UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

CARLOS HENRIQUE GOMES SOARES

ESTUDO COMPARATIVO DE MÉTODOS PARA RECONHECIMENTO VISUAL DE LOCAIS EM AMBIENTES EXTERNOS

RECIFE

2025

UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

ENGENHARIA DA COMPUTAÇÃO

CARLOS HENRIQUE GOMES SOARES

ESTUDO COMPARATIVO DE MÉTODOS PARA RECONHECIMENTO VISUAL DE LOCAIS EM AMBIENTES EXTERNOS

TCC apresentado ao Curso de engenharia da computação da Universidade Federal de Pernambuco, Centro de Informática, como requisito para a obtenção do título de graduado em engenharia da computação.

Orientador(a): Prof. Aluizio Fausto Ribeiro Araújo
Coorientador(a): Prof. Adrien Durand-Petiteville

RECIFE
2025

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Soares, Carlos Henrique Gomes.
Estudo comparativo de métodos para reconhecimento visual de locais em ambientes externos / Carlos Henrique Gomes Soares. - Recife, 2025.
61 : il., tab.

Orientador(a): Aluizio Fausto Ribeiro Araújo
Cooorientador(a): Adrien Durand-Petiteville
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Engenharia da Computação - Bacharelado, 2025.
8,5.
Inclui referências.

1. Reconhecimento visual de locais. 2. SeqSLAM. 3. Multi-Process Fusion. 4. Robótica móvel. 5. Processamento de imagens. 6. Inteligência artificial. I. Araújo, Aluizio Fausto Ribeiro . (Orientação). II. Durand-Petiteville, Adrien. (Coorientação). IV. Título.

000 CDD (22.ed.)

CARLOS HENRIQUE GOMES SOARES

ESTUDO COMPARATIVO DE MÉTODOS PARA RECONHECIMENTO VISUAL DE LOCAIS EM AMBIENTES EXTERNOS

TCC apresentado ao Curso de engenharia da computação da Universidade Federal de Pernambuco, Centro de Informática, como requisito para a obtenção do título de graduado em engenharia da computação.

Aprovado em: 03/04/2025.

BANCA EXAMINADORA

Profº. Dr. Aluizio Fausto Ribeiro Araújo
Universidade Federal de Pernambuco

Profº. Dr. João Paulo Cerquinho Cajueiro
Universidade Federal de Pernambuco

AGRADECIMENTOS

Gostaria de expressar minha mais profunda gratidão à minha família, por todo apoio e encorajamento durante cada etapa desta jornada. Vocês foram a base que sustentou meus esforços, proporcionando força nos momentos difíceis e compartilhando alegria nas conquistas.

Agradeço ao professor Adrien e ao professor Aluízio, pela orientação, dedicação e incentivo ao longo do meu percurso. E por me colocar de volta no caminho das vezes que me perdi.

Agradeço a minha namorada Luana, por estar ao meu lado em cada passo desta jornada.

Agradeço a Adriel e Lucas, por estarem sempre prontos para me ajudar com atenção e paciência nas diversas dúvidas e obstáculos que enfrentei.

"A persistência é o caminho do êxito."
— Charles Chaplin

RESUMO

Este estudo examina técnicas de reconhecimento visual de locais empregando os métodos SeqSLAM e Multi-Process Fusion (MPF). A análise considera o desempenho dessas abordagens em condições adversas, tais como variações na iluminação e mudanças sazonais. O SeqSLAM emprega um método baseado em sequências de imagens e diferenças absolutas, enquanto MPF combina descritores (SAD, HOG e CNN) para maior robustez. Ambos os métodos foram avaliados com base em critérios de precisão e revocação, e os resultados obtidos destacam as principais vantagens e limitações de cada abordagem.

Palavras-chave: Reconhecimento visual de locais; SeqSLAM; Multi-Process Fusion; Robótica móvel; Processamento de imagens; Inteligência artificial

ABSTRACT

This study examines visual location recognition techniques using SeqSLAM and Multi-Process Fusion (MPF) methods. The analysis considers the performance of these approaches under adverse conditions, such as variations in illumination and seasonal changes. SeqSLAM employs a method based on image sequences and absolute differences, while MPF combines descriptors (SAD, HOG and CNN) for greater robustness. Both methods were evaluated based on precision and recall criteria, and the results

Keywords: Visual place recognition; SeqSLAM; Multi-Process Fusion; Mobile robotics; Image processing; Artificial intelligence

LISTA DE FIGURAS

Figura 1:Pipeline genérico de reconhecimento de locais com os seguintes módulos: representação de lugares, mapeamento de lugares e geração de crenças..... 19

Figura 2: Principais paradigmas de representação do ambiente. Em cima: mapa métrico. Note-se que os objetos identificados são apenas ilustrativos, pois, tipicamente, os mapas representam estruturas de mais baixo nível, como arestas e cantos. Em baixo: mapa topológico. Aqui, a posição dos nós sobre a planta é também ilustrativa, pois estes mapas podem não incluir informação métrica..... 22

Figura 3: Observações de um mesmo local extraídas em diferentes ocasiões..... 25

Figura 4:Observações semelhantes extraídas em diferentes locais.....25

Figura 5: Mesmo local visto de diferentes pontos de vista.....25

Figura 6: Modelo do problema de SLAM ilustrado por meio de um grafo..... 26

Figura 7: O problema da mudança perceptual. O lugar A e B são lugares diferentes, mas parecem muito mais semelhantes do que o lugar A a si mesmo durante uma noite chuvosa (A'), tanto em uma imagem inteira quanto em uma base de características individuais. Nossa abordagem usa correspondência local dentro de sequências para corresponder com sucesso A a A', mas não A a B.. 27

Figura 8: Este diagrama mostra a formulação da matriz de emissão usando quatro métodos de processamento de imagem. Um vetor de características é extraído para cada método, que é então comparado aos modelos de banco de dados para produzir um vetor de distância. Logaritmos e normalização são usados para formatar cada vetor antes de criar a matriz de emissão..... 28

Figura 9: Diagrama teórico em bloco da execução do SeqSlam.....30

Figura 10: Diagrama teórico em bloco da execução do SeqSlam.....33

Figura 11: amostra de imagens do Nord - amostra de imagens..... 37

Figura 12: pista de Nürburgring.....37

figura 13: Comparação entre a imagem original passada para a escala de cinza em MATLAB, Python e C++.MSE MATLAB X Python= 0.00000000 MSE MATLAB X C++ = 0.23125462.....38

Figura 14: Comparação entre a imagem na escala de cinza re-dimensionalizada em MATLAB, Python e C++.....38

Figura 15: Comparação entre a imagem clipadas e normalizadas MATLAB, Python e C++..... 39

Figura 16: Figura ilustra a construção da matriz de pré-processamento:(a) representação dos cortes na imagem original(b) corte normalizado aleatório destacado como uma matriz,(c) Corte normalizado transformado em um vetor coluna e (d) simulação da matriz de pré-processamento destacando o corte normalizado em sua posição.....40

Figura 17: Precisão, Revocação e Tempo de execução médios em cada linguagem para o algoritmos analisado.....47

Figura 18: Cada linha representa frames capturados em diferentes momentos das sequências "2014/12/02", "2015/02/13", "2014/12/10" e "2014/11/14", respectivamente, no conjunto de dados Oxford Robotcar.....49

Figura 19: As sequências que serão posteriormente enumeradas de 1 a 5 são dadas respectivamente por: (a) 2014/12/09 com início na imagem 400 com 2050 amostras para referência e 2014/12/10 com início na imagem 1750 com 2050 amostras para busca, (b) 2014/12/09 com início na imagem 1200 com 1250 amostras para referência e 2014/12/10 com início na imagem 2645 com 1410 amostras para busca, (c) 2014/12/09 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) 2014/11/14 para referência com início na imagem 8750 com 2050 amostras e 2014/12/10 com início na imagem 9150 com 2050 amostras para busca, (e) 2014/12/09 para referência com início na imagem 1000 com 2050 amostras e 2015/02/13 com início na imagem 1750 com 2550 amostras para busca.....50
Figura 20: As subfiguras (a),(b),(c),(d) e (e) são referentes a gráficos de precisão-revocação das sequências de 1 a 5 respectivamente.....57

LISTA DE TABELAS

Tabela 1: Comparação entre configurações de baseadas em diferentes sensores para o reconhecimento de locais..... 19
Tabela 2:MSE comparado a implementação de matriz de pré-processamento de entre cada dataserComp..... 40
Tabela 3: MSE comparado a implementação de matriz de diferenças entre MATLAB e Python..... 41
Tabela 4: MSE comparando a implementação de matriz de diferenças entre MATLAB e C++..... 42
Tabela 5: MSE comparado a implementação de matriz de diferenças aprimorada entre MATLAB e Python..... 42
Tabela 6: MSE comparado a implementação de matriz de diferenças aprimorada entre MATLAB e C++. 43
Tabela 7: Comparando entre as métricas entre MATLAB..... 44
Tabela 8: Comparando entre as métricas entre Python..... 45
Tabela 9: Comparando entre as métricas entre C++..... 45
Tabela 10: Tabela com a média e o desvio padrão de todos os casos em cada lingagem..... 46
A tabela 11: Mostra o erro médio quadrado (MSE), entre a uma image número 400 da sequência 2014/12/09 do dataset do Oxford Robotcar, redimensionamento e normalização em matlab e em python:..... 51
Tabela 12: o valor dos ângulos Θmédio e Θmax para a diferença entre os ângulos das imagens vetorizadas em matlab e em python para cada método, para o conjunto de referência das sequências 1-5..... 51
Tabela 13: O valor dos ângulos Θmédio e Θmax para a diferença entre os ângulos das imagens vetorizadas em matlab e em python para cada método, para o conjunto de match das sequências 1-5 52
Tabela 14: Distância dos cossenos média em python subtraida da distância dos cossenos média em matlab CNN e HOG..... 53
Tabela 15: o MSE para a CNN-D e SAD..... 53
Tabela 16: MSE entre as matrizes de observação da sequência de 1 a 5..... 54
Tabela 17: Vetor worstIDCounter para cada sequência..... 55
Tabela 18: Comparação de precisão para trechos das sequência no Oxford Robotcar em Matlab e em Python..... 58

LISTA DE ABREVIATURAS E SIGLAS

CNN	*Convolutional Neural Network*
HOG	*Histogram of Oriented Gradients*
HMM	*Hidden Markov Model*
MPF	*Multi-Sensor Fusion*
MSE	*Mean Squared Error*
ORB	*Oriented FAST and Rotated BRIEF*
SAD	*Sum-of-Absolute-Differences*
SIFT	*Scale-Invariant Feature Transform*
SLAM	*Simultaneous Localization and Mapping*
SURF	*Speeded Up Robust Features*
VPR	*Visual Place Recognition*

SUMÁRIO

1 INTRODUÇÃO..... 15
1.1 OBJETIVOS..... 17
1.1.1 Objetivos específicos:..... 17
1.2 ESTRUTURA DO TEXTO..... 17
2 FUNDAMENTAÇÃO TEÓRICA..... 18
2.1 O QUE É UM LOCAL..... 18
2.2 COMO RECONHECER LOCAL..... 18
2.3 RECONHECIMENTO VISUAL DE LOCAIS..... 19
2.3.1 Representação de locais..... 20
2.3.1.1 Sem extratores..... 20
2.3.1.2 Abordagem Clássica..... 20
2.3.1.3 Tecnicas baseadas em Deep learning e CNN..... 21
2.3.2 Mapeamento de locais..... 22
2.3.2.1 Mapas Metricos..... 22
2.3.2.2 Mapas Topológicos..... 22
2.3.3 Geração de crença..... 23
2.3.3.1 Método quadro a quando de imagens..... 23
2.3.3.2 Método baseado em Sequência de quadros de imagens..... 23
2.3.3.3 Métodos Hierárquicos..... 24
2.4 DESAFIOS NO RECONHECIMENTO VISUAL DE LOCAIS..... 24
2.4.1 Mudança de aparência de um Local..... 24
2.4.2 Aliasing perceptual..... 25
2.4.3 Mudanças de ponto de vista..... 25
2.5 TÉCNICAS PARA SOLUCIONAR O PROBLEMA DE RECONHECIMENTO VISUAL DE LOCAIS..... 26
2.5.1 Seqslam..... 26
2.5.2.MPF..... 28
2.6 Métricas para Análise no Reconhecimento Visual de Lugares..... 29
3 METODOLOGIA..... 30
3.1 SEQSLAM..... 30
3.1.1 Aquisição de imagens..... 30
3.1.2 Pré-processamento das imagens:..... 30
3.1.3 Comparação entre as imagens nos conjuntos de referência e de teste..... 31
3.1.4 Aplicação do processo de aprimoramento de contrastes locais..... 31
3.1.5 Construção das sequências com diferentes velocidades e determinação do melhor match entre a sequência teste e as sequências com diferentes velocidades..... 32
3.1.6 Determinação da localização do robô a partir da sequência com melhor correspondência..... 32
3.1.7 Atualizar o mapa..... 32
3.2 MPF..... 33

3.2.1 Aquisição de imagens..... 33
3.2.2 Pré-processamento das imagens..... 33
3.2.3 Extração de Características..... 33
3.2.4 Cálculo das Diferenças entre Imagens..... 34
3.2.5 Normalização das diferenças..... 34
3.2.6 Encontrasse o pior caso..... 34
3.2.7 Executa o algoritmo de Viterbi..... 34
3.2.8 Identifica o estado mais provável..... 35
4 RESULTADOS.....36
4.1 SEQSLAM..... 36
4.1.1 Aquisição de imagens..... 36
4.1.2 Pré-processamento das imagens..... 37
4.1.3 Comparação entre as imagens nos conjuntos de referência e de teste.....41
4.1.4 Aplicação o processo de aprimoramento de contraste local..... 42
4.1.5 Construção das sequências com diferentes velocidades e determinação do melhor match entre a sequência teste e as sequência com diferentes velocidades.....43
4.2 COMPARAÇÕES do SeqSlam..... 44
4.3 MPF..... 48
4.3.1 Aquisição de imagens:..... 48
4.3.2 Pré-processamento das imagens:..... 50
4.3.3 Extração de Características..... 51
4.3.4 Cálculo das Diferenças entre Imagens..... 53
4.3.5 Normalização das diferenças..... 54
4.3.6 Encontra-se o pior caso.....55
4.3.7 Executa o algoritmo de Viterbi..... 56
4.3.8 Identifica o estado mais provável..... 56
4.4 COMPARAÇÕES DO MPF..... 57
5 CONCLUSÃO..... 59
REFERÊNCIAS..... 60

1 INTRODUÇÃO

A navegação é um problema fundamental na nas áreas da robótica móvel, pois possibilita que o robô evite perigos e alcance objetivos. O desenvolvimento destas competências possibilitam inúmeras aplicações como navegação de veículos autônomos, robótica social e em operações que envolvem algum risco para o ser humano. Para navegar é essencial que o robô saiba localizar-se no ambiente através de modelos que realizam o reconhecimento de lugares. O avanço das técnicas de identificação de lugares permite que robôs naveguem por ambientes mais complexos e desafiadores. Um dos principais tratamentos para reconhecimento de lugares envolve a classe de problemas localização e mapeamento simultâneos (SLAM -*Simultaneous Localization and Mapping* -). Tipicamente, o SLAM é um método no qual um robô vai navegando por um ambiente desconhecido ao mesmo tempo em que vai criando um mapa métrico do ambiente, ou seja, uma representação geométrica acurada de um ambiente na qual o robô determina sua localização.

Entretanto, as soluções para um SLAM baseados em mapas métricos costumam apresentar limitações quando precisam lidar com mudanças e variações em ambientes dinâmicos, como variações de iluminação e variações climáticas, que são agravados devido a hardwares com baixa capacidade de processamento. Uma forma de contornar esse entrave é abrir mão do uso de mapas métricos, que são mais detalhados e precisam de sensores georreferenciados e passar a empregar mapas topológicos. Estes podem ser entendidos como um mapa que trata essencialmente de localizações relativas de objetos presentes no mapa, portanto, tais mapas não lidam com localizações exatas. Os mapas topológicos podem ser representados por grafos, de modo que os locais se conectam no espaço através de suas arestas. Ainda vale a pena mencionar que os mapas topológicos são geralmente mais simples e exigem menos esforço computacional que os mapas métricos.

Neste trabalho, o foco central compreende os mapas topológicos visuais empregados para representar um dado ambiente, mapas que utilizam grafos para representar ligações entre pontos ou regiões, baseadas em informações visuais adquiridas por sensores de varredura tais como câmeras. Os mapas topológicos visuais também podem viabilizar a localização e navegação em condições de incertezas decorrentes de mudança de iluminação ou perspectiva da aquisição de imagens. Para haver o reconhecimento de um lugar a partir de um mapa topológico, pode-se lançar mão (encontra um substituto pois não é um termo tecnico)de um modelo para lidar com o problema de Reconhecimento Visual de Lugar, VPR (*Visual Place Recognition*). Esta classe de problemas tem por objetivo reconhecer lugares

anteriormente já visitados, atrás de características visuais. O reconhecimento deve ocorrer mesmo que o ambiente tenha mudado parcialmente, que as imagens sejam coletadas de diferentes perspectivas ou distâncias das imagens de referência.

Na literatura o VPR tem algumas famílias que são geralmente categorizadas como supervisionadas, não supervisionadas, semi-supervisionadas, paralelas e hierárquicas. A família escolhida como foco das soluções tratadas neste trabalho é a família de frameworks paralelos, justificada por sua habilidade de combinar múltiplos fluxos de informações (como descritores e dados visuais) para melhorar a precisão e robustez em ambientes desafiadores e dinâmicos. Esses métodos são particularmente eficientes em lidar com mudanças de aparência e alta complexidade dos ambientes, como mostrado pelos trabalhos com SeqSLAM e MPF, que são centrais no desenvolvimento deste trabalho.

O SeqSLAM utiliza sequências de imagens para reconhecer locais, mesmo em condições de grandes mudanças, como variações sazonais ou horários distintos. Pode-se fazer uma analogia do SeqSLAM como uma memória de viagem, que lembra o caminho baseado em uma sequência de fotos. Já o MPF (Multi-Process Fusion) combina diferentes métodos de análise de imagens, integrando múltiplos descritores para alcançar maior precisão e robustez em cenários complexos, como se o algoritmo fosse juntar várias opiniões para tomar uma decisão mais acertada. Esses dois modelos foram escolhidos porque, dentro da família de frameworks paralelos, eles se destacam por resolver desafios práticos de ambientes reais, além de serem exemplos de abordagens amplamente utilizadas.

Além disso, trabalhos como o RAT-SLAM, que utiliza um modelo inspirado no hipocampo para localização e mapeamento simultâneos, e o FAB-MAP, que aplica métodos probabilísticos para reconhecimento de lugares com base na aparência, destacam avanços significativos no campo de reconhecimento visual de lugares. Outro exemplo é o ORB-SLAM, que combina recursos visuais e geométricos para alcançar uma localização robusta e precisa. Esses estudos, junto com abordagens mais recentes como o NetVLAD, que utiliza aprendizado profundo para criar descritores compactos e eficazes, e o SegMap, que aplica aprendizado baseado em segmentos para mapeamento e localização, destacam avanços significativos no campo de reconhecimento visual de lugares.

Também foram utilizados alguns dos conjuntos de dados publicamente disponíveis, como o Database Nordland, para a avaliação de mudanças sazonais, e o Oxford RobotCar Dataset, que contém imagens de ambientes urbanos e suburbanos sob diversas condições.

1.1 OBJETIVOS

O objetivo deste trabalho é implementar técnicas de reconhecimento visual de lugares com base em imagens SeqSLAM e MPF em diferentes linguagens, como Python e C++. Além disso, analisar seus desempenhos em diferentes situações climáticas e de iluminação.

1.1.1 Objetivos específicos:

- Compreender, implementar e executar os métodos de reconhecimento visual de lugares presentes na literatura;

- Implementação de métodos de validação entre o modelo desenvolvido e com base na literatura;

- Comparar os modelos com base em métricas de desempenho tais como precisão, revogação, tempo de execução e erro médio quadrado(MSE)

1.2 ESTRUTURA DO TEXTO

O resto do trabalho é organizado da seguinte forma:

- O Capítulo 2 serão abordados conceitos a respeito de mapas métricos e topológicos, os algoritmos SeqSLAM e MPF e o que são os descritores;

- O Capítulo 3 detalha o passo a passo do funcionamento dos algoritmos escolhidos para análise;

- O Capítulo 4 apresenta os experimentos para validação dos métodos implementados com referência aos métodos encontrados na literatura e posteriormente comparação entre eles utilizando métricas de desempenho;

- O Capítulo 5 traz a conclusão do trabalho e uma reflexão sobre trabalhos futuros que podem ser realizados.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será apresentado uma contextualização sobre o problema de reconhecimento visual de lugares, conceitos de mapeamento e descritores, bem como algumas técnicas que são consideradas importantes para tentar resolver o problema de reconhecimento visual de lugares.

2.1 O QUE É UM LOCAL

Segundo LOWRY et al., (2016), um local é um segmento do mundo físico que possui relevância. Um lugar pode ser localizado através de coordenadas. Para um robô, um lugar pode ser uma área onde ele deve executar tarefas específicas, um ponto de referência para se localizar no ambiente ou até mesmo uma região a ser evitada.

2.2 COMO RECONHECER LOCAL

Para Yin et al, (2022), o primeiro módulo essencial para a localização de robôs no mundo real é o sensor adequado para as tarefas de reconhecimento de local. Isso dá, por conta que sensores diferentes, percebem o mundo de forma diferente. As considerações críticas para a seleção do sensor incluem 1) campo de visão, 2) riqueza geométrica e 3) robustez ao ruído ambiental. Sensores populares em reconhecimento de local incluem, LiDAR, RADAR e Câmeras dos mais diferentes tipos, ex: RGB, Estéreo, Térmico, etc. Para cada tipo de sensor pode ser aplicado um conjunto de técnicas que formam uma configuração base para atender a determinadas propriedades para a tarefa de reconhecimento de locais. Algumas configurações-base populares são descritas na Tabela 1.

Tabela 1: Comparação entre configurações de baseadas em diferentes sensores para o reconhecimento de locais

Sensor	**Propriedades para o reconhecimento de local**
LiDAR	Oferece uma medição do ambiente em 3D altamente precisa. No entanto, geralmente possui resolução mais baixa e não funciona bem em ambientes confinados ou áreas com textura limitada
RADAR	Não são afetados pela aparência ou por condições climáticas extremas. No entanto, eles têm baixa resolução e podem registrar clusters de ruído
Câmeras	São amplamente utilizadas em diversos métodos de reconhecimento de locais, mas são sensíveis a variações de perspectiva, mudanças na aparência e à presença de objetos dinâmicos. Sua abrangência pode variar até 360 graus.

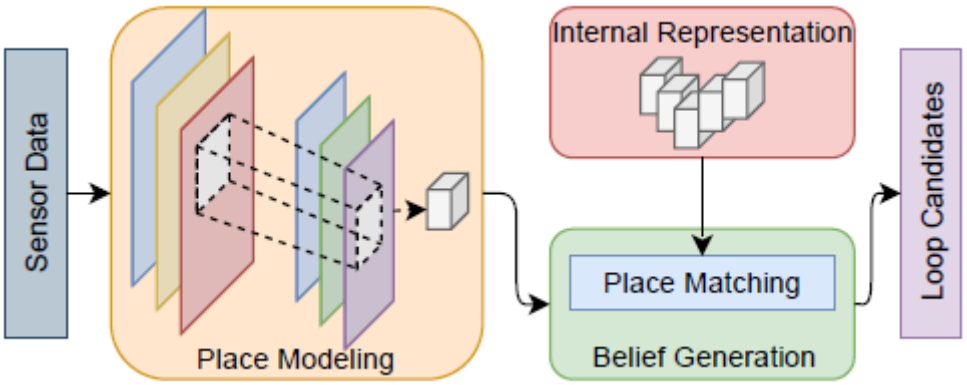
(**Fonte**: feita pelo autor, baseado em Yin et al, (2022))

Novamente segundo Yin et al, (2022), o reconhecimento de locais com base visual é a técnica mais investigada na área de reconhecimento de locais.

2.3 RECONHECIMENTO VISUAL DE LOCAIS

De acordo com Barros et al. (2021), o processo de reconhecer um local pode ser dividido em três partes. *Conforme ilustrado na Figura 1 representação de lugares, mapeamento de lugares e geração de crenças.*

Figura 1:Pipeline genérico de reconhecimento de locais com os seguintes módulos: representação de lugares, mapeamento de lugares e geração de crenças.



Fonte:Barros et al. (2021)

Cada um dos módulos, com algumas técnicas possíveis de implementação, serão descritos a seguir:

2.3.1 Representação de locais

Este módulo tem a função de mapear os dados de um espaço de sensor para um espaço descritor. Os dados sensoriais das câmeras são utilizados para modelar o ambiente circundante, obtendo informações relevantes por meio da extração de características.

2.3.1.1 Sem extratores

Estas técnicas são usadas para representar locais sem fazer uso de descritores ou de redes neurais, fazer uso de operações matemáticas ou filtros e atuam em diferentes contextos:

Sum of Absolute Differences (SAD): É uma métrica usada em processamento de imagens e visão computacional. Ele quantifica a diferença entre duas imagens ou regiões de imagens.

2.3.1.2 Abordagem Clássica

A abordagem clássica, utiliza extratores de características (ou descritores) que são técnicas que transformam as imagens captadas por um sensor em um conjunto de dados que podem ser interpretadas através de algoritmos, ele faz isso através de reconhecer pontos de interesses e padrões específicos nas imagens que queremos tratar. Estas características podem ser, por exemplo,cantos, bordas e texturas. Um ponto importante sobre os descritores é que eles são imunes a variação de várias características, o que os torna capazes de se adequar a diferentes cenários. Utilizar técnicas de rotação e translação para determinar, qual imagem vinda da entrada do sistema tem o conjunto de descritores correspondentes aos da imagem no banco de dados.

Tipos de descritores:

1. **Descritores Globais**: Esses métodos representam a imagem usando um descritor geral calculado a partir de toda a informação visual disponível. São úteis para classificação de imagens e recuperação de informações

2. **Descritores Locais**: Aqui, identificamos pontos de interesse na imagem e descrevemos uma área ao redor desses pontos para identificá-los em outras

imagens. Esses descritores são úteis para correspondência de imagens e reconhecimento de objetos. Alguns descritores locais mais comuns são:

SIFT: Ele extrai os descritores de áreas ao redor dos pontos de interesse, considerando a orientação, não considera nem escola nem rotação.

SURF: Visa ser mais rápido e eficiente do que o SIFT, utiliza um descritor baseado em caixas retangulares e utiliza aproximações para calcular os gradientes. Ele também detecta pontos de interesse em várias escalas.

ORG: Utiliza descritores em forma binária, o que o torna mais leve e eficiente, é otimizado para utilizar menos recursos de memória e apresentar bom desempenho, mas pode não apresentar bom resultados em certas situações.

Bag of Words (BoW): Nesse método, as características locais da imagem são quantizadas com base em um conjunto de modelos de características chamado “dicionário visual”. As imagens são representadas como histogramas das ocorrências de cada palavra no dicionário. O BoW é comumente usado em tarefas de recuperação de imagens e classificação.

3. **Descritores Combinados globais e locais. Esses** métodos utilizam várias técnicas descritas acima em conjunto para criar uma nova solução. Por exemplo, combinar descritores globais e locais pode melhorar a robustez e a precisão da representação da imagem.

2.3.1.3 Tecnicas baseadas em Deep learning e CNN

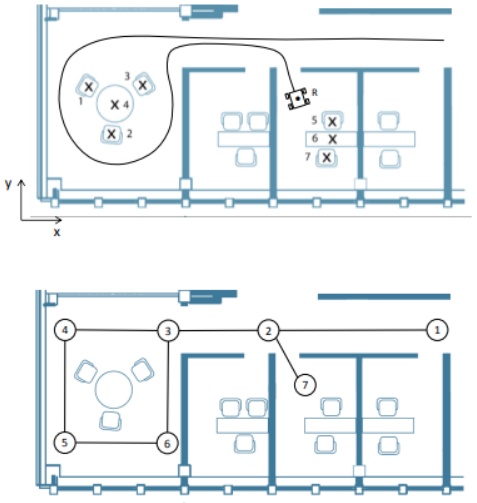
De acordo com Barros et al. (2021), as redes neurais convolucionais (CNNs) são amplamente utilizadas para extrair características em tarefas de reconhecimento de lugares. As camadas convolucionais das CNNs são essenciais para capturar padrões locais nos dados, permitindo que a rede identifique características importantes, como bordas e texturas. Além disso, as CNNs podem ser usadas como extratores de características pré-treinadas, evitando a necessidade de treinar a rede do zero. O artigo também destaca a importância do NetVLAD para o aprendizado supervisionado end-to-end. O NetVLAD é uma arquitetura de rede neural frequentemente usada para extrair características em tarefas de reconhecimento de lugares.

Além disso, também pudessem destacar as vantagens das abordagens não supervisionadas no reconhecimento de lugares, especialmente para aplicações de domínio cruzado. Essas abordagens de aprendizado profundo podem aprender a extrair características úteis de imagens sem depender de rótulos de treinamento, o que é particularmente útil quando se trabalha com grandes conjuntos de dados não rotulados ou ao reconhecer lugares em diferentes domínios ou condições.

2.3.2 Mapeamento de locais

Segundo Pacheco et al. (2018) o mapeamento integra as observações parciais do entorno em um único modelo consistente. Pode-se dividir os mapas em dois principais tipos, mapas métricos e mapas topológicos.

Figura 2: Principais paradigmas de representação do ambiente. Em cima: mapa métrico. Note-se que os objetos identificados são apenas ilustrativos, pois, tipicamente, os mapas representam estruturas de mais baixo nível, como arestas e cantos. Em baixo: mapa topológico. Aqui, a posição dos nós sobre a planta é também ilustrativa, pois estes mapas podem não incluir informação métrica.



Fonte: CAMPOS, F. M. M. O (2015)

2.3.2.1 Mapas Metricos

O ambiente é mapeado por meio de um sistema de coordenadas que detalha a posição de objetos e barreiras, utilizando informações primárias de odometria e propriedades geométricas. Essa técnica de mapeamento e localização pode alcançar uma precisão extremamente alta, fornecendo uma representação detalhada do espaço. No entanto, essa precisão vem com o custo de exigir um grande poder de processamento computacional e depende de forma crucial da identificação correta dos objetos que são re-observados

2.3.2.2 Mapas Topológicos

São representações simplificadas do ambiente, focando apenas nas informações essenciais e eliminando detalhes desnecessários. Diferentemente dos mapas métricos, os mapas topológicos não possuem escala fixa, o que significa que a

distância e a direção podem variar, mas a relação topológica entre pontos é preservada. Estes mapas destacam as conexões e relações entre elementos geográficos sem se preocupar com a precisão absoluta das medidas.

2.3.3 Geração de crença

A Geração de Crenças é responsável por criar uma distribuição de crenças, representando o quão provável ou confiante o sistema está de que os dados de entrada correspondem a um local no mapa. Em termos mais simples, a "geração de crença" refere-se à capacidade do sistema de inferir ou estimar a probabilidade de uma determinada localização ser o local atual com base em várias fontes de informação sensorial e conhecimento prévio, como dados de GPS, câmeras e sensores de proximidade.

Existem diferentes maneiras de calcular as pontuações de crença. Alguns métodos incluem abordagens baseadas em comparação de quadro a quadro de imagens, sequências de quadros, métodos hierárquicos, grafos ou abordagens probabilísticas:

2.3.3.1 Método quadro a quando de imagens

Os métodos quadro a quadro calculam as pontuações de crença comparando cada quadro individualmente, utilizando medidas de similaridade entre os quadros. Esse processo envolve a comparação direta de cada quadro com os outros disponíveis, utilizando métricas como distância euclidiana, distância de Hamming ou similaridade de cosseno. Para encontrar correspondências entre os quadros, técnicas de busca de vizinho mais próximo, como árvores KD ou árvores Chow Liu, são frequentemente empregadas. Este método é eficaz para reconhecimento de locais e sistemas de rastreamento de objetos em tempo real devido à sua simplicidade e capacidade de lidar com dados de entrada em tempo real. No entanto, ele pode ser sensível a ruídos e variações na cena, e seu desempenho pode diminuir em ambientes complexos ou com grandes variações de iluminação.

2.3.3.2 Método baseado em Sequência de quadros de imagens

Neste método, as pontuações de crença são calculadas com base em sequências de observações consecutivas, considerando a ordem temporal das observações. As sequências de observações são comparadas utilizando técnicas como minimização de fluxo de custo em uma matriz de similaridade, levando em conta a ordem temporal. Modelos probabilísticos, como Modelos Markovianos Ocultos (HMMs) ou Campos Aleatórios Condicionalmente (CRFs), são comumente utilizados para modelar a transição entre diferentes estados ou observações na sequência. Isso permite capturar padrões temporais complexos e lidar com variações temporais e ruídos nos dados de entrada. No entanto, este método pode exigir uma grande quantidade de dados para treinamento eficaz e pode ser computacionalmente intensivo, especialmente para modelos probabilísticos mais complexos.

2.3.3.3 Métodos Hierárquicos

Os métodos hierárquicos combinam múltiplas abordagens de correspondência em um único framework, permitindo uma busca mais eficiente por correspondências. Eles empregam uma abordagem em duas etapas: uma camada grosseira seleciona os principais candidatos, enquanto uma camada fina escolhe a melhor correspondência entre esses candidatos. Não há algoritmos ou modelos específicos associados aos métodos hierárquicos, já que podem integrar diferentes técnicas de correspondência em cada camada. Essa estratégia melhora a eficiência computacional ao reduzir o número de comparações necessárias, potencialmente fornecendo resultados mais robustos em ambientes desafiadores. No entanto, sua eficácia depende da seleção apropriada das abordagens de correspondência em cada camada, o que pode representar um desafio na busca pelo equilíbrio ideal entre eficiência e precisão.

2.4 DESAFIOS NO RECONHECIMENTO VISUAL DE LOCAIS

Para LOWRY et al., (2016) o problema do Reconhecimento Visual de Locais pode ser definido como um desafio que combina uma série de fatores:

- A aparência de um local pode mudar drasticamente.
- Múltiplos locais em um ambiente podem parecer muito semelhantes, um problema conhecido como “aliasing perceptual”.
- Os locais podem nem sempre ser revisitados do mesmo ponto de vista e posição que antes.

2.4.1 Mudança de aparência de um Local

Em operações de longo prazo, o ambiente pode sofrer alterações drásticas na aparência devido a fatores como mudança de iluminação, clima e horário do dia. Diferenças entre duas observações do mesmo lugar podem ser tão significativas que sedificulta identificar semelhanças. Um exemplo de alteração é a mudança de iluminação do dia para a noite: pontos de luz antes apagados podem se tornar pontos brilhantes, ou a textura de uma árvore pode se transformar em um borrão escuro uniforme (MILFORD; WYETH, 2012; SÜNDERHAUF; NEUBERT; PROTZEL, 2013).

Figura 3: Observações de um mesmo local extraídas em diferentes ocasiões

(**Fonte**: Milford e Wyeth (2012).)

2.4.2 Aliasing perceptual

Mas especificamente no problema de VPR, o aliansing perceptual acontece quando duas imagens de lugares distintos parecem semelhantes. Isso pode levar a erros de localização do robô. Um exemplo disso é um robô com uma camera que está explorando um parque e vê dois bancos de madeira cercados por árvores similares. Apesar de estarem em locais diferentes do parque, as imagens dos dois bancos parecem tão parecidas que o robô pode confundi-las e achar que está vendo o mesmo lugar.

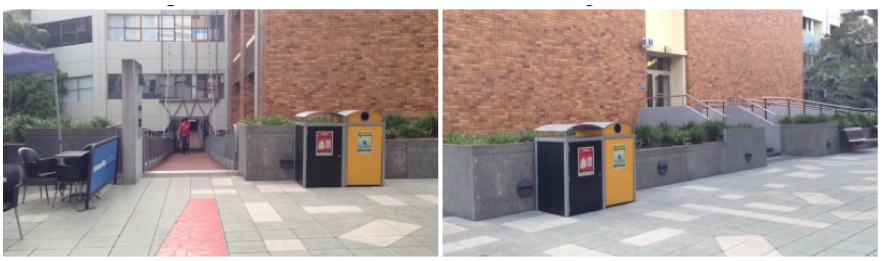
Figura 4:Observações semelhantes extraídas em diferentes locais

(**Fonte**: Milford e Wyeth (2012).)

2.4.3 Mudanças de ponto de vista

Também é um desafio enfrentado o de reconhecer um mesmo local quando observado de diferentes pontos de vista, como ilustrado na Figura 1.xxx. Com a mudança no ponto de vista, além de objetos em comum serem observados a partir de ângulos distintos, também pode surgir nova informação não compartilhada entre as diferentes observações.

Figura 5: Mesmo local visto de diferentes pontos de vista



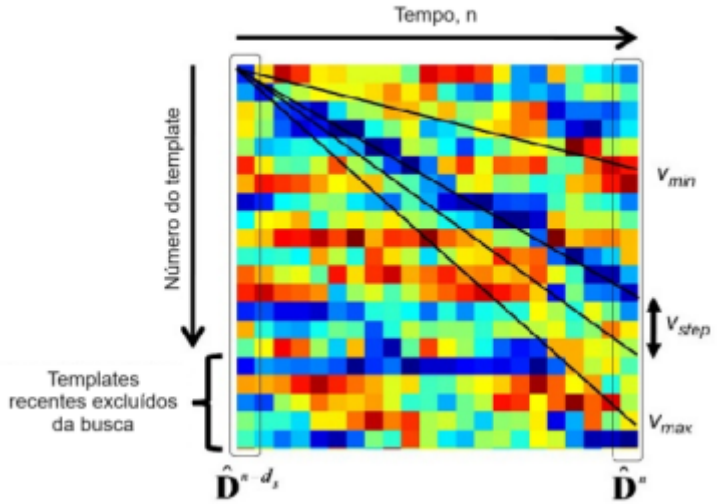
(**Fonte**: Glover (2014).)

2.5 TÉCNICAS PARA SOLUCIONAR O PROBLEMA DE RECONHECIMENTO VISUAL DE LOCAIS

2.5.1 Seqslam

O SeqSLAM, introduzido por Milford e Wyeth em 2012, é um método que provê uma solução eficaz para o problema de VPR. Ele determina a melhor localização em uma sequência de imagens conhecidas, utilizando mapas topológicos e o SAD para representar os locais. Este método é especialmente útil em ambientes dinâmicos, como diferentes condições de tempo, horários do dia e estações do ano. Ele alinha uma sequência de teste com uma de referência para encontrar as melhores correspondências entre elas, capturando com precisão as semelhanças geométricas contínuas, mesmo diante das mudanças condicionais mencionadas.

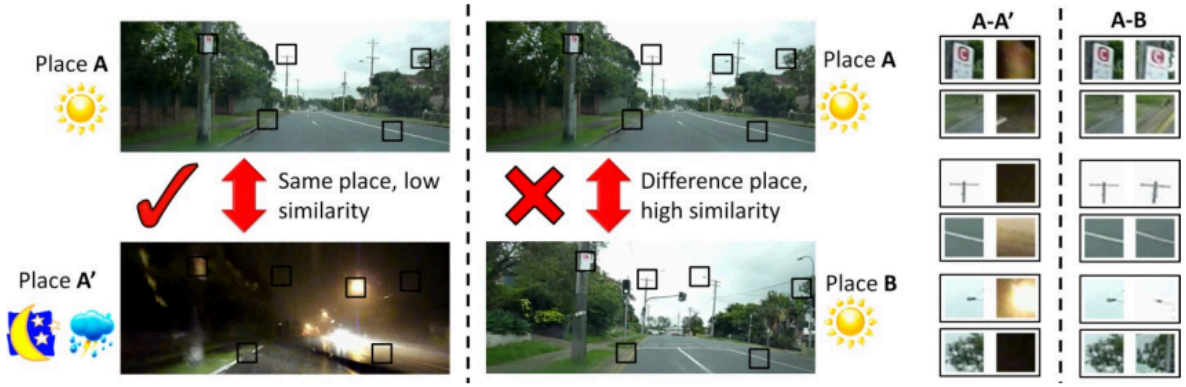
Figura 6: Modelo do problema de SLAM ilustrado por meio de um grafo



(**Fonte**: Milford e Wyeth (2012).)

Ele busca encontrar modelos nas vizinhanças locais que melhor correspondem à imagem atual. A localização mais precisa é encontrada verificando a sequência mais coerente entre estes correspondentes locais. Este processo gera muitos correspondentes locais candidatos para cada intervalo de tempo analisado, por sua vez, estes são analisados em busca do que apresenta uma trajetória de pontuação mínima, ou seja, daquele que melhor corresponde.

Figura 7: O problema da mudança perceptual. O lugar A e B são lugares diferentes, mas parecem muito mais semelhantes do que o lugar A a si mesmo durante uma noite chuvosa (A'), tanto em uma imagem inteira quanto em uma base de características individuais. Nossa abordagem usa correspondência local dentro de sequências para corresponder com sucesso A a A', mas não A a B.

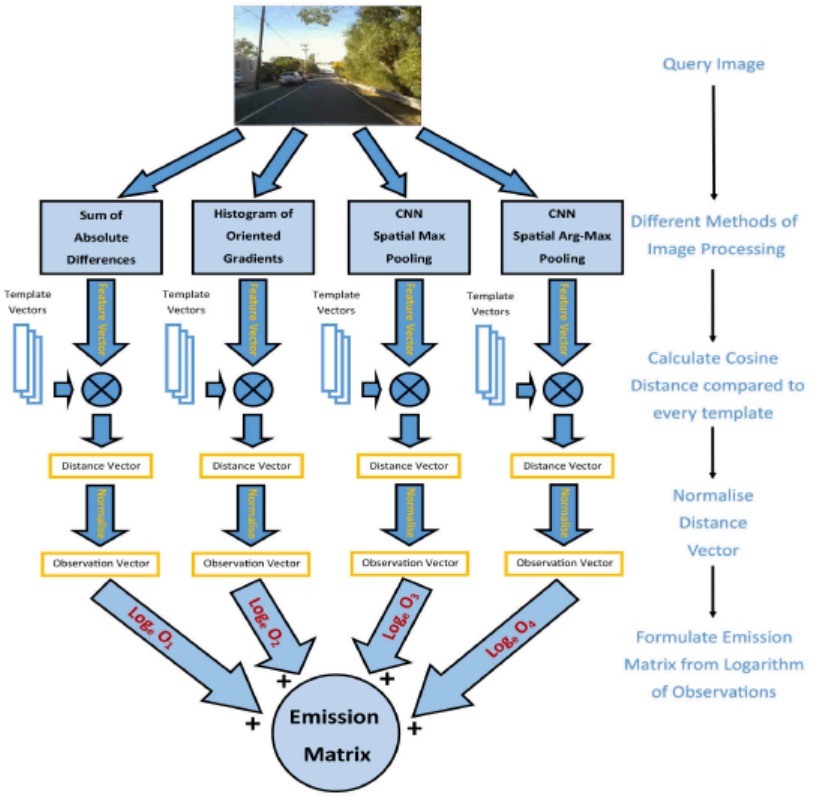


(**Fonte**: Milford e Wyeth (2012).)

Por conta disso o SeqSLAM tem um custo computacional elevado devido à sua rotina de verificação de correspondências. Isso ocorre porque o método calcula a similaridade entre a sequência das últimas imagens capturadas em relação às sequências de todas as imagens armazenadas. Esse custo é proporcional ao tamanho da sequência e ao número de imagens do mapa.

2.5.2.MPF

Figura 8: Este diagrama mostra a formulação da matriz de emissão usando quatro métodos de processamento de imagem. Um vetor de características é extraído para cada método, que é então comparado aos modelos de banco de dados para produzir um vetor de distância. Logaritmos e normalização são usados para formatar cada vetor antes de criar a matriz de emissão.



(**Fonte**: S. Hausler, A. Jacobson and M. Milford(2019)

É um método de reconhecimento visual de lugares que utiliza múltiplos métodos de processamento de imagens aplicados a um único conjunto de dados de entrada, ela propõe que um dos métodos tenha sempre um bom desempenho em algum dos múltiplos ambientes possíveis, em seguida, serão utilizados métricas de qualidade para determinar qual das técnicas deverão ser dados maior peso para aquele ambiente no resultado final.

O método utiliza uma técnica dinâmica de correspondência de sequência que permite latências de localização menores, através da análise de métricas de qualidade de reconhecimento ao revisitar locais familiares.

Diferente do SeqSLAM que utiliza a soma de diferenças absolutas (SAD) para dá um match entre os melhores correspondentes, o MPF utiliza descritores para determinar pontos de interesse em uma imagem.

2.6 Métricas para Análise no Reconhecimento Visual de Lugares

Uma parte significativa dos estudos realizados em VPR emprega precisão e revocação como métricas de análise. Nesse cenário, cada comparação feita entre imagens de referência e imagens de consulta pelo sistema gera um índice de correspondência. Para determinar se essa correspondência é verdadeira ou não, é necessário estabelecer um limiar de confiança. Com a variação desse limiar, podem-se calcular os valores de precisão e revocação, definidos, respectivamente, pelas equações abaixo:

$$Precisão = \frac{TP}{TP+FP}$$

$$Revocação = \frac{TP}{TP+FN}$$

Nessas equações, a quantidade de correspondências identificadas corretamente como verdadeiras é representada por TP (verdadeiro positivo); a quantidade de correspondências identificadas corretamente como falsas é representada por TN (verdadeiro negativo); a quantidade de correspondências erroneamente identificadas como verdadeiras é designada como FP (falso positivo); e, finalmente, a quantidade de correspondências erroneamente identificadas como falsas é classificada como FN (falso negativo). Ademais, ao variar o limiar de confiança, é possível traçar um gráfico que relaciona precisão e revocação.

Além disso, em aplicações embarcadas, é altamente relevante obter informações sobre a utilização ou não de GPU pelo sistema VPR, o consumo de memória RAM, o tempo de execução e o tamanho dos descritores. Esses fatores desempenham um papel essencial na definição da viabilidade de uso do sistema em aplicações de navegação autônoma, como no caso do tempo de execução, que apresenta desafios importantes.

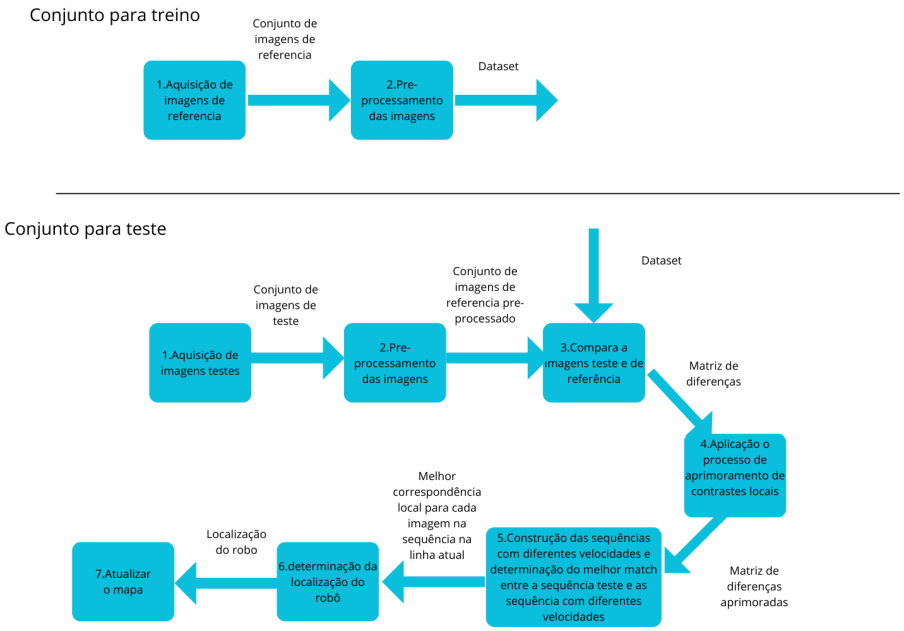
3 METODOLOGIA

Neste capítulo, será descrito o funcionamento passo a passo de algumas técnicas para resolver o problema de reconhecimento visual de lugares.

3.1 SEQSLAM

Segundo Milford e Wyeth, podemos descrever o passo a passo do funcionamento do algoritmo do seqslam assim:

Figura 9: Diagrama teórico em bloco da execução do SeqSlam



(**Fonte**: feita pelo autor)

3.1.1 Aquisição de imagens

Usando uma câmera acoplada ao robô, obter uma série de fotos do cenário atual.

3.1.2 Pré-processamento das imagens:

Visando diminuir ruídos, padronizar as imagens e destacar características relevantes, é feito o pré-processamento das imagens.

1. Converter as imagens em escala de cinza: A escala de cinza é uma forma de representar as imagens usando apenas tons de cinza, que variam do preto ao branco. A escala de cinza pode simplificar o processamento das imagens, reduzindo a quantidade de dados e eliminando as informações de cor.

2. Redimensionamento da Imagem: O redimensionamento é feito para ajustar o tamanho das imagens para um novo formato específico. Esta é uma prática comum durante o pré-processamento pois ajuda na economia de recursos computacionais e acelera o processo de treinamento do modelo.

3. Recorte da Imagem:É feito o recorte das imagens, se necessário, para obter um campo de visão aproximadamente correspondentes entre os diferentes conjuntos de dados do mesmo dataset, pois estes conjuntos de dados podem ter sido obtidos com modelos de câmeras diferentes, com posições dos veículos em relação a pista diferentes, ou com a altura da câmera em relação ao solo diferente, tudo isso pode acabar gerando diferença no campo de visão dos conjuntos de dados no mesmo datasets.

4. Normalização de patches: A imagem original então é dividida em regiões quadradas menores com comprimento lateral L, denominados patches, estes patche são então normalizados, esta é uma prática normal durante o pré-processamento feita para realçar os detalhes em uma imagem.

5. Construção da matriz de pré-processamento para cada conjunto de dados do dataset: No fim, as imagens são adicionadas na matriz de pré-processamento. pois trabalhar com uma única estrutura de dados que guarde informações sobre cada imagem por conjunto do dataset é mais prático é economizar espaço na memória do sistema.

3.1.3 Comparação entre as imagens nos conjuntos de referência e de teste

Neste passo agora será feita a comparação entre as imagens presente no conjunto de dados de referência com as imagens presentes no conjunto de teste.

3.1.4 Aplicação do processo de aprimoramento de contrastes locais

Em seguida, é realizado um dos principais diferenciais do SeqSLAM em relação a outros métodos para resolver o VPR. Ao invés de buscar um único modelo que melhor corresponda à imagem atual (melhor correspondência global), o SeqSLAM procura identificar todos os modelos nas vizinhanças locais que melhor se ajustem à imagem atual (melhor correspondência local). Para atingir esse objetivo, aplica-se um processo de aprimoramento de contraste local. Isso é relevante, pois dá mais ênfase às imagens com melhor correspondência entre si.

3.1.5 Construção das sequências com diferentes velocidades e determinação do melhor match entre a sequência teste e as sequências com diferentes velocidades

Nesta etapa é feito a Construção das sequências com diferentes velocidades com base na matriz de diferenças aprimorar e a determinação do melhor match entre a sequência teste e as sequências com diferentes velocidades, isso é feito aplicando o método de reconhecimento de sequência localizada. Ele é eficiente em termos de tempo, pois leva em consideração a variação temporal dos dados, o que é especialmente útil em aplicações de tempo real onde os dados de entrada mudam dinamicamente ao longo do tempo. Além disso, é robusto a variações de velocidade, pois ao variar a velocidade da trajetória entre Vmin e Vmax, o método pode lidar com variações na velocidade do objeto em movimento. Isso é útil em cenários onde a velocidade do objeto pode mudar ao longo do tempo. Outra vantagem é que ele evita correspondências com modelos recentemente aprendidos, o que pode ajudar a evitar falsos positivos e melhorar a precisão da correspondência. Além disso, é capaz de reconhecer sequências de lugares familiares, o que pode ser útil em aplicações como navegação de robôs ou rastreamento de objetos. Finalmente, o método permite ajustar vários parâmetros, como o alcance da janela deslizante (Rwindow) e o fator μ, para otimizar a correspondência de modelos para diferentes cenários de aplicação. No entanto, a eficácia deste método pode variar dependendo do cenário de aplicação específico e dos dados de entrada.

3.1.6 Determinação da localização do robô a partir da sequência com melhor correspondência

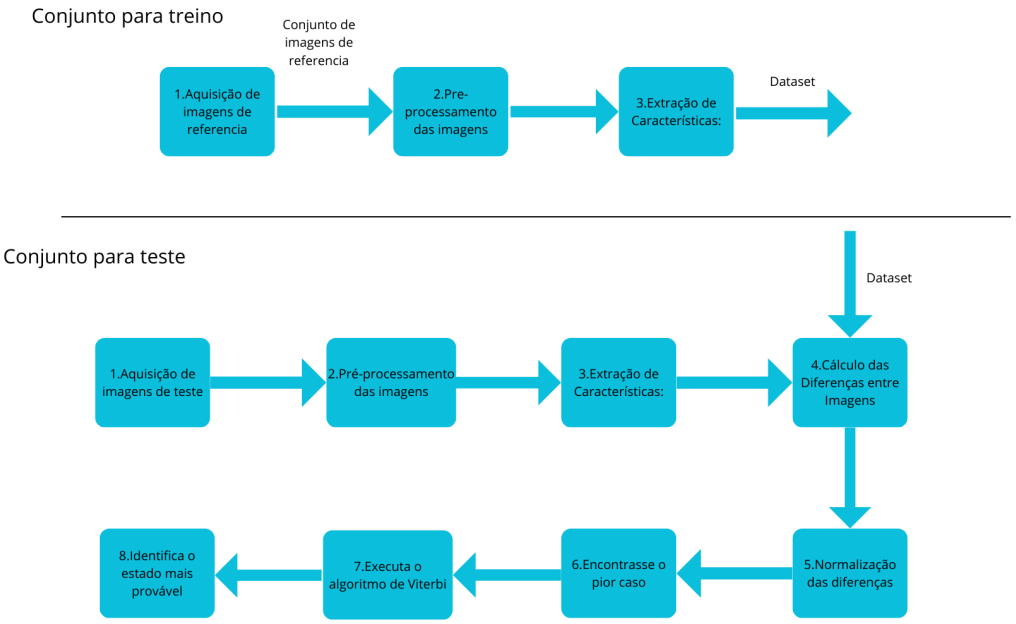
Isso é feito encontrando a maior e mais consistente cadeia de correspondências locais através de algoritmos de programação dinâmica.

3.1.7 Atualizar o mapa

Atualiza a posição do robô no mapa

3.2 MPF

Figura 10: Diagrama teórico em bloco da execução do SeqSlam



(**Fonte**: feita pelo autor)

3.2.1 Aquisição de imagens

É o ponto inicial do algoritmo e refere-se à coleta das imagens que serão usadas tanto como referência quanto para consulta. Esta etapa define a base de dados para o processamento, envolvendo a obtenção e organização das imagens de forma que estejam prontas para a análise nas próximas etapas.

3.2.2 Pré-processamento das imagens

É realizado o recorte das imagens, em seguida o redimensionamento para uma resolução correspondente a entrada de cada um dos descritores:

- CNN e CNN-D: 227x227 pixels para a camada de entrada HybridNet.
- HOG: 640x320 pixels
- SAD: 64x32 pixels

3.2.3 Extração de Características

É realizado a extração das características pelo método correspondente para cada um dos diferentes descritores (como CNN, CNN-D, HOG e SAD), em seguida e o armazenamento dessas características em vetores apropriados para futuras comparações.

- CNN: Calcula os valores máximos da matriz de ativação.
- CNN-D: Encontrasse as coordenadas do maior valor de ativação
- HOG: Extrai vetores de gradiente das imagens com células de 32×32 pixels e 9 bins de orientação.
- SAD: similar ao SeqSlam, após reduzir a resolução da imagem realiza normalização de patches.

3.2.4 Cálculo das Diferenças entre Imagens

As imagens processadas são comparadas às imagens de referência usando diferentes métricas:

- CNN: Pooling espacial piramidal das ativações, normalizado e comparado com distância de cosseno.
- CNN-D: Comparação baseada apenas nas coordenadas (x, y) das ativações máximas, utilizando distância euclidiana.
- HOG: Calcula-se a distância dos cossenos entre vetores HOG.
- SAD: Diferença absoluta entre patches normalizados.

3.2.5 Normalização das diferenças

Os vetores de diferença entre imagens são normalizados para ficarem entre -0.001 e 0.999, garantindo que a melhor correspondência seja representada pelo maior valor. Um limiar O_{thresh} é aplicado para penalizar observações de baixa confiança.

3.2.6 Encontrasse o pior caso

O processo é utilizado para identificar e registrar casos críticos nas matrizes de observação (O1, O2, O3 e O4) dentro de uma janela específica de dados (Rwindow). Ele prioriza o monitoramento de cenários desafiadores, como discrepâncias significativas entre observações e valores esperados.

Identificar o pior caso permite compreender os maiores desvios ou inconsistências, ajustando o algoritmo e melhorando o desempenho do sistema. Além disso, o processo possibilita a análise da frequência dessas situações críticas, facilitando refinamentos em áreas problemáticas e tornando a análise mais robusta e precisa.

3.2.7 Executa o algoritmo de Viterbi

O algoritmo de Viterbi é utilizado para encontrar a sequência mais provável de estados ocultos em um Modelo de Markov Oculto (HMM), a partir de uma sequência observada. Ele é amplamente aplicado em áreas como reconhecimento de padrões,

processamento de linguagem natural e bioinformática devido à sua eficiência e capacidade de lidar com dados sequenciais.

Primeiramente, o algoritmo calcula as probabilidades iniciais de cada estado oculto com base na primeira observação. Em seguida, avalia iterativamente as melhores probabilidades de chegada a cada estado para as observações seguintes. Durante esse processo, registra as transições mais prováveis, facilitando a reconstrução da sequência de estados.

Após processar todas as observações, o algoritmo retrocede pelas transições registradas para reconstruir a sequência mais provável. Esse método evita a combinação exaustiva de todas as possibilidades, otimizando os cálculos.

No caso analisado, o Viterbi foi adaptado para usar matrizes de observação *(O1, O2, O3 e O4)* e logaritmos, aumentando a eficiência e a sensibilidade na análise de qualidades. A inclusão de janelas de análise e da taxa de variação *(Rate of Change, ROC)* aprimora a identificação da melhor sequência, permitindo lidar com características dinâmicas de forma eficaz.

3.2.8 Identifica o estado mais provável

Com base no algoritmo de Viterbi, o processo envolve ajustar a qualidade calculada para torná-la proporcional ao comprimento da sequência analisada, garantindo que o resultado seja representativo e não distorcido por diferenças no tamanho das sequências. Além disso, o último estado da sequência mais provável é identificado como o ponto final mais relevante.

Esse ajuste na qualidade é essencial para obter uma métrica equilibrada que reflete com precisão a confiabilidade da sequência processada. Já a identificação do estado final fornece informações úteis sobre o desfecho da análise, sendo fundamental para entender a sequência mais provável de eventos que ocorreram, especialmente em sistemas baseados em modelos de Markov. Esses passos são cruciais para otimizar resultados e garantir maior consistência na interpretação das observações analisadas.

4 RESULTADOS

Neste capítulo, foi realizada a implementação das técnicas descritas no capítulo anterior em Python e C++, seguida pela comparação. Para isso, conduzimos um estudo minucioso do algoritmo original implantado no MATLAB, conforme descrito no artigo base do SeqSlam e do MPF. Em seguida, comparamos cada parte do código enquanto o implementámos em Python e C++. Durante esse processo, foram observadas as nuances e diferenças entre as duas implementações, garantindo a fidelidade aos métodos propostos nos artigos.

Foi executada a implementação original dos modelos em MATLAB e em seguida comparada passo-a-passo com a implementação dos desenvolvidos em Python e C++. O computador utilizado para os testes possui um processador i5 de 10 geração padrão e 8 gigabytes de memória RAM.

4.1 SEQSLAM

4.1.1 Aquisição de imagens

Para avaliar os métodos propostos, selecionamos alguns datasets desafiadores que apresentam mudanças de aparência e variações de ponto de vista. Esses datasets são amplamente utilizados na área, o que nos permite comparar os resultados obtidos com nossa proposta em relação a outros trabalhos. Para nossos testes, é fundamental que todos os datasets contenham conjuntos de imagens organizadas conforme a sequência de coleta. Mais detalhes podem ser encontrados a seguir

- Nordland

O dataset Nordland (SÜNDERHAUF; NEUBERT; PROTZEL, 2013) consiste em quatro vídeos que documentam uma viagem de trem de 728 km pelo norte da Noruega. Cada vídeo foi gravado em uma estação do ano diferente, permitindo a captura de mudanças significativas na aparência do ambiente. Para os experimentos realizados, foram utilizados conjuntos de imagens extraídas dos vídeos correspondentes a cada estação do ano. As imagens foram extraídas com uma frequência de 1 imagem a cada 100 imagens do dataset, resultando em um total de 357 imagens por conjunto de cada estação.

Figura 11: amostra de imagens do Nord - amostra de imagens

(**Fonte**: SÜNDERHAUF; NEUBERT; PROTZEL, 2013)

4.1.2 Pré-processamento das imagens

Para garantir que os modelos funcionem de maneira eficaz e produzam resultados confiáveis foi realizado o pré-processamento das imagens do dataset analisado, conforme descrito anteriormente. Para exemplificar o pré-processamento, será utilizada uma imagem do circuito de nordland encontrada na internet. Esta imagem tem tamanho original de 1200x750.

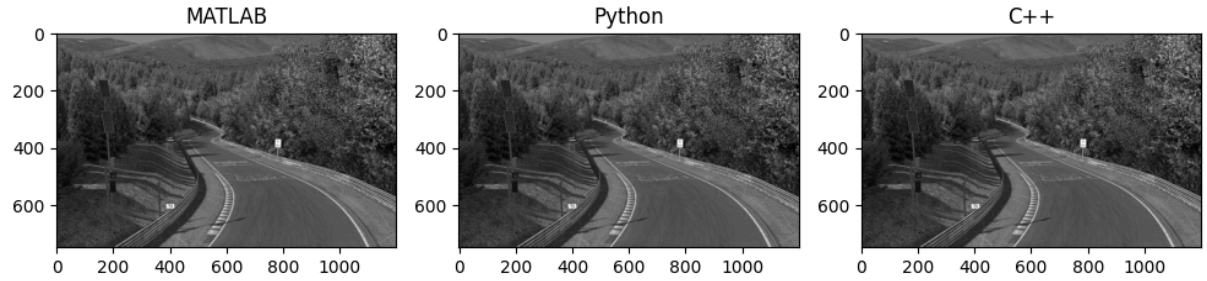
Figura 12: pista de Nürburgring

(**Fonte:**https://www.revistacarros.pt/10-melhores-tempos-no-nurburgring-para-carros-de-producao/)

Em seguida, será mostrada a imagem de referência após cada etapa do pré-processamento, em matlab, como base da implantação original, e em python e C++, que foram implantados e serão comparados com a implementação original, e também o valor do erro médio quadrado (EQM, ou MSE em inglês) para poder demonstrar a similaridade entre a implantação base as outras duas implementações.

1. Converter as imagens em escala de cinza: Para fazer isso aplica-se funções que calculam a média ou o peso das componentes vermelha, verde e azul de cada pixel, e atribuem um valor de cinza correspondente.

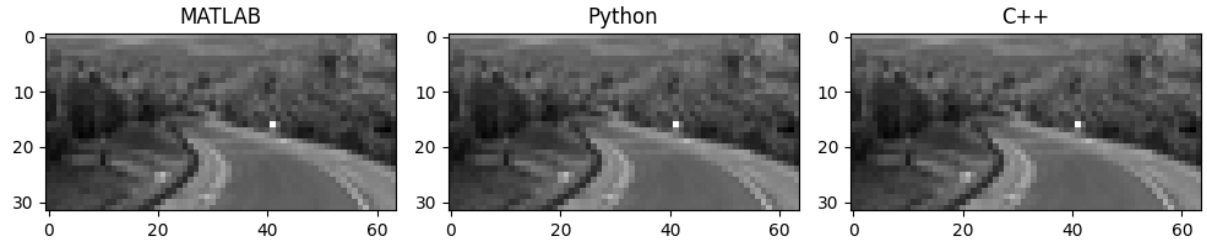
figura 13: Comparação entre a imagem original passada para a escala de cinza em MATLAB, Python e C++.



MSE MATLAB X Python= 0.00000000 MSE MATLAB X C++ = 0.23125462

2. Redimensionamento da Imagem: Em seguida foi feito o redimensionamento da imagem de 1200x750 para 64x32, a imagem é redimensionada aplicando algum método de interpolação como BILINEAR, BICUBIC, LANCZOS, etc.

Figura 14: Comparação entre a imagem na escala de cinza re-dimensionalizada em MATLAB, Python e C++.



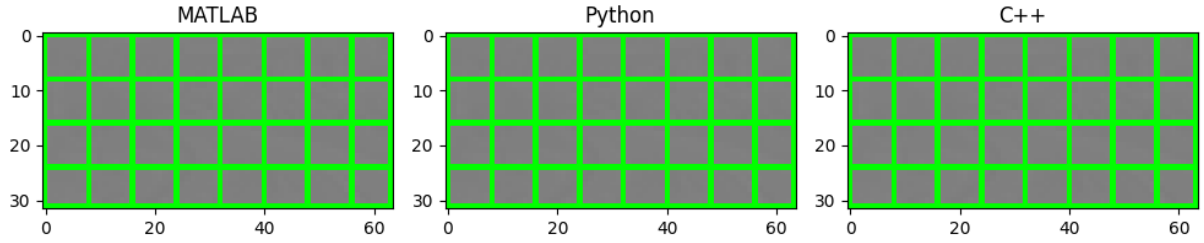
(**Fonte**: feita pelo autor)

MSE MATLAB X Python= 0.35156250 MSE MATLAB X C++ = 0.21785764

3. Recorte da Imagem: Para os conjuntos de dados dos datasets analisados, não foi necessário realizar este passo.

4. Normalização de patches: Na etapa subsequente da normalização de patches, a imagem é segmentada em quadrados menores, cada um com lado L, denominados ‘patches’. Cada patch é então transformado em um vetor e normalizado. A normalização é realizada aplicando uma das seguintes técnicas: para cada pixel, subtrai-se a média e divide-se pelo desvio padrão dos pixels circundantes, ou então, o ‘patch’ pode ser escalonado linearmente para o intervalo compreendido entre seus valores mínimo e máximo. A técnica de normalização empregada é determinada pelo modo selecionado. No caso em análise, o primeiro modo foi o escolhido

Figura 15: Comparação entre a imagem clipadas e normalizadas MATLAB, Python e C++



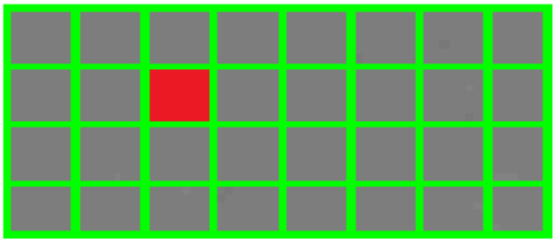
(**Fonte**: feita pelo autor)

MSE MATLAB X Python= 0.10187554 MSE MATLAB X C++ = 0.08417525

Observa-se que o MSE apresenta um valor diferente de zero, o que ocorre devido às variações na maneira como cada biblioteca, em diferentes linguagens, implementa as conversões. Isso gera problemas de Aliasing, uma forma de contornar isso é comparando cada implementação com e sem o anti-aliasing ligado e utilizando métodos de interpolação semelhantes entre as implantações base e desenvolvidas.

5. Construção da matriz de pré-processamento para cada conjunto de dados do dataset: O passo seguinte será juntar cada um dos cortes normalizados da imagem em uma única coluna da matriz de pré-processamento e empilhá-los cada um em sequência e cada imagem da sequência de imagens do dataset ficará representado em sequência na matriz da pré-processamento. Toma-se, por exemplo, a imagem de referência pré-processada. Até agora ela tem resolução 64x32 e pode estar em alguma posição a de uma sequência de n imagens, e pegasse um de seus cortes normalizados de forma aleatória, ele ocupa da linha b até a linha b+63 da coluna a na matriz de pré-processamento.

Figura 16: Figura ilustra a construção da matriz de pré-processamento:(a) representação dos cortes na imagem original(b) corte normalizado aleatório destacado como uma matriz,(c) Corte normalizado transformado em um vetor coluna e (d) simulação da matriz de pré-processamento destacando o corte normalizado em sua posição.



(a)

126	126	128	129	128	129	128	129
127	127	127	128	129	129	129	129
127	127	127	127	127	128	128	128
126	127	127	127	126	127	128	129
127	127	127	126	126	126	127	127
126	127	127	126	126	126	127	127
126	127	127	126	126	126	126	127
126	126	126	126	126	126	126	126

(b)

126
127
...
129
127
126
...
127
126

(c)

Linha/Coluna	1	...	a	...	n
1	127	...	127	...	127
...	127	...	126	...	126
...	...	...	...	...	...
m	127	...	126	...	127
...	...	...	...	...	...
m+63	126	...	126	...	128
...	...	...	...	...	...
2048	126	...	127	...	127

(d)

(**Fonte**: feita pelo autor)

Como cada imagem pré-processada até agora tem 64x32 e temos 357 imagens por sequência no dataset, logo cada matriz de pré-processamento terá 2048 linhas e 357 colunas no total. Elas também foram construídas em Python e C++ e comparadas com a implementação original em MATLAB.

Tabela 2:MSE comparado a implementação de matriz de pré-processamento de entre cada dataserComp

	Spring	Summer	Fall	Winter
MSE MATLAB X Python	0,486625	0,453872	0,587901	0,421345
MSE MATLAB X C++	0,616579	0,398836	0,532764	0,471288

4.1.3 Comparação entre as imagens nos conjuntos de referência e de teste

Para comparar imagens de teste com uma matriz de referência, seguimos os seguintes passos:

Primeiro, criamos uma matriz de distância. Isso é feito subtraindo a matriz de pré-processamento de referência pela matriz de m copias da coluna pertinente a imagem de teste na matriz de pré-processamento de teste. Essa operação resulta em uma matriz de distância para cada imagem de teste.

Em seguida, precisamos de uma métrica de similaridade para quantificar o grau de semelhança entre as imagens. Duas métricas comuns são a Média da Diferença Absoluta (MAD ou SAD) e a Distância Euclidiana.

O SAD (Média da Diferença Absoluta) mede a distância média entre os valores originais e a média de um conjunto de números. A fórmula para o SAD é:

$$SAD = \frac{1}{n}\sum_{i=1}^{n}\left|x_i - \overline{x}\right|$$

Como as várias matrizes de distâncias anterior já guarda a diferença entre os pré-processamento de treino e de teste, tem-se os valores de $x_i - \overline{x}$. Com isso basta aplicar a fórmula do SAD para cada matriz de distância, e tem-se a construção da matriz de diferença. Com essa abordagem, pode-se avaliar a similaridade entre as imagens de teste e as imagens de referência. Valores próximos de 1 indicam maior similaridade, enquanto valores próximos de 0 indicam menor similaridade.

Tabela 3: MSE comparado a implementação de matriz de diferenças entre MATLAB e Python

MSE MATLAB X Python	Spring	Summer	Fall	Winter
Spring	0,0675	0,0843	0,0865	0,0902
Summer	0,0852	0,0871	0,0312	0,0471
Fall	0,0418	0,0397	0,0657	0,0439
Winter	0,0955	0,0789	0,0453	0,0702

Tabela 4: MSE comparando a implementação de matriz de diferenças entre MATLAB e C++

MSE MATLAB X C++	Spring	Summer	Fall	Winter
Spring	0,0594	0,0609	0,0863	0,0623
Summer	0,0894	0,0836	0,0782	0,0794
Fall	0,0561	0,0589	0,0578	0,0595
Winter	0,0827	0,0667	0,0851	0,0807

(**Fonte**: feita pelo autor)

4.1.4 Aplicação o processo de aprimoramento de contraste local

O processo de aprimoramento de constantes é aplicado pegando cada elemento da matriz de diferenças, subtrai-se a média e divide-se pelo desvio padrão dos elementos ao seu redor, gerando assim a matriz de distâncias aprimorada, este processo é análogo a uma versão 1D de normalização de patch:

Tabela 5: MSE comparado a implementação de matriz de diferenças aprimorada entre MATLAB e Python

MSE MATLAB X C++	Spring	Summer	Fall	Winter
Spring	0,0886	0,0834	0,0768	0,0883
Summer	0,0816	0,0589	0,0625	0,0736
Fall	0,0713	0,0855	0,0682	0,0513
Winter	0,0422	0,0457	0,0811	0,0719

(**Fonte**: feita pelo autor)

Tabela 6: MSE comparado a implementação de matriz de diferenças aprimorada entre MATLAB e C++

MSE MATLAB X Python	Spring	Summer	Fall	Winter
Spring	0,0148	0,0286	0,0138	0,0369
Summer	0,0369	0,0138	0,0286	0,0148
Fall	0,0138	0,0369	0,0148	0,0286
Winter	0,0286	0,0148	0,0369	0,0138

(**Fonte**: feita pelo autor)

4.1.5 Construção das sequências com diferentes velocidades e determinação do melhor match entre a sequência teste e as sequência com diferentes velocidades

O método de reconhecimento de sequência localizada é empregado para identificar sequências de imagens que são consideradas parecidas. Inicialmente é gerado um número de possíveis correspondências de modelos a cada etapa do tempo, o que é feito para reconhecer sequências de lugares familiares.Para isso, uma busca é realizada através do espaço M de vetores recentes de diferença de imagem. O espaço M é uma matriz composta por uma sequência de vetores de diferença, representado pela equação:

$$M = D^{(T-ds)}, D^{(T-ds+1)}, \ldots, D^{(T)}$$

Uma pontuação de diferença S é calculada para cada linha de trajetória com base nos valores de diferença que a linha passa ao viajar do tempo T-ds até o tempo atual T. A equação para isso é:

$$S = \sum_{t=T-ds}^{T} D^{t}$$

Onde ‘D’ representa os vetores de diferença e ‘t’ o tempo.

Além disso, o algoritmo considera o valor particular de diferença que a trajetória passa no tempo t, representado pela equação

$$K = S + V(ds - T + t)$$

Onde ‘k’ é o valor da posição no tempo ‘t’, ‘s’ é uma constante, ‘V’ representa velocidade, ‘ds’ é outra constante relacionada à distância, e ‘T’ é uma constante relacionada ao tempo.

As buscas de trajetória são realizadas a partir de todos os modelos, exceto para os modelos recentemente aprendidos dentro de Rrecent do modelo atual, a fim de evitar a correspondência com a trajetória atual. Depois que todas as pontuações de trajetória foram avaliadas, a trajetória de pontuação mínima (ou seja, a melhor correspondência) para cada modelo é colocada no vetor S. Se a trajetória de pontuação mínima dentro de uma janela deslizante de alcance Rwindow for um fator de μ menor do que qualquer uma das pontuações de trajetória fora da janela, essa trajetória é considerada uma correspondência.

4.2 COMPARAÇÕES do SeqSlam

Com o intuito de realizar uma análise comparativa do algoritmo implementado em diferentes linguagem, constrói-se então as tabelas 7,8 e 9 a partir dos dados obtidos após a execução do algoritmo variando o conjunto de treino e de teste. O objetivo é avaliar o desempenho do algoritmo implementado nas 3 diferentes linguagens (Python, MATLAB - método de referência, e C++) com base em três principais métricas: precisão, revocação e tempo de execução (em segundos).

Tabela 7: Comparando entre as métricas entre MATLAB

Precisão	Spring	Summer	Fall	Winter
Spring	0,941826	0,815857	0,793918	0,941826
Summer	0,771751	0,95212	0,806551	0,771751
Fall	0,751478	0,783018	0,882106	0,751478
Winter	0,786641	0,790615	0,814348	0,786641
Revocação	Spring	Summer	Fall	Winter
Spring	0,817416	0,651743	0,629468	0,679415
Summer	0,629461	0,791925	0,618494	0,638941
Fall	0,659456	0,702509	0,829752	0,667253
Winter	0,687506	0,695167	0,689559	0,778654
Tempo de execução	Spring	Summer	Fall	Winter
Spring	13,54	13,62	13,45	13,81
Summer	14,65	14,75	14,16	14,05

Fall	13,72	12,96	13,34	14,18
Winter	14,56	13,76	13,57	13,29

Tabela 8: Comparando entre as métricas entre Python

Precisão	Spring	Summer	Fall	Winter
Spring	0,954274	0,797291	0,759162	0,774381
Summer	0,769254	0,961463	0,774866	0,793289
Fall	0,796541	0,781452	0,938137	0,747985
Winter	0,789841	0,780934	0,816692	0,965187
Revocação	Spring	Summer	Fall	Winter
Spring	0,806156	0,66413	0,644088	0,584185
Summer	0,615461	0,80039	0,608649	0,647581
Fall	0,655663	0,688768	0,838914	0,675173
Winter	0,67506	0,704923	0,701194	0,794594
Tempo de execução	Spring	Summer	Fall	Winter
Spring	14,78	14,93	14,62	14,67
Summer	15,27	16,13	14,91	14,88
Fall	14,83	13,10	14,11	14,18
Winter	16,11	14,77	14,27	14,11

Tabela 9: Comparando entre as métricas entre C++

Precisão	Spring	Summer	Fall	Winter
Spring	0,941826	0,815857	0,793918	0,695565
Summer	0,771751	0,95212	0,806551	0,941627
Fall	0,751478	0,853018	0,882106	0,773472
Winter	0,786641	0,810615	0,844348	0,971

Revocação	Spring	Summer	Fall	Winter
Spring	0,826996	0,643281	0,614848	0,572875
Summer	0,648361	0,787272	0,628339	0,648061
Fall	0,665987	0,71625	0,82059	0,673673
Winter	0,702966	0,704851	0,708022	0,699454
Tempo de execução	Spring	Summer	Fall	Winter
Spring	13,78	13,88	13,76	14,02
Summer	14,27	15,07	14,49	14,29
Fall	13,83	13,29	13,62	14,43
Winter	14,11	13,94	13,81	13,49

(**Fonte**: feita pelo autor)

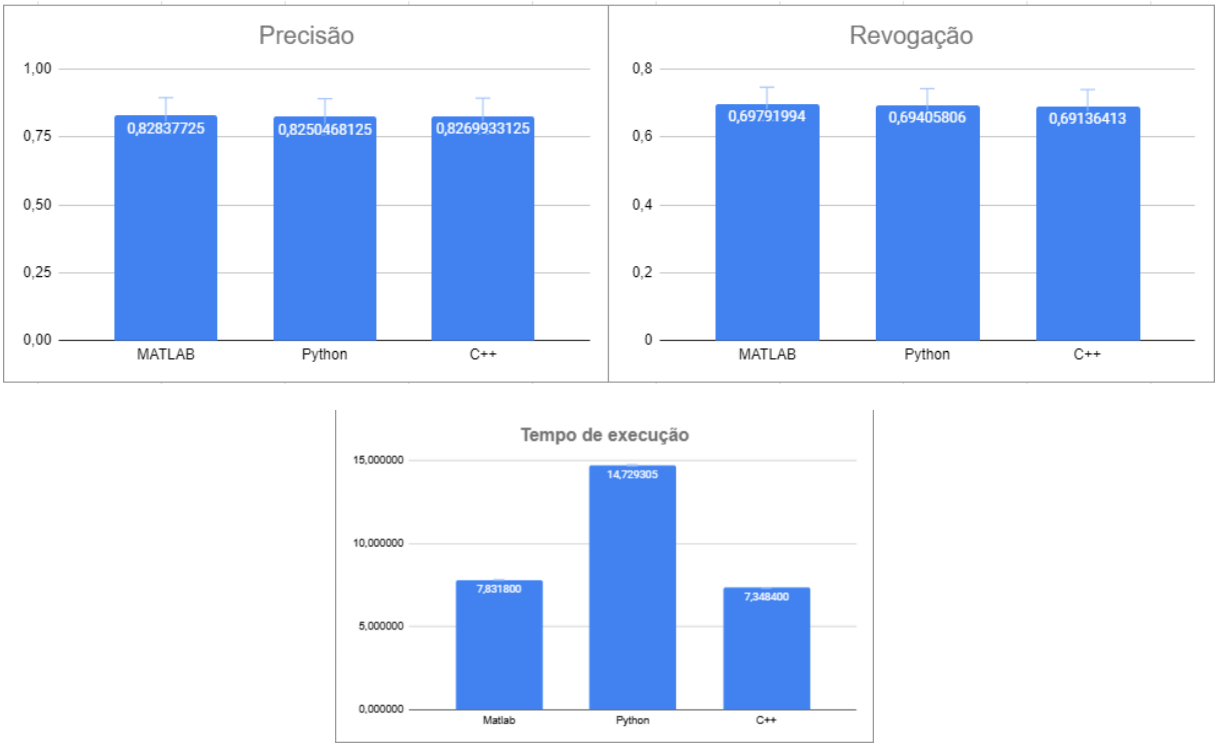
Estes dados foram sintetizados em uma tabela com a média e o desvio padrão de todos os casos em cada linguagem.

Tabela 10: Tabela com a média e o desvio padrão de todos os casos em cada lingagem

Precisão	Média	desvio padrão
MATLAB	0,82837725	0,0791709217
Python	0,82504681	0,0791463783
C++	0,82699331	0,0790621140
Revogação	Média	desvio padrão
MATLAB	0,69791994	0,06893229
Python	0,69405806	0,0767893054
C++	0,69136413	0,0713826949
Tempo de execução	Média	desvio padrão
MATLAB	7,831800	0,2934861217
Python	9,308250	0,3268212136
C++	8,346000	0,2914534354

Uma outra forma de sintetizando em uma imagem:

Figura 17: Precisão, Revocação e Tempo de execução médios em cada linguagem para o algoritmos analisado



(**Fonte**: feita pelo autor)

As três tabelas analisadas permitem identificar características interessantes sobre os dados apresentados. O MATLAB (método de referência) demonstra maior consistência em termos de precisão e revocação ao longo das estações do ano, enquanto Python e C++ apresentam variações mais acentuadas. A primavera se destaca como a estação com os melhores resultados de precisão para todas as ferramentas, enquanto o inverno apresenta os piores desempenhos, especialmente na métrica de revogação.

Quando o foco é o tempo de execução, o C++ e o MATLAB (método de referência) se sobressaem com valores menores e mais constantes, enquanto o Python é ligeiramente mais lento em todas as estações. Assim, o MATLAB (método de referência) se mostra ideal para quem busca equilíbrio entre precisão e revogação, enquanto o C++ pode ser a escolha para priorizar velocidade. O Python, por sua vez, oferece grande desempenho em precisão na primavera, apesar de tempos de execução mais altos.

4.3 MPF

4.3.1 Aquisição de imagens:

A aquisição de imagens envolve a execução de um script que carrega as imagens das pastas especificadas, verificando o tipo de arquivo e a pasta de origem, além de coletar as imagens de acordo com o intervalo e a quantidade definidos. Neste contexto, a base de dados escolhida foi a Oxford Robotcar, amplamente utilizada em aplicações de VPR e SLAM. Essa base de dados é fundamental para a determinação dos melhores parâmetros da rede, a fim de validar o modelo.

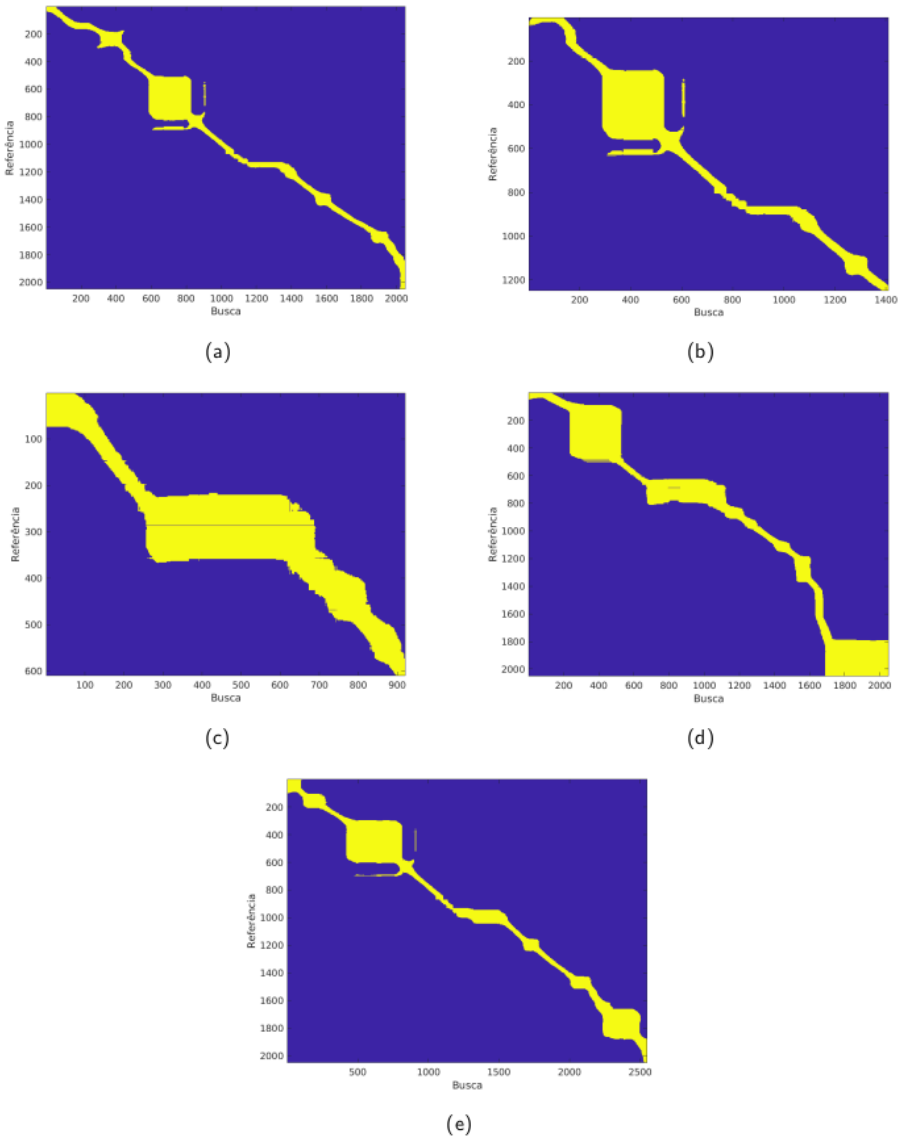
A Oxford Robotcar, conforme descrito por Maddern et al. (2017), possui imagens coletadas de diferentes ângulos por dois tipos de câmeras acopladas a um carro. Nas laterais e na traseira, a câmera Glasshopper foi utilizada, capturando imagens com resolução de 1024x1024. Na frente, a câmera Bumblebee XB3, equipada com três lentes, sendo uma delas posicionada no centro do veículo, foi empregada. Essa base contém uma vasta quantidade de imagens coletadas em uma mesma região, abrangendo diversas condições, como dia, noite, inverno, verão, chuva e ambientes nublados, além de cenários com prédios em diferentes estágios de construção, obras rodoviárias e desvios de rota.

Para os experimentos, foram utilizadas imagens capturadas pela lente esquerda da câmera Bumblebee XB3, assim como descrito no artigo de referência. As imagens correspondem às sequências 2014/12/09 (dia), 2014/12/10 (noite), 2015/02/13 (dia) e 2014/11/14 (noite). A Figura 18 ilustra imagens das sequências nessa base de dados utilizadas para os experimentos. A distância máxima considerada para o ground truth em todos os experimentos realizados com essa base de dados foi de 40 metros, referindo-se à distância entre a posição estimada, com base nos dados armazenados que mais corresponde ao dado atual, e a posição real.

Figura 18: Cada linha representa frames capturados em diferentes momentos das sequências "2014/12/02", "2015/02/13", "2014/12/10" e "2014/11/14", respectivamente, no conjunto de dados Oxford Robotcar.

Com base nestas sequências, foram construídas 5 subsequências, para abranger diferentes condições de horário e clima, para todas as sequências os frames coletados possuem intervalo de 3 em 3 imagens da base de dados, na Figura 19 tem-se o ground truth para cada uma das sequências:

Figura 19: As sequências que serão posteriormente enumeradas de 1 a 5 são dadas respectivamente por: (a) 2014/12/09 com início na imagem 400 com 2050 amostras para referência e 2014/12/10 com início na imagem 1750 com 2050 amostras para busca, (b) 2014/12/09 com início na imagem 1200 com 1250 amostras para referência e 2014/12/10 com início na imagem 2645 com 1410 amostras para busca, (c) 2014/12/09 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) 2014/11/14 para referência com início na imagem 8750 com 2050 amostras e 2014/12/10 com início na imagem 9150 com 2050 amostras para busca, (e) 2014/12/09 para referência com início na imagem 1000 com 2050 amostras e 2015/02/13 com início na imagem 1750 com 2550 amostras para busca.



4.3.2 Pré-processamento das imagens:

Recorte e Redimensionamento das imagens: É realizado o carregamento das imagens do dataset e aplicando técnicas pré-processamento das imagens como o

recorte e redimensionamento das imagens para uma resolução correspondente a entrada de cada um dos descritores. Durante este processo foi constatada uma diferença no redimensionalização entre as implementações em MATLAB e Python. O método utilizado no MATLAB foi o Lanczos3, enquanto, no Python, foi utilizado o método mais próximo disponível, Lanczos. Essa diferença resultou em variações no erro médio quadrado (MSE) entre as duas linguagens, como demonstrado na Tabela 11.

A tabela 11: Mostra o erro médio quadrado (MSE), entre a uma image número 400 da sequência 2014/12/09 do dataset do Oxford Robotcar, redimensionamento e normalização em matlab e em python:

	im1-CNN	im2-CNN-D	im3-HOG	im4-SAD
MSE	0.0000707015	0.0000707015	0.0001918411	0,0000230119

(**Fonte**: feita pelo autor)

Nota-se que, apesar de essa diferença ser bastante pequena, ainda há uma discrepância entre as imagens redimensionadas e normalizadas em MATLAB e Python. Além disso, a diferença é maior para o HOG, intermediária para a CNN e CNN-D, e menor para o SAD. Isso ocorre porque a imagem original foi redimensionada para uma dimensão maior no HOG (640x320), intermediária na CNN e CNN-D (227x227) e menor no SAD (64x32). Essa diferença acompanhou proporcionalmente o tamanho do redimensionamento da imagem original.

4.3.3 Extração de Características

Características das imagens são extraídas utilizando métodos como CNN, CNN-D, HOG e SAD, e os resultados são armazenados em arrays para facilitar a comparação com as imagens de referência.

Na tabelas 12 e 13, temos o valor dos ângulos $\Theta_{médio}$ e Θ_{max} para a diferença entre os ângulos das imagens vetorizadas em matlab e em python para cada método, de referência e de match, respectivamente.

Tabela 12: o valor dos ângulos $\Theta_{médio}$ e Θ_{max} para a diferença entre os ângulos das imagens vetorizadas em matlab e em python para cada método, para o conjunto de referência das sequências 1-5

	CNN		CNN-D		HOG		SAD	
	$\Theta_{médio}$	Θ_{max}	$\Theta_{médio}$	Θ_{max}	$\Theta_{médio}$	Θ_{max}	$\Theta_{médio}$	Θ_{max}
Sequência 1	6,13 E-05°	3,77 E-04°	2,27 E-05°	1,91 E-05°	4,98 E-05°	9,41 E-04°	2,27 E-03°	1,42 E-03°
Sequência 2	1,07 E-05°	6,60 E-04°	3,97 E-05°	3,34 E-05°	8,71 E-05°	1,64 E-03°	3,97 E-03°	2,49 E-03°
Sequência 3	5,20 E-05°	4,21 E-04°	3,91 E-05°	2,97 E-05°	7,22 E-05°	1,18 E-03°	2,89 E-03°	1,76 E-03°
Sequência 4	2,30 E-05°	1,45 E-04°	1,02 E-05°	8,89 E-06°	2,10 E-05°	3,81 E-04°	1,02 E-04°	6,43 E-03°
Sequência 5	4,96 E-05°	3,52 E-04°	2,47 E-05°	1,91 E-05°	4,98 E-05°	9,41 E-04°	2,27 E-03°	1,42 E-03°

Tabela 13: O valor dos ângulos $\Theta_{médio}$ e Θ_{max} para a diferença entre os ângulos das imagens vetorizadas em matlab e em python para cada método, para o conjunto de match das sequências 1-5

	CNN		CNN-D		HOG		SAD	
	$\Theta_{médio}$	Θ_{max}	$\Theta_{médio}$	Θ_{max}	$\Theta_{médio}$	Θ_{max}	$\Theta_{médio}$	Θ_{max}
Sequência 1	1,53 E-05°	6,23 E-04°	4,00 E-05°	3,34 E-05°	8,71 E-05°	1,49 E-03°	3,47 E-03°	2,25 E-03°
Sequência 2	7,09 E-05°	3,30 E-04°	2,23 E-05°	1,72 E-05°	4,48 E-05°	8,46 E-04°	2,09 E-03°	1,30 E-03°
Sequência 3	8,81 E-05°	2,12 E-04°	1,25 E-05°	1,05 E-05°	2,86 E-05°	5,18 E-04°	1,25 E-03°	7,89 E-04°
Sequência 4	5,00 E-05°	4,21 E-04°	3,97 E-05°	3,25 E-05°	7,20 E-05°	1,18 E-03°	2,89 E-03°	1,76 E-03°
Sequência 5	2,47 E-05°	3,68 E-04°	2,87 E-05°	2,20 E-05°	5,74 E-05°	1,08 E-03°	2,67 E-03°	1,67 E-03°

Verifica que os valores do SAD apresentam maior divergência entre os ângulos dos métodos das imagens vetorizadas em matlab e em python, ficando em torno dos E-03° enquanto os outros métodos ficam em torno dos E-05°, o que acontece devido a especificações da quantidade de casas decimais de como realiza as operações em cada linguagem.

4.3.4 Cálculo das Diferenças entre Imagens

Para cada imagens de query para todas as em relação ao dataset de referência. Para a CNN e HOG utiliza-se a distância dos cossenos, enquanto para a CNN-D e SAD utiliza-se distâncias euclidianas.

Na tabela 14 temos a distância dos cossenos média em python subtraída da distância dos cossenos média em matlab CNN e HOG e na tabela 15 temos o MSE entre os vetore df para a CNN-D e SAD para as implementações em python e em matlab

Tabela 14: Distância dos cossenos média em python subtraida da distância dos cossenos média em matlab CNN e HOG

Distância dos cossenos	$df1_{python} - df1_{matlab}$	$df3_{python} - df3_{matlab}$
Sequência 1	0,01248	0,00584
Sequência 2	0,01375	0,00867
Sequência 3	0,00987	0,00342
Sequência 4	0,00812	0,01401
Sequência 5	0,01453	0,01029

(**Fonte**: feita pelo autor)

Tabela 15: o MSE para a CNN-D e SAD.

MSE	$MSE(df2_{python}, df2_{matlab})$	$MSE(df4_{python}, df4_{matlab})$
Sequência 1	0,00672	0,00915
Sequência 2	0,00289	0,00468
Sequência 3	0,01198	0,00785
Sequência 4	0,01356	0,01503
Sequência 5	0,00423	0,01134

4.3.5 Normalização das diferenças

Para cada vetor de diferenças, calcula-se o valor máximo mx correspondente. Em seguida, determina-se o denominador de normalização df, que é obtido subtraindo-se o valor mínimo do vetor de diferenças do valor máximo.

$$df = mx - min(diffVector)$$

Depois, para cada elemento *k* dentro do vetor, realiza-se a normalização utilizando a fórmula:

$$O\_diff = \frac{(mx - diffVector(k))}{df - epsilon}$$

O parâmetro *epsilon* é subtraído para ajustar o valor ao intervalo desejado (0.001 a 0.999). Esse processo é repetido para cada vetor de diferenças*(diffVector1, diffVector2, diffVector3, diffVector4)*, garantindo que todos os elementos sejam normalizados de forma consistente.

Por fim verificamos para cada valor *o_diff* se eles são menores que o valor do limiar de observação *obsThresh,* se for substituímos o valor por epsilon, se não, mantemos o valor de *o_diff* na matriz de observação *(O1,O2, O3, O4).*

Na tabela 16, a seguir, temos o MSE entre as matrizes de observação da sequência de 1 a 5.

Tabela 16: MSE entre as matrizes de observação da sequência de 1 a 5.

	O1	O2	O3	O4
Sequência 1	0,00833	0,00717	0,00247	0,01014
Sequência 2	0,00950	0,01076	0,00371	0,01521
Sequência 3	0,00417	0,00359	0,00124	0,00507
Sequência 4	0,00625	0,00538	0,00185	0,00760
Sequência 5	0,01042	0,00896	0,00278	0,01268

(**Fonte**: feita pelo autor)

O1 e O4 têm os valores mais extremos dentro das sequências, enquanto O2 e O3 mantêm uma faixa bem mais restrita.

4.3.6 Encontra-se o pior caso

Com base nas matrizes de observação (*O1*, *O2*, *O3*, *O4*), o processo de identificar o "pior caso" consiste em analisar os valores normalizados dessas matrizes para determinar qual apresenta a maior discrepância ou desempenho mais crítico. Essa análise é realizada em uma janela específica de observação (*Rwindow*), conforme configurado pelos parâmetros.

Inicialmente, ela identifica os índices dos valores máximos em cada matriz de observação, que representam os pontos de maior relevância em cada sensor. Em seguida, calcula-se as distâncias entre esses índices, definidas como 'distâncias de coerência espacial'. Essas distâncias são comparadas a um limite definido pela janela de observação *(Rwindow).* O identificador resultante é armazenado em um array específico (*worstIDArray*) para registro, enquanto um contador (*worstIDCounter*) é incrementado para rastrear a frequência com que esse "pior caso" ocorre em relação às observações.

Esse processo permite monitorar os dados capturados pelas matrizes de observação e identificar padrões ou anomalias, sendo útil para ajustar e melhorar o desempenho do sistema analisado.

Abaixo, na tabela 17 temos o vetor do pior ID Array, para as cinco sequências analisadas primeira sequência analisada. ele está na seguinte ordem [nenhum dos métodos, CNN, CNN-D, HOG, SAD], O valor 'nenhum dos métodos' é utilizado para representar casos em que todos os métodos apresentam valores superiores à janela específica de observação *(Rwindow)* ou em que todos os métodos possuem valores inferiores a essa mesma janela. Essa abordagem é fundamental para identificar cenários em que nenhuma correspondência relevante é encontrada dentro do intervalo definido.

Tabela 17: Vetor worstIDCounter para cada sequência

worstIDCounter	Matlab	Python
Sequência 1	[711, 18, 93, 231, 997]	[718, 18, 92, 231, 991]
Sequência 2	[420, 16, 99, 172, 703]	[423, 17, 99, 170, 701]
Sequência 3	[367, 13, 68, 11, 461]	[369, 13, 68, 11, 459]
Sequência 4	[873, 144, 352, 140, 541]	[881, 144, 350, 140, 535]
Sequência 5	[1856, 7, 288, 49, 350]	[1873, 7, 284, 49, 337]

(**Fonte**: feita pelo autor)

Pode-se perceber que o vetor em Python apresenta valores ligeiramente diferentes do vetor em Matlab para quase todas as posições, exceto para o terceiro da CNN-D e quarto posição do HOG, que são quase idênticos para a maioria dos casos analisados. A quantidade de 'piores casos' aumentou para na primeira posição do vetor, representada por 'nenhum dos métodos', o que indica que a discrepância entre os métodos diminuiu, enquanto as posições relacionadas à CNN e ao SAD apresentaram ligeiras quedas.

4.3.7 Executa o algoritmo de Viterbi

O algoritmo de Viterbi é usado para analisar sequências de observações de forma dinâmica, extraindo a sequência mais provável de estados ocultos e avaliando a confiabilidade das correspondências entre os estados detectados e as observações fornecidas. A execução ocorre apenas quando o índice atual ultrapassa o comprimento máximo definido, evitando cálculos desnecessários. Nesse caso, é criada uma janela com as últimas observações dentro desse limite.

Com base nas matrizes de observação (*O1*, *O2*, *O3* e *O4*), na sequência atual analisada *(s)*, na matriz de transição *(t)* em algum parâmetros configurados, chama-se o algoritmo de viterbi que irá determinar a sequência mais provável de estados *(seq)*, avalia o quão confiável é essa sequência (quality) e ajusta a métrica ao comprimento da sequência processada *(newSeqLength)*. Após isso, o algoritmo identifica o estado final mais relevante para a análise.

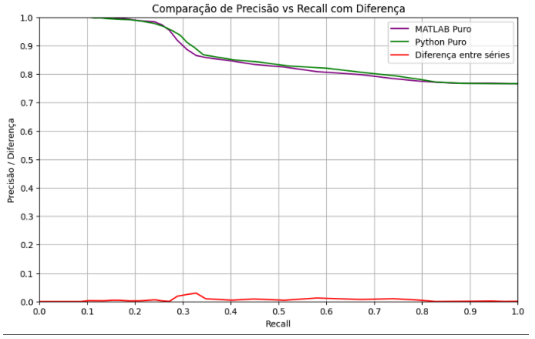
Esse método é projetado para lidar com mudanças nas características das observações, facilitando a identificação de padrões e estados de forma eficiente e alinhada aos limites predefinidos, o que melhora a confiabilidade dos resultados obtidos.

4.3.8 Identifica o estado mais provável

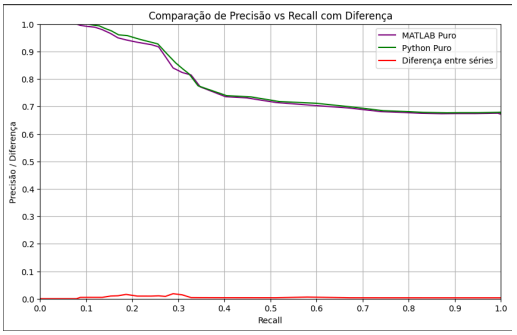
A qualidade *(quality)* é normalizada com base no comprimento da sequência analisada, garantindo uma métrica proporcional e precisa. Além disso, utilizando a sequência mais provável é identificado que seu último estado é armazenado em *id*, representando o estado final mais relevante no contexto analisado. Este procedimento é utilizado para assegurar a avaliação precisa e consistente da confiabilidade da sequência e, ao mesmo tempo, fornece um ponto de referência crítico para aplicações práticas, como localização ou análise de padrões.

4.4 COMPARAÇÕES DO MPF

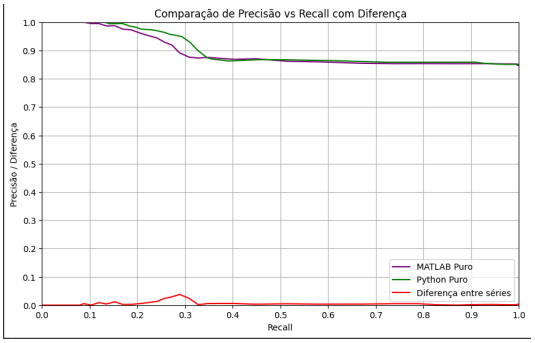
Figura 20: As subfiguras (a),(b),(c),(d) e (e) são referentes a gráficos de precisão-revocação das sequências de 1 a 5 respectivamente.



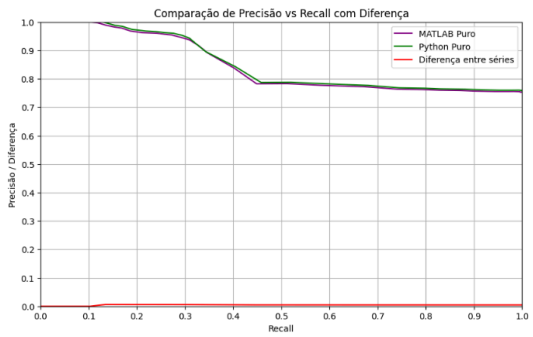
(a)



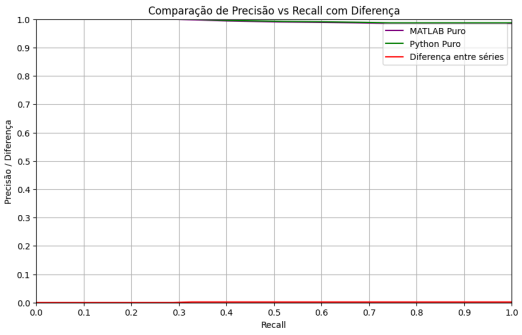
(b)



(c)



(d)



(e)

Tabela 18: Comparação de precisão para trechos das sequência no Oxford Robotcar em Matlab e em Python

	Matlab	Python
Sequência 1	0.765517241379310	0.7660098522167488
Sequência 2	0.671223021582734	0.678273381294964
Sequência 3	0.847777777777778	0.8511111111111112
Sequência 4	0.753201970443350	0.7577211822668424
Sequência 5	0.985375494071146	0.9847885091629525

(**Fonte**: feita pelo autor)

Com base na figura 20 e na tabela 18, que apresentam a comparação entre o método do MPF implementado em Python e o método de referência em MATLAB, foi possível observar que o modelo do MPF desenvolvido em Python apresentou uma leve melhora na curva de precisão x revocação em relação ao modelo de referência construído no MATLAB. Essa melhora, embora inferior a 1%, foi identificada nas 5 sequências analisadas. Uma possível explicação para esse resultado está na forma como cada linguagem trata a precisão das casas decimais durante as operações.

Segundo Silva, A. F. L. A. (2023), o método MPF mostrou desempenho superior ao SeqSLAM em termos de precisão e revocação nos testes realizados, garantindo maior eficiência na identificação de lugares visuais em diferentes cenários. No entanto, essa vantagem vem acompanhada de um custo computacional significativamente maior, tornando o MPF menos adequado para aplicações em tempo real ou embarcadas. Por outro lado, o SeqSLAM, embora tenha apresentado resultados inferiores em precisão e revocação, demonstra maior viabilidade para cenários com restrições computacionais, devido ao seu menor tempo de execução. Assim, a escolha entre os dois modelos depende diretamente das demandas específicas de cada aplicação, equilibrando precisão e eficiência computacional.

5 CONCLUSÃO

Este trabalho teve como objetivo implementar e comparar dois métodos de reconhecimento visual de locais, o SeqSLAM e o MPF, em diferentes linguagens de programação: MATLAB, Python e C++. Para isso, foram realizadas análises detalhadas do desempenho dos modelos em condições adversas, utilizando datasets consagrados na área, como o Nordland e o Oxford Robotcar, que apresentam condições adversas, incluindo mudanças climáticas e de iluminação

Entre os principais resultados, destaca-se a validação das implementações, que permitiu replicar os algoritmos descritos na literatura e ajustá-los às especificidades de cada linguagem de programação, assegurando fidelidade aos métodos de referência. Também foi conduzida uma análise comparativa que revelou diferenças importantes, como variações nas curvas de precisão e revocação, associadas às particularidades no tratamento de operações numéricas e ao desempenho de execução.

Os resultados obtidos fornecem uma base robusta para futuras pesquisas na área de reconhecimento visual de locais, ampliando as possibilidades de aplicação desses algoritmos em sistemas robóticos e destacando sua relevância em cenários adversos. Além disso, uma perspectiva futura inclui testar outros métodos e explorar mais datasets, além de embarcar os modelos implementados em uma plataforma robótica para testes práticos. A implementação dos métodos estudados em diferentes linguagens possibilita essa integração, permitindo expandir os horizontes do trabalho realizado.

REFERÊNCIAS

Barros, T., Pereira, R., Garrote, L., Premebida, C., & Nunes, U. J. (2021). Place recognition survey: An update on deep learning approaches. *arXiv preprint arXiv:2106.10458*.

Campos, F. M. M. O. (2015). Contribuições para a Localização e Mapeamento em Robótica através da Identificação Visual de Lugares. *Tese (Doutorado)* – Universidade de Lisboa, Faculdade de Ciências, Departamento de Informática. Disponível em: Projeto Acadêmico. Acesso em: 23 abr. 2024.

Hausler, S., Jacobson, A., & Milford, M. (2019). Multi-Process Fusion: Visual Place Recognition Using Multiple Image Processing Methods. *IEEE Robotics and Automation Letters, 4*(2), 1924–1931. doi:10.1109/LRA.2019.2898427.

Lowry, S., Sünderhauf, N., Newman, P., Leonard, J. J., Cox, D., Corke, P., & Milford, M. J. (2015). Visual place recognition: A survey. *IEEE Transactions on Robotics, 32*(1), 1-19.

Milford, M. J., & Wyeth, G. F. (2012). SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights. *2012 IEEE International Conference on Robotics and Automation*, Saint Paul, MN, USA, 1643–1649. doi:10.1109/ICRA.2012.6224623.

NEULAND, Renata das Chagas. Análise de Intervalos e Restrições Temporais Aplicadas ao Problema de Reconhecimento de Regiões. *Tese (Doutorado em Computação)* - Programa de Pós-Graduação em Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2019.

Siam, S. M., & Zhang, H. (2017). Fast-SeqSLAM: A fast appearance-based place recognition algorithm. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, 5702–5708. doi:10.1109/ICRA.2017.7989671.

SILVA, A. F. L. A. da; SILVA JÚNIOR, M. R. da; ARAÚJO, A. F. R.; DURAND-PETITEVILLE, A. Visual Place Recognition under a High-Dimensional Subspace Clustering Perspective. 2023 Latin American Robotics Symposium (LARS), Brazilian Symposium on Robotics (SBR), and Workshop on Robotics in Education (WRE). DOI: 10.1109/LARS/SBR/WRE59448.2023.10332995.

SÜNDERHAUF, N.; NEUBERT, P.; PROTZEL, P. Are we there yet? challenging seqslam on a 3000 km journey across all four seasons. In: Proceedings of Workshop on Long-Term Autonomy, International Conference on Robotics and Automation (ICRA). IEEE, 2013. p. 2013. Dataset download: http://nrkbeta.no/2013/01/15/.

Sünderhauf, N., Shirazi, S., Jacobson, A., Dayoub, F., Pepperell, E., Upcroft, B., & Milford, M. (2024). Place Recognition with ConvNet Landmarks: Viewpoint-Robust, Condition-Robust, Training-Free. *Proceedings of Robotics: Science and Systems (RSS), 11*, 22. 1.

Yin, P., Jiao, J., Zhao, S., Xu, L., Huang, G., Choset, H., Scherer, S., & Han, J. (2024). General Place Recognition Survey: Towards Real-World Autonomy. *arXiv preprint arXiv:2405.04812*.