UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E INFORMÁTICA INDUSTRIAL

RICARDO DA ROSA

MÉTODO BIOINSPIRADO PARA EXPLORAÇÃO E MAPEAMENTO DE AMBIENTES INTERNOS COM MÚLTIPLOS VANT

TESE

CURITIBA

2020

RICARDO DA ROSA

MÉTODO BIOINSPIRADO PARA EXPLORAÇÃO E MAPEAMENTO DE AMBIENTES INTERNOS COM MÚLTIPLOS VANT

Tese apresentada ao Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do grau de "Doutor em Ciências" – Área de Concentração: Engenharia de Computação.

Orientador: Prof. Dr. Marco Aurélio Wehrmeister

CURITIBA

2020

Dados Internacionais de Catalogação na Publicação

Rosa, Ricardo da
Método bioinspirado para exploração e mapeamento de ambientes internos com múltiplos VANT [recurso eletrônico] / Ricardo da Rosa
1 arquivo texto (209 f.): PDF; 20,1 MB.

Modo de acesso: World Wide Web
Título extraído da tela de título (visualizado em 1 jun. 2020)
Texto em português com resumo em inglês
Tese (Doutorado) - Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Curitiba, 2020
Bibliografia: f. 105-111.

1. Engenharia elétrica - Teses. 2. Aeronave não-tripulada. 3. Drone. 4. Espaços topológicos lineares. 5. Mapeamentos (Matemática) - Simulação por computador. 6. Métodos de simulação. I. Wehrmeister, Marco Aurelio. II. Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial. III. Título.

CDD: Ed. 23 – 621.3

Biblioteca Central da UTFPR, Câmpus Curitiba
Bibliotecário: Adriano Lopes CRB-9/1429

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Pesquisa e Pós-Graduação

TERMO DE APROVAÇÃO DE TESE Nº_____

A Tese de Doutorado intitulada **Método Bioinspirado para Exploração e Mapeamento de Ambientes Internos com Múltiplos VANT,** defendida em sessão pública pelo(a) candidato(a) **Ricardo da Rosa,** no dia **30 de abril de 2020**, foi julgada aprovada em sua forma final para obtenção do título de Doutorem Ciências, Área de Concentração – **Engenharia De Computação**, Linha de Pesquisa – **Engenharia De Sistemas Computacionais,** pelo Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial - CPGEI.

BANCA EXAMINADORA:

Prof. Dr. Marco Aurélio Wehrmeister – (UTFPR) - Orientador
Prof. Dr. Edson Pignaton de Freitas – (UFRGS)
Prof. Dr. Eduardo Todt – (UFPR)
Prof. Dr. Cesar Tecla – (UTFPR)
Prof. Dr. André Schneider de Oliveira – (UTFPR)

A via original deste documento encontra-se arquivada na Secretaria do Programa, contendo a assinatura da Coordenação após a entrega da versão corrigida do trabalho.

Curitiba, 30 de abril de 2020.

_____

Carimbo e Assinatura do(a) Coordenador(a) do Programa

Com carinho, para Márcia e Marina...

AGRADECIMENTOS

Agradeço ao Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial da Universidade Tecnológica Federal do Paraná - Campus Curitiba, pela oportunidade de realizar esta pesquisa.

Ao professor doutor Marco Aurélio Wehrmeister, meu orientador, pela confiança, pelo olhar sábio e paciência, o que tornou esse doutorado um processo de crescimento e evolução profissional e pessoal. Meus agradecimentos, respeito e admiração.

Aos professores doutores que compuseram a banca de qualificação e de defesa de doutorado, os quais contribuíram expressivamente para a elaboração desta pesquisa: André Schneider de Oliveira, Cesar Augusto Tacla, Edison Pignaton de Freitas e Eduardo Todt.

A todos os professores que ministraram disciplinas durante o doutorado, os quais ampliaram os horizontes para minha atuação profissional como docente.

Aos parceiros de pesquisa do Instituto Politécnico de Bragança, Ana Isabel Pereira, José Luis Lima e Thadeu Brito, pelas contribuições na definição de estratégias adotadas nesta tese.

Aos colegas do Laboratório de Engenharia de Sistemas Computacionais.

Ao apoio institucional proporcionado pelo Instituto Federal do Paraná, o qual possibilitou o afastamento integral de minhas atividades para realização deste doutorado e, em especial, ao Campus Cascavel, que disponibilizou o equipamento utilizado nos experimentos desta tese.

Aos meus familiares, em especial meus irmãos Renata e Rodrigo da Rosa, os quais sempre acompanharam minha trajetória acadêmica e profissional.

Aos meus pais, Celomar da Rosa e Tânia Maria Martinasso da Rosa, por todo carinho e ensinamentos de vida passados desde minha infância e até hoje, pois ainda aprendo muito com eles.

A minha esposa, Márcia Souza da Rosa, que também realizou seus estudos de doutorado juntamente comigo, compartilhando o caminho e sempre me apoiando e incentivando em todas as horas. Você é uma pessoa imprescindível na minha vida. Te amo.

A minha amada filha, Marina Aparecida Souza da Rosa, que nasceu durante a realização deste doutorado, sendo fonte de luz, inspiração e renovação das minhas energias. Você é a melhor parte de mim. Te amo.

A Deus, por me proporcionar uma vida com uma família linda, todos meus estudos acadêmicos em instituições renomadas e por me inserir em uma profissão da qual me orgulho.

“Em tudo dai graças, porque esta é a vontade de Deus em Cristo Jesus para convosco”.

1 Tessalonicenses 5:18

RESUMO

ROSA, Ricardo da. MÉTODO BIOINSPIRADO PARA EXPLORAÇÃO E MAPEAMENTO DE AMBIENTES INTERNOS COM MÚLTIPLOS VANT . 209 f. Tese – Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná. Curitiba, 2020.

O uso de veículos aéreos não tripulados autônomos surge como uma opção de apoio em diversas atividades realizadas pelo ser humano, especialmente naquelas em que o processo de execução pode ser perigoso. A busca de vítimas feita por equipes de resgate em ambientes atingidos por catástrofes, tais como terremotos e furacões, é um exemplo disso, uma vez que, geralmente, não há conhecimento prévio do estado do espaço físico. A exploração e o mapeamento de ambientes com múltiplos Veículos Aéreos Não Tripulados (VANT) pode auxiliar equipes na identificação das condições do local em um ambiente interno. A construção de mapas a partir de vários VANT autônomos exige um método de exploração do ambiente que pode fornecer resultados plausíveis do mapeamento e apresentar visibilidade para equipes de resgate em um período de tempo adequado. O método de exploração e mapeamento proposto faz a utilização de uma estrutura topológica para a representação do mapa, inspirado na estrutura hexagonal que as colmeias de abelhas possuem. Os VANT possuem um comportamento semelhante ao das abelhas no processo de construção dos favos da colmeia para as atividades de mapeamento de novos espaços. Nesse sentido, foram desenvolvidas três estratégias diferentes para a alocação de hexágonos a serem explorados: First In First Out, Distância Euclidiana e Distância Euclidiana Relativa. A estratégia de Distância Euclidiana Relativa (DER) apresentou resultados mais satisfatórios em intensidade do tráfego dos robôs aéreos e tempo total de exploração e mapeamento do ambiente. Experimentos foram realizados, fazendo uso de ambientes de simulação para validação do método proposto, variando o número de cenários e as estratégias adotadas. A redução dos deslocamentos dos VANT influencia diretamente no tempo de execução da exploração. O mapeamento topológico bioinspirado proposto nesta tese apresenta-se como uma opção viável para times de robôs aéreos na atividade de exploração e mapeamento de ambientes internos desconhecidos.

Palavras-chave: Mútiplos VANT, Exploração de Ambientes Internos, Método inspirado em colmeia de abelhas

ABSTRACT

ROSA, Ricardo da. BIOINSPIRED EXPLORATION AND MAPPING INDOOR ENVIRONMENTS METHOD WITH MULTIPLE UAVS. 209 f. Tese – Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná. Curitiba, 2020.

The use of autonomous unmanned aerial vehicles appears as an option to support various activities carried out by the human being, specially when the execution of these activities can be dangerous. An example of this is the search and rescue of victims in environments affected by catastrophes such as earthquakes or hurricanes, once there is no prior knowledge of the state of space. The exploration and mapping of the environment with multiple Unmanned Aerial Vehicles (UAV) can assist these teams in identifying the conditions of the place. The construction of maps from data provided by several autonomous UAVs requires a method for exploring the environment so that plausible results of the mapping are visible to the rescue staff within an acceptable period of time. The proposed exploration and mapping method uses a topological structure for the map representation inspired by the hexagonal structure that the bee hives present. Therefor each UAV has a behavior, in the mapping activities of new spaces, similar to that of bees in the beehive construction process. Three strategies for the allocation of hexagons to be explored were developed: First In First Out, Euclidean Distance and Relative Euclidean Distance. The Relative Euclidean Distance (DER) strategy showed satisfactory results in traffic intensity of UAVs leading to lesser total time for the exploration and the mapping of the environment. Experiments were carried out using simulation environments to validate the proposed method, varying the number of scenarios and the strategies adopted. The reduction of UAV displacements directly influences the exploration execution time. The bio-inspired topological mapping proposed in this thesis presents itself as a viable option for teams of aerial robots in the activity of exploration and mapping of unknown internal environments.

Keywords: Multiple UAVs, Internal Environment Exploration, Honeycombs-inspired method

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

6DOF	Six Degree of Freedom
AR	Augmented Reality
DE	Distância Euclidiana
DER	Distância Euclidiana Relativa
DDOF	Differentable Degrees of Freedom
DOF	Degree of Freedom - Graus de Liberdade
DP-SLAM	Distributed Particle Mapping SLAM
EKF-SLAM	Extended Kalman Filter
FIFO	First In First Out
GA	Grau de Adjacência
GPS	Global Positioning System
HALE	Hight Altitude Long Endurance
HTOL	Horizontal Takeoff and Landing
IFPR	Instituto Federal do Paraná
IMU	Inertial Measurement Unit
IPB	Instituto Politécnico de Bragança
IR	Infrared
MALE	Medium Altitude Long Endurance
MAV	Micro UAV
MLE	Maximum Likelihood Estimation
MUAV	Mini UAV
NAV	Nano UAV
OACI	Organização de Aviação Civil Internacional
PHD-SLAM	Probability Hyphotesis Density SLAM
PTAM	Parallel Tracking and Mapping
RGB-D	Red, Green, Blue and Depth
RPA	Remotely Piloted Aircraft
RPAS	Remotely Piloted Aircraft System
SANT	Sistemas de Aeronaves Não Tripuladas
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SURF	Speeded-Up Robust Features
TUAV	Medium Range or Tatical UAV
UAV	Unmanned Aerial Vehicles
UC	Unidade Central
UGV	Unmanned Ground Vehicles
UTFPR	Universidade Tecnológica Federal do Paraná
VANT	Veículo Aéreo Não Tripulado
VOXEL	Volumetric Pixel
VTNT	Veículo Terrestre Não Tripulado
VTOL	Vertical Takeoff and Landing

LISTA DE SÍMBOLOS

DE	Distância Euclidiana
DER	Distância Euclidiana Relativa
Δh	Variação da altura da fuselagem do VANT
Δs	Variação do comprimento da fuselagem do VANT
H	Altura do hexágono
HN	Lista de hexágonos não visitados
R	Raio do hexágono
SU	Conjunto de VANTs

SUMÁRIO

1 INTRODUÇÃO

O uso de robôs para resolução de problemas vem se tornando cada vez mais frequente. Além de ampla aplicação na indústria, o desenvolvimento de robôs móveis cresce com a popularização e redução de custos de sensores e atuadores. Avanços significativos têm sido feitos no sentido de desenvolver robôs que possam realizar tarefas de forma autônoma ou que tenham alguma base operacional, a qual controla o robô e ainda mantém dados de suas ações. Alguns exemplos de uso de robôs móveis estão relacionados ao transporte de materiais em hospitais (EVANS et al., 1992), depósitos e fábricas (POUDEL, 2013), veículos autônomos para agricultura (EDAN, 1995), veículos urbanos autônomos (LIDORIS et al., 2009), guias em museus (THRUN et al., 2000), entre outros. Essas novas aplicações demostram que a robótica móvel pode realizar tarefas complexas que até então eram apenas realizadas por seres humanos com experiência e conhecimento. Os robôs podem navegar em ambientes desconhecidos e desviarem de obstáculos inesperados, reagindo de forma inteligente aos estímulos do ambiente (THRUN, 2002). Outra aplicação da robótica móvel é no suporte a times de resgate (MICHAEL et al., 2012) em situações de desastres naturais como deslizamentos de terra, inundações e terremotos, quando o acesso a construções pode estar limitado ou ainda oferecer perigo aos seres humanos. Algumas vezes a atividade de resgate pode por em risco a vida de profissionais de apoio, seja da defesa civil, bombeiros e até mesmo policiais. O uso de Veículos Aéreos Não Tripulados (VANT - em inglês *Unmanned Aerial Vehicles* - UAV) pode auxiliar nessas atividades de resgate, especialmente em áreas onde desastres naturais tornam a movimentação de um robô terrestre difícil e mesmo a chegada de humanos. Doravante, será utilizado neste trabalho a sigla VANT, quando for necessária a referência a veículos aéreos não tripulados, sendo sinônimo de robô aéreo.

Para acessar locais internos e desconhecidos, os robôs necessitam de meios para identificar o espaço onde se encontram, gerando mapas do ambiente que possam auxiliar no próprio controle de movimento e de suas tarefas, bem como nas ações a serem tomadas por humanos, como decisões feitas por equipes de busca e resgate. Assim, um robô autônomo precisa tratar dois problemas críticos para navegar em seu ambiente: mapear o ambiente e

buscar sua própria localização no mapa (SAEEDI et al., 2016), sendo estes uns dos principais desafios da robótica móvel (AULINAS et al., 2008). O mapeamento consiste na representação gráfica ou lógica de um ambiente, e pode ser classificado de diversas formas. Thrun (2002) faz uma divisão das técnicas como mapeamento métrico e mapeamento topológico. O mapeamento métrico é o método de representação mais comum, pois fornece uma representação métrica do ambiente, no qual cada objeto é representado nas devidas proporções e distâncias, geralmente em escalas diferentes à escala real. O mapeamento topológico representa o ambiente mediante relações entre vários objetos e áreas e é geralmente representado por grafos.

A localização é o registro da posição do robô em relação a um mapa ou pontos de referência. Em um mapa métrico, a localização é representada pela distância entre o robô e os referenciais fixos do mapa. Exemplos de aplicações que utilizam localização em mapeamento métrico são aqueles baseados em sinais de GPS (*Global Positioning System* - Sistema de Posicionamento Global), que tem como função principal localizar-se dentro destes mapas através de triangulação de sinais de satélites (RAY et al., 2009). Em ambientes externos, o uso de GPS facilita o mapeamento do espaço e a localização do robô. Entretanto, em ambientes internos, o uso destes recursos não é possível na maioria das vezes. Para isso, outras técnicas precisam ser aplicadas para definir a localização de um robô móvel.

A atividade de exploração de um ambiente com robôs pode ter diversas aplicações, desde o reconhecimento de espaços específicos, missões de resgates e até tarefas de varredura/limpeza, quando é necessária uma cobertura completa de determinada área (BURGARD et al., 2005). Em algumas situações, o fornecimento prévio de um mapa ou planta do ambiente pode ser útil na otimização das tarefas a serem realizadas, porém em ambientes internos desestruturados, como locais atingidos por terremotos ou outras catástrofes, a complexidade pode ser maior devido aos mapas conhecidos não estarem atualizados ou simplesmente não existirem. Assim, ações exploratórias nesses tipos de ambientes necessitam da criação de um mapa durante a atividade de exploração, o qual também pode ser útil para equipes de resgate utilizarem como apoio na tomada de decisão em alguma ação planejada.

1.1 PROBLEMAS DE PESQUISA E HIPÓTESES

O desenvolvimento de algoritmos, técnicas e dispositivos para o mapeamento de ambientes é abordado em diversos trabalhos (como pode ser visto no capítulo 3). Em grande parte, esses trabalhos apresentam aprofundamentos e refinamentos de soluções para espaços internos explorados por um único agente. Entretanto, o uso de apenas um agente para exploração pode apresentar alguns problemas que podem restringir em alguns aspectos

o mapeamento desejado (BURGARD et al., 2005):

- tempo para realizar o mapeamento: pode ser inaceitável;
- as características do espaço podem limitar o acesso de alguns robôs, por exemplo espaços onde um armário está derrubado pode restringir a passagem de um robô terrestre;
- falhas de comunicação: um robô pode afastar-se consideravelmente de uma base de comunicação, de forma que trocas de dados são perdidas;

O tempo de mapeamento é considerada uma questão fundamental, em se tratando da busca de vítimas em ambientes atingidos por desastres naturais, já que pode haver vítimas a serem socorridas, mas que ainda não foram localizadas. Neste sentido, o uso de times de robôs na resolução de tarefas de exploração e mapeamento pode tornar sua execução mais eficiente (WAHARTE; TRIGONI, 2010). Além disso, se considerarmos que o robô em uso é um VANT, o tempo de autonomia torna-se vital para o sucesso da missão. Ou seja, caso haja energia suficiente para realizar o mapeamento, o uso de um único robô para um mapeamento completo do espaço demandará mais tempo do que o uso de múltiplos robôs. Ainda, caso o robô seja um VANT do tipo multirotor (comumente conhecido como drone), o mapeamento completo pode não ser finalizado devido à sua baixa autonomia de voo. Assim, em operações de busca e resgate, a rapidez no uso de múltiplos agentes no mapeamento de espaços, auxilia no planejamento e execução de tarefas estratégicas.

Dadas as características de um espaço a ser mapeado, alguns tipos de robôs podem ter a sua movimentação no ambiente limitada por obstáculos. Um exemplo disso são veículos terrestres não tripulados (VTNT - do inglês UGV - *Unmanned Ground Vehicles*) os quais não possuem capacidade de superar determinados tipos de obstáculos, deixando partes do ambiente sem mapeamento, ou seja, desconhecidas no mapa. Abordagens heterogêneas (MICHAEL et al., 2012) têm surgido como uma solução para este problema, integrando o uso de VTNT com VANT e, dessa forma, obtendo êxito em mapear um espaço de forma completa.

Um dos desafios do mapeamento de ambientes internos é a exploração com diversos robôs em um mesmo espaço, de modo que haja geração de dados em tempo suficientemente rápido para equipes de busca e resgate. Logo, o problema de pesquisa desta tese de doutorado define-se como: é possível utilizar múltiplos robôs aéreos[1] (VANT[2]) para realizar a exploração e mapeamento de ambientes internos, de forma que o mapa gerado possa ser usado por humanos

[1]Neste texto os termos "robô aéreo" e "VANT" são usados intercaladamente como sinônimos um do outro
[2]O termo "VANT" é usado como sinônimo de "VANT do tipo VTOL com quatro rotores (quadcoptero)

no planejamento de ações em missões de busca e resgate? Para responder esta pergunta, foram propostas as seguintes hipóteses:

- H1: é possível realizar a exploração e o mapeamento colaborativo de ambientes internos usando múltiplos robôs aéreos;
- H2: é possível fazer a distribuição de tarefas de exploração e mapeamento, de forma que o tempo de exploração seja reduzido em comparação ao uso de um único robô;
- H3: é possível haver a circulação de diversos VANT em um mesmo ambiente, de modo que as colisões entre eles sejam evitadas sem o uso de informações globais de localização dos VANT no ambiente;

1.2 OBJETIVOS

O objetivo geral desta tese de doutorado é propor um método bioinspirado de exploração e mapeamento de ambientes internos desconhecidos, utilizando múltiplos robôs aéreos para construção de um mapa topológico de forma colaborativa e com resultados dentro de um espaço de tempo aceitável por equipes, para que possam utilizar esses mapas no planejamento de suas ações nas missões de busca e resgate.

Dado o objetivo geral, os objetivos específicos de pesquisa são:

- Definir uma estrutura de mapa;
- Realizar a exploração e o mapeamento do ambiente com múltiplos VANT;
- Implementar estratégias de definição de locais a serem explorados, de forma que a evolução do mapeamento seja realizado no menor tempo em relação ao uso de um único robô;
- Gerar uma representação gráfica do ambiente para que humanos possam visualizar como é a estrutura do ambiente e sua ocupação;
- Fazer a identificação de pontos onde haja informação relevante para missões, de forma que dados, como temperatura elevada, estejam disponíveis para as equipes de busca e resgate;

1.3 CONTRIBUIÇÕES E ORIGINALIDADE DA PESQUISA

Na última década diversas pesquisas se dedicaram ao problema do mapeamento de ambientes internos, em que o acesso às informações de posicionamento global não é possível. O robô utilizado comumente é o terrestre (SAEEDI et al., 2011; HOWARD, 2006), entretanto algumas soluções que utilizam VANT foram desenvolvidas (WILLIAMS et al., 2014; LOIANNO et al., 2015). Quando se foca no uso de soluções cooperativas ou com múltiplos robôs, surgem trabalhos que fazem o uso de estruturas híbridas, combinando veículos terrestres e voadores. Ao restringir o mapeamento de ambientes internos utilizando apenas VANT, somente alguns trabalhos tratam de tal perspectiva, a saber: McCune e Madey (2013), Williams et al. (2014), Li e Aouf (2012) e Loianno et al. (2015).

O resultado da tarefa de mapeamento pode ser representado por mapas com características diferentes, por exemplo, mapas métricos e mapas topológicos (SIEGWART; NOURBAKHSH, 2004). O mapeamento topológico faz uso de grafos que fornece uma representação mais compacta em termos de uso de memória para seu armazenamento, porém não permite fornecer uma representação visual do espaço de forma métrica. O mapeamento métrico possibilita a modelagem do espaço como um todo, sendo que uma das principais técnicas métricas é a grade de ocupação, a qual discretiza o ambiente em células que identificam obstáculos ou espaços livres. Entretanto, a representação por grades de ocupação, em uma representação 3D, pode aumentar consideravelmente o consumo de memória.

Outro aspecto importante no processo de mapeamento com múltiplos robôs está no agrupamento dos dados de leitura de cada robô aéreo. Essa fusão de dados pode consumir um tempo de processamento alto e, muitas vezes, esse trabalho de junção dos diversos mapas é feito de forma *off-line* (JESSUP et al., 2014), não atendendo ao requisito de resultados em tempo hábil para ações de resgate. Outra questão importante a considerar é que em um cenário de desastres ou catástrofes, o acesso à planta baixa do espaço não está disponível de forma rápida para estas equipes de resgate, ou ainda esta planta pode não existir. Assim, o mapeamento do espaço deve considerar o fator de desconhecimento do local a ser explorado.

A originalidade desta pesquisa está na proposição de um método de mapeamento *bioinspirado* de ambientes internos desconhecidos com o uso de múltiplos robôs aéreos, de forma que atendam requisitos de tempo de processamento aceitável na modelagem do espaço para equipes de busca e resgate comparado com o uso de um único robô. Para diminuir o tempo de processamento, propõe-se que os robôs aéreos trabalhem de forma semelhante às abelhas na construção de favos nas colmeias. Para essa construção, uma topologia em formato hexagonal

auxilia na definição de pontos a serem explorados, assim como na identificação de espaços já mapeados. Os robôs aéreos transitarão dentro destes hexágonos do mapa topológico, os quais representam espaços trafegáveis dentro do ambiente, de modo que colisões sejam evitadas a partir da identificação dos locais que cada um dos robôs ocupam bem como do próximo movimento a ser realizado. O objetivo de cada robô é explorar espaços trafegáveis identificados e verificar se, a partir desse local, é possível acessar outros espaços. Cada descoberta feita por um robô aéreo é compartilhada com os demais através de uma estrutura global centralizada, a qual controla a identificação, o gerenciamento de novos hexágonos descobertos e delega aos VANT as tarefas de exploração que deverão realizar. Assim, busca-se gerar mapas onde os espaços explorados possam fornecer dados para humanos no suporte à tomada de decisão em missões de busca e resgate.

1.4 ESCOPO DA PESQUISA

O escopo principal desta tese de doutorado é a exploração e o mapeamento de ambientes internos desconhecidos, mediante o uso de múltiplos VANT. A exploração a ser realizada pelos robôs aéreos será definida por estratégias que podem considerar a ordem de descoberta de um novo local ou a distância entre locais a serem explorados até um ponto referencial. Não é considerada a informação global da localização de todos os VANT para realizar a escolha do próximo hexágono a ser explorado, apenas a alocação de tarefas de forma individual.

O método proposto baseia-se no uso exclusivo de VANT semelhantes entre si em configuração. O mapeamento faz uso de uma estrutura topológica de grafos combinado com uma representação em 3D do espaço mapeado.

O presente trabalho faz uso de ambientes de simulação para validação do método desenvolvido. Dessa forma, detalhes e restrições de tecnologias de comunicação são abstraídas, devido a configuração dos equipamentos utilizados. Considera-se aqui que a localização relativa dos robôs é conhecida, assim a identificação da localização dos VANT não faz parte do foco desta tese. Além disso, o tempo de processamento da simulação de um ambiente virtual demanda uma maior capacidade de processamento, o que pode acarretar em tempos de execução maiores do que os obtidos em ambientes reais.

O cenário considerado no processo de mapeamento é um ambiente onde não há conhecimento prévio da sua estrutura interna, e estabelece-se que os diversos VANT irão iniciar a navegação a partir de uma mesma base de lançamento comum. Após a primeira exploração

de um robô aéreo sentinela, os demais seguirão a exploração dos novos espaços identificados, adjacentes ao primeiro hexágono.

1.5 ORGANIZAÇÃO DO DOCUMENTO

Este documento está organizado da seguinte forma: o capítulo 2 apresenta conceitos fundamentais para o desenvolvimento deste trabalho. O capítulo 3 apresenta os trabalhos relacionados, destacando as contribuições deste trabalho. O capítulo 4 apresenta a proposta para o método de exploração e mapeamento bioinspirado na construção das colmeias de abelhas. O capítulo 5 apresenta os experimentos e simulações realizados, bem como os resultados alcançados. Por fim, o capítulo 6 apresenta as conclusões desta tese de doutorado.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os fundamentos teóricos necessários para a compreensão e para o desenvolvimento deste trabalho. São abordadas a concepção dos termos VANT, amplamente utilizado durante este texto, suas classificações e topologias, bem como técnicas de exploração e mapeamento.

2.1 VEÍCULOS AÉREOS NÃO TRIPULADOS (VANT)

Veículos aéreos não tripulados (VANT) possuem pesquisas relacionadas já no século 19 para fins militares e com seu uso ampliado na Primeira Guerra Mundial ((SANDY, 2017) *apud* (RONCONI et al., 2014)). Desde então, pesquisas continuaram e nas últimas décadas intensificaram-se no sentido de possibilitar soluções aplicadas na ciência e no mundo civil. Hoje podemos identificar VANT sendo utilizados na captação de imagens aéreas, agricultura, monitoramento e vistoria de estruturas (KOSLOSKY, 2018), entre outros.

No Brasil, o comando da aeronáutica definiu a Instrução do Comando da Aeronáutica (ICA) 100-40 (DECEA, 2015), a qual regulamenta a operação de sistemas de aeronaves remotamente pilotadas e o acesso a espaço aéreo brasileiro, bem como trata a nomenclatura de veículos aéreos não tripulados como aeronaves. O documento define três tipos de aeronaves:

- aeronave não tripulada totalmente autônoma: apresenta parâmetros e perfil de voo previamente programados sem a possibilidade de intervenção externa no controle do voo;
- aeronave de acompanhamento: é tripulada e voltada para o acompanhamento de voos experimentais de RPA;
- aeronave de remotamente pilotada (RPA): é uma aeronave não tripulada pilotada a partir de uma estação de pilotagem remota.

No ICA 100-40, o termo VANT é substituído por “aeronaves remotamente pilotadas” (RPA - em inglês *Remotely Piloted Aircraft*), enquanto os sistemas de associados são

denominados *Remotely Piloted Aircraft System* (RPAS). A necessidade de deixar explícita o controle remoto vem da definição que não foram regulamentadas as aeronaves totalmente autônomas pela Organização de Aviação Civil Internacional (OACI), pois elas não teriam responsabilidades previstas em normas.

Os VANT, por serem desprovidos de tripulação, possuem vantagens técnicas e econômicas, podendo ter tamanhos reduzidos ou distribuição de sua estrutura sem se preocupar com o espaço a ser ocupado por um piloto. Para que um VANT consiga realizar o seu deslocamento em uma trajetória específica, ele deve (ROSKAM, 2006):

- ter a capacidade de controlar seus movimentos para manter-se em voo nivelado e sair de um estado em equilíbrio para outro de forma segura;
- ter o seu projeto prevendo forças necessárias para o controle de seus movimentos;
- conseguir manter-se estável durante toda a sua trajetória de deslocamento.

Dessa forma, o uso de VANT não dependem apenas do próprio corpo do veículo e seus motores, mas também de um sistema que planeja todo o seu funcionamento, considerando sua configuração de fuselagem e topologia.

2.1.1 SISTEMAS DE AERONAVES NÃO TRIPULADAS (SANT)

Sistemas de Aeronaves não Tripuladas (SANT - do inglês *Unmanned Aircraft Systems* - UAS) são sistemas compostos por aeronave e base de controle (DECEA, 2015). Austin (2010) define SANT como um sistema onde a tripulação foi removida da aeronave e então substituída por um sistema computacional e redes de comunicação. Entretanto, uma aeronave com estas características não pode ser uma adaptação de uma aeronave tripulada. É necessário que seja projetado desde o início sem acomodação para a tripulação, levando em consideração vários aspectos relacionados aos requisitos para realizar um voo sem piloto a bordo.

Um sistema completo de uma aeronave não tripulada necessita englobar estações de controle, considerar o peso de sua carga útil (*pay load* - elementos da aeronave não necessários para o voo e pilotagem, mas que são destinados para a realização de alguma tarefa específica (DECEA, 2015)), prover um sistema de comunicação entre a estação de controle e aeronave, além de questões de manutenção e suporte, que pode envolver transporte de equipamentos. As estações de controle servem como uma base de apoio para a aeronave, que pode receber comandos ou obter referências para seu voo, ou ainda a aeronave repassar dados coletados

enquanto realiza seu voo. Dessa forma, o sistema de comunicação entre base de controle e VANT tem um importante papel na transmissão dos dados de entrada e saída, os quais podem eventualmente ser usados para realizar o lançamento ou recuperação de uma aeronave. Além disso, como o controle de altitude e estabilidade são realizados automaticamente pelo sistema eletrônico embarcado no VANT, eles devem considerar as restrições de peso que o VANT pode carregar. Por exemplo, além do peso da própria estrutura, também existem as cargas de combustível, sensores e outros equipamentos. Então cada carga que precisa ser carregada pelo VANT deve ser considerada no sistema. Apesar de toda infraestrutura que um SANT deve prover, não se pode confundir tal solução com modelos de aeronaves que são rádio-controlados por seres humanos, pois no modelo de VANT em um SANT, considera-se a autonomia de atuação da aeronave. Em (AUSTIN, 2010) são apresentadas algumas categorias de sistemas baseados em veículos aéreos:

- HALE - *hight altitude long endurance*: alcança aproximadamente 15000 m de altitude com autonomia de aproximadamente 24 horas;
- MALE - *medium altitude long endurance*: altitude entre 5000 m e 15000 m com autonomia de aproximadamente 24 horas;
- TUAV - *medium range or tatical UAV*: voos de 100 a 300 km de distância da estação de controle;
- *Close-Range UAV*: aproximadamente 100 km de distância da estação de controle;
- MUAV - *Mini UAV*: com peso entre 20 kg e 30 kg;
- MAV - *Micro UAV*: voltado para ambientes urbanos, possuem um voo mais lento ou ainda em estado estacionário;
- NAV - *Nano UAV*: usados em técnicas de *swarm* (enxame) para confusão de radar.

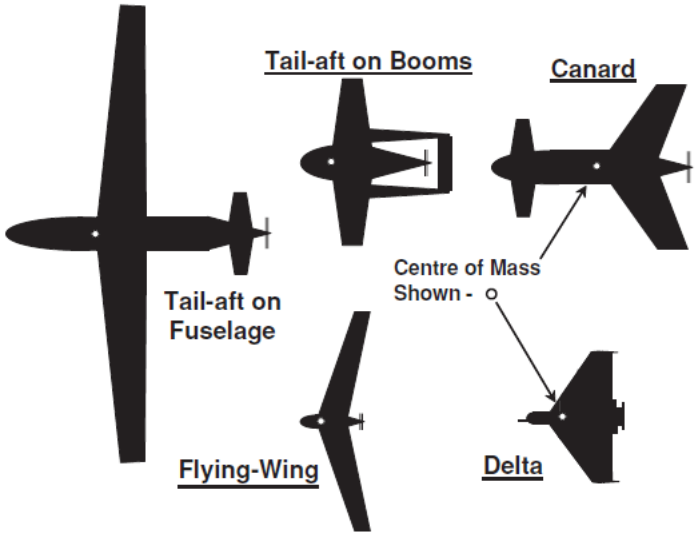
O projeto de um SANT pode ser dividido em: fase conceitual, projeto preliminar, análise de aerodinâmica e seleção do sistema. A fase conceitual é a responsável por definir a viabilidade do projeto, técnica e financeiramente. O projeto preliminar visa realizar a definição dos materiais, dimensões e componentes, além de definir o que será fabricado pela própria equipe e o que será buscado externamente e quais profissionais estarão envolvidos. Após o projeto preliminar, é feita uma análise de aerodinâmica, estrutura, ambientes e sistemas de estações de controle e demais subsistemas. Então é realizada a seleção do sistema, que irá considerar algumas características, requisitos e restrições, incluindo:

- Definição da aplicação do projeto: civil, militar, econômica, entre outras;
- Características: *pay load*, autonomia, raio de ação, velocidade, decolagem e pouso;
- Requisitos e restrições que afetam o sistema como um todo: outros elementos externos ao sistema, como pista de decolagem ou rampa de lançamento;
- Condições do ambiente: impacto do ambiente no sistema e o impacto do sistema no ambiente.

2.1.2 CLASSIFICAÇÃO DE VANT

A fuselagem de uma aeronave pode ser classificada pela forma de decolagem e aterrissagem que realiza. Quando uma aeronave faz uma ação de decolagem ou aterrissagem de forma horizontal, é dito que a configuração da fuselagem é HTOL (*Horizontal Takeoff and Landing*). A figura 1 apresenta modelos de fuselagem HTOL. Nesta categoria de fuselagem com asas fixas, quanto maior for a velocidade a força de reação aumenta. As asas sofrem efeito de arrasto, e com isso para alcançar maiores velocidades é necessário atingir espaços com menor densidade de ar, interferindo na relação de altitude/velocidade (AUSTIN, 2010).

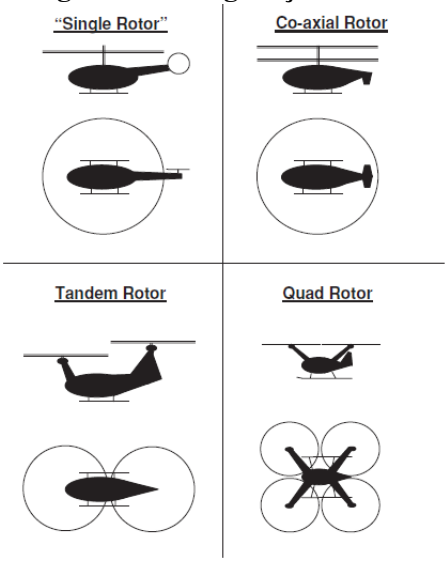
Figura 1: Configuração HTOL.



Fonte: (AUSTIN, 2010)

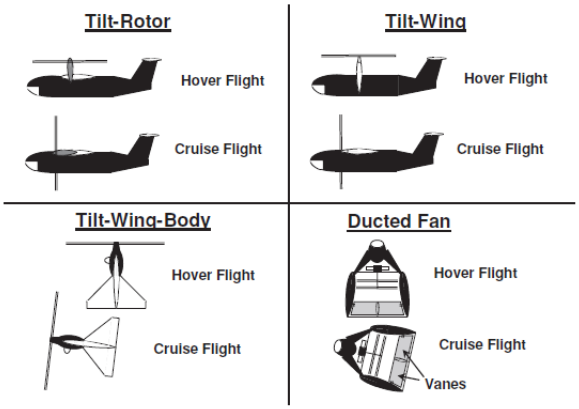
Quando uma aeronave realiza uma ação de decolagem ou aterrissagem de forma vertical, a configuração de fuselagem é denominada VTOL (*vertical takeoff and landing*). A figura 2 apresenta os modelos de fuselagem VTOL. Nesta categoria de fuselagem com asas giratórias (hélices), a aerodinâmica é mais complexa. O giro da hélice move o ar em um

Figura 2: Configuração VTOL.



Fonte: (AUSTIN, 2010)

Figura 3: Configuração híbrida.



Fonte: (AUSTIN, 2010)

caminho circular. A força de elevação produz então um arrasto que exige maior torque do motor, consequentemente existe a necessidade de maior densidade de ar. Esta configuração não tem como característica a alta velocidade e altitude alcançada pelos HTOL, entretanto possui a capacidade de se manter sustentado em um ponto especifico do espaço, tornando-se extremamente útil em ações táticas. Buscando unir as vantagens das fuselagens de ambas configurações, modelos híbridos podem trazer a conveniência de ações táticas de HTOL com a velocidade e altitude alcançadas de VTOL. A figura 3 apresenta modelos de fuselagem híbrida.

O foco deste trabalho são os veículos aéreos do tipo VTOL com múltiplos rotores. Assim, ao longo do texto o termo VANT refere-se exclusivamente a aeronaves deste tipo.

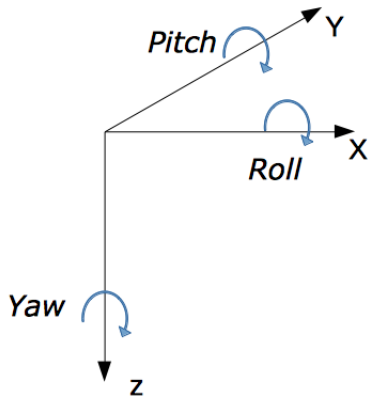
2.1.3 GRAUS DE LIBERDADE

Na robótica móvel é importante entender como será o comportamento mecânico do robô para projetar a execução de tarefas ou entender como criar sistemas que controlam tais robôs (SIEGWART et al., 2011).

O termo *Graus de Liberdade* (*degree of freedom* - DOF) busca apresentar a capacidade de movimento do robô no ambiente em que se encontra. Em Arkin (1998), os DOFs são considerados um conjunto de posições variáveis independentes, em relação à fuselagem de referência, necessário para especificar a posição de um objeto com o mundo. Em LaValle (2006), o termo DOF é usado para referenciar o número máximo de parâmetros independentes que são necessários para caracterizar uma transformação completa aplicada ao robô. Por exemplo, se o conjunto de possíveis valores para x e y formam um subconjunto de duas dimensões em $\mathbb{R}^2$, então o grau de liberdade é dois. Em Spong et al. (2006) e em Siciliano et al. (2009), é feita uma consideração para o exemplo de robôs manipuladores com braços articulados, como os presentes em linhas de montagem automotiva. Eles supõem que um robô tem n graus de liberdade se pode ser minimamente especificado por n parâmetros, de forma que o DOF é igual a dimensão do espaço de configuração. Para um robô manipulador, o número de *joints* (articulações) determinam o número de DOF neste caso. Ainda em Baginski (1998), são separados os graus de liberdade cinemáticos e geométricos para um sistema manipulador. Os graus de liberdade cinemáticos são o número de *joints* do sistema. O número de graus de liberdade geométricos depende do número de *joints*, mas não sendo mais do que seis (translação e rotação em $\mathbb{R}^3$).

Um robô terrestre pode fazer uso de rodas para se mover, e a disposição e restrições dessas rodas são o que possibilitam maior ou menor agilidade em seus movimentos. Devido a ele estar apenas no chão, um robô com estas características pode mover-se em duas dimensões e possui apenas um movimento de deslocamento angular (SIEGWART; NOURBAKHSH, 2004). Quando consideramos os VANT como robôs móveis, eles apresentam um maior nível de mobilidade. O movimento de uma aeronave é tipicamente expressa em seis tipos de movimentos, também chamados de seis graus de liberdade (6DOF) (MANSSON; STENBERG, 2014). Desde que o VANT esteja em um ponto fixo no espaço, ele pode realizar três movimento de translação através dos eixos x, y e z, e realizar movimentos rotacionais nos ângulos ϕ, θ e ψ, conhecidos como *roll, pitch* e *yaw*. A figura 4 mostra estes movimentos que um VANT pode realizar. Em um VANT com múltiplos propulsores, a sua distribuição na estrutura e o modo de operação de cada propulsor influenciará em uma maior ou menor complexidade no balanceamento dos seus movimentos para obter estabilidade (MANSSON; STENBERG, 2014).

Figura 4: Seis Graus de Liberdade (6DOF).



Fonte: Autoria própria.

A seguir são apresentadas algumas topologias de VANT, em que o controle de movimentos angulares e de translação são fundamentais para o controle de estabilidade em voo.

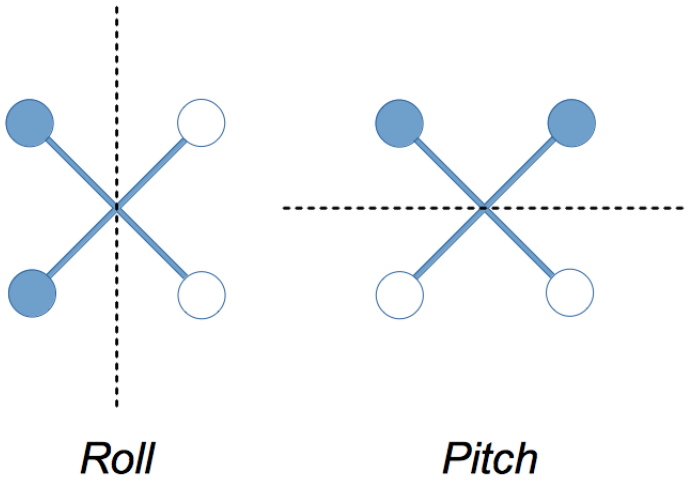
2.1.4 TOPOLOGIAS DE VANT

Diferentemente dos robôs terrestres, os VANT multirotores podem realizar movimentos de translação e rotação em mais eixos. Enquanto um robô terrestre pode rotacionar apenas em z (*yaw*), o VANT pode realizar a rotação nos eixos x (*roll*) e y (*pitch*). Então, a distribuição da força necessária para os movimentos nos três eixos será de forma balanceada. Em algumas topologias, as distribuição dos propulsores são diferentes. Considerando a topologia de um quadrotor na configuração "X", os movimentos de *pitch* e *roll* fazem uso da mesma quantidade de propulsores para realizar o movimento, como pode ser visto na figura 5. O mesmo acontece com um quadrotor "+" (*plus*), como pode ser visto na figura 6. Contudo, pode-se perceber que na configuração "X" são necessários mais rotores para fazer o mesmo movimento.

Outra distribuição com mais propulsores é a dos hexacópteros, os quais possuem 6 propulsores distribuídos pela fuselagem do VANT. A figura 7 mostra esta topologia.

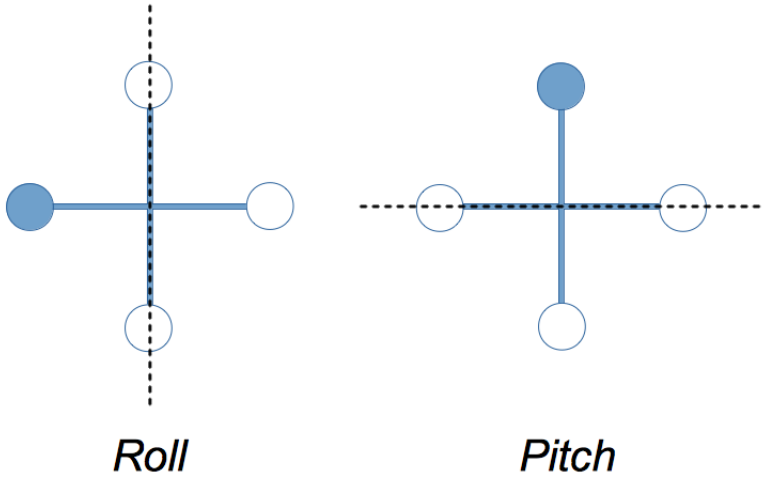
Nas topologias de quadrotor "X", "+" e hexacóptero, o propulsores são fixados na fuselagem do VANT e a movimentação será realizada a partir da definição de forças aplicadas em cada propulsor (SANDY, 2017). Outras topologias apresentam uma proposta de movimentação do propulsor em relação a fuselagem do VANT. Um exemplo disto é a topologia "X4S" (MANSSON; STENBERG, 2014), onde cada braço de sustentação dos propulsores possuem um servo motor que adiciona opções de atuação para algoritmos de controle, entretanto

Figura 5: Quadrotor topologia "X".



Fonte: Autoria própria.

Figura 6: Quadrotor topologia "+".

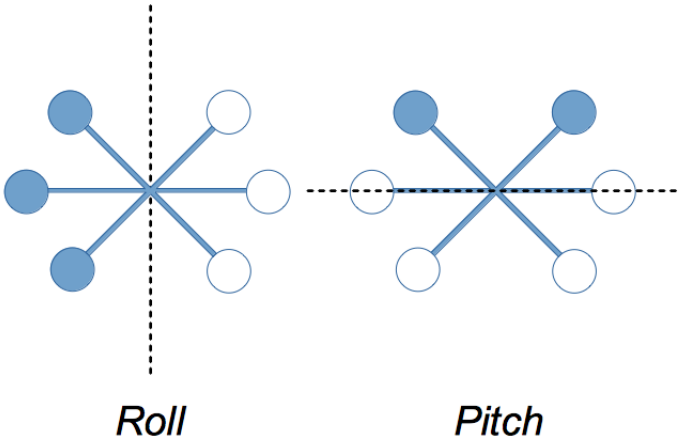


Fonte: Autoria própria.

aumenta-se a complexidade. A figura 8 mostra esta topologia.

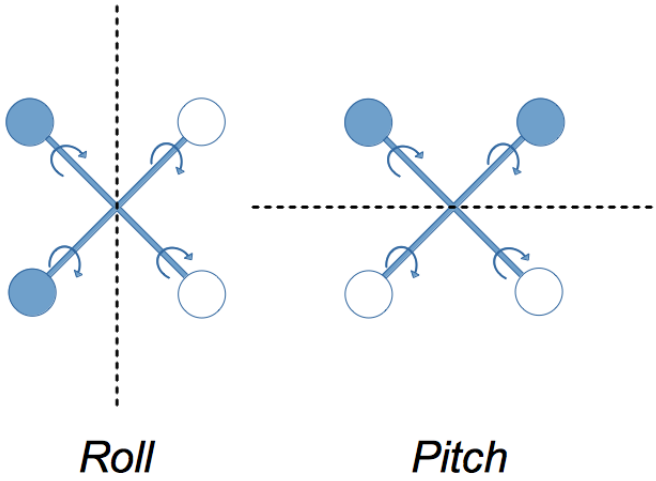
Em (MANSSON; STENBERG, 2014) outras topologias são apresentadas, como a I8, V8, Y3, T3 e Y6. A topologia I8 assemelha-se com a distribuição de propulsores de "X" e do hexacóptero, porém com oito propulsores distribuídos no VANT. Em uma topologia em formado de "V", a proposta V8 apresenta-se como uma solução para a atividade de fotografias aéreas devido ao seu ângulo de abertura e capacidade de carga como a I8, entretanto é mais complexo na manutenção da estabilidade de voo. Já as topologias Y3 e T3 apresentam-se bastante semelhantes, com dois propulsores à frente da aeronave e um propulsor na parte traseira, sendo que o braço de sustentação deste possui um servo motor, dando maior possibilidade de movimentos. A topologia Y6 traz uma distribuição uniforme de 3 braços na aeronave, porém em cada uma de suas extremidades são acoplados dois propulsores, sendo um sobre o outro e

Figura 7: Topologia Hexacóptero.



Fonte: Autoria própria.

Figura 8: Topologia "X4S".

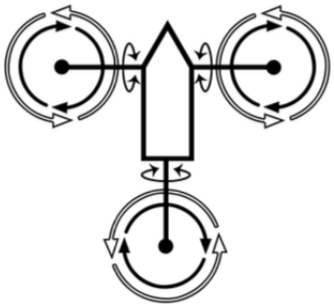


Fonte: Autoria própria.

girando em direções opostas. Uma proposta de fusão entre os modelos Y6 e X4S é apresentada em (MANSSON; STENBERG, 2014), onde a estabilidade dos duplos propulsores do Y6 seriam somados a flexibilidade de movimentos dos servo motores dos braços de sustentação do X4S. A figura 9 apresenta a topologia proposta.

Desse modo, a utilização de VANT com a configuração VTOL torna-se adequada para o processo de exploração de ambientes desconhecidos (e por vezes desestruturados). Caso o robô encontre um obstáculo, uma ou mais trajetórias de deslocamento podem ser calculadas considerando a alteração da posição de altura que ele se encontra, bem como variar sua posição angular para a exploração de adjacências no seu entorno.

Figura 9: Topologia de fusão Y6 e X4S.



Fonte: (MANSSON; STENBERG, 2014).

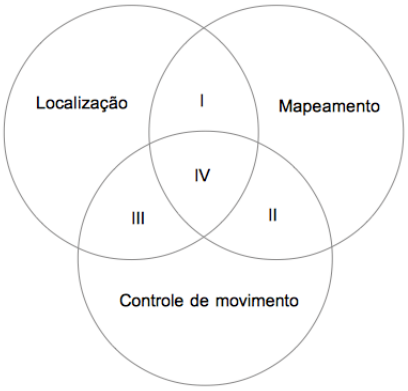
2.2 EXPLORAÇÃO E MAPEAMENTO DE AMBIENTES

A atividade de mapear e construir um modelo de um ambiente necessita que robôs móveis tenham a capacidade de realizar eficientemente a exploração, minimizando o tempo necessário para que essa exploração seja feita de forma completa (BURGARD et al., 2000). Dessa forma, é essencial que cada robô saiba quais áreas ainda precisam ser exploradas e, para isso, eles devem construir um mapa global para poder planejar seus caminhos e coordenar suas ações (BURGARD et al., 2005).

O trabalho de Makarenko et al. (2002) aborda a combinação das atividades de localização, mapeamento e de controle de movimento, como pode ser visto na figura 10. A região I representa a integração de localização e mapeamento implementados por muitos algoritmos de SLAM. A região II representa a atividade de exploração, que compreende as ações de integração do controle de movimento e mapeamento. A região III integra localização e controle de movimento, enquanto a região IV é a integração completa dos três componentes. Assim, para que um robô possa realizar a exploração de um ambiente, ele deve ser capaz de localizar-se no ambiente, mapear o ambiente, determinar qual será seu próximo destino e como se deslocará até lá (MAKARENKO et al., 2002).

A atividade de exploração é uma ação necessária em diversas soluções como, por exemplo, exploração planetária, atividades de busca e resgate, e até de limpeza de ambientes por robôs móveis, nos quais a cobertura total de terrenos é uma parte importante de sua missão (BURGARD et al., 2005). Estratégias de exploração são feitas de diversas formas, as quais são abordadas na seção 3.2.

Figura 10: Diagrama integrando localização, mapeamento e controle de movimento.



Fonte: Baseado no trabalho de Makarenko et al. (2002).

2.2.1 EXPLORAÇÃO DE AMBIENTES

O objetivo da atividade de exploração é aumentar o conhecimento do robô a respeito do seu ambiente, a partir da seleção de ações de controle apropriadas, que o leve a visitar locais inicialmente desconhecidos (SHADE, 2011). Para isso, é essencial que os robôs mantenham o conhecimento a respeito das áreas do ambiente onde já foram exploradas. Além disso, os robôs devem construir um mapa global em tempo de exploração, de forma que possam planejar suas trajetórias e coordenar suas ações (STACHNISS, 2009). Deve-se assumir que, tanto o mapa da área explorada quanto as posições dos robôs, podem ser comunicadas entre os robôs. A questão central é como coordenar os robôs de forma eficiente cobrindo todo o ambiente.

Diversas técnicas de exploração são propostas na literatura. A exploração topológica descreve um método de exploração na qual um robô constrói um grafo conectado do ambiente (SHADE, 2011). O trabalho de Kuipers e Byun (1991) apresenta uma abordagem topológica onde locais distintos no mundo são identificados e permitem que o robô diferencie áreas já exploradas das não exploradas.

O uso de métodos potenciais possibilita o uso de gradientes para definir trajetórias (SHADE, 2011). Ao assinar o ponto de origem com um potencial alto e a posição destino com um potencial baixo, o caminho entre esses dois pontos é definido pelo gradiente descendente através do campo resultante, assumindo que condições de fronteiras apropriadas tenham sido definidas.

Uma das questões centrais da exploração é colocada como: “Dado o conhecimento sobre o mundo, onde deve-se ir para obter o maior ganho de nova informação possível?”(YAMAUCHI, 1997). A exploração baseada em fronteiras busca com que o robô se mova até um local limite entre um espaço livre conhecido e um espaço desconhecido, onde ele

poderá usar seus sensores para captar informações além da fronteira, e então expandir o mapa. O algoritmo de ganho de informação busca um local de fronteira onde o robô conseguirá obter maior número de informações, ou seja, maior descoberta de espaços desconhecidos. Contudo, esta abordagem pode fazer com que o robô desloque-se por longas distâncias para alcançar a região a ser explorada (SHADE, 2011). Para González-Baños e Latombe (2002), a proposta de ganho de informações com pesos busca balancear o ganho de informação e a distância entre os pontos alvos.

2.2.2 MAPEAMENTO DE AMBIENTES

O desenvolvimento de robôs móveis para aplicações em ambientes internos necessita de um mapeamento robusto e técnicas de percepção que possibilitem a navegação autônoma em ambientes complexos. O problema da representação de mapas do ambiente com o robô em movimento está também na representação da possível posição do robô e a fidelidade desta posição irá interferir na fidelidade do mapa (SIEGWART; NOURBAKHSH, 2004). A construção de um sistema que faça este mapeamento deve considerar os seguintes quesitos (SAEEDI et al., 2016):

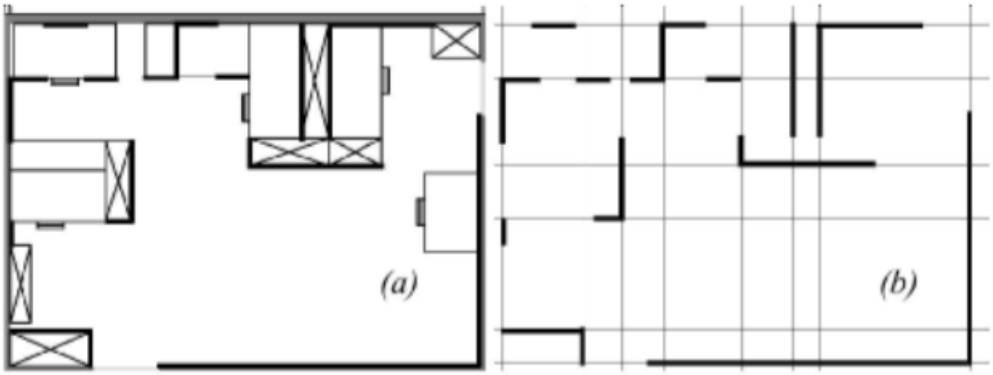
- Sensores;
- Processamento de dados;
- Representação do mapa.

A partir do momento em que os sensores capturam dados, o processamento pode iniciar. Alguns dos processamentos mais comuns incluem filtragem, suavização e técnicas de inteligência artificial para a construção dos mapas (SAEEDI et al., 2016). Neste processamento, diferentes tipos de incertezas são tratadas, para então finalmente ser feita a representação no mapa, o qual é um modelo que retrata o ambiente a partir da trajetória que o robô realizou. O processamento dos dados e a geração do mapa interage um com o outro, fornecendo resultados mais precisos.

O mapeamento contínuo é um método para a decomposição exata do ambiente, onde a posição das características do ambiente são registradas precisamente, entretanto pode haver uma explosão computacional na representação destes dados (SIEGWART et al., 2011). Um mapa geométrico pode representar a localização física de um objeto sem fazer referências a texturas, cores ou qualquer outra característica do ambiente além da localização. A partir desta simplificação pode haver redução do uso de memória. Um exemplo de aplicação que faz apenas

extração geométrica é a identificação de linhas contínuas (SIEGWART et al., 2011). Ainda que haja uma redução de características no mapeamento, a característica contínua ainda pode trazer custo computacional. A figura 11 apresenta um exemplo de ambiente mapeado de forma contínua.

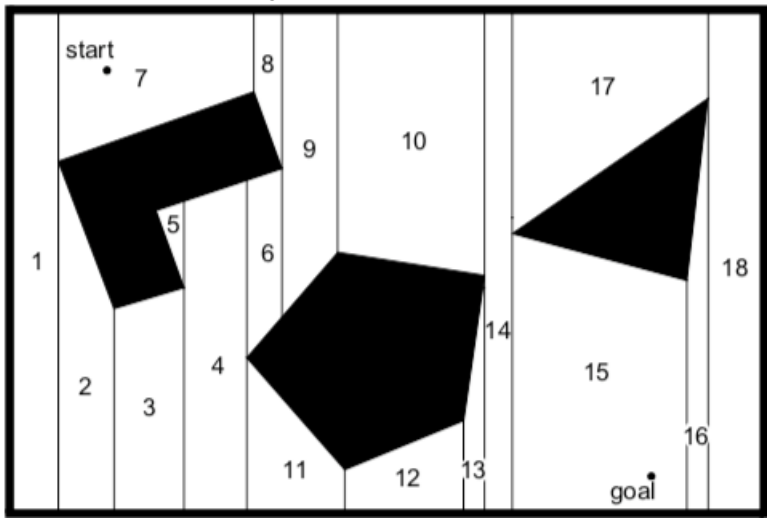
Figura 11: Representação contínua: (a) mapa real, (b) representação com conjunto de linhas infinitas.



Fonte: (SIEGWART et al., 2011).

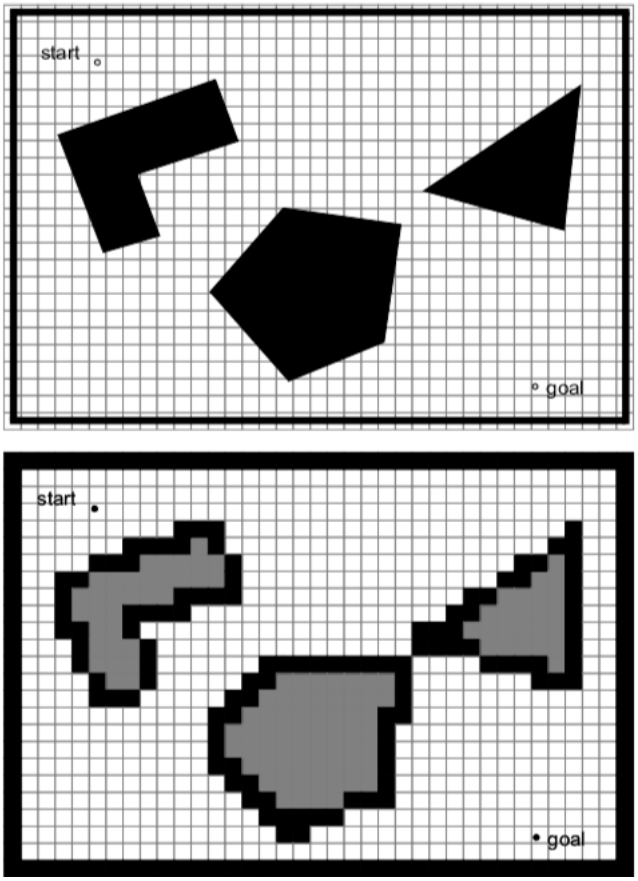
Algumas estratégias de decomposição podem auxiliar na identificação de informações relevantes e minimização de dados representados. A decomposição por células faz a seleção dos limites entre as células discretas baseando-se na geometria (como pode ser visto na figura 12). Já a decomposição fixa transforma o ambiente contínuo real em uma aproximação discretizada para o mapa (SIEGWART et al., 2011), como pode ser visto na figura 13. Em DHIMAN et al. (2015) a geração de mapas é particionada em três partes: métricos, topológicos e híbridos. Elas são detalhadas a seguir.

Figura 12: Decomposição por células (SIEGWART et al., 2011).



Fonte: (SIEGWART et al., 2011).

Figura 13: Decomposição fixa (SIEGWART et al., 2011).



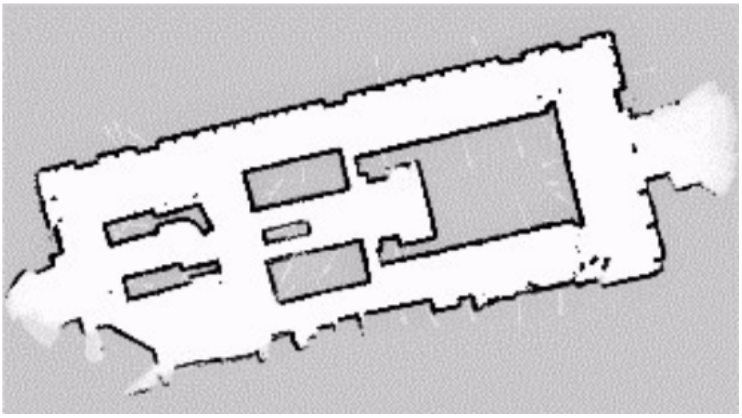
Fonte: (SIEGWART et al., 2011).

2.2.2.1 MAPAS MÉTRICOS

Os mapas métricos buscam extrair as propriedades geométricas e características do ambiente, e então representá-las em forma de grade (*grid map*) ou ainda em mapas de características (*feature map*) (STACHNISS, 2009). Normalmente, mapas métricos são probabilísticos, e estabelecem métodos para modelar o ruído e seus efeitos na modelagem do ambiente (DHIMAN et al., 2015). Os *grid maps* ou mapas de grade, representam o espaço em uma matriz de células. *Occupancy grid* ou grades de ocupação são um dos mais populares mapas de grades, em especial em soluções de algoritmos de SLAM 2D (ELFES, 1990). Nesta solução cada célula representa uma pequena seção retangular do mundo. A célula é modelada com uma variável binária que indica se um objeto existe naquele espaço que representa. *Occupancy grid* podem ser também estendido para modelagem tridimensional (3D), o qual é conhecido como *Volumetric Pixel (VOXEL) maps*. Os *Feature maps* ou mapas de características, também conhecidos como *landmark maps*, representam o mundo com posições globais das características extraídas do ambiente. Normalmente a posição da característica mapeada é acompanhada de sua assinatura, que é um identificador único que a caracteriza. Técnicas

voltadas para *feature maps* como *Scale-Invariant Feature Transformm* (SIFT) e *Speeded-Up Robust Features* (SURF) foram introduzidas em (FRAUNDORFER; SCARAMUZZA, 2012). A figura 14 apresenta um exemplo de mapa métrico baseado em grades de ocupação.

Figura 14: Exemplo de mapa métrico.



Fonte: (SIEGWART et al., 2011).

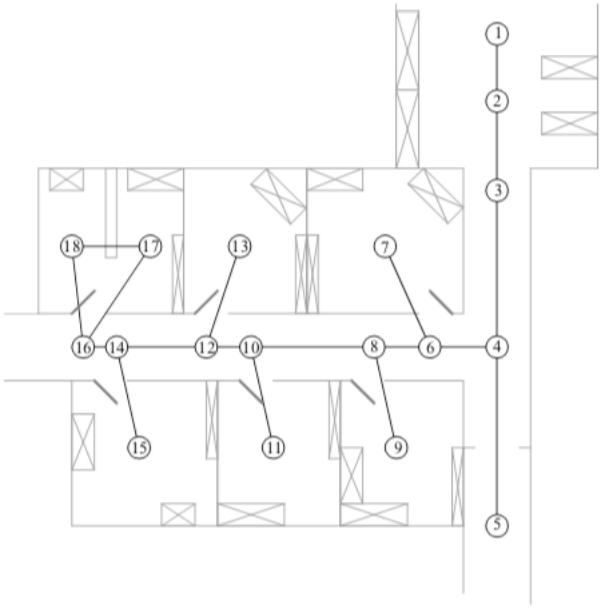
2.2.2.2 MAPAS TOPOLÓGICOS

Os mapas topológicos representam o ambiente de forma abstrata (KUIPERS; BYUN, 1991) em uma estrutura de grafos, onde os vértices representam locais e objetos de interesse, enquanto as arestas representam o relacionamento espacial ou o caminho entre os vértices. Além disso, para fornecer uma representação mais compacta do ambiente que os mapas métricos, os mapas topológicos apresentam uma compreensão simbólica de alto nível para tarefas de planejamento e navegação. Enquanto os mapas métricos podem acumular erros de odometria, os mapas topológicos são construídos sem preocupação de aspectos métricos: os erros de odometria são acumulados entre os vértices do grafo, não necessariamente sendo acumulados pelo mapa global (DHIMAN et al., 2015). A abordagem topológica para percepção ou autonomia também podem ser interpretadas como um mapeamento simbólico (BEESON et al., 2010), onde é fornecido uma representação concisa para alternativas estruturais. A figura 15 apresenta um exemplo de mapa topológico.

2.2.2.3 MAPAS HÍBRIDOS

A geração dos mapas híbridos é feita combinando as vantagens dos mapeamentos métricos e topológicos Tomatis et al. (2003). O mapeamento topológico é aplicado para uma visão global do ambiente, enquanto o mapeamento métrico é aplicado em áreas pequenas, de forma que é reduzida a complexidade computacional durante o processamento das informações métricas. Assim, uma forma de mapa híbrido é fazer uso de um mapa topológico em que os

Figura 15: Exemplo de mapa topológico.



Fonte: (SIEGWART et al., 2011).

vértices são utilizados para representar pequenos mapas métricos, e as arestas entre os vértices representam o caminho do ponto central de um mapa métrico até o ponto central do próximo mapa métrico (DHIMAN et al., 2015).

2.2.2.4 MAPAS SEMÂNTICOS E DE APARÊNCIA

Saeedi et al. (2016) adiciona mais duas classificações de mapas: mapas semânticos e mapas de aparência. Mapas semânticos (*semantic maps*) contém funcionalidades e relacionamentos de objetos do ambiente, sendo um formato mais abstrato (NÜCHTER; HERTZBERG, 2008). Esta proposta de mapa é útil para processamento em alto-nível e comportamentos orientados a objetivos que requerem reações em tempo real sobre ambiguidades espacias e de perspectiva. Existe uma similaridade com mapas topológicos, porém os mapas semânticos trazem maior detalhamento de informação sobre os objetos e locais. Os mapas de aparência (*appearance maps*) são voltados para sistemas de visão e incluem diferentes visualizações associadas em um grafo não direcionado com pesos. Cada nó do grafo é uma imagem e os arcos unem as imagens suficientemente similares (ERINC; CARPIN, 2014).

2.3 DISCUSSÃO

Nesse capítulo buscou-se apresentar os principais conceitos existentes relacionados com o tema desta tese de doutorado. O crescente uso de VANT para diversas situações nos faz

refletir sobre a necessidade de considerar diversos fatores quando se propõe um sistema com robôs aéreos: além de ser constituído por VANT também necessita de estruturas como bases de controle em solo, estudo de arquiteturas e sensores a serem empregados entre outros.

Dependendo do contexto, a escolha de uma determinada fuselagem de um VANT em relação a outra pode ser determinante para a eficiência e eficácia da execução das missões existentes. Para ambientes amplos e abertos, a fuselagem HTOL com asas fixas apresenta-se como uma solução conveniente, entretanto quando foca-se em ambientes internos, a fuselagem VTOL é a mais adequada, pois não necessita de pista de decolagem para tomar impulso e ganha sustentação (AUSTIN, 2010). Dentro das fuselagens VTOL, diversas topologias são encontradas. Os quadrotores são bastante empregados e possuem uma estrutura simétrica e de complexidade não tão elevada quanto outras topologias, como por exemplo V8, Y3 e Y6 (MANSSON; STENBERG, 2014). Assim, no método de exploração e mapeamento proposto neste trabalho, são considerados os VANT do tipo VTOL com quatro rotores (quadrotores) como a topologia adotada.

A tarefa de exploração e mapeamento pode adotar diversas estratégias. As estratégias de exploração do ambiente baseado em fronteiras desconhecidas possibilita o avanço do mapeamento, em que trabalhos que ponderam tanto o ganho de conhecimento quanto a distância dos pontos alvos de exploração evitam deslocamentos de robôs para pontos distantes. O uso de mapas métricos faz a discretização do ambiente. O mapa de grade de ocupação é uma das formas mais populares dessa técnica (SIEGWART et al., 2011). Já os mapas topológicos buscam representar o ambiente em uma estrutura de grafos, nos quais os vértices representam pontos específicos no espaço e as arestas o caminho entre eles. Os mapas métricos têm a vantagem de poder fazer uma representação gráfica mais detalhada de um ambiente, porém pode haver alto uso de memória e processamento. O mapeamento topológico possui uma estrutura mais leve para o processador. Soluções híbridas buscam unir as vantagens de cada uma das formas de representação de mapas, onde o mapa topológico faz a representação global do espaço, enquanto o mapa métrico faz a representação local do ambiente.

O mapa gerado pela abordagem proposta nesta tese de doutorado assemelha-se com a proposta de mapas híbridos: os hexágonos são vértices e as adjacências são as arestas que ligam estes vértices. A leitura do espaço 3D gera uma visão do ambiente explorado, entretanto essa informação métrica não é utilizada pelos robôs para seu controle, apenas é voltado para humanos terem uma compreensão do local que está sendo explorado pelos múltiplos VANT.

3 TRABALHOS RELACIONADOS

Neste capítulo serão apresentados os trabalhos relacionados ao trabalho de pesquisa desenvolvido. O foco desta tese de doutorado é a atividade de exploração e mapeamento, entretanto algumas soluções de trabalhos relacionados fazem o mapeamento dentro de técnicas de SLAM. Dessa forma, detalhes sobre SLAM podem ser visto no apêndice A, no qual é exposto como algumas técnicas de SLAM definem a representação dos mapas gerados. Aqui são apresentadas soluções de mapeamento de ambientes internos utilizando múltiplos robôs que podem apresentar aspectos homogêneos e heterogêneos, integrando diversas formas de processamento das leituras para representá-las. Por fim, diversas abordagens de exploração de ambientes com múltiplos robôs são apresentadas.

3.1 MAPEAMENTO DE AMBIENTES USANDO MÚLTIPLOS ROBÔS

Para muitas aplicações, o uso de múltiplos robôs para as atividades de exploração e busca fornece melhores resultados, seja com uma tarefa sendo realizada em menor tempo do que a realizada por apenas um robô ou ainda pelo aumento da redundância que reflete em maior tolerância a falhas (BURGARD et al., 2000).

3.1.1 SOLUÇÕES HOMOGÊNEAS

Uma solução baseada em filtro de partículas é apresentada em Howard (2006) para SLAM utilizando múltiplos robôs. Em uma abordagem de mapeamento *on-line*, cada robô gera seu próprio mapa usando métodos de filtro de partícula *Rao-Blackwellised*. É considerado que durante o tempo de vida da operação, cada robô irá observar outros robôs envolvidos, assumindo-se assim que eles conseguem identificar uns aos outros. Ao se identificarem, eles tem acesso à posição relativa um do outro, e as leituras efetuadas por um robô são incluídas nas leituras do outro. Os robôs usados são terrestres.

Em Howard (2004) é apresentada uma abordagem de SLAM para espaços de

duas dimensões usando representações variadas de mapas planos (*manifold representations*). Assume-se que os dados de todos os robôs são enviados para uma localização central e então processados. Uma técnica de estimação máxima da vizinhança (*Maximum Likelihood Estimation - MLE*) é adaptada para definir um conjunto de poses projetadas que o robô pode ter. Esta variação é discretizada em um conjunto de caminhos sobrepostos. Cada caminho possui um sistema de coordenadas locais planas e de extensão finita (sem ocorrência de *loops*). Um conjunto de relações de poses é definido entre os caminhos. O mapa local será o conjunto de caminhos onde todos estes caminhos encaixam-se juntos. Robôs são terrestres.

Em Saeedi et al. (2011) é apresentada uma fusão de mapas baseados em redes neurais. Cada robô gera um mapa de grade de ocupação usando EKF e sensores lasers. Estes mapas são fundidos para gerar um mapa global consistente. Cada mapa de grade de ocupação de cada robô é pré-processado usando segmentação. Cada segmento alimenta um algoritmo de mapas auto organizados para clusterização. Dado que o número de *clusters* é muito menor que o número de células no mapa, o processamento de *clusters* requer menos tempo durante a fusão dos mapas. Para a contagem de incertezas das células do mapa de grade de ocupação, estas células com maior probabilidade de serem obstáculos são usados mais frequentemente como entrada de treinamento para o algoritmo de mapas auto organizados. O mapa do mundo é abstraído como um arranjo de pontos de *clusters* por este algoritmo. Robôs são terrestres.

Outras abordagens baseiam-se *Graph SLAM*. É o caso da proposta de Olson et al. (2013), onde é demonstrado um sistema de mapeamento centralizado para um time de 14 robôs terrestres. Cada robô gera periodicamente um pequeno mapa do seu entorno e transmite a uma estação de controle. Esta estação funde este pequeno mapa realizando uma inferência em um grafo de fator. Cada pequeno mapa atua como um nó no grafo. Os arcos são gerados usando odometria, IMU e leituras de sensores.

O trabalho de Williams et al. (2014) faz uma modificação no algoritmo PTAM para múltiplos agentes usando câmera monocular. A exploração do ambiente é feita cooperativamente com o armazenamento de pontos de interesse. A definição da exploração é feita via leilão, onde cada lance é a distância linear de cada VANT para o ponto a ser explorado. A menor distância vence o leilão.

No trabalho de Loianno et al. (2015), uma adaptação do PTAM (PTAMM) faz uso de dados advindos de sensores RGB-D, IMU e infra-vermelho (IR) de dois VANT que estão no ambiente. É feita a localização e mapeamento utilizando o sensor RGB-D. Uma característica deste trabalho é a decomposição do problema de SLAM 3D em um SLAM monocular com representação esparsa.

3.1.2 SOLUÇÕES HETEROGÊNEAS

Em Mahendran et al. (2013), é realizado o mapeamento colaborativos do espaço com VANT e VTNT, modelando mapas complementares. Enquanto o VTNT realiza o mapeamento 2D da área, o VANT faz o mapeamento 3D de objetos ortogonais no ambiente, identificando bordas de portas, mesas, janelas e demais elementos com estruturas retas.

No trabalho de Michael et al. (2012), é apresentado uma aplicação prática onde é feito o mapeamento de áreas atingidas por terremotos, em uma operação semi-autônoma: o VTNT é controlado remotamente, mas quando se depara com um obstáculo que não pode transpor, o VANT, de forma autônoma, realiza o mapeamento da área. A execução de um SLAM 3D é feito pelo VANT através de sensores RGB-Ds, e pelo VTNT através de um sensor laser.

Em Dewan et al. (2013), o VANT implementa um PTAM com base nas leituras de um sensor sonar, enquanto o VTNT executa um Visual SLAM alimentado por sensores laser e RGB-D. O objetivo é a exploração heterogênea usando programação inteira. Os dados gerados pelo VANT via PTAM são enviados para o VTNT e integrados ao Visual SLAM.

3.2 EXPLORAÇÃO DE AMBIENTES COM MÚLTIPLOS ROBÔS

A utilização de múltiplos robôs no processo de exploração de um ambiente pode trazer vantagens no sentido de obter resultados a partir de alguma premissa. Robôs cooperativos podem realizar tarefas de forma mais rápida que um único robô. A redundância gerada por vários robôs no ambiente reflete em maior tolerância a falhas do que soluções que fazem uso de apenas um robô, além da possibilidade de fusão de dados sobrescritos, que podem compensar as incertezas das leituras dos sensores. Entretanto, quando robôs trabalham em times, existem riscos de interferências entre eles, além de que quanto maior o número de robôs em um mesmo ambiente, maior a importância de fornecer mecanismos de prevenção de colisões entre os robôs envolvidos (BURGARD et al., 2005).

Algumas abordagens de exploração com múltiplos robôs são baseadas em princípios de mercados. É o caso do trabalho de Simmons et al. (2000), onde é proposta a coordenação da exploração combinando um processamento distribuído com uma tomada de decisão global. Cada robô prepara “sugestões”, que descrevem sua estimativa de ganho de informações (conhecimento do mapa) e custo de deslocamento para vários locais. Uma central recebe as “sugestões” e define as tarefas na tentativa de maximizar a utilidade geral dos robôs, enquanto busca minimizar a sobreposição de áreas já cobertas pelos robôs. Assim, a central executiva pode definir um local de exploração diferente de uma localização de preferência do robô. Cada

robô submete novas sugestões quando seus mapas são atualizados. Já o trabalho de Zhang et al. (2020) apresenta uma proposta de exploração a partir de árvores aleatórias de exploração rápida, a qual é uma abordagem probabilística, aplicando uma estratégia de alocação de tarefas baseada em mercados. Quando o robô se move para seu destino de exploração, o tamanho da trajetória torna-se uma restrição para a otimização do processo exploratório.

O trabalho de Dai et al. (2020) apresenta uma proposta de coordenação da exploração de múltiplos robôs através de uma estratégia baseada em lógica dos predicados. A estratégia define os ambientes com máxima utilidade para o robô que possui propriedades de exploração ótima, e realiza a exploração cooperativa considerando a capacidade do ambiente.

A abordagem feita por Burgard et al. (2005) faz a exploração por uso de fronteiras desconhecidas utilizando mapas de grades de ocupação. Nessa estratégia, o conceito de "fronteiras úteis" pode incluir fatores relacionados a proximidade de comunicação, fazendo com que os robôs explorem menos os espaços que estão mais distantes da área de comunicação. Essa técnica possui uma aproximação com a proposta de método de exploração nesta tese de doutorado que também busca identificar espaços acessíveis, porém desconhecidos, mas mapeando de forma topológica, e não métrica como é o trabalho de Burgard et al. (2005). Outras técnicas buscam criar uma "zona de conforto" para manter os robôs uns próximos dos outros (VAZQUEZ; MALCOLM, 2004).

O trabalho de Hoog et al. (2010) propõe uma exploração baseada em papeis para aumentar a conectividade no processo exploratório. Dois papeis são definidos para os robôs: explorador e retransmissor. Os exploradores procuram explorar o ambiente nos pontos mais distantes. Para comunicar seus resultados, eles retornam periodicamente a pontos de encontro previamente estabelecidos, onde repassam seus conhecimentos para os retransmissores. Os retransmissores carregam as informações entre os exploradores e o comando central, através de acessos periódicos aos pontos de encontro. Se um retransmissor descobre alguma informação sobre o ambiente, esta é então adicionada ao conjunto de conhecimento do time de robôs. Assim, é formada uma hierarquia de árvore, com um robô em cada nó, a estação base é a raiz e os exploradores são as folhas. A árvore pode ter vários retransmissores entre a estação base e um explorador.

Em Singh e Fujimura (1993) é apresentada uma abordagem descentralizada para robôs heterogêneos. Sempre que um robô descobre uma abertura para um espaço que não pode explorar, é selecionado outro robô o qual pode realizar a tarefa de exploração.

Os trabalhos apresentados até aqui são focados na exploração de ambientes com múltiplos robôs terrestres. O trabalho de Cesare et al. (2015) apresenta uma proposta de

exploração com múltiplos VANT que possuem quatro estados possíveis: explorar, encontrar, sacrificar e transmitir. Quando o VANT está no estado explorar, ele busca em um algoritmo uma tarefa a realizar. Quando está no estado encontrar, ele retorna a uma localização prévia com os membros do time. O estado sacrificar será acionado caso o VANT não possua mais bateria suficiente para retornar a estação base, de forma que ele continua a exploração até esgotar toda a carga que possui. Se o VANT está com a carga da bateria baixa e deseja recarregar, ele pode aterrizar e atuar como um transmissor, que é o último estado possível. O mapeamento é feito em um mapa métrico em 3D utilizando OctoMap, porém o planejamento de rotas considera um espaço de navegação 2D.

3.3 DISCUSSÃO

Diversos trabalhos propõem soluções para a exploração e mapeamento de ambientes internos. Os algoritmos de SLAM fazem uso de diversas estruturas para construírem os mapas, como grades de ocupação, *OcTrees-OctoMap*, estrutura de grafos, entre outros. Entretanto, não é o foco do SLAM garantir a exploração completa do ambiente, por mais que alguns trabalhos definam pontos a serem mapeados e estabeleçam algum critério para definir qual robô será responsável por mapeá-lo. Muitos trabalhos são focados no mapeamento métrico, o que pode gerar alto consumo de memória e processamento. A tabela 1 apresenta um resumo dos trabalhos relacionados ao mapeamento de ambientes com múltiplos robôs.

Tabela 1: Relacionamento de trabalho voltados ao mapeamento de ambientes.

Trabalho	Homogêneos	Tipo robôs	Mapa	Construção
Howard (2006)	S	VTNT	Métrico	Distribuída
Howard (2004)	S	VTNT	Métrico	Centralizada
Saeedi et al. (2011)	S	VTNT	Métrico	Distribuída
Olson et al. (2013)	S	VTNT	Topológico	Centralizada
Williams et al. (2014)	S	VANT	Métrico	Centralizada
Loianno et al. (2015)	S	VANT	Métrico	Distribuída
Mahendran et al. (2013)	N	VANT e VTNT	Métrico	Distribuída
Michael et al. (2012)	N	VANT e VTNT	Métrico	Distribuída
Dewan et al. (2013)	N	VANT e VTNT	Métrico	Distribuída
Este trabalho	S	VANT	Topológico	Centralizado

Fonte: Autoria própria.

Os trabalhos que focam na exploração de ambientes propõem soluções baseadas em sugestões do próprio robô para uma base central de locais que pode explorar, baseando-se no seu conhecimento do ambiente (SIMMONS et al., 2000), aplicando estratégias de mercado para definir as ações exploratórias (ZHANG et al., 2020). Estratégias utilizando a lógica de

predicados e a combinação de escolher o robô mais preparado para explorar um ambiente é apresentado por Dai et al. (2020). Outras soluções propõem que os robôs explorem os espaços desconhecidos mais próximos possíveis de uma área de comunicação, com a representação em mapas métricos (BURGARD et al., 2005). Essa proposta necessita que um perímetro seja estabelecido para que o mapa métrico possa representar o desconhecido também. Outro viés de atividade de exploração foca na conectividade entre os robôs durante o processo (HOOG et al., 2010). Todas estas propostas são voltadas para robôs terrestres. A tabela 2 apresenta um resumo dos trabalhos relacionados à atividade de exploração do ambiente com múltiplos robôs.

Tabela 2: Relacionamento de trabalho voltados a exploração de ambientes.

Trabalho	Estratégia de exploração	Foco
Simmons et al. (2000)	estratégias de mercado	Cobertura
Zhang et al. (2020)	estratégias de mercado	Cobertura
Dai et al. (2020)	lógica de predicados	Cobertura
Burgard et al. (2005)	fronteiras úteis	Comunicação
Vazquez e Malcolm (2004)	fronteiras úteis	Comunicação
Hoog et al. (2010)	papeis no processo de exploração	Comunicação
Singh e Fujimura (1993)	uso de robôs heterogêneos	Cobertura
Cesare et al. (2015)	papeis no processo de exploração	Comunicação
Este trabalho	comportamento bioinspirado em abelhas	Cobertura

Fonte: Autoria própria.

O trabalho de Cesare et al. (2015) faz uso de múltiplos VANT para a exploração de ambientes internos, com definição de estados de atuação. Uma representação 3D é feita utilizando OctoMap, porém o planejamento de trajetórias dos VANT considera apenas coordenadas em 2D.

Dessa forma, esta tese de doutorado propõe um método de mapeamento de ambientes internos com múltiplos VANT utilizando uma estrutura topológica para a construção do mapa do espaço explorado. Esta estrutura possibilita que a identificação dos espaços considerem o espaço em 3D, onde o VANT poderá definir trajetórias dentro do mapa, de forma que obstáculos possam ser transpostos através de movimentação do robô aéreo nos eixos x, y e z. Esse mapeamento será alcançado a partir do processo de exploração com múltiplos VANT, fazendo uso de estratégias para coordenar o comportamento coletivo, buscando reduzir o tempo de exploração do ambiente.

4 MÉTODO BIOINSPIRADO PARA EXPLORAÇÃO E MAPEAMENTO DE AMBIENTES INTERNOS

Este capítulo apresenta o método proposto neste trabalho. São explicitados os detalhes de como ocorre o processo de exploração e mapeamento do ambiente, definindo os meios de captura de dados pelos diversos sensores a serem utilizados, as formas de tratamento de bloqueio de caminhos entre trajetórias sobrepostas e uma forma de visualização do mapa topológico bioinspirado na forma em que as abelhas constroem as suas colmeias.

4.1 VISÃO GERAL DO MÉTODO PROPOSTO

Um time de VANT que precisa fazer a exploração e o mapeamento com o intuito de auxiliar equipes de resgate, necessita ter a capacidade de identificar tanto os espaços livres para circulação quanto os obstáculos existentes, visando definir uma área acessível para a atuação dos humanos. Tanto as restrições de urgência na obtenção do mapeamento quanto as restrições tecnológicas dos VANT (poder de processamento ou o tempo de voo) podem limitar o uso de soluções de mapeamento completo do ambiente. Soluções de mapeamento métrico 3D podem exigir maior poder de processamento por parte dos robôs envolvidos, e consequentemente o tempo de ação é afetado.

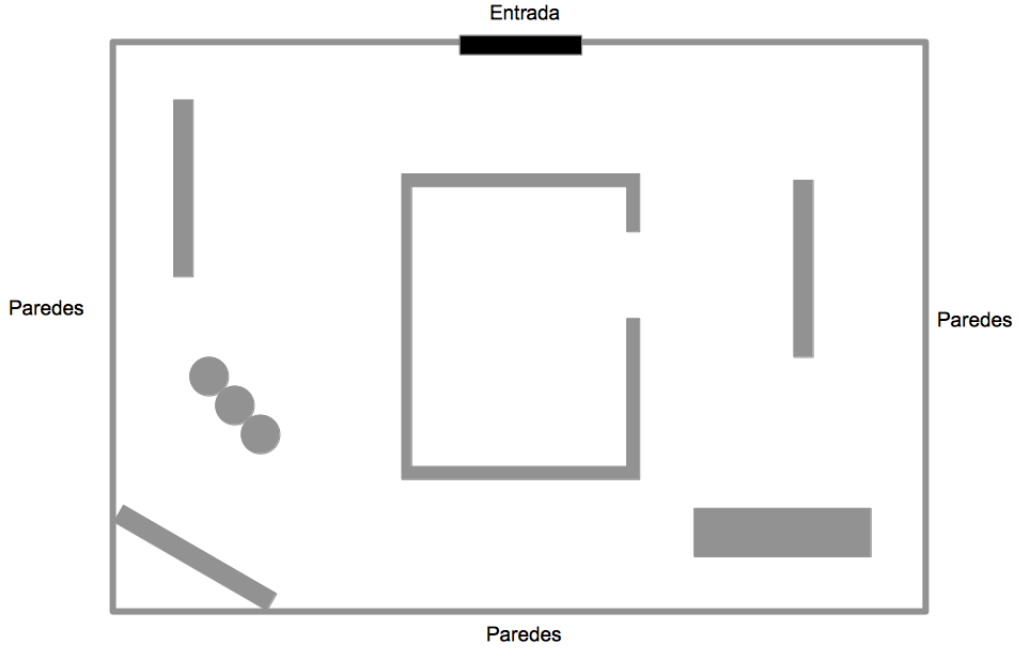
Dessa forma, este trabalho propõe a construção de um método cooperativo para o mapeamento de ambientes internos. O método é inspirado em como as abelhas constroem as colmeias, fornecendo uma estrutura topológica para o mapa que os múltiplos VANT usarão para realizar suas tarefas e paralelamente gerando uma visualização do espaço em um modelo em três dimensões. Assim, o método é dividido em:

- Representação do mapa topológico em um grafo inspirado na estrutura de favos da colmeia de abelhas;
- Cada célula da colmeia (ou nó do grafo) contém uma representação local de ocupação em 3D que pode ser visualizado logo após a célula ser incluída na colmeia;

- Exploração do espaço a partir dos pontos existentes no mapa topológico.

A figura 16 apresenta um possível cenário de mapeamento. Nesta configuração, imagina-se que exista um prédio com pelo menos uma área de entrada (porta, janela ou algum buraco suficientemente grande para a passagem de VANT) e diversos obstáculos como paredes, objetos, escombros, entre outros.

Figura 16: Exemplo de ambiente a ser modelado.

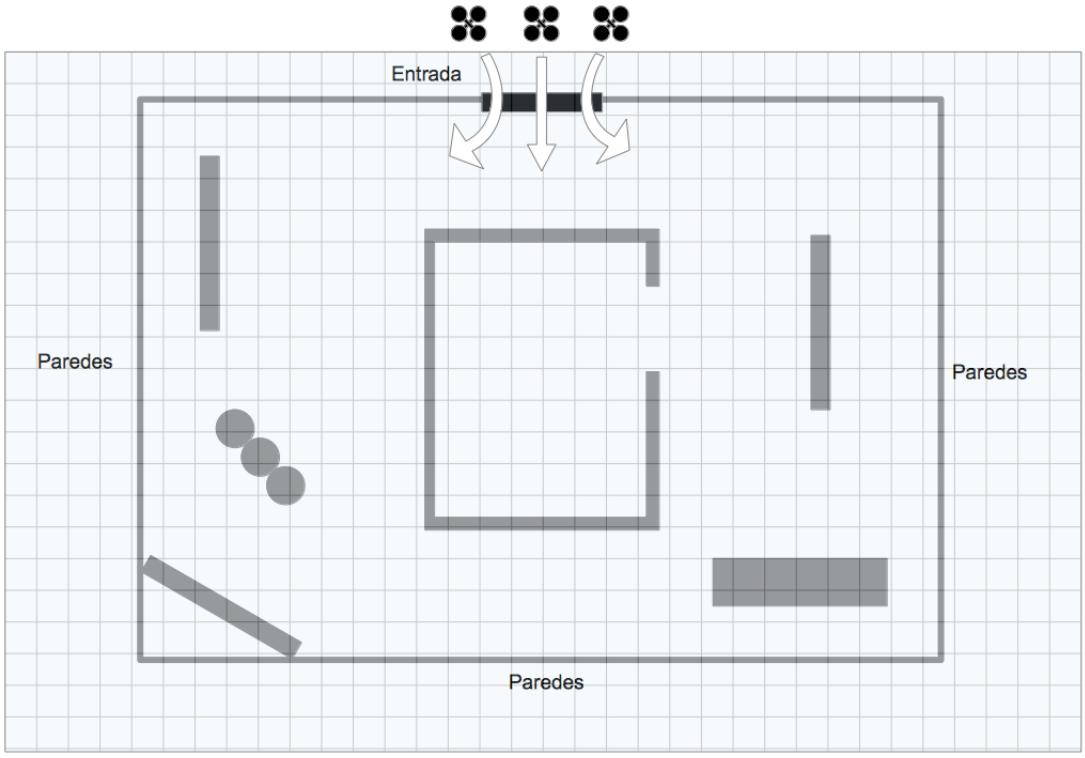


Fonte: Autoria própria.

Ao definir um local de entrada, inicia-se o processo de coleta de dados para o mapeamento do espaço, com os VANT iniciando a circulação pela abertura existente no ambiente, como pode ser visto na figura 17.

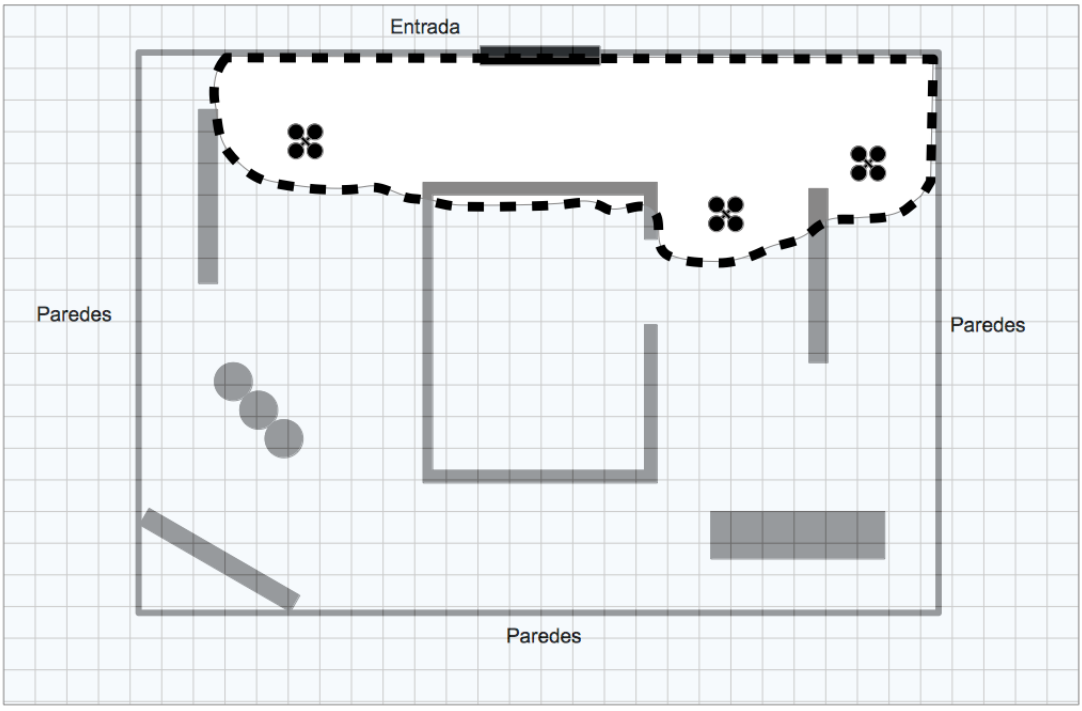
Conforme os VANT circulam e exploram o espaço, o mapeamento começa a ser feito e pouco a pouco o ambiente fica conhecido, conforme ilustra a figura 18.

Figura 17: VANT na entrada do espaço a ser explorado.



Fonte: Autoria própria.

Figura 18: VANT explorando o espaço.



Fonte: Autoria própria.

4.2 MAPEAMENTO BIOINSPIRADO DO AMBIENTE

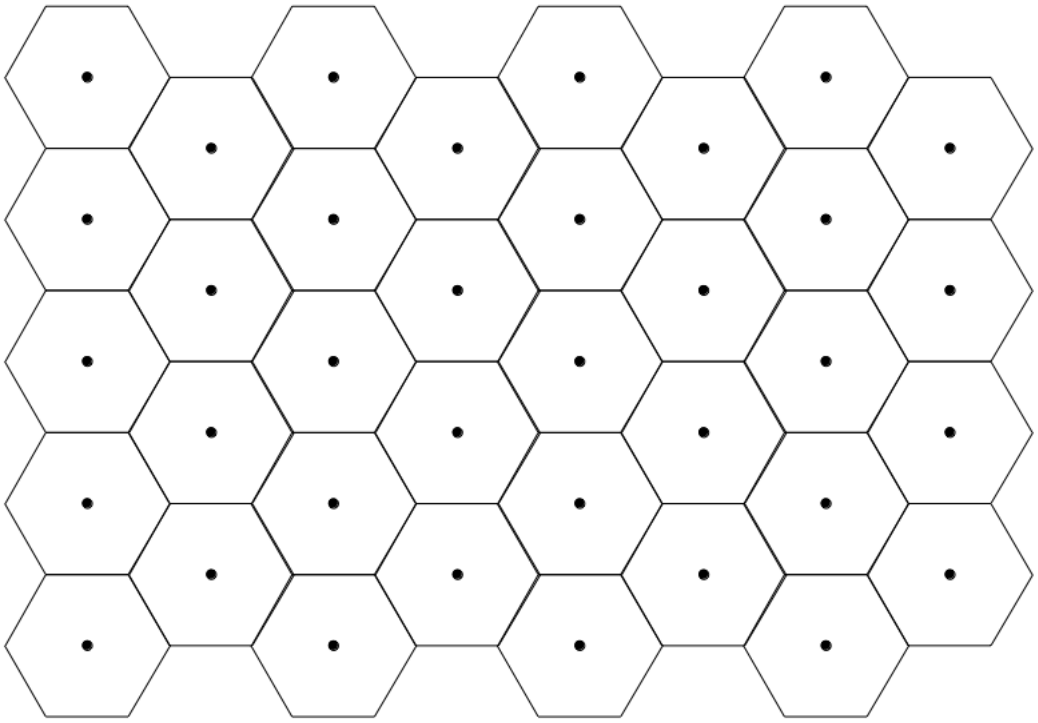
O mapa do ambiente será produzido a partir da leitura de sensores dos diversos VANT que movimentam-se pelo espaço. Com os dados coletados pelos VANT, é construído um mapa topológico na forma de um grafo interconectado cuja estrutura é bioinspirada em favos de colmeias de abelhas. Alguns trabalhos já foram inspirados pelo uso de estrutura de favos de colmeia, porém voltados para a estrutura de comunicação de redes sem fio ((LIU et al., 2006; STOJMENOVIC, 1997)). Outros trabalhos se dedicaram a representação de mapas com formatos hexagonais, porém sem aplicação na robótica móvel ((YANG; BIUK-AGHAI, 2015; POLISCIUC et al., 2016; SAHR, 2011)).

A construção dos favos de uma colmeia (*honeycomb*) faz uso de um material a prova d'água a partir de ceras produzidas pelas próprias abelhas, juntamente com a estrutura hexagonal de cada favo, proporciona uma estrutura leve e resistente o suficiente para maximizar a produção de mel e outras necessidades delas (GOULD; GOULD, 2012). Em geral, as abelhas fazem uma organização de cilindros de padrão hexagonal onde elas progressivamente formam as laterais desses cilindros pela adição de ceras produzidas e manipuladas pelas abelhas construtoras. A estrutura hexagonal permite a construção de favos com menor quantidade de cera necessária (economia de material) com a capacidade de maior armazenamento. O conjunto dos favos da colmeia é o resultado do trabalho coletivo de centenas de abelhas onde não há um comando central, e os indivíduos constroem apenas novos cilindros ao lado de outros existentes, promovendo a ampliação da colmeia, de forma que este ambiente influencia o comportamento, o qual por sua vez transforma o ambiente, sendo um mecanismo de sinergia (NAZZI, 2016). A figura 19 apresenta a estrutura de favos de colmeia.

Na natureza as abelhas começam a estruturar os favos a partir de buracos em árvores, galhos e outras estruturas existentes (GOULD; GOULD, 2012). Já na apicultura, produtores fornecem uma estrutura retangular onde as abelhas começam a construção em uma base na parte superior e vão aos poucos construindo novos favos adjacentes aos já existentes (EMBRAPA, 2018). Da mesma forma, o método aqui apresentado busca iniciar a construção do mapeamento a partir de um ponto inicial gerado por um VANT, enquanto os demais VANT irão sempre realizar suas tarefas em espaços diretamente adjacentes aos já existentes no mapa.

Esta estrutura de favos possibilita que seja definida uma estrutura topológica no mapeamento de ambientes. O centro de cada favo representa um nó, enquanto as paredes do hexágono representam a adjacência existente entre os favos. Dessa forma, o método proposto considera que se existe uma adjacência entre os favos, existe um espaço livre no ambiente e,

Figura 19: Estrutura hexagonal dos favos de uma colmeia.



Fonte: Autoria própria.

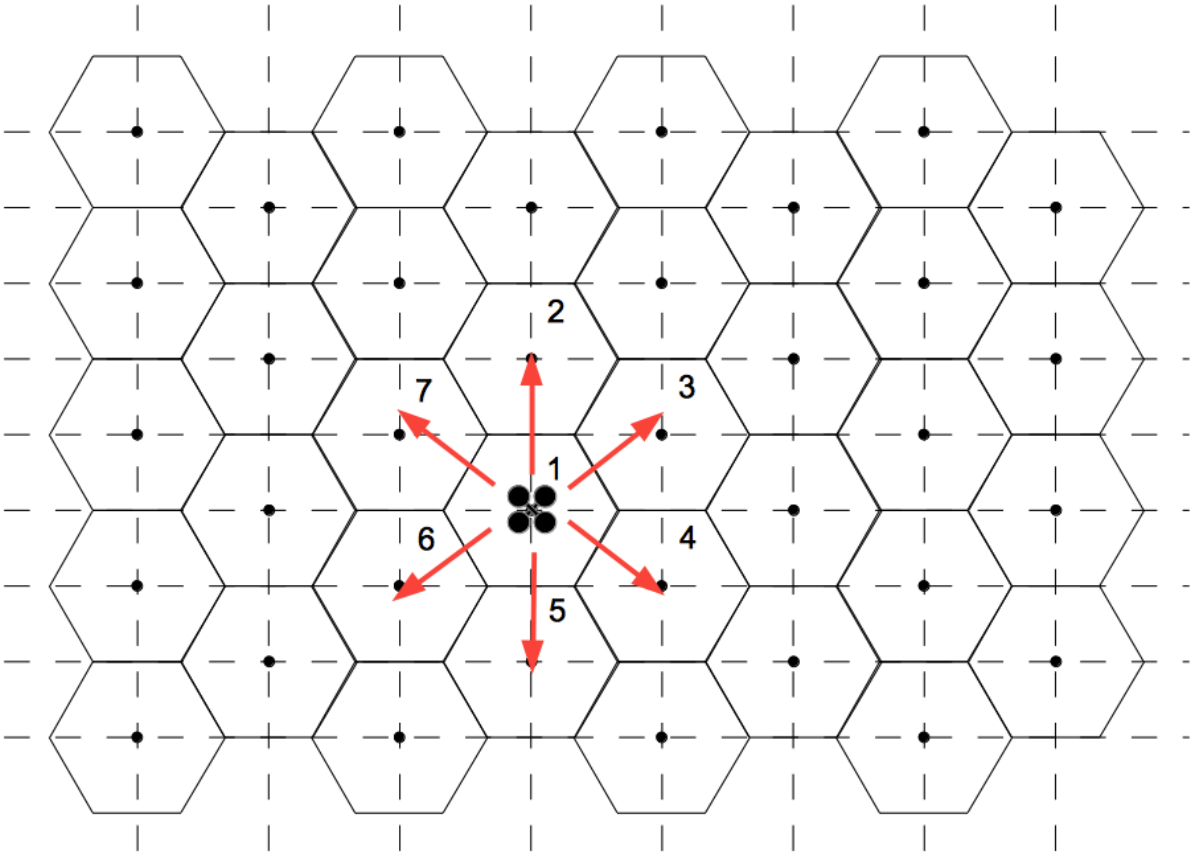
assim, é possível que o VANT ou um humano se desloque por esse espaço.

Ao sobrepor e combinar a estrutura de favos de colmeias com uma área do ambiente, é possível definir um avanço da exploração e do mapeamento baseando-se no comportamento das abelhas neste processo. A figura 20 apresenta esta combinação.

Por exemplo, considere o ponto (1) da figura 20 onde está localizado o VANT, o qual pode ser considerado o ponto inicial de exploração do ambiente. Caso não existam obstáculos, ele possui adjacência com os pontos (2), (3), (4), (5), (6) e (7). Assim, esse mapeamento topológico possibilita o processamento de forma mais leve em comparação com o processamento de mapas métricos (SIEGWART et al., 2011).

A distribuição do espaço a ser mapeado é iniciado em um ponto de entrada pré-estabelecido, onde a equipe de resgate irá posicionar o primeiro VANT que atuará como sentinela, de forma semelhante que as abelhas constroem os favos na natureza. Um primeiro VANT registra o primeiro favo (ou hexágono) do mapa a partir de sua localização inicial e então identifica onde existem adjacências a partir de leitura de sensores localizados na sua parte frontal, superior e inferior, inserindo esses novos identificadores no mapa topológico em uma unidade centralizadora de informações com acesso global por todos os robôs. Estes novos pontos identificados estão disponíveis para serem mapeamentos pelos demais VANT. Os VANT

Figura 20: Combinação entre área do ambiente e a estrutura hexagonal dos favos de uma colmeia.



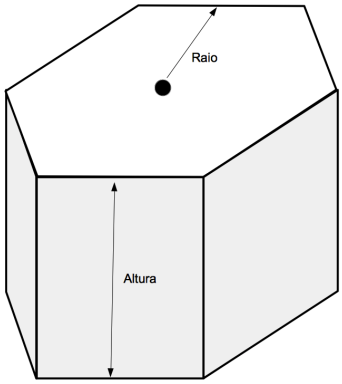
Fonte: Autoria própria

podem estar em três possíveis estados: em exploração, em deslocamento ou parado. O estado *em exploração* significa que o VANT está fazendo a leitura e mapeamento da faces de um determinado hexágono. O estado *em deslocamento* significa que o VANT recebeu a tarefa de explorar um determinado hexágono e está se dirigindo até ele pelo mapa topológico. O VANT estará no estado *parado* quando estiver esperando a definição de algum hexágono para mapear. Assim, cada VANT que não está em estado de exploração de um favo ou deslocando-se, busca na unidade centralizadora um novo espaço que ainda precisa ser mapeado a partir das adjacências existentes dos favos já identificados. O processo de mapeamento de cada favo gera uma representação em três dimensões da ocupação do ambiente, mas também pode incluir mais informações que podem ser adquiridas dependendo da tecnologia de sensores embarcados no VANT, como imagens térmicas por exemplo. Desse modo, o mapa fornecerá uma visão gráfica para as equipes de resgate ao mesmo tempo que a estrutura topológica em favos de colmeia possibilitam um processamento de menor custo comparado com uma representação em grade de ocupação.

As dimensões dos hexágonos a ser mapeado deverão ser estabelecidas antes do

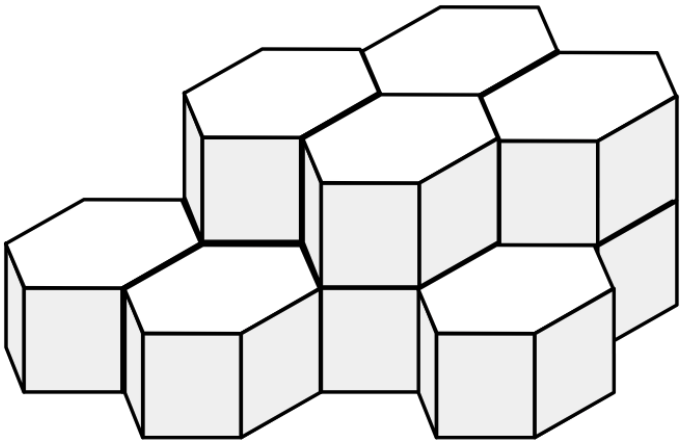
processo de exploração. Devem ser definidas as dimensões de raio do hexágono (que é a distância do centro até o ponto mais próximo de uma das faces), a sua altura, bem como qual a altitude máxima de atuação dos VANT. A figura 21 apresenta um hexágono e os pontos que devem ser configurados. A definição do raio de um hexágono para um mesmo espaço irá interferir no grau de detalhamento do mapa gerado, bem como a quantidade de hexágonos necessários para cobrir a área total explorada, ou seja, quanto menor o raio, maior será o número de hexágonos para um mesmo espaço. Além disso, a altura do hexágono e a altitude máxima de atuação dos VANT irão interferir no número de camadas de hexágono sobrepostos. A figura 22 apresenta um exemplo dos hexágonos em duas camadas sobrepostas.

Figura 21: Hexágono com configuração de raio e altura.



Fonte: Autoria própria.

Figura 22: Hexágono em camadas sobrepostas.



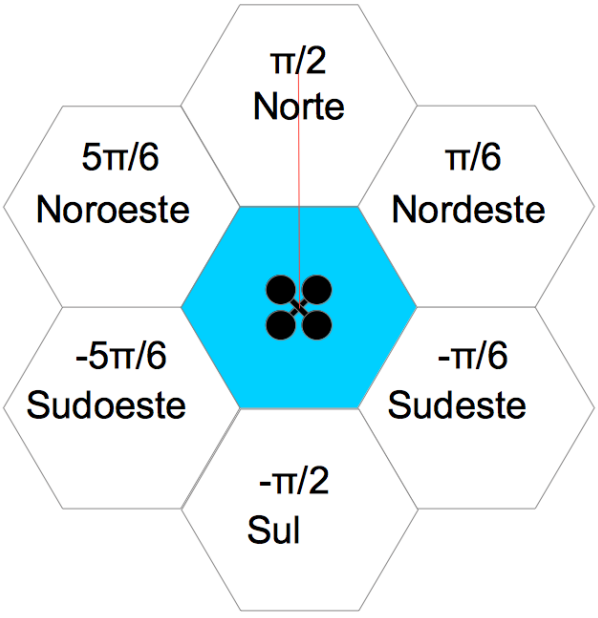
Fonte: Autoria própria.

4.3 EXPLORAÇÃO E MAPEAMENTO DO AMBIENTE

4.3.1 MÉTODO DE EXPLORAÇÃO

A construção da colmeia, que é representada como um conjunto de células hexagonais, inicia com o trabalho de uma primeira abelha operária construindo o primeiro favo usando cera para edificar as paredes. De forma semelhante, no método proposto, um primeiro VANT gera o primeiro hexágono do mapa, verificando se há adjacência para cada um dos seis lados e nas partes superior e inferior. É importante destacar que, no contexto deste trabalho o termo *adjacência* significa a possibilidade de um VANT se mover de um hexágono para outro. Então, um hexágono existirá no mapa se e somente se é possível que um VANT possa acessá-lo completamente a partir de outro hexágono através de, ao menos, um dos seus seis lados ou pelas partes superior e inferior. Dessa forma, não podem existir obstáculos entre o centro de um hexágono origem e o centro do outro hexágono destino que impeçam um VANT de se movimentar do ponto de origem ao ponto destino. A figura 23 apresenta um VANT explorando um hexágono que deve rotacionar nos seis ângulos além de verificar as adjacências superiores e inferiores: $\pi/2, \pi/6, -\pi/6, -\pi/2, -5\pi/6$, e $5\pi/6$. Caso haja adjacência, o centro do próximo hexágono adjacente será definido por uma distância baseada no tamanho do hexágono e o ângulo que ele foi identificado, não considerando a posição atual do VANT, mas sempre referente ao hexágonos já existentes. Cada hexágono avaliado é marcado com um identificador único (*id*), conforme é apresentado na figura 20.

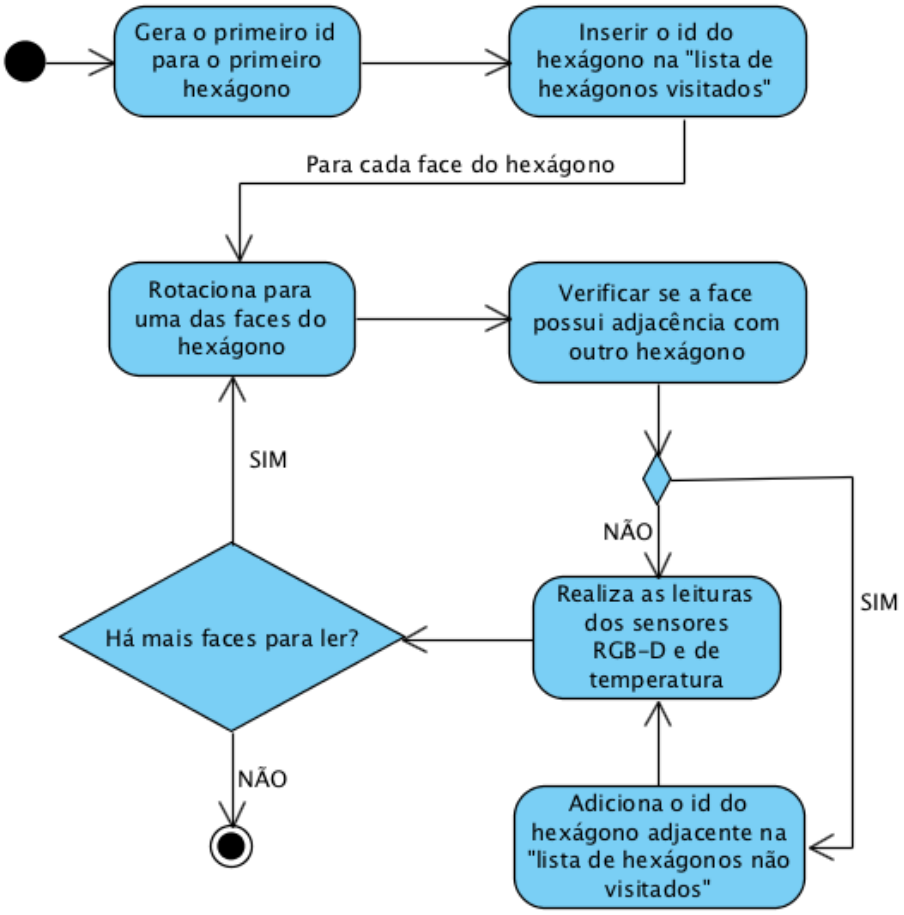
Figura 23: VANT na exploração de um hexágono.



Fonte: Autoria própria.

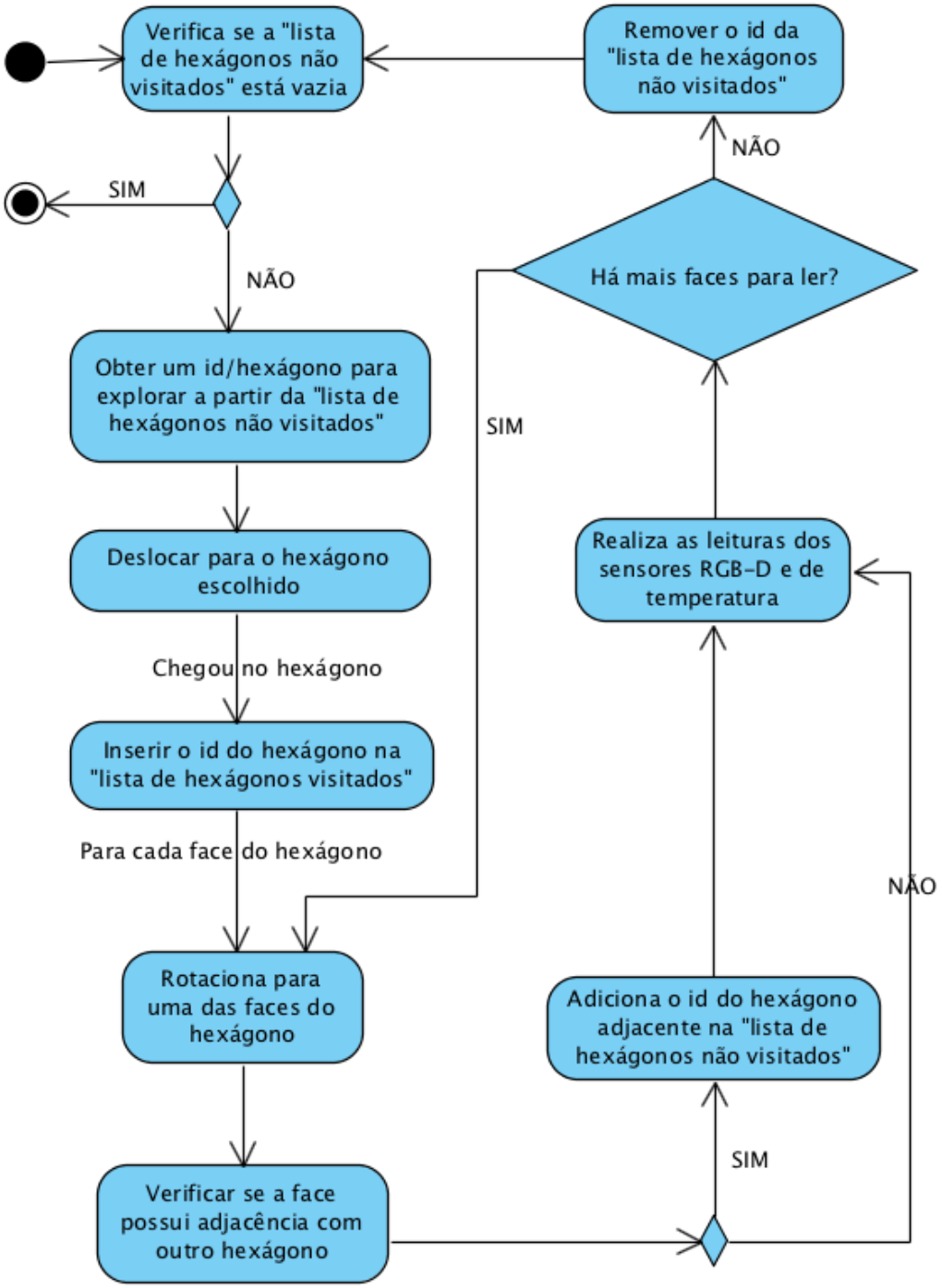
As figuras 24 e 25 apresentam os diagramas de atividade das ações de exploração do VANT sentinela e dos demais, respectivamente. O primeiro VANT a iniciar a exploração tem um comportamento diferente dos demais na primeira tarefa, pois é o responsável por gerar o primeiro *id* do primeiro hexágono do mapa. Nas próximas explorações, ele terá o comportamento padrão dos demais VANT.

Figura 24: Processo de exploração: primeira exploração do ambiente com o primeiro VANT.



Fonte: Autoria própria.

Figura 25: Processo de exploração: exploração de todos os VANT depois da primeira exploração.



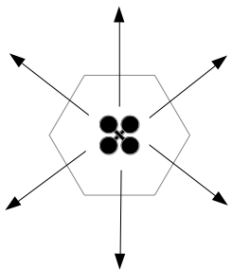
Fonte: Autoria própria.

A figura 26 mostra a ação de exploração de um VANT. De forma geral, o VANT explora o hexágono verificando se é possível transpor os seus limites em cada um dos seis ângulos/lados e nas partes superior e inferior (figura 26a), escolhe um novo hexágono para explorar (figura 26b) e desloca-se até ele, iniciando uma nova exploração (figura 26c). A busca por locais de exploração bem como a identificação dos hexágonos encontrados e demais

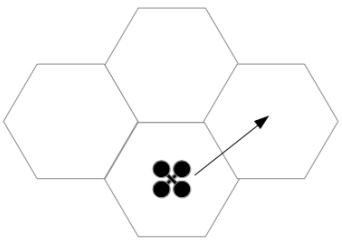
informações são concentradas em uma unidade central (UC), a qual é a responsável por controlar o acesso concorrente dos VANT nas estruturas de dados que armazenam dados da exploração e do mapeamento. Essa abordagem baseada em UC pode acarretar em constante comunicação entre os VANT e a base, porém detalhes de comunicação são abstraídos neste trabalho.

Uma vez que o VANT sentinela finaliza a primeira exploração identificando o primeiro hexágono e a existência de adjacentes, ele notifica a *unidade central* - UC, de forma que os demais VANT possam iniciar sua busca por espaços para explorar. Para controlar os hexágonos identificados no processo de leitura das suas faces, são usadas duas listas e uma matriz. Para registrar os identificadores (*id*s) dos hexágonos explorados, é definida uma lista chamada "lista de hexágonos visitados". Quando o VANT rotaciona e encontra adjacência para um novo hexágono, um novo *id* é gerado e adicionado em uma estrutura chamada "lista de hexágonos não visitados", enquanto as informações de adjacências são armazenadas na *matriz de adjacências*, a qual relaciona os *id*s adjacentes. Então, um VANT que está buscando um hexágono para explorar deverá realizar esta busca na "lista de hexágonos não visitados". Ao final da exploração do hexágono, o *id* é removido da "lista de hexágonos não visitados".

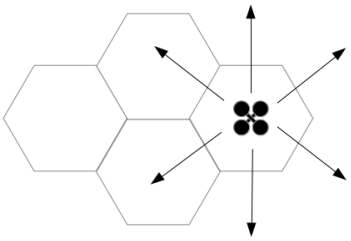
Figura 26: Sequência de ações da exploração.



(**a**) Ação 1: explorar o primeiro hexágono.



(**b**) Ação 2: obter um novo hexágono para explorar e se deslocar até ele.



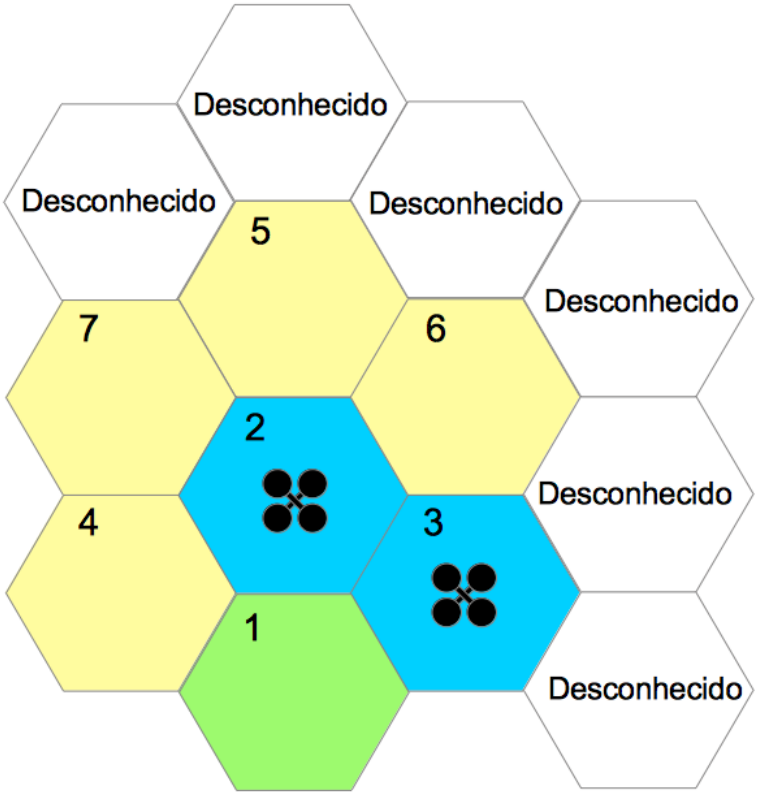
(**c**) Ação 3: explorar o novo hexágono.

Fonte: Autoria própria.

A figura 27 apresenta dois VANT explorando um determinado espaço. Neste caso, a exploração iniciou com o hexágono 1, o qual já foi completamente avaliado. Para fins de ilustração, o hexágono 1 é pintado na cor verde, indicando que ele já foi completamente explorado. No processo de exploração, adjacências foram identificadas com os hexágonos 2, 3 e 4, os quais foram inseridos na "lista de hexágonos não visitados". Quando um VANT inicia a exploração do hexágono 2, os hexágonos 5, 6 e 7 são identificados. Na figura 27, os hexágonos azuis indicam um espaço em exploração, enquanto os amarelos são aqueles que foram identificados, mas ainda não foram explorados. O processo de exploração termina quando

a “lista de hexágonos não visitados” estiver vazia.

Figura 27: Exploração com múltiplos VANT: hexágonos verdes representam locais explorados; hexágonos amarelos representam locais que têm adjacência, mas não foram explorados ainda; hexágonos azuis representam locais que estão sendo explorados por algum VANT.



Fonte: Autoria própria.

Algumas suposições e simplificações são assumidas:

- Ao se deslocar, o VANT desvia de eventuais obstáculos até chegar no próximo hexágono, definindo uma trajetória pelos espaços livres já mapeados. Os elementos dinâmicos considerados na prevenção de colisões são os demais VANT circulando no ambiente, os quais informam a unidade central em qual hexágono estão localizados;
- Os erros de odometria são desconsiderados;
- Como o mapeamento inicial é identificado em um ponto fixo, as adjacências são relativas a este ponto, mantendo as proporções e distâncias iguais entre hexágonos;
- Quando um VANT precisa saber em qual hexágono ele ou outro VANT se encontra, ele repassa um conjunto de coordenadas xyz[1] para a unidade central, a qual verifica qual

[1]Neste trabalho são abstraídos os detalhes de precisão de posicionamento, entretanto esse é um tema que necessita ser considerado quando da sua implantação em VANT reais.

hexágono tem em seu espaço aquela coordenada informada, e retorna o *id* correspondente. Assim, o VANT irá se deslocar de forma que fique centralizado no hexágono, e então realize as leituras de adjacências. Caso exista a adjacência em um novo ponto ainda não mapeado, este novo ponto será baseado em uma distância fixa do hexágono de onde ele se encontra, independente do valor de distância capturado pelo sensor, evitando o acúmulo de erros;

- O consumo de energia e o nível de carga da bateria são ignorados. Em outras palavras, o VANT sempre tem energia suficiente para se deslocar até o hexágono alvo e realizar o mapeamento do local.

A seguir são detalhadas todas as atividades do método de exploração que são mostrados na figura 25.

4.3.2 CRITÉRIO DE PARADA DO MÉTODO

Verificar se a "lista de hexágonos não visitados" está vazia é o critério de parada do algoritmo de exploração. Cada hexágono descoberto que não tenha sido explorado por algum VANT é inserido na "lista de hexágonos não visistados". Quando um VANT solicita à UC um *id* de um hexágono para realizar a tarefa de explorá-lo, ele recebe um *id* que permanece nesta lista de não visitados até o fim da sua exploração, entretanto o VANT marca nessa lista que a ele foi atribuído a exploração e mapeamento do hexágono em questão, evitando que não seja alocado para outro VANT. Isso garante que nenhum VANT irá parar o processo de exploração sem que realmente não exista mais espaços para explorar. Por exemplo, em um ponto da exploração, um VANT pode ter finalizado a sua tarefa, enquanto outro está trabalhando. Se não há hexágonos não explorados disponíveis, o primeiro VANT irá esperar possíveis novas descobertas do segundo, o qual continua sua atividade de exploração. Se nenhum novo hexágono é descoberto, então a exploração como um todo é finalizada, pois a "lista de hexágonos não visitados" está vazia.

4.3.3 DEFINIÇÃO DO PRÓXIMO LUGAR A SER EXPLORADO

A definição de qual hexágono será explorado por um VANT interfere em todo o processo de mapeamento, pois afeta os deslocamentos dos robôs e, consequentemente, o tempo de exploração. Assim, uma questão importante é a definição de como um conjunto de n VANT irão trabalhar em um ambiente dinâmico. Considere o ponto onde o VANT sentinela inicia a exploração no hexágono 1, h_1. O VANT 1 analisa o entorno do hexágono e identifica se

há hexágonos vizinhos/adjacentes. Os hexágonos não explorados serão inseridos na "lista de hexágonos não visitados".

Dessa forma, considerando um conjunto de n VANT, $SU = \{u_1, u_2, ..., u_n\}$, e um conjunto de hexágonos não explorados definida como $HN = \{h_2, ..., h_n\}$. O conjunto HN é elaborado por todos os VANT de uma forma colaborativa, onde cada VANT, u_j, insere elementos no conjunto global HN. Para identificar o caminho entre cada VANT u_i e um hexágono h_j, é aplicado o algoritmo de Dijkstra (DIJKSTRA, 1959). Assim, é necessário definir qual hexágono h_j será explorado. Para isso, três estratégias foram definidas[2] para buscar uma forma de exploração com o tempo de execução e circulação no ambiente otimizados.

A estratégia I é baseada na ordem *First In First Out* (FIFO). Neste caso, o *id* do hexágono a mais tempo inserido no conjunto HN será o primeiro a ser escolhido para a tarefa de mapeamento e exploração. Como a escolha do hexágono a ser explorado é sempre o primeiro elemento encontrado na lista, a complexidade computacional é $\Theta(1)$.

A estratégia II é baseada na distância euclidiana com respeito ao primeiro hexágono mapeado h_1. Considerando a posição $x_1y_1z_1$ do hexágono h_1 e a posição $x_jy_jz_j$ de cada hexágono, o escolhido será aquele h_j que possui o menor valor de distância euclidiana (DE), o qual é calculado através da equação 1. Desse modo, como no conjunto HN há m elementos, o custo de buscar o hexágono com a menor distância euclidiana em relação ao primeiro hexágono mapeado é $O(m)$.

$$DE = \sqrt{(x_1 - x_j)^2 + (y_1 - y_j)^2 + (z_1 - z_j)^2} \tag{1}$$

onde $j = 2, ..., n$, sendo n o número de hexágonos do conjunto HN.

A estratégia III é baseada na distância euclidiana com respeito ao primeiro hexágono h_1 e a localização atual do VANT, que é um determinado hexágono h_l onde ele se encontra. Considerando a posição $x_jy_jz_j$ de cada hexágono, o escolhido será aquele h_j que possui o menor valor de distância euclidiana relativa DER, o qual é calculado na equação 2, considerando o valor calculado na equação 1. De forma semelhante a estratégia II, como no conjunto HN há m elementos, o custo de buscar o hexágono com a menor distância euclidiana relativa em relação ao primeiro hexágono mapeado é $O(m)$.

$$DER = DE_j + \sqrt{(x_l - x_j)^2 + (y_l - y_j)^2 + (z_l - z_j)^2} \tag{2}$$

[2]Estas estratégias são resultado de um trabalho integrado entre Universidade Tecnológica do Paraná (UTFPR) e o Instituto Politécnico de Bragança - Portugal (IPB), a partir do Edital UTFPR/IPB 01/2017, em que foi gerada uma publicação (ROSA et al., 2020), a qual encontra-se no apêndice D.

onde $j = 2, ..., n$, sendo n o número de hexágonos do conjunto HN.

A escolha do hexágono a ser explorado, baseada em uma das três estratégias apresentadas acima, é feita pela unidade central (UC). Salienta-se que a estratégia adotada é aplicada individualmente para cada VANT, não considerando a informação global sobre a disposição dos robôs no ambiente. Após ser definido qual hexágono (*id*) deve ser explorado por um determinado VANT u_j, este hexágono (da "lista de hexágonos não visitados") é marcado com o identificador do VANT que o explorará, de forma que um mesmo hexágono não seja atribuído para dois VANT ao mesmo tempo. A partir do momento que o VANT possui o *id* do hexágono que irá explorar e mapear, ele calcula uma trajetória e se desloca até o ponto do centro do hexágono.

4.3.4 DESLOCAMENTO DO VANT

Uma vez que o VANT obtém um hexágono para explorar a partir da "lista de hexágonos não visitados" , ele se deslocará até o ponto que representa o centro do hexágono. O VANT transita apenas através de espaços livres e conhecidos. Então, dado um hexágono onde o VANT está localizado e o seu destino, um caminho de deslocamento é definido para ele percorrer. Este caminho é construído com o algoritmo de Dijkstra (DIJKSTRA, 1959)[3]. Com esse caminho estabelecido, que é uma lista de *id*s de hexágonos adjacentes, o VANT transita pelo mapa, através de trajetórias retas do centro de um hexágono até o centro do outro, até alcançar seu destino. Assume-se aqui que durante o deslocamento não serão encontrados demais obstáculos dinâmicos, com exceção dos VANT que estão circulando no espaço e que o tratamento de colisões para este caso é tratado na seção 4.4.

Por exemplo considere o cenário da figura 27. Suponha que um VANT que se encontra no hexágono 1 obteve como local a ser explorado o hexágono 5. O algoritmo Dijkstra forneceria um caminho composto pelos hexágonos 1, 4, 7 e 5, pois os hexágonos 2 e 3 estariam ocupados e não possibilitariam o tráfego por eles.

Quando o VANT finaliza sua movimentação através do caminho definido pelo algoritmo de Dijkstra, ele se encontra no hexágono definido para ser explorado. No início da atividade de exploração, o *id* do hexágono é inserido na "lista de hexágonos visitados". Isso irá possibilitar a identificação da ordem de exploração dos hexágonos para análises futuras.

[3] O algoritmo utilizado é uma implementação em Matlab feita por Aryo (2020).

4.3.5 VERIFICAÇÃO DE ADJACÊNCIAS

O processo de definição de um hexágono consiste no VANT rotacionar 360^o na direção horária e, para cada um dos seis lados do hexágono, o VANT verifica se há adjacência: um sensor de distância faz a leitura e identifica se é possível que o VANT acesse o centro de um hexágono vizinho, em outras palavras, se não existem obstáculos entre os pontos centrais dos dois hexágonos. Neste caso, a adjacência é adicionada na *matriz de adjacências*. Para as leituras de adjacência superior e inferior, a ação de rotação não é necessária, sendo feitas apenas as leituras dos sensores posicionados nas partes inferior e superior do VANT.

A adjacência é considerada existente se o VANT u_i consegue sair de um ponto $x_a y_a z_a$ e ir para um ponto $x_b y_b z_b$, onde o ponto $x_a y_a z_a$ é um hexágono conhecido h_a possuindo seu $id = a$. É definido um valor de raio R que representa a distância do centro do hexágono até a sua borda e uma altura H que representa a sua altura, possibilitando que, por exemplo, uma sala seja mapeada com duas ou mais camadas de hexágonos sobrepostos, e.g. caso a altitude de atuação seja $2,5m$ e a altura do hexágono seja $H = 1m$. Essas duas informações devem ser previamente configuradas.

O sensor de distância horizontal é configurado para identificar obstáculos a uma distância de $2R + \Delta s$, onde Δs é a variação do comprimento da fuselagem do VANT. Os sensores de distância verticais são configurados para identificar obstáculos a uma distância de $H + \Delta h$, em que Δh é a variação da altura da fuselagem do VANT. Para ilustrar um exemplo de detecção de adjacência, um VANT está em um determinado hexágono no mapa e o seu sensor de distância horizontal faz a leitura de um valor de $4,5m$ para um ângulo específico. Caso a configuração do raio do hexágono seja $R = 0,5$ e o $\Delta s = 0,4$, então a leitura do sensor é maior que $2R + \Delta s$, indicando a existência de adjacência, pois não existem obstáculos fixos entre estes pontos.

Nas leituras nos seis ângulos do hexágono, caso haja esta adjacência, é calculado o ponto $x_b y_b z_b$ do hexágono h_b, com base no valor do ângulo α que o VANT está, o ponto $x_a y_a z_a$ do hexágono h_a e o raio R. São definidos os valores dx e dy para as variações nos eixos x e y entre os dois pontos, baseando-se nas leis dos senos, conforme as equações 3 e 4.

$$dx = \frac{2 \times R \times \sin(\alpha)}{\sin(90)} \tag{3}$$

$$dy = \frac{2 \times R \times \sin(90 - \alpha)}{\sin(90)} \tag{4}$$

Dessa forma, a posição $x_b y_b z_b$ do novo hexágono h_b é definida na equação 5, sendo que $z_b = z_a$, pois estão no mesmo eixo z.

$$(x_b, y_b, z_b) = \begin{cases} x_b = x_a + dx \\ y_b = y_a + dy \\ z_b = z_a \end{cases} \tag{5}$$

Para as verificações de adjacência das partes superior e inferior, os sensores destinados para essa medição consideram a detecção da adjacência, caso nada seja detectado a uma distância de $H + \Delta h$. As equações 6 e 7 apresentam os cálculos dos pontos $x_s y_s z_s$ e $x_i y_i z_i$ para as leituras superior e inferior, respectivamente.

$$(x_s, y_s, z_s) = \begin{cases} x_s = x_a \\ y_s = y_a \\ z_s = z_a + H \end{cases} \tag{6}$$

$$(x_i, y_i, z_i) = \begin{cases} x_i = x_a \\ y_i = y_a \\ z_i = z_a - H \end{cases} \tag{7}$$

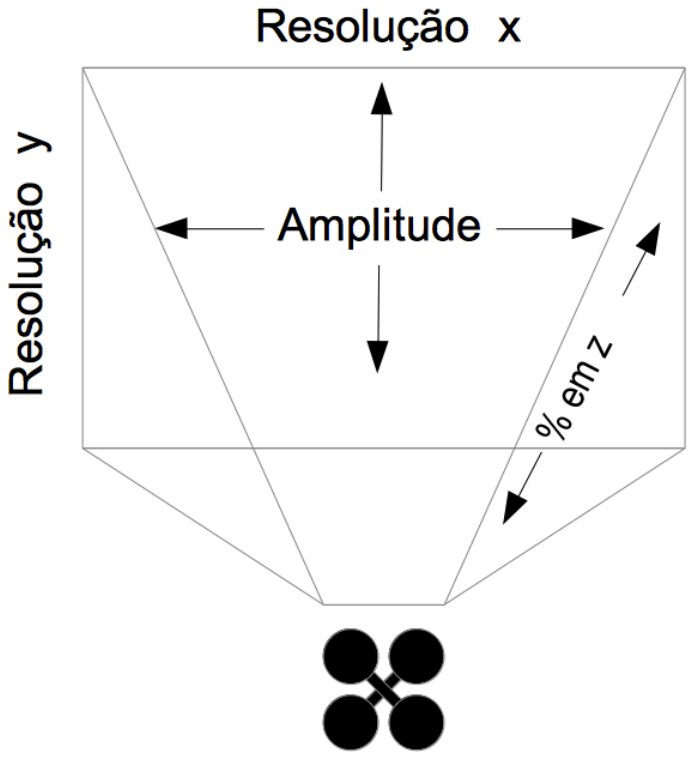
Quando a leitura dos sensores descobrem a adjacência entre hexágonos, é feita a inserção dessa adjacência na "matriz de adjacências". Esse processo é realizado a partir do envio da posição *xyz* do hexágono descoberto para a UC. Na UC, um módulo irá verificar se aquele ponto *xyz* já foi identificado por algum outro VANT u_j possuindo um identificador h_b. Se essa identificação já existir, significa que outro VANT já mapeou aquele ponto no espaço e que o mesmo ou já foi explorado ou está na "lista de hexágonos não visitados" esperanto exploração, e então é feita apenas o registro da adjacência entre os dois hexágonos na *matriz de adjacências*. Caso esse ponto nunca tenha sido encontrado anteriormente, será então gerado um novo *id* para o hexágono h_b o qual é inserido na "lista de hexágonos não visitados", as suas coordenadas *xyz* serão armazenados em uma lista que registra todos os pontos *xyz* de cada hexágono identificado e, por fim, será feito o registro da adjacência na "matriz de adjacências".

Ao final da leitura de todos os lados do hexágono (seis laterais e lados superior e inferior), ele é removido da "lista de hexágonos não visitados". Assim, o VANT pode começar a busca por um novo hexágono para exploração se a "lista de hexágonos não visitados" não estiver vazia, de outra forma, a atividade de exploração do VANT é finalizada.

4.3.6 AQUISIÇÃO DE INFORMAÇÕES LOCAIS DO AMBIENTE

As informações do mapa topológico gerado estão disponíveis tanto para os VANT utilizarem no controle dos seu movimentos no ambiente, quanto para os times de busca e resgate para saberem o espaço que podem se deslocar no ambiente. Além dos sensores de detecção de obstáculos, sensores RGB-D e de temperatura podem ser utilizados. Os sensores RGB-D fazem a leitura 3D para cada lado do hexágono resultando em uma nuvem de pontos para cada lado. Uma visão em cubos é construída para auxiliar as equipes no reconhecimento do espaço. Ao mesmo tempo, um sensor térmico faz leitura de temperatura do mesmo lado do hexágono. Se uma temperatura é maior que um valor de referência definido como parâmetro, essa informação é identificada e uma imagem do espaço é armazenada em uma estrutura apropriada. A figura 28 mostra uma representação da leitura de dados RGB-D do VANT.

Figura 28: Leitura RGB-D por um VANT.



Fonte: Autoria própria.

Para fazer a captura dos pontos obtidos pela câmera RGB-D, alguns dados são definidos:

- *matrizRGBD*: matriz de dados com a leitura RGB-D. O valor de dado dentro da matriz é uma porcentagem de quão longe ou perto está de um ponto (entre 0 e 1). Por exemplo, se o sensor está configurado para uma distância máxima de leitura de $1m$ e a leitura de um ponto do *matrizRGBD* for 0.1, então aquele ponto está à $10cm$ de distância do sensor

RGB-D, e se a configuração for para $2m$ e a leitura do ponto no *matrizRGBD* for 0.1 também, a distância será $20cm$;

- xn e yn: definem a resolução da leitura RGB-D. O *matrizRGBD* será uma matriz de dimensão $xn \times yn$. Neste trabalho é considerado $xn = yn$;
- dz: distância máxima que o sensor RGB-D fará a leitura. Essa informação é configurada no sensor;
- nz: leitura de um valor calculado com base no valor proporcional que está na *matrizRGBD* e no valor de dz;
- $amplitude$: é a abertura angular que o sensor possui. Essa configuração no sensor unida com a distância dz fornecerão o tamanho da janela de leitura do sensor;
- α: são os ângulos de Euler que o VANT possui no momento da captura do sensor RGB-D. $\alpha(1)$ traz a variação no eixo x, $\alpha(2)$ no eixo y e $\alpha(3)$ em z;
- $x_u y_u z_u$: é a posição em que o VANT se encontra no momento da captura do sensor RGB-D;
- Dc: é a dimensão do cubo que se pretende projetar a partir dos ponto capturados pelo sensor RGB-D. Quanto maior o cubo, menor será o detalhamento do ambiente e, de forma inversa, quanto menor o cubo, maior será o detalhamento.

Inicialmente, é necessário definir a variação angular da leitura RGB-D de forma que, dada a $amplitude$ configurada e a resolução, tenha-se o ângulo da leitura de cada ponto no *matrizRGBD*. Assim, para todos os pontos de leitura do *matrizRGBD* será considerado o valor definido na equação 8.

$$\Delta\alpha = \frac{amplitude}{xn} \tag{8}$$

Tendo o valor $\Delta\alpha$, deve-se varrer cada ponto armazenado em *matrizRGBD* e transformá-lo para pontos em coordenadas globais. Sejam $i = 1,...,yn$ e $j = 1,...,xn$, $nz = matrizRGBD(i,j) \times dz$ e α a variação angular do VANT. β será a variação angular do ponto $matrizRGBD(i,j)$ em relação ao ponto central da leitura da câmera RGB-D, conforme:

$$\beta = \begin{cases} 0, & \text{se } j = 1 \\ amplitude, & \text{se } j = xn \\ \Delta\alpha \times j, & \text{para os demais casos} \end{cases} \tag{9}$$

A variação angular no eixo z será βz, conforme a equação 10

$$\beta z = \begin{cases} 0, & \text{se } i = 1 \\ amplitude, & \text{se } i = xn \\ \Delta\alpha \times i, & \text{para os demais casos} \end{cases} \tag{10}$$

Definido a variação angular β do ponto, é necessário calcular o ângulo global do ponto θ conforme a equação 11:

$$\theta = \begin{cases} \alpha(3) + ((\frac{amplitude}{2}) - \beta), & \text{se } \beta < (\frac{amplitude}{2}) \\ \alpha(3) - (\beta - (\frac{amplitude}{2})), & \text{se } \beta >= (\frac{amplitude}{2}) \end{cases} \tag{11}$$

Já o ângulo global do ponto θz é definido pela equação 12:

$$\theta z = \begin{cases} \alpha(1) + ((\frac{amplitude}{2}) - \beta z), & \text{se } \beta z < (\frac{amplitude}{2}) \\ \alpha(1) - (\beta z - (\frac{amplitude}{2})), & \text{se } \beta z >= (\frac{amplitude}{2}) \end{cases} \tag{12}$$

Com o valor de θ e θz, pela lei dos senos, calcula-se as variações nos eixos xyz (δx, δy e δz), conforme as equações 13, 14 e 15:

$$\delta x = \frac{nz \times \sin(\theta)}{\sin(90)} \tag{13}$$

$$\delta y = \frac{nz \times \sin(90 - \theta)}{\sin(90)} \tag{14}$$

$$\delta z = \frac{nz \times \sin(90 - \theta z)}{\sin(90)} \tag{15}$$

Desse modo, as coordenadas globais do ponto extraído em $matrizRGBD(i, j)$, dada a posição $x_u y_u z_u$ do VANT, será $x_b y_b z_b$:

$$(x_b, y_b, z_b) = \begin{cases} x_b = x_u + \delta x \\ y_b = y_u + \delta y \\ z_b = z_u + \delta z \end{cases} \tag{16}$$

A representação dos pontos como cubos faz com que pontos próximos gerem cubos sobrepostos interseccionados. Por isso, é feita um agrupamento de dados onde os pontos que estão dentro de um determinado volume (considerando o valor de Dc configurado) do ambiente sejam representados por um único cubo. Os valores $x_b y_b z_b$ resultantes da equação 16 juntamente

com o valor $Rc = Dc/2$ são ajustados pela equação 17, onde o valor de $x_c y_c z_c$ será a coordenada do cubo a ser armazenada em uma estrutura adequada para posterior exibição.

$$(x_c, y_c, z_c) = \begin{cases} x_c = round(\frac{x_b}{Rc}) \times Rc \\ y_c = round(\frac{y_b}{Rc}) \times Rc \\ z_c = round(\frac{z_b}{Rc}) \times Rc \end{cases} \quad (17)$$

Como no ambiente em exploração haverão diversos VANT e diversas leituras, pode-se encontrar leituras repetidas do mesmo ponto $x_c y_c z_c$ obtidas em diferentes explorações. Assim, deve-se excluir dos registros leituras repetidas. Um processo de filtragem é feito pela unidade central UC. A configuração do valor de *Dc* influencia na qualidade da representação da visão em cubos 3D e na quantidade de dados armazenados: se o valor do *Dc* diminui, maior será o detalhamento da visão e consequentemente maior será a quantidade de dados armazenados, porém se este valor aumenta, menor será o detalhamento da visão gerada e menos dados serão armazenados.

4.4 CONTROLE DE CAMINHOS BLOQUEADOS

Durante o processo de exploração, os vários VANT irão se mover para os hexágonos, aos quais foram atribuídas as tarefas de mapeamento e, consequentemente, seus caminhos irão se cruzar. Evitar colisões é um ponto crítico para a exploração de um ambiente com múltiplos robôs. Diversas abordagens têm sido apresentadas, nas quais os meios de prevenção são propostos por programação otimizada (FUKUSHIMA et al., 2013; GAN et al., 2012; MORGAN et al., 2014), campos potenciais (PAMOSOAJI; HONG, 2013), métodos baseados em exemplos (BEKRIS et al., 2012), entre outros.

De modo geral, dois conceitos são aplicados (ZHOU et al., 2017): um onde os robôs são livres e podem alterar seus caminhos e outro onde os robôs possuem um caminho fixo sem possibilidade de mudanças. Dessa forma, no primeiro conceito o foco está na mudança do caminho a ser percorrido, enquanto no segundo o foco está no controle de movimento e tempo. Zhou et al. (2017) discutem um método para tratamento de *deadlock* para múltiplos robôs onde o caminho é fixo. Para melhorar o desempenho, algumas políticas de paradas são propostas. Cada robô toma a decisão de mudar ou esperar o outro. Alonso-Mora et al. (2018), apresenta uma abordagem de síntese de correção por construção para planejamento de missões para múltiplos robôs que garante evitar colisões com respeito à obstáculos que se movem. É feita a integração de um módulo de planejamento em alto-nível com um módulo local que garante a movimentação livre de colisões em um espaço tridimensional quando deparado com

obstáculos ambos estáticos ou dinâmicos.

Para evitar colisões, a abordagem proposta neste trabalho define que apenas um VANT pode ocupar um hexágono (favo da colmeia) por vez, de forma semelhante a construção de uma célula em uma colmeia real. Então, é necessário ter os registros dos hexágonos que estão sendo atualmente ocupados pelos VANT. Para isso, cada vez que um VANT se move de um hexágono para outro, ele registra ambas informações: onde está e qual o próximo hexágono para onde vai. Desse modo, as colisões são evitadas pois antes de um VANT se mover para um hexágono que faz parte de sua trajetória, ele verifica se o mesmo está ocupado para, então, ir até lá, apenas se estiver livre. Entretanto, estados de bloqueios dos caminhos podem acontecer.

Na abordagem proposta, assume-se que um VANT pode se encontrar em três possíveis estados: “em exploração”, “em deslocamento” ou “parado”. O estado “em exploração” significa que o VANT está fazendo a leitura de todas as faces do hexágono e gerando os dados do mapeamento. O estado “em deslocamento” significa que o VANT está movendo-se para um hexágono para fazer sua exploração. O estado “parado” significa que o VANT não possui nenhum hexágono alvo para exploração alocada a ele, de forma que ele não está se movendo para qualquer hexágono. Situações de bloqueio de caminhos podem acontecer entre dois VANT que estão se deslocando para seus alvos de exploração ou entre um VANT que está se deslocando e o outro está parado.

Um elemento chave para resolver o bloqueio de caminhos nesta abordagem é o Grau de Adjacência (GA) do VANT. O GA é o número de hexágonos adjacentes, direta ou indiretamente, para os quais o VANT pode trafegar em caminhos livres. Para obter o GA, cada VANT verifica quantos hexágonos são diretamente adjacentes ao que ele se encontra, excluindo aqueles que estão ocupados por algum outro VANT. Se o valor de GA é maior que 1, isso significa que há espaço para realizar alguma manobra que libere a passagem para um outro VANT. Se o valor de GA é igual a 1, é calculado o valor de GA deste único hexágono adjacente. O valor encontrado de GA para este hexágono adjacente será o valor de GA, indiretamente, para o hexágono original.

Considerando a existência de um conjunto $SU = \{u_1, u_2, ..., u_n\}$ com n VANT, $u_i \rightarrow u_j$ representa o VANT u_i que quer se mover para o hexágono que se encontra o VANT u_j, GA_i e GA_j representando os seus graus de adjacência, respectivamente. Os casos de bloqueio de caminho que podem ocorrer são:

- **Caso 1 ($u_i \rightarrow u_j$ e $u_j \rightarrow u_i$):**

  Neste cenário, o VANT u_i deseja se mover para o hexágono que encontra-se u_j, e ao

mesmo tempo, o VANT u_j deseja se deslocar para o hexágono onde encontra-se u_i. Aqui, cada VANT calcula seu *GA*, e aquele que possuir o maior valor irá ceder espaço para o outro VANT passar, deslocando-se para um hexágono adjacente ao que ele se encontra;

- **Caso 2 ($u_i \rightarrow u_j$):**

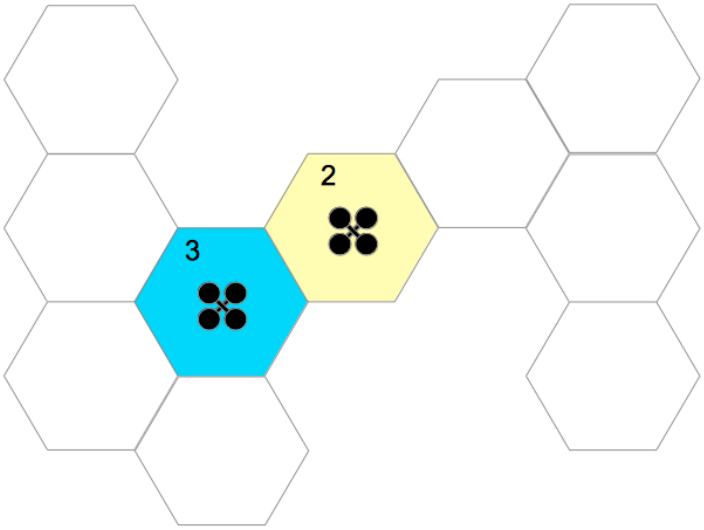
  Neste cenário, apenas o VANT u_i informa que deseja se mover para o hexágono onde encontra-se u_j, o qual não deseja deslocar-se para o hexágono onde está u_i. Neste caso, o VANT u_j pode estar em um estado de "em exploração" ou "parado". Se ele está no estado "em exploração", o VANT u_i irá recalcular um novo caminho buscando desviar do hexágono ocupado por u_j. Se houver apenas um caminho, u_i espera u_j completar sua exploração. Por outro lado, se o VANT u_j está no estado "parado", ele mesmo irá identificar que u_i deseja ir para o local em que ele se encontra. Nesse sentido, u_j irá calcular seu GA_j e comparar o valor GA_i de u_i. Se $GA_j > GA_i$, u_i irá se mover para algum dos seus hexágonos adjacentes livres, e caso contrário, ele irá tentar se mover para um hexágono livre adjacente ao hexágono onde se encontra u_i, o que faz com que eles se encontrem no **caso 1**;

- **Caso 3 ($u_i \rightarrow u_j$, $u_j \rightarrow u_k$ e $u_k \rightarrow u_i$):**

  Neste caso, dois VANT não conseguem mutuamente identificar um bloqueio. Então, é necessário verificar se existe um bloqueio cíclico. Para isso, caso exista algum VANT no hexágono que o u_i deseja mover-se, ele irá verificar se este u_j deseja deslocar-se para um local livre. Se sim, não há bloqueio, se não, ele irá verificar se o local que o VANT u_j deseja deslocar-se é onde ele mesmo se encontra. Caso não seja o mesmo, verificará o local de destino do próximo VANT u_k e, assim, sucessivamente. Porém, caso seja o mesmo hexágono, o bloqueio é detectado. Dada a detecção, o VANT u_i calcula seu GA_i, e caso seja maior que 0, irá dar espaço para a resolução do bloqueio. Depois disso, o caminho de u_i será recalculado e ele continuará na realização de sua tarefa de exploração.

Dado que dois VANT podem estar em um bloqueio de caminho, será feita a comparação entre os valores de GA de cada um. Aquele que possuir o maior valor cederá espaço para o que possuir menor, movendo-se para um dos hexágonos adjacentes para resolver o bloqueio do caminho. Quando o VANT finaliza este movimento, ele refaz sua trajetória para o hexágono alvo da exploração, e então retorna a sua tarefa, isto é, deslocar-se para seu alvo. A figura 29 apresenta um cenário com dois VANT, onde o VANT no hexágono azul possui três hexágonos adjacentes diretos, de modo que o seu valor de GA é 3. O outro VANT que está no hexágono amarelo possui um único hexágono adjacente, entretanto este único possui dois hexágonos adjacentes, fazendo com que o seu GA seja 2.

Figura 29: Graus de Adjacência: o hexágono azul possui GA = 3, enquanto o hexágono amarelo possuir GA = 2.



Fonte: Autoria própria.

4.5 VISUALIZAÇÃO DO MAPA DE FAVOS DE COLMEIA

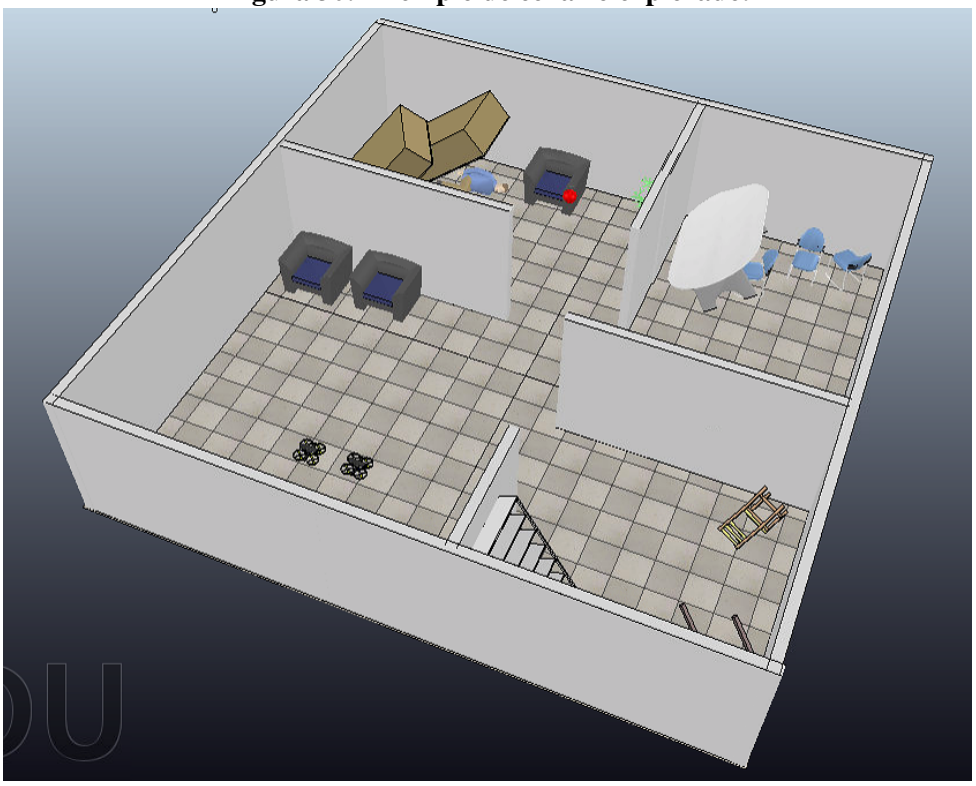
O presente trabalho propõe a representação do mapa topológico gerado em um formato de hexágonos, formando uma estrutura inspirada em favos de colmeias de abelhas. Cada hexágono tem suas paredes representadas por linhas, as quais são contínuas entre hexágonos que não possuem adjacência (ou seja, caminhos bloqueados), e tracejadas para aquelas que as possuem (ou seja, caminho livre).

Além disso, o mapa proposto busca fornecer também informações relacionadas ao ambiente explorado, como por exemplo temperatura do ambiente. Para isso, durante as leituras das faces horizontais do hexágono, a variação de temperatura é analisada a partir de leituras de um sensor específico. Dessa forma, caso a leitura do sensor ultrapasse um valor referencial adotado, esta informação é registrada. Esta variação térmica é representada no mapa topológico a partir da alteração da cor da parede do hexágono que a leitura foi realizada. Assim, caso seja detectada a temperatura acima do valor de referência em uma leitura, independente de haver ou não adjacência, a representação será feita na cor vermelha.

A figura 30 apresenta um exemplo de cenário usado em um estudo de caso (ver seção 5.2), onde a esfera vermelha representa uma fonte de calor, sendo que a temperatura emitida por ela é capturada e identificada no mapa topológico, conforme pode ser visto na parte superior da figura 31.

Na figura 32, é possível ver um mapeamento onde a configuração de altura do

Figura 30: Exemplo de cenário explorado.



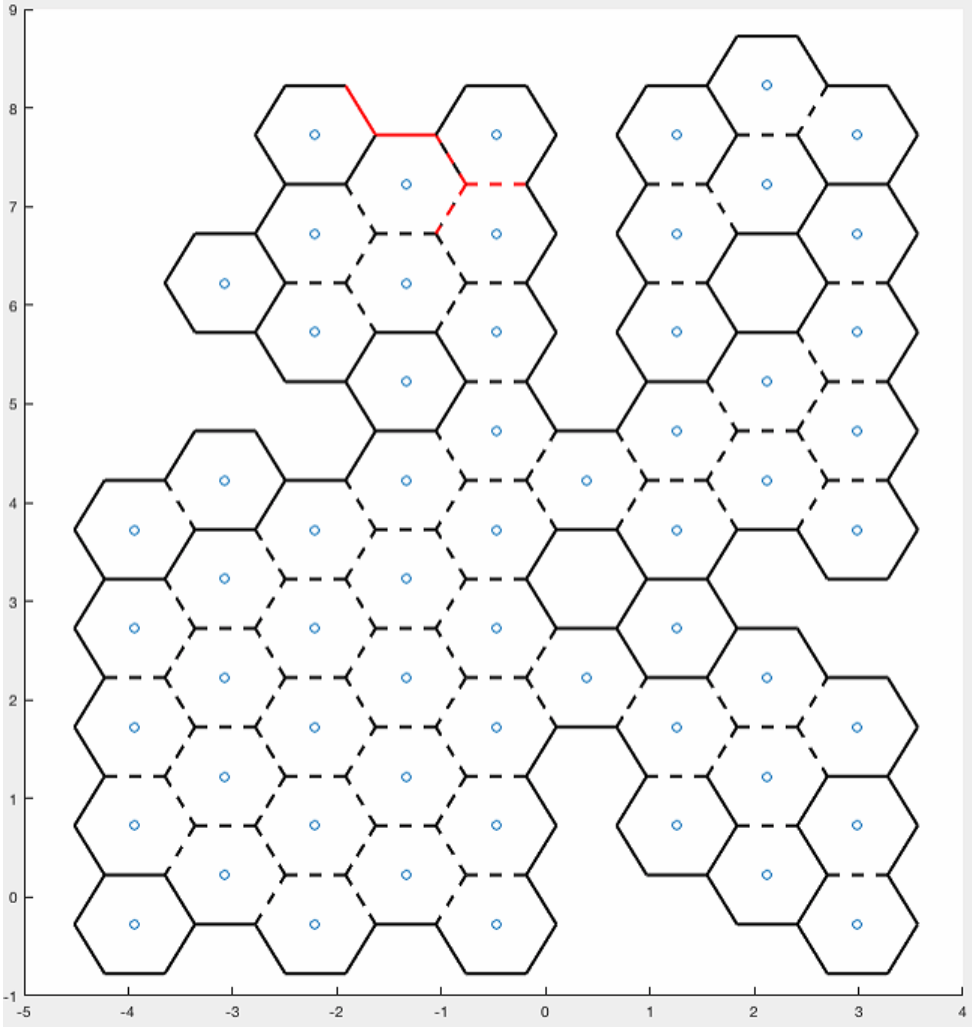
Fonte: Autoria própria.

hexágono e a altitude de atuação dos VANT, definidas previamente, possibilitou a geração de duas camadas de hexágonos no mapa.

Como a exploração pode ser feita em vários níveis, dependendo da configuração e parâmetros adotados, pode haver hexágonos identificados e diferentes pontos do eixo z. A figura 32 apresenta o mapa topológico em favos de colmeia da figura 31 em uma perspectiva em que se pode identificar os diversos hexágonos mapeados.

A partir da identificação de cada hexágono no mapa topológico gerado, pode-se realizar uma verificação individualizada dos favos, onde pode-se visualizar a projeção em cubos 3D bem como imagens geradas dos pontos, em que os valores de temperaturas são superiores a um referencial adotado. As figuras 33 e 34 apresentam uma representação em cubos 3D de um determinado hexágono (que é o ponto identificado com temperatura elevada na figura 31) e a imagem capturada em uma de suas faces, respectivamente.

Figura 31: Mapa topológico em favos de colmeia: visão superior.



Fonte: Autoria própria.

4.6 DISCUSSÃO

Esse capítulo apresentou o método bioinspirado para exploração e mapeamento de ambientes internos. De forma geral, busca-se que um time de robôs aéreos façam a exploração coordenada do espaço em um ambiente interno desconhecido, mapeando os locais de livre circulação para que equipes de busca e resgate tenham conhecimento prévio que auxilie na tomada de decisão sobre ações a serem feitas. A forma de construção do mapa topológico é inspirado em como as abelhas constroem suas colmeias, em que os hexágonos são mapeados como vértices de um grafo e as adjacências como arestas. A definição das adjacências considera as leituras efetuadas por sensores de distância e as projeções do ambiente em uma visão 3D é feita a partir de leituras realizadas por câmeras RGB-D. Informações adicionais do ambiente podem ser obtidas através de sensores diversos, como o de temperatura que é utilizado neste

trabalho. A representação da temperatura ocorre na coloração diferenciada na marcação das adjacências entre os hexágonos.

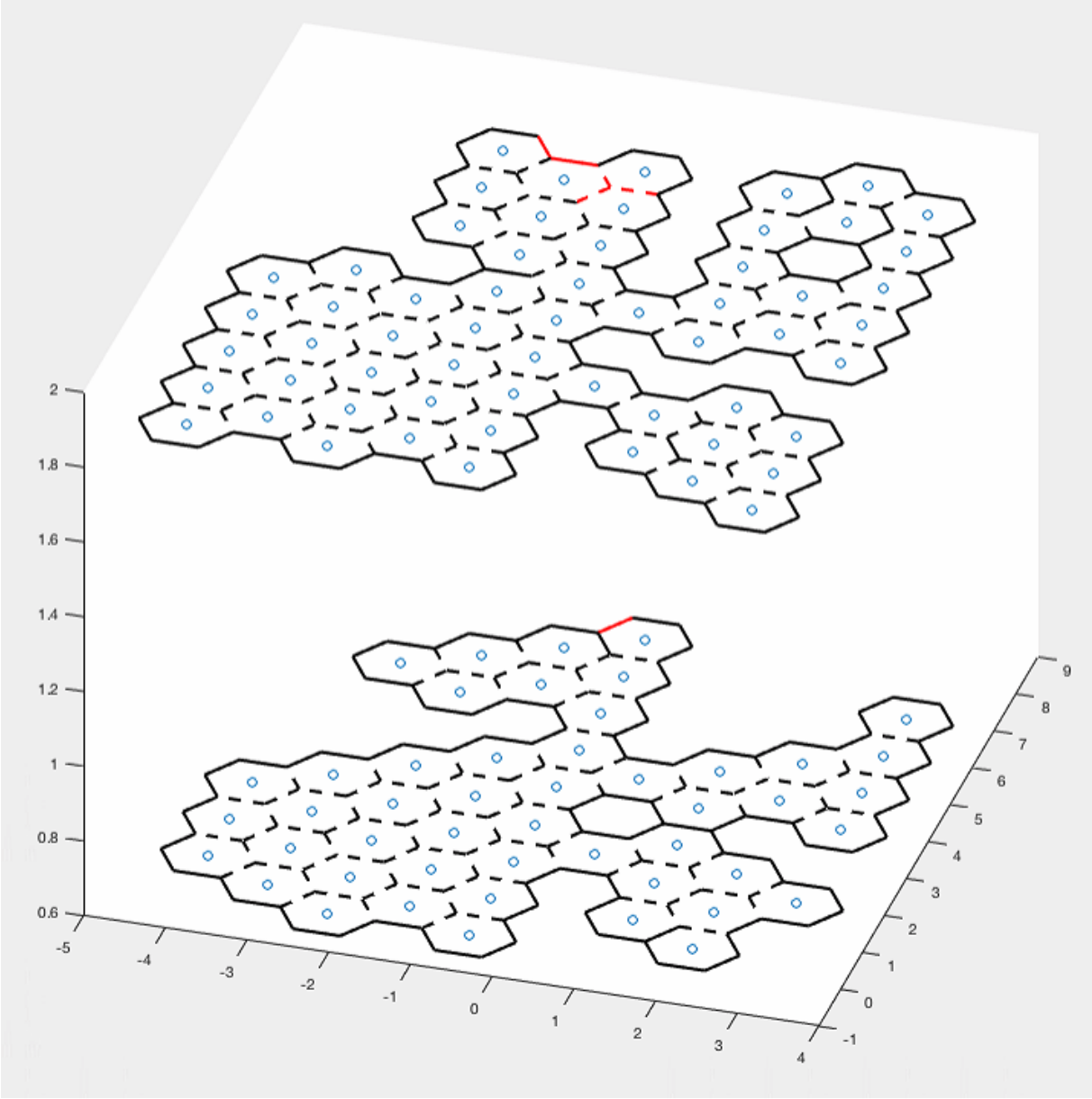
Para realizar a exploração, três estratégias foram implementadas: FIFO, DE e DER. Essas estratégias buscam organizar a forma de distribuição das tarefas de exploração. Por mais que o custo computacional para processar a estratégia FIFO na definição de um hexágono tenha custo computacional $\Theta(1)$ enquanto DE e DER tenha custo de $O(m)$, é importante considerar que ainda é necessário contabilizar o custo de deslocamento do VANT até o determinado hexágono definido pela estratégia adotada. Desse modo, a estratégia DER buscará o hexágono mais próximo a posição atual do VANT, reduzindo tempo de deslocamento. Dada a definição de um hexágono a ser explorado por um VANT, ele irá deslocar-se apenas por espaços conhecidos e livres de circulação. Como vários VANT estão circulando no mesmo ambiente e, por vezes, com trajetórias se cruzando, podem acontecer bloqueios desses caminhos. Um algoritmo baseado no grau de adjacência do hexágono em que o robô se encontra influenciará na resolução desse bloqueio.

Algumas limitações ainda são encontradas nesse método. A suposição de que os únicos elementos dinâmicos no ambiente sejam os VANT limita o uso do método proposto em um cenário em que ocorreu alguma catástrofe e o prédio pode não estar estável devido a constantes desabamentos. Assim, escombros podem cair em locais que teriam sido mapeados como livres de circulação, como portas, janelas e corredores, necessitando fazer a identificação destes novos obstáculos e, consequente atualização do mapa. Os erros de odometria que são desconsiderados aqui podem refletir em uma representação distorcida do ambiente na visualização em cubos 3D, pois o posicionamento da câmera RGB-D não seria precisa. Além disso, o posicionamento do VANT no local correto que representa o centro de um hexágono no ambiente físico também poderia ser impactado pelo erro de odometria.

Desconsiderar o consumo de energia e o nível de carga de bateria pode significar que um VANT deixe de completar a sua tarefa de exploração e mapeamento, levando a “buracos” no mapa que nunca serão “descobertos” , pois aquele espaço fica alocado ao VANT que desligou por falta de energia.

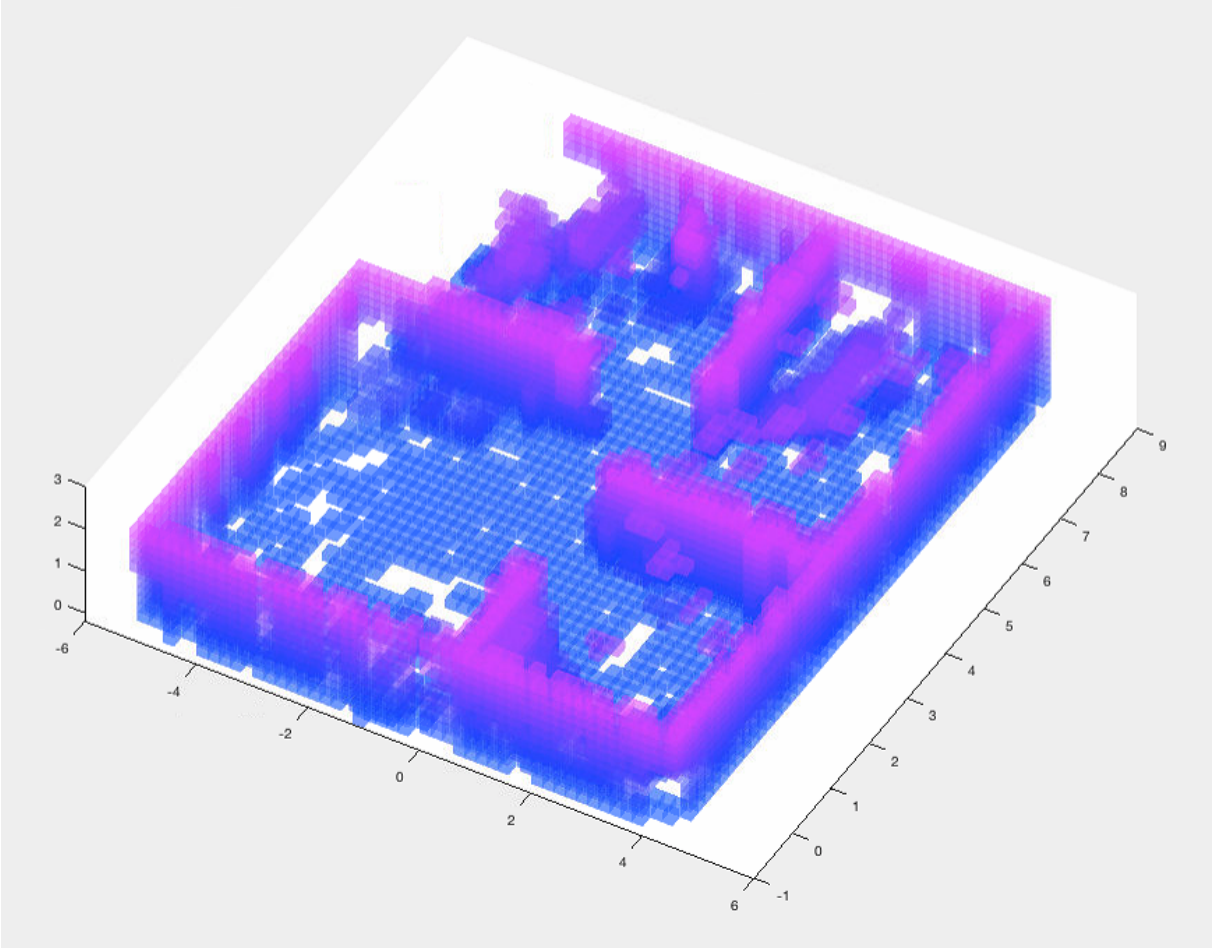
No entanto, apesar destas simplificações, pode-se observar na discussão dos resultados apresentados no próximo capítulo que o método proposto traz contribuições bastante relevantes para a aplicação de múltiplos VANT na exploração de ambientes internos e mapeamento de informações que podem ser usadas por equipes de socorristas no planejamento de ações em missões de busca e resgate.

Figura 32: Mapa topológico em favos de colmeia: visão em perspectiva.



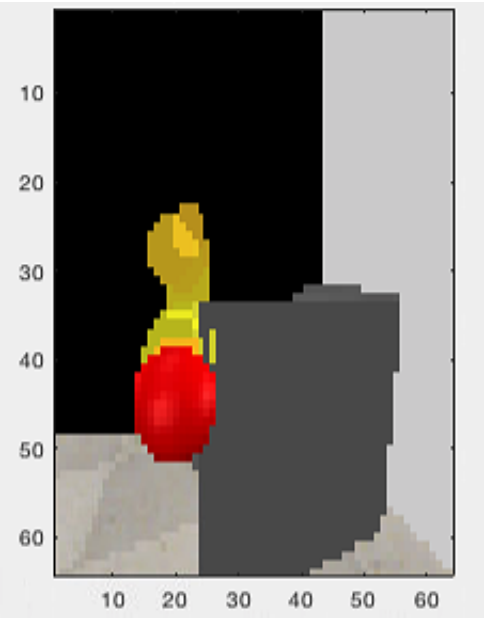
Fonte: Autoria própria.

Figura 33: Visão em cubos 3D.



Fonte: Autoria própria.

Figura 34: Imagem capturada a partir da detecção de temperatura pelo sensor térmico.



Fonte: Autoria própria.

5 EXPERIMENTOS E RESULTADOS

O presente capítulo apresenta detalhes sobre os experimentos realizados para avalizar e validar este trabalho. São apresentadas as ferramentas utilizadas, configurações de VANT, os ambientes simulados e os resultados obtidos da execução de cada um dos algoritmos FIFO, DE e DER para alocação de tarefas de exploração.

5.1 VISÃO GERAL DO AMBIENTE DE SIMULAÇÃO

A ferramenta V-REP (*Virtual Robot Experimentation Plataform*) (ROBOTICS, 2015) é um simulador de ambientes e equipamentos robóticos desenvolvido pela Coppelia Robotics. Nele é possível realizar simulações de elementos fixos ou móveis, como braços robóticos, robôs terrestres, aéreos e aquáticos. Por mais que não seja uma ferramenta de código aberto, é disponibilizada uma versão educacional gratuita, a qual pode ser executada em sistemas Windows, Linux e MacOS.

O V-REP, além de já disponibilizar diversos modelos prontos, possibilita a importação de modelos 3D. Além disso, ele possibilita a integração com outras tecnologias, baseando-se em uma arquitetura de controle distribuída, onde cada objeto pode ser controlado por meio de *scripts*, *plugins*, APIs de clientes remotos ou até mesmo com nós ROS (*Robot Operating System*) (MARTINEZ; FERNNDEZ, 2013). Além das diversas formas de comunicação, ele aceita diversas linguagens de programação, como C/C++, Matlab, Java, Urbi, Python e Lua, sendo esta última a padrão da ferramenta.

Para simular a realidade física dos ambientes, o V-REP disponibiliza diferentes motores físicos, entre eles o *Open Motor Dynamics* (ODE), o *Bullet physics library*, o *Vortex Dynamics engine* e o *Newton Dynamics engine*. Essa variedade de motores possibilita que o usuário escolha o que melhor se adeque às necessidades de sua simulação: graus de precisão, velocidade ou suporte às diversas características.

O motor ODE é um motor físico de código aberto com dois principais componentes:

dinâmicas de corpos rígidos e detecção de colisão. É um motor mais voltado para jogos. Já o motor *Bullet* é um motor físico de código aberto voltado para detecção de colisões 3D, dinâmica de corpos rígidos e macios, sendo também voltado para jogos e efeitos visuais em filmes. O *Vortex Dynamics* é um motor físico de código fechado comercial, que produz simulações físicas de alta qualidade. Fornece parâmetros do mundo real para diversas propriedades físicas, sendo aplicado em indústrias de alto desempenho/precisão e pesquisas. O motor físico *Newton Dynamics* implementa uma solução determinística não baseada em métodos iterativos. É voltado tanto para jogos, mas também para simulações físicas de tempo real. Sua atual implementação está em versão beta. O motor físico utilizado neste trabalho nas simulações foi o *Bullet*.

Neste trabalho, foi utilizado o ambiente de simulação V-REP integrado com a ferramenta Matlab. No simulador serão definidos os VANT com seus devidos controles de movimento, configuração de sensores e demais componentes do cenário utilizado para os experimentos. No Matlab foram implementados todos os controles de construção dos mapas e organização da exploração como, por exemplo, as tarefas que os VANT deverão realizar.

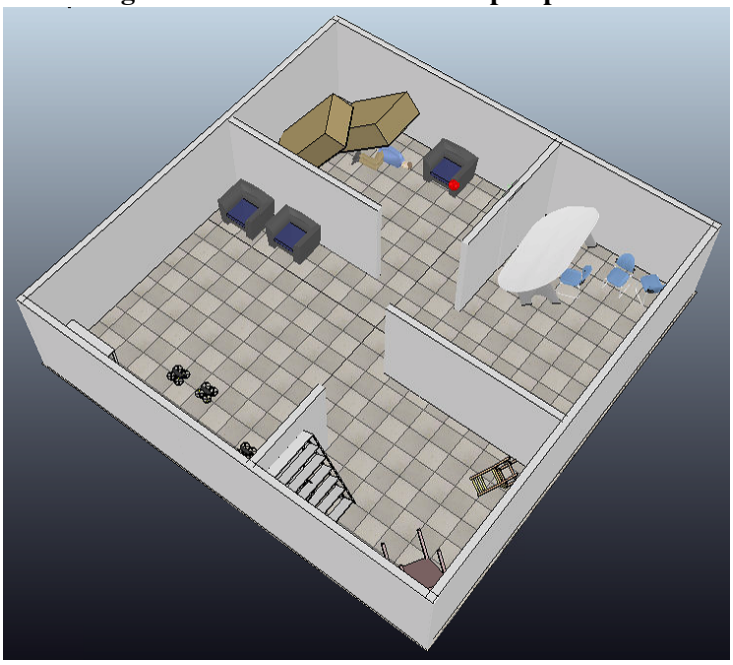
A integração realizada entre V-REP e Matlab foi feita a partir da API disponibilizada na distribuição do simulador. Para isso, é necessário que elas estejam disponíveis para o código Matlab que irá se conectar. As bibliotecas são denominadas *"remoteApi.so", "remoteApi.dylib"* e *"remoteApi.dll"*, para execução em sistemas operacionais Linux, MacOS e Microsoft Windows, respectivamente. Com a configuração do *script* no V-REP e as APIs disponíveis, basta definir as conexões nos códigos Matlab para realizar a interação com os componentes do modelo simulado.

Para realizar os experimentos que visam avaliar os algoritmos desenvolvidos para a exploração e o mapeamento de ambientes internos, foram criados dois modelos de ambientes simulados, com paredes, corredores, mobiliário, uma pessoa e um foco de calor. Os dois ambientes simulados possuem uma dimensão de $10m^2$. As figuras 35 e 36 apresentam o cenário 1 simulado em visões superiores e de perspectiva, respectivamente. As figuras 37 e 38 apresentam as mesmas visões para o cenário 2. As esferas em vermelho nos dois cenários representam fontes de calor, as quais são detectadas pelos sensores de temperatura de cada VANT.

Figura 35: Cenário 1: visão superior.

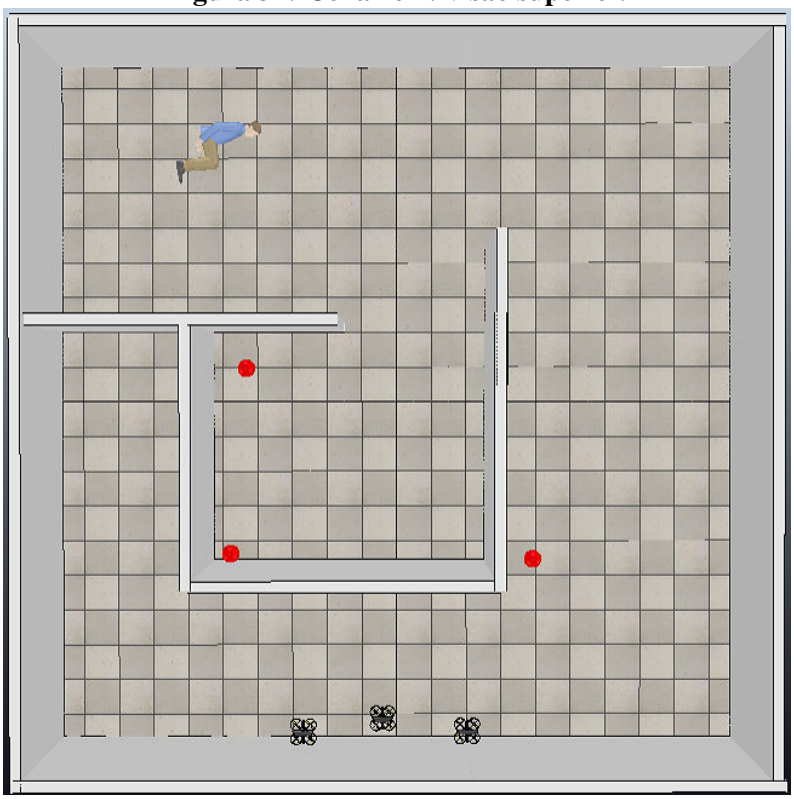
Fonte: Autoria própria.

Figura 36: Cenário 1: visão em perspectiva.



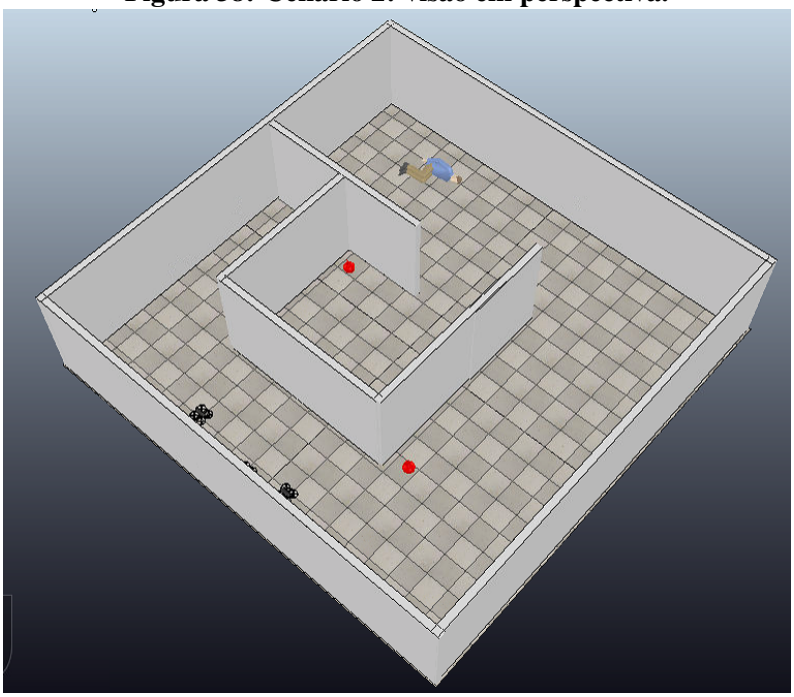
Fonte: Autoria própria.

Figura 37: Cenário 2: visão superior.



Fonte: Autoria própria.

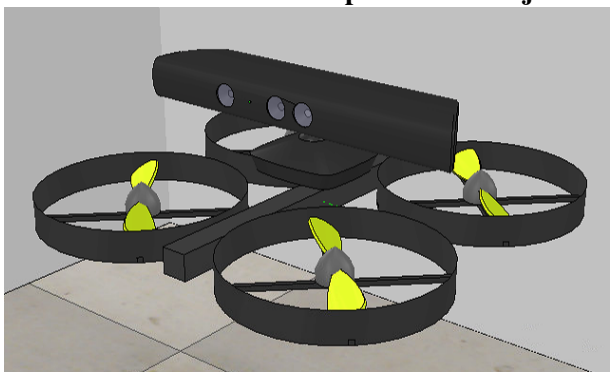
Figura 38: Cenário 2: visão em perspectiva.



Fonte: Autoria própria.

Os modelos de VANT utilizados são de topologia V-TOL quadrotor, todos semelhantes em configuração. Dessa forma, cada VANT é composto por: (i) Três sensores de distância: frontal, superior e inferior, (ii) sensor térmico e (iii) sensor RGB-D. A figura 39 apresenta o modelo de VANT utilizado.

Figura 39: Modelos de VANT utilizado: base disponível no conjunto de modelos do v-REP.

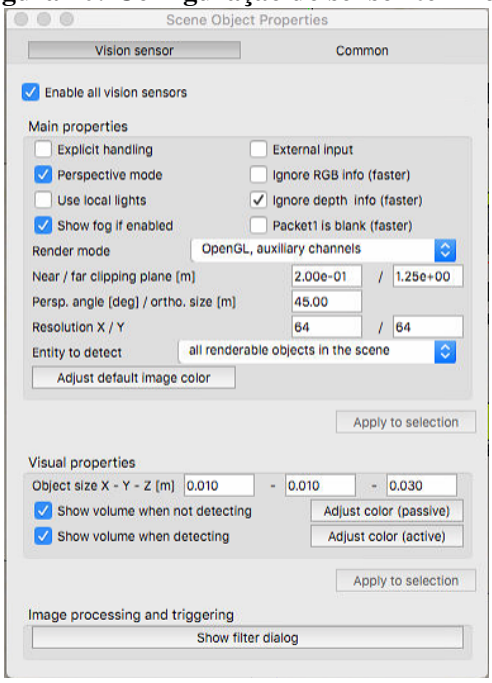


Fonte: Autoria própria.

A leitura de dados dos sensores de distância é feita a partir do comando `vrep.simxReadProximitySensor`, onde o retorno desta função trará, além de outras informações, uma validação se a leitura foi realizada com sucesso ou houve algum erro e o ponto detectado.

O sensor térmico é posicionado para fazer a leitura da parte frontal do VANT. O V-REP não possui um modelo de sensor térmico nativo, entretanto, ele possibilita o uso de um sensor de visão (como uma câmera RGB) e configurar canais auxiliares para essa detecção, onde o canal da cor vermelha (red) indica a leitura de temperatura (que varia 0 a 1, e 0.5 é a temperatura ambiente). Essa configuração é feita no campo *"Render mode"* com o valor *"OpenGL, auxiliary channels"*. A configuração da distância que o sensor irá ler é configurada no campo *"Near/far clipping plane"*. Para os experimentos, esta configuração foi alimentada com o valor de $1.25m$. Além da distância, é possível configurar um ângulo de leitura do sensor. Esta configuração é feita no campo *"Persp. angle/ortho. size"*, e para o experimento foi definido um valor de 45^o. A resolução da leitura é configurada no campo *"Resolution X/Y"*, e para as simulações realizadas foram definidos os valores de $X = 64$ e de $Y = 64$.

Figura 40: Configuração do sensor térmico.

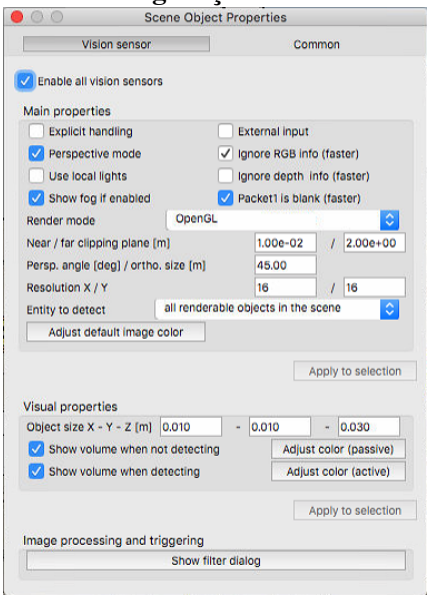


Fonte: Autoria própria.

A leitura de dados do sensor térmico é feita a partir do comando `vrep.simxReadVisionSensor`, que possui valores de retorno como o validador da leitura e um conjunto de dados que incluem, devido a configuração no V-REP do canal auxiliar, o valor de temperatura no campo do valor de leitura da cor vermelha (red). Caso seja detectada a temperatura acima de um limite definido, é acionado a leitura de uma câmera RGB, a qual nesse modelo está acoplada juntamente com o modelo de câmera RGB-D utilizado. Nesse caso, o comando para capturar a imagem do ponto em questão é `vrep.simxGetVisionSensorImage2`, que também retorna um validador, a resolução da imagem capturada e uma matriz que armazena os dados da imagem.

Por fim, a captura de informações do sensor RGB-D é feita a partir do comando `vrep.simxGetVisionSensorDepthBuffer2`, o qual retorna o validador da leitura, a resolução e uma matriz que armazena o valor de leitura de profundidade de cada ponto dentro da configuração do sensor. A figura 41 apresenta a tela de configuração do sensor RGB-D no ambiente V-REP. Os campos configuráveis são os mesmos disponíveis na configuração do sensor térmico, porém neste caso o campo *"render mode"* possui o valor *"OpenGL"* apenas, e as leituras RGB são ignoradas. A configuração adotada para as simulações, em todos os VANT, foi de $2m$ para distância, 45^o para o grau de abertura do sensor e resolução de $X = 16$ e $Y = 16$.

Figura 41: Configuração do sensor RGB-D.



Fonte: Autoria própria.

5.2 EXPERIMENTOS

Como forma de validar o método proposto, foram realizados seis experimentos, os quais consistem em realizar simulações nos dois cenários apresentados na seção 5.1. Para cada cenário, o algoritmo de definição do próximo hexágono a ser explorado variou conforme as três estratégias apresentadas na seção 4.3.3: FIFO, distância euclidiana (DE) e distância euclidiana relativa (DER). Essa variação possibilita averiguar se a configuração da distribuição de tarefas de exploração influencia no tempo geral da exploração, bem como na intensidade do tráfego no ambiente. Dessa forma, serão avaliados os deslocamentos realizados por cada VANT por influenciarem no tráfego dentro do ambiente, podendo afetar o número de bloqueios de caminhos encontrados por cada um deles. A resolução desses bloqueios consiste em: (i) suspender de forma temporária o deslocamento de um VANT para o hexágono destino, (ii) liberar a passagem para um dos robôs envolvidos no bloqueio e, então, (iii) recalcular a trajetória e retomar o deslocamento para o hexágono destino.

O equipamento utilizado para fazer o processamento das simulações foi: CPU Intel Xeon de 3.33 GHz 6 Núcleos, 6 GB 1333 MHz DDR3 de memória, e processamento gráfico com GPU ATI Radeon HD 5770 1024 MB[1]. Inicialmente, foram realizados testes em um equipamento laptop com CPU Intel Core i5 2.5 GHz, com 16 GB 1600 MHz DDR3 de memória, e processamento gráfico com GPU Intel HD Graphics 4000 com 1536 MB, entretanto esse

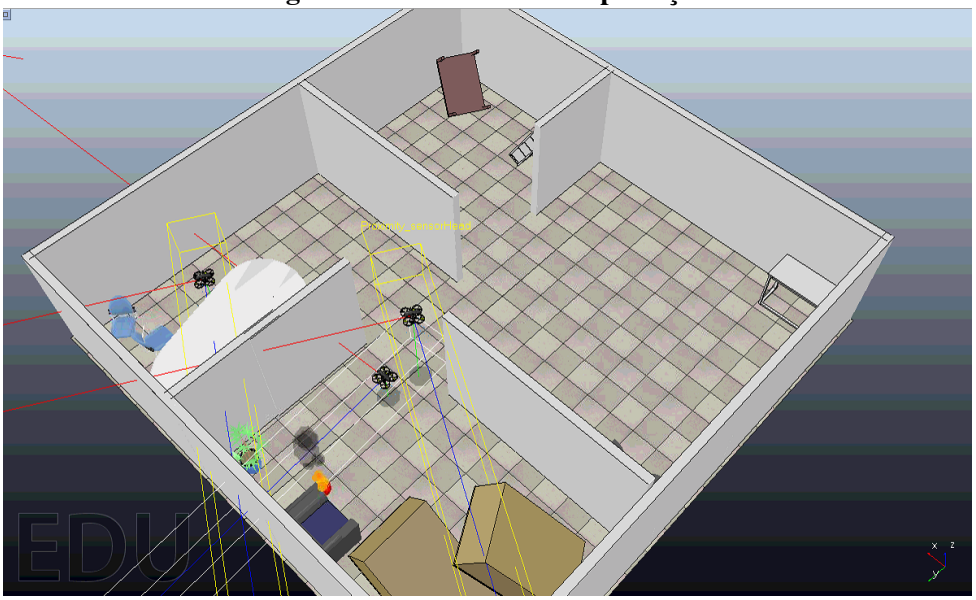
[1]Disponibilizado pelo Instituto Federal do Paraná (IFPR) - Campus Cascavel.

equipamento não suportou executar a simulação com dois VANT simulados ao mesmo tempo.

Todas as simulações realizadas fazem uso de três VANT na exploração do cenário. Simulações com dois VANT foram feitas para as estratégias de *First In First Out* (FIFO) e Distância Euclidiana (DE) publicados em Rosa et al. (2020) (ver apêndice D). Apesar da capacidade de processamento do equipamento utilizado, a simulação de mundo, dos três VANT e suas dinâmicas demandam alto processamento pelo simulador V-REP, podendo muitas vezes ocorrer falhas de comunicação. Para contornar esse problema, mecanismos de detecção de falhas de comunicação foram implementados e, assim, as conexões são reestabelecidas quando as falhas ocorrem.

Nas simulações, considera-se que os VANT terão sempre acesso aos dados globais, mesmo que estejam sendo tratadas as falhas de comunicação entre V-REP e Matlab. Para que os VANT acessem estas estruturas globais mantendo a integridade das informações, um controle de acesso concorrente é feito, de forma que em ações de escrita em estrutura de dados compartilhadas seja feito por apenas um VANT por vez. Cada VANT possui um identificador gerado pelo V-REP (*"handler"*), que é utilizado neste controle de acesso concorrente. Por exemplo, suponha que o VANT com identificação 1 deseja inserir um hexágono na "lista de hexágonos não visitados". Para ele ter acesso a essa lista, ele insere esse seu *"handler"* em uma fila de espera para o acesso a estrutura. Assim, quando ele obtiver o acesso, faz o seu processamento, e então retira sua identificação da lista de espera. A figura 42 apresenta o cenário 1 com os três VANT fazendo a exploração do ambiente.

Figura 42: Cenário 1 em exploração.



Fonte: Autoria própria.

5.3 RESULTADOS

5.3.1 EXPLORAÇÃO DO AMBIENTE

As simulações realizadas nos dois cenários propostos, variando o algoritmo de definição de hexágono a ser explorado definidos na seção 4.3.3 (FIFO, DE e DER), geraram os mapas que podem ser vistos nas figuras 44, 45 e 46 para o cenário 1, e nas figuras 61, 62 e 63 do apêndice B para o cenário 2. As figuras 44a, 45a, 46a, 61a, 62a e 63a apresentam uma visão superior dos mapas gerados, enquanto as figuras 44b, 45b, 46b, 61b, 62b e 63b apresentam o mapa topológico gerado em perspectiva, de forma semelhante ao apresentado na figura 22. Nesses cenários foi definido um valor de altitude máxima de atuação dos VANT de $2,5m$, o raio do hexágono $0,5m$ com altura de $1m$. Portanto, foram alcançada duas camadas de hexágonos. Caso os sensores de distância, superior ou inferior, indiquem que há espaço para deslocamento em uma determinada área, ela será inserida no mapa topológico da mesma forma que são inseridas as adjacências identificadas pelo sensor frontal do VANT. Em todos os experimentos, todos os VANT iniciam a exploração do ambiente pelo primeiro hexágono mapeado, o qual é tomado como referência pelas estratégias DE e DER.

São definidas algumas premissas para os valores de tempo de execução do método nos experimentos. Esses valores representam estimativas dos tempos que um VANT real necessita para realizar cada ação. São elas:

- unidade tempo de deslocamento: é o tempo gasto para um VANT se deslocar do centro de um hexágono para o centro de outro hexágono imediatamente adjacente. Considerando a velocidade[2] e aceleração[3] médias de um VANT comercial, o tempo de deslocamento estimado é de $5s$ (OLIVEIRA; WEHRMEISTER, 2018);
- unidade tempo de exploração do hexágono: é o tempo necessário para que um VANT faça a leitura das laterais do hexágono que está explorando e identifique as adjacências. É considerado um valor de tempo de mapeamento de 30 segundos;
- tempo de resolução de bloqueio: é o tempo necessário para que um bloqueio entre trajetórias de VANT seja resolvido. É considerado o valor de 4 segundos;

As tabelas 3, 4 e 5 apresentam os dados obtidos nos experimentos realizados, onde a tabela 3 apresenta os dados das simulações dos cenários 1 e 2 com a utilização de apenas um

[2] https://dronerush.com/how-fast-can-a-drone-fly-science-of-flight-10953/
[3] https://www.research-drone.com/en/extreme_climb_rate.html

VANT, enquanto as tabelas 4 e 5 apresentam os dados das simulações com o uso de três VANT, para os cenários 1 e 2, respectivamente. O campo D representa o número de deslocamentos realizados pelo VANT, B é o número de bloqueios tratados pelo VANT e H é o número de hexágonos explorados pelo VANT até que o ambiente seja completamente explorado conforme a condição de parada descrita na seção 4.3.2. Considerando as premissas estabelecidas, e que o tempo de execução dos experimentos é o maior valor de tempo de execução obtido por um dos três VANT, temos os tempos de execução da exploração e mapeamento nos experimentos T, que são apresentados nas tabelas 6 e 8 para simulações com um VANT e com três VANT, respectivamente. Os valores foram calculados da seguinte forma:

$$T = (D \times 1segundos) + (H \times 30segundos) + (B \times 4segundos) \tag{18}$$

Tabela 3: Dados dos experimentos com apena um VANT.

	Cenário 1			Cenário 2		
Estratégia	**D**	**B**	**H**	**D**	**B**	**H**
FIFO	318	0	103	377	0	132
DE	375	0	103	449	0	131
DER	142	0	103	190	0	128

Fonte: Autoria própria.

Tabela 4: Dados dos experimentos com três VANT- Cenário 1.

	VANT 1			VANT 2			VANT 3		
Estratégia	**D**	**B**	**H**	**D**	**B**	**H**	**D**	**B**	**H**
FIFO	151	7	35	152	6	34	161	7	32
DE	111	5	33	96	4	33	99	7	33
DER	45	0	37	46	0	32	58	1	33

Fonte: Autoria própria.

Tabela 5: Dados dos experimentos com três VANT - Cenário 2.

	VANT 1			VANT 2			VANT 3		
Estratégia	**D**	**B**	**H**	**D**	**B**	**H**	**D**	**B**	**H**
FIFO	182	4	45	183	11	42	189	8	41
DE	159	23	42	170	19	44	166	11	42
DER	59	0	46	68	0	41	64	0	43

Fonte: Autoria própria.

Por sua vez, os tempos de execução das simulações são apresentados nas tabelas 7 e 9, para simulações com um VANT e com três VANT, respectivamente. O percentual de redução apresentado em ambas as tabelas é coerente, mostrando que as estimativas do tempo real de

Tabela 6: Tempo estimados de execução das simulações realizadas com um VANT.

Estratégia	Cenário 1	Diferença - FIFO	Cenário 2	Diferença - FIFO
FIFO	00:56:48	-	01:12:16	-
DE	00:57:45	+1,67%	01:12:58	+0,96%
DER	00:53:51	-5,16%	01:07:09	-7,07%

Fonte: Autoria própria.

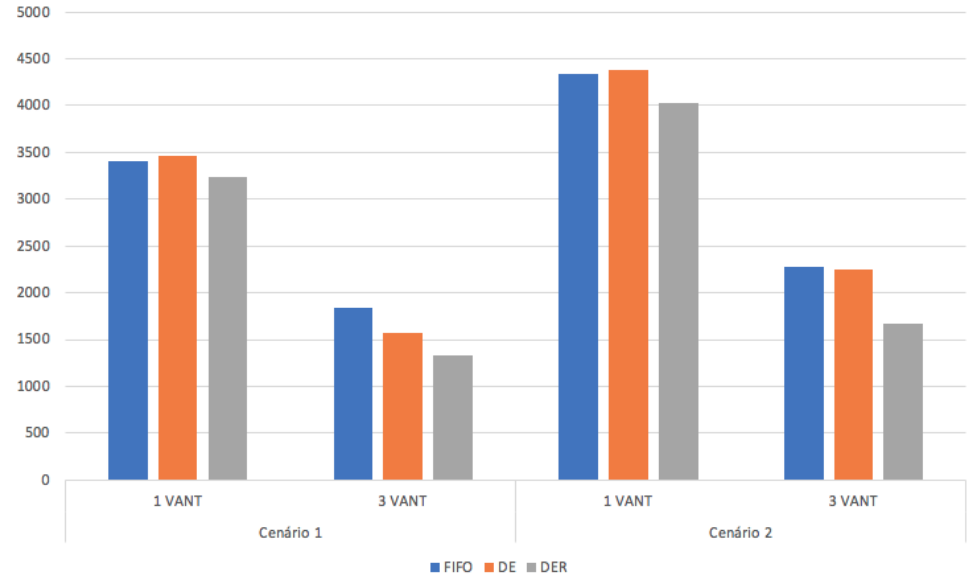
Tabela 7: Tempo de execução das simulações realizadas no V-REP com um VANT.

Estratégia	Cenário 1	Diferença - FIFO	Cenário 2	Diferença - FIFO
FIFO	12:20:01	-	14:40:05	-
DE	12:50:56	+4,17%	15:03:47	+2,69%
DER	11:12:43	-9,09%	12:03:31	-17,78%

Fonte: Autoria própria.

exploração são condizentes com o comportamento do método proposto durante a simulação. Os valores de tempo de exploração apresentados nas tabelas 8 e 9 tanto para o cenário 1 quanto para o cenário 2 consideram o tempo total da simulação (do momento que o primeiro VANT inicia a exploração e mapeamento do primeiro hexágono até o momento que a *lista de hexágonos não visitados* fica vazia, conforme é apresentado na seção 4.3.2), de forma que todos os VANT finalizam suas execuções ao mesmo tempo. A figura 43 apresenta uma comparação dos tempos de simulação das diferentes estratégias definidas na seção 4.3.3, baseado nas informações das tabelas 6 e 8.

Figura 43: Comparação dos tempos de simulação estimados (em segundos) para os cenários 1 e 2 em experimentos com 1 e 3 VANT.



Fonte: Autoria própria.

Tabela 8: Tempo estimados de execução das simulações realizadas com três VANT.

Estratégia	Cenário 1	Diferença - FIFO	Cenário 2	Diferença - FIFO
FIFO	00:30:33	-	00:37:56	-
DE	00:26:05	-14,62%	00:37:26	-1,31%
DER	00:22:15	-27,16%	00:27:55	-26,40%

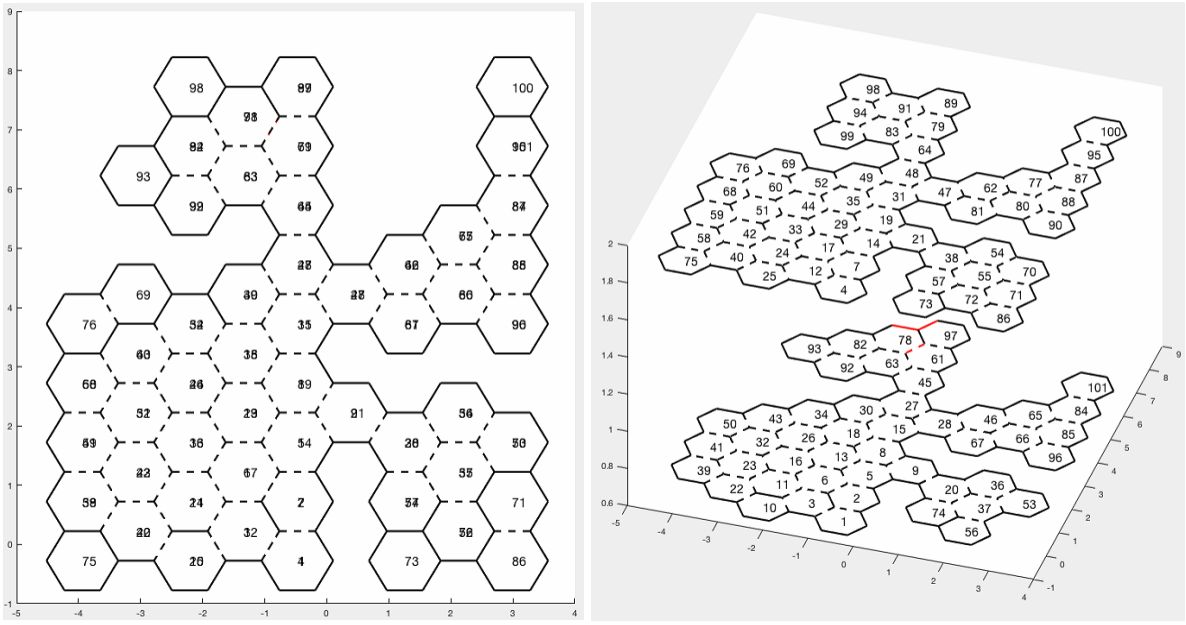
Fonte: Autoria própria.

Tabela 9: Tempo de execução das simulações realizadas no V-REP com três VANT.

Estratégia	Cenário 1	Diferença - FIFO	Cenário 2	Diferença - FIFO
FIFO	03:40:47	-	5:35:25	-
DE	03:20:57	-8,98%	05:30:16	-1,53%
DER	02:49:52	-23,06%	04:13:10	-24,52%

Fonte: Autoria própria.

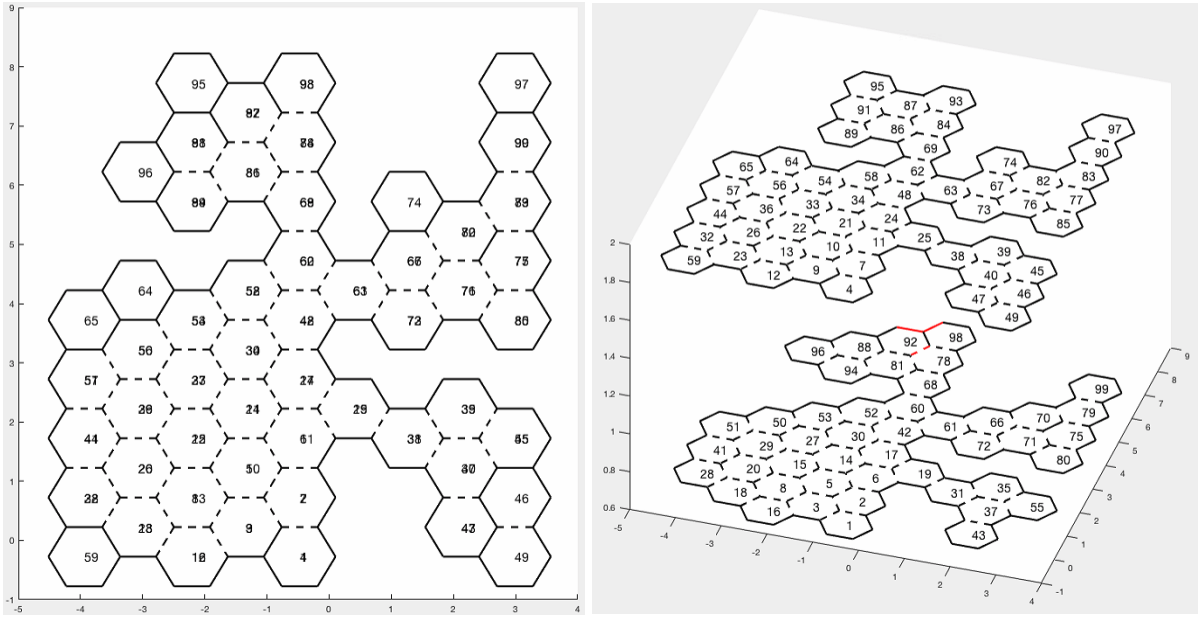
Figura 44: Mapeamento cenário 1 - FIFO.



(**a**) Visão superior. (**b**) Visão em perspectiva.

Fonte: Autoria própria.

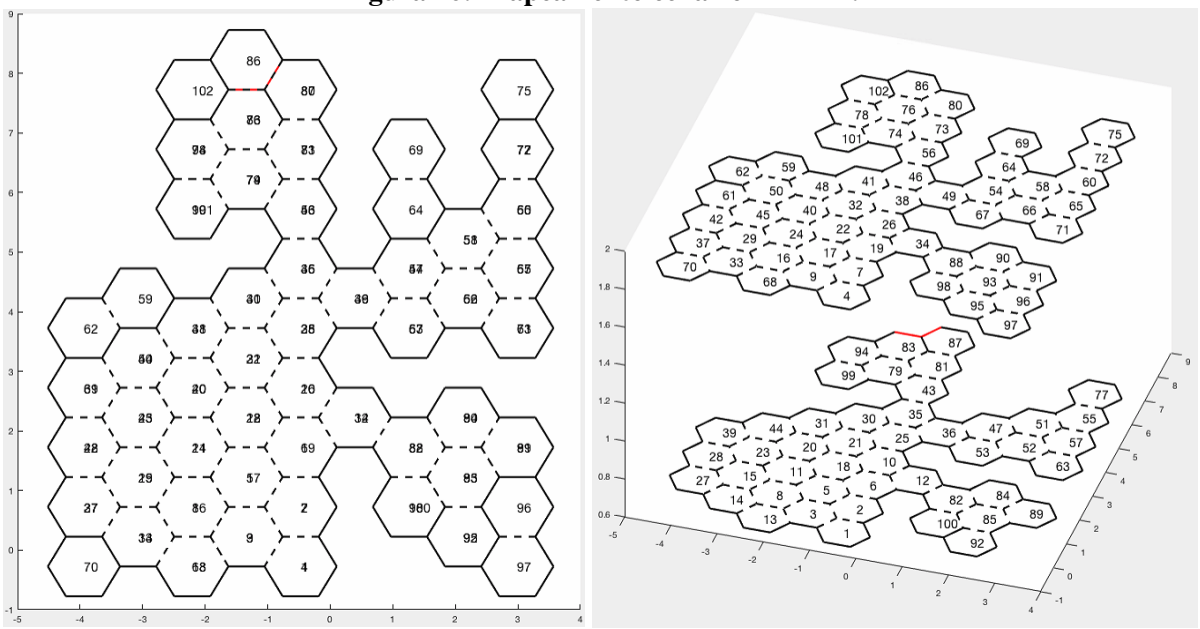
Figura 45: Mapeamento cenário 1 - DE.



(**a**) Visão superior. (**b**) Visão em perspectiva.

Fonte: Autoria própria.

Figura 46: Mapeamento cenário 1 - DER.



(**a**) Visão superior. (**b**) Visão em perspectiva.

Fonte: Autoria própria.

O processo de deslocamento de um VANT consiste em ele sair do ponto central do hexágono onde ele se encontra e ir para o centro do próximo hexágono adjacente, dentro de uma trajetória calculada. Nas simulações realizadas os números de deslocamentos foram registrados

e são apresentados nas tabelas 10 e 11, enquanto o gráfico da figura 47 mostra visualmente as diferenças destes valores. Nota-se uma redução considerável no número de deslocamentos quando é comparado o algoritmo DER com os algoritmos DE e FIFO, para ambos os cenários. As figuras 48a, 49a, 50a, 52a, 52a e 53a apresentam linhas indicando o espaço que cada VANT moveu-se dentro do experimento em cada cenário, sempre dentro da área mapeada. As figuras 48b, 49b, 50b, 51b, 52b e 53b apresentam a ordem de hexágonos explorados em cada estratégia (FIFO, DE e DER), respectivamente. A linha azul representa o VANT 1, a vermelha o VANT 2 e a verde o VANT 3.

Tabela 10: Número de deslocamentos nas simulações com 1 VANT.

Estratégia	Cenário 1	Diferença - FIFO	Cenário 2	Diferença - FIFO
FIFO	318	-	377	-
DE	375	+17,92%	449	+19,09%
DER	142	-55,34%	190	-49,60%

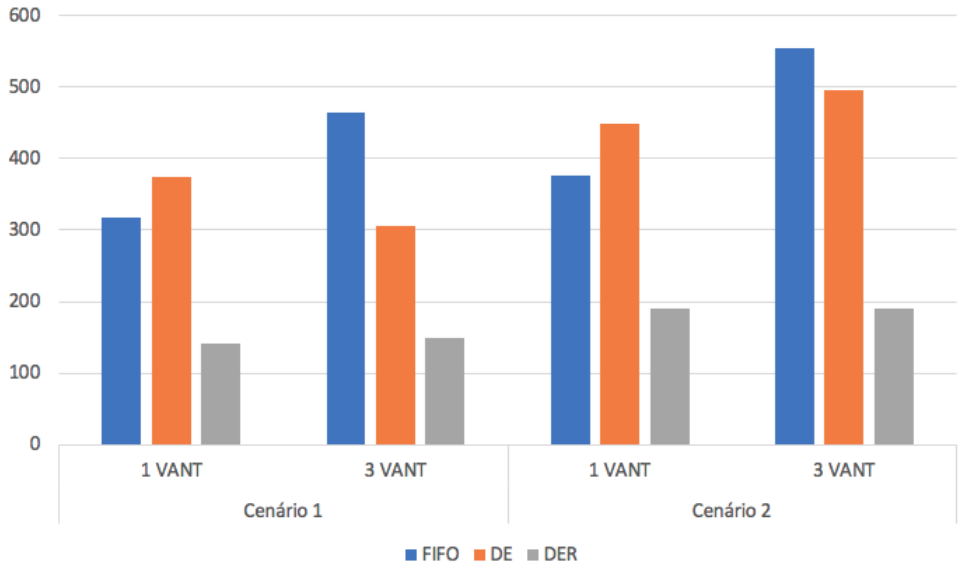
Fonte: Autoria própria.

Tabela 11: Número de deslocamentos nas simulações com 3 VANT.

Estratégia	Cenário 1	Diferença - FIFO	Cenário 2	Diferença - FIFO
FIFO	464	-	554	-
DE	306	-34,05%	495	-10,6%
DER	149	-67,88%	191	-65,52%

Fonte: Autoria própria.

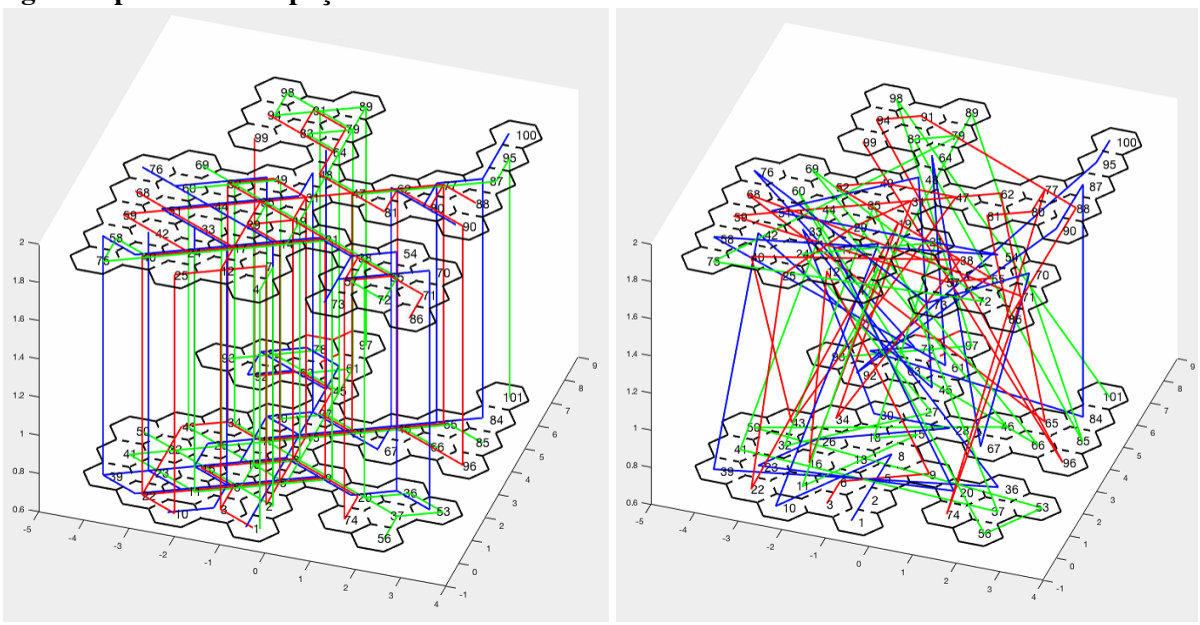
Figura 47: Comparação do número de deslocamentos realizados nas simulações para os cenários 1 e 2 em experimentos com 1 e 3 VANT.



Fonte: Autoria própria.

Na figura 47, é possível notar que a estratégia DER reduziu significativamente o número de deslocamentos do ambiente em comparação com as estratégias FIFO e DE. O maior número de deslocamentos nas estratégias FIFO e DE para as simulações com três VANT em relação as simulações com um único VANT reflete como sendo consequência do número de bloqueios de caminhos que necessitaram de resolução. Como na estratégia DER os VANT ficam concentrados em áreas específicas, resoluções de bloqueio de caminhos são menores, influenciando em um número menor de deslocamentos para esta estratégia. Ainda, a quantidade de deslocamentos para a estratégia DER ficaram com valores aproximados tanto para o uso de apenas um VANT quanto para o uso de três VANT. Entretanto, é possível verificar nas tabelas 6 e 8 que o tempo de execução das simulações são reduzidos quando utilizados múltiplos robôs.

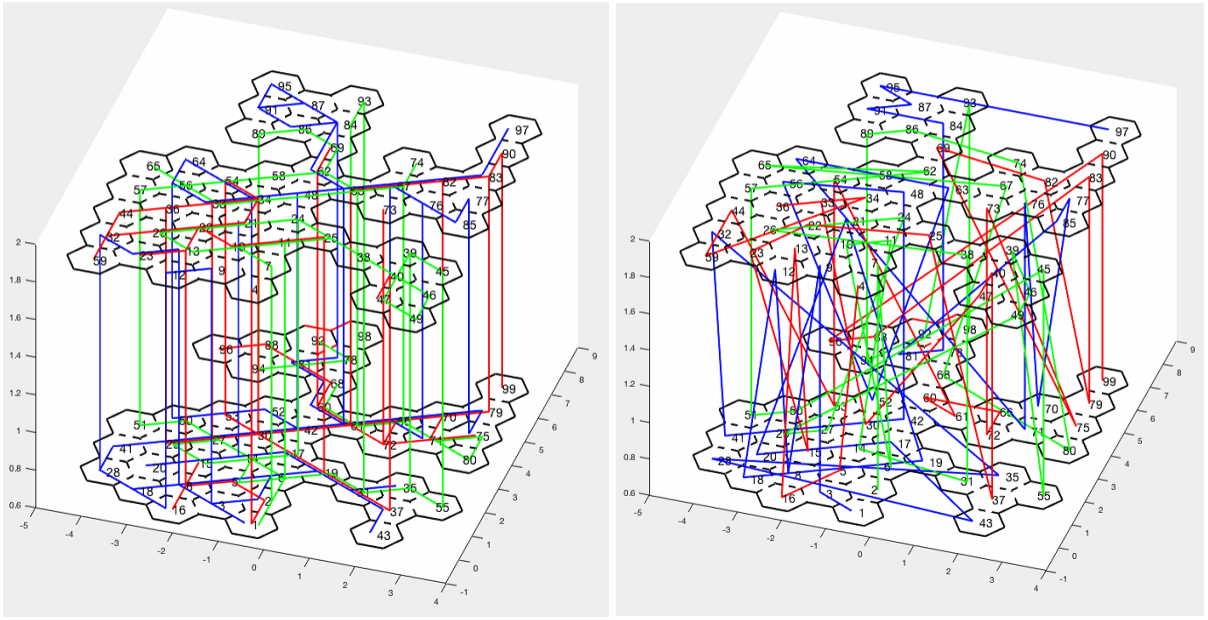
Figura 48: Deslocamentos e exploração no cenário 1 - FIFO. É possível notar que há uma maior sobreposição de trajetórias realizadas em um ambiente, além dos VANT não se concentrarem em regiões específicas do espaço.



(**a**) Deslocamentos dos VANT. (**b**) Exploração dos VANT.

Fonte: Autoria própria.

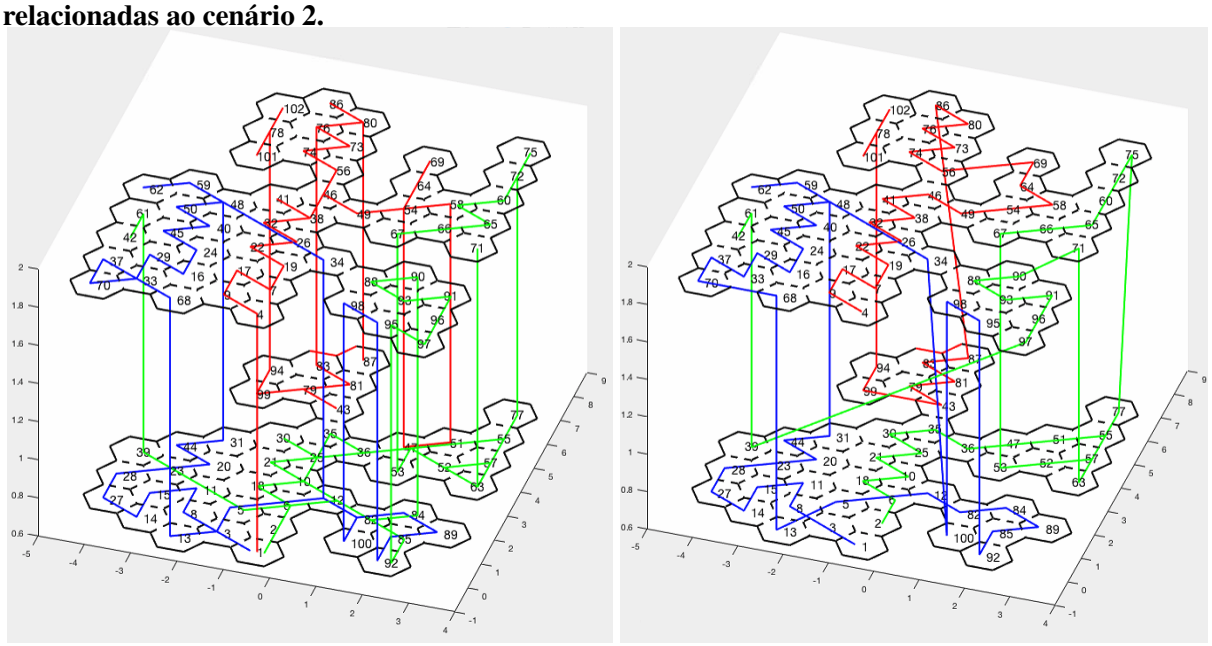
Figura 49: Deslocamentos e exploração no cenário 1 - DE. Da mesma forma que ocorre com a abordagem FIFO, a abordagem DE apresenta considerável sobreposição de trajetórias realizadas pelos VANT, porém em quantidade levemente reduzida.



(**a**) Deslocamentos dos VANT. (**b**) Exploração dos VANT.

Fonte: Autoria própria.

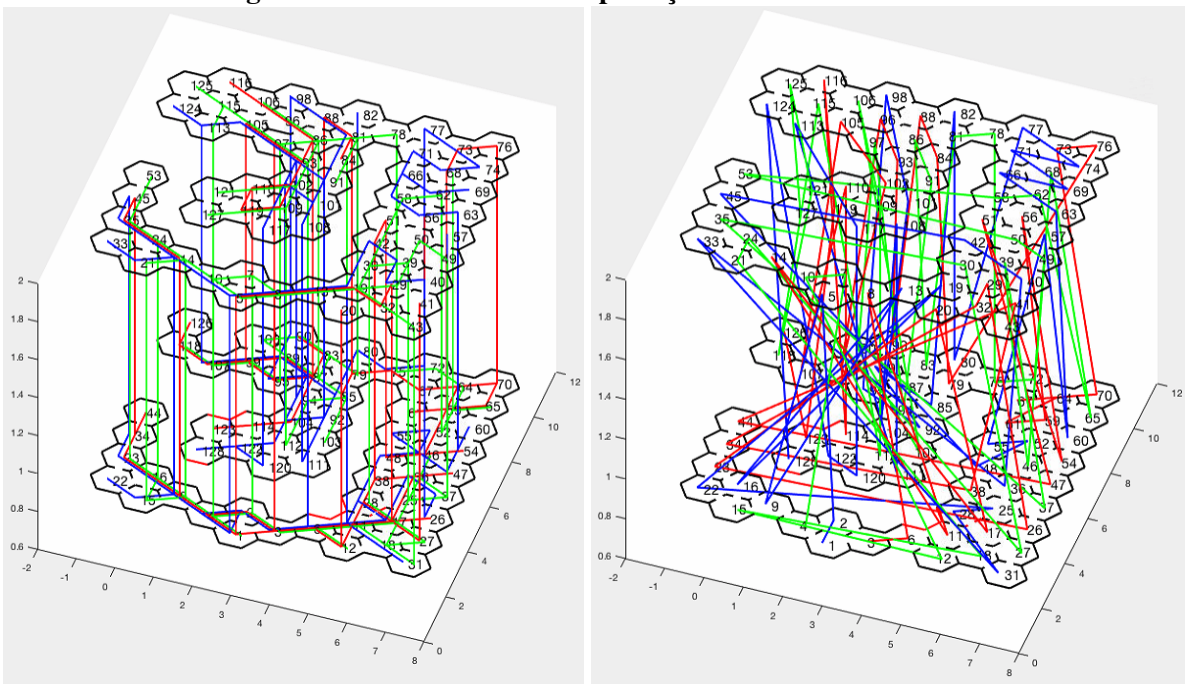
Figura 50: Deslocamentos e exploração no cenário 1 - DER. É possível verificar visualmente que as sobreposições de trajetórias são consideravelmente reduzidas, onde os VANT se concentram em determinados espaços do ambiente. Esse mesmo comportamento pode ser visto nas figuras relacionadas ao cenário 2.



(**a**) Deslocamentos dos VANT. (**b**) Exploração dos VANT.

Fonte: Autoria própria.

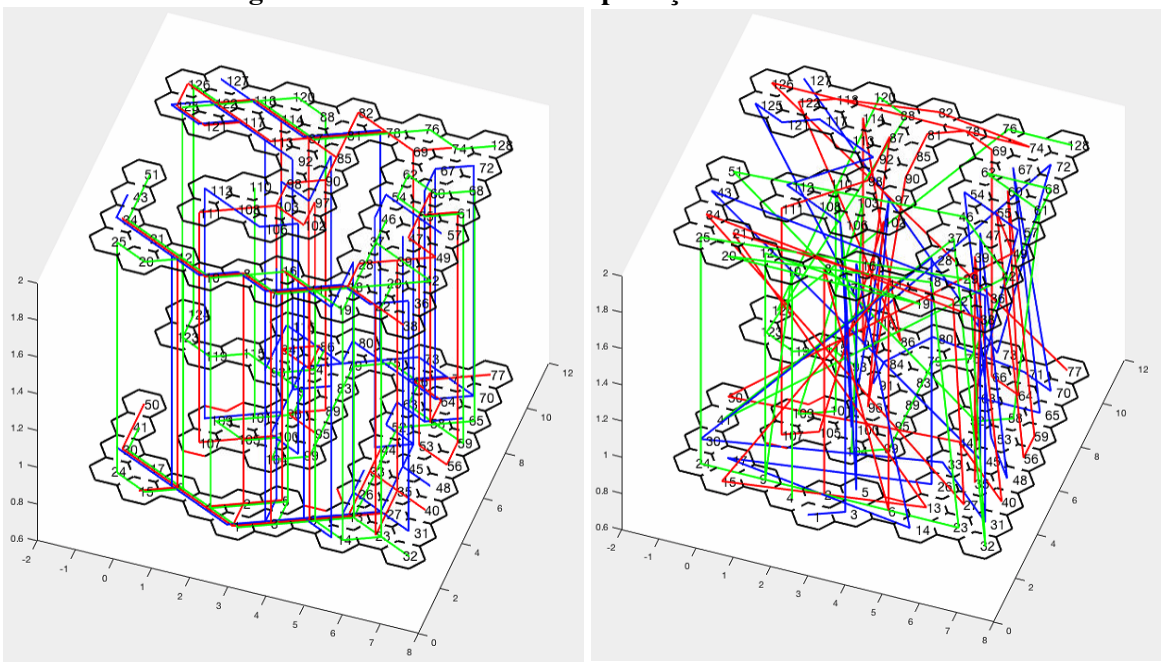
Figura 51: Deslocamentos e exploração no cenário 2 - FIFO.



(**a**) Deslocamentos dos VANT. (**b**) Exploração dos VANT.

Fonte: Autoria própria.

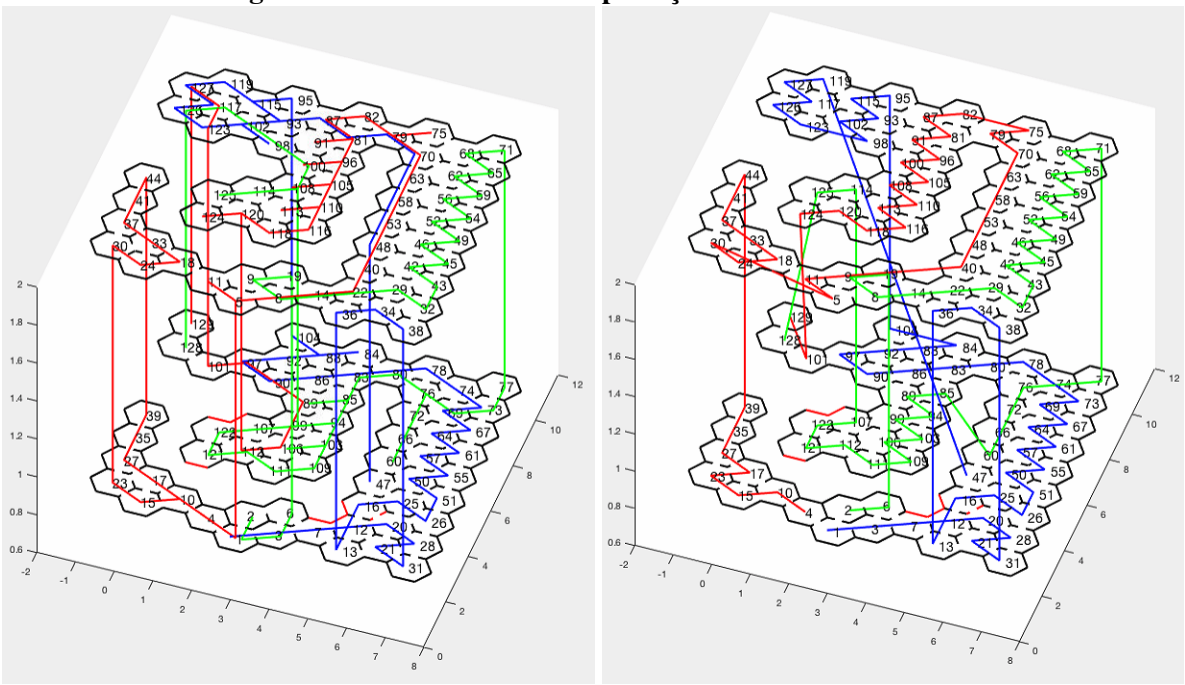
Figura 52: Deslocamentos e exploração no cenário 2 - DE.



(**a**) Deslocamentos dos VANT. (**b**) Exploração dos VANT.

Fonte: Autoria própria.

Figura 53: Deslocamentos e exploração no cenário 2 - DER.



(**a**) Deslocamentos dos VANT. (**b**) Exploração dos VANT.

Fonte: Autoria própria.

A partir das figuras acima apresentadas e da tabela 11, é possível verificar que no algoritmo DER as movimentações e explorações de cada VANT ficam concentradas em determinadas áreas do ambiente, enquanto nos algoritmos FIFO e DE isso não ocorre. O algoritmo DE irá trazer o hexágono mais perto do hexágono inicial explorado, independente de onde esteja o VANT. Com isso, a alocação pode ser de um hexágono que está em um lugar muito distante da localização atual do VANT, acarretando maior número de deslocamentos. No algoritmo FIFO, a alocação é feita baseada na ordem de descoberta e inserção na "lista de hexágonos não visitados", o que faz com que o VANT não explore um local perto de sua localização, ocasionando um aumento no número de deslocamentos efetuados. Já no algoritmo DER, é considerada a posição do VANT juntamente com o hexágono com menor distância do hexágono inicial, de modo que sejam alocados hexágonos para exploração mais próximos a sua localidade, reduzindo o número de deslocamentos e, consequentemente, tempo de exploração geral.

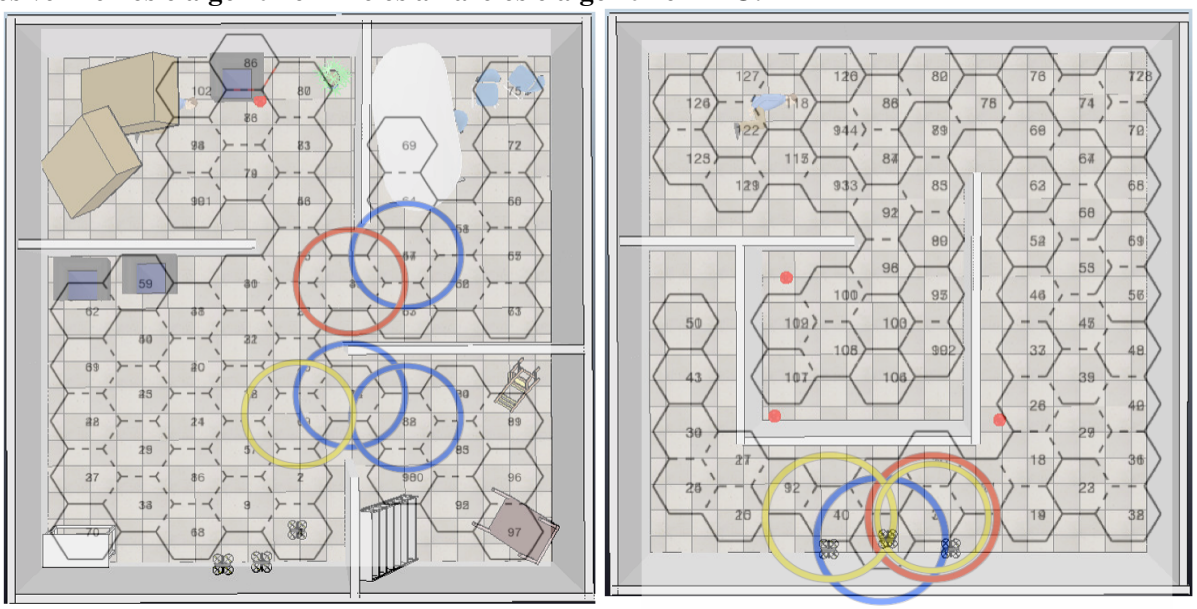
Durante o processo de exploração, a passagem de cada VANT por cada hexágono é registrada, de modo que é possível identificar qual hexágono foi o mais trafegado durante a simulação. Nesse sentido, a tabela 12 apresenta o número do hexágono com maior tráfego e a quantidade desse tráfego, enquanto a figura 54 apresenta graficamente onde se localizam esses gargalos no ambiente mapeado.

Tabela 12: Maiores tráfegos em hexágonos por algoritmo.

Estratégia	Cenário 1	Cenário 2
FIFO	1 hexágono com 16 passagens	2 hexágonos com 24 passagens cada
DE	1 hexágono com 10 passagens	1 hexágono com 23 passagens
DER	3 hexágonos com 4 passagens cada	1 hexágono com 5 passagens

Fonte: Autoria própria.

Figura 54: Gargalos no processo de exploração: os círculos azuis representam o algoritmo DER, os vermelhos o algoritmo DE e os amarelos o algoritmo FIFO.



(**a**) Gargalos nas simulações do cenário 1.

(**b**) Gargalos nas simulações do cenário 2.

Fonte: Autoria própria.

Tanto na figura 54a quanto na figura 54b é possível verificar que os hexágonos com maior número de passagens se concentram em divisas de ambientes, como portas, ou em passagens estreitas, como corredores. Além disso, quando comparamos os dados presentes na tabela 12, no cenário 1 o algoritmo DER mesmo com maior número de hexágonos com valores máximos (3 hexágonos com 4 passagens cada) possui um menor tráfego que o único hexágono com maior tráfego no algoritmo FIFO (1 hexágono com 16 passagens), e por ser uma área com grande circulação de VANT, está sujeita a encontro entre eles e, consequentemente, bloqueios de caminhos.

A média de deslocamentos por algoritmo em cada cenário é apresentada na tabela 13. A diferença no número médio de deslocamentos por hexágono no cenário 1 utilizando o algoritmo DE é $33,03\%$ menor do que a abordagem FIFO, enquanto a abordagem DER é $67,84\%$ menor do que a abordagem FIFO. Já para o cenário 2, o algoritmo DE gera $8,8\%$

menos deslocamentos no ambiente do que o algoritmo FIFO e a abordagem DER tem uma diferença de 64,76% da abordagem FIFO.

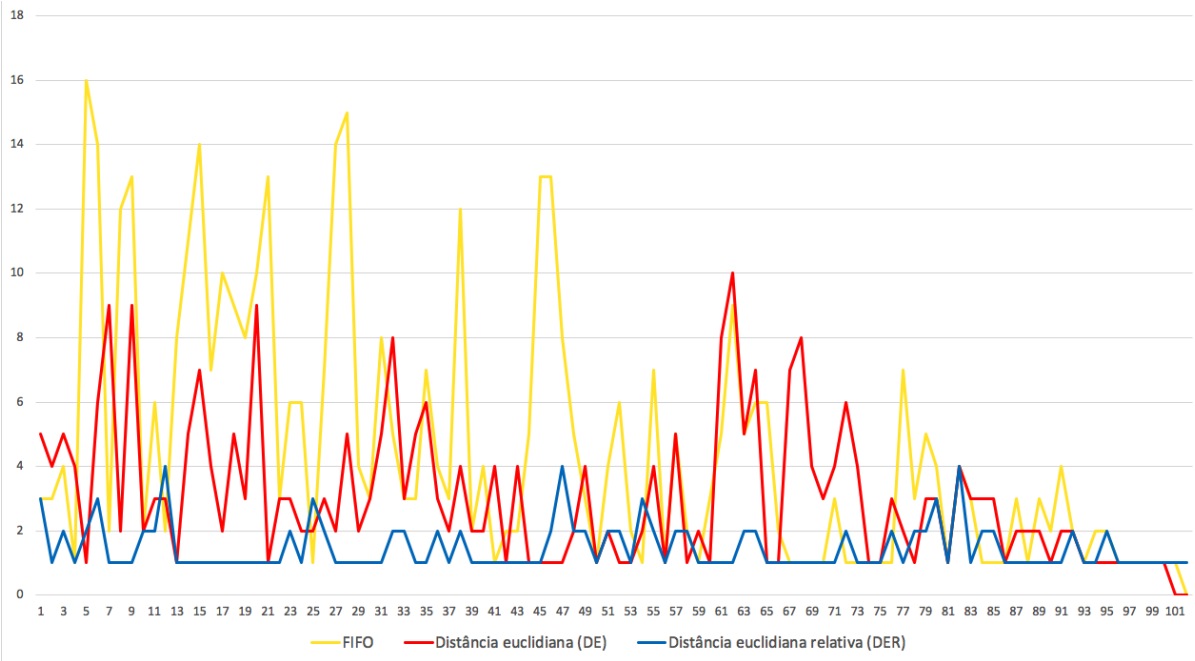
Tabela 13: Média de deslocamentos nos hexágonos.

Estratégia	Cenário 1	Diferença - FIFO	Cenário 2	Diferença - FIFO
FIFO	4,54	-	4,20	-
DE	3,04	-33,03%	3,83	-8,8%
DER	1,46	-67,84%	1,48	-64,76%

Fonte: Autoria própria.

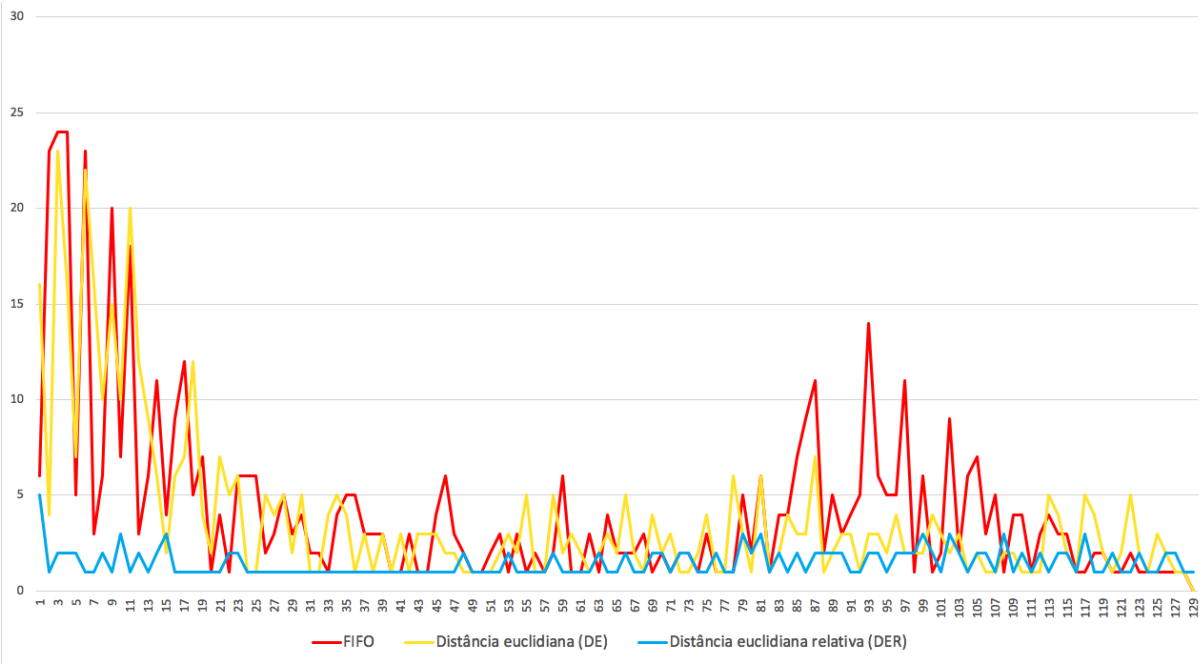
Além disso, quando é avaliado o número de passagens em todos os hexágonos, o tráfego fica distribuído por quase todos os hexágonos com valores próximos, como pode ser visto nos gráficos das figuras 55 e 56. Utilizando a abordagem DER, o tráfego no ambiente fica distribuído por todo o mapa, onde os picos de tráfego são relativamente baixos comparados com as abordagens DE e FIFO.

Figura 55: Tráfego no cenário 1.



Fonte: Autoria própria.

Figura 56: Tráfego no cenário 2.



Fonte: Autoria própria.

Durante o processo de exploração, cada VANT busca um hexágono para explorar na "lista de hexágonos não visitados", planeja uma trajetória a percorrer e então se desloca até o seu destino. Durante o deslocamento, bloqueios de caminhos podem ocorrer devido a coincidência de trajetórias calculadas se sobreporem. A tabela 14 apresenta o número de bloqueios de caminhos detectados e tratados. Como em um determinado hexágono podem haver diversos outros hexágonos adjacentes com a mesma distância, dependendo da estratégia adotada (FIFO, DE ou DER) o hexágono pode ser diferente. Assim, dependendo da estratégia escolhida o número de bloqueios podem não ser o mesmo. Além disso, o algoritmo Dijkstra responsável por calcular a trajetória a ser realizada será influenciado pelo conhecimento já existente do ambiente, que é influenciado também pela evolução da construção do mapa.

Tabela 14: Bloqueios de caminhos tratados.

Estratégia	Cenário 1	Diferença - FIFO	Cenário 2	Diferença - FIFO
FIFO	20	-	23	-
DE	16	-20%	53	130%
DER	1	-95%	0	-100%

Fonte: Autoria própria.

Além do número de deslocamentos em cada hexágono, tempo de simulação e número de bloqueios tratados, a escolha dos diferentes algoritmos influencia na evolução do mapeamento. As figuras 64, 65, 66, 67, 68 e 69 do apêndice B apresentam a evolução

do mapeamento em cada cenário simulado com as diferentes estratégias da seção 4.3.3, apresentando em seis proporções (figuras (a), (b), (c), (d), (e) e (f)) na ordem de mapeamento. Conforme o ambiente é descoberto pelos VANT, o mapeamento topológico utilizando o algoritmo DE evolui com os hexágonos mais próximos ao primeiro hexágono descoberto, sendo identificados. No mapeamento utilizando o algoritmo DER, as regiões são descobertas a partir da proximidade com o VANT daquela área. Já o mapeamento utilizando o algoritmo FIFO irá evoluir conforme os hexágonos são descobertos. Essa evolução acontece tanto com os hexágonos que possuem adjacência nas faces laterais quanto com os hexágonos adjacentes nas partes superior ou inferior de forma igual, pois dentro da estrutura topológica não há distinção de camadas, e o que influencia a escolha por explorar um hexágono que está na mesma camada (eixo z) ou um que se encontre em outra será a ordem de descoberta (para o FIFO) ou a distância euclidiana adotada (para DE ou DER). Adicionalmente, é importante resolver os bloqueios de caminhos, que, caso não tratados, os VANT ficarão parados no ambiente, sempre um esperando o outro dar passagem, paralisando todo o processo de exploração e mapeamento. Além disso, adotar uma estratégia que reduz o número de bloqueios, como o DER, significa que haverão menos trajetórias se sobrepondo, e se menos trajetórias se sobrepõem, os VANT provavelmente estarão concentrados em áreas específicas, reduzindo o número de deslocamentos, que reflete em menor tempo de exploração e mapeamento total do ambiente.

5.3.2 VISUALIZAÇÃO DOS DADOS OBTIDOS SOBRE O AMBIENTE

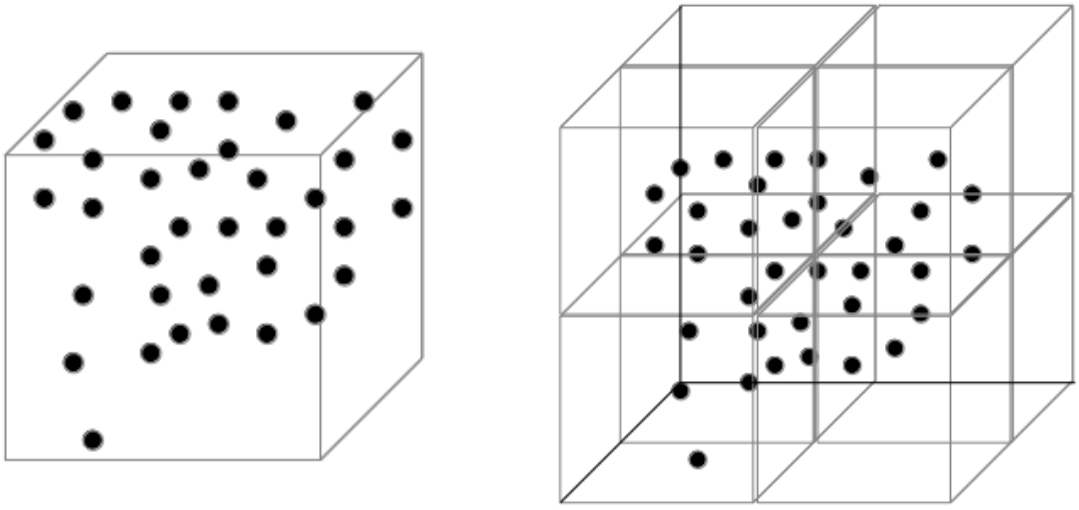
Conforme apresentado na seção 4.3.6 com as leituras dos pontos RGB-D de cada face do hexágono e o processamento de cada leitura, uma visualização em três dimensões utilizando cubos é realizada.

A seção 4.3.6 descreve como cada cubo é gerado a partir de uma representação discretizada de pontos mapeados pelas câmeras RGB-D, de forma que seu tamanho pode ser configurado para: (i) uma dimensão maior, que resultará em mais pontos RGB-D, sendo representados dentro de um mesmo cubo, diminuindo o detalhamento do ambiente; ou (ii) menor, que resultará em mais cubos, sendo gerados com maior detalhamento do espaço, porém com mais dados a serem processados na visualização. A figura 57 apresenta a variação na dimensão do cubo. A figura 57a apresenta diversos pontos RGB-D sendo discretizados em um único cubo, enquanto a figura 57b apresenta os mesmos pontos sendo projetados em oito cubos quando a dimensão é reduzida pela metade.

A figura 58 apresenta essa representação para o cenário 1 mapeado com o algoritmo DER. Além da visualização completa (todos os hexágonos mapeados), é possível criar uma

visão parcial selecionando quais hexágonos deseja-se mostrar. A figura 58d apresentam o recortes feito nos cenário. Como durante o processo de exploração e mapeamento cada VANT envia instantaneamente os dados coletados para a unidade central (UC), o processamento das leituras das câmeras RGB-D pode ser iniciado, mesmo que o processo de exploração não tenha finalizado. A visão parcial do ambiente possibilita que as equipes de busca e resgate consigam reconhecer partes do ambiente e então estabelecer suas estratégias de ação, juntamente com o mapa topológico baseado em colmeias de abelhas obtido parcialmente.

Figura 57: Discretização das leituras dos sensores RGB-D em cubos.



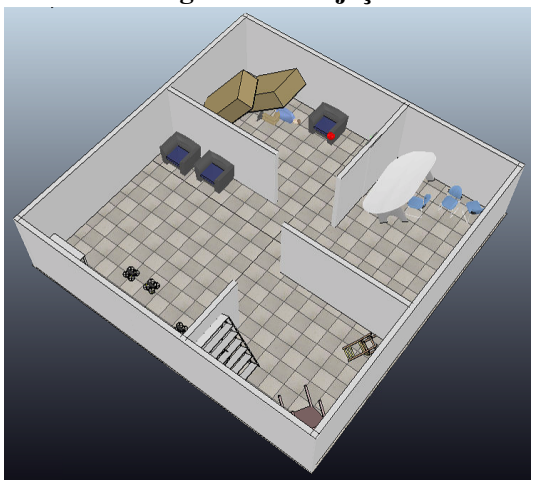
(**a**) Pontos RGB-D discretizados em um cubo com uma dimensão de tamanho D.

(**b**) Pontos RGB-D discretizados em cubos com a dimensão reduzida para $D/2$.

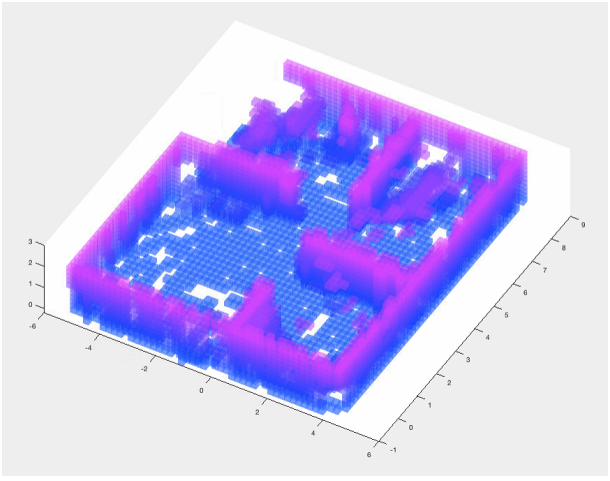
Fonte: Autoria própria.

No processo de discretização são contabilizados os pontos capturados pela câmera RBG-D que pertencem (ou estão contidos em) a um determinado cubo, gerando um índice de ocupação do cubo que reflete na sua opacidade quando ele é visualizado; Quanto mais pontos pertencem a um cubo, mais opaco é sua visualização, indicando que existem obstáculos que possivelmente são intransponíveis pelo VANT ou seres humanos. A figura 58 apresenta a visualização dos cubos em 3D a partir das leituras realizadas no cenário 1.

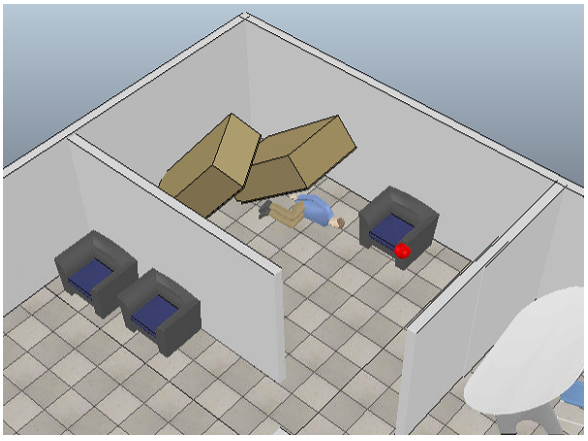
Figura 58: Projeção em cubos 3D do ambiente mapeado - cenário 1.



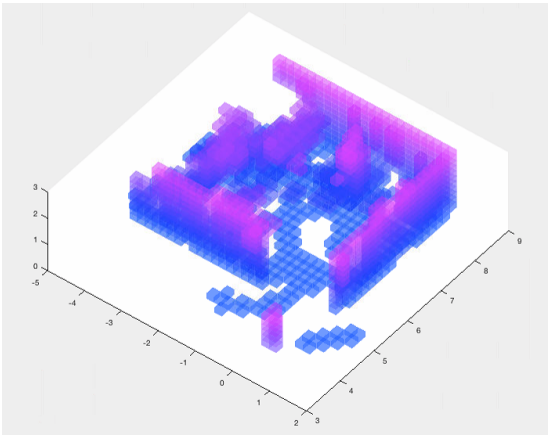
(**a**) Cenário explorado.



(**b**) Visualização da representação em cubos 3D



(**c**) Recorte do cenário explorado.

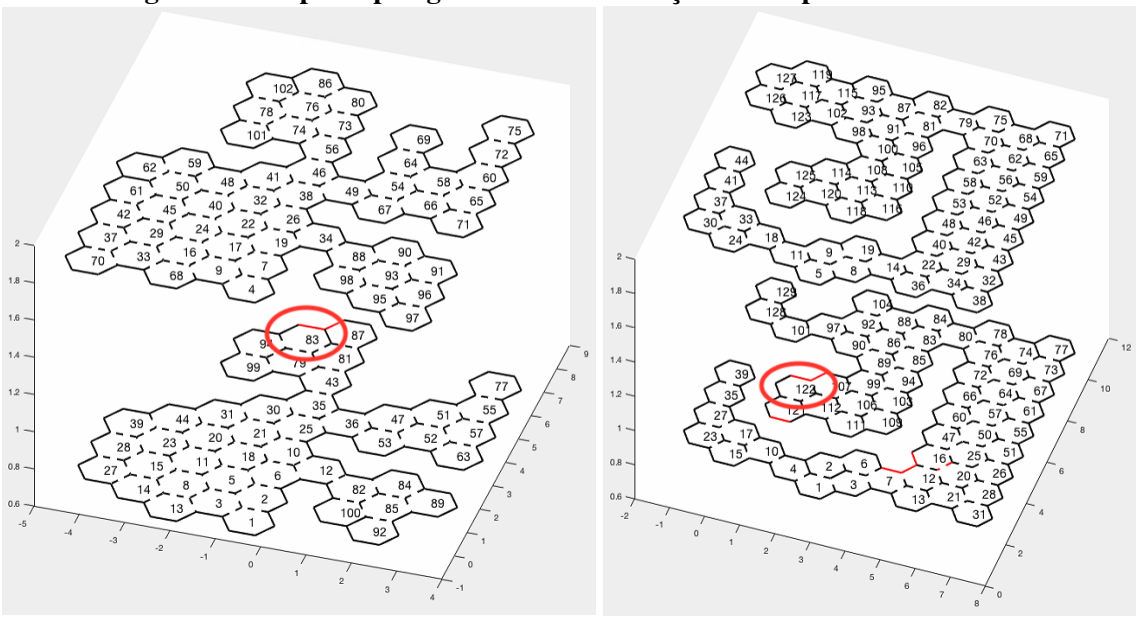


(**d**) Visualização dos cubos 3D gerados a partir de um subconjunto de hexágonos baseados no mapeamento da figura 46b da página 84: hexágonos 76, 80, 81, 83, 86, 87, 94 e 102.

Fonte: Autoria própria.

No processo de mapeamento, durante a leitura das faces, é feita a captura de informação térmica, ou seja, temperatura do ambiente. Caso seja detectada uma temperatura superior a um valor de referência adotado, um registro dessa ocorrência é feito. Dessa forma, a representação do mapa em favos de colmeia apresenta uma cor diferenciada nas faces do hexágono com esta temperatura elevada, ou seja, a face do hexágono é identificada pela cor vermelha, conforme pode ser visto nas figuras 59a e 59b. Além dessa identificação no mapa, uma imagem é registrada para que as equipes de resgate tenham uma melhor identificação do que pode ser a fonte de calor encontrada. A figura 60 apresenta imagens capturadas no processo de exploração e mapeamento utilizando o algoritmo DER nos dois cenários simulados.

Figura 59: Mapas topológicos com identificação de temperatura elevada.

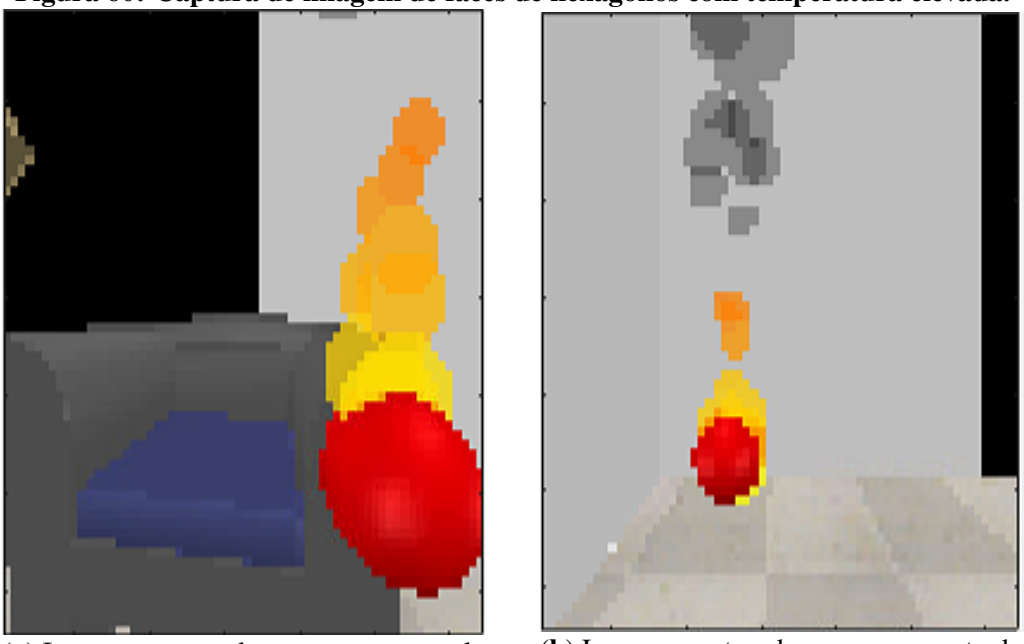


(**a**) Mapa topológico cenário 1 - identificação de foco de incêndio.

(**b**) Mapa topológico cenário 2 - identificação de foco de incêndio.

Fonte: Autoria própria.

Figura 60: Captura de imagem de faces de hexágonos com temperatura elevada.



(**a**) Imagem capturada no mapeamento do hexágono 83 do mapa da figura 59a.

(**b**) Imagem capturada no mapeamento do hexágono 122 do mapa da figura 59b.

Fonte: Autoria própria.

5.4 DISCUSSÃO

O uso de ferramentas de simulação de ambientes possibilita que sejam experimentados os diferentes métodos que foram propostos para a exploração de ambientes internos com múltiplos VANT. O ambiente de simulação V-REP possibilita a simulação de um ambiente dinâmico onde diversos VANT são inseridos para realizarem tarefas de forma conjunta. A partir de APIs específicas, é possível integrar V-REP com alguma linguagem de programação. Neste trabalho foi adotada a integração com a ferramenta MATLAB, através de chamadas a funções específicas para conexão, envio de comandos para os robôs e recebimento de informações dos sensores.

Os ambientes simulados possibilitaram delimitar um espaço de atuação dos VANT para avaliar como os diferentes algoritmos de alocação de tarefas de exploração podem influenciar no desempenho da atividade de mapeamento como um todo. Os resultados demostram que quanto menos deslocamentos possíveis, menor será o tempo geral de exploração e mapeamento, como pode ser visto na tabela 8, em que a estratégia DER obteve diferenças de tempo em relação a estratégia FIFO de 27,16% e 26,40%, para o cenário 1 e 2 respectivamente. O algoritmo DER apresentou tempos de execução consideravelmente menores que os algoritmos DE e FIFO. Isso ocorre devido ao VANT explorar o hexágono mais próximo dele que não tenha sido explorado por algum outro robô. Assim, o deslocamento é menor e o VANT concentra suas atividades em regiões específicas do ambiente. Com cada VANT concentrado em áreas específicas, menor é a possibilidade de trajetórias de deslocamento entre os pares de VANT se cruzarem, evitando bloqueios e consequentes resoluções desses, como é possível ver na tabela 11 e 13. Na estratégia DER o número de deslocamentos realizados pelos VANT foi 67,88% e 65,52% menor do que na estratégia FIFO, para os cenários 1 e 2 respectivamente. Com menor número de deslocamento dos robôs no ambiente, o tráfego geral é reduzido e gargalos de movimentação no ambiente são amenizados. Por exemplo, para as simulações no cenário 1, o algoritmo DER resultou em três hexágonos com número máximo de passagens de VANT, porém com número menor de passagens em cada um se comparado com os valores do algoritmo FIFO. Se a quantidade de tráfego em um determinado espaço é reduzido, menos encontros entre VANT ocorrerão, e com isso menos bloqueios de caminhos deverão ser tratados, contribuindo para a redução do tempo de exploração (conforme mostra a tabela 8).

Os mapas topológicos bioinspirados em colmeias de abelhas gerados fornecem informações, tanto para os robôs no seu controle de movimentação e realização de tarefas quanto para equipes de resgate obterem informações sobre o ambiente mapeado. A visualização dos espaços mapeados em cada hexágono a partir de uma representação de cubos 3D possibilita

o reconhecimento de paredes, portas, corredores e outros elementos presentes no espaço. Sensores térmicos contribuem fornecendo dados do ambiente e identificando locais que estão com temperaturas superiores a um valor de referência definido. As imagens capturadas nesses pontos podem auxiliar equipes de resgate a prepararem-se para lidar da melhor forma possível com os obstáculos encontrados ao adentrarem no ambiente, que até então era desconhecido.

Devido ao uso de simuladores para executar os experimentos, foi definida uma estimativa de tempo de execução das simulações baseadas em parâmetros (número de hexágonos explorados, número de deslocamentos efetuados e número de tratamento de bloqueios de caminho) para cada VANT. Como o mapeamento é feito de forma topológica, os erros de imprecisão de posicionamento não se propagam pelo mapa. Entretanto, por não ser considerados os erros de estabilização durante a leitura das câmeras RGB-D, as visualizações podem ter maior ruído na sua representação, necessitando ainda de técnicas para atenuá-los.

No método proposto são ignoradas as falhas de comunicação entre VANT e UC, porém dependendo do estado catastrófico do ambiente interno que os VANT possam se encontrar, podem haver interferências que influenciem na qualidade de comunicação, sendo um ponto que precisa ser considerado futuramente. Além disso, não é considerado nos experimentos a capacidade de energia disponível para os VANT continuarem suas tarefas, de forma que tanto o controle de trajetória quanto as estratégias de alocação de tarefas devem também considerar estas variáveis para obterem a maior quantidade de espaço mapeado possível.

O método apresentado é flexível quanto a definição de parâmetros para a exploração e o mapeamento do ambiente interno. As dimensões do hexágono no mapa topológico podem ser reduzidas ou ampliadas antes de iniciar a execução. Reduzir o tamanho do hexágono implicará em um maior número de pontos a serem mapeados, aumentando a precisão de representação do ambiente, porém aumentará o número de tarefas de exploração e o tempo total de mapeamento. Aumentar as dimensões do hexágono, de forma inversa, irá diminuir o tempo total de mapeamento por haver menos hexágonos a serem explorados, entretanto a precisão do mapa em refletir a estrutura real do ambiente poderá ser prejudicada, pois espaços menores que poderiam ser identificados mediante um hexágono menor e onde um VANT conseguiria transitar para ter acesso a outros locais inexplorados, não seriam identificados. Então, caso as equipes de busca e resgate queiram alterar os parâmetros do método, bastaria saber os limites de alcance dos sensores a serem utilizados. Dessa forma o método pode ser considerado genérico no sentido de aceitar diversas configurações de sensores de distância e possibilitar a configuração dos seus parâmetros de funcionamento.

O uso de simuladores no desenvolvimento deste trabalho possibilita que diversos

experimentos possam ser realizados e as falhas sejam identificadas sem um possível custo de manutenção e uso de equipamentos reais. Entretanto, as dinâmicas de representação virtual de um mundo real geram uma carga de processamento elevada, o que influencia precisão e duração de simulação. Para que essas limitações não interferissem no comportamento dos algoritmos apresentados, foram implementados formas de detecção de falhas de comunicação entre MATLAB e V-REP, o que influenciou também nos tempos de execução da simulação de exploração e mapeamento realizados nos experimentos. Assim, espera-se que a aplicação em robôs reais (não-simulados) possam apresentar tempos ainda melhores.

6 CONCLUSÕES

6.1 CONSIDERAÇÕES FINAIS

O desenvolvimento de tecnologias para robótica móvel está possibilitando a sua aplicação nos mais diferenciados setores. Seja na indústria ou em serviços, os robôs autônomos ganham espaço por propiciar atividades mais específicas, permitindo abstrair ações para os usuários finais. Além disso, em setores onde o acesso de um humano pode ser perigoso, estas soluções autônomas fornecem uma opção de segurança, como, por exemplo, para equipes de resgate, quando o ambiente desconhecido pode trazer riscos à vida.

Para o uso de robôs autônomos em ambientes desconhecidos é necessário que o robô tenha meios para reconhecer o ambiente em sua volta e sua própria localização dentro dele. Diversas pesquisas trazem soluções com técnicas diferentes, onde cada uma pode ser mais adequada dependendo do tipo da aplicação. No mapeamento de espaços internos, considerando ambientes onde pode ter acontecido uma possível catástrofe ou desastre natural, realizar o mapeamento do ambiente em um período de tempo aceitável é fundamental para equipes de busca e resgate, já que precisam de um entendimento de todo espaço para planejar e decidir suas ações.

Entretanto, o uso de apenas um único robô pode trazer algumas restrições, como tempo de mapeamento muito alto além de poder haver restrição no acesso terrestre. O problema de acesso a locais que são, inicialmente, inacessíveis para um robô terrestre pode ser solucionado com o uso de um VANT, que pode sobrevoar espaços remotos e sem possibilidade de acesso de outra forma. Porém, o uso de um único VANT traz outras restrições, como tempo disponível de voo. O uso de times de robôs aéreos (isto é, múltiplos VANT) trabalhando de forma cooperativa apresenta-se como uma alternativa, quando a partir das leituras realizadas por eles, o trabalho de mapeamento é feito em menor tempo (BURGARD et al., 2005). O uso de múltiplos VANT em uma tarefa de mapeamento de ambientes internos possibilita atingir uma maior amplitude do espaço, pois os VANT acessam locais que veículos terrestres não conseguem.

A diversidade de técnicas de exploração e mapeamento possibilitam soluções de

formas variadas. Enquanto mapas métricos permitem representar um ambiente completo de forma discretizada, mapas topológicos trazem a baixa carga de processamento e consumo de memória (SIEGWART et al., 2011), o que é salutar quando o robô em questão é um VANT. Técnicas de exploração diversas buscam atender demandas com foco na maior cobertura possível de um espaço ou ainda na comunicação e tolerância a falhas durante o processo.

O método aqui proposto define uma estrutura de mapa topológico baseado na organização dos favos de uma colmeia de abelhas. Cada vértice desse mapa topológico representa um ponto específico no ambiente que abrange o espaço de um hexágono, e a adjacência entre os pontos representa a possibilidade de circulação livre entre eles. Com esta estrutura definida, os múltiplos VANT fazem a exploração e o mapeamento do ambiente interno de forma cooperativa, buscando repetir o comportamento de abelhas no processo de construção dos favos: um primeiro VANT mapeia o primeiro hexágono, verificando se em cada um dos seus lados é possível alçar um novo espaço livre. Após o primeiro hexágono ser completamente mapeado, os demais VANT seguem no processo de exploração dos novos hexágonos descobertos. O processo de exploração finaliza quando não houver novas descobertas. Uma técnica de resolução de caminhos bloqueados tratou momentos em que trajetórias de diferentes VANT ficam sobrepostas e coincidem a busca por ocupação de um mesmo espaço em um mesmo tempo.

Para organizar a ordem de exploração dos hexágonos, três diferentes estratégias foram aplicadas, conforme pode ser visto na seção 4.3.3. A estratégia FIFO organizou a exploração baseada na ordem de descoberta dos hexágonos, porém gerou muito tráfego no ambiente, acarretando em maiores tempo de execução do processo de exploração e mapeamento. A estratégia de distância euclidiana (DE) fez com que o conhecimento do ambiente (crescimento do grafo topológico) ocorresse em largura, ou seja, os espaços mais próximos do ponto inicial são explorados antes. Porém, esta estratégia também apresenta um tráfego elevando no ambiente, pois a alocação dos hexágonos não considera o posicionamento geral de todos os VANT. A abordagem de distância euclidiana relativa (DER) considera, além da distância do hexágono não visitado para o hexágono inicial, a posição em que o VANT está localizado. Isso permite que as tarefas de exploração e mapeamento do VANT concentrem-se em determinadas áreas do ambiente, diminuindo o tráfego, o número de caminhos bloqueados e consequentemente o tempo total de exploração e mapeamento de todo o ambiente. Nos experimentos realizados utilizando essa estratégia, houve redução do tempo de mapeamento total em 27,16% e 26,4% em comparação ao FIFO para os cenários 1 e 2, respectivamente. Quando é analisado o número de deslocamentos realizados, a diferença é de 67,88% e 65,52% a menos dos experimentos que utilizaram a estratégia FIFO, para os cenários 1 e 2,

respectivamente.

A captura de informações de temperatura e sua respectiva representação no mapa topológico gerado possibilita que as equipes de busca e resgate consigam identificar focos de incêndios no ambiente. Esta aplicação pode ser estendida para outros tipos de sensores, como de umidade, gás, entre outros, ampliando a possibilidade de aplicação do método e dos mapas propostos no planejamento e decisão das ações a serem realizadas em uma missão de resgate.

A representação em cubos 3D do espaço a partir de leituras das câmeras RGB-D geram uma visualização do ambiente de forma que as equipes de resgate possam ter uma percepção do estado físico interno. A definição do tamanho destes cubos influencia em um maior ou menor detalhamento do espaço.

A utilização do ambiente virtual de simulação V-REP possibilitou realizar os experimentos variando o cenário e os algoritmos adotados como estratégia de alocação de hexágonos para explorar (FIFO, DE e DER). Entretanto, a representação de dinâmicas de vários VANT e do ambiente como um todo demandam alta capacidade de processamento. O equipamento utilizado para executar as simulações possui capacidade de processamento satisfatório, porém mesmo assim apresenta falhas de comunicação entre V-REP e Matlab, o que por muitas vezes refletia em leitura errada dos sensores. Assim, formas de detecção desta falhas foram inseridas nos códigos, buscando a reconexão quando elas aconteciam.

Considerando as hipóteses desta tese que foram descritos na seção 1.1, verifica-se que é possível realizar a exploração e o mapeamento colaborativo de ambientes internos usando múltiplos robôs aéreos. Para tanto, é necessário o uso de um método que tenha em seu arcabouço o controle das atividades exploratórias com alocação de tarefas de exploração e mapeamento, coordenadas entre os múltiplos robôs aéreos, e também mantendo um controle da evolução do mapeamento como um todo e das informações adicionais que dão suporte ao controle do comportamento dos VANT autônomos. Esse controle é obtido a partir de algumas premissas, por exemplo: (i) o VANT circula apenas dentro de área conhecidas (hexágonos identificados), (ii) escolha de um hexágono a explorar a partir de uma estratégia que busque fornecer o menor tempo global de exploração e mapeamento com o menor deslocamento possível para cada VANT, como pode ser visto nas seção 5.3. Ainda, o conhecimento da localização de cada VANT e de suas trajetórias possibilita tratar possíveis bloqueios de caminhos que por ventura se cruzem, de forma que esse processamento é feito no momento em que ocorrem os bloqueios. A estratégia a ser adotada (FIFO, DE ou DER) influencia no número de bloqueios de caminhos a serem tratados.

A realização dos experimentos em simuladores necessitou da definição de estimativas

baseadas em parâmetros da exploração para o tempo de execução da exploração e mapeamento, que poderiam acontecer em uma missão real usando VANT reais. Os erros de imprecisão de posicionamento não se propagam pelo mapa topológico (SIEGWART et al., 2011), porém, ao ignorar falhas de precisão no posicionamento durante a captura das informações pelas câmeras RGB-D, ruídos podem ocorrer na sua representação. As falhas de comunicação entre os VANT e a UC são abstraídas neste método, porém em um cenário real podem haver interferências que influenciem na qualidade de comunicação. Alguns aspectos gerais de funcionamento de VANT também são ignorados, de forma que questões de tempo de voo disponível *vs* mapeamento completo do ambiente não são avaliados neste trabalho. O método apresentado não considera objetos dinâmicos não identificados (como outros VANT), de forma que caso um obstáculo surja em um local anteriormente mapeado como livre, tal obstrução não será identificada com atualizações de adjacências.

Esta tese de doutorado gerou as seguintes contribuições:

- Um método de exploração e mapeamento de ambientes internos inspirados em como as abelhas constroem as colmeias com sua implementação em MATLAB;
- Representações do ambiente explorado através de mapas topológicos e suas respectivas visualizações em cubos que representam a ocupação 3D do ambiente;
- Modelos de ambientes de simulação com múltiplos VANT equipados com sensores de proximidade e câmera RGB-D na plataforma V-REP;
- Um artigo sobre estratégias de alocação de tarefas de exploração e mapeamento de ambientes internos com múltiplos robôs aéreos publicado em revista científica Sensor (Basel) ((ROSA et al., 2020) - ver apêndice D - fator de impacto 3.031;
- Um artigo de revisão sistemática da literatura submetido para revista científica IEEE Latin America, o qual está em processo de revisão - fator de impacto 0.804;
- Um artigo submetido para evento científico (CONTROLO 2020, de 01 a 03 de julho de 2020, Bragança - Portugal), o qual encontra-se em processo de revisão;

6.2 TRABALHOS FUTUROS

O desenvolvimento deste trabalho possibilitou identificar novas lacunas que ainda precisam sem exploradas e representam novos problemas a serem abordados em futuros trabalhos de pesquisa.

Para o método aqui proposto, após a sua definição, o tamanho dos hexágonos para o mapa é fixo e homogêneo. Assim, o mapeamento pode tomar maior tempo de execução se o tamanho do hexágono for reduzido, ou, então, por outro lado, pode-se não conseguir atingir determinados espaços se o tamanho do hexágono for aumentado. Por isso, um foco de trabalho futuro é a exploração e o mapeamento usando tamanhos de hexágonos variáveis.

A alocação de hexágonos a serem explorados para um VANT não considera a informação da disposição global de todos os VANT no ambiente e, portanto, a alocação de um hexágono para um VANT pode ser feita para um local próximo a outro VANT que, por ventura, pode estar prestes a finalizar seu trabalho, e que este último seria a melhor opção para a alocação da tarefa de exploração do hexágono escolhido. Assim, a alocação dos hexágonos a serem explorados, considerando a disposição de todos os VANT no ambiente e o custo de trajeto, pode ser foco de outras pesquisas.

Além disso, o método proposto não considera objetos não identificados em espaços livres já mapeados, de forma que os únicos elementos dinâmicos existentes são os próprios VANT. Estudos com ambientes onde os obstáculos podem ser dinâmicos, focando também a atualização da informação dos obstáculos nos mapas em tempo de mapeamento, também são sugestões de trabalhos futuros.

Como neste trabalho o acesso ao mapa é feita de forma centralizada, trabalhos futuros poderão descentralizar o processo de construção dos mapas, sendo que a partir dos encontros entre robôs os dados sejam atualizados e os mapas evoluam.

Por fim, outro trabalho futuro pode ser analisar as características físicas e o estado atual do VANT, como, por exemplo, o nível de carga da bateria e quanta energia é necessária para alcançar o próximo hexágono, no processo de escolha e atribuição das tarefas de exploração e mapeamento.

REFERÊNCIAS

ADAMS, M. et al. Slam gets a phd: New concepts in map estimation. **IEEE Robotics & Automation Magazine**, IEEE, v. 21, n. 2, p. 26–37, 2014.

ALONSO-MORA, J. et al. Reactive mission and motion planning with deadlock resolution avoiding dynamic obstacles. **Autonomous Robots**, v. 42, n. 4, p. 801–824, 2018. Disponível em: <https://doi.org/10.1007/s10514-017-9665-6>.

ARKIN, R. C. **Behavior-based robotics**. 1. ed. Cambridge, Massachusetts: MIT press, 1998. (Intelligent Robotics and Autonomous Agents series, v. 1). ISBN 0-262-01165-4.

ARYO, D. **Dijkstra Algorithm**. 2020. MATLAB Central File Exchange. Disponível em: <https://www.mathworks.com/matlabcentral/fileexchange/36140-dijkstra-algorithm>.

AULINAS, J. et al. The slam problem: a survey. **11th International Conference of the Catalan Association for Artificial Intelligence, CCIA'08**, 2008.

AUSTIN, R. **Unmanned aircraft systems: UAVS desing, development and deployment**. West Sussex, UK: Wiley and Sons Ltd., 2010.

BAGINSKI, B. Motion planning for manipulators with many degrees of freedom-the bb-method. Citeseer, 1998.

BEESON, P.; MODAYIL, J.; KUIPERS, B. Factoring the mapping problem: Mobile robot map-building in the hybrid spatial semantic hierarchy. **The International Journal of Robotics Research**, SAGE Publications Sage UK: London, England, v. 29, n. 4, p. 428–459, 2010.

BEKRIS, K. E. et al. Safe distributed motion coordination for second-order systems with different planning cycles. **The International Journal of Robotics Research**, v. 31, n. 2, p. 129–150, 2012. Disponível em: <https://doi.org/10.1177/0278364911430420>.

BURGARD, W. et al. Collaborative multi-robot exploration. In: **Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)**. San Francisco, CA: IEEE, 2000. v. 1, n. 1, p. 476–481 vol.1. ISSN 1050-4729.

BURGARD, W. et al. Coordinated multi-robot exploration. **IEEE Transactions on Robotics**, v. 21, n. 3, p. 376–386, June 2005. ISSN 1941-0468.

CESARE, K. et al. Multi-uav exploration with limited communication and battery. In: IEEE. **2015 IEEE international conference on robotics and automation (ICRA)**. Seattle, WA, 2015. p. 2230–2235.

DAI, X. et al. Predicate logic reasoning for exploration coordination of multi-robot systems in structured environments. **Concurrency and Computation: Practice and Experience**, Wiley Online Library, p. e5806, 2020.

DECEA. **ICA 100-40: Sistemas de Aeronaves Remotamente Pilotadas e o Acesso ao Espaço Aéreo Brasileiro**. 2015. Disponível em: <https://www.decea.gov.br/static/uploads/2015/12/Instrucao-do-Comando-da-Aeronautica-ICA-100-40.pdf>. Acesso em: 10 de abril de 2020.

DEWAN, A. et al. Heterogeneous ugv-mav exploration using integer programming. In: IEEE. **2013 IEEE/RSJ International Conference on Intelligent Robots and Systems**. Tokyo, Japan, 2013. p. 5742–5749.

DHIMAN, N. K.; DEODHARE, D.; KHEMANI, D. Where am i? creating spatial awareness in unmanned ground robots using slam: A survey. **Sadhana**, v. 40, n. 5, p. 1385–1433, 2015. Disponível em: <https://doi.org/10.1007/s12046-015-0402-6>.

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

DURRANT-WHYTE, H. F. Uncertain geometry in robotics. **IEEE Journal on Robotics and Automation**, IEEE, v. 4, n. 1, p. 23–31, 1988.

EDAN, Y. Design of an autonomous agricultural robot. **Applied Intelligence**, Springer, v. 5, n. 1, p. 41–50, 1995.

ELFES, A. Occupancy grids: A stochastic spatial representation for active robot perception. In: **Proceedings of the Sixth Conference on Uncertainty in AI**. Cambridge, MA: Elsevier, 1990. v. 2929.

ELIAZAR, A. **DP-SLAM**. Tese (Doutorado) — Duke University, Durham, NC, 2005.

ELIAZAR, A.; PARR, R. Dp-slam: Fast, robust simultaneous localization and mapping without predetermined landmarks. In: **IJCAI**. Acapulco, Mexico: IJCAI, 2003. v. 3, p. 1135–1142.

EMBRAPA. **Sistema de Produção de Mel para a Região Sul do Rio Grande do Sul**. 2018. Disponível em: <https://www.infoteca.cnptia.embrapa.br/infoteca/bitstream/doc/1104382/1/Sistema26web.pdf>. Acesso em: 10 de abril de 2020.

ERINC, G.; CARPIN, S. Anytime merging of appearance-based maps. **Autonomous Robots**, Springer, v. 36, n. 3, p. 241–256, 2014.

EVANS, J. et al. Handling real-world motion planning: a hospital transport robot. **IEEE Control Systems Magazine**, v. 12, n. 1, p. 15–19, 1992.

FRAUNDORFER, F.; SCARAMUZZA, D. Visual odometry: Part ii: Matching, robustness, optimization, and applications. **IEEE Robotics & Automation Magazine**, IEEE, v. 19, n. 2, p. 78–90, 2012.

FUKUSHIMA, H.; KON, K.; MATSUNO, F. Model predictive formation control using branch-and-bound compatible with collision avoidance problems. **IEEE Transactions on Robotics**, v. 29, n. 5, p. 1308–1317, Oct 2013. ISSN 1941-0468.

GAN, S. K.; FITCH, R.; SUKKARIEH, S. Real-time decentralized search with inter-agent collision avoidance. In: **2012 IEEE International Conference on Robotics and Automation**. St Paul, MN: IEEE, 2012. p. 504–510. ISSN 1050-4729.

GONZáLEZ-BAñOS, H. H.; LATOMBE, J.-C. Navigation strategies for exploring indoor environments. **The International Journal of Robotics Research**, v. 21, n. 10-11, p. 829–848, 2002. Disponível em: <https://doi.org/10.1177/027836490202101083 4>.

GOULD, J.; GOULD, C. **Animal Architects: Building and the Evolution of Intelligence**. Basic Books, 2012. ISBN 9780465028399. Disponível em: <https://books.google.com.br/books?id=B1l-0dw7o_sC>.

HOOG, J. D.; CAMERON, S.; VISSER, A. Autonomous multi-robot exploration in communication-limited environments. In: CITESEER. **Proceedings of the Conference on Towards Autonomous Robotic Systems**. Plymouth, UK, 2010. p. 68–75.

HORNUNG, A. et al. Octomap: An efficient probabilistic 3d mapping framework based on octrees. **Autonomous Robots**, Springer, v. 34, n. 3, p. 189–206, 2013.

HOWARD, A. Multi-robot mapping using manifold representations. In: IEEE. **Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on**. [S.l.], 2004. v. 4, p. 4198–4203.

HOWARD, A. Multi-robot simultaneous localization and mapping using particle filters. **The International Journal of Robotics Research**, Sage Publications Sage CA: Thousand Oaks, CA, v. 25, n. 12, p. 1243–1256, 2006.

JESSUP, J.; GIVIGI, S.; BEAULIEU, A. Merging of octree based 3d occupancy grid maps. In: IEEE. **Systems Conference (SysCon), 2014 8th Annual IEEE**. Ottawa, Canada, 2014. p. 371–377.

KLEIN, G.; MURRAY, D. Parallel tracking and mapping for small ar workspaces. In: **2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality**. Washington, DC: IEEE Computer Society, 2007. p. 225–234.

KOSLOSKY, E. **Geração de trajetória em espiral e navegação com desvio de obstáculos para veículos aéreos não-tripulados**. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná - CPGEI, 2018.

KUIPERS, B.; BYUN, Y.-T. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. **Robotics and autonomous systems**, Elsevier, v. 8, n. 1, p. 47–63, 1991.

LAVALLE, S. M. **Planning algorithms**. Cambridge, Massachusetts: Cambridge University Press, 2006.

LEE, C. S.; CLARK, D. E.; SALVI, J. Slam with dynamic targets via single-cluster phd filtering. **IEEE Journal of Selected Topics in Signal Processing**, IEEE, v. 7, n. 3, p. 543–552, 2013.

LI, X.; AOUF, N. Cooperative vslam based on uav application. In: IEEE. **2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)**. Guangzhou, China, 2012. p. 914–919.

LIDORIS, G. et al. The autonomous city explorer (ace) project—mobile robot navigation in highly populated urban environments. In: IEEE. **2009 IEEE International Conference on Robotics and Automation**. Kobe, Japan, 2009. p. 1416–1422.

LIU, R. P.; ROGERS, G.; ZHOU, S. Wsn14-3: Honeycomb architecture for energy conservation in wireless sensor networks. In: **IEEE Globecom 2006**. San Francisco, CA: IEEE, 2006. p. 1–5. ISSN 1930-529X.

LOIANNO, G.; THOMAS, J.; KUMAR, V. Cooperative localization and mapping of mavs using rgb-d sensors. In: IEEE. **2015 IEEE International Conference on Robotics and Automation (ICRA)**. Seattle, Washington, 2015. p. 4021–4028.

LUCAS, A. E. **Mapeamento e Localização Baseados em Mosaicos Visuais**. Dissertação (Mestrado) — Universidade Técnica de Lisboa, 2009.

MAHENDRAN, A. et al. Ugv-mav collaboration for augmented 2d maps. In: **Proceedings of Conference on Advances In Robotics**. New York, NY, USA: ACM, 2013. (AIR '13), p. 38:1–38:6. ISBN 978-1-4503-2347-5. Disponível em: <http://doi.acm.org/10.1145/2506095.2506116>.

MAKARENKO, A. A. et al. An experiment in integrated exploration. In: **IEEE/RSJ International Conference on Intelligent Robots and Systems**. Lausanne, Switzerland: IEEE, 2002. v. 1, p. 534–539 vol.1. ISSN null.

MANSSON, C.; STENBERG, D. **Model-based Design Development and Control of a Wind Resistant Multirotor UAV**. Dissertação (Mestrado) — Lunds University - Department of Automatic Control, 2014.

MARTINEZ, A.; FERNNDEZ, E. **Learning ROS for Robotics Programming**. [S.l.]: Packt Publishing, 2013. ISBN 1782161449, 9781782161448.

MCCUNE, R. R.; MADEY, G. R. Agent-based simulation of cooperative hunting with uavs. In: **Proceedings of the Agent-Directed Simulation Symposium**. San Diego, CA, USA: Society for Computer Simulation International, 2013. (ADSS 13), p. 8:1–8:6. ISBN 978-1-62748-029-1. Disponível em: <http://dl.acm.org/citation.cfm?id=2499592.2499600>.

MICHAEL, N. et al. Collaborative mapping of an earthquake-damaged building via ground and aerial robots. **Journal of Field Robotics**, Wiley Online Library, v. 29, n. 5, p. 832–841, 2012.

MONTEMERLO, M.; THRUN, S. Simultaneous localization and mapping with unknown data association using fastslam. In: IEEE. **Proceedings of ICRA'03: IEEE International Conference on Robotics and Automation, 2003.** Taipei, Taiwan, 2003. v. 2, p. 1985–1991.

MORATUWAGE, D. et al. Rfs collaborative multivehicle slam: Slam in dynamic high-clutter environments. **IEEE Robotics & Automation Magazine**, IEEE, v. 21, n. 2, p. 53–59, 2014.

MORGAN, D.; CHUNG, S.-J.; HADAEGH, F. Model predictive control of swarms of spacecraft using sequential convex programming. **Journal of Guidance, Control, and Dynamics**, v. 37, p. 1–16, 04 2014.

MULLANE, J. et al. A random-finite-set approach to bayesian slam. **IEEE Transactions on Robotics**, IEEE, v. 27, n. 2, p. 268–282, 2011.

NAZZI, F. The hexagonal shape of the honeycomb cells depends on the construction behavior of bees. **Scientific Reports**, v. 6, p. 28341, 06 2016.

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

OLIVEIRA, D. de; WEHRMEISTER, M. Using deep learning and low-cost rgb and thermal cameras to detect pedestrians in aerial images captured by multirotor uav. **Sensors**, MDPI AG, v. 18, n. 7, p. 2244, Jul 2018. ISSN 1424-8220. Disponível em: <http://dx.doi.org/10.3390/s18072244>.

OLIVEIRA, T. E. A. de. **Sistema autônomo para a estimaçao da pose de um objeto com faces planas em ambiente não estruturado**. Dissertação (Mestrado) — Instituto Militar de Engenharia, 2012.

OLSON, E. et al. Exploration and mapping with autonomous robot teams. **Communications of the ACM**, ACM, v. 56, n. 3, p. 62–70, 2013.

PAMOSOAJI, A. K.; HONG, K. A path-planning algorithm using vector potential functions in triangular regions. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, v. 43, n. 4, p. 832–842, July 2013. ISSN 2168-2232.

POLISCIUC, E. et al. Hexagonal gridded maps and information layers: A novel approach for the exploration and analysis of retail data. In: **SIGGRAPH ASIA 2016 Symposium on Visualization**. New York, NY, USA: Association for Computing Machinery, 2016. (SA '16). ISBN 9781450345477. Disponível em: <https://doi.org/10.1145/3002151.3002160>.

POUDEL, D. B. Coordinating hundreds of cooperative, autonomous robots in a warehouse. **Jan**, v. 27, p. 1–13, 2013.

RAY, A. K.; BEHERA, L.; JAMSHIDI, M. Gps and sonar based area mapping and navigation by mobile robots. In: IEEE. **2009 7th IEEE International Conference on Industrial Informatics**. Cardiff, UK, 2009. p. 801–806.

ROBOTICS, C. **V-REP: Virtual robot experimentation platform**. Zürich, Switzerland: Coppelia, 2015.

RONCONI, G. B. A.; BATISTA, T. J.; MEROLA, V. The utilization of unmanned aerial vehicles (uav) for military action in foreign airspace. **UFRGSMUN**, v. 2, p. 137–180, 2014. ISSN 2318-3195.

ROSA, R. d. et al. Honeycomb map: A bioinspired topological map for indoor search and rescue unmanned aerial vehicles. **Sensors**, MDPI AG, v. 20, n. 3, p. 907, Feb 2020. ISSN 1424-8220. Disponível em: <http://dx.doi.org/10.3390/s20030907>.

ROSKAM, J. **Airplane flight dynamics and automatic flight controls - Part I**. Lawrence, KS: DARcorporation, 2006.

SAEEDI, S. et al. Neural network-based multiple robot simultaneous localization and mapping. **IEEE Transactions on Neural Networks**, IEEE, v. 22, n. 12, p. 2376–2387, 2011.

SAEEDI, S. et al. Multiple-robot simultaneous localization and mapping: A review. **Journal of Field Robotics**, v. 33, n. 1, p. 3–46, 2016. ISSN 1556-4967. Disponível em: <http://dx.doi.org/10.1002/rob.21620>.

SAHR, K. Hexagonal discrete global grid systems for geospatial computing. **Archiwum Fotogrametrii, Kartografii i Teledetekcji**, v. 22, p. 363–376, 2011.

SANDY, N. **Modelagem e análise de topologias para veículos aéreos não-tripulados do tipo multirotor**. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná - CPGEI, 2017.

SHADE, R. **Choosing where to go: mobile robot exploration**. Tese (Doutorado) — University of Oxford, 2011.

SICILIANO, B. et al. **Robotics: Modelling, Planning and Control**. London, UK: Springer, 2009.

SIEGWART, R.; NOURBAKHSH, I. R. **Introduction to Autonomous Mobile Robots**. Scituate, MA, USA: Bradford Company, 2004. ISBN 026219502X.

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. **Introduction to Autonomous Mobile Robots - Intelligent Robotics and Autonomous Agents Series**. Cambridge, Massachusetts: The MIT Press, 2011.

SIMMONS, R. G. et al. Coordination for multi-robot exploration and mapping. In: **Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence**. Austin, Texas: AAAI Press, 2000. p. 852–858. ISBN 0262511126.

SINGH, K.; FUJIMURA, K. Map making by cooperating mobile robots. In: IEEE. **[1993] Proceedings IEEE International Conference on Robotics and Automation**. Atlanta, GA, 1993. p. 254–259.

SMITH, A. J. The task of the referee. **IEEE Computer**, v. 23, n. 4, p. 65–71, April 1990.

SMITH, R. C.; CHEESEMAN, P. On the representation and estimation of spatial uncertainty. **The international journal of Robotics Research**, Sage Publications Sage CA: Thousand Oaks, CA, v. 5, n. 4, p. 56–68, 1986.

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

STACHNISS, C. Coordinated multi-robot exploration. In: **Robotic Mapping and Exploration**. Berlin, Heidelberg: Springer, 2009. p. 43–71.

STENTZ, A.; FOX, D.; MONTEMERLO, M. Fastslam: A factored solution to the simultaneous localization and mapping problem with unknown data association. In: CITESEER. **In Proceedings of the AAAI National Conference on Artificial Intelligence**. Edmonton, Canada, 2002.

STOJMENOVIC, I. Honeycomb networks: Topological properties and communication algorithms. **IEEE Transactions on Parallel and Distributed Systems**, v. 8, n. 10, p. 1036–1042, Oct 1997. ISSN 2161-9883.

THRUN, S. **Robotic Mapping: A Survey**. Pittsburgh, Pennsylvania: School of Computer Science, Carnegie Mellon University, 2002.

THRUN, S. et al. Probabilistic algorithms and the interactive museum tour-guide robot minerva. **The International Journal of Robotics Research**, SAGE Publications, v. 19, n. 11, p. 972–999, 2000.

THRUN, S.; MONTEMERLO, M. The graph slam algorithm with applications to large-scale mapping of urban structures. **The International Journal of Robotics Research**, SAGE Publications, v. 25, n. 5-6, p. 403–429, 2006.

TOMATIS, N.; NOURBAKHSH, I.; SIEGWART, R. Hybrid simultaneous localization and map building: a natural integration of topological and metric. **Robotics and Autonomous systems**, Elsevier, v. 44, n. 1, p. 3–14, 2003.

VAZQUEZ, J.; MALCOLM, C. Distributed multirobot exploration maintaining a mobile network. In: **2004 2nd International IEEE Conference on Intelligent Systems.** Varna, Bulgaria: IEEE, 2004. v. 3, p. 113–118 Vol.3.

WAHARTE, S.; TRIGONI, N. Supporting search and rescue operations with uavs. In: IEEE. **2010 International Conference on Emerging Security Technologies**. Canterbury, UK, 2010. p. 142–147.

WILLIAMS, R.; KONEV, B.; COENEN, F. Multi-agent environment exploration with ar.drones. In: **Advances in Autonomous Robotics Systems: 15th Annual Conference (TAROS 2014)**. Birmingham, UK: Springer International Publishing, 2014. p. 60–71. ISBN 978-3-319-10401-0.

YAMAUCHI, B. A frontier-based approach for autonomous exploration. In: **Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation**. USA: IEEE Computer Society, 1997. (CIRA '97), p. 146. ISBN 0818681381.

YANG, M.; BIUK-AGHAI, R. P. Enhanced hexagon-tiling algorithm for map-like information visualisation. In: **Proceedings of the 8th International Symposium on Visual Information Communication and Interaction**. New York, NY, USA: Association for Computing Machinery, 2015. (VINCI '15), p. 137–142. ISBN 9781450334822. Disponível em: <https://doi.org/10.1145/2801040.2801056>.

ZHANG, L. et al. Rapidly-exploring random trees multi-robot map exploration under optimization framework. **Robotics and Autonomous Systems**, p. 103565, 2020. ISSN 0921-8890. Disponível em: <http://www.sciencedirect.com/science/article/pii/S092188902030405X>.

ZHOU, Y. et al. Collision and deadlock avoidance in multirobot systems: A distributed approach. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, v. 47, n. 7, p. 1712–1726, July 2017. ISSN 2168-2232.

APÊNDICE A – ALGORITMOS DE LOCALIZAÇÃO E MAPEAMENTO SIMULTÂNEOS

No início da década de 80, o problema de localização de um robô móvel em um determinado ambiente começou a ser pesquisado. Este problema ficou conhecido como SLAM (*Simultaneous Localization and Mapping* - Mapeamento e Localização Simultâneos) (AULINAS et al., 2008), que busca a definição do local onde um robô autônomo está inserido e sua posição neste ambiente. Este mapeamento e localização de forma simultânea tem despertado a atenção da comunidade acadêmica, onde diversos algoritmos para SLAM são propostos. As pesquisas desenvolvidas por Smith e Cheeseman (1986) e por Durrant-Whyte (1988) definiram os primeiros modelos baseados em estruturas probabilísticas, com algoritmos baseados em Filtros de Kalman (EKF-SLAM) (SMITH, 1990). Alguns outros modelos são propostos, como FastSLAM (STENTZ et al., 2002), GraphSLAM (THRUN; MONTEMERLO, 2006), PHD-SLAM (MULLANE et al., 2011), DP-SLAM (ELIAZAR; PARR, 2003), Visual SLAM (PTAM (*Parallel Tracking and Mapping*) (KLEIN; MURRAY, 2007) e OctoMap (HORNUNG et al., 2013)). O estudo dos algoritmos de SLAM neste trabalho se dá pela necessidade de compreender as diversas formas e técnicas empregadas em especial na construção dos mapas pela comunidade científica. Dessa forma, a atividade de localização não é o foco deste trabalho.

A.1 EKF-SLAM

O algoritmo EKF-SLAM foi proposto por Smith e Cheeseman (1986) e Durrant-Whyte (1988), o qual baseia-se em filtros de Kalman. Estes filtros representam as distribuições dos espaços (trajetórias do robô) com gaussianas, fazendo o cálculo em duas etapas: predição e correção. A etapa de predição tem como objetivo estimar a próxima posição do robô baseando-se na sua posição atual no espaço, bem como as leituras dos demais sensores sobre o mapa existente/conhecido. A etapa de correção ajusta as estimativas levantadas a partir das leituras dos sensores e do posicionamento de objetos encontrados no mapa. Tanto na etapa de predição quanto na etapa de correção, os erros identificados a partir das incertezas medidas pelos sensores

são representados por matrizes de covariância, as quais determinam o peso que novas leituras do espaço irão exercer na matriz calculada resultante. Quando uma leitura obtiver um grau muito alto de incerteza, menos ele influenciará na estimativa final. O mapa gerado é um mapa métrico.

A implementação utilizando filtros de Kalman adéqua-se bem para distribuições lineares. Porém, muitos modelos de movimentação e observação de robôs utilizam funções não-lineares. Como forma de contornar esta limitação, os Filtros de Kalman Estendidos (EKF) aplicam uma aproximação das funções não-lineares de movimentação e observação dos robôs através de séries de Taylor. Esta abordagem é vantajosa para os casos onde os modelos de movimentação e observação retornem leituras com não-linearidade moderada, entretanto quando o grau de incerteza das leituras dos sensores é relativamente alta, a linearização pode gerar um aumento de erros.

A.2 FASTSLAM

O algoritmo FastSLAM, proposto por Stentz et al. (2002) , faz uma abordagem voltada para filtros de partículas, baseando-se em amostragens recursivas de Monte Carlo. Diferentemente do EKF-SLAM, a trajetória do robô não é modelada por distribuições gaussianas (com necessidade de linearidade ou aproximações por séries de Taylon), mas sim a partir de um conjunto de amostras (partículas). Estas partículas representam as trajetórias possíveis do robô e, para cada trajetória, um mapa diferente é computado. O filtro de partículas adota um algoritmo recursivo que se divide nas seguintes etapas: amostragem, definição dos pesos, e reamostragem. A amostragem faz a extração de partículas de uma distribuição proposta. A definição dos pesos atribui um fator de importância para a partícula extraída na fase de amostragem, o qual é calculado pela razão entre a distribuição alvo e a distribuição proposta. Por fim, a reamostragem faz a seleção de partículas de acordo com os pesos atribuídos. O mapa gerado é um mapa métrico.

O FastSLAM aborda a incorporação de múltiplas hipóteses a partir de novas partículas, possibilitando que o problema de associação de dados seja tratado internamente (MONTEMERLO; THRUN, 2003). Além disso, com um grande número de amostras é possível representar eficientemente modelos não-lineares ou não-gaussianos. Entretanto, o número de partículas aumenta exponencialmente com a dimensão do estado do mapa (LUCAS, 2009).

A.3 GRAPHSLAM

A abordagem adotada pelo GraphSLAM (THRUN; MONTEMERLO, 2006) é a transformação do problema de SLAM em um problema de otimização de grafo. Com a pose do robô (posição e orientação do robô no mapa), monta-se um grafo onde as poses são os nós, e as arestas definem as restrições entre as poses. Este algoritmo não calcula o mapa diretamente, mas apenas o faz quando uma trajetória é encontrada e, dessa forma, necessita de menos processamento para sua execução. O mapa gerado é topológico.

A otimização do grafo é tratada como um problema de quadrados mínimos, minimizando os erros impostos pelas relações dos nós. O problema dos quadrados mínimos normalmente emprega algoritmos iterativos Gauss-Newton e Levenberg-Marquardt, mas sempre de um ponto inicial para encontrar um valor mínimo para o cálculo do erro. Desse modo, não se pode garantir que um valor mínimo global será encontrado.

A.4 PHD-SLAM

O PHD-SLAM (*Probability Hyphotesis Density - SLAM*) (MULLANE et al., 2011) utiliza conjuntos aleatórios como descritores, ao invés de vetores, na descrição de modelos. Desse modo, o valor da pose do robô (posição e orientação), da localização dos elementos no mapa e a estimativa do número de elementos no mapa encontram-se em um mesmo framework.

Diferentemente dos algoritmos anteriores, nos quais as formas de redução de erros ou taxas de valores aproximados ao valor real armazenado eram calculados, o PHD-SLAM processa todos os dados dos sensores, levando a uma detecção mais fiel do número de referências no mapa (MORATUWAGE et al., 2014) (ADAMS et al., 2014), porém podem levar a altas taxas de falsos positivos (LEE et al., 2013).

Em outras palavras, a vantagem o PHD-SLAM é obter uma representação mais fiel possível, porém, uma melhor precisão pode significar maiores custos computacionais e de armazenamento. Tal situação deve ser cuidadosamente considerada na definição da plataforma de hardware que irá ser usada no robô autônomo. O mapa gerado é métrico.

A.5 DP-SLAM

O algoritmo de DP-SLAM é voltado para o uso de laser (LRF), e apresenta uma solução precisa mantendo uma distribuição conjunta através de mapas e as posições do robô

móvel usando filtro de partículas, fornecendo mapas precisos (ELIAZAR; PARR, 2003). Quando se utiliza filtro de partículas para resolver o problema de SLAM, cada partícula corresponde a uma trajetória específica através do ambiente, e possui um mapa específico associado a ele. Quando uma partícula é reamostrada, o mapa inteiro é tratado como parte do estudo oculto que está sendo mostrado.

Para que um filtro de partículas acompanhe corretamente a distribuição conjunta sobre a posição do robô móvel e o mapa, é necessário manter um mapa separado para cada partícula. Durante a fase de reamostragem do filtro de partículas, cada partícula pode ser reamostrada várias vezes. Com isso, é necessário copiar o mapa inteiro para cada nova partícula, para manter cada hipótese do conjunto de atualizações do mapa. Esta distribuição conjunta, e a correspondente necessidade de múltiplas hipóteses de mapas, é ponto de partida crucial para o processo de SLAM (ELIAZAR, 2005).

O DP-SLAM usa uma representação eficiente para fazer cópias de mapas, ao mesmo tempo reduzindo a memória total exigida para representar este grande número de mapas através de um método de mapeamento chamado partículas distribuídas (DP-Mapping). Tal abordagem explora as redundâncias significativas entre os diferentes mapas. A principal vantagem é que o DP-Mapping foi proposto para o uso com sensores laser, quando este for o sensor principal de captura de dados para mapeamento e localização. Contudo, a principal desvantagem é ele apresenta complexidade na integração em filtros de partículas já existentes, dificultado a integração com sistemas de localização pré-existentes (OLIVEIRA, 2012). O mapa gerado é métrico.

A.6 VISUAL SLAM

Algoritmos de SLAM que utilizam sensor de visão são conhecidos como Visual SLAM (VSLAM). Um ambiente pode ser representado como uma coleção de observações de um sensor de visão, ou seja, um conjunto de imagens. Uma imagem pode ser representada com uma coleção de características visuais. Com isso há uma alta probabilidade de que muitas imagens possuam observações comuns do ambiente, consequentemente características comuns. A coleção destas características do conjunto de imagens provê uma representação abstrata do ambiente. Assim, o VSLAM estima a localização 3D correta destas características a partir da localização da câmera que faz a captura das imagens.

Um exemplo de algoritmo que estende o conceito de VSLAM é o PTAM (KLEIN; MURRAY, 2007). Com o propósito voltado para pequenos ambientes de realidade aumentada

(AR), o PTAM é um algoritmo de SLAM baseado em características que rastreia e mapeia diversas características para alcançar robustez. O seu diferencial está em executar paralelamente em tempo real a estimação de movimento e tarefas de mapeamento além de entregar um ajuste de bordas ao invés de filtragem Bayesianas para o refinamento do mapa e da pose. Entretanto, o PTAM é projetado especificamente para aplicações de AR e apresenta bons resultados para espaços pequenos sem gerenciamento de um mapa global. O mapa gerado é métrico.

A.7 OCTOMAP

OctoMap (HORNUNG et al., 2013) é um algoritmo de geração de mapas 3D o qual faz uso de uma representação *Octree* para evolução do mapa. Um *Octree* é uma estrutura de dados hierárquica para subdivisão espacial do ambiente em três dimensões, onde cada nó representa um volume de cubo, normalmente chamado de VOXEL. Cada VOXEL pode ser dividido em oito sub-volumes de mesmo tamanho. Um nó em um *Octree* será dividido apenas se dados do sensor estão disponíveis. A ocupação do VOXEL é modelada probabilisticamente. Uma das grandes vantagens do *Octree* é que ele mapeia células livres e ocupadas, possibilitando uma visão ampla do espaço em especial para equipes de resgate. A estrutura do mapa é híbrida.

A abordagem OctoMap faz o armazenamento de dados 3D em formatos semelhantes ao propostos neste trabalho para representação visual do ambiente mapeado, porém a abordagem proposta nesta tese não utiliza essas leituras 3D para representar o mapa do ambiente, e sim a representação através de um mapa topológico. Na abordagem OctoMap, a informação 3D é parte do mapa para uso dos robôs. Além disso, o trabalho de Hornung et al. (2013) é voltado para apenas um robô.

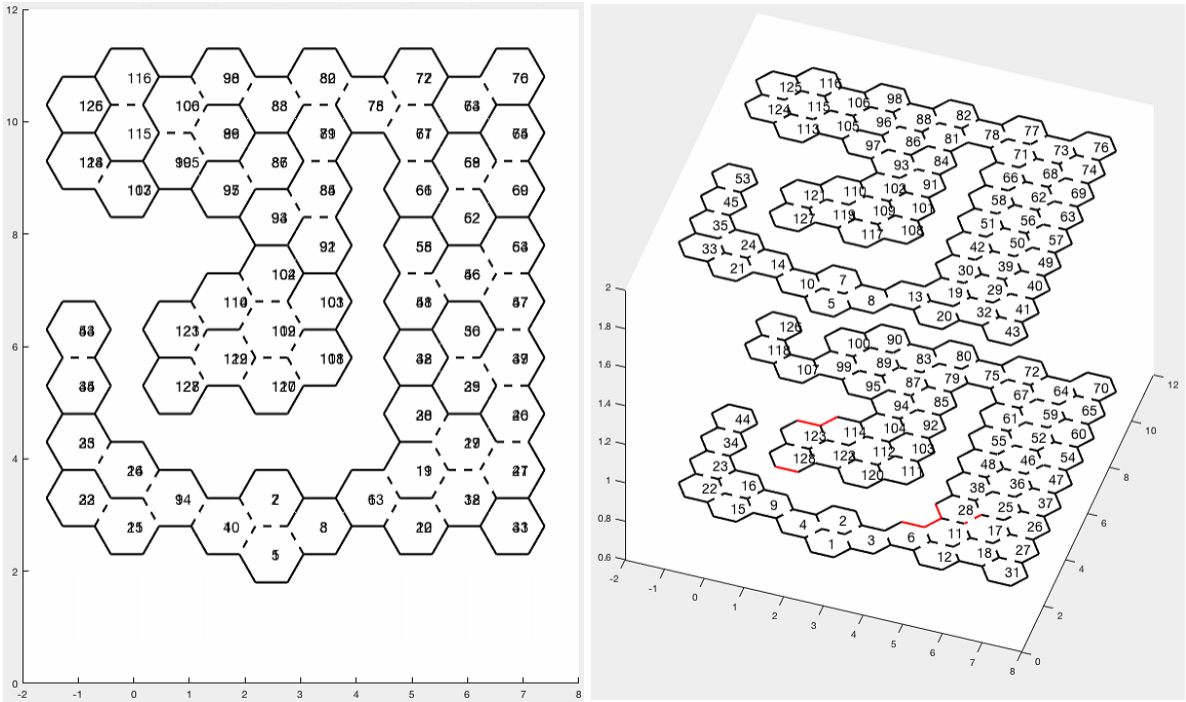
APÊNDICE B – FIGURAS DO MAPEAMENTO

A seguir são apresentadas as figuras que mostram o mapeamento do cenário 2 e a evolução do mapeamento realizado nos cenários 1 e 2, variando a estratégia adotada para a alocação de tarefas de exploração (FIFO, DE e DER), conforme discutido no capítulo 5.

As figuras 61, 62 e 63 apresentam as simulações realizadas no cenário 2, utilizando as estratégias definidas na seção 4.3.3 (FIFO, DE e DER).

As figuras 64, 65, 66, 67, 68 e 69 apresentam a evolução do mapeamento em cada cenário simulado com as diferentes estratégias da seção 4.3.3, apresentando em seis proporções (figuras (a), (b), (c), (d), (e) e (f)) na ordem de mapeamento.

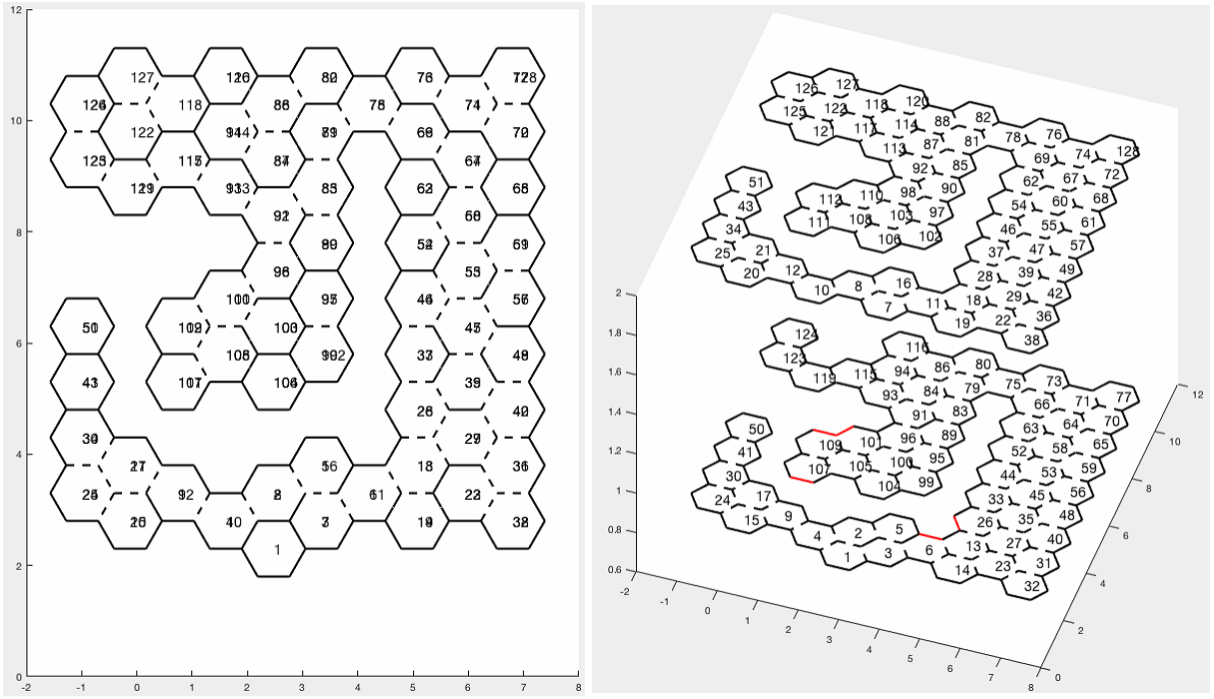
Figura 61: Mapeamento cenário 2 - FIFO.



(**a**) Visão superior. (**b**) Visão em perspectiva.

Fonte: Autoria própria.

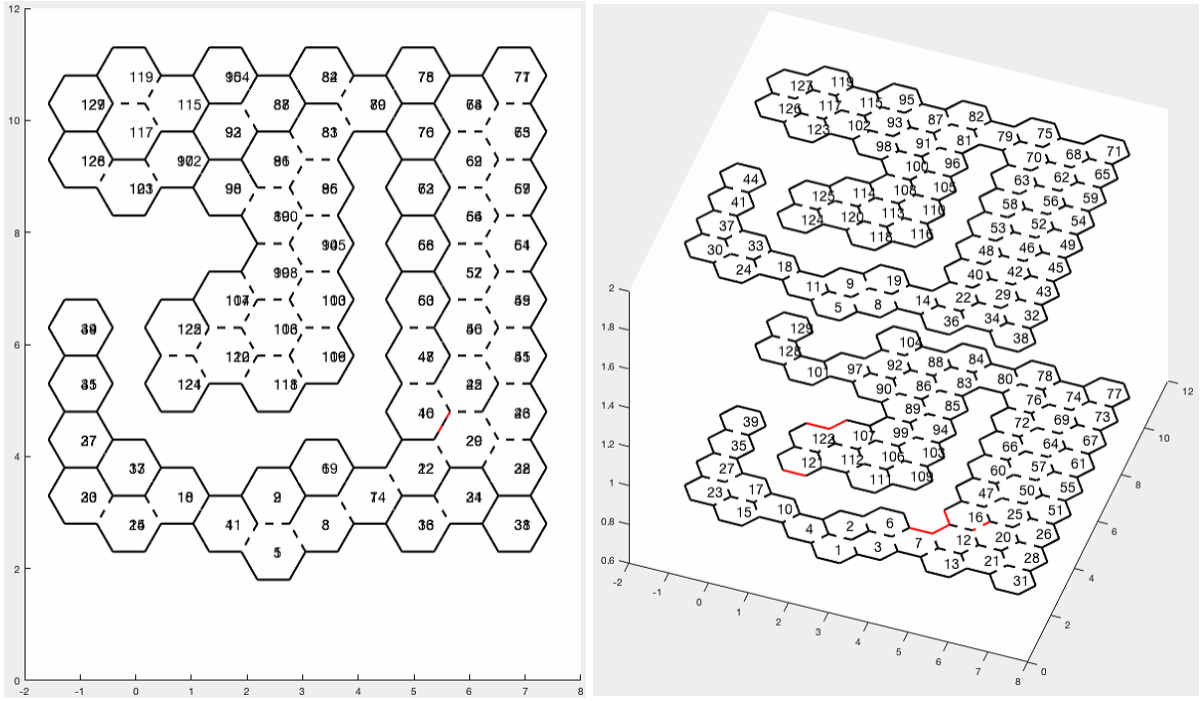
Figura 62: Mapeamento cenário 2 - DE.



(**a**) Visão superior. (**b**) Visão em perspectiva.

Fonte: Autoria própria.

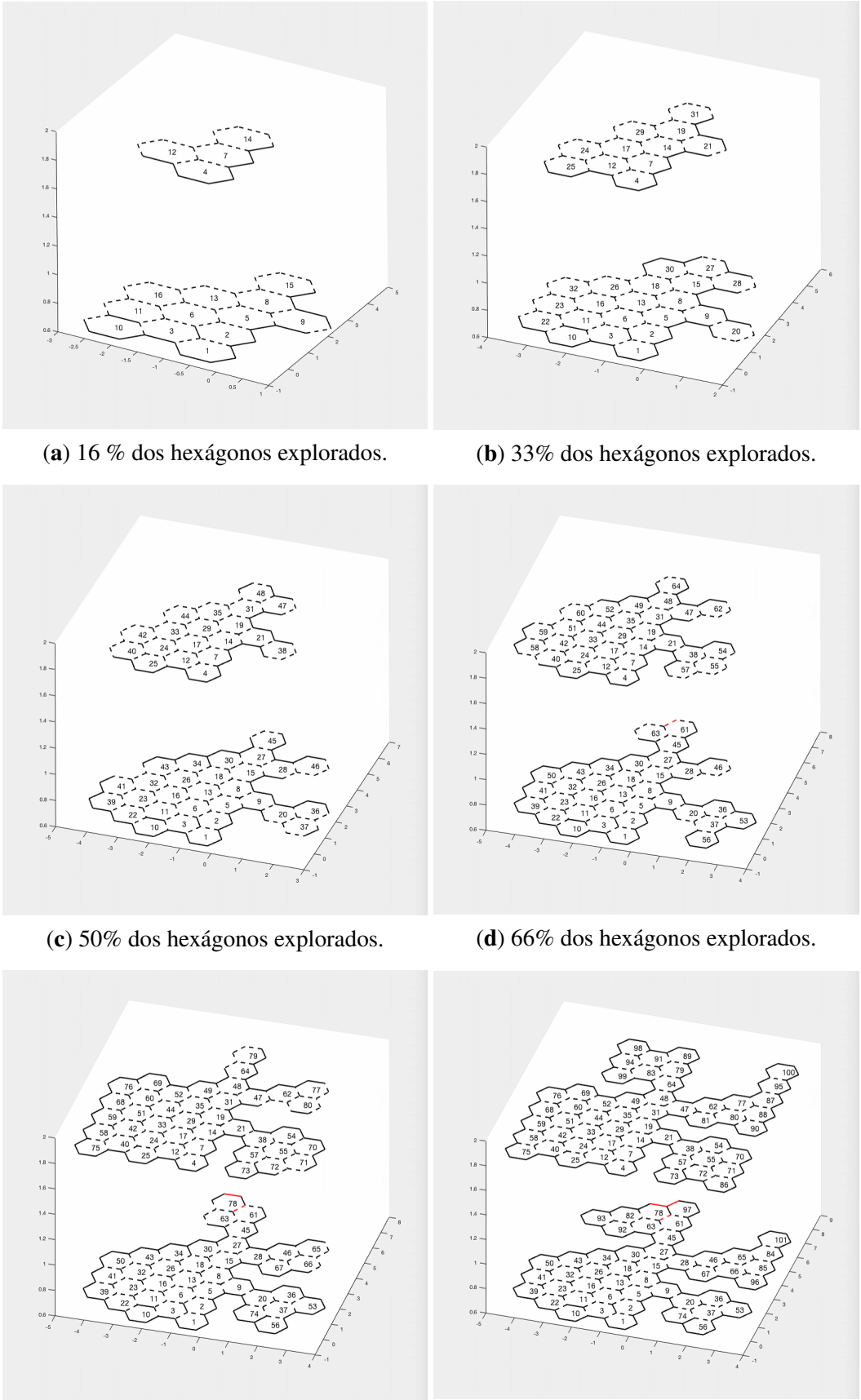
Figura 63: Mapeamento cenário 2 - DER.



(**a**) Visão superior. (**b**) Visão em perspectiva.

Fonte: Autoria própria.

Figura 64: Evolução do mapeamento do cenário 1 - FIFO. É possível verificar que a evolução do mapa, considerando uma estrutura de grafo, apresenta-se em largura.



(**a**) 16 % dos hexágonos explorados.

(**b**) 33% dos hexágonos explorados.

(**c**) 50% dos hexágonos explorados.

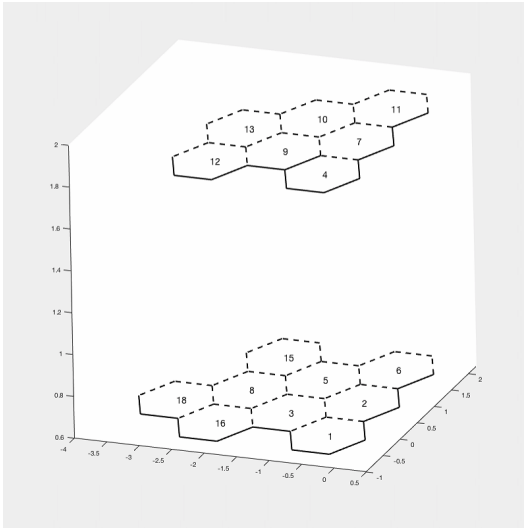
(**d**) 66% dos hexágonos explorados.

(**e**) 83% dos hexágonos explorados.

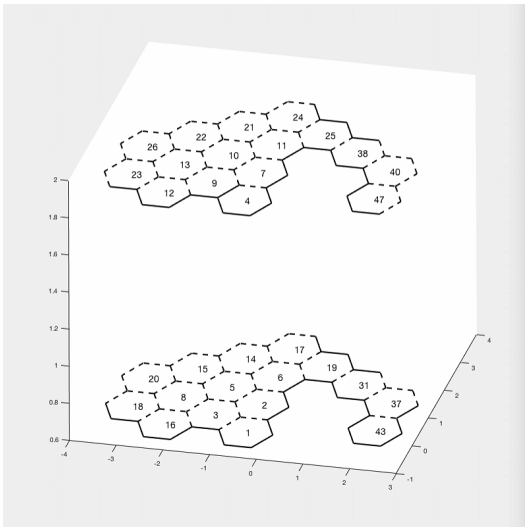
(**f**) 100% dos hexágonos explorados.

Fonte: Autoria própria.

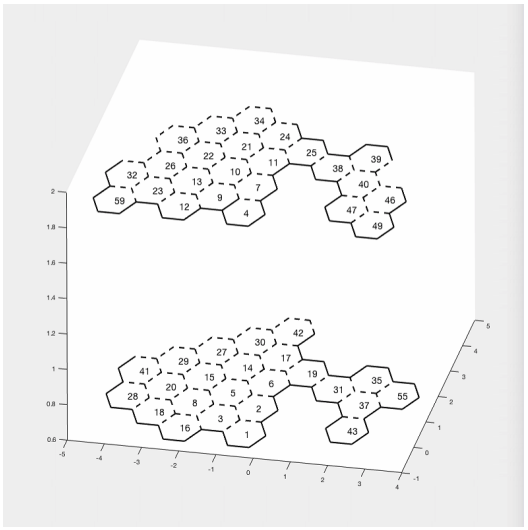
Figura 65: Evolução do mapeamento do cenário 1 - DE. De forma semelhante ao FIFO, a evolução do mapeamento, considerando uma estrutura de grafos, apresenta-se em largura.



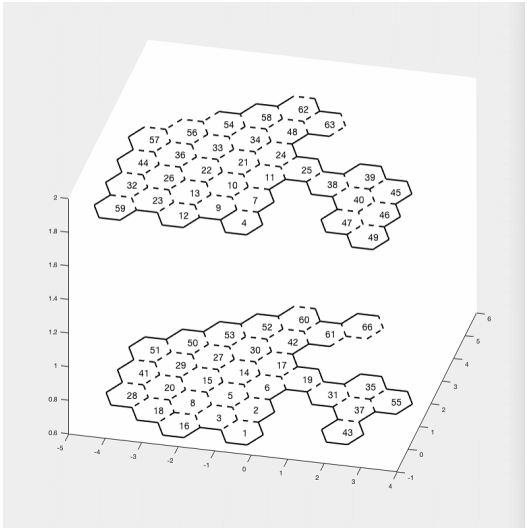
(**a**) 16 % dos hexágonos explorados.



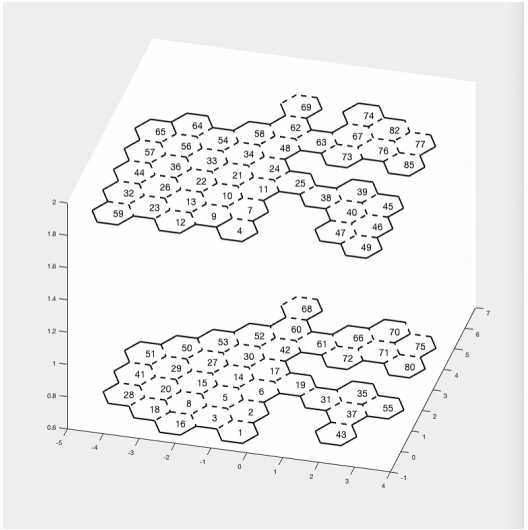
(**b**) 33% dos hexágonos explorados.



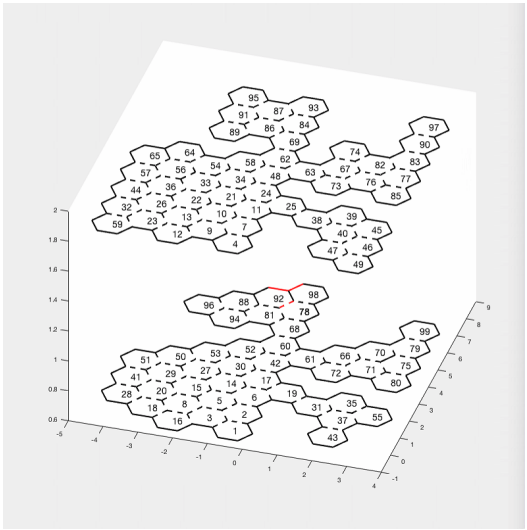
(**c**) 50% dos hexágonos explorados.



(**d**) 66% dos hexágonos explorados.



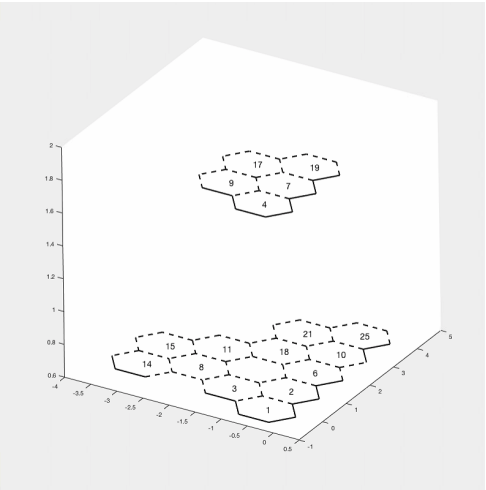
(**e**) 83% dos hexágonos explorados.



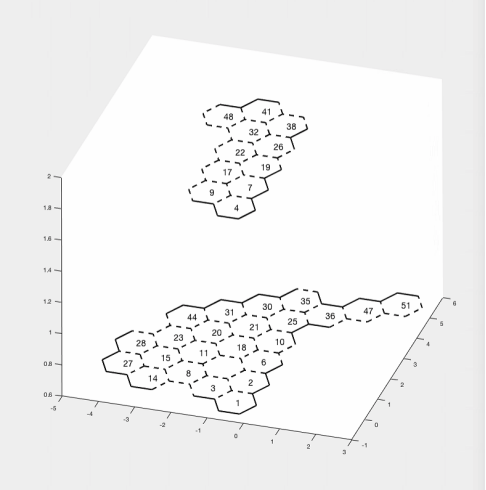
(**f**) 100% dos hexágonos explorados.

Fonte: Autoria própria.

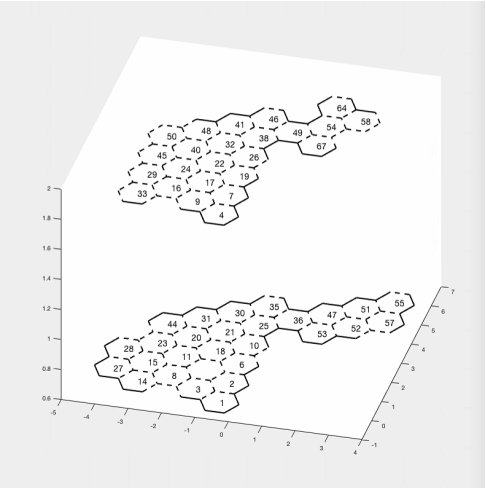
Figura 66: Evolução do mapeamento do cenário 1 - DER. Diferentemente das estratégias FIFO e DE, aqui a evolução do mapeamento, considerando a estrutura de grafos, evolui com um comportamento semelhante a varredura em profundidade. Esse comportamento pode ser verificado nas figuras relacionadas ao cenário 2.



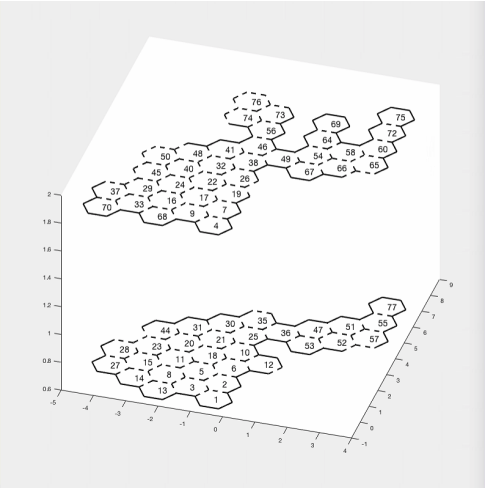
(**a**) 16 % dos hexágonos explorados.



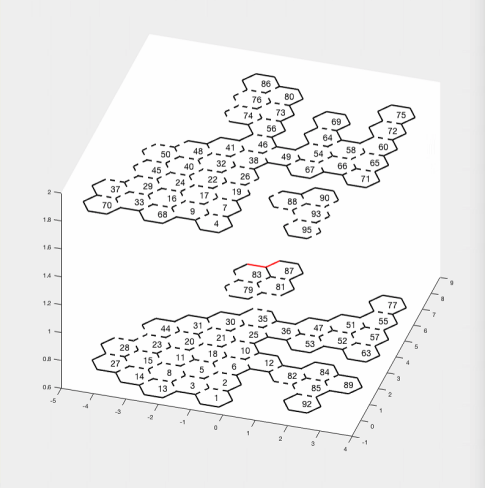
(**b**) 33% dos hexágonos explorados.



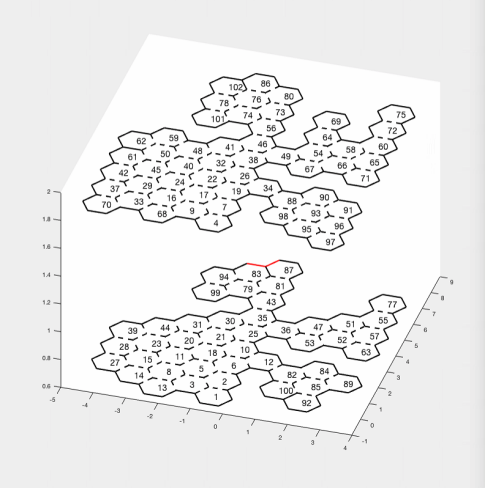
(**c**) 50% dos hexágonos explorados.



(**d**) 66% dos hexágonos explorados.



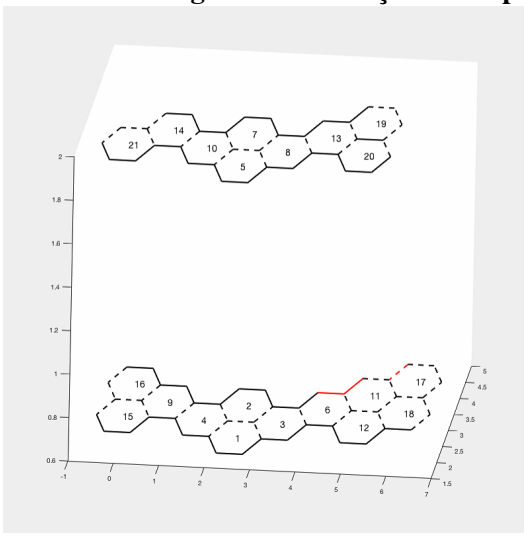
(**e**) 83% dos hexágonos explorados.



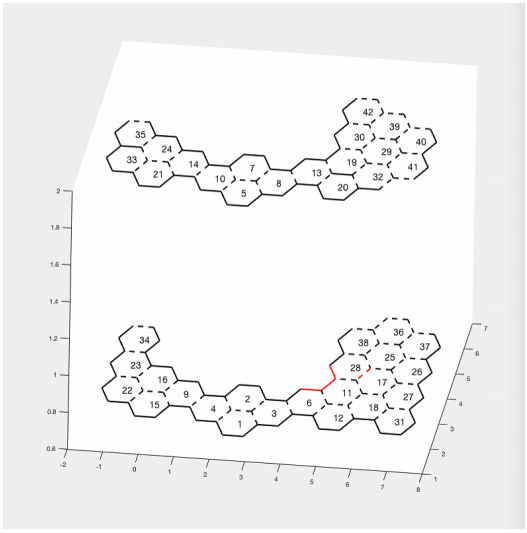
(**f**) 100% dos hexágonos explorados.

Fonte: Autoria própria.

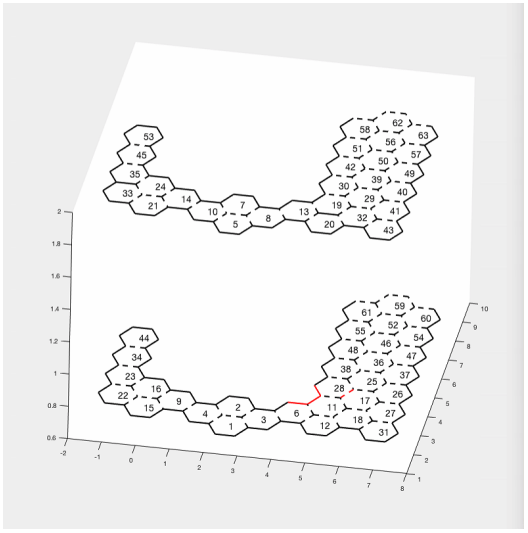
Figura 67: Evolução do mapeamento do cenário 2 - FIFO .



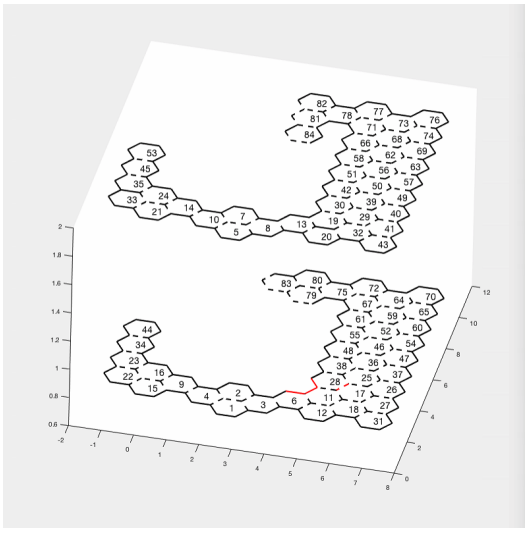
(**a**) 16 % dos hexágonos explorados.



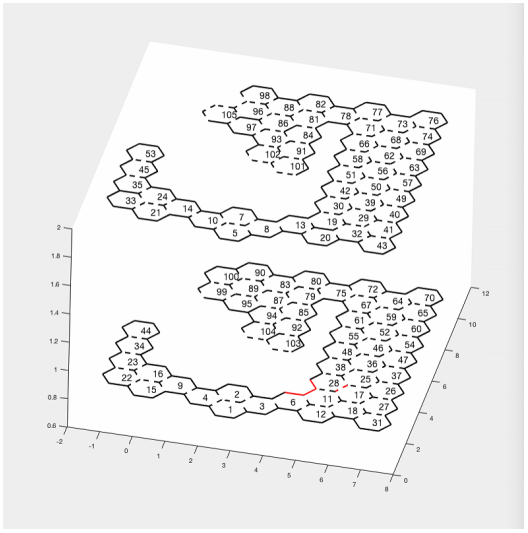
(**b**) 33% dos hexágonos explorados.



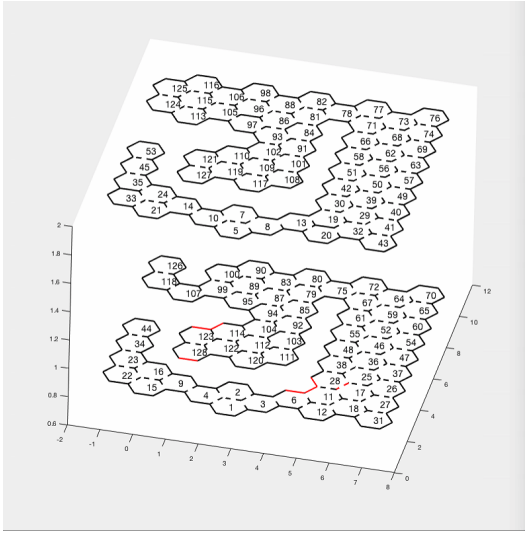
(**c**) 50% dos hexágonos explorados.



(**d**) 66% dos hexágonos explorados.



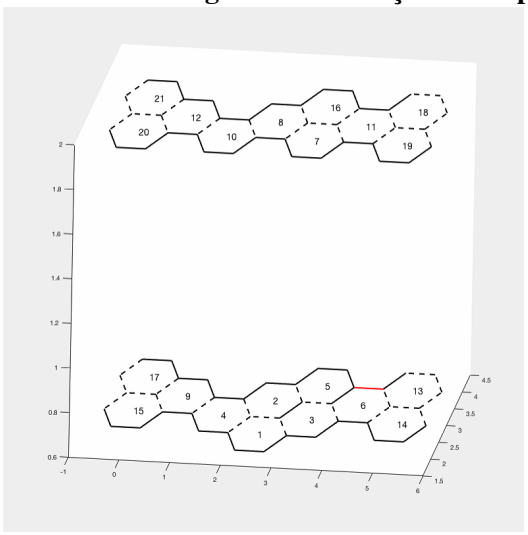
(**e**) 83% dos hexágonos explorados.



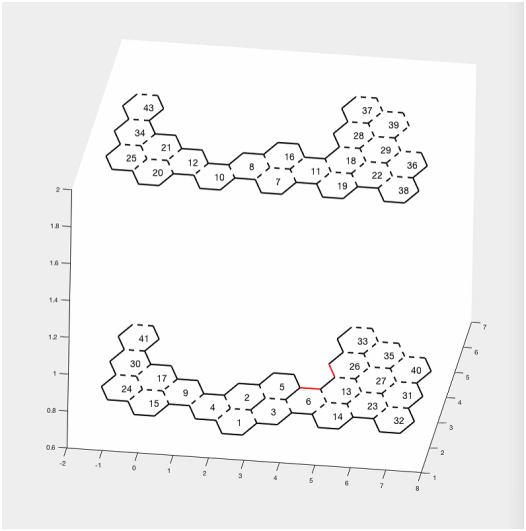
(**f**) 100% dos hexágonos explorados.

Fonte: Autoria própria.

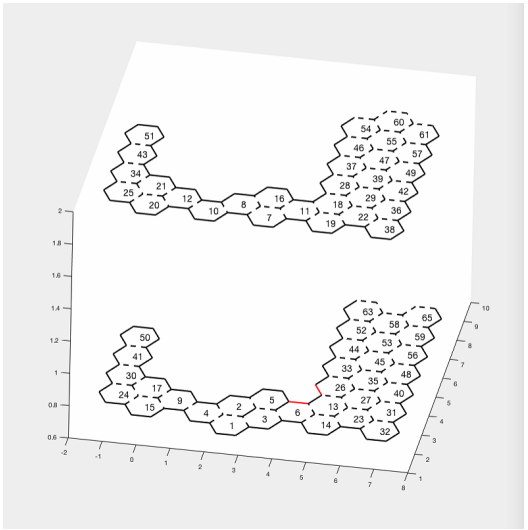
Figura 68: Evolução do mapeamento do cenário 2 - DE .



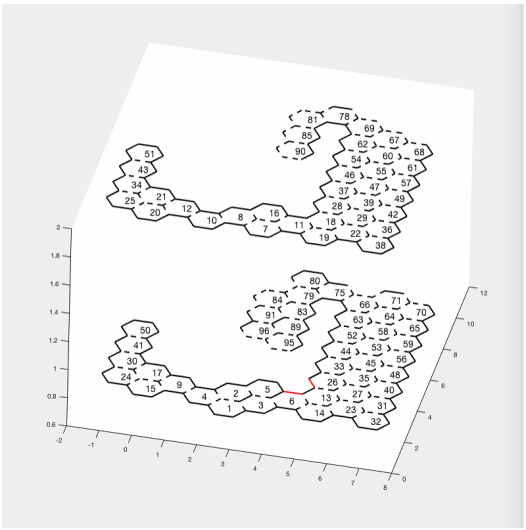
(**a**) 16 % dos hexágonos explorados.



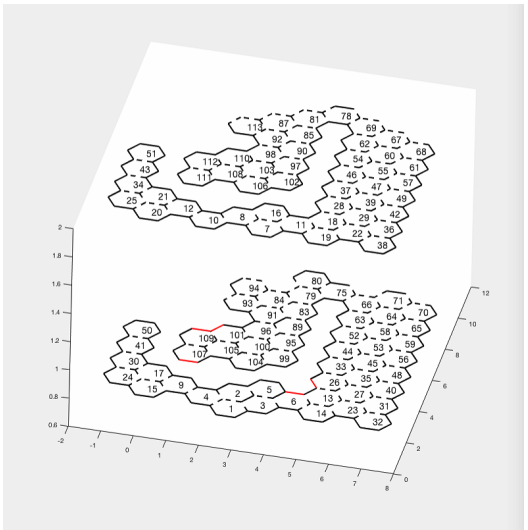
(**b**) 33% dos hexágonos explorados.



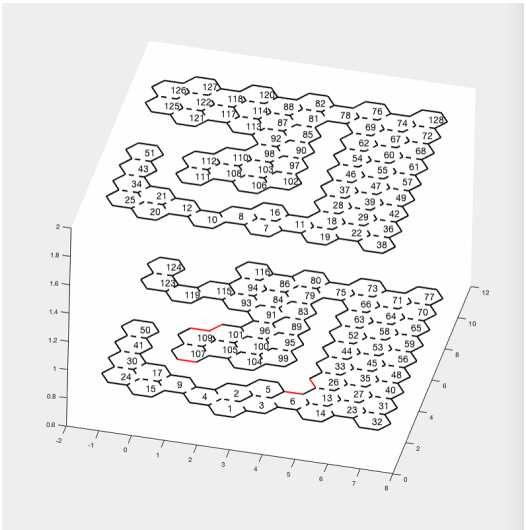
(**c**) 50% dos hexágonos explorados.



(**d**) 66% dos hexágonos explorados.



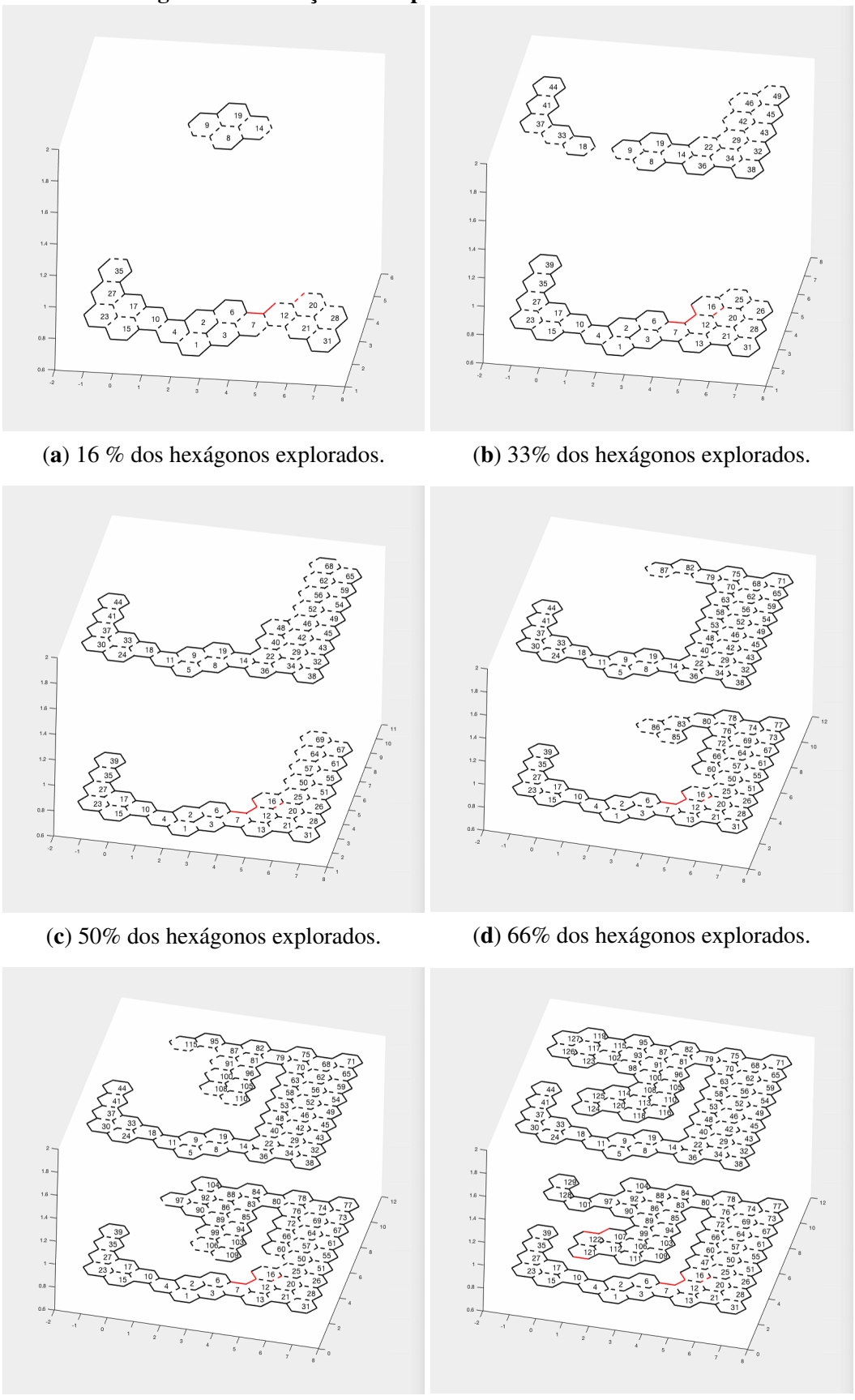
(**e**) 83% dos hexágonos explorados.



(**f**) 100% dos hexágonos explorados.

Fonte: Autoria própria.

Figura 69: Evolução do mapeamento do cenário 2 - DER .



(**a**) 16 % dos hexágonos explorados.

(**b**) 33% dos hexágonos explorados.

(**c**) 50% dos hexágonos explorados.

(**d**) 66% dos hexágonos explorados.

(**e**) 83% dos hexágonos explorados.

(**f**) 100% dos hexágonos explorados.

Fonte: Autoria própria.

APÊNDICE C – ARTIGO EM PROCESSO DE REVISÃO

Artigo submetido e em processo de revisão no periódico IEEE Latin America, com o título “UAV Indoor Mapping: Methods, Challenges and Open Issues”.

UAV Indoor Mapping: Methods, Challenges and Open Issues

R. D. Rosa and M. A. Wehrmeister *Member, IEEE,*

***Abstract*—The focus of this work is the mapping of indoor places with cooperative robots, in order to explore environments that have suffered natural disasters or are considered hostile, which would provide safety to rescue workers and agility in this process. The objective is to perform a systematic review in repositories that contain works with focus on the mapping of closed spaces with cooperative robots. For that, we use publications from the last decade in scientific repositories that address the problem of cooperative mapping and simultaneous location (SLAM) including unmanned aerial vehicles (UAV). Our questions seek to identify not only the main solutions in indoor mapping with multiple robots but also the main challenges in the area. The answers of research questions indicate that the data environment processing, the characteristic identification and restrictions of the space to be mapped and the communication issues that deliver a solution in fast time are points that might be a focus of studies.**

***Index Terms*—Mobile Robotic, UAV, cooperative SLAM, 3D mapping.**

I. Introdução

O uso de robôs para resolver problemas torna-se cada vez mais frequente. Além da ampla aplicação na indústria, o desenvolvimento e aplicação de modelos móveis cresce com a popularização e redução de custos de sensores e atuadores. Estas novas aplicações demostram que a robótica móvel pode realizar tarefas complexas, podendo navegar entorno de ambientes desconhecidos e desviando de obstáculos inesperados, reagindo de forma inteligente aos estímulos do ambiente [1]. Outra aplicação da robótica móvel é no suporte a times de resgate em situações de desastres naturais ou catástrofes. Algumas vezes a atividade de exploração pode por em risco a vida de profissionais de equipes de resgate. O uso de veículos aéreos não-tripulados (*Unmanned Aerial Vehicles - UAV*) pode auxiliar nas atividades de resgate, especialmente em áreas fechadas (*indoor*) onde por vezes é impossível a chegada ou movimentação de um robô terrestre.

O acesso a locais fechados e desconhecidos necessita de técnicas para a definição do espaço onde o robô se encontra, gerando mapeamentos do ambiente com o objetivo de auxiliar equipes no reconhecimento destes espaços onde o uso de sistemas de posicionamento por satélite (GPS) é inviável. Assim, um robô autônomo precisa tratar dois problemas críticos para sobreviver e navegar em seu ambiente: mapear o ambiente e buscar sua própria localização no mapa [2].

Muitos algoritmos de localização e mapeamento simultâneos (*Simultaneous localization and mapping - SLAM*) foram desenvolvidos abordando diferentes técnicas em diferentes tipos de robôs, os quais possuem uma grande variedade de sensores. De uma forma geral, as técnicas de SLAM trabalham com a criação de mapas que podem ser subdivididos em diversas categorias, como mapas *métricos, topológicos ou híbridos* [3]. Os mapas métricos buscam extrair as caractísticas e propriedades geométricas do ambiente e são representados como *grid map, feature map ou geometric map*[4]. Os mapas topológicos apresentam o ambiente como uma representação em grafos, onde os nós representam lugares e objetos de interesse, enquanto os arcos representam a relação espacial ou o caminho entre os vértices. Outras classificações de técnicas de SLAM podem ser realizadas, como a encontrada em [5]. Considerando um cenário onde equipes de resgate necessitam obter conhecimento prévio da estrutura interna de um espaço desconhecido, o uso de múltiplos agentes no processo de mapeamento pode trazer informações de forma mais rápida do que apenas uma solução com apenas um único robô.

O nosso trabalho tem como objetivo realizar uma revisão sistemática das técnicas de mapeamento e modelagem de ambientes fechados utilizando múltiplos robôs móveis autônomos. Busca-se aqui apresentar os principais trabalhos desenvolvidos nos últimos anos considerando critérios de seleção/exclusão previamente definidos, quais são as soluções apresentadas tanto para UAV quanto para demais robôs móveis.

II. A Revisão Sistemática

A busca pela compreensão e levantamento de questões a respeito do desenvolvimento de trabalhos que foquem na resolução de problemas de modelagem de ambientes internos com múltiplos robôs móveis sem o auxílio de sistemas de localização por satélite (GPS) pautou-se pelas considerações dos trabalhos de [6] e [7]. Com isso, buscamos agregar informações empíricas no sentido de apresentar as diferente soluções existentes e o desafios a serem superados por pesquisadores. Outras revisões da literatura abordam técnicas de SLAM ([3] e [8]), porém não possuem um enfoque no mapeamento de ambientes fechadados com múltiplos agentes, em especial, UAVs.

A. Questões de Pesquisa

O objetivo do nosso trabalho é identificar as principais formas de mapeamento cooperativo de ambientes internos com o uso de múltiplos robôs móveis, em especial veículos aéreos não-tripulados (UAV). Dessa forma, definimos as seguintes questões de pesquisa que buscamos responder com este trabalho:

- QP1: Quais os problemas existentes que necessitam de algoritmos de mapeamento e localização cooperativos *indoor*?

- QP2: Quais as principais técnicas e métodos aplicados na resolução de mapeamento e localização cooperativos *indoor* para UAV?
- QP3: Quais característica de cada solução pesquisada?
- QP4: As soluções apresentam implementação real?
- QP5: Quais problemas/desafios/limitações são encontrados no desenvolvimento de algoritmos de mapeamento e localização cooperativos *indoor* para UAV?

As possíveis respostas para a QP1 são prioritariamente respondidas a partir das leituras da introdução de cada trabalho. Desse modo, buscamos apresentar em uma visão macro os principais problemas existentes na construção cooperativa de mapas e do processo de localização de múltiplos robôs autônomos dentro de um espaço fechado.

Na questão de pesquisa QP2 buscamos elencar quais são as principais técnicas de SLAM ou mapeamento 3D de espaços fechados, trazendo quais soluções são desenvolvidas no trabalho cooperativo de múltiplos robôs UAVs *indoor*. As respostas desta pergunta são respondidas a partir das fundamentações teóricas bem como na explicação da solução apresentada por cada trabalho.

A questão de pesquisa QP3 é responsável por explorar as particularidades de cada trabalho, como arquiteturas utilizadas, tipos e quantidades de sensores empregados além das extensões de técnicas já previamente conhecidas. Nossas leituras tanto na introdução quanto no desenvolvimento e relato de experiência respondem tal pergunta de pesquisa.

Com o objetivo de buscar soluções que já são existentes no mundo real (não apenas simulado), é definido a questão de pesquisa QP4. Encontramos as respostas majoritariamente no relato dos experimentos.

Na busca pelas respostas das questões de pesquisa QP1, QP2, QP3 e QP4, a partir das leituras do corpo dos artigos pesquisados bem como em suas considerações e conclusões, formamos a reposta para a questão de pesquisa QP5, onde a qual questiona quais são as dificuldades ou limitações encontradas pelos pesquisadores autores dos trabalhos aqui considerados.

B. Processo de Pesquisa

O proceso de pesquisa foi o primeiro passo na identificação dos trabalhos analisados. Com o objetivo de dar amplitude nas buscas em repositórios, fizemos as consultas considerando os seguintes repositórios: ACM Computing Survey (http://csur.acm.org/), ACM Digital Library (http://dl.acm.org/), Google Scholar (http://scholar.google.com), IEEE Xplorer (http://ieeexplore.ieee.org/ Xplore/home.jsp) e Springer (http://www.springer.com).

No processo de pesquisa para todos os repositórios, realizamos as bucas considerando o seguinte termo: *"cooperative UAV SLAM indoor 3D"*. Para restringir a pesquisa em trabalhos desenvolvidos na última década, foi definida a limitação de trabalho desenvolvidos entre 2006 e 2019. O número de artigos encontrados em cada repositório são: ACM Computing Survey - 35; ACM Digital Library - 18101; Google Scholar - 816; IEEE Xplorer - 205; Springer - 25. A data de finalização da seleção dos trabalhos foi 30 de agosto de 2019. Dada a quantidade de trabalhos listados, definimos os seguintes critérios de inclusão e exclusão, baseando-se nas informações contídas no título e no resumo:

1) Ser um trabalho voltado para robótica móvel: busca-se aqui que o artigo apresente uma solução aplicada a um robô móvel, independentemente de arquitetura;
2) Ser *indoor* ou *GPS-denied*: o objetivo desta restrição é focar em trabalhos que atentem-se para soluções em espaços fechados;
3) Ser SLAM ou de mapeamento: busca-se aqui restringir a pesquisa aos trabalhos que abordem alguma técnica de SLAM ou apenas mapeamento de espaços fechados, excluindo trabalhos voltados para outras áreas da robótica móvel;
4) Estar entre os 100 trabalhos mais relevantes de cada repositório;
5) Trabalho escrito em língua inglesa ou portuguesa;
6) Obter pontuação superior a 60: pontuação dada pela existência das palavras-chave do termo pesquisado no título e/ou resumo;

Para o último critério de inclusão e exclusão, a pontuação de cada palavra-chave é a seguinte: *cooperative*: 30, *UAV*: 30, *SLAM*: 35, *indoor*: 15 e *3D*: 10.

Considerando os critérios aplicados no processo de leitura do título e resumo dos diversos artigos apresentados por cada repositório, foram selecionados um total de 89 trabalhos que previamente atendiam os critérios estabelecidos. Com a definição dos artigos passados pelos critérios de inclusão e exclusão baseado na leitura do título e resumo, demos início a leitura completa dos textos. Com esta leitura mais aprofundada, observamos que, mesmo fazendo a filtragem pelos critérios previamente estabelecidos, alguns artigos não se encaixaram com os objetivos deste trabalho. Assim, uma nova quantidade de 74 artigos aptos a avaliação foi encontrado. Dos excuídos, 14 trabalhos não foram analisados devido ao seu enfoque ser diferente do assunto aqui pesquisado, e consequentemente não atender aos critérios de inclusão e exclusão. Durante a leitura, 4 trabalhos apresentam-se com o enfoque de solução *outdoor*, 7 abodam posicionamento e planejamento de trajetória, 2 fazem revisão teórica e de bibliografia geral, enquanto 1 faz referência apenas a tecnologia de sensor. Esta situação aconteceu devido a tanto o título quanto o resumo apresentarem palavras relacionadas com os critério de inclusão e exclusão previamente definidos, como *UAV, SLAM, 3D* e *indoor*.

C. Coleta de Dados

Para fazer a avaliação dos trabalhos selecionados no processo de pesquisa nos repositórios, realizamos a extração de algumas informações de forma tabulada e que consideramos comum a todos os artigos. Assim, esta extração auxiliou nas respostas as questões de pesquisa previamente definidas. Em cada trabalho avaliado, buscamos as seguintes informações:

- Repositório;
- Tipo do robô: UAV, UUV(*unmanned underwater vehicles*), UGV (*unmanned ground vehicle*) e humanóide;

- Trabalho teórico ou trabalho com experimento e resultado prático;
- Aplicação prática: este item verifica se a solução apresentada no artigo já está em uso, sendo aplicado em algum problema da sociedade;
- Ano;
- Técnica de mapeamento: verifca-se se o artigo desenvolve ou aplica alguma implementação de modelagem de ambientes ou técnicas de SLAM ou ainda faz alguma extensão de algum trabalho existente;
- Autônomo: verifica se o robô possui a capacidade de transitar de forma autônoma no ambiente ou se algum controle de forma remota é realizado;
- Sensores;
- Ser múltiplo;
- Problema que pretente resolver;
- Característica;
- Implementação Real ou Simulada.

D. Análise dos dados

Dada a coleta dos trabalhos considerando os critérios de inclusão e exclusão definidos, constatamos que alguns os quais possuem o objetivo de trabalhar com problemas relacionados a SLAM *indoor* faziam uso apenas da câmera/sensor para resolução do problema, sem aplicar de forma prática em um robô. Dessa forma, pelo objetivo dos trabalhos ser atender uma demanda da robótica móvel, estes foram considerados aptos a avaliação.

1) Robôs por repositórios: As pesquisas realizadas no repositório *IEEE Xplorer* retornaram um maior número de artigos, seguidos pelos repositórios *ACM Digital Library*. A divisão dos trabalhos no sentido de separação por tipo de robô apresentou uma maior quantidade de trabalhos voltados para UGV (terrestres), 31 artigos, já os trabalhos que focam em apenas UAV são um total de 21 artigos. Salienta-se ainda uma considerável quantidade de trabalhos que possuem apenas a câmera ou sensor em seus experimentos, contando com 15 textos. Os trabalhos que combinam UAV e UGV são 5, apenas 1 trabalho combina UAV, UGV e UUV (aquáticos), assim como 1 trabalho aborda humanóides.

Ao analizarmos o período de publicação, os anos 2012 e 2014 houve uma intensa atividade de pesquisas relacionadas a UAV, tendo seu ápice em 2013. Já os trabalhos voltados para UGV são objetos de pesquisa com maior quantidade de trabalhos em 2006, 2012 e 2014. Poucos trabalhos focaram em SLAM ou mapeamento fazendo uso de uma estrutura híbrida (UAV, UGV e UUV) ou ainda com humanoides foram abordados dentro deste período de tempo. Já os trabalhos em que fazem apenas uso de câmeras ou sensores para os experimentos possuíram uma grande quantidade de publicações entre os anos de 2013 e 2015.

2) Características dos experimentos: A leitura dos artigos selecionados mostrou que os trabalhos, quase em sua totalidade, apresentam experimentos práticos, sejam eles simulados, com robôs reais ou ainda ambos. Apenas o trabalho [9] possui uma abordagem teórica. Do total de trabalhos avaliados, verificou-se que uma ampla maioria apresenta experimentos reais (61), enquantos algums poucos trabalhos apresentam experimentos reais e simulados (8) ou ainda apenas simulados (4).

No que diz respeito a autonomia dos robôs empregados, dos 74 trabalhos, 26 informam que seus controles são autônomos enquanto o controle de 11 são feitos de alguma forma remota. Já os 15 trabalhos que possuem apenas câmera ou sensor não foram considerados como autônomos, já que não fazem emprego de robôs nos experimentos práticos. Apenas o trabalho [10] apresenta uma abordagem semi-autônoma.

3) Técnicas de SLAM e mapeamento abordadas:: Diversos trabalhos fazem uso de técnicas semelhantes de SLAM ou mapeamento em um espaço *indoor*. Algumas dessa técnicas são aplicadas em sua essência, enquanto outras fazem adaptações ou extensões. Há ainda soluções próprias, as quais não baseiam-se em nenhuma abordagem já previamente estabelecida.

Os trabalhos de [11], [12], [13], [14], [15], [16], [17], [18], [19], [20] e [21] apresentam a implementação do VSLAM. Já uma variação do VSLAM chamada PTAM é tratada por [22], [23], [24], [25], [26] e [27]. Os trabalhos que adotam o FastSLAM são [28] e [29]. Trabalhos que tem como base algoritmos de SLAM baseados em filtros de Kalman são apresentados em [19] e [20] (que empregam filtros de Kalman em implementações de monocular SLAM), [30], e em [31], [32], [33] e [34] (os quais fazem uso do filtro de Kalman extendido - EKF). Abordagens que fazem o mapeamento em mapas Octree são [35], [36] e [37], já os trabalhos [38], [39] e [40] são focados em técnica de mapeamento em grids de ocupação. A adoção de soluções baseadas em *Graph* SLAM são feitas em [41], [42], [43] e [44], enquanto a abordagem RO-SLAM é feita em [45] e em [46].

Com a popularização e consequente redução de custos de sensores RGB-D como o Microsoft Kinect, muitas abordagens de RGB-D SLAM foram desenvolvidas, entre as quais citamos [47], [48], [49], [50], [43] e [44], onde estes dois últimos também fazem uso de grafos (*graph-based*). Diversos outros trabalhos fazem uso do sensor Kinect, porém não desenvolvem um algoritmo de RGB-D SLAM, apenas fazem o uso das leituras para desenvolverem as suas técnicas. Um exemplo disso é a solução apresentada em [51], onde busca-se decompor a leitura 3D em dados 2D para alcançar uma representação simples e rápida do espaço. Outra solução que também decompõe a leitura de informações das câmeras/sensores é a de [52], o qual usa a os dados do RGB-D reduzidos de 3D para 2D como se fosse uma leitura laser, porém com maior nível de detalhamento. Considerando ainda a questão de níveis dimensionais, existe grande variações em implementações as quais focam em SLAM ou apenas mapeamento 2D, 2.5D, 3D, 4D e 6D. O trabalho apresentado em [53] apresenta uma solução 2D SLAM. Entre os trabalhos que buscam apresentar uma solução híbrida que combine 2D e 3D SLAM encontram-se [53] onde o UAV faz o mapeamento 3D enquanto o UGV faz o mapeamento 2D, e também os trabalhos de [54], [55] e [56]. Já os trabalhos [57], [58], [58], [10], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71] e [72] apresentam uma abordagem 3D de mapeamento. Em [73] é apresentada uma abordagem 2.5D *mapping*, onde é construido

um mapa de informações com dados geométrico obtidos do sensor LRF e com imagens visuais obtidas da camera stereo. Já no trabalho apresentado em [9], restringe-se os movimentos de *pitch* e *roll* de um UAV para o 4D SLAM, apresentando uma solução que localiza-se entre o 3D SLAM e o 6D SLAM. Uma solução 6D SLAM é apresentada em [74]. O trabalho de [75] apresenta o uso de RGB-D em UAV e LRF para o UGV, enquanto o trabalho de [76] também integra UGV e UAV, porém sendo o UGV responsável pelas leituras RGB-D enquanto o UAV faz um refinamento do mapa produzido pelo UGV através de leituras 2D.

Algumas outras técnicas de SLAM ou modelagem apresentam um foco distinto das demais. É o caso do trabalho de [77], onde é feito um *swarm* de UAVs, onde um deles tem o papel de "sentinela" e faz a subdivisão do espaço a ser explorado. Em [78], é feita a localização baseada em landmarks naturais com sistema de visão independente da pose do UAV. No trabalho de [79], busca-se fazer localização em ambientes conhecidos através de uma rede wireless com nós fixos e móveis, onde cada um deles possui uma identificação única. A solução apresentada em [80] faz a representação local do ambiente de forma métrica, enquanto a representação global é feita de forma topológica, obtendo um SLAM de escala média e grande.

4) Soluções cooperativas ou com múltiplos robôs: O trabalho de [81] procura fazer a modelagem de forma colaborativa através de mapas complementares. Nessa abordagem são utilizados tanto um UAV quanto um UGV. A proposta é que enquanto o UGV faz o mapeamento 2D do espaço, o UAV faz o mapeamento 3D de objetos ortogonais no ambiente, como por exemplo uma mesa. Em [75] também é realizada uma integração entre leituras do UGV feitas com sensores LRF com as leituras do UAV obtidas a partir de sensores RGB-D. Os robôs são reais e não autônomos. Já em [76], o UGV faz o mapeamnto 3D com sensor RGB-D, o UAV faz um refinamento das leituras do UGV com uma leitura 2D do espaço.

Uma abordagem com *swarm* é apresentada em [77], onde o foco do trabalho é o uso de UAVs tanto para caça quanto para limpeza. Aqui, dentro do grupo de vários UAVs, um deles é definido como um sentinela que faz o particionamento do espaço para a exploração. É feito uso de robôs simulados e autônomos.

A implementação de um algoritmo de SLAM PTAM modificado para múltiplos agentes é apresentado em [24]. A exploração do ambiente é feita de forma cooperativa com reconhecimento de pontos de interesse. A definição da exploração é feita a partir de um leilão onde os lances são as distâncias lineares de cada um dos UAVs em relação ao ponto explorado. Vence a menor distância. Os Robôs utilizados são simulados e reais com comportamento autônomo.

O trabalho de [10] é o único que apresenta a aplicação prática da sua solução na sociedade, que é o mapeamento de espaços atingidos por terremotos. Sendo uma implementação que faz uso de UAV e UGV, o funcionamento é semi-autônomo. Isso se dá pela razão do UGV ser controlado remotamente, e quando ele se depara com obstáculos que não pode superar, é o UAV de forma autônoma que faz o mapeamento do espaço. Os robôs são reais.

A utilização de robôs UGV heterogêneos é feita em [55]. Busca-se neste trabalho o mapeamento cooperativo 3D e 2D a partir da navegação autônoma dos agentes. Aqui, cada robô constroi um mapa local e envia os seus dados relevantes para um servidor central onde os dados são associados com as informações já existentes fazendo uso de uma implementação JCBB (*Join-compatibility branch and bound*). Os robôs são reais.

Outro trabalho que também considera o uso de *swarm* é o apresentado em [28]. Ao apresentar um versão do FastSLAM adaptado para múltiplos robôs UGV, faz o mapeamento cooperativo com o uso da técnica SPF (*Stigmergic potential fields*), o qual representa influências comportamentais de informações coletadas do ambiente operacional de um dos agentes. Os robôs são reais.

O artigo apresentado por [13] traz uma abordagem teórica abordando o algoritmos C-VSLAM para UAVs. Na construção do mapa cooperativo, faz uso do EIF (*Extended Information Filter*) para a fusão de dados advindos dos diversos agentes. Os robôs são simulados e não autônomos.

O uso de UAV com UGV para exploração do espaço é aplicado em [25]. O objetivo do trabalho é fazer a exploração heterogênea usando programação inteira. O UGV possui um VSLAM próprio. Para os lugares que ele não consegue explorar, o UAV é acionado fazendo uso do PTAM. Os dados do UAV via PTAM são então associados ao UGV e integrados em um VSLAM. Os Robôs utilizados são simulados e reais com comportamento autônomo.

A implementação apresentada em [35] faz o uso de UAV, UGV e UUV. Aqui os espaços são mapeados em 3D e representados em mapas *Octree*. Robôs são simulados e não autônomos.

Em [26] é apresentado uma versão PTAMM adaptado para múltiplos agentes UAV. O trabalho consiste em fazer a localização e o mapeamento utilizando sensores RGB-D. Uma característica deste trabalho é que ele decompõe o problema do SLAM 3D em um problema monocular SLAM com representação esparsa. Os Robôs utilizados são simulados e reais com comportamento autônomo.

Na implementação de um VSLAM monocular cooperativo apresentado em [20], a construção do SLAM cooperativo baseia-se em landmarks salientes para a representação de características promenientes. Para isso, cada robô realiza seu próprio monocular-SLAM com EKF. O algoritmo de merge usa os landmarks duplicados para dar acurácia ao mapa centralizado. Os robôs são reais e autônomos.

Também empregando múltiplos UGVs autônomos, [64] faz a exploração com times de robôs para aprendizado. Cada robô constrói um mapa 3D parcial, o qual compartilha com outros robôs que estejam em sua faixa de comunicação. Um mapa global é construído a partir do matching das poses e características mútuas encontradas nos mapas individuais.

Por fim, o último trabalho a apresentar uma abordagem cooperativa ou com múltiplos robôs é o de [41]. Aqui, UGVs autônomos buscam fazer um mapeamento 6D do espaço, propondo uma topologia de grafo para desacoplar a integração

das estimativas de incertezas dos filtros locais dos múltiplos robôs em um graph SLAM. Os robôs são reais e autônomos.

Dos 73 trabalhos pesquisados, 14 focam em SLAM ou modelagem cooperativa com múltiplos robôs no ambiente. Destes, 4 tratam exclusivamente UAVs, 5 exclusivamente UGVs, 4 fazem uma abordagem híbrida de UAV e UGV, enquanto apenas um apresenta, ainda que de forma teórica, o uso de UAV, UGV e UUV.

5) Sensores usados: A análise realizada nos artigos selecionados possibilitou fazer um levantamento do uso de sensores em cada solução. Quando trata-se do uso de sensores de uma forma geral, verificamos que os principais utilizados nas soluções apresentadas são respectivamente RGB-D (23), tecnologias laser (22), IMU (19) e câmera monocular (14). Quando o foco é dado apenas em soluções exclusivamente UAV, a ordem de sensores mais utilizados são IMU (14), câmera monocular (5), tecnologias laser (5), sensores barométricos/altitude (4) e RGB-D (2). Ao considerarmos apenas as soluções cooperatívas/múltiplas com emprego de diversos robôs, os sensores mais utilizados são os de tecnologias laser (7), IMU (4), câmera monocular (3) e RGB-D (5). Ao restringirmos o foco em soluções cooperatívas/múltiplas que fazem apenas uso de UAVs, temos entre os sensores mais utilizados o IMU (2), câmera stereo (1), sensor infra-vermelho (1) e sensores RGB-D (1).

Fica claro a popularização do uso de sensores RGB-D em diversos trabalhos na última década. Este sensor é bastante explorado em soluções UGV e também pelos trabalhos que não fizeram uso de um robô móvel de fato mas que apenas fixaram o sensor em um ambiente e fizeram a modelagem. Já a sua aplicação em UAVs demonstra-se possível, porém são poucas as soluções que o aplicam até então. O sensor IMU é um elemento bastante aplicado em soluções para UAVs autônomos, dado que o mesmo fornece medições de movimento do robô a partir de informações inerciais. Já as câmeras monoculares e tecnologias laser são sensores de detecção de pontos que alimentam bem algoritmos de VSLAM, e por isso são sensores bastante empregados em UAVs.

III. Respostas as Questões de Pesquisa

A. QP1: Quais os problemas existentes que necessitam de algoritmos de mapeamento e localização cooperativos indoor?

Considerando os trabalhos aqui analisados que tratam do mapeamento de um espaço *indoor* de forma cooperativa com a atuação de mútiplos robôs, verificamos que os principais problemas existentes estão relacionados ao: tempo de mapeamento, características do espaço e consequente acesso dos robôs, e falhas de comunicação.

O tempo de mapeamento torna-se uma questão fundamental quando trata-se de missões de resgate. Neste sentido, o uso de times de agentes na resolução de tarefas torna sua execução mais eficiente. Além disso, se considerarmos o uso de UAVs, o tempo de autonomia torna-se vital para o sucesso da missão. Ou seja, caso haja energia suficiente para o mapeamento, o uso de um único robô para o mapeamento completo do espaço tomará mais tempo do que o uso de múltiplo robôs, e ainda caso o robô seja um UAV, o mapeamento completo pode não ser finalizado devido sua autonomia de vôo. Assim, o uso de múltiplos agentes no mapeamento rápido do espaço auxilia no planejamento e execução de tarefas estratégicas.

Alguns tipos de robôs ficam limitados por obstáculos. Um exemplo disso são robôs terrestres (UGV) os quais não possuem capacidade de superar determinadas restrições, deixando pontos não mapeados. Abordagens heterogênas tem surgido como uma solução para o mapeamento completo, integrando o uso de UGVs com UAVs.

Se um determinado local a ser mapeado possui uma grande extensão ou determinada característica estrutural, a comunicação pode ser prejudicada. Falhas de comunicação entre o robô e sua estação central podem fazer com que o mesmo se perca em sua missão. O uso de times de robôs traz tolerância a falhas.

B. QP2: Quais as principais técnicas e métodos aplicados na resolução de mapeamento e localização cooperativos indoor para UAV?

Dentro dos trabalhos analisamos, aqueles que apresentam soluções cooperativas para mapeamento de localização focam no uso heterogêneo de UGV e UAV ou apenas UAV.

O trabalho de [81] faz o mapemanto do espaço com o uso de UAVs e UGVs. Enquanto o UAV faz o mapeamento 3D a partir de uma câmera monocular, o UGV faz o mapeamento 2D com o uso de um escaner laser. Já em [10] e em [75], a execução de um SLAM 3D é feito pelo UAV a partir de um sensor RGB-D e pelo UGV com um escaner laser. Já em [76], a distribuição dos sensores é trocada, sendo que o UGV é quem faz as leituras com o sensor RGB-D enquanto o UAV refina o mapa gerado previamente com uma leitura 2D do espaço. Em [25] o UAV implementa um PTAM a partir de leituras de um sonar, enquanto o UGV executa um VSLAM alimentado por sensores RGB-D e laser. Uma abordagem teórica que envolve UAV, UGV e UUV é apresentada em [35], onde a modelagem do ambiente faz uso de mapas Octree.

Já os trabalhos que empregam apenas o uso de UAVs são: [77] o qual faz uso de *swarm* para a distribuição dos espaços a serem explorados pelos UAVs; [24] faz uma modificação do algoritmos PTAM para múltiplos agentes usando câmeras monoculares; em [26] é apresentado uma adaptação do PTAM (PTAMM) com o uso de sensores RGB-D, IMU e IR; em [13] é apresentado uma abordagem de UAVs com câmeras stereo para a execução de um C-SLAM baseado em VSLAM.

Já soluções que implementam o mapeamento cooperativo de espaços indoor usando apenas UGVs são [55], [28], [20], [64] e [41]. Em [55] é feito o uso de robôs heterogêneos no mapeamento 3D e 2D do espaço com o uso de escaners laser. Em [28] é feita uma adaptação do algoritmo de FastSLAM para múltiplos agentes também utilizando escaner laser. No trabalho de [20] os UGVs executam um VSLAM a partir de câmera monocular. Em [64] é feito o uso de laser e webcam para a modelagem do espaço. O trabalho de [41] apresentam uma implementação de múltiplo GraphSLAM, fazendo uso de câmera stereo.

C. QP3: Quais característica de cada solução pesquisada?

O trabalho de [81] apresenta a modelagem colaborativa com mapas complementares, onde o UGV faz o mapeamento 2D do espaço e o UAV faz o mapeamento 3D de objetos ortogonais no ambiente. De forma semelhante, em [75] é feito uso de um RGB-D SLAM para o UAV integrado ao LRF SLAM através de um algoritmo de ICP (*Iterative Closest Point*), o qual o alimenta de forma a trazer maior precisão para o mapa do UGV. Em [76] este conceito é invertido, onde o UGV faz a leitura 3D enquanto o UAV a refina com uma leitura 2D.

O trabalho de [77] faz uso de UAVs em *swarm* para tarefas de caça/limpeza. Na distribuição dos espaços a serem explorados, um UAV sentinela faz o particionamento do espaço.

Em [24] a exploração cooperativa é feita baseada em um leilão para escolha do agente que irá explorar determinado ponto. Os vencedores deste leilão são aqueles que estão em uma menor distância linear daquele ponto.

O trabalho de [10] é aplicado na exploração de espaços atingidos por terremotos. Um UGV é remotamente comandado, e quando há obstáculos que não pode superar, um UAV que o mesmo carrega é acionado de forma que faça o mapeamento autônomo, tendo como ponto de origem o local do UGV. O refinamento dos mapas gerados pelo UAV e UGV é feito utilizando o algoritmo 3D ICP (*Interactive closest point*).

O uso de robôs terrestres heterogêneos na construção rápida de mapas em [55] adota que cada robô constroi um mapa local e envia os dados relevantes para um servidor central, onde os dados são associados com as informações já existentes através de uma implementação de JCBB (*Join-compatibility branch and bound*).

O trabalho de [28] foca no uso da técnica SPF (*Stigmergic potential fiels*) para representar influências comportamentais de informações coletadas do ambiente operacional em que se encontra um agente da *swarm*.

Em [13] faz uso do filtro EIF (*Extended Information Filter*) para a fusão dos dados vindos dos diversos agentes.

No trabalho de [25], busca-se fazer a exploração heterogênea usando como técnica a programação inteira. Neste trabalho, o UGV possui um SLAM próprio, e os locais que ele não consegue explorar são explorados pelo UAV que executa um PTAM. Os dados do UAV via PTAM são associados ao UGV e integrados em um VSLAM. A comunicação é feita constantemente entre pequenos times organizados.

A proposta de [35] é focar no consumo de memória a partir de mapas *Octree*, produzindo mapas maiores em menor tempo.

Em [26] é feito uso do sensor RGB-D, onde decompõe-se o problema do SLAM 3D em um monocular SLAM com represetação esparsa.

O trabalho de [20] foca no uso de landmarks salientes para representação de características promenientes. Cada robô realiza seu próprio monocular SLAM com EKF. O algoritmo de *merge* usa os *landmarks* duplicados para dar acurácia ao mapa centralizado.

Na abordagem de [64] é feita a exploração da seguinte forma: cada robô constroi um mapa 3D parcial o qual compartilha com outros robôs que estejam em sua faixa de comunicação. Um mapa global é contruído a partir do *matching* das poses e características mútuas encontradas no mapas individuais.

Já o trabalho de [41] busca fazer um mapeamento 6D com múltiplos robôs, onde propõe-se uma topologia de grafo para desacoplar a integração das estimativas de incertezas dos filtros locais dos múltiplos robôs em um GraphSLAM.

D. QP4: As soluções apresentam implementação real?

Entre os trabalhos que abordam soluções cooperativas na modelagem de espaços *indoor*, 7 deles apresentaram experimentos reais ([81], [10], [55], [28], [20], [64], [41], [75] e [76]), 3 apresentaram experimentos reias e também simulados ([24], [25] e [26]) enquanto 3 apresentaram experimentos simulados ([77], [13] e [35]).

E. QP5: Quais problemas/desafios/limitações são encontrados no desenvolvimento de algoritmos de mapeamento e localização cooperativos indoor para UAV?

De forma sucinta, o processamento e custos de integração dos mapas dos múltiplos agentes, controle do comportamento dos agentes e representação do espaço e de características do ambientes são desafios ainda.

A integração dos mapas deve considerar a associação dos dados capturados pelos múltiplos agentes [24][13][35][26][41]. O problema do *merge* dos mapas pode ser tratado em duas abordagens: com cada veículo sabendo sua posição inicial no sentido de haver conhecimento para estabelecer uma correlação entre os mapas gerados; e no tratamento de mapas dispersos usando características ou determinando probalilidades de sobreposição (*overlap*) ou fusão dos mapas usando filtros de partículas [28] ou outras técnicas. A detecção de características do ambiente com identificação de objetos com uso de *landmarks* (naturais ou não) possibilitam uma maior extração de informação do espaço a ser mapeado. Focos destes estudos podem ser aprofundados em extração de pontos de interesse complexos [24], landmarks de alto-nível [55], bem como outras formas de características [64]. Ao considerar ainda que todo esse trabalho de mapeamento e fusão/integração de mapas de múltiplos agentes estão sustentados no uso de UAVs, as restrições de recusos energéticos são obstáculos a serem considerados no desenvolvimento de técnicas.

Assim, este processamento de dados advindos de múltiplos agentes confrontados com a necessidade de respostas rápidas para UAVs em ambientes hostís ainda é espaço para pesquisas aprofundadas. Além disso, quando trata-se da representação dos mapas, o problema de inserír níveis de escala pode-se ter um olhar mais dedicado [81]. A representação do ambiente a ser modelado traz algumas restrições, em especial quando considera-se a modelagem 3D do espaço: a modelagem de grades de ocupação 3D não fazem uma modelagem fina do ambiente e é necessário limitar o espaço a ser modelado ou conhecer previamente as dimensões a serem exploradas; núvens de pontos 3D evitam a discretização que a abordagem de grades de ocupação provocam, entretanto além do consumo de memória, não possibilita modelar o espaço livre ou desconhecido, o que para tarefas de resgate pode ser uma

informação importante; já a abordagem de mapas de elevação 2.5D apresentam um bom consumo de memória, entretanto não é possível fazer uma representação volumétrica [35]. Uma proposta de computação de mapeamento em núvens é apresentado em [82].

Outro ponto de importância na modelagem 3D de ambiente é na representação do espaço de ambientes desestruturados. Alguma soluções abordam a identificação de linhas verticais e horizontais [64], mas em espaços como os de desastres, obstáculos de diversos formatos podem impedir a movimentação do robô [54].

No que tange a comunicação em propostas multi agentes, soluções com comunicação centralizada são mais comuns [28], mas podem sofrer com latência e perda de dados. Assim, é necessário estudos sobre como os agentes podem se comunicar com maior eficácia e eficiência [77][24][10][25][41]. Soluções descentralizadas [41] podem auxiliar nesse sentido. A questão então torna-se como fazer esta distribuição do controle de comunicação entre os múltiplos agentes. Abordagens que implementem *swarm* [77][28] podem prover meios de coodenar o comportamento individual dos robôs para alcançar sinergia no processo de exploração.

IV. Conclusão

Os diversos repositórios pesquisados mostraram trabalhos voltados também para UAVs e mapeamento cooperativo na última década. O emprego de múltiplos agentes neste processo de mapeamente demostrou que determinados sensores como o RGB-D tornam-se mais populares a medida que o seu custo diminui, e que diversos experimentos o usam como forma de validar suas propostas. Técnas de SLAM que adotam dados advindos destes sensores e também de sensores como câmera monocular ou stereo receberam diversas adptações e extensões, como por exemplo o PTAM. Apesar da grande quantidade de trabalhos, ainda são apontados como desafios superar o mapeamento de ambientes *indoor* dadas as restrições de processamento em tempo satisfatório e consumo de memória, a identificação de características e restrições do espaço a serem mapeados bem como a forma de gerenciamento da comunicação entre os agentes que compões um time de exploração e o mapeamento de forma cooperativa, e se considerarmos como agente um UAV, a sua autonomia torna-se fator de extrema importância no sucesso de um mapeamento completo do ambiente.

Referências

[1] S. Thrun, *Robotic Mapping: A Survey*, ser. Research paper. School of Computer Science, Carnegie Mellon University, 2002.

[2] S. Saeedi, M. Trentini, M. Seto, and H. Li, "Multiple-robot simultaneous localization and mapping: A review," *Journal of Field Robotics*, vol. 33, no. 1, pp. 3–46, 2016. [Online]. Available: http://dx.doi.org/10.1002/rob.21620

[3] N. K. Dhiman, D. Deodhare, and D. Khemani, "Where am i? creating spatial awareness in unmanned ground robots using slam: A survey," *Sadhana*, vol. 40, no. 5, pp. 1385–1433, 2015.

[4] C. Stachniss, "Coordinated multi-robot exploration," in *Robotic Mapping and Exploration*. Springer, 2009, pp. 43–71.

[5] S. Thrun and J. J. Leonard, "Simultaneous localization and mapping," in *Springer handbook of robotics*. Springer, 2008, pp. 871–889.

[6] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele University*, vol. 33, no. 2004, pp. 1–26, 2004.

[7] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering–a systematic literature review," *Information and software technology*, vol. 51, no. 1, pp. 7–15, 2009.

[8] S. Saeedi, M. Trentini, M. Seto, and H. Li, "Multiple-robot simultaneous localization and mapping: A review," *Journal of Field Robotics*, vol. 33, no. 1, pp. 3–46, 2016.

[9] A. Aditya, "Implementation of a 4d fast slam including volumetric sum of the uav," in *Sensing Technology (ICST), 2012 Sixth International Conference on*, Dec 2012, pp. 78–84.

[10] N. Michael, S. Shen, K. Mohta, Y. Mulgaonkar, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida *et al.*, "Collaborative mapping of an earthquake-damaged building via ground and aerial robots," *Journal of Field Robotics*, vol. 29, no. 5, pp. 832–841, 2012.

[11] V. Ghadiok, J. Goldin, and W. Ren, "On the design and development of attitude stabilization, vision-based navigation, and aerial gripping for a low-cost quadrotor," *Autonomous Robots*, vol. 33, no. 1, pp. 41–68, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10514-012-9286-z

[12] E. Hernandez, J. M. Ibarra, J. Neira, R. Cisneros, and J. E. Lavin, "Visual slam with oriented landmarks and partial odometry," in *Electrical Communications and Computers (CONIELECOMP), 2011 21st International Conference on*, Feb 2011, pp. 39–45.

[13] X. Li and N. Aouf, "Cooperative vslam based on uav application," in *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*. IEEE, 2012, pp. 914–919.

[14] R. Huang, P. Tan, and B. M. Chen, "Monocular vision-based autonomous navigation system on a toy quadcopter in unknown environments," in *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*, June 2015, pp. 1260–1269.

[15] L. Teixeira, A. B. Raposo, and M. Gattass, "Indoor localization using slam in parallel with a natural marker detector," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, ser. SAC '13. New York, NY, USA: ACM, 2013, pp. 27–33. [Online]. Available: http://doi.acm.org/10.1145/2480362.2480370

[16] D. M. A. Latif, M. A. M. Salem, H. Ramadan, and M. I. Roushdy, "Comparison of 3d feature registration techniques for indoor mapping," in *Computer Engineering Systems (ICCES), 2013 8th International Conference on*, Nov 2013, pp. 239–244.

[17] C. Fu, M. A. Olivares-Mendez, R. Suarez-Fernandez, and P. Campoy, "Monocular visual-inertial slam-based collision avoidance strategy for fail-safe uav using fuzzy logic controllers," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1, pp. 513–533, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10846-013-9918-3

[18] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-slam–based navigation for autonomous micro helicopters in gps-denied environments," *Journal of Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011.

[19] N. Zhang, M. Li, and B. Hong, "Simultaneous localization and mapping using invariant natural features," in *2006 IEEE International Conference on Robotics and Biomimetics*, Dec 2006, pp. 1682–1687.

[20] M. Wu, F. Huang, L. Wang, and J. Sun, "Cooperative multi-robot monocular-slam using salient landmarks," in *2009 International Asia Conference on Informatics in Control, Automation and Robotics*, Feb 2009, pp. 151–155.

[21] G. Zhang and I. H. Suh, "Building a partial 3d line-based map using a monocular slam," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 1497–1502.

[22] S. Upadhyay, A. Dewan, A. K. Singh, and M. Krishna, "Trajectory planning for monocular slam based exploration system," in *Proceedings of the 2015 Conference on Advances In Robotics*, ser. AIR '15. New York, NY, USA: ACM, 2015, pp. 27:1–27:6. [Online]. Available: http://doi.acm.org/10.1145/2783449.2783476

[23] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. von Stryk, *Comprehensive Simulation of Quadrotor UAVs Using ROS and Gazebo*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 400–411.

[24] R. Williams, B. Konev, and F. Coenen, *Multi-agent Environment Exploration with AR.Drones*. Cham: Springer International Publishing, 2014, pp. 60–71.

[25] A. Dewan, A. Mahendran, N. Soni, and K. M. Krishna, "Heterogeneous ugv-mav exploration using integer programming," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 5742–5749.

[26] G. Loianno, J. Thomas, and V. Kumar, "Cooperative localization and mapping of mavs using rgb-d sensors," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4021–4028.

[27] A. Concha and J. Civera, "Using superpixels in monocular slam," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 365–372.

[28] J. Stipes, R. Hawthorne, D. Scheidt, and D. Pacifico, "Cooperative localization and mapping," in *2006 IEEE International Conference on Networking, Sensing and Control*. IEEE, 2006, pp. 596–601.

[29] B. X. Hon, H. Tian, F. Wang, B. M. Chen, and T. H. Lee, "A customized fastslam algorithm using scanning laser range finder in structured indoor environments," in *2013 10th IEEE International Conference on Control and Automation (ICCA)*, June 2013, pp. 640–645.

[30] R. Li, J. Liu, L. Zhang, and Y. Hang, "Lidar/mems imu integrated navigation (slam) method for a small uav in indoor environments," in *2014 DGON Inertial Sensors and Systems (ISS)*, Sept 2014, pp. 1–15.

[31] P. Jensfelt, D. Kragic, J. Folkesson, and M. Bjorkman, "A framework for vision based bearing only 3d slam," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, May 2006, pp. 1944–1950.

[32] H. Lim and Y. S. Lee, "Real-time single camera slam using fiducial markers," in *ICCAS-SICE, 2009*, Aug 2009, pp. 177–182.

[33] S. Hochdorfer and C. Schlegel, "6 dof slam using a tof camera: The challenge of a continuously growing number of landmarks," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, Oct 2010, pp. 3981–3986.

[34] Q. Zeng, Y. Wang, J. Liu, R. Chen, and X. Deng, "Integrating monocular vision and laser point for indoor uav slam," in *Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS), 2014*, Nov 2014, pp. 170–179.

[35] J. Jessup, S. Givigi, and A. Beaulieu, "Merging of octree based 3d occupancy grid maps," in *Systems Conference (SysCon), 2014 8th Annual IEEE*. IEEE, 2014, pp. 371–377.

[36] J. Fossel, D. Hennes, D. Claes, S. Alers, and K. Tuyls, "Octoslam: A 3d mapping approach to situational awareness of unmanned aerial vehicles," in *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, May 2013, pp. 179–188.

[37] Y. k. Wang, J. Huo, and X. s. Wang, "A real-time robotic indoor 3d mapping system using duel 2d laser range finders," in *Control Conference (CCC), 2014 33rd Chinese*, July 2014, pp. 8542–8546.

[38] N. Zhang, M. Li, and B. Hong, "Active mobile robot simultaneous localization and mapping," in *2006 IEEE International Conference on Robotics and Biomimetics*, Dec 2006, pp. 1676–1681.

[39] S. Shaker, D. Asmar, and I. H. Elhajj, "3d reconstruction of indoor scenes by casting visual rays in an occupancy grid," in *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on*, Dec 2010, pp. 1176–1182.

[40] R. Zask and M. N. Dailey, "Rapid 3d visualization of indoor scenes using 3d occupancy grid isosurfaces," in *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2009. ECTI-CON 2009. 6th International Conference on*, vol. 02, May 2009, pp. 672–675.

[41] M. J. Schuster, C. Brand, H. Hirschmuller, M. Suppa, and M. Beetz, "Multi-robot 6d graph slam connecting decoupled local reference filters," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, Sept 2015, pp. 5093–5100.

[42] H. Y. Chen and C. Y. Lin, "Rgb-d sensor based real-time 6dof-slam," in *Advanced Robotics and Intelligent Systems (ARIS), 2014 International Conference on*, June 2014, pp. 61–65.

[43] H. P. Quang and N. L. Quoc, "Some improvements in the rgb-d slam system," in *Computing Communication Technologies - Research, Innovation, and Vision for the Future (RIVF), 2015 IEEE RIVF International Conference on*, Jan 2015, pp. 112–116.

[44] H. E. Elghor, D. Roussel, F. Ababsa, and E. H. Bouyakhf, "Planes detection for robust localization and mapping in rgb-d slam systems," in *3D Vision (3DV), 2015 International Conference on*, Oct 2015, pp. 452–459.

[45] F. R. Fabresse, F. Caballero, I. Maza, and A. Ollero, "Robust range-only slam for unmanned aerial systems," *Journal of Intelligent & Robotic Systems*, pp. 1–14, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10846-015-0322-z

[46] A. Torres-González, J. Martinez-de Dios, A. Jiménez-Cano, and A. Ollero, "An efficient fast-mapping slam method for uas applications using only range measurements," *Unmanned Systems*, vol. 4, no. 02, pp. 155–165, 2016.

[47] G. Hu, S. Huang, L. Zhao, A. Alempijevic, and G. Dissanayake, "A robust rgb-d slam algorithm," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 1714–1719.

[48] C. Lin, B. He, and J. Zhang, "Study on indoor robot of self-localization based on rgb-d sensor," in *Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on*, Dec 2014, pp. 2619–2624.

[49] T. Emter and A. Stein, "Simultaneous localization and mapping with the kinect sensor," in *Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on*, May 2012, pp. 1–6.

[50] H. Dong, N. Figueroa, and A. E. Saddik, "Towards consistent reconstructions of indoor spaces based on 6d rgb-d odometry and kinectfusion," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 1796–1803.

[51] H. C. Roh, C. H. Sung, and M. J. Chung, "Rapid slam using simple map representation in indoor environment," in *Frontiers of Computer Vision, (FCV), 2013 19th Korea-Japan Joint Workshop on*, Jan 2013, pp. 225–229.

[52] M. F. A. Ghani, K. S. M. Sahari, and L. C. Kiong, "Improvement of the 2d slam system using kinect sensor for indoor mapping," in *Soft Computing and Intelligent Systems (SCIS), 2014 Joint 7th International Conference on and Advanced Intelligent Systems (ISIS), 15th International Symposium on*, Dec 2014, pp. 776–781.

[53] M. Leichtfried, C. Kaltenriner, A. Mossel, and H. Kaufmann, "Autonomous flight using a smartphone as on-board processing unit in gps-denied environments," in *Proceedings of International Conference on Advances in Mobile Computing & Multimedia*, ser. MoMM '13. New York, NY, USA: ACM, 2013, pp. 341:341–341:350. [Online]. Available: http://doi.acm.org/10.1145/2536853.2536898

[54] A. Oliver, S. Kang, B. C. Wünsche, and B. MacDonald, "Using the kinect as a navigation sensor for mobile robotics," in *Proceedings of the 27th Conference on Image and Vision Computing New Zealand*, ser. IVCNZ '12. New York, NY, USA: ACM, 2012, pp. 509–514. [Online]. Available: http://doi.acm.org/10.1145/2425836.2425932

[55] J. G. Rogers, D. Baran, E. Stump, S. Young, and H. I. Christensen, "Cooperative 3d and 2d mapping with heterogenous ground robots," in *SPIE Defense, Security, and Sensing*. International Society for Optics and Photonics, 2012, pp. 838 708–838 708.

[56] I. Dryanovski, W. Morris, and J. Xiao, "An open-source pose estimation system for micro-air vehicles," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 4449–4454.

[57] Q. Li, D.-C. Li, Q.-F. Wu, L.-W. Tang, Y. Huo, Y.-X. Zhang, and N. Cheng, "Autonomous navigation and environment modeling for mavs in 3-d enclosed industrial environments," *Comput. Ind.*, vol. 64, no. 9, pp. 1161–1177, Dec. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.compind.2013.06.010

[58] D. Yun, H. Chang, and T. V. Lakshman, "Accelerating vision-based 3d indoor localization by distributing image processing over space and time," in *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*, ser. VRST '14. New York, NY, USA: ACM, 2014, pp. 77–86. [Online]. Available: http://doi.acm.org/10.1145/2671015.2671018

[59] P. Kohlhepp, G. Bretthauer, M. Walther, and R. Dillmann, "Using orthogonal surface directions for autonomous 3d-exploration of indoor environments," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2006, pp. 3086–3092.

[60] L. Iocchi, S. Pellegrini, and G. D. Tipaldi, "Building multi-level planar maps integrating lrf, stereo vision and imu sensors," in *2007 IEEE International Workshop on Safety, Security and Rescue Robotics*, Sept 2007, pp. 1–6.

[61] V. Nguyen, A. Harati, and R. Siegwart, "A lightweight slam algorithm using orthogonal planes for indoor mobile robotics," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2007, pp. 658–663.

[62] A. Zureiki and M. Devy, "Appearance-based data association for 3d and multisensory slam in structured environment," in *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on*, April 2008, pp. 1–6.

[63] G. Michalicek, D. Klimentjew, and J. Zhang, "A 3d simultaneous localization and mapping exploration system," in *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, Dec 2011, pp. 1059–1065.

[64] S. R. u. N. Jafri and R. Chellali, "A distributed multi robot slam system for environment learning," in *Robotic Intelligence In Informationally Structured Space (RiiSS), 2013 IEEE Workshop on*, April 2013, pp. 82–88.

[65] K. Wang, S. Jia, B. Guo, and Y. Li, "Mobile robot 3d map building based on rtm," in *Information and Automation (ICIA), 2013 IEEE International Conference on*, Aug 2013, pp. 1224–1229.

[66] M. Tomono, "Robust 3d slam with a stereo camera based on an edge-point icp algorithm," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, May 2009, pp. 4306–4311.

[67] J. Ryde and J. J. Corso, "Fast voxel maps with counting bloom filters," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 4413–4418.

[68] P. Manojkumar and G. R. M. Reddy, "Parallel implementation of 3d modelling of indoor environment using microsoft kinect sensor," in *Computing, Communications and Networking Technologies (ICCCNT),2013 Fourth International Conference on*, July 2013, pp. 1–6.

[69] P. C. Su, J. Shen, and S. c. S. Cheung, "A robust rgb-d slam system for 3d environment with planar surfaces," in *2013 IEEE International Conference on Image Processing*, Sept 2013, pp. 275–279.

[70] E. Ataer-Cansizoglu, Y. Taguchi, S. Ramalingam, and T. Garaas, "Tracking an rgb-d camera using points and planes," in *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on*, Dec 2013, pp. 51–58.

[71] Z. Zhao and X. Chen, "Semantic mapping for object category and structural class," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 724–729.

[72] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012.

[73] R. C. Luo and C. C. Lai, "Concurrent indoor map construction and patterns of interests recognition using sensory fusion approach for service robotics," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, May 2012, pp. 2243–2248.

[74] D. Osmankovi and J. Velagi, "Reconstructing the 3d thermal model of indoor environment from unorganized data set acquired by 3d laser scans and thermal imaging camera," in *2012 IEEE International Symposium on Intelligent Control*, Oct 2012, pp. 13–18.

[75] J. H. Shim and Y. I. Cho, "A visual localization technique for unmanned ground and aerial robots," in *2017 First IEEE International Conference on Robotic Computing (IRC)*, April 2017, pp. 399–403.

[76] H. Qin, Z. Meng, W. Meng, X. Chen, H. Sun, F. Lin, and M. H. Ang, "Autonomous exploration and mapping system using heterogeneous uavs and ugvs in gps-denied environments," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1339–1350, Feb 2019.

[77] R. R. McCune and G. R. Madey, "Agent-based simulation of cooperative hunting with uavs," in *Proceedings of the Agent-Directed Simulation Symposium*, ser. ADSS 13. San Diego, CA, USA: Society for Computer Simulation International, 2013, pp. 8:1–8:6. [Online]. Available: http://dl.acm.org/citation.cfm?id=2499592.2499600

[78] J.-O. Lee, T. Kang, K.-H. Lee, S. K. Im, and J. Park, "Vision-based indoor localization for unmanned aerial vehicles," *Journal of Aerospace Engineering*, vol. 24, no. 3, pp. 373–377, 2010.

[79] L. Yut, Q. Fei, and Q. Geng, "Combining zigbee and inertial sensors for quadrotor uav indoor localization," in *2013 10th IEEE International Conference on Control and Automation (ICCA)*. IEEE, 2013, pp. 1912–1916.

[80] V. Pradeep, G. Medioni, and J. Weiland, "Visual loop closing using multi-resolution sift grids in metric-topological slam," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, June 2009, pp. 1438–1445.

[81] A. Mahendran, A. Dewan, N. Soni, and K. M. Krishna, "Ugv-mav collaboration for augmented 2d maps," in *Proceedings of Conference on Advances In Robotics*, ser. AIR '13. New York, NY, USA: ACM, 2013, pp. 38:1–38:6. [Online]. Available: http://doi.acm.org/10.1145/2506095.2506116

[82] P. Zhang, H. Wang, B. Ding, and S. Shang, "Cloud-based framework for scalable and real-time multi-robot slam," in *2018 IEEE International Conference on Web Services (ICWS)*, July 2018, pp. 147–154.

Ricardo da Rosa earned a bachelor's degree in Informatics from the Western Parana State University (Unioeste) in 2006, master's degree in Computer Science from the State University of Campinas (Unicamp) in 2010 and is currently doctorate in Electrical Engineering and Industrial Informatics by the Technological Federal University of Paraná (UTFPR). He is also a teacher in the field of computation in the Federal Institute of Paraná Campus Cascavel (IFPR). His research interests are mobile robotics, software engineering, educational robotics and aerial robots.

Marco Aurélio Wehrmeister Marco Aurélio Wehrmeister earned his doctor's degree in Computer Science from the Federal University of Rio Grande do Sul (Brazil) and from the University of Paderborn (Germany) in 2009 (joint degree). In 2009, he worked as professor and post- doctorate researcher for the Federal University of Santa Catarina (Brazil). From 2010 to 2013, he worked as professor of the Computer Science Department of University of the State of Santa Catarina (UDESC, Brazil). Since 2013 he has worked as professor in the Informatics Department in the Technological Federal University of Paraná (UTFPR, Brazil). From 2014 to 2016 he was coordinator of the Master's in Applied Computation program of UTFPR. In 2015 he was a visiting professor of the School of Electronic, Electrical and Systems Engineering of the University of Birmingham (United Kingdom). He had his thesis selected by the Brazilian Computer Society as one of the six best theses in Computer Science in 2009. He is a member of the special commission in computational systems engineering of the Brazilian Computer Society. Since 2015 he is a member of the IFIP Working Group 10.2 in Embedded Systems. His research interests are in the fields of embedded and real time systems, aerial robots, model driven engineering, and hardware/software engineering for embedded systems and robotics. He co-authored more than 70 articles in journals and conferences reviewed by peers. He has worked in various research programs funded by Brazilian R&D agencies.

APÊNDICE D – ARTIGO PUBLICADO

Artigo publicado com o título “Honeycomb Map: A Bioinspired Topological Map for Indoor Search and Rescue Unmanned Aerial Vehicles” (ROSA et al., 2020).

sensors

MDPI

Article

Honeycomb Map: A Bioinspired Topological Map for Indoor Search and Rescue Unmanned Aerial Vehicles

Ricardo da Rosa [1,2,*,†], Marco Aurelio Wehrmeister [2,†], Thadeu Brito [3,†], José Luís Lima [3,4,†] and Ana Isabel Pinheiro Nunes Pereira [3,†]

1 Federal Institute of Education, Science and Technology—Parana (IFPR), 85814-800 Campus Cascavel, Brazil
2 Campus Curitiba, Federal University of Technology—Parana (UTFPR), 80230-901 Curitiba, Brazil; wehrmeister@utfpr.edu.br
3 Campus de Santa Apolónia, Instituto Politécnico de Bragança (IPB), Research Centre in Digitalization and Intelligent Robotics (CeDRI), 5300-253 Bragança, Portugal; brito@ipb.pt (T.B.); jllima@ipb.pt (J.L.L.); apereira@ipb.pt (A.I.P.N.P.)
4 INESC TEC - INESC Technology and Science, 4200-465 Porto, Portugal
* Correspondence: ricardo.rosa@ifpr.edu.br; Tel.: +55-45-99141-8255
† These authors contributed equally to this work.

Received: 16 January 2020; Accepted: 4 February 2020; Published: 8 February 2020

Abstract: The use of robots to map disaster-stricken environments can prevent rescuers from being harmed when exploring an unknown space. In addition, mapping a multi-robot environment can help these teams plan their actions with prior knowledge. The present work proposes the use of multiple unmanned aerial vehicles (UAVs) in the construction of a topological map inspired by the way that bees build their hives. A UAV can map a honeycomb only if it is adjacent to a known one. Different metrics to choose the honeycomb to be explored were applied. At the same time, as UAVs scan honeycomb adjacencies, RGB-D and thermal sensors capture other data types, and then generate a 3D view of the space and images of spaces where there may be fire spots, respectively. Simulations in different environments showed that the choice of metric and variation in the number of UAVs influence the number of performed displacements in the environment, consequently affecting exploration time and energy use.

Keywords: multi-robot; UAV; bioinspired map; topologic mapping; map exploration

1. Introduction

Mobile robotics is being applied more often to not only solve problems found in industrial environments, but also applied to services and home uses. For example, robots can be used in the process of warehouse automation, space monitoring, and house cleaning. These new applications show that a mobile robot can perform complex tasks while navigating unknown environments and avoiding unexpected obstacles by reacting to environmental stimuli [1]. Another application of mobile robotics is in the support of rescue teams in natural-disaster or catastrophe situations. Exploration might put the life of rescue-team professionals in danger. The use of Unmanned Aerial Vehicles (UAV) may assist rescue activities, especially in indoor areas where the arrival or movement of a ground robot is sometimes impossible. Access to unknown indoor areas requires techniques for defining the space where a robot is positioned, generating environmental mappings in order to aid teams in the reconnaissance of these areas where the use of global positioning systems (GPS) is unavailable. Thus, an autonomous robot must deal with two critical problems to survive and navigate in its environment: mapping the environment, and searching for its own location in the map [2].

For rescue environments, the time for space recognition becomes critical. Thus, the use of multiple robots can reduce environment exploration time. The collective construction of a map that is used to

displace both multiple robots and the rescue team must represent spaces where it is possible to move and points that need more attention, such as human-temperature recognition, toxic elements, fires, and other factors that could be life-threatening.

This work proposes a mapping approach that was bioinspired by honeycomb construction. Honeybees use hexagonal-pattern cylinders to progressively build a complex structure by adding wax produced and manipulated by several bees [3]. This hexagonal structure allows the construction of combs with less wax (material saving), with the capacity for more storage. The construction of a honeycomb structure starts from a cell floor. Then, the structure is progressively extended in depth by adding more materials around the cell walls. The hive combs are the result of the collective work of hundreds of bees. There is no central commander/master for the building process. The individuals follow simple rules related to environmental construction (e.g., only one bee at a time can build a particular comb, and a new cell must be adjacent to an existing cell), so that this environment influences behavior, which, in turn, transforms the environment, it being a mechanism of synergy [3].

The scope of this work is in the application of simulated models of UAVs with similar configuration, and in addition, it will make use of simulation environments to validate the developed method. In this way, details and restrictions of communication technologies are abstracted.

2. Related Works

2.1. Map Generation

Building an environment map is necessary for both robot exploration and in simultaneous localization and mapping (SLAM) tasks. In [4], map generation was partitioned into three parts: metric, topological, and hybrid maps. Cartographic maps are able to make use of Vector map ([5–7]); however, they are not the focus of this work.

2.1.1. Metric Maps

Metric maps try to extract the features and geometric properties of the environment, and they are represented as a grid, geometric, or feature map [8] . Often, metric maps are probabilistic [4], and establish methods for modeling noise and its effects on environmental modeling. The approaches are based on a Bayesian filter, graph-based SLAM, and submap-joining SLAM.

2.1.2. Topological Maps

Topological maps represent the environment in graphs, where nodes represent places and objects of interest, and edges represent the spatial relationship or path between nodes [4]. In addition to providing a more compact representation of the environment than metric maps, topological maps provide a higher-level symbolic understanding for planning and navigation tasks. While metric maps are achieved with odometry-error accumulation, topological maps are built without the worry of metric aspects. Odometry errors that are accumulated between graph nodes do not necessarily accumulate through the global map.

2.1.3. Hybrid Maps

Hybrid maps combine the advantages of metric and topological mapping. Topological mapping is applied for a global view of the environment, while metric mapping is applied to smaller areas, which reduces computational complexity during metric-information processing. A hybrid-map form is the use of each topological-map node to represent a small metric map, and edges between nodes represent the path from the center point of one metric map to the center point of the next metric map [4].

2.2. Multiple Robots in Environment Mapping

Solutions that use multiple robots are characterized by the application of homogeneous and heterogeneous robots. Many related works make use of SLAM algorithms, but the focus of this work is environment exploration. Thus, works that make use of SLAM were considered for understanding the way they build the maps.

In [9], the authors performed collaborative space mapping with UAV and Unmanned Ground Vehicle (UGV) modeling through complementary maps. While the UGV does 2D area mapping, the UAV does 3D mapping of orthogonal objects in the environment. In [10], the authors presented a practical application, which is the mapping of areas struck by earthquakes. This being an implementation that uses a UAV and UGV, operation is semiautonomous. That happens because the UGV is remotely controlled, but when it faces obstacles it cannot overcome, the UAV autonomously does the mapping of the area. The execution of a 3D SLAM is done by the UAV via an RGB-D sensor, and by the UGV with a laser scanner. In [11], the UAV implements a Parallel Tracking and Mapping (PTAM)on the basis of sonar readings, while the UGV executes a Visual SLAM (VSLAM) fed by RGB-D and laser sensors. The work's goal was heterogeneous exploration using integer programming. The UGV has its own VSLAM and, for places that it cannot explore, the UAV is put in action using PTAM. UAV data via PTAM are then sent to the UGV and integrated in a VSLAM.

Some works that only use UAVs are presented: [12] uses a swarm to distribute areas to be explored by the UAVs. The focus is the use of UAVs for both hunting and cleaning. Here, in a group of many UAVs, one is defined as a sentinel and partitions the area for exploration. The work of [13] modified the PTAM algorithm for multiple agents using monocular cameras. Environment exploration is done cooperatively with recognition of points of interest. The definition of exploration is done via auction, where each bid is the linear distance of each UAV to the point being explored. The shortest distance wins the auction. In [14], an adaptation of PTAM (Parallel Tracking and Multiple Mapping—PTAMM) with the use of RGB-D, inertial measurement unit (IMU), and infrared (IR) sensors was presented. The work did localization and mapping using RGB-D sensors. A characteristic of this work is that it decomposed a 3D SLAM problem in a monocular SLAM with sparse representation.

There are solutions that implemented cooperative indoor mapping by using only UGVs [15–19]. In [15], heterogeneous robots were used in 2D and 3D area mapping using laser scanners, performing 3D and 2D cooperative mapping via autonomous agent navigation. Here, each robot builds a local map and sends the relevant data to a central server, where the data are joined with existing data using join-compatibility branch and bound (JCBB) implementation. In [16], the authors adapted the FastSLAM algorithm for multiple agents by also using laser scanners. Presenting a version of FastSLAM adapted to multiple UGV robots, it could perform cooperative mapping with the stigmergic potential field (SPF) technique, which represents behavioral influences of gathered data from the operational environment of one of the agents. In [17], the UGVs executed a VSLAM via a monocular camera. The creation of cooperative SLAM was based on salient landmarks to represent prominent characteristics. For that, each robot performs its own monocular SLAM with Extended Kalman Filter (EKF). The merge algorithm uses duplicated landmarks to increase the accuracy of the centralized map. In [18], a laser and webcam were used to model an area. By employing multiple autonomous UGVs, this work performs exploration with teams of robots for learning. Each robot creates a partial 3D map that it shares with other robots in its communication range. A global map is created on the basis of matching poses and mutual characteristics found in individual maps. The authors in [19] presented an implementation of multiple GraphSLAM using a stereo camera. Here, autonomous UGVs perform 6D mapping of an area using graph topology to separate uncertainty estimates of the local filters of multiple robots in a SLAM graph.

3. Methodology—Bioinspired Mapping Method

For [20], an exploration task is the combination of both mapping and robot motion-control activity.

This work proposes an environment exploration method with multiple UAVs inspired by how bees build hives. The authors in [3] discussed how bees perform hive construction. Following the behavior of bees in the construction of each honeycomb, UAVs perform the build and exploration map in a similar way, where combs are represented as hexagons. Each honeycomb can have only one bee occupying its space, so each hexagon can hold a maximum of one UAV. The built map is a collection of hexagons.

The construction of a beehive begins with the work of the first bee, which begins construction of the first honeycomb using wax to build its walls. Similarly, in the proposed method, a first UAV, identified as the sentinel, generates the first map hexagon, checking whether there are adjacencies for each of the six sides (honeycomb walls). In this case, the term adjacency means the possibility for a UAV to move from one hexagon to another. Thus, a hexagon exists on the map if and only if it is possible for a UAV to fully access it from another hexagon on at least one of its six sides, so obstacles cannot exist between the center of one hexagon and the center of the other hexagon. Figure 1 shows a UAV exploring a hex that should rotate at six angles: $\pi/2$, $\pi/6$, $-\pi/6$, $-\pi/2$, $-5\pi/6$, and $5\pi/6$. Each evaluated hexagon with possible adjacency is marked with an identifier.

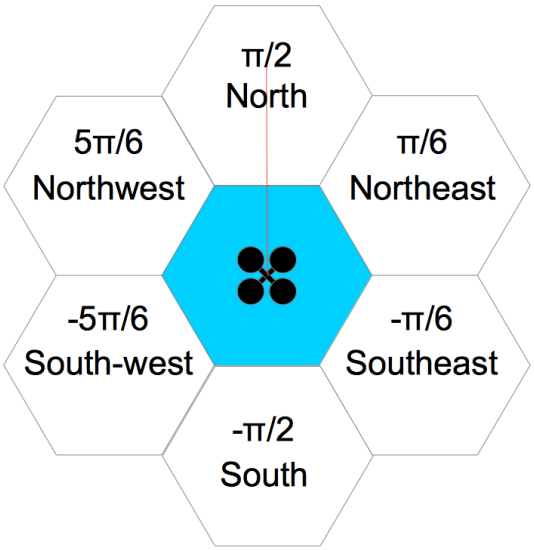
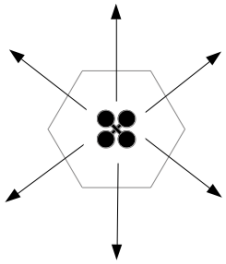


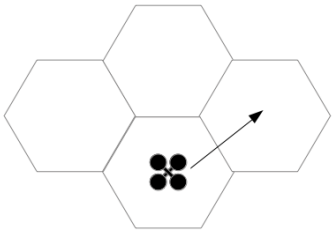
Figure 1. Unmanned Aerial Vehicle (UAV) in hexagon exploration.

Briefly, the UAV explores the hexagon in each of its six angles, sets a new hexagon to explore and moves to this, starts a new exploration. The Figure 2 shows this action.

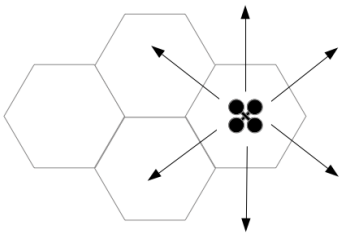
Once the sentinel UAV finishes the first scan, all UAVs can start searching for spaces to explore. To control the hexagons identified in the reading process from each of the six angles, some structures are used. To record the identifiers (ids) of the explored hexagons, a list called "visited hexagon list" is used. When the UAV rotates and finds adjacency for a new hexagon, a new id is generated and added into a structure called a "not visited hexagon list". Thus, a UAV searching for a hexagon to explore should perform this search in the "not visited hexagon list".



(**a**) Action 1: explore the first hexagon.



(**b**) Action 2: get a new hexagon to explore and scroll to it.



(**c**) Action 3: explore the new hexagon

Figure 2. Exploration stages.

At the end of hexagon exploitation, id is removed from the latter list. Figure 3 presents two UAVs exploring a given space. In this case, exploration started with hexagon 1, which was already explored. For illustration purposes, hexagon 1 is green, indicating that it was already fully explored. In its exploration process, adjacencies were identified with hexagons 2–4, which were inserted into the "not visited hexagon list". When a UAV began exploring hexagon 2, hexagons 5–7 were identified. Blue hexagons represent spaces in exploration, while yellow ones are those that were identified but not yet explored. The exploration process ends when the "not visited hexagon list" is empty.

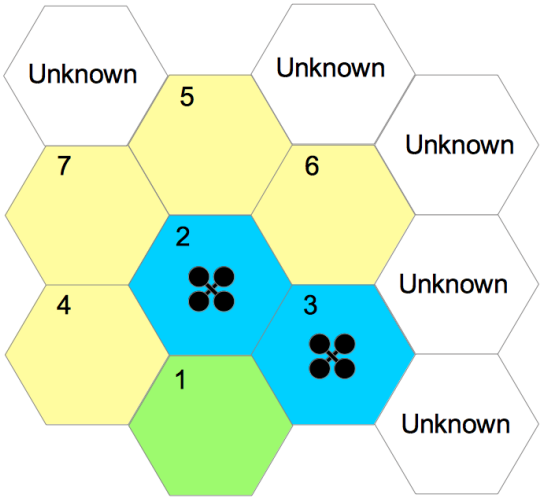


Figure 3. Multiple UAV exploration: green hexagon, explored place; yellow hexagons, places that have adjacency, but not yet explored; blue hexagons, places that are explored by UAV; white hexagons, unknown places that are mapped in future steps.

3.1. Environment Exploration

Figures 4 and 5 present state diagrams of the scanning activity of both sentinel and other UAVs. The sentinel UAV only behaves differently in the first exploration (where it generates the first *id* from point *xyz* from its placement); in the others, it has the default behavior of the other UAVs.

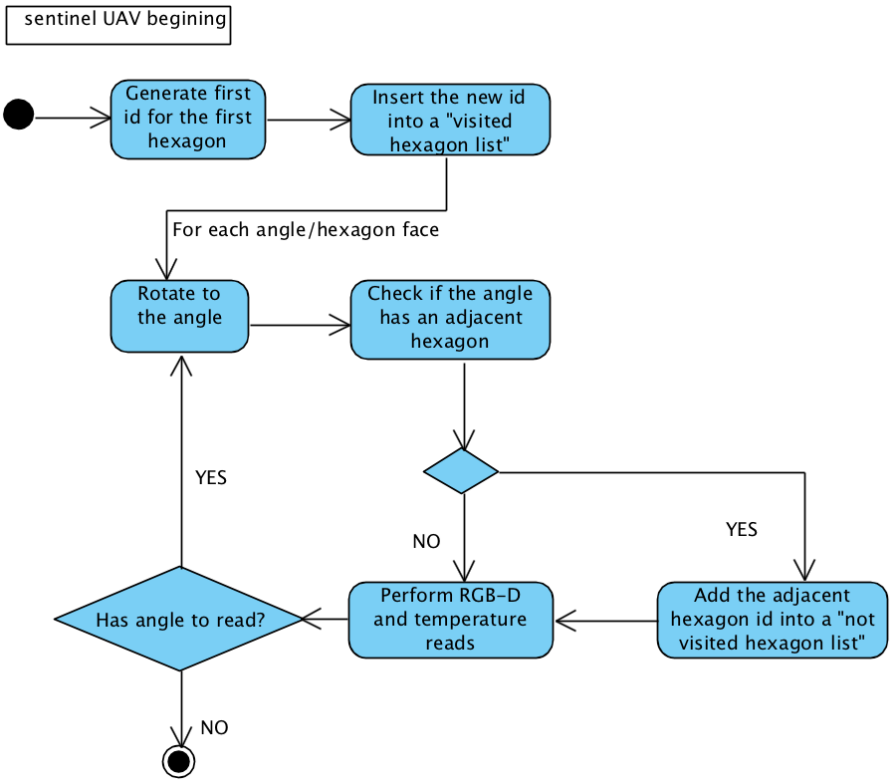


Figure 4. First sentinel UAV exploration.

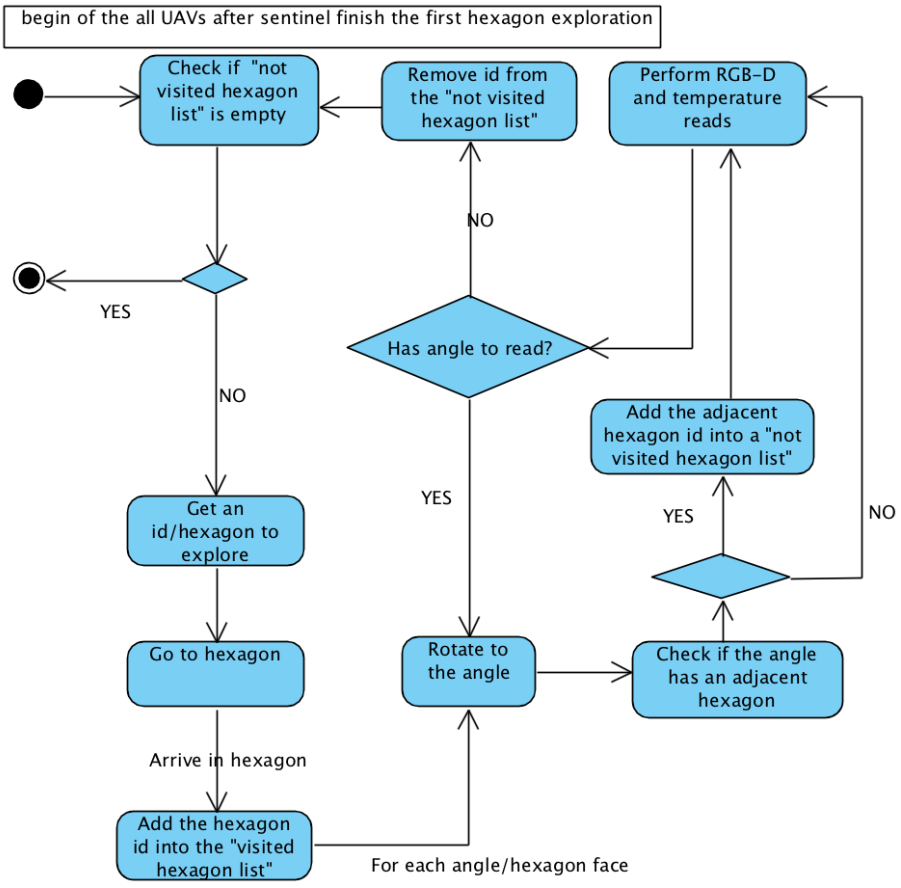


Figure 5. All UAVs after first sentinel exploration.

3.1.1. Checking If *"Not Visited Hexagon List"* Is Empty

This is the stopping criterion of the exploring algorithm. Each uncovered discovered hexagon is inserted into the "not visited hexagon list". When a UAV receives a id to explore, it remains in the "not visited hexagon list" to the end of the exploration, but the UAVs that it exploits are registered. This ensures that no UAV stops the exploration process without actually having any new spaces to explore. For example, at one point in the exploration, one UAV may have finished its exploration, while another is working. If no unvisited hexagons are currently available, the first UAV waits for possible discoveries of the second UAV, which is still in exploration activity. If no new hexagon is discovered, then exploration is finished, or new explorations process are done again.

3.1.2. Getting Id/Hexagon to Explore

When a UAV is free (no hexagons in the exploration process), it seeks a new place to explore, which is done in the "not visited hexagon list". To define which is assigned to the UAV, two metrics were defined for different simulations: First-In–First-Out (FIFO) and Euclidean distance. The FIFO metric assigns to the UAV that unvisited hexagon than has been awaiting exploration for the longest, so the first discoveries are the first to be explored. The second metric defines that the hexagon to be explored by the UAV is the one with the smallest Euclidean distance from the initial hexagon (id 1).

3.1.3. Go to Hexagon

Once the UAV gets a hexagon to explore, it must travel there. The UAV only transitions through familiar and accessible spaces. Thus, given the hexagon where the UAV is located and the target, one path is defined to go. This path is built from Dijkstra's algorithm [21], with an adapted version from [22]. With this path, the UAV travels the map until it reaches its target.

3.1.4. Add Hexagon Id into "Visited Hexagon List"

When the UAV finishes moving along the path defined by Dijkstra's algorithm, it is in the hexagon to explore. At the beginning of the exploration activity, hexagon id is inserted into the "visited hexagon list" . This ensures that a UAV identifies a hexagon already found and identified by another UAV, not creating a new id for the same space.

3.1.5. Rotate to Angle and Check If Angle Has Adjacent Hexagon

For each six sides of the hexagon (six angles), the UAV should rotate and check for adjacency: a sensor checks if it is possible for the UAV to access the center of the neighboring hexagon; in other words, if there are no obstacles between the two hexagons. If so, adjacency is added to an adjacency matrix. Each angle of the explored hexagon is identified in the honeycomb map with dashed lines if there is adjacency at that angle, or with continuous lines if there is none, as shown in Figure 6.

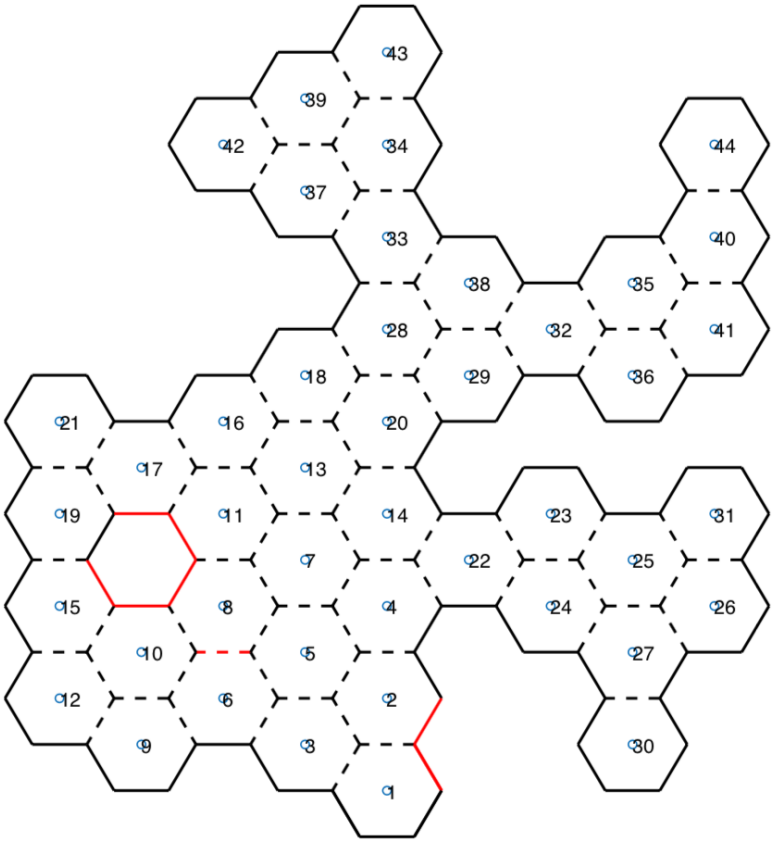


Figure 6. Honeycomb map.

3.1.6. Add Adjacent Hexagon Id into "Not Visited Hexagon List"

When the sensor reading discovers an adjacent hexagon for each of the six angles, and it is not in the "visited hexagon list", it sends it to the "not visited hexagon list", if it is not already there (this is a newly discovered hexagon).

3.1.7. Perform RFB-D and Temperature Reads

Map information is available for both UAVs to control their movements in the environment, and for rescue teams to know the space that can be navigated. In addition to obstacle sensors, RGB-D and temperature sensors are used. RGB-D sensors read the 3D angle of the UAV, and a cube view is then built to aid rescue teams in space recognition. At the same time, a thermal sensor reads the temperature from the same angle. If a temperature higher than a reference value is found, it is identified and a photo of the location is taken. In honeycomb map, this scenario is represented with red lines, as shown in Figure 6.

The RGB-D reading returns a matrix structure. The matrix size (nxm) and the range of RGB-D sensor are set in a V-REP simulator in the sensor settings. Matrix values are between 0 and 1, where 0 is very close to and 1 very far from the sensor. When the UAV reads RGB-D, the 3D data, and the angle and position of the UAV during the reading are recorded. These data are transformed from a perspective to a global point. For instance, let R be the cube size, $amplitudeRGBD$ the extent of the RGB-D sensor, $buffer$ the RGB-D matrix read, xn and yn the $buffer$ dimension, $angUAV$ the UAV Euler angle, and $posUAV$ the xyz UAV position. Algorithm 1 brings the data transformation.

Algorithm 1 RGB-D transformation algorithm.

```
function [xc,yc,zc] = TransformRGBD(R, amplitudeRGBD,
                                    buffer, xn, yn, angUAV, posUAV)
xc = 0;
yc = 0;
zc = 0;
deltaAngleRGBD = double(amplitudeRGBD)/double(xn);
for i=1:yn
    for j=1:xn
        if (double(buffer(i,j))<0.99)
            angUAVz=rad2deg(angUAV(3));
            ---nz is the distance in meter.
            ---RGB-D is set to 2m
            nz=double(buffer(i,j))*2;
            if (j==1)
                dtAng=0;
            elseif (j==xn)
                dtAng=amplitudeRGBD;
            else
                dtAng=deltaAngleRGBD*double(j);
            end
            if (dtAng < amplitudeRGBD/2)
                alfa=double(angUAVz)+((amplitudeRGBD/2)-dtAng);
            else
                alfa=double(angUAVz)-(dtAng-(amplitudeRGBD/2));
            end
            alfarad=deg2rad(alfa);
            ---Sine's Law
            dy = double(nz*sin(double(alfarad))/sin(deg2rad(90)));
            dx = nz*sin(double(deg2rad(180-90-alfa)))/sin(deg2rad(90));
            ---calculate dz
            angUAVx = rad2deg(angUAV(1));
            if (i==1)
                dtAngz=0;
            elseif (i==xn)
                dtAngz=amplitudeRGBD;
            else
                dtAngz = deltaAngleRGBD*double(i);
            end
            if (dtAngz < amplitudeRGBD/2)
                alfaz=double(angUAVx)+((amplitudeRGBD/2)-dtAngz);
            else
                alfaz=double(angUAVx)-(dtAngz-(amplitudeRGBD/2));
            end
            dz = nz*sin(double(deg2rad(alfaz)))/sin(deg2rad(90));
            xp = posUAV(1) + dx;
            yp = posUAV(2) + dy;
            zp = posUAV(3) + dz;
            ---Discretizing values.
            xc = round(xp/R)*R;
            yc = round(yp/R)*R;
            zc = round(zp/R)*R;
        end
    end
end
end
```

3.1.8. Remove Id from "Not Visited Hexagon List"

At the end of the reading of the six sides of the hexagon, id is removed from the "not visited hexagon list". Then, the UAV can begin the search for a hexagon to explore again if the "not visited hexagon list" is not empty; otherwise, the UAV's exploration activity is finished.

3.2. Lock Path Resolution

Throughout the exploration process, the various UAVs will be moving towards their targets, and consequently, their paths may cross. Avoiding collisions is a critical point for an environment with multiple robots. Several approaches have been presented, where means of prevention are proposed by optimized programming [23–25], potential fields [26], sampling-based methods [27], and others. In general, two concepts are applied [28]: one where robots are free and can change their paths, and another where robots have a fixed path with no possibility of changes. Thus, in the first concept, the focus is on changing paths, while in the second, the focus is on controlling movement and time. In [28] a method for treating deadlock for multiple robots where the path is fixed is discussed. To improve performance, some stopping policies are proposed. With these policies, each robot makes the decision to change or wait for another one. A correct-by-construction synthesis approach to multi-robot mission planning that guarantees collision avoidance with respect to moving obstacles are approach in [29], where has done an integration of a high-level mission planner with a local planner that guarantees collision-free motion in three-dimensional workspaces, when faced with both static and dynamic obstacles.

To avoid collisions, the proposed architecture defines that only one UAV can occupy one hexagon (honeycomb) at a time. Thus, it is necessary to have a record of the hexagon that the UAV currently occupies. To do this, each time a UAV moves from one hexagon to another, it records both which one it is in and what is the next move. Collision is avoided in this way; however, deadlock states can happen.

In the proposed approach, it is assumed that a UAV can be found in three possible states: "in exploration", "in displacement" or "stopped". The state "in exploration" means that the UAV is reading the six angles of the hexagon (honeycomb), and generating the mapping data. "In displacement" means that the UAV is moving to a hexagon and make their exploitation. The "stopped" state means that the UAV has no allocated exploration, and is not moving to any honeycomb.

A key element for resolving path blocks in this approach is the Adjacent Degree (AD), which is the number of adjacent hexagons, directly or indirectly, to which the UAV can travel, in order to free the paths. To obtain the AD, each UAV checks how many hexagons are directly adjacent to it, excluding those in which they are occupied by other UAVs. If the AD value is greater than 1, it means that there is space to perform a maneuver to release the passage. If the AD value is 1, the AD value for this adjacent single is searched. The AD value found for the adjacent one will be its value as well.

A comparison between the AD values of each UAV is made, and if there is a conflict between UAVs, the one with the highest AD must give way to the one with the smallest, moving to one of its adjacent hexagons to resolve the deadlock . When he finishes moving, he retraces his trajectory for his hexagon to explore and returns to his tasks. The Figure 7 presents a scenario with two UAVs, where the UAV in the blue hexagon has three adjacent hexagons directly and its AD value is 3, while the one in yellow has a single adjacent one; however, this, in turn, it has two adjacent hexagons, making the UAV AD in the yellow hexagon to be 2.

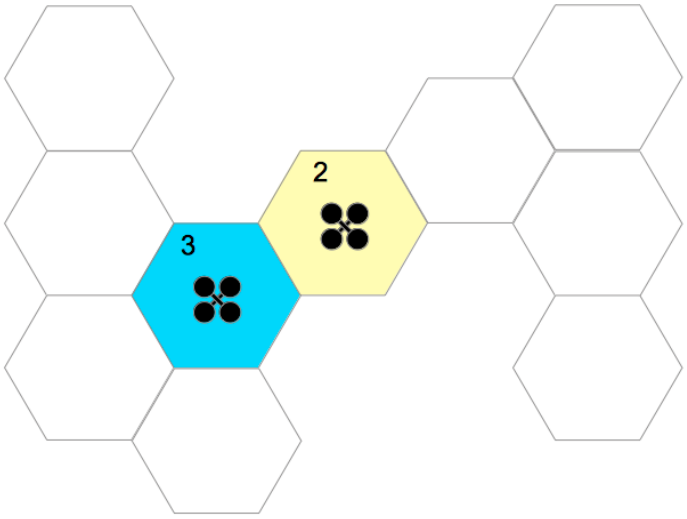


Figure 7. Adjacent Degree: blue hexagon has AD = 3, while yellow hexagon AD = 2.

Considering the existence of several UAVs in the environment, each one identified as $A, B, C, ..., Z$, and $A \rightarrow B$ representing the UAV who wants to move to the hexagon which is the UAV B. Some cases of path blocking can happen:

- **Case 1—$A \rightarrow B$ and $B \rightarrow A$:**

  In this scenario, UAV A wants to move to the hexagon of UAV B, and at the same time, UAV B wants to move to the hexagon where UAV A. Here, each UAV calculates its adjacent degree (AD). The UAV that has the largest AD will open the way to the other UAV.
- **Case 2—$A \rightarrow B$:**

  In this scenario, only UAV A shows that it wants to move to the hexagon of UAV B; however, B will not go to the hexagon of A is. In this case, UAV B may be in an "in exploration" or "stopped" state. If it is in an "in exploration" state, UAV A will recalculate a new path trying to deflect the hexagon occupied by B. If there is only one path, UAV A waits for UAV B to complete its exploration. On the other hand, if UAV B is in a "stopped" state, UAV B itself will identify that UAV A wants to go to the hexagon it occupies. That way, it will calculate your AD and compare it with the UAV A. If your AD is greater, it will move to a free adjacent hexagon, and otherwise, it will try to move to a hexagon adjacent to the UAV A, which causes them to find themselves in Case 1.
- **Case 3—$A \rightarrow B$, $B \rightarrow C$ and $C \rightarrow A$:**

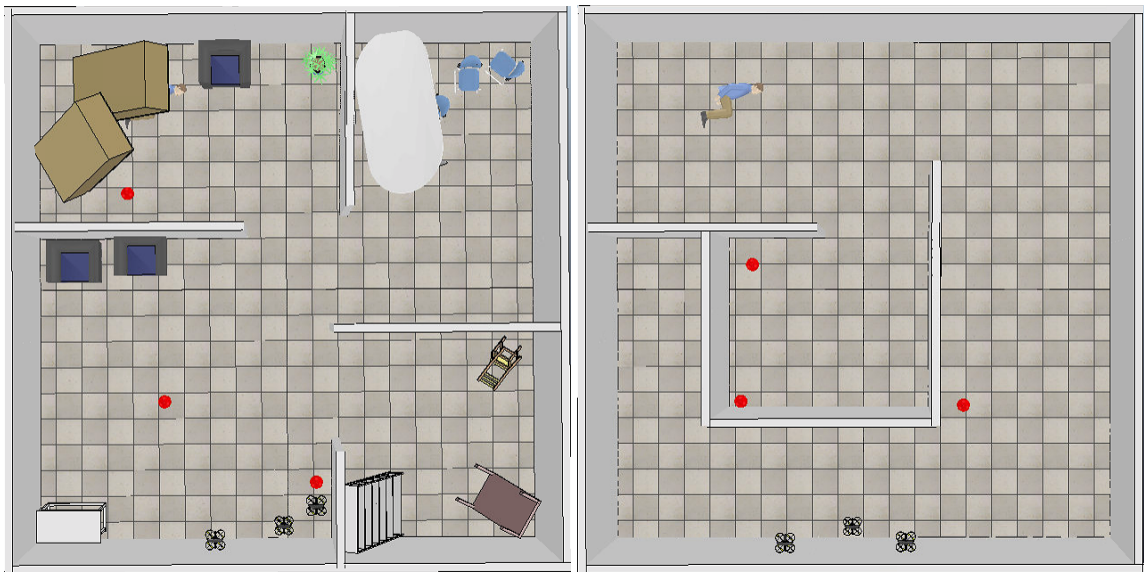
  In this case, two UAVs are unable to mutually identify a deadlock. So, it is necessary to check if there is a cyclically blocking. Thus, from the hexagon to which you want to move, UAV A checks if there are any others that want to move to where it is. If this block is detected, the UAV calculates its AD, and if it is greater than 0, it will give space for the resolution of the deadlock. After that, the path to the defined hexagon will be recalculated, and then continue your task.

3.3. Simulation

To validate the proposed method, simulations were performed with different scenarios and UAV numbers. Through the simulation, it was possible to verify the proposed approach, i.e., to plan the UAV tasks to map a catastrophic environment. Simulations were created with the following setup: CPU, Intel Xeon with 3.33 GHz 6 Core, 6 GB 1333 MHz DDR3 memory, and GPU ATI Radeon HD 5770 1024 MB.

There are several robot simulation environments, such as Open HRP [30], Gazebo [31], Webots [32] and Virtual Robot Experimentation Platform (V-REP) [33]. In this work, we chose V-REP, which has application programming interfaces (API) that allow communication with many programming languages. The proposed approach was implemented in MATLAB [34].

Figure 8 shows the simulation scenarios. Both were 10 × 10 m locations. Scenario 1 (Figure 8a) presents a place characterized by rooms with furniture that were knocked down, like an earthquake scene, while Scenario 2 (Figure 8b) is a place with passages; red dots represent fire spots.



(**a**) Scenario 1. (**b**) Scenario 2.

Figure 8. V-REP simulation scenarios.

To perform exploration, the simulations made use of two and three similar UAVs. Figure 9 shows a used UAV. The UAV was equipped with an RGB-D camera, a thermal sensor, and a laser sensor. The laser sensor took a 0.5 cm radio to the honeycomb, so distance from a hexagon center to another was 1 m. For each scenario and each configuration (two or three UAVs), simulations were performed with the FIFO and Euclidean Distance algorithms.

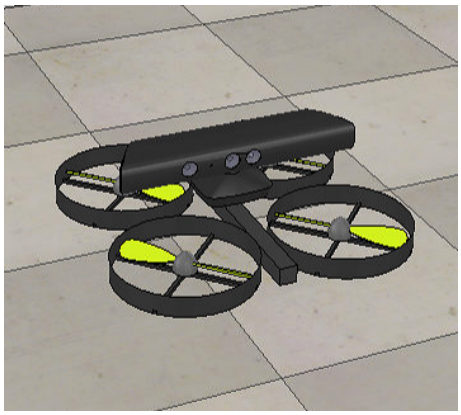
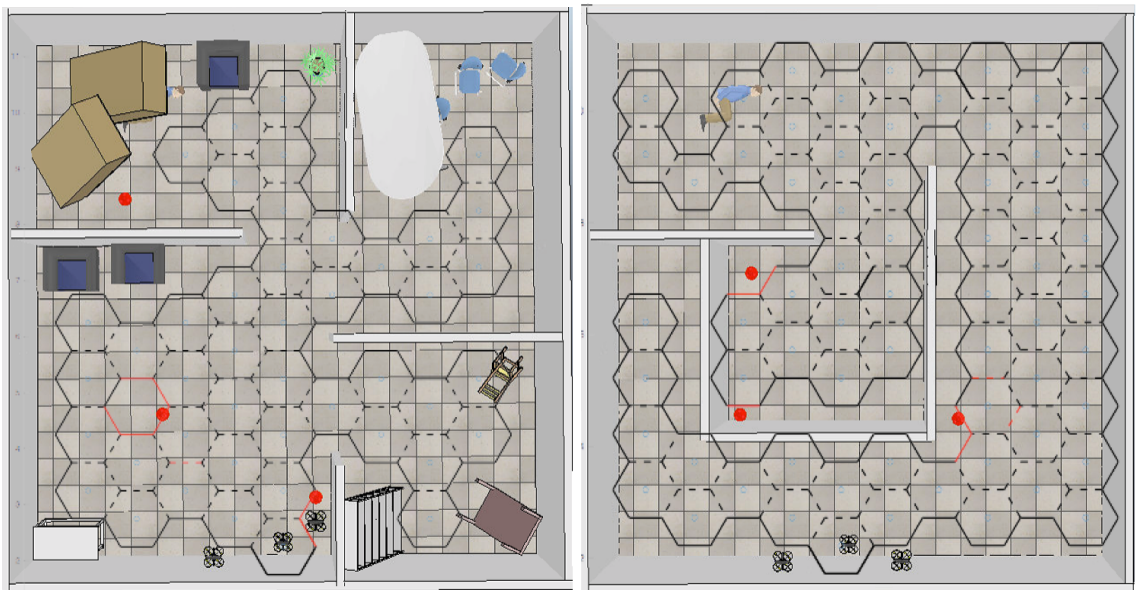


Figure 9. UAV in simulation.

4. Results

Figure 10 shows scenarios merged with the honeycomb-map build. After the simulations were performed, it was possible to verify the displacements of each UAV within the generated map, as well as the order of honeycomb exploration by each UAV.



(**a**) Scenario 1 with honeycomb map. (**b**) Scenario 2 with honeycomb map.

Figure 10. Scenarios merged with honeycomb map.

4.1. Scenarios and Honeycomb-Map Generation

Figures 11 and 12 show the movements made by the UAVs in the simulations of Scenarios 1 and 2, respectively. Figures 13 and 14 present the exploration order of each UAV with the respective hexagon-definition algorithm to be explored, FIFO and Euclidean distance. The blue line correspond to UAV 1, the red is UAV 2, and the green is UAV 3 (when the simulation had three UAVs).

The yellow circle identifies the highest-traffic hexagon. Hexagon traffic means how many times a UAV went through the hexagon. Table 1 shows the max traffic number in the simulations. Considering the *ids* of Table 1, and relating them in Figures 11 and 12, these most accessed hexagons were located in places characterized as doors or passageways.

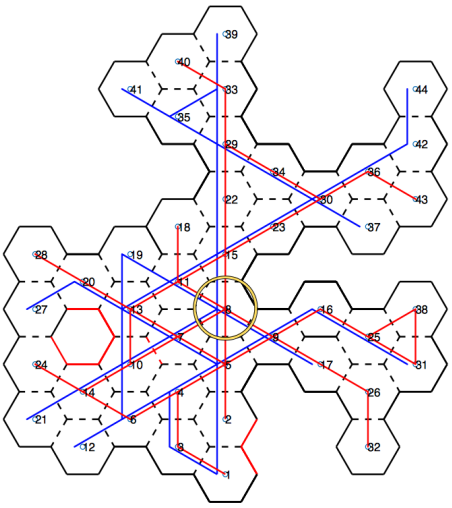
In the simulations, the movements of each UAV were recorded. Displacement means that a UAV moved from a hexagon to an adjacent one. Table 2 shows the displacement number and average per UAV in each simulation in Scenario 1. Table 3 shows the same for Scenario 2. Table 4 bring the exploration time. Tables 5 and 6 details data from both exploration order and displacement.

Table 1. Hexagon traffic.

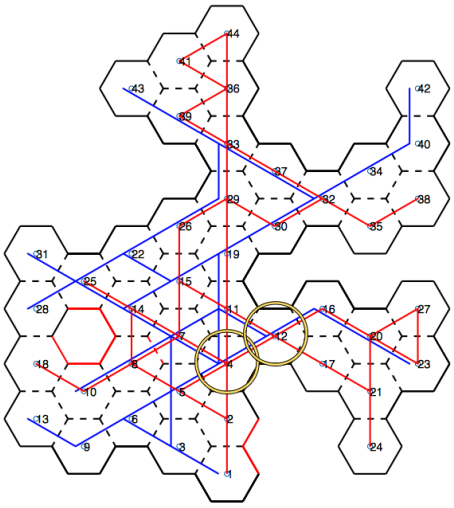
Simulation	Scenario 1		Scenario 2	
	Id Hexagon	Traffic Number	Id Hexagon	Traffic Number
Two UAVs–FIFO	8	17	3	19
Two UAVs–Euclidean distance	4 and 12	8	3	13
Three UAVs–FIFO	5 and 10	15	2	21
Three UAVs–Euclidean distance	10 and 13	9	12	14

Table 2. Displacement number—Scenario 1.

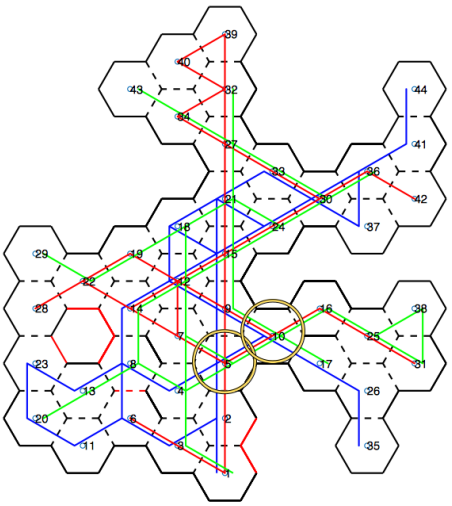
Simulation	FIFO	Average/UAV	Euclidean Distance	Average/UAV
Two UAVs	196	98	143	71.5
Three UAVs	203	67	169	56.33
Variation	-	31.63%	-	21.21%



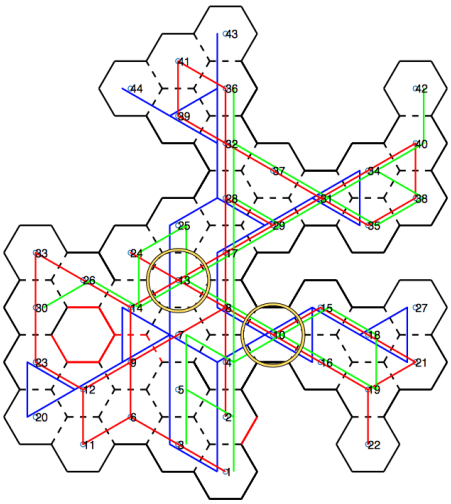
(**a**) Scenario 1 - 2 UAVs—First-In–First-Out (FIFO).



(**b**) Scenario 1. Two UAVs–Euclidean distance (ED).



(**c**) Scenario 1. Three UAVs–FIFO.

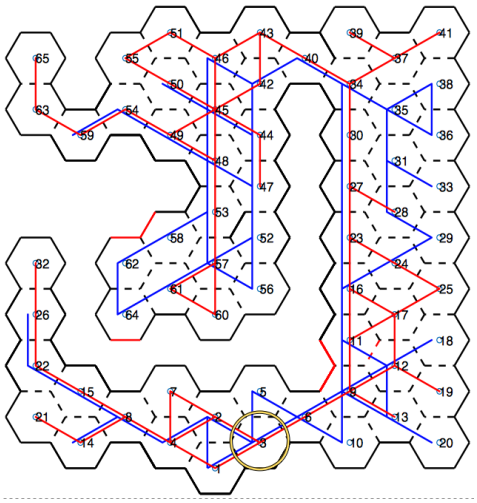


(**d**) Scenario 1. Three UAVs–Euclidean distance.

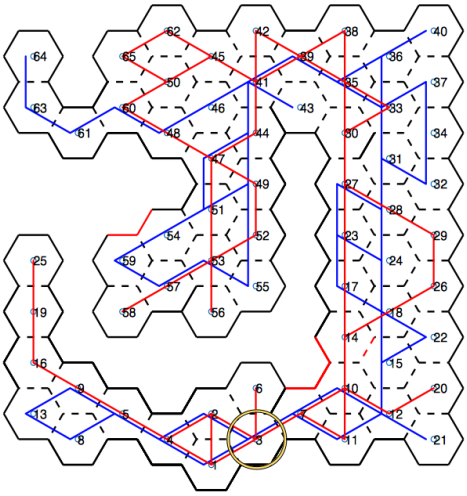
Figure 11. Displacement—Scenario 1.

Table 3. Displacement number—-Scenario 2.

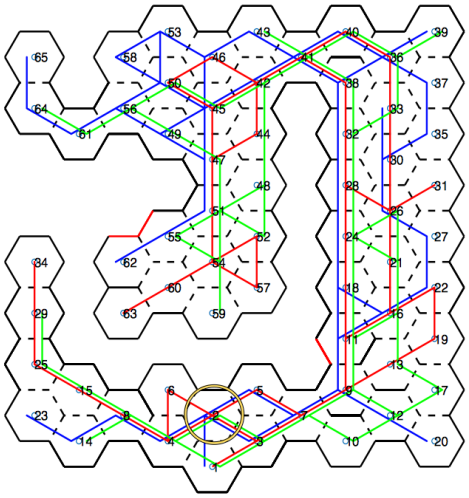
Simulation	FIFO	Average/UAV	Euclidean Distance	Average/UAV
Two UAVs	290	145	206	103
Three UAVs	324	108	271	90.33
Variation	-	25.51%	-	12.29%



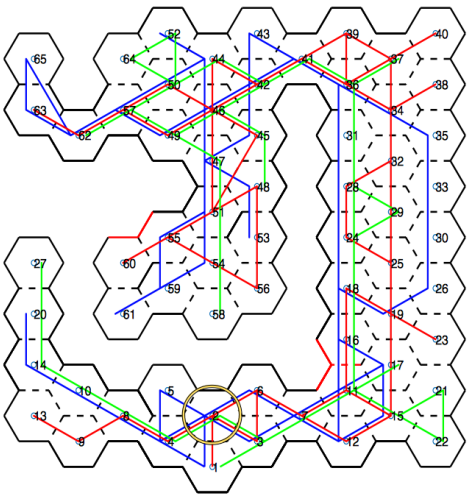
(**a**) Scenario 2-2. UAV–FIFO.



(**b**) Scenario 2-2. UAV–Euclidean distance.

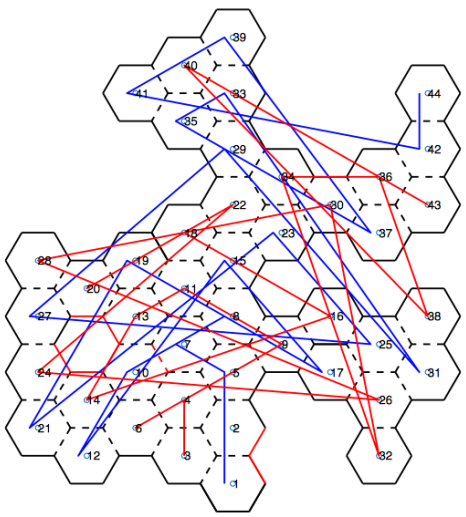


(**c**) Scenario 2-3. UAV–FIFO.

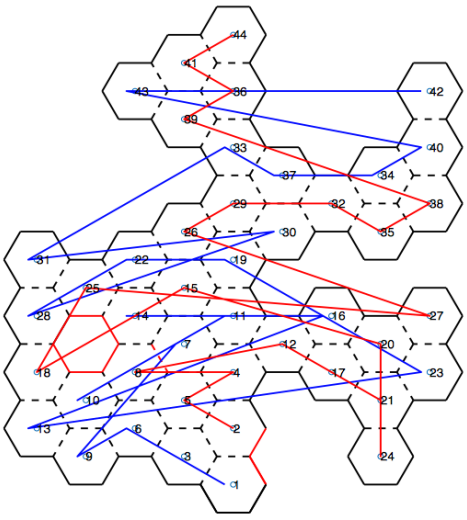


(**d**) Scenario 2-3. UAV–Euclidean distance.

Figure 12. Displacement—Scenario 2.

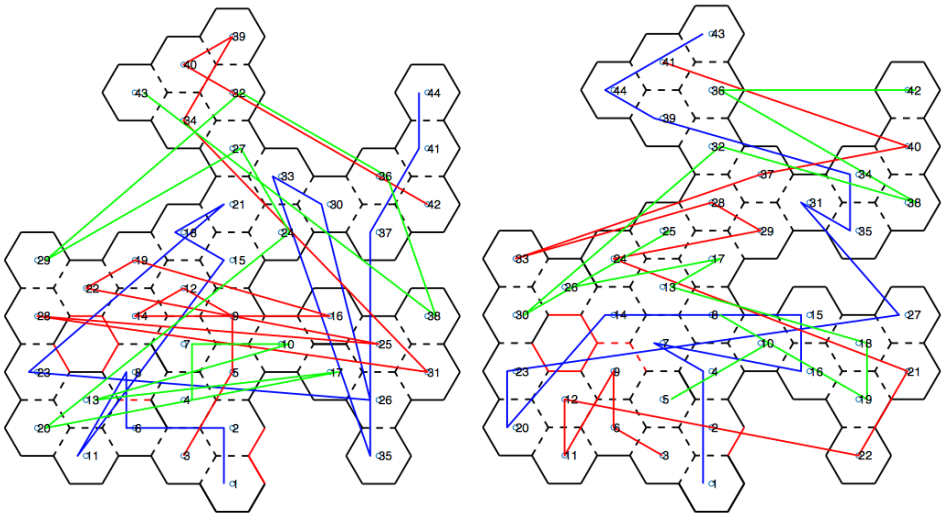


(**a**) Scenario 1-2. UAV–FIFO.



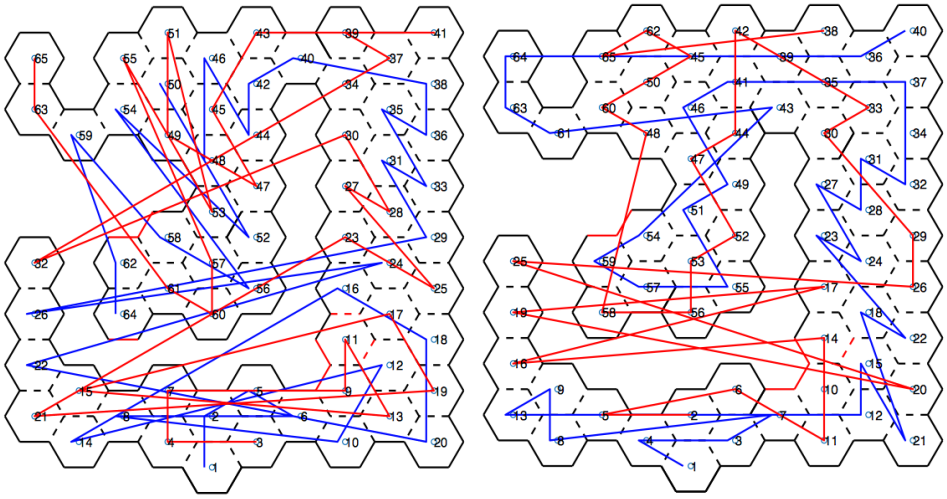
(**b**) Scenario 1-2. UAV–Euclidean distance.

Figure 13. *Cont.*

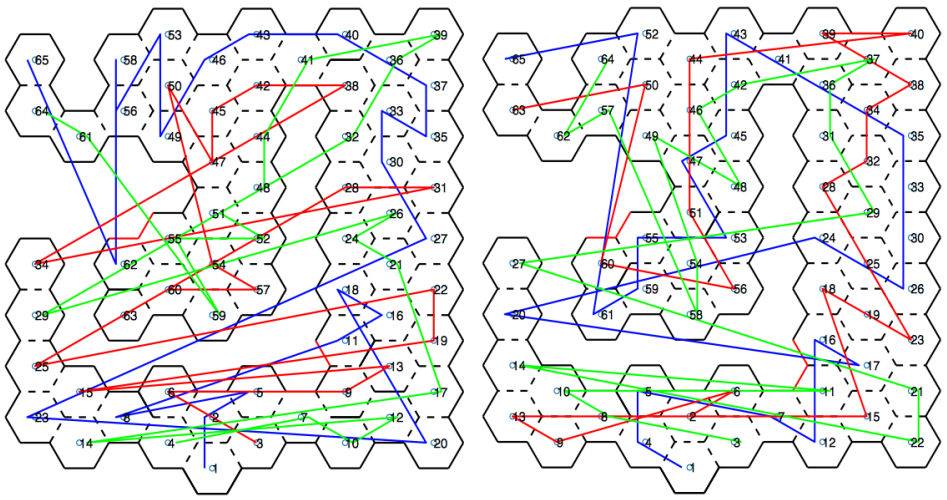


(**c**) Scenario 1-3. UAV–FIFO. (**d**) Scenario 1-3. UAV–Euclidean distance.

Figure 13. Exploration order—Scenario 1.



(**a**) Scenario 2-2. UAV–FIFO. (**b**) Scenario 2-2. UAV–Euclidean distance.



(**c**) Scenario 2-3. UAV–FIFO. (**d**) Scenario 2-3. UAV–Euclidean distance.

Figure 14. Exploration order—Scenario 2.

Table 4. Exploration time.

Simulation	Scenario 1	Scenario 2
Two UAVs–FIFO	2:30:32	3:00:17
Two UAVs–Euclidean distance	2:24:56	2:27:58
Three UAVs–FIFO	3:44:25	2:08:09
Three UAVs–Euclidean distance	3:54:17	1:56:08

Table 5. Exploration order.

Scenarios	UAV Number	Exploration Order
Scenario 1		**Three UAV - Euclidean Distance**
	UAV 1	1 2 4 7 16 15 14 20 23 27 31 35 34 39 44 43
	UAV 2	3 6 9 11 12 22 21 24 29 28 33 37 40 41
	UAV 3	5 10 8 19 18 13 17 26 25 30 32 38 36 42
		Two UAV - Euclidean Distance
	UAV 1	1 3 6 9 7 10 11 14 16 13 23 19 22 28 30 31 33 37 34 40 43 42
	UAV 2	2 5 4 8 12 17 21 24 20 15 18 25 27 26 29 32 35 38 39 36 41 44
		Three UAV - FIFO
	UAV 1	1 2 6 8 11 15 18 21 23 26 30 33 35 37 41 44
	UAV 2	3 5 9 12 14 16 19 22 25 28 31 34 39 40 42
	UAV 3	4 7 10 13 17 20 24 27 29 32 36 38 43
		Two UAV - FIFO
	UAV 1	1 2 5 7 8 10 12 15 17 19 21 23 25 27 29 31 33 35 37 39 41 42 44
	UAV 2	3 4 6 9 11 13 14 16 18 20 22 24 26 28 30 32 34 36 38 40 43
Scenario 2		**Three UAV - Euclidean Distance**
	UAV 1	1 4 5 7 12 16 17 20 24 26 30 33 35 41 43 45 47 53 55 59 61 52 65
	UAV 2	2 6 9 13 15 18 19 23 25 28 32 34 38 39 40 44 51 56 60 50 63
	UAV 3	3 8 10 11 14 22 21 27 29 31 36 37 42 46 48 49 54 58 57 62 64
		Two UAV - Euclidean Distance
	UAV 1	1 4 3 7 8 9 13 12 15 21 18 22 23 24 27 28 31 32 34 37 41 46 49 51 55 57 59 54 43 61 63 64 36 40
	UAV 2	2 5 6 11 10 14 16 17 19 20 25 26 29 30 33 35 39 42 44 47 52 53 56 58 48 60 50 45 62 65 38
		Three UAV - FIFO
	UAV 1	1 2 5 8 11 16 18 20 23 27 30 33 35 37 40 43 46 49 53 56 58 62 65
	UAV 2	3 6 9 13 15 19 22 25 28 31 34 38 42 45 47 50 54 57 60 63
	UAV 3	4 7 10 12 14 17 21 24 26 29 32 36 39 41 44 48 51 52 55 59 61 64
		Two UAV - FIFO
	UAV 1	1 2 5 6 8 10 12 14 16 18 20 22 24 26 29 31 33 35 36 38 40 42 44 46 48 50 52 54 56 58 59 62 64
	UAV 2	3 4 7 9 11 13 15 17 19 21 23 25 27 28 30 32 34 37 39 41 43 45 47 49 51 53 55 57 60 61 63 65

Table 6. Displacement order.

Scenarios	UAV Number	Exploration Order
Scenario 1		**Three UAV - Euclidean Distance**
	UAV 1	1 2 4 7 4 10 16 15 10 4 7 14 9 12 20 23 12 9 7 4 10 15 18 21 27 18 15 10 8 17 29 31 35 34 31 29 28 32 39 44 39 36 43 36 32 28 25 13 7 5 3 1
	UAV 2	1 3 6 9 6 11 12 9 7 8 10 16 19 22 19 21 18 15 10 8 13 24 13 17 29 28 17 13 14 9 12 23 30 33 26 14 13 17 28 32 37 31 35 38 40 34 31 29 28 32 36 41 39 32 28 17 8 4 2
	UAV 3	1 2 5 2 4 10 8 10 16 19 18 15 10 8 13 17 13 14 26 14 24 25 13 14 26 30 26 14 13 17 28 32 28 29 31 35 38 34 31 37 32 36 32 37 31 34 40 42 40 34 31 29 17 8 4 7 5 3
		Two UAV - Euclidean Distance
	UAV 1	1 3 6 9 6 5 7 8 10 8 7 11 7 14 7 4 12 16 12 4 5 6 9 13 9 6 5 4 12 16 20 23 20 16 12 11 19 15 22 25 28 25 14 15 19 30 19 15 14 25 31 25 22 26 29 33 37 32 34 40 34 32 37 33 39 43 39 33 37 32 34 40 42 40 34 32 30 19 11 4 5 3 1
	UAV 2	1 2 5 4 5 8 5 4 12 17 21 24 21 20 16 12 11 15 7 8 10 18 10 8 14 25 14 7 4 12 16 20 23 27 20 16 12 11 15 26 29 30 32 35 38 35 32 37 33 39 36 41 44 36 33 29 19 11 4 2
		Three UAV - FIFO
	UAV 1	1 2 5 2 3 6 8 6 11 6 8 4 5 9 15 18 21 15 12 14 8 6 11 20 23 13 8 4 5 10 17 26 17 10 9 15 24 30 33 21 15 9 10 17 26 35 26 17 10 9 12 18 21 24 30 37 36 41 44 41 36 30 24 15 9 5 4
	UAV 2	1 3 6 3 1 2 5 9 12 14 7 5 10 16 10 9 12 19 22 14 7 5 10 16 25 16 10 5 7 14 22 28 22 14 7 5 10 16 25 31 25 16 10 5 7 12 15 21 27 34 32 39 40 32 27 33 30 36 42 36 30 24 15 9 5 2
	UAV 3	1 3 4 7 5 10 5 4 8 13 8 4 5 10 17 10 5 4 8 13 20 13 8 14 12 15 24 21 27 21 18 19 22 29 22 19 18 21 27 32 27 33 30 36 30 24 15 9 10 16 25 31 38 25 16 10 9 15 21 27 34 43 34 27 21 18 12 7 4 3
		Two UAV - FIFO
	UAV 1	1 2 5 7 8 7 10 6 12 6 4 5 8 15 8 9 17 9 8 11 19 13 10 14 21 14 10 7 8 15 23 15 8 9 16 25 16 9 5 7 13 20 27 20 13 11 15 22 29 22 15 8 9 16 25 31 25 16 9 8 15 22 29 33 35 29 34 30 37 30 34 29 33 39 33 35 41 35 29 34 30 36 42 44 42 36 30 23 15 8 5 4 3 1
	UAV 2	1 3 4 6 4 5 9 8 11 13 10 14 6 4 5 9 16 9 8 11 18 11 13 20 13 11 15 22 15 8 7 10 14 24 14 6 4 5 9 17 26 17 9 5 7 13 20 28 20 13 11 15 23 30 23 15 8 9 17 26 32 26 17 9 8 15 22 29 34 30 36 30 23 15 8 9 16 25 31 38 25 16 9 8 15 22 29 33 40 33 29 34 30 36 43 36 30 23 15 8 5 2

Table 6. *Cont.*

Scenarios	UAV Number	Exploration Order
Scenario 2	UAV 1	**Three UAV - Euclidean Distance** 1 4 5 2 3 7 12 11 16 17 11 7 3 2 4 8 10 14 20 14 10 8 4 2 3 7 12 15 17 16 18 24 18 19 26 30 33 35 34 36 41 43 42 45 47 46 47 48 53 48 47 51 55 59 61 59 54 51 47 46 44 52 44 50 57 62 65 63 62 57 49 46 42 41 36 31 28 24 18 16 11 7 6 2 1
	UAV 2	1 2 3 6 2 4 8 9 13 9 8 4 2 6 7 11 15 11 16 18 19 23 19 25 24 28 32 34 38 34 36 39 37 40 37 36 41 42 44 46 45 51 48 51 55 51 48 53 56 54 55 60 55 51 47 46 50 57 49 57 50 57 50 57 49 57 62 63 62 57 49 46 42 41 39 37 34 32 29 25 19 17 15 12 7 6 2
	UAV 3	1 3 7 3 2 4 8 10 8 4 2 3 7 11 7 3 2 4 8 10 14 10 8 4 2 3 7 11 15 22 21 15 11 17 11 7 3 2 4 8 10 14 20 27 20 14 10 8 4 2 3 7 11 16 18 24 29 28 31 36 37 36 41 42 46 45 48 45 46 49 47 51 54 58 54 51 47 49 57 62 57 50 57 50 52 64 50 44 42 41 36 31 28 24 18 16 11 7 3
	UAV 1	**Two UAV - Euclidean Distance** 1 4 1 3 7 3 2 4 5 8 5 9 13 8 5 4 2 3 7 10 12 15 12 21 12 15 18 15 22 18 17 23 24 23 27 28 31 32 34 37 33 35 39 41 46 41 44 49 51 49 52 55 53 57 59 54 51 47 44 41 43 41 46 48 60 61 63 64 63 61 60 48 46 41 39 35 36 40 36 33 31 28 24 18 15 12 11 7 3 1
	UAV 2	1 2 4 5 4 2 3 6 3 7 11 10 14 10 7 3 1 4 5 9 16 9 5 4 2 3 7 10 14 17 14 10 7 3 2 4 5 9 16 19 16 9 5 4 2 3 7 10 12 20 12 10 7 3 2 4 5 9 16 19 25 19 16 9 5 4 2 3 7 10 14 18 26 29 28 27 30 33 35 39 42 41 44 47 49 52 53 56 53 57 58 57 53 51 47 48 60 50 45 62 65 50 45 41 39 38 35 30 27 23 17 14 10 7 3 2
	UAV 1	**Three UAV - FIFO** 1 2 5 2 4 8 4 2 3 7 9 11 16 18 11 9 12 20 12 9 7 3 2 4 8 14 23 14 8 4 2 3 7 9 11 16 22 27 26 30 33 30 35 37 36 40 36 39 36 38 41 40 41 38 41 40 41 40 41 38 43 46 45 49 45 46 53 50 56 50 53 58 50 45 47 51 55 62 55 51 47 49 56 61 56 49 56 61 64 65 64 61 56 49 45 42 41 38 32 28 24 18 11 9 7 5 2 1
	UAV 2	1 3 2 3 2 3 2 4 6 2 3 7 9 13 9 7 3 2 4 8 15 8 4 2 3 7 9 13 19 22 16 11 9 7 3 2 4 8 15 25 15 8 4 2 3 7 9 11 18 24 28 26 31 26 21 16 11 9 7 3 2 4 8 15 25 29 34 29 25 15 8 4 2 3 7 9 11 18 24 28 32 38 41 42 45 47 44 42 46 50 45 47 51 54 52 57 54 60 63 60 54 51 47 44 42 41 40 36 33 30 26 21 16 11 9 7 5 2
	UAV 3	1 3 7 3 2 4 8 10 8 4 2 3 7 11 7 3 2 4 8 10 14 10 8 4 2 3 7 11 15 22 21 15 11 17 11 7 3 2 4 8 10 14 20 27 20 14 10 8 4 2 3 7 11 16 18 24 29 28 31 36 37 36 41 42 46 45 48 45 46 49 47 51 54 58 54 51 47 49 57 62 57 50 57 50 52 64 50 44 42 41 36 31 28 24 18 16 11 7 3
	UAV 1	**Two UAV - FIFO** 1 2 3 5 6 3 2 4 8 4 2 3 6 10 9 12 9 6 3 2 4 8 14 8 4 2 3 6 9 11 16 11 12 18 12 13 20 13 9 6 3 2 4 8 15 22 15 8 4 2 3 6 9 11 16 24 16 11 9 6 3 2 4 8 15 22 26 22 15 8 4 2 3 6 9 11 16 24 29 28 31 33 31 35 36 38 35 34 40 42 44 42 46 45 48 45 50 45 44 47 52 47 48 49 54 49 48 47 52 56 57 53 58 53 48 49 54 59 54 49 48 53 58 62 64 61 57 52 47 44 42 40 34 30 27 23 16 11 9 6 3 1
	UAV 2	1 3 1 4 7 2 3 6 9 11 9 13 9 6 3 2 4 8 15 8 4 2 3 6 9 11 17 12 19 12 9 6 3 2 4 8 14 21 14 8 4 2 3 6 9 11 16 23 16 17 25 24 23 27 28 27 30 27 23 16 11 9 6 3 2 4 8 15 22 26 32 26 22 15 8 4 2 3 6 9 11 16 23 27 30 34 37 39 37 41 37 34 40 43 42 45 44 47 44 45 49 45 46 51 46 45 48 53 48 45 46 51 55 50 45 48 53 48 53 57 60 61 57 53 48 49 54 59 63 65 63 59 54 49 45 46 43 40 34 30 27 23 16 11 9 6 3 2

4.2. Cube View and Temperature Caption

In addition to performing the mapping honeycomb in a hexagonal shape, further information can be generated for rescue teams. With the RGB-D sensor, a cube view can be generated. Figure 15 shows the Scenario 1 cube projection (Figure 8a), while Figure 16 shows the honeycomb map.

A cutout of 41 and 44 hexagons from the generated map of Figure 16, and the location of these hexagons in the simulator, are shown in Figure 17. In Section 3.1.7, the TransformRGB-D Algorithm 1 shows how RGB-D points are converted into 3D cubes.

In Figure 16, red lines (continuous or dashed) indicate the temperature reading above a reference value. Then, a fire-spot photo was recorded. Figure 18 shows the caption of hexagon 14 in Scenario 1.

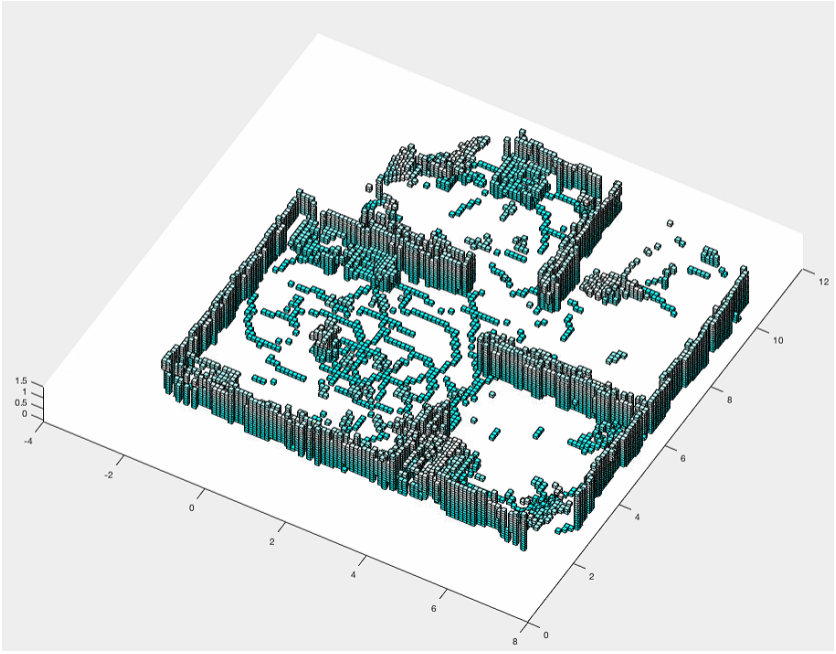


Figure 15. Three-dimensional cube view.

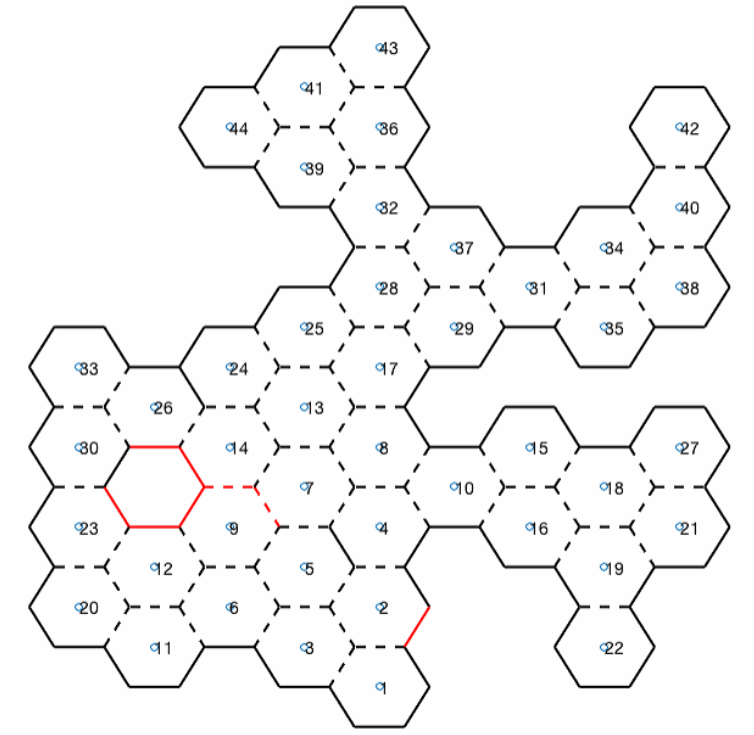
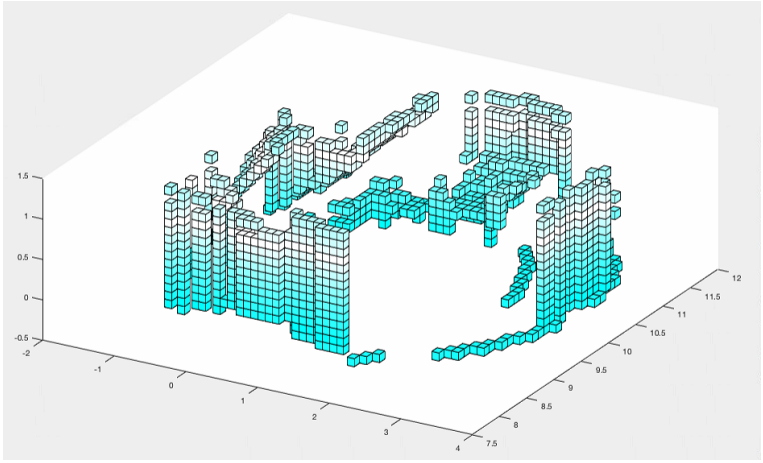
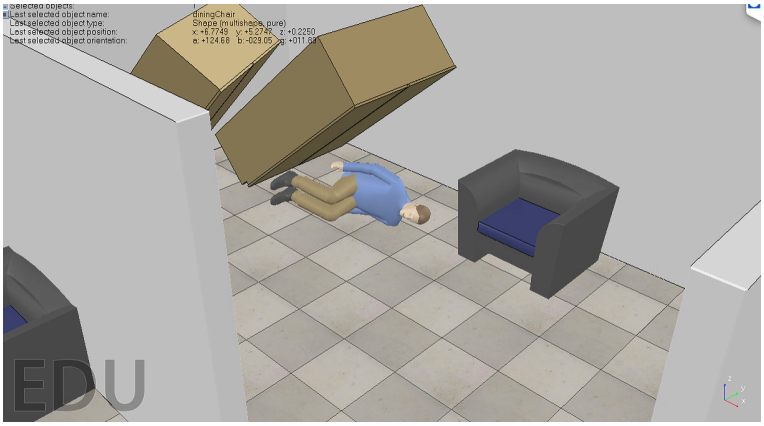


Figure 16. Honeycomb-map simulation—Scenario 1, three UAVs, Euclidean distance algorithm.



(**a**) Three-dimensional cube view—hexagons 41 and 44.



(**b**) V-REP scenario 1—hexagons 41 and 44.

Figure 17. Clipping of hexagons 41 and 44 of Figure 16.

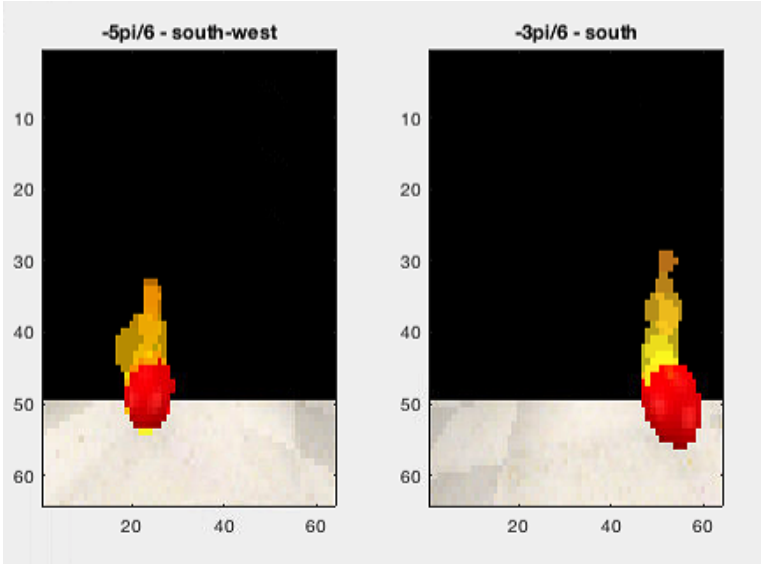


Figure 18. Fire-spot—hexagon 14 of Scenario 1.

5. Discussion

Topological mapping reduces information processing compared to metric mappings. The graph structure allows the execution of generic algorithms, such as the Dijkstra algorithm, used in trajectory planning. The data presented in Section 4 show the behavioral differences in the simulations considering number of UAVs, and algorithms in the definition of places to be explored, besides environment characteristics. By comparing the simulations, we verified that traffic in the hexagons was reduced when there was an algorithm change (FIFO for Euclidean Distance), as can be seen in Table 1. When comparing the change in UAV number, there was a slight increase in the maximum traffic value and exploration time, as can be seen in Table 4. When Scenario 2 is analyzed, the variation in the number of UAVs from two to three, in both FIFO and Euclidean Distance algorithms, reduces the exploration time. Already in Scenario 1 the opposite occurs. This happens due to the characteristics of the scenarios, where Scenario 2 has wide passages and more space for maneuvers, while Scenario 1 is composed of rooms and narrow doors, which influences the processing to avoid collisions in this points that were bottlenecks on the map. To decrease these values, an algorithm that considers not only Euclidean distance, but also the arrangement of UAVs and hexagons as a whole, should be evaluated.

On UAV displacement in the simulated scenario, Tables 2 and 3 exhibited a strong reduction in UAV movement when increasing the number of UAVs and changing the algorithm of choosing hexagons to explore. For Scenario 1, the change in the number of UAVs in the FIFO algorithm showed 31.63% reduction in average displacement per UAV. For the Euclidean distance algorithm, the reduction was 21.21%. By changing the simulation with two UAVs to three, and the FIFO algorithm for the Euclidean distance algorithm, reducing displacement in the scenario reached 42.52%; the same analysis for Scenario 2 showed a displacement decrease of 37.7%. This saves both energy and exploration time .

6. Conclusions

This work presented an environment mapping method inspired by how bees build their hives. Since only one bee constructs and occupies the space of a honeycomb, a topological map was constructed so that UAVs involved in the mapping process behaved similarly to bees. The definition of which honeycomb the UAV should map depends on a metric. The performed simulations considered two metrics to define which honeycomb should be mapped, FIFO and Euclidean Distance. In addition, simulations were performed by changing the number of UAVs. This demonstrated that setting the exploration order has direct impact on the number of offsets and a UAV in the environment, considering its position on the map. This can result in saving both energy and exploration time. Generating RGB-D and thermal-reading information enables rescuers to be prepared for obstacles and dropped objects, but also life-threatening elements such as high temperatures.

Future Work

Improvements in the definition of the spaces to be explored can be made, with metrics that consider not only distance from the initial hexagon (Euclidean distance), but also UAV location and environmental characteristics. In addition, in identifying points that may endanger the life of the rescue team, the use of gas or other toxic-element sensors may be applied. There is still the challenge of gathering this information and processing it with the use of game theory and machine learning. So far, each UAV works independently; however, it is not identified when a failure occurs with another one. A way of detecting failures and generating contingency plans needs to be implemented in future work. In this work, the representation of the hexagons is made in a projection of the x and y axes. In future work, the z axis will be added, so that this representation has several layers.

Author Contributions: R.d.R. developed the software and contributed to methodology, investigation, data curation, formal analysis, resourses, validation and writing (original draft). M.A.W. contributed to resourses, project administration, supervision, validation and writing (review and editing). T.B., J.L.L. and A.I.P.N.P. contributed to conceptualization, resourses and supervision. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Federal University of Technology (UTFPR), Federal Institute of Education, Science and Technology (IFPR) and Polytechnic of Bragança (IPB).

Acknowledgments: We would like to thank UTFPR and IFPR for their support in providing the equipment to run the simulations.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AD	Adjacent Degree
ED	Euclidean Distance
EKF	Extended Kalman Filter
FIFO	First-In–First-Out
GPS	Global Positioning Systems
IMU	Inertial Measurement Unit
IR	Infrared
JCBB	Join-compatibility Branch and Bound
PTAM	Parallel Tracking and Mapping
PTAMM	Parallel Tracking and Multiple Mapping
RGB-D	Red, Green, Blue and depth
SLAM	Simultaneous Localization and Mapping
SPF	Stigmergic Potential Field
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
VSLAM	Visual SLAM

References

1. Thrun, S. *Robotic Mapping: A Survey*; Research paper; School of Computer Science, Carnegie Mellon University: Pittsburgh, PA, USA, 2002.
2. Saeedi, S.; Trentini, M.; Seto, M.; Li, H. Multiple-Robot Simultaneous Localization and Mapping: A Review. *J. Field Robot.* **2016**, *33*, 3–46, doi:10.1002/rob.21620. [CrossRef]
3. Nazzi, F. The hexagonal shape of the honeycomb cells depends on the construction behavior of bees. *Sci. Rep.* **2016**, *6*, 28341, doi:10.1038/srep28341. [CrossRef] [PubMed]
4. Dhiman, N.K.; Deodhare, D.; Khemani, D. Where am I? Creating spatial awareness in unmanned ground robots using SLAM: A survey. *Sadhana* **2015**, *40*, 1385–1433. [CrossRef]
5. Douglas, D.H.; Peucker, T.K. Algorithms for the Reduction of the Number of Points Required to Represent a Diditized Line or its Caricature. *Cartographica* **1973**, *10*, 112–122. [CrossRef]
6. Jelinek, A. Vector Maps in Mobile Robotics. *Acta Polytech. CTU Proc.* **2015**, *2*, 22–28. [CrossRef]
7. Reimer, A. Cartographic Modelling for Automated Map Generation. Ph.D. Thesis, Technische Universiteit Eindhoven., Eindhoven, The Netherlands, 2015.
8. Stachniss, C. Coordinated multi-robot exploration. In *Robotic Mapping and Exploration*; Springer: Berlin, Germany, 2009; pp. 43–71.
9. Mahendran, A.; Dewan, A.; Soni, N.; Krishna, K.M. UGV-MAV Collaboration for Augmented 2D Maps. In *AIR '13, Proceedings of the Conference on Advances in Robotics*; ACM: New York, NY, USA, 2013; pp. 1–6, doi:10.1145/2506095.2506116. [CrossRef]
10. Michael, N.; Shen, S.; Mohta, K.; Mulgaonkar, Y.; Kumar, V.; Nagatani, K.; Okada, Y.; Kiribayashi, S.; Otake, K.; Yoshida, K.; et al. Collaborative mapping of an earthquake-damaged building via ground and aerial robots. *J. Field Robot.* **2012**, *29*, 832–841. [CrossRef]
11. Dewan, A.; Mahendran, A.; Soni, N.; Krishna, K.M. Heterogeneous UGV-MAV exploration using integer programming. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 5742–5749.

12. McCune, R.R.; Madey, G.R. Agent-based Simulation of Cooperative Hunting with UAVs. In Proceedings of the Agent-Directed Simulation Symposium, Society for Computer Simulation International, San Diego, CA, USA, 7–10 April 2013; p. 8.
13. Williams, R.; Konev, B.; Coenen, F., Multi-agent Environment Exploration with AR.Drones. In *Advances in Autonomous Robotics Systems, Proceedings of the 15th Annual Conference, TAROS 2014, Birmingham, UK, 1–3 September 2014*; Mistry, M., Leonardis, A., Witkowski, M., Melhuish, C., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 60–71. doi:10.1007/978-3-319-10401-0_6. [CrossRef]
14. Loianno, G.; Thomas, J.; Kumar, V. Cooperative localization and mapping of MAVs using RGB-D sensors. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Washington, DC, USA, 26–30 May 2015; pp. 4021–4028.
15. Rogers, J.G.; Baran, D.; Stump, E.; Young, S.; Christensen, H.I. Cooperative 3D and 2D mapping with heterogenous ground robots. In *SPIE Defense, Security, and Sensing*; International Society for Optics and Photonics: Bellingham, WA, USA, 2012; p. 838708.
16. Stipes, J.; Hawthorne, R.; Scheidt, D.; Pacifico, D. Cooperative localization and mapping. In Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control, Ft. Lauderdale, FL, USA, 23–25 April 2006; pp. 596–601.
17. Wu, M.; Huang, F.; Wang, L.; Sun, J. Cooperative Multi-Robot Monocular-SLAM Using Salient Landmarks. In Proceedings of the 2009 International Asia Conference on Informatics in Control, Automation and Robotics, Bangkok, Thailand, 1–2 February 2009; pp. 151–155, doi:10.1109/CAR.2009.22. [CrossRef]
18. Chellali, R. A distributed multi robot SLAM system for environment learning. In Proceedings of the 2013 IEEE Workshop on Robotic Intelligence in Informationally Structured Space (RiiSS), Singapore, 16–19 April 2013; pp. 82–88, doi:10.1109/RiiSS.2013.6607933. [CrossRef]
19. Schuster, M.J.; Brand, C.; Hirschmuller, H.; Suppa, M.; Beetz, M. Multi-robot 6D graph SLAM connecting decoupled local reference filters. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 5093–5100, doi:10.1109/IROS.2015.7354094. [CrossRef]
20. Makarenko, A.A.; Williams, S.B.; Bourgault, F.; Durrant-Whyte, H.F. An experiment in integrated exploration. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, 30 September–4 October 2002; Volume 1, pp. 534–539, doi:10.1109/IRDS.2002.1041445. [CrossRef]
21. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271. [CrossRef]
22. Aryo, D. Dijkstra Algorithm. MATLAB Central File Exchange. 2020. Available online: https://www.mathworks.com/matlabcentral/fileexchange/36140-dijkstra-algorithm (accessed on 2 January 2020)
23. Fukushima, H.; Kon, K; Matsuno, F. Model Predictive Formation Control Using Branch-and-Bound Compatible With Collision Avoidance Problems. *IEEE Trans. Robot.* **2013**, *29*, 1308–1317, doi:10.1109/TRO.2013.2262751. [CrossRef]
24. Gan, S. K.; Fitch, R.; Sukkarieh, s. Real-time decentralized search with inter-agent collision avoidance. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation, St Paul, MN, USA, 14–19 May 2012; pp. 504–510, doi:10.1109/ICRA.2012.6224975. [CrossRef]
25. Morgan, D.; Chung, S.; Hadaegh, F. Model Predictive Control of Swarms of Spacecraft Using Sequential Convex Programming. *J. Guid. Control Dyn.* **2014**, *37*, 1–16, doi:10.2514/1.G000218. [CrossRef]
26. Pamosoaji, A. K.; Hong, K. A Path-Planning Algorithm Using Vector Potential Functions in Triangular Regions. *IEEE Trans. Syst. Man Cybern. Syst.* **2013**, *43*, 832–842, doi:10.1109/TSMCA.2012.2221457. [CrossRef]
27. Bekris, K.E.; Grandy, D.K.; Moll, M.; Kavraki, L.E. Safe distributed motion coordination for second-order systems with different planning cycles. *Int. J. Robot. Res.* **2012**, *31*, 129–150, doi:10.1177/0278364911430420. [CrossRef]
28. Zhou, Y.; Hu, H.; Liu, Y.; Ding, Z. Collision and Deadlock Avoidance in Multirobot Systems: A Distributed Approach. *IEEE Trans. Syst. Man Cybern. Syst.* **2017**, *47*, 1712–1726, doi:10.1109/TSMC.2017.2670643. [CrossRef]
29. Alonso-Mora, J.; DeCastro, J.A.; Raman, V.; Rus, D.; Kress-Gazit, H. Reactive mission and motion planning with deadlock resolution avoiding dynamic obstacles. *Auton. Robot.* **2018**, *42*, 801–824, doi:10.1007/s10514-017-9665-6. [CrossRef]

30. Kanehiro, F.; Hirukawa, H.; Kajita, S. OpenHRP: Open Architecture Humanoid Robotics Platform. *Int. J. Robot. Res.* **2004**, *23*, 155–165, doi:10.1177/0278364904041324. [CrossRef]
31. Koenig, N.P.; Howard, A. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), Sendai, Japan, 28 September–2 October 2004; Volume 3, pp. 2149–2154.
32. Michel, O. WebotsTM: Professional Mobile Robot Simulation. *Int. J. Adv. Robot. Syst.* **2004**, *1*, doi:10.5772/5618. [CrossRef]
33. Rohmer, E.; Singh, S.P.N.; Freese, M. V-REP: A versatile and scalable robot simulation framework. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 1321–1326, doi:10.1109/IROS.2013.6696520. [CrossRef]
34. The Mathworks, Inc. MATLAB Version 9.6.0.1114505 (R2019a). Available online: https://www.mathworks.com/ (accessed on 2 January 2020)

APÊNDICE E – ALGORITMOS DESENVOLVIDOS

```
----------------------------------------
function rtn = AdicionarVerticeVisitado(xyz,QuadricopterHandle)
MatrizIDx = GetMatrizIDx();
MatrizIDy = GetMatrizIDy();
MatrizIDz = GetMatrizIDz();
AddListaChegadaVisitados(QuadricopterHandle);
while(GetListaVisitados(1)~=QuadricopterHandle)
    %espera chegar sua vez
end
tam = size(MatrizIDx);
n = tam(2);
existe = 0;
for i=1:n
    if (MatrizIDx(i) == xyz(1) && MatrizIDy(i) == xyz(2) && MatrizIDz(i) == xyz(3))
        existe = i;
        break;
    end
end
if (existe==0)
    id = geraID(QuadricopterHandle);
    SalvaMatrizIDxyz(xyz(1),xyz(2),xyz(3),id);
    SalvarMatrizAdjacencia(id,id,0);
    AddVerticesVisitadosID(id);
else
    id = existe;
    AddVerticesVisitadosID(id);
end
rtn = id;
RemoverListaVisitados(1);
End

----------------------------------------
function rtn = BuscarIDporXYZ(xyz)
MatrizIDx = GetMatrizIDx();
MatrizIDy = GetMatrizIDy();
MatrizIDz = GetMatrizIDz();
dxyz = GetDxDyDz();
Dx = dxyz(1);
Dy = dxyz(2);
Dz = dxyz(3);
Raioz = Dz/4;
Raio = GetRaio();
tam = size(MatrizIDx);
id = 0;
xp = xyz(1);
yp = xyz(2);
zp = xyz(3);
for i=1:tam(2)
    if (MatrizIDx(i)>xp)
        erroX = abs(MatrizIDx(i)-xp);
    else
        erroX = abs(xp-MatrizIDx(i));
    end
    if (MatrizIDy(i)>yp)
        erroY = abs(MatrizIDy(i)-yp);
    else
        erroY = abs(yp-MatrizIDy(i));
    end
    if (MatrizIDz(i)>zp)
        erroZ = abs(MatrizIDz(i)-zp);
    else
        erroZ = abs(zp-MatrizIDz(i));
    end
    if(erroX<Raio && erroY<Raio && erroZ<Raioz)
        id = i;
        break;
    end
end
rtn = id;
end
```

```
----------------------------------------
function CapturaPonto3D(KinectHandle, QuadricopterHandle, clientID, port,
idOrigem,angle)

vrep=remApi('remoteApi');
vrep.simxFinish(-1);
clientID=vrep.simxStart('127.0.0.1',port,true,false,0,5);

%eulerQuad1 contem os angulos de Euler dos eixos x,y,z
[resOrientation, eulerUAV] = vrep.simxGetObjectOrientation(clientID, QuadricopterHandle,
-1, vrep.simx_opmode_streaming);
a = now;
saiu = 0;
while (eulerUAV(3)==0)%espera o inicio de leituras diferentes de zero
    [resOrientation, eulerUAV] = vrep.simxGetObjectOrientation(clientID,
QuadricopterHandle, -1, vrep.simx_opmode_streaming);
    b = (now - a)*3600;
    if(b>1)
        saiu = 1;
        break;
    end
end
if (saiu == 1)
    CapturaPonto3D(KinectHandle, QuadricopterHandle, clientID, port, idOrigem,angle);
else
    % buffer contem a leitrua do sensor RGB-D, enquanto resolution contem
    % a resolucao da leitura RGB-D EX.: 64 linhas por 64 colunas
    filename = [num2str(idOrigem) '_honeycomb_3dread.mat'];
    load(filename);
    id_honeycomb_3dread;
    buffer =  [];
    [returnCode, resolution, buffer] = vrep.simxGetVisionSensorDepthBuffer2(clientID,
KinectHandle, vrep.simx_opmode_streaming);
    a = now;
    saiu = 0;
    while (resolution(1)==0)%espera o inicio de leituras diferentes de zero
        [returnCode, resolution, buffer] =
vrep.simxGetVisionSensorDepthBuffer2(clientID, KinectHandle, vrep.simx_opmode_buffer);
        b = (now - a)*3600;
        if(b>1)
            saiu = 1;
            break;
        end
    end
    if (saiu == 1)
        CapturaPonto3D(KinectHandle, QuadricopterHandle, clientID, port,
idOrigem,angle);
    else
        if(angle == (3*pi/6))
            % durante a primeira leitura armazena a resolucao do rgbd
            id_honeycomb_3dread.resolucao_x = resolution(1);
            id_honeycomb_3dread.resolucao_y = resolution(2);
            id_honeycomb_3dread.leitura_3pi_6 = buffer;
            id_honeycomb_3dread.angulo_3pi_6 = eulerUAV;
        elseif(angle == (pi/6))
            id_honeycomb_3dread.resolucao_x = resolution(1);
            id_honeycomb_3dread.resolucao_y = resolution(2);
            id_honeycomb_3dread.leitura_pi_6 = buffer;
            id_honeycomb_3dread.angulo_pi_6 = eulerUAV;
        elseif(angle == (-pi/6))
            id_honeycomb_3dread.resolucao_x = resolution(1);
            id_honeycomb_3dread.resolucao_y = resolution(2);
            id_honeycomb_3dread.leitura_menospi_6 = buffer;
            id_honeycomb_3dread.angulo_menospi_6 = eulerUAV;
        elseif(angle == (-3*pi/6))
            id_honeycomb_3dread.resolucao_x = resolution(1);
            id_honeycomb_3dread.resolucao_y = resolution(2);
            id_honeycomb_3dread.leitura_menos3pi_6 = buffer;
            id_honeycomb_3dread.angulo_menos3pi_6 = eulerUAV;
        elseif(angle == (-5*pi/6))
            id_honeycomb_3dread.resolucao_x = resolution(1);
            id_honeycomb_3dread.resolucao_y = resolution(2);
            id_honeycomb_3dread.leitura_menos5pi_6 = buffer;
            id_honeycomb_3dread.angulo_menos5pi_6 = eulerUAV;
        elseif(angle == (5*pi/6))
            id_honeycomb_3dread.resolucao_x = resolution(1);
```

```
            id_honeycomb_3dread.resolucao_y = resolution(2);
            id_honeycomb_3dread.leitura_5pi_6 = buffer;
            id_honeycomb_3dread.angulo_5pi_6 = eulerUAV;
        end
        save(filename,'id_honeycomb_3dread', '-v6');
    end
end
vrep.simxFinish(clientID);  % fecha a comunicacao
vrep.delete();              % destroi os elementos
end
```

```
----------------------------------------
function rtn =
DeslocarTrajetoria(id,idOrigemHoneycomb,QuadricopterHandle,Quadricopter_targetHandle,por
t)
vrep=remApi('remoteApi');
vrep.simxFinish(-1);
clientID=vrep.simxStart('127.0.0.1',port,true,false,0,5);
% Caso retorne algum ID VALIDO, TRABALHAR, SENAO NAO FAZ NADA
try
    if(id==0)
        proximo = 0;
        filename = [num2str(QuadricopterHandle) 'proximo.mat'];
        save(filename,'proximo','-v6');

        [cont, lista, tamanho] =
ObterGrauAdjacencia(QuadricopterHandle,idOrigemHoneycomb,0);
        clt.cont = cont;
        clt.lista = lista;
        clt.tamanho = tamanho;
        clt.caminho = [];
        filename = [num2str(QuadricopterHandle) 'clt.mat'];
        save(filename, 'clt','-v6');
        % verficar se n„o est· bloqueando alguÈm. Se sim, buscar resolver o
        % caminho, se nao nada a fazer ocupado = 0;
        lista = GetListaQuadricopterHandle();
        idx = 0;
        travando = 0;
        pause(1);
        for i=1:length(lista)
            filename = [num2str(lista(i)) 'proximo.mat'];
            if exist(filename,'file')
                load(filename,'proximo');
                if (proximo==idOrigemHoneycomb)
                    travando = 1;
                    idx = lista(i);
                    break;
                end
            end
        end
        if(travando==1)
            [cont, lista, tamanho] =
ObterGrauAdjacencia(QuadricopterHandle,idOrigemHoneycomb,idx);
            clt.cont = cont;
            clt.lista = lista;
            clt.tamanho = tamanho;
            clt.caminho=[];
            filename = [num2str(QuadricopterHandle) 'clt.mat'];
            save(filename, 'clt','-v6');
            mycont = cont;
            mylista = lista;
            mytamanho = tamanho;
            mycaminho = [];
            flag = 0;
            while(flag==0)
                filename = [num2str(idx) 'clt.mat'];
                if exist(filename,'file')
                    load(filename);
                    flag = 1;
                end
            end
            if(mytamanho > clt.tamanho)
                %deve fazer o deslocamento
                novaPosicao = ObterNovaPosicao(mylista,clt.caminho);

idOrigemHoneycomb=DeslocarTrajetoria(novaPosicao,idOrigemHoneycomb,QuadricopterHandle,Qu
adricopter_targetHandle,port);
                pause(randi(10));
            else
                if(mytamanho > 1)
                    novaPosicao = ObterNovaPosicao(mylista,clt.caminho);

idOrigemHoneycomb=DeslocarTrajetoria(novaPosicao,idOrigemHoneycomb,QuadricopterHandle,Qu
adricopter_targetHandle,port);
                else
                    novaPosicao = ObterNovaPosicao(clt.lista,clt.caminho);
```

```
idOrigemHoneycomb=DeslocarTrajetoria(novaPosicao,idOrigemHoneycomb,QuadricopterHandle,Qu
adricopter_targetHandle,port);
                    pause(randi(10));
                end
            end
            rtn = idOrigemHoneycomb;
        else
            rtn = idOrigemHoneycomb;
        end
    else
        % Gerar um caminho para o ID nao visitado
        MatrizAdjacencia = GetMatrizAdjacencia();
        [custo caminho]=dijkstra(MatrizAdjacencia,idOrigemHoneycomb,id);
        origem = idOrigemHoneycomb;
        n = size(caminho);
        i=n(2)-1;
        p = 0;
        if(i>0)
            p = caminho(i);
        end
        [cont, lista, tamanho] =
ObterGrauAdjacencia(QuadricopterHandle,idOrigemHoneycomb,p);
        clt.cont = cont;
        clt.lista = lista;
        clt.tamanho = tamanho;
        clt.caminho = caminho;
        filename = [num2str(QuadricopterHandle) 'clt.mat'];
        save(filename, 'clt','-v6');
        while(i~=0)
            %     MatrizAdjacencia = GetMatrizAdjacencia();
            %salva o prÛximo caminho a ser explorado para detectar
            %deadlock
            n = size(caminho);
            proximo = caminho(i);
            filename = [num2str(QuadricopterHandle) 'proximo.mat'];
            save(filename,'proximo','-v6');
            [cont, lista, tamanho] =
ObterGrauAdjacencia(QuadricopterHandle,idOrigemHoneycomb,proximo);
            clt.cont = cont;
            clt.lista = lista;
            clt.tamanho = tamanho;
            clt.caminho =  caminho;
            filename = [num2str(QuadricopterHandle) 'clt.mat'];
            save(filename, 'clt','-v6');
            %deslocar-se para o ID nao visitado
            pos = caminho(i);
            [resp, idx] = EstaOcupado(QuadricopterHandle,pos);
            if(resp==1)
                bloqueio = 0;
                filename = [num2str(QuadricopterHandle) 'uavbloqueio.mat'];
                if exist(filename,'file')
                    load(filename);

                end
                bloqueio  = bloqueio + 1;
                save(filename, 'bloqueio','-v6');
                while(resp==1)
                    pause(1);
                    filename = [num2str(idx) 'proximo.mat'];
                    load(filename,'proximo');
                    origem = GetPosicaoUAV(QuadricopterHandle);
                    if(proximo == origem)
                        %detectou deadlock
                        [cont, lista, tamanho] =
ObterGrauAdjacencia(QuadricopterHandle,idOrigemHoneycomb,pos);
                        clt.cont = cont;
                        clt.lista = lista;
                        clt.tamanho = tamanho;
                        clt.caminho = caminho;
                        filename = [num2str(QuadricopterHandle) 'clt.mat'];
                        save(filename, 'clt','-v6');
                        mycont = cont;
                        mytamanho = tamanho;
                        mycaminho = caminho;
                        flag = 0;
                        while(flag==0)
```

```
                            filename = [num2str(idx) 'clt.mat'];
                            if exist(filename,'file')
                                load(filename);
                                flag = 1;
                            end
                        end
                        if(mytamanho > clt.tamanho)
                            %deve fazer o deslocamento
                            novaPosicao = ObterNovaPosicao(lista,clt.caminho);

idOrigemHoneycomb=DeslocarTrajetoria(novaPosicao,idOrigemHoneycomb,QuadricopterHandle,Qu
adricopter_targetHandle,port);
                            %quando terminar, retornar um novo idOrigemHoneycomb
                            %para calcular um novo caminho
                            pause(randi(10));
                            MatrizAdjacencia = GetMatrizAdjacencia();
                            [custo
caminho]=dijkstra(MatrizAdjacencia,idOrigemHoneycomb,id);
                            n = size(caminho);
                            i=n(2)-1;
                            pos = caminho(i);
                            proximo = caminho(i);
                            origem = idOrigemHoneycomb;
                            [cont, lista, tamanho] =
ObterGrauAdjacencia(QuadricopterHandle,idOrigemHoneycomb,pos);
                            clt.cont = cont;
                            clt.lista = lista;
                            clt.tamanho = tamanho;
                            clt.caminho = caminho;
                            filename = [num2str(QuadricopterHandle) 'clt.mat'];
                            save(filename, 'clt','-v6');
                            filename = [num2str(QuadricopterHandle) 'proximo.mat'];
                            save(filename,'proximo','-v6');
                            [resp, idx] = EstaOcupado(QuadricopterHandle,pos);
                        else
                            if(mytamanho == clt.tamanho)
                                if(QuadricopterHandle>idx)
                                    %deve fazer o deslocamento
                                    %deslocar para a ˙ltima posicao da lista
                                    novaPosicao = ObterNovaPosicao(lista,clt.caminho);

idOrigemHoneycomb=DeslocarTrajetoria(novaPosicao,idOrigemHoneycomb,QuadricopterHandle,Qu
adricopter_targetHandle,port);
                                    %quando terminar, retornar um novo idOrigemHoneycomb
                                    %para calcular um novo caminho
                                    pause(randi(10));
                                    MatrizAdjacencia = GetMatrizAdjacencia();
                                    [custo
caminho]=dijkstra(MatrizAdjacencia,idOrigemHoneycomb,id);
                                    n = size(caminho);
                                    i=n(2)-1;
                                    pos = caminho(i);
                                    proximo = caminho(i);
                                    origem = idOrigemHoneycomb;
                                    [cont, lista, tamanho] =
ObterGrauAdjacencia(QuadricopterHandle,idOrigemHoneycomb,pos);
                                    clt.cont = cont;
                                    clt.lista = lista;
                                    clt.tamanho = tamanho;
                                    clt.caminho = caminho;
                                    filename = [num2str(QuadricopterHandle) 'clt.mat'];
                                    save(filename, 'clt','-v6');
                                    filename = [num2str(QuadricopterHandle)
'proximo.mat'];
                                    save(filename,'proximo','-v6');
                                    [resp, idx] = EstaOcupado(QuadricopterHandle,pos);
                                else
                                    [resp, idx] = EstaOcupado(QuadricopterHandle,pos);
                                end
                            else
                                [resp, idx] = EstaOcupado(QuadricopterHandle,pos);
                            end
                        end
                    else
                        if(proximo==0)
                            [cont, lista, tamanho] =
ObterGrauAdjacencia(QuadricopterHandle,idOrigemHoneycomb,pos);
```

```
                            clt.cont = cont;
                            clt.lista = lista;
                            clt.tamanho = tamanho;
                            clt.caminho = caminho;
                            filename = [num2str(QuadricopterHandle) 'clt.mat'];
                            save(filename, 'clt','-v6');
                            mycont = cont;
                            mytamanho = tamanho;
                            mycaminho = caminho;
                            flag = 0;
                            while(flag==0)
                                filename = [num2str(idx) 'clt.mat'];
                                if exist(filename,'file')
                                    load(filename);
                                    flag = 1;
                                end
                            end
                            if(mytamanho > clt.tamanho)
                                %deve fazer o deslocamento
                                novaPosicao = ObterNovaPosicao(lista,clt.caminho);

idOrigemHoneycomb=DeslocarTrajetoria(novaPosicao,idOrigemHoneycomb,QuadricopterHandle,Qu
adricopter_targetHandle,port);%quando terminar, retornar um novo idOrigemHoneycomb
                                %para calcular um novo caminho
                                pause(randi(10));
                                MatrizAdjacencia = GetMatrizAdjacencia();
                                [custo
caminho]=dijkstra(MatrizAdjacencia,idOrigemHoneycomb,id);
                                n = size(caminho);
                                i=n(2)-1;
                                pos = caminho(i);
                                proximo = caminho(i);
                                origem = idOrigemHoneycomb;
                                [cont, lista, tamanho] =
ObterGrauAdjacencia(QuadricopterHandle,idOrigemHoneycomb,pos);
                                clt.cont = cont;
                                clt.lista = lista;
                                clt.tamanho = tamanho;
                                clt.caminho = caminho;
                                filename = [num2str(QuadricopterHandle) 'clt.mat'];
                                save(filename, 'clt','-v6');
                                filename = [num2str(QuadricopterHandle) 'proximo.mat'];
                                save(filename,'proximo','-v6');
                                [resp, idx] = EstaOcupado(QuadricopterHandle,pos);
                            else
                                if(mytamanho == clt.tamanho)
                                    if(QuadricopterHandle>idx)
                                        %deve fazer o deslocamento
                                        novaPosicao =
ObterNovaPosicao(lista,clt.caminho);

idOrigemHoneycomb=DeslocarTrajetoria(novaPosicao,idOrigemHoneycomb,QuadricopterHandle,Qu
adricopter_targetHandle,port);%quando terminar, retornar um novo idOrigemHoneycomb
                                        %para calcular um novo caminho
                                        pause(randi(10));
                                        MatrizAdjacencia = GetMatrizAdjacencia();
                                        [custo
caminho]=dijkstra(MatrizAdjacencia,idOrigemHoneycomb,id);
                                        n = size(caminho);
                                        i=n(2)-1;
                                        pos = caminho(i);
                                        proximo = caminho(i);
                                        origem = idOrigemHoneycomb;
                                        [cont, lista, tamanho] =
ObterGrauAdjacencia(QuadricopterHandle,idOrigemHoneycomb,pos);
                                        clt.cont = cont;
                                        clt.lista = lista;
                                        clt.tamanho = tamanho;
                                        clt.caminho = caminho;
                                        filename = [num2str(QuadricopterHandle)
'clt.mat'];
                                        save(filename, 'clt','-v6');
                                        filename = [num2str(QuadricopterHandle)
'proximo.mat'];
                                        save(filename,'proximo','-v6');
                                        [resp, idx] =
EstaOcupado(QuadricopterHandle,pos);
```

```
                                    else
                                        [resp, idx] =
EstaOcupado(QuadricopterHandle,pos);
                                    end
                                else
                                    [resp, idx] = EstaOcupado(QuadricopterHandle,pos);
                                end
                            end
                        else
                            posicao = GetPosicaoUAV(QuadricopterHandle);
                            [temCiclo, idx] = EstaEmCiclo(QuadricopterHandle,posicao);
                            if(temCiclo)
                                [cont, lista, tamanho] =
ObterGrauAdjacencia(QuadricopterHandle,idOrigemHoneycomb,pos);
                                if(length(lista)>0)
                                    novaPosicao = ObterNovaPosicao(lista,clt.caminho);

idOrigemHoneycomb=DeslocarTrajetoria(novaPosicao,idOrigemHoneycomb,QuadricopterHandle,Qu
adricopter_targetHandle,port);
                                    pause(randi(10));
                                end
                                MatrizAdjacencia = GetMatrizAdjacencia();
                                [custo
caminho]=dijkstra(MatrizAdjacencia,idOrigemHoneycomb,id);
                                n = size(caminho);
                                i=n(2)-1;
                                pos = caminho(i);
                                proximo = caminho(i);
                                origem = idOrigemHoneycomb;
                                [cont, lista, tamanho] =
ObterGrauAdjacencia(QuadricopterHandle,idOrigemHoneycomb,pos);
                                clt.cont = cont;
                                clt.lista = lista;
                                clt.tamanho = tamanho;
                                clt.caminho = caminho;
                                filename = [num2str(QuadricopterHandle) 'clt.mat'];
                                save(filename, 'clt','-v6');
                                filename = [num2str(QuadricopterHandle) 'proximo.mat'];
                                save(filename,'proximo','-v6');
                                [resp, idx] = EstaOcupado(QuadricopterHandle,pos);
                            else
                                MatrizAdjacencia = GetMatrizAdjacencia();
                                %gerar novo caminho
                                tam = size(MatrizAdjacencia);
                                for j=1:tam(1)
                                    if(MatrizAdjacencia(j,pos)==1)
                                        MatrizAdjacencia(j,pos)=5;
                                        MatrizAdjacencia(pos,j)=5;
                                    end
                                end
                                [custo
caminho]=dijkstra(MatrizAdjacencia,idOrigemHoneycomb,id);
                                n = size(caminho);
                                i=n(2)-1;
                                pos = caminho(i);
                                proximo = caminho(i);
                                origem = idOrigemHoneycomb;
                                [cont, lista, tamanho] =
ObterGrauAdjacencia(QuadricopterHandle,idOrigemHoneycomb,pos);
                                clt.cont = cont;
                                clt.lista = lista;
                                clt.tamanho = tamanho;
                                clt.caminho = caminho;
                                filename = [num2str(QuadricopterHandle) 'clt.mat'];
                                save(filename, 'clt','-v6');
                                filename = [num2str(QuadricopterHandle) 'proximo.mat'];
                                save(filename,'proximo','-v6');
                                [resp, idx] = EstaOcupado(QuadricopterHandle,pos);
                            end
                        end
                    end

                end
            end
            ocupou = MarcarOcupado(QuadricopterHandle,pos);
            if(ocupou == 1)
                MatrizIDx = GetMatrizIDx();
```

```
                MatrizIDy = GetMatrizIDy();
                MatrizIDz = GetMatrizIDz();
                xyz(1) = MatrizIDx(pos);
                xyz(2) = MatrizIDy(pos);
                xyz(3) = MatrizIDz(pos);

DeslocarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,xyz,port);
                LiberaPosicaoOldUAV(QuadricopterHandle);
                pause(randi(20));
                %SALVAR O TRAJETO DE EXPLORACAO DO UAV
                filename = [num2str(QuadricopterHandle) 'listatrajeto.dat'];
                fileID = fopen(filename,'a');
                fwrite(fileID,pos);
                fclose(fileID);
                idOrigemHoneycomb = pos;
                origem = idOrigemHoneycomb;
                MatrizAdjacencia = GetMatrizAdjacencia();
                i=i-1;
            else

idOrigemHoneycomb=DeslocarTrajetoria(id,idOrigemHoneycomb,QuadricopterHandle,Quadricopte
r_targetHandle,port);
                pause(randi(10));
                origem = idOrigemHoneycomb;
            end
        end
    end
catch

idOrigemHoneycomb=DeslocarTrajetoria(id,idOrigemHoneycomb,QuadricopterHandle,Quadricopte
r_targetHandle,port);
    pause(5);
    origem = idOrigemHoneycomb;
end
rtn = idOrigemHoneycomb;
end
```

```
----------------------------------------
function [rtn, id] = EstaOcupado(QuadricopterHandle, posicao)
AddListaEstaOcupado(QuadricopterHandle);
nid = GetListaEstaOcupado(1);
while(nid~=QuadricopterHandle)
    if(nid==0)
        AddListaEstaOcupado(QuadricopterHandle);
    end
    nid = GetListaEstaOcupado(1);
    %espera chegar sua vez
end
try
    ocupado = 0;
    lista = GetListaQuadricopterHandle();
    id = 0;
    for i=1:length(lista)
        filename = [num2str(lista(i)) '.mat'];
        load(filename,'pos');
        if (pos==posicao) && (lista(i)~=QuadricopterHandle)
            ocupado = 1;
            id = lista(i);
        else
            filename = [num2str(lista(i)) 'old.mat'];
            load(filename);
            if (posOld==posicao) && (lista(i)~=QuadricopterHandle)
                ocupado = 1;
                id = lista(i);
            end
        end
    end
    rtn = ocupado;
catch
    rtn = -1;
    id = -1;
end
RemoverListaEstaOcupado(QuadricopterHandle,1);
End

----------------------------------------
function rtn = ExisteVerticeNaoVisitado()
fileID = fopen('verticesNaoVisitadosID.dat');
lista = fread(fileID);
fclose(fileID);

existe = 0;

if (length(lista)>0)
    existe = 1;
end

rtn = existe;

end
```

```
----------------------------------------
function rtn = Explorar(QuadricopterHandle, Quadricopter_targetHandle,
Proximity_sensorHandle, Proximity_sensorUpHandle,Proximity_sensorDownHandle,
KinectHandle,Kinec_rgb_tHandle,VisionSensortHandle, port)

MatrizIDx = GetMatrizIDx();
MatrizIDy = GetMatrizIDy();
MatrizIDz = GetMatrizIDz();

vrep=remApi('remoteApi');
vrep.simxFinish(-1);
clientID=vrep.simxStart('127.0.0.1',port,true,false,0,2);

%posicionar-se em um espaco visitado
%RECUPERAR XYZ A PARTIR DO IDORIGEMHONEYCOMB PARA ALIMENTAR O DeslocarUAV
a = now;
saiu = 0;
[resQuad1, posQuad1] = vrep.simxGetObjectPosition(clientID, QuadricopterHandle, -1,
vrep.simx_opmode_streaming);
while (posQuad1(3)==0)
    [resQuad1, posQuad1] = vrep.simxGetObjectPosition(clientID, QuadricopterHandle, -1,
vrep.simx_opmode_buffer);
    b = (now - a)*3600;
    if(b>1)
        saiu = 1;
        break;
    end
end
if (saiu == 1)
    rtn = Explorar(QuadricopterHandle, Quadricopter_targetHandle,
Proximity_sensorHandle, Proximity_sensorUpHandle,Proximity_sensorDownHandle,
KinectHandle,Kinec_rgb_tHandle,VisionSensortHandle, port);
else
    dx = posQuad1(1);
    dy = posQuad1(2);
    dz = posQuad1(3);
    pos=[dx dy dz];
    idOrigemHoneycomb = BuscarIDporXYZ(pos);
    if (idOrigemHoneycomb==0)
        x = MatrizIDx(1);
        y = MatrizIDy(1);
        z = MatrizIDz(1);
        idOrigemHoneycomb=1;
    else
        x = MatrizIDx(idOrigemHoneycomb);
        y = MatrizIDy(idOrigemHoneycomb);
        z = MatrizIDz(idOrigemHoneycomb);
    end
    pos = idOrigemHoneycomb;
    [resp, idx]  = EstaOcupado(QuadricopterHandle,pos);
    while(resp~=0)
        % Nada a fazer. Aguardar liberar caminho para iniciar exploracao
        [resp, idx] = EstaOcupado(QuadricopterHandle,pos);
        if(resp==-1)
            %caso ocorra excecao (acessando quando o arquivo est· sendo
            %atualizado
            [resp, idx] = EstaOcupado(QuadricopterHandle,pos);
        end
    end
    marcouOcupado = MarcarOcupado(QuadricopterHandle,pos);
    if (marcouOcupado==1)
        xyz = [x y z];
        DeslocarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,xyz,port);
        LiberaPosicaoOldUAV(QuadricopterHandle);
        % Obtem a lista de todos os UAVs existentes no ambiente
        listaQuadricopterHandle = GetListaQuadricopterHandle();
        %Obtem a posicao id do UAV
        posicao = GetPosicaoUAV(QuadricopterHandle);
        x = MatrizIDx(posicao);
        y = MatrizIDy(posicao);
        z = MatrizIDz(posicao);

        %busca o id a ser explorado a partir de algum algoritmo: FIFO,
        %DISTANCIA EUCLIDIANA OU DISTANCIA EUCLIDIANA DISTRIBUIDA - UTFPR EM
        %PARCERIA COM IPB - PORTUGAL
        id = MarcarNaoVisitado(QuadricopterHandle);
```

```
        if (id~=0) % Caso retorne algum ID VALIDO, TRABALHAR, SENAO NAO FAZ NADA

            % deslocar o UAV para o HONEYCOMB definido para exploracao a
            % partir do id
            pos =
DeslocarTrajetoria(id,idOrigemHoneycomb,QuadricopterHandle,Quadricopter_targetHandle,por
t);
            Raio = GetRaio();
            dxyz = GetDxDyDz();

            Dz = dxyz(3);
            Raioz = Dz/4;
            MatrizIDx = GetMatrizIDx();
            MatrizIDy = GetMatrizIDy();
            MatrizIDz = GetMatrizIDz();
            xyz(1) = MatrizIDx(pos);
            xyz(2) = MatrizIDy(pos);
            xyz(3) = MatrizIDz(pos);

            %visitar o ID e adicionar as adjac?ncias
            fprintf('Adicionando o Vertice visitado\n');
            idOrigem = AdicionarVerticeVisitado(xyz,QuadricopterHandle);%adiciona o
ponto central do honeycomb a estrutura de vertices visitados

            %apos deslocar o UAV para o local de exploracao, deve-se fazer o
            %giro no honeycomb afim de mapear todas as direcoes
            %rotacionar para norte 3pi/6




            filename = [num2str(idOrigem) '_honeycomb_3dread.mat'];
            id_honeycomb_3dread.posUAV = xyz;
            id_honeycomb_3dread.leitura_3pi_6 = [];
            id_honeycomb_3dread.angulo_3pi_6 = [];
            id_honeycomb_3dread.leitura_pi_6 = [];
            id_honeycomb_3dread.angulo_pi_6 = [];
            id_honeycomb_3dread.leitura_menospi_6 = [];
            id_honeycomb_3dread.angulo_menospi_6 = [];
            id_honeycomb_3dread.leitura_menos3pi_6 = [];
            id_honeycomb_3dread.angulo_menos3pi_6 = [];
            id_honeycomb_3dread.leitura_menos5pi_6 = [];
            id_honeycomb_3dread.angulo_menos5pi_6 = [];
            id_honeycomb_3dread.leitura_5pi_6 = [];
            id_honeycomb_3dread.angulo_5pi_6 = [];
            id_honeycomb_3dread.resolucao_x = 0;
            id_honeycomb_3dread.resolucao_y = 0;
            save(filename,'id_honeycomb_3dread');

            angle = 3*pi/6;

            dy = 2*double(Raio)*sin(double(angle))/sin(degtorad(90));
            dx = 2*double(Raio)*sin(double(degtorad(180-90-
radtodeg(angle))))/sin(degtorad(90));
            xp = xyz(1) + dx;
            yp = xyz(2) + dy;
            zp = xyz(3);
            idExistente = GetIDbyXYZ(xp,yp,zp);
            matAdj = GetMatrizAdjacencia();
            if (idExistente==0 | (matAdj(idOrigem,idExistente)==0))
                fprintf('Rotacionando o UAV a norte 3pi/6\n');

RotacionarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,angle,port);

DeslocarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,xyz,port);
                fprintf('Capturando a leitura RGBD e gerando os pontos 3D\n');
                CapturaPonto3D(KinectHandle, QuadricopterHandle, clientID,
port,idOrigem,angle);
                fprintf('Verificando as adjacencias do vertice\n');

VerificaAdjacencias(clientID,QuadricopterHandle,Proximity_sensorHandle,idOrigem,angle,xy
z,port);
```

```
VerificaTemperatura(clientID,QuadricopterHandle,Kinec_rgb_tHandle,VisionSensortHandle,
idOrigem, angle, xyz,port);
            end

            %rotacionar para nordeste pi/6
            angle = pi/6;
            dy = 2*double(Raio)*sin(double(angle))/sin(degtorad(90));
            dx = 2*double(Raio)*sin(double(degtorad(180-90-
radtodeg(angle))))/sin(degtorad(90));
            xp = xyz(1) + dx;
            yp = xyz(2) + dy;
            zp = xyz(3);
            idExistente = GetIDbyXYZ(xp,yp,zp);
            matAdj = GetMatrizAdjacencia();
            if (idExistente==0 | (matAdj(idOrigem,idExistente)==0))
                fprintf('Rotacionando o UAV a nordeste pi/6\n');

RotacionarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,angle,port);

DeslocarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,xyz,port);
                fprintf('Capturando a leitura RGBD e gerando os pontos 3D\n');
                CapturaPonto3D(KinectHandle, QuadricopterHandle, clientID,
port,idOrigem,angle);
                fprintf('Verificando as adjacencias do vertice\n');

VerificaAdjacencias(clientID,QuadricopterHandle,Proximity_sensorHandle,idOrigem,angle,xy
z,port);

VerificaTemperatura(clientID,QuadricopterHandle,Kinec_rgb_tHandle,VisionSensortHandle,
idOrigem, angle, xyz,port);
            end
            %rotacionar para sudeste -pi/6
            angle = -pi/6;
            dy = 2*double(Raio)*sin(double(angle))/sin(degtorad(90));
            dx = 2*double(Raio)*sin(double(degtorad(180-90-
radtodeg(angle))))/sin(degtorad(90));
            xp = xyz(1) + dx;
            yp = xyz(2) + dy;
            zp = xyz(3);
            idExistente = GetIDbyXYZ(xp,yp,zp);
            matAdj = GetMatrizAdjacencia();
            if (idExistente==0 | (matAdj(idOrigem,idExistente)==0))
                fprintf('Rotacionando o UAV a sudeste -pi/6\n');

RotacionarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,angle,port);

DeslocarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,xyz,port);
                fprintf('Capturando a leitura RGBD e gerando os pontos 3D\n');
                CapturaPonto3D(KinectHandle, QuadricopterHandle, clientID,
port,idOrigem,angle);
                fprintf('Verificando as adjacencias do vertice\n');

VerificaAdjacencias(clientID,QuadricopterHandle,Proximity_sensorHandle,idOrigem,angle,xy
z,port);

VerificaTemperatura(clientID,QuadricopterHandle,Kinec_rgb_tHandle,VisionSensortHandle,
idOrigem, angle, xyz,port);
            end

            %rotacionar para sul -3pi/6
            angle = -3*pi/6;
            dy = 2*double(Raio)*sin(double(angle))/sin(degtorad(90));
            dx = 2*double(Raio)*sin(double(degtorad(180-90-
radtodeg(angle))))/sin(degtorad(90));
            xp = xyz(1) + dx;
            yp = xyz(2) + dy;
            zp = xyz(3);
            idExistente = GetIDbyXYZ(xp,yp,zp);
            matAdj = GetMatrizAdjacencia();
            if (idExistente==0 | (matAdj(idOrigem,idExistente)==0))
                fprintf('Rotacionando o UAV a sul -3pi/6\n');

RotacionarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,angle,port);

DeslocarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,xyz,port);
                fprintf('Capturando a leitura RGBD e gerando os pontos 3D\n');
```

```
                CapturaPonto3D(KinectHandle, QuadricopterHandle, clientID,
port,idOrigem,angle);
                fprintf('Verificando as adjacencias do vertice\n');

VerificaAdjacencias(clientID,QuadricopterHandle,Proximity_sensorHandle,idOrigem,angle,xy
z,port);

VerificaTemperatura(clientID,QuadricopterHandle,Kinec_rgb_tHandle,VisionSensortHandle,
idOrigem, angle, xyz,port);
            end

            %rotacionar para sudoeste -5pi/6
            angle = -5*pi/6;
            dy = 2*double(Raio)*sin(double(angle))/sin(degtorad(90));
            dx = 2*double(Raio)*sin(double(degtorad(180-90-
radtodeg(angle))))/sin(degtorad(90));
            xp = xyz(1) + dx;
            yp = xyz(2) + dy;
            zp = xyz(3);
            idExistente = GetIDbyXYZ(xp,yp,zp);
            matAdj = GetMatrizAdjacencia();
            if (idExistente==0 | (matAdj(idOrigem,idExistente)==0))
                fprintf('Rotacionando o UAV a sudoeste -5pi/6\n');

RotacionarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,angle,port);

DeslocarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,xyz,port);
                fprintf('Capturando a leitura RGBD e gerando os pontos 3D\n');
                CapturaPonto3D(KinectHandle, QuadricopterHandle, clientID,
port,idOrigem,angle);
                fprintf('Verificando as adjacencias do vertice\n');

VerificaAdjacencias(clientID,QuadricopterHandle,Proximity_sensorHandle,idOrigem,angle,xy
z,port);

VerificaTemperatura(clientID,QuadricopterHandle,Kinec_rgb_tHandle,VisionSensortHandle,
idOrigem, angle, xyz,port);
            end

            % rotacionar para noroeste 5pi/6
            angle = 5*pi/6;
            dy = 2*double(Raio)*sin(double(angle))/sin(degtorad(90));
            dx = 2*double(Raio)*sin(double(degtorad(180-90-
radtodeg(angle))))/sin(degtorad(90));
            xp = xyz(1) + dx;
            yp = xyz(2) + dy;
            zp = xyz(3);
            idExistente = GetIDbyXYZ(xp,yp,zp);
            matAdj = GetMatrizAdjacencia();
            if (idExistente==0 | (matAdj(idOrigem,idExistente)==0))
                fprintf('Rotacionando o UAV a noroeste 5pi/6\n');

RotacionarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,angle,port);

DeslocarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,xyz,port);
                fprintf('Capturando a leitura RGBD e gerando os pontos 3D\n');
                CapturaPonto3D(KinectHandle, QuadricopterHandle, clientID,
port,idOrigem,angle);
                fprintf('Verificando as adjacencias do vertice\n');

VerificaAdjacencias(clientID,QuadricopterHandle,Proximity_sensorHandle,idOrigem,angle,xy
z,port);

VerificaTemperatura(clientID,QuadricopterHandle,Kinec_rgb_tHandle,VisionSensortHandle,
idOrigem, angle, xyz,port);
            end

            % %Verificar adjacencia superior

VerificaAdjacenciasSuperior(clientID,QuadricopterHandle,Proximity_sensorUpHandle,idOrige
m,xyz,port);
            % %Verificar adjacencia inferior

VerificaAdjacenciasInferior(clientID,QuadricopterHandle,Proximity_sensorDownHandle,idOri
gem,xyz,port);
            % ADICIONAR O ID A LISTA DE HONEYCOMBS COM LEITURA RGBD REALIZADOS;
            fileID = fopen('listaIDHoneycombLeituraRGBDpre.dat','a');
```

```
            fwrite(fileID,idOrigem);
            fclose(fileID);
            RemoverNaoVisitado(id,QuadricopterHandle);

            rtn = idOrigem;

        else

posicao=DeslocarTrajetoria(id,idOrigemHoneycomb,QuadricopterHandle,Quadricopter_targetHa
ndle,port);
            rtn = posicao;
        end
    else
        rtn =Explorar(QuadricopterHandle, Quadricopter_targetHandle,
Proximity_sensorHandle, Proximity_sensorUpHandle,Proximity_sensorDownHandle,
KinectHandle,Kinec_rgb_tHandle,VisionSensortHandle, port);
    end
end
end
```

```
----------------------------------------
function rtn = ExplorarInicial(QuadricopterHandle, Quadricopter_targetHandle,
Proximity_sensorHandle, Proximity_sensorUpHandle,Proximity_sensorDownHandle,
KinectHandle,Kinec_rgb_tHandle,VisionSensortHandle, port)

inicioUAV1 = now;
save('inicioUAV1','-v6');

vrep=remApi('remoteApi');
vrep.simxFinish(-1);
clientID=vrep.simxStart('127.0.0.1',port,true,false,0,5);

[resQuad1, posUAV] = vrep.simxGetObjectPosition(clientID, QuadricopterHandle, -1,
vrep.simx_opmode_streaming);
while (posUAV(3)==0)%espera o inicio de leituras diferentes de zero
    [resQuad1, posUAV] = vrep.simxGetObjectPosition(clientID, QuadricopterHandle, -1,
vrep.simx_opmode_buffer);
end

dxyz = GetDxDyDz(); %obtem os limites de exploracao

Dz = dxyz(3);
x = posUAV(1);
y = posUAV(2);
z = Dz/4;%posiciona o UAV em um espaco baseado no valor da altura de exploracao
xyz = [x y z];

vrep.simxFinish(clientID);  % fecha a comunicacao
vrep.delete();

DeslocarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,xyz,port);

%adiciona o ponto central do honeycomb a estrutura de vertices visitados
idOrigem = AdicionarVerticeVisitado(xyz,QuadricopterHandle);
pos = idOrigem;
MarcarOcupado(QuadricopterHandle,pos);
%SALVAR A ATRIBUI«√O DE EXPLORACAO DO UAV
filename = [num2str(QuadricopterHandle) 'listaexploracao.dat'];
fileID = fopen(filename,'a');
fwrite(fileID,pos);
fclose(fileID);

%SALVAR O TRAJETO DE EXPLORACAO DO UAV
filename = [num2str(QuadricopterHandle) 'listatrajeto.dat'];
fileID = fopen(filename,'a');
fwrite(fileID,pos);
fclose(fileID);


%giro no honeycomb afim de mapear todas as direcoes
%-------------------------------------------------------
%rotacionar para norte 3pi/6
angle = 3*pi/6;
fprintf('Rotacionando o UAV a norte 3pi/6\n');
RotacionarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,angle,port);
DeslocarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,xyz,port);
fprintf('Capturando a leitura RGBD e gerando os pontos 3D\n');

filename = [num2str(idOrigem) '_honeycomb_3dread.mat'];
id_honeycomb_3dread.posUAV = xyz;
id_honeycomb_3dread.leitura_3pi_6 = [];
id_honeycomb_3dread.angulo_3pi_6 = [];
id_honeycomb_3dread.leitura_pi_6 = [];
id_honeycomb_3dread.angulo_pi_6 = [];
id_honeycomb_3dread.leitura_menospi_6 = [];
id_honeycomb_3dread.angulo_menospi_6 = [];
id_honeycomb_3dread.leitura_menos3pi_6 = [];
id_honeycomb_3dread.angulo_menos3pi_6 = [];
id_honeycomb_3dread.leitura_menos5pi_6 = [];
id_honeycomb_3dread.angulo_menos5pi_6 = [];
id_honeycomb_3dread.leitura_5pi_6 = [];
id_honeycomb_3dread.angulo_5pi_6 = [];
id_honeycomb_3dread.resolucao_x = 0;
id_honeycomb_3dread.resolucao_y = 0;
save(filename,'id_honeycomb_3dread');
CapturaPonto3D(KinectHandle, QuadricopterHandle, clientID, port,idOrigem,angle);
```

```
fprintf('Verificando as adjacencias do vertice\n');
VerificaAdjacencias(clientID,QuadricopterHandle,Proximity_sensorHandle,idOrigem,angle,xy
z,port);% retorna o id da adjacencia
VerificaTemperatura(clientID,QuadricopterHandle,Kinec_rgb_tHandle,VisionSensortHandle,
idOrigem, angle, xyz,port);

%rotacionar para nordeste pi/6
angle = pi/6;
fprintf('Rotacionando o UAV a nordeste pi/6\n');
RotacionarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,angle,port);
DeslocarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,xyz,port);
fprintf('Capturando a leitura RGBD e gerando os pontos 3D\n');
CapturaPonto3D(KinectHandle, QuadricopterHandle, clientID, port,idOrigem,angle);
fprintf('Verificando as adjacencias do vertice\n');
VerificaAdjacencias(clientID,QuadricopterHandle,Proximity_sensorHandle,idOrigem,angle,xy
z,port);% retorna o id da adjacencia
VerificaTemperatura(clientID,QuadricopterHandle,Kinec_rgb_tHandle,VisionSensortHandle,
idOrigem, angle, xyz,port);

%rotacionar para sudeste -pi/6
angle = -pi/6;
fprintf('Rotacionando o UAV a sudeste -pi/6\n');
RotacionarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,angle,port);
DeslocarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,xyz,port);
fprintf('Capturando a leitura RGBD e gerando os pontos 3D\n');
CapturaPonto3D(KinectHandle, QuadricopterHandle, clientID, port,idOrigem,angle);
fprintf('Verificando as adjacencias do vertice\n');
VerificaAdjacencias(clientID,QuadricopterHandle,Proximity_sensorHandle,idOrigem,angle,xy
z,port);
VerificaTemperatura(clientID,QuadricopterHandle,Kinec_rgb_tHandle,VisionSensortHandle,
idOrigem, angle, xyz,port);

%rotacionar para sul -3pi/6
angle = -3*pi/6;
fprintf('Rotacionando o UAV a sul -3pi/6\n');
RotacionarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,angle,port);
DeslocarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,xyz,port);
fprintf('Capturando a leitura RGBD e gerando os pontos 3D\n');
CapturaPonto3D(KinectHandle, QuadricopterHandle, clientID, port,idOrigem,angle);
fprintf('Verificando as adjacencias do vertice\n');
VerificaAdjacencias(clientID,QuadricopterHandle,Proximity_sensorHandle,idOrigem,angle,xy
z,port);
VerificaTemperatura(clientID,QuadricopterHandle,Kinec_rgb_tHandle,VisionSensortHandle,
idOrigem, angle, xyz,port);

%rotacionar para sudoeste -5pi/6
angle = -5*pi/6;
fprintf('Rotacionando o UAV a sudoeste -5pi/6\n');
RotacionarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,angle,port);
DeslocarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,xyz,port);
fprintf('Capturando a leitura RGBD e gerando os pontos 3D\n');
CapturaPonto3D(KinectHandle, QuadricopterHandle, clientID, port,idOrigem,angle);
fprintf('Verificando as adjacencias do vertice\n');
VerificaAdjacencias(clientID,QuadricopterHandle,Proximity_sensorHandle,idOrigem,angle,xy
z,port);
VerificaTemperatura(clientID,QuadricopterHandle,Kinec_rgb_tHandle,VisionSensortHandle,
idOrigem, angle, xyz,port);

% rotacionar para noroeste 5pi/6
angle = 5*pi/6;
fprintf('Rotacionando o UAV a noroeste 5pi/6\n');
RotacionarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,angle,port);
DeslocarUAV(QuadricopterHandle,Quadricopter_targetHandle,clientID,xyz,port);
fprintf('Capturando a leitura RGBD e gerando os pontos 3D\n');
CapturaPonto3D(KinectHandle, QuadricopterHandle, clientID, port,idOrigem,angle);
fprintf('Verificando as adjacencias do vertice\n');
VerificaAdjacencias(clientID,QuadricopterHandle,Proximity_sensorHandle,idOrigem,angle,xy
z,port);
VerificaTemperatura(clientID,QuadricopterHandle,Kinec_rgb_tHandle,VisionSensortHandle,
idOrigem, angle, xyz,port);

% %Verificar adjacencia superior

VerificaAdjacenciasSuperior(clientID,QuadricopterHandle,Proximity_sensorUpHandle,idOrige
m,xyz,port);
%
% %Verificar adjacencia inferior
```

```
VerificaAdjacenciasInferior(clientID,QuadricopterHandle,Proximity_sensorDownHandle,idOri
gem,xyz,port);

% ADICIONAR O ID A LISTA DE HONEYCOMBS COM LEITURA RGBD REALIZADOS;
fileID = fopen('listaIDHoneycombLeituraRGBDpre.dat','a');
fwrite(fileID,idOrigem);
fclose(fileID);

RemoverNaoVisitado(idOrigem, QuadricopterHandle)

rtn = idOrigem;
end
```

```
----------------------------------------
function rtn = GetListaQuadricopterHandle()

load listaQuadricopterHandle.mat;
rtn = listaQuadricopterHandle;

end
```

```
----------------------------------------
function rtn = GetPosicaoUAV(QuadricopterHandle)

      filename = [num2str(QuadricopterHandle) '.mat'];
      leitura =   load(filename);
      rtn =  leitura.pos;

end
```

```
----------------------------------------
function ImprimirMapaCubos(idToPlot)

listaId = str2num(idToPlot);
tam = length(listaId);

if (tam==0 | listaId(1)==0)
    %gerar todos os Honeycombs
    fileID = fopen('listaIDHoneycombLeituraRGBDpre.dat');
    listaId = fread(fileID);

end

NXC = [];
NYC = [];
NZC = [];
CONT = [];

for i=1:length(listaId)
    filename2 = [num2str(listaId(i)) 'xc.mat'];
    if(isfile(filename2))
        load(filename2);
    end
    filename2 = [num2str(listaId(i)) 'yc.mat'];
    if(isfile(filename2))
        load(filename2);
    end
    filename2 = [num2str(listaId(i)) 'zc.mat'];
    if(isfile(filename2))
        load(filename2);
    end
    filename2 = [num2str(listaId(i)) 'cont.mat'];
    if(isfile(filename2))
        load(filename2);
    end

    NXC = [NXC xc];
    NYC = [NYC yc];
    NZC = [NZC zc];
    CONT = [CONT cont];

end
maior = max(CONT);
R = GetR();
figure('Name','3D Cube View', 'NumberTitle','off');
hold on;
    for j=1:length(NXC)
        xyz = [NXC(j) NYC(j) NZC(j)];
        PlotCube3D(xyz, R, maior,CONT(j));
        hold on;
    end

end
```

```
----------------------------------------
function LiberaPosicaoOldUAV(QuadricopterHandle)
posOld = -1;
filename = [num2str(QuadricopterHandle) 'old.mat'];
save(filename,'posOld');

end
```

```
----------------------------------------
function mapa()

delete *.mat;
delete *.dat;


Dx=200;
Dy=200;
Dz=2.5;
GravaDxDyDz(Dx,Dy,Dz); % LIMITA O ESPACO ONDE OS UAVS PODERAO ATUAR

Raio = 0.5;%SE ALTERAR O VALOR DO RAIO DO HONEYCOM, PRECISA RECONFIGURAR O SENSOR DE
PROXIMIDADE DO UAV NO VREP (EST? CONFIGURADO PARA DUAS VEZES O RAIO)
GravaRaio(Raio);


% cria a matriz de ID com zeros
matrizidx = [];
save matrizidx.dat matrizidx -ascii;
matrizidy = [];
save matrizidy.dat matrizidy -ascii;
matrizidz = [];
save matrizidz.dat matrizidz -ascii;

% para controle de ID ?nico para cada ponto
ultimoID = 1;
fileID = fopen('id.dat','w');
fwrite(fileID,ultimoID);
fclose(fileID);


%MatrizAdjacencia;
matrizadjacencia = [];
save matrizadjacencia.dat matrizadjacencia -ascii;

%cria as listas de v?rtices visitados e n?o visitados
fileID = fopen('verticesvisitadoid.dat','w');
fclose(fileID);
fileID = fopen('listachegadavisitados.dat','w');

%controle de concorr?ncia na manipula??o dos dados
fclose(fileID);
fileID = fopen('listahandleid.dat','w');
fclose(fileID);
fileID = fopen('listaxyzquadricopter.dat','w');
fclose(fileID);
fileID = fopen('listaxyzcquadricopter.dat','w');
fclose(fileID);
fileID = fopen('verticesNaoVisitadosID.dat','w');
fclose(fileID);
fileID = fopen('verticesNaoVisitadosIDowner.dat','w');
fclose(fileID);
fileID = fopen('listachegadanaovisitados.dat','w');
fclose(fileID);
fileID = fopen('listachegadaocupado.dat','w');
fclose(fileID);

%Configuracao da amplitude RGBD;
% SE ALTERAR ESTE VALOR, O MESMO PRECISA SER CONFIGURADO NOS PARAMETROS RGBD DO KINECT
NO VREP
amplitudeRGBD = 45;
GravaAmplitudeRGBD(amplitudeRGBD);


%REPRESENTACAO DE CUBOS
contxyzc=[];
save contxyzc.dat contxyzc -ascii;
%dimensao do cubo no mapa 3D de cubos
R = 0.2;
GravaR(R);

liberar_exploracao = 0;
save liberar_exploracao.dat liberar_exploracao -ascii;

end
```

```
----------------------------------------

function rtn = MarcarNaoVisitado(QuadricopterHandle)
AddListaChegadaNaoVisitados(QuadricopterHandle);
while(GetListaNaoVisitados(1)~=QuadricopterHandle)
    %espera chegar sua vez
end
% --- VERIFICAR SE O QUADRICOPTER J¡ N√O TEM UM HONEYCOMB ATRIBUIDO A ELE
% PARA EXPLORAR...
fileID = fopen('verticesNaoVisitadosID.dat');
lista = fread(fileID);
fclose(fileID);
fileID = fopen('verticesNaoVisitadosIDowner.dat');
listaowner = fread(fileID);
fclose(fileID);
tam = length(lista);
achou = 0;

for i=1:tam
    vn = lista(i);
    owner = listaowner(i);
    if (owner==QuadricopterHandle)
        achou = 1;
        break;
    end
end

if(achou == 0) % n„o h· vertice atribuido ao UAV
    %---- FIFO ------
    %rtn = ObterIDParaExplorarFIFO(QuadricopterHandle);

    %---- ALGORITMO DISTANCIA EUCLIDIANA ---------

    %rtn = ObterIDParaExplorarDistanciaEuclidiana(QuadricopterHandle);

    %---- ALGORITMO DISTANCIA EUCLIDIANA DISTRIBUIDA -----
    rtn = ObterIDParaExplorarDistanciaEuclidianaDistdro(QuadricopterHandle);

else
    % O UAV j· tem um honeycomb atribuido e dever continuar sua exploraÁ„o
    rtn = vn;
end
RemoverListaNaoVisitados(1);
end
```

```
----------------------------------------
function rtn = MarcarOcupado(QuadricopterHandle,pos)
posOld =-1;
if(pos==0)
    filename = [num2str(QuadricopterHandle) '.mat'];
    save(filename,'pos');
    filename = [num2str(QuadricopterHandle) 'old.mat'];
    save(filename,'posOld');
    rtn = 1;
else
    [resp, idx] = EstaOcupado(QuadricopterHandle,pos);
    if (resp==0)
        filename = [num2str(QuadricopterHandle) 'old.mat'];
        posOld = GetPosicaoUAV(QuadricopterHandle);
        save(filename,'posOld');
        filename = [num2str(QuadricopterHandle) '.mat'];
        save(filename,'pos');
        rtn = 1;
    else
        rtn = 0;
    end
end
end
```

```
----------------------------------------
function rtn = MarcarOcupado(QuadricopterHandle,pos)
posOld =-1;
if(pos==0)
    filename = [num2str(QuadricopterHandle) '.mat'];
    save(filename,'pos');
    filename = [num2str(QuadricopterHandle) 'old.mat'];
    save(filename,'posOld');
    rtn = 1;
else
    [resp, idx] = EstaOcupado(QuadricopterHandle,pos);
    if (resp==0)
        filename = [num2str(QuadricopterHandle) 'old.mat'];
        posOld = GetPosicaoUAV(QuadricopterHandle);
        save(filename,'posOld');
        filename = [num2str(QuadricopterHandle) '.mat'];
        save(filename,'pos');
        rtn = 1;
    else
        rtn = 0;
    end
end
end
```

```
----------------------------------------
function rtn = ObterIDParaExplorarDistanciaEuclidiana(QuadricopterHandle)

fileID = fopen('verticesNaoVisitadosID.dat');
lista = fread(fileID);
fclose(fileID);
fileID = fopen('verticesNaoVisitadosIDowner.dat');
listaowner = fread(fileID);
fclose(fileID);
tam = length(lista);
rtn = 0;
MatrizIDx = GetMatrizIDx();
MatrizIDy = GetMatrizIDy();
MatrizIDz = GetMatrizIDz();
x = MatrizIDx(1);
y = MatrizIDy(1);
z = MatrizIDz(1);
clear distn;
distn =[];
for i=1:tam
    vn = lista(i);
    owner = listaowner(i);
    if (owner==0)
        xi = MatrizIDx(vn);
        yi = MatrizIDy(vn);
        zi = MatrizIDz(vn);
        if (x>xi)
            dx = x - xi;
        else
            dx = xi - x;
        end
        if (y>yi)
            dy = y -  yi;
        else
            dy = yi - y;
        end
        if (z>zi)
            dz = z -  zi;
        else
            dz = zi - z;
        end
         %calcula a distancia euclidiana para cada
         %vertice n„o visitado sem owner
        distn(i) = sqrt(dx^2+dy^2+dz^2);
    else
        %caso tenha um owner, jogar a distancia ao "infinito"
        distn(i) = 10000000;
    end
end

if(length(distn)>0)
    %retorna o ponto com a menor dist,ncia euclidiana
    %a ser explorada a partir do ponto inicial
    [v, iv] = min(distn);
    if(distn(iv)~=10000000)
        %SALVA O NOVO OWNER DO HEXAGONO A SER EXPLORADO
        listaowner(iv) = QuadricopterHandle;
        rtn = lista(iv);
        fileID = fopen('verticesNaoVisitadosIDowner.dat', 'w');
        fwrite(fileID,listaowner);
        fclose(fileID);
        %SALVAR A ATRIBUI«√O DE EXPLORACAO DO UAV
        filename = [num2str(QuadricopterHandle) 'listaexploracao.dat'];
        fileID = fopen(filename,'a');
        fwrite(fileID,rtn);
        fclose(fileID);
    else
        rtn = 0;
    end
else
    rtn = 0;
end
end
```

```
----------------------------------------
function rtn = ObterIDParaExplorarDistanciaEuclidianaDistdro(QuadricopterHandle)
fileID = fopen('verticesNaoVisitadosID.dat');
lista = fread(fileID);
fclose(fileID);
fileID = fopen('verticesNaoVisitadosIDowner.dat');
listaowner = fread(fileID);
fclose(fileID);
tam = length(lista);
rtn = 0;
MatrizIDx = GetMatrizIDx();
MatrizIDy = GetMatrizIDy();
MatrizIDz = GetMatrizIDz();
x = MatrizIDx(1);
y = MatrizIDy(1);
z = MatrizIDz(1);
lista2 = GetListaQuadricopterHandle();
filename = [num2str(lista2(1)) 'listaexploracao.dat'];
fileID = fopen(filename);
trajetoUAV1 = fread(fileID);
fclose(fileID);
filename = [num2str(QuadricopterHandle) 'listaexploracao.dat'];
fileID= fopen(filename);
trajeto = fread(fileID);
fclose(fileID);
clear distn;
listdist = [];
listdistd = [];
listdistz = [];
for i=1:tam
    vn = lista(i);
    owner = listaowner(i);
    if (owner==0)
        xi = MatrizIDx(vn);
        yi = MatrizIDy(vn);
        zi = MatrizIDz(vn);
        if (x>xi)
            dx = x - xi;
        else
            dx = xi - x;
        end
        if (y>yi)
            dy = y -  yi;
        else
            dy = yi - y;
        end
        if (z>zi)
            dz = z -  zi;
        else
            dz = zi - z;
        end
        %calcula a distancia euclidiana para cada
        %vertice n„o visitado sem owner
        listdistz(i) = sqrt(dx^2+dy^2+dz^2);
        if(length(trajeto)>0)
            dro = trajeto(end);
            x = MatrizIDx(dro);
            y = MatrizIDy(dro);
            z = MatrizIDz(dro);
            if (x>xi)
                dx = x - xi;
            else
                dx = xi - x;
            end
            if (y>yi)
                dy = y -  yi;
            else
                dy = yi - y;
            end
            if (z>zi)
                dz = z -  zi;
            else
                dz = zi - z;
            end
            listdistd(i) = sqrt(dx^2+dy^2+dz^2);
        else
```

```
            dro = 1;
            x = MatrizIDx(dro);
            y = MatrizIDy(dro);
            z = MatrizIDz(dro);
            if (x>xi)
                dx = x - xi;
            else
                dx = xi - x;
            end
            if (y>yi)
                dy = y -  yi;
            else
                dy = yi - y;
            end
            if (z>zi)
                dz = z -  zi;
            else
                dz = zi - z;
            end
            listdistd(i) = sqrt(dx^2+dy^2+dz^2);
        end
    else
         %caso tenha um owner, jogar a distancia ao "infinito"
        listdistz(i) = 10000000;
        listdistd(i) = 10000000;
    end
end
listdist = listdistd+listdistz;
if(~isempty(listdist))
    %retorna o ponto com a menor dist,ncia euclidiana
    %a ser explorada a partir do ponto inicial
    [v, iv] = min(listdist);
    if(listdist(iv)~=10000000)
        if (v<10000000)
            %SALVA O NOVO OWNER DO HEXAGONO A SER EXPLORADO
            listaowner(iv) = QuadricopterHandle;
            rtn = lista(iv);
            fileID = fopen('verticesNaoVisitadosIDowner.dat', 'w');
            fwrite(fileID,listaowner);
            fclose(fileID);
            %SALVAR A ATRIBUI«√O DE EXPLORACAO DO UAV
            filename = [num2str(QuadricopterHandle) 'listaexploracao.dat'];
            fileID = fopen(filename,'a');
            fwrite(fileID,rtn);
            fclose(fileID);
        else
            rtn = 0;
        end
    else
        rtn = 0;
    end
else
    rtn = 0;
end
end
```

```
----------------------------------------
function rtn = ObterIDParaExplorarFIFO(QuadricopterHandle)
fileID = fopen('verticesNaoVisitadosID.dat');
lista = fread(fileID);
fclose(fileID);
fileID = fopen('verticesNaoVisitadosIDowner.dat');
listaowner = fread(fileID);
fclose(fileID);
tam = length(lista);
rtn = 0;
for i=1:tam
    vn = lista(i);
    owner = listaowner(i);
    if (owner==0)
        listaowner(i) = QuadricopterHandle;
        fileID = fopen('verticesNaoVisitadosID.dat', 'w');
        fwrite(fileID,lista);
        fclose(fileID);
        fileID = fopen('verticesNaoVisitadosIDowner.dat', 'w');
        fwrite(fileID,listaowner);
        fclose(fileID);
        rtn = vn;
        %SALVAR A ATRIBUI«√O DE EXPLORACAO DO UAV
        filename = [num2str(QuadricopterHandle) 'listaexploracao.dat'];
        fileID = fopen(filename,'a');
        fwrite(fileID,rtn);
        fclose(fileID);
        break;
    end
end
end
```

```
----------------------------------------
function PlotarHexagono(x,y,z)

XPlot = [];
YPlot = [];
ZPlot = [];
angles = [3*pi/6; pi/6; -pi/6; -3*pi/6; -5*pi/6; 5*pi/6];

R = GetRaio();
%nz = sqrt(R^2+0.5773^2);
nz = (R*sin(deg2rad(90)))/sin(deg2rad(60));
%----------------
for i=1:length(angles)
    angle = angles(i);

    alfarad = angle - deg2rad(30);
    dy = double(nz*sin(double(alfarad))/sin(deg2rad(90)));%varia??o em y
    dx = nz*sin(double(deg2rad(180-90-rad2deg(alfarad))))/sin(deg2rad(90)); %varia??o e
x
    xp = x + dx;
    yp = y + dy;
    XPlot(1) = xp;
    YPlot(1) = yp;
    ZPlot(1) = z;
%     XPlot(4) = xp;
%     YPlot(4) = yp;
%     ZPlot(4) = z+R;

    alfarad = angle + deg2rad(30);
    dy = double(nz*sin(double(alfarad))/sin(deg2rad(90)));%varia??o em y
    dx = nz*sin(double(deg2rad(180-90-rad2deg(alfarad))))/sin(deg2rad(90)); %varia??o e
x
    xp = x + dx;
    yp = y + dy;
    XPlot(2) = xp;
    YPlot(2) = yp;
    ZPlot(2) = z;
%      XPlot(3) = xp;
%     YPlot(3) = yp;
%     ZPlot(3) = z+R;


     filename = ['adjacencias_plotadas.mat'];
    if isfile(filename)
        load(filename);
        flag=1;
        while(flag==1)
            try
                MatrizAdjacencia = GetMatrizAdjacencia();
                flag = 0;
            catch
                flag=1;
            end
        end
        adjacencias_plotadas;
        if (length(MatrizAdjacencia)>length(adjacencias_plotadas))
                aux = zeros(length(MatrizAdjacencia));
                for k=1:length(adjacencias_plotadas)
                    for j=1:length(adjacencias_plotadas)
                        aux(k,j) = adjacencias_plotadas(k,j);
                    end
                end
                adjacencias_plotadas = aux;
        end

    else
        MatrizAdjacencia = GetMatrizAdjacencia();
        adjacencias_plotadas = zeros(length(MatrizAdjacencia));

    end

    lw = 2;


    %verificar se o xy possui adjacÍncias
    id1 = GetIDbyXYZ(x,y,z);
```

```
    xyz = [x y z];
    temAdjacencia = VerificaAnguloAdjacencia(id1,angle,xyz);
    temperatura = VerificaTemperaturaAlta(id1,angle);
     filename2 = [num2str(id1) '_honeycomb_temp_read.mat'];
            if(~isfile(filename2))
                temperatura=0;
            end
    if(temAdjacencia==0)
        %id nao existente e consequentemente nao possui adjacencia
        %VERIFICAR INFORMA«√O DE SENSOR DE TEMPERATURA
        if (temperatura==0)
         plot3(XPlot,YPlot,ZPlot,'k-', 'linewidth', lw);
        else
          plot3(XPlot,YPlot,ZPlot,'r-', 'linewidth', lw);
        end
    else
        %id existe, verificar se ha adjacencia
        %VERIFICAR INFORMA«√O DE SENSOR DE TEMPERATURA
        if (adjacencias_plotadas(temAdjacencia,id1)==0 | temperatura~=0)

            if (temperatura==0)
             plot3(XPlot,YPlot,ZPlot,'k--','linewidth', lw);
            else
              plot3(XPlot,YPlot,ZPlot,'w-','linewidth', lw);
              plot3(XPlot,YPlot,ZPlot,'r--','linewidth', lw);
            end
            adjacencias_plotadas(id1,temAdjacencia)=1;
            adjacencias_plotadas(temAdjacencia,id1)=1;
            save(filename,'adjacencias_plotadas', '-v6');
        else
            adjacencias_plotadas(id1,temAdjacencia)=1;
            save(filename,'adjacencias_plotadas', '-v6');
        end


    end
    hold on;

    XPlot = [];
    YPlot = [];
    ZPlot = [];
end


%-----------------




End
```

```
----------------------------------------
function PlotCube3D(xyz, R, maior,CONT) % passar os parametros de xyz como uma array.
Ex: xyz = [2.3 5.6 8.1];

%defini??o dos v?rtices do cubo a partir do ponto passado e a resolu??o do
%cubo
xf=round(xyz(1)/R)*R;
xR=xf+R;
yf=round(xyz(2)/R)*R;
yR=yf+R;
zf=round(xyz(3)/R)*R;
zR=zf+R;

vert = [xf yf zf; xf yR zf; xR yR zf; xR yf zf;...
        xf yf zR; xf yR zR; xR yR zR; xR yf zR];

fac = [1 2 3 4; ...
    2 6 7 3; ...
    4 3 7 8; ...
    1 5 8 4; ...
    1 2 6 5; ...
    5 6 7 8];

dxyz = GetDxDyDz();
Dz = dxyz(3);
cz = abs(xyz(3)/Dz);

% if(abs(xyz(3))<1)
%     cz = abs(xyz(3));
% else
%     cz = abs(1/xyz(3));
% end

ct = (CONT/maior);

if(ct<0.3)
    ct = 0.3;
end

cor = [cz 0.3 1];
patch('Faces',fac,'Vertices',vert,'EdgeColor','none','FaceColor',cor,'FaceAlpha', ct);
% patch function
% material shiny;
% alpha('color');
% alphamap('rampdown');
view(45,60);

%------second cube-----
%hold on;
end


----------------------------------------
function PlotHoneycomb()
clear
filename = 'adjacencias_plotadas.mat';
if isfile(filename)
    delete (filename);
end
listaQuadricopterHandle = GetListaQuadricopterHandle();
cores = ["blue" "red"  "green"  "magenta"  "yellow"];
contagem = [];
MatrizIDx = GetMatrizIDx();
MatrizIDy = GetMatrizIDy();
MatrizIDz = GetMatrizIDz();

f1 = figure('Name','Honeycomb Topologic Map', 'NumberTitle','off');
f1.Position = [100 100 800 900];
 hold on;


hc = 0.5773;
xs = [MatrizIDx(1)];
ys = [MatrizIDy(1)];
zs = [MatrizIDz(1)];
```

```
for i=1:length(listaQuadricopterHandle)
    % descomentar para plotar a ordem de exploracao
    % filename = [num2str(listaQuadricopterHandle(i)) 'listaexploracao.dat'];
filename = [num2str(listaQuadricopterHandle(i)) 'listatrajeto.dat'];
    fileID = fopen(filename);
    trajetoUAV = fread(fileID);

    tamUAV = size(trajetoUAV);
    flag=1;
    while(flag==1)
        try
            MatrizIDx = GetMatrizIDx();
            MatrizIDy = GetMatrizIDy();
            MatrizIDz = GetMatrizIDz();
        flag=0;
        catch
            flag=1;
        end
    end


    for j=1:tamUAV(1)
        id = trajetoUAV(j,1);
        xs = [xs MatrizIDx(id)];
        ys = [ys MatrizIDy(id)];
        zs = [zs MatrizIDz(id)];


        PlotarHexagono(MatrizIDx(id), MatrizIDy(id), MatrizIDz(id));


    end
    contagem(i) = tamUAV(1);
end


%plot(pgon, 'FaceColor','w');
for i=1:length(contagem)
    if (i==1)
        inicio = 1;
        fim = contagem(i)+1;
    end
    if(i>1)
        inicio = fim+1;
        fim = inicio + contagem(i) -1;
    end
    xc = [];
    yc = [];
    zc = [];
    j = 0;
    if(i==1)
        j = -0.15;
    end
    if(i==2)
        j = 0;
    end
    if(i==3)
        j = 0.15;
    end
    xc=xs(inicio:fim)+ j;
    yc = ys(inicio:fim);
    zc = zs(inicio:fim);
    plot3(xc,yc,zc,cores(i),'linewidth', 2);
end


tam = size(MatrizIDx);
for i=1:tam(2)
    vn = num2str(i);
    text(MatrizIDx(i),MatrizIDy(i),MatrizIDz(i),vn, 'FontSize',14);

end

end
```

```
----------------------------------------
function PrincipalUAVExplorer()
clear all;
close all;
clc;
port = 20000;
disp('Inicio do programa');
vrep=remApi('remoteApi');
vrep.simxFinish(-1);
clientID=vrep.simxStart('127.0.0.1',port,true,false,2000,5);
if (clientID>-1)
    % Busca os handles de todos os objetos do ambiente

[res,objs]=vrep.simxGetObjects(clientID,vrep.sim_handle_all,vrep.simx_opmode_oneshot);
    %PRIMEIRO UAV DETECTADO - DEFINIDO OS HANDLES DE ACESSO
    % handler do 'Quadricopter'
    [rtnArducopter,QuadricopterHandle] =
vrep.simxGetObjectHandle(clientID,'Quadricopter',vrep.simx_opmode_oneshot_wait);
    [rtnQuadricopter_target,Quadricopter_targetHandle] =
vrep.simxGetObjectHandle(clientID,'Quadricopter_target',vrep.simx_opmode_oneshot_wait);
    [rtnKinect,KinectHandle] =
vrep.simxGetObjectHandle(clientID,'kinect_depth',vrep.simx_opmode_oneshot_wait);
    [rtnKinect,Kinec_rgb_tHandle] =
vrep.simxGetObjectHandle(clientID,'kinect_rgb',vrep.simx_opmode_oneshot_wait);
    [rtnProximity_sensor,Proximity_sensorHandle] =
vrep.simxGetObjectHandle(clientID,'Proximity_sensorHead',vrep.simx_opmode_oneshot_wait);
    [rtnProximity_sensorUp,Proximity_sensorUpHandle] =
vrep.simxGetObjectHandle(clientID,'Proximity_sensorUp',vrep.simx_opmode_oneshot_wait);
    [rtnProximity_sensorDown,Proximity_sensorDownHandle] =
vrep.simxGetObjectHandle(clientID,'Proximity_sensorDown',vrep.simx_opmode_oneshot_wait);
    [rtnVisionSensor,VisionSensortHandle] =
vrep.simxGetObjectHandle(clientID,'Vision_sensor',vrep.simx_opmode_oneshot_wait);


    %SEGUNDO UAV DETECTADO - DEFINIDO OS HANDLES DE ACESSO
    % handler do 'Quadricopter'
    [rtnKinect1,KinectHandle1] =
vrep.simxGetObjectHandle(clientID,'kinect_depth#0',vrep.simx_opmode_oneshot_wait);
    [rtnKinect,Kinec_rgb_tHandle1] =
vrep.simxGetObjectHandle(clientID,'kinect_rgb#0',vrep.simx_opmode_oneshot_wait);
    [rtnArducopter1,QuadricopterHandle1] =
vrep.simxGetObjectHandle(clientID,'Quadricopter#0',vrep.simx_opmode_oneshot_wait);
    [rtnQuadricopter_target1,Quadricopter_targetHandle1] =
vrep.simxGetObjectHandle(clientID,'Quadricopter_target#0',vrep.simx_opmode_oneshot_wait)
;
    [rtnProximity_sensor1,Proximity_sensorHandle1] =
vrep.simxGetObjectHandle(clientID,'Proximity_sensorHead#0',vrep.simx_opmode_oneshot_wait
);
    [rtnProximity_sensorUp1,Proximity_sensorUpHandle1] =
vrep.simxGetObjectHandle(clientID,'Proximity_sensorUp#0',vrep.simx_opmode_oneshot_wait);
    [rtnProximity_sensorDown1,Proximity_sensorDownHandle1] =
vrep.simxGetObjectHandle(clientID,'Proximity_sensorDown#0',vrep.simx_opmode_oneshot_wait
);


    %TERCEIRO UAV DETECTADO
    % handler do 'Quadricopter'
    [rtnArducopter2,QuadricopterHandle2] =
vrep.simxGetObjectHandle(clientID,'Quadricopter#1',vrep.simx_opmode_oneshot_wait);
    [rtnKinect2,KinectHandle2] =
vrep.simxGetObjectHandle(clientID,'kinect_depth#1',vrep.simx_opmode_oneshot_wait);
    [rtnKinect,Kinec_rgb_tHandle2] =
vrep.simxGetObjectHandle(clientID,'kinect_rgb#1',vrep.simx_opmode_oneshot_wait);
    [rtnQuadricopter_target2,Quadricopter_targetHandle2] =
vrep.simxGetObjectHandle(clientID,'Quadricopter_target#1',vrep.simx_opmode_oneshot_wait)
;
    [rtnProximity_sensor2,Proximity_sensorHandle2] =
vrep.simxGetObjectHandle(clientID,'Proximity_sensorHead#1',vrep.simx_opmode_oneshot_wait
);
    [rtnProximity_sensorUp2,Proximity_sensorUpHandle2] =
vrep.simxGetObjectHandle(clientID,'Proximity_sensorUp#1',vrep.simx_opmode_oneshot_wait);
    [rtnProximity_sensorDown2,Proximity_sensorDownHandle2] =
vrep.simxGetObjectHandle(clientID,'Proximity_sensorDown#1',vrep.simx_opmode_oneshot_wait
);
```

```
    vrep.simxGetObjectOrientation(clientID, QuadricopterHandle, -1,
vrep.simx_opmode_streaming);
    vrep.simxGetObjectPosition(clientID, QuadricopterHandle, -1,
vrep.simx_opmode_streaming);
    vrep.simxGetObjectOrientation(clientID, QuadricopterHandle1, -1,
vrep.simx_opmode_streaming);
    vrep.simxGetObjectPosition(clientID, QuadricopterHandle1, -1,
vrep.simx_opmode_streaming);
    vrep.simxGetObjectOrientation(clientID, QuadricopterHandle2, -1,
vrep.simx_opmode_streaming);
    vrep.simxGetObjectPosition(clientID, QuadricopterHandle2, -1,
vrep.simx_opmode_streaming);



    uav.QuadricopterHandle = QuadricopterHandle;
    uav.Quadricopter_targetHandle = Quadricopter_targetHandle;
    uav.Proximity_sensorHandle = Proximity_sensorHandle;
    uav.Proximity_sensorUpHandle = Proximity_sensorUpHandle;
    uav.Proximity_sensorDownHandle = Proximity_sensorDownHandle;
    uav.KinectHandle = KinectHandle;
    uav.Kinec_rgb_tHandle = Kinec_rgb_tHandle;
    uav.VisionSensortHandle = VisionSensortHandle;
    uav.clientID = clientID;
    UAVS(1) = uav;
    vrep.simxFinish(clientID);  % fecha a comunicacao
    vrep.delete();              % destroi os elementos

    mapa();%inicializa os dados da simulacao

    %Inicia a exploracao pelo UAV sentinela
    vrep=remApi('remoteApi');
    vrep.simxFinish(-1);
    clientID=vrep.simxStart('127.0.0.1',port,true,false,0,5);
    [resQuad1, posUAV] = vrep.simxGetObjectPosition(clientID, QuadricopterHandle, -1,
vrep.simx_opmode_streaming);
    while (posUAV(3)==0)%espera o inicio de leituras diferentes de zero
        [resQuad1, posUAV] = vrep.simxGetObjectPosition(clientID, QuadricopterHandle, -
1, vrep.simx_opmode_buffer);
    end

    vrep.simxFinish(clientID);  % fecha a comunicacao
    vrep.delete();              % destroi os elementos


    %Salva todos os QuadricopterHandler existentes (ou seja, qual o
    %identificador de cada UAV existente para posterior recuperaÁ„o de sua
    %localizacao. Caso exista mais um UAV, o mesmo dever· ser inserido na
    %lista.

    %PARA 3 UAVS
    listaQuadricopterHandle = [QuadricopterHandle QuadricopterHandle1
QuadricopterHandle2];

    %PARA 2 UAVS
    %listaQuadricopterHandle = [QuadricopterHandle QuadricopterHandle1];
    save listaQuadricopterHandle.mat listaQuadricopterHandle -mat;

    lista = GetListaQuadricopterHandle();
    zero = [];
    filename = [num2str(lista(1)) 'listaexploracao.dat'];
    fileID = fopen(filename,'w');
    fwrite(fileID,zero);
    fclose(fileID);

    filename = [num2str(lista(1)) 'listatrajeto.dat'];
    fileID = fopen(filename,'w');
    fwrite(fileID,zero);
    fclose(fileID);

    filename = [num2str(lista(2)) 'listaexploracao.dat'];
    fileID = fopen(filename,'w');
    fwrite(fileID,zero);
    fclose(fileID);

    filename = [num2str(lista(2)) 'listatrajeto.dat'];
    fileID = fopen(filename,'w');
```

```
    fwrite(fileID,zero);
    fclose(fileID);


    %CASO HAJA UM TERCEIRO UAV, DESCOMENTAR A PARTE ABAIXO
    %-----------------------------------------------------
    filename = [num2str(lista(3)) 'listaexploracao.dat'];
    fileID = fopen(filename,'w');
    fwrite(fileID,zero);
    fclose(fileID);

    filename = [num2str(lista(3)) 'listatrajeto.dat'];
    fileID = fopen(filename,'w');
    fwrite(fileID,zero);
    fclose(fileID);
    %-----------------------------------------------------


    %inicializar o registro da posiÁ„o inicial de cada UAV como 0 (n„o
    %alocado ainda)
    pos = 0;
    MarcarOcupado(QuadricopterHandle,pos);
    MarcarOcupado(QuadricopterHandle1,pos);
    MarcarOcupado(QuadricopterHandle2,pos); %DESCOMENTAR CASO 3 UAVS

    %SALVAR O TRAJETO DE EXPLORACAO DO UAV 2
    filename = [num2str(QuadricopterHandle1) 'listatrajeto.dat'];
    fileID = fopen(filename,'a');
    fwrite(fileID,1);
    fclose(fileID);

    %SALVAR O TRAJETO DE EXPLORACAO DO UAV3 - DESCOMENTAR CASO 3
    %UAVS
    filename = [num2str(QuadricopterHandle2) 'listatrajeto.dat'];
    fileID = fopen(filename,'a');
    fwrite(fileID,1);
    fclose(fileID);



ExplorarInicial(uav.QuadricopterHandle,uav.Quadricopter_targetHandle,uav.Proximity_senso
rHandle,uav.Proximity_sensorUpHandle,uav.Proximity_sensorDownHandle,uav.KinectHandle,uav
.Kinec_rgb_tHandle, uav.VisionSensortHandle, port);
    fprintf('Finalizou a exploracao inicial\n');
    liberar_exploracao = 1;
    save liberar_exploracao.dat liberar_exploracao -ascii;

    while(ExisteVerticeNaoVisitado()==1)
        try
            pos =
Explorar(uav.QuadricopterHandle,uav.Quadricopter_targetHandle,uav.Proximity_sensorHandle
,uav.Proximity_sensorUpHandle,uav.Proximity_sensorDownHandle,uav.KinectHandle,uav.Kinec_
rgb_tHandle, uav.VisionSensortHandle, port);
            MarcarOcupado(uav.QuadricopterHandle,pos);
            MatrizIDx = GetMatrizIDx();
            MatrizIDy = GetMatrizIDy();
            MatrizIDz = GetMatrizIDz();
            x = MatrizIDx(pos);
            y = MatrizIDy(pos);
            z = MatrizIDz(pos);
            xyz = [x y z];

DeslocarUAV(uav.QuadricopterHandle,uav.Quadricopter_targetHandle,clientID,xyz,port);
        catch
            pos =
Explorar(uav.QuadricopterHandle,uav.Quadricopter_targetHandle,uav.Proximity_sensorHandle
,uav.Proximity_sensorUpHandle,uav.Proximity_sensorDownHandle,uav.KinectHandle,uav.Kinec_
rgb_tHandle, uav.VisionSensortHandle, port);
            MarcarOcupado(uav.QuadricopterHandle,pos);
            MatrizIDx = GetMatrizIDx();
            MatrizIDy = GetMatrizIDy();
            MatrizIDz = GetMatrizIDz();
            x = MatrizIDx(pos);
            y = MatrizIDy(pos);
            z = MatrizIDz(pos);
            xyz = [x y z];
```

```
DeslocarUAV(uav.QuadricopterHandle,uav.Quadricopter_targetHandle,clientID,xyz,port);
        end
    end
    fimUAV1 = now;
    save('fimUAV1','-v6');
    vrep.simxFinish(clientID);  % fecha a comunicacao
    vrep.delete();              % destroi os elementos
    disp('Fim do Programa');
end
```

```
----------------------------------------
function ProcessarLeituraRGBD()
% Para cada idhoneycomb da lista de leitura rgbd pre, pegar o id e abrir o
% arquivo correspondente em _honeycomb_3dread.mat. Para cada angulo do
% honeycomb, chamar o TransformarRGBD() para gerar a transformacao dos
% dados;


fileID = fopen('listaIDHoneycombLeituraRGBDpre.dat');
listaHoneycomb = fread(fileID);
R = GetR();
amplitudeRGBD = GetAmplitudeRGBD();

id_honeycomb_3dread.posUAV = [];
id_honeycomb_3dread.leitura_3pi_6 = [];
id_honeycomb_3dread.angulo_3pi_6 = [];
id_honeycomb_3dread.leitura_pi_6 = [];
id_honeycomb_3dread.angulo_pi_6 = [];
id_honeycomb_3dread.leitura_menospi_6 = [];
id_honeycomb_3dread.angulo_menospi_6 = [];
id_honeycomb_3dread.leitura_menos3pi_6 = [];
id_honeycomb_3dread.angulo_menos3pi_6 = [];
id_honeycomb_3dread.leitura_menos5pi_6 = [];
id_honeycomb_3dread.angulo_menos5pi_6 = [];
id_honeycomb_3dread.leitura_5pi_6 = [];
id_honeycomb_3dread.angulo_5pi_6 = [];
id_honeycomb_3dread.resolucao_x = 0;
id_honeycomb_3dread.resolucao_y = 0;


parfor i=1:length(listaHoneycomb)
    idHoneycomb = listaHoneycomb(i);


    filename = [num2str(idHoneycomb) '_honeycomb_3dread.mat'];
    x = load(filename);

    posUAV = x.id_honeycomb_3dread.posUAV;
    xn = x.id_honeycomb_3dread.resolucao_x;
    yn = x.id_honeycomb_3dread.resolucao_y ;
    buffer = x.id_honeycomb_3dread.leitura_3pi_6;
    angUAV = x.id_honeycomb_3dread.angulo_3pi_6;
    TranformarRGBD(idHoneycomb, R, amplitudeRGBD, buffer, xn, yn, angUAV, posUAV);

    buffer = x.id_honeycomb_3dread.leitura_pi_6;
    angUAV = x.id_honeycomb_3dread.angulo_pi_6;
    TranformarRGBD(idHoneycomb, R, amplitudeRGBD, buffer, xn, yn, angUAV, posUAV);

    buffer = x.id_honeycomb_3dread.leitura_menospi_6;
    angUAV = x.id_honeycomb_3dread.angulo_menospi_6;
    TranformarRGBD(idHoneycomb, R, amplitudeRGBD, buffer, xn, yn, angUAV, posUAV);

    buffer = x.id_honeycomb_3dread.leitura_menos3pi_6;
    angUAV = x.id_honeycomb_3dread.angulo_menos3pi_6;
    TranformarRGBD(idHoneycomb, R, amplitudeRGBD, buffer, xn, yn, angUAV, posUAV);

    buffer = x.id_honeycomb_3dread.leitura_menos5pi_6;
    angUAV = x.id_honeycomb_3dread.angulo_menos5pi_6;
    TranformarRGBD(idHoneycomb, R, amplitudeRGBD, buffer, xn, yn, angUAV, posUAV);

    buffer = x.id_honeycomb_3dread.leitura_5pi_6;
    angUAV = x.id_honeycomb_3dread.angulo_5pi_6;
    TranformarRGBD(idHoneycomb, R, amplitudeRGBD, buffer, xn, yn, angUAV, posUAV);

end
texto = "terminou processamento paralelo"

%   filename = [num2str(idOrigem) '_honeycomb_3dread.mat'];
%     load(filename);
%     id_honeycomb_3dread
%
%  filename2 = [num2str(idHoneycomb) 'xc.dat'];
% load(filename2);
% filename2 = [num2str(idHoneycomb) 'yc.dat'];
% load(filename2);
% filename2 = [num2str(idHoneycomb) 'zc.dat'];
```

```
% load(filename2);

End
```

```
----------------------------------------
function resp=RotacionarUAV(QuadricopterHandle, Quadricopter_targetHandle, clientID,
angle,port)

vrep=remApi('remoteApi');
vrep.simxFinish(-1);
clientID=vrep.simxStart('127.0.0.1',port,true,false,0,5);
%rotaciona o UAV de acordo com o angulo de giro passado

ex = 0;
ey = 0;
ez = angle;
pos=[ex ey ez];

cont = 0;
[res2, eulerQuadricopter] = vrep.simxGetObjectOrientation(clientID, QuadricopterHandle,
-1, vrep.simx_opmode_streaming);
a = now;
saiu = 0;
while(res2~=0)
    pause(2);
    [res2, eulerQuadricopter] = vrep.simxGetObjectOrientation(clientID,
QuadricopterHandle, -1, vrep.simx_opmode_buffer);
    b = (now - a)*3600;
    if(b>1)
        saiu = 1;
        break;
    end
end
if (saiu == 1)
    RotacionarUAV(QuadricopterHandle, Quadricopter_targetHandle, clientID, angle,port);
else
    vrep.simxSetObjectOrientation(clientID, Quadricopter_targetHandle, -1, pos,
vrep.simx_opmode_oneshot);
    [res2, eulerQuadricopter] = vrep.simxGetObjectOrientation(clientID,
QuadricopterHandle, -1, vrep.simx_opmode_streaming);
    while(abs(eulerQuadricopter(3)-ez)>0.015)
        pause(5);
        [res2, eulerQuadricopter] = vrep.simxGetObjectOrientation(clientID,
QuadricopterHandle, -1, vrep.simx_opmode_buffer);
    end
end
resp = 1;
vrep.simxFinish(clientID);  % fecha a comunicacao
vrep.delete();              % destroi os elementos
end
```

```
----------------------------------------
function
rtn=VerificaAdjacencias(clientID,QuadricopterHandle,Proximity_sensorHandle,idOrigem,angl
e,xyz, port)
vrep=remApi('remoteApi');
vrep.simxFinish(-1);
clientID=vrep.simxStart('127.0.0.1',port,true,false,0,2);
Raio = GetRaio();
rtn = -1;
[rtnProximity_sensor,detectionStateProximity_sensor,detectedPointProximity_sensor,detect
edObjectHandleProximity_sensor,detectedSurfaceNormalVectorProximity_sensor]=
vrep.simxReadProximitySensor(clientID,Proximity_sensorHandle,vrep.simx_opmode_streaming)
;
a = now;
saiu = 0;
while (detectedPointProximity_sensor(3) == 0)

[rtnProximity_sensor,detectionStateProximity_sensor,detectedPointProximity_sensor,detect
edObjectHandleProximity_sensor,detectedSurfaceNormalVectorProximity_sensor]=
vrep.simxReadProximitySensor(clientID,Proximity_sensorHandle,vrep.simx_opmode_buffer);
    b = (now - a)*3600;
    if(b>1)
        saiu = 1;
        break;
    end
    if (detectedPointProximity_sensor(3)>3*Raio)
        pause(randi(3));

[rtnProximity_sensor,detectionStateProximity_sensor,detectedPointProximity_sensor,detect
edObjectHandleProximity_sensor,detectedSurfaceNormalVectorProximity_sensor]=
vrep.simxReadProximitySensor(clientID,Proximity_sensorHandle,vrep.simx_opmode_buffer);
        cont = 1;
        while(detectedPointProximity_sensor(3)>3*Raio & cont<5 )
            pause(randi(3));

[rtnProximity_sensor,detectionStateProximity_sensor,detectedPointProximity_sensor,detect
edObjectHandleProximity_sensor,detectedSurfaceNormalVectorProximity_sensor]=
vrep.simxReadProximitySensor(clientID,Proximity_sensorHandle,vrep.simx_opmode_buffer);
            cont = cont + 1;
        end
    end
end
if (saiu == 1)

VerificaAdjacencias(clientID,QuadricopterHandle,Proximity_sensorHandle,idOrigem,angle,xy
z, port);
else
    dy = 2*double(Raio)*sin(double(angle))/sin(degtorad(90));
    dx = 2*double(Raio)*sin(double(degtorad(180-90-radtodeg(angle))))/sin(degtorad(90));
    xp = xyz(1) + dx;
    yp = xyz(2) + dy;
    zp = xyz(3);
    if (detectedPointProximity_sensor(3)>3*Raio)
        % NADA DETECTADO - PODE ADICIONAR ADJACENCIA a distancia ponto ? 2*Raio
        %adicionar os pontos xp yp e zp na lista de adjacencias
        rtn = AdicionarAdjacencias(xp,yp,zp,QuadricopterHandle,idOrigem);
    else
        delta_size_uav = 0.2;
        % O UAV tem aproximadamente 0.4 cm de tamanho lateral, e como ele esta no centro
do hexagono, considera-se a metade deste tamanho

        %verifica se o ponto detectado eh outro UAV
        if (detectedPointProximity_sensor(3)<(2*double(Raio)-delta_size_uav))
            % realmente nao ha adjacencia, nada a fazer
            rtn = -1;
        else
            % pode ser outro UAV, ent„o busca-se se alguem esta no hexagono
            % que se esta verificando a adjacencia

            n_xyz(1) = xp;
            n_xyz(2) = yp;
            n_xyz(3) = zp;

            %obtem-se o id do hexagono pela posicao xyz
            idAng = BuscarIDporXYZ(n_xyz);
            if(idAng>0)
```

```
                %verifica se ha algum uav neste id/hexagono
                [resp, idx]  = EstaOcupado(QuadricopterHandle,idAng);
                while(resp==-1)%caso ocorra excecao (acessando quando o arquivo est·
sendo
                    %atualizado
                    [resp, idx] = EstaOcupado(QuadricopterHandle,idAng);
                end
                if(resp~=0)
                    %algum uav esta neste espaco, entao ha adjacencia
                    rtn = AdicionarAdjacencias(xp,yp,zp,QuadricopterHandle,idOrigem);
                else
                    %ninguem esta neste espaco, entao ha obstaculo e
                    %consequentemente nao ha adjacencia: NADA A FAZER
                    rtn = -1;
                end
            else
                rtn = -1;
            end
        end
    end
    vrep.simxFinish(clientID);  % fecha a comunicacao
    vrep.delete();              % destroi os elementos
end
end
```

```
----------------------------------------
function
rtn=VerificaAdjacenciasInferior(clientID,QuadricopterHandle,Proximity_sensorHandle,idOri
gem,xyz,port)
vrep=remApi('remoteApi');
vrep.simxFinish(-1);
clientID=vrep.simxStart('127.0.0.1',port,true,false,0,2);
dxyz = GetDxDyDz();
Dx = dxyz(1);
Dy = dxyz(2);
Dz = dxyz(3);
Raio = Dz/4;
rtn = -1;
[rtnProximity_sensor,detectionStateProximity_sensor,detectedPointProximity_sensor,detect
edObjectHandleProximity_sensor,detectedSurfaceNormalVectorProximity_sensor]=
vrep.simxReadProximitySensor(clientID,Proximity_sensorHandle,vrep.simx_opmode_streaming)
;
[rtnProximity_sensor,detectionStateProximity_sensor,detectedPointProximity_sensor,detect
edObjectHandleProximity_sensor,detectedSurfaceNormalVectorProximity_sensor]=
vrep.simxReadProximitySensor(clientID,Proximity_sensorHandle,vrep.simx_opmode_buffer);
a = now;
saiu = 0;
while (rtnProximity_sensor ~= 0)

[rtnProximity_sensor,detectionStateProximity_sensor,detectedPointProximity_sensor,detect
edObjectHandleProximity_sensor,detectedSurfaceNormalVectorProximity_sensor]=
vrep.simxReadProximitySensor(clientID,Proximity_sensorHandle,vrep.simx_opmode_buffer);
    b = (now - a)*3600;
    if(b>1)
        saiu = 1;
        break;
    end
end
if (saiu == 1)

VerificaAdjacenciasInferior(clientID,QuadricopterHandle,Proximity_sensorHandle,idOrigem,
xyz, port);
else
    xp = xyz(1);
    yp = xyz(2);
    zp = xyz(3) - 2*Raio;
    if ((detectedPointProximity_sensor(3)>3*Raio))
        %o ponto detectado est· alÈm do hexagono ou nada foi detectado
        %adicionar os pontos xp yp e zp na lista de adjacencias
        AdicionarAdjacencias(xp,yp,zp,QuadricopterHandle,idOrigem);
        rtn = 1;
    else
        delta_size_uav = 0.1;
        if (detectedPointProximity_sensor(3)<(2*double(Raio)-delta_size_uav))
            % realmente nao ha adjacencia, nada a fazer
            rtn = -1;
        else
            % pode ser outro UAV, ent„o busca-se se alguem esta no hexagono
            % que se esta verificando a adjacencia
            n_xyz(1) = xp;
            n_xyz(2) = yp;
            n_xyz(3) = zp;
            %obtem-se o id do hexagono pela posicao xyz
            idAng = BuscarIDporXYZ(n_xyz);
            if(idAng>0)
                %verifica se ha algum uav neste id/hexagono
                [resp, idx]  = EstaOcupado(QuadricopterHandle,idAng);
                while(resp==-1)%caso ocorra excecao (acessando quando o arquivo est·
sendo
                    %atualizado
                    [resp, idx] = EstaOcupado(QuadricopterHandle,idAng);
                end
                if(resp~=0)
                    %algum uav esta neste espaco, entao ha adjacencia
                    rtn = AdicionarAdjacencias(xp,yp,zp,QuadricopterHandle,idOrigem);
                else
                    %ninguem esta neste espaco, entao ha obstaculo e
                    %consequentemente nao ha adjacencia: NADA A FAZER
                    rtn = -1;
                end
            else
```

```
                rtn = -1;
            end
        end
    end
end
vrep.simxFinish(clientID);  % fecha a comunicacao
vrep.delete();              % destroi os elementos
end
```

```
----------------------------------------
function
rtn=VerificaAdjacenciasSuperior(clientID,QuadricopterHandle,Proximity_sensorHandle,idOri
gem,xyz,port)
vrep=remApi('remoteApi');
vrep.simxFinish(-1);
clientID=vrep.simxStart('127.0.0.1',port,true,false,0,2);
dxyz = GetDxDyDz();
Dx = dxyz(1);
Dy = dxyz(2);
Dz = dxyz(3);
Raio = Dz/4;
rtn = -1;
[rtnProximity_sensor,detectionStateProximity_sensor,detectedPointProximity_sensor,detect
edObjectHandleProximity_sensor,detectedSurfaceNormalVectorProximity_sensor]=
vrep.simxReadProximitySensor(clientID,Proximity_sensorHandle,vrep.simx_opmode_streaming)
;
[rtnProximity_sensor,detectionStateProximity_sensor,detectedPointProximity_sensor,detect
edObjectHandleProximity_sensor,detectedSurfaceNormalVectorProximity_sensor]=
vrep.simxReadProximitySensor(clientID,Proximity_sensorHandle,vrep.simx_opmode_buffer);
a = now;
saiu = 0;
while (rtnProximity_sensor ~= 0)

[rtnProximity_sensor,detectionStateProximity_sensor,detectedPointProximity_sensor,detect
edObjectHandleProximity_sensor,detectedSurfaceNormalVectorProximity_sensor]=
vrep.simxReadProximitySensor(clientID,Proximity_sensorHandle,vrep.simx_opmode_buffer);
    b = (now - a)*3600;
    if(b>1)
        saiu = 1;
        break;
    end
end
if (saiu == 1)

VerificaAdjacenciasSuperior(clientID,QuadricopterHandle,Proximity_sensorHandle,idOrigem,
xyz,port);
else
    % NADA DETECTADO - PODE ADICIONAR ADJACENCIA a distancia ponto ? 2*Raio
    xp = xyz(1);
    yp = xyz(2);
    zp = xyz(3) + 2*Raio;
    if ((detectedPointProximity_sensor(3)>3*Raio)|| detectedPointProximity_sensor(3)==0)
        %o ponto detectado est· alÈm do hexagono ou nada foi detectado
        %adicionar os pontos xp yp e zp na lista de adjacencias
        AdicionarAdjacencias(xp,yp,zp,QuadricopterHandle,idOrigem);
        rtn = 1;
    else
        delta_size_uav = 0.1;
        if (detectedPointProximity_sensor(3)<(2*double(Raio)-delta_size_uav))
            % realmente nao ha adjacencia, nada a fazer
            rtn = -1;
        else
            % pode ser outro UAV, ent„o busca-se se alguem esta no hexagono
            % que se esta verificando a adjacencia
            n_xyz(1) = xp;
            n_xyz(2) = yp;
            n_xyz(3) = zp;
            %obtem-se o id do hexagono pela posicao xyz
            idAng = BuscarIDporXYZ(n_xyz);
            if(idAng>0)
                %verifica se ha algum uav neste id/hexagono
                [resp, idx]  = EstaOcupado(QuadricopterHandle,idAng);
                while(resp==-1)%caso ocorra excecao (acessando quando o arquivo est·
sendo
                    %atualizado
                    [resp, idx] = EstaOcupado(QuadricopterHandle,idAng);
                end
                if(resp~=0)
                    %algum uav esta neste espaco, entao ha adjacencia
                    rtn = AdicionarAdjacencias(xp,yp,zp,QuadricopterHandle,idOrigem);
                else
                    %ninguem esta neste espaco, entao ha obstaculo e
                    %consequentemente nao ha adjacencia: NADA A FAZER
                    rtn = -1;
                end
```

```
            else
                rtn = -1;
            end
        end
    end
end
vrep.simxFinish(clientID);  % fecha a comunicacao
vrep.delete();              % destroi os elementos
end
```

```
----------------------------------------
function rtn =
VerificaTemperatura(clientID,QuadricopterHandle,Kinec_rgb_tHandle,VisionSensortHandle,
idOrigem, angle, xyz,port)
rtn = 0;

vrep=remApi('remoteApi');
vrep.simxFinish(-1);
clientID=vrep.simxStart('127.0.0.1',port,true,false,2000,5);
matrix = [];
[returnCode, detectionState, auxData,
auxPacketInfo]=vrep.simxReadVisionSensor(clientID,VisionSensortHandle,vrep.simx_opmode_s
treaming);
[returnCode, detectionState, auxData,
auxPacketInfo]=vrep.simxReadVisionSensor(clientID,VisionSensortHandle,vrep.simx_opmode_b
uffer);
[returnCode2,resolution,matrix]=vrep.simxGetVisionSensorImage2(clientID,
VisionSensortHandle, 0,vrep.simx_opmode_streaming);
[returnCode2,resolution,matrix]=vrep.simxGetVisionSensorImage2(clientID,
VisionSensortHandle, 0,vrep.simx_opmode_buffer);
[ErrorC,ResolColor,imagem]=vrep.simxGetVisionSensorImage2(clientID,Kinec_rgb_tHandle,0,v
rep.simx_opmode_streaming_split+4000);
a = now;
saiu = 0;
while(returnCode~=0)
    pause(2);
    [returnCode, detectionState, auxData,
auxPacketInfo]=vrep.simxReadVisionSensor(clientID,VisionSensortHandle,vrep.simx_opmode_s
treaming);
    [returnCode, detectionState, auxData,
auxPacketInfo]=vrep.simxReadVisionSensor(clientID,VisionSensortHandle,vrep.simx_opmode_b
uffer);
    [returnCode2,resolution,matrix]=vrep.simxGetVisionSensorImage2(clientID,
VisionSensortHandle, 0,vrep.simx_opmode_streaming);
    [returnCode2,resolution,matrix]=vrep.simxGetVisionSensorImage2(clientID,
VisionSensortHandle, 0,vrep.simx_opmode_buffer);

[ErrorC,ResolColor,imagem]=vrep.simxGetVisionSensorImage2(clientID,Kinec_rgb_tHandle,0,v
rep.simx_opmode_streaming_split+4000);
    b = (now - a)*3600;
    if(b>1)
        saiu = 1;
        break;
    end
    if(auxData(7)>0.5)
        pause(randi(3));
        [returnCode, detectionState, auxData,
auxPacketInfo]=vrep.simxReadVisionSensor(clientID,VisionSensortHandle,vrep.simx_opmode_b
uffer);
        [returnCode2,resolution,matrix]=vrep.simxGetVisionSensorImage2(clientID,
VisionSensortHandle, 0,vrep.simx_opmode_buffer);

[ErrorC,ResolColor,imagem]=vrep.simxGetVisionSensorImage2(clientID,Kinec_rgb_tHandle,0,v
rep.simx_opmode_streaming_split+4000);
        cont=1;
        while(auxData(7)<0.7 & cont<5 )
            pause(randi(3));
            [returnCode, detectionState, auxData,
auxPacketInfo]=vrep.simxReadVisionSensor(clientID,VisionSensortHandle,vrep.simx_opmode_b
uffer);
            [returnCode2,resolution,matrix]=vrep.simxGetVisionSensorImage2(clientID,
VisionSensortHandle, 0,vrep.simx_opmode_buffer);

[ErrorC,ResolColor,imagem]=vrep.simxGetVisionSensorImage2(clientID,Kinec_rgb_tHandle,0,v
rep.simx_opmode_streaming_split+4000);
            cont=cont+1;
        end
    end
end

if (saiu == 1)

VerificaTemperatura(clientID,QuadricopterHandle,Kinec_rgb_tHandle,VisionSensortHandle,
idOrigem, angle, xyz,port);
else
```

```
    %inicializando a variavel que sera carregada do arquivo
    honeycomb_temp_read.temperatura_3pi_6 = 0;
    honeycomb_temp_read.leitura_3pi_6 = [];
    honeycomb_temp_read.temperatura_pi_6 = 0;
    honeycomb_temp_read.leitura_pi_6 = [];
    honeycomb_temp_read.temperatura_menospi_6 = 0;
    honeycomb_temp_read.leitura_menospi_6 = [];
    honeycomb_temp_read.temperatura_menos3pi_6 = 0;
    honeycomb_temp_read.leitura_menos3pi_6 = [];
    honeycomb_temp_read.temperatura_menos5pi_6 = 0;
    honeycomb_temp_read.leitura_menos5pi_6 = [];
    honeycomb_temp_read.temperatura_5pi_6 = 0;
    honeycomb_temp_read.leitura_5pi_6 = [];

    temperatura = auxData(7);
    if (temperatura > 0.7)
        rtn = 1;
        filename = [num2str(idOrigem) '_honeycomb_temp_read.mat'];
        if exist(filename,'file')
            load(filename);
            honeycomb_temp_read;
            if(angle == (3*pi/6))
                honeycomb_temp_read.temperatura_3pi_6 = temperatura;
                honeycomb_temp_read.leitura_3pi_6 = imagem;
            elseif(angle == (pi/6))
                honeycomb_temp_read.temperatura_pi_6 = temperatura;
                honeycomb_temp_read.leitura_pi_6 = imagem;
            elseif(angle == (-pi/6))
                honeycomb_temp_read.temperatura_menospi_6 = temperatura;
                honeycomb_temp_read.leitura_menospi_6 = imagem;
            elseif(angle == (-3*pi/6))
                honeycomb_temp_read.temperatura_menos3pi_6 = temperatura;
                honeycomb_temp_read.leitura_menos3pi_6 = imagem;
            elseif(angle == (-5*pi/6))
                honeycomb_temp_read.temperatura_menos5pi_6 = temperatura;
                honeycomb_temp_read.leitura_menos5pi_6 = imagem;
            elseif(angle == (5*pi/6))
                honeycomb_temp_read.temperatura_5pi_6 = temperatura;
                honeycomb_temp_read.leitura_5pi_6 = imagem;
            end
            save(filename,'honeycomb_temp_read', '-v6');
        else
            % durante a primeira leitura armazena
            % temperatura e leitura rgb
            if(angle == (3*pi/6))
                honeycomb_temp_read.temperatura_3pi_6 = temperatura;
                honeycomb_temp_read.leitura_3pi_6 = imagem;
            elseif(angle == (pi/6))
                honeycomb_temp_read.temperatura_pi_6 = temperatura;
                honeycomb_temp_read.leitura_pi_6 = imagem;
            elseif(angle == (-pi/6))
                honeycomb_temp_read.temperatura_menospi_6 = temperatura;
                honeycomb_temp_read.leitura_menospi_6 = imagem;
            elseif(angle == (-3*pi/6))
                honeycomb_temp_read.temperatura_menos3pi_6 = temperatura;
                honeycomb_temp_read.leitura_menos3pi_6 = imagem;
            elseif(angle == (-5*pi/6))
                honeycomb_temp_read.temperatura_menos5pi_6 = temperatura;
                honeycomb_temp_read.leitura_menos5pi_6 = imagem;
            elseif(angle == (5*pi/6))
                honeycomb_temp_read.temperatura_5pi_6 = temperatura;
                honeycomb_temp_read.leitura_5pi_6 = imagem;
            end
            save(filename,'honeycomb_temp_read', '-v6');
        end
    end
end
end
```