UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ADRIEL FILIPE LINS ALVES DA SILVA

Agrupamento Dinâmico para Reconhecimento Visual de Lugares

Recife

2024

ADRIEL FILIPE LINS ALVES DA SILVA

Agrupamento Dinâmico para Reconhecimento Visual de Lugares

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Inteligência Computacional

Orientador: Aluizio Fausto Ribeiro Araújo

Coorientador: Adrien Joan Sylvain Durand-Petiteville

Recife

2024

.Catalogação de Publicação na Fonte. UFPE - Biblioteca Central

Silva, Adriel Filipe Lins Alves da.
Agrupamento dinâmico para reconhecimento visual de lugares / Adriel Filipe Lins Alves da Silva. - Recife, 2024.
98f.: il.

Dissertação (Mestrado), Universidade Federal de Pernambuco, Centro de Informática, Programa de Pós-Graduação em Ciência da Computação, 2024.
Orientação: Aluizio Fausto Ribeiro Araújo.
Coorientação: Adrien Joan Sylvain Durand- Petiteville.

1. Mapas auto-organizáveis; 2. Reconhecimento visual de lugares; 3. Geo-localização visual. I. Araújo, Aluizio Fausto Ribeiro. II. Petiteville, Adrien Joan Sylvain Durand. III. Título.

UFPE-Biblioteca Central CDD 006.31

Adriel Filipe Lins Alves da Silva

“Agrupamento Dinâmico para Reconhecimento Visual de Lugares”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de Concentração: Inteligência Computacional.

Aprovado em: 28/02/2024.

BANCA EXAMINADORA

_____

Prof. Dr. Carlos Alexandre Barros de Mello
Centro de Informática / UFPE

_____

Prof. Dr. José Alfredo Ferreira Costa
Departamento de Engenharia Elétrica / UFRN

_____

Prof. Dr. Aluizio Fausto Ribeiro Araújo
Centro de Informática/UFPE
(**orientador**)

Dedico à minha família.

AGRADECIMENTOS

Agradeço imensamente à minha mãe e irmã por serem minha rocha nos bons e maus momentos, âncoras da minha existência e presenças insubstituíveis em minha vida.

Agradeço ao professor Adrien e ao professor Aluizio por terem me orientado e me motivado a chegar até aqui.

Agradeço à Mário, Lucas e Marcondes por terem atenção e paciência de me ajudar com várias dúvidas que tive durante essa jornada.

RESUMO

Navegação autônoma desempenha um papel muito importante em aplicações que mostraram uma grande evolução nos últimos anos como, por exemplo, missões espaciais, agricultura, veículos autônomos, robôs de limpeza e logística. Para a realização dessa tarefa, é importante que o robô tenha informação a respeito do ambiente e da sua localização dentro dele. Reconhecimento Visual de Lugares (*Visual Place Recognition* ou VPR) usa somente sensores visuais para o agente autônomo saber o local onde ele está. Esse processo pode ser utilizado de forma independente ou dentro de um sistema de mapeamento e localização simultâneos, como detector de fechamento de laço ou para relocalização. Nesse contexto, há alguns problemas inerentes ao reconhecimento visual de lugares, sendo um dos principais o custo computacional (principalmente para aplicações em longos trajetos), visto que, de uma forma geral, para uma única imagem de busca ser comparada com n imagens de referência armazenadas, a complexidade computacional seria de $O(n)$, o que seria um problema para aplicações em tempo real a longo prazo. Assim, para m imagens de busca, a complexidade computacional seria de $O(m * n)$. Dentro desse contexto, a abordagem proposta neste trabalho consiste na utilização de um Mapa Auto-Organizável com topologia variante no tempo, empregando ciclos de treinamento para analisar o agrupamento das amostras de dados conforme um critério de erro estabelecido para cada protótipo da rede. Esse método leva em consideração a disposição espacial das amostras agrupadas e ajusta os protótipos quando o critério não é atendido. Adicionalmente, o modelo proposto pode lidar com grande quantidade de dados e permite aprendizagem incremental que incorpora novos conhecimentos com pouca ou nenhuma perda de conhecimento anterior. Por último, o modelo proposto determina o número de categorias para o reconhecimento tendo como parâmetro apenas uma relação de semelhança entre o protótipo de uma categoria e as amostras que a integra. O modelo proposto, enquanto se mantém competitivo com relação a outros modelos bem estabelecidos, possui uma menor complexidade computacional na tarefa de reconhecimento de lugares, devido ao agrupamento por meio de SOM, que reduz substancialmente o número de comparações para determinação de um local.

Palavras-chave: mapas auto-organizáveis; reconhecimento visual de lugares; geo-localização visual.

ABSTRACT

Autonomous navigation plays a significant role in applications that have shown huge advancements in recent years, such as space missions, agriculture, autonomous vehicles, cleaning robots, and logistics. To accomplish this task, it is important for the robot to have information about the environment and its location within it. Visual Place Recognition (VPR) uses only visual sensors for the autonomous agent to know its location. This process can be used either independently or within a simultaneous mapping and localization system, such as loop closure detection or for relocalization. In this context, there are some inherent problems in visual place recognition, with one of the main ones being computational cost (especially for applications over long trajectories), since, in general, for a single query image to be compared with n stored reference images, the computational complexity would be $O(n)$, which would be problematic for long-term real-time applications. Thus, for m query images, the computational complexity would be $O(m * n)$. Within this framework, the approach proposed in this work consists of utilizing a Self-Organising Map with time-varying topology, employing training cycles to analyze the clustering of data samples according to an established error criterion for each network prototype. This method takes into account the spatial arrangement of clustered samples and adjusts the prototypes when the criterion is not met. Additionally, the proposed model can handle large amounts of data and allows incremental learning that incorporates new knowledge with little or no loss of previous knowledge. Lastly, the proposed model determines the number of categories for recognition based solely on a similarity relationship between the prototype of a category and the samples it comprises. While remaining competitive compared to other well-established models, the proposed model has lower computational complexity in the task of place recognition, due to clustering through SOM, which substantially reduces the number of comparisons required to determine a location.

Keywords: self-organising maps; visual place recognition; visual geo-location.

LISTA DE FIGURAS

Figura 1 – Modelo do problema de *Simultaneous Localization and Mapping* (SLAM) ilustrado por meio de um grafo. 22

Figura 2 – Esta figura ilustra as duas etapas. Primeiramente a imagem "*Query*"e as imagens do conjunto de dados são transformadas em vetores os quais irão representá-las durante o resto do processo. Posteriormente, é realizada uma busca da imagem "*Query*"em meio ao conjunto de imagens armazenadas. . 24

Figura 3 – Ilustração esquemática do ORB-SLAM 25

Figura 4 – Ilustração da matriz de diferenças entre *frames* recentes e os *templates* de imagem aprendidos. A seta vermelha aponta para uma versão alternativa desta matriz obtida pelo processo de "melhora de contraste". O uso da "melhora de contraste"acentua a discrepância entre os *matchs* realizados na matriz de diferenças. 27

Figura 5 – Busca por trajetórias de melhor score para a primeira imagem de referência. Na imagem, as diferentes linhas indicam as diferentes trajetórias sendo calculadas, as quais, nesse caso, partem da imagem de referência de índice 1. 29

Figura 6 – Uma arquitetura CNN para extração de características profundas. Cada um dos blocos brancos representa um filtro convolucional, enquanto que os blocos azuis são camadas completamente conectadas, combinadas com uma camada *softmax* na saída. 32

Figura 7 – Ilustração esquemática do modelo *Multi-Sensor Fusion* (MPF), em que a imagem de entrada tem suas características extraídas e os diferentes processamentos desses vetores de características extraídos da imagem são ilustrados até o cálculo final da matriz de emissão. 33

Figura 8 – Ilustração do *Self-organising Maps* (SOM), com dimensões x e y, para um vetor de entrada de dimensão D. 37

Figura 9 – Arquitetura DSOM com uma camada escondida. 42

Figura 10 – Arquitetura E-DSOM com uma camada escondida. 43

Figura 11 – Ilustração de uma tarefa de agrupamento. 59

Figura 12 – Diagrama do processo de treinamento do modelo desenvolvido. A sequência gerada pelo filtro é a entrada do sistema e a saída é uma representação da estrutura da rede. 61
Figura 13 – Diagrama da tarefa de *Visual Place Recognition* (VPR) do modelo desenvolvido. As entradas são: a sequência gerada pelo filtro, o conjunto de dados e a entrada atual. A saída é a indicação de qual imagem mais corresponde à entrada atual. No passo 1, as comparações são realizadas entre a saída do filtro e os pesos do nodo (indicados em amarelo), enquanto que no passo 2 as comparações são entre a entrada atual e as imagens agrupadas no nodo selecionado, sendo o nodo selecionado indicado em verde e as imagens indicadas em amarelo. 69
Figura 14 – Ilustração dos dados de posição da trajetória percorrida para formar o base de dados Oxford Robotcar em 32 momentos diferentes, um sobreposto ao outro. 71
Figura 15 – Cada linha ilustra *frames* coletados em diferentes momentos das sequências "2014/12/02", "2015/02/13", "2014/12/10" e "2014/11/14" respectivamente no *data set* Oxford Robotcar. 72
Figura 16 – Ilustração dos dados de posição da trajetória percorrida para formar o base de dados St. Lucia. 72
Figura 17 – Cada linha ilustra *frames* coletados em diferentes momentos das sequências "0845" e "1545" respectivamente no *data set* St. Lucia. 72
Figura 18 – As sequências que serão posteriormente enumeradas de 1 a 5 são dadas respectivamente por: (a) 2014/12/02 com início na imagem 400 com 2050 amostras para referência e 2014/12/10 com início na imagem 1750 com 2050 amostras, (b) 2014/12/02 com início na imagem 1200 com 1250 amostras para referência e 2015/02/13 com início na imagem 2645 com 1410 amostras, (c) 2014/12/02 com início na imagem 8700 com 610 amostras para referência e 2014/12/10 com início na imagem 10360 com 920 amostras para busca, (d) 2014/12/02 para referência com início na imagem 8750 com 2050 amostras e 2015/02/13 com início na imagem 9150 com 2050 amostras para busca, (e) 2014/11/14 para referência com início na imagem 1000 com 2050 amostras e 2014/12/10 com início na imagem 1750 com 2550 amostras para busca. 86

Figura 19 – As subfiguras (a),(b),(c),(d) e (e) são referentes a gráficos de precisão-evocação das sequências 1 à 5 respectivamente. 87
Figura 20 – As sequências (a),(b),(c) e (d) que serão posteriormente enumeradas de 6 a 9 tem as imagens de referência retiradas de toda a trajetória de 2014/12/02, enquanto que as imagens de busca são da sequência 2014/12/10, todas com 2050 amostras e respectivamente com início nas imagens: 120, 2489, 4900 e 7001. Além disso, (e) se refere ao uso das sequências completas de 2014/12/02 e 2014/12/10. 88
Figura 21 – As sequências (a),(b),(c) e (d) que serão posteriormente enumeradas de 10 a 13 tem as imagens de referência retiradas de toda a trajetória de 8:45 am, enquanto que as imagens de busca são da sequência 3:45 pm, todas com 2050 amostras e respectivamente com início nas imagens: 101, 2500, 4800 e 7300. Além disso, (e) se refere ao uso das sequências completas de 8:45 am e 3:45 pm 89
Figura 22 – As subfiguras (a),(b),(c),(d) e (e) são referentes a gráficos de precisão-evocação das sequências 6 à 9 e à sequência completa do base de dados St. Lucia respectivamente. "LARFSOM–ARC neigh" indica que neste caso a tarefa de reconhecimento de lugares foi realizada considerando tanto as imagens agrupadas no nodo vencedor como os seus vizinhos. 90
Figura 23 – As subfiguras (a),(b),(c),(d) e (e) são referentes a gráficos de precisão-evocação das sequências 10 à 13 e à sequência completa do base de dados St. Lucia respectivamente. "LARFSOM–ARC neigh" indica que neste caso a tarefa de reconhecimento de lugares foi realizada considerando tanto as imagens agrupadas no nodo vencedor como os seus vizinhos. 91

LISTA DE TABELAS

Tabela 1 – Precisão para diferentes combinações de experimentos entre as subsequências 2014/12/02 (dia) e 2014/12/10 (noite). 76
Tabela 2 – Precisão para diferentes combinações de experimentos entre as subsequências 2014/12/02 (Dia 1) e 2015/02/13 (Dia 2). 76
Tabela 3 – Precisão para diferentes combinações de experimentos entre as sequências 2014/11/14 (Noite 1) e 2014/12/10 (Noite 2). 77
Tabela 4 – Comparações de desempenho para trechos de sequência no Oxford Robotcar. 78
Tabela 5 – Comparações de tempo para trechos de sequência no Oxford Robotcar. . . 78
Tabela 6 – Comparações de F1 máximo para trechos de sequência no Oxford Robotcar. 79
Tabela 7 – Comparações de evocação para trechos de sequência no Oxford Robotcar. . 79
Tabela 8 – Tabela para as 4 subsequências, comparando resultados do uso de mais de 1 nodo vencedor para a tarefa de reconhecimento. 80
Tabela 9 – Precisão para os diferentes modelos no *dataset* Oxford Robotcar. 81
Tabela 10 – Tempo de execução para os diferentes modelos no *dataset* Oxford Robotcar. 82
Tabela 11 – Máximo escore F1 para os diferentes modelos no *dataset* Oxford Robotcar. 82
Tabela 12 – Evocação para os diferentes modelos no dataset Oxford Robotcar. 82
Tabela 13 – Precisão para os diferentes modelos no data set St. Lucia. 83
Tabela 14 – Tempo de execução para os diferentes modelos no data set St. Lucia. . . . 83
Tabela 15 – Máximo escore F1 para os diferentes modelos no data set St. Lucia. 84
Tabela 16 – Evocação para os diferentes modelos no data set St. Lucia. 84
Tabela 17 – Tabela para todas as sequências consideradas para a base de dados Oxford Robotcar, comparando os resultados do uso de mais de 1 nó vencedor para a tarefa de reconhecimento. 85

LISTA DE ABREVIATURAS E SIGLAS

BA	*Bundle Adjustment*
BMU	*Best Matching Unit*
BOW	*Bag of Words*
BRIEF	*Binary Robust Independent Elementary Features*
CNN	*Convolutional Neural Network*
DA	*Denoising Autoencoders*
DSOM	*Deep Self-Organising Map*
E-DSOM	*Extended Deep Self-Organising Map*
GCS	*Growing Cell Structures*
GHNG	*Growing Hierarchical Neural Gas*
GHSOM	*Growing Hierarchical Self-Organising Map*
GHSORM	Growing Hierarchical Self-Organising Representation Map
GISONN	Growing Incremental Self-Organising Neural Network
GNG	*Growing Neural Gas*
GWR	*Growing When Required*
HMM	*Hidden Markov Model*
HOG	*Histogram of Oriented Gradients*
LARFSOM	*Local Adaptive Receptive Field Self-organising Map*
LARFSOM-ARC	*Local Adaptive Receptive Field Self-organising Map with Adjustable Resemblance for Category*
LASOM	*Location Aware Self-Organising Map*
MPF	*Multi-Sensor Fusion*
ORB	*Oriented FAST and Rotated BRIEF*
SAD	*Sum-of-Absolute-Differences*
SIFT	*Scale-Invariant Feature Transform*

SLAM	*Simultaneous Localization and Mapping*
SOM	*Self-organising Maps*
SOM-TVS	*Self-Organising Maps with Time-Varying Structure*
SURF	*Speeded Up Robust Features*
VLAD	*Vectors of Locally Aggregated Descriptors*
VPR	*Visual Place Recognition*

SUMÁRIO

1 INTRODUÇÃO 16

1.1 OBJETIVOS 18

1.1.1 Objetivo principal 19

1.1.1.1 Objetivos secundários 19

1.2 ESTRUTURA DO TEXTO 19

2 RECONHECIMENTO VISUAL DE LUGARES 21

2.1 TÉCNICAS DE RECONHECIMENTO VISUAL DE LUGARES 25

2.1.1 Técnicas de VPR sem Extração de Características 26

2.1.2 Técnicas de VPR Artesanais 29

2.1.3 Técnicas de VPR baseadas em Aprendizado Profundo 31

2.2 MÉTRICAS DE AVALIAÇÃO PARA RECONHECIMENTO VISUAL DE LUGARES 34

2.3 DESAFIOS E TENDÊNCIAS 35

3 MAPAS AUTO-ORGANIZAVEIS ÚTEIS PARA VPR 37

3.1 SOM PARA EXTRAÇÃO DE CARACTERÍSTICAS E RECONHECIMENTO VISUAL 38

3.1.1 *Localization Aware SOM* 38

3.1.2 *Deep Self-Organising Map* 42

3.1.3 *Extended Deep Self-Organising Map* 43

3.1.4 Relação dos Modelos Mencionados com VPR 44

3.2 SOM COM ESTRUTURA VARIANTE NO TEMPO 44

3.2.1 *Growing Cell Structures* 45

3.2.2 *Growing Neural Gas* 46

3.2.3 *Growing When Required* 48

3.2.4 Local Adaptive Receptive Field Self-Organising Map 49

3.2.5 Growing Incremental Self-Organising Neural Network 52

3.2.6 Growing Hierarchical Neural Gas 53

3.2.7 *Growing Hierarchical Self-Organising Representation Map* 54

3.3 DISCUSSÃO 56

4 MAPA AUTO-ORGANIZÁVEL DE CAMPO RECEPTIVO ADAPTATIVO LOCAL COM SEMELHANÇA AJUSTÁVEL POR CATEGORIA LARFSOM-ARC 58
4.1 EXTRAÇÃO DE CARACTERÍSTICAS 60
4.2 TREINAMENTO 60
4.3 PROCEDIMENTO DE RECONHECIMENTO 68
5 EXPERIMENTOS 70
5.1 BASES DE DADOS 70
5.2 CONFIGURAÇÃO EXPERIMENTAL 73
5.3 ANÁLISE DE RESULTADOS 74
6 CONCLUSÕES 92
REFERÊNCIAS 94

1 INTRODUÇÃO

Sistemas robóticos são uma das soluções mais promissoras quando se trata de lidar com os maiores desafios do futuro e do mundo moderno, tais como produção de comida, transporte e medicina.

A agricultura do século 21 tem de enfrentar dois importantes desafios. O primeiro consiste em aumentar significativamente a produção de comida de alta qualidade e seguro consumo, ração, fibra e biocombustíveis para saciar as necessidades de uma população mundial em constante crescimento (FOLEY et al., 2011). O segundo é que isso deve ser feito de forma economicamente e ambientalmente sustentáveis, de forma a conservar os recursos base, incluindo a biodiversidade, água e solo (LENAIN; TRICOT; BERDUCAT, 2019). Para superar esses desafios, tecnologias de robótica agricultural são essenciais por proverem sensoriamento móvel, computação e atuação que permitam agricultura de precisão, *i.e.*, aplicando os tipos certos e quantidades de insumos nos locais e tempos certos. Com o objetivo de aumentar a segurança nas estradas, o tráfego e a eficiência no transporte, desde o início do século passado, pensava-se no desenvolvimento de carros autônomos. Um dos primeiros projetos que envolveu uma grande quantidade de pessoas e organizações envolvidas foi o projeto Prometheus, iniciado em 1986(GILLAN, 1989). Nesse projeto foi desenvolvido um modelo dinâmico espaço-temporal, *i.e.* que considerava a movimentação no espaço tridimensional de um objeto ao decorrer do tempo, predizendo sua próxima posição(DICKMANNS, 1988). Já hoje, diversas empresas como Ford, Waymo, Tesla, Audi, Toyota, Volvo, Nissan e Volkswagen investem em carros autônomos. Contudo, há ainda muitas preocupações quanto à segurança e problemas legais, visto acidentes fatais envolvendo veículos autônomos, assim como o fato de ainda custarem muito dinheiro.

Robôs autônomos também têm espaço em ambientes médicos, como entrega de suprimentos médicos, transporte de pacientes e robôs assistivos (MARMAGLIO et al., 2023). Eles são uma possível alternativa para evitar possíveis infecções, enquanto aumentam a produtividade no trabalho. Há muitas prospectivas para o uso de robôs em hospitais, já sendo utilizados comercialmente em diferentes situações, como nos seguintes casos: os transportadores autônomos da Clevon são utilizados para transporte de amostras para análise entre hospitais na Estônia; moxi é um robô de entrega de medicamentos, auxiliando e interagindo com enfermeiros nos Estados Unidos; assim como robôs usando luz UV foram utilizados para desinfecção

da COVID-19 em hospitais na China(ZHAO et al., 2022).

Uma ferramenta necessária para autonomamente realizar tarefas é a habilidade de navegar em um ambiente dinâmico em constante mudança. Realmente, em geral o robô tem a necessidade de se locomover para determinados destinos para interagir com o ambiente, *e.g.*, coletar frutas, carregar medicamentos ou levar passageiros aos seus destinos.

O processo de navegação é composto por um conjunto de subprocessos: (i) mapeamento do ambiente em que o robô tem que navegar, (ii) localização do robô em meio ao ambiente, (iii) planejamento do caminho o qual o levará a determinado objetivo e (iv) seguimento de trajetória para atingir dado destino.

Nesse contexto, este trabalho se propõe a focar nos processos de mapeamento e localização em ambientes variáveis no tempo e dinâmicos.

O subprocesso (i) consiste no mapeamento do ambiente usando uma representação geométrica, em que o robô pode localizar a si mesmo - *i.e.*, computar sua posição no mapa, planejar um caminho e o seguir.

Vários trabalhos seguiram essa abordagem e optaram pelo uso de SLAM (*Simultaneous Localization and Mapping*) (DURRANT-WHYTE; BAILEY; DURRANT-WHYTE; BAILEY), em que os processos de mapeamento e localização são desempenhados simultaneamente. Essa solução foi extensivamente explorada, levando a diversos tipos de SLAM, tais como SLAM baseado no filtro de Kalman(HUANG; DISSANAYAKE, 2007), SLAM baseado em filtro de partículas (MONTEMERLO et al., 2003), SLAM baseado em grafos (GRISETTI et al., 2010) ou SLAM Visual (MUR-ARTAL; MONTIEL; TARDOS, 2015). Apesar de resultados mostrarem a possibilidade de navegação em ambientes internos sem muita dinamicidade, abordagens geométricas tiveram dificuldade em lidar com desafios como: ambientes extensos os quais requerem alta capacidade de armazenamento, ambientes em constante mudança tornando mapas obsoletos e elementos dinâmicos os quais não devem ser mapeados.

Roboticistas propuseram, dessa forma, mapear ambientes utilizando mapas topológicos (THRUN et al., 1998). Fazendo isso, eles esperavam mapear ambientes extensos e criar mapas menos sensíveis a mudanças e a elementos dinâmicos. Embora mapas topológicos possam usar a representação geométrica (ĆWIAN et al., 2021), muitos trabalhos propõem a caracterização do ambiente no espaço de aparências. De fato, como câmeras são capazes de prover informações mais abundantes do que sensores de distância, obter representações mais robustas do ambiente deve ser possível. Nesse caso, a localização e mapeamento visuais são chamados de reconhecimento visual de lugares (VPR).

No decorrer da última década, vários trabalhos propuseram métodos para criar mapas topológicos visuais de forma a permitir reconhecimento de lugares de diferente pontos de vista e para mudanças significativas de aparência, seja de período do dia ou de estação do ano (MILFORD; WYETH, 2012). Contudo, a maioria dos métodos propostos são avaliados *offline*, não considerando a necessidade de embarcar o processo de VPR em um sistema robótico e rodá-lo no próprio robô dentro de um intervalo de tempo compatível com o processo de navegação, *i.e.*, ao menos 1 Hz. Isso passa a se tornar um desafio à medida que a trajetória vai aumentando, visto que, de forma geral, sistemas de VPR utilizam técnicas de comparação dos dados armazenados com o dado de entrada atual para determinar a localização do agente autônomo, aumentando o tempo de execução linearmente com a quantidade de dados armazenados.

Com base na argumentação acima, esta dissertação estuda como SOM pode ser utilizado em um processo de VPR de forma a reduzir o tempo de processamento, assim como obter um sistema que possa ser embarcado em um robô para aplicações de longas trajetórias.

Para esse feito, dessarte, o sistema visa atingir os requisitos de baixo tempo de execução, escalabilidade e robustez. Nesse contexto, o SOM desenvolvido agrupa os dados de entrada e, ao invés de realizar comparações entre o dado de entrada com todos os dados armazenados, compara somente com os nodos e, posteriormente, com os dados agrupados nesse nodo, reduzindo o número total de comparações.

Além disso, o mapa desenvolvido é um SOM de estrutura variante no tempo, sendo capaz de se adaptar a novos dados de entrada, possuindo aprendizagem incremental, robusto a mudanças de cenário do ambiente em que o agente autônomo se encontra.

Por fim, testes foram realizados em duas bases de dados bem conhecidas no contexto de VPR em diferentes situações de iluminação e ambientação. Por meio deles foi visto que o sistema apresentado consegue se manter competitivo em termos de desempenho em meio a sistemas bem estabelecidos na literatura, enquanto apresenta um tempo de execução ordens de grandeza abaixo dos sistemas ao qual foi comparado.

1.1 OBJETIVOS

O modelo desenvolvido nesta dissertação utiliza o conceito de mapas auto-organizáveis SOMs com a finalidade de redução de complexidade computacional na tarefa de VPR, aprendizagem incremental, solução escalável, adaptativa às mudanças no ambiente apresentando robustez a perturbações visuais, sendo o objetivo principal e secundários descritos a seguir.

1.1.1 Objetivo principal

- Propor um modelo de baseado em mapa auto-organizável de estrutura variante no tempo o qual é capaz de realizar a tarefa de VPR de forma computacionalmente mais efetiva e com precisão comparável a modelos bem estabelecidos na literatura.

1.1.1.1 Objetivos secundários

- Realizar análise do sistema em diferentes condições de ambientação e iluminação;
- Realizar comparações do desempenho da abordagem proposta com abordagens atuais eficazes que localização visual;
- Realizar experimentos em longos trajetos.

1.2 ESTRUTURA DO TEXTO

O resto da dissertação é dividida da seguinte forma:

- O Capítulo 2 é dedicado e explicar com mais profundidade sobre VPR, detalhando seu procedimento, citando extratores de características comumente utilizados, explicando também sobre SLAM e estabelecendo uma diferenciação entre SLAM visual, localização visual e VPR. Em seguida, esse Capítulo esmiúça algumas técnicas de VPR mais relevantes nos últimos anos, prosseguindo com uma explicação sobre métricas de avaliação nessa área e concluindo com com desafios e tendências.
- No Capítulo 3 há uma apresentação inicial sobre o que é o SOM e uma revisão sobre os modelos de SOM de estrutura variantes no tempo, assim como modelos com aplicações similares ao que foi desenvolvido nesta dissertação, encerrando com uma seção explicando sobre partes similares do modelo desenvolvido a modelos anteriores e capacitações que foram introduzidas.
- No Capítulo 4 é apresentado o modelo proposto, as motivações e adequações de sua utilização e como é realizado o processo de treinamento e teste.

- No Capítulo 5 são apresentados os experimentos realizados com o modelo em múltiplas sequências e subsequências em cenários sob diferentes condições de ambiente, tanto para determinar a melhor configuração do modelo, quanto para comparar com outros modelos.
- No Capítulo 6 é apresentada a conclusão deste trabalho, destacando os pontos importantes trazidos pelo modelo e uma discussão sobre trabalhos a serem realizados futuramente.

2 RECONHECIMENTO VISUAL DE LUGARES

No contexto de um agente autônomo iniciar sua movimentação em uma dada posição e explorar o ambiente no qual está imerso, coletanto dados ruidosos de sensores para se localizar ao longo do tempo, o SLAM visa empregar os dados recebidos desses sensores para estabelecer a localização do robô, enquanto que, simultaneamente mapeia todo ambiente explorado.

O SLAM começou sendo visto como um problema de filtragem (SMITH; CHEESEMAN), formalmente podendo ser descrito como em SICILIANO (2016), cujos conceitos fundamentais são detalhados a seguir.

Defina o caminho como,

$$\boldsymbol{X_T} = \{x_1, x_2, ..., x_T\}. \tag{2.1}$$

Nessa equação, $\boldsymbol{T}$ é o tempo terminal e cada elemento é definido como um local x_n, com $n = 1, 2, ..., T$. Além disso, $\boldsymbol{U_T}$ é definido como sendo a movimentação do robô,

$$\boldsymbol{U_T} = \{u_1, u_2, ..., u_T\}, \tag{2.2}$$

sendo u_n a odometria no instante n. Os dados odométricos poderiam ser o suficiente para saber o posicionamento do robô durante toda a trajetória. Contudo, na realidade, esses dados são ruidosos e o erro é integrado conforme o decorrer do tempo. Sendo assim, adicionam-se as medições do ambiente ($\boldsymbol{Z_T}$) em cada posição no instante n,

$$\boldsymbol{Z_T} = \{z_1, z_2, ..., z_T\}. \tag{2.3}$$

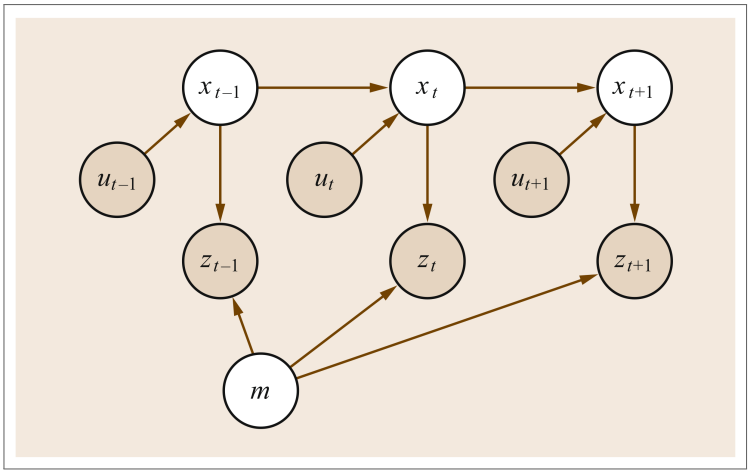
Destarte, considerando m como sendo o mapa do ambiente, SLAM é o problema de tentar obter o modelo do ambiente ao mesmo tempo que procura descobrir as posições do robô no meio desse ambiente, como descrito na equação a seguir,

$$p(\boldsymbol{X_T}, m|\boldsymbol{Z_T}, \boldsymbol{U_T}), \tag{2.4}$$

em que $p(.)$ é a probabilidade do novo local inserido no mapa, visto que se conhecem os dados prévios de odometria e medições do ambiente. Esse processo é ilustrado na Figura 1

Há diversos sensores utilizados para estimativa da posição em SLAM, cada um possuindo suas peculiaridades, vantagens e desvantagens. Os sensores Lidar são reconhecidos por sua alta

Figura 1 – Modelo do problema de SLAM ilustrado por meio de um grafo.



Fonte: MUR-ARTAL; MONTIEL; TARDOS (2015)

precisão e detalhamento do ambiente, embora tenham um pior desempenho melhor no caso de chuva e neblina, devido a fenômenos de refração e reflexão da luz, além de possuírem um maior custo, quando comparados a outros sensores em geral. Já câmeras são sensores mais baratos, que também trazem uma riqueza de detalhes por meio de imagens, embora as técnicas até o momento utilizando esses sensores possuem um pior desempenho se comparadas aos sensores Lidar. Sensores acústicos, em contrapartida, são encontrados geralmente em situações em que o agente autônomo se encontra submerso na água, fazendo com que outros sensores não atuem tão bem por diferentes razões. Por sua vez, radares são encontrados em cenários que podem trazer adversidades no caso do uso de outros sensores, como em situações de chuva ou de neve. Contudo, na literatura o desempenho desses sensores para a tarefa de localização depende muito do tipo de algoritmo sendo utilizado (ZAFFAR et al., 2018).

Mais recentemente, SLAM Visual é algo que vem ganhando cada vez mais espaço principalmente devido ao custo das câmeras com relação a outros sensores responsáveis pela aquisição de dados do ambiente em conjunto com a capacidade de informação que esses sensores conseguem adquirir (CHEN et al., 2022). Dois dos sistemas mais bem estabelecidos atualmente são os chamados ORB-SLAM (MUR-ARTAL; MONTIEL; TARDOS, 2015) e S-PTAM (PIRE et al., 2015).

Outrossim, Sistemas SLAM mais recentemente são sistemas complexos, bastante distintos entre si, e comumente compostos por diversos módulos, sendo importante enfatizar que cada módulo enseja um ou múltiplos temas de pesquisa. Extração de características (LEE et al., 2019), técnicas de fechamento de laço (LABBE; MICHAUD, 2014), métodos de otimização (LABBE; MICHAUD, 2014) e métodos de redução de dimensionalidade (DONG et al., 2023) são

exemplos desses temas.

No problema tratado nesta dissertação, o agente autônomo possui câmeras embutidas e possui a tarefa de se localizar no ambiente sem a necessidade de saber sua localização precisamente. Esse caso específico é tratado como SLAM topológico visual ou comumente conhecido por reconhecimento visual de lugares (VPR), aproveitando o conceito de recuperação de imagem (*image retrieval* em inglês) do campo de estudo de visão computacional. De uma maneira formal, VPR diz respeito ao problema de determinar o lugar específico da imagem em análise, a partir da representação do ambiente composta por imagens anteriormente capturadas. A representação de uma imagem pode ser formada de diversas formas, sendo a mais comum delas, um vetor de características (dados extraídos) de uma ou de múltiplas imagens. O VPR pode ser dividido em 2 etapas como mostra o fluxograma apresentado na Figura 2(MASONE; CAPUTO, 2021).

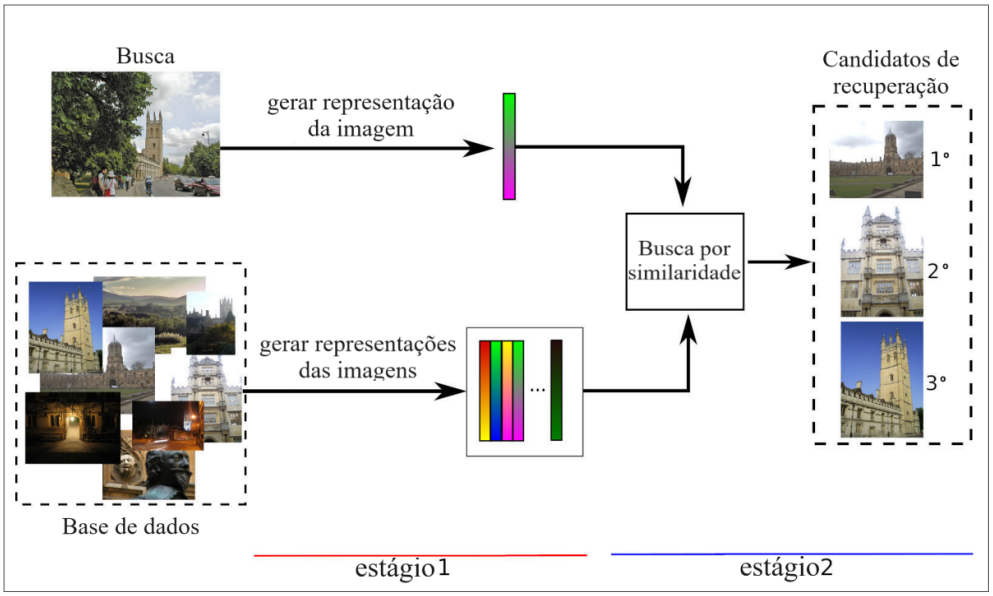
A primeira etapa usualmente emprega extratores de características ou utiliza os dados crus da imagem para realizar o reconhecimento como em (MILFORD; WYETH, 2012). Os extratores podem ser globais, criando uma representação geral de uma imagem (DALAL; TRIGGS, 2005) considerada por meio de vetores, ou extratores locais que extraem características de regiões específicos de uma imagem (*e.g.* pontos-chave), como no caso de descritores artesanais (do inglês *handcrafted descriptors*) mais comuns SURF (BAY; TUYTELAARS; GOOL, 2006), SIFT (LOWE, 2004), BRIEF (CALONDER et al., 2010) ou ORB (RUBLEE et al., 2011). Além de artesanais, os descritores também podem ser baseados em treinamento, por exemplo, aqueles que empregam redes convolucionais profundas como em SIMONYAN; ZISSERMAN (2014), CHEN et al. (2017) ou LI et al. (2020).

A segunda etapa do processamento do VPR é caracterizada pelas comparações entre imagens para determinar a localização da imagem atual. Nesse caso, a busca pode ser por meio das representações das imagens (GLOVER et al., 2012), sequências (MILFORD; WYETH, 2012) ou conjunto de imagens (MUR-ARTAL; MONTIEL; TARDOS, 2015). Essas comparações variam de acordo com as aplicações tais como recuperação de imagens, detecção de sobreposição visual, reconhecimento e recuperação de pontos de referência visuais e recuperação de vídeo (GARG; FISCHER; MILFORD, 2021).

Por fim, em casos mais práticos, há uma terceira etapa que consiste no pós-processamento, geralmente utilizado em sistemas como tratados em MUR-ARTAL; MONTIEL; TARDOS (2015) e KRAJNÍK et al. (2017), ela corresponde a utilização de métodos como verificação espacial, reordenação não-geométrica, expansão de consulta e difusão, para a melhora de desempenho de

um sistema de VPR. Os métodos mais populares em robótica são os de verificação espacial, que têm como ideia principal primeiramente detectar correspondências entre características de um par de imagens e então verificar a confiabilidade das correspondências analisando a consistência das transformações espaciais entre elas. Como exemplo uma utilização muito comum realizada nesses casos é a do RANSAC (MARTÍNEZ-OTZETA et al., 2022).

Figura 2 – Esta figura ilustra as duas etapas. Primeiramente a imagem "*Query*"e as imagens do conjunto de dados são transformadas em vetores os quais irão representá-las durante o resto do processo. Posteriormente, é realizada uma busca da imagem "*Query*"em meio ao conjunto de imagens armazenadas.



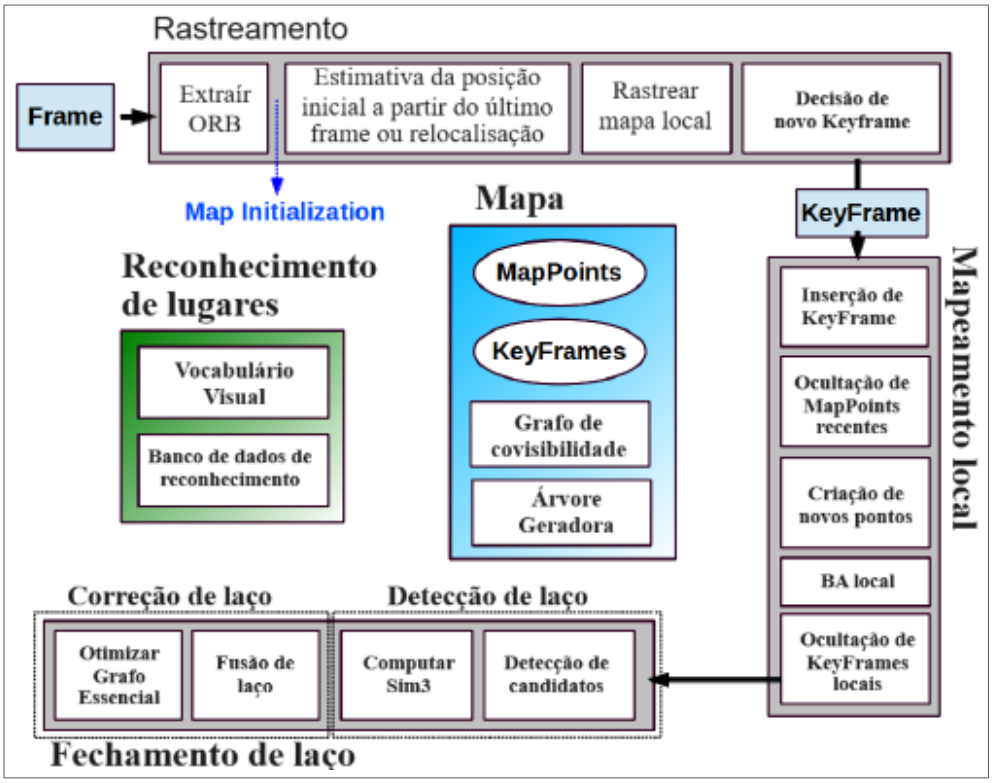
Fonte: Autor, adaptada de MASONE; CAPUTO (2021)

Além de atuar sozinho como um SLAM topológico visual, VPR pode ser encontrado também como componente de um SLAM visual métrico, o qual pode ser dividido, de forma geral, em duas partes: localização visual e o fechamento de laço - em que o VPR se encontra.

Para exemplificar, o modelo ORB-SLAM (Figura 3) possui 5 componentes ou 5 etapas. Rastreamento e Mapeamento local são responsáveis pela localização visual do robô no mapa, enquanto que a tarefa de VPR ocorre em Fechamento de laço (ou para relocalização, em caso de problema na estimativa da posição). Na parte de localização visual, nesse caso, pontos-chave extraídos do *frame* são processados utilizando técnicas de projeção para determinar a posição do robô por meio de um método de otimização chamado *Bundle Adjustment* (BA). Por sua vez, o módulo de fechamento de laço levanta candidatos de possíveis locais os quais o robô possa estar com auxílio da tarefa de VPR e determina se o local no qual o robô está foi algum lugar já explorado ou não. Essa última parte é de vital importância para esse SLAM métrico como um todo, pois caso ela não ocorra, o erro irá constantemente aumentar acu-

mulativamente. Além disso, caso haja um erro no fechamento de laço, as consequências para o mapa como representação do ambiente podem ser fatais, dado que a posição determinada para o fechamento de laço em caso de erro geralmente se encontra muito distante da posição correta.

Figura 3 – Ilustração esquemática do ORB-SLAM



Fonte: Autor, adaptada de MUR-ARTAL; MONTIEL; TARDOS (2015)

Neste capítulo, a Seção 2.1 apresenta detalhas diferentes técnicas de VPR, a Seção 2.2 aborda as métricas para a avaliação do VPR e a Seção 2.3 encerra o capítulo, tratando de desafios e tendências das pesquisas sobre VPR.

2.1 TÉCNICAS DE RECONHECIMENTO VISUAL DE LUGARES

Esta seção é dedicada aos diferentes tipos de sistemas de VPR separados conforme os tipos de extratores de características utilizados. Além disso, o SeqSLAM e o MPF, modelos os quais são comparados com o modelo proposto posteriormente, são aqui descritos.

2.1.1 Técnicas de VPR sem Extração de Características

Essa seção é dedicada ao SeqSLAM, sistema VPR que não possui um extrator de características, mas sim uma técnica de processamento de imagens para seu processo de busca de correspondências para o local atual.

Sum-of-Absolute-Differences (SAD) (HAUSLER; JACOBSON; MILFORD, 2019) é uma técnica de comparação de imagens baseada na comparação de intensidade de pixels. Para operar adequadamente, SAD necessita que o ambiente das imagens a serem comparadas não tenha sofrido variações nas suas condições nem o o ponto de vista tenha se alterado. Como na grande maioria dos casos não é possível atender a estas condições, o modelo demanda controle sobre as variações mencionadas.

SeqSLAM (MILFORD; WYETH, 2012) se baseia nessa técnica, como é descrito posteriormente, para realizar o VPR mesmo que ocorra variações significativas das condições ambientais, tais como mudança de clima e iluminação, embora SeqSLAM perca desempenho em cenários em que ocorra variação de pontos de vista. Contudo, sua abordagem simples que utiliza sequências de imagens ao invés de imagens isoladas para o processo de reconhecimento de lugares se provou ser muito eficaz. SeqSLAM se tornou fonte de inspiração para trabalhos futuros e é visto como um método bem estabelecido de rápida execução, portanto, ele é considerado nesta dissertação para comparação com o método a ser proposto.

No SeqSLAM, a primeira etapa é converter as imagens para escala de cinza e realizar uma redução na amostragem da imagem. Em seguida, apenas uma região no centro da imagem é selecionada, removendo todo o resto da imagem, completando o pré-processamento do método. As imagens de referência podem ser determinadas por um conjunto de imagens previamente armazenadas no robô ou pelo armazenamento dos *frames* de busca mais antigos. Uma vez que todas as imagens de referência (*templates* aprendidos) e os *frames* de busca mais recentes são pré-processados, a primeira etapa do processo de reconhecimento visual consiste na criação de uma matriz de diferenças, na qual esses *frames* mais recentes são comparados com as imagens de referência, como mostrado na matriz à esquerda da Figura 4, por meio da equação do SAD, *i.e.*,

$$D_{i,j} = \frac{1}{R_x R_y} \sum_{x=0}^{R_x} \sum_{y=0}^{R_y} \left| p^i_{x,y} - p^j_{x,y} \right|, \tag{2.5}$$

em que $D_{i,j}$ corresponde a cada elemento da matriz de diferenças $\boldsymbol{D}$, R_x e R_y são os números

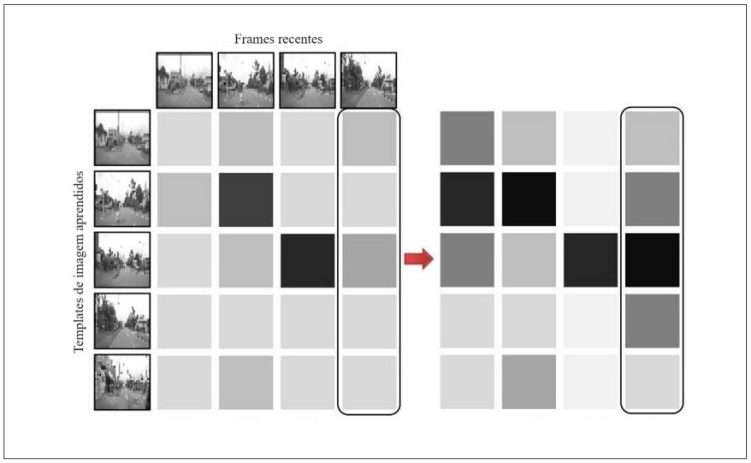
de linhas e de colunas das imagens após o pré-processamento, $p^i_{x,y}$ é a intensidade de cada pixel da imagem de referência de índice i e $p^j_{x,y}$ é a intensidade de cada pixel da imagem de busca de índice j. Nessa figura, j assume os valores de 1 até 4, enquanto que i assume os valores de 1 até 5 (por ser uma matriz $5x4$).

Após o cálculo da matriz de diferenças, ela é submetida a um processo denominado "melhora de contraste local". O objetivo desse processo é fazer com que as comparações entre o *frame* de busca e as imagens de referência sejam determinadas pelo quanto uma imagem de referência inserida em um subconjunto representativo do local mais se assemelha a um *frame* de busca. Para isso, as imagens de referência precisam estar em sequência e determina-se as R imagens de referência mais próximas de uma dada imagem de referência de índice i no texto não fica clara como esta imagem de referência é determinada, determinando o subconjunto de imagens definidores do local. Com esse local definido, para uma imagem de busca j, o cálculo de cada elemento da nova matriz é dado por,

$$\hat{D}_{i,j} = \frac{D_{i,j} - \overline{\boldsymbol{D_l}}^{\boldsymbol{j}}}{\sigma_j}, \tag{2.6}$$

em que $\overline{\boldsymbol{D_l}}^{\boldsymbol{j}}$ é a média de um subconjunto de elementos, dentro dessa região de índice l, contido no vetor coluna j (índice do *frame* de busca correspondente). Por sua vez, σ_j é o desvio padrão desse mesmo subconjunto. Essa operação faz com que correspondências mais fortes apareçam na matriz de diferenças, fazendo com que a matriz ilustrada à esquerda da Figura 4 passe a ter a aparência da matriz à direita.

Figura 4 – Ilustração da matriz de diferenças entre *frames* recentes e os *templates* de imagem aprendidos. A seta vermelha aponta para uma versão alternativa desta matriz obtida pelo processo de "melhora de contraste". O uso da "melhora de contraste"acentua a discrepância entre os *matchs* realizados na matriz de diferenças.



Fonte: Autor, adaptada de MILFORD; WYETH (2012)

Para cada imagem de busca de índice j há uma submatriz da matriz de diferenças utilizada para a tarefa de reconhecimento, como ilustrado na Figura 5, em que as colunas vão de $j - d_s$ até j, com d_s sendo o tamanho da sequência definido pelo usuário. Essa sequência é composta pelas d_s últimas imagens e calcula-se um *score* para aquela imagem pela equação,

$$S_{i,k} = \sum_{n=j-d_s}^{j} \hat{D}_{k,n}. \tag{2.7}$$

em que $\hat{D}_{k,n}$ é um elemento da matriz, com n sendo os índices da sequência correspondente ao *frame* de referência de índice j e, por fim, $S_{i,k}$ é um *score* referente à imagem de referência i para uma trajetória k. Nessa equação,

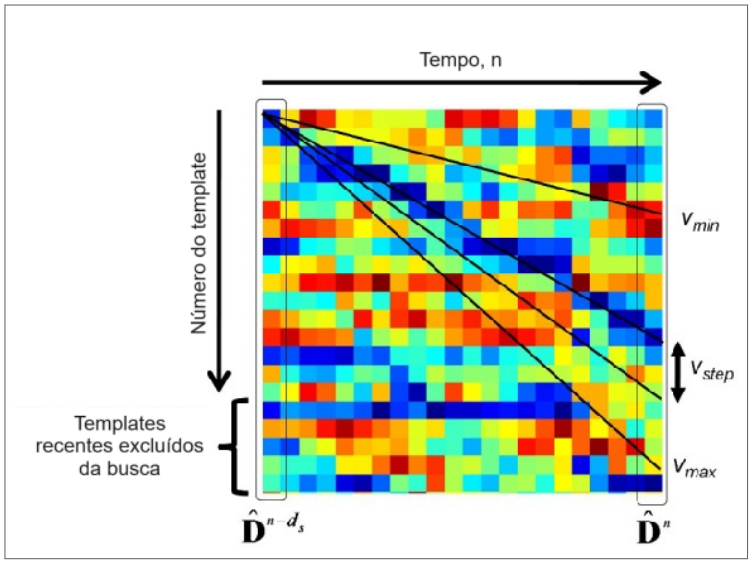
$$k = \lfloor s + V(d_s - j + n) \rfloor, \tag{2.8}$$

em que k é um índice para cada imagem de referência em função de cada índice dos *frames* pertencentes à sequência, assim como s é o valor inicial do índice da imagem de referência, pela qual diferentes trajetórias com diferentes velocidades têm seu início, *e.g.* no caso da Figura 5 $s = 0$, pois nesse exemplo as trajetórias (indicadas pelas linhas pretas) partem da primeira imagem de referência. Além disso, para criação dessas diferentes trajetórias, V determina a velocidade com que as comparações serão realizadas, *e.g.*, se $V = 0$ todas as imagens da sequência serão comparadas com a imagem de referência inicial s. Na Figura 5, V varia de V_{min} até $V_{máx}$ com passo V_{step}. Diferentes valores de V são coletados para cada valor de s e os valores de s são alterados, conforme a imagem de referência a ser comparada com a sequência correspondente ao *frame* atual de busca até que todos os *scores* $S_{i,k}$ tenham sido calculados. Posteriormente, o valor mínimo de do vetor $\boldsymbol{S}_i$, resultante dos cálculos das diferentes trajetórias, é encontrado para determinar o *score* da imagem de referência de índice i.

Por fim, cada imagem de referência terá um *score* e para determinar se a imagem é realmente similar a imagem atual, uma janela de largura R (determinado pelo usuário) irá separar as R imagens mais próximas da imagem de referência de *score* mínimo das imagens mais distantes. Sendo assim, um valor mínimo de *score* das imagens distantes também é recolhido e, caso a razão entre o *score* da imagem escolhida dentro da janela seja pelo menos μ vezes menor que o da fora da janela, essa imagem é considerada correspondente à imagem de busca. Nesse caso, μ é um parâmetro determinado pelo usuário, sendo sua variação determinante da revocação (subseção 2.3), visto que, quão maior for esse parâmetro, mais difícil será de haver

uma imagem de referência considerada como sendo correspondente do *frame* de busca. A implementação oficial dessa lógica está disponível neste site (acessado pela última vez em 15/03/2024).

Figura 5 – Busca por trajetórias de melhor score para a primeira imagem de referência. Na imagem, as diferentes linhas indicam as diferentes trajetórias sendo calculadas, as quais, nesse caso, partem da imagem de referência de índice 1.



Fonte: Autor, adaptada de MILFORD; WYETH (2012)

Em anos conseguintes, técnicas baseadas no SeqSLAM foram propostas, como o caso do SMART citesmart, que introduziu dados de odometria para melhorar o desempenho do SeqSLAM aumentando a invariância a mudanças na iluminação e na velocidade do veículo, e demonstrando alguma capacidade para lidar com *aliasing* ambiental.

2.1.2 Técnicas de VPR Artesanais

Antes do surgimento de descritores treináveis, especialmente antes do amplo uso de modelos de aprendizado profundo, uma abordagem muito comum envolvia o uso de descritores artesanais que não eram modelados especificamente para a resolução de tarefas de VPR. Esses descritores podem ser divididos em descritores globais e locais.

Scale-Invariant Feature Transform (SIFT) (LOWE, 2004) foi um dos primeiros descritores que possui processamento que busca ser invariante às escala, rotação e iluminação. SIFT foi utilizado em sistemas de VPR para aplicações de mapeamento topológico (ANDREASSON; DUCKETT, 2004) e técnicas de fechamento de laço (PRADEEP; MEDIONI; WEILAND, 2009). Já o *Speeded Up Robust Features* (SURF) (BAY; TUYTELAARS; GOOL, 2006) foi projetado para

ser mais rápido que o SIFT e está presente em sistemas de SLAM populares tal como o FABMAP(GLOVER et al., 2012), um SLAM topológico que emprega abordagem probabilística recebidos do SURF, o descritor utilizado para extração de características. Por sua vez, o *Binary Robust Independent Elementary Features* (BRIEF) (CALONDER et al., 2010) é um descritor que, para cada ponto-chave na imagem, de maneira geral, produz uma saída composta por 128, 256 ou 512 bits. Nela, cada bit resulta da comparação de intensidade para cada par de pontos considerados, em um tamanho de 48x48 ao redor de um dado ponto-chave. Por ser um descritor binário, é mais rápido que os descritores anteriores, porém, não é tão robusto quanto a variações de escala e rotação. BRIEF pode ser encontrado em sistemas de fechamento de laço como em GÁLVEZ-LÓPEZ; TARDOS (2012a) e SÜNDERHAUF; PROTZEL (2011). *Oriented FAST and Rotated BRIEF* (ORB) (RUBLEE et al., 2011) é uma variação do BRIEF mais robusta a variações de escala e rotação (GÁLVEZ-LÓPEZ; TARDOS, 2012a), sendo utilizado dentro do sistema usado para a tarefa de fechamento de laço no ORB-SLAM (MUR-ARTAL; MONTIEL; TARDOS, 2015). Nesse caso, o sistema de VPR utilizado foi o DBoW2 (GÁLVEZ-LÓPEZ; TARDOS, 2012b), o qual cria uma *Bag of Words* (BOW) a partir do conjunto de imagens coletadas, com o objetivo de se acessar os dados mais rapidamente a posteriori.

Os descritores locais, anteriormente definidos, possuem como restrição a necessidade haver correspondência entre os pontos chave a serem comparados, o que não acontece com descritores globais. Nesse contexto, BOW é um descritor global que cria uma "BOW visuais" a partir de um conjunto de imagens, nas quais as ditas palavras visuais se constituem em características presentes nesse conjunto de dados. Assim, é possível criar um histograma a partir de qualquer imagem, o qual indica a quantidade de vezes em que cada palavra visual presente na sacola aparece em uma imagem considerada. Além disso, o *Vectors of Locally Aggregated Descriptors* (VLAD)(LOWRY; ANDREASSON, 2018) é construído criando k diferentes palavras visuais a partir de um conjunto de imagens, descrevendo a imagem a partir da soma da diferença entre cada *cluster* com elementos das descrições locais da imagem (no artigo original o descritor usado foi o SIFT) mais próximos do respectivo cluster, sendo o vetor resultante normalizado no fim. Por outro lado, o *Histogram of Oriented Gradients* (HOG) divide a imagem em células, calculando gradientes nessas células, as quais são agrupadas em blocos, sendo os mesmos usados para formar um histograma de orientações. Assim, a representação da imagem se dá por um vetor o qual é construído pelo concatenamento dos histogramas de todas os blocos.

2.1.3 Técnicas de VPR baseadas em Aprendizado Profundo

Redes neurais convolucionais são redes neurais especializadas no processamento de imagens, aplicando operações de convolução, de forma a aprender características cada vez mais complexas à medida que os dados passam por essas camadas. Figura 6 ilustra um exemplo de rede neural convolucional.

O primeiro trabalho utilizando *Convolutional Neural Network* (CNN) para VPR foi apresentado em CHEN et al. (2014). Neste modelo, eles usaram uma rede pré-treinada chamada Overfeat no conjunto de dado ImageNet, composto por 1,2 milhões de imagens e 1000 classes. Essa rede é composta por 5 camadas convolucionais e 3 camadas completamente conectadas. Além disso, ele cria uma matriz de confusão similar a do SeqSLAM (Figura 4), para assim ele conseguir aplicar um filtro que elimine falsos positivos, posteriormente implementando uma versão mais sofisticada da técnica usada pelo SeqSLAM. ZHANG; WANG; SU (2021) apresenta trabalhos representativos em VPR baseados em CNN, apresentados em ordem cronológica.

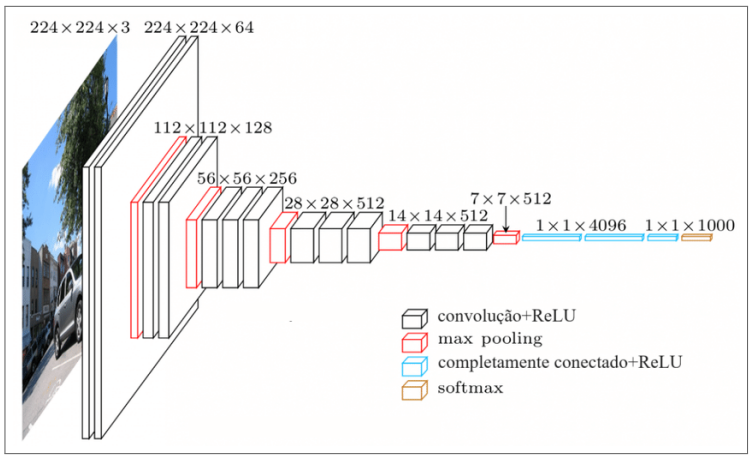
Posteriormente, redes cada vez mais profundas foram sendo desenvolvidas. Um exemplo disso é a rede VGG(SIMONYAN; ZISSERMAN, 2014), a qual possui 13 camadas convolucionais, sendo utilizada em trabalhos na área desde 2016.

CHEN et al. (2017) desenvolveram um conjunto de dados em larga escala (contendo 2,5 milhões de imagens), denominado SPED, especialmente para o reconhecimento de lugares. Com base nisso, eles ajustaram finamente uma rede, chamada HybridNet, essa rede pós processada foi futuramente utilizada no trabalho de HAUSLER; JACOBSON; MILFORD (2019), posteriormente mencionado aqui e cujo modelo foi utilizado como base de comparação para o modelo desta dissertação.

Outra rede importante é o NetVLAD(ARANDJELOVIC et al., 2016), treinada ponta a ponta diretamente para tarefa de VPR, sendo seu principal componente uma nova camada chamada VLAD, devido a sua inspiração no descritor usado para VPR. Além disso, ela possui um procedimento de treinamento baseado em uma perda de classificação supervisionada de forma fraca, para aprender os parâmetros da arquitetura de forma completa a partir de imagens que representam os mesmos lugares ao longo do tempo, baixadas do Google Street View Time Machine.

MPF é uma abordagem para VPR que combina vários sensores em um único fluxo de imagens visuais, utilizando métodos de processamento de imagem dinâmicos e um esquema automático de ponderação. Ao invés de depender exclusivamente de um único método de

Figura 6 – Uma arquitetura CNN para extração de características profundas. Cada um dos blocos brancos representa um filtro convolucional, enquanto que os blocos azuis são camadas completamente conectadas, combinadas com uma camada *softmax* na saída.



Fonte: Autor, adaptação da imagem disponível neste site (acessado pela última vez em 20/03/2024)

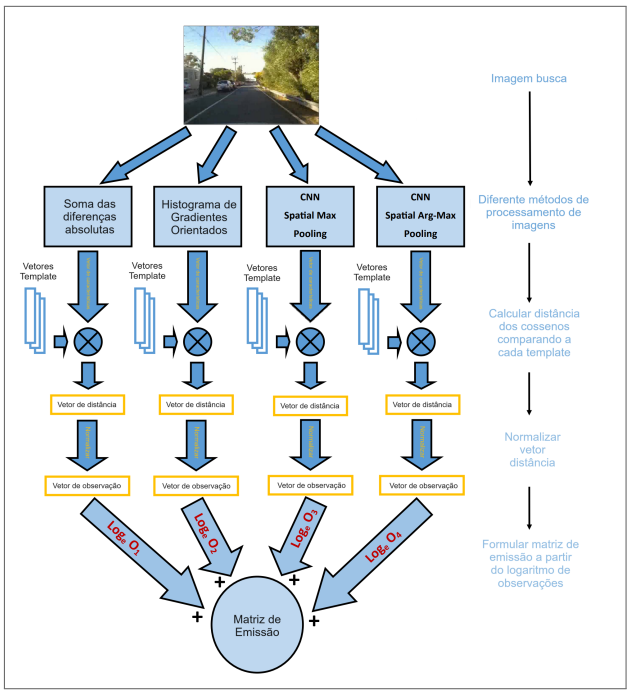
processamento de imagem, o MPF requer que pelo menos um método apresente bom desempenho em um ambiente específico. Essa abordagem dinâmica permite reduzir as latências de localização ao analisar métricas de qualidade de reconhecimento. Um esquema do MPF é ilustrado na Figura 7.

Ao receber a imagem, ele extrai características por meio de 4 processos diferentes, o SAD, como no SeqSLAM, o HOG, um descritor artesanal global, o HybridNet e o mesmo Hybridnet utilizando um método de GARG; SUENDERHAUF; MILFORD (2018). Para o SAD as correspondências são calculadas como no SeqSLAM, já para o HOG e HybridNet, as correspondências são calculadas pela distância dos cossenos, enquanto que para a alteração do HybridNet as correspondências são calculadas pela distância euclidiana. Assim, cada vetor de distância e normalizado e tem seu logarítmocomputado, resultando na matriz de observação para cada descritor. Uma votação ocorre entre as quatro matrizes de observação, e as escolhidas são somadas resultando na matriz de emissão.

A partir da matriz de emissão, o MPF segue conforme a teoria da Cadeia Escondida de Markov ou *Hidden Markov Model* (HMM), com o processo seguindo conforme o Algoritmo 1, em que a matriz $\boldsymbol{T}$ é estimada, a matriz $\boldsymbol{D}$ é a matriz de diferenças, $\boldsymbol{e}_i$ é um vetor coluna da matriz de emissão, τ é o tamanho da sequência, N é o número de imagens de referência e S é a sequência otimizada.

Nesse pseudocódigo, as iterações são realizadas para cada índice da sequência, sendo nelas atualizadas a matriz $\boldsymbol{D}$, considerando os pesos de transição e emissão e $\boldsymbol{H}$, que armazena os índices de referência k, que obtiveram o maior valor de pesos. Vale a pena salientar que

Figura 7 – Ilustração esquemática do modelo MPF, em que a imagem de entrada tem suas características extraídas e os diferentes processamentos desses vetores de características extraídos da imagem são ilustrados até o cálculo final da matriz de emissão.



Fonte: Autor, adaptada de MUR-ARTAL; MONTIEL; TARDOS (2015)

ao contrário do HMM, aqui são considerados valores de peso e não de probabilidades, mas o conceito é exatamente o mesmo. Por fim, $\boldsymbol{s}(\tau)$ armazena o último índice da sequência de maior valor e posteriormente todos os índices de maior valor anteriormente, sendo o vetor $\boldsymbol{s}$ a saída desse algoritmo.

Algoritmo 1 HMM

Inicializar $\boldsymbol{D}$ para primeira emissão
for $i = 2$ to τ **do**
    $\boldsymbol{D}[k,i] \leftarrow \max_{1 \leq k \leq N} (\boldsymbol{D}[k,i-1] + \boldsymbol{T}) + \boldsymbol{e_i}$
    $\boldsymbol{H}[k,i] \leftarrow \arg\max_{1 \leq k \leq N} (D[k,i-1] + T)$
end for
$\boldsymbol{s}(\tau) \leftarrow \arg\max_{1 \leq k \leq N} \boldsymbol{D}[k,\tau]$
for $k = \tau, \tau-1, \ldots, 1$ **do**
    $\boldsymbol{s}(k-1) \leftarrow \boldsymbol{H}(\boldsymbol{s}(k), k)$
end for

2.2 MÉTRICAS DE AVALIAÇÃO PARA RECONHECIMENTO VISUAL DE LUGARES

Esta seção fala sobre as métricas usualmente utilizadas em VPR, dando uma contextualização da metodologia de avaliação utilizada para comparar desempenhos do sistema desenvolvido neste trabalho com outros sistemas já existentes.

Uma parte considerável dos trabalhos desenvolvidos em VPR utiliza precisão e evocação como métricas de avaliação. Nesse contexto, cada comparação entre imagens de referência e imagens de busca realizadas pelo sistema produzem um índice de correspondência, sendo necessário para determinar se a correspondência é verdadeira ou não, a determinação de um limiar de confiança. Baseado na variação desse limiar, é possível calcular a precisão e o evocação, determinados, respectivamente pela Equação 2.9 e pela Equação 2.10. Nessas equações, a quantidade de correspondências que são determinadas como sendo verdadeiras sendo realmente verdadeiras é determinada como sendo TP (verdadeiro positivo, do inglês *true positive*), a quantidade de correspondências que são determinadas como sendo falsas sendo realmente falsas é determinada como sendo TN (verdadeiro negativo, do ingles *true negative*), a quantidade de correspondências que são determinadas como sendo verdadeiras sendo na realidade falsas é determinada como sendo FP (falso positivo, do inglês *false positive*) e, por fim, a quantidade de correspondências que são determinadas como sendo falsas sendo na realidade verdadeiras é determinada como sendo FN (falso negativo, do inglês *false positive*). Além disso, com a variação desse limiar é possível plotar um gráfico da precisão em função do evocação.

$$Precisão = \frac{TP}{TP + FP} \tag{2.9}$$

$$Evocação = \frac{TP}{TP + FN} \tag{2.10}$$

Além disso, outras métricas de precisão podem ser consideradas como a métrica "Evocação@K", que indica a capacidade do sistema VPR recuperar a imagem correta dentro das top K melhores correspondências, assim como a área abaixo da curva de precisão-evocação e o escore F(GARG; FISCHER; MILFORD, 2021).

Em aplicações embarcadas, é de extremo interesse também se ter a informação de se o sistema VPR utiliza GPU ou não, consumo de RAM, tempo de execução e tamanho dos descritores. Sendo algumas dessas métricas definidoras da possibilidade de utilização ou não

do sistema em aplicações de navegação autônoma, como por exemplo o tempo de execução, havendo importantes desafios descritos na seção a seguir.

2.3 DESAFIOS E TENDÊNCIAS

No que concerne a desafios a serem enfrentados no campo de pesquisa de VPR, pode-se falar da eficiência de descritores do estado da arte, os quais possuem dimensões altas desde centenas a dezenas de milhares. Geralmente técnicas como PCA são utilizadas para a redução de dimensionalidade, porém, poderia ser considerado uma técnica de redução de dimensionalidade como em MCINNES; HEALY; MELVILLE (2018), a qual levasse em conta que no caso de VPR poderia ser considerada uma sequência de dados ou até mesmo considerado a odometria como fonte de informação adicional.

Além disso, esse campo carece de um estudo sistemático acerca da escolha da métrica de distância a ser utilizada para estabelecer a correspondência entre as imagens. Existem diversas métricas comumente usadas, como a distância euclidiana, dos cossenos, e de Hamming. Contudo, mesmo que essas métricas possam ser eficazes a depender do descritor utilizado, a distribuição das distâncias em uma aplicação costuma ser estreita, mesmo em situações em que as aparências das imagens são bastante diferentes umas das outras. Assim, seria interessante um estudo tanto em questões teóricas, quanto em questões práticas com relação ao impacto que isso implica no desempenho.

Um outro problema já explorado, mas de forma sucinta, é a aprendizagem com relação ao processo de busca por correspondências. Embora o estudo com relação ao aprendizado das características dos descritores seja amplo, ainda há poucos trabalhos que levem em conta o processo de correspondência dentro do aprendizado.

Ademais, há um *tradeoff* entre a invariância à posição (reconhecer os lugares a partir de pontos de vista diferentes) e invariância à mudança de condição do ambiente (*e.g.* iluminação e clima), o que impõe um desafio a muitos sistemas atualmente.

Garg *et al.* (GARG; VANKADARI; MILFORD, 2022) argumentam que a maioria dos métodos recentes baseados em CNNs é projetada para VPRs baseados em imagens individuais. Eles utilizam informações sequenciais como um procedimento pós-aprendizado para filtrar os escores de correspondência de imagens individuais. Embora os métodos baseados em CNN geralmente tenham um desempenho melhor do que os métodos anteriores baseados em sequências feitas manualmente, sua eficácia pode melhorar quando mais informações do que imagens individuais

são consideradas durante a fase de treinamento, reduzindo ou evitando problemas como a *aliasing* perceptual.

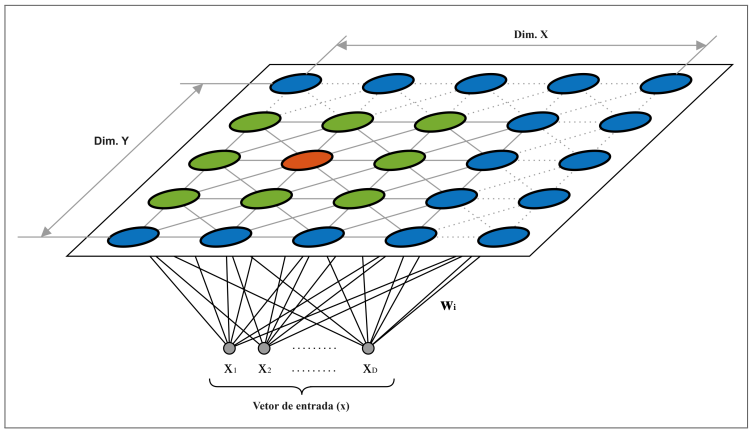
Portanto, métodos baseados em sequências, de modo geral se destacam em termos de desempenho, mas um dos principais problemas com sistemas VPR no geral, principalmente com os métodos baseados em sequências mais conhecidos, como SeqSLAM (MILFORD; WYETH, 2012) e MPF (HAUSLER; JACOBSON; MILFORD, 2019), têm sido o tempo de processamento. O esforço para calcular descritores depende linearmente dos números de imagens de referência (n) e de busca (m), mas os custos para uma comparação exaustiva entre pares crescem significativamente mais rápido. De fato, poucos trabalhos abordam esse problema (SIAM; ZHANG, 2017), e, até onde se sabe, há uma escassez de trabalhos recentes sobre esse assunto, crucial para aplicações em robótica móvel.

Sendo assim, nesta dissertação é proposto um Mapa Auto-organizável para tarefa de VPR para agrupar representações dos dados de referência fazendo com que as comparações sejam entre a representação da imagem de busca e os nodos da rede e a representação da imagem de busca e somente as imagens do nodo mais próximo, diminuindo o total de comparações necessárias. Contudo, como em um contexto de navegação autônoma novos dados surgem constantemente, é necessário que a rede tenha a capacidade de se adaptar aos novos dados ao decorrer do tempo. Dessa forma, a rede proposta é baseada em um SOM de estrutura variante no tempo. Tendo isso em vista, a próxima seção é voltada para Mapas Auto-organizáveis utilizados para tarefas de reconhecimento visual assim como de estrutura variante no tempo, justificando o caminho tomado para o desenvolvimento da rede proposta neste trabalho.

3 MAPAS AUTO-ORGANIZAVEIS ÚTEIS PARA VPR

SOM é um conjunto de modelos de rede neural de aprendizagem não-supervisionada por competição, introduzido por KOHONEN (1982). O SOM introduzido por KOHONEN é dado por um conjunto de unidades de processamento diretamente ligadas à entrada, como ilustrado na Figura 8.

Figura 8 – Ilustração do SOM, com dimensões x e y, para um vetor de entrada de dimensão D.



Fonte: ANDRADES et al. (2020)

O processo de treinamento desta rede está sintetizado no Algoritmo 2. Dado um conjunto de entrada $\boldsymbol{\Xi}$, para cada entrada $\boldsymbol{\xi}_i$ pertencente a esse conjunto dentro de uma época, busca-se encontrar o nodo mais próximo desta entrada, sendo ele denominado unidade de melhor ajuste ou *Best Matching Unit* (BMU) e essa etapa é definida como a etapa de competição (linha 6). Em seguida, é realizada a etapa de cooperação (linha 8), em que cada nodo possui um valor correspondente à função de vizinhança $h_j(n, \boldsymbol{\xi}_i)$. Esse valor é utilizado posteriormente na etapa de adaptação (linha 9), em que cada peso tem seu valor atualizado conforme a entrada fornecida. Assim, sendo o valor dessa atualização maior conforme a unidade de processamento seja mais próxima da BMU, devido à etapa de cooperação. Além disso, o usuário pode definir a taxa de aprendizagem $\eta(n)$ a qual é geralmente atribuída uma constante ou uma exponencial decrescente.

Algoritmo 2 Treinamento do Mapa Auto-Organizável (SOM)

1: **Entrada:** $\boldsymbol{\Xi}$
2: Inicializar a grade do SOM com pesos aleatórios
3: Definir taxa de aprendizado inicial η e tamanho do vizinho σ
4: **while** não atingir o critério de convergência **do**
5: **for** para cada $\boldsymbol{\xi}_i \in \boldsymbol{\Xi}$ **do**
6: Competição: encontrar o nodo mais próximo de $\boldsymbol{\xi}_i$
7: **for** para $\boldsymbol{w_j} \forall j \in 1, 2, ..., N$ **do**
8: Cooperação: $h_j(n, \boldsymbol{\xi}_i) = exp(\frac{\|\boldsymbol{w}_{bmu} - \boldsymbol{w_j}\|^2}{2\sigma^2(n)})$
9: Adaptação: $\boldsymbol{w_j}(n+1) = \boldsymbol{w_j}(n) + \eta(n) h_j(n, \boldsymbol{\xi}_i)(\boldsymbol{\varepsilon(n)} - \boldsymbol{w_j(n)})$
10: **end for**
11: **end for**
12: **end while**

3.1 SOM PARA EXTRAÇÃO DE CARACTERÍSTICAS E RECONHECIMENTO VISUAL

Esta seção é dedicada a algumas aplicações recentes de SOM para extração de características e recuperação de imagens, levando em conta a sua relevância e proximidade com o problema foco desta dissertação.

3.1.1 *Localization Aware SOM*

Location Aware Self-Organising Map (LASOM)(KIT; KONG; FU, 2014) é um sistema baseado em SOM utilizado para a tarefa de recuperação de imagens de diferentes regiões, reduzindo custo computacional e necessidade de armazenamento. A distância $H(\cdot)$ é definida como a distância de Haversine, $\boldsymbol{c}_1$ é o nodo vencedor, $\boldsymbol{c}_2$ é o segundo vencedor, $\boldsymbol{L}_{c1}$ é o local do primeiro vencedor e $\boldsymbol{L}_{c2}$ é o local do segundo vencedor e $\boldsymbol{\mathcal{N}}$ é o conjunto de nodos vizinhos em torno do vencedor. O Algoritmo 3 mostra o processo de treinamento do LASOM.

Para cada vetor de entrada $\boldsymbol{f}_t$, em que t é o índice correspondente à amostra atual, primeira ação é a busca de $\boldsymbol{c}_1$ e $\boldsymbol{c}_2$ através do cálculo da similaridade, a depender do descritor utilizado (linhas 3 e 4). Após essa determinação, se a distância entre $\boldsymbol{L}_{\boldsymbol{f}_t}$ e $\boldsymbol{L}_{c1}$ for maior do que o limiar determinado e $\boldsymbol{c}_1$ possuir um vizinho muito similar à entrada e cuja distância é inferior ao limiar, esse vizinho passa a ser $\boldsymbol{c}_1$ e $\boldsymbol{c}_1$ se torna $\boldsymbol{c}_2$ (linha 6 à linha 9). Para esse vizinho ser considerado similar o suficiente, é calculada a média de todas as distâncias encontradas e esse vizinho deve estar dentro de um desvio padrão de magnitude 3 ao redor dessa média.

Se houver um $\boldsymbol{c}_1$ cuja distância à $\boldsymbol{f}_t$ respeite o limiar, então $\boldsymbol{c}_1$ é atualizado de forma a reduzir $\|\boldsymbol{f}_t - \boldsymbol{W}_{c1}\|$, enquanto que o contador de vitórias de $\boldsymbol{c}_1$ ($\tau(\boldsymbol{c}_1)$) é incrementado

por 1 (linhas 10 e 12). Neste caso, se a conexão entre $\boldsymbol{c}_1$ e $\boldsymbol{c}_2$ não existia, $\boldsymbol{c}_1$ é conectado à $\boldsymbol{c}_2$ e a idade da conexão entre $\boldsymbol{c}_1$ e $\boldsymbol{c}_2$ (AgeMatrix(n,c1)) é inicializada em 0 (linhas 13 até 15), assim como para todos vizinhos de $\boldsymbol{c}_1$ há um incremento dessa idade de conexão por 1. Caso esse limiar de distância não seja respeitado, um novo nodo com as informações de $\boldsymbol{f}_t$ é adicionado à rede (linhas 21 até 23). Em seguida, todas as arestas de idade maior que um parâmetro definido à priori Γ. Por fim, se o dado de treinamento atual é múltiplo de um parâmetro predefinido λ, nodos são fundidos por meio da função "FundirClusters($\cdot$)", como descrito a seguir.

Algoritmo 3 Algoritmo de Treinamento LASOM

1: Dados: (Lf, f) : Conjunto de Amostras de Treinamento
2: **while** houver mais amostras de treinamento **do**
3: $\quad \boldsymbol{c}_1 \leftarrow \text{argmin}_{c \in \mathcal{G}} \|\boldsymbol{f}_t - \boldsymbol{W}_c\|$;
4: $\quad \boldsymbol{c}_2 \leftarrow \text{argmin}_{c \in \mathcal{G} \setminus \boldsymbol{c}_1} \|\boldsymbol{f}_t - \boldsymbol{W}_c\|$;
5: $\quad \boldsymbol{\mathcal{N}} =$ Conjunto dos vizinhos de $\boldsymbol{c}_1$;
6: $\quad$ **if** $\exists \boldsymbol{n} \in \boldsymbol{\mathcal{N}}$ tal que $\|\boldsymbol{f}_t - \boldsymbol{W}_{\boldsymbol{n}}\|$ está dentro de um desvio de magnitude 3 **E** $H(\boldsymbol{L}_{\boldsymbol{f}_t}, \boldsymbol{L}_{\boldsymbol{n}}) < \delta$ **then**
7: $\quad\quad \boldsymbol{c}_2 \leftarrow \boldsymbol{c}_1$;
8: $\quad\quad \boldsymbol{c}_1 \leftarrow \boldsymbol{n}$;
9: $\quad$ **end if**
10: $\quad$ **if** $H(\boldsymbol{L}_{\boldsymbol{f}_t}, \boldsymbol{L}_{\boldsymbol{c}_1}) < \delta$ **then**
11: $\quad\quad \boldsymbol{W}_{\boldsymbol{c}_1} = \boldsymbol{W}_{\boldsymbol{c}_1} + \frac{\epsilon}{\tau(c1)} \times (\boldsymbol{f} - \boldsymbol{W}_i)$;
12: $\quad\quad$ incrementar $\tau(\boldsymbol{c}_1)$ por 1;
13: $\quad\quad$ **if** $(\boldsymbol{c}_1, \boldsymbol{c}_2) \notin \boldsymbol{\mathcal{E}}$ **then**
14: $\quad\quad\quad$ adicionar uma aresta $(c1, c2)$ a $\boldsymbol{\mathcal{E}}$;
15: $\quad\quad\quad$ AgeMatrix$(c1, c2) = 0$;
16: $\quad\quad\quad$ **for all** $n \in N \setminus c2$ **do**
17: $\quad\quad\quad\quad$ incrementar AgeMatrix$(n, c1)$ por 1;
18: $\quad\quad\quad$ **end for**
19: $\quad\quad$ **end if**
20: $\quad$ **else**
21: $\quad\quad \boldsymbol{C} = (\boldsymbol{L}_{\boldsymbol{f}_t}, \boldsymbol{f}_t)$;
22: $\quad\quad$ Adicionar o nodo $\boldsymbol{C}$ a $\boldsymbol{\mathcal{V}}$;
23: $\quad\quad$ Adicionar uma aresta $(\boldsymbol{C}, \boldsymbol{c}_1)$ a $\boldsymbol{\mathcal{E}}$;
24: $\quad$ **end if**
25: $\quad$ **for all** $(e_1, e_2) \in \boldsymbol{\mathcal{E}}$ **do**
26: $\quad\quad$ **if** AgeMatrix$((e_1, e_2)) > \Gamma$ **then**
27: $\quad\quad\quad$ remover (e_1, e_2) de $\boldsymbol{\mathcal{E}}$;
28: $\quad\quad$ **end if**
29: $\quad$ **end for**
30: $\quad$ **if** o teste atual é um múltiplo de λ **then**
31: $\quad\quad \boldsymbol{\mathcal{G}} \leftarrow$ FundirClusters$(\boldsymbol{\mathcal{G}}, \text{AgeMatrix}, \tau)$;
32: $\quad$ **end if**
33: **end while**

O Algoritmo 4 busca agrupar nodos similares e criar novos pontos de referência para regiões que não atendem a um critério de similaridade visual. Ele ajusta as conexões entre os vetores para se originarem desses novos pontos, reduzindo redundâncias e simplificando a estrutura. Sendo assim, há um laço percorrendo cada vértice $\boldsymbol{v} \in \boldsymbol{\mathcal{V}}$ representante de cada nodo pertencente ao estado atual da rede (linha 4). Para cada um desses vértices é calculada a distância entre os pesos do nodo atualmente percorrido pelo laço e todos os outros nodos da rede, sendo o conjunto dessas distâncias e os vértices correspondentes adicionados ao conjunto $\boldsymbol{\mathcal{P}}$, ordenados decrescentemente (linhas 5 até 9). Com isso, todos os dados de $\boldsymbol{\mathcal{P}}$ são acessados de forma a decidir se a distância geográfica entre o vértice atual e cada um dos outros vértices é menor do que $0,5 * \delta$, em que δ é um parâmetro predefinido (linhas 10 e 11). Caso isso ocorra, $\boldsymbol{v'}$ é adicionado ao conjunto $\boldsymbol{\mathcal{M}}$, caso contrário o laço é interrompido. Por fim, caso $\boldsymbol{\mathcal{M}}$, não esteja vazio, *i.e.*, há vértices geograficamente muito próximos um do outro, sendo assim necessária a realização de um processo de fusão. Por conseguinte, $\boldsymbol{n}$ é um novo vetor que substituirá todos os outros nodos que foram adicionados a $\boldsymbol{\mathcal{M}}$, sendo $\boldsymbol{w_n}$ a média dos pesos desses nodos e $\boldsymbol{l_n}$ a média de suas localizações geográficas, sendo também as arestas que eram conectadas a todos esses nodos, reconectadas ao nodo $\boldsymbol{n}$ e as arestas (linhas 17 até 23).

Para atribuir uma localização a uma imagem sem rótulo, primeiramente se realiza uma busca por $\boldsymbol{c}_1$, como mostrado na linha 3 do Algoritmo 5. A partir disso, a localização do vetor de características da imagem de busca pode estar em qualquer lugar na área de influência de $\boldsymbol{c}_1$. Para fornecer estimativas melhores, o vetor de característica da imagem de busca é novamente comparada a $\boldsymbol{c}_1$ e todos os seus nodos vizinhos. Esses erros são então convertidos para uma distribuição gaussiana padrão, subtraindo cada erro pela média e dividindo pelo desvio padrão, como na linha 5. Após deslocar os valores de erro de forma que o valor mínimo seja 0, uma função exponencial é usada para calcular os pesos para cada nodo (linha 6). Os pesos são normalizados e a localização é calculada como uma média ponderada das localizações dos vetores de peso dos nodos vizinhos a $\boldsymbol{c}_1$, assim como o próprio $\boldsymbol{c}_1$ (linhas 8 até 12). Segundo o artigo, experimentos mostraram que o uso de uma exponencial com uma base de $1,5$ resultou em desempenho superior para o cálculo da linha 6.

Algoritmo 4 Fundir Clusters

Resultado: $\boldsymbol{\mathcal{G}} = (\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{V}})$, AgeMatrix: Idade das arestas, τ: # de vezes que $\boldsymbol{v} \in \boldsymbol{\mathcal{V}}$ foi um vetor vencedor
Dados: $\boldsymbol{\mathcal{G}} = (\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{V}})$, AgeMatrix, τ
Inicializar $\boldsymbol{\mathcal{P}} = \{\}$, $\boldsymbol{\mathcal{M}} = \{\}$
for all $\boldsymbol{v} \in \boldsymbol{\mathcal{V}}$ **do**
    for all $\boldsymbol{v'} \in \boldsymbol{\mathcal{V}} \setminus \{\boldsymbol{v}\}$ **do**
        distância visual $= \|\boldsymbol{W_v} - \boldsymbol{W_{v'}}\|$
        Adicionar $(\boldsymbol{v'}$, distância visual$)$ a $\boldsymbol{\mathcal{P}}$
    end for
    Ordenar $\boldsymbol{\mathcal{P}}$ por distância visual
    for all $(\boldsymbol{v'}$, distância visual$) \in \boldsymbol{\mathcal{P}}$ **do**
        if $H(\boldsymbol{L_v}, \boldsymbol{L_{v'}}) < 0,5 \times \delta$ **then**
            adicionar $\boldsymbol{v'}$ a $\boldsymbol{\mathcal{M}}$
        else
            interromper
        end if
    end for
    if $\boldsymbol{\mathcal{M}}$ não está vazio **then**
        Faça $\boldsymbol{n}$ ser um nodo cujo peso é a média de todos $\boldsymbol{w'_v}$ e seu local correspondente é a média de todos vetores $\boldsymbol{l'_v}$;
        Adicionar $\boldsymbol{n}$ a $\boldsymbol{\mathcal{V}}$;
        Alterar todas as arestas originárias dos vértices pertencentes ao conjunto $\boldsymbol{\mathcal{M}}$ para se originarem de $\boldsymbol{n}$;
        Redefinir contadores de AgeMatrix para essas arestas;
        $\boldsymbol{\mathcal{E}} = \boldsymbol{\mathcal{E}} \setminus \boldsymbol{M}$;
    end if
end for

Algoritmo 5 Consulta LASOM

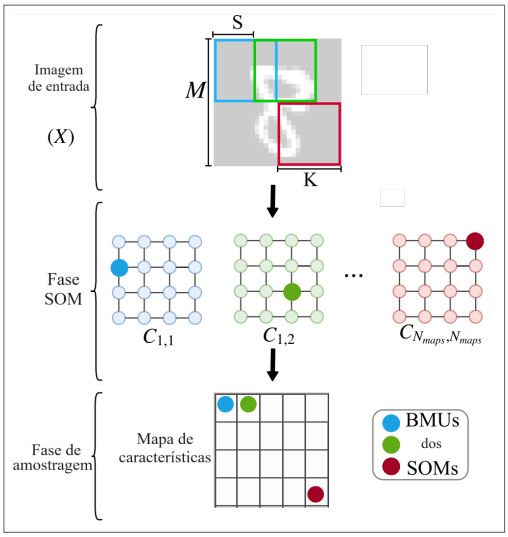
Resultado: Localização
Dados: $\boldsymbol{\mathcal{G}} = (\boldsymbol{\mathcal{E}}$, $\boldsymbol{f}$: Vetor de características da imagem de consulta, $\boldsymbol{\mathcal{N}}$
$c1 \leftarrow \text{argmin}_{\boldsymbol{c} \in \boldsymbol{\mathcal{G}}} \|\boldsymbol{f} - \boldsymbol{W_c}\|$;
Faça com que erros $= \{\|\boldsymbol{f} - \boldsymbol{W_v}\| : \forall \boldsymbol{v} \in \boldsymbol{\mathcal{N}} \cup \{\boldsymbol{c}_1\}\}$
$\boldsymbol{erros}_i := \frac{(\boldsymbol{erros}_i - \boldsymbol{média}(\boldsymbol{erros}))}{\sigma(\boldsymbol{erros})} \forall i \in \{1, ..., |\boldsymbol{\mathcal{N}} \cup \{\boldsymbol{c}_1\}|\}$;
Faça com que $\boldsymbol{w} = 1,5^{-(\boldsymbol{erros}_i + \min(\boldsymbol{erros}))\forall i \in \{1,...,|\boldsymbol{\mathcal{N}} \cup \{\boldsymbol{c}_1\}|\}}$;
$\boldsymbol{w} \leftarrow \boldsymbol{w}/\Sigma(\boldsymbol{w})$
$\boldsymbol{l} = \boldsymbol{0}$;
for all $\boldsymbol{v} \in \boldsymbol{\mathcal{N}} \cup \{\boldsymbol{c}_1\}$ **do**
    $\boldsymbol{l} \leftarrow \boldsymbol{l} + w_v \cdot L_v$;
end for

3.1.2 *Deep Self-Organising Map*

Deep Self-Organising Map (DSOM)(LIU; WANG; GONG, 2015) é uma junção do conceito de SOM com CNN com a finalidade de desenvolvimento de uma rede de treinamento não-supervisionado. Sua arquitetura é ilustrada na Figura 9. A ideia é que cada *patch* da imagem de entrada passa por um SOM e a saída é a BMU, definida de forma similar ao Algoritmo 2, o qual vai formar um componente de uma nova camada na "fase de amostragem".

Como pode ser visto nessa figura, diferentes regiões da imagem são selecionadas por meio de uma janela que percorre através da imagem com passo S, sendo a janela da imagem de dimensão KxK e a imagem de dimensão M. Cada uma dessas janelas serve de entrada para um SOM $C_{i,j}$, em que $i = 1, ..., N_{maps}$ e $j = 1, ..., N_{maps}$. Como mencionado anteriormente cada um desses SOMs atua como mostrado no Algoritmo 2, sendo a função de vizinhança $h(n, \epsilon_i) = 0,49(1 - \frac{e}{epochs}) + 0,01$. Na "fase de amostragem", as BMUs de cada SOM são coletadas para se criar um novo mapa de características, o qual é submetido ao mesmo processo da imagem de entrada, passando por uma nova camada de SOMs e repetindo o processo, conforme a estrutura da rede desejada.

Figura 9 – Arquitetura DSOM com uma camada escondida.



Fonte: Autor, adaptado de LIU; WANG; GONG (2015).

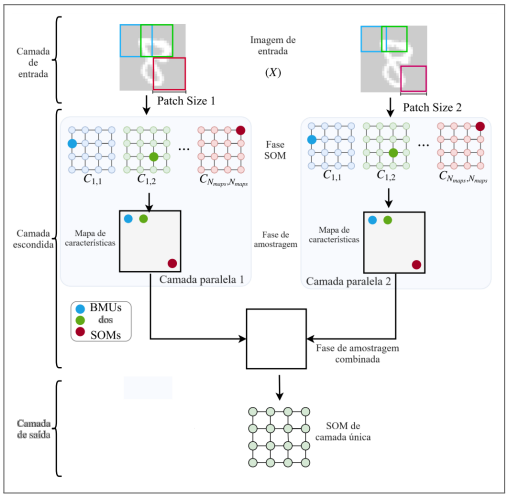
O treinamento desta rede é supervisionado. Ainda mais, cada nodo na camada de saída tem um contador de vitórias, assim, cada vez que uma entrada de treinamento rotulada é utilizada como entrada, o nodo vencedor da camada de saída tem seu contador incrementado de 1. No caso de empate, escolhe-se o vencedor aleatoriamente entre aqueles que se igualam.

Portanto, cada nodo de saída é associado a uma classe e, assim, é possível associar cada imagem de entrada a uma classe através das ativações da camada de saída da rede.

3.1.3 *Extended Deep Self-Organising Map*

Extended Deep Self-Organising Map (E-DSOM)(WICKRAMASINGHE; AMARASINGHE; MANIC, 2019) estende o conceito do DSOM alterando a etapa de amostragem e introduzindo uma amostragem combinada, como ilustrado na Figura 10. Essa mudança se deve ao fato de ao invés de um único tamanho de *patch*, aqui pode-se definir múltiplos tamanhos de *patch* os quais serão computados em paralelo, sob a motivação de que abordagens de *patches* em múltiplas escalas ajudam a melhorar a precisão da classificação ao extrair informações complementares. Cada um desses mapas de características é convertido em *arrays* unidimensionais e concatenados em um único vetor. Por conseguinte, o vetor resultante é reestruturado em um gradil 2-D para ser a saída da respectiva camada e entrada para a próxima camada ou saída da rede. Um detalhe importante é que caso esse vetor unidimensional não possa ser diretamente reestruturado para o gradil, uma operação de *zero-padding* é utilizada para que essa reestruturação seja possível. Contudo, o resto do procedimento é aquele do DSOM, portanto, o acréscimo de camadas e suas consequências caracterizaram a novidade desse trabalho.

Figura 10 – Arquitetura E-DSOM com uma camada escondida.



Fonte: Autor, adaptado de WICKRAMASINGHE; AMARASINGHE; MANIC (2019).

3.1.4 Relação dos Modelos Mencionados com VPR

Embora modelos de rede profunda possam ser utilizados para lidar com o VPR como um problema de classificação, há graves problemas práticos quanto a isso, visto que, quando ocorrerem mudanças de trajetória a rede precisaria ser redesenhada devido ao surgimento de novos lugares, assim como a rede precisaria ser retreinada em casos de mudanças impactantes no cenário do ambiente.

Em contrapartida, o LASOM possui relações próximas ao modelo proposto nesta dissertação por ser uma rede de topologia variante no tempo, podendo se adaptar ao aumento e mudança no conjunto de dados de entrada. Contudo, ele não foi desenvolvido no contexto de navegação autônoma, de forma a não tirar proveito da continuidade dos dados para melhora de seu desempenho, tratando cada imagem de maneira individual. Além disso, o LASOM é baseado no *Growing Neural Gas* (GNG), mas, como é explicado mais adiante, o mapa auto-organizável de topologia variante no tempo *Local Adaptive Receptive Field Self-organising Map* (LARFSOM) possui características de maior interesse no caso do trabalho desenvolvido nesta dissertação.

3.2 SOM COM ESTRUTURA VARIANTE NO TEMPO

Redes SOM com estrutura variante no tempo - ou *Self-Organising Maps with Time-Varying Structure* (SOM-TVS) - são desenvolvidos para seguir distribuições não estacionárias. Sendo assim, ele possui importantes características como a determinação do número de nodos durante o processo de treinamento em consonância com a aprendizagem da topologia do mapa. Suas vantagens incluem a possibilidade de usar medidas de erro dependentes do problema para determinar onde novas unidades são inseridas (inserção onde necessário), assim como de interromper o processo de auto-organização ou continuar um processo previamente interrompido: devido aos parâmetros constantes, não existem diferentes fases na auto-organização. Além disso, menos parâmetros para definir: o tamanho da rede não precisa ser definido antecipadamente, mas pode ser definido indiretamente ao fornecer um critério de desempenho que deve ser atendido. Para cada parâmetro, apenas seu valor precisa ser definido, não sendo necessário o valor inicial, o valor final, nem sua função ao longo do tempo, como em muitas outras abordagens(FRITZKE, 1996).

Suas aplicações abrangem diversas áreas como categorização de dados, visualização de

dados de alta dimensionalidade, detecção de novidade, navegação robótica, processamento de imagens e reconstrução de imagens. Sendo, portanto, algumas das aplicações mais similares ao modelo proposto nesta dissertação descritas na próxima subseção. Nesta subseção, uma breve descrição em ordem cronológica de importantes redes auto-organizáveis variantes no tempo é realizada.

3.2.1 *Growing Cell Structures*

O *Growing Cell Structures* (GCS) (FRITZKE, 1994a) tem como objetivo criar um mapeamento de um conjunto de vetores de entrada ($\boldsymbol{\Xi}$) para o conjunto de pesos pertencentes ao mapa (A) com uma topologia de k dimensões. Ou seja, se $k = 1$, a topologia será linear; com $k = 2$, a topologia será formada por triângulos; com $k = 3$, serão tetraedros; e para valores maiores que 3, serão hiper-tetraedros. O Algoritmo 6 ilustra o processo padrão de treinamento dessa rede, conforme explicado a seguir.

Dado um conjunto $\boldsymbol{\Xi}$ de entrada, uma entrada aleatória $\boldsymbol{\xi}$ é escolhida. A entrada escolhida é então comparada com cada um dos nodos da rede, sendo escolhido como nodo vencedor o nodo mais próximo à entrada. Sabendo qual o nodo vencedor s, esse nodo assim como seus vizinhos são atualizados conforme às suas distâncias com relação à entrada. Cada uma dessas distâncias é multiplicada por uma constante definida pelo usuário, sendo a constante ϵ_b para o nodo vencedor diferente da ϵ_c referente aos nodos vizinhos, com c sendo o índice referente aos nodos. Após ajustes, o contador τ_s do vencedor é incrementado em 1 e o vetor de contadores dos nodos é decrementado por uma fração α. Além disso, realizar a inserção de um novo nodo, define-se o sinal de frequência relativa h_c de cada nodo, selecionando o nodo q, cujo h_q é maior que todos os sinais referentes aos outros nodos. Assim, um nodo r é inserido na rede por meio de uma ponderação entre o nodo c e o nodo de maior distância f. Por fim, cada τ_c é atualizado levando em consideração o volume do campo de Voronoi do nodo c pertencente a vizinhança do nodo inserido, assim como τ_r é definido como o negativo do somatório de todos Δ_{τ_c} em torno da vizinhança de r.

Algoritmo 6 Algoritmo GCS

1: **Entrada:** $\boldsymbol{\Xi}$
2: Inicialize os parâmetros: $\|A\| = k + 1$, $\boldsymbol{\tau_c} = \mathbf{0}$
3: **while** não atingir o critério de convergência **do**
4: Escolha aleatoriamente uma entrada $\boldsymbol{\xi}$ do conjunto de dados $\boldsymbol{\Xi}$
5: Encontre o vencedor $\boldsymbol{w}_s$ para $\boldsymbol{\xi}$
6: Ajuste o peso do vencedor por $\boldsymbol{\Delta_{w_s}} = \epsilon_b(\xi - \boldsymbol{w}_s)$
7: Ajuste os pesos dos vizinhos por $\boldsymbol{\Delta_{w_c}} = \epsilon_n(\xi - \boldsymbol{w}_c)$
8: Incremente 1 ao contador τ_s
9: Decremente o vetor de contadores $\boldsymbol{\tau}$ por $\boldsymbol{\Delta_{tau}} = -\alpha\boldsymbol{\tau}$
10: computar $h_c = \frac{\tau_c}{\sum_{j \in A} \tau_j}$
11: Definir h_q em que $h_q \geq h_c \forall c \in A$
12: Definir w_f em que $\|\|\boldsymbol{w}_f - \boldsymbol{w}_q\|\| \geq \|\|\boldsymbol{w}_c - \boldsymbol{w}_q\|\| \forall c \in N_q$
13: Inserir $\boldsymbol{w}_r = 0.5(\boldsymbol{w}_q + \boldsymbol{w}_f)$ na rede
14: Atualize $\boldsymbol{\tau}_c$ por $\boldsymbol{\Delta_{\tau_c}} = \frac{\|F_c^{new}\| - \|F_c^{old}\|}{\|F_c^{old}\|}\boldsymbol{\tau}_c \forall c \in N_r$
15: Defina $\tau_r = -\sum_{c \in N_r} \Delta_{\tau_c}$
16: **end while**

3.2.2 *Growing Neural Gas*

GNG, desenvolvido por FRITZKE (1994b), é um mapa auto-organizável capaz de crescer e se adaptar aos dados com o tempo. Essa capacidade de adaptação lhe permite maior flexibilidade, aumentando a robustez, isto é, ter capacidade para responder a diferentes conjuntos de dados, sem precisar antever o tamanho da rede. O crescimento é periódico, ocorrendo a cada λ apresentações de instâncias. Nesse caso, o processo de aprendizagem é mostrado no Algoritmo 7.

Dentro de um laço, uma amostra aleatória $\boldsymbol{x}_i$ do conjunto de dados $\boldsymbol{\Xi}$ é escolhida aleatoriamente, posteriormente sendo realizada uma busca por dois nodos vencedores s_1 e s_2 (linhas 5 e 6). Por sua vez, o nodo vencedor tem todas as idades de suas conexões incrementadas por 1 para cada vitória (linha 7). Além disso, cada nodo tem um contador local (relacionado ao erro não-supervisionado), o qual é incrementado pelo quadrado da distância entre a amostra atual e o respectivo nodo, como mostrado na linha 8. Já na linha 9, os nodos, assim como seus vizinhos, são atualizados em direção à amostra atual e ϵ_b e ϵ_n são taxas de aprendizagem para essas atualizações, em que ambos são valores entre 0 e 1 e ϵ_n é muito menor que ϵ_b. As conexões do primeiro e do segundo nodo vencedor são então atualizadas. Neste sentido, se eles já estão conectados a idade dessa conexão é zerada, caso não haja conexão, uma nova conexão é criada com idade zero (linha 10). As conexões que tiverem o seu contador de idade

maior que um limiar age_{max} são removidas da rede e, os nodos que ficarem sem conexões após essa operação, também são removidos (linha 11). Por sua vez, se o número de amostras processadas pela rede for múltiplo de um parâmetro predefinido λ, um novo nodo é inserido, sendo seu peso o ponto médio entre o nodo com maior erro acumulado e o nodo vizinho desse nodo que possui maior erro acumulado entre os vizinhos (linhas 12 até 15). Para a atualização das conexões, se houver uma conexão entre o nodo de maior erro acumulado e seu vizinho, esta conexão é removida, em seguida uma conexão é criada entre o novo nodo e o nodo de maior erro acumulado e seu vizinho. O contador local do nodo de maior erro acumulado é então decrementado, assim como o contador do seu vizinho de maior erro, pela multiplicação por um fator α e o nodo r tem seu contador local inicializado com o mesmo valor do nodo q. Ademais, todas os contadores dos outros nodos tem seu erro atualizado pela multiplicação por uma constante β.

Algoritmo 7 Algoritmo GNG

1: **Entrada:** Ξ
2: Inicialize os parâmetros: ρ_f, ε, a_T, $age = 0$, d_m, $t = 0$, e_{min}, $N = 2$
3: Inicialize os N pesos a partir de Ξ
4: **while** não atingir o critério de convergência **do**
5: Amostragem: Selecione uma amostra $\boldsymbol{\xi}$ de acordo com uma dada probabilidade $P(\boldsymbol{\xi})$ do conjunto Ξ.
6: Casamento por similaridade: Encontre o nodo, $\boldsymbol{s}_1$, com os vetores de peso mais próximo e o segundo mais próximo, $\boldsymbol{s}_2$, da amostra $\boldsymbol{\xi}$.
7: Envelhecimento: Incremente a idade de todas as conexões que partem de $\boldsymbol{s}_1$.
8: Incremento do contador local: Incremente o contador local com o quadrado da distância entre a amostra e o nodo mais próximo ($\boldsymbol{s}_1$):
$counter(s_1) = counter(s_1) + \Delta_{erro}(s_1)$, em que $\Delta_{erro}(s_1) = \|\boldsymbol{w}_{\boldsymbol{s_1}} - \boldsymbol{\xi}\|^2$
9: Atualização dos pesos: Movimente o nodo $\boldsymbol{s}_1$ e os seus vizinhos topológicos em direção da amostra ξ pela fração de ϵ_b e ϵ_n, respectivamente, da distância total:
$\boldsymbol{\Delta w}_{\boldsymbol{s_1}} = \varepsilon_b(\boldsymbol{\xi} - \boldsymbol{w}_{\boldsymbol{s_1}})$
$\boldsymbol{\Delta w_n} = \varepsilon_n(\boldsymbol{\xi} - \boldsymbol{w_n})$, em que $\varepsilon_b > \varepsilon_n$
10: Conectando os nodos mais próximos: Se $\boldsymbol{s}_1$ e $\boldsymbol{s}_2$ estão conectados, iguale a zero a idade da conexão. Se a conexão não existe, crie-a.
11: Remoção de conexões antigas: Remova uma dada conexão se tiver uma idade maior que age_{max}. Se isso resultar em nodos sem nenhuma conexão, remova o nodo também.
12: Inserção de novos nodos: Se o número de amostragens realizadas for múltiplo de um inteiro λ, insira um novo nodo da seguinte maneira:
13: Determine o nodo q com o maior erro acumulado.
14: Insira um novo nodo r no ponto médio entre o nodo q e o nodo vizinho f com o maior erro acumulado.
15: $w_r = 0,5(w_q + w_f)$
16: Insira uma conexão nova ligando r a q e a f e remova a conexão original entre q e f.
17: Decremente a variável de erro de q e f pela multiplicação por um fator constante de α e inicialize o erro de r com o mesmo valor do erro de q, já atualizado.
18: Decremento do erro: Decremente todas as variáveis de erro pela multiplicação delas por uma constante β.
19: Critério de parada: Finalize o treinamento se o critério de parada foi atingido, se não, continue a partir de 1 (O Algoritmo do GNG não fixa nenhum critério de parada específico, pode ser o tamanho da rede, alguma medida de desempenho, entre outros).
20: **end while**

3.2.3 *Growing When Required*

Growing When Required (GWR)(MARSLAND; SHAPIRO; NEHMZOW, 2002) é um mapa auto-organizável a qual cresce e se adapta ao decorrer do tempo, assim como GNG, contudo ambos divergem quanto ao processo de inserção de nodos. No caso do GWR, enquanto o mapa não contiver nodos em quantidade suficiente para representar adequadamente o conjunto de treinamento com um erro aceitável, o processo de treinamento persiste na inserção de novos

nodos. Similar ao GNG, a adição dos novos nodos é sempre feita na proximidade do nodo com o maior erro acumulado. Seu processo de treinamento pode ser visto no Algoritmo 8.

Nesse algoritmo, um laço é realizado e nele primeiramente cada variável de treino é selecionada aleatoriamente e o nodo vencedor (s_1) e o segundo vencedor (s_2) são encontrados, assim como no GNG (linhas 5, 6 e 7). Posteriormente, se na existir uma conexão entre o nodo s_1 e s_2, uma nova conexão é criada, caso contrário, a idade da conexão entre s_1e s_1 é zerada (linhas 8 até 12). Então a ativação do nodo vencedor é calculada e caso ela ultrapasse um limiar, um novo nodo com vetor de peso igual ao vetor de entrada é adicionado à rede (linhas 14 até 17). Caso contrário, os pesos do nodo vencedor e seus vizinhos são atualizados de forma similar ao caso do GNG, contudo, agora há uma multiplicação pelas funções $h_s(t)$ e $h_i(t)$ para o nodo vencedor e para os seus vizinhos respectivamente, em que t é um contador de quantas vezes o laço foi executado até o dado momento (linhas 18 até 20). Após a inserção ou atualização dos nodos, a idade de todas as conexões que partem de s_1 são atualizadas (linha 21) e h_s e h_i são reduzidos conforme mostrado nas 22 e 23. Por fim, todas conexões que ultrapassam um limiar de idade são removidas e os nodos que terminam ficando sem conexão após essa operação são também removidos (linhas 24 e 25).

3.2.4 Local Adaptive Receptive Field Self-Organising Map

O LARFSOM(ARAÚJO; COSTA, 2009) é um modelo de mapa auto-organizável baseado no GWR, inicialmente proposto com o intuito de lidar com compressão de imagens. Ele tenta preservar as características de aprendizagem competitiva e de agrupamento do SOM. LARFSOM é mais simples que o GWR no sentido de que somente a unidade de melhor correspondência tem suas conexões atualizadas. Ao contrário do GNG, o LARFSOM cresce apenas quando o valor de ativação de um protótipo não é satisfatório de acordo com um limiar e não em intervalos fixos. Além disso, ele apresenta uma convergência mais rápida. Seu processo de treinamento pode ser visto no Algoritmo 9.

No início do laço, amostras $\boldsymbol{\xi}$ são aleatoriamente selecionadas do conjunto de dados de treinamento $\boldsymbol{\Xi}$ e os os nodos vencedores s_1 e s_2 são encontrados por meio do cálculo da distância (os nodos de menor distância são selecionados) - linhas 6, 7 e 8. Além disso, cada nodo possui um contador de vitórias associado, o qual é atualizado cada vez que o respectivo nodo vence (linha 9). Além disso, uma nova conexão é inserida entre o nodo vencedor e segundo nodo vencedor, seguido do calculo da atividade do nodo s_1, a qual é calculada em função de

Algoritmo 8 Algoritmo GWR

Entrada: $\boldsymbol{\Xi}$
Inicialize os parâmetros: ρ_f, ε, a_T, $age = 0$, d_m, $t = 0$, e_{min}, $N = 2$
Inicialize os N pesos a partir de $\boldsymbol{\Xi}$
while não atingir o critério de convergência **do**
    Selecione um vetor $\boldsymbol{\xi}$ de $\boldsymbol{\Xi}$ como entrada
    Calcule as distâncias euclidianas entre $\boldsymbol{\xi}$ e os vetores de peso $(\boldsymbol{w_i})$ como: $\boldsymbol{d}(\boldsymbol{\xi}, \boldsymbol{w_i}) \forall i \in \{1, ..., N\}$
    Encontre os dois vetores peso $\boldsymbol{w_{s1}}$ e $\boldsymbol{w_{s2}}$ mais próximos de $\boldsymbol{\xi}$, i.e.: $\boldsymbol{d}(\boldsymbol{\xi}, \boldsymbol{w_{s1}}) \leq \boldsymbol{d}(\boldsymbol{\xi}, \boldsymbol{w_{s2}}) \leq \boldsymbol{d}(\boldsymbol{\xi}, \boldsymbol{w_i}) \forall i \in \{1, ..., N\} \setminus \{s1, s2\}$
    if $\nexists$ conexão entre $\boldsymbol{s}_1$ e $\boldsymbol{s}_2$ **then**
        Insira uma nova conexão entre $\boldsymbol{s}_1$ e $\boldsymbol{s}_2$
    else
        $age_{s_1 s_2} = 0$
    end if
    Calcule a atividade de $\boldsymbol{s}_1$: $a_{s_1} = exp(-\|\boldsymbol{\xi} - \boldsymbol{w_{s_1}}\|)$
    if $a_{s_1} > a_T$ e $c_d < h_T$ **then**
        Adicione um novo nodo à rede com vetor de peso $w_n = \boldsymbol{\xi}$
        Atualize o número de nodos $N = N + 1$
    else
        atualize o peso do nodo s_1 somando o resultado da equação $\Delta_{\boldsymbol{w}_{s_1}} = \epsilon_b * h_s * (\xi - \boldsymbol{w}_{s_1})$ ao seu peso atual;
        atualize o peso dos nodos vizinhos ao vencedor somando o resultado da equação $\Delta_{\boldsymbol{w}_n} = \epsilon_n h_i (\xi - \boldsymbol{w}_n)$ ao seus pesos atuais;
    end if
    Incremente a age de todas as conexões que partem de $\boldsymbol{s}_1$
    Reduza o contador de acordo com a frequência que o nodo $\boldsymbol{s}_1$ vencedor tem sido ativado: $h_s(t) = h_0 - \frac{S(t)}{\alpha_n}(1 - \frac{e^{-\alpha_n}}{\tau_n})$
    Realize esse processo de redução também para seus vizinhos: $h_i(t) = h_0 - \frac{S(t)}{\alpha_n}(1 - \frac{e^{-\alpha_n}}{\tau_n})$
    Remova todas as conexões com $age_i > age_{max} \forall i \in 1, ..., Nx1, ..., N$
    Remova nodos desconectados
end while

campo receptivo, calculado através da distância entre os pesos do nodo vencedor e segundo vencedor (linhas 10 até 12). Caso a atividade de s_1 seja menor que um limiar predefinido, um novo nodo é adicionado, o seu peso é dado pelo vetor de entrada e as conexões são atualizadas da seguinte forma: a conexão entre s_1 e s_2 é removida e uma nova conexão é estabelecida entre 2 dos 3 nodos (inserido e vencedores), os quais possuírem pesos de menor distância (linha 14 até linha 18). Caso contrário, o peso do nodo vencedor é atualizado conforme a diferença entre seu vetor de peso e a amostra atual, multiplicada por um parâmetro predefinido ρ (linhas 20 até 23).

Algoritmo 9 Algoritmo LARFSOM

1: **Entrada:** $\boldsymbol{\Xi}$
2: **Saída:**
3: Inicialize os parâmetros: ρ_f, ε, a_T, $d_i = 0$, d_m, $t = 0$, e_{min}, $N = 2$
4: Inicialize os N pesos a partir de $\boldsymbol{\Xi}$
5: **while** não atingir o critério de convergência **do**
6: Selecione um vetor $\boldsymbol{\xi}$ de $\boldsymbol{\Xi}$ como entrada
7: Calcule as distâncias euclidianas entre $\boldsymbol{\xi}$ e os vetores de peso $(\boldsymbol{w_i})$ como: $\boldsymbol{d}(\boldsymbol{\xi}, \boldsymbol{w_i}) \forall i \in \{1, ..., N\}$
8: Encontre os dois vetores peso $\boldsymbol{w_{s1}}$ e $\boldsymbol{w_{s2}}$ mais próximos de $\boldsymbol{\xi}$, i.e.: $\boldsymbol{d}(\boldsymbol{\xi}, \boldsymbol{w_{s1}}) \leq \boldsymbol{d}(\boldsymbol{\xi}, \boldsymbol{w_{s2}}) \leq \boldsymbol{d}(\boldsymbol{\xi}, \boldsymbol{w_i}) \forall i \in \{1, ..., N\} \setminus \{s1, s2\}$
9: Incremente o contador de vitórias da BMU: $d_{s1} = d_{s1} + 1$
10: Insira uma nova conexão entre $\boldsymbol{s}_1$ e $\boldsymbol{s}_2$
11: Calcule o campo receptivo de $\boldsymbol{s}_1$: $r_{s_1} = \|\boldsymbol{w_{s_1}} - \boldsymbol{w_{s_2}}\|$
12: Calcule a atividade de $\boldsymbol{s}_1$: $a_{s_1} = \frac{exp(-\|\boldsymbol{\xi} - \boldsymbol{w_{s_1}}\|)}{r_{s_1}}$
13: **if** $a_{s1} < a_T$ **then**
14: Adicione um novo nodo à rede com vetor de peso $w_n = \boldsymbol{\xi}$
15: Atualize o número de nodos $N = N + 1$
16: Remova a conexão entre $\boldsymbol{s}_1$ e $\boldsymbol{s}_2$
17: Calcule: $\boldsymbol{d}(\boldsymbol{w_n}, \boldsymbol{w_{s_1}}), \boldsymbol{d}(\boldsymbol{w_n}, \boldsymbol{w_{s_2}}), \boldsymbol{d}(\boldsymbol{w_{s_1}}, \boldsymbol{w_{s_2}})$
18: Insira conexões entre os 2 nodos com menores distâncias
19: **else**
20: $\Delta_{s_1} = \rho * (\boldsymbol{\xi} - \boldsymbol{w_{s_1}})$
21: em que,
22: Se $d_i \leq d_m$, então $\rho = \varepsilon * \rho_f^{\frac{d_i}{d_m}}$
23: Se $d_i > d_m$, então $\rho = \varepsilon * \rho_f$
24: **end if**
25: Atualize a contagem de iterações $t = t + 1$
26: **end while**
27: Remova nodos desconectados

3.2.5 Growing Incremental Self-Organising Neural Network

O modelo Growing Incremental Self-Organising Neural Network (GISONN) (LIU; BAN, 2015) foi desenvolvido pensando em ter aspectos de uma rede SOM de estrutura variante no tempo com a capacidade de remoção de *outliers*, assim como visualização da rede em 2D reportando claramente os agrupamentos realizados, sendo o crescimento dessa rede dado de uma forma hexagonal, um pouco mais diferente se comparado com outros modelos descritos nesta seção.

O processo desse modelo é dado pelo Algoritmo 10, em que inicialmente se é escolhida uma amostra aleatória do conjunto de dados $\mathcal{D}$, cuja informação é utilizada para criação de uma estrutura SOM formada por 3 hexágonos, com um nodo central $\tilde{\mathrm{n}}$ de peso de mesmo valor de $\boldsymbol{x}_{1st}$. Os outros nodos possuem sempre uma distância menor ou igual a uma variável ϵ, posicionados todos de forma a criar hexágonos. A partir daí, o treinamento passa a varrer todos os dados, primeiramente encontrando o nodo $\tilde{\mathrm{n}}*$ mais próximo do respectivo dado. Se $\boldsymbol{x}$ é suficientemente próximo do peso de $\tilde{\mathrm{n}}*$, mas ainda não foi incluído em nenhum local de sua vizinhança, ele é considerado parte da vizinhança de $\tilde{\mathrm{n}}*$ e a rede não é incrementada. Caso contrário, é verificado se os dados são próximos (distância menor do que ϵ) do peso do nodo mais próximo, mantendo a rede de mesmo tamanho e atualizando seus pesos ou aumentando a rede de forma hexagonal.

Por fim, pode ocorrer de aparecerem *outliers* após o processo de treinamento, sendo eles detectados a partir de um cálculo de densidade dos dados e reduzidos por meio da adição de novos nodos.

Algoritmo 10 Algoritmo GISONN

1: A partir do conjunto de dados selecione aleatoriamente uma entrada $\boldsymbol{x}_{1st}$;
2: Criar uma estrutura de 3 hexágonos;
3: **for** $\forall \boldsymbol{x} \in \mathcal{D} \setminus \{\boldsymbol{x}_1 st\}$ **do**
4: $\quad \tilde{\mathrm{n}}* = \arg\min_{\tilde{\mathrm{n}}* \in G_N} d(\boldsymbol{W}(\tilde{\mathrm{n}}), \boldsymbol{x})$;
5: $\quad$ **if** $d(\boldsymbol{W}(\tilde{\mathrm{n}}*), \boldsymbol{x}) \leq \epsilon E(\boldsymbol{x} \notin \mathcal{N}(\tilde{\mathrm{n}}*))$ **then**
6: $\quad\quad \mathcal{N}(\tilde{\mathrm{n}}*) = \mathcal{N}(\tilde{\mathrm{n}}*) \cup \boldsymbol{x}$;
7: $\quad$ **else**
8: $\quad\quad$ Treinamento incremental;
9: $\quad$ **end if**
10: **end for**
11: Reduzir *outliers*;

3.2.6 Growing Hierarchical Neural Gas

O *Growing Hierarchical Neural Gas* (GHNG) (PALOMO; LÓPEZ-RUBIO, 2016) é um modelo hierárquico baseado no GNG o qual cria novos SOMs conforme o necessário, sendo cada SOM treinado individualmente, conforme mostrado no Algoritmo 11. Inicialmente, nesse processo, os parâmetros, estados e valores iniciais das variáveis são definidos, assim como algumas considerações devem ser feitas. Dito isso, o número de nodos iniciais é $H = 2$, as conexões entre eles são dadas por $a_{w_1 w_2} = a_{w_2 w_1} = 0$, os erros iniciais de cada nodo são dados por $e_1 = e_2 = 0$ e, por sua vez, $\boldsymbol{w}_1$ em consonância com $\boldsymbol{w}_2$ são aleatoriamente determinados, sendo a quantidade de conexões, dos erros iniciais de cada nodo, dos pesos e dos erros variável conforme o número de nodos da rede. O conjunto de treinamento é denotado por $\mathcal{S}$, com $\mathcal{S} \subset \mathbb{R}^D$, sendo D a dimensão no espaço dos dados. Além disso, cada conexão tem uma variável que indica o tempo desde que a conexão foi estabelecida (age), sendo as conexões formadas entre nodos e o conjunto $\mathcal{A} \subseteq \{1, .., H\} \times \{1, .., H\}$ é a notação do conjunto de conexões entre nodos.

Após inicialização, o treinamento do primeiro SOM se dá pelo sorteio aleatório de uma amostra de $\mathcal{S}$ (linha 8) sendo comparada com os nodos da rede de forma a determinar o primeiro e segundo nodo mais próximos (linhas 9 e 10). Caso tenham nodos conectados a q, a variável age de todas essas conexões é incrementada de 1 (linha 11). Já a atualização dos nodos é determinada pelos parâmetros ϵ_b e ϵ_n, pois os pesos do nodo vencedor e seus vizinhos são alterados conforme ϵ_b, como mostrado nas linhas 13 a 20, enquanto que os nodos que não pertencem a nenhum desses conjuntos é inalterado. As conexões dos nodos são controladas através da variável age, um contador de tempo de duração daquela conexão, sendo resetado para q e s (linha 24), pois já que eles possuem dados próximos com relação a outros nodos, é interessante manter sua conexão intacta, removendo conexões dos nodos relativamente mais distantes de todas as variáveis de entrada (linha 28). Caso entre os nodos q e s não haja nenhuma conexão, uma nova conexão é criada. Ademais, periodicamente em valores de t múltiplos de valores de um parâmetro λ é criado um novo nodo a partir do nodo de maior erro e seu respectivo vizinho de maior erro (linhas 29 a 36). Quanto ao crescimento da rede, a cada vez que a condição da linha 38 for respeitada, é realizada a análise da condição de inserção da rede, determinada pela variação do erro supervisionado médio da rede como um todo, tendo que superar um parâmetro predefinido τ e, caso esse parâmetro não seja superado, a rede entra no estado de convergência, em que não há mais o processo de inserção de novos nodos.

Por fim, os erros associados a cada nodo são reduzidos por meio da multiplicação com um parâmetro predefinido d.

Além do processo de treinamento de cada mapa, há o controle de inserção de novos mapas, sendo cada novo mapa criado a partir de cada nodo e tendo como conjunto de treino,

$\mathcal{S}_i = \{x \in S \mid i = \arg\min_{j \in \{1,\dots,H\}} \|\boldsymbol{w}_j - \boldsymbol{x}\|\}$.

O treinamento para cada novo mapa criado segue o mesmo processo descrito anteriormente, acontecendo ao final de treinamento da rede hierarquicamente superior. Dessas novas redes criadas e treinadas, caso alguma possua somente 2 nodos ao final do processo, elas são podadas. Esse processo se repete até o momento em que um número de níveis de hierarquia predefinido é atingido.

3.2.7 *Growing Hierarchical Self-Organising Representation Map*

Growing Hierarchical Self-Organising Representation Map (GHSORM) é uma rede hierárquica de estrutura variante no tempo a qual se aproveita de conceitos de *Denoising Autoencoders* (DA) e *Growing Hierarchical Self-Organising Map* (GHSOM). O Algoritmo 12 explica o funcionamento do processo de treinamento em alto nível. Primeiramente todos os dados, após sofrerem inserção de ruído gaussiano, são utilizados para o treinamento do DA, o qual é utilizado para criar uma representação de cada dado de entrada (linha 1). Com a representação dos dados criada, é criado o primeiro nodo a partir da média de todos os dados (linha 2). Sendo assim, é calculado um erro a partir das distâncias entre o peso criado e o conjunto de dados (linha 3), sendo de uma maneira geral a expressão para o cálculo deste erro:

$$qe_{\nu}^{(\sigma)} = \sum_{p=1}^{P} \left\| u_{\nu} - y^{(p)} \right\|, \tag{3.1}$$

em que o índice superior σ indica a numeração do mapa auto-organizável, enquanto que o inferior ν é referente ao índice da BMU e, por fim, P é a quantidade de amostras referentes ao nodo. Nesse caso inicial, somente há um nodo.

Todos os nodos criados posteriormente, devem obedecer a condição $qe(\sigma) < \tau_2 qe^{(0)}$, sendo terminado o processo de treinamento (linha 4). Enquanto isso não ocorre, utiliza-se uma representação para todos os SOMs ou uma específica para cada SOM a depender se a rede foi definida para operar em modo adaptativo ou não. Com isso é criada uma rede SOM 2x2 a qual é treinada enquanto seu erro médio não-supervisionado for maior que $qe_u T_1$,

Algoritmo 11 Algoritmo GHNG

Inicialização:
$H = 2$
Pesos iniciais da rede: $\boldsymbol{w}_1$ e $\boldsymbol{w}_2$
$e_1 = e_2 = 0$
$a_{\boldsymbol{w}_1\boldsymbol{w}_2} = a_{\boldsymbol{w}_2\boldsymbol{w}_1} = 0$
Código:
for cada passo de tempo t **do**
    Sortear uma amostra $\boldsymbol{x_t} \in \mathbb{R}^D$ aleatoriamente de $\mathcal{S}$;
    $q = argmin_{i\in\{1,..,H\}}\|\boldsymbol{w}_i - \boldsymbol{x}_t\|$;
    $s = argmin_{i\in\{1,..,H\}-\{q\}}\|\boldsymbol{w}_i - \boldsymbol{x}_t\|$;
    Atualizar age de todas conexões do nodo q;
    $e_q(t+1) = e_q(t) + \|\boldsymbol{w}_q - \boldsymbol{x}_t\|^2$;
    for $i \in H$ **do**
        if $i == q$ **then**
            $\epsilon(t,i) = \epsilon_b$;
        else if $(i \neq q)E(i,q) \in A$ **then**
            $\epsilon(t,i) = \epsilon_n$;
        else
            $\epsilon(t,i) = 0$;
        end if
        $\boldsymbol{w}_i(t+1) = (1-\epsilon(t,i))\boldsymbol{w}_i(t) + \epsilon(t,i)\boldsymbol{x}_t$;
    end for
    if q e s são conectados **then**
        Resetar idade da conexão;
    else
        Criar conexão;
    end if
    Remover conexões que atingirem valor máximo de sua respectiva variável age e unidades sem saídas;
    if t é inteiro múltiplo de λE SOM está na fase de crescimento **then**
        Selecionar o nodo de maior erro r;
        Selecionar o nodo de maior erro entre os vizinhos de r, chamado de z;
        Crie um novo nodo k conectado aos nodos r e z;
        $\boldsymbol{w}_k(t) = \frac{1}{2}(\boldsymbol{w}_r(t) + \boldsymbol{w}_z(t))$;
        Remova a conexão entre r e z;
        Reduza os erros associados à r e z multiplicando seus valores por α;
        Erro associado a k é criado com o mesmo valor de e_r;
    end if
    if $mod(t, 2\lambda) == \lfloor\frac{3}{2}\lambda\rfloor E$ SOM está na fase de crescimento **then**
        if $\frac{MQE_{\text{old}} - MQE_{\text{new}}}{MQE_{\text{old}}} < \tau$ **then**;
            Aceitar inserção;
        else
            Manter a rede como estava antes da inserção;
        end if
    end if
    Diminuir todos erros associados e_i multiplicando por um parâmetro d;
end for

sendo u o nodo a partir do qual essa nova rede foi criada e T_1 um parâmetro predefinido. Esse mapa cresce verticalmente a partir de linhas ou horizontalmente a partir de colunas a depender de quais o nodo que apresenta maior erro não-supervisionado e seu vizinho de maior erro não-supervisionado.

Algoritmo 12 Algoritmo GHSORM

```
Treinar DA e criar representações dos dados de entrada;
Inicializar a rede com um único nodo centralizado em meio aos dados;
Calcular qe_0 a partir desse nodo;
while Há pelo menos um nodo em que qe_i > qe_0 * T_2 do
    if Verifica se o GHSORM é adaptativo then
        Usar as representações criadas a partir do DA;
    else
        Copiar o DA para ser ajustado durante o treinamento;
    end if
    Criar um mapa-filho e treinar com todos os dados cujo nodo pai é BMU;
    while MQE > qe_u T_1 do
        Mapa é treinado;
        Uma nova coluna ou linha é adicionada entre a unidade de erro e seu vizinho de
        pior desempenho;
        Os novos nodos são gerados a partir da média dos pesos dos vizinhos;
    end while
end while
```

3.3 DISCUSSÃO

Não foi visto em meio a literatura um modelo de SOM previamente aplicado em um problema de reconhecimento visual de lugares no contexto de navegação autônoma, como nesta dissertação. Sendo assim, modelos similares são abordados aqui, geralmente utilizados para classificação e extração de características, sendo o mais similar o LASOM, baseado no GNG.

A capacidade da rede de se adaptar ao surgimento de novos dados é de extrema importância no caso da aplicação abordada nesta dissertação, visto que no problema de VPR novos dados constantemente surgem e são comparados com dados antigos com a finalidade de saber a localização do agente autônomo, i.e., se uma rede é utilizada como meio para essa comparação, é importante que ela seja capaz de se adaptar a esses novos dados gerados ao decorrer do tempo. Por isso, um SOM de estrutura variante no tempo foi o modelo escolhido para essa tarefa.

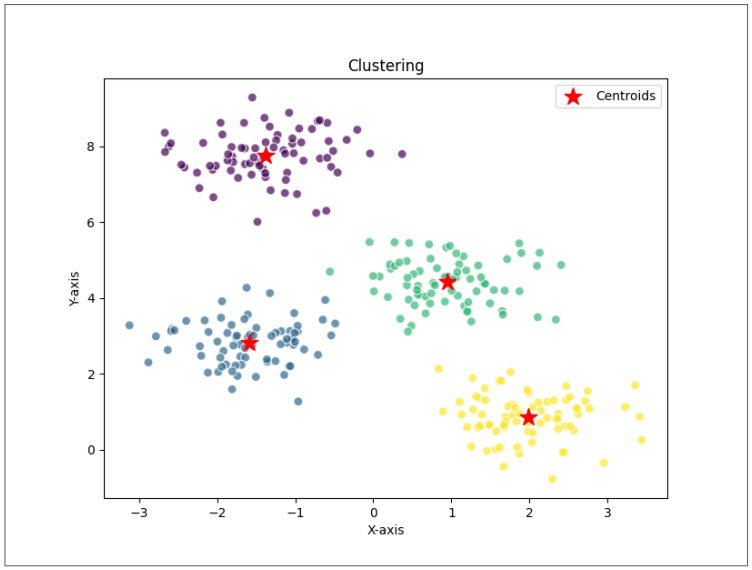
Modelos de SOM de estrutura variante no tempo, de uma maneira geral, possuem diversos atributos similares, como a definição de uma rede inicial a partir da qual a rede irá crescer, um método de ajuste de pesos, uma relação de vizinhança e uma forma de atualização da topologia dos nodos e conexões entre os mesmos. Contudo, cada uma dessas redes possui suas desvantagens a partir das quais levam a uma conclusão de qual a melhor rede para se ter de ponto de partida para o caso da aplicação deste trabalho. Tanto o GCS, quanto o GNG tem a quantidade de nodos aumentada sem um critério que leva em conta o desempenho, não seguindo as distribuições dos dados de entrada tão bem como nos outras redes. GISONN, GHNG e GHSORM demonstram ter um treinamento mais custoso, devido a sua forma de crescimento da rede, aparentemente produzindo mais nodos e/ou arestas do que o necessário, por diferentes razões: o GISONN cresce necessariamente de forma hexagonal, conceitualmente fazendo com que conexões e nodos sigam um padrão topológico já predeterminado e não necessariamente porque foi estritamente necessário; enquanto que o GHNG e GHSORM criam mapas inteiros a partir do nodo selecionado, processo no qual ocorre em cadeia. Ambos GWR e LARFSOM crescem conforme um critério de necessidade, porém, o LARFSOM é mais simples que o GWR, no sentido que o contador do nodo vencedor é mais simples de se calcular e somente a BMU tem suas conexões alteradas. Sendo assim, o LARFSOM foi o ponto de partida para a rede desenvolvida, detalhada no próximo capítulo.

4 MAPA AUTO-ORGANIZÁVEL DE CAMPO RECEPTIVO ADAPTATIVO LOCAL COM SEMELHANÇA AJUSTÁVEL POR CATEGORIA LARFSOM-ARC

O Mapa Auto-organizável de Campo Receptivo Adaptativo Local com Semelhança Ajustável por Categoria ou *Local Adaptive Receptive Field Self-organising Map with Adjustable Resemblance for Category* (LARFSOM-ARC), proposto nesta dissertação, realiza agrupamento de dados de entrada para tarefa de VPR. Neste caso, espera-se que o LARFSOM-ARC atue com celeridade, precisão alta, aprendizagem incremental e rápida, capacidade de atuar em diferentes ambientes e contextos de aquisição de dados, e, capacidade para lidar com problemas maiores em termos de dimensões, número de instâncias e/ou categorias dos dados. Dessa maneira, vetores representando características extraídas de conjuntos de dados de entrada são agrupados e são representados por protótipos (centroides) dos grupos, as estrelas esboçadas na Figura 11. Para a identificação do lugar, uma imagem a ser reconhecida é inicialmente comparada com os centroides. Escolhido aquele protótipo que mais se pareça com a imagem de teste, esta é comparada com todas as imagens agrupadas no grupo vencedor para se determinar a imagem agrupada mais semelhante a imagem teste (*best match*). A escolha determina o lugar que é considerado correto se ele estiver dentro de um intervalo de distância geométrica da posição da imagem teste. Em síntese, uma imagem teste é apresentada ao sistema, o LARFSOM-ARC determina o centroide que mais se parece com ela e verifica-se se a posição espacial da imagem escolhida está dentro de um intervalo máximo de distância da imagem teste para confirmar a escolha como acerto. Para realizar tal tarefa, decidiu-se trabalhar com um SOM capaz de se adaptar dinamicamente aos dados de entrada ajustando o número de nodos da rede e o campo receptivo (ARAÚJO; COSTA, 2009), enquanto simultaneamente pode ajustar a similaridade entre amostras de uma mesma categoria.

A abordagem proposta deve oferecer redução da complexidade computacional com respeito a um algoritmo que compare uma imagem teste a todas as imagens de uma base de dados. Considerando uma única imagem de entrada, a complexidade computacional de um algoritmo que compare a entrada com todas N imagens armazenadas em um conjunto de dados é dada por $O(N)$. No caso do LARFSOM-ARC, a complexidade computacional é dada por $O(K+\frac{N}{K})$, em que K é o número de nodos da rede. Essa equação da complexidade computacional do modelo é devido às K comparações realizadas entre a imagem e os protótipos dos nodos, somada a quantidade de imagens por nodo (categoria), visto que a imagem de teste será somente comparada com imagens armazenadas que foram agrupadas no nodo de protótipo

Figura 11 – Ilustração de uma tarefa de agrupamento.



Fonte: Autor (2024).

mais similar à imagem de teste e supondo que as imagens são homogeneamente distribuídas pelos nodos. Vale ressaltar, embora haja essa consideração, na prática a distribuição não é exatamente homogênea, portanto, quanto maior for a variância mais custoso (e demorado) é o processo de comparação.

Ademais, o valor ótimo - o qual minimiza a complexidade computacional do modelo - pode ser encontrado igualando-se a derivada da equação de número de comparações do modelo a zero,

$$F(K) = K + \frac{N}{K}, \tag{4.1}$$

sendo assim,

$$1 - K^{-2}N = 0, \tag{4.2}$$

sendo $K = \sqrt{N}$ a solução ótima obtida por meio dessa equação e $O(2\sqrt{N})$ a complexidade computacional.

A melhor estrutura para redução da complexidade computacional seria em árvore, com uma complexidade computacional de $O(log(N))$, sendo mencionada na seção dedicada a trabalhos futuros.

Este capítulo é dividido em 3 partes: a primeira aborda a forma como foram extraídos os vetores de característica; a segunda trata do treinamento do LARFSOM-ARC e, por fim, a terceira parte discute o passo-a-passo da tarefa de reconhecimento.

4.1 EXTRAÇÃO DE CARACTERÍSTICAS

Inicialmente, trata-se da representação das imagens para serem processadas. Ao invés de se considerar apenas uma única representação de imagem como entrada, esta é alterada por um filtro temporal que emprega uma janela de imagens. Com esta representação busca-se desambiguar possíveis imagens parecidas, adicionando informações de um conjunto de imagens precedentes, capturando o contexto temporal no qual uma imagem individual se encontra. O mencionado filtro é definido pela Equação 4.3.

$$\boldsymbol{s}_n = \sum_{l=n}^{n-L} \alpha^{n-l} \boldsymbol{x}_l. \tag{4.3}$$

onde $\boldsymbol{x}_l$ é a imagem de entrada de índice l, que varia de n até $n - L$, em que n indica o instante de tempo atual e L determina o tamanho da janela das imagens consideradas. Além disso, α é uma constante utilizada para dar mais peso à recência pois $0 \leq \alpha \leq 1$, portanto o peso de cada imagem de contexto diminui com o aumento da defasagem temporal da imagem atual.

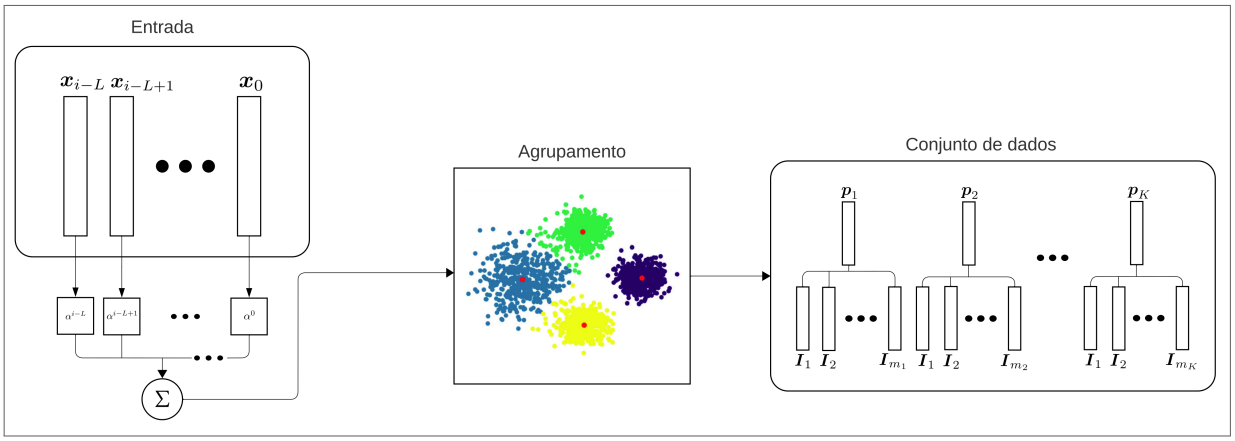
Um esboço visual desse filtro é exibido na Figura 12, em que cada entrada passa por um bloco de α elevado a uma potência determinada pelo atraso e cada valor ponderado é somado para produzir a imagem contextualizada. Esta imagem contextualizada será chamada de imagem de entrada ou apenas imagem daqui em diante.

Como extrator de características, escolheu-se a rede neural HybridNet proposta por CHEN et al. (2017), devido a seu desempenho alcançado no trabalho de HAUSLER; JACOBSON; MILFORD (2019), o qual a utilizou e obteve melhores resultados do que a bem estabelecida rede NetVLAD ((ARANDJELOVIC et al., 2016)).

4.2 TREINAMENTO

A Figura 12 ilustra um diagrama geral sobre o treinamento do mapa auto-organizável que se constitui o cerne do sistema de reconhecimento. Até chegar ao bloco de agrupamento, todas as operações realizadas são referentes ao extrator de características mencionado na seção 4.1. Além disso, nessa figura a rede possui K nodos (protótipos) p e cada nodo agrupa respectivamente m_r imagens, com $r = 1, 2, ..., K$. Agora, a discussão passa a ser o bloco de agrupamento, o qual gera o conjunto de grupos empregado para a tarefa de reconhecimento.

Figura 12 – Diagrama do processo de treinamento do modelo desenvolvido. A sequência gerada pelo filtro é a entrada do sistema e a saída é uma representação da estrutura da rede.



Fonte: Autor (2024).

Como LARFSOM-ARC se trata de um SOM de estrutura variante no tempo, ele é composto por um conjunto de nodos e arestas, podendo ser visto como um grafo. O Algoritmo 13 apresenta o pseudocódigo referente à etapa de treinamento dessa rede. As inicializações compreendem o trecho do código da linha 01 à linha 12. O vetor de pesos referentes aos dois nodos iniciais é inicializado a partir de duas amostras sorteadas do conjunto de dados de treinamento. Por sua vez, a variável n_etapas diz respeito à quantidade de etapas, em que cada etapa diz respeito à atualização de limiares da rede (linha 41). O vetor $\boldsymbol{limiares}$ indica os valores dos limiares de ativação de cada nodo no mapa, os quais determinam quais amostras são agrupadas ou não no nodo. Esse vetor passa a ter uma maior dimensionalidade conforme for ocorrendo a inserção de novos nodos e pode ter valores de limiares alterados (linha 24). Por sua vez, matriz $\boldsymbol{arestas}$ indica as conexões entre nodos, enquanto que a variável $\boldsymbol{vitórias}$ é um contador que indica quantas vezes um determinado nodo foi vencedor, sendo que $v_{máx}$ é o número máximo de vitórias por nodo. Por fim há uma métrica, $contador\_erro$, para estabelecer a condição de parada que será explicada mais adiante.

Para cada época, é realizada uma permutação na ordem dos componentes de cada vetor de amostras (linha 15), de forma que as amostras sempre serão selecionadas para treinamento em ordem aleatória. Em seguida, para cada dado de entrada o algoritmo faz a busca do protótipo vencedor (linha 18). A busca está descrita no Algoritmo 14, o qual compara a amostra atual com todos pesos dos nodos do mapa para encontrar aquele com menor distância para o vetor de entrada (primeiro vencedor) e em seguida, encontra-se o protótipo com menor distância para o vetor de entrada (segundo vencedor), excetuando-se o primeiro vencedor.

O cálculo das distâncias no Algoritmo 14 tem o mesmo critério de distância do extrator

Algoritmo 13 Código do treinamento

Inicialização:
% Variáveis referentes à estrutura da rede
$\boldsymbol{peso1} = \boldsymbol{amostras}$[inteiro aleatório entre 1 e tamanho de amostras];
$\boldsymbol{peso2} = \boldsymbol{amostras}$[inteiro aleatório entre 1 e tamanho de amostras];
$\boldsymbol{pesos} = \{\boldsymbol{peso1}\ \boldsymbol{peso2}\}$;
$\boldsymbol{limiares} = \{l_0\ l_0\}$;
$\boldsymbol{arestas} = \{0\ 0; 0\ 0\}$;
$\boldsymbol{vitórias} = \{0\ 0\}$;
% Demais variáveis;
n_etapas = 1;
$v_{máx} = 100$;
contador_erro = 0;
Código:
for $i = 1, 2, \ldots, número\_de\_épocas$ **do**
    $\boldsymbol{amostras} = \boldsymbol{amostras}$[rand_perm()];
    for $j = 1, 2, \ldots, número\_de\_amostras$ **do**
        $\boldsymbol{amostra\_atual} = \boldsymbol{amostras}$[j];
        ($\boldsymbol{v1}$, $\boldsymbol{v2}$, $v1_{idx}$, $v2_{idx}$) = achar_vencedores($\boldsymbol{pesos}$, $\boldsymbol{amostra\_atual}$);
        $ativação = exp(-dist(\boldsymbol{v1}, \boldsymbol{amostra\_atual}))$;
        $\boldsymbol{vitórias}[v1_{idx}]$++;
        if $ativação < \boldsymbol{limiares}[v1_{idx}]$ **then**
            Adiciona novo peso igual a $\boldsymbol{amostra\_atual}$ à variável $\boldsymbol{pesos}$;
            Adiciona ativação ao $\boldsymbol{limiares}$;
            Incrementar dimensão de $\boldsymbol{vitórias}$ com um elemento inicializado em 0;
            atualizar_arestas($\boldsymbol{arestas}$, dist($\boldsymbol{v1}$,$\boldsymbol{v2}$), dist($\boldsymbol{v1}$, $\boldsymbol{amostra\_atual}$), dist($\boldsymbol{v2}$,$\boldsymbol{amostra\_atual}$), $v1_{idx}$, $v2_{idx}$);
        else
            atualizar_protótipos($v_{máx}$, ρ, $\boldsymbol{v_1}$, $\boldsymbol{amostra\_atual}$, $\boldsymbol{pesos}$);
        end if
    end for
    remover_protótipos_isolados($\boldsymbol{arestas}$, $\boldsymbol{protótipos}$, $\boldsymbol{pesos}$, $\boldsymbol{limiares}$);
    poda($\boldsymbol{pesos}$, $\boldsymbol{arestas}$, $\boldsymbol{vitórias}$);
    erro_atual = calcular_erro($\boldsymbol{pesos}$, $\boldsymbol{amostras}$);
    erro_relativo = (erro_atual - erro_médio)/erro_médio;
    if $erro\_relativo < err_{th}\ e\ i > n$ **then**
        contador_erro++;
    end if
    if contador_erro == 10 **then**
        contador_erro = 0;
        if n_etapas < 3 **then**
            atualizar_limiares($\boldsymbol{limiares}$);
        else
            poda($\boldsymbol{pesos}$, $\boldsymbol{arestas}$, $\boldsymbol{vitórias}$);
            break;
        end if
        n_etapas = n_ etapas + 1;
    end if
end for

Algoritmo 14 Achar vencedores ("achar_vencedores")

1: **Entrada:**
2: $\boldsymbol{pesos}$, $\boldsymbol{amostra\_atual}$
3: **Saída:**
4: $\boldsymbol{v1}$, $\boldsymbol{v2}$, $v1_{idx}$
5: **Inicialização:**
6: $\boldsymbol{dist\_pesos} = \{0\ 0\ ...\ 0\}$;
7: contador_distância $= 0$;
8: **Código:**
9: $\boldsymbol{dist\_pesos}$ = dist($\boldsymbol{pesos}$, $\boldsymbol{amostra\_atual}$);
10: ($\boldsymbol{v1}$, $v1_{idx}$) = min($\boldsymbol{dist\_pesos}$);
11: Remover $\boldsymbol{dist\_pesos}[v1_{idx}]$ do vetor $\boldsymbol{dist\_pesos}$;
12: ($\boldsymbol{v2}$, $v2_{idx}$) = min($\boldsymbol{dist\_pesos}$);

de características utilizado, a distância dos cossenos no HybridNet. Como pode ser visto no Algoritmo 15, nessa função, cada par de vetores de pesos tem o cosseno entre eles calculado, posteriormente calcula-se a diferença de 1 (um) e o cosseno encontrado (linha 11). Este retorno da função cresce com o aumento da distância entre os vetores.

Algoritmo 15 Distância dos cossenos ("dist")

1: **Entrada:**
2: $\boldsymbol{pesos}$;
3: $\boldsymbol{amostra\_atual}$
4: **Saida:**
5: $\boldsymbol{distância\_dos\_pesos}$;
6: **Inicialização:**
7: $\boldsymbol{distância\_dos\_pesos} = \{0\ 0\ ...\ 0\}$;
8: contador_distância $= 0$;
9: **Código:**
10: **for** $\boldsymbol{peso} \in \boldsymbol{pesos}$ **do**
11: distância $= 1 - \frac{\boldsymbol{peso} \cdot \text{amostra\_atual}}{||\boldsymbol{peso}||\ ||\boldsymbol{amostra\_atual}||}$;
12: $\boldsymbol{distância\_dos\_pesos}$[contador_distância] = distância;
13: contador_distância++;
14: **end for**

Voltando ao Algoritmo 13, tem-se o processo de busca que se inicia com o cálculo da ativação do nodo vencedor (linha 19) por uma função de base radial (gaussiana neste caso). A ativação considera a distância entre o primeiro vencedor e a instância atual. A unidade vencedora tem o contador de vitórias atualizado, somando-se um (linha 20). Se o valor da ativação do primeiro vencedor for menor que o limiar de ativação (linha 21), limiar que estabelece o grau de semelhança mínimo entre o protótipo vencedor e a instância de entrada, então insere-se um novo nodo que compreende (linhas 22 a 27): criar um vetor de pesos igual

ao vetor de entrada, definir o limiar da nova unidade como o limiar da unidade vencedora, acrescentar um elemento nulo ao contador de vitórias e atualizar as conexões do novo nodo e os dois vencedores da competição, removendo a conexão entre os dois vencedores (se existir) e criando duas conexões entre os dois pares de vetores mais próximos dos três possíveis pares. A inserção de um novo nodo ocorre porque a semelhança entre o vetor de entrada e o protótipo vencedor não é próxima o suficiente a ponto de pertencer à categoria vencedora. Essa inserção se dá pela concatenação do valor da amostra atual à variável $\boldsymbol{pesos}$, assim como a concatenação da ativação à variável $\boldsymbol{limiares}$ e $\boldsymbol{vitórias}$ com um valor inicializado em zero. Após isso, é necessária a atualização das conexões entre nodos dada pela função "$atualizar\_arestas$", mostrada no Algoritmo 16. Nessa função, as arestas entre os 2 nodos que possuem a menor distância entre as possíveis combinações de arestas, enquanto que a de maior distância é removida, caso já exista, mantendo somente uma conexão entre nodos de maior proximidade.

Algoritmo 16 Atualizar Arestas.

1: **Entrada:**
2: $\boldsymbol{arestas}$, d12, d13, d23,$v1_{idx}$, $v2_{idx}$;
3: **Saéda:**
4: $\boldsymbol{arestas}$;
5: **if** $(d31 \leq d32)$ e $(d32 \leq d12)$ **then**
6: Remover conexões entre $vencedor1$ e $vencedor2$
7: Inserir conexões entre $vencedor1$ e r, e entre $vencedor2$ e r
8: **end if**
9: **if** $(d31 \leq d32)$ e $(d32 > d12)$ **then**
10: Remover conexões entre $vencedor1$ e $vencedor2$
11: Inserir conexões entre $vencedor1$ e r
12: Inserir/manter conexões entre $vencedor1$ e $vencedor2$
13: **end if**
14: **if** $(d31 \geq d32)$ e $(d31 \geq d12)$ **then**
15: Remover conexões entre $vencedor1$ e $vencedor2$
16: Inserir conexões entre $vencedor2$ e r
17: Inserir/manter conexões entre $vencedor1$ e $vencedor2$
18: **end if**

Se a distância entre o vetor de pesos do nodo vencedor e o vetor de entrada for próxima o suficiente (determinada pelo limiar de ativação), o vetor de pesos do nodo vencedor é atualizado pela função $atualizar\_protótipos$ (linha 27 do Algoritmo 13). Nessa função (Algoritmo 17), ρ_f é a taxa de aprendizado dada por $\rho^{\frac{número\_de\_vitórias\_do\_vencedor}{v_{máx}}}$ (linha 7), *i.e.* decrescente enquanto o número de vitórias da BMU for menor que o parâmetro $v_{máx}$ e constante durante o resto do tempo de treinamento. Assim, o ajuste no peso vencedor é resultado da multiplicação

entre a diferença de sua posição e a da amostra atual com essa taxa de aprendizado.

Algoritmo 17 Atualizar Protótipos

1: **Entrada:**
2: $v_{máx}$, ρ, $\boldsymbol{v_1}$, $\boldsymbol{amostra\_atual}$, $número\_de\_vitórias\_do\_vencedor$, $\boldsymbol{pesos}$
3: **Saída:**
4: $\boldsymbol{pesos}$
5: **Código:**
6: **if** $número\_de\_vitórias\_do\_vencedor \leq v_{máx}$ **then**
7: $\quad \rho_f = \rho^{\frac{número\_de\_vitórias\_do\_vencedor}{v_{máx}}}$;
8: **else**
9: $\quad \rho_f = \rho$;
10: **end if**
11: $\boldsymbol{ajuste} = \rho_f$ * ($\boldsymbol{amostra\_atual}$ - $\boldsymbol{v_1}$);
12: $\boldsymbol{v_1} = \boldsymbol{v_1} + \boldsymbol{ajuste}$;
13: $\boldsymbol{pesos}[v1_{idx}] = \boldsymbol{v_1}$;

Ao final de cada época, é realizada a remoção de protótipos isolados (linha 32) e a poda (linha 33). A função $remover\_protótipos\_isolados$ (Algoritmo 18), encara a rede como uma matriz, em que 0 e 1 determinam se não há ou há uma conexão, respectivamente. Assim, caso uma coluna esteja completamente zerada, significa que o nodo representante daquela coluna está isolado, sendo completamente removido da rede. Já a poda (Algoritmo 19), a cada 5% épocas do número total de épocas, remove nodos que não possuem nenhuma vitória desde a última poda.

Algoritmo 18 Remover Protótipos Isolados

1: **Entrada:**
2: $\boldsymbol{arestas}$, $\boldsymbol{protótipos}$, $\boldsymbol{pesos}$, $\boldsymbol{limiares}$
3: **Código:**
4: Atribuir dimensão da matriz arestas a $arestas\_dim$;
5: **for** $i = 1, 2, \ldots, arestas\_dim$ **do**
6: $\quad$ soma_elementos = 0;
7: $\quad$ **for** $j = 1, 2, \ldots, arestas\_dim$ **do**
8: $\quad\quad$ soma_elementos += $\boldsymbol{arestas}$[i,j];
9: $\quad$ **end for**
10: $\quad$ **if** $soma\_elementos == 0$ **then**
11: $\quad\quad$ Remover linha e coluna de coordenada i da matriz $\boldsymbol{arestas}$;
12: $\quad\quad$ Remover peso de coordenada i;
13: $\quad\quad$ Remover limiar de coordenada i;
14: $\quad$ **end if**
15: **end for**

Por fim, o erro não-supervisionado (Algoritmo 20) é chamado (linha 34 do Algoritmo 13) e caso o erro não supervisionado relativo (linha 35 do Algoritmo 13) não varie acima de um

Algoritmo 19 Poda

1: **Entrada:**
2: $\boldsymbol{pesos}$, $\boldsymbol{arestas}$, $\boldsymbol{vitórias}$, $\boldsymbol{limiares}$
3: **Código:**
4: **if** $i/número\_de\_épocas\ mod(0.05 * número\_de\_épocas) == 0$ **then**
5: Atribuir dimensão de $\boldsymbol{vitórias}$ a vit_dim;
6: **for** $i = 1, 2, .., vit\_dim$ **do**
7: **if** $\boldsymbol{vitórias} == 0$ **then**
8: Remover $\boldsymbol{pesos}$[i] de $\boldsymbol{pesos}$;
9: Remover $\boldsymbol{limiares}$[i] de $\boldsymbol{limiares}$;
10: Remover linha e coluna de coordenada i de $\boldsymbol{arestas}$;
11: **end if**
12: **end for**
13: Zerar todos elementos de $\boldsymbol{vitórias}$;
14: **end if**

limiar err_{th}, durante 10 épocas (parâmetro definido experimentalmente), ocorre a atualização dos limiares, alternativa que ocorre 2 vezes durante todo o processo de treinamento. Logo, o treinamento é dividido em etapas, encerrando o treinamento após a terceira ocorrência. Como mostrado no Algoritmo 20, o erro não-supervisionado é calculado como sendo a soma das distâncias entre o vetor de pesos do protótipo e o vetor de pesos de cada uma das amostras no grupo, dividido pelo número de pesos.

Algoritmo 20 Calcular Erro Não Supervisionado

1: **Entrada:**
2: $\boldsymbol{pesos}$, $\boldsymbol{amostras}$
3: **Saída:**
4: erro $= 0$;
5: **Código:**
6: erro $= 0$;
7: Atribuir número de $\boldsymbol{pesos}$ à variável dim_pesos;
8: Atribuir número de $\boldsymbol{amostras}$ à variável dim_amostras;
9: **for** $i = 1, 2, ..., dim\_amostras$ **do**
10: $\boldsymbol{dist\_pesos}$ = dist($\boldsymbol{pesos}$, $\boldsymbol{amostras}$[i]);
11: $(\boldsymbol{v1}, v1_{idx})$ = min($\boldsymbol{dist\_pesos}$);
12: erro = erro + $\boldsymbol{dist\_pesos}[v1_{idx}]$;
13: **end for**
14: erro $= \frac{erro}{dim\_pesos}$;

A atualização dos limiares é realizada no Algoritmo 21, em que ocorre a seleção dos protótipos a terem seus limiares alterados (linha 4). Nesse sentido, duas possibilidades de seleção de protótipos foram cogitadas, a seleção conforme o Algoritmo 22 e conforme o Algoritmo 23.

No primeiro caso, os nodos selecionados a serem alterados possuem uma quantidade de amostras acima de um quantil determinado pelo parâmetro $quant_{sel_{th}}$ (linha 7 do Algoritmo 22). Por sua vez, o segundo caso consiste em separar a trajetória de treino em diversos trechos, já que se sabe a posição onde cada amostra foi coletada. Já para determinar o trecho em que o nodo pertence, visto que ele não possui uma posição pré-definida, é estimado um trecho a partir da maior quantidade de amostras pertencentes a um mesmo trecho. Então, conta-se a quantidade de imagens de trechos diferentes armazenadas em cada nodo, sendo os nodos com ocorrências de erro acima de um quantil $quant_{sel_{th}}$ selecionados para terem seus limiares alterados. Ambos quantis serão mencionados daqui em diante como "quantil critério".

Após isso, os nodos selecionados tem seu limiar de ativação alterado conforme a ativação das amostras agrupadas em cada nodo, sendo esse limiar de alteração determinado pela amostra acima de uma quantidade de amostras pertencentes ao nodo determinadas por $quant_{at_{th}}$ (linha 6 do Algoritmo 21), *i.e.* quantil de ativação,vezes o número total de amostras no nodo.

No fim, ao atingir convergência, a estrutura da rede é armazenada com os nodos e os vetores de características das imagens armazenadas nos seus respectivos nodos.

Algoritmo 21 Atualizar Limiares

1: **Entrada:**
2: $\boldsymbol{limiares}$, $\boldsymbol{pesos}$, $\boldsymbol{amostras}$
3: **Código:**
4: $\boldsymbol{limiares}$ = selecionar_protótipos($\boldsymbol{pesos}$, $\boldsymbol{amostras}$);
5: **for** Cada protótipo selecionado **do**
6: Aumentar o limiar de ativação do no selecionado conforme a mediana;
7: das ativações de amostras pertencentes a ele;
8: Atribuir limiar ao elemento de $\boldsymbol{limiares}$;
9: **end for**

Algoritmo 22 Seleção de nodos com base em suas amostras.

1: **Entradas:**
2: $feat\_test\_norm, features\_train\_normalized, w$
3: $distance\_type$
4: **Saída:**
5: $idxs$
6: Contar quantidade de amostras pertencentes a cada nodo;
7: Identificar os nodos com a maior quantidade de amostras, conforme o "quantil critério";
8: Retornar um indicador para as amostras associadas a esses vetores;

Algoritmo 23 Seleção de nodos com base no erro.

Entradas:
$feat\_test\_norm, features\_train\_normalized, w, clustered\_samples\_array, pos1,$
$distance\_type, quantile\_crit$
Saída:
$idxs$
Inicializar variáveis para cálculos futuros;
Calcular a distância percorrida a partir das posições de cada amostra e dividir em trechos para o conjunto de treinamento;
for cada grupo de amostras em nodos do conjunto de treinamento **do**
    Calcular os trechos baseados nas distâncias das amostras em relação aos nodos;
    Identificar o trecho predominante em cada nodo;
end for
Inicializar vetores de armazenamento para trechos relacionados às amostras de teste e aos nodos;
for cada amostra no conjunto de teste **do**
    Calcular as distâncias entre a amostra atual e os vetores de peso;
    Identificar os vetores de peso mais próximos da amostra;
    Determinar o trecho predominante associado aos vetores de peso selecionados;
    Associar os trechos da amostra e do nodo correspondente aos vetores de peso;
    Verificar distâncias entre os trechos e os armazenar;
end for
Ordenar as distâncias em relação aos vetores de peso;
Identificar os vetores de peso com as maiores distâncias em relação aos trechos, conforme o “quantil critério";
Retornar um indicador para as amostras que estão associadas a esses vetores;

4.3 PROCEDIMENTO DE RECONHECIMENTO

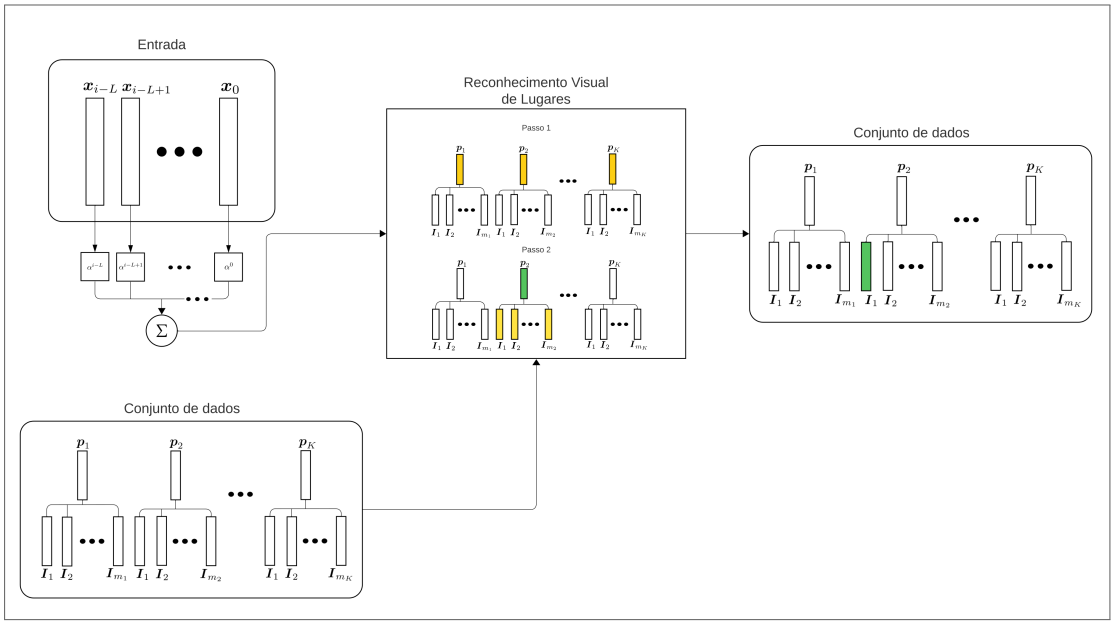
Para o reconhecimento, uma imagem de teste é apresentada, passando pelo filtro temporal para sua contextualização, *i.e.*, considerando não somente as características da imagem atual, mas de imagens próximas também. Assim, no passo 1, a imagem contextualizada é comparada com cada um dos protótipos, vetores de pesos associados aos nodos do LARFSOM-ARC que aparecem em amarelo na Figura 13, para determinar o nodo vencedor. No passo 2, o nodo vencedor (em verde), determina as imagens de treinamento (em amarelo na Figura 13) contidas na base de dados que pertencem ao grupo vencedor. Tais imagens serão comparadas com a imagem teste, de forma que o resultado final é a identificação da imagem do grupo vencedor que seja a mais parecida com a entrada. É conveniente lembrar que tanto a imagem de teste como os protótipos são todas imagens que passaram pelo filtro temporal.

Uma alternativa a comparar somente com as imagens do melhor nodo, é a comparação também com as imagens dos k melhores nodos (ou o melhor nodo e seus nodos vizinhos), na

busca de se obter uma melhor precisão à custa de acréscimo do tempo de execução.

As imagens de teste são apresentadas em ordem temporal, como aparecem em uma sequência ou em um trecho dela, embora a aprendizagem se dê em ordem aleatória. A identificação da imagem mais parecida com aquela de teste é sucedida pela avaliação da distância a posição geográfica desta imagem para a posição real de onde a imagem teste foi coletada para validar o reconhecimento do lugar como verdadeiro para fins de avaliar o desempenho. Portanto, o reconhecimento verdadeiro implica que a imagem teste está dentro de uma distância máxima admissível entre as duas imagens.

Figura 13 – Diagrama da tarefa de VPR do modelo desenvolvido. As entradas são: a sequência gerada pelo filtro, o conjunto de dados e a entrada atual. A saída é a indicação de qual imagem mais corresponde à entrada atual. No passo 1, as comparações são realizadas entre a saída do filtro e os pesos do nodo (indicados em amarelo), enquanto que no passo 2 as comparações são entre a entrada atual e as imagens agrupadas no nodo selecionado, sendo o nodo selecionado indicado em verde e as imagens indicadas em amarelo.



Fonte: Autor (2024).

5 EXPERIMENTOS

Neste capítulo, são descritos os experimentos realizados para a parametrização do modelo, como foram organizadas as bases de dados, assim como os testes realizados para validação do modelo. Para os testes, foram selecionados os modelos de comparação MPF(HAUSLER; JACOBSON; MILFORD, 2019) e o SeqSLAM(MILFORD; WYETH, 2012). A escolha do MPF se deve a seu desempenho e recência, enquanto que o SeqSLAM é um modelo bem consolidado. Além disso, o modelo proposto também foi comparado com o uso HybridNet sozinho para a tarefa de VPR. Essa rede foi já pré-treinada por CHEN et al. (2017), em uma base de dados contendo 2,5 milhões de imagens. Além disso, ela é também utilizada tanto no modelo proposto, quanto no MPF, o qual possui 4 diferentes formas de extração de características, como descrito no Capítulo 2 (SAD, HOG, HybridNet e HybridNet somente com as coordenadas das ativações máximas). Os experimentos compreenderam treinamento com as bases de dados completas e testes com sequência completa ou trechos de sequências treinadas que foram coletados em outros contextos (horário do dia, estação do ano, clima). Quando o treinamento se deu com trechos de sequências documentadas, os testes envolveram apenas sequências com mesmo comprimento que foram coletadas em diferentes contextos, como anteriormente.

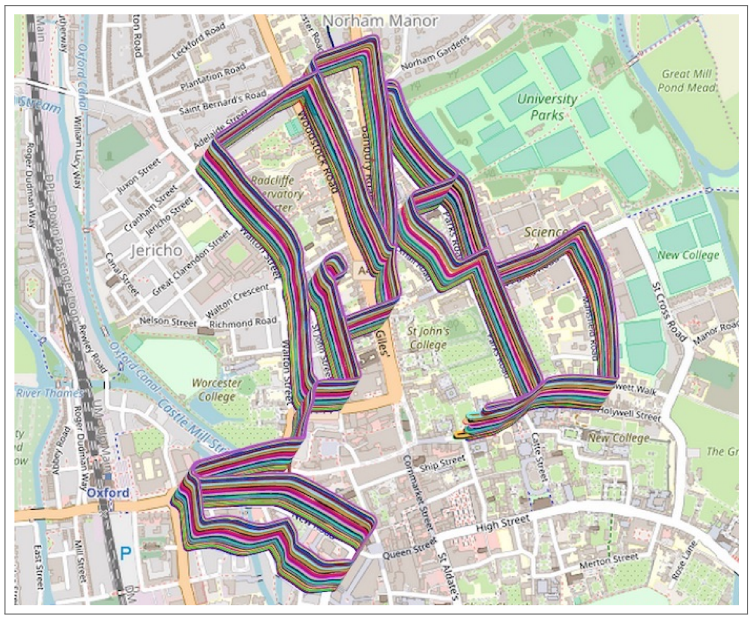
5.1 BASES DE DADOS

As bases de dados escolhidas foram Oxford Robotcar e St. Lucia, ambas comumente utilizadas em aplicações de VPR e SLAM. Oxford Robotcar foi a base em que foram determinados os melhores parâmetros para a rede, com fim de validar o modelo; enquanto que St. Lucia foi utilizada com a intenção de demonstrar a robustez do sistema em diferentes situações.

Oxford Robotcar (MADDERN et al., 2017) possui imagens coletadas de diferentes ângulos por dois tipos de câmeras acopladas a um carro: nas laterais e na traseira o tipo de câmera utilizada foi Glasshopper, coletando imagens de 1024x1024 pixels de resolução; enquanto que na frente, empregou-se uma Bumblebee XB3 com 3 lentes, sendo uma delas posicionada no centro do carro. Nessa base de dados há uma grande quantidade de imagens coletadas em uma mesma região (Figura 14) em diferentes situações tais como dia, noite, inverno, verão, chuva, ambiente nublado, com prédios em diferentes estágios de construção, com obras rodoviárias e com alguns desvios de rota. As imagens utilizadas para os experimentos foram coletadas

por meio da lente esquerda da câmera Bumblebee XB3 para as sequências 2014/12/02 (dia), 2015/02/13 (dia), 2014/12/10 (noite) e 2014/11/14 (noite). A Figura 15 ilustra imagens das sequências nessa base de dados utilizadas para os experimentos. A distância máxima considerada para o *ground truth* de todos experimentos realizados para essa base de dados é de 40 m (mesmo parâmetro de tolerância escolhido em HAUSLER; JACOBSON; MILFORD). Essa distância se refere à distância entra a posição estimada a partir do dado armazenado o qual mais corresponde ao dado atual e a posição atual.

Figura 14 – Ilustração dos dados de posição da trajetória percorrida para formar o base de dados Oxford Robotcar em 32 momentos diferentes, um sobreposto ao outro.



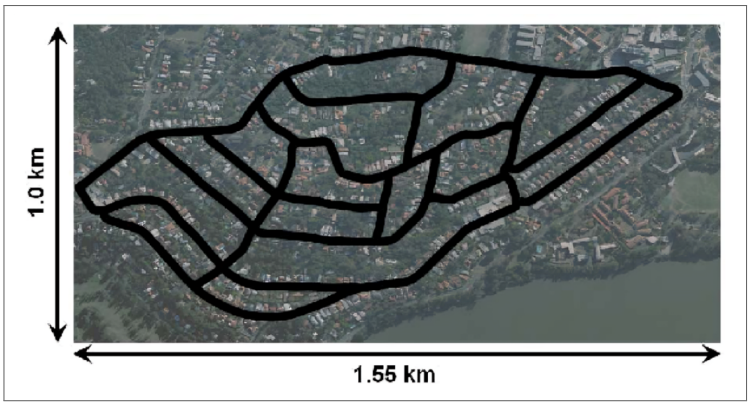
Fonte: BARNES et al. (2020).

Os dados da base St. Lucia (GLOVER et al., 2010a) foram coletados por uma câmera Logitech QuickCam Pro a 15 *frames* por segundo com uma resolução de 640x480 pixels, possuindo 5 sequências gravadas em vídeo em cinco dias diferentes nos seguintes horários: 8:45 am; 10 am; 12:10 pm; 2:10 pm; e 3:45pm. Assim, as sequências utilizadas para os experimentos foram 8:45 am e e 3:45 pm respectivamente. A trajetória em que as imagens foram coletadas pode ser vista na Figura 16 e imagens de diferentes pontos da trajetória são mostradas na Figura 17. A distância máxima considerada para o *ground truth* nessa base de dados é de 20 m (mesmo parâmetro definido em HAUSLER; JACOBSON; MILFORD).

Figura 15 – Cada linha ilustra *frames* coletados em diferentes momentos das sequências "2014/12/02", "2015/02/13", "2014/12/10" e "2014/11/14" respectivamente no *data set* Oxford Robotcar.

Fonte: Autor (2024).

Figura 16 – Ilustração dos dados de posição da trajetória percorrida para formar o base de dados St. Lucia.



Fonte: GLOVER et al. 2010b.

Figura 17 – Cada linha ilustra *frames* coletados em diferentes momentos das sequências "0845" e "1545" respectivamente no *data set* St. Lucia.

Fonte: Autor (2024).

5.2 CONFIGURAÇÃO EXPERIMENTAL

O modelo desenvolvido foi implementado em MATLAB e o Hybridnet foi utilizado para a extração de características, o qual pode ser obtido por meio do *download* de uma rede pré-treinada por meio deste site. O computador utilizado para os testes possui 78 gigabytes de memória RAM e processador AMD Ryzen 7 5700X, com 8 núcleos e 16 *threads*.

Para sintonizar os parâmetros do modelo, foi escolhido o subconjunto "2014/12/02" da base de dados Oxford Robotcar para treinar a rede. Já para o teste, o subconjunto "2014/12/10", da mesma base foi selecionado. Em ambos os casos, 2050 imagens foram selecionadas, iniciando na imagem número 450 para a sequência de treinamento e na imagem número 1750 para a sequência de teste, os inícios foram estabelecidos aleatoriamente. Os *frames* coletados possuem intervalo de 3 em 3 imagens da base de dados.

Dessa forma, os parâmetros foram sendo definidos no decorrer de 3 etapas, levando em consideração que vários experimentos já ocorreram até então, fornecendo uma boa ideia dos melhores intervalos para cada parâmetro. Na primeira etapa, utilizou-se o Hipercubo Latino(LOH, 1996) para estimar: o limiar de ativação inicial a_{t_0}; o limiar para o erro percentual que foi utilizado para determinar quando a fase de mudança dos limiares poderia ser utilizada p_{err}; assim como o número de épocas em que o erro não-supervisionado poderia ficar abaixo desse limiar sem que ocorresse mudança de limiares seguida de retreinamento n_{err}.

A partir da média e desvio padrão de cada métrica considerada, um teste estatístico não paramétrico foi realizado para a determinação dos parâmetros a_{t_0}, p_{err} e n_{err}. Nesse caso, foi optado pelo teste estatístico da soma de postos de Wilcoxon com um nível de significância de 5%. Assim, os valores para os parâmetros citados foram de "$0,921$", "$0,089$" e "18" respectivamente.

Após isso, foram variados os quantis que são empregados para determinar novos valores para os limiares dos nodos selecionados para tal. Há dois critérios testados para seleção dos nodos com limiares a serem alterados, sendo necessária mais duas etapas para cada um. Um dos critérios é a quantidade de amostras, pois um nodo com muitas amostras ali agrupadas tende a tornar mais lento o reconhecimento dos lugares, pois aumenta o número de comparações. Os experimentos indicaram que o valor para o quantil desse critério foi definido como sendo "$0,890$", enquanto que o quantil de ativação foi "$0,750$". O segundo critério considera a quantidade de amostras categorizadas erradamente em cada nodo, portanto, os nodos com maiores quantidades de amostras categorizadas erradamente têm seus limiares ajustados. Neste

caso, o quantil de ativação foi "0,75", e o quantil de critério foi de "$0,085$", considerando dados de treino, "$0,080$", considerando dados de teste e "$0,080$", considerando dados de treinamento corrompido.

Recapitulando, a trajetória de treinamento é separada em diversos trechos, por meio dos dados de posição das amostras. Assim, os nodos tem seus trechos definidos pela moda dos trechos das amostras agrupadas nele. Logo, um agrupamento é considerado errado se o trecho do nodo e da imagem agrupada forem diferentes. Além disso, ruído (*salt and pepper*) foi inserido nos dados de treinamento para estudar o efeito dessas perturbações no treinamento e no teste. Este teste de robustez foi empregado para verificar a tolerância do modelo a este tipo de ruído.

5.3 ANÁLISE DE RESULTADOS

Com os parâmetros definidos, experimentos para analisar as diferentes configurações de critério de alteração do limiar do modelo foram realizados inicialmente para as mesmas subsequências utilizadas anteriormente (das sequências 2014/12/02 e 2014/12/10 ou sequências de "Dia" e "Noite"). Esses experimentos compreendem em criar uma versão "corrompida" ("Dia corr.") dos dados de treino (2014/12/02) e testas todas possíveis combinações possíveis entre essas 3 subsequências para se ganhar conhecimento sobre o funcionamento do modelo proposto. Na Tabela 1, são mostradas as médias de precisão resultantes de cada combinação dos experimentos realizados. Foram considerados três conjuntos de dados: treino, treino corrompido e teste. O primeiro compreende os dados de trechos de sequências selecionados para treinamento, o segundo engloba estes dados com inserção de ruídos e o terceiro os dados separados para testes de desempenho. Estes últimos diferem do primeiro grupo por serem coletados em outro período com diferenças de luminosidade e nitidez na imagem além de diferenças dos pontos de tomada das imagens. Para esses primeiros experimentos, cada um dos conjuntos é empregado para treinar a rede que é posteriormente testado com os três conjuntos. O objetivo foi avaliar como o conjunto usado para treinamento repercutiria no desempenho do modelo proposto.

Na Tabela 1, cada sub-tabela ("Dia", "Dia corr." e "noite") informa sobre resultados obtidos em diferentes situações e corresponde a que tipos de dados foram utilizados para treinamento: "Dia", "Dia corr." ou " Noite". Em cada sub-tabela, as linhas indicam os tipos de dado que foram utilizados para teste, enquanto que as colunas indicam as formas de critério de

alteração de limiar que foram utilizados. Nesse caso, na primeira coluna não ocorrem alterações dos limiares dos nodos, correspondendo ao LARFSOM, com alteração na poda; enquanto que a segunda, terceira e quarta colunas correspondem aos tipos de dados que são usados para realizar a contagem dos erros com a finalidade de escolher os nodos que terão seus limiares alterados; e, por fim, a última coluna considera o critério pelo número de amostras agrupadas no nodo, ao invés do número de erros. Nessa tabela, assim como em todas as tabelas deste capítulo, todas as células que apresentam dois dados separados por uma barra indicam que o resultado se trata de uma média de 15 experimentos, mostrada à esquerda, assim como seu desvio padrão, mostrado a direita.

Pode-se ver que quando se usa os dados de treinamento íntegros e os dados de dia corrompidos, um para aprender e outro para testar, a precisão é superior a 90% na maioria dos casos. Além disso, para a escolha dos nodos que alteram seus limiares de ativação, o emprego do erro não-supervisionado produz melhores resultados que a quantidade de amostra erradamente categorizadas.

Contudo, de maneira inesperada, ao se utilizar os dados de dia corrompidos para contabilizar o erro para seleção dos nodos que teriam seus limiares alterados, os resultados obtidos foram melhores do que utilizando os dados de noite para essa contabilização, no caso em que os dados de noite foram utilizados para treinar a rede. Além disso, usar dados corrompidos produziu resultados melhores do que utilizar dados de dia em todos os casos, com exceção do caso no qual os dados de dia foram utilizados para realização do teste, enquanto que os de dia corrompido foram utilizados para o treino. Tendo isso em vista, um padrão em geral que pode se enxergar dessa tabela, é que utilizar dados que trazem características diferentes dos dados de treino fazem com que o desempenho sempre tenha uma pequena melhora.

Os testes incluíram os trechos de sequências com dados coletados de dia e de noite, também foram testadas combinações entre duas subsequências de aquisição diurna (Tabela 2). Busca-se avaliar se a o reconhecimento melhora quando se lida com dados adquiridos em contextos semelhantes. Para o "Dia 1", selecionou-se a subsequência 2014/12/02, enquanto que para o "Dia 2", escolheu-se a subsequência de 2015/02/13. A imagem inicial de cada subsequência foi a de número 1000 e 1750 respectivamente. Enquanto isso, a quantidade de imagens totaliza 2050 e 2550 para cada uma dos trechos e sequência respectivamente. Como no caso anterior, foram utilizadas imagens íntegras, imagens corrompidas e imagens de teste. Utilizar os dados corrompidos para o processo de alteração do limiar continuou apresentando os melhores resultados em todos os casos, embora de forma sutil, perdendo para a utilização

Tabela 1 – Precisão para diferentes combinações de experimentos entre as subsequências 2014/12/02 (dia) e 2014/12/10 (noite).

Dia						
Alteração do limiar por:		Sem alteração	Noite	Dia	Dia corr.	Amostras
Dados experimentais	Noite	50,66%/4,5%	**61,29%**/3,21%	60,63%/2,83%	60,79%/3,17%	56,27%/3,96%
	Dia	100%/0%	100%/0%	100%/0%	100%/0%	100%/0%
	Dia corr.	89,71%/1,84%	**95,74%**/0,83%	94,69%/1,19%	95,21%/0,36%	94,46%/1,22%
Dia corr.						
Alteração do limiar por:		Sem alteração	Noite	Dia	Dia corr.	Amostras
Dados experimentais	Noite	29,14%/2,79%	**36,83%**/5,53%	36,07%/4,85%	35,69%/4,13%	35,62%/5,51%
	Dia	77,13%/5,66%	94,43%/0,98%	**95,48%**/0,71%	95,27%/0,73%	87,51%/2,81%
	Dia corr.	100%/0%	100%/0%	100%/0%	100%/0%	100%/0%
Noite						
Alteração do limiar por:		Sem alteração	Noite	Dia	Dia corr.	Amostras
Dados experimentais	Noite	100%/0%	100%/0%	100%/0%	100%/0%	100%/0%
	Dia	37,7%/1,59%	43,25%/0,68%	42,76%/0,69%	**43,99%**/0,7%	42,2%/1,61%
	Dia corr.	27,17%/2,65	31,16%/1,32%	31,33%/1,54%	**31,81%**/1,46%	30,99%/2,15%

Fonte: Autor (2024).

Tabela 2 – Precisão para diferentes combinações de experimentos entre as subsequências 2014/12/02 (Dia 1) e 2015/02/13 (Dia 2).

Dia 1						
Alteração do limiar por:		Sem alteração	Dia 2	Dia 1	Dia 1 corr.	amostras
Dados experimentais	Dia 2	93,22%/1,62%	**98,45%**/0,46%	97,98%/0,33%	98,25%/0,36%	96,18%/1,93%
	Dia	100%/0%	100%/0%	100%/0%	100%/0%	100%/0%
	Dia 1 corr.	89,80%/3,88%	**96,11%**/0,55%	95,14%/1,40%	96,49%/0,67%	89,70%/1,05%
Dia 1 corr.						
Alteração do limiar por:		amostras	Dia 2	Dia	Dia 1 corr.	sem alteração
Dados experimentais	Dia 2	70,17%/6,59%	**85,80%**/0,83%	85,79%/0,33%	86,86%/1,38%	80,36%/1,92%
	Dia	75,91%/8,82%	94,60%/0,86%	94,60%/0,42%	**95,27%**/0,95%	88,47%/1,52%
	Dia 1 corr.	100%/0%	100%/0%	100%/0%	100%/0%	100%/0%
Dia 2						
Alteração do limiar por:		amostras	Dia 2	Dia	Dia 1 corr.	sem alteração
Dados experimentais	Dia 2	100%/0%	100%/0%	100%/0%	100%/0%	100%/0%
	Dia	91,64%/1,61%	94,02%/0,68%	94,15%/0,69%	**94,22%**/0,70%	88,35%/1,59%
	Dia 1 corr.	55,47%/6,36%	60,71%/3,71%	61,19%/4,74%	**61,42%**/3,44%	50,55%/5,40%

Fonte: Autor (2024).

do "Dia 2" no caso em que a subsequência de 2014/12/02 foi utilizada para o treinamento e a subsequência de 2015/02/13 foi utilizada para o teste.

Por fim, a Tabela 3 informa resultados quando os dados para aprendizagem e teste foram coletados a noite. A ideia é avaliar o quanto de informação se consegue capturar com dados coletados com pouca iluminação (subsequências 2014/11/14 e 2014/12/10 para "Noite 1" e "Noite 2" respectivamente). A organização dos experimentos é a mesma daquela apresentada nas duas tabelas anteriores. Nesse caso, as subsequências se iniciam na imagem de número 8750 e 9150 respectivamente, enquanto que a quantidade de imagens para cada uma delas é de

Tabela 3 – Precisão para diferentes combinações de experimentos entre as sequências 2014/11/14 (Noite 1) e 2014/12/10 (Noite 2).

Noite 1						
Alteração do limiar por:		Sem alteração	Noite 2	Noite 1	Noite 1 corr.	amostras
Dados experimentais	Noite 2	25,76%/4,67%	27,03%/0,62%	**27,51%**/2,43%	26,55%/2,33%	24,24%/3,84%
	Noite 1	100%/0%	100%/0%	100%/0%	100%/0%	100%
	Noite 1 corr.	**12,03%**/7,54%	8,3%/2,22%	10,57%/3,2%	9,8%/1,63%	10,13%/1,09%
Noite 1 corr.						
Alteração do limiar por:		Sem alteração	Noite 2	Noite 1	Noite 1 corr.	amostras
Dados experimentais	Noite 2	**17,75%**/0,16%	14,11%/7,83%	16,5%/0,19%	17,61%/0,21%	**17,75%**/0,20%
	Noite 1	16,07%/3,43%	**17,62%**/1,39%	16,03%/3,84%	15,37%/3,4%	16,78%/2,16%
	Noite 1 corr.	100%/0%	100%/0%	100%/0%	100%/0%	100%
Noite 2						
Alteração do limiar por:		Sem alteração	Noite 2	Noite 1	Noite 1 corr.	amostras
Dados experimentais	Noite 2	100%/0%	100%/0%	100%/0%	100%/0%	100%
	Noite 1	6,49%/1,82%	6,16%/1,47%	6,99%/0,69%	**19,25%**/1,44%	15,93%/0,95%
	Noite 1 corr.	8,75%/2,11%	9,46%/1,38%	9,32%/1,11%	**10,58%**/1,49%	10,38%/2,04%

Fonte: Autor (2024).

2050. A Tabela 3 mostra uma queda drástica de desempenho em todos os casos, sugerindo que o extrator empregado não atua adequadamente para dados adquiridos à noite. Cabe ressaltar que dados de treinamento adquiridos durante o dia são mais efetivos para testes com dados noturnos. Esta é mais uma propriedade interessante do modelo proposto que como sempre há um vencedor, ele atua bem mesmo com poucas pistas, expressas por algumas características que são comuns entre os dados de treinamento e de teste.

A Figura 18 ilustra a relação da trajetória de referência com a trajetória de busca em que as coordenadas em amarelo representam imagens que obedecem a tolerância, sendo roxas caso contrário. Essa figura mostra 5 subsequências, em que as três primeiras estão relacionadas com os experimentos realizados anteriormente, enquanto que as duas últimas são referentes a testes em que o veículo esteve parado ou se movendo mais lentamente durante múltiplas ocasiões e quando houve várias curvas. É importante que a subsequência de busca selecionada sempre tenha uma imagem que atenda a tolerância entre os dados de treinamento, para garantir que a precisão máxima que pode ser alcançada seja de 100%. Além disso, borrões retangulares indicam que o veículo parou em ambas as situações de busca e referência, enquanto que traços fora da trajetória principal indicam que o mesmo lugar foi visitado mais de uma vez. Nos casos dos experimentos anteriores, foi necessário que isso ocorresse tanto para os dados de treinamento, quanto para os dados de teste, visto que houve alternações entre qual sequência seria usada para teste ou treinamento.

A Tabela 4 ilustra os resultados de precisão. As sequências de 1 a 5 são os mesmos

trechos referidos na Figura 18, sendo as sequências 1, 4 e 5 as sequências dos 3 primeiros experimentos mostrados anteriormente. MPF se sobressaiu em todos os casos. O SeqSLAM é o pior em desempenho devido à sua forma de comparar as imagens, fazendo com que mudanças de ponto de vista impactem muito em seus resultados. Além disso, nas linhas 4 e 5 da tabela são mostrados os resultados considerando as representações das imagens agrupadas também nos vizinhos do vencedor, assim como os vizinhos dos vizinhos respectivamente. Assim, para o LARFSOM-ARC é visto que a utilização das vizinhanças não causou alteração nos resultados do modelo.

Tabela 4 – Comparações de desempenho para trechos de sequência no Oxford Robotcar.

	Seq. 1	Seq. 2	Seq. 3	Seq. 4	Seq. 5
SeqSLAM	20,54%	23,67%	32%	0%	13,50%
Hybridnet	59,41%	59,08%	79,02%	20,54%	98,35%
MPF	**76,50%**	**67,05%**	**84,78%**	14,29%	**98,53%**
LARFSOM-ARC	65,80%	60,07%	84,13%	**20,93%**	98,39%
LARFSOM-ARC 1º vizinhança	65,07%	60,07%	84,13%	**20,93%**	98,39%
LARFSOM-ARC 2º vizinhança	65,80%	60,07%	84,13%	**20,93%**	98,39%

Fonte: Autor (2024).

Embora o LARFSOM-ARC não tenha sido o melhor modelo quanto às precisões, ele se foi bem mais rápido que todos os demais algoritmos como mostrado na Tabela 5.

Tabela 5 – Comparações de tempo para trechos de sequência no Oxford Robotcar.

	Seq. 1	Seq. 2	Seq. 3	Seq. 4	Seq. 5
SeqSLAM	16,09 s	5,33 s	1,30 s	15,20 s	9,31 s
Hybridnet	20,38 s	7,30 s	0,99 s	18,63 s	23,34 s
MPF	175,23 s	43,86 s	11,66 s	203,73 s	238,22 s
LARFSOM-ARC	**2,33** s	**1,25** s	**0,64** s	**2,35** s	**2,96** s
LARFSOM-ARC 1º vizinhança	2,64 s	1,42 s	0,75 s	5,95 s	3,62 s
LARFSOM-ARC 2º vizinhança	3,01 s	1,53 s	0,83 s	8,50 s	4,00 s

Fonte: Autor (2024).

Para o escore F1, diferentemente da precisão, há uma diferença pequena entre o LARFSOM-ARC e o MPF de uma maneira geral, como mostrado na Tabela 6. Além disso, o LARFSOM-ARC apresenta melhores resultados para as subsequências 2 e 5.

O valor máximo do *evocação* para uma precisão de 100%, assim como os gráficos de precisão-evocação (Figura 19), são interessantes em casos de fechamento de laço e reloca-

Tabela 6 – Comparações de F1 máximo para trechos de sequência no Oxford Robotcar.

	Seq. 1	Seq. 2	Seq. 3	Seq. 4	Seq. 5
SeqSLAM	56,57%	47,98%	57,37%	4,00%	42,72%
Hybridnet	78,67%	74,52%	89,06%	64,48%	99,17%
MPF	**86,69%**	80,28%	**91,76%**	**47,54%**	99,26%
LARFSOM-ARC	81,23%	**83,45%**	91,32%	34,54%	**99,23%**
LARFSOM-ARC 1º vizinhança	82,59%	**83,45%**	91,32%	34,54%	**99,23%**
LARFSOM-ARC 2º vizinhança	81,23%	**83,45%**	91,32%	34,54%	**99,23%**

Fonte: Autor (2024).

lização em tarefa de SLAM, visto que nessas situações, a ocorrência de algum erro pode comprometer o mapeamento todo. Nessa tabela vemos o LARFSOM-ARC se destacar na maioria dos casos, perdendo somente para o MPF na última coluna.

Tabela 7 – Comparações de evocação para trechos de sequência no Oxford Robotcar.

	Seq. 1	Seq. 2	Seq. 3	Seq. 4	Seq. 5
SeqSLAM	16,79%	14,59%	5,90%	0%	2,92%
Hybridnet	32,10%	22,69%	14,72%	0%	79,39%
MPF	21,18%	14,96%	22,00%	0%	**96,36%**
LARFSOM-ARC	**40,80%**	**28,11%**	**23,48%**	0%	84,43%
LARFSOM-ARC 1º vizinhança	39,90%	**28,11%**	**23,48%**	0%	84,43%
LARFSOM-ARC 2º vizinhança	**40,80%**	**28,11%**	**23,48%**	0%	84,43%

Fonte: Autor (2024).

Os gráficos de precisão-evocação comparando os diferentes modelos são mostrados na Figura 19, aparentemente seguindo o mesmo padrão de resultados para as métricas de F1 máximo e precisão. Esses gráficos são gerados considerando a variação de um parâmetro de confiabilidade, i.e., como cada correspondência estabelecida entre imagem de referência e busca tem uma distância já calculada, quão mais próxima ela for, maior as chances dessa correspondência ser correta. Assim, é possível definir um limiar que define valores aceitáveis de distância, gerando assim verdadeiros e falsos negativos, visto que com a utilização desse limiar algumas correspondências são certamente ou erroneamente rejeitadas. Dessa forma, com o surgimento de verdadeiros e falsos negativos, é possível calcular também a evocação, havendo um valor específico de precisão e evocação para cada valor do limiar, sendo possível gerar os gráficos abaixo.

Visto que a consideração dos vizinhos para a tarefa de reconhecimento para o LARFSOM-ARC não causou uma melhora nos resultados, é apresentado aqui a Tabela 8, que mostra resultados para todas as subsequências considerando, ao invés dos vizinhos, os k nodos mais próximos da imagem de teste. Nesse caso, a subsequência 4 não foi utilizada, visto que a quantidade de nodos gerados na rede foi muito pequena, com média de aproximadamente 6 nodos, para os 15 casos de testes realizados nessa subsequência, em alguns dos experimentos menor do que o número mínimo de k considerado.

Tabela 8 – Tabela para as 4 subsequências, comparando resultados do uso de mais de 1 nodo vencedor para a tarefa de reconhecimento.

Seq. 1							
	Precisão (média)	Precisão (máx.)	Tempo (s)	F1 máx. (média)	F1 máx. (máx.)	Evocação (média)	Evocação (máx.)
k1	60,63%/2,83%	65,81%	2,33/0,11	81,23%/1,44%	81,23%	30,70%/5,25%	40,80%
k3	60,63%/2,83%	65,81%	2,29/0,04	77,60%/2,01%	81,55%	7,24%/1,42%	9,46%
k5	60,63%/2,83%	77,60%	2,28/0,04	77,60%/2,01%	81,55%	7,24%/1,42%	9,46%
k10	60,63%/2,83%	65,81%	2,29/0,05	77,60%/2,01%	81,55%	7,24%/1,42%	9,46%
Seq. 2							
	Precisão (média)	Precisão (máx.)	Tempo (s)	F1 máx. (média)	F1 máx. (máx.)	Evocação (média)	Evocação (máx.)
k1	58,34%/1,19%	60,07%	1,25/0,05	83,45%/1,50%	83,45%	21,09%/5,60%	28,11%
k3	58,34%/1,19%	60,07%	1,24/0,03	79,46%/1,10%	81,16%	4,19%/0,26%	4,60%
k5	58,34%/1,19%	79,46%	1,22/0,02	79,46%/1,10%	81,16%	4,19%/0,26%	4,60%
k10	58,34%/1,19%	60,07%	1,24/0,04	79,46%/1,10%	81,16%	4,19%/0,26%	4,60%
Seq. 3							
	Precisão (média)	Precisão (máx.)	Tempo (s)	F1 máx. (média)	F1 máx. (máx.)	Evocação (média)	Evocação (máx.)
k1	80,19%/1,51%	84,13%	0,64/0,02	91,32%/0,81%	91,32%	16,58%/4,21%	23,48%
k3	80,19%/1,51%	84,13%	0,62/0,02	88,99%/0,90%	91,32%	5,87%/0,19%	6,10%
k5	80,19%/1,51%	88,99%	0,62/0,02	88,99%/0,90%	91,32%	5,87%/0,19%	6,10%
k10	80,19%/1,51%	84,13%	0,62/0,01	88,99%/0,90%	91,32%	5,87%/0,19%	6,10%
Seq. 5							
	Precisão (média)	Precisão (máx.)	Tempo (s)	F1 máx. (média)	F1 máx. (máx.)	Evocação (média)	Evocação (máx.)
k1	97,94%/0,35%	98,39%	2,96/0,19	99,23%/0,09%	99,23%	72,01%/4,64%	84,43%
k3	97,94%/0,35%	98,39%	2,90/0,04	99,09%/0,09%	99,23%	72,01%/4,64%	84,43%
k5	97,94%/0,35%	99,09%	2,90/0,05	99,09%/0,09%	99,23%	72,01%/4,64%	84,43%
k10	97,94%/0,35%	98,39%	2,89/0,07	99,09%/0,09%	99,23%	72,01%/4,64%	84,43%

Fonte: Autor (2024).

As subsequências de busca foram aleatoriamente escolhidas, cada uma correspondente a cada coluna das tabelas de resultados, sendo a última correspondente ao percurso completo da base de dados, assim como a sequência de treinamento (ou simplesmente sequência de referência para os outros modelos, já que eles não possuem treinamento). A Figura 20 mostra a relação entre a trajetória de referência, dessa vez sendo a sequência completa, com a trajetória de busca, uma subsequência, sendo os pixels amarelos representantes de um par de imagem de referência com imagem de busca que respeitaram a tolerância, os pixels roxos violam a tolerância. Já Figura 21 mostra a mesma relação, contudo para sequências do St. Lucia, ao

invés do Oxford Robotcar, o qual foi considerado para todos os experimentos até aqui.

Dessa vez, os casos considerados são os casos em que a sequência completa foi utilizada para o treinamento. Além disso, anteriormente, os dados de treino e teste eram referentes a uma mesma trajetória, todavia, agora o teste representa somente um pedaço da trajetória em 4 dos 5 casos, somente correspondendo a mesma trajetória no quinto caso. Vale ressaltar que para esses novos casos de trajetórias mais longas, o SeqSLAM e MPF apresentaram problemas em sua execução devido ao uso de RAM (de 16 GB), sendo necessário um aumento da capacidade para a realização desses experimentos. Para a sequência de referência (utilizada para o processo de treinamento do LARFSOM-ARC), foram realizados treinando a rede na base de dados completo de 34128 *frames*, coletando um *frame* a cada 3, sendo coletadas 11376 imagens, assim como na base de dados St. Lucia, sendo um total de 21100 *frames*, coletando os *frames* de 2 em 2, sendo um total de 10550 imagens coletadas. Assim, as subsequências da Figura 20 são referentes ao Oxford Robotcar, enquanto que da Figura 21 são referentes ao St. Lucia. Para o Oxford Robotcar foram consideradas as sequências 2014/12/02 (dia) e 2014/12/10 (noite) respectivamente, enquanto que para o St. Lucia, foram consideradas as sequências 0845 e 1545 para treinamento e testes respectivamente.

A Tabela 9 mostra as precisões para os diferentes modelos, sendo que dessa vez considerando a sequência completa para o treinamento na base de dados Oxford Robotcar. Dessa vez é observado um desempenho melhor do LARFSOM-ARC na subsequência 6 com relação a todos os outros modelos, mesmo o MPF.

Tabela 9 – Precisão para os diferentes modelos no *dataset* Oxford Robotcar.

	Seq. 6	Seq. 7	Seq. 8	Seq. 9	Seq. Completa
SeqSLAM	8,77%	14,03%	11,11%	13,10%	8,98%
Hybridnet	37,71%	53,51%	58,93%	64,63%	44,52%
MPF	41,63%	**59,78%**	**63,25%**	**75,57%**	**48,81%**
LARFSOM-ARC	**42,63%**	58,88%	59,46%	67,80%	45,81%
LARFSOM-ARC 1º vizinhança	**42,63%**	58,88%	59,46%	67,80%	45,81%
LARFSOM-ARC 2º vizinhança	**42,63%**	58,88%	59,46%	67,80%	45,81%

Fonte: Autor (2024).

Também nesse experimento, o tempo de execução ainda continua ordens de grandeza abaixo, sendo mais do que 100 vezes a razão entre o LARFSOM-ARC e o MPF, reflexo da menor complexidade computacional do modelo, como pode ser visto na Tabela 10. Contudo, mesmo nesses casos o uso da vizinhança ainda não apresentou nenhuma melhora.

Tabela 10 – Tempo de execução para os diferentes modelos no *dataset* Oxford Robotcar.

	Seq. 6	Seq. 7	Seq. 8	Seq. 9	Seq. Completa
SeqSLAM	152,61 s	153,20 s	225,81 s	224,64 s	4533,31 s
Hybridnet	272,47 s	267,00 s	263,84 s	265,77 s	1694,19 s
MPF	3576,23 s	3890,25 s	3860,48 s	3298,63 s	22200,01 s
LARFSOM-ARC	**32,18 s**	**32,94 s**	**50,85 s**	**51,82 s**	**197,07 s**
LARFSOM-ARC 1º vizinhança	53,16 s	54,02 s	51,00 s	57,97 s	250,97 s
LARFSOM-ARC 2º vizinhança	57,42 s	55,86 s	53,71 s	57,07 s	260,65 s

Fonte: Autor (2024).

Por sua vez, o máximo escore do LARFSOM-ARC apresentou um resultado melhor para todos os casos, como mostrado na Tabela 11.

Tabela 11 – Máximo escore F1 para os diferentes modelos no *dataset* Oxford Robotcar.

	Seq. 6	Seq. 7	Seq. 8	Seq. 9	Seq. Completa
SeqSLAM	39,09%	53,83%	53,01%	55,63%	49,37%
Hybridnet	68,48%	69,81%	74,20%	78,55%	65,17%
MPF	58,78%	**74,88%**	**77,49%**	**86,08%**	65,61%
LARFSOM-ARC	**70,86%**	74,43%	74,65%	80,93%	**67,70%**
LARFSOM-ARC 1º vizinhança	**70,86%**	74,43%	74,65%	80,93%	**67,70%**
LARFSOM-ARC 2º vizinhança	**70,86%**	74,43%	74,65%	80,93%	**67,70%**

Fonte: Autor (2024).

Contudo, o *evocação* nesses casos foi maior para o SeqSLAM, como mostrado na Tabela 12.

Tabela 12 – Evocação para os diferentes modelos no dataset Oxford Robotcar.

	Seq. 6	Seq. 7	Seq. 8	Seq. 9	Seq. Completa
SeqSLAM	10,67%	**25,61%**	6,61%	4,89%	12,05%
Hybridnet	12,81%	14,13%	7,86%	11,55%	7,02%
MPF	2,27%	4,53%	2,96%	20,94%	2,94%
LARFSOM-ARC	**24,17%**	23,65%	**12,73%**	**13,77%**	**11,26%**
LARFSOM-ARC 1º vizinhança	**24,17%**	23,65%	**12,73%**	**13,77%**	**11,26%**
LARFSOM-ARC 2º vizinhança	**24,17%**	23,65%	**12,73%**	**13,77%**	**11,26%**

Fonte: Autor (2024).

Os *plots* de precisão-evocação podem ser vistos na Figura 22, as quais mostram uma clara competitividade entre o LARFSOM-ARC e outros modelos.

Os bons resultados no St. Lucia indicam capacidade de generalização do modelo a diferentes situações. A Tabela 13 mostra resultados de precisão para os diferentes casos considerados no St. Lucia.

Tabela 13 – Precisão para os diferentes modelos no data set St. Lucia.

	Seq. 10	Seq. 11	Seq. 12	Seq. 13	Seq. Completa
SeqSLAM	74,31%	81,77%	73,65%	62,24%	71,36%
Hybridnet	83,66%	92,29%	85,61%	79,66%	81,38%
MPF	**92,86%**	**98,55%**	**94,98%**	**91,66%**	**93,10%**
LARFSOM-ARC	83,71%	92,59%	85,41%	79,90%	80,65%
LARFSOM-ARC 1º vizinhança	83,71%	92,59%	85,41%	79,90%	80,65%
LARFSOM-ARC 2º vizinhança	83,71%	92,59%	85,41%	79,90%	80,65%

Fonte: Autor (2024).

Novamente, os tempos de execução para a tarefa de reconhecimento no LARFSOM-ARC são ordens de grandeza menores do que o MPF e uma orgem de grandeza a menos do que o SeqSLAM para a sequência completa, conforme a Tabela 14.

Tabela 14 – Tempo de execução para os diferentes modelos no data set St. Lucia.

	Seq. 10	Seq. 11	Seq. 12	Seq. 13	Seq. Completa
SeqSLAM	135,54	134,90	133,90	134,30	2671,27
Hybridnet	245,36	252,58	246,11	252,05	1184,42
MPF	3448,78	3647,20	3409,60	3552,77	15738,12
LARFSOM-ARC	**52,18**	**51,84**	**56,41**	**53,83**	**277,87**
LARFSOM-ARC 1º vizinhança	59,17	60,18	58,00	59,45	287,54
LARFSOM-ARC 2º vizinhança	58,22	60,12	59,38	58,41	285,37

Fonte: Autor (2024).

Contudo, quanto ao máximo escore de F1, o MPF continua com o melhor desempenho entre todos os modelos, como ilustrado na Figura 15.

Pela primeira vez entre todos os casos, o LARFSOM-ARC mostra resultados melhores na maioria das trajetórias, perdendo somente nas duas últimas colunas (Tabela 16).

A Figura 23 mostra que todos os modelos em geral melhoraram suas curvas de precisão-evocação, devido ao horário em que os dados foram coletados para esta base de dados. Em geral, o LARFSOM-ARC se mantém competitivo em meio aos outros modelos.

Por fim, como anteriormente a utilização dos k melhores nodos não teve bons resultados, foi realizados novos experimentos da mesma forma, contudo com as sequência de 6 à 9, i.e.,

Tabela 15 – Máximo escore F1 para os diferentes modelos no data set St. Lucia.

	Seq. 10	Seq. 11	Seq. 12	Seq. 13	Seq. Completa
SeqSLAM	86,07%	90,06%	85,02%	78,67%	83,84%
Hybridnet	91,45%	96,04%	92,34%	88,73%	89,73%
MPF	**96,38%**	**99,33%**	**97,47%**	**95,74%**	**96,42%**
LARFSOM-ARC	91,50%	96,17%	92,25%	88,88%	89,29%
LARFSOM-ARC 1º vizinhança	91,50%	96,17%	92,25%	88,88%	89,29%
LARFSOM-ARC 2º vizinhança	91,50%	96,17%	92,25%	88,88%	89,29%

Fonte: Autor (2024).

Tabela 16 – Evocação para os diferentes modelos no data set St. Lucia.

	Seq. 10	Seq. 11	Seq. 12	Seq. 13	Seq. Completa
SeqSLAM	32,80%	2,35%	20,07%	12,42%	3,59%
Hybridnet	40,29%	31,92%	24,05%	17,58%	9,19%
MPF	21,67%	19,95%	18,47%	**27,09%**	**22,77%**
LARFSOM-ARC	**40,83%**	**44,60%**	**23,97%**	22,26%	15,93%
LARFSOM-ARC 1º vizinhança	**40,83%**	**44,60%**	**23,97%**	22,26%	15,93%
LARFSOM-ARC 2º vizinhança	**40,83%**	**44,60%**	**23,97%**	22,26%	15,93%

Fonte: Autor (2024).

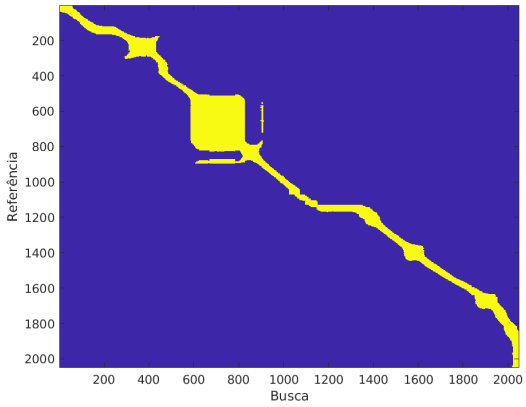
considerando a sequência completa para o treinamento na base de dados Oxford Robotcar, sendo os resultados mostrados na Tabela 17. Nessa tabela, embora os tempos aumentem conforme um maior número de nodos é considerado, a precisão e desempenho em geral não melhoram em média, com exceção de alguns casos para $k = 5$.

Tabela 17 – Tabela para todas as sequências consideradas para a base de dados Oxford Robotcar, comparando os resultados do uso de mais de 1 nó vencedor para a tarefa de reconhecimento.

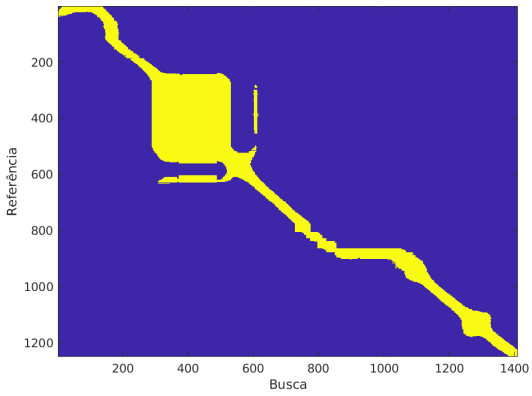
Seq. 6							
	Precisão (média)	Precisão (máx.)	Tempo (s)	F1 máx. (média)	F1 máx. (máx.)	Evocação (média)	Evocação (máx.)
k1	38,16%/3,40%	42,63%	32,18/1,75	70,86%/2,61%	70,86%	21,51%/1,77%	24,17%
k3	38,16%/3,40%	42,63%	54,08/4,35	73,68%/1,56%	75,74%	0,13%/0,01%	0,14%
k5	38,16%/3,40%	73,68%	53,06/3,94	73,68%/1,56%	75,74%	0,13%/0,01%	0,14%
k10	38,16%/3,40%	42,63%	54,56/2,20	73,68%/1,56%	75,74%	0,13%/0,01%	0,14%
Seq. 7							
	Precisão (média)	Precisão (máx.)	Tempo (s)	F1 máx. (média)	F1 máx. (máx.)	Evocação (média)	Evocação (máx.)
k1	53,82%/3,08%	58,88%	32,94/2,30	74,43%/2,47%	74,43%	21,98%/1,33%	23,65%
k3	53,82%/3,08%	58,88%	50,04/4,29	73,93%/0,88%	75,46%	3,24%/0,24%	3,55%
k5	53,82%/3,08%	73,93%	49,95/4,74	73,93%/0,88%	75,46%	3,24%/0,24%	3,55%
k10	53,82%/3,08%	58,88%	53,44/4,16	73,93%/0,88%	75,46%	3,24%/0,24%	3,55%
Seq. 8							
	Precisão (média)	Precisão (máx.)	Tempo (s)	F1 máx. (média)	F1 máx. (máx.)	Evocação (média)	Evocação (máx.)
k1	57,56%/1,51%	59,46%	50,85/1,37	74,65%/1,13%	74,65%	11,82%/0,62%	12,73%
k3	57,56%/1,51%	59,46%	49,74/4,90	74,01%/1,15%	75,48%	2,54%/0,06%	2,63%
k5	57,56%/1,51%	74,01%	49,53/4,50	74,01%/1,15%	75,48%	2,54%/0,06%	2,63%
k10	57,56%/1,51%	59,46%	50,05/5,52	74,01%/1,15%	75,48%	2,54%/0,06%	2,63%
Seq. 9							
	Precisão (média)	Precisão (máx.)	Tempo (s)	F1 máx. (média)	F1 máx. (máx.)	Evocação (média)	Evocação (máx.)
k1	62,77%/3,63%	67,81%	51,82/1,85	80,93%/2,05%	80,93%	10,80%/1,31%	13,77%
k3	62,77%/3,63%	67,81%	41,38/7,90	78,19%/1,89%	81,29%	3,73%/1,01%	5,66%
k5	62,77%/3,63%	78,19%	41,48/7,57	78,19%/1,89%	81,29%	3,73%/1,01%	5,66%
k10	62,77%/3,63%	67,81%	41,95/8,61	78,19%/1,89%	81,29%	3,73%/1,01%	5,66%
Seq. Completa							
	Precisão (média)	Precisão (máx.)	Tempo (s)	F1 máx. (média)	F1 máx. (máx.)	Evocação (média)	Evocação (máx.)
k1	43,53%/1,49%	45,81%	311,10/18,92	67,70%/1,31%	67,70%	7,57%/1,19%	11,27%
k3	43,53%/1,49%	45,81%	220,37/38,69	67,75%/1,32%	69,67%	1,75%/0,39%	2,90%
k5	43,53%/1,49%	69,67%	219,16/38,14	67,75%/1,32%	69,67%	1,75%/0,39%	2,90%
k10	43,53%/1,49%	45,81%	219,41/37,58	67,75%/1,32%	69,67%	1,75%/0,39%	2,90%

Fonte: Autor (2024).

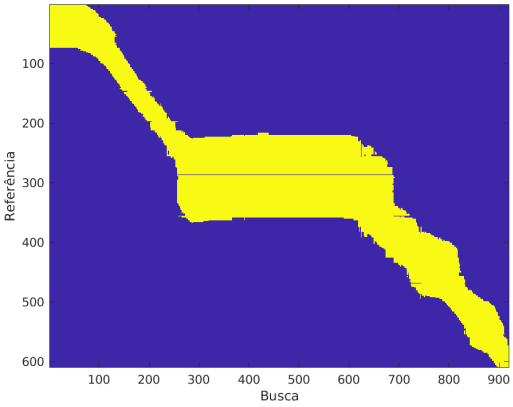
Figura 18 – As sequências que serão posteriormente enumeradas de 1 a 5 são dadas respectivamente por: (a) 2014/12/02 com início na imagem 400 com 2050 amostras para referência e 2014/12/10 com início na imagem 1750 com 2050 amostras, (b) 2014/12/02 com início na imagem 1200 com 1250 amostras para referência e 2015/02/13 com início na imagem 2645 com 1410 amostras, (c) 2014/12/02 com início na imagem 8700 com 610 amostras para referência e 2014/12/10 com início na imagem 10360 com 920 amostras para busca, (d) 2014/12/02 para referência com início na imagem 8750 com 2050 amostras e 2015/02/13 com início na imagem 9150 com 2050 amostras para busca, (e) 2014/11/14 para referência com início na imagem 1000 com 2050 amostras e 2014/12/10 com início na imagem 1750 com 2550 amostras para busca.



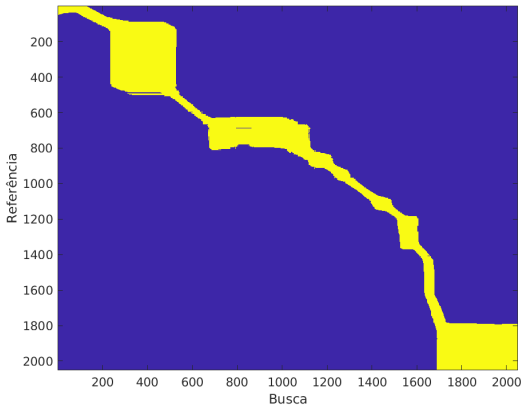
(a)



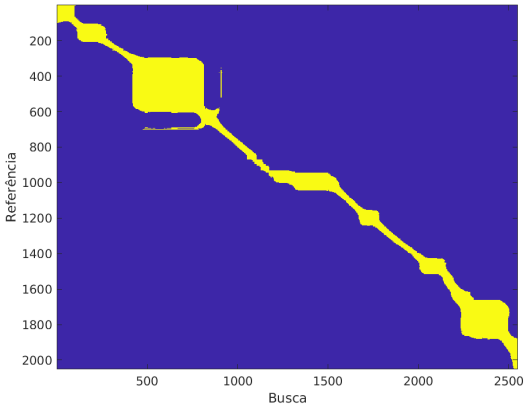
(b)



(c)



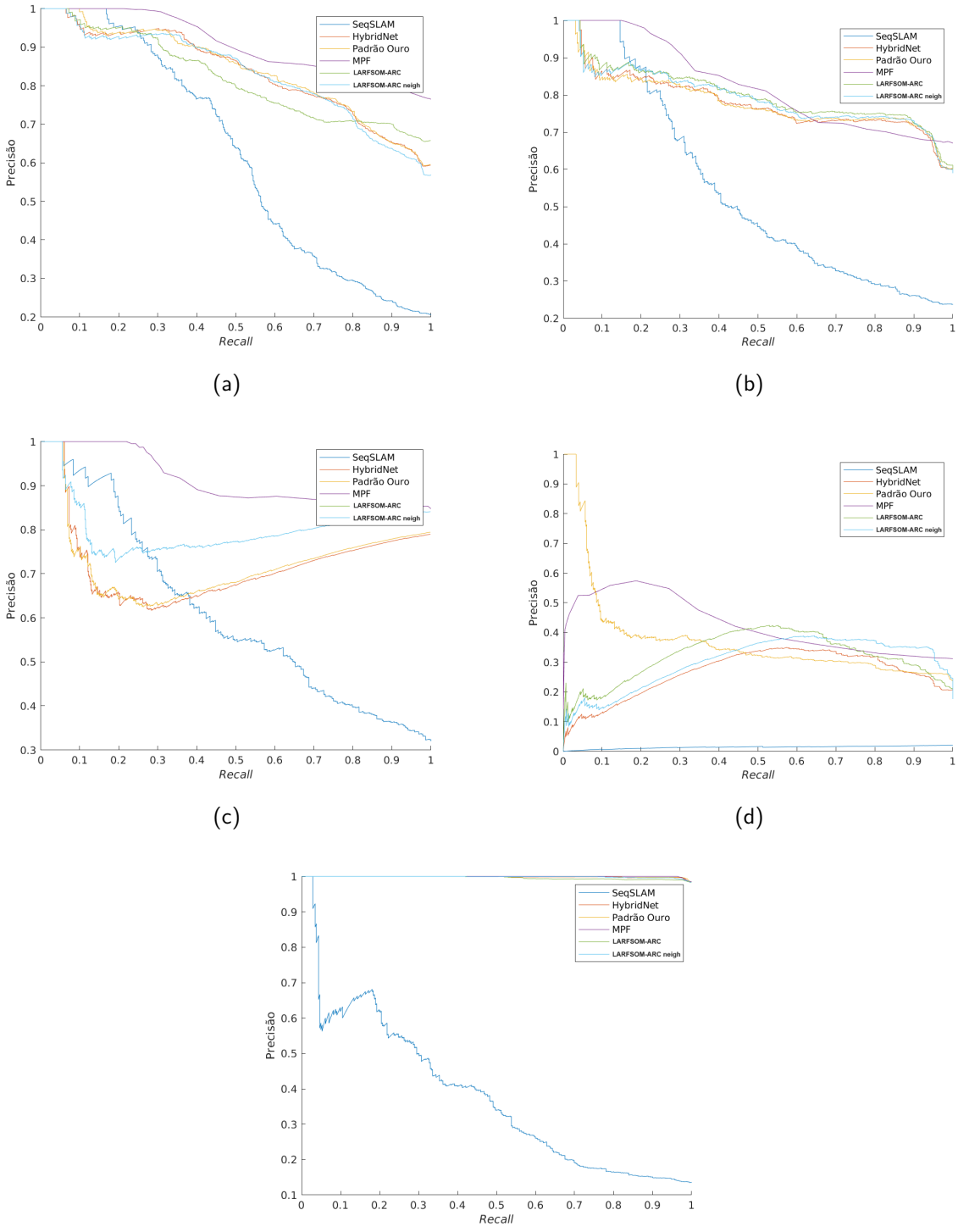
(d)



(e)

Fonte: Autor (2024).

Figura 19 – As subfiguras (a),(b),(c),(d) e (e) são referentes a gráficos de precisão-evocação das sequências 1 à 5 respectivamente.



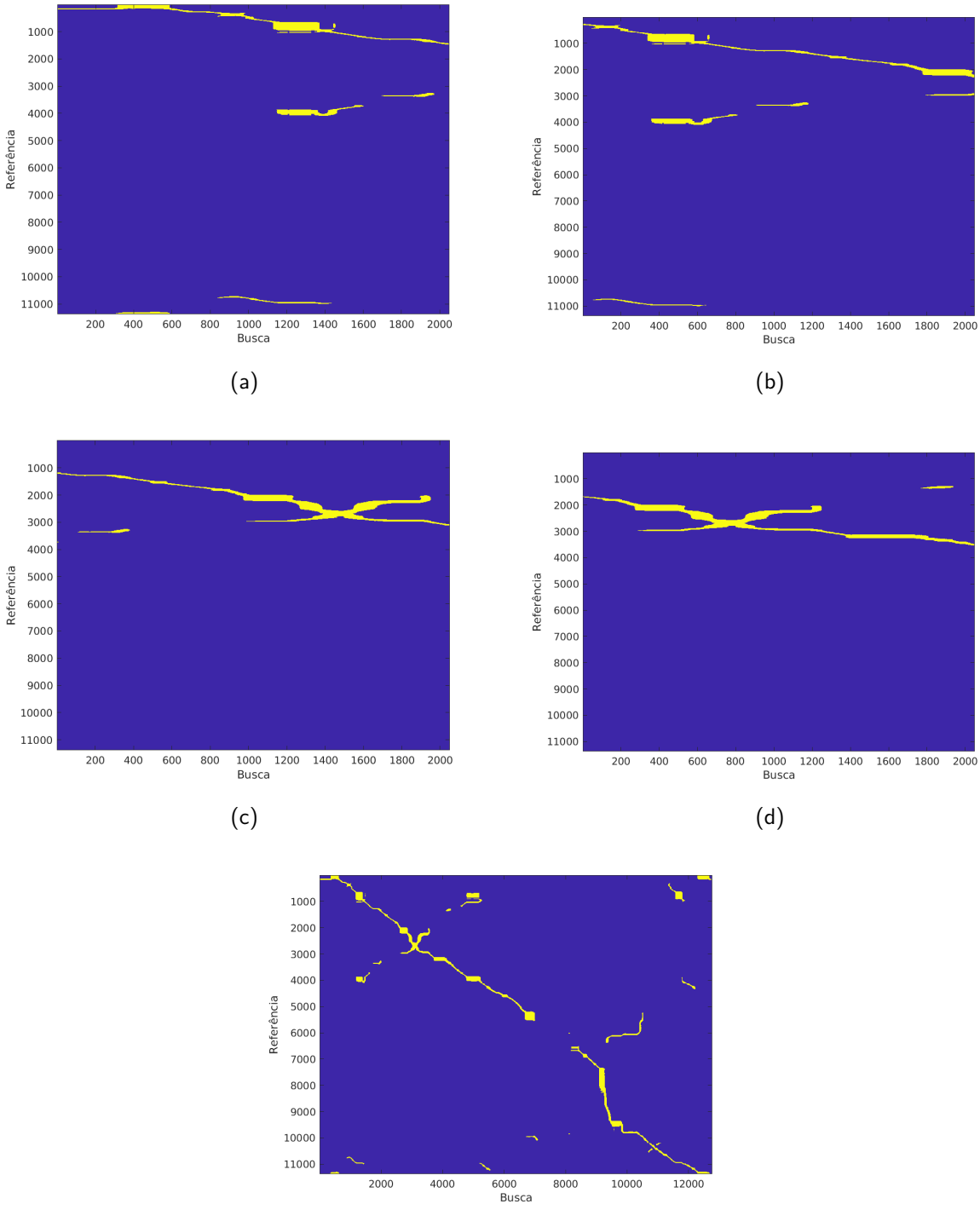
(a) (b)

(c) (d)

(e)

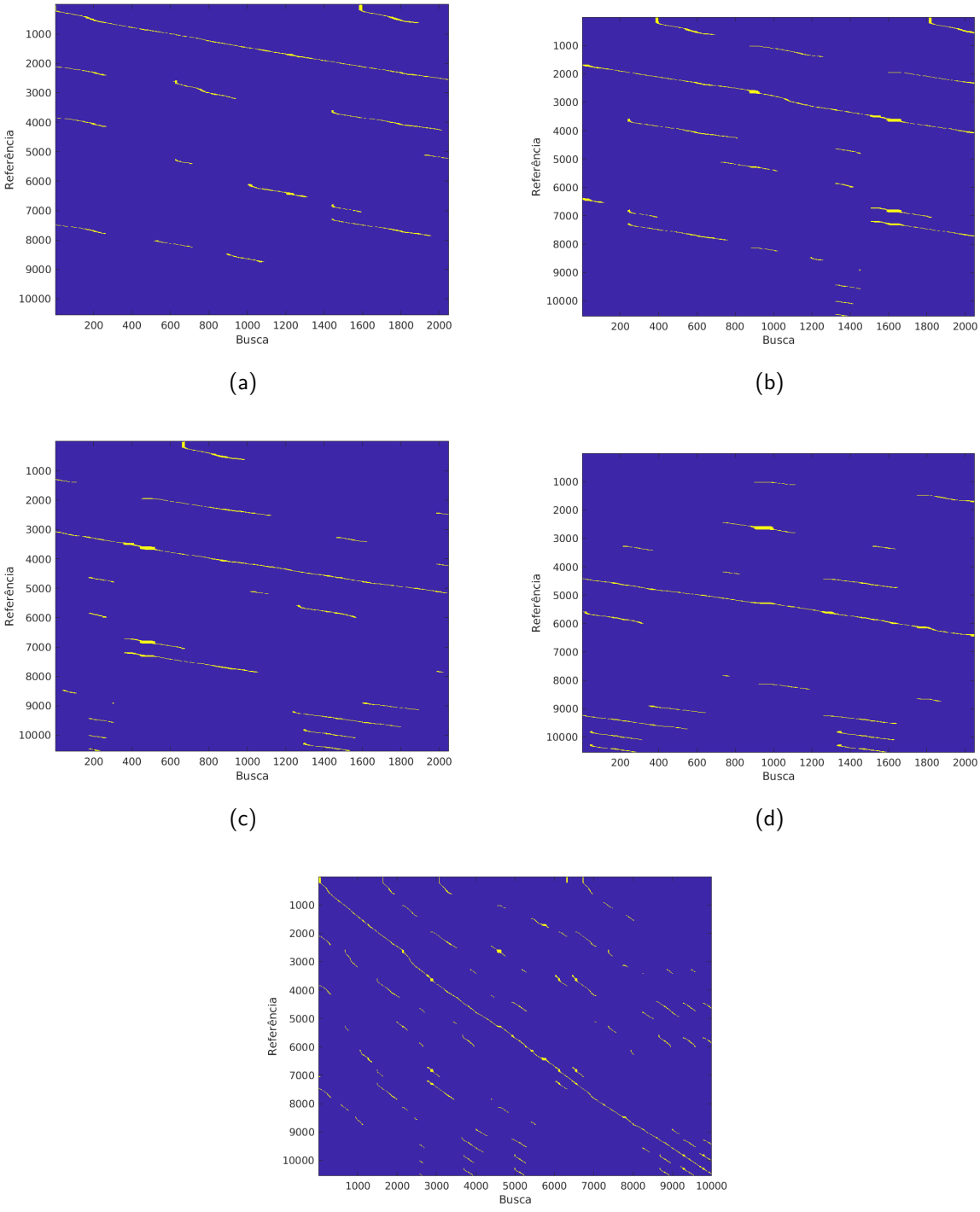
Fonte: Autor (2024).

Figura 20 – As sequências (a),(b),(c) e (d) que serão posteriormente enumeradas de 6 a 9 tem as imagens de referência retiradas de toda a trajetória de 2014/12/02, enquanto que as imagens de busca são da sequência 2014/12/10, todas com 2050 amostras e respectivamente com início nas imagens: 120, 2489, 4900 e 7001. Além disso, (e) se refere ao uso das sequências completas de 2014/12/02 e 2014/12/10.



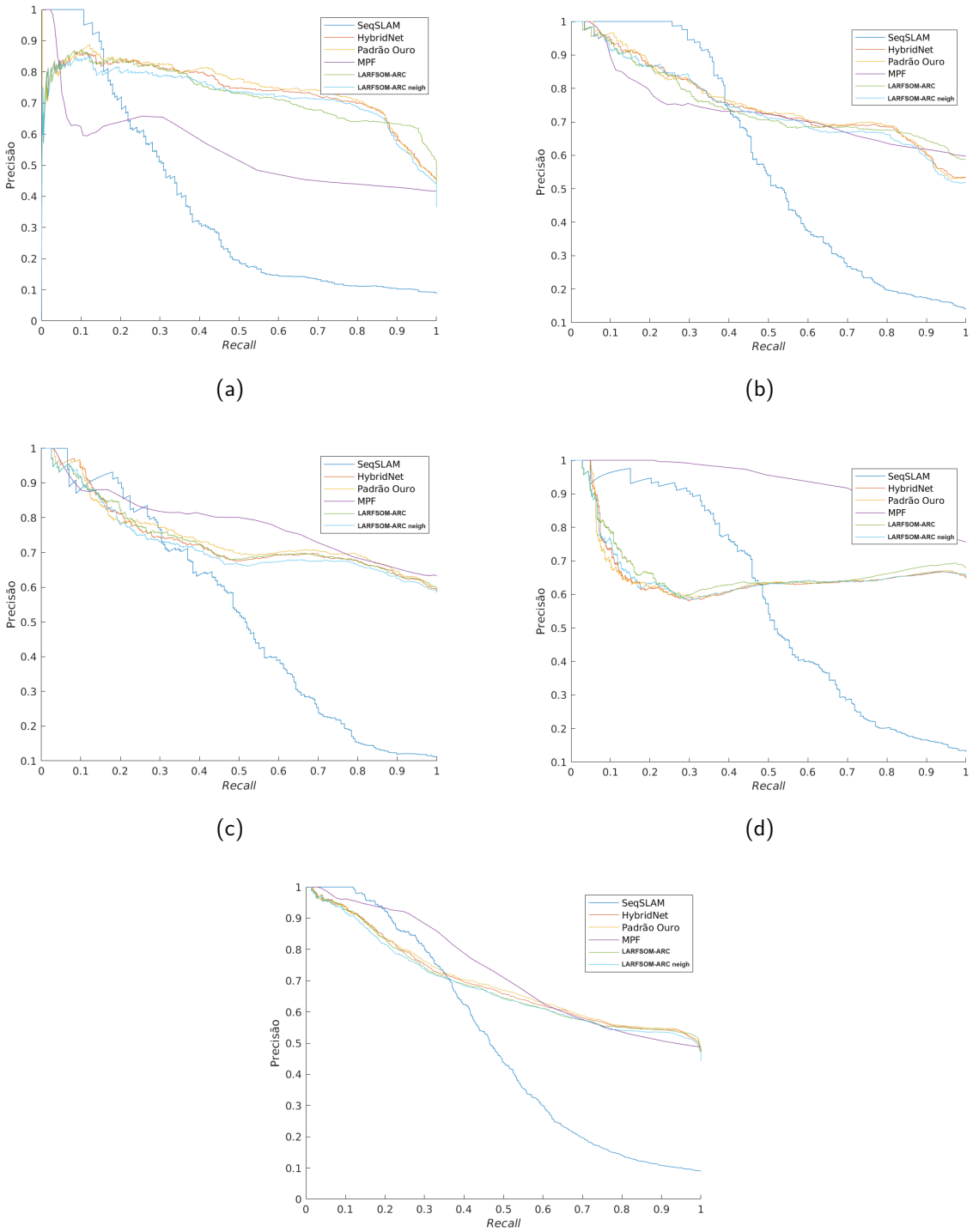
Fonte: Autor (2024).

Figura 21 – As sequências (a),(b),(c) e (d) que serão posteriormente enumeradas de 10 a 13 tem as imagens de referência retiradas de toda a trajetória de 8:45 am, enquanto que as imagens de busca são da sequência 3:45 pm, todas com 2050 amostras e respectivamente com início nas imagens: 101, 2500, 4800 e 7300. Além disso, (e) se refere ao uso das sequências completas de 8:45 am e 3:45 pm



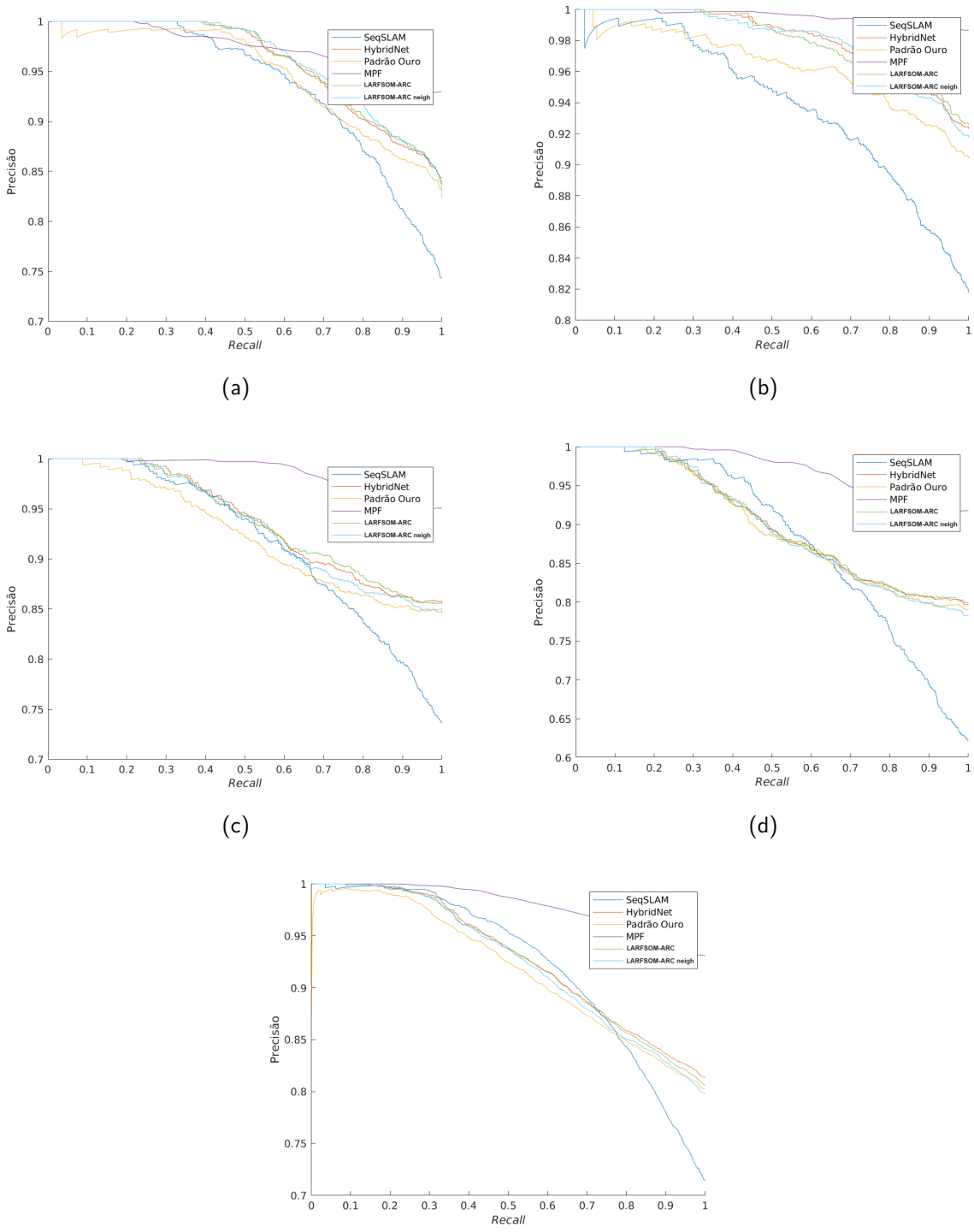
Fonte: Autor (2024).

Figura 22 – As subfiguras (a),(b),(c),(d) e (e) são referentes a gráficos de precisão-evocação das sequências 6 à 9 e à sequência completa do base de dados St. Lucia respectivamente. "LARFSOM–ARC neigh" indica que neste caso a tarefa de reconhecimento de lugares foi realizada considerando tanto as imagens agrupadas no nodo vencedor como os seus vizinhos.



Fonte: Autor (2024).

Figura 23 – As subfiguras (a),(b),(c),(d) e (e) são referentes a gráficos de precisão-evocação das sequências 10 à 13 e à sequência completa do base de dados St. Lucia respectivamente. "LARFSOM–ARC neigh" indica que neste caso a tarefa de reconhecimento de lugares foi realizada considerando tanto as imagens agrupadas no nodo vencedor como os seus vizinhos.



Fonte: Autor (2024).

6 CONCLUSÕES

Este trabalho tem como foco no desenvolvimento de uma rede SOM de estrutura variante no tempo para a tarefa de VPR. Esse sistema tem o objetivo de ser competitivo com sistemas bem estabelecidos na literatura, a um tempo de execução muito menor, principalmente a longo termo, devido a sua complexidade computacional. Dessa forma, o sistema foi desenvolvido de maneira a se adaptar aos dados da trajetória de referência, agrupando as imagens mais parecidas entre si em cada um dos nodos da rede. Assim, posteriormente, para o processo de reconhecimento de lugar somente os nodos são utilizados para a comparação inicialmente, sendo a comparação para achar a imagem correspondente realizada somente com as imagens que foram agrupadas no respectivo nodo. Em contraste com as aplicações em geral, em que o processo de comparação geralmente se dá considerando todas as imagens da base de dados.

Os experimentos foram realizados em diferentes trajetórias nas bases de dados Oxford Robotcar e St. Lucia, considerando as trajetórias completas ou somente trechos, podendo esses trechos serem considerados somente para a sequência de busca ou para as sequências de referência também. Inicialmente os experimentos foram considerados para a definição dos parâmetros, sendo posteriormente realizados para comparar o modelo proposto com os diferentes modelos da literatura.

Em quesitos de precisão, o modelo proposto somente ficou atrás do MPF na grande maioria das vezes, considerando as diferentes sequências de ambas bases de dados. Contudo, enquanto o MPF se trata de um modelo mais robusto, devido a sua técnica de comparação, o nosso modelo somente realiza comparações diretas imagem-nodo e imagem-imagem. Assim, nosso modelo foi capaz de apresentar um tempo de execução ordens de grandeza menor do que o MPF em todos os casos, sendo também muito mais rápido do que todos os outros modelos apresentados.

Ademais, também houve a tentativa de utilizar a informação da vizinhança dos nodos para a realização das comparações, na expectativa de que os vizinhos, possuindo características semelhantes do nodo ganhador, fossem auxiliar no desempenho da rede. Porém, os resultados dessa alteração não apresentaram melhora. Além disso, mesmo a tentativa de utilizar nodos mais similares ao vencedor não apresentou melhores resultados, indicando que o problema não era necessariamente a topologia da vizinhança.

O modelo apresenta limitações, como a vizinhança não estar espelhando bem a similaridade

entre os nodos da rede, o que é um fator que limita as formas de utilizar a topologia da rede para obtenção de melhores resultados. Ainda nesse contexto, o método de comparação ainda é simples, faltando ainda uma exploração de novas alternativas.

Dessarte, há pontos notórios de melhora do modelo, sendo um desses pontos a utilização do mesmo conceito de HMM utilizado pelo MPF, sendo a matriz de transição gerada a partir da relação de vizinhança dos nodos da rede. Nesse contexto, como mostrado no capítulo de resultados, é necessário também uma melhor análise com respeito a vizinhança da rede para que a vizinhança represente de melhor forma as relações entre os nodos da rede. Além disso, seria interessante que a rede possuísse uma hierarquia entre seus nodos, formando uma árvore, dado que isso influenciaria na complexidade computacional, sendo para isso necessário uma mudança no processo de treinamento. Além disso, há métodos mais sofisticados para a busca pelo vencedor (BMU), o que poderia tornar o sistema proposto ainda mais rápido.

REFERÊNCIAS

ANDRADES, I. S.; AGUILAR, J. J. C.; GARCÍA, J. M. V.; CARRILLO, J. A. C.; LOZANO, M. S. Low-cost road-surface classification system based on self-organizing maps. *Sensors*, MDPI, v. 20, n. 21, p. 6009, 2020.

ANDREASSON, H.; DUCKETT, T. Topological localization for mobile robots using omni-directional vision and local features. *IFAC Proceedings Volumes*, Elsevier, v. 37, n. 8, p. 36–41, 2004.

ARANDJELOVIC, R.; GRONAT, P.; TORII, A.; PAJDLA, T.; SIVIC, J. Netvlad: Cnn architecture for weakly supervised place recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 5297–5307.

ARAÚJO, A. R.; COSTA, D. C. Local adaptive receptive field self-organizing map for image color segmentation. *Image and Vision Computing*, Elsevier, v. 27, n. 9, p. 1229–1239, 2009.

BARNES, D.; GADD, M.; MURCUTT, P.; NEWMAN, P.; POSNER, I. The oxford radar robotcar dataset: A radar extension to the oxford robotcar dataset. In: IEEE. *2020 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2020. p. 6433–6438.

BAY, H.; TUYTELAARS, T.; GOOL, L. V. Surf: Speeded up robust features. In: SPRINGER. *European conference on computer vision*. [S.l.], 2006. p. 404–417.

CALONDER, M.; LEPETIT, V.; STRECHA, C.; FUA, P. Brief: Binary robust independent elementary features. In: SPRINGER. *European conference on computer vision*. [S.l.], 2010. p. 778–792.

CHEN, W.; SHANG, G.; JI, A.; ZHOU, C.; WANG, X.; XU, C.; LI, Z.; HU, K. An overview on visual slam: From tradition to semantic. *Remote Sensing*, MDPI, v. 14, n. 13, p. 3010, 2022.

CHEN, Z.; JACOBSON, A.; SÜNDERHAUF, N.; UPCROFT, B.; LIU, L.; SHEN, C.; REID, I.; MILFORD, M. Deep learning features at scale for visual place recognition. In: IEEE. *2017 IEEE international conference on robotics and automation (ICRA)*. [S.l.], 2017. p. 3223–3230.

CHEN, Z.; LAM, O.; JACOBSON, A.; MILFORD, M. Convolutional neural network-based place recognition. *arXiv preprint arXiv:1411.1509*, 2014.

ĆWIAN, K.; NOWICKI, M. R.; WIETRZYKOWSKI, J.; SKRZYPCZYŃSKI, P. Large-scale lidar slam with factor graph optimization on high-level geometric features. *Sensors*, MDPI, v. 21, n. 10, p. 3445, 2021.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: IEEE. *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. [S.l.], 2005. v. 1, p. 886–893.

DICKMANNS, E. D. An integrated approach to feature based dynamic vision. In: IEEE COMPUTER SOCIETY. *Proceedings CVPR'88: The Computer Society Conference on Computer Vision and Pattern Recognition*. [S.l.], 1988. p. 820–821.

DONG, H.; CHEN, X.; DUSMANU, M.; LARSSON, V.; POLLEFEYS, M.; STACHNISS, C. Learning-based dimensionality reduction for computing compact and effective local feature descriptors. In: IEEE. *2023 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2023. p. 6189–6195.

DURRANT-WHYTE, H.; BAILEY, T. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, IEEE, v. 13, n. 2, p. 99–110, 2006.

DURRANT-WHYTE, H.; BAILEY, T. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, IEEE, v. 13, n. 2, p. 99–110, 2006.

FOLEY, J. A.; RAMANKUTTY, N.; BRAUMAN, K. A.; CASSIDY, E. S.; GERBER, J. S.; JOHNSTON, M.; MUELLER, N. D.; O'CONNELL, C.; RAY, D. K.; WEST, P. C. et al. Solutions for a cultivated planet. *Nature*, Nature Publishing Group UK London, v. 478, n. 7369, p. 337–342, 2011.

FRITZKE, B. Growing cell structures—a self-organizing network for unsupervised and supervised learning. *Neural networks*, Elsevier, v. 7, n. 9, p. 1441–1460, 1994.

FRITZKE, B. A growing neural gas network learns topologies. *Advances in neural information processing systems*, v. 7, 1994.

FRITZKE, B. Growing self-organizing networks-why? In: *ESANN*. [S.l.: s.n.], 1996. v. 96, p. 61–72.

GÁLVEZ-LÓPEZ, D.; TARDOS, J. D. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, IEEE, v. 28, n. 5, p. 1188–1197, 2012.

GÁLVEZ-LÓPEZ, D.; TARDOS, J. D. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, IEEE, v. 28, n. 5, p. 1188–1197, 2012.

GARG, S.; FISCHER, T.; MILFORD, M. Where is your place, visual place recognition? *CoRR*, abs/2103.06443, 2021. Disponível em: <https://arxiv.org/abs/2103.06443>.

GARG, S.; SUENDERHAUF, N.; MILFORD, M. Don't look back: Robustifying place categorization for viewpoint-and condition-invariant place recognition. In: IEEE. *2018 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2018. p. 3645–3652.

GARG, S.; VANKADARI, M.; MILFORD, M. Seqmatchnet: Contrastive learning with sequence matching for place recognition & relocalization. In: PMLR. *Conference on Robot Learning*. [S.l.], 2022. p. 429–443.

GILLAN, W. Prometheus and drive: their implications for traffic managers. In: IEEE. *Conference Record of papers presented at the First Vehicle Navigation and Information Systems Conference (VNIS'89)*. [S.l.], 1989. p. 237–243.

GLOVER, A.; MADDERN, W.; WARREN, M.; REID, S.; MILFORD, M.; WYETH, G. Openfabmap: An open source toolbox for appearance-based loop closure detection. In: *2012 IEEE International Conference on Robotics and Automation*. [S.l.: s.n.], 2012. p. 4730–4735.

GLOVER, A. J.; MADDERN, W. P.; MILFORD, M. J.; WYETH, G. F. Fab-map+ ratslam: Appearance-based slam for multiple times of day. In: IEEE. *2010 IEEE international conference on robotics and automation*. [S.l.], 2010. p. 3507–3512.

GLOVER, A. J.; MADDERN, W. P.; MILFORD, M. J.; WYETH, G. F. Fab-map+ ratslam: Appearance-based slam for multiple times of day. In: IEEE. *2010 IEEE international conference on robotics and automation*. [S.l.], 2010. p. 3507–3512.

GRISETTI, G.; KÜMMERLE, R.; STACHNISS, C.; BURGARD, W. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, IEEE, v. 2, n. 4, p. 31–43, 2010.

HAUSLER, S.; JACOBSON, A.; MILFORD, M. Multi-process fusion: Visual place recognition using multiple image processing methods. *IEEE Robotics and Automation Letters*, IEEE, v. 4, n. 2, p. 1924–1931, 2019.

HUANG, S.; DISSANAYAKE, G. Convergence and consistency analysis for extended kalman filter based slam. *IEEE Transactions on robotics*, IEEE, v. 23, n. 5, p. 1036–1049, 2007.

KIT, D.; KONG, Y.; FU, Y. Lasom: Location aware self-organizing map for discovering similar and unique visual features of geographical locations. In: IEEE. *2014 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2014. p. 263–270.

KOHONEN, T. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, Springer, v. 43, n. 1, p. 59–69, 1982.

KRAJNÍK, T.; CRISTÓFORIS, P.; KUSUMAM, K.; NEUBERT, P.; DUCKETT, T. Image features for visual teach-and-repeat navigation in changing environments. *Robotics and Autonomous Systems*, Elsevier, v. 88, p. 127–141, 2017.

LABBE, M.; MICHAUD, F. Online global loop closure detection for large-scale multi-session graph-based slam. In: IEEE. *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. [S.l.], 2014. p. 2661–2666.

LEE, S.-W.; HSU, C.-M.; LEE, M.-C.; FU, Y.-T.; ATAS, F.; TSAI, A. Fast point cloud feature extraction for real-time slam. In: IEEE. *2019 International Automatic Control Conference (CACS)*. [S.l.], 2019. p. 1–6.

LENAIN, R.; TRICOT, N.; BERDUCAT, M. La robotique agricole: l'essor de nouveaux outils pour l'agroécologie. *Sciences Eaux & Territoires*, n. 29, p. 64–67, 2019.

LI, X.; HAN, K.; LI, S.; PRISACARIU, V. Dual-resolution correspondence networks. *Advances in Neural Information Processing Systems*, v. 33, p. 17346–17357, 2020.

LIU, H.; BAN, X.-j. Clustering by growing incremental self-organizing neural network. *Expert Systems with Applications*, Elsevier, v. 42, n. 11, p. 4965–4981, 2015.

LIU, N.; WANG, J.; GONG, Y. Deep self-organizing map for visual classification. In: IEEE. *2015 international joint conference on neural networks (IJCNN)*. [S.l.], 2015. p. 1–6.

LOH, W.-L. On latin hypercube sampling. *The annals of statistics*, Institute of Mathematical Statistics, v. 24, n. 5, p. 2058–2080, 1996.

LOWE, D. G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, Springer, v. 60, p. 91–110, 2004.

LOWRY, S.; ANDREASSON, H. Lightweight, viewpoint-invariant visual place recognition in changing environments. *IEEE Robotics and Automation Letters*, IEEE, v. 3, n. 2, p. 957–964, 2018.

MADDERN, W.; PASCOE, G.; LINEGAR, C.; NEWMAN, P. 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, SAGE Publications Sage UK: London, England, v. 36, n. 1, p. 3–15, 2017.

MARMAGLIO, P.; CONSOLATI, D.; AMICI, C.; TIBONI, M. Autonomous vehicles for healthcare applications: A review on mobile robotic systems and drones in hospital and clinical environments. *Electronics*, MDPI, v. 12, n. 23, p. 4791, 2023.

MARSLAND, S.; SHAPIRO, J.; NEHMZOW, U. A self-organising network that grows when required. *Neural networks*, Elsevier, v. 15, n. 8-9, p. 1041–1058, 2002.

MARTÍNEZ-OTZETA, J. M.; RODRÍGUEZ-MORENO, I.; MENDIALDUA, I.; SIERRA, B. Ransac for robotic applications: A survey. *Sensors*, MDPI, v. 23, n. 1, p. 327, 2022.

MASONE, C.; CAPUTO, B. A survey on deep visual place recognition. *IEEE Access*, IEEE, v. 9, p. 19516–19547, 2021.

MCINNES, L.; HEALY, J.; MELVILLE, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

MILFORD, M. J.; WYETH, G. F. Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. In: *2012 IEEE International Conference on Robotics and Automation*. [S.l.: s.n.], 2012. p. 1643–1649.

MONTEMERLO, M.; THRUN, S.; KOLLER, D.; WEGBREIT, B. et al. Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In: *IJCAI*. [S.l.: s.n.], 2003. v. 3, n. 2003, p. 1151–1156.

MUR-ARTAL, R.; MONTIEL, J. M. M.; TARDOS, J. D. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, IEEE, v. 31, n. 5, p. 1147–1163, 2015.

PALOMO, E. J.; LÓPEZ-RUBIO, E. The growing hierarchical neural gas self-organizing neural network. *IEEE transactions on neural networks and learning systems*, IEEE, v. 28, n. 9, p. 2000–2009, 2016.

PIRE, T.; FISCHER, T.; CIVERA, J.; CRISTÓFORIS, P. D.; BERLLES, J. J. Stereo parallel tracking and mapping for robot localization. In: IEEE. *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. [S.l.], 2015. p. 1373–1378.

PRADEEP, V.; MEDIONI, G.; WEILAND, J. Visual loop closing using multi-resolution sift grids in metric-topological slam. In: IEEE. *2009 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.], 2009. p. 1438–1445.

RUBLEE, E.; RABAUD, V.; KONOLIGE, K.; BRADSKI, G. Orb: An efficient alternative to sift or surf. In: IEEE. *2011 International conference on computer vision*. [S.l.], 2011. p. 2564–2571.

SIAM, S. M.; ZHANG, H. Fast-seqslam: A fast appearance based place recognition algorithm. In: IEEE. *2017 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2017. p. 5702–5708.

SICILIANO, O. K. B. *Springer Handbook of Robotics*. Springer, 2016. (Springer Handbooks). Disponível em: <http://gen.lib.rus.ec/book/index.php?md5=25a40a3f9cf05cc9f0b6f9a75ec5ddc9>.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

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

SÜNDERHAUF, N.; PROTZEL, P. Brief-gist-closing the loop by simple means. In: IEEE. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. [S.l.], 2011. p. 1234–1241.

THRUN, S.; GUTMANN, J.-S.; FOX, D.; BURGARD, W.; KUIPERS, B. et al. Integrating topological and metric maps for mobile robot navigation: A statistical approach. *AAAI/IAAI*, v. 9, p. 989–995, 1998.

WICKRAMASINGHE, C. S.; AMARASINGHE, K.; MANIC, M. Deep self-organizing maps for unsupervised image classification. *IEEE Transactions on Industrial Informatics*, IEEE, v. 15, n. 11, p. 5837–5845, 2019.

ZAFFAR, M.; EHSAN, S.; STOLKIN, R.; MAIER, K. M. Sensors, slam and long-term autonomy: A review. In: IEEE. *2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. [S.l.], 2018. p. 285–290.

ZHANG, X.; WANG, L.; SU, Y. Visual place recognition: A survey from deep learning perspective. *Pattern Recognition*, Elsevier, v. 113, p. 107760, 2021.

ZHAO, Z.; MA, Y.; MUSHTAQ, A.; RAJPER, A. M. A.; SHEHAB, M.; HEYBOURNE, A.; SONG, W.; REN, H.; TSE, Z. T. H. Applications of robotics, artificial intelligence, and digital technologies during covid-19: a review. *Disaster Medicine and Public Health Preparedness*, Cambridge University Press, v. 16, n. 4, p. 1634–1644, 2022.