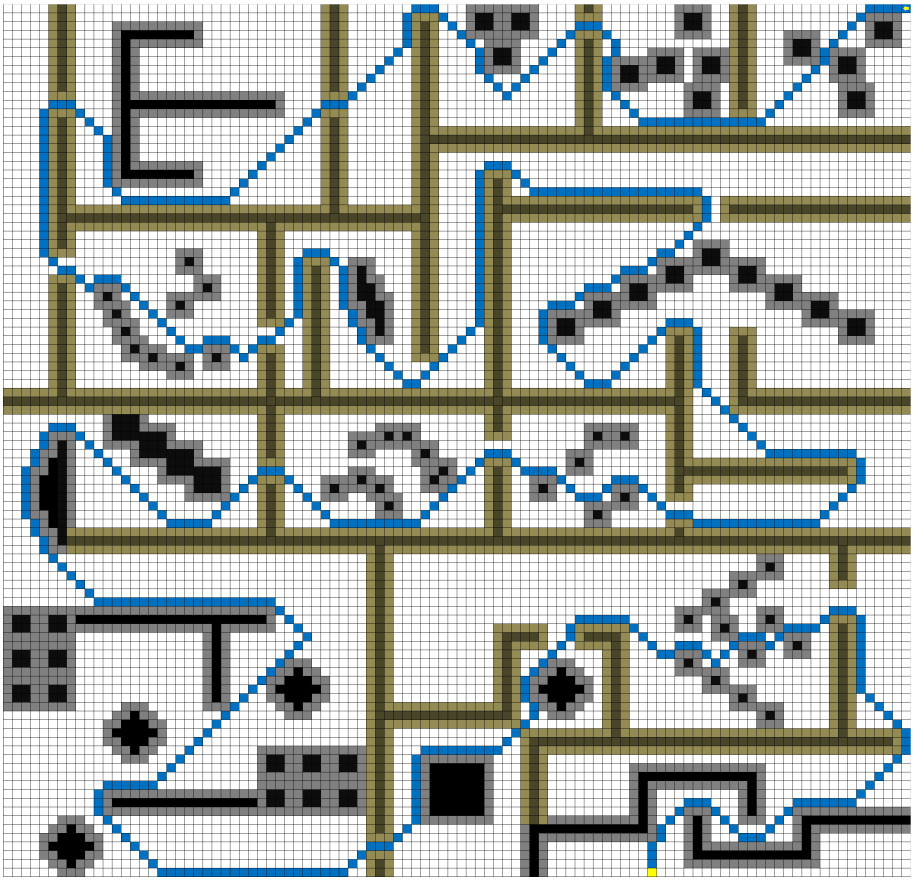
(a) Ambiente A1 (b) Ambiente A2

Figura 46 – Ambientes virtuais usados nos experimentos.

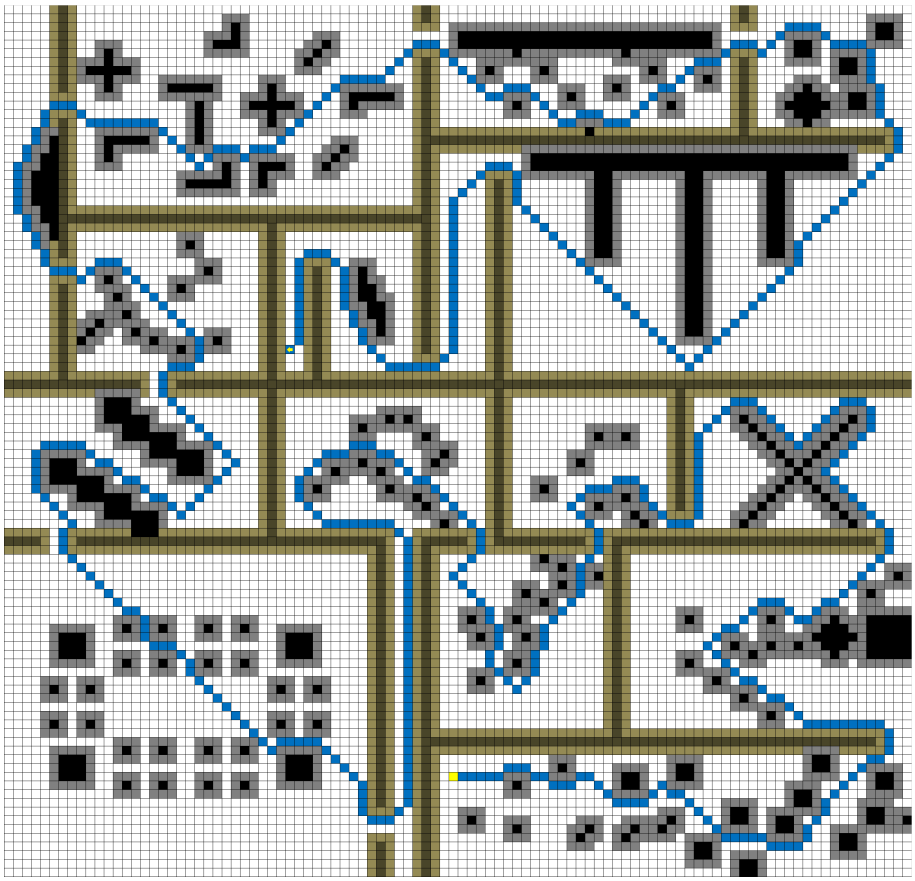
Com o objetivo de esclarecer a contribuição de cada modificação incorporada ao modelo investigado, foram elaboradas duas versões do algoritmo de planejamento de caminhos. A primeira versão apenas difere entre movimentos diagonais e laterais (regras GDLS e GDDS), mas não aplica o sombreamento de regiões côncavas nos obstáculos (regra ARC). Essa versão simplificada é chamada de PCD (planejamento de caminhos com diferenciação diagonal). A segunda é uma versão completa do modelo descrito no capítulo 3. Ou seja, além de usar as regras GDLS e GDDS, ele também sombreia as regiões côncavas usando a regra ARC. Essa versão é chamada PCDC (planejamento de caminho com diferenciação diagonal e sombreamento das regiões côncavas dos obstáculos). As duas versões foram confrontadas com os resultados do algoritmo original para cálculo de rotas usado em (MARTINS et al., 2018) (aqui denominado PC) para verificar o desempenho do novo modelo.

Cada versão do algoritmo de planejamento de caminho foi executado em ambos os ambientes (A1 e A2), conforme ilustrado nas Figuras 47, 48 e 49. Em todas as simulações foram consideradas as mesmas células iniciais e finais (objetivo) para o robô, cujas posições foram estrategicamente definidas de modo a garantir que o robô atravesse todas as salas para encontrar o alvo. A rota planejada por cada modelo é destacada em azul, com a célula objetivo do robô em amarelo.

Por meio das simulações realizadas, observa-se que não há alteração na distância a ser percorrida pelo robô para chegar ao seu objetivo, ou seja, na quantidade de células presentes nos caminhos planejados pelos modelos. Entretanto, é possível perceber uma redução no custo da rota, cujo cálculo é baseado nos movimentos do robô (somando 1 para movimentos laterais e 1,5 para os diagonais, e na quantidade de rotações realizadas durante o percurso da posição atual do robô até o seu objetivo.

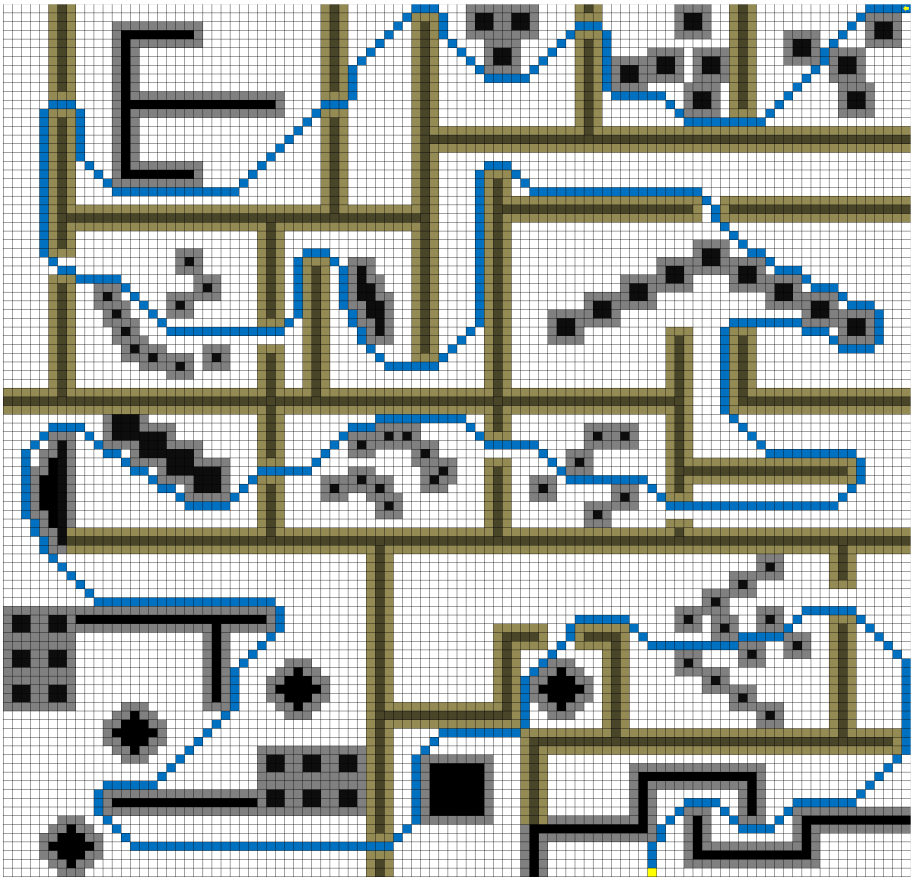


(a) Ambiente A1

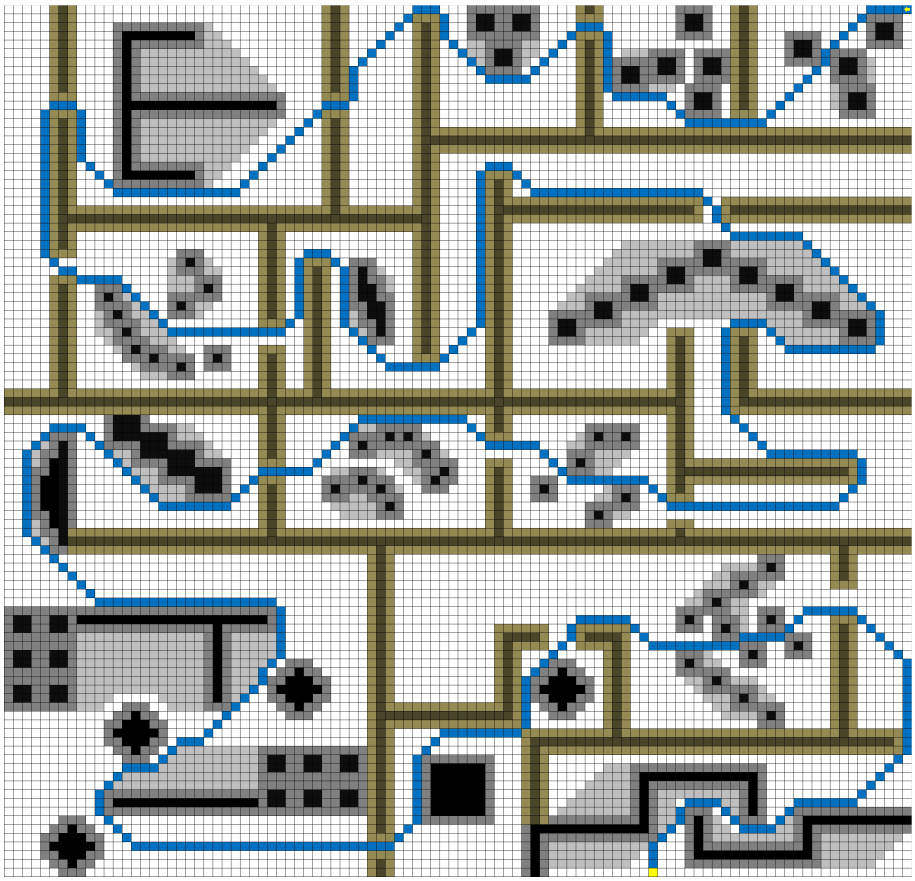


(b) Ambiente A2

Figura 47 – Rotas planejadas em A1 e A2 pela PC.

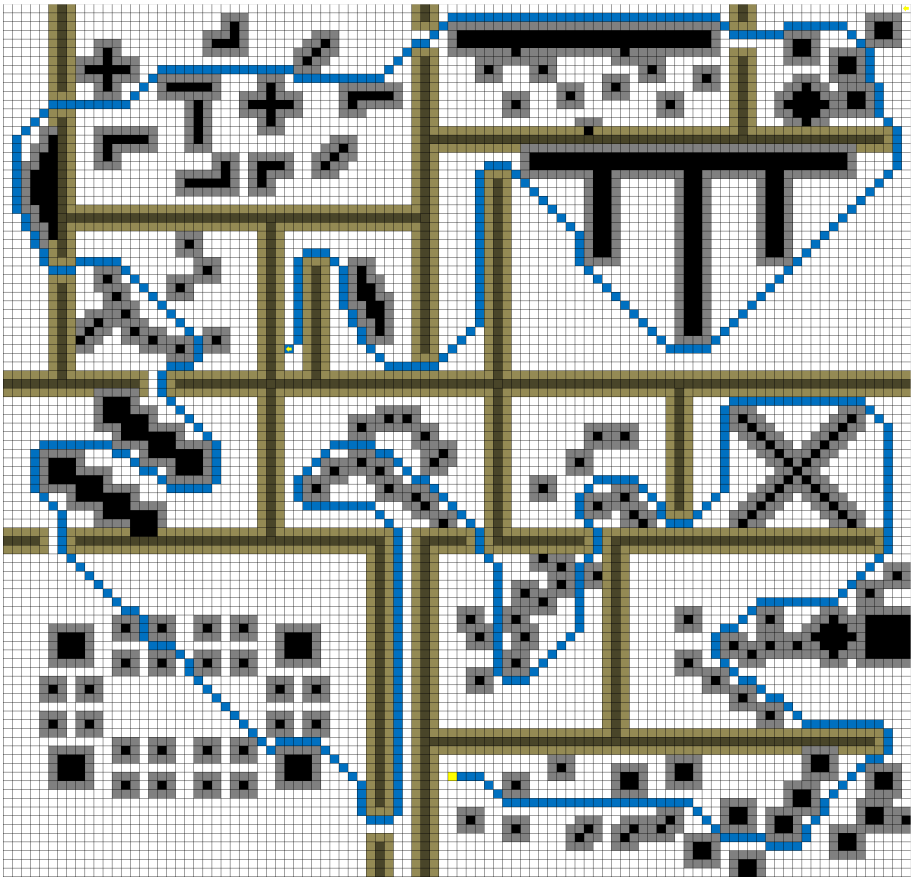


(a) PCD

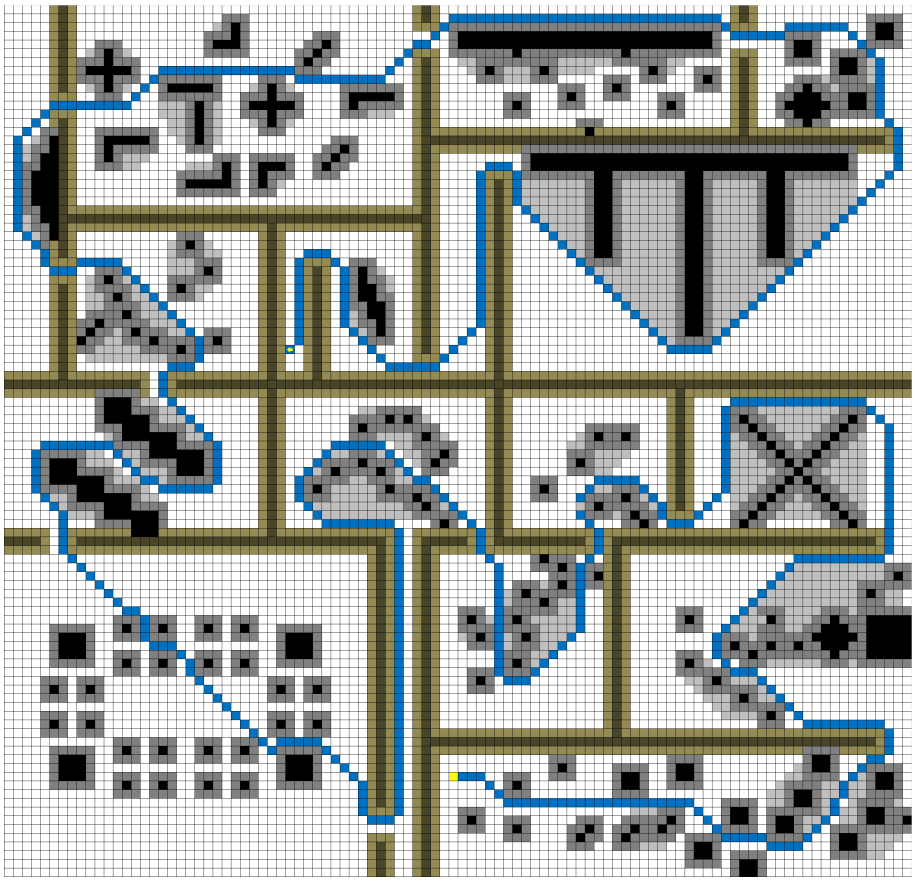


(b) PCDC

Figura 48 – Rotas planejadas em A1 pela PCD e PCDC.



(a) PCD



(b) PCDC

Figura 49 – Rotas planejadas em A2 pela PCD e PCDC.

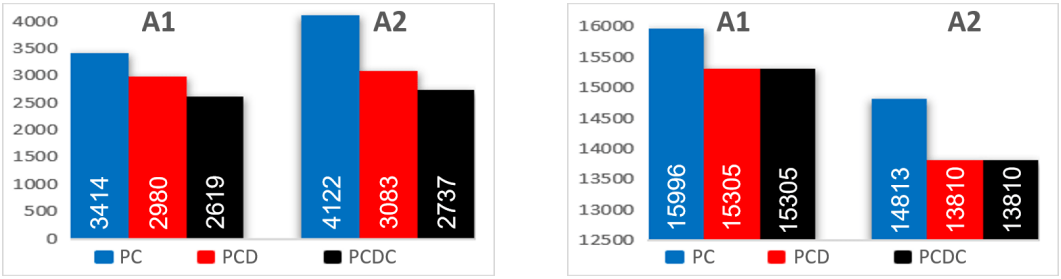
Na Figura 47 são apresentadas as rotas planejadas em cada ambiente utilizando o modelo original (PC). A Figura 47(a) destaca a rota calculada no ambiente 1, a qual possui um custo de 738 e demanda 158 rotações do robô. Na Figura 47(b) é apresentada a rota calculada para o ambiente 2, com custo da rota igual 747 e a quantidade de giros igual a 211. Para facilitar a comparação visual entre os modelos PCD e PCDC, os resultados foram agrupados de acordo com o ambiente. Assim, a Figura 48 apresenta as rotas planejadas para o ambiente A1 e a Figura 49 mostra as rotas obtidas no ambiente A2. Considerando o primeiro ambiente (A1), a verão PCD obteve uma rota com custo de 702 e que exige 144 rotações durante seu percurso (Figura 48(a)), enquanto a versão PCDC resultou em uma rota de mesmo custo (702) e 122 giros (Figura 48(b)). No ambiente A2, a rota planejada pelo PCD tem um custo de 697 e exige 150 rotações do robô (Figura 49(a)), enquanto a versão PCDC obteve uma rota com o custo de 697 e que demanda 132 giros (Figura 49(b)).

Como se pode notar, PCD diminuiu o custo da rota de 738 para 702 e também reduziu a quantidade de giros de 158 para 144 em A1, quando comparado com PC. Por outro lado, PCDC foi capaz de atingir uma redução extra na quantidade de giros, exigindo 122 rotações do robô durante o percurso planejado para A1, com o mesmo custo da rota de PCD (702). Resultados semelhantes foram obtidos a partir das rotas planejadas para o ambiente A2. PCD e PCDC foram capazes de reduzir o custo da rota de 747 para 697 e também diminuíram a quantidade de giros necessários para o robô percorrer o caminho planejado de 211 para 150 e 132, respectivamente.

Esses resultados preliminares destacam a capacidade de cada aprimoramento proposto para o modelo de planejamento de caminho. A diferenciação entre movimentos diagonais e laterais permite encontrar rotas com menor custo e também reduz a quantidade de rotações do robô, pois evita movimentos diagonais. Já, o sombreamento de regiões côncavas consegue reduzir ainda mais a quantidade de rotações, retornando trajetórias mais suaves para os robôs, sem alterar o seu custo.

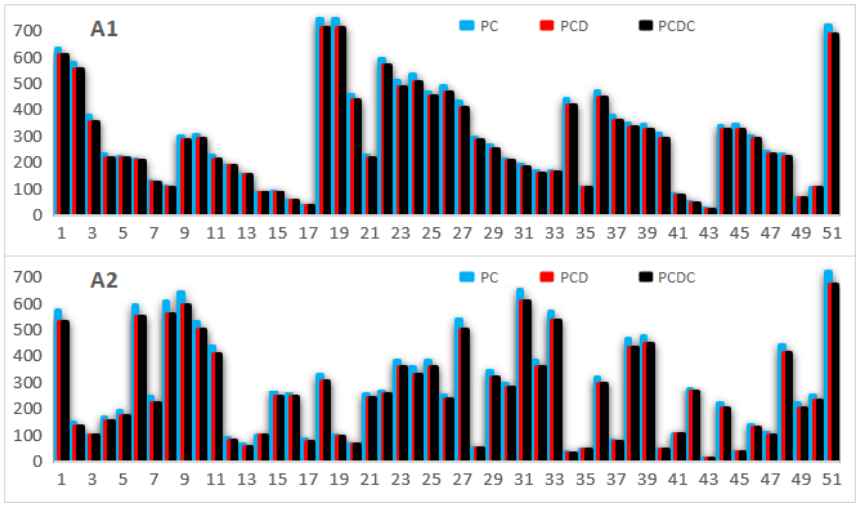
Em seguida, uma série de 50 novas execuções foram realizadas para cada ambiente, nas quais as posições inicial e objetivo foram geradas aleatoriamente. O objetivo é quantificar as melhorias do modelo proposto em termos de custo da rota e quantidade de rotações do robô. Os resultados consolidados por abordagem, obtidos a partir da soma dos valores de todas as 50 execuções para cada ambiente, são apresentados na Figura 50, enquanto a Figura 51 mostra os resultados individuais dos modelos em cada simulação. Em ambas as figuras, as barras azuis representam o desempenho do PC, as vermelhas representam o PCD e as pretas representam o PCDC. A versão completa do modelo (PCDC), quando aplicada ao ambiente A1, foi capaz de reduzir em 4,3% o custo da rota e em 23,2% a quantidade de giros em relação ao modelo original (PC). Para o ambiente A2, foi possível reduzir em 5,1% o custo da rota e em 25,7% a quantidade de rotações. Comparando as duas versões do algoritmo proposto, a adição do sombreamento das regiões côncavas

retornou o mesmo custo da rota em todas as 100 execuções e atingiu uma redução extra na quantidade de giros de 10,4% em A1 e 8,0% em A2. Além disso, a quantidade de rotações em 73 simulações diminuiu devido ao sombreamento da concavidade, enquanto em apenas 2 delas esse número aumentou ligeiramente.

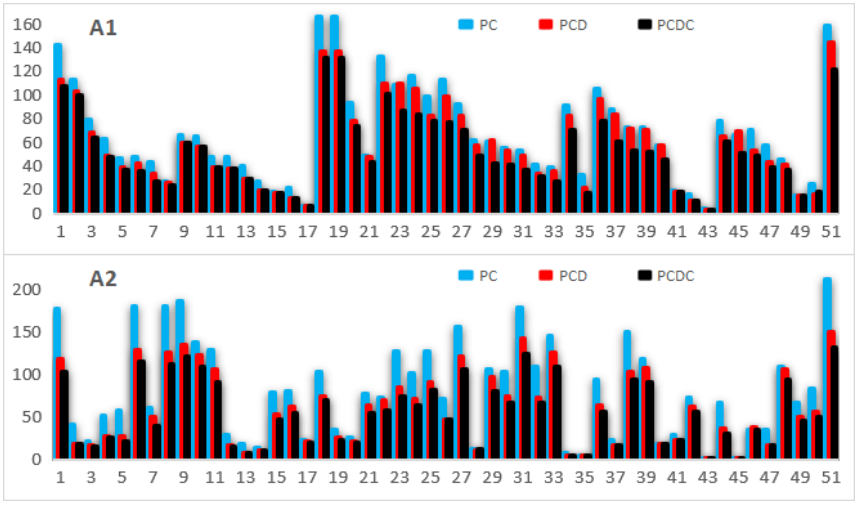


(a) Custo da rota (b) Quantidade de rotações do robô

Figura 50 – Desempenho consolidado das abordagens investigadas.



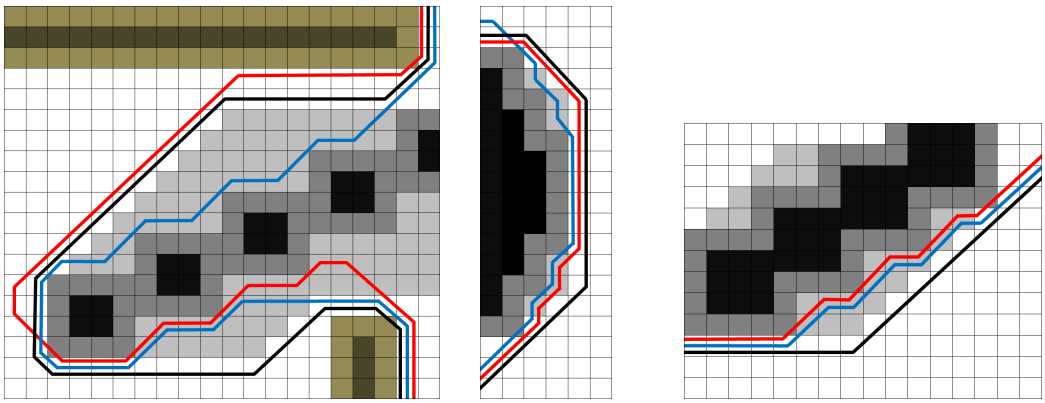
(a) Custo da rota



(b) Quantidade de giros

Figura 51 – Desempenho das abordagens investigadas por simulação.

A Figura 52 mostra em detalhes as mudanças no planejamento da rota extraídas das simulações nos ambientes A1 e A2, quando os algoritmos PCD ou PCDC são usados em vez de PC. O trajeto original (PC) é mostrado em azul, enquanto o trajeto vermelho foi construído usando PCD e o trajeto preto foi planejado usando PCDC. Pode-se observar que o PCDC é capaz de construir rotas mais suaves, evitando os obstáculos das regiões côncavas. É importante notar que a etapa de sombreamento transforma células livres em objetos virtuais através da aplicação sucessiva da regra de ARC, enquanto houver um caminho alternativo possível para o robô. Por exemplo, na Figura 52(a), a região côncava abaixo dos obstáculos foi coberta em direção à parede vertical alargada até que pelo menos uma célula livre permanecesse entre a área de sombra e a parede. Este comportamento é consequência dos padrões de disparo da regra ARC apresentados na Figura 19, de modo que a célula central passará a obstáculo virtual apenas se o canto oposto for uma célula livre, garantindo assim a existência de pelo menos um caminho livre. Por outro lado, as Figura 52(b) e Figura 52(c) ilustram situações em que a região côncava é totalmente coberta, uma vez que não há outro obstáculo nas proximidades. Dessa forma, o caminho planejado é ainda mais linear, evitando movimentos em zigue-zague e reduzindo giros.



(a) A1 (b) A1 (c) A2

Figura 52 – Visão detalhada de três trechos das rotas calculadas.

5.2 Impacto do Modelo Proposto na Navegação de um Único Robô

Após a confirmação dos efeitos positivos para a construção da rota planejada, simulações mais realistas foram realizadas, utilizando a plataforma Webots (CYBERBOTICS, 2021a). Ao contrário dos experimentos anteriores, os ambientes implementados no Webots simulam vários aspectos físicos, possibilitando avaliar, de forma mais realista, a eficiência do robô na execução das rotas planejadas. Para esses experimentos, foi implementado o mesmo modelo de navegação proposto em (MARTINS et al., 2018), o qual controla os movimentos de um único robô e-puck pelo ambiente simulado, após o planejamento da

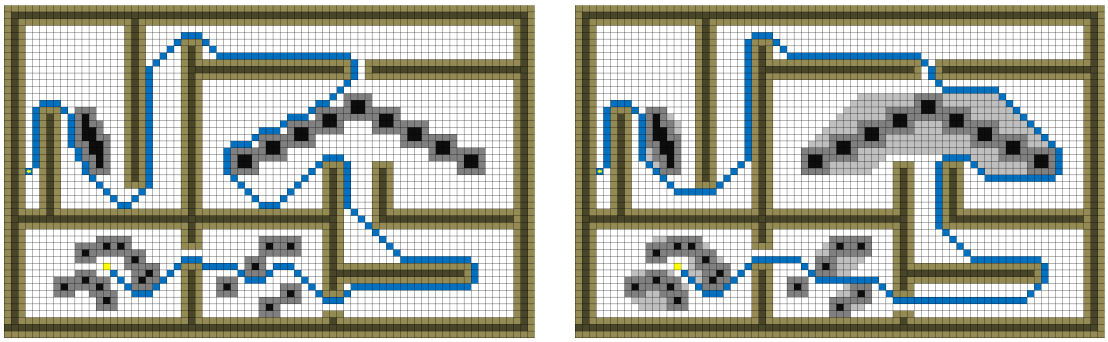
rota. Esse modelo usa apenas os sensores de posição do robô (sensores de roda esquerda e direita) para o cálculo da odometria.

Nossa implementação também considera um segundo robô e-puck (robô emissor), o qual foi adicionado para emular o comportamento de um dispositivo que captura e processa a imagem do ambiente, uma vez que todos os experimentos são simulados e o desenvolvimento do sistema de captura e processamento de imagens está fora do escopo deste trabalho. Neste trabalho, os dados enviados pelo emissor ao robô navegante para o planejamento da rota são adicionados manualmente, de forma estática, de acordo com cada ambiente simulado. As localizações dos objetos correspondem às coordenadas (x, y) de suas respectivas células no mapa do ambiente. Assim, o mapa enviado do robô emissor para o navegante é representado por um vetor que armazena somente as informações importantes desse mapa (posições de cada obstáculo, parede, robô e da célula objetivo). A conexão entre os dois robôs é feita por bluetooth dentro da plataforma Webots (CYBERBOTICS, 2021a).

Uma vez que a simulação no Webots é uma tarefa que exige mais memória e tempo de processamento do que as simulações iniciais no programa implementado em C, optamos por elaborar um novo ambiente mais simples que A1 e A2, para validar a navegação do robô em uma plataforma de simulação robótica mais realística. Esse ambiente foi denominado A3, e foi baseado em trechos do A1 e do A2 representado por um reticulado de 70×45 células, cada uma com 7×7 cm. Portanto, cada passo do robô leva 7 cm se for um movimento lateral e aproximadamente 10 cm se for um movimento diagonal.

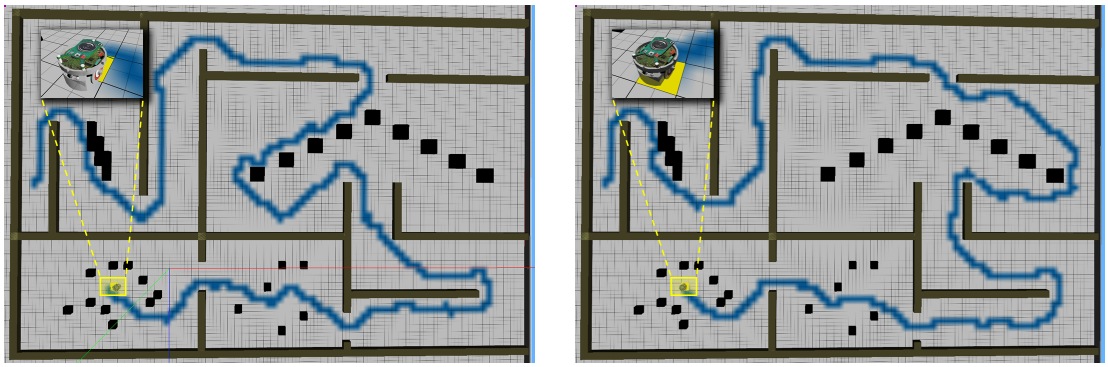
A Figura 53 apresenta as rotas planejadas para o A3 usando o PC original e o PCDC proposto. É possível observar que a etapa de sombreamento junto com a diferenciação diagonal gera uma rota que evite passar nas regiões de concavidade formadas por obstáculos próximos retornando uma trajetória mais linear, com menos mudanças de direção. Além disso, a rota planejada usando PCDC é mais curta. Numericamente, PC (Figura 53(a)) calculou uma rota com distância total (quantidade de células para chegar ao objetivo) igual a 235 e 54 rotações do robô, enquanto PCDC (Figura 53(b)) gerou uma rota com distância total igual a 225 e 38 rotações.

A Figura 54 mostra a trajetória do robô simulado pelo Webots ao navegar de acordo com a rota planejada calculada usando o algoritmo PC e PCDC. Nos dois casos é possível observar que o robô foi capaz de atingir a célula objetivo a partir de sua posição inicial, fazendo uma trajetória livre de colisões muito próxima da planejada. No entanto, a navegação usando PCDC foi quase 1 minuto mais rápida do que usando PC (14min15s contra 15min05s), uma consequência de sua rota mais suave. Portanto, através desse experimento pode-se concluir que o novo modelo de planejamento de caminhos (PCDC) é capaz de construir rotas curtas e suaves, enquanto o modelo de controle faz com que o robô execute sua trajetória muito próximo da rota planejada. O vídeo deste experimento pode ser encontrado em (NAMETALA, 2021b)



(a) PC (b) PCDC

Figura 53 – Rotas planejadas para o ambiente A3.



(a) PC (b) PCDC

Figura 54 – Rotas executadas em A3 pelo simulador Webots.

Para confirmar a capacidade do novo modelo em evitar regiões côncavas, foram criados cinco novos ambientes denominados AF1, AF2, AF3, AF4 e AF5. Eles foram formados por um reticulado de 50×28 células de tamanho 7×7 cm. Nesses ambientes os obstáculos são distribuídos propositalmente pelo ambiente para formarem várias regiões côncavas, as quais devem ser evitadas pelo robô durante sua navegação. A Figura 55 mostra os cinco ambientes utilizados, bem como as rotas planejadas no ambiente simplificado usando o modelo PC (figuras da esquerda) e o novo modelo PCDC (figuras da direita). Já a Figura 56 apresenta a trajetória realizada pelo robô na plataforma de simulação Webots, ao navegar de acordo com as rotas planejadas. Novamente, as figuras da esquerda mostram as trajetórias do robô para as rotas calculadas pelo PC, enquanto que as figuras da direita representam as trajetórias obtidas a partir das rotas planejadas pelo PCDC.

A Tabela 2 mostra os valores obtidos para as métricas de desempenho (custo da rota e quantidade de rotações) para as rotas planejadas nas execuções dos ambientes da Figura 56, utilizando ambos os modelos (PC e PCDC). Ainda na Tabela 2, é destacado em negrito a redução alcançada pelo novo modelo em cada métrica por ambiente. A Tabela 3 mostra o tempo necessário para o planejamento e navegação das rotas de acordo com o ambiente e o modelo simulados no Webots. Ela também destaca (em negrito) o quanto

foi possível reduzir o tempo de simulação utilizando o modelo proposto. De forma geral, considerando-se os 5 ambientes da Figura 53, pudemos alcançar com o uso do PCDC, em média, uma redução de 68% no número de rotações, 16% na distância percorrida (custo) e 9% no tempo de navegação.

	Quantidade de rotações		Custo da rota	
Ambientes	PC	PCDC	PC	PCDC
AF1	30	12 **(60%)**	109,5	97,5 **(11%)**
AF2	51	10 **(80%)**	116,5	102,5 **(12%)**
AF3	21	9 **(57%)**	121,5	103,5 **(15%)**
AF4	21	6 **(71%)**	118,0	93,0 **(21%)**
AF5	26	8 **(69%)**	130,0	105,0 **(19%)**

Tabela 2 – Desempenho alcançado a partir das rotas planejadas em cada ambiente.

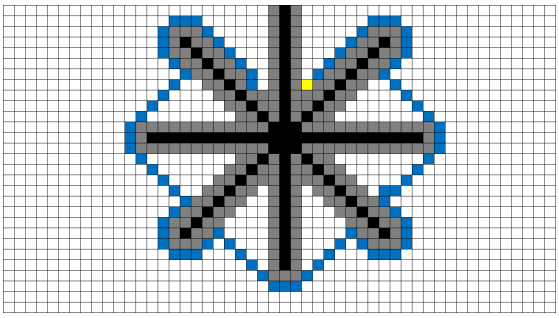
	Tempo de execução	
Ambientes	PC	PCDC
AF1	6:42 min	6:13 min **(7%)**
AF2	8:18 min	7:10 min **(12%)**
AF3	8:07 min	7:36 min **(6%)**
AF4	7:29 min	6:34 min **(12%)**
AF5	8:09 min	7:26 min **(9%)**

Tabela 3 – Tempo de simulação da navegação do robô pelos ambientes.

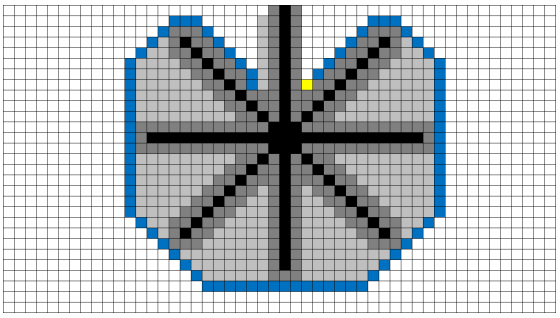
Comparando as rotas planejadas (Figura 55) com aquelas efetivamente executadas pelo robô durante as simulações no Webots (Figura 56), é possível notar que elas não são exatamente iguais. Isso acontece porque mesmo em simulação, o robô pode acumular vários erros de odometria, já que a plataforma Webots tenta aproximar ao máximo o comportamento simulado daquele observado no mundo real, tanto dinâmica quanto fisicamente. Apesar disto, através dos resultados apresentados nas tabelas, é possível concluir que o modelo PCDC consegue evitar de forma eficiente as regiões côncavas e suavizar a trajetória do robô, melhorando consequentemente as métricas de desempenho em comparação ao modelo original (PC). Além disso, vale ressaltar que em nenhuma das simulações houve colisão. Os vídeos com essas simulações estão disponíveis em (NAMETALA, 2021b).

Através deste experimento concluímos que a etapa de planejamento de caminhos combinada com estratégias de navegação com tomada de decisões locais pode retornar um modelo eficiente para o planejamento de caminhos de navegação de um time de robôs.

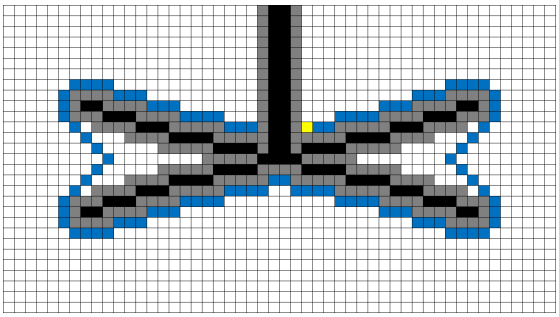
Essa análise relativa ao módulo de planejamento de caminhos serviu de base para elaborarmos e publicarmos um artigo em evento internacional (NAMETALA; MARTINS; OLIVEIRA, 2020). O evento ocorreu de forma remota e o vídeo da apresentação pode ser visualizado em (NAMETALA, 2020).



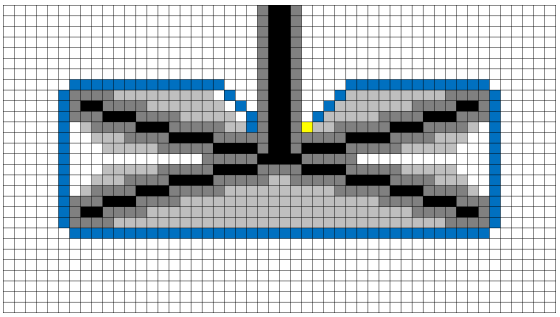
(a) Rota planejada usando PC para AF1



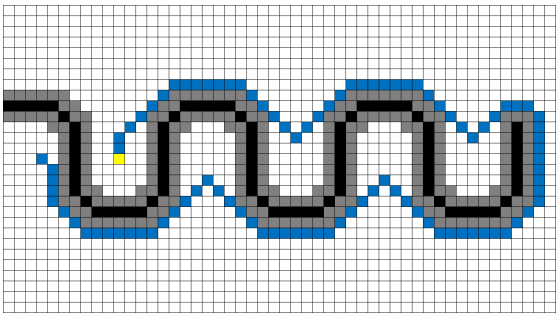
(b) Rota planejada usando PCDC para AF1



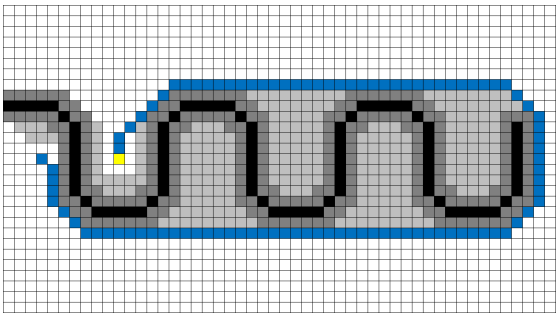
(c) Rota planejada usando PC para AF2



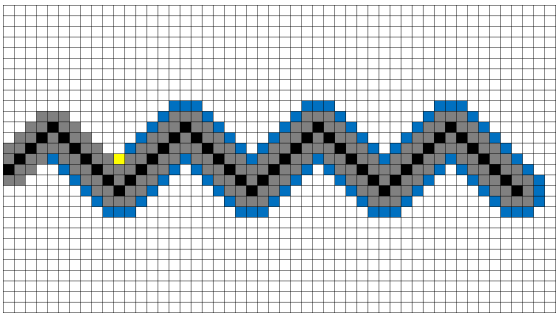
(d) Rota planejada usando PCDC para AF2



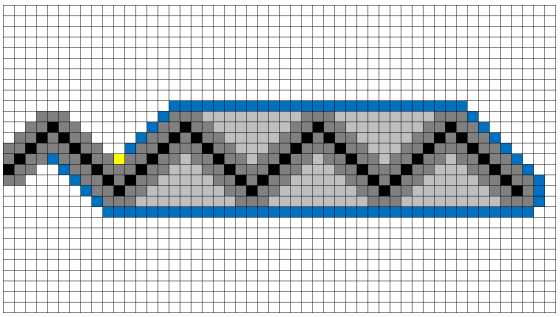
(e) Rota planejada usando PC para AF3



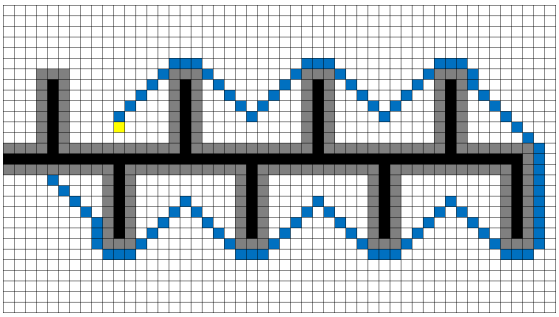
(f) Rota planejada usando PCDC para AF3



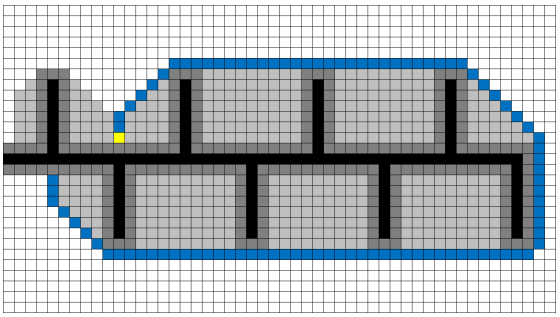
(g) Rota planejada usando PC para AF4



(h) Rota planejada usando PCDC para AF4

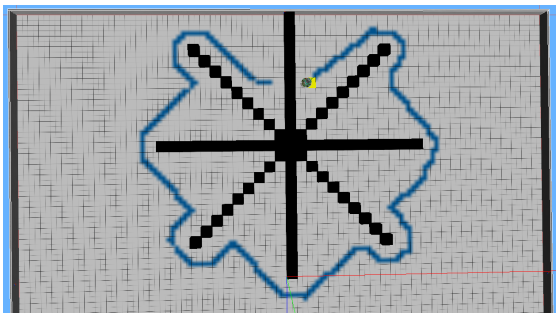


(i) Rota planejada usando PC para AF5

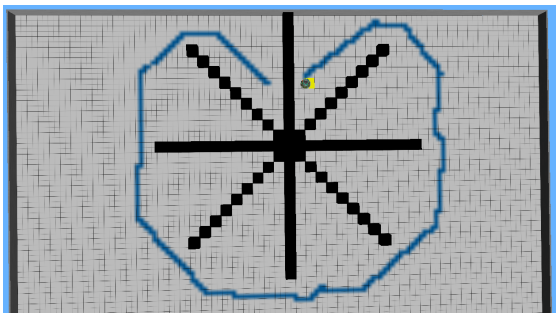


(j) Rota planejada usando PCDC para AF5

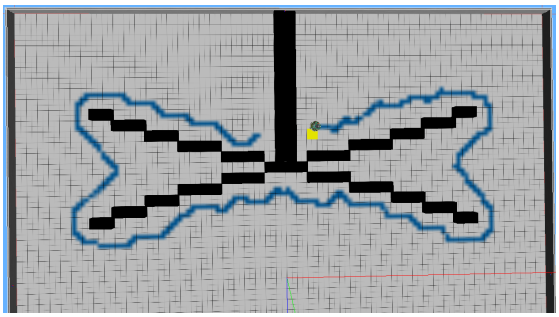
Figura 55 – Rotas planejadas de acordo com o modelo e ambiente simulados..



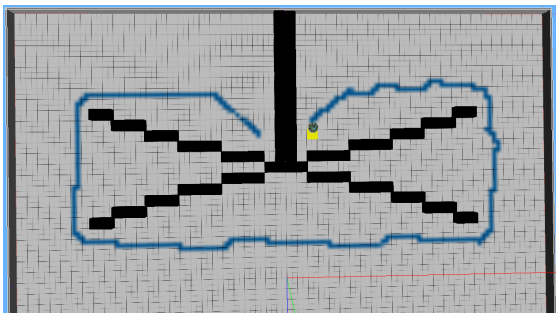
(a) Rota executada usando PC para AF1



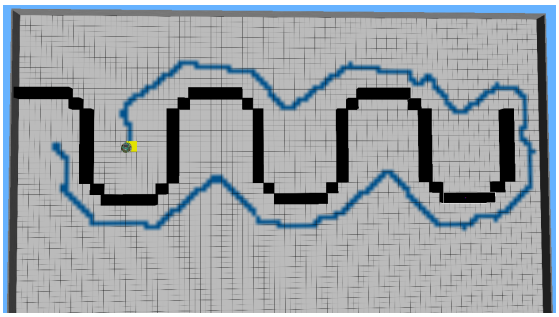
(b) Rota executada usando PCDC para AF1



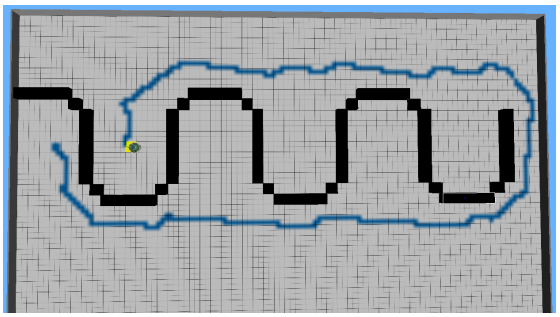
(c) Rota executada usando PC para AF2



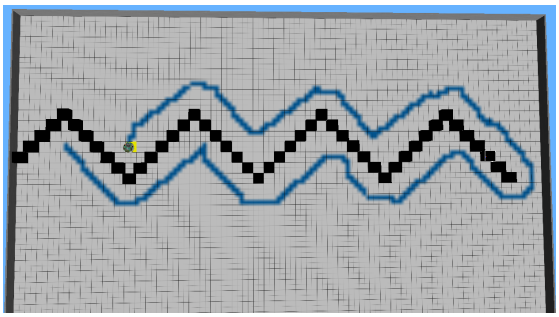
(d) Rota executada usando PCDC para AF2



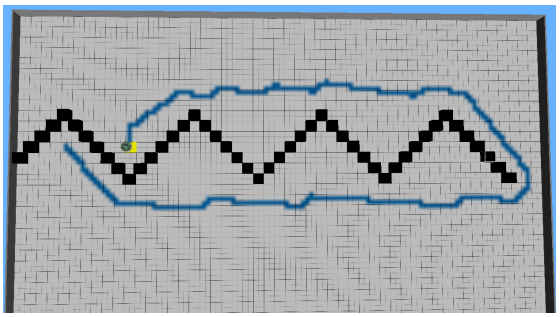
(e) Rota executada usando PC para AF3



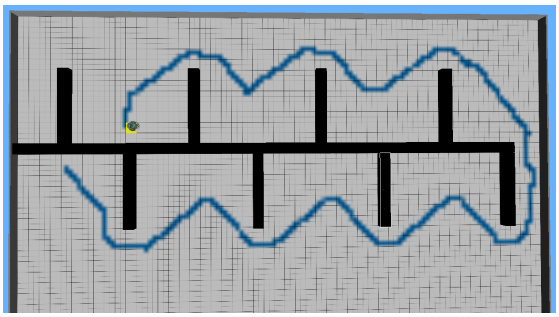
(f) Rota executada usando PCDC para AF3



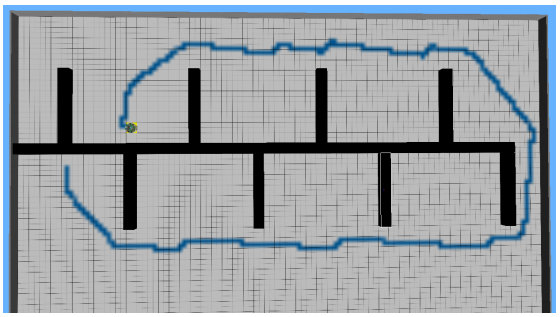
(g) Rota executada usando PC para AF4



(h) Rota executada usando PCDC para AF4



(i) Rota executada usando PC para AF5



(j) Rota executada usando PCDC para AF5

Figura 56 – Rotas executadas pelo robô nas simulações dos ambientes e modelos no Webots.

5.3 Avaliação do Novo Modelo de Navegação para Times de Robôs

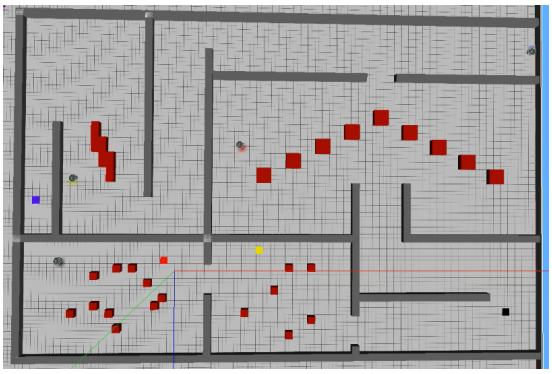
Com o modelo de planejamento de caminho refinado e com resultados promissores obtidos nas simulações com um único robô, o modelo foi estendido para controlar, de forma distribuída e descentralizada, a navegação de vários robôs pelo ambiente, realizando a avaliação do modelo BioTeam como um todo: planejamento de rota e navegação. Cada robô do time planeja sua rota, utilizando o modelo PCDC avaliado nas sessões anteriores e percorre essa rota até a célula objetivo (meta), de forma independente através de decisões locais.

Diferentes simulações foram realizadas a fim de validar o modelo proposto BioTeam. Essas simulações foram executadas utilizando a plataforma de simulação robótica Webots.

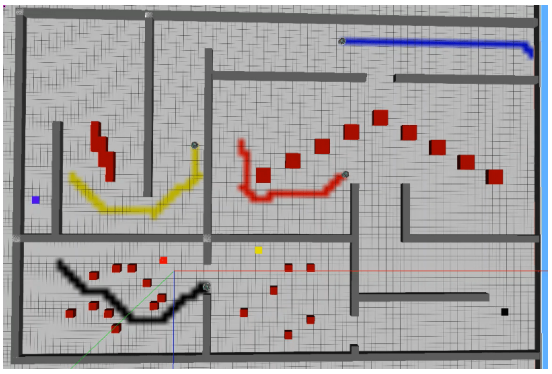
5.3.1 Análise das manobras de resolução de conflitos

Inicialmente, foi avaliada a eficácia de cada manobra na resolução de conflitos entre robôs durante o percurso das respectivas rotas. Esse experimento foi realizado no ambiente A3 (Figura 53), utilizando quatro robôs com rastros em cores diferentes para facilitar a visualização da rota executada por cada um. Para esse experimento a posição inicial e final de cada objetivo foi alterada diversas vezes a fim de buscar situações de conflito envolvendo 2 robôs.

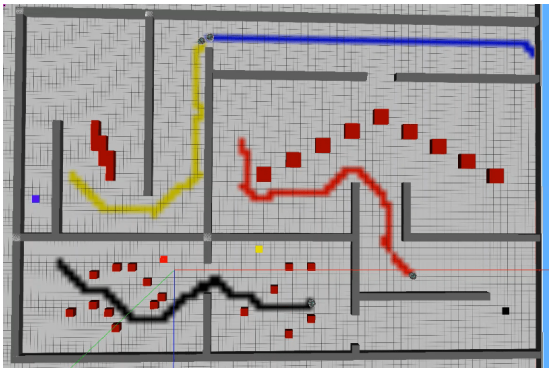
A Figura 57 mostra a evolução dos robôs partindo de suas posições iniciais até seus objetivos, em uma das configurações (posições iniciais e objetivos) aplicadas. Para esse exemplo, os robôs são nomeados de acordo com a cor de seus rastros. Na Figura 57(a) (t = 0m0s), pode-se ver o início da execução onde os robôs ainda não iniciaram seus movimentos. Na Figura 57(b) (t = 1m50s), os robôs já se moveram por quase 2 minutos em direção aos seus objetivos. No tempo 2m65 (Figura 57(c)), os robôs amarelo e azul se aproximaram o suficiente para se detectarem, resultando em uma situação de conflito que deve ser resolvida entre eles. No tempo 4m35s (Figura 57(d)), pode-se ver que o conflito entre os robôs amarelo e azul foi superado e, outra situação de conflito é identificada entre os robôs preto e vermelho. Na Figura 57(e) (t = 5m50s), o robô preto alcança seu objetivo, após superar o conflito identificado anteriormente. No instante 7m35s (Figura 57(f)), os robôs azul e vermelho também alcançaram seus objetivos. Na Figura 57(g) (t = 10m0s), o robô amarelo ainda está em movimento até seu objetivo. Por fim, no tempo 12m80s (Figura 57(h)), o robô amarelo também alcança seu objetivo e o experimento é encerrado com todos os robôs localizados em suas metas.



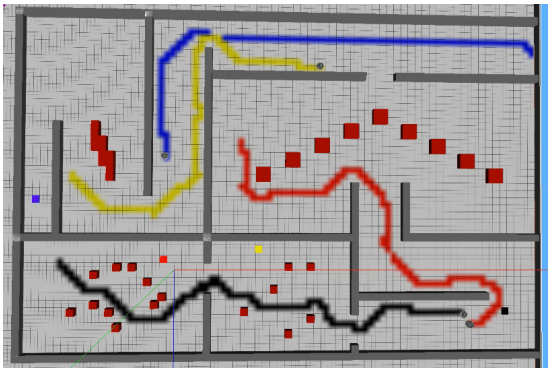
(a) t = 0m0s



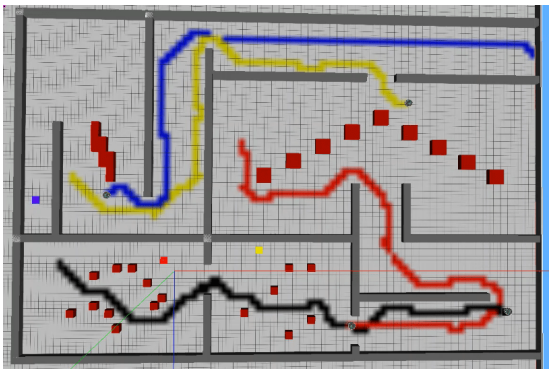
(b) t = 1m50s



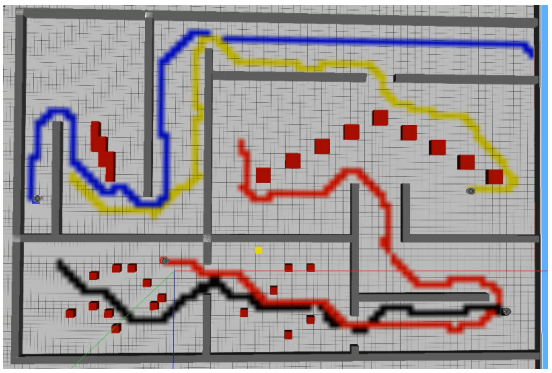
(c) t = 2m65s



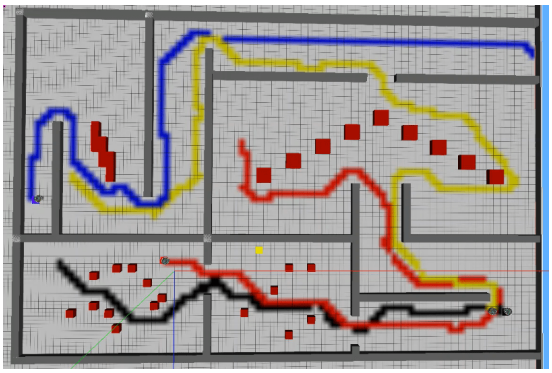
(d) t = 4m35s



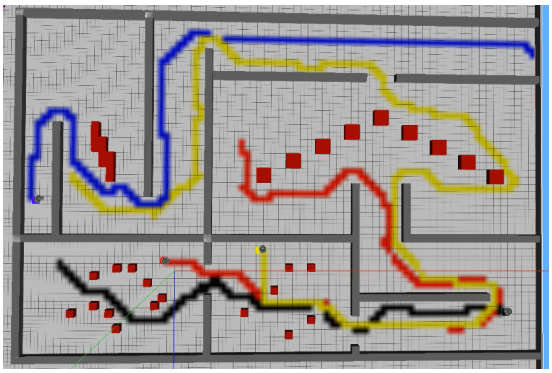
(e) t = 5m50s



(f) t = 7m35s



(g) t = 10m0s



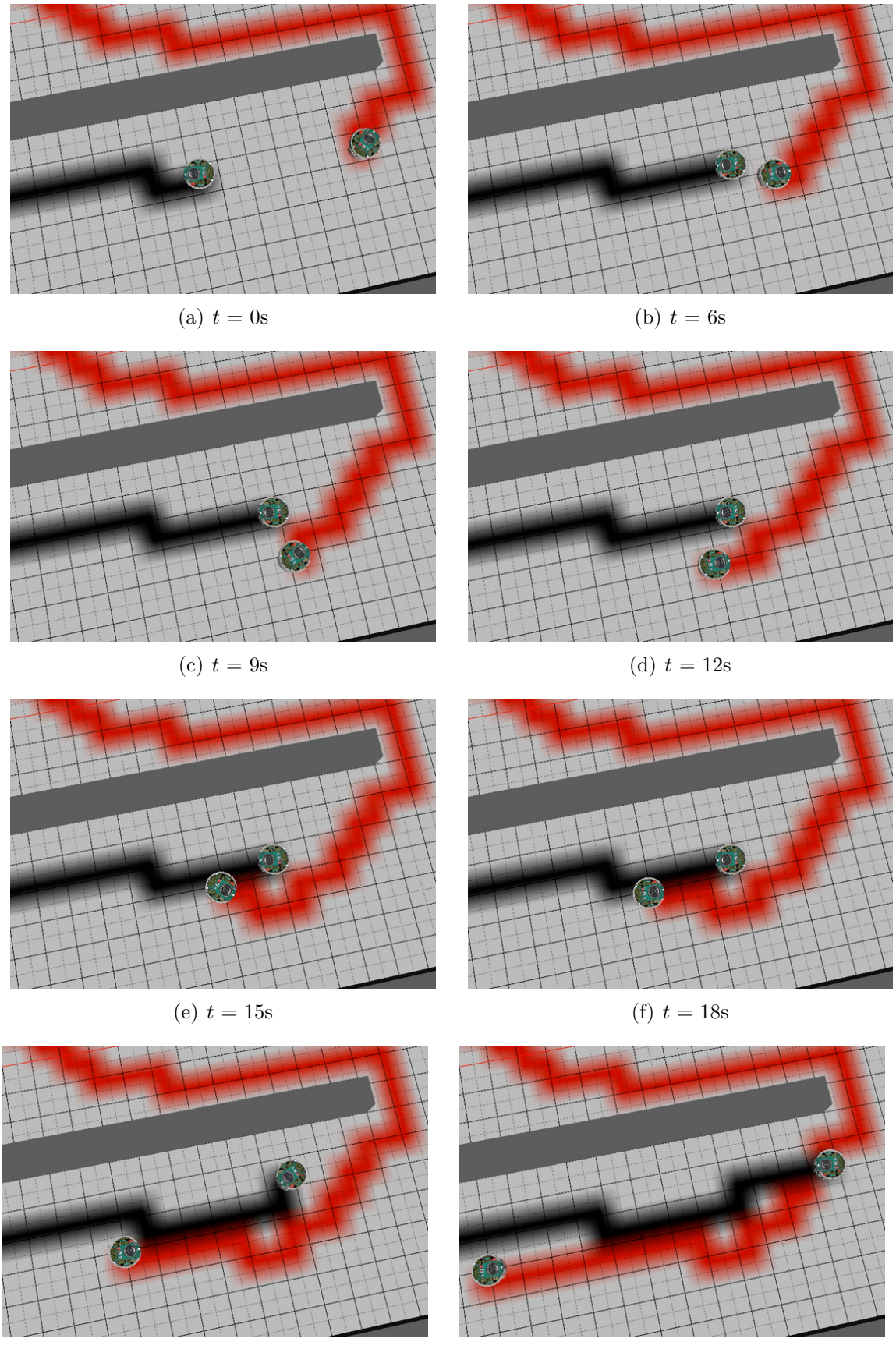
(h) t = 12m80s

Figura 57 – Evolução temporal da navegação de 4 robôs pelo ambiente A3.

A partir das simulações realizadas, como a do exemplo da Figura 57, foi possível identificar e analisar cada manobra envolvendo 2 robôs. As Figuras 58 e 59 mostram, respectivamente, a evolução de manobras de desvio e de recuo em diferentes instantes de tempo (t), focando nos robôs que estão em conflito. A manobra de espera também está representada nas figuras, uma vez que ela é realizada por um dos robôs nos dois exemplos.

No exemplo ilustrado na Figura 58 (manobra de desvio), o robô com rastro preto é de menor prioridade, enquanto o rastro vermelho representa o percurso daquele de maior prioridade. Na Figura 58(a) ($t = 0$s), os robôs já estão em movimento e ainda não se detectaram, pois existe uma distância segura entre eles. No instante $t = 6$s (Figura 58(b)), os robôs se aproximaram o suficiente para se detectarem e assim, interromperem seus percursos. No instante $t = 9$s (Figura 58(c)), o robô superior (vermelho) utiliza da manobra de desvio para calcular uma nova rota que considera o robô inferior (preto) e sua respectiva área alargada, possibilitando então que ele retome sua trajetória. Ao mesmo tempo, o robô inferior utiliza da manobra de espera, que faz com que ele fique parado por 12 segundos (4 ciclos de 3 segundos), o suficiente para o robô superior distanciar e não ser mais detectado. Nos instantes $t = 12$s (Figura 58(d)) e $t = 15$s (Figura 58(e)), o robô superior realiza mais alguns passos e consegue superar o robô inferior. Na Figura 58(f) ($t = 18$s), a distância entre os robôs é grande o suficiente para que eles não se detectem. Assim, quando o tempo de espera do robô inferior acaba, ele volta a se movimentar pelo ambiente, o que pode ser percebido no instante $t = 24$s (Figura 58(g)). Na Figura 58(h) ($t = 33$s), os robôs continuam suas trajetórias até os respectivos objetivos, podendo executar outras manobras, caso ocorram novas situações de conflito.

É interessante ressaltar que em momento algum dessas manobras ocorreu uma comunicação direta entre os robôs. Também não foi necessária a atuação de qualquer mediador externo. O sucesso e escolha das manobras é decorrência das informações da vizinhança recebidas a partir do processamento da imagem e decisões locais de cada robô de forma autônoma.



(a) $t = 0$s (b) $t = 6$s

(c) $t = 9$s (d) $t = 12$s

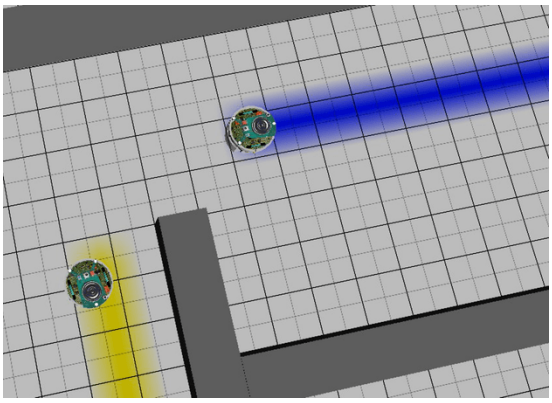
(e) $t = 15$s (f) $t = 18$s

(g) $t = 24$s (h) $t = 33$s

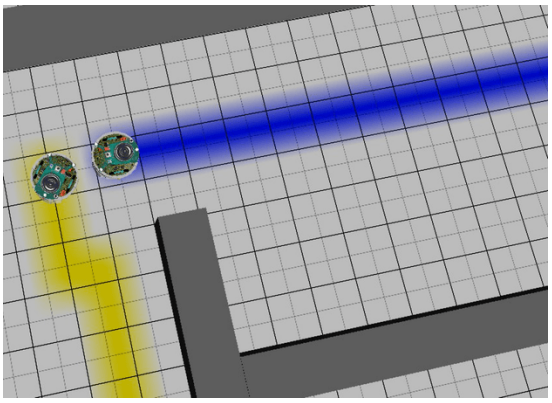
Figura 58 – Evolução temporal de uma manobra de desvio realizada durante a navegação de 2 robôs pelo ambiente A3.

A Figura 59 ilustra o comportamento da manobra de recuo. Nesse exemplo, o robô com rastro amarelo é de menor prioridade e o outro com rastro azul possui uma prioridade superior. Na Figura 59(a) (instante $t = 0$s), os robôs estão em movimento, pois a distância entre eles ainda não é suficiente para que sejam identificados como um obstáculo na vizinhança. Entretanto, no instante $t = 6$s (Figura 59(b)), os robôs já se aproximaram o suficiente para que possam se detectar e, assim, interromper sua navegação. Como a única passagem para o objetivo do robô superior está bloqueada pelo outro robô (amarelo), ele não consegue aplicar uma manobra de desvio, como no exemplo anterior. Após identificar essa impossibilidade de rota viável, o robô superior (azul) inicia uma a manobra de recuo, que pode ser percebida no instante $t = 9$s (Figura 59(c)). Para realizar esse recuo, é definida uma posição temporária a ser ocupada por ele, a fim de liberar o caminho do robô inferior (amarelo). Durante esse processo, o robô inferior está em manobra de espera, na qual ele fica parado até que o robô superior se distancie suficientemente e seu tempo de espera termine ou até perceber uma situação que precise realizar alguma manobra de recuo, o que não é o caso. No instante $t = 12$s (Figura 59(d)), o robô superior se distancia o suficiente para sair do alcance de detecção do inferior, possibilitando sua movimentação ao final do tempo de espera. No tempo $t = 15$s (Figura 59(e)), o robô inferior volta a se movimentar, enquanto o robô superior alcança a posição estabelecida para o recuo (objetivo temporário) e inicia sua manobra de espera. Na Figura 59(f) ($t = 24$s), o robô recuado conclui o seu tempo de espera e o deslocamento do outro robô foi suficiente para liberar a passagem que estava bloqueada e, ao mesmo tempo, sair da área de detecção (vizinhança). Assim, no instante seguinte ($t = 27$s), o robô superior retoma sua trajetória em direção ao objetivo original. Por fim, a Figura 59(h) ($t = 39$s) mostra que, após a manobra, os dois robôs conseguem seguir seus respectivos caminhos, podendo voltar a executar outras manobras em novas situações de conflito. Novamente, ressaltamos que as manobras são baseadas em informações e decisões locais tomadas por cada robô de forma autônoma, sem a existência de comunicação direta ou negociador externo.

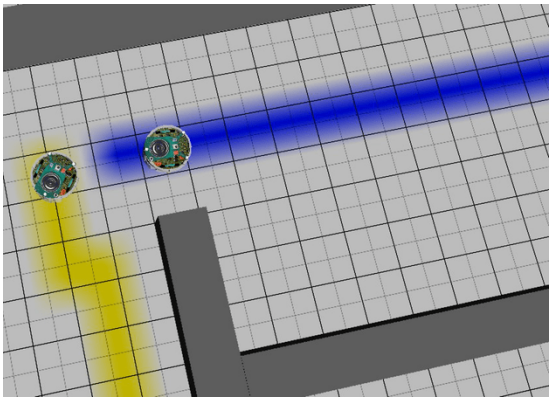
Embora não tenha ocorrido no exemplo da Figura 59, durante a manobra de recuo, os erros de odometria podem fazer com que, em alguns casos, o robô inferior se movimente em direção ao robô superior que está parado na posição de recuo, aguardando o fim do tempo de espera. Neste caso, ocorrerá uma nova detecção entre os robôs. Entretanto, como o robô inferior inicia seu deslocamento enquanto o robô superior ainda está se deslocando até a posição de recuo, a passagem que antes estava bloqueada é liberada. Assim, ao final do tempo de espera do robô superior, ele executa uma manobra de desvio para contornar o robô inferior, que estará em manobra de espera devido à nova detecção.



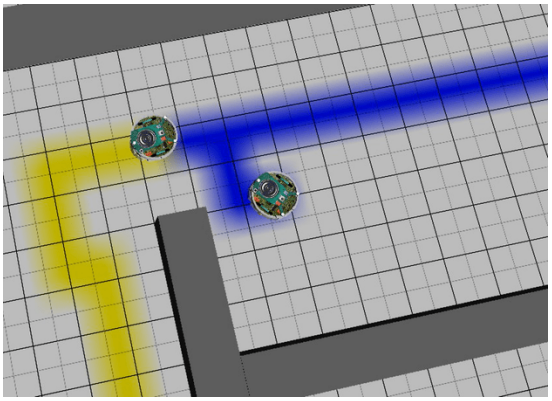
(a) $t = 0$s



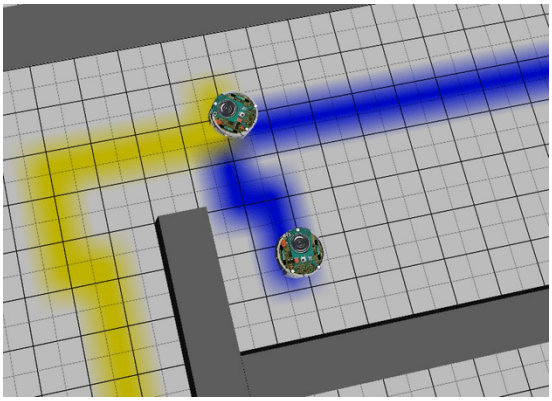
(b) $t = 6$s



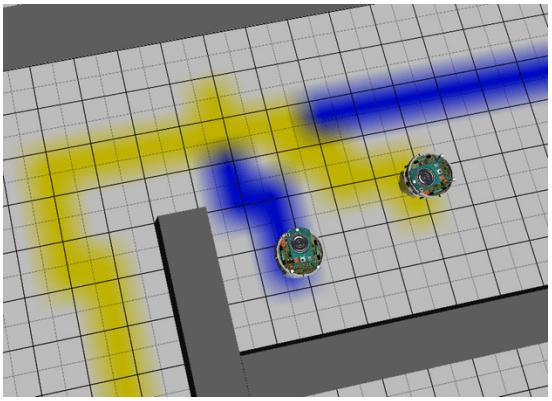
(c) $t = 9$s



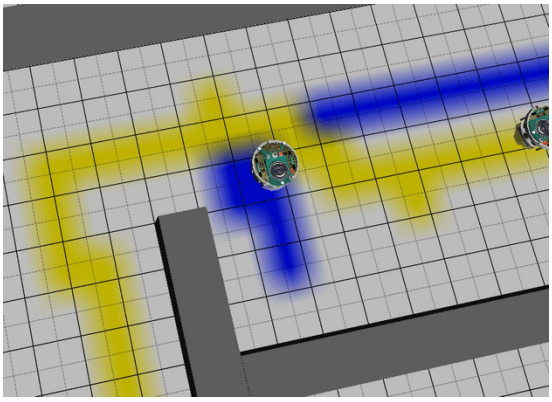
(d) $t = 12$s



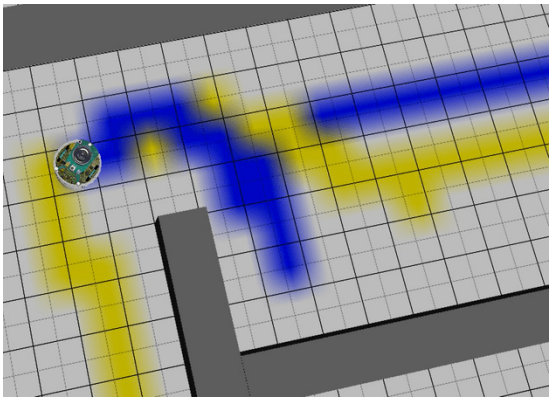
(e) $t = 15$s



(f) $t = 24$s



(g) $t = 27$s



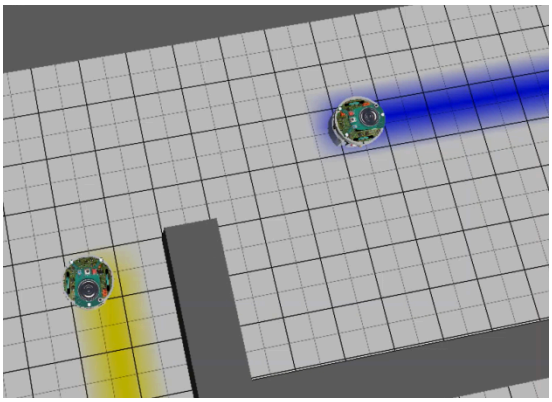
(h) $t = 39$s

Figura 59 – Evolução temporal de uma manobra de recuo realizada durante a navegação de 2 robôs pelo ambiente A3.

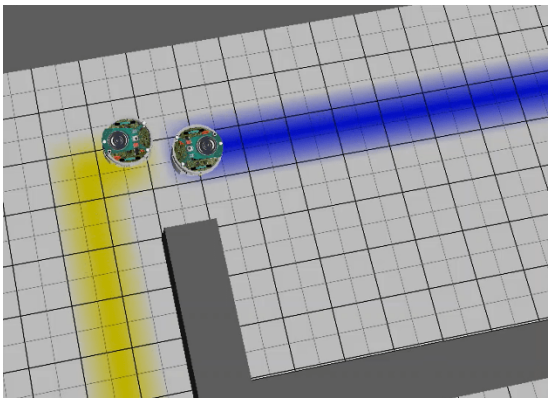
A Figura 60 ilustra esse comportamento. Nesse exemplo, assim como o exemplo da Figura 59 o robô com rastro amarelo tem menor prioridade e o outro (rastro azul) possui uma prioridade superior. Na Figura 60(a) (instante $t = 0$s), os robôs estão em movimento, pois a distância entre eles ainda não é suficiente para que sejam identificados entre si. Entretanto, no instante $t = 6$s (Figura 60(b)), os robôs já se aproximaram o suficiente para que possam se detectar e, assim, interromper sua navegação. Como não existe passagem para o robô superior (azul) até seu objetivo, ele utiliza da manobra de recuo para dar caminho para o robô que o bloqueou e consequentemente liberar sua passagem. Na Figura 60(c) ($t = 12$s), o robô inferior que estava utilizando a manobra de espera já voltou a movimentar e o robô superior alcança seu objetivo temporário, obtido pela manobra de recuo e, interrompe seu movimento para aguardar o fim do seu tempo de espera. No instante $t = 18$s (Figura 60(d)), por causa de erros de odometria, o robô inferior se movimenta em direção ao robô superior e novamente o detecta e, então, utiliza da manobra de espera, que interrompe seu movimento por mais 21 segundos (7 ciclos de 3 segundo). Na Figura 60(e) ($t = 24$s), quando o tempo de espera da manobra de recuo do robô superior termina, ele inicia seu movimento até seu objetivo, mas, ao rotacionar, seus sensores frontais detectam novamente o robô inferior. Contudo, nesse novo conflito, apesar do robô inferior não ter conseguido seguir seu caminho, ele se movimentou o suficiente para liberar uma passagem para o robô superior até seu objetivo e, então, nesse caso, o superior utiliza da manobra de desvio. No tempo $t = 28$s (Figura 59(f)) o robô superior já iniciou a manobra de desvio e no tempo $t = 40$s ((Figura 59(g))) ele supera o robô inferior e, eles já estão distantes o suficiente para não se detectarem. Por fim, a Figura 59(h) ($t = 50$s) mostra que, os dois robôs conseguem seguir seus respectivos caminhos, podendo executar outras manobras em novas situações de conflito.

Comparando-se as evoluções da Figuras 59 e 60, vemos que embora representem duas situações iniciais similares, a segunda demandou um tempo bem maior para que o conflito entre os dois robôs fosse superado. O tempo adicional na Figura 60 decorre do erro de odometria que fez com que os robôs envolvidos mudassem de manobra. Entretanto, é importante ressaltar que mesmo na ocorrência desse erro, os robôs conseguiram se adaptar, de forma autônoma, e resolveram o conflito apesar da nova aproximação inesperada.

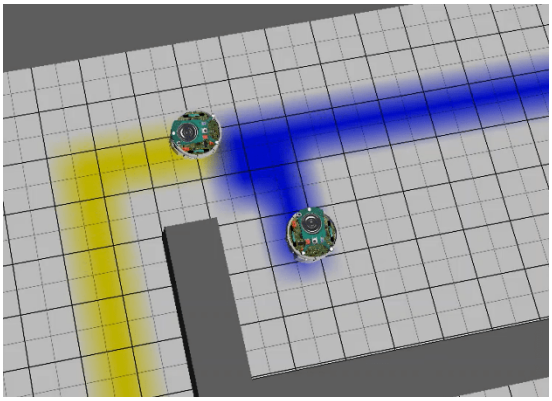
Analisando o comportamento das manobras em situações diferentes e em várias simulações do ambiente, foi possível observar que cada robô consegue superar seus conflitos de trajetória sem colisões e sem prejudicar significativamente a eficácia geral da sua navegação e do time. Os vídeos elaborados a partir desses experimentos podem ser acessados em (NAMETALA, 2021a).



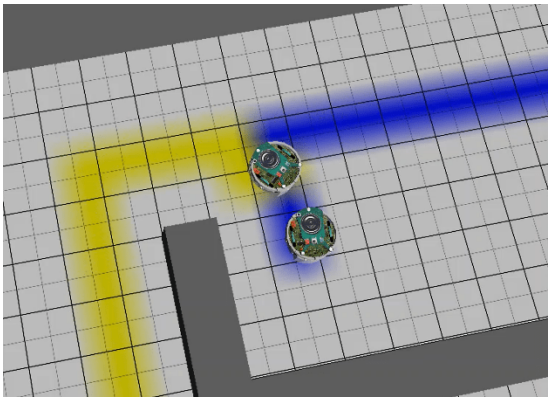
(a) $t = 0$s



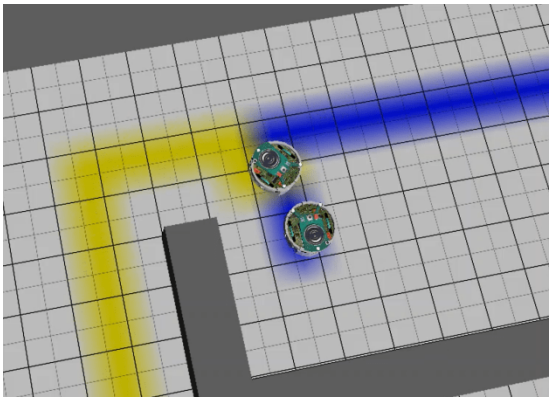
(b) $t = 6$s



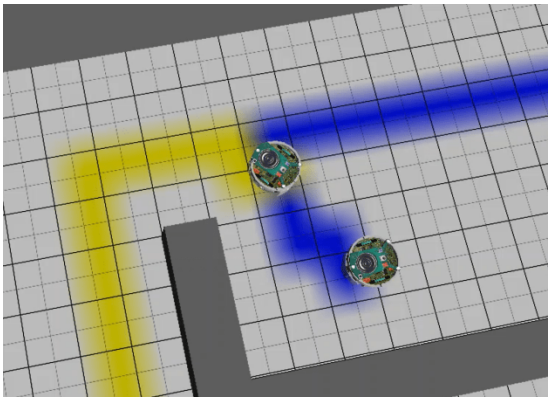
(c) $t = 12$s



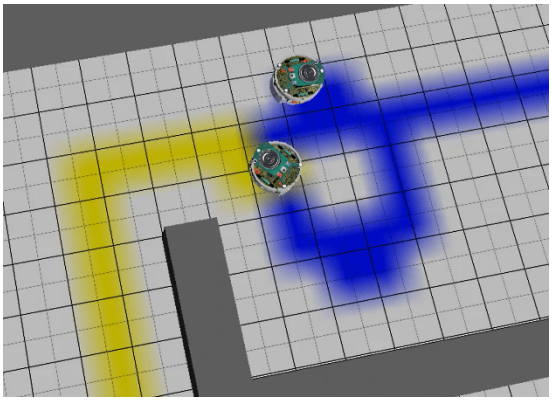
(d) $t = 18$s



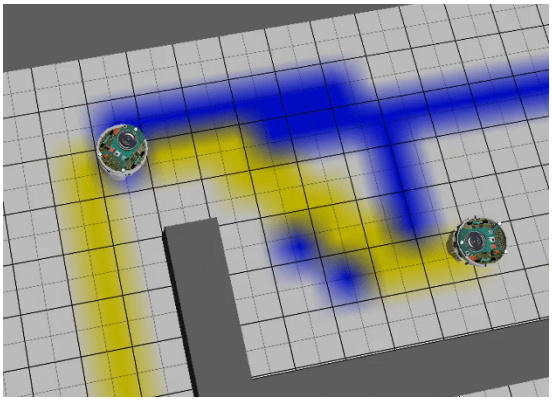
(e) $t = 24$s



(f) $t = 28$s



(g) $t = 40$s



(h) $t = 49$s

Figura 60 – Evolução temporal de uma situação entre 2 robôs que envolve mais de uma manobra para superar o conflito pelo ambiente A3.

5.3.2 Análise do comportamento do time e da escalabilidade do modelo

Após constatarmos que as manobras propostas conseguem superar os conflitos de trajetórias entre quatro robôs, passou-se a avaliar a dinâmica da navegação de uma maior quantidade de robôs pelo ambiente. Isso foi feito com o objetivo de encontrar situações de conflito que envolvam mais robôs e observar se, nesses casos, eles conseguem resolvê-las, de modo que o modelo consiga manter a navegabilidade do time mesmo com uma grande quantidade de robôs no ambiente, sem comprometer a eficácia de qualquer membro e do próprio time.

Com o objetivo de analisar a robustez do modelo, simulou-se a navegação de um time com 10 robôs pelo ambiente A3, onde a posição inicial e a célula objetivo de cada robô foram distribuídas de forma dispersa, na tentativa de criar várias situações de conflito. Durante esse experimento, foram utilizados 5 cenários diferentes, cada qual com uma configuração diferente, ou seja, as posições iniciais e objetivo de cada robô distribuídas de forma diferente, buscando uma maior variedade nas situações de conflito. Além do aumento no tamanho do time, os robôs passaram a se movimentar continuamente durante toda a simulação, com o objetivo de provocar mais situações de conflito. Essa alteração foi implementada de modo que, quando um robô alcança seu objetivo, ele passa a adotar a posição onde iniciou a simulação (posição inicial) como seu próximo objetivo. Dessa forma ele fica em um movimento de ida e volta em sua trajetória, até que a simulação seja finalizada. Vale destacar que essa estratégia foi escolhida devido à facilidade de implementação no ambiente de simulação. Entretanto, como já discutido anteriormente, a definição dos objetivos é ortogonal ao modelo de planejamento e navegação proposto, podendo ser adotadas outras abordagens, uma vez que o robô recebe seu objetivo a partir da imagem enviada pelo módulo CPI no início de uma nova navegação. Portanto, diferentes objetivos poderiam ser definidos, por exemplo, a partir de uma lista pré-definida ou da interação com o ambiente. O final da simulação ocorre quando todos os robôs realizam pelo menos um movimento completo de ida e volta pelo ambiente. Nos cenários testados, todas as simulações executadas demandaram entre 30 e 37 minutos. A Figura 61 ilustra o comportamento do time de robôs durante a navegação em um dos cenários executados. Ela registra o percurso de todo o time, adotando uma cor diferente para cada robô. Pode-se notar a existência de regiões com alto fluxo de robôs, proporcionando várias situações de conflito, como esperado.

Pelas simulações, foi possível observar que o modelo distribuído BioTeam foi capaz de manter a navegabilidade de todos os membros do time, mesmo com vários robôs no mesmo ambiente, conseguindo resolver todas as situações de conflito geradas. Inclusive, cabe ressaltar que o BioTeam foi capaz de resolver conflitos envolvendo mais de dois robôs e que exigem a realização de diferentes manobras, ou seja, cada robô consegue executar suas funções corretamente e de forma descentralizada.

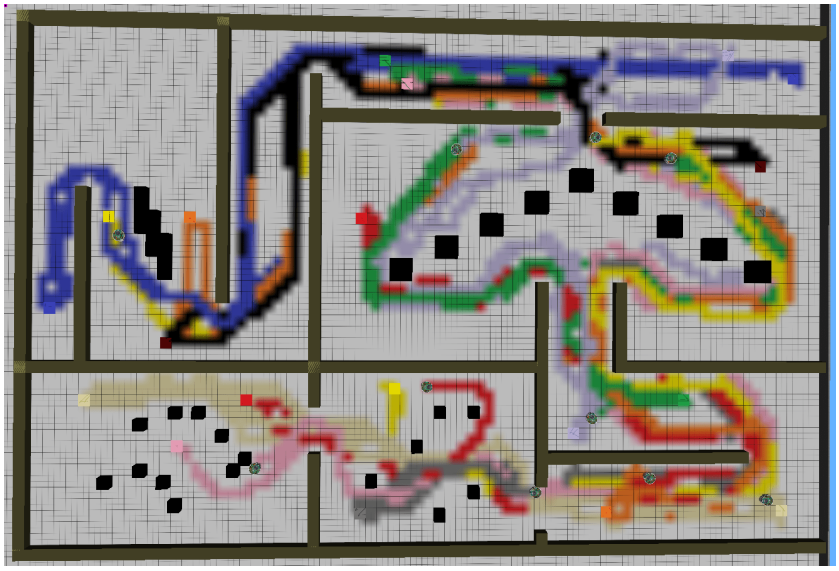
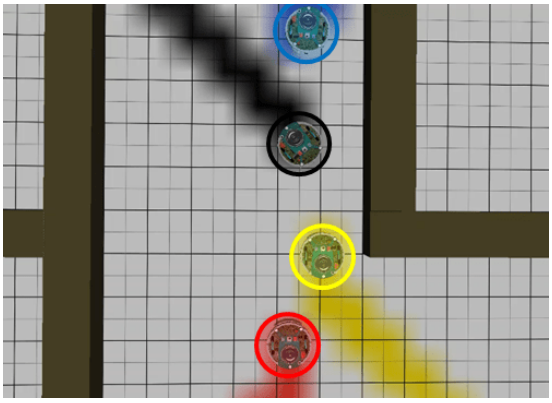
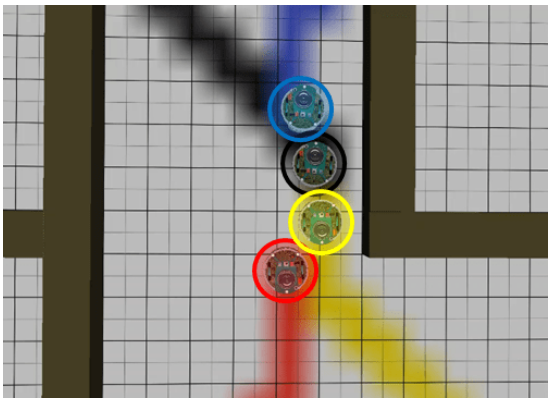


Figura 61 – Navegação do time com 10 robôs em um dos cenários do abmiente A3.

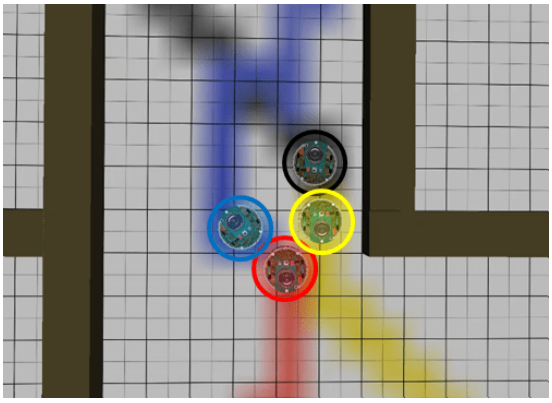
A Figura 62 ilustra uma situação de conflito de trajetória que envolve 4 robôs. Nesse exemplo a ordem de prioridade do menor para o maior é: (*i*) robô de rastro preto; (*ii*) vermelho; (*iii*) amarelo; e (*iv*) azul. Na Figura 62(a) (instante $t = 0$s), os robôs estão em movimento, pois a distância entre eles ainda não é suficiente para que sejam identificados como um obstáculo na vizinhança. Entretanto, no instante $t = 6$s (Figura 62(b)), os robôs já se aproximaram o suficiente para que possam se detectar e, assim, interromper sua navegação. Nessa situação, o robô azul tem prioridade superior aos demais e, portanto, como existe um caminho livre até seu objetivo ele vai usar a manobra de desvio. Para os outros três robôs, o amarelo tem prioridade superior entre eles, porém, apesar de visualmente existir uma passagem entre eles, no mapa interno do robô amarelo, ele está cercado pela área alargada do robô preto e vermelho, que é a margem de segurança para evitar possíveis colisões, ou seja, nesse caso o robô amarelo não consegue utilizar a manobra de desvio nem de recuo. Portanto, o robô amarelo vai ficar parado verificando sua vizinhança no final de seu tempo de espera, até que algum caminho seja liberado, para que ele possa utilizar a manobra de desvio ou de recuo. Por outro lado, os robôs vermelho e preto, que possuem prioridade inferior, vão utilizar a manobra de espera. Porém, nessa situação, como o robô superior (amarelo) não consegue agir, eles vão ficar parados até que o tempo de espera de cada um chegue ao seu limite (7 ciclos de 3 segundos) e, no final, agir utilizando uma manobra diferente. Portanto, no instante $t = 18$s (Figura 62(c)) enquanto esse limite de tempo não é alcançado, o robô azul que utilizou a manobra de desvio, novamente detecta outro robô e interrompe seu movimento. Como sua prioridade é superior aos demais, ele se movimenta utilizando a manobra de desvio mais uma vez e supera o conflito no tempo $t = 27$s (Figura 62(d)). Na Figura 62(e) ($t = 30$s), o tempo de espera dos robôs vermelho e preto chegam ao limite e os dois se movimentam utilizando a manobra de desvio, enquanto o robô amarelo aguarda em espera. No tempo $t = 36$s



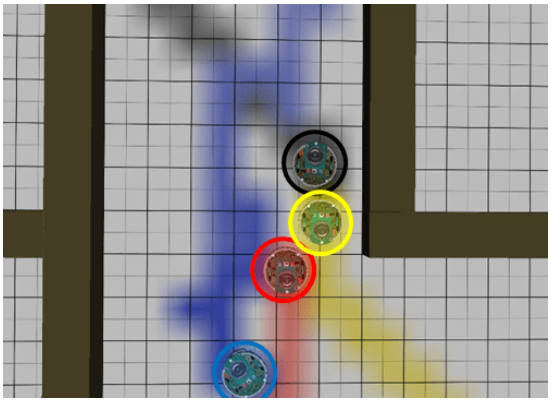
(a) $t = 0$s



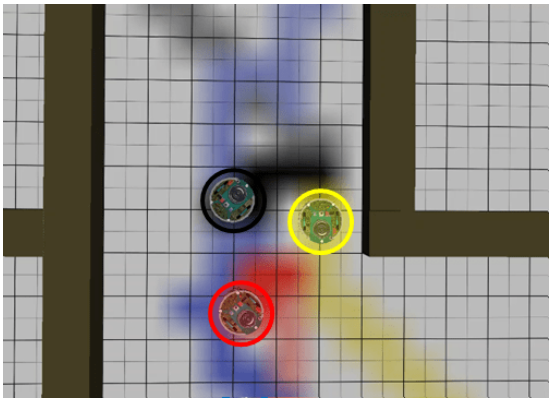
(b) $t = 6$s



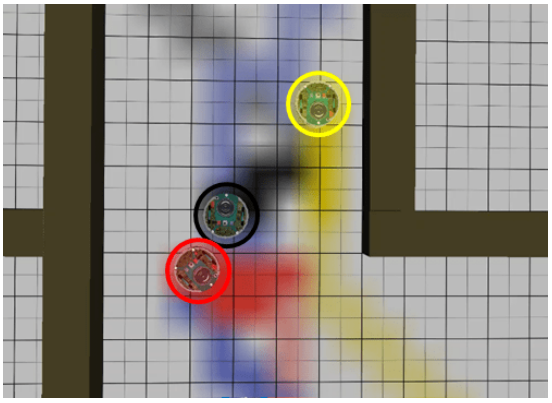
(c) $t = 18$s



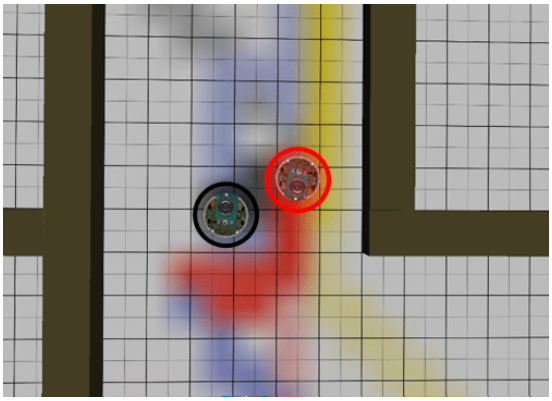
(d) $t = 27$s



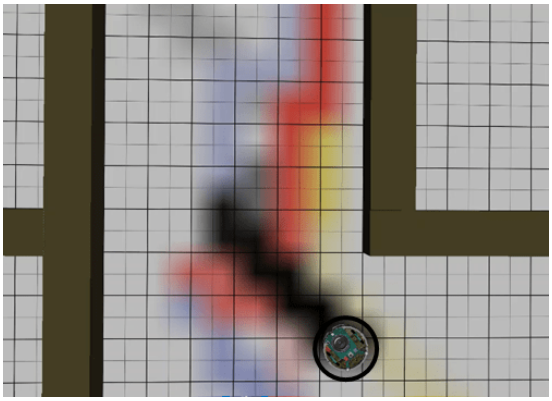
(e) $t = 30$s



(f) $t = 36$s



(g) $t = 45$s



(h) $t = 57$s

Figura 62 – Evolução temporal da resolução de um conflito envolvendo 4 robôs pelo ambiente A3.

(Figura 62(f)), o robô amarelo não identifica mais robôs em sua vizinhança e consegue seguir seu caminho, sem a necessidade de utilizar outra manobra. Entretanto, os robôs vermelho e preto interrompem seus movimentos após se detectarem. No tempo $t = 45$s (Figura 62(g)), o robô vermelho consegue superar o preto, após utilizar a manobra de desvio, uma vez que ele tem a maior prioridade dentre os dois. Por fim, na Figura 59(h) ($t = 57$s) o conflito envolvendo os 4 robôs foi resolvido e o robô preto segue seu caminho até seu objetivo.

Com base nos resultados obtidos no experimento anterior, novas simulações foram realizadas a fim de avaliar a escalabilidade do modelo. Esse experimento foi realizado considerando o ambiente A1, por ser um ambiente mais amplo que poderia comportar a navegação simultânea de um número maior de robôs, com uma certa folga, de forma a permitir a observação dos conflitos. Nesse ambiente, foram avaliados cinco diferentes cenários, sendo que a única diferença entre eles reside na quantidade de robôs que compõem o time. O primeiro é formado por 10 robôs e acrescenta-se mais 10 em cada cenário, até totalizar 50 robôs no mesmo ambiente.

A Figura 63 ilustra em detalhes a navegação do time de robôs no cenário simulado com 50 robôs no ambiente A1, enquanto a Figura 64 ilustra os demais cenários com 10, 20, 30 e 40 robôs. Devido à quantidade elevada de robôs, seus rastros são apagados conforme se movimentam pelo ambiente para facilitar a visualização e todos possuem a mesma cor (azul). Os vídeos com as simulações realizadas dos experimentos estão disponíveis em (NAMETALA, 2021a).

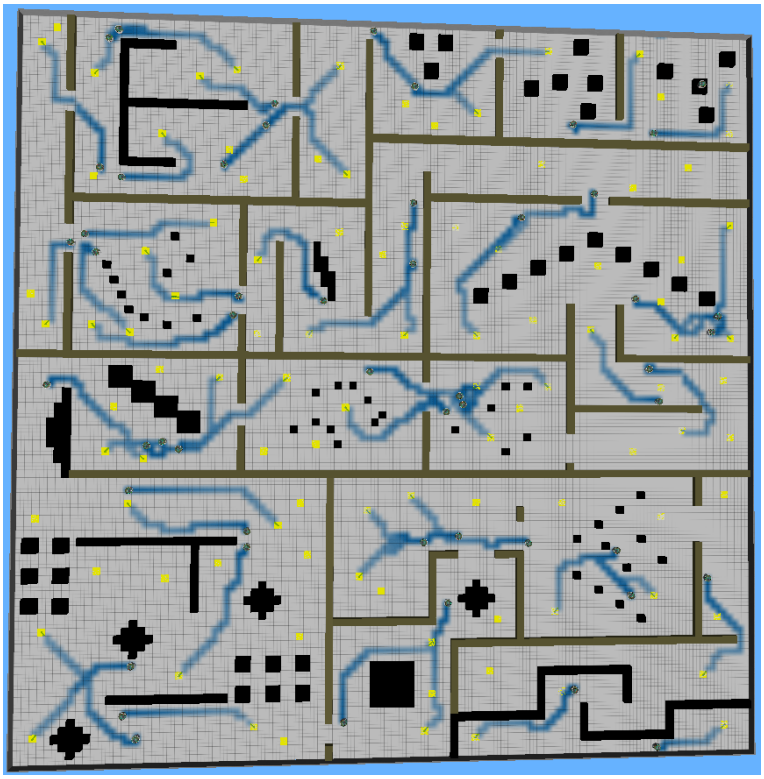
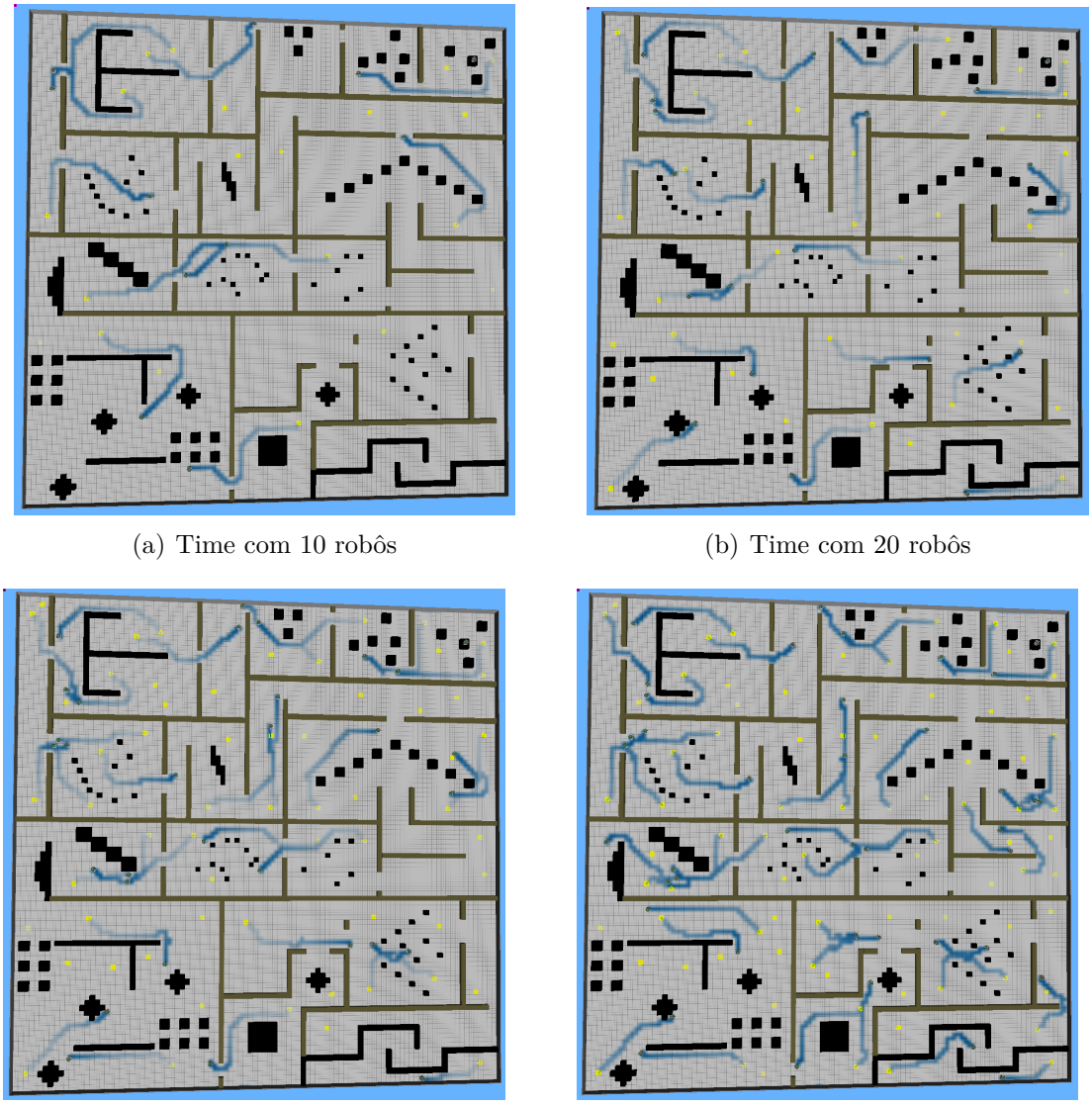


Figura 63 – Navegação do time com 50 robôs pelo amiente A1.



(a) Time com 10 robôs

(b) Time com 20 robôs

(c) Time com 30 robôs

(d) Time com 40 robôs

Figura 64 – Navegação do time com 10, 20, 30 e 40 robôs pelo amiente A1.

Algumas métricas foram computadas para possibilitar a análise da performance dos times. Para tal, foram realizadas três execuções de 30 minutos, para cada tamanho de time (10, 20, 30, 40 e 50 robôs). Para cada execução, as métricas foram computadas em dois momentos da execução: aos 15 minutos, na metade do tempo total, e aos 30 minutos, ao final da execução. A Tabela 4 apresenta os resultados médios obtidos para cada robô do time, considerando-se as 3 execuções no ambiente A1, para cada instante da execução: 15 e 30 minutos. As métricas médias computadas nesses experimentos para cada robô do time são apresentadas na tabela: distância percorrida (aproximada), número de passos, número de rotações, quantidade de objetivos alcançados, quantidade de manobras de recuo, quantidade de manobras de desvio, quantidade de manobras de espera e, finalmente, quantidade média de manobras executadas, considerando-se a soma das manobras recuo, desvio e espera.

Tempo de Execução	**15 minutos**					**30 minutos**				
Quantidade de robôs	**10**	**20**	**30**	**40**	**50**	**10**	**20**	**30**	**40**	**50**
Distância percorrida (cm)	1601	1526	1444	1405	1355	2882	2938	2753	2625	2471
Rotações	70,47	64,75	68,43	67,74	70,86	128,90	132,68	140,10	134,36	140,30
Passos	195,47	186,02	173,94	168,64	161,27	346,70	356,07	329,86	310,83	288,79
Objetivos	1,27	1,48	1,77	1,79	1,55	2,50	3,20	3,61	3,51	2,77
Manobras de Recuo	0,13	0,72	1,62	1,63	5,20	0,33	1,48	2,40	2,87	8,11
Manobras de Desvio	15,13	16,43	18,19	17,71	18,86	29,33	33,53	39,00	38,10	40,90
Manobras de Espera	0,20	0,72	1,92	2,58	3,95	0,37	1,42	3,13	4,88	7,35
Manobras (Total)	15,47	18,10	22,56	23,93	29,92	34,13	37,80	47,04	48,44	62,22

Tabela 4 – Valor médio de cada robô do time, obtidos através das 3 execuções em A1.

É possível perceber que com o aumento do número de robôs do time, existe uma queda na performance individual dos robôs, ou seja, eles percorrem uma distância menor, através de um número menor de passos. Em relação às rotações, embora existam algumas oscilações, alterando-se o time de 10 a 50 robôs, não ocorreu uma mudança tão significativa. Essa queda individual ocorre devido ao fato do ambiente ser mantido fixo e um número maior de robôs precisa disputar o mesmo espaço livre, levando a um número maior de conflitos e à necessidade de mais manobras para resolverem essas situações. Em relação ao número de objetivos alcançados em média por cada agente do time, existe uma melhoria de performance quando o tamanho do time é ampliado de 10 a 30 robôs, permanece estável até 40 robôs, mas sofre uma queda para 50 robôs. Entretanto, ao verificar os valores absolutos, multiplicando-se o número de objetivos pelo tamanho do time, pode-se ver que o time como um todo é capaz de alcançar os objetivos um número maior de vezes. Ou seja, por mais que o desempenho individual possa cair, o aumento do número de membros do time consegue efetivamente lidar com situações críticas e aumentar a quantidade total de trabalho realizado. Assim, foi possível concluir que as manobras propostas para o BioTeam conseguem resolver os conflitos, sem exigir uma grande sobrecarga individual.

Considerando-se as manobras, pode-se observar que o desvio é a estratégia mais acionada. Isso acontece porque enquanto as manobras de espera e recuo são acionadas apenas em situações envolvendo outros robôs na vizinhança, a manobra de desvio também é utilizada para impedir que o robô colida com paredes e obstáculos internos. Essa situação de proximidade com obstáculos fixos ocorre devido aos erros de navegação em relação à rota planejada. Dessa forma, a utilização dos sensores e recálculo de rotas de desvio impedem que o robô colida por erros de odometria e torna a manobra de desvio a mais utilizada.

Portanto, foi considerado que o modelo distribuído BioTeam provê um comportamento adequado do time, permitindo a resolução de eventuais conflitos de navegação e mantendo uma boa performance coletiva quando o número de robôs é elevado de 10 a 40, com uma pequena redução de performance com 50 robôs. Essa queda final pode nos indicar que, para o tamanho do ambiente A1, o número de robôs ideal para a navegação seria em torno de 40 robôs.

Entretanto, comparando as métricas quando o tempo de execução foi aumentado de 15 para 30 minutos, pode-se perceber uma queda de desempenho coletivo a partir de 40

robôs e, principalmente, para os times com 50 agentes. Assim, após uma análise cuidadosa dos vídeos das execuções, foi possível identificar algumas situações em que alguns robôs do time deixavam de contribuir efetivamente para a execução da tarefa, diminuindo a performance do time como um todo. A próxima seção descreve essas limitações e possíveis soluções a serem desenvolvidas no futuro.

5.3.3 Limitações do modelo proposto

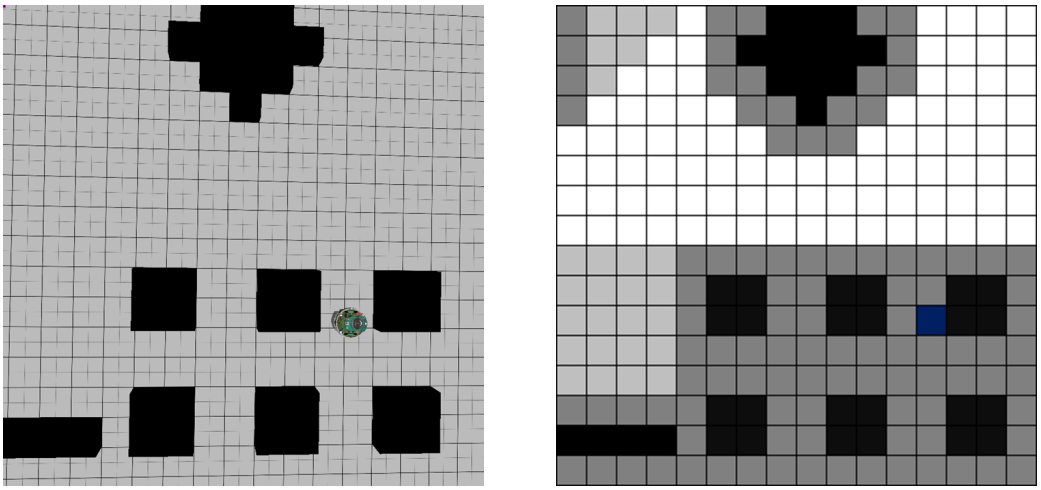
De forma geral, as simulações permitiram observar que os robôs também conseguem alcançar seus objetivos, evitando obstáculos e conflitos de trajetória. Porém, conforme o tempo de execução e o número de robôs aumentam, a quantidade de manobras executadas por eles também aumenta consideravelmente. Esse é um comportamento natural, uma vez que o ambiente de navegação é o mesmo e, portanto, quanto mais robôs no ambiente, mais conflitos de trajetórias irão acontecer, que são superados com o uso de diferentes manobras. Adicionalmente, quanto mais manobras são executadas, maior o número de rotações realizadas pelo robô, e esse comportamento tende a elevar os erros de odometria relacionados com o ângulo do robô. Na grande maioria das execuções, mesmo com a redução na precisão da odometria devido aos erros gerados pelas rotações extras, a eficácia do modelo ainda é mantida. Contudo, quando o acúmulo de erro é muito elevado, o robô pode sair de sua rota e não conseguir alcançar seu objetivo.

Identificamos que duas situações diferentes podem fazer com que o robô não alcance seu objetivo.

A primeira delas é quando o robô "perde a sua orientação". Isso acontece porque a informação referente ao ângulo está excessivamente incorreta e nesse caso o robô passa a ter um comportamento fora do padrão. Por exemplo, quando o robô percorre toda a rota planejada e, ao verificar se de fato está posicionado na célula *objetivo*, através de uma requisição de imagem, ele identifica que está em uma posição diferente. Nesse caso, é feito um novo cálculo de rota da sua posição atual até o objetivo. Porém, quando a informação referente ao ângulo é muito diferente da orientação real do robô, ele fica "desorientado"e pode nunca alcançar a célula objetivo e, assim, fica indefinidamente se movimentando em torno dela.

A segunda situação que impede o robô de alcançar seu objetivo é quando ele se movimenta até uma região alargada entre dois ou mais obstáculos, incluindo paredes. Nesse caso, o robô irá detectar um dos obstáculos e o módulo de navegação irá solicitar uma nova rota para superá-lo. Entretanto, o módulo PC não será capaz de calcular essa rota. Isso ocorre porque a posição atual do robô não pode ser alcançada pelo método de propagação de distância, uma vez que não existem células livres em torno do robô. Nesse momento, ele está cercado por células de estado *obstáculo*, *obstáculo_virtual*, *parede* ou *parede_virtual*. Assim, o robô continuará parado e nunca alcançará seu objetivo. Essa situação é demonstrada na Figura 65, onde o robô (azul) está cercado por células no es-

tado *obstáculo_virtual* (cinza escuro) e obstáculo (preto), as quais impedem que a difusão da distância alcance suas células adjacentes (vizinhança). Em (OLIVEIRA; VARGAS; FERREIRA, 2015a), casos semelhantes foram apresentados.



(a) Parte do ambiente de simulação A1 (b) Parte do reticulado celular A1

Figura 65 – Exemplo de situação onde o robô fica preso entre obstáculos.

A Tabela 5 apresenta o número médio de vezes que essas situações indesejadas foram identificadas nas simulações realizadas. É possível perceber que, embora sejam menos prováveis nos experimentos com poucos robôs e com um tempo menor de execução, à medida que aumentamos tanto o tempo quanto o tamanho do time, essas situações se tornam mais prováveis, justificando a queda de performance observada na Tabela 4. Entretanto, apesar dos problemas relatados, como o modelo é descentralizado, mesmo quando um dos robôs do time falha, o sistema não é interrompido, uma vez que os demais robôs continuam a se movimentar pelo ambiente até seus respectivos objetivos. Portanto, é possível concluir que o modelo tem um comportamento aceitável, mesmo em um ambiente mais complexo onde vários robôs trafegam.

O problema do cálculo da rota a partir de áreas sombreadas de obstáculos foi discutido em (OLIVEIRA; VARGAS; FERREIRA, 2015a), onde uma técnica que trata situações semelhantes foi apresentada. Assim, pretende-se incorporar essa técnica no modelo descentralizado aqui proposto, em trabalhos futuros. Apesar do BioTeam ainda não ser considerado um modelo de enxame, acreditamos que ao corrigir os problemas relatados, essa característica pode ser alcançada e será possível executar simulações com um número ainda maior de robôs, e até mesmo superar a barreira de 100 robôs.

Tempo de Execução	**15 minutos**					**30 minutos**				
Quantidade de robôs	**10**	**20**	**30**	**40**	**50**	**10**	**20**	**30**	**40**	**50**
Número de robôs desorientados	0,00	0,00	0,67	0,67	0,00	0,00	0,67	2,67	4,00	16,67
Número de robôs presos	0,00	0,33	0,67	1,00	1,33	1,00	0,67	1,67	4,00	6,33

Tabela 5 – Número médio de vezes que os robôs perderam sua orientação ou ficaram presos entre obstáculos.

Capítulo 6

Conclusão

Este trabalho propôs um novo modelo de planejamento de caminhos e navegação distribuída e descentralizada de um time de robôs. O modelo foi denominado BioTeam e foi aplicado na tarefa de navegação por ambientes previamente conhecidos. Esse tipo de sistema pode ser útil em centros de distribuição e logística, em vigilância patrimonial e de perímetro, entre outras aplicações.

Nessas aplicações é fundamental que o algoritmo crie um caminho da posição inicial do robô até sua meta, evitando qualquer obstáculo no ambiente. Dessa forma, um novo modelo de planejamento de caminho baseado em regras de autômatos celulares e difusão a distância foi proposto. Este modelo foi baseado nos modelos discutidos em (OLIVEIRA; VARGAS; FERREIRA, 2015a) e (MARTINS et al., 2018). As principais modificações referem-se ao cálculo da rota e podem ser resumidas por: (i) diferenciar os movimentos diagonais e cardinais, penalizando o primeiro tipo no cálculo da rota; (ii) implementar uma nova etapa de regras do autômato celular que visa evitar as regiões côncavas resultantes de obstáculos; e (iii) diferenciar paredes e obstáculos internos. A primeira modificação provoca a geração de rotas que evitam deslocamentos diagonais, os quais costumam elevar o custo da rota final. A segunda faz com que os robôs executem rotas mais suaves ao alargar regiões côncavas, tornando obstáculos independentes em uma região compacta, que será evitada. A terceira modificação é necessária para aplicar algumas regras de alargamento apenas em obstáculos internos, pois nos modelos anteriores não há diferença entre eles e as paredes.

Experimentos realizados para avaliar a rota planejada em ambientes com apenas um robô mostraram que o uso do nosso método retorna rotas mais curtas e com trajetórias suaves, ou seja, os robôs fazem menos rotações durante seu percurso, quando comparado com as rotas originais planejadas usando o modelo apresentado em (MARTINS et al., 2018). Além disso, simulações de Webots mostraram que o modelo proposto é capaz de calcular rotas que podem ser percorridas com sucesso e sem colisões usando um robô e-puck e pode ser uma abordagem eficiente para o planejamento e navegação de robôs.

Além do módulo de planejamento de caminho, nesta pesquisa também foi desenvolvido

um modelo de navegação distribuída e descentralizada para um time de robôs em um ambiente previamente conhecido e parcialmente observável. Este modelo foi simulado em um ambiente com vários robôs, com o objetivo de capacitar cada agente a tomar suas próprias decisões baseadas apenas em informações locais (vizinhança), coletadas a partir de sensores e de imagens do ambiente, sem a necessidade de comunicação com qualquer membro do time. Essas decisões locais são referentes a situações de possível colisão e principalmente conflitos de trajetória envolvendo um ou mais robôs do time. Em caso de conflito de trajetória, cada robô consegue decidir qual manobra deve ser aplicada para superá-lo, sem modificar significativamente a sua trajetória ou a de qualquer outro membro do time.

Simulações com o modelo BioTeam na navegação de times de robôs apontam que, no geral, os agentes conseguem superar diferentes situações de conflito de trajetória, sem prejudicar a sua eficiência ou do restante do time e de forma totalmente descentralizada, considerando-se o comportamento do time, ou seja, sem um robô controlador e sem qualquer comunicação com os outros robôs. Esse comportamento foi observado em dois ambientes complexos e com uma grande quantidade de robôs (de 10 a 50 agentes). Dessa forma, conclui-se que o objetivo principal do trabalho foi atingido, uma vez que foi desenvolvido um modelo de navegação distribuída e descentralizada para um time de robôs simples. Entretanto, vale destacar que, embora o planejamento e a navegação dos robôs sejam totalmente descentralizados, o funcionamento atual do modelo depende do envio de imagens do ambiente para os robôs do time, o qual é feito pelo módulo PCI. O uso de um único componente para geração dessas imagens limita a característica de descentralização do sistema. Uma evolução natural deste trabalho é o emprego de múltiplas unidades de processamento de imagens para prover a redundância necessária. Além da escalabilidade, robustez e flexibilidade, o modelo mostrou-se capaz de atender outras características importantes para a navegação inteligente de enxame, a saber: (*i*) autonomia dos robôs, onde cada um realiza as suas próprias tomadas de decisão; (*ii*) decisões baseadas em informações locais; (*iii*) cooperação entre os membros do time para atingir um objetivo comum; e (*iv*) inexistência de um controle centralizado e global.

6.1 Produções científicas

Os resultados apresentados na Seção 5.1 e 5.2, relacionados com a técnica de planejamento de caminho envolvendo apenas um robô, possibilitaram a elaboração, submissão e publicação de um artigo (NAMETALA; MARTINS; OLIVEIRA, 2020) no *IEEE Congress on Evolutionary Computation* (CEC). Os resultados da Seção 5.3, relacionados ao sistema de navegação descentralizado e distribuído, estão sendo utilizados na elaboração de um novo artigo.

6.2 Trabalhos Futuros

Durante o desenvolvimento do modelo, identificou-se diferentes pontos que necessitam de uma investigação mais aprofundada, mas que não puderam ser devidamente tratados nesta pesquisa, devido à restrição de tempo para sua conclusão. Dentre eles, se destacam: (*i*) realizar uma análise comparativa com outras abordagens de planejamento de caminho, como os algoritmos A*e D*-lite (SETIAWAN et al., 2014); (*ii*) realizar uma análise comparativa com modelos de navegação distribuída que utilizam técnicas convencionais; (*iii*) Integrar a técnica apresentada em (OLIVEIRA; VARGAS; FERREIRA, 2015a) para resolver o problema relacionado com robôs presos em regiões alargadas por obstáculos; (*iv*) elaborar uma estratégia para corrigir os erros de odometria relacionados com o ângulo do robô, utilizando o sensor giroscópio presente no robô e-puck; (*v*) aplicar um algoritmo genético para ajustar as variáveis que envolvem tempo de espera nas manobras para resolução de conflito; (*vi*) atualizar o fluxo de decisão local do módulo de navegação, para possibilitar a navegação do robô em um ambiente onde os obstáculos também serão dinâmicos e não apenas os robôs; (*vii*) desenvolver novos métodos de classificação de robôs, podendo envolver diferentes níveis de classes de identificadores de prioridade entre eles, ou seja, além de uma classe de prioridade com identificadores estáticos presentes em cada robô (como apresentado neste trabalho), pode-se incluir novas classes com identificadores dinâmicos, que se ajustam conforme um fator definido, como a importância de uma tarefa executada ou a distância entre o robô até seu objetivo. Por exemplo, em um sistema de envio e entrega de correspondências, onde os robôs seriam os veículos autônomos responsáveis pela locomoção, em caso de conflito de trajetória, aquele com uma correspondência considerada urgente teria prioridade superior aos demais. Porém, se outro robô também estivesse transportando um item urgente, dentre eles, o mais distante do seu ponto de entrega teria maior prioridade. Contudo, em último caso, se a distância entre eles fosse a mesma, aquele com maior identificador fixo teria prioridade de movimento; (*viii*) implementar e simular outros ambientes complexos, capazes de estressar o modelo investigado, os quais também podem ser utilizado por outros trabalhos; (*ix*) aumentar a quantidade de robôs nos experimentos futuros até que a dinâmica de exame de robôs seja alcançada; (*x*) integrar o modelo ao módulo de captura e processamento de imagem desenvolvido em (MARTINS et al., 2018) a fim de viabilizar a realização de experimentos com robôs reais; (*xi*) melhorar a eficiência do processo de requisição de imagem, possibilitando a recepção de apenas uma parte da imagem e não a imagem completa; (*xii*) aplicar e analisar diferentes métodos de busca inteligentes na escolha do objetivo temporário na vizinhança do robô, a fim de melhorar a eficiência dessa tarefa; (*xiii*) avaliar a possibilidade de utilizar as áreas de sombreamento por concavidade em rotas de fuga dos conflitos entre robôs; e (*xiv*) adaptar o modelo para o uso no controle autônomo de um time em partidas de futebol de robô.

Referências

AKBARIMAJD, A.; HASSANZADEH, A. A novel cellular automata based real time path planning method for mobile robots. **J. of Engineering Research and Applications**, v. 1, n. 4, p. 1262–1267, 2011. Disponível em: <http://www.ijera.com/papers/Vol%201%20issue%204/C01412621267.pdf>.

_____. Autonomously implemented versatile path planning for mobile robots based on cellular automata and ant colony. **international journal of computational intelligence systems**, Taylor & Francis, v. 5, n. 1, p. 39–52, 2012. Disponível em: <https://doi.org/10.1080/18756891.2012.670520>.

AKBARIMAJD, A.; LUCAS, C. A new architecture to execute cas-based path-planning algorithm in mobile robots. In: IEEE. **2006 IEEE International Conference on Mechatronics**. 2006. p. 478–482. Disponível em: <https://doi.org/10.1109/ICMECH.2006.252574>.

ALVES, R. M. F. et al. Coordenação, localização e navegação para robôs de serviço em ambientes internos. Universidade Federal de Uberlândia, 2017. Disponível em: <http://dx.doi.org/10.14393/ufu.te.2017.21>.

ARKIN, R. C. Behavior-based robotics. **MIT press**, 1998.

BALCH, T.; ARKIN, R. C. Behavior-based formation control for multirobot teams. **IEEE transactions on robotics and automation**, IEEE, v. 14, n. 6, p. 926–939, 1998. Disponível em: <https://doi.org/10.1109/70.736776>.

BARRAQUAND, J.; KAVRAKI, L.; LATOMBE, J.-C.; MOTWANI, R.; LI, T.-Y.; RAGHAVAN, P. A random sampling scheme for path planning. **The International Journal of Robotics Research**, Sage Publications Sage CA: Thousand Oaks, CA, v. 16, n. 6, p. 759–774, 1997. Disponível em: <https://doi.org/10.1177/027836499701600604>.

BARRAQUAND, J.; LANGLOIS, B.; LATOMBE, J.-C. Numerical potential field techniques for robot path planning. **IEEE transactions on systems, man, and cybernetics**, IEEE, v. 22, n. 2, p. 224–241, 1992. Disponível em: <https://doi.org/10.1109/21.148426>.

BEHRING, C.; BRACHO, M.; CASTRO, M.; MORENO, J. An algorithm for robot path planning with cellular automata. In: **Theory and practical issues on cellular automata**. Springer, 2001. p. 11–19. Disponível em: <https://doi.org/10.1007/978-1-4471-0709-5_2>.

BORENSTEIN, J.; FENG, L. **UMBmark: A method for measuring, comparing, and correcting dead-reckoning errors in mobile robots**. [S.l.], 1994. Disponível em: <https://deepblue.lib.umich.edu/bitstream/handle/2027.42/3753/bac6477.0001.001.pdf?sequence=5>.

CÂNDIDO, R. da P.; OLIVEIRA, G. M. B. de; MARTINS, L. G. A. Refinamento de modelos de navegação de robôs autônomos através da calibração do sistema de odometria. **Revista Eletrônica de Iniciação Científica em Computação**, v. 16, n. 4, 2018. Disponível em: <https://doi.org/10.5753/reic.2018.1068>.

CAPI, G.; MOHAMED, Z. Multiple robots formation–a multiobjctive evolution approach. **Procedia Engineering**, Elsevier, v. 41, p. 156–162, 2012. Disponível em: <https://doi.org/10.1016/j.proeng.2012.07.156>.

CYBERBOTICS. **Webots User Guide R2019a - Tutorial 4: More about Controllers (20 Minutes)**. 2019. Disponível em: <https://www.cyberbotics.com/doc/guide/tutorial-4-more-about-controllers-20-minutes?version=R2019a>.

_____. **Webots simulator**. 2021. Disponível em: <https://www.cyberbotics.com>.

_____. **Webots User Guide master - GCTronic' e-puck**. 2021. Disponível em: <https://www.cyberbotics.com/doc/guide/epuck?version=master>.

DASGUPTA, P.; WHIPPLE, T.; CHENG, K. Effects of multi-robot team formations on distributed area coverage. **International Journal of Swarm Intelligence Research (IJSIR)**, IGI Global, v. 2, n. 1, p. 44–69, 2011. Disponível em: <https://doi.org/10.4018/jsir.2011010103>.

EPFL, P. F. d. L. . **E-puck education robot**. 2021. Disponível em: <http://www.e-puck.org/index.php?option=com_content&view=article&id=2&Itemid=8>.

FERREIRA, G. B.; VARGAS, P. A.; OLIVEIRA, G. M. An improved cellular automata-based model for robot path-planning. In: SPRINGER. **Conference Towards Autonomous Robotic Systems**. 2014. p. 25–36. Disponível em: <https://doi.org/10.1007/978-3-319-10401-0_3>.

FERREIRA, G. B. S. et al. Modelos baseados em autômatos celulares para o planejamento de caminhos em robôs autônomos. Universidade Federal de Uberlândia, 2014. Disponível em: <https://doi.org/10.14393/ufu.di.2014.25>.

FU, Y.; LI, H.; MA, Y. Path planning of cooperative robotics and robot team. In: IEEE. **2006 IEEE International Conference on Robotics and Biomimetics**. 2006. p. 1250–1255. Disponível em: <https://doi.org/10.1109/ROBIO.2006.340107>.

GARDNER, M. Mathematical games: The fantastic combinations of john conway's new solitaire game "life". **Scientific American**, vol, v. 223, n. 4, p. 120–123, 1970. Disponível em: <https://www.jstor.org/stable/24927642>.

GHAFFARI, A.; MEGHDARI, A.; NADERI, D.; ESLAMI, S. International journal of advanced robotic systems. **Stability enhancement of mobile manipulators via soft computing**, v. 12, n. 3, p. 191–198, 2004. Disponível em: <https://doi.org/10.5772/5739>.

GUANGHUA, W.; DEYI, L.; WENYAN, G.; PENG, J. Study on formation control of multi-robot systems. In: IEEE. **2013 Third International Conference on Intelligent System Design and Engineering Applications**. 2013. p. 1335–1339. Disponível em: <https://doi.org/10.1109/ISDEA.2012.316>.

IOANNIDIS, K.; SIRAKOULIS, G. C.; ANDREADIS, I. A cellular automaton collision-free path planner suitable for cooperative robots. In: IEEE. **2008 Panhellenic Conference on Informatics**. 2008. p. 256–260. Disponível em: <https://doi.org/10.1109/PCI.2008.43>.

_____. A path planning method based on cellular automata for cooperative robots. **Applied Artificial Intelligence**, Taylor & Francis, v. 25, n. 8, p. 721–745, 2011. Disponível em: <https://doi.org/10.1080/08839514.2011.606767>.

JIANJUN, Y.; HONGWEI, D.; GUANWEI, W.; LU, Z. Research about local path planning of moving robot based on improved artificial potential field. In: IEEE. **2013 25th Chinese Control and Decision Conference (CCDC)**. 2013. p. 2861–2865. Disponível em: <https://doi.org/10.1109/CCDC.2013.6561433>.

KAVRAKI, L. E.; SVESTKA, P.; LATOMBE, J.-C.; OVERMARS, M. H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. **IEEE transactions on Robotics and Automation**, IEEE, v. 12, n. 4, p. 566–580, 1996. Disponível em: <https://doi.org/10.1109/70.508439>.

KOSTAVELIS, I.; BOUKAS, E.; NALPANTIDIS, L.; GASTERATOS, A. Path tracing on polar depth maps for robot navigation. In: SPRINGER. **International Conference on Cellular Automata**. 2012. p. 395–404. Disponível em: <https://doi.org/10.1007/978-3-642-33350-7_41>.

LIMA, D. A.; OLIVEIRA, G. M. A cellular automata ant memory model of foraging in a swarm of robots. **Applied Mathematical Modelling**, Elsevier, v. 47, p. 551–572, 2017. Disponível em: <https://doi.org/10.1016/j.apm.2017.03.021>.

LIMA, D. A.; TINOCO, C. R.; VIEDMAN, J. M.; OLIVEIRA, G. M. Coordination, synchronization and localization investigations in a parallel intelligent robot cellular automata model that performs foraging task. In: **ICAART (2)**. [s.n.], 2017. p. 355–363. Disponível em: <https://doi.org/10.5220/0006081403550363>.

LINGELBACH, F. Path planning using probabilistic cell decomposition. In: IEEE. **IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004**. 2004. v. 1, p. 467–472. Disponível em: <https://doi.org/10.1109/ROBOT.2004.1307193>.

LUO, C.; GAO, J.; MURPHEY, Y. L.; JAN, G. E. A computationally efficient neural dynamics approach to trajectory planning of an intelligent vehicle. In: **Int. Joint Conf. on Neural Networks**. [s.n.], 2014. p. 934–939. Disponível em: <https://doi.org/10.1109/IJCNN.2014.6889604>.

MARCHESE, F. Cellular automata in robot path planning. In: **Advanced Mobile Robots, Euromicro Workshop on**. [s.n.], 1996. p. 116–116. Disponível em: <https://doi.org/10.1109/EURBOT.1996.551890>.

MARCHESE, F. M. A directional diffusion algorithm on cellular automata for robot path-planning. **Future Generation Computer Systems**, Elsevier, v. 18, n. 7, p. 983–994, 2002. Disponível em: <https://doi.org/10.1016/S0167-739X(02)00077-8>.

_____. A reactive planner for mobile robots with generic shapes and kinematics on variable terrains. In: IEEE. **ICAR'05. Proceedings., 12th International Conference on Advanced Robotics, 2005.** 2005. p. 23–30. Disponível em: <https://doi.org/10.1109/ICAR.2005.1507386>.

_____. **Spatiotemporal MCA Approach for the Motion Coordination of Heterogeneous MRS**. INTECH Open Access Publisher, 2008. Disponível em: <https://doi.org/10.5772/5480>.

_____. Time-invariant motion planner in discretized c-spacetime for mrs. **Multi-Robot Systems. Trends and Development**, p. 307–324, 2011.

MARTINS, L. G.; CÂNDIDO, R. d. P.; ESCARPINATI, M. C.; VARGAS, P. A.; OLIVEIRA, G. M. de. An improved robot path planning model using cellular automata. In: SPRINGER. **Annual Conference Towards Autonomous Robotic Systems**. 2018. p. 183–194. Disponível em: <https://doi.org/10.1007/978-3-319-96728-8_16>.

MARTINS, L. G.; FYNN, E.; OLIVEIRA, G. Algoritmos genéticos usados na obtenção de autômatos celulares para resolução da tarefa de classificação de densidade no espaço bidimensional. **Horizonte Científico**, 2011. Disponível em: <http://www.seer.ufu.br/index.php/horizontecientifico/article/view/7288>.

MATARIĆ, M. J.; MAJA, J.; ARKIN, R. C. et al. **The robotics primer**. [S.l.]: Mit Press, 2007.

MENG, Y.; GUO, H.; JIN, Y. A morphogenetic approach to flexible and robust shape formation for swarm robotic systems. **Robotics and Autonomous Systems**, Elsevier, v. 61, n. 1, p. 25–38, 2013. Disponível em: <https://doi.org/10.1016/j.robot.2012.09.009>.

MONDADA, F.; BONANI, M.; RAEMY, X.; PUGH, J.; CIANCI, C.; KLAPTOCZ, A.; MAGNENAT, S.; ZUFFEREY, J.-C.; FLOREANO, D.; MARTINOLI, A. The e-puck, a robot designed for education in engineering. In: IPCB: INSTITUTO POLITÉCNICO DE CASTELO BRANCO. **Proceedings of the 9th conference on autonomous robot systems and competitions**. 2009. v. 1, n. CONF, p. 59–65. Disponível em: <https://infoscience.epfl.ch/record/135236>.

NAMETALA, S. **LCBio Talk 1:A New Distance Diffusion Algorithm for a Path-Planning Model based on Cellular Automata**. 2020. Disponível em: <https://youtu.be/L-amaWaHIwc>.

_____. **Distributed navigation (Robotics)**. 2021. Disponível em: <https://www.youtube.com/watch?v=Cf_iKb1ZCi4&list=PLhS2nTblfgy49NytM8vD9cbnGopkSHjaz>.

_____. **PATHP model (Robotics)**. 2021. Disponível em: <https://www.youtube.com/watch?v=g_QNkcmWpL4&list=PLhS2nTblfgy6a4F7PsrOpHcAdYsCL9jVW>.

NAMETALA, S. C.; MARTINS, L. G.; OLIVEIRA, G. M. A new distance diffusion algorithm for a path-planning model based on cellular automata. In: IEEE. **2020 IEEE Congress on Evolutionary Computation (CEC)**. 2020. p. 1–8. Disponível em: <https://doi.org/10.1109/CEC48606.2020.9185824>.

NAVARRO, I.; MATÍA, F. A survey of collective movement of mobile robots. **International Journal of Advanced Robotic Systems**, SAGE Publications Sage UK: London, England, v. 10, n. 1, p. 73, 2013. Disponível em: <https://doi.org/10.5772/54600>.

OLIVEIRA, G. M.; SILVA, R. G.; FERREIRA, G. B.; COUCEIRO, M. S.; AMARAL, L. R. D.; VARGAS, P. A.; MARTINS, L. G. A. A cellular automata-based path-planning for a cooperative and decentralized team of robots. In: IEEE. **2019 IEEE Congress on Evolutionary Computation (CEC)**. 2019. p. 739–746. Disponível em: <https://doi.org/10.1109/CEC.2019.8790205>.

OLIVEIRA, G. M.; VARGAS, P. A.; FERREIRA, G. B. Investigating a cellular automata model that performs three distance diffusion on a robot path planning. In: MIT PRESS. **Artificial Life Conference Proceedings 13**. 2015. p. 271–278. Disponível em: <https://dx.doi.org/10.7551/978-0-262-33027-5-ch052>.

_____. A local decision making cellular automata-based path-planning. In: **The European Conference on Artificial Life (ECAL 2015), Procedings, York, United Kingdom, Monday**. [S.l.: s.n.], 2015.

OLIVEIRA, G. M. de; SILVA, R. G.; AMARAL, L. R. do; MARTINS, L. G. An evolutionary-cooperative model based on cellular automata and genetic algorithms for the navigation of robots under formation control. In: IEEE. **2018 7th Brazilian Conference on Intelligent Systems (BRACIS)**. 2018. p. 426–431. Disponível em: <https://doi.org/10.1109/BRACIS.2018.00080>.

PEI, H.; LOU, Y.; YE, F. Robot path planning based autonomous vehicles.on cellular automata with mixed neighborhoods. In: IEEE. **2018 11th International Symposium on Computational Intelligence and Design (ISCID)**. 2018. v. 1, p. 114–117. Disponível em: <https://doi.org/10.1109/ISCID.2018.00033>.

PENDLETON, S. D.; ANDERSEN, H.; DU, X.; SHEN, X.; MEGHJANI, M.; ENG, Y. H.; RUS, D.; ANG, M. H. Perception, planning, control, and coordination for autonomous vehicles. **Machines**, Multidisciplinary Digital Publishing Institute, v. 5, n. 1, p. 6, 2017. Disponível em: <https://doi.org/10.3390/machines5010006>.

PÉREZ, I. D. V. Autómata celular para la planeación de trayectoria. IPN, 2020.

PROROK, A.; BAHR, A.; MARTINOLI, A. Low-cost collaborative localization for large-scale multi-robot systems. In: IEEE. **2012 IEEE International Conference on Robotics and Automation**. 2012. p. 4236–4241. Disponível em: <https://doi.org/10.1109/ICRA.2012.6225016>.

RAMER, C.; REITELSHÖFER, S.; FRANKE, J. A robot motion planner for 6-dof industrial robots based on the cell decomposition of the workspace. In: IEEE. **IEEE ISR 2013**. 2013. p. 1–4. Disponível em: <https://doi.org/10.1109/ISR.2013.6695611>.

REZAEE, H.; ABDOLLAHI, F. A decentralized cooperative control scheme with obstacle avoidance for a team of mobile robots. **IEEE Transactions on Industrial Electronics**, IEEE, v. 61, n. 1, p. 347–354, 2013. Disponível em: <https://doi.org/10.1109/TIE.2013.2245612>.

RODRÍGUEZ-SEDA, E. J.; RICO, C. K. Cellular automata based decentralized cooperative collision avoidance control for multiple mobile robots. In: IEEE. **2019 IEEE International Symposium on Measurement and Control in Robotics (ISMCR)**. 2019. p. A3–3. Disponível em: <https://doi.org/10.1109/ISMCR47492.2019.8955689>.

ROSENBERG, A. L. Cellular antomata. In: STOJMENOVIC, I.; THULASIRAM, R. K.; YANG, L. T.; JIA, W.; GUO, M.; MELLO, R. F. de (Ed.). **Parallel and Distributed Processing and Applications**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 78–90.

_____. Cellular antomata: food-finding and maze-threading. In: IEEE. **2008 37th International Conference on Parallel Processing**. 2008. p. 528–535. Disponível em: <https://doi.org/10.1109/ICPP.2008.13>.

ROUMELIOTIS, S. I.; BEKEY, G. A. Distributed multirobot localization. **IEEE transactions on robotics and automation**, IEEE, v. 18, n. 5, p. 781–795, 2002. Disponível em: <https://doi.org/10.1109/TRA.2002.803461>.

SANTANA, A. M.; SOUSA, A. A.; BRITTO, R. S.; ALSINA, P. J.; MEDEIROS, A. A. D. d. Localization of a mobile robot based on odometry and natural landmarks using extended kalman filter. In: INTERNATIONAL CONFERENCE ON INFORMATICS IN CONTROL, AUTOMATION AND ROBOTICS. 2008. Disponível em: <https://repositorio.ufrn.br/jspui/handle/1/6150>.

SETIAWAN, Y. D.; PRATAMA, P. S.; JEONG, S. K.; DUY, V. H.; KIM, S. B. Experimental comparison of a* and d* lite path planning algorithms for differential drive automated guided vehicle. In: **Recent Advances in Electrical Engineering and Related Sciences**. [s.n.], 2014. p. 555–564. Disponível em: <https://doi.org/10.1007/978-3-642-41968-3_55>.

SHU, C.; BUXTON, H. Parallel path planning on the distributed array processor. **Parallel computing**, Elsevier, v. 21, n. 11, p. 1749–1767, 1995. Disponível em: <https://doi.org/10.1016/0167-8191(96)80006-8>.

SILVA, C. E. d. et al. Coordenação de múltiplos veículos autônomos de entrega usando k-means e algoritmos bio-inspirados. Universidade Federal de Uberlândia, 2020. Disponível em: <http://doi.org/10.14393/ufu.di.2020.563>.

SILVA, R. G. O. et al. Um modelo baseado em autômatos celulares e algoritmos genéticos para a navegação de um time de robôs visando o controle de formação e o desvio de obstáculos. Universidade Federal de Uberlândia, 2015. Disponível em: <https://doi.org/10.14393/ufu.di.2015.476>.

SOOFIYANI, F. R.; RAHMANI, A. M.; MOHSENZADEH, M. A straight moving path planner for mobile robots in static environments using cellular automata. In: IEEE. **2010 2nd International Conference on Computational Intelligence, Communication Systems and Networks**. 2010. p. 67–71. Disponível em: <https://doi.org/10.1109/CICSyN.2010.28>.

TARIQ, J.; KUMARAVEL, A. Construction of cellular automata over hexagonal and triangular tessellations for path planning of multi-robots. In: IEEE. **2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)**. 2016. p. 1–6. Disponível em: <https://doi.org/10.1109/ICCIC.2016.7919686>.

TAVAKOLI, Y.; JAVADI, H. H. S.; ADABI, S. A cellular automata based algorithm for path planning in multi-agent systems with a common goal. **International journal of computer science and network security**, v. 8, n. 7, p. 119–123, 2008. Disponível em: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.461.8423&rep=rep1&type=pdf>.

TINOCO, C. R.; OLIVEIRA, G. M. Heterogeneous teams of robots using a coordinating model for surveillance task based on cellular automata and repulsive pheromone. In: IEEE. **2019 IEEE Congress on Evolutionary Computation (CEC)**. 2019. p. 747–754. Disponível em: <https://doi.org/10.1109/CEC.2019.8790266>.

TINOCO, C. R. et al. **Coordenação de times de robôs baseada em mapas descentralizados de feromônio repulsivo e regras locais de autômatos celulares**. Dissertação (Mestrado), 2019. Disponível em: <http://dx.doi.org/10.14393/ufu.di.2019.325>.

TZIONAS, P. G.; THANAILAKIS, A.; TSALIDES, P. G. Collision-free path planning for a diamond-shaped robot using two-dimensional cellular automata. **IEEE Transactions on Robotics and Automation**, IEEE, v. 13, n. 2, p. 237–250, 1997. Disponível em: <https://doi.org/10.1109/70.563646>.

VIVALDINI, K. C. T.; GALDAMES, J. P. M.; PASQUAL, T. B.; SOBRAL, R. M.; ARAÚJO, R. C.; BECKER, M.; CAURIN, G. Automatic routing system for intelligent warehouses. In: **IEEE Int. Conf. on Robotics and Automation**. [s.n.], 2010. v. 1, p. 1–6. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.385.7900&rep=rep1&type=pdf>.

WEISSTEIN, E. W. **Piano Mover's Problem."From MathWorld–A Wolfram Web Resource**. 2019. Disponível em: <https://mathworld.wolfram.com/PianoMoversProblem.html>.

ZHANG, Y.; FATTAHI, N.; LI, W. Probabilistic roadmap with self-learning for path planning of a mobile robot in a dynamic and unstructured environment. In: IEEE. **2013 IEEE International Conference on Mechatronics and Automation**. 2013. p. 1074–1079. Disponível em: <https://doi.org/10.1109/ICMA.2013.6618064>.

ZHU, P.; DAI, W.; YAO, W.; MA, J.; ZENG, Z.; LU, H. Multi-robot flocking control based on deep reinforcement learning. **IEEE Access**, IEEE, v. 8, p. 150397–150406, 2020. Disponível em: <https://doi.org/10.1109/ACCESS.2020.3016951>.