Universidade Federal do Piauí
Centro de Ciências da Natureza
Programa de Pós-Graduação em Ciência da Computação

Mapeamento Topológico usando Informações de Sensores para *Cellbots*

Francisco Bruno de Sousa Rocha

Teresina-PI, Novembro de 2017

Francisco Bruno de Sousa Rocha

Mapeamento Topológico usando Informações de Sensores para *Cellbots*

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFPI (área de concentração: Sistemas de Computação), como parte dos requisitos necessários para a obtenção do Título de Mestre em Ciência da Computação.

Universidade Federal do Piauí - UFPI

Centro de Ciências da Natureza

Programa de Pós-Graduação em Ciência da Computação

Orientador: André Macedo Santana

Teresina-PI

Novembro de 2017

Francisco Bruno de Sousa Rocha

Mapeamento Topológico usando Informações de Sensores para *Cellbots*/ Francisco Bruno de Sousa Rocha. – Teresina-PI, Novembro de 2017-

105 p. : il. (algumas color.) ; 30 cm.

Orientador: André Macedo Santana

Dissertação (Mestrado) – Universidade Federal do Piauí - UFPI

Centro de Ciências da Natureza

Programa de Pós-Graduação em Ciência da Computação, Novembro de 2017.

1. Mapeamento Topológico. 2. Cellbots. I. Orientador. II. Universidade Federal do Piauí. III. Departamento de Computação. IV. Título

CDU 02:141:005.7

Francisco Bruno de Sousa Rocha

Mapeamento Topológico usando Informações de Sensores para *Cellbots*

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFPI (área de concentração: Sistemas de Computação), como parte dos requisitos necessários para a obtenção do Título de Mestre em Ciência da Computação.

Trabalho aprovado. Teresina-PI, 28 de novembro de 2017:

André Macedo Santana
Orientador

Professor
Ricardo de Andrade Lira Rabelo

Professor
Anderson Abner de Santana Souza

Teresina-PI
Novembro de 2017

A minha família,
por sempre estarem comigo em todos os momentos e por me apoiarem durante está longa trajetória.

Agradecimentos

Agradeço a minha mãe e meu irmão, pela confiança e incentivo. Por me apoiarem em momentos difíceis e me encorajarem.

Agradeço ao meu orientador, Dr. André Macedo Santana, por todos os conselhos, pela paciência e ajuda nesse período. E ao professor Dr. Ricardo de Andrade Lira Rabêlo, pela ajuda na qualificação e todo o mestrado.

Aos meus amigos que se preocuparam comigo nos tempos difíceis.

À FAPEPI pelo apoio financeiro para realização deste trabalho de pesquisa.

Resumo

Um dos grandes desafios da robótica é tornar um robô autônomo, ou seja, o robô deve navegar no ambiente sem a intervenção humana. Essa autonomia pode ser abstraída em cinco etapas hierarquizadas: Mapeamento do Ambiente, Localização, Planejamento de Caminho, Geração de Trajetória e Execução de Trajetória. Este trabalho propõe uma solução para a primeira etapa da navegação robótica, o mapeamento do ambiente. O mapeamento tem o objetivo de adquirir um modelo espacial do ambiente em que o robô se encontra, este ambiente é representado por meio de um mapa. Este trabalho faz uso de mapas topológicos, que são representados computacionalmente por um grafo, onde os nós representam regiões com informações sensoriais homogêneas e são conectados entre si por arestas. O grafo descreve os espaços livres para execução de tarefas. Por isso, o mapa topológico, devido a sua estrutura, é uma solução compacta para o armazenamento do grafo na memória e pode ser usado para resolução de problemas de alto nível, como planejamento de tarefas. O objetivo deste trabalho é implementar um sistema de mapeamento topológico usando informações de redes sem fio para um *cellbot*, afim de mostrar sua eficiência em mapear um ambiente *indoor*. Com a finalidade de reduzir os custos financeiros do projeto optou-se por utilizar um robô da arquitetura *cellbot*, um robô que utiliza um dispositivo móvel (smartphone) para processar informações. Geralmente já integrados ao smartphone dispomos de giroscópio, câmera e acelerômetro, componentes que podem ser usados na navegação robótica. O trabalho apresenta resultados do sistema de mapeamento topológico com informações de localização e sinais *WiFi*, executando em três cenários com complexidade crescente. Os resultados apresentados dos tempos de execução no trabalho em smartphones são positivos.

Palavras-chaves: informações de redes sem fio. mapeamento. topológico. *cellbot*.

Abstract

One of the greatest challenges of robotics is to make an autonomous robot, i.e., the robot must navigate in a environment without human intervention. This autonomy can be abstracted in five hierarchical steps: Environment Mapping, Localization, Path Planning, Trajectory Generation and Trajectory Execution. This work proposes a solution for the first stage of robotics navigation, environment mapping. The mapping aims to acquire a spatial model of the environment in which the robot is, the environment is represented by a map. This work makes use of topological maps, which are represented computationally by a graph, where the nodes represent regions with homogeneous sensory information and are connected by edges (links or arcs). The graph describes the free space for performing tasks. Therefore, the topological map, due to its structure, is a compact solution for the graph storage in memory and can be used to solve problems of high level, and task scheduling. The objective of this work is to implement a topological mapping system using wireless network information for a cellbot in order to show their efficiency in mapping an indoor environment. In order to reduce the financial costs of the project it was decided to use a robot cellbot architecture, a robot that uses a mobile device (smartphone) to process information. Usually already integrated into the smartphone we have gyroscope, camera and accelerometer components that can be used in robotic navigation. It is not necessary to buy control boards and sensors separately for the construction of the robot increase its monetary cost, requiring only a base with wheels to be attached to the smartphone. The paper presents results of topological mapping system with information of localization and *WiFi* signals, running in three scenarios with increasing complexity. The results presented from execution times on smartphones are positive.

Keywords: mapping. topological. wifi data. cellbot.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AP	*Acess Point*
API	*Application Programming Interface*
CellBot	*Cell roBot*
ID	*IDentity*
LCS	*Local Control Strategies*
MAC	*Media Access Control*
MCL	*Monte Carlo Localization*
OpenCV	*Open source Computer Vision library*
OV	*Odometria Visual*
RSSI	*Received Signal Strength Indication*
SO	Sistema Operacional
UFPI	Universidade Federal do Piauí
VO	*Visual Odometry*

Lista de símbolos

e	Aresta do mapa
G	Base informações teóricas de características
f	Categoria da característica
O	Classe, componente da tupla informação-característica
C	*Codebook*, mapa de *codewords*
c	*Codeword*, entrada do *codebook*
E	Conjunto de arestas do mapa
Q	Conjunto de características extraidas das imagens
V	Conjunto de vértices do mapa
x	Coordenada x plano cartesiano
y	Coordenada y plano cartesiano
σ	Desvio padrão das forças de sinal *WiFi*
eo	Erro da odometria
ev	Erro da visão
ew	Erro da *WiFi*
fp	*Fingerprint*
H	Grau de entropia
ID	Identificador *WiFi fingerprint*
Sf	Função de similaridade entre *fingerprints*
Sp	Função de similaridade entre *poses*
ss	Intensidade sinal *WiFi*
Γ	Largura e comprimento da aresta
$thsO$	Limiar de sobreposição odometria

$thsS$	Limiar de sobreposição sinais WiFi
P	Mapa de *poses*
W	Mapa de *WiFi fingerprints*.
$\mathbb{M}$	Mapa Topológico
μ	Média das forças de sinal *WiFi*
Ψ	Número máximo de *APS*
Ω	Número de classes
Λ	Número de leituras de força de sinal *WiFi*
θ	Orientação
pi	Ponto de interseção circunferência
p	*Pose*
$\in$	Símbolo de pertence
v	Vértice do mapa

Sumário

1 Introdução

A presença de robôs é uma realidade em vários ramos da sociedade e sua integração vem aumentando com o passar do tempo. O crescimento inicial baseou-se no esforço de automatizar as operações industriais procurando realizar e reproduzir determinadas tarefas automaticamente. Desde então, os robôs industriais foram aperfeiçoados para serem utilizados nas linhas de produção realizando trabalhos com alta precisão, agilidade e repetitividade, destacando-se para estas tarefas os robôs manipuladores. Os robôs também são utilizados para tarefas perigosas ou que precisam ser realizadas em ambiente insalubre, de difícil acesso ou com alguma condição de risco, usando para isso principalmente os robôs móveis.

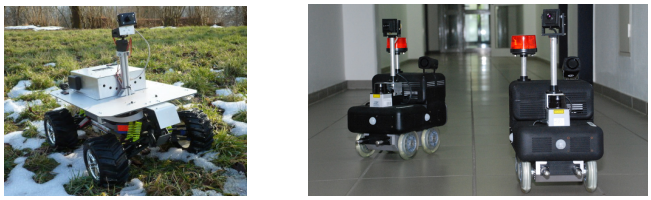
Além disso, robôs já têm aplicações em exploração submarina (WHITCOMB; YOERGER; SINGH, 1999) (KUNZ et al., 2008), exploração espacial (BARTSCH et al., 2010) (GOLDBERG et al., 2002), cirúrgicas (KANG et al., 2009) (CORCIONE et al., 2005), busca e resgate (DAVIDS, 2002) (KITANO et al., 1999), carros autônomos (THRUN et al., 2006) (PETROVSKAYA; THRUN, 2009), educação infantil (BENITTI, 2012), entre outras. Wolf et al. (2009) discorre que esse uso diversificado demonstra a grande gama de aplicações atuais dos robôs móveis e os interesses econômicos envolvidos em relação ao seu desenvolvimento e aplicação. Na Figura 1 alguns exemplos de aplicações.

(a) exploração subaquática (b) uso militar (c) automação industrial

Figura 1 – Algumas aplicações de robôs

A classe de robôs móveis tem capacidade de locomoção no espaço tridimensional (robôs aereoespaciais e subaquáticos) ou no espaço planar (robôs terrestres). As aplicações envolvendo robôs móveis vêm crescendo significantemente nos últimos anos devido a uma classe de tarefas que podem realizar, pois, ao contrário dos manipuladores, possuem capacidade de se locomover livremente pelo espaço de trabalho, limitando-se apenas a eventuais obstáculos. Existe inúmeras aplicações para os robôs móveis terrestres, elas dividem-se em duas classes de acordo com ambiente, o aberto (*outdoor*) (Figura 2(a)) e o interno (*indoor*) (Figura 2(b)). Foram separadas desta forma, porque de acordo com o ambiente de trabalho escolhido para robô os métodos usados para navegação, mapeamento,

localização e até mesmo em sua estrutura como rodas, formato, peso e entre outros, distinguem-se significantemente nestes dois ambientes.



(a) outdoor (b) indoor

Figura 2 – Ambientes de navegação.

O principal diferencial destes ambientes é o mapeamento. O desafio em mapear ambientes *outdoors* é maior comparado ao mapeamento de ambientes estruturados *indoors*, pois a complexidade do ambiente, a escala e a irregularidade do terreno tornam o problema difícil. Em ambientes *indoors*, como Horswill (1993) comenta eles raramente tem pisos irregulares e são normalmente constituídos por um ou mais planos bem definidos ligados por elevadores ou escadas e tem normalmente texturas uniformes devido aos revestimentos de azulejos, ladrilhos ou carpetes o que facilita o mapeamento do ambiente.

Independente do ambiente, como Santana (2011) esclarece, um fator essencial para um sistema robótico é torna-lo autônomo. A autonomia citada refere-se à capacidade de um robô manter um senso de posição e navegar sem intervenção humana. Assim, robôs autônomos devem apresentar reações às mudanças do ambiente, comportamento inteligente, integração entre vários sensores, resolução de múltiplas tarefas, robustez, capacidade de operar sem falha, programabilidade, modularidade, flexibilidade, expansibilidade, adaptabilidade e raciocino global.

Assim para trabalhar com um robô autônomo é necessário usar diversas técnicas computacionais, como navegação robótica, inteligência artificial, redes de sensores, etc.; além de conhecimento em mecânica e eletrônica. Neste trabalho é abordado o mapeamento robótico que é uma subárea da navegação robótica como mostrado na Figura 3 adaptada de Alsina et al. (2002). Em destaque na figura mostra que no mapeamento é necessário a localização do robô e para a localização é preciso do mapa do ambiente. Este impasse é conhecido como o problema da Localização e Mapeamento Simultâneos (*Simultaneous Localization and Mapping*, SLAM).

Figura 3 – Fluxo de dados navegação robótica (Adaptação de Alsina et al. (2002)).

Para realizar o mapeamento o robô escolhido neste trabalho é o *cellbot* (*Cell roBot*), um robô que usa um smartphone como principal computador de controle e sensoriamento. Geralmente já integrados no smartphone já dispomos de giroscópio, câmera e acelerômetro componentes que podem ser usados na navegação. Dispensando assim, a necessidade de lidar com os detalhes de microcontroladores ou outros dispositivos embarcados de computação. Outro beneficio é a redução do custo na construção dos robôs, pois, com o uso do smartphone, tem-se um sistema embarcado completo para navegação e com sua popularização, tornou-se um item comum de fácil aquisição com relação ao baixo custo. Mais detalhes sobre *cellbot* serão abordados nas Seções 2.1 e 3.1.

O sistema de mapeamento criado neste trabalho é o topológico, que cria um mapa do ambiente em forma de grafo. Ele trabalha em ambientes *indoors* estáticos, onde os objetos do cenário não se movem durante o mapeamento. Pelo sistema os nós do grafo podem ser agrupados pelo seu grau de similaridade em termos de informações sensoriais, quantidade de nós e distância. Com a identificação de portas do ambiente o sistema pode agrupar os grupos formados em novos, representam uma sala ou quarto. Neste trabalho, não abordaremos o método de identificação das portas, assim o teste do agrupamento é simulado e as portas são definidas empiricamente como entrada do sistema. O algoritmo criado para o mapeamento é capaz de trabalhar com os sensores que sejam modelados adequadamente para o sistema. Os dados capturados são utilizados para calcular a pose do robô no ambiente.

O trabalho apresenta resultados positivos em termos de tempo de processamento médio do sistema e eficácia no mapeamento topológico e nos agrupamentos desenvolvidos neste trabalho. Os resultados são divididos em testes de eficácia e em tempo de processamento que executam o mapeamento em três cenários com nível crescente de complexidade. Os testes são executados em três plataformas, computador e dois smartphone.

1.1 Motivação

Um robô autônomo precisa do mapa completo ou parcial para poder locomover-se no ambiente. Para mapear, o robô usa sensores acoplados para obter informações (características) relevantes no ambiente. O uso de diversos sensores aumenta a dificuldade no processo de desenvolvimento do sistema robótico. Nos trabalhos recentes, vem crescendo o uso da câmera como o principal sensor do robô, porque uma única imagem capturada pode adquirir-se, usando algoritmos de processamento de imagens, diversas características do ambiente como a detecção, distância e formas geométricas de obstáculos, reconhecimento, etc.

Além de fornecer a câmera, o smartphone oferece outros recursos como *GPS*, redes sem fio, giroscópio, acelerômetro, entre outros. Estes recursos podem ser usados para

navegação do robô tendo como vantagem que todos estes recursos estão unidos em um sistema embarcado.

O problema do mapeamento topológico é bastante abordado na literatura (Capítulo 2), por isso, existem várias técnicas e algoritmos que resolvem eficientemente o problema. Porém, é necessário avaliar a eficiência destas técnicas em um smartphone para testar a viabilidade de seu uso para resolver o problema e consequentemente o uso em um *cellbot*.

1.2 Objetivos

O objetivo deste trabalho é implementar e avaliar um sistema de mapeamento topológico em um *cellbot* afim de mostrar sua eficiência em mapear um ambiente *indoor* estático. Para a avaliação, o sistema é executado em um computador pessoal e alguns smartphones em diversos ambientes simulados. Avaliação entre computador e smartphone serve para destacar a diferença em tempo de execução entre arquiteturas diferentes, sendo o computador por definição mais potente em termos de quantidade de memória e capacidade de processamento. A comparação entre as execuções em diferentes smartphones objetiva mostrar o desempenho do sistema em capacidades diferentes de memória e processamento entre modelos de smartphones escolhidos.

O mapeamento topológico é testado em ambientes simulados variando de forma crescente o nível de complexidade (número de salas no ambiente) com o objetivo de testar a eficiência tanto do mapeamento topológico quanto dos agrupamentos. O sistema, com o uso do smartphone, objetiva obter um mapa do ambiente em forma de grafo que permite a execução de tarefas em alto nível como planejamento de rotas e estratégias de locomoção. O sistema usa um algoritmo de mapeamento topológico com informações extraídas de sensores acoplados ao *cellbot*.

1.3 Contribuições científicas

Este trabalho contribui com um sistema robótico para mapeamento topológico com o uso de informações de sensores em ambientes *indoor* estáticos para um robô de baixo custo monetário, um *cellbot*. Com isso, permitir que um robô mapeie o ambiente eficientemente e execute tarefas de alto nível. O sistema desenvolvido mostra para comunidade científica a capacidade executar sistemas complexos em um celular eficientemente, inspirando o avanço da pesquisa nessa área utilizando esta arquitetura robótica.

Para o mapeamento topológico este trabalho apresenta um algoritmo que combina as informações dos sensores, cria os nós e arestas do grafo e agrupa-os seguindo algum critério, como distância sensorial entre os nós ou número de nós. Com os sensores modelados

matematicamente, o algoritmo compara as informações obtidas de um estado anterior ao atual e como resultado cria os nós e as arestas, caso alcance o critério de agrupamento os nós e arestas criados até então são agrupados. O agrupamento tem o objetivo de agrupar os nós com informações sensoriais similares, e assim tentar determinar regiões no ambiente.

O trabalho contribui com o sistema de mapeamento topológico executando em ambientes *indoors* com uso de informações extraídas de uma antena *WiFi* e da localização que fornece a *pose* (posição e orientação) do robô. Essas informações podem ser extraídas dos componentes do smartphone, tratando-os como sensores do robô. A antena *WiFi* já vem embarcada na placa do smartphone. E a localização usada pode ser a Odometria Visual (OV) calculada usando as imagens capturadas pela câmera do smartphone (NISTER; NARODITSKY; BERGEN, 2004). No entanto, algoritmo apresentado pode trabalhar com diversos outros tipos de sensores que estão embutidos no celular como magnetômetro, giroscópio, acelerômetro ou sensores externos ao smartphone como sonares, *encoders*, *lasers*, entre outros.

Foi desenvolvido também dois aplicativos para smartphone. O primeiro, um aplicativo para executar o sistema de mapeamento topológico para capturar os tempos de execuções. O sistema executa tendo como entrada informações sensoriais de um ambiente simulado. Embora execute ambientes simulados, os erros usados na simulação foram extraídos em testes reais de ambientes *indoors*.

Dois outros aplicativos foram criados para capturar os sinais *WiFi* do ambiente em que se encontra. Um com a funcionalidade de medir o tempo de captura desta informação, mas que também pode servir futuramente para um sistema de localização *indoor* como em Fox et al. (1999). O outro foi usado para calcular o erro desta informação sensorial. O erro de localização usado na simulação foi do trabalho Bayramoglu et al. (2009) que usa um ambiente *indoor* semelhante aos testados nos experimentos. Este erro foi inserido para dar mais veracidade a simulação.

1.4 Organização do documento

O restante do trabalho encontra-se organizado como descrito a seguir:

- O Capítulo 2 apresenta o referencial teórico usado neste trabalho e os trabalhos relacionados.
- O Capítulo 3 descreve os materiais e métodos utilizados neste trabalho.
- O Capítulo 4 apresenta o sistema para resolver o problema do mapeamento topológico.
- O Capítulo 5 os experimentos e resultados.

- O Capítulo 6 são apresentadas as principais conclusões sobre o sistema e trabalhos futuros.

2 Referencial Teórico

Este capítulo apresenta o referencial teórico usado neste trabalho para o uso de cellbots, localização, informações de redes sem fio e para resolver o problema do mapeamento topológico.

2.1 *Cellbots*

Nas últimas décadas houve uma popularização dos smartphones. Crescimento justificado pelo barateamento, pelo tamanho compacto feito para caber na palma da mão, pelo conjunto de periféricos embarcados em sua placa como câmera, giroscópio, etc., pelo uso de programação de alto nível e pelo poder computacional. Por isso, os *Cellbots* por possuírem o smartphone como principal componente são bons por seu baixo custo monetário e eficiência, ainda mais quando ele obtém bons resultados (AROCA et al., 2013).

O número de aplicações computacionais usando os smartphones vem aumentando proporcionalmente e diversificando-se cada vez mais, como exemplo jogos, aplicações médicas, visão computacional, monitoramento cardíaco, realidade aumentada, etc. Na robótica não podia ser diferente. Nos últimos anos o smartphone vem sendo integrado ao robô como o principal meio computacional, que neste trabalho é usado para executar algoritmo de mapeamento topológico.

Em Aroca, Marcos e Gonçalves (2012) é apresentado um compilado de trabalhos desenvolvidos que usam robôs controlados por smartphones e ainda propõem um sistema de controle mecatrônico usando canais de áudio de dispositivos móveis (conector do fone de ouvido). Através dessa abordagem os motores do robô são controlados diretamente pelo smartphone, sem qualquer processamento intermediário. No trabalho o *smartphone* é usado como "cérebro"do robô, os comandos de ação são enviados pelo canal de áudio e as leituras dos sensores são recebidas também pelo canal de áudio usando somente sinais analógicos sem intermediários. A bússola e o acelerômetro são os sensores usados do celular e os externos são os *encoders* das rodas que é usado para calcular a odometria (Figura 4).

Na dissertação de mestrado Araújo (2013) apresenta uma metodologia para o ensino de Física com a robótica, sendo que em suas experimentações o aluno monta seu próprio *cellbot*. O trabalho Aroca et al. (2013) introduz o uso de um robô educacional flexível e acessível desenvolvido para experimentos práticos inerentes as disciplinas tecnológicas. No trabalho Buhl-Brown e D. (2015), também sobre robótica na educação, propõe projetar uma plataforma educacional robótica para *cellbots*.

Figura 4 – Robô usado em Aroca, Marcos e Gonçalves (2012).

No trabalho (BODENSTEIN et al., 2015) apresenta um robô autônomo controlado por um smartphone. Descreve um sistema de controle dinâmico que permite o robô navegar em volta de uma sala em direção a um alvo, evitando obstáculos. A posição do robô é estimada por um sistema visual. Usa um notebook para computação pesada, conectando com o celular via *WiFi*.

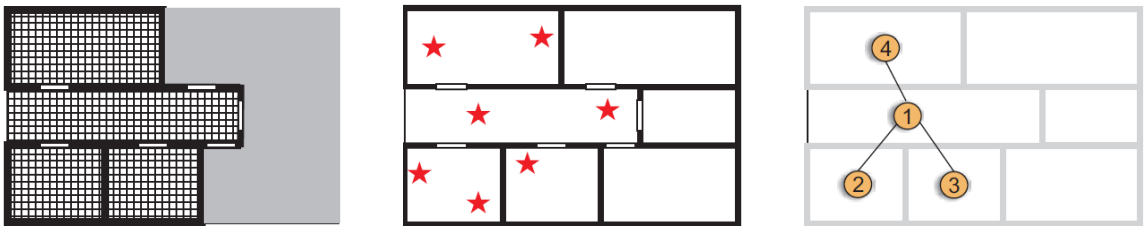
2.2 Mapeamento

O problema do mapeamento robótico é adquirir um modelo espacial do ambiente de um robô. Para construir o mapa o robô deve possuir sensores que o habilitam a perceber o mundo exterior (THRUN, 2003). Para construir o mapa um sistema robótico precisa determinar a posição e orientação (*pose*) do robô com algum referencial fixo no mundo. Thrun (2003) divide o problema do mapeamento robótico em cinco problemas. O primeiro é o movimento do robô está sujeito há erros, pois todos os sensores o estão e muitos são restritos a limitações de alcance.

O segundo problema do mapeamento robótico advêm da alta dimensionalidade das entidades que serão mapeadas, que pode necessitar de uma grande quantidade de memória para o mapa ser armazenado. O terceiro diz respeito à correspondência ou problema da associação de dados que determina se as medidas de um determinado sensor usando diferentes leituras no tempo correspondem ao mesmo objeto no mundo. O quarto está relacionado a ambientes dinâmicos que tem mudanças relativamente lentas como mudanças estruturais de uma construção, outras mais rápidas como o estado de uma porta e outras mais rápidas ainda como a localização de agentes em movimento no ambiente. O quinto diz respeito a qual caminho o robô deve escolher para explorar o mapa. Todos estes aspectos devem ser resolvidos para o correto mapeamento do ambiente, no entanto, em na maioria das vezes no quarto aspecto define-se no problema que o ambiente é estático para facilitar sua modelagem e implementação, como em Meyer-Delius et al. (2010) e Hahnel, Schulz e

Burgard (2012).

No problema de mapeamento robótico o ambiente deve ser armazenado em uma estrutura de dados e na literatura há três correntes de representação do ambiente, métrica Figura 5(a)(b), topológica Figura 5(c) e híbrida que é composta pelas outras duas. A Figura 5 foi retirada de Souza et al. (2014).



(a) métrica (b) métrica com características (c) topológica

Figura 5 – Representações de um mapa (SOUZA et al., 2014).

2.2.1 Mapeamento Métrico

Os mapas métricos podem ser construídos por grade de ocupação (MORAVEC; ELFES, 1985) (Figura 5(a)) ou por mapa de características (CROWLEY, 1985) (Figura 5(b)). Na representação por grade de ocupação consiste em usar um sensor para retornar a distância de um obstáculo dentro de um ambiente fechado, assim cada medida coletada traz uma informação associada sobre a existência ou não de um obstáculo, essa informação é então projetada num mapa bidimensional considerando o conhecimento da *pose* do robô. Um problema da representação métrica é o armazenamento das informações métricas que dependendo da dimensionalidade do ambiente e da resolução para representá-lo a quantidade dados pode ser muito grande. Moravec e Elfes (1985) expandiu essa representação para uma configuração discreta 3D. Yguel, Aycard e Laugier (2006) aborda os problemas da representação e armazenamento de dados para mapas grandes e propõe uma forma de representação em grade de ocupação baseada em wavelets (*Wavelet Occupancy Grids*). Hata, Shinzato e Wolf (2010) propõe a utilização de uma grade 2D construída a partir de nuvens de pontos 3D visando identificar regiões navegáveis de um determinado terreno.

No mapa de características são armazenados dados que descrevem alguma forma geométrica que podem ser encontrados no ambiente a ser mapeado, como pontos, retas ou figuras geométricas, por exemplo, triângulos, retângulos e círculos. Por isso, este tipo de mapeamento é mais usado em ambientes *indoor* (AMIGONI; GASPARINI; GINI, 2004)(PFISTER; ROUMELIOTIS; BURDICK, 2003).

2.2.2 Mapeamento Topológico

Os mapas topológicos são representados computacionalmente por um grafo uma estrutura de dados formada por nós, regiões com informações sensoriais homogêneas, que são conectadas entre si por arestas (elos ou arcos). O grafo descreve os espaços livres para execução de tarefas. Por isso, o mapa topológico é uma solução compacta para o armazenamento na memória e devido sua estrutura podem ser usados para resolução de problemas de alto nível como planejamento de tarefas. A localização do robô no mapa é abstrata, ou seja, não há como definir explicitamente qual a *pose* do robô, entretanto, pode-se afirmar em qual nó do grafo ou em qual região ele se encontra (SOUZA et al., 2014). A Figura 6 (SOUZA et al., 2014) ilustra intuitivamente o processo de construção de um mapa topológico de um determinado ambiente representado por uma planta baixa.

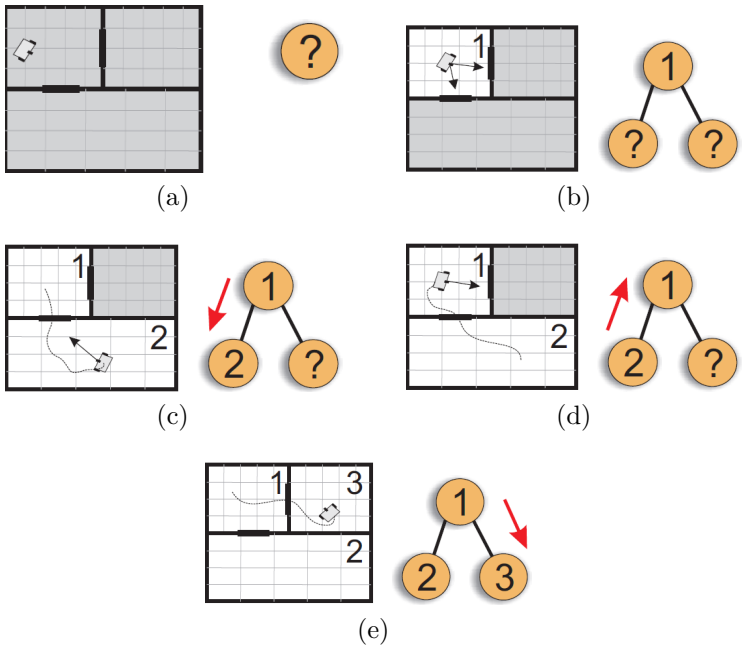


Figura 6 – Construção de um mapa topológico (SOUZA et al., 2014).

O mapeamento topológico possui algumas desvantagens ou problemas. Um problema é a ausência de um padrão de definição no mapa para poder ser considerado um vértice e quais relações serão descritas para serem utilizadas como arestas. Outro problema é a dimensionalidade do ambiente. Ambientes grandes e complexos podem apresentar informações sensoriais ambíguas que acarretam falhas na identificação de compartimentos já mapeados e de compartimentos não mapeados. Kuipers et al. (2004) usam os nós do mapa para representar lugares, caracterizados por dados sensoriais, as arestas para representar caminhos entre lugares, caracterizadas por estratégias de controle. Em Choset e Burdick (2000) e Choi et al. (2002) é gerado um mapa do espaço livre do ambiente através de diagrama de Voronoi e esqueletização, respectivamente. Fabrizi e Saffiotti (2000) e Galindo et al. (2005) usam um grafo que descreve a topologia do ambiente de trabalho do robô.

Kortenkamp e Weymouth (1994) desenvolve um mapa topológico para robôs móveis com uma abordagem que combina informações baseadas em sonar e informação visual para distinguir locais. Os sonares são usados para determinar onde capturar as imagens e usar as características usadas nessa imagem para ajudar no reconhecimento do local. A informação visual é adicionada a do sonar para reduzir a ambiguidade dos locais que parecem idênticos na visão dos sensores.

Romero e Cazorla (2012) apresenta um método para mapeamento visual usando informação topológica. Usa um algoritmo de combinação para pegar características da imagem e sua estrutura, então usa um método de comparação de imagens para construir o mapa topológico.

A partir de Fabrizi e Saffiotti (2000) a abordagem do mapeamento híbrido começou a aumentar nos trabalhos publicados sobre mapeamento. Os mapas híbridos são uma junção da representação topológica que é enriquecida com informações métricas de uma grade de ocupação. Cada trabalho apresenta meios diferentes de uso deste tipo mapeamento. No trabalho Fabrizi e Saffiotti (2000) os nós representam as salas e os corredores e as arestas as passagens entre as salas. Os nós e arestas armazenam informações métricas relativas ao ambiente que permitem a reconstrução do mapa de forma métrica. Em Thrun (1998) usa um mapa topológico obtido a partir de uma grade de ocupação probabilística particionada em regiões (nós) separadas por passagens estreitas (arestas).

2.3 Informações de Redes sem Fio

Redes *Wireless* se tornaram um componente necessário para as infraestruturas de redes e estão disponíveis em muitos ambientes corporativos (universidades, aeroportos, hospitais, etc.) e em muitas construções comerciais (restaurante, cinema, *shoppings*, etc.). Isto faz esse sistema atrativo em ambientes *indoor*(OCANA et al., 2005).

A informação usada das redes sem fio chama-se força de sinal recebida, *Received Signal Strength* (RSS) uma unidade arbitrária que indica o nível de força recebida pela onda de rádio ((IEEE. . . , 2009)). Ela pode ser obtida facilmente pela antena *WiFi* integrada ao smartphone.

Para cada leitura *WiFi* é retornado como resultado o conjunto de pontos de acesso, *Acess Point* (AP), em sua área de alcance com sua medida RSS. Para uso do mapeamento é montado uma tabela com a média e desvio padrão de cada AP a partir de um conjunto de leituras lidas de uma determinada posição. Com erro do sensor *WiFi* modelado é possível usar essa tabela como uma impressão digital (*fingerprint*) definindo assim, um identificador único para cada local.

O *fingerprint* servirá para ajudar a resolver o problema da criação dos nós e arestas

do grafo no mapeamento topológico. Os *fingerprints* são usados para distinguir locais do mapa auxiliando na criação de vértices evitando o remapeamento de locais, pois através da leitura RSS de determinado local é possível comparar com os *fingerprints* salvos e verificar se o local já foi mapeado.

Shin e Cha (2010) constrói o mapa topológico baseado em *fingerprint WiFi* para descrever uma trajetória de um usuário em ambientes *indoor* (Figura 7). O sistema associa títulos semânticos no mapa e estima a localização semântica do usuário usando um smartphone baseado na atual observação do *WiFi*. Usa os *fingerprints* para fazer a distinção dos locais para criar o mapa topológico.

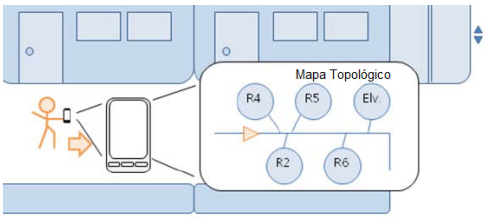


Figura 7 – Mapa topológico gerado a partir da trajetória do usuário(Adaptado de Shin e Cha (2010)).

2.4 Localização

Na localização os sensores do robô são usados para determinar a *pose*. Há diversas técnicas entre elas temos os trabalhos de: Santana et al. (2010) propõe uma técnica de localização em ambientes planos e fechado fundindo informação da odometria e processamento de imagem; Bodenstein et al. (2015) desenvolve um robô autônomo de baixo custo controlado por um smartphone. A *pose* é estimada a partir da odometria calculada pelos *encoders*. O câmera do smartphone é usada em um sistema de espelhos para o uso no sistema de controle dinâmico (Figura 8).

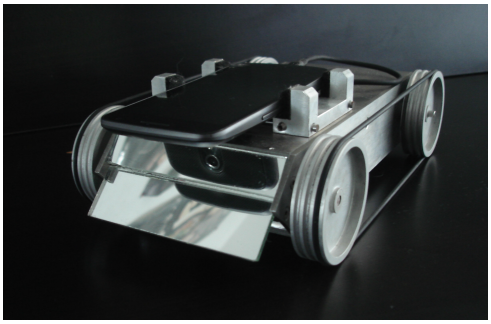


Figura 8 – Robô controlado por smartphone com um sistema de espelhos (BODENSTEIN et al., 2015).

Em Bayramoglu et al. (2009) desenvolve a navegação de um robô móvel com smartphone em um corredor usando Odometria Visual (OV) (Figura 9). O trabalho tem o

escopo de gerar a localização do robô através da fusão imagens extraídas do smartphone e os *encoders* das rodas. O robô opera em um corredor de um ambiente *indoor*.

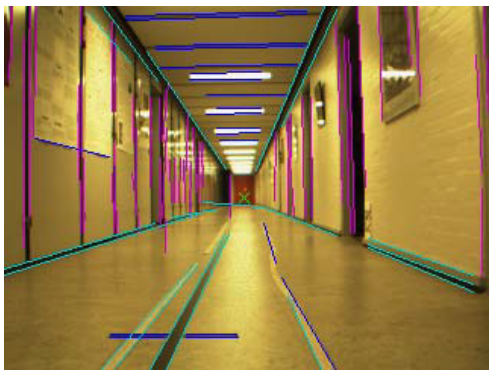


Figura 9 – Robô navegando em um corredor de um ambiente indoor usando OV para estimar a *pose* (BAYRAMOGLU et al., 2009).

Como mencionado para mapear é necessário a localização do robô, o problema do mapeamento e localização simultâneos. Na literatura sobre localização geralmente usa a odometria, mecânica e/ou visual, como entrada para os algoritmos para estimar a pose do robô. No entanto, em trabalhos recentes está se popularizando o uso de informações de sensores sem fio para localização em ambientes *indoor* devido ao uso de roteadores *WiFi* dentro de instalações públicas, privadas e comerciais.

Uma localização que usa o valor *RSS* é a Localização de Monte Carlo apresentada em Fox et al. (1999). Um método que usa uma abordagem para representar a incerteza que em vez de descrever a função de densidade de probabilidade em si, ela é representada por manter um conjunto de amostras (*samples*) que são randomicamente gerados e posicionados. O aplicativo mostrado na Seção 5.4 foi projetado para executar esta localização. Mas neste projeto é usado para medir tempo necessário para a antena *WiFi* ler a quantidade necessária de valores *RSS* dos APs do ambiente.

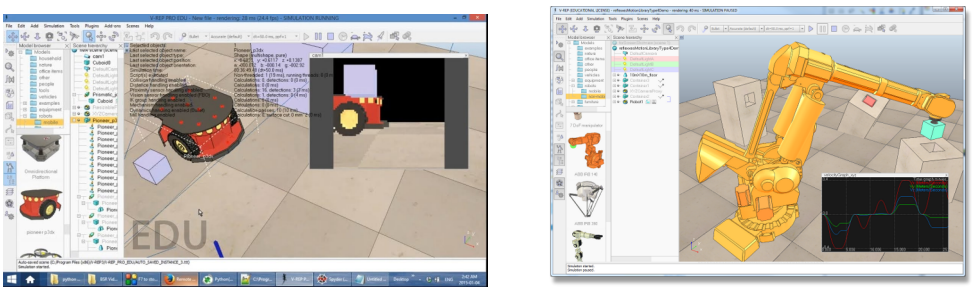
3 Materiais e Métodos

Neste capítulo serão apresentados os materiais e métodos que serão usados neste trabalho. Para simulação do sistema robótico a ferramenta *V-Rep* (V-REP, Acessado em Julho 2017) e para o mapeamento topológico um smartphone com sistema *Android*(ANDROID, Acessado em Julho 2017). Por último, a especificação do robô *cellbot*.

3.0.1 *V-Rep*

V-Rep é um simulador robótico com um ambiente integrado de desenvolvimento, baseado em uma arquitetura distribuída de controle. Cada objeto pode ser controlado individualmente via um código, um *plugin*, uma *API* (*Application Programming Interface*) remota, ou uma solução customizada.

O simulador tem suporte aplicações multi-robô. Seus controladores podem ser escritos em *C/C++*, *Python*, *Java*, *Lua*, *Matlab*, *Octave* ou *Urbi*. Ele pode ser usado nas aplicações: desenvolvimento de algoritmos para robôs, simulações de automação em fábricas, prototipação e verificação rápida, robótica na educação, etc. Na Figura 10 um pouco do seu ambiente de desenvolvimento.



(a) (b)

Figura 10 – Visão geral do *V-Rep*

Neste trabalho é construído um robô no simulador para extrair as informações do ambiente nele foram acoplados mecanismos para medir a *pose* exata do robô e as distâncias dos pontos de acesso *WiFi* (Figura 11). A partir das leituras das distâncias são calculados, considerando um ambiente livre de obstáculos, os sinais de força usando um modelo empírico chamado *Path-Loss* (MEHRA; SINGH, 2013), mais detalhes na Seção 4.1 subseção Processamento - *WiFi*.

Foram criados ambientes *indoors* com textura de parede e chão bem definidas com propósito de aumentar a eficiência na execução dos algoritmos de visão. As informações coletadas são armazenadas em bases de dados. Como são adquiridas do simulador essas

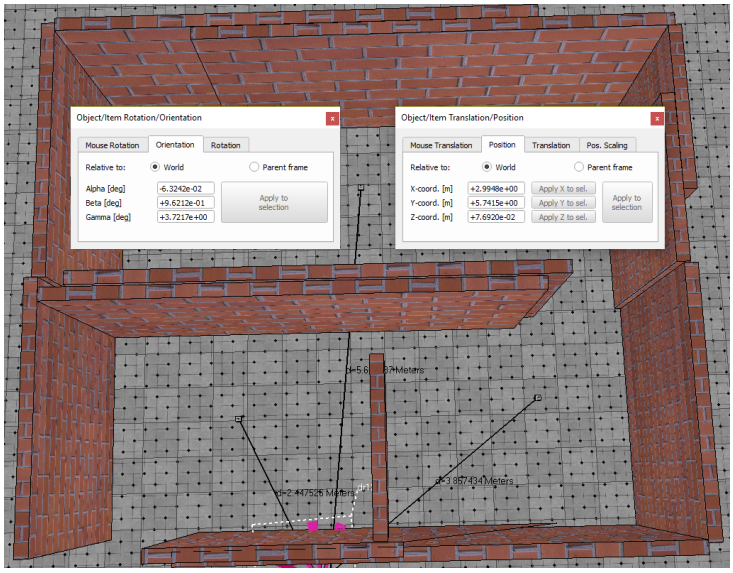


Figura 11 – Exemplo de captura de informações do *V-Rep*

informações são exatas, por isso erros randômicos são adicionados com referência as margens de erros extraídos de ambientes *indoors* reais, na Seção 4.1 subseção Processamento - Localização e *WiFi* demonstra como são adquiridos esses erros. Essas informações simuladas adicionados os erros são usadas como entrada pelo sistema de mapeamento topológico.

3.0.2 Plataforma *Android*

Android é um sistema operacional (SO) baseado em Linux e atualmente mantido pela empresa de tecnologia *Google*, algumas telas do sistema na Figura 12. Projetado principalmente para dispositivos móveis com tela sensível ao toque como smartphones e *tablets*, com interface específica para *TV* (*Android TV*), carro (*Android Auto*) e relógio de pulso (*Android Wear*).

Oferece uma *API* rica, disponibilizando fácil acesso a vários recursos de hardware, como Wi-Fi, GPS e acelerômetro, além de apresentar diversas ferramentas para o desenvolvimento de aplicativos. Na Figura 12 exemplo de smartphones com *Android*.

Os modelos de smartphones utilizados para os testes são mostrados na Tabela 1. Para comparação de desempenho com os smartphones foi usado um computador com as especificações da Tabela 2.

O aplicativo, chamado *TopologicalMap*, usado para executar os testes do sistema de mapeamento topológico pode ser visto na Figura 13. Em '*Choose Data*' o usuário escolhe o banco de dados usado no teste, as opções são: '*Small*', '*Medium*', '*Large*'; que fazem referência ao tamanho do banco de dados, mas indicam os ambientes com baixa, média e alta complexidades, respectivamente. Em '*Choose Mode*' é possível determinar qual parte do sistema será testado. Na opção '*FULL*', executa os testes do sistema completo, parte

(a) (b)

Figura 12 – Visão geral do *Android*.

Tabela 1 – Modelos de smartphones usados nos testes.

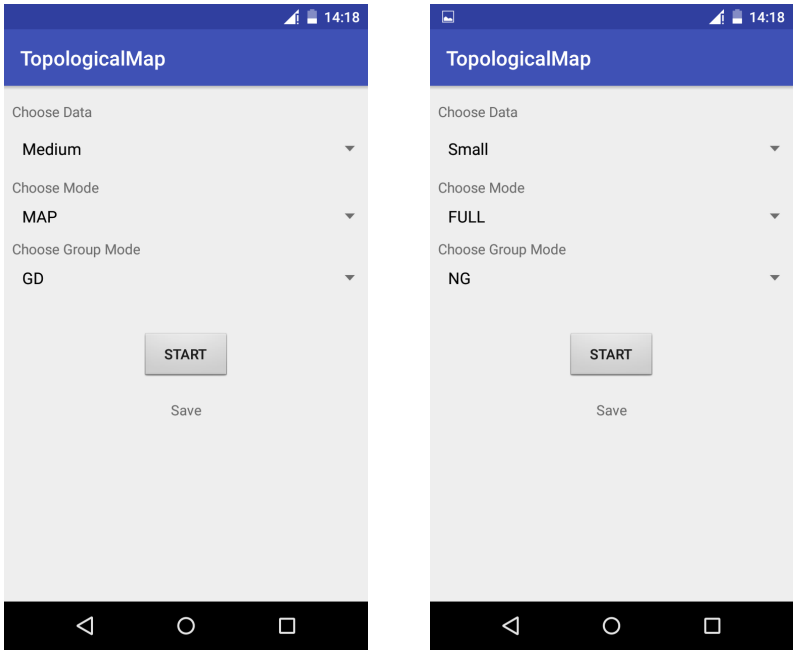
	Sistema Operacional	CPU	GPU	RAM
Motorola Moto G Dual Sim	Android 5.1.1 (Lollipop)	Quad-core 1.2 GHz Cortex-A7	Adreno 305	1 GB
Samsung Galaxy E7 Dual Sim	Android 5.1.1 (Lollipop)	Quad-core 1.2 GHz Cortex-A53	Adreno 306	2GB

Tabela 2 – Computador usado nos testes.

	Sistema Operacional	CPU	GPU	RAM
Dell	Windows 10 Home 64 bits	Intel Core i7-4500U 1.8 GHz	Intel(R) HD Graphics Family	8 GB

offline e *online*. Na opção '*TRNT*', executa a parte *offline*. E por último, a opção '*MAP*' executa a parte online. Em '*Choose Group Mode*' seleciona o tipo de agrupamento usado nos testes. Na opção '*NG*' indica sem agrupamento. A opção '*G*' ativa o agrupamento. E a opção '*GD*' ativa o agrupamento por salas.

Para medir o tempo de captura das informações *WiFi* foi usado o aplicativo chamado de *MuteDroid*. As telas do aplicativo pode ser visto na Figura 14. Ao iniciar o aplicativo é carregado o grafo de *Wifi fingerprints* dos arquivos, caso existam. O grafo pode ser usado para um sistema de localização *indoor* como em Fox et al. (1999). Considerando que não existam dados nos arquivos, para começar o mapeamento o usuário escolhe a opção de *Map* (Figura 60(a)), colocando a aplicação no modo *Mapa*. Nesta aba tem uma imagem do mapa Figura 60(b) que é proporcional ao tamanho real do mapa e possui a



(a) (b)

Figura 13 – Telas do aplicativo *TopologicalMap* com diferentes configurações para execução dos testes.

escala de 0,03050 metros por *pixel* da imagem. O usuário deve tocar no mapa da imagem no mesmo local que ele se encontra no mundo real para fazer a leitura do *Wifi fingerprint* corretamente, depois é criado um índice único para ele. É possível aplicar o zoom na imagem para facilitar o processo (Figura 60(b)). Esse processo pode ser executado repetidas vezes. As leituras e os tempos para adquiri-las são salvas em arquivos.

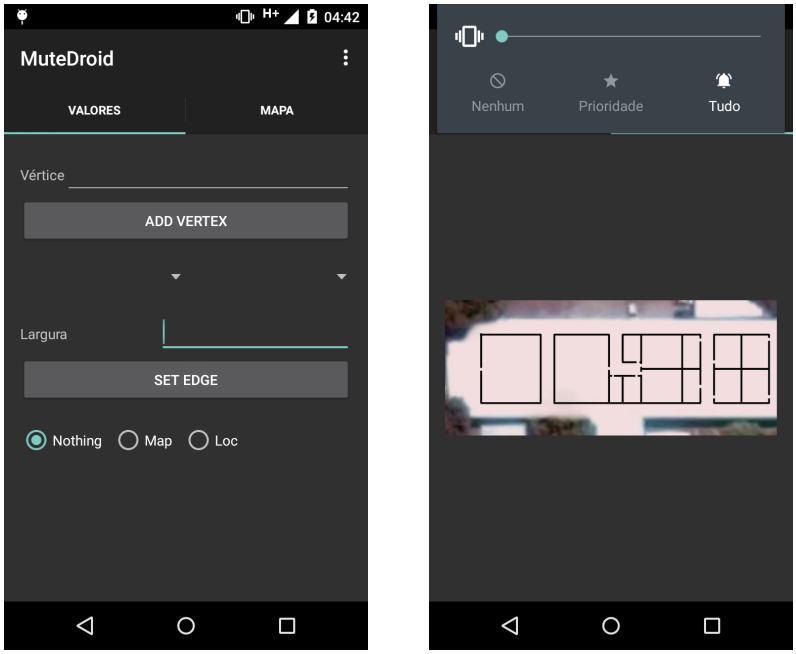


Figura 14 – Telas do aplicativo *MuteDroid*, (a) Tela para inserir informações para os vértices e arestas, (b) Tela com mapa para criar os *Wifi fingerprints* e (c) tela com um *zoom* no mapa.

3.1 *Cellbot*

Um sistema robótico *cellbot* possui muitas vantagens como mencionado na Seção 2.1 e por isso, além do baixo custo monetário, é um ótimo robô tanto para começar a desenvolver e testar os algoritmos nos primeiros momentos como para projetos finais robustos. Geralmente em um projeto de robótica um protótipo de robô móvel com rodas

possui diversas placas, para controlá-lo, sensores que variam de acordo com a aplicação, motores com *encoders*, fios de alimentação energética e comunicação e uma base para colocar tudo.

Essa complexidade para montar um robô ocasiona diversos problemas. O problema do erro ou atraso de comunicação entre as placas pode causar erro nos algoritmos e caso não tenham capacidade de corrigi-lo fará o robô ter um comportamento inesperado. E estas mesmas consequências são ocasionadas pela falha na comunicação entre os sensores e as placas. Além do trabalho de etiquetagem e agrupamentos de fios que conectam os diversos componentes do robô, da dificuldade de montagem e da manutenção. Esses problemas realçam ainda mais as vantagens do *cellbot* que possui quase todos os componentes necessários (memória, armazenamento, processamento e sensores) para o funcionamento do robô em um única placa pronta para o uso.

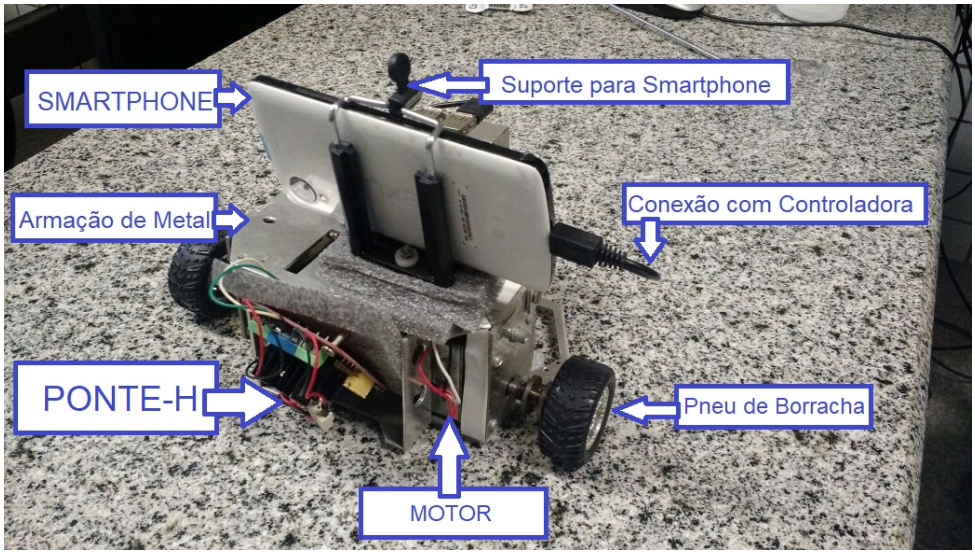
O robô é construído pelos componentes: uma base de ferro onde são anexados dois motores de corrente contínua; três rodas com *encoders* acionadas com os motores e sendo que uma roda é "boba" que serve para manter o equilíbrio do robô; uma placa controladora, no caso deste trabalho uma Raspberry Pi 3 (RASPBERRY, Acessado em Julho 2017), que faz a conexão e comunicação entre a ponte-H e o smartphone; um suporte para o smartphone; uma ponte-H para controlar a potência dos motores; um controlador de tensão entre as baterias e a placa controladora; e o smartphone. Os componentes do *cellbot* são mostrados nas Figura 15(a) e Figura 15(b).

O *cellbot* é montado colocando o smartphone no suporte e a placa controladora na base de metal, a conexão entre eles é feita por um cabo *mini USB*. Através de alguns fios (*jumpers*) é feita a conexão entre a placa e a ponte-H. A alimentação do sistema robótico é feita por três baterias, uma de 7v para alimentação da placa controladora, com um regulador de tensão entre eles, e as outras de 11v para os motores, com a ponte-H entre eles. E a terceira bateria a do próprio smartphone que pode variar de acordo com o modelo e só é usada para alimentação do mesmo.

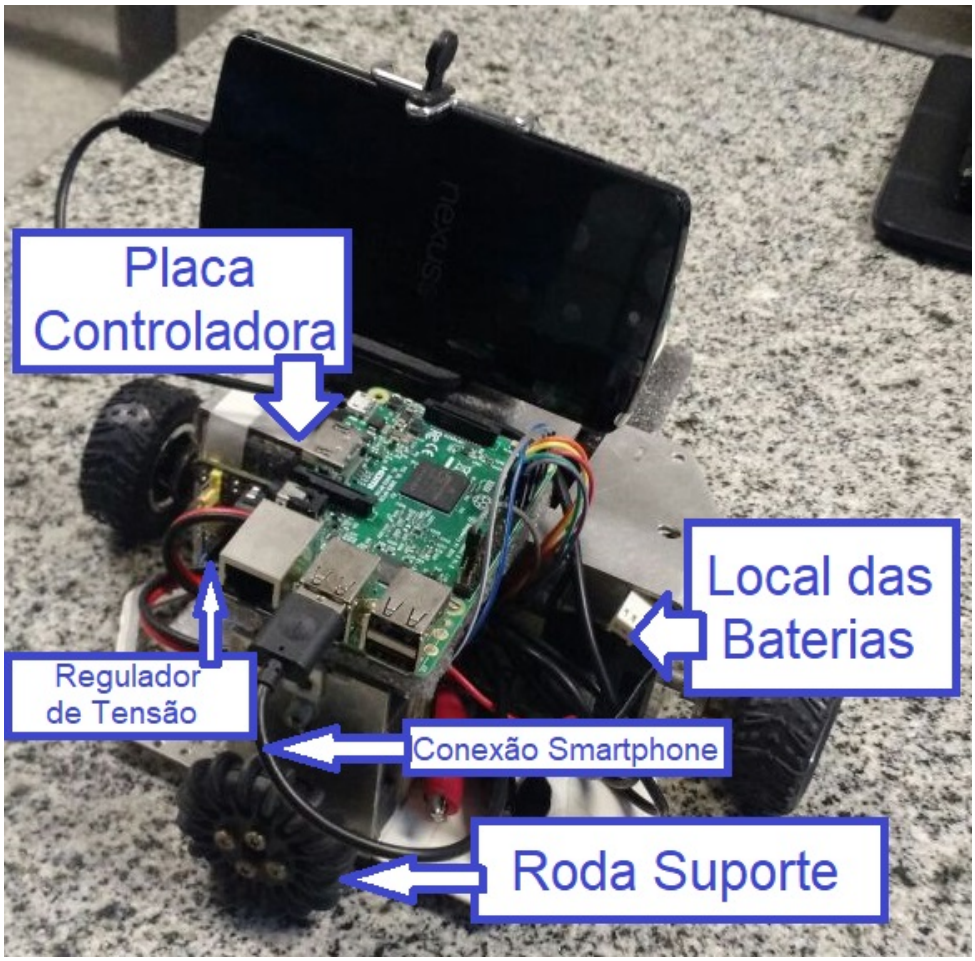
Com o sistema montado o smartphone faz o processamento e captura informações com seus sensores e envia as informações necessárias para os acionamentos das rodas, mas também é possível uma divisão de processamento entre o smartphone e a placa. Embora, neste trabalho, apenas o celular tenha sido usado para os testes do mapeamento topológico, o sistema robótico se encontra funcional pode ser controlado remotamente por outro celular ou um controle autônomo pode ser instalado tanto no *smartphone* quanto na placa controladora, mais detalhes no Capítulo 5 Resultados.

O maior custo do *cellbot* está nos motores e na placa controladora, que pode ser trocada por uma de baixo custo caso não necessite de processamento árduo designado a ela. O *smartphone* possui fácil acesso atualmente e a base pode ser construída com materiais de baixo custo, contanto que comporte os componentes do robô, pode ser feita de

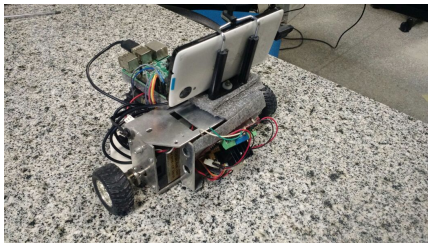
diversos materiais como metal, madeira, plástico, etc. Assim, esta subseção tem objetivo de mostrar a construção de um robô de baixo custo monetário que pode ser usado para diversos propósitos inclusive o mapeamento topológico.



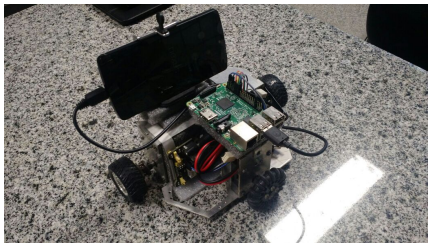
(a) Componentes da parte frontal do robô.



(b) Componentes da parte traseira do robô.



(c)



(d)

Figura 15 – *Cellbot* montado.

4 Mapeamento Topológico

Este trabalho desenvolve um sistema de mapeamento topológico usando a fusão de informações de redes de sensores sem fio *WiFi* e da localização, para ambientes *indoors* estáticos. A informação utilizada das redes *WiFi* são as leituras *RSS* dos APs. A informação da localização extraída é a *pose* do robô.

A *pose* do robô é necessária para mapeamento, assim uma informação obrigatória para o algoritmo, outras informações podem ser adicionadas ao algoritmo de sensores diferentes enriquecendo o mapa topológico. Para cada informação uma função de distinguibilidade que define o quanto estão sobrepostas entre si deve ser definida. Um limar (*threshold*) é usado para fazer a distinção de informações do mesmo tipo, pode ser adquirido através de experimentação. Essas informações são usadas como entrada do sistema, uma visão geral dele é mostrada na Figura 16.

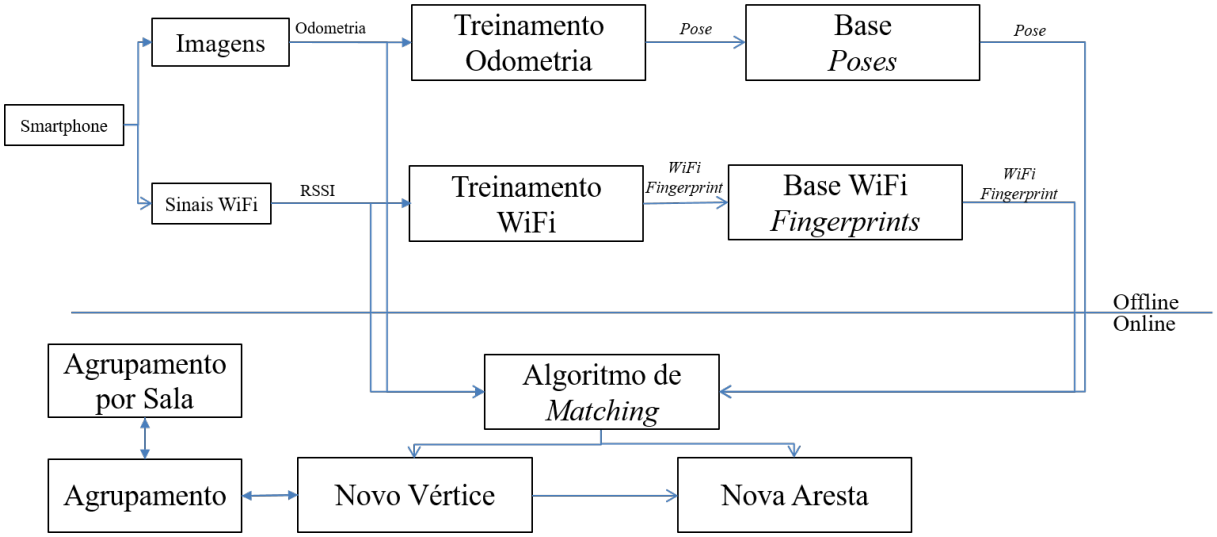


Figura 16 – Mapeamento Topológico - visão geral.

O sistema é dividido em duas partes: online e offline. As duas são independentes e podem ser executadas separadamente, em duas etapas, ou em paralelo, na execução da trajetória do robô.

Na etapa offline o robô percorre a trajetória seguindo algum algoritmo de exploração de ambiente e coleta os dados do smartphone que são direcionados para o processamento apropriado. É chamada assim porque o robô não cria o mapa nesta etapa. Neste trabalho o algoritmo usado é um controlador Fuzzy para um robô seguidor de paredes desenvolvido em Scaramuzza e Fraundorfer (2016). A navegação tem o objetivo de evitar colisões e manter o robô a uma distância segura da parede.

As informações processadas são armazenadas em sua base de dados. Na etapa

online o robô percorre a trajetória e cria o mapa topológico recolhendo as informações atuais e comparando-as com as armazenadas nos bancos de dados usando o algoritmo de *Matching*, veja em Algoritmo 1.

No outro método as duas etapas podem ser executadas juntas sequencialmente ou em paralelo, neste caso aproveitando o processamento multinúcleo dos celulares modernos. No sistema proposto as etapas online e offline são executadas sequencialmente para mapear o ambiente.

4.1 Etapa Offline

Nesta seção é abordado o funcionamento offline do sistema em três subseções correspondentes aos processamento mostrados na Figura 16. Os processamento são abordados detalhadamente nas próximas seções.

4.1.1 Processamento - Localização

Localização, no contexto de robótica, é o de determinar a *pose* de um robô relativo a um dado mapa do ambiente (THRUN; BURGARD; FOX, 2005). Em Bayramoglu et al. (2009) é desenvolvido a navegação de um robô em corredor usando Odometria Visual, são adicionados os erros adquiridos em seus experimentos a *pose* extraída do simulador.

Em Bayramoglu et al. (2009) apresenta uma OV para localização de um robô com smartphone. O método baseia-se em indentificar linhas retas ao longo do corredor, cuja largura e altura são assumidas como sendo conhecidas a priori. As linhas das imagens são então comparadas para obter as restrições de *pose*. A informação da *pose* é então fundida usando um filtro Kalman estendido. O erro de posição medido é mantido abaixo de 3cm e o erro de orientação é estimado abaixo de 1 grau.

A *pose* é definida como $p = (x, y, \theta)$, onde x, y e θ são posição no plano cartesiano e a orientação do robô respectivamente. E o erro de posição causado pela localização é definido por eo.

Devido ao erro da localização a *pose* lida pode ser diferente da *pose* real do robô. De fato, o robô pode está em qualquer lugar dentro do raio de erro da localização, veja a Figura 17 e a circunferência formada pelo erro. Assim, para o cálculo de distinguibilidade entre duas poses deve ser considerado os seus centros e raios.

Com as duas circunferências das poses, nas quais se desejam distinguir, é possível calcular os pontos de interseções das circunferências, no máximo 2, sendo definidos como

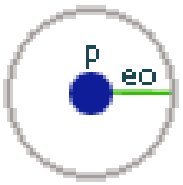


Figura 17 – Circunferência com centro a *pose* (p) lida e raio o erro da localização (eo).

pi, veja a Figura 18. Assim, o grau de distinguibilidade pode ser definido como:

$$\begin{cases} 1, & \text{se} número(pi) \text{ menor que } 2 \\ 1 - erf(\frac{A^2}{\pi r_1 r_2}), & \text{outros casos} \end{cases} \tag{4.1}$$

onde erf é uma função de erro, A é área de interseção entre as duas circunferências, veja a Figura 18, r_1 e r_2 os erros das duas *poses*. A Equação 4.1 calcula a probabilidade de o robô está na área de interseção. Para valores altos de probabilidade indica que as *poses* são distinguíveis.

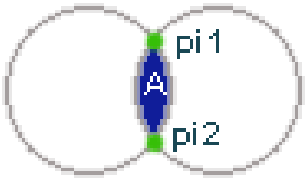


Figura 18 – Área de interseção A e os pontos de interseção $pi1$ e $pi2$ entre as circunferências.

Para agrupamento dos nós do grafo usando a informação da *pose* do robô é usada uma função de similaridade entre duas poses, que é definida por:

$$Sp_{a,b} = \frac{1}{||\overline{a} - \overline{b}||} \tag{4.2}$$

onde $Sp_{a,b}$ é o grau de similaridade entre a e b, $\overline{a}$ e $\overline{b}$ são as poses. Na Equação 4.2 a similaridade aumenta quanto mais próximo de 1.

4.1.2 Processamento - *WiFi*

Este processamento tem como entrada as leituras das força dos sinais *WiFi* dos *APs* que podem ser fornecidas pela antena do smartphone. Com essas leituras, é formado o *WiFi fingerprint* definido como $fp = (ID, \{ss_{i,j}\}, \mu_j, \sigma_j)$, onde ID é o identificador único do *fingerprint*. O $ss_{i,j}$ é a força de sinal i lida no AP j, sendo $i = 1, 2..\Lambda$, onde Λ é o número de leituras feitas e $j = 1, 2, ..\Psi$, onde Ψ é o número de APs detectados. O μ e σ são média e desvio padrão das leituras de cada AP, respectivamente.

As informações retornadas pelo simulador são as distâncias entre o robô e os APs do ambiente e são usadas para o cálculo das forças de sinais através do modelo empírico

Path-Loss (MEHRA; SINGH, 2013). O ambiente simulado é considerado, para efeitos de cálculo, livre de obstáculos, incluindo as paredes que causariam atenuação no sinal.

Para obter os erros das leituras das forças de sinais foi feito uma experimentação com o smartphone capturando informações da rede sem fio de um único *AP*. Com a distância do *AP* para o smartphone de 1 metro, sem obstáculos, foi medido a cada 10 centímetros a força de sinal deste AP. Com isso, foi comparado a força de sinal lida com a calculada do modelo.

O resultado do experimento pode ser visto na Figura 19. A nuvem de pontos são os valores *RSS* lidos e a linha o valor *RSS* calculado pelo modelo *Path-Loss* dado a distância do *AP*. Com a comparação entre os valores foi obtido um erro *RSS* médio de 4.635 com um desvio padrão médio de 3.138. Os erros foram adicionados randomicamente as leituras feitas na simulação.

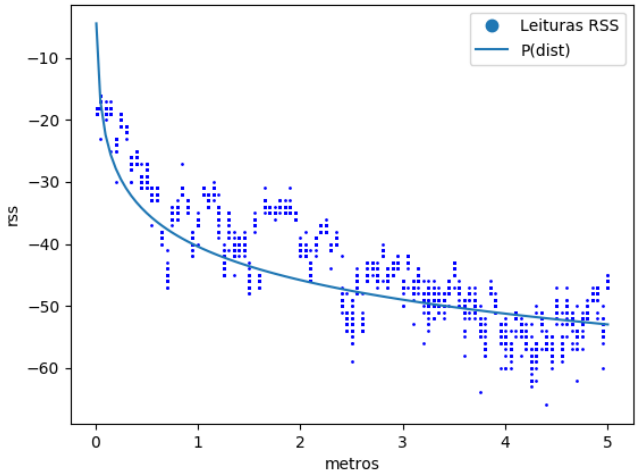


Figura 19 – Leituras *RSS* do *WiFi* nos pontos e valor *RSS* esperado pelo modelo *Path-Loss*(P(dist)).

O mapa topológico é composto por *fingerprints* distinguíveis. Quando dois tem uma distância grande em espaço de força de sinal, os dois *fingerprints* são distinguíveis. Shin e Cha (2010) define o grau de distinguibilidade como a possibilidade de que as distribuições de intensidades de sinal medidos em dois *fingerprints* não se sobrepõem. Assumindo que os dois tem o mesmo desvio padrão, a probabilidade de sobreposição é definida por:

$$1 - erf\left(\frac{\mu_1 - \mu_2}{2\sqrt{2\sigma}}\right) \tag{4.3}$$

onde erf é a função de erro. Com a Equação 4.3, dado dois *fingerprints* a alta probabilidade de sobreposição significa que são similares e a baixa indica que são diferentes um do outro. As leituras são comparadas com os *fingerprints* para distinção, caso não haja correspondência a nova *fingerprint* é adicionada a sua base de dados.

Para agrupamento dos nós do grafo usando a informação dos *fingerprints* do robô é usada uma função de similaridade entre dois *fingerprints*, que é definida por:

$$Sf_{c,d} = \frac{1}{||\overline{c} - \overline{d}||} \tag{4.4}$$

onde $Sf_{c,d}$ é o grau de similaridade entre c e d, $\overline{c}$ e $\overline{d}$ as médias de forças de sinal para cada ponto de acesso. Na Equação 4.4 a similaridade aumenta quanto mais próximo de 1.

4.2 Etapa Online - Mapeamento Topológico

O mapa topológico deste trabalho é um grafo definido em $\mathbb{M}$, com $\mathbb{M} = (V, E)$ onde V é o conjunto de vértices, E é o conjunto de arestas que os conecta. Para cada $v \in V$ corresponde a uma única localização no mapa e é definido por $v = (p, fp)$. Onde p é o conjunto de *poses* e fp é o conjunto de *fingerprints*. A aresta $e = (v_a, v_b, \Gamma)$, onde v_a e v_b são vértices e Γ é o peso da aresta. Assim, para cada $e \in E$ indica que há um caminho navegável entre os vértices v_a e v_b. O peso da aresta é definido por $\Gamma = (Sp, Sf)$, onde Sp é o grau de similaridade entre *poses* e o Sf entre *fingerprints*.

As criações dos vértices e arestas do mapa topológico são feitas na etapa online, como mostra na Figura 16. As informações recebidas pelo smartphone são comparadas com as informações das bases de dados pelo algoritmo de *Matching* que coordena a criação de vértices e arestas.

A chave para correlacionar as informações das bases de dados são seus erros. A posição do vértice é a *pose* obtida pela localização e não é criado um novo vértice até que o robô saia do erro máximo das informações. O erro é definido na Equação 4.5, onde eo é o erro da localização e ew é o erro do WiFi.

$$max(eo, ew, ...) \tag{4.5}$$

A medida que o robô percorre seu trajeto e reconhece lugares distintos através de suas medidas sensoriais ele cria novos vértices e arestas interconectando-os. Na Figura 20 simula a criação do grafo em ambiente com três salas e três passagens percorrendo um trajeto em formato de "L". As informações são simuladas em uma escala real, sendo 14 *pixels* da imagem equivalente a 1 metro. O ambiente possui 668,59 m^2 de área, a posição do robô e o erro da localização é dada em metros e a força de sinal *WiFi* é medida em valores *RSS* desprezando a resistência de obstáculos.

Ao iniciar o mapeamento, Figura 20(a), ele executa o processamento com as novas informações, como não existe nada nas bases de dados o vértice não é criado nesta iteração.

Na próxima iteração, não há uma mudança de erro grande comparado com as novas leituras, o algoritmo indica como sendo o mesmo local, mas como não há vértice com essas informações, então este é adicionado no grafo e o processamento é reiniciado com as novas leituras. Na Figura 20(b) o algoritmo também indica como um local não distinto, mas agora existe um vértice com estas informações. Assim, o grafo permanece inalterado e pula para a etapa de processamento.

Na Figura 20(c) o algoritmo já reconhece o local como distinto e adiciona um novo vértice com as informações lidas ao grafo. Como neste momento já existe um vértice anterior uma aresta é criada ligando os dois vértices. Essas etapas se sucedem até a Figura 20(d).

Para resolver o problema do mapeamento topológico foi desenvolvido o Algoritmo 1. O algoritmo resolve os dois dos cinco problemas do mapeamento divididos por Thrun (2003), Seção 2.2. O problema da associação de dados são resolvidos com a fusão sensorial dos sinais *WiFi* e da localização. O problema da dimensionalidade é resolvido pela característica do mapa que é topológico, assim ambientes são representados com pequena quantidade de dados. O mapeamento proposto é projetado para ambientes estáticos, assim não resolve o problema de ambientes dinâmicos. E o último problema, o da exploração, não é abordado neste trabalho.

O algoritmo tem como entradas o *fingerprint* lido definido por fp, a *pose* lida definida por p, o peso atual da aresta candidata definido por Γ e as informações das bases de dados são definidas por P e W, sendo P um mapa de poses, W mapa de *WiFi fingerprints*. Também é passado o grafo onde está sendo feito o mapa topológico.

Nas linhas 2 e 3 inicializa as variáveis que serão usadas. O $v\_anterior$ é o último vértice adicionado ao grafo. Da linha 5 à 11 se encontra o algoritmo de *Matching* para criação de vértices e arestas. Das linhas 12 à 14 se encontra a fase de atualização do algoritmo. Nas linhas 15 à 22 a condição de fechamento de ciclo no grafo (*closed loop*). Na linha 23 é feito processamento com as novas informações lidas.

O algoritmo de *Matching* é separado em três para cada tipo de informação sendo advindas da localização (Algoritmo 11) e do *WiFi* (Algoritmo 2). Caso as informações fornecidas não sejam distintas o suficiente de suas bases de dados, ou seja, um local já visitado em termos de informações sensoriais ocorre o *match*. Se não ocorrer o *match* e as leituras não forem completamente nulas ou se ocorrer o *match* e as informações não estiverem no grafo é criado o vértice e a aresta com essas informações e adicionados ao grafo. O $v\_anterior$ é atualizado com o novo vértice.

O algoritmo está preparado para funcionar mesmo quando com a informação da *pose* esteja disponível. A quantidade menor de informações, tem como consequência uma menor precisão do mapa criado, ou seja, nas localizações dos vértices e arestas criados.

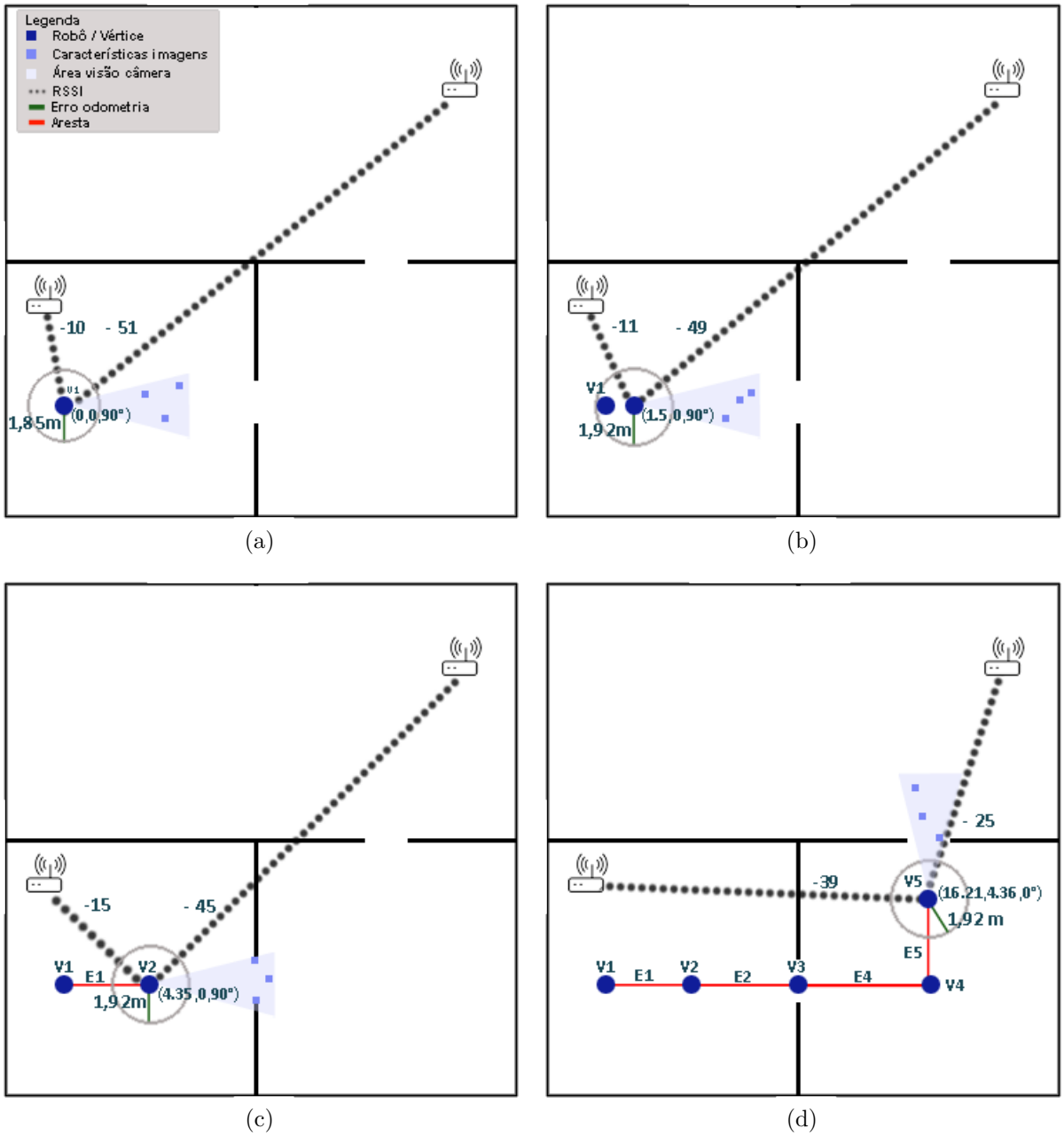


Figura 20 – Demonstração da execução do algoritmo de mapeamento topológico.

Entretanto, a medida que as informações são recebidas o mapa é atualizado aumentando assim sua precisão. Esta etapa é executada na fase de atualização do algoritmo. Este recurso tem como vantagem a adição de novos tipos de informações futuramente, por exemplo sonar, necessitando apenas que seja modelado seu algoritmo de *matching*.

O Algoritmo 11, *Matching* Localização, compara a *pose p* lida as poses *pi* da base de dados P calculando o grau de distinguibilidade de acordo com a Equação 4.1. Caso esse grau seja maior que o grau de sobreposição definido pelo limiar $thsO$ a leitura recebida se encontra na base de dados. Os limiares são definidos pelos experimentos.

O Algoritmo 2, *Matching WiFi*, compara o *WiFi fingerprint* (fp) lido aos *finger-*

Algoritmo 1: MAPEAMENTO TOPOLÓGICO

Entrada: $fp, p, eo, \Gamma, P, W, \mathbb{M}$
Saída: $\mathbb{M}$

início
  $v \leftarrow (P, F)$
  $e \leftarrow (v, v\_anterior, \Gamma)$
  `// Algoritmo de Matching.`
  $match \leftarrow$ max(MATCHING LOCALIZAÇÃO(p, eo, P), MATCHING WIFI(fp, W))
  se *(não match e não (p é* ***nulo*** *e fp é* ***nulo****)) ou (match e v não contido* $\mathbb{M}$*)* **então**
    $\mathbb{M} \leftarrow v$
    se *e não contido* $\mathbb{M}$ **então**
      $\mathbb{M} \leftarrow e$
    fim
    $v\_anterior \leftarrow v$
  fim
  `// Fase de Atualização.`
  se *match e v contido* $\mathbb{M}$ **então**
    ATUALIZA$(v, \mathbb{M})$
  fim
  `// Algoritmo Closed Loop.`
  se *match* **então**
    se *v contido* $\mathbb{M}$ *e v_anterior não é* ***nulo*** *e v_anterior diferente v* **então**
      se *e não contido* $\mathbb{M}$ **então**
        $\mathbb{M} \leftarrow e$
        $v\_anterior \leftarrow v$
      fim
    fim
  fim
  PROCESSAMENTO(fp, p, eo, W, P)
fim
retorna $\mathbb{M}$

prints (w) da base de dados (W) calculando o grau de distinguibilidade de acordo com a Equação 4.3. Caso esse grau seja maior que o grau de sobreposição definido pelo limiar $thsS$ a leitura recebida se encontra na base de dados.

Uns dos problemas da teoria de grafos é a identificação do *closed loop*. Uma "caminhada"em uma sequência de vértices, no mínimo três, que começam e terminam no mesmo vértice, caracteriza um *closed loop*. O Algoritmo 1 identifica o *closed loop*. Primeiro, verifica se o vértice criado a partir das informações lidas já existe no grafo indicando que o vértice está sendo visitado novamente. Depois, se existe um vértice anterior ao atual indica que há mais de um vértice no grafo. Por último, verifica se o vértice anterior não é igual ao atual verificando assim, se não está apenas voltando do local antigo. Caso a aresta não

Algoritmo 2: MATCHING LOCALIZAÇÃO

Entrada: p, eo, P
Saída: Boleana

```
1  início
2      para cada pi em P faça
3          Pi ← PONTOS INTERSEÇÃO(p, pi, eo, pi.eo)
4          A ← ÁREA INTERSEÇÃO(p, pi, eo, pi.eo)
5          err ← erf(A² / (π*eo*pi.eo))
6          se numero(Pi) < 2 ou err < thesO então
7              retorna verdadeiro
8          fim
9      fim
10 fim
11 retorna falso
```

Algoritmo 3: MATCHING WIFI

Entrada: fp, W
Saída: Boleana

```
1  início
2      para cada w em W faça
3          dMax ← fp.dw.d
4          err ← 1 − erf((fp.m − w.m) / (2.sqrt(2.dmax)))
5          se err menor igual thsS então
6              retorna verdadeiro
7          fim
8      fim
9  fim
10 retorna falso
```

exista é adicionada ao grafo, fechando o ciclo e depois atualiza o $v\_anterior$ para o v.

Ao término do Algoritmo 1 é realizado o processamento com as novas informações lidas (Seção 4.1). Observando que o processamento é offline não é necessário fazer-lo a cada iteração do algoritmo, podendo ser feito obedecendo uma determinada condição, por exemplo, intervalo de tempo para execução.

4.3 Agrupamento (Clustering)

Com objetivo de diminuir a densidade do grafo gerado pelo mapeamento topológico (Figura 21) neste trabalho os nós ou vértices podem ser agrupados seguindo até três critérios similaridade, número de vértices por grupo e distância. O agrupamento é realizado na etapa online do mapeamento, depois da criação do vértice (Figura 16). O algoritmo de agrupamento inicia com um grupo vazio onde é inserido os novos vértices que são criados

pelo mapeamento topológico. Caso um novo vértice em relação ao anterior atinja qualquer dos limiares dos critérios citados um novo grupo é criado para o novo e os próximos vértices até que um limiar seja alcançado novamente. A prioridade de agrupamento dos critérios seguem a ordem apresentada anteriormente da maior para a menor prioridade.

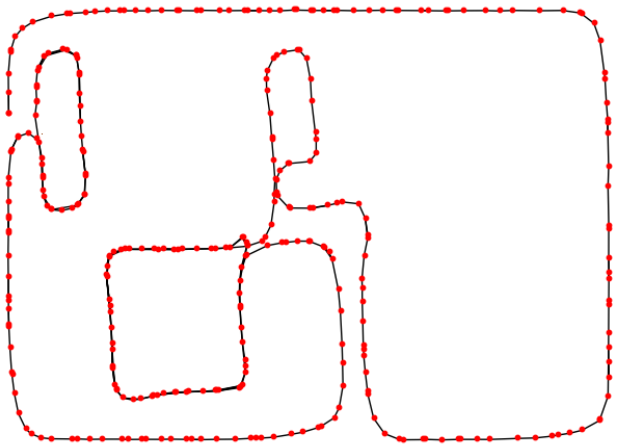
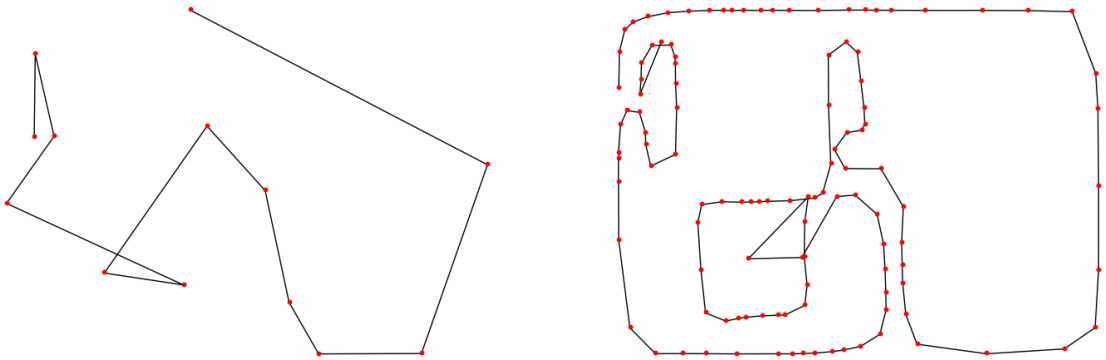


Figura 21 – Mapa topológico com densidade alta de vértices (pontos vermelhos).

Caso as informações sensoriais combinadas, neste trabalho a *pose* e forças de sinal *WiFi*, de dois vértices não ultrapassem o limiar de similaridade eles são agrupados. Caso contrario, um novo grupo é criado para comportar as novas informações sensoriais. Este agrupamento pode trazer vantagens agrupando vértices sensorialmente muito similares, no entanto se o limiar não for ajustado adequadamente pode causar a formação de poucos grupos com muitos vértices. Se o limiar for muito baixo e muitos grupos com poucos vértices se o limiar for muito alto. Veja na Figura 22 esses dois casos, o agrupamento é realizado sobre o mapa topológico da Figura 21.

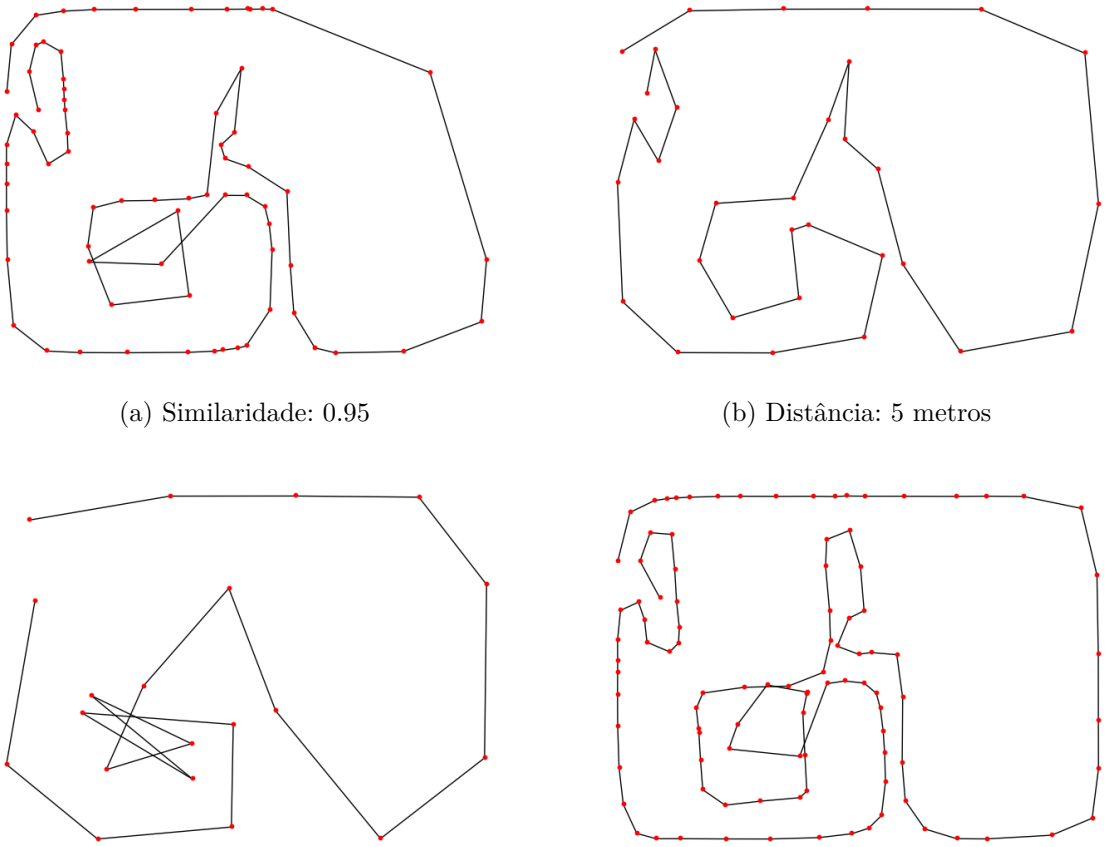


(a) Limiar de Similaridade: 0.8 (b) Limiar de Similaridade: 0.99

Figura 22 – Agrupamento seguindo o limiar de similaridade (a) baixo e (b) alto

Os limiares dos outros critérios de agrupamento são definidos empiricamente e

visam melhorar os grupos formados pelo critério de similaridade. O agrupamento pelo critério da distância é dependente da posição do vértice no mapa, caso dois vértices ultrapassem a distância euclidiana definida pelo limiar, um novo grupo é criado. O critério de número de vértices por grupo funciona de maneira similar, caso um grupo atinja o limiar, um novo grupo é criado. Os critérios podem ser usados em qualquer combinação para gerar os grupos do mapa topológico. Na Figura 23, exemplos de agrupamentos com os critérios funcionando separadamente e os três ao mesmo tempo.



(a) Similaridade: 0.95

(b) Distância: 5 metros

(c) Número de Vértices por Grupo: 10

(d) Sim.: 0.95, Dist.: 8 metros, Num. de Verts.: 5

Figura 23 – Exemplos de agrupamentos com os três critérios similaridade, distância e número de vértices por grupo.

Como objetivo de criar um grafo mais 'amigável' ao usuário do sistema, nós representando salas conectando-se a corredores e outras salas como exemplo a Figura 6, foi desenvolvido neste trabalho um agrupamento para este propósito e será definido aqui por agrupamento por salas. Este agrupamento é criado a partir dos grupos formados pelos critérios anteriores. O agrupamento por salas funciona de maneira diferente dependendo de onde o robô está localizado, sendo duas opções em um corredor ou dentro de uma sala. Ao explorar o corredor, o robô cria nós seguindo um critério de distância da mesma forma agrupamento anterior, mas também usa um critério de curva. Quando o robô faz uma

curva e ultrapassa um limiar angular em relação a sua posição anterior um novo grupo é criado.

Quando robô está no corredor e entra em uma sala, um novo grupo é criado e só é fechado quando o robô sair da sala. Assim, para o funcionamento do agrupamento por salas é necessário que o robô identifique as portas ou já tenha conhecimento de suas posições no ambiente. Neste trabalho as posições das portas são fornecidas ao robô, além disso, cada porta está associada a informação de conexão corredor-sala ou sala-sala. Na Figura 24 um exemplo de agrupamento por salas que é feito a partir do agrupamento mostrado na Figura 23 (d).Os pontos são vértices, as arestas em linha vermelha e em baixo do grafo uma planta baixa do ambiente mapeado, os ambientes dos testes são detalhados no Capítulo 5. O algoritmo de fechamento de *Loop* proposto neste trabalho não funciona para os agrupamentos feitos, sendo um recurso a ser concluído em trabalhos futuros.

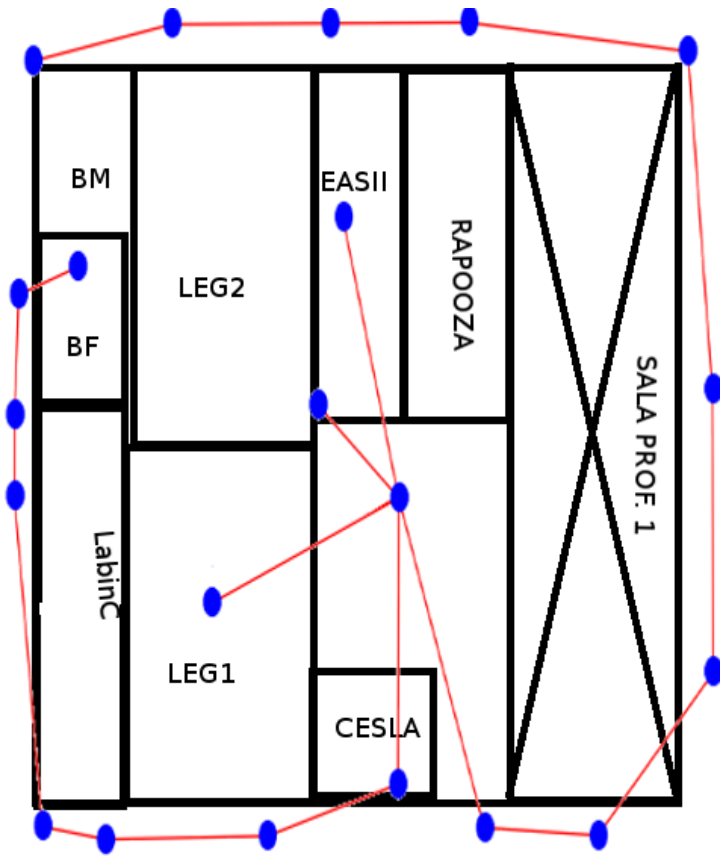


Figura 24 – Mapa topológico depois do agrupamento por portas, os pontos azuis vértices e as linhas vermelhas as arestas.

Depois do agrupamento por salas, os grupos representam os lugares na qual serão classificados semanticamente. A classificação semântica ajuda o usuário a reconhecer o local atual. O sistema tenta extrair um nome semanticamente significativo do local similar ao trabalho Shin e Cha (2010).

O método busca o ponto de acesso *WiFi* com o sinal mais forte do local atual. Se ele

possui um forte sinal assume-se que ele se encontra no local atual e o vértice é renomeado. Os pontos de acesso geralmente representam filiações que usam esse ponto, como exemplo laboratórios de pesquisa em um único prédio como: 'Raposa', 'Disnel', 'Cesla', entre outros. Assim, locais sem nome são nomeados com o nome do ponto de acesso mais próximo.

5 Experimentos e Resultados

Neste trabalho os resultados são apresentados em três cenários com nível de complexidade crescente. A complexidade do cenário é determinada pelo número de salas exploradas pelo robô em sua navegação no ambiente. Cada cenário está dividido basicamente em duas partes. A primeira mostra um Estudo de Caso do cenário avaliando resultados do mapeamento topológico desenvolvido variando os limiares e os resultados dos agrupamentos variando seus parâmetros. Ainda no Estudo de Caso, são apresentados resultados com as quantidades de vértices gerados com as variações. Apenas o Estudo de Caso do Cenário 1 é detalhado neste trabalho os outros Estudos de Casos dos cenários sobressalentes são encontrados nos Anexos. O motivo é explicado no Estudo de Caso do Cenário 1.

No segundo cenário, é feita uma análise dos tempos de execução para do mapeamento topológico com limiares e parâmetros fixos em todas as plataformas testadas. Por último, os resultados do experimento para estimar o custo da criação dos *WiFi fingerprints*.

5.1 Cenário 1

Nesta seção, o cenário de menor complexidade, o robô navega por todas as três salas do ambiente coletando dados de *pose* e distância dos *APs*. O ambiente simulado fictício, não possuí uma relação com um ambiente real, foi criado para ser o mapa inicial. A Figura 25 mostra o ambiente criado, possui uma grande sala com uma entrada. Essa sala é divida três outras sendo conectadas entre si por uma única porta. O ambiente possui 3 *APs* um em cada sala. Uma planta baixa do ambiente pode ser visto na Figura 26, onde é mostrada mais claramente a estrutura do ambiente.

Com o método de exploração do ambiente escolhido, o robô seguidor de paredes começa do lado de fora da grande sala, assim seguindo as paredes pelo lado direito ele navega por todas as salas. As informações coletadas do ambiente são guardadas em banco de dados. Os testes como os mapeamentos são executados lendo os bancos de dados. Esta operação ocorre igual para todos os cenários e esses banco de dados gerados são usados em todas as plataformas testadas neste trabalho, pc e smartphones. A Figura 27 é um exemplo de execução do sistema de mapeamento topológico no ambiente.

5.1.1 Estudo de Caso

No Estudo de Caso do Cenário 1 são apresentados os resultados de mapeamentos topológicos que mantém pelo menos um dos erros das informações coletadas no ambiente

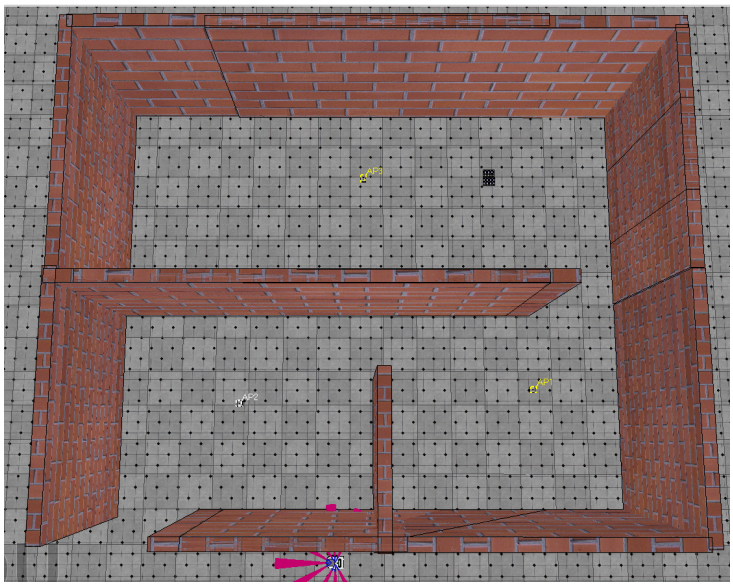


Figura 25 – Ambiente simulado no *V-Rep* usado no Cenário 1.

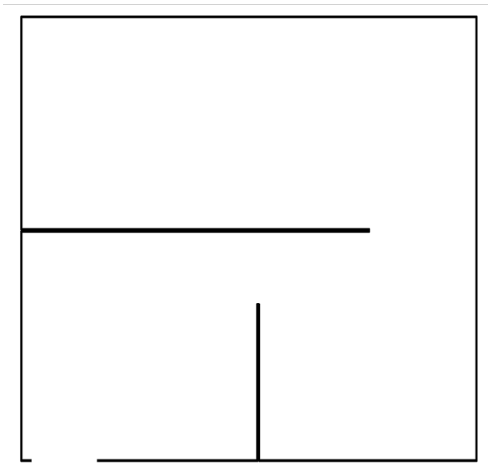


Figura 26 – Planta baixa do ambiente usado no Cenário 1.

ou parâmetros de agrupamento fixo. Esta subseção tem objetivo de mostrar a influência de mudança no mapeamento topológico variando determinado limiar ou no agrupamento alterando um único parâmetro. Esses testes também foram usados para uma boa seleção de limiares e parâmetros para os outros testes deste trabalho.

Para cada cenário temos um Estudo de Caso. No entanto, por apresentarem muitas similaridades em suas densidades de vértices nos grafos gerados pelo mapeamento topológico e similaridades nas execução dos testes os Estudos de Casos dos outros cenários estão em Anexos neste trabalho. Assim, este Estudo de Caso serve para ter uma visão geral das diferenças geradas por alteração dos limiares e parâmetros do mapeamento topológico e agrupamento, respectivamente, em todos os cenários, salvo algumas mudanças que podem ser encontradas nos Anexos.

Para o primeiro conjunto de testes é abordado com o erro de localização fixo em

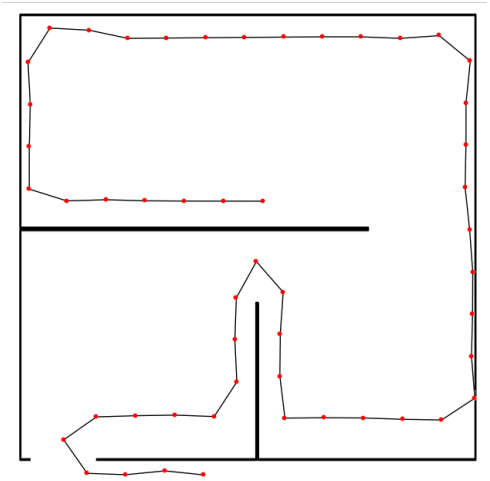


Figura 27 – Exemplo do mapa topológico gerado em cima da planta baixa do cenário. Erro localização: 0.3m. Limiar distinguibilidade localização: 0.1

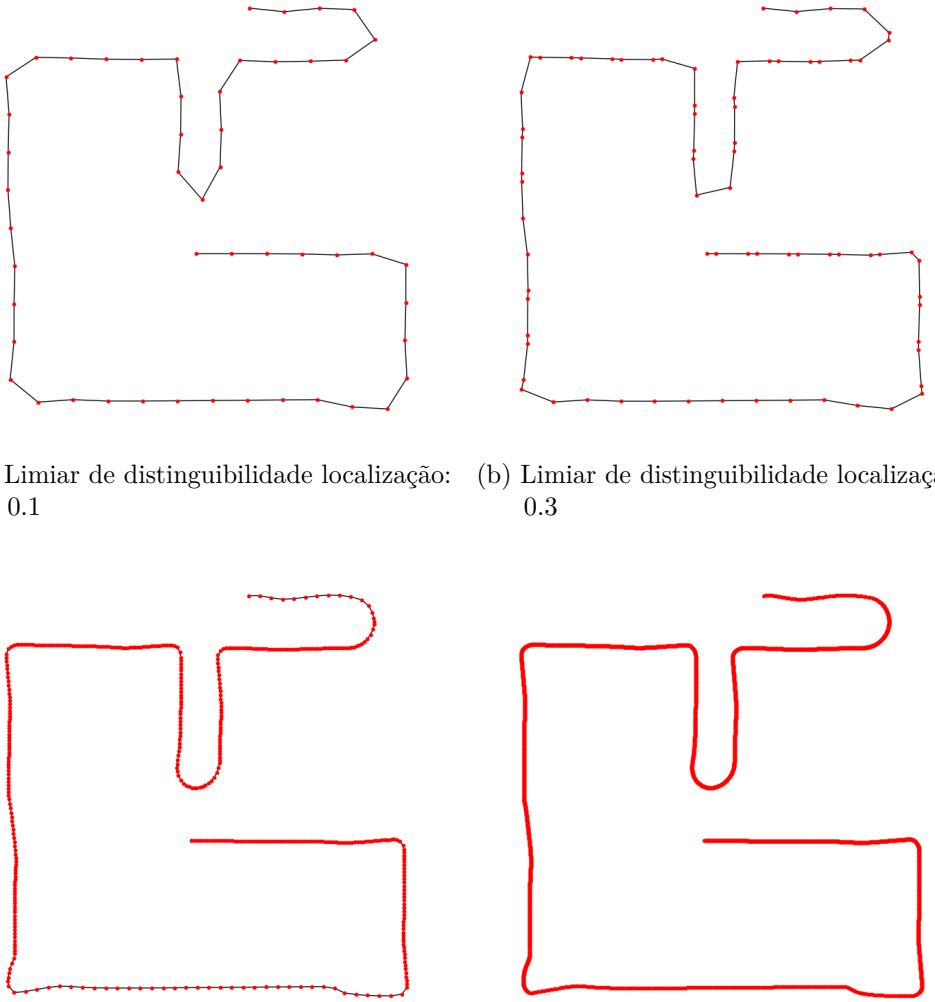
0.3m, variando o limiar de distinguibilidade da localização. Neste conjunto de testes, a informação *WiFi fingerprint* não é usada no mapeamento topológico. Assim, o grafo é criado apenas com as informações de pose. Os resultados dos testes são mostrados na Figura 28, em cada imagem o limiar de distinguibilidade usado no teste.

Um novo vértice é criado quando as informações coletadas no ambiente atualmente se diferem significativamente das informações armazenadas pelo vértice anterior criado. Quanto mais próximo de 1 o limiar de distinguibilidade da localização mais as informações coletadas são distinguíveis, porque dificilmente o grau de distinguibilidade alcança o limiar. Assim, por ter apenas a *pose* do robô como informação no mapeamento, a medida que o grau de distinguibilidade aumenta a quantidade de vértices no grafo também aumenta na mesma proporção, como pode ser visto na Figura 29.

O segundo conjunto de testes é abordado com o limiar de distinguibilidade fixo em 0.3, variando o erro de localização. Neste conjunto de testes a informação *WiFi fingerprint* não é usada no mapeamento. Os resultados dos testes são mostrados na Figura 30, em cada imagem o erro de localização usado no teste.

Duas *poses* próximas se distinguem menos a medida que o seus erros aumentam. Então, quanto menos os vértices se distinguirem entre si, menos vértices são criados no grafo do mapa topológico, como mostrado na Figura 30. O aumento do erro implica numa diminuição da quantidade de vértices (Figura 31), mas a medida que o erro aumenta, o erro do mapeamento topológico também aumenta. Na Figura 30(d) vértices são criados dentro de uma estrutura e um fechamento ciclo no grafo é formado erroneamente.

Para o terceiro conjunto de testes de mapeamento topológico usa o erro de localização 0.03m adquirido por OV em Bayramoglu et al. (2009), veja mais detalhes na Subseção Processamento - Localização na Seção 4.1. Neste caso, o limiar de distinguibilidade da localização é fixo em 0.3 e novamente a informação *WiFi fingerprint* não é usada. O



(a) Limiar de distinguibilidade localização: 0.1

(b) Limiar de distinguibilidade localização: 0.3

(c) Limiar de distinguibilidade localização: 0.6

(d) Limiar de distinguibilidade localização: 0.9

Figura 28 – Mapas topológicos gerados com o erro de localização fixo em 0.3m.

resultado é mostrado na Figura 32.

Erros de localização pequenos como 0.03m proporcionam uma localização do robô no ambiente com uma pequena margem de erro. Para o mapeamento topológico significa que vértices com posições maiores que esse erro são lugares distintos em termos de informações sensoriais. Assim, com um erro pequeno o mapa topológico fica preciso, mas em contrapartida apenas com a informação *pose* do robô, o grafo apresenta uma quantidade muito grande vértices como mostrado no resultado da Figura 32, na imagem (b) o grafo é ampliado para mostrar densidade dos vértices mais de perto.

Para os próximos conjuntos de testes são usadas as informações *WiFi fingerprint* e são adotados o erro da localização 0.03m e limiar de distinguibilidade da localização 0.1. O limiar baixo foi selecionado para diminuir a densidade do grafo gerado pelo mapeamento topológico, sendo a esta configuração final da localização para os testes.

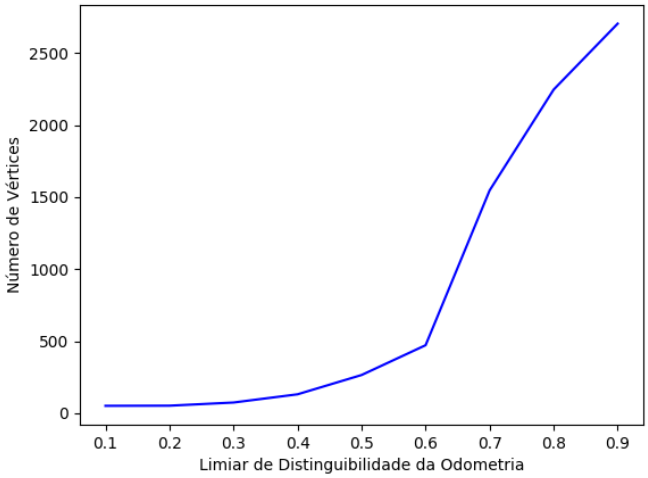


Figura 29 – Número de vértices em média criados com o erro de localização fixo em 0.3m.

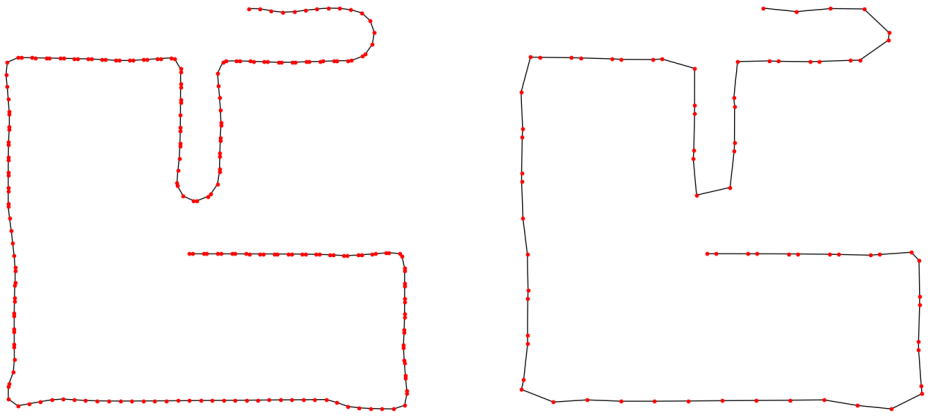
O quarto conjunto de testes usa erro *RSS* do *WiFi* fixo em 1, variando o grau de distinguibilidade do *WiFi*. Os resultados dos testes são mostrados na Figura 34, em cada imagem o limiar de distinguibilidade do *WiFi* usado no teste.

Como as informações de *WiFi fingerprint* e *poses* no sistema de mapeamento topológico a quantidade de vértices do grafo diminuiu significativamente, como mostra as Figuras 33 e 35. Com um erro *RSS* fixo a medida que o limiar de distinguibilidade do *WiFi* aumenta as informações tornam-se cada vez mais distinguíveis entre si, por dificuldade destas alcançarem o limiar determinado. Este efeito, gera um aumento na quantidade de vértices do mapa como a Figura 35 apresenta.

O quinto conjunto de testes mantêm o limiar de distinguibilidade *WiFi* fixo em 0.1, variando o erro *RSS* do *WiFi*. Os resultados podem ser visto na Figura 36, em cada imagem o erro *RSS* do *WiFi*. Erros *RSS* altos faz com que o *WiFi fingerprint* ocupe uma grande área em termos de informações sensoriais, com erro de localização baixo a quantidade de vértices no grafo cai abruptamente (Figura 37).

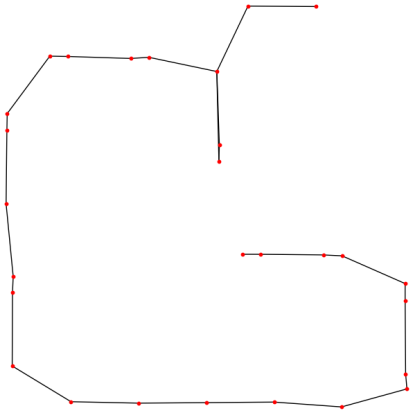
O sexto conjunto de testes usa o erro *RSS* do *WiFi* 4.635 ± 3.138 obtido em experimento neste trabalho, mais detalhes na Subseção Processamento - Wifi na Seção 4.1. Nos testes a variação é no limiar de distinguibilidade do *WiFi*. Os resultados são apresentados na Figura 38, em cada imagem o limar de distinguibilidade *WiFi* usado no teste. Mesmo com limiares altos não há muita diferença na densidade dos vértices do grafo gerado pelo mapeamento topológico. Os erros das informações utilizados são pequenos por consequência, os vértices possuem graus de distinguibilidade próximos de 0. Assim, mesmo com um grau de distinguibilidade das informações a 0.1 a quantidade de vértices do grafo se mantém quase constante com o aumento dos limiares de distinguibilidade (Figura 39).

A partir daqui os conjuntos de testes deste Estudo de Caso são focados nos

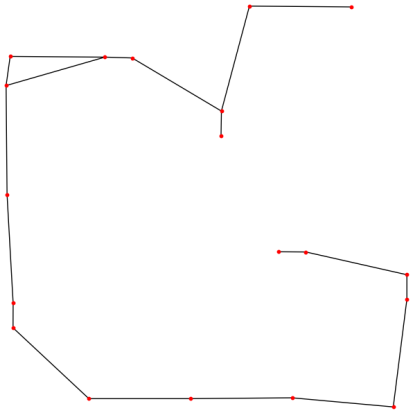


(a) Erro localização: 0.1

(b) Erro localização: 0.3



(c) Erro localização: 0.6



(d) Erro localização: 0.9

Figura 30 – Mapas topológicos gerados com limiar de distinguibilidade da localização fixo em 0.3.

agrupamentos. Os testes são realizados fixando e/ou variando um determinado parâmetro de agrupamento. A contagem dos conjuntos de testes não reinicia, assim o próximo é o sétimo. O agrupamento tem a função de diminuir a densidade perdendo o mínimo de informação do mapa topológico gerado. E também criar um agrupamento semântico dos vértices no caso deste trabalho, o agrupamento por salas.

O mapa topológico usado para os agrupamentos possui os parâmetros, selecionados a partir dos experimentos, erro de localização de 0.03m, grau de distinguibilidade da localização de 0.1, erro *RSS WiFi* de 4.635 ± 3.138 e grau de distinguibilidade do *WiFi* de 0.1. Esta é a configuração final para os testes fora do Estudo de Caso. O mapa topológico gerado por essas configurações é apresentado na Figura 38(a).

O sétimo conjunto de testes é o agrupamento apenas com o critério da similaridade, variando seu limiar. O resultados são apresentados na Figura 40, em cada imagem o limiar

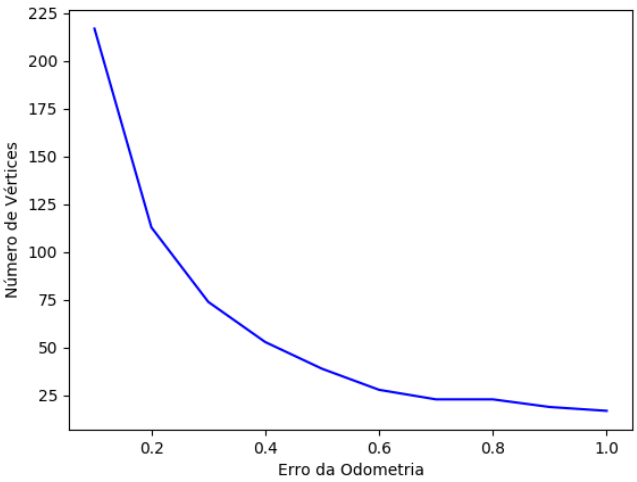
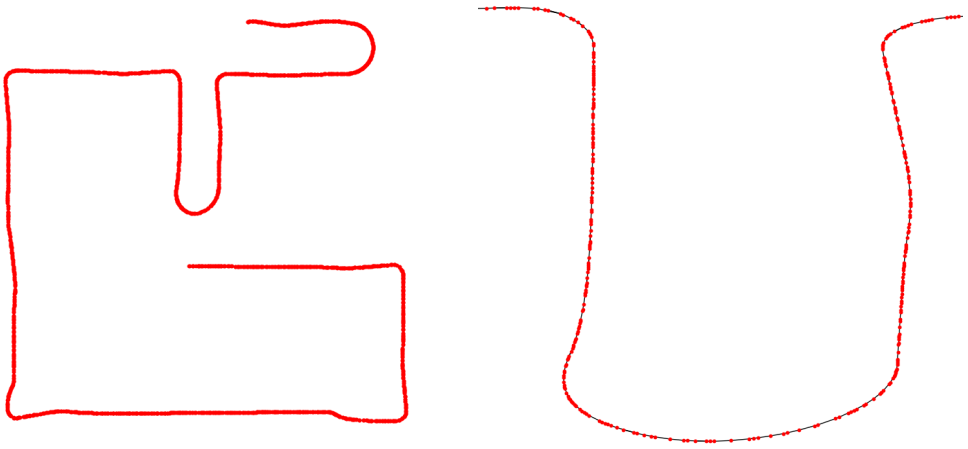


Figura 31 – Número de vértices em média criados com limiar de distinguibilidade da localização fixo em 0.3.



(a) Erro localização: 0.03m (b) Imagem aumentada

Figura 32 – Mapas topológicos gerados com limiar de distinguibilidade da localização fixo em 0.3.

de similaridade usado. Através dos experimentos detectou-se, que os vértices possuem um limiar de similaridade bem estreito. Os vértices tem um grau de similaridade acima de 0.165, assim, para limiares menores ou iguais, todos os vértices são agrupados em um único grupo. Com limiares acima de 0.18 o número de grupos criados sobe exponencialmente. A média com o número de vértices e grupos criados entre essa faixa é apresentada na Figura 41.

O oitavo conjunto de testes é o agrupamento apenas com o critério da número máximo de vértices por grupo, variando seu número. O resultados são apresentados na Figura 42, em cada imagem o limiar de número máximo de vértices usado. Por consequência a alta densidade do grafo, os grupos criados são quase equidistantes. Na Figura 43 a média

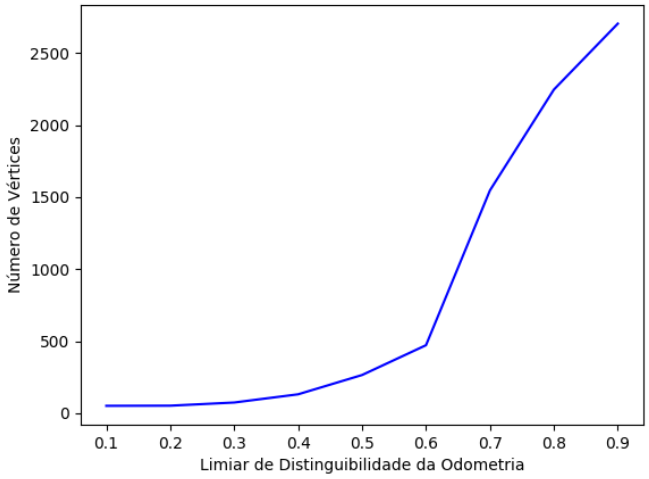


Figura 33 – Número de vértices em média criados com o erro da localização de 0.03m e limiar de distinguibilidade da localização fixo em 0.3.

com o número de vértices e grupos criados.

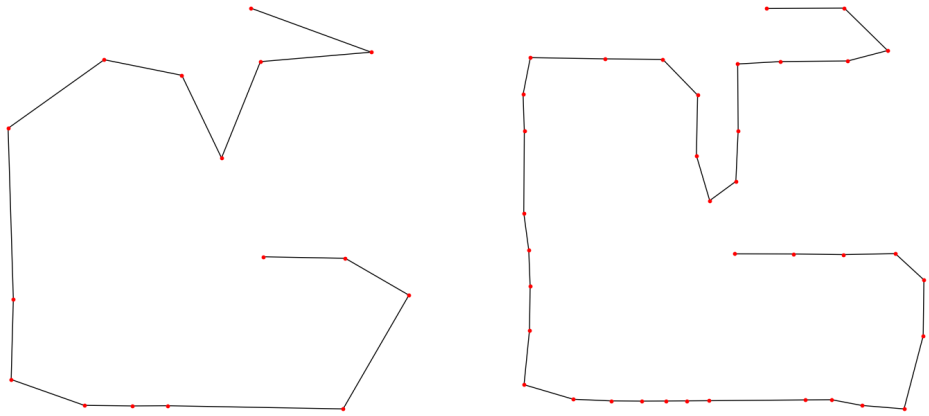
O nono conjunto de testes é o agrupamento apenas com o critério da distância máxima entre os vértices, variando a distância. O resultados são apresentados na Figura 44, em cada imagem o limiar de distância máxima entre os vértices usada. Com este critério é possível criar grupos equidistantes, a distância ideal para uma boa densidade vértices no grafo varia de acordo com ambiente usado. Na Figura 45 a média com o número de vértices e grupos criados.

O décimo, e último conjunto de testes deste Estudo de Caso, apresenta agrupamentos fixando o critério de similaridade em 0.17 variando os outros dois critérios. Os resultados são apresentados na Figura 46, em cada imagem os limiares usados no teste. Por fim, depois dos experimentos de agrupamento, a configuração de parâmetros final selecionada para os testes fora do Estudo de Caso são: limiar de similaridade 0.17, número de vértices máximos por grupo 10 e distância máxima entre os vértices 2m.

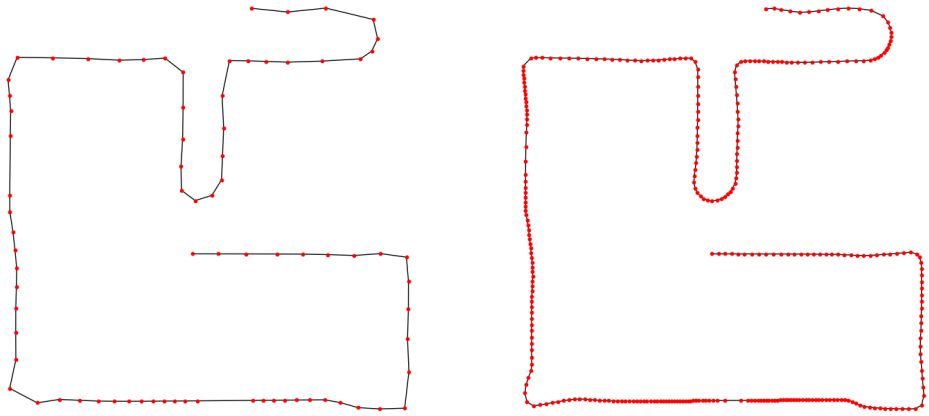
5.1.2 Resultados

Com os experimentos realizados no Estudo de Caso deste cenário foram selecionados limares para a execução do mapeamento topológico e os parâmetros para o agrupamento dos vértices do grafo gerado. Na Figura 47 é apresentado o mapa topológico gerado com os limiares selecionados e na Figura 48 o agrupamento com os parâmetros selecionados. Em sequência, na Figura 49 é apresentado os resultados do agrupamento por salas que é criado a partir do agrupamento anterior, onde cada vértice já está rotulado semanticamente com o nome do AP mais próximo.

O sistema de mapeamento topológico é executado utilizando os limiares e parâmetros



(a) Limiar de distinguibilidade *WiFi*: 0.1 (b) Limiar de distinguibilidade *WiFi*: 0.3



(c) Limiar de distinguibilidade *WiFi*: 0.6 (d) Limiar de distinguibilidade *WiFi*: 0.9

Figura 34 – Mapas topológicos gerados com erro *RSS* do *WiFi* fixo em 1.

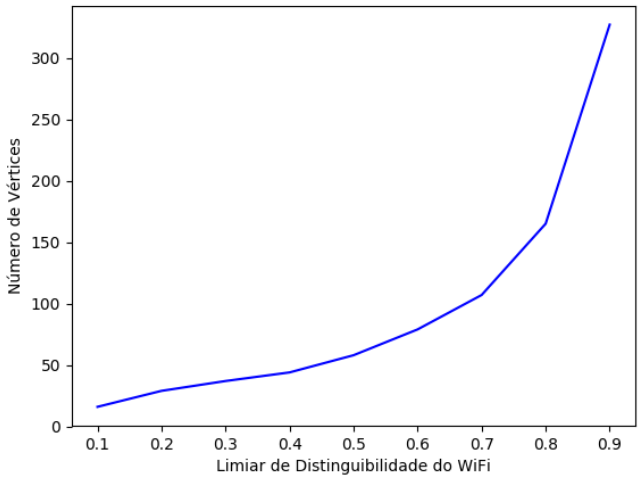
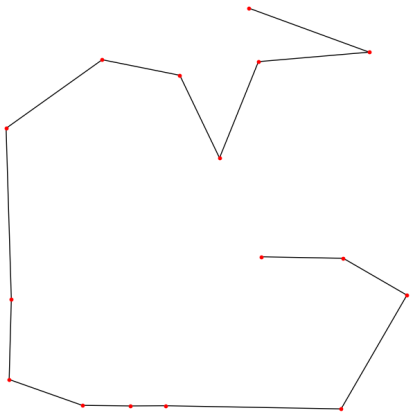
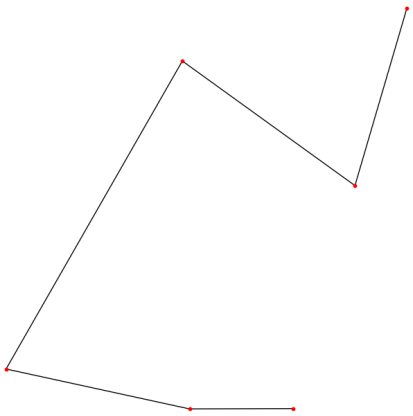


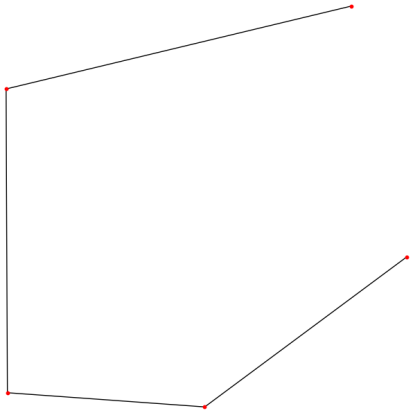
Figura 35 – Número de vértices em média criados com erro *RSS* do *WiFi* fixo em 1.



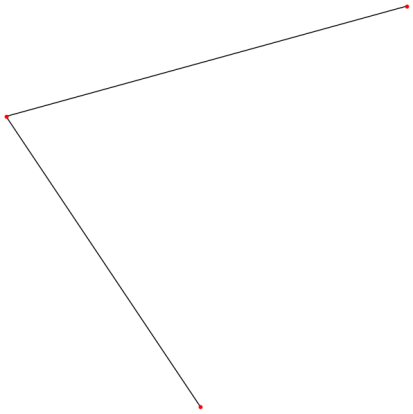
(a) Erro *RSS WiFi*: 1



(b) Erro *RSS WiFi*: 3



(c) Erro *RSS WiFi*: 6



(d) Erro *RSS WiFi*: 9

Figura 36 – Mapas topológicos gerados com limiar de distinguibilidade fixo *WiFi* em 0.1.

selecionados e depois é medido os tempos de execuções médios em segundos nas plataformas das Tabelas 1 e 2. São feitas 5 análises do sistema. Na primeira análise a etapa offline do sistema (Tabela 3) é avaliada separadamente. Na segunda análise o sistema completo é avaliado (Tabela 4). Pode-se calcular o tempo de execução médio da etapa online do sistema subtraindo os tempos das duas análises.

Os agrupamentos não influenciam na execução da etapa offline do sistema, assim os tempos médios com os agrupamentos são analisados apenas na etapa online. As três ultimas análises são feitas nos smartphones. É analisado o uso percentual do processador (*Cpu*) e da memória durante a execução do mapeamento topológico sem agrupamento (Tabela 5), com agrupamento (Tabela 6) e com agrupamento por salas (Tabela 7).

O sistema consegue executar a etapa offline em um tempo muito baixo em todas as plataformas em relação a execução completa do sistema. Com isso, a execução da etapa offline e da online é viável em termos de tempo de execução. Os desvios padrões dos

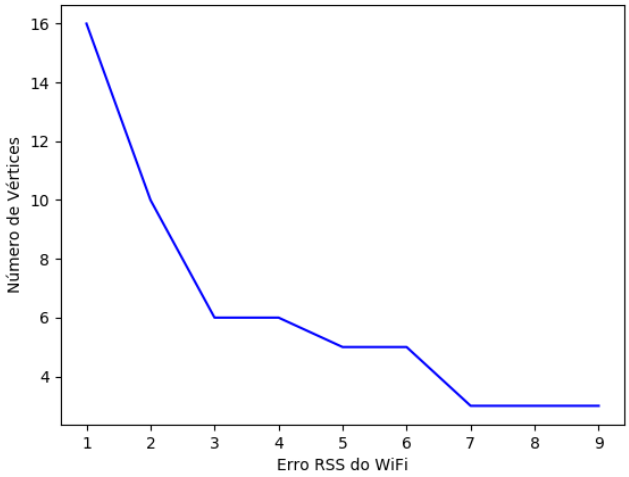


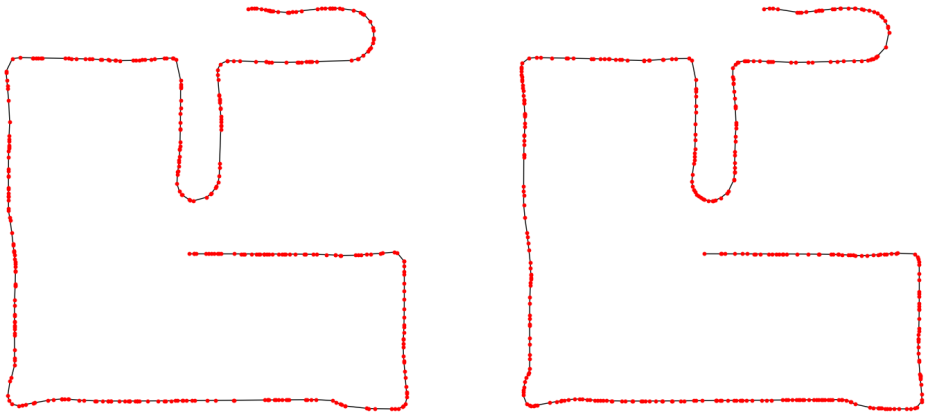
Figura 37 – Número de vértices em média criados com limiar de distinguibilidade *WiFi* fixo em 0.1.

Tabela 3 – Tempo(s) de execução médio da etapa offline do sistema de mapeamento topológico para cada plataforma.

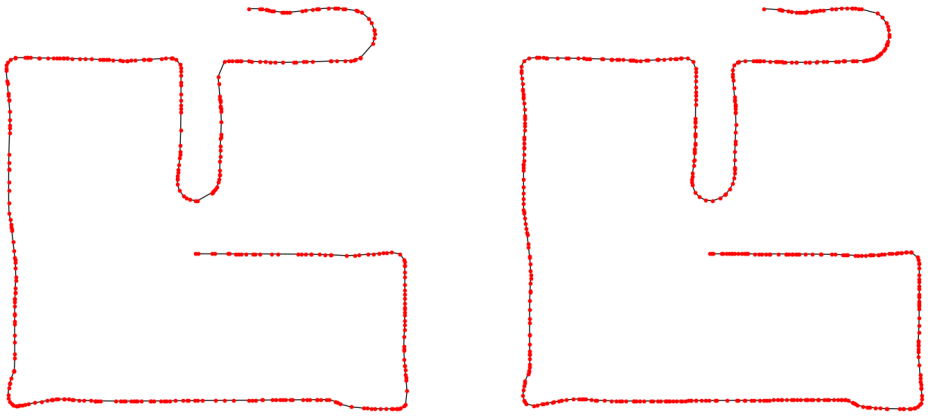
Notebook	$7,152 * 10^{-5} \pm 1,952 * 10^{-5}$
Moto G	$8,974 * 10^{-4} \pm 1,226 * 10^{-4}$
Galaxy S7	$1,166 * 10^{-4} \pm 2,202 * 10^{-4}$

tempos de execuções também são pequenos mostrando que o sistema executa quase no tempo constante mostrado nas tabelas. O tempo de execução média no notebook é $0,17$ ms sem agrupamento, cerca de 10 vezes mais rápido que os smartphones. Os agrupamentos tiveram pouco impacto na execução no computador variando cerca de 0.03 ms em média.

A execução nos celulares foram em média cerca de $1,8$ ms sem agrupamento, aproximadamente 550 execuções por segundo, sendo que mesmo com a adição dos agrupamentos não houve mudança significativa nos tempos de execuções médios nos celulares. Mostrando que os agrupamentos podem ser realizados na etapa online sem impactar no tempo de de execução do sistema. Os tempos médios similares pode ter ocorrido devido ao uso extensivo do processador (*Cpu*) que nos testes usou em média de 50%. E também, o sistema utilizou em média de 21% da memória aumentando aproximadamente 4% com o uso dos agrupamentos.



(a) Limiar de distinguibilidade *WiFi*: 0.1 (b) Limiar de distinguibilidade *WiFi*: 0.3



(c) Limiar de distinguibilidade *WiFi*: 0.6 (d) Limiar de distinguibilidade *WiFi*: 0.9

Figura 38 – Mapas topológicos gerados com erro *RSS* do *WiFi* 4.635 ± 3.138.

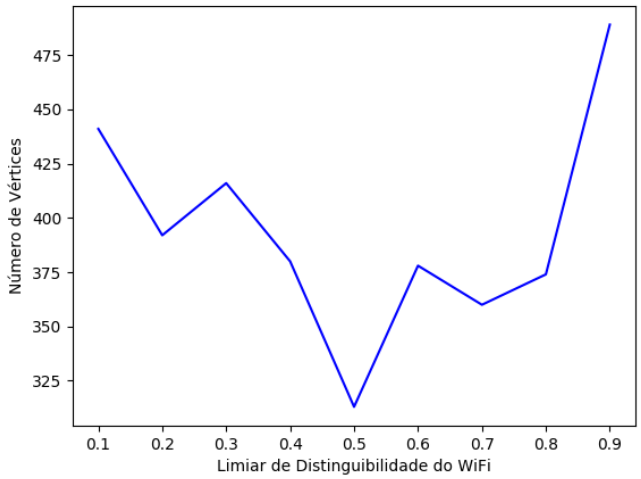
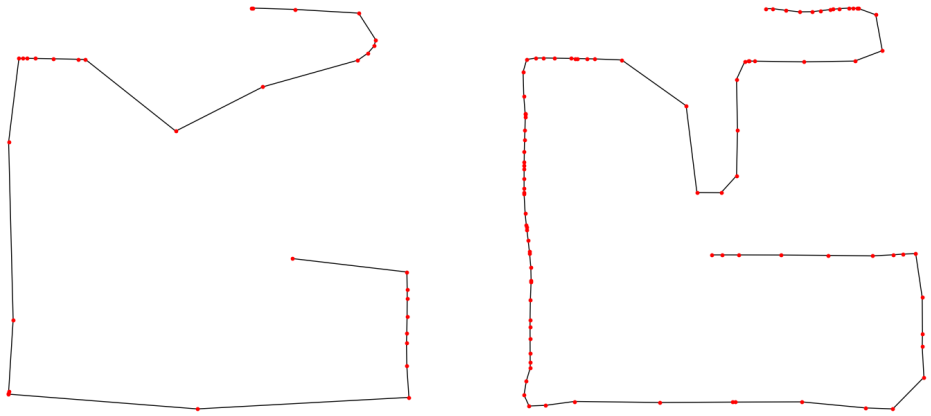
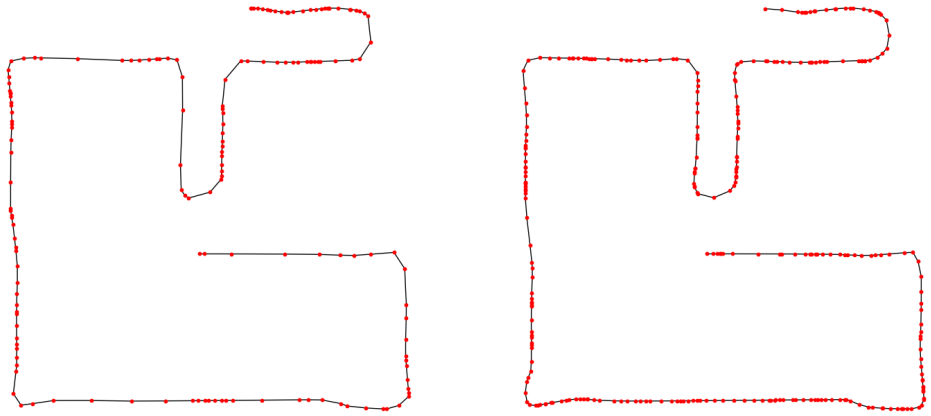


Figura 39 – Número de vértices em média criados com erro *RSS* do *WiFi* 4.635 ± 3.138.



(a) Similaridade: 0.168 (b) Similaridade: 0.17



(c) Similaridade: 0.175 (d) Similaridade: 0.18

Figura 40 – Agrupamentos gerados apenas com o critério de similaridade.

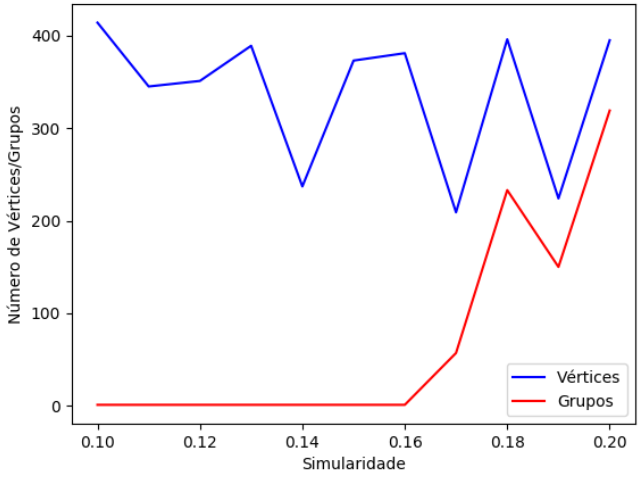
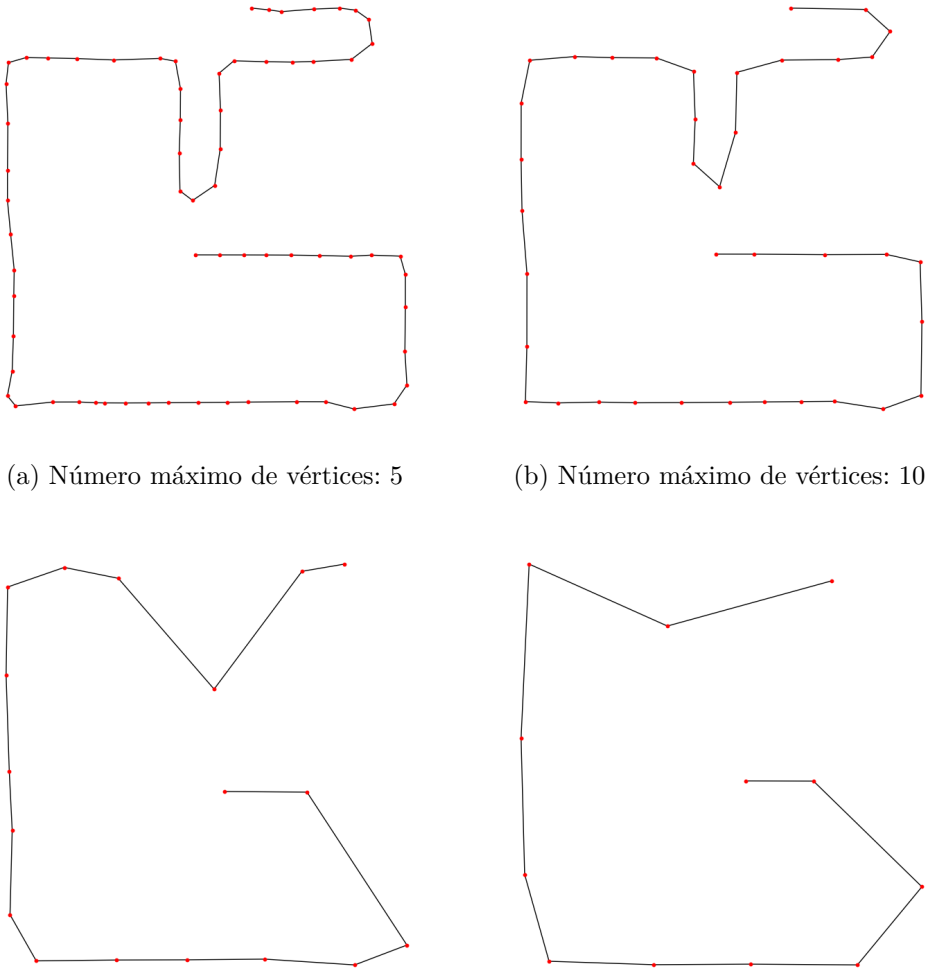


Figura 41 – Número de vértices e grupos em média criados com o critério de similaridade.



(a) Número máximo de vértices: 5

(b) Número máximo de vértices: 10

(c) Número máximo de vértices: 15

(d) Número máximo de vértices: 20

Figura 42 – Agrupamentos gerados apenas com o critério de número máximo de vértices por grupo.

Tabela 4 – Tempo(s) de execução médio do sistema de mapeamento topológico completo para cada plataforma.

	Normal	Agrupamento	Agrupamento por Sala
Notebook	$1,765 * 10^{-4} \pm 2.513 * 10^{-5}$	$2,021 * 10^{-4} \pm 3.832 * 10^{-5}$	$1,796 * 10^{-4} \pm 4.967 * 10^{-5}$
Moto G	$1,873 * 10^{-3} \pm 3,440 * 10^{-4}$	$1,962 * 10^{-3} \pm 3,519 * 10^{-4}$	$2,216 * 10^{-3} \pm 4,746 * 10^{-4}$
Galaxy S7	$1,739 * 10^{-3} \pm 3,709 * 10^{-4}$	$1,803 * 10^{-3} \pm 2,678 * 10^{-4}$	$2,580 * 10^{-3} \pm 3,515 * 10^{-4}$

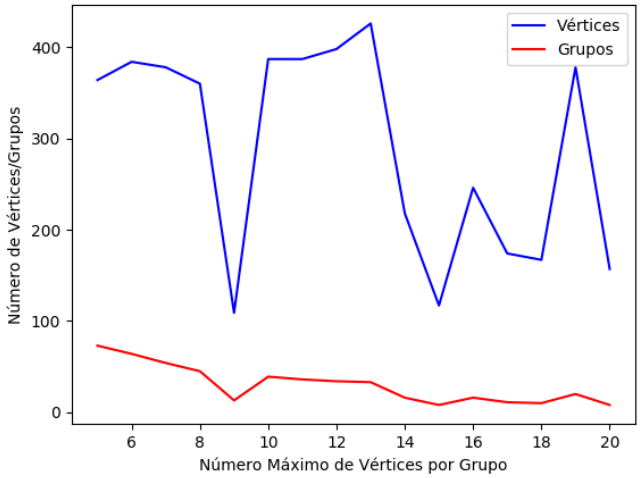


Figura 43 – Número de vértices e grupos em média criados com o critério de número máximo de vértices por grupo.

Tabela 5 – Uso de Cpu e Memória médio pelos smartphones na execução do mapeamento topológico sem agrupamento.

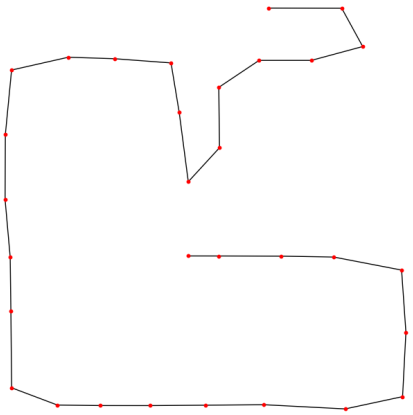
	Cpu (%)	Memória (%)
Moto G	$49 \pm 3,3$	$21,57 \pm 2,26$
Galaxy S7	$52 \pm 7,1$	$21.41 \pm 2,31$

Tabela 6 – Uso(%) de Cpu e Memória médio pelos smartphones na execução do mapeamento topológico com agrupamento

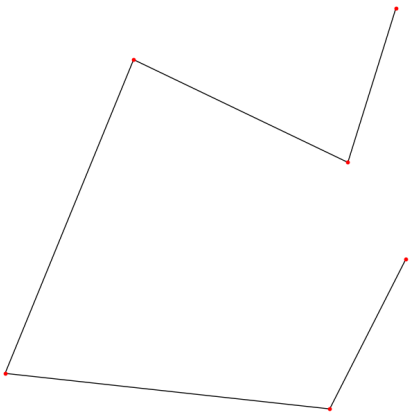
	Cpu	Memória
Moto G	$47 \pm 4,5$	$25,1 \pm 0,6$
Galaxy S7	53 ± 8	$23,13 \pm 1,64$

Tabela 7 – Uso(%) de Cpu e Memória médio pelos smartphones na execução do mapeamento topológico com agrupamento por salas

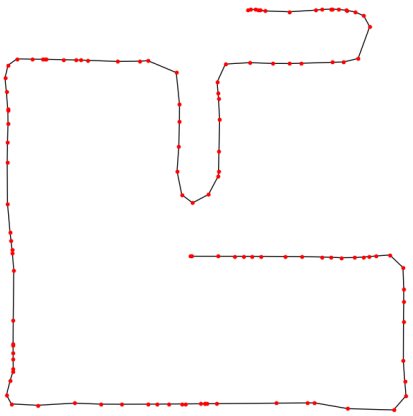
	Cpu	Memória
Moto G	$50 \pm 2,4$	25.24 ± 0.56
Galaxy S7	51 ± 7	27.40 ± 0.29



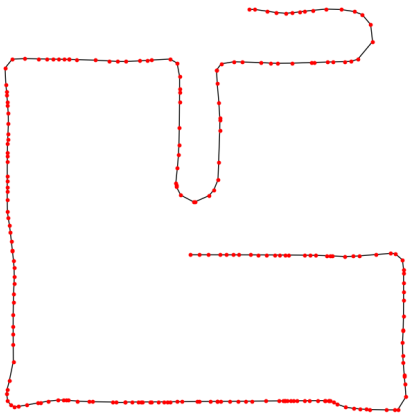
(a) Distância máxima entre os vértices: 0.5m



(b) Distância máxima entre os vértices: 1.0m



(c) Distância máxima entre os vértices: 1.5m



(d) Distância máxima entre os vértices: 2.0m

Figura 44 – Agrupamentos gerados apenas com o critério de distância máxima entre os vértices.

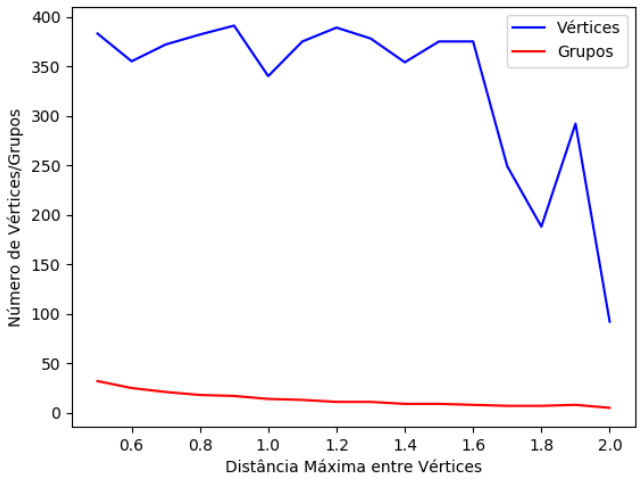
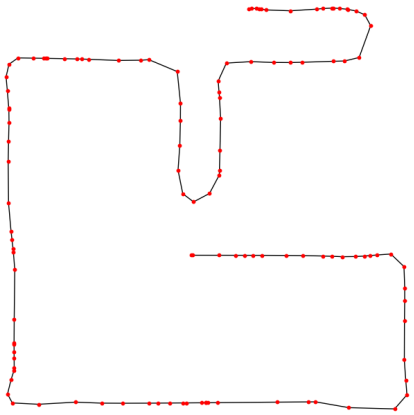
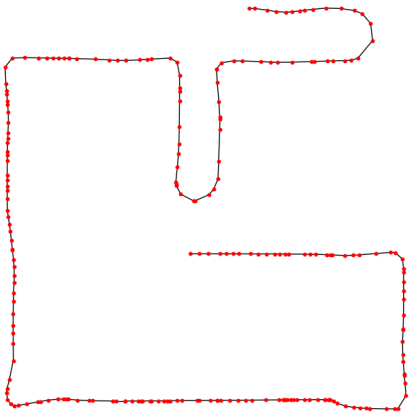


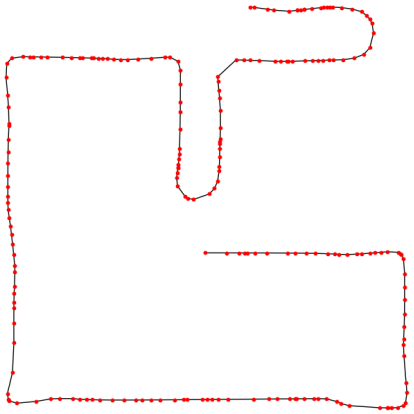
Figura 45 – Número de vértices e grupos em média criados com o critério de distância máxima entre os vértices.



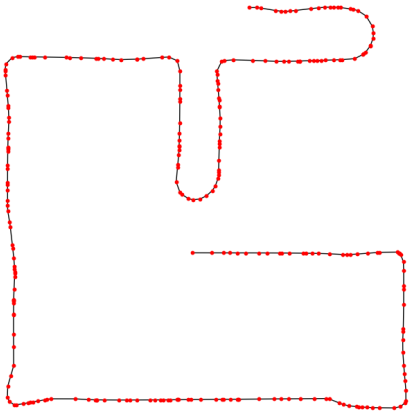
(a) Num.Verts.: 5, Dist.:1m



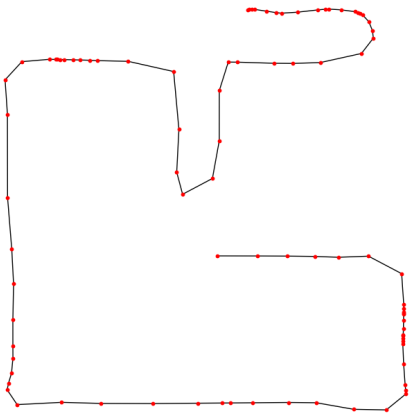
(b) Num.Verts.: 5, Dist.:0.1m



(c) Num.Verts.: 2, Dist.:1m



(d) Num.Verts.: 2, Dist.:0.1m



(e) Num.Verts.: 10, Dist.:2m

Figura 46 – Agrupamentos gerados os três critérios, sendo o de similaridade fixo em 0.17.

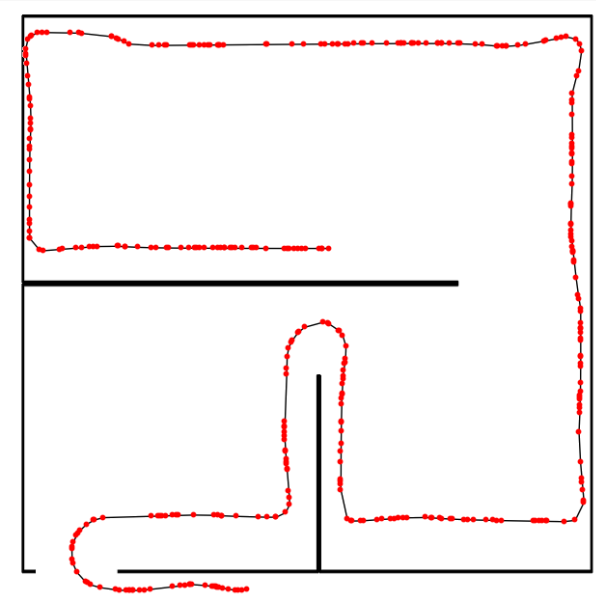


Figura 47 – Mapa topológico do Cenário 1.

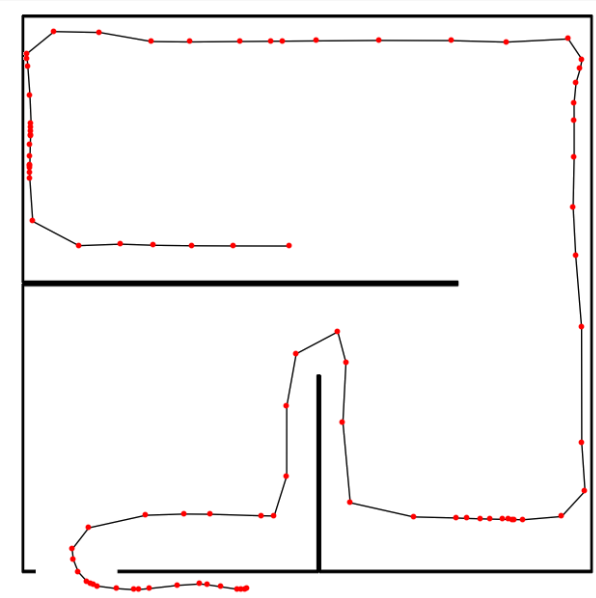


Figura 48 – Agrupamento do Cenário 1.

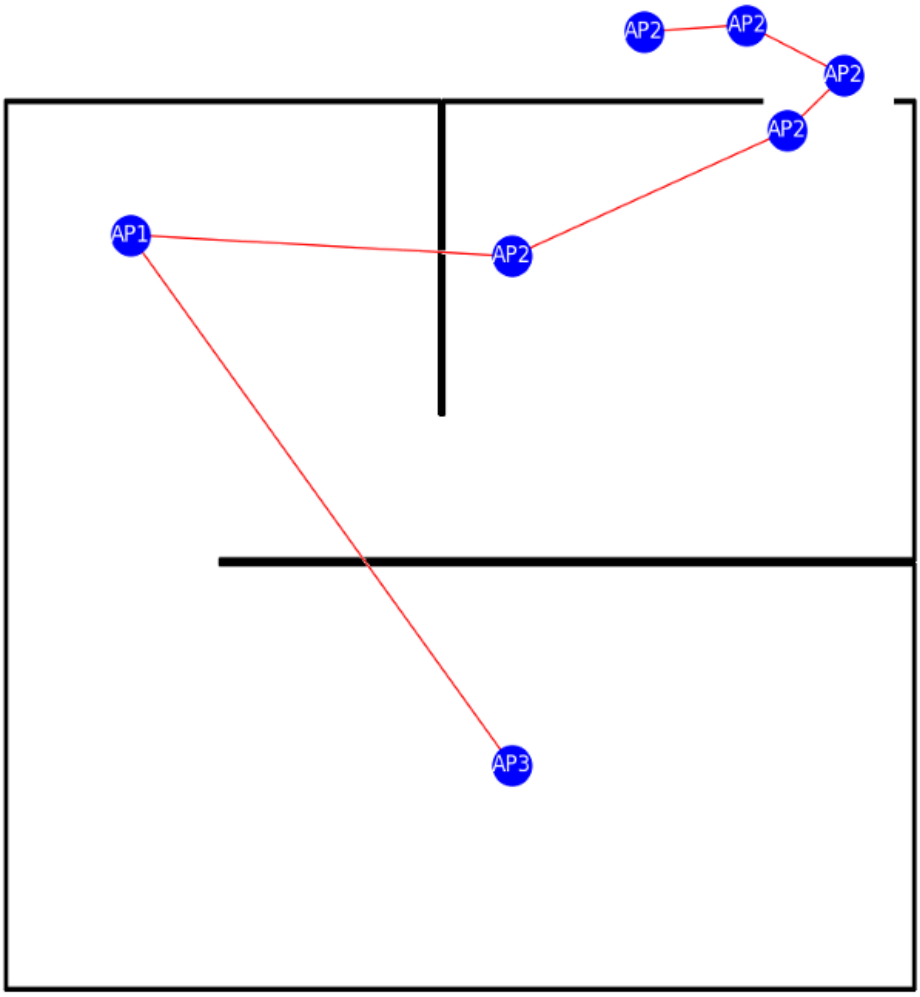


Figura 49 – Agrupamento por salas do Cenário 1.

5.2 Cenário 2

Nesta seção o cenário de média complexidade, o robô navega coletando dados de *pose* e distância dos *APs*. Na Figura 50 mostra o ambiente criado, o robô navega pela área demarcada em azul e entra em quatro salas no ambiente, algumas portas foram fechadas para diminuir a complexidade do ambiente. O ambiente possui 3 *APs* em salas separadas.

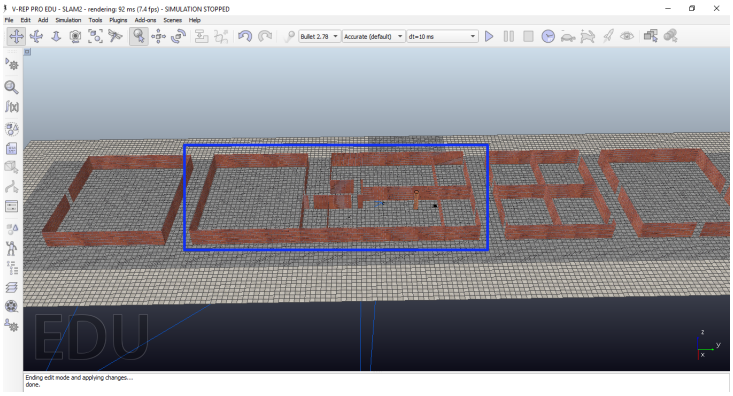


Figura 50 – Ambiente simulado no *V-Rep* do Cenário 2.

O segundo ambiente simulado é o mapa do Departamento de Computação da Universidade Federal do Piauí (UFPI) entre a Chefia do Curso e a Comissão Permanente de Seleção (COPESE). Foram tiradas a medidas reais desta área e a planta baixa do departamento é mostrado na Figura 51.

Figura 51 – Parte do mapa do Departamento de Computação

5.2.1 Resultados

Com os experimentos realizados no Estudo de Caso deste cenário (Apêndice A) foram selecionados limares para a execução do mapeamento topológico e os parâmetros para o agrupamento dos vértices do grafo gerado. Na Figura 52 é apresentado o mapa topológico gerado com os limiares selecionados. Destacado em azul um exemplo de fechamento de laço feito corretamente. Na Figura 53 o agrupamento com os parâmetros selecionados. Em

sequência, na Figura 54 é apresentado os resultados do agrupamento por salas que é criado a partir do agrupamento anterior, onde cada vértice já está rotulado semanticamente com o nome do AP mais próximo.

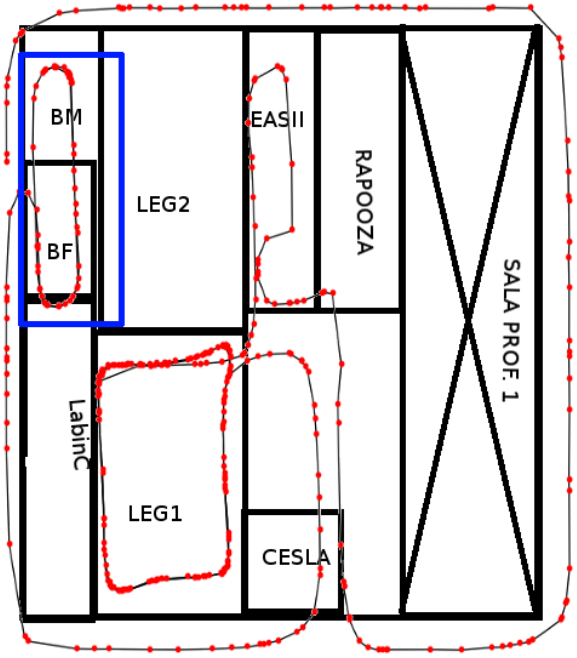


Figura 52 – Mapa topológico do Cenário 2.

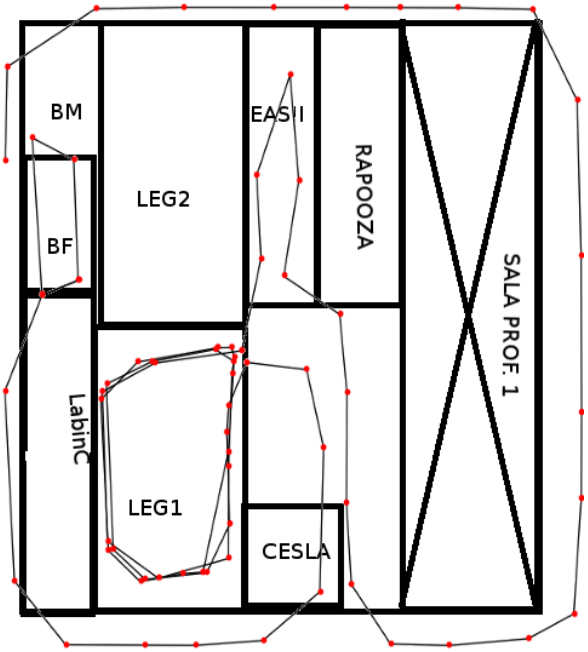


Figura 53 – Agrupamento do Cenário 2.

O sistema de mapeamento topológico é executado utilizando os limiares e parâmetros selecionados e depois é medido os tempos de execuções médios em segundos nas plataformas das Tabelas 1 e 2. Assim como no cenário anterior, são feitas as mesmas 5 análises do

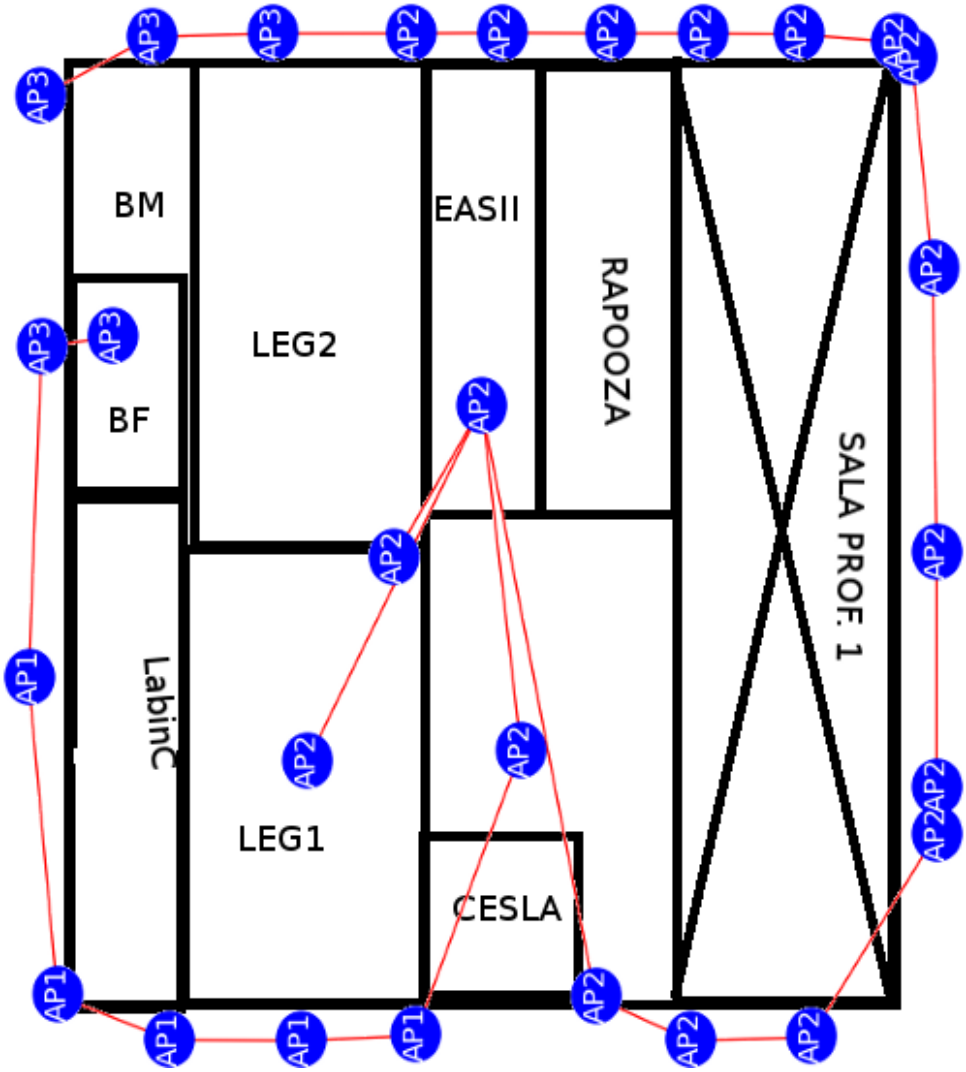


Figura 54 – Agrupamento por salas do Cenário 2.

sistema. Na primeira análise a etapa offline do sistema (Tabela 8) é avaliada separadamente. Na segunda análise o sistema completo é avaliado (Tabela 9).

As três ultimas análises são feitas nos smartphones. É analisado o uso percentual do processador (*Cpu*) e da memória durante a execução do mapeamento topológico sem agrupamento (Tabela 10), com agrupamento (Tabela 11) e com agrupamento por salas (Tabela 12).

Em relação ao Cenário 1 houve um aumento do tempo de execução médio na etapa offline em até 10 vezes em média. A causa disto é também o aumento quantidade de informações capturadas do ambiente e armazenada nos bancos de dados. Quando uma nova informação chega no processamento ela verifica se já existe no seu respectivo banco de dados. Quanto maior o banco mais essa busca vai demorar, aumentando o tempo de execução médio. Outra grande diferença do cenário anterior, foi da quantidade de memória utilizada pelo mapeamento topológico que passou de 25% para 75%, até mesmo 95%. O aumento está relacionado a quantidade vértices armazenados no mapa topológico.

Tabela 8 – Tempo(s) de execução médio da etapa offline do sistema de mapeamento topológico para cada plataforma.

Notebook	$5.443 * 10^{-5} \pm 4.980 * 10^{-6}$
Moto G	$1,139 * 10^{-3} \pm 8.462 * 10^{-5}$
Galaxy S7	$1,252 * 10^{-3} \pm 1,045 * 10^{-4}$

Tabela 9 – Tempo(s) de execução médio do sistema de mapeamento topológico completo para cada plataforma.

	Normal	Agrupamento	Agrupamento por Sala
Notebook	$1,365 * 10^{-4} \pm 1.479 * 10^{-5}$	$1,479 * 10^{-4} \pm 1.021 * 10^{-5}$	$1,439065 * 10^{-4} \pm 1.495 * 10^{-5}$
Moto G	$2,579 * 10^{-3} \pm 2,419 * 10^{-4}$	$2,585 * 10^{-3} \pm 1,999 * 10^{-4}$	$2,746 * 10^{-3} \pm 2,537 * 10^{-4}$
Galaxy S7	$2,632 * 10^{-3} \pm 1,854 * 10^{-4}$	$2,724 * 10^{-3} \pm 2,402 * 10^{-4}$	$2,748 * 10^{-3} \pm 2,080 * 10^{-4}$

Tabela 10 – Uso de Cpu e Memória médio pelos smartphones na execução do mapeamento topológico sem agrupamento.

	Cpu (%)	Memória (%)
Moto G	$51 \pm 3,3$	$74,68 \pm 4,98$
Galaxy S7	$54 \pm 6,9$	$95,911 \pm 1,41$

Tabela 11 – Uso(%) de Cpu e Memória médio pelos smartphones na execução do mapeamento topológico com agrupamento

	Cpu	Memória
Moto G	$51 \pm 4,0$	$71,84 \pm 1,09$
Galaxy S7	$52 \pm 6,35$	$95,75 \pm 2,49$

Tabela 12 – Uso(%) de Cpu e Memória médio pelos smartphones na execução do mapeamento topológico com agrupamento por salas

	Cpu	Memória
Moto G	$51 \pm 3,6$	74.06 ± 1.25
Galaxy S7	52 ± 6	75.84 ± 1.70

5.3 Cenário 3

Nesta seção o cenário com maior complexidade testado. Na Figura 55 mostra o ambiente criado, o robô navega pelo ambiente e entra em todas as suas salas, portanto o ambiente com maior quantidade de leituras capturadas e o maior número de salas exploradas. O ambiente possui 3 *APs* em locais. Este ambiente também é fictício e foi criado para simular um corredor com diversas salas, o que ocorre em um hospital, por exemplo. A planta baixa do ambiente é apresentado na Figura 56.

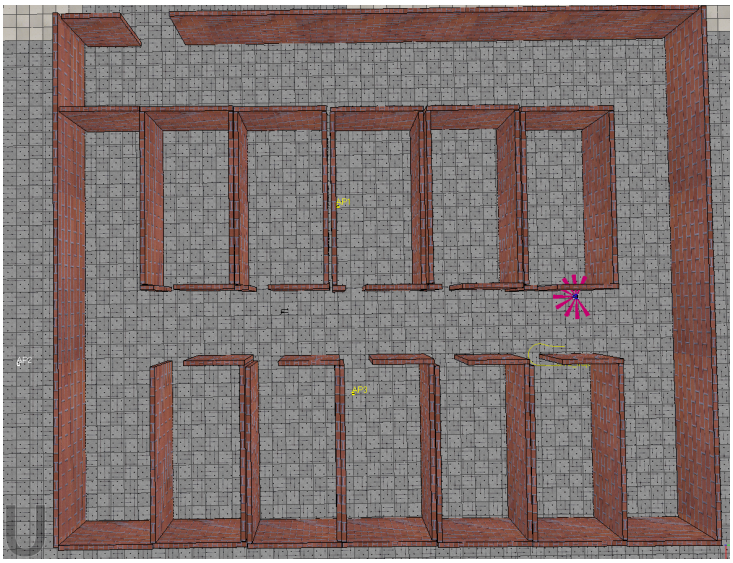


Figura 55 – Ambiente simulado no *V-Rep* do Cenário 3.

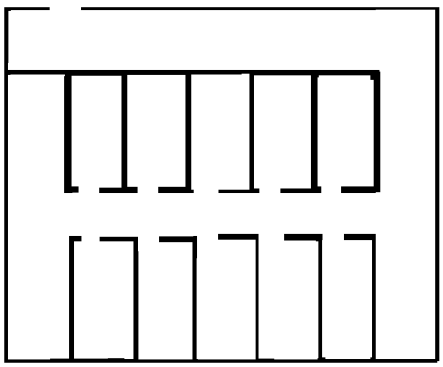


Figura 56 – Planta baixa do ambiente simulado.

5.3.1 Resultados

Com os experimentos realizados no Estudo de Caso deste cenário (Apêndice B) foram selecionados limares para a execução do mapeamento topológico e os parâmetros para o agrupamento dos vértices do grafo gerado. Na Figura 57 é apresentado o mapa topológico gerado com os limiares selecionados. Na Figura 58 o agrupamento com os parâmetros selecionados. Em sequência, na Figura 59 é apresentado os resultados do

agrupamento por salas que é criado a partir do agrupamento anterior, onde cada vértice já está rotulado semanticamente com o nome do AP mais próximo. Neste cenário mais complexo que o robô não identifica corretamente as portas e erra no agrupamento de salas.

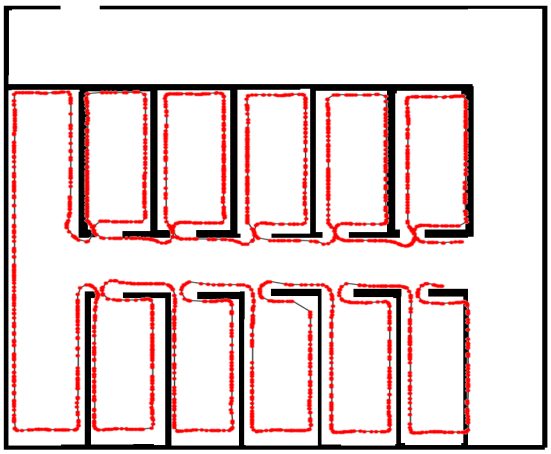


Figura 57 – Mapa topológico do Cenário 2.

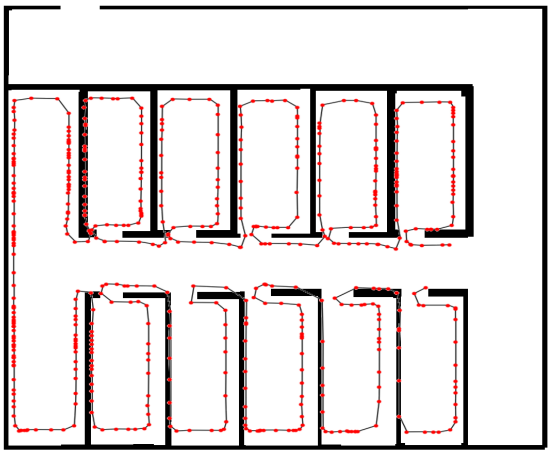


Figura 58 – Agrupamento do Cenário 2.

O sistema de mapeamento topológico é executado utilizando os limiares e parâmetros selecionados e depois é medido os tempos de execuções médios em segundos nas plataformas das Tabelas 1 e 2. Assim como no cenário anterior, são feitas as mesmas 5 análises do sistema. Na primeira análise a etapa offline do sistema (Tabela 13) é avaliada separadamente. Na segunda análise o sistema completo é avaliado (Tabela 14).

As três ultimas análises são feitas nos smartphones. É analisado o uso percentual do processador (Cpu) e da memória durante a execução do mapeamento topológico sem agrupamento (Tabela 15), com agrupamento (Tabela 16) e com agrupamento por salas (Tabela 17).

Em relação ao Cenário 2 há uma acentuamento no tempo de execução médio na etapa offline e na online em até 10 vezes em média. O mesmo motivo do cenário

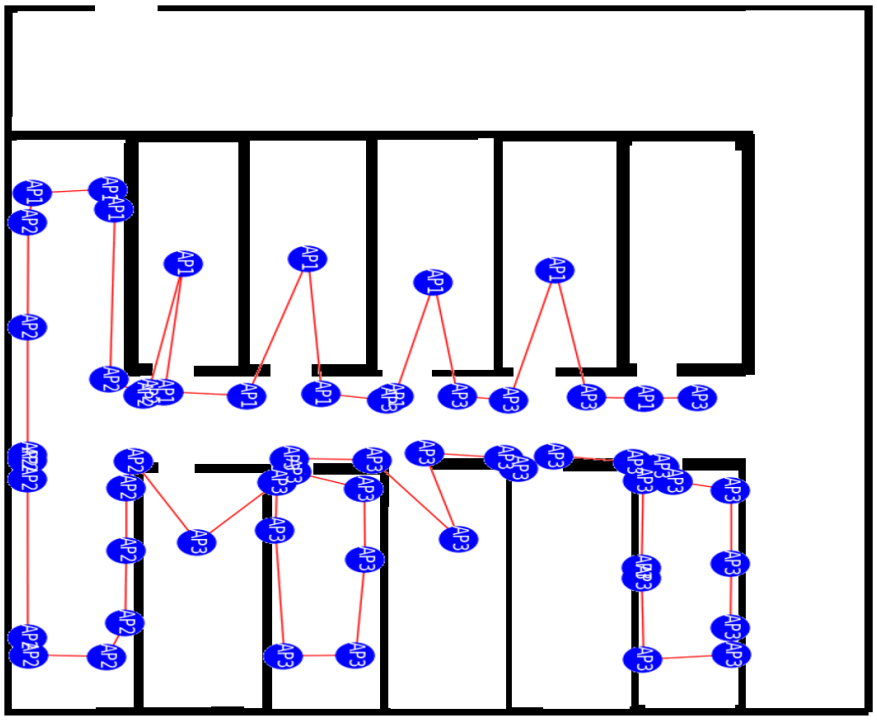


Figura 59 – Agrupamento por salas do Cenário 2.

Tabela 13 – Tempo(s) de execução médio da etapa offline do sistema de mapeamento topológico para cada plataforma.

Notebook	$1,487 * 10^{-3} \pm 3,920 * 10^{-4}$
Moto G	$7,850 * 10^{-3} \pm 6,595 * 10^{-4}$
Galaxy S7	$7,863 * 10^{-3} \pm 4,397 * 10^{-2}$

anterior é a causa para o aumento da quantidade de informações capturadas do ambiente e armazenada nos bancos de dados. Os tempos de execuções da etapa offline das plataformas se aproximaram neste cenário. E nos smartphones as médias dos tempos de execuções do sistema já são menores que seu desvio padrão.

Tabela 14 – Tempo(s) de execução médio do sistema de mapeamento topológico completo para cada plataforma.

	Normal	Agrupamento	Agrupamento por Sala
Notebook	$3,131 * 10^{-3} \pm 9,909 * 10^{-4}$	$3,501 * 10^{-3} \pm 1,114 * 10^{-3}$	$3,381914 * 10^{-3} \pm 6,488 * 10^{-4}$
Moto G	$1,065 * 10^{-3} \pm 5,813 * 10^{-2}$	$4,304 * 10^{-3} \pm 4,500 * 10^{-2}$	$1,471 * 10^{-3} \pm 4,48676205353 * 10^{-2}$
Galaxy S7	$3,325 * 10^{-3} \pm 4,379 * 10^{-2}$	$2,713 * 10^{-3} \pm 4,390 * 10^{-2}$	$2,572 * 10^{-3} \pm 4,47848981086 * 10^{-2}$

Tabela 15 – Uso de Cpu e Memória médio pelos smartphones na execução do mapeamento topológico sem agrupamento.

	Cpu (%)	Memória (%)
Moto G	$51 \pm 1,2$	$73,45 \pm 11,78$
Galaxy S7	$54 \pm 2,4$	$76,67 \pm 14,62$

Tabela 16 – Uso(%) de Cpu e Memória médio pelos smartphones na execução do mapeamento topológico com agrupamento

	Cpu	Memória
Moto G	$52 \pm 0,8$	$81,46 \pm 4,59$
Galaxy S7	$55 \pm 1,35$	$85,27 \pm 6,33$

Tabela 17 – Uso(%) de Cpu e Memória médio pelos smartphones na execução do mapeamento topológico com agrupamento por salas

	Cpu	Memória
Moto G	$52 \pm 0,6$	78.61 ± 5.42
Galaxy S7	$55 \pm 1,15$	88.34 ± 5.00

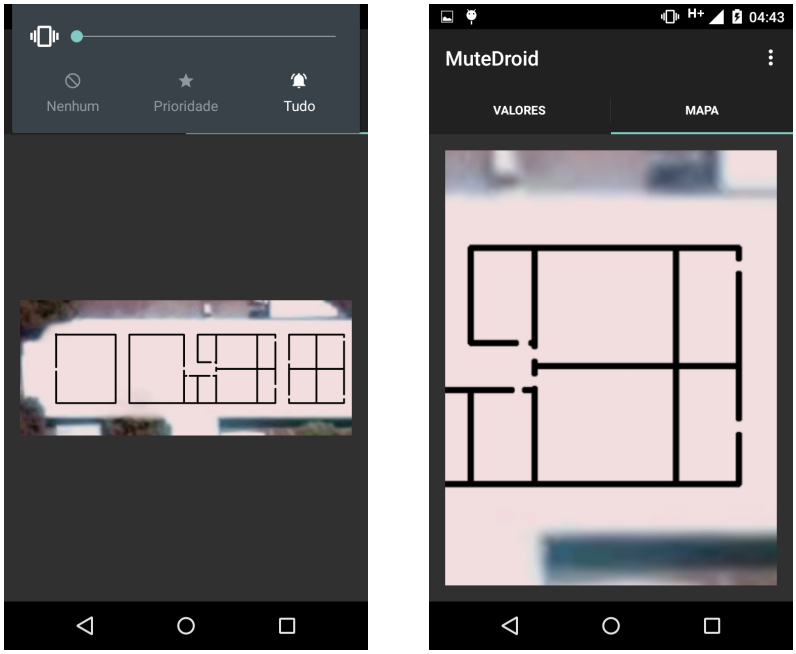
5.4 Coleta do *Wifi fingerprint*

Foi realizado um experimento para estimar o custo da criação dos *WiFi fingerprints*. Os resultados adquiridos são as extrações das informações das leituras RSS *WiFi* que são usadas no mapeamento topológico. É avaliado o tempo necessário para gerar um *fingerprint* do mapa e o erro com as leituras.

Como já explicado na Subseção 3.0.2 foi desenvolvido um aplicativo *Android* chamado *MuteDroid* para captura dos *WiFi fingerprints* e futuramente para um sistema de localização baseado em Biswas e Veloso (2010). O local das medições é o Departamento de Computação da Universidade Federal do Piauí (UFPI), na Figura 60 uma foto tirada por satélite com a planta baixa do apartamento em destaque.

Para o teste de avaliação do tempo necessário para coletar um *fingerprint*, com o auxilio da aplicação (Figura 60), foram feitos 10 medições em locais diferente do mapa e em cada medição foram feitas 100 leituras do *WiFi*. O tempo médio para criar um *fingerprint* foi de 10101.41 milissegundos com um desvio padrão de 910.50 milissegundos. O tempo médio e o desvio relativamente alto para uma aplicação de tempo real deve-se ao Sistema Operacional *Android* que faz as leituras *WiFi* em intervalo de tempo determinado, consequentemente não é possível obter-se as informações das redes sem fio sobre demanda.

Um fator importante que influenciará no controle de trajetória do robô e no algoritmo de mapeamento topológico apresentado na seção 4.2. De fato, o robô só terá essas informações quando novas leituras forem feitas pelo sistema. Consequentemente, ao projetar o algoritmo permitiu-se a necessidade dele trabalhar sem essa informação. Com



(a) (b)

Figura 60 – Telas do aplicativo *MuteDroid*, (a) Tela com mapa para criar os *Wifi fingerprints* e (c) tela com um *zoom* no mapa.

isso, o algoritmo teve um ganho na flexibilidade de execução.

6 Conclusões e Trabalhos Futuros

O processo de mapeamento é um dos problemas fundamentais na navegação autônoma de robôs e mais especificamente, resolver o problema do *SLAM*. Com essa base, este trabalho propõe uma solução de mapeamento topológico para ambientes *indoors* estáticos para um *CellBot*. O mapeamento permite que o robô mapeie o ambiente fundindo informações de odometria e redes sem fio.

Um algoritmo para resolver o problema é desenvolvido e, a partir dele, é desenvolvido um sistema de mapeamento topológico. As entradas dos sistema são as *poses* do robô calculada através para Odometria Visual (NISTER; NARODITSKY; BERGEN, 2004) e as leituras *RSS* do *WiFi*. Estas são utilizadas para montar *Wifi fingerprints*, pegadas, que podem ser distinguidas (SHIN; CHA, 2010)). Com essas informações o algoritmo cria vértices apenas em locais não visitados e a arestas interligando esses vértices criando, assim, o mapa topológico.

A representação do mapa topológico descreve o ambiente em uma estrutura compacta para o armazenamento na memória do sistema computacional do robô além de permitir a execução de tarefas de alto nível. O algoritmo resolve o grande problema do mapeamento topológico que consiste na ausência de um padrão de definição de quais estruturas serão associadas ao vértice e quais relações serão descritas para serem utilizadas como elos (KUIPERS; BYUN, 1993).

O sistema mapeamento topológico desenvolvido é dividido em duas etapas: offline e online. As coletas de informações do ambiente são realizadas na etapa offline. Essas informações brutas são encapsuladas e salvas em bancos de dados próprios para cada informação encapsulada, essa fase é chamada de treinamento. Na etapa online as informações são lidas dos bancos de dados servindo de entrada para o algoritmo de mapeamento. Nesta etapa, onde são criadas os vértices, as arestas e os grupos.

Para tornar o mapa topológico mais 'amigável' ao usuário, foi implementado neste trabalho um sistema de agrupamento. Tornar-lo mais 'amigável' significa dar a um vértice do mapa uma informação semântica que o identifica no ambiente, como número da sala, nome do estabelecimento, instituição, etc. Dois tipos de agrupamento foram implementados. Um agrupa os vértices seguindo um critério de similaridade entre os eles, assim agrupa vértices muitos similares diminuindo a densidade do grafo gerado pelo mapeamento topológico sem perdas para o mapa.

O outro agrupamento agrupa os grupos gerados anteriormente para tentar criar grupos isolados dentro de salas, este processo é chamado de agrupamento por salas. O agrupamento diminui ainda mais a densidade de vértices do grafo, mas com risco de

perda de informação do mapa. Para o agrupamento por salas funcionar é preciso que o robô identifique as portas do ambiente ou já tenha conhecimento delas previamente, essas posições podem ser adquiridas na etapa offline do sistema. Informação semântica dada ao vértice gerado pelo agrupamento por salas é o nome do AP mais próximo em força de sinal do vértice. Essa informação foi escolhida, porque geralmente os APs são nomeados para identificar uma localidade como escola, laboratório, loja, etc.

Os resultados deste trabalho são obtidos através de testes em três cenários distintos com um nível de complexidade crescente. A complexidade do cenário é definida pela quantidade de salas do ambiente por onde robô navegou. Em cada cenário é feito um Estudo de Caso para avaliar a criação dos mapas topológicos e a quantidade de vértices e grupos criados em cada um. Depois do Estudo de Caso resultados com os tempos de execução do sistema de mapeamento topológico executando em diversas plataformas, especificamente um computador notebook e dois smartphones diferentes. Alem disso, nos smartphones são avaliados também o uso percentual do processador e da memória pelo sistema.

No primeiro cenário, ambiente menos complexo, o sistema executou no smartphone em um tempo médio de $0,17ms$ sem agrupamento, $0,18ms$ com agrupamento e $0,25ms$ com o agrupamento por salas. Um resultado excelente permitindo 4000 execuções em média por segundo do sistema em cenários menos complexos.

No primeiro cenário, ambiente menos complexo, o sistema executou no smartphone em um tempo médio de $2,6ms$ sem agrupamento, $2,7ms$ com agrupamento e $2,7ms$ com o agrupamento por salas. Ainda um resultado bom permitindo 370 execuções em média por segundo do sistema em cenários menos com complexidade mediana. Para cenários mais complexos esses resultados são acentuados.

Por último, foi apresentado um experimento para avaliar o tempo médio necessário para criar um *fingerprint*. No experimento, verificou-se que a leitura *WiFi* é restringida a um intervalo de tempo determinado pelo próprio sistema *Android*. Com esta observação percebeu-se a necessidade do algoritmo criar o mapa mesmo que alguma informação não esteja disponível no momento de sua execução.

6.1 Trabalhos Futuros

Alguns testes e implementações do sistema de mapeamento topológico são necessárias para aprimorar-lo. Assim, pretende-se em trabalhos futuros:

- Implementar o sistema de mapeamento topológico em *CellBot* real.
- Testar o sistema em ambientes reais.

- Estender a abrangência do algoritmo de fechamento de laço de grafos para os agrupamentos.
- E criar uma aplicativo para mostrar o mapa topológico, onde será possível renomear os vértices das salas e direcionar o robô para as salas.

Referências

ALSINA, P. J. et al. Navegação e controle de robôs móveis. *Micurso : Congresso Brasileiro de Automática*, 2002. Citado 2 vezes nas páginas 13 e 2.

AMIGONI, F.; GASPARINI, S.; GINI, M. Map building without odometry information. In: *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*. [S.l.: s.n.], 2004. v. 4, p. 3753–3758 Vol.4. ISSN 1050-4729. Citado na página 9.

ANDROID. *Página principal do android*. Acessado em Julho 2017. <http://www.android.com/>. Citado na página 15.

ARAúJO, A. V. P. R. de. *Uma proposta de metodologia para o ensino de Física usando robótica de baixíssimo custo*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, Natal, mar. 2013. Citado na página 7.

AROCA, R. V. et al. Increasing students' interest with low-cost cellbots. *IEEE Transactions on Education*, v. 56, n. 1, p. 3–8, Fev 2013. ISSN 0018-9359. Citado na página 7.

AROCA, R. V.; MARCOS, L.; GONçALVES, L. M. G. Towards smarter robots with smartphones. *5th workshop in applied robotics and automation*, Robocontrol, 2012. Citado 3 vezes nas páginas 13, 7 e 8.

BARTSCH, S. et al. Spaceclimber: Development of a six-legged climbing robot for space exploration. In: *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*. IEEE, 2010. p. 1–8. Disponível em: <http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5756731>. Citado na página 1.

BAYRAMOGLU, E. et al. Mobile robot navigation in a corridor using visual odometry. In: _____. *Prooceedings of the 14th International Conference on Advanced Robotics*. [S.l.]: IEEE, 2009. p. 58. ISBN 978-1-4244-4855-5. Copyright: 2009 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE. Citado 7 vezes nas páginas 13, 5, 12, 24, 39, 79 e 93.

BENITTI, F. B. V. Exploring the educational potential of robotics in schools: A systematic review. *Computers and Education*, v. 58, n. 3, p. 978 – 988, 2012. ISSN 0360-1315. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0360131511002508>. Citado na página 1.

BISWAS, J.; VELOSO, M. Wifi localization and navigation for autonomous indoor mobile robots. In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. [S.l.: s.n.], 2010. p. 4379–4384. Citado na página 64.

BODENSTEIN, C. et al. A smartphone-controlled autonomous robot. In: *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*. [S.l.: s.n.], 2015. p. 2314–2321. Citado 3 vezes nas páginas 13, 8 e 12.

BORENSTEIN, J.; FENG, L. Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation*, v. 12, n. 6, p. 869–880, Dec 1996. ISSN 1042-296X. Nenhuma citação no texto.

BUHL-BROWN; D., D. Developing a robotics education platform using android based cellbots (abstract only). In: *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. New York, NY, USA: ACM, 2015. (SIGCSE '15), p. 714–714. ISBN 978-1-4503-2966-8. Disponível em: <http://doi.acm.org/10.1145/2676723.2693625>. Citado na página 7.

CHOI, C.-H. et al. Topological map building based on thinning and its application to localization. In: *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*. [S.l.: s.n.], 2002. v. 1, p. 552–557 vol.1. Citado na página 10.

CHONG, K. S.; KLEEMAN, L. Accurate odometry and error modelling for a mobile robot. In: *ICRA*. [S.l.: s.n.], 1997. Nenhuma citação no texto.

CHOSET, H.; BURDICK, J. *Sensor Based Exploration: The Hierarchical Generalized Voronoi Graph*. 2000. Citado na página 10.

CORCIONE, F. et al. Advantages and limits of robot-assisted laparoscopic surgery: preliminary experience. *Surgical Endoscopy And Other Interventional Techniques*, v. 19, n. 1, p. 117–119, 2005. ISSN 1432-2218. Disponível em: <http://dx.doi.org/10.1007/s00464-004-9004-9>. Citado na página 1.

CROWLEY, J. Navigation for an intelligent mobile robot. *IEEE Journal on Robotics and Automation*, v. 1, n. 1, p. 31–41, Mar 1985. ISSN 0882-4967. Citado na página 9.

DAVIDS, A. Urban search and rescue robots: from tragedy to technology. *IEEE Intelligent Systems*, v. 17, n. 2, p. 81–83, March 2002. ISSN 1541-1672. Citado na página 1.

FABRIZI, E.; SAFFIOTTI, A. Extracting topology-based maps from gridmaps. In: *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*. [S.l.: s.n.], 2000. v. 3, p. 2972–2978 vol.3. ISSN 1050-4729. Citado 2 vezes nas páginas 10 e 11.

FOX, D. et al. Monte carlo localization: Efficient position estimation for mobile robots. In: *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1999. (AAAI '99/IAAI '99), p. 343–349. ISBN 0-262-51106-1. Disponível em: <http://dl.acm.org/citation.cfm?id=315149.315322>. Citado 3 vezes nas páginas 5, 13 e 17.

GALINDO, C. et al. Multi-hierarchical semantic maps for mobile robotics. In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. [S.l.: s.n.], 2005. p. 2278–2283. ISSN 2153-0858. Citado na página 10.

GOLDBERG, D. et al. A distributed layered architecture for mobile robot coordination: Application to space exploration. In: *In Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*. [S.l.: s.n.], 2002. Citado na página 1.

HAHNEL, D.; SCHULZ, D.; BURGARD, W. Temporary maps for robust localization in semi-static environments. In: *Advanced Robotics*. [S.l.: s.n.], 2012. p. 579–597. Citado na página 9.

HATA, A. Y.; SHINZATO, P.; WOLF, D. F. Mapeamento e classificação de terrenos utilizando aprendizado supervisionado. Congresso Brasileiro de Automática - CBA, p. 122–129, 2010. Citado na página 9.

HORSWILL, I. Polly: A vision-based artificial agent. *AAAI*, 1993. Citado na página 2.

IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC)and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput. *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)*, p. 1–565, Out 2009. Citado na página 11.

KANG, S.-W. et al. Robot-assisted endoscopic surgery for thyroid cancer: experience with the first 100 patients. *Surgical Endoscopy*, v. 23, n. 11, p. 2399–2406, 2009. ISSN 1432-2218. Disponível em: <http://dx.doi.org/10.1007/s00464-009-0366-x>. Citado na página 1.

KITANO, H. et al. Robocup rescue: search and rescue in large-scale disasters as a domain for autonomous agents research. In: *Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on*. [S.l.: s.n.], 1999. v. 6, p. 739–743 vol.6. ISSN 1062-922X. Citado na página 1.

KORTENKAMP, D.; WEYMOUTH, T. Topological mapping for mobile robots using a combination of sonar and vision sensing. In: *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 2)*. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1994. (AAAI 94), p. 979–984. ISBN 0-262-61102-3. Disponível em: <http://dl.acm.org/citation.cfm?id=199480.199508>. Citado na página 11.

KUIPERS, B.; BYUN, Y.-T. Toward learning robots. In: VELDE, W. Van de (Ed.). Cambridge, MA, USA: MIT Press, 1993. cap. A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations, p. 47–63. ISBN 0-262-72017-5. Disponível em: <http://dl.acm.org/citation.cfm?id=164605.164614>. Citado na página 67.

KUIPERS, B. et al. Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In: *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*. [S.l.: s.n.], 2004. v. 5, p. 4845–4851 Vol.5. ISSN 1050-4729. Citado na página 10.

KUNZ, C. et al. Deep sea underwater robotic exploration in the ice-covered arctic ocean with auvs. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008. p. 3654–3660. Disponível em: <http://dx.doi.org/10.1109/IROS.2008.4651097>. Citado na página 1.

MEHRA, R.; SINGH, A. Real time rssi error reduction in distance estimation using rls algorithm. In: *2013 3rd IEEE International Advance Computing Conference (IACC)*. [S.l.: s.n.], 2013. p. 661–665. Citado 2 vezes nas páginas 15 e 26.

MEYER-DELIUS, D. et al. Mobile robot mapping in populated environments. In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. [S.l.: s.n.], 2010. p. 5750–5755. ISSN 2153-0858. Citado na página 8.

MORAVEC, H.; ELFES, A. High resolution maps from wide angle sonar. In: *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*. [S.l.: s.n.], 1985. v. 2, p. 116–121. Citado na página 9.

NISTER, D.; NARODITSKY, O.; BERGEN, J. Visual odometry. In: *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. [S.l.: s.n.], 2004. v. 1, p. I–652–I–659 Vol.1. ISSN 1063-6919. Citado 2 vezes nas páginas 5 e 67.

OCANA, M. et al. Indoor robot localization system using wifi signal measure and minimizing calibration effort. In: *Proceedings of the IEEE International Symposium on Industrial Electronics, 2005. ISIE 2005.* [S.l.: s.n.], 2005. v. 4, p. 1545–1550. ISSN 2163-5137. Citado na página 11.

PETROVSKAYA, A.; THRUN, S. Model based vehicle detection and tracking for autonomous urban driving. *Autonomous Robots*, v. 26, n. 2, p. 123–139, 2009. ISSN 1573-7527. Disponível em: <http://dx.doi.org/10.1007/s10514-009-9115-1>. Citado na página 1.

PFISTER, S. T.; ROUMELIOTIS, S. I.; BURDICK, J. W. Weighted line fitting algorithms for mobile robot map building and efficient data representation. In: *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*. [S.l.: s.n.], 2003. v. 1, p. 1304–1311 vol.1. ISSN 1050-4729. Citado na página 9.

RASPBERRY. *Página principal da placa raspberry pi 3*. Acessado em Julho 2017. <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. Citado na página 20.

ROMERO, A.; CAZORLA, M. Topological visual mapping in robotics. *Cognitive Processing*, v. 13, n. 1, p. 305–308, 2012. ISSN 1612-4790. Disponível em: <http://dx.doi.org/10.1007/s10339-012-0502-8>. Citado na página 11.

SANTANA, A. et al. Fusion of odometry and visual datas to localization a mobile robot using extended kalman filter. In: _____. *Sensor Fusion and Its Applications*. Sciyo, Austria: InTech, 2010. p. 407–422. Citado na página 12.

SANTANA, A. M. *Localização e mapeamento simultâneos de ambientes planos usando visão monocular e representação híbrida do ambiente*. Tese (Doutorado) — Universidade Federal do Rio Grande do Norte, Natal, 2011. Disponível em: <https://repositorio.ufrn.br/jspui/bitstream/123456789/15150/1/AndreMS_TESE.pdf>. Citado na página 2.

SCARAMUZZA, D.; FRAUNDORFER, F. Visual odometry: Part i - the first 30 years and fundamentals. *IEEE Robotics and Automation Magazine*, v. 8, 2011. Nenhuma citação no texto.

SCARAMUZZA, D.; FRAUNDORFER, F. Aplicação de controladores fuzzy e proporcional para um robô seguidor de parede autônomo em ambiente estático. *Revista de Sistemas e Computação, Salvador*, v. 6, n. 1, p. 55–63, 2016. Citado na página 23.

SHIN, H.; CHA, H. Wi-fi fingerprint-based topological map building for indoor user tracking. In: *2010 IEEE 16th International Conference on Embedded and Real-Time Computing Systems and Applications*. [S.l.: s.n.], 2010. p. 105–113. ISSN 2325-1271. Citado 5 vezes nas páginas 13, 12, 26, 34 e 67.

SOUZA, A. A. de S. et al. Mapeamento. In: _____. *Robótica Móvel. 1ed.* [S.l.]: LTC, 2014. v. 1, p. 161–175. Citado 3 vezes nas páginas 13, 9 e 10.

TARDIF, J. P.; PAVLIDIS, Y.; DANIILIDIS, K. Monocular visual odometry in urban environments using an omnidirectional camera. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. [S.l.: s.n.], 2008. p. 2531–2538. ISSN 2153-0858. Nenhuma citação no texto.

THRUN, S. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, v. 99, n. 1, p. 21 – 71, 1998. ISSN 0004-3702. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0004370297000787>. Citado na página 11.

THRUN, S. Exploring artificial intelligence in the new millennium. In: LAKEMEYER, G.; NEBEL, B. (Ed.). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003. cap. Robotic Mapping: A Survey, p. 1–35. ISBN 1-55860-811-7. Disponível em: <http://dl.acm.org/citation.cfm?id=779343.779345>. Citado 2 vezes nas páginas 8 e 28.

THRUN, S.; BURGARD, W.; FOX, D. Localization. In: _____. *Probabilistic Robotics*. [S.l.]: The MIT Press, 2005. p. 157–184. Citado na página 24.

THRUN, S. et al. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, Wiley Subscription Services, Inc., A Wiley Company, v. 23, n. 9, p. 661–692, 2006. ISSN 1556-4967. Disponível em: <http://dx.doi.org/10.1002/rob.20147>. Citado na página 1.

V-REP. *Página principal do v-rep*. Acessado em Julho 2017. <http://http://www.coppeliarobotics.com/>. Citado na página 15.

WHITCOMB, L.; YOERGER, D.; SINGH, H. Advances in doppler-based navigation of underwater robotic vehicles. In: *Proceedings of the IEEE International Conference on Robotics and Automation, 1999*. IEEE, 1999. p. 399–406 vol.1. Disponível em: <http://dx.doi.org/10.1109/ROBOT.1999.770011>. Citado na página 1.

WOLF, D. F. et al. Robótica móvel inteligente: Da simulação às aplicações no mundo real. *Micurso : Jornada de Atualização em Informática (JAI)*, Congresso da SBC, 2009. Citado na página 1.

YGUEL, M.; AYCARD, O.; LAUGIER, C. Wavelet occupancy grids: A method for compact map building. In: _____. *Field and Service Robotics: Results of the 5th International Conference*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 219–230. ISBN 978-3-540-33453-8. Disponível em: <http://dx.doi.org/10.1007/978-3-540-33453-8_19>. Citado na página 9.

Apêndices

APÊNDICE A – Estudo de Caso Cenário 2

Neste Apêndice é apresentado o Estudo de Caso do Cenário 2. São apresentados os resultados de mapeamentos topológicos mantendo pelo menos um dos erros das informações coletadas no ambiente ou parâmetros de agrupamento fixo. Assim, este Estudo de Caso serve para ter uma visão geral das diferenças geradas por alteração dos limiares e parâmetros do mapeamento topológico e agrupamento, respectivamente.

Para o primeiro conjunto de testes é abordado com o erro da localização fixo em 0.3m, variando o limiar de distinguibilidade da localização. Neste conjunto de testes a informação *WiFi fingerprint* não é usada no mapeamento topológico. Assim, o grafo é criado apenas com as informações de pose. Os resultados dos testes são mostrados na Figura 61, em cada imagem o limiar de distinguibilidade usado no teste. Na Figura 62 a média com o número de vértices criados.

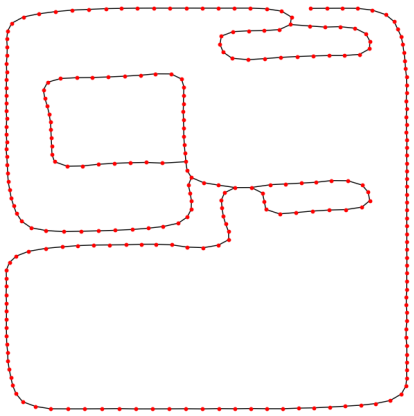
O segundo conjunto de testes é abordado com o limiar de distinguibilidade fixo em 0.3, variando o erro da localização. Neste conjunto de testes a informação *WiFi fingerprint* não é usada no mapeamento. Os resultados dos testes são mostrados na Figura 63, em cada imagem o da localização usado no teste.Na Figura 64 a média com o número de vértices criados.

Para o terceiro conjunto de testes de mapeamento topológico usa o erro da localização 0.03m adquirido por OV em Bayramoglu et al. (2009), veja mais detalhes na Subseção Processamento - localização na Seção 4.1. Neste caso, o limiar de distinguibilidade da localização é fixo em 0.3 e novamente a informação *WiFi fingerprint* não é usada. O resultado é mostrado na Figura 65. Na Figura 66 a média com o número de vértices criados.

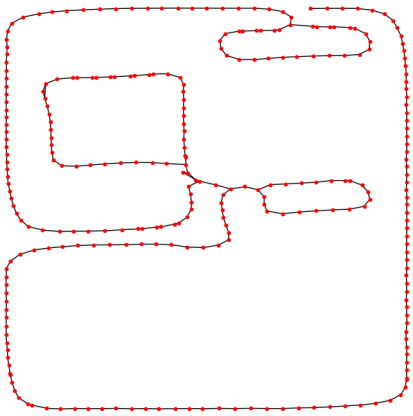
Para os próximos conjuntos de testes são usadas as informações *WiFi fingerprint* e são adotados o erro da localização 0.03m e limiar de distinguibilidade da localização 0.1. O limiar baixo foi selecionado para diminuir a densidade do grafo gerado pelo mapeamento topológico, sendo a esta configuração final da localização para os testes.

O quarto conjunto de testes usa erro *RSS* do *WiFi* fixo em 1, variando o grau de distinguibilidade do *WiFi*. Os resultados dos testes são mostrados na Figura 67, em cada imagem o limiar de distinguibilidade do *WiFi* usado no teste. Na Figura 68 a média com o número de vértices criados.

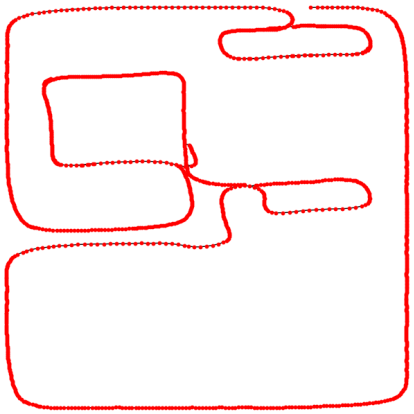
O quinto conjunto de testes mantêm o limiar de distinguibilidade *WiFi* fixo em 0.1, variando o erro *RSS* do *WiFi*. Os resultados podem ser visto na Figura 69, em cada imagem o erro *RSS* do *WiFi*. Na Figura 70 a média com o número de vértices criados.



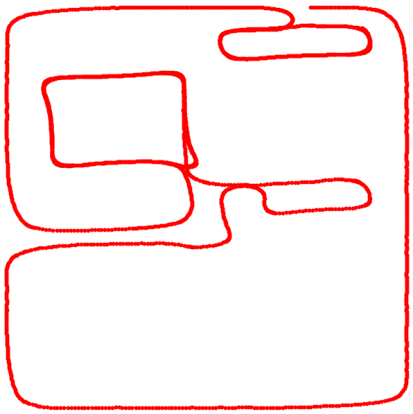
(a) Limiar de distinguibilidade localização: 0.1



(b) Limiar de distinguibilidade localização: 0.3



(c) Limiar de distinguibilidade localização: 0.6



(d) Limiar de distinguibilidade localização: 0.9

Figura 61 – Mapas topológicos gerados com o erro da localização fixo em 0.3m.

O sexto conjunto de testes usa o erro *RSS* do *WiFi* 4.635 ± 3.138 obtido em experimento neste trabalho, mais detalhes na Subseção Processamento - Wifi na Seção 4.1. Nos testes a variação é no limiar de distinguibilidade do *WiFi*. Os resultados são apresentados na Figura 71, em cada imagem o limar de distinguibilidade *WiFi* usado no teste. Na Figura 72 a média com o número de vértices criados.

A partir daqui os conjuntos de testes deste Estudo de Caso são focados nos agrupamentos. Os testes são realizados fixando e/ou variando um determinado parâmetro de agrupamento. A contagem dos conjuntos de testes não reinicia, assim o próximo é o sétimo. O agrupamento tem a função de diminuir a densidade perdendo o mínimo de informação do mapa topológico gerado. E também criar um agrupamento semântico dos vértices no caso deste trabalho, o agrupamento por salas.

O mapa topológico usado para os agrupamentos possui os parâmetros, selecionados

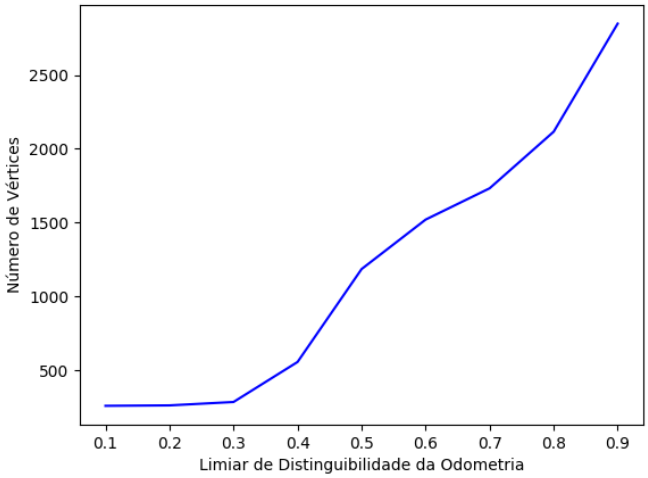


Figura 62 – Número de vértices em média criados com o erro da localização fixo em 0.3m.

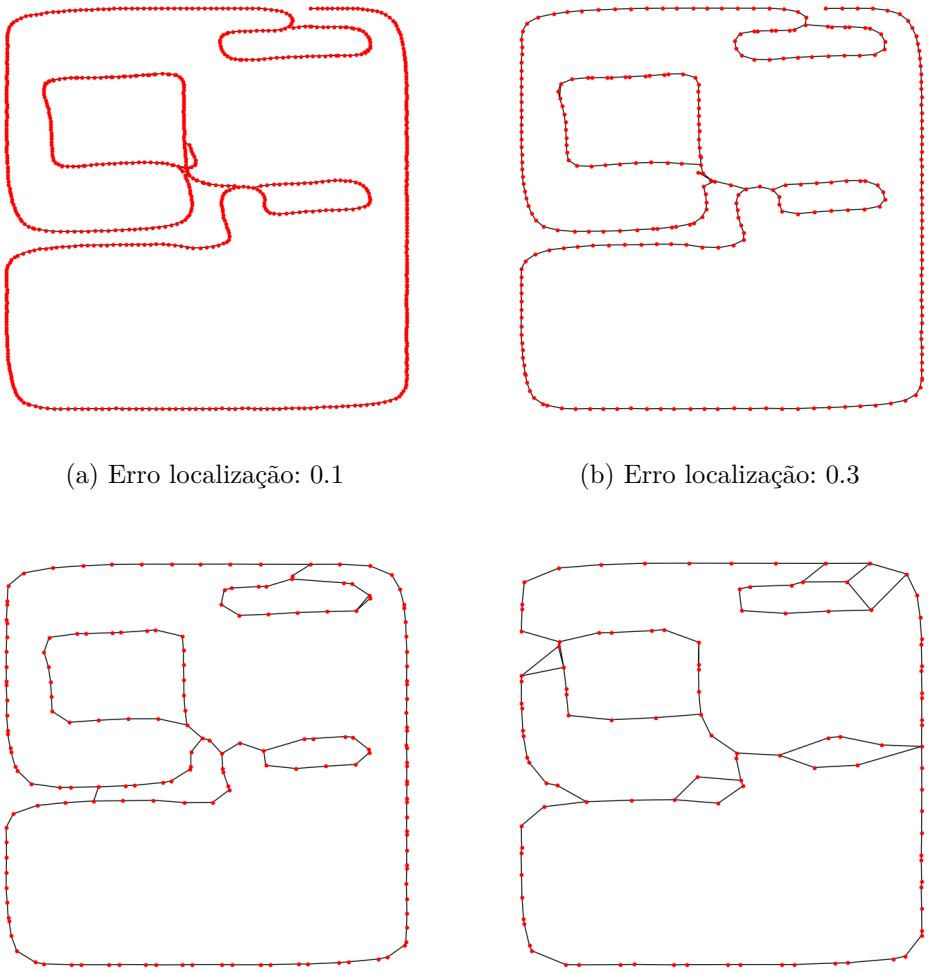
a partir dos experimentos, erro de localização de 0.03m, grau de distinguibilidade da localização de 0.1, erro *RSS WiFi* de 4.635 ± 3.138 e grau de distinguibilidade do *WiFi* de 0.1. Esta é a configuração final para os testes fora do Estudo de Caso. O mapa topológico gerado por essas configurações é apresentado na Figura 71(a).

O sétimo conjunto de testes é o agrupamento apenas com o critério da similaridade, variando seu limiar. O resultados são apresentados na Figura 73, em cada imagem o limiar de similaridade usado. Através dos experimentos detectou-se, que os vértices possuem um limiar de similaridade bem estreito. Na Figura 74 a média com o número de vértices e grupos criados.

O oitavo conjunto de testes é o agrupamento apenas com o critério da número máximo de vértices por grupo, variando seu número. O resultados são apresentados na Figura 75, em cada imagem o limiar de número máximo de vértices usado. Por consequência a alta densidade do grafo, os grupos criados são quase equidistantes. Na Figura 76 a média com o número de vértices e grupos criados.

O nono conjunto de testes é o agrupamento apenas com o critério da distância máxima entre os vértices, variando a distância. O resultados são apresentados na Figura 77, em cada imagem o limiar de distância máxima entre os vértices usada. Com este critério é possível criar grupos equidistantes, a distância ideal para uma boa densidade vértices no grafo varia de acordo com ambiente usado. Na Figura 78 a média com o número de vértices e grupos criados.

O décimo, e último conjunto de testes deste Estudo de Caso, apresenta agrupamentos fixando o critério de similaridade em 0.17 variando os outros dois critérios. Os resultados são apresentados na Figura 79, em cada imagem os limiares usados no teste. Por fim, depois dos experimentos de agrupamento, a configuração de parâmetros final selecionada



(a) Erro localização: 0.1

(b) Erro localização: 0.3

(c) Erro localização: 0.6

(d) Erro localização: 0.9

Figura 63 – Mapas topológicos gerados com limiar de distinguibilidade da localização fixo em 0.3.

para os testes fora do Estudo de Caso são: limiar de similaridade 0.17, número de vértices máximos por grupo 10 e distância máxima entre os vértices 2m.

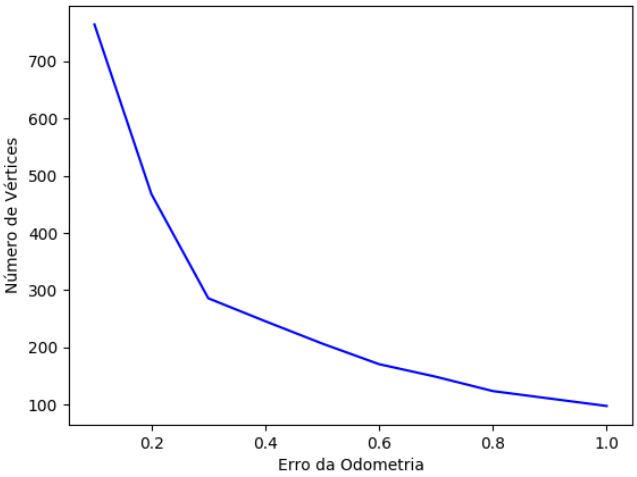
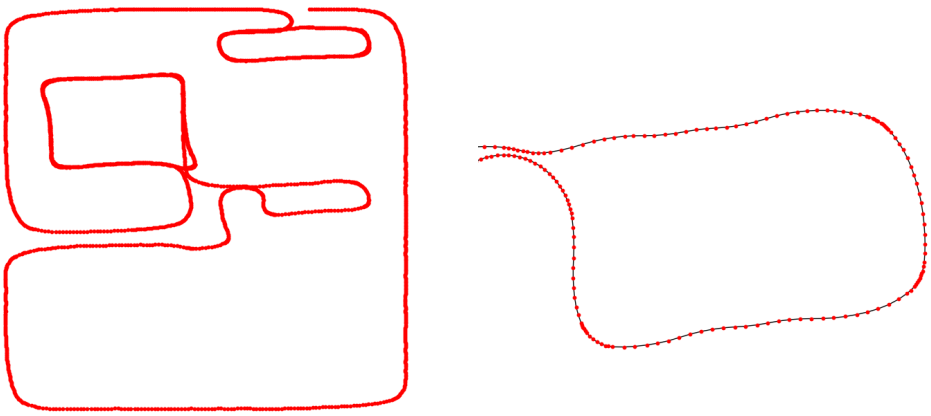


Figura 64 – Número de vértices em média criados com limiar de distinguibilidade da localização fixo em 0.3.



(a) Erro localização: 0.03m (b) Imagem aumentada

Figura 65 – Mapas topológicos gerados com limiar de distinguibilidade da localização fixo em 0.3.

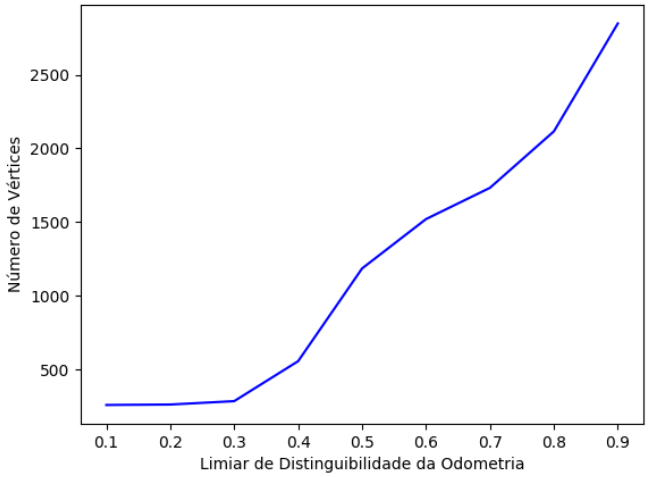
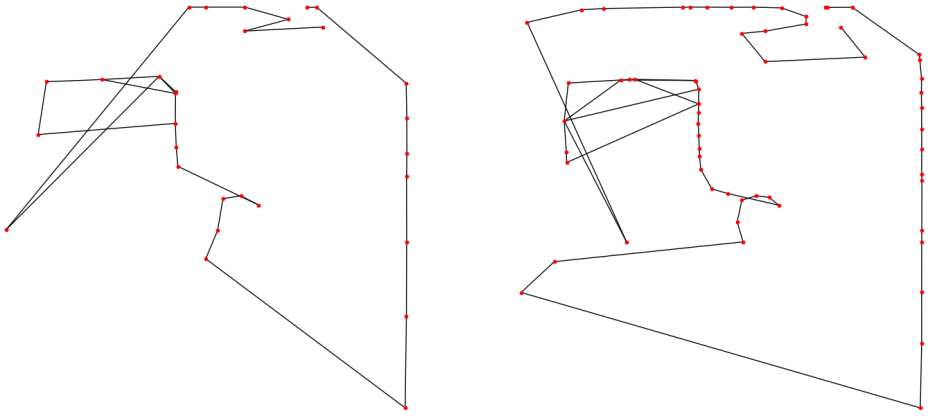
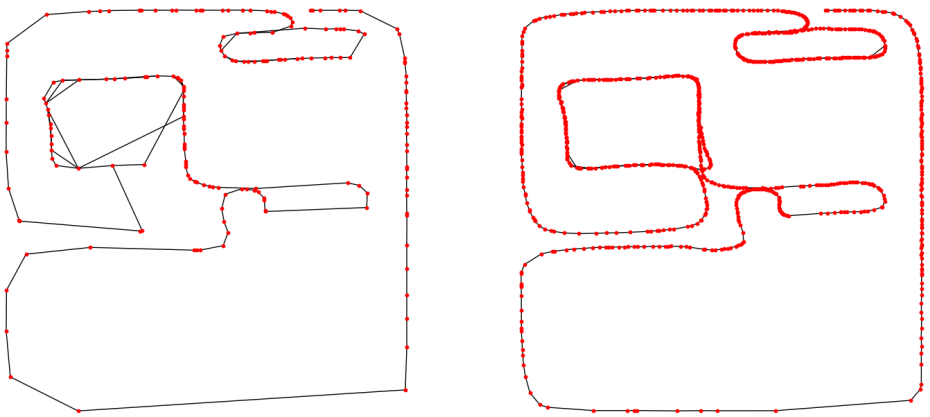


Figura 66 – Número de vértices em média criados com o erro da localização de 0.03m e limiar de distinguibilidade da localização fixo em 0.3.



(a) Limiar de distinguibilidade *WiFi*: 0.1 (b) Limiar de distinguibilidade *WiFi*: 0.3



(c) Limiar de distinguibilidade *WiFi*: 0.6 (d) Limiar de distinguibilidade *WiFi*: 0.9

Figura 67 – Mapas topológicos gerados com erro *RSS* do *WiFi* fixo em 1.

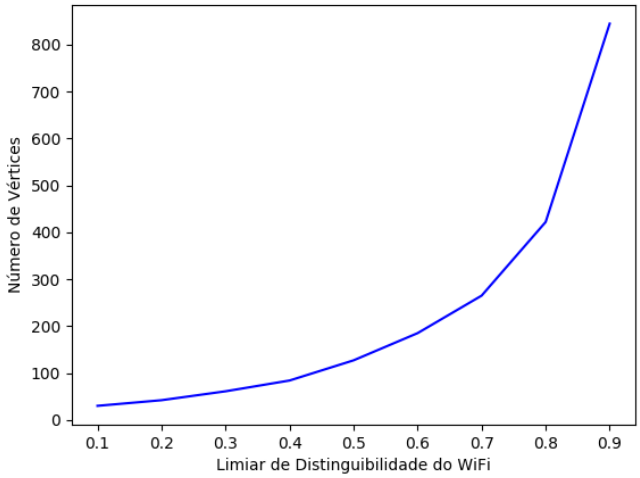
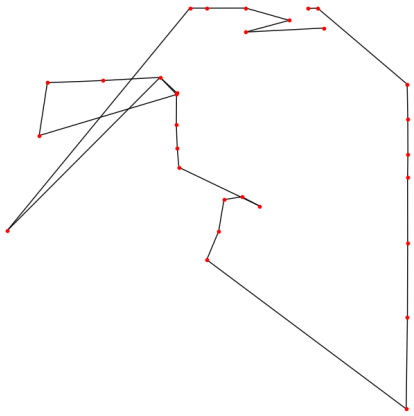
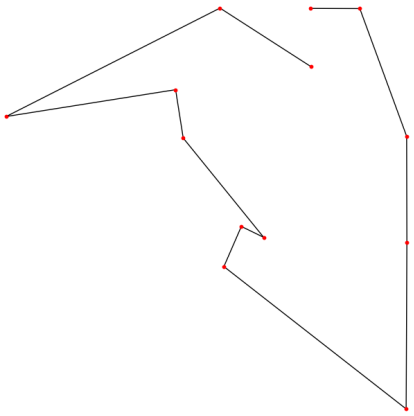


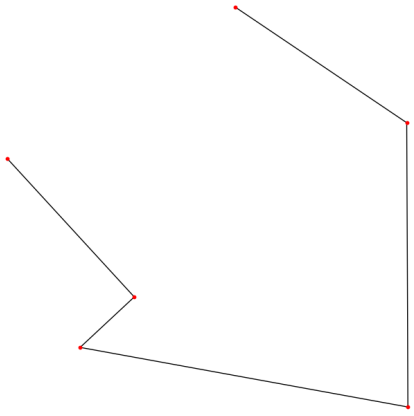
Figura 68 – Número de vértices em média criados com erro *RSS* do *WiFi* fixo em 1.



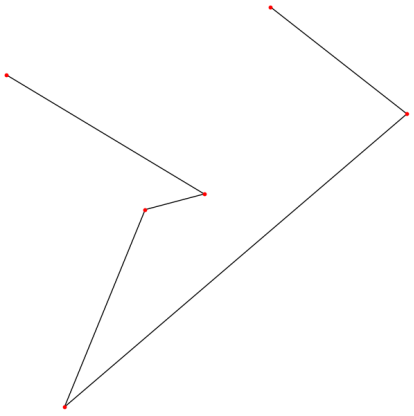
(a) Erro *RSS WiFi*: 1



(b) Erro *RSS WiFi*: 3



(c) Erro *RSS WiFi*: 6



(d) Erro *RSS WiFi*: 9

Figura 69 – Mapas topológicos gerados com limiar de distinguibilidade fixo *WiFi* em 0.1.

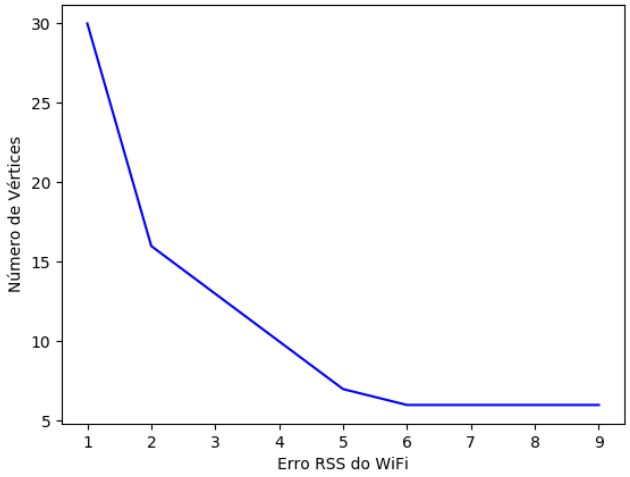
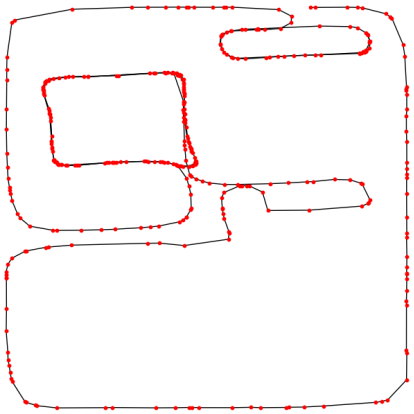
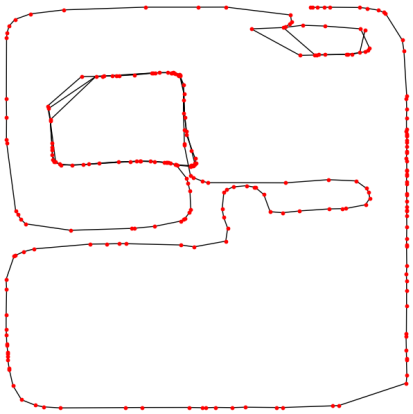


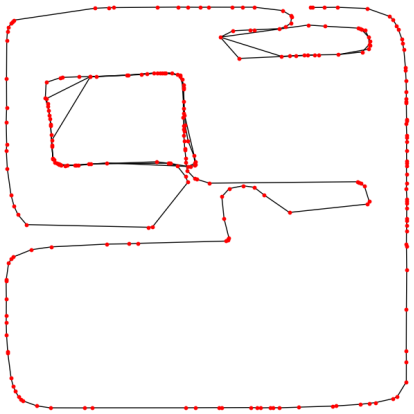
Figura 70 – Número de vértices em média criados com limiar de distinguibilidade *WiFi* fixo em 0.1.



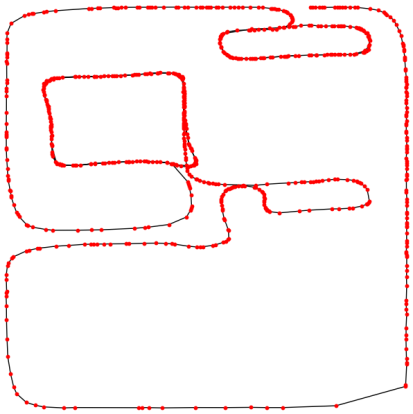
(a) Limiar de distinguibilidade *WiFi*: 0.1



(b) Limiar de distinguibilidade *WiFi*: 0.3



(c) Limiar de distinguibilidade *WiFi*: 0.6



(d) Limiar de distinguibilidade *WiFi*: 0.9

Figura 71 – Mapas topológicos gerados com erro *RSS* do *WiFi* 4.635 ± 3.138.

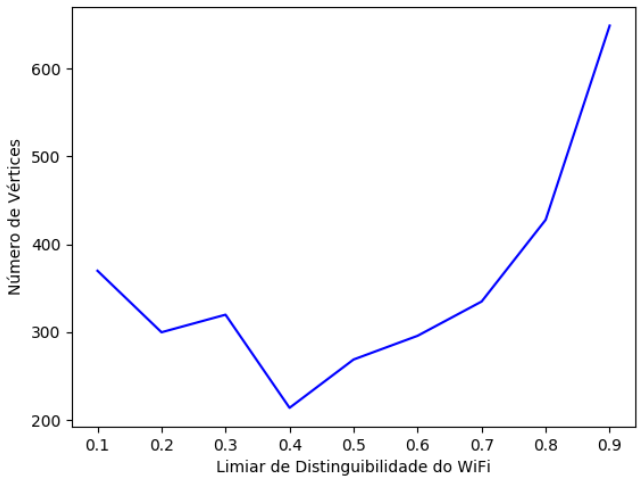
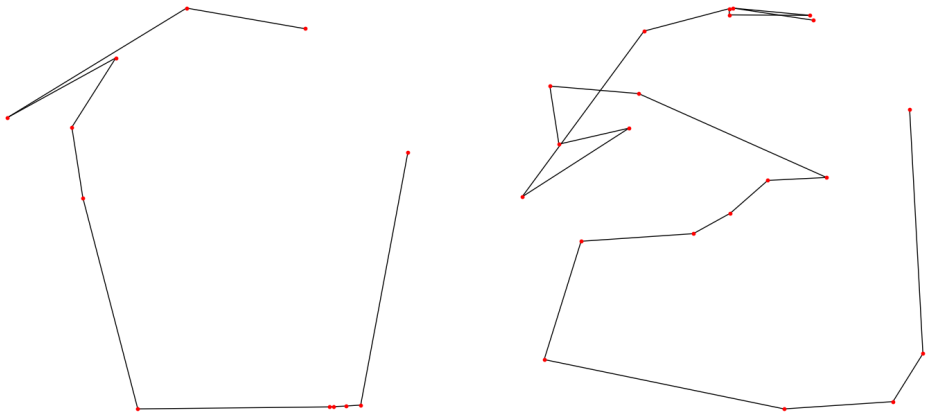
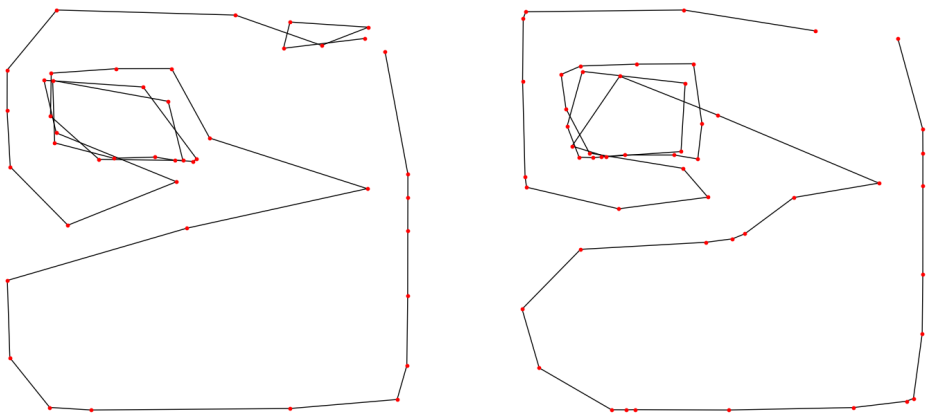


Figura 72 – Número de vértices em média criados com erro *RSS* do *WiFi* 4.635 ± 3.138.



(a) Similaridade: 0.168

(b) Similaridade: 0.17



(c) Similaridade: 0.175

(d) Similaridade: 0.18

Figura 73 – Agrupamentos gerados apenas com o critério de similaridade.

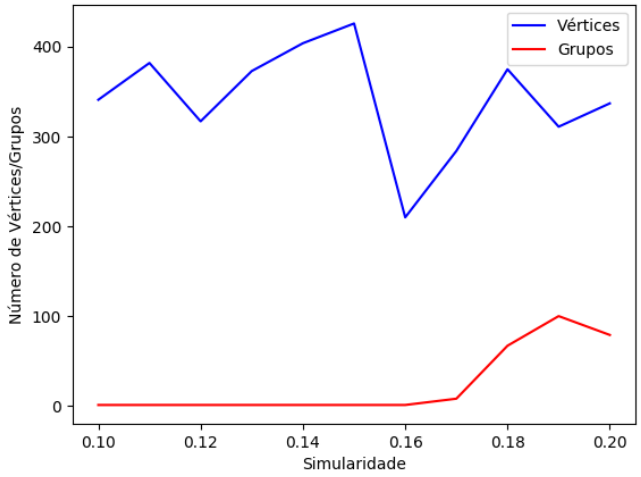
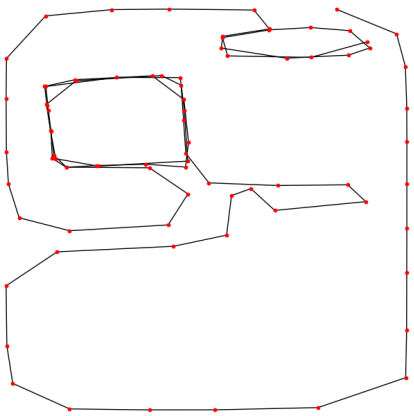
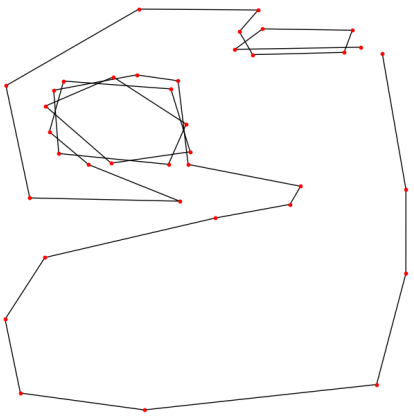


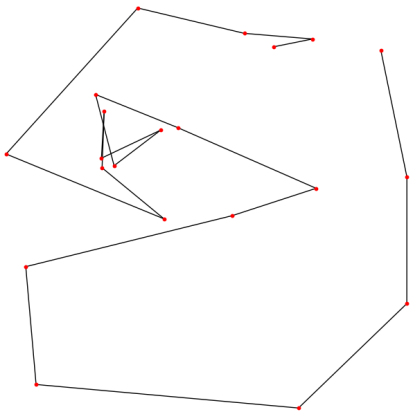
Figura 74 – Número de vértices e grupos em média criados com o critério de similaridade.



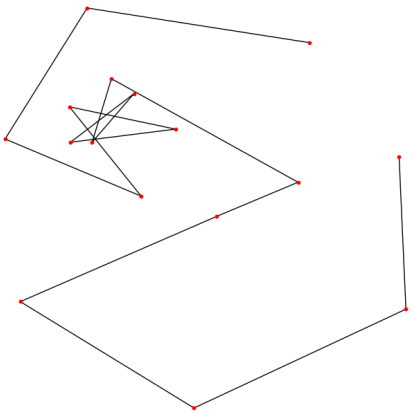
(a) Número máximo de vértices: 5



(b) Número máximo de vértices: 10



(c) Número máximo de vértices: 15



(d) Número máximo de vértices: 20

Figura 75 – Agrupamentos gerados apenas com o critério de número máximo de vértices por grupo.

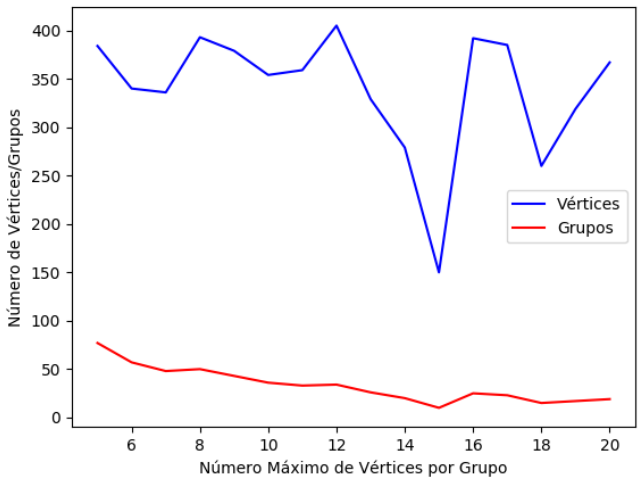
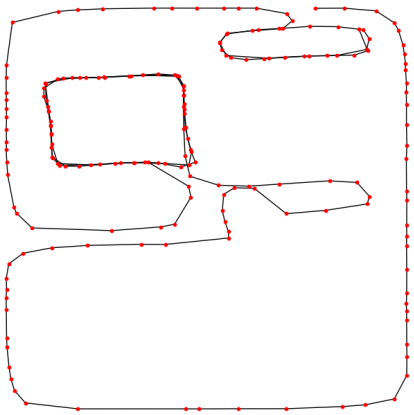
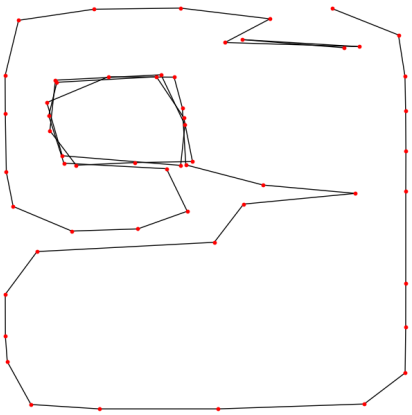


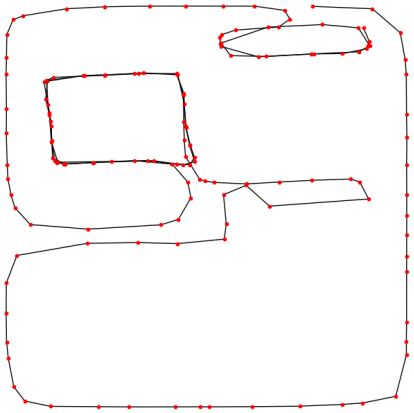
Figura 76 – Número de vértices e grupos em média criados o critério de número máximo de vértices por grupo.



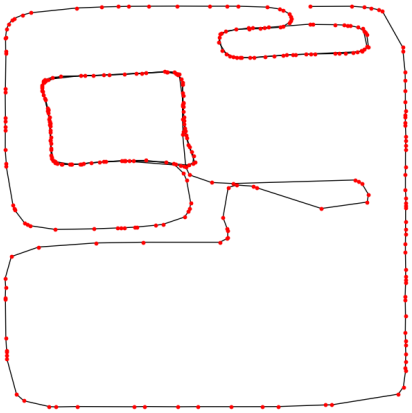
(a) Distância máxima entre os vértices: 0.5m



(b) Distância máxima entre os vértices: 1.0m



(c) Distância máxima entre os vértices: 1.5m



(d) Distância máxima entre os vértices: 2.0m

Figura 77 – Agrupamentos gerados apenas com o critério de distância máxima entre os vértices.

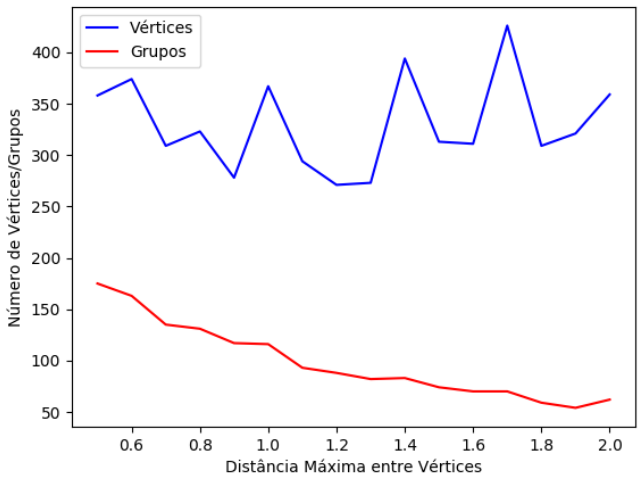
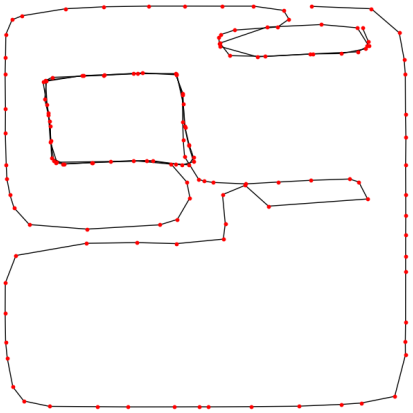
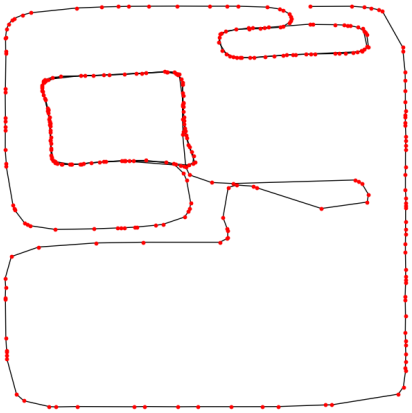


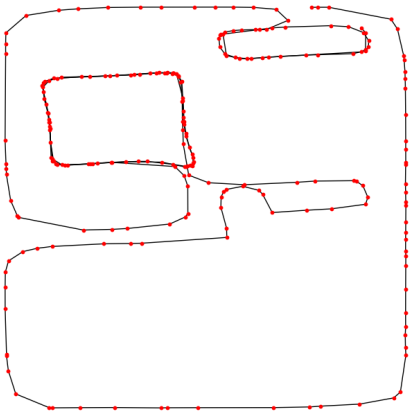
Figura 78 – Número de vértices e grupos em média criados com o critério de distância máxima entre os vértices.



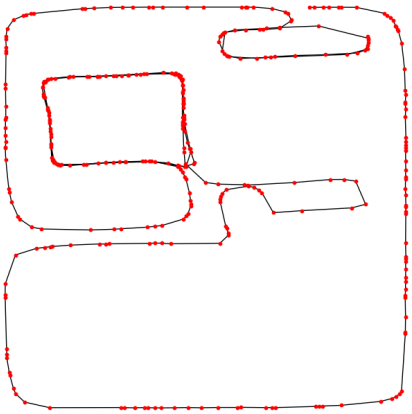
(a) Num.Verts.: 5, Dist.:1m



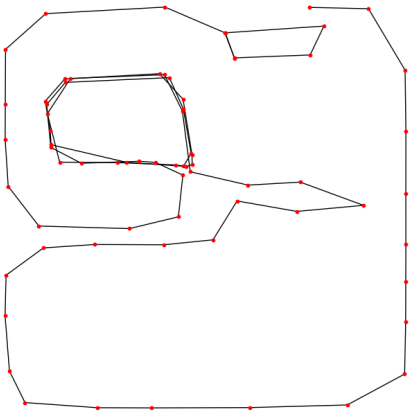
(b) Num.Verts.: 5, Dist.:0.1m



(c) Num.Verts.: 2, Dist.:1m



(d) Num.Verts.: 2, Dist.:0.1m



(e) Num.Verts.: 10, Dist.:2m

Figura 79 – Agrupamentos gerados os três critérios, sendo o de similaridade fixo em 0.17.

APÊNDICE B – Estudo de Caso Cenário 3

Neste Apêndice é apresentado o Estudo de Caso do Cenário 3. São apresentados os resultados de mapeamentos topológicos mantendo pelo menos um dos erros das informações coletadas no ambiente ou parâmetros de agrupamento fixo. Assim, este Estudo de Caso serve para ter uma visão geral das diferenças geradas por alteração dos limiares e parâmetros do mapeamento topológico e agrupamento, respectivamente.

Para o primeiro conjunto de testes é abordado com o erro da localização fixo em 0.3m, variando o limiar de distinguibilidade da localização. Neste conjunto de testes a informação *WiFi fingerprint* não é usada no mapeamento topológico. Assim, o grafo é criado apenas com as informações de pose. Os resultados dos testes são mostrados na Figura 81, em cada imagem o limiar de distinguibilidade usado no teste. Na Figura 82 a média com o número de vértices criados. Não foi possível gerar o gráfico com o grafo com o limiar de distinguibilidade localização de 0.9, pois o mapeamento criou muitos vértices e não teve memória suficiente para completar a operação. O erro mostrado na Figura 80.

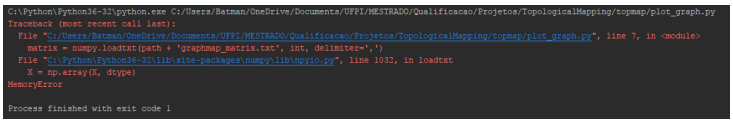


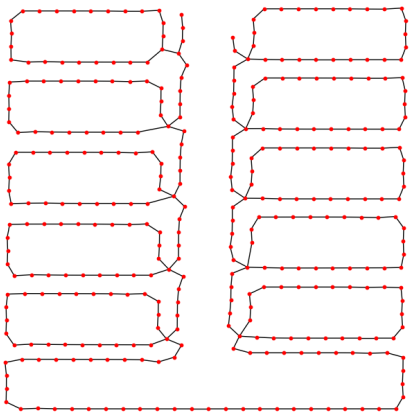
Figura 80 – Erro de memoria ao tentar gerar gráfico de um grafo com muitos vértices.

O segundo conjunto de testes é abordado com o limiar de distinguibilidade fixo em 0.3, variando o erro da localização. Neste conjunto de testes a informação *WiFi fingerprint* não é usada no mapeamento. Os resultados dos testes são mostrados na Figura 83, em cada imagem o da localização usado no teste.Na Figura 84 a média com o número de vértices criados.

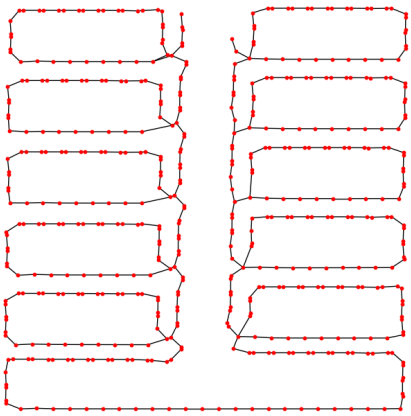
Para o terceiro conjunto de testes de mapeamento topológico usa o erro da localização 0.03m adquirido por OV em Bayramoglu et al. (2009), veja mais detalhes na Subseção Processamento - localização na Seção 4.1. Neste caso, o limiar de distinguibilidade da localização é fixo em 0.3 e novamente a informação *WiFi fingerprint* não é usada. Não foi possível gerar o resultado. Na Figura 85 a média com o número de vértices criados.

Para os próximos conjuntos de testes são usadas as informações *WiFi fingerprint* e são adotados o erro da localização 0.03m e limiar de distinguibilidade da localização 0.1. O limiar baixo foi selecionado para diminuir a densidade do grafo gerado pelo mapeamento topológico, sendo a esta configuração final da localização para os testes.

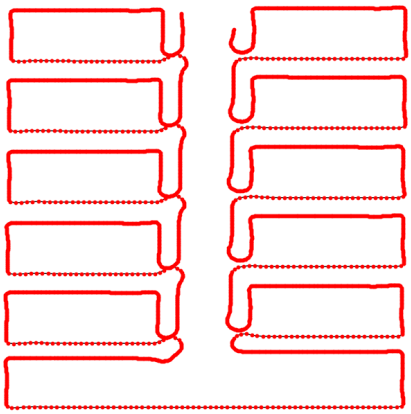
O quarto conjunto de testes usa erro *RSS* do *WiFi* fixo em 1, variando o grau de distinguibilidade do *WiFi*. Os resultados dos testes são mostrados na Figura 86, em cada



(a) Limiar de distinguibilidade localização: 0.1



(b) Limiar de distinguibilidade localização: 0.3



(c) Limiar de distinguibilidade localização: 0.6

Figura 81 – Mapas topológicos gerados com o erro da localização fixo em 0.3m.

imagem o limiar de distinguibilidade do *WiFi* usado no teste. Na Figura 87 a média com o número de vértices criados.

O quinto conjunto de testes mantêm o limiar de distinguibilidade *WiFi* fixo em 0.1, variando o erro *RSS* do *WiFi*. Os resultados podem ser visto na Figura 88, em cada imagem o erro *RSS* do *WiFi*. Na Figura 89 a média com o número de vértices criados.

O sexto conjunto de testes usa o erro *RSS* do *WiFi* 4.635 ± 3.138 obtido em experimento neste trabalho, mais detalhes na Subseção Processamento - Wifi na Seção 4.1. Nos testes a variação é no limiar de distinguibilidade do *WiFi*. Os resultados são apresentados na Figura 90, em cada imagem o limar de distinguibilidade *WiFi* usado no teste. Na Figura 91 a média com o número de vértices criados.

A partir daqui os conjuntos de testes deste Estudo de Caso são focados nos

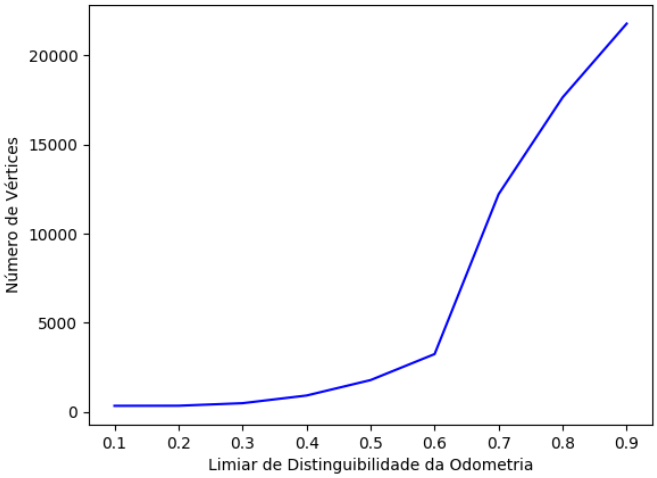


Figura 82 – Número de vértices em média criados com o erro da localização fixo em 0.3m.

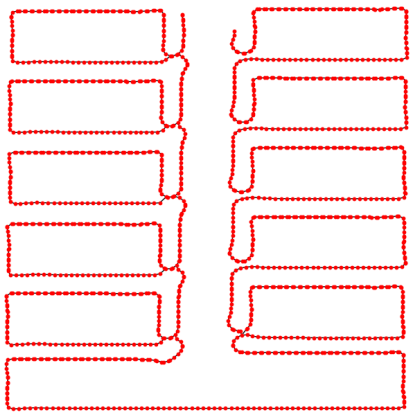
agrupamentos. Os testes são realizados fixando e/ou variando um determinado parâmetro de agrupamento. A contagem dos conjuntos de testes não reinicia, assim o próximo é o sétimo. O agrupamento tem a função de diminuir a densidade perdendo o mínimo de informação do mapa topológico gerado. E também criar um agrupamento semântico dos vértices no caso deste trabalho, o agrupamento por salas.

O mapa topológico usado para os agrupamentos possui os parâmetros, selecionados a partir dos experimentos, erro de localização de 0.03m, grau de distinguibilidade da localização de 0.1, erro *RSS WiFi* de 4.635 ± 3.138 e grau de distinguibilidade do *WiFi* de 0.1. Esta é a configuração final para os testes fora do Estudo de Caso.

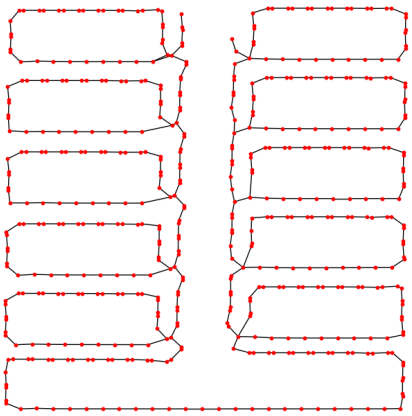
O sétimo conjunto de testes é o agrupamento apenas com o critério da similaridade, variando seu limiar. O resultados são apresentados na Figura 92, em cada imagem o limiar de similaridade usado. Através dos experimentos detectou-se, que os vértices possuem um limiar de similaridade bem estreito. Na Figura 93 a média com o número de vértices e grupos criados.

O oitavo conjunto de testes é o agrupamento apenas com o critério da número máximo de vértices por grupo, variando seu número. O resultados são apresentados na Figura 94, em cada imagem o limiar de número máximo de vértices usado. Por consequência a alta densidade do grafo, os grupos criados são quase equidistantes. Na Figura 95 a média com o número de vértices e grupos criados.

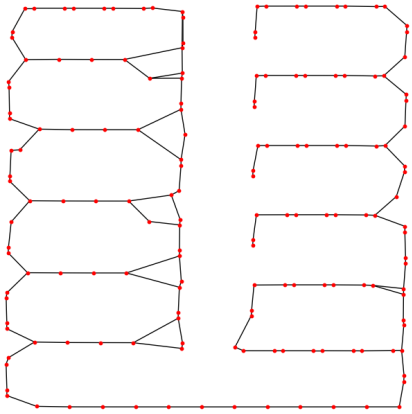
O nono conjunto de testes é o agrupamento apenas com o critério da distância máxima entre os vértices, variando a distância. O resultados são apresentados na Figura 96, em cada imagem o limiar de distância máxima entre os vértices usada. Com este critério é possível criar grupos equidistantes, a distância ideal para uma boa densidade vértices no grafo varia de acordo com ambiente usado. Na Figura 97 a média com o número de



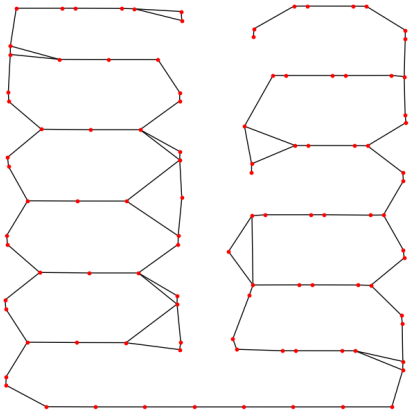
(a) Erro localização: 0.1



(b) Erro localização: 0.3



(c) Erro localização: 0.6



(d) Erro localização: 0.9

Figura 83 – Mapas topológicos gerados com limiar de distinguibilidade da localização fixo em 0.3.

vértices e grupos criados.

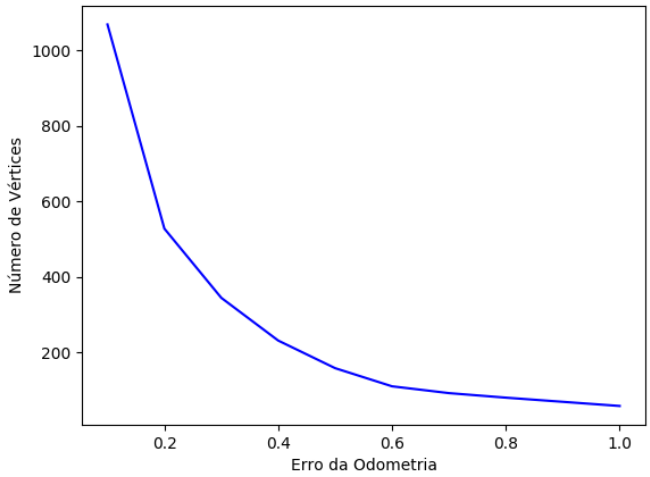


Figura 84 – Número de vértices em média criados com limiar de distinguibilidade da localização fixo em 0.3.

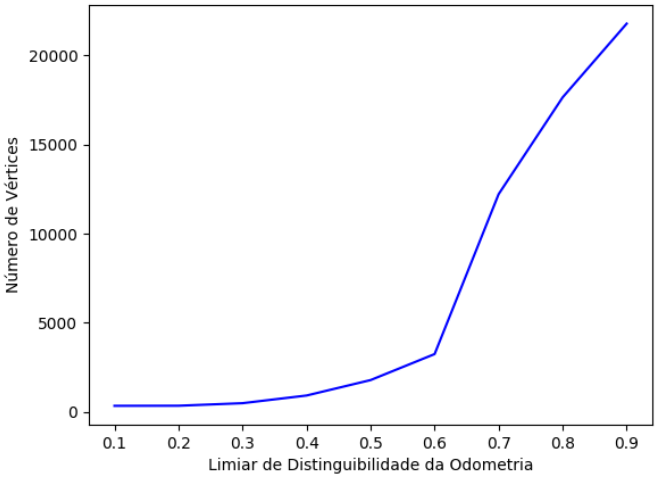
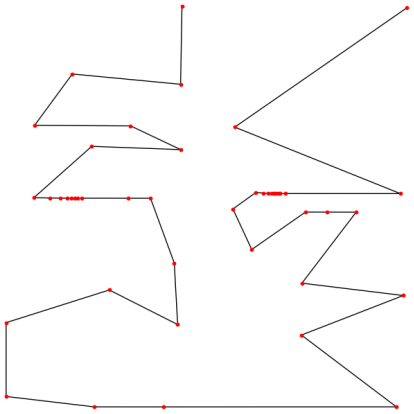
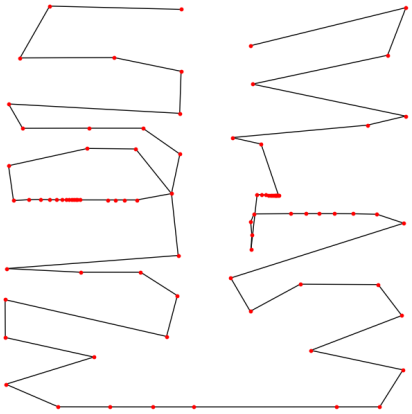


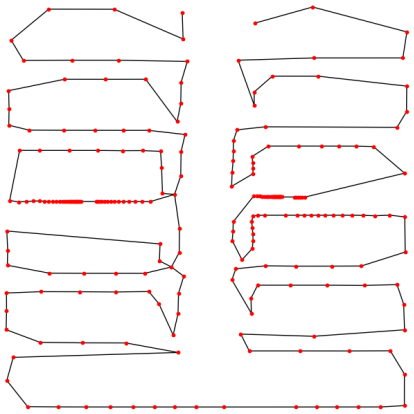
Figura 85 – Número de vértices em média criados com o erro da localização de 0.03m e limiar de distinguibilidade da localização fixo em 0.3.



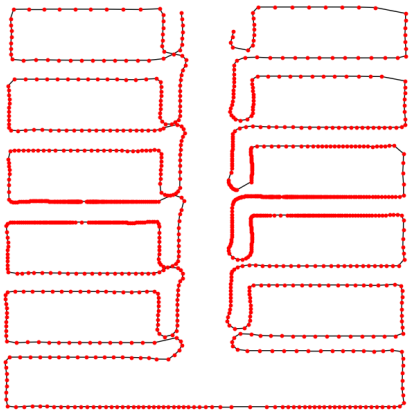
(a) Limiar de distinguibilidade *WiFi*: 0.1



(b) Limiar de distinguibilidade *WiFi*: 0.3



(c) Limiar de distinguibilidade *WiFi*: 0.6



(d) Limiar de distinguibilidade *WiFi*: 0.9

Figura 86 – Mapas topológicos gerados com erro *RSS* do *WiFi* fixo em 1.

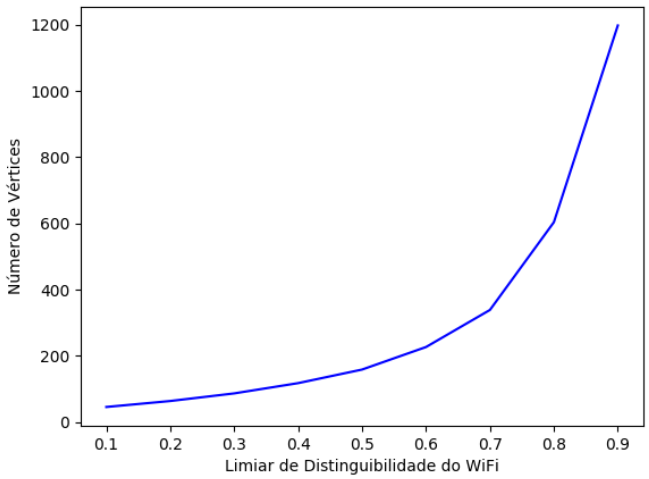
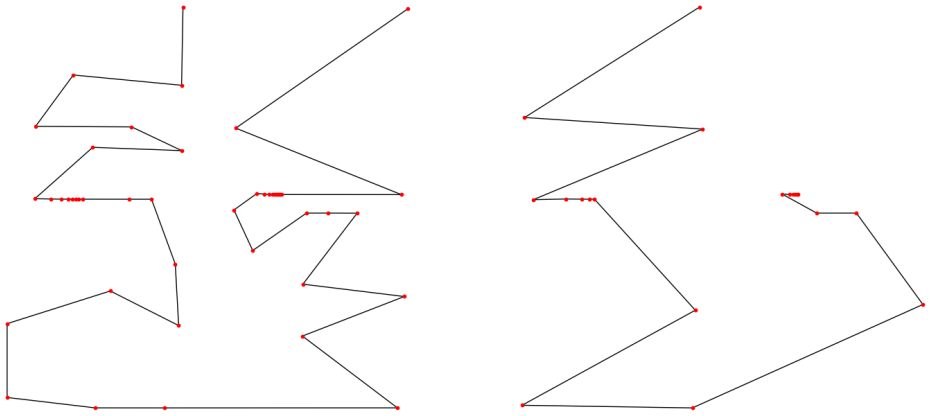
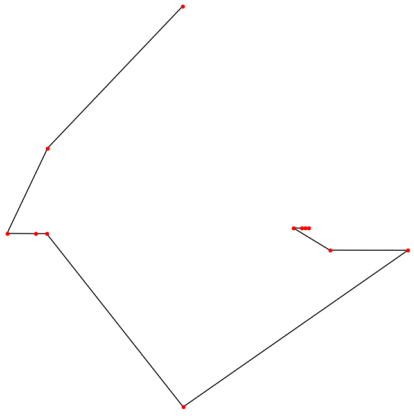


Figura 87 – Número de vértices em média criados com erro *RSS* do *WiFi* fixo em 1.

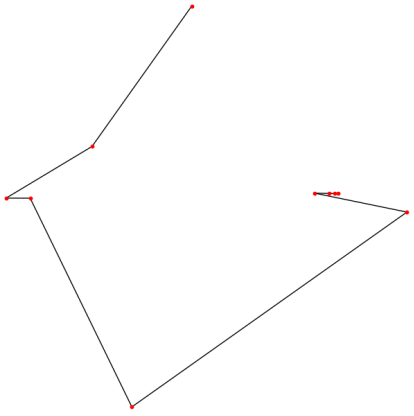


(a) Erro *RSS WiFi*: 1

(b) Erro *RSS WiFi*: 3



(c) Erro *RSS WiFi*: 6



(d) Erro *RSS WiFi*: 9

Figura 88 – Mapas topológicos gerados com limiar de distinguibilidade fixo *WiFi* em 0.1.

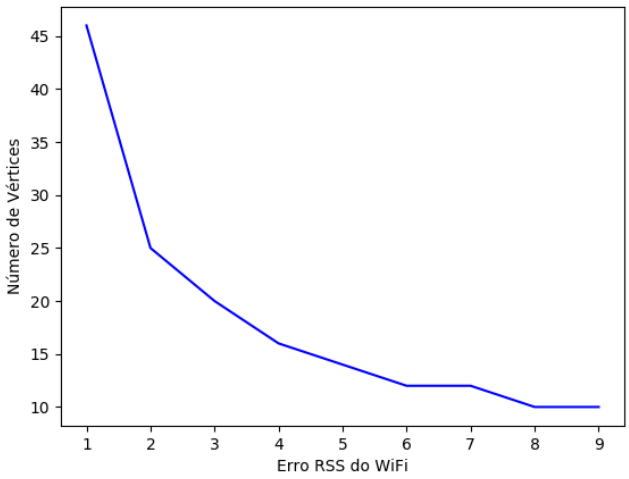
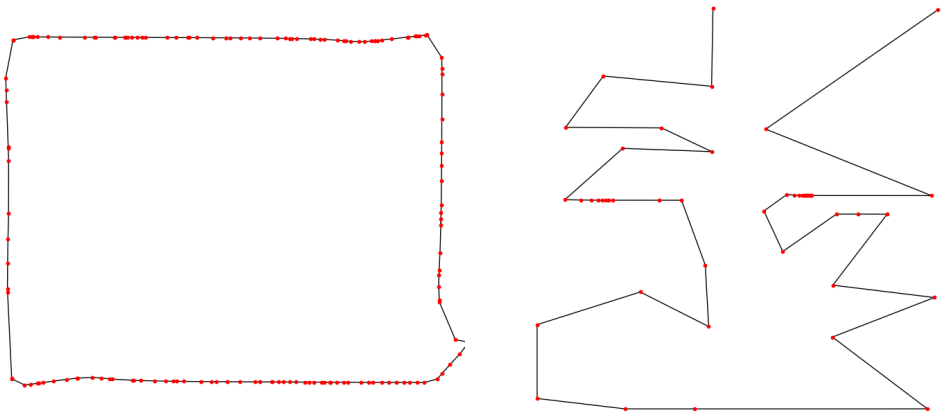


Figura 89 – Número de vértices em média criados com limiar de distinguibilidade *WiFi* fixo em 0.1.



(a) Limiar de distinguibilidade *WiFi*: 0.1 (b) Imagem Ampliada

Figura 90 – Mapas topológicos gerados com erro *RSS* do *WiFi* 4.635 ± 3.138.

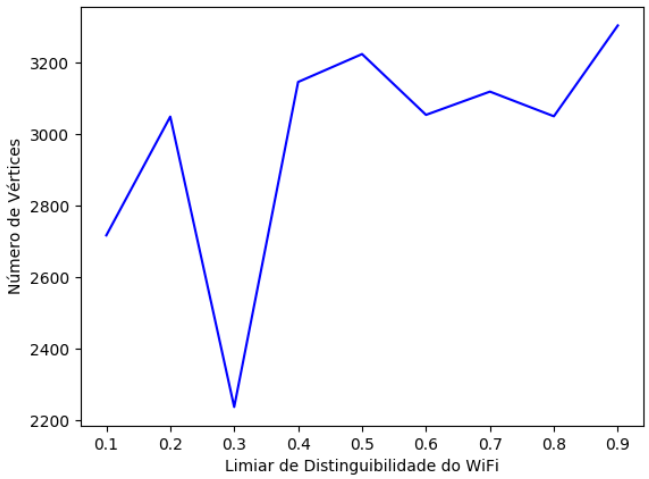
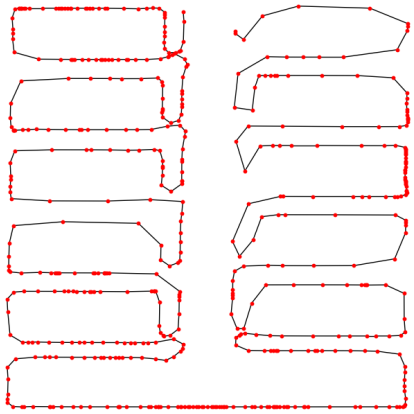
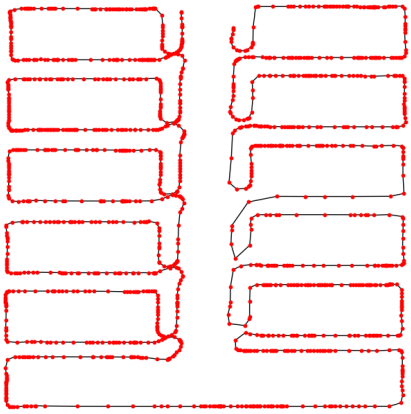


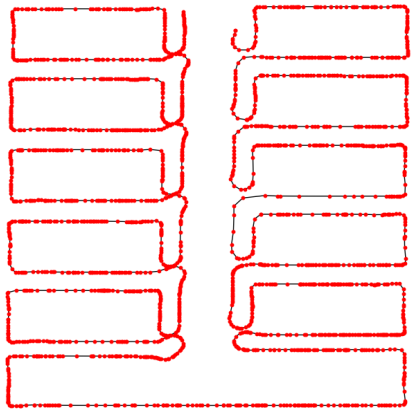
Figura 91 – Número de vértices em média criados com erro *RSS* do *WiFi* 4.635 ± 3.138.



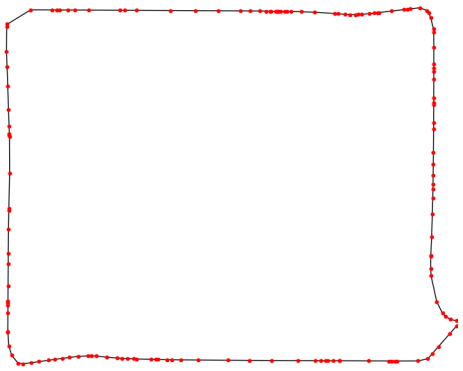
(a) Similaridade: 0.168



(b) Similaridade: 0.17



(c) Similaridade: 0.175



(d) Similaridade: 0.18

Figura 92 – Agrupamentos gerados apenas com o critério de similaridade.

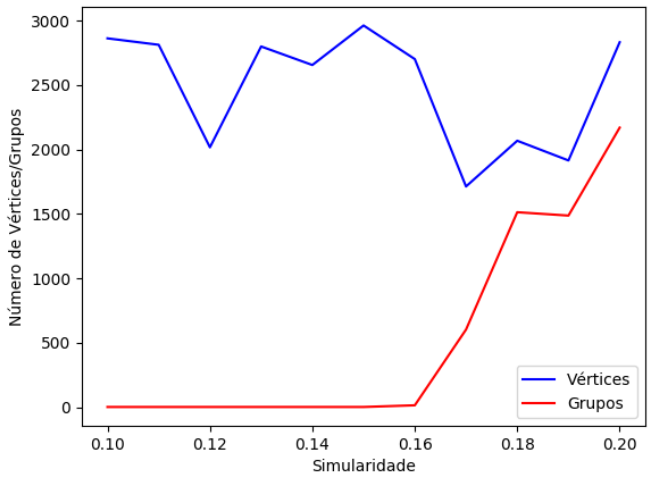
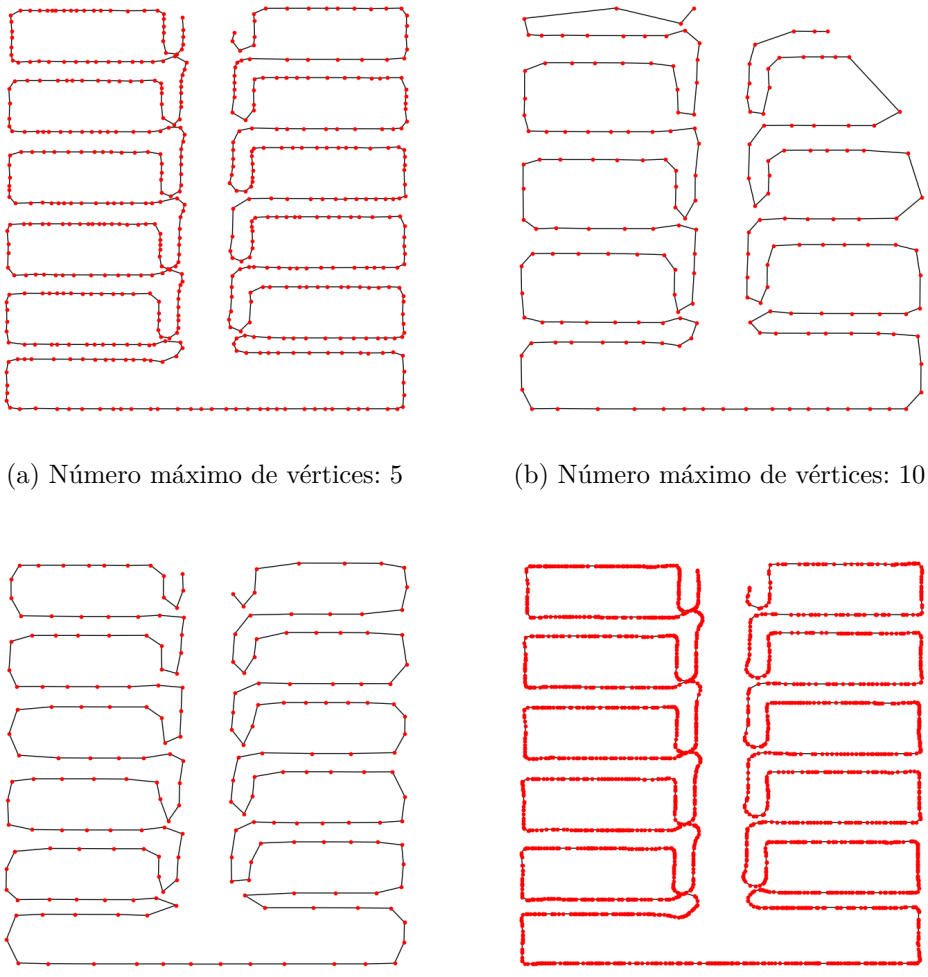


Figura 93 – Número de vértices e grupos em média criados com o critério de similaridade.



(a) Número máximo de vértices: 5

(b) Número máximo de vértices: 10

(c) Número máximo de vértices: 15

(d) Número máximo de vértices: 20

Figura 94 – Agrupamentos gerados apenas com o critério de número máximo de vértices por grupo.

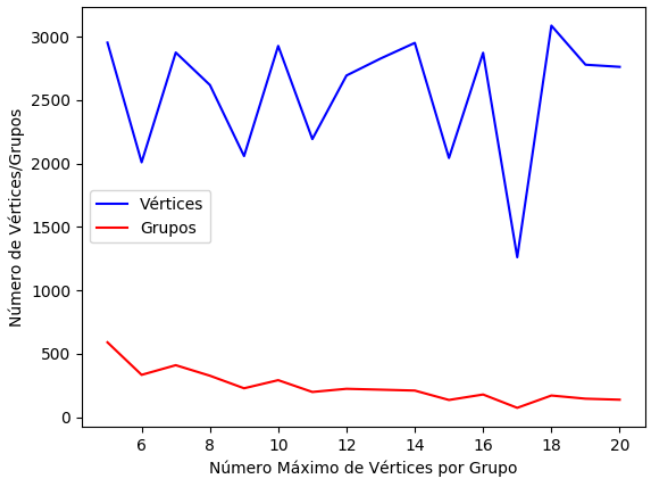
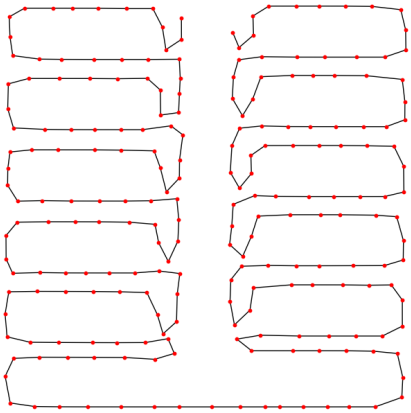
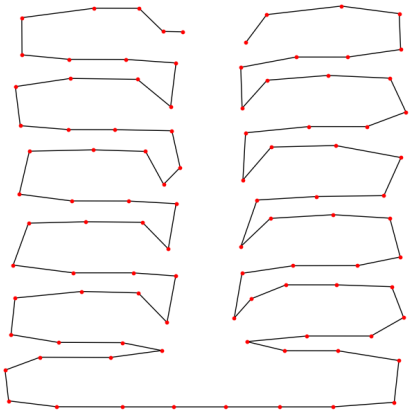


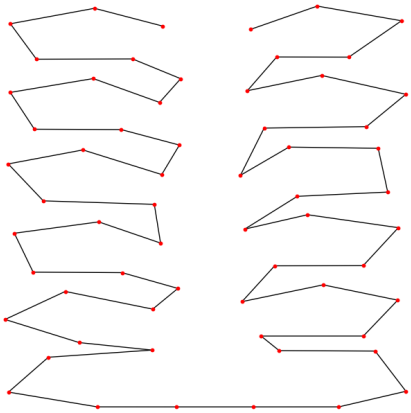
Figura 95 – Número de vértices e grupos em média criados o critério de número máximo de vértices por grupo.



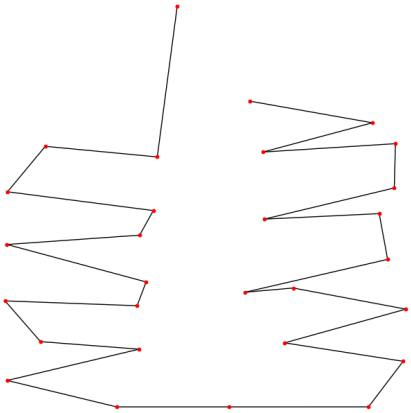
(a) Distância máxima entre os vértices: 0.5m



(b) Distância máxima entre os vértices: 1.0m



(c) Distância máxima entre os vértices: 1.5m



(d) Distância máxima entre os vértices: 2.0m

Figura 96 – Agrupamentos gerados apenas com o critério de distância máxima entre os vértices.

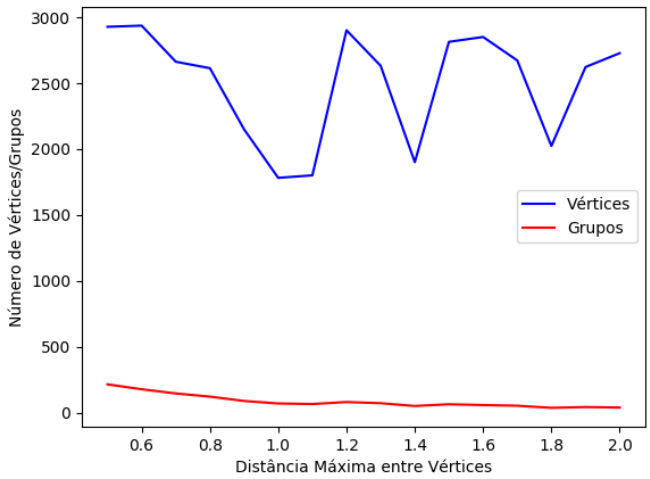


Figura 97 – Número de vértices e grupos em média criados com o critério de distância máxima entre os vértices.