UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

# Desenvolvimento de um sistema de detecção e rastreamento de veículos para análise de anomalias de tráfego em rodovias utilizando estruturas espaciais e temporais por meio de Visão Computacional.

**Ana Rosalia Huaman Reyna**

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

**Ana Rosalia Huaman Reyna**

# Desenvolvimento de um sistema de detecção e rastreamento de veículos para análise de anomalias de tráfego em rodovias utilizando estruturas espaciais e temporais por meio de Visão Computacional.

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestra em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Rodolfo Ipolito Meneguette

**USP – São Carlos**
**Fevereiro de 2024**

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

H459d

Huaman Reyna, Ana Rosalia
Desenvolvimento de um sistema de detecção e rastreamento deveículos para análise de anomalias de tráfego em rodovias utilizando estruturas espaciais e temporais por meio de Visão Computacional. / Ana Rosalia Huaman Reyna; orientador Rodolfo Ipolito Meneguette. -- São Carlos, 2024.
96 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2024.

1. . I. Ipolito Meneguette, Rodolfo, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

**Ana Rosalia Huaman Reyna**

# Development of a vehicle detection and tracking system for analyzing traffic anomalies on highways using spatial and temporal structures through Computer Vision.

Master dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Rodolfo Ipolito Meneguette

**USP – São Carlos**
**February 2024**

*Este trabalho é dedicado aos meus pais ♡, Jonatan e Ana, por me ensinarem a ser uma pessoa de princípios, motivando-me a explorar o mundo e encorajando-me a perseguir meus objetivos. Eles têm sido uma fonte inesgotável de inspiração e apoio em minha jornada, moldando quem sou e me incentivando a sempre buscar o melhor em tudo que faço. Suas lições e amor incondicional são o alicerce sobre o qual construí minha vida, e sou profundamente grata por tudo o que tem feito por mim.*

# AGRADECIMENTOS

Agradeço, em primeiro lugar, a Deus por todas as bênçãos e oportunidades que tenho recebido. Sou profundamente grata à minha família, que tem sido um pilar inabalável de apoio ao longo dos meus estudos. Em especial, quero expressar minha gratidão aos meus pais, Jonatan e Ana, e aos meus irmãos, Heiner e Maira. Eles sempre acreditaram que a educação era o melhor caminho para o crescimento e desenvolvimento pessoal.

Também quero estender meus agradecimentos a todas as pessoas que me incentivaram, ofereceram ideias valiosas e palavras de motivação ao longo dessa jornada. Amigos e conhecidos, como o Alex, por contribuir com ideias para o meu projeto, à Rosalia, que me incentivou a seguir o caminho do mestrado.

Não posso deixar de reconhecer a importância da CAPES e do ICMC, que proporcionaram a oportunidade de uma bolsa integral e apoio financeiro para todas as minhas atividades. Quero agradecer ao Restaurante Universitário, por fornecer as refeições que foram essenciais para minha sustentação ao longo dessa jornada.

Além disso, minha jornada de mestrado foi enriquecida graças aos pesquisadores, professores e amigos do grupo *Urban Smart Solutions Lab (USSL)*. Eles me orientaram com dicas de apresentação, estruturação de artigos e conhecimentos relevantes, que foram fundamentais para o sucesso do meu projeto. Quero agradecer em particular ao professor Rodolfo I. Meneguette, que desde o início acreditou em mim para a realização deste projeto.

Por último, mas não menos importante, quero expressar minha eterna gratidão a Adhara, Joaquin, Joaquina e Federico pelo amor puro e incondicional que eles me proporcionaram. Seus gestos afetuosos e latidos trouxeram imensa alegria a cada dia da minha jornada, mesmo estando longe.

*"As maiores descobertas muitas vezes*
*não residem em encontrar coisas novas,*
*mas em ver coisas familiares*
*de novas maneiras."*
*(Alexander Fleming)*

# RESUMO

REYNA, A. H. **Desenvolvimento de um sistema de detecção e rastreamento de veículos para análise de anomalias de tráfego em rodovias utilizando estruturas espaciais e temporais por meio de Visão Computacional.**. 2024. 96 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2024.

Atualmente, existem sistemas de visão computacional que nos auxiliam em tarefas que seriam maçantes para o ser humano, como vigilância e rastreamento de veículos. Uma parte essencial desta análise é identificar anomalias de tráfego. Uma anomalia nos diz que algo incomum aconteceu, neste caso, na rodovia. Este projeto tem como objetivo modelar a detecção e o rastreamento de veículos usando visão computacional para detectar anomalias de tráfego nas estradas. Para o desenvolvimento deste trabalho, seguimos as etapas de detecção, rastreamento e análise de tráfego: a detecção de veículos a partir de vídeos de tráfego urbano, o rastreamento de veículos utilizando um gráfico bipartido e o algoritmo *Convex Hull* para delimitar áreas móveis. Finalmente, para detecção de anomalias, utilizamos duas estruturas de dados para detectar o início e o fim da anomalia. A primeira é o *QuadTree*, que agrupa veículos que ficam muito tempo parados na estrada. A segunda abordagem trata de veículos que estão obstruídos. Os resultados experimentais mostram que nosso método é aceitável no conjunto de testes *Track 4*, com uma pontuação $F1$ de 85,7% e um erro quadrático médio de 25,432 segundos.

**Palavras-chave:** Detecção de anomalias, Rastreamento veicular, Visão Computacional.

# ABSTRACT

REYNA, A. H. **Development of a vehicle detection and tracking system for analyzing traffic anomalies on highways using spatial and temporal structures through Computer Vision.**. 2024. 96 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2024.

Currently, there are computer vision systems that help us with tasks that would be dull for humans, such as surveillance and vehicle tracking. An essential part of this analysis is to identify traffic anomalies. An anomaly tells us that something unusual has happened, in this case, on the highway. This project aims to model vehicle detection and tracking using computer vision to detect traffic anomalies on the road. For the development of this work, we follow the steps of detection, tracking and analysis of traffic: the detection of vehicles from video of urban traffic, the tracking of vehicles using a bipartite graph and the Convex Hull algorithm to delimit moving areas. Finally, for anomaly detection, we use two data structures to detect the beginning and end of the anomaly. The first is the QuadTree, which groups vehicles that are stopped for a long time on the road. The second approach handles vehicles that are occluded. Experimental results show that our method is acceptable on the Track4 test set, with an $F1$ score of 85.7% and a mean squared error of 25.432 seconds.

**Keywords:** Anomaly Detection, Vehicle Tracking, Computer Vision..

# LISTA DE ILUSTRAÇÕES


Figura 1 – Distribuição de mortes por tipo de usuário das estradas por região da *WHO* no 2018 . . . . . 23
Figura 2 – Exemplo de aplicações da visão computacional. . . . . 28
Figura 3 – Etapas de um sistema de Visão Computacional. . . . . 28
Figura 4 – Visão computacional parte de outras disciplinas . . . . . 29
Figura 5 – *Bounding boxes* em torno dos objetos detectados . . . . . 31
Figura 6 – Processo geral de detecção de objetos da arquitetura YOLO . . . . . 33
Figura 7 – Rastreamento de objetos . . . . . 33
Figura 8 – Pipeline do rastreamento de objetos . . . . . 34
Figura 9 – Exemplos de aplicação dos filtros. Imagem de padrão de granulados(esquerda), filtro Gaussiano (centro) e resultado da Diferença Gaussiana(direita). . . . . 35
Figura 10 – Pontos-Chaves localizados em duas imagens. . . . . 36
Figura 11 – Atribuição de orientação e magnitude a cada ponto-chave. . . . . 36
Figura 12 – Construção do descritor para um ponto-chave de 2x2 com 48 elementos. . . 37
Figura 13 – Sistema de Medição de Similaridade Estrutural. . . . . 37
Figura 14 – O *CLIP* pré-treina um codificador de imagem e um codificador de texto. . . 39
Figura 15 – Exemplo de grafo bipartido e processo de poda aplicando um limiar. . . . . 40
Figura 16 – Representação de um objeto usando *QuadTree*. . . . . 41
Figura 17 – Estrutura da *QuadTree* para a Figura 16. . . . . 42
Figura 18 – Exemplo de *Convex Hull* . . . . . 43
Figura 19 – Processo de poda em nós vizinhos . . . . . 46
Figura 20 – Arquitetura da estrutura de detecção de anomalias . . . . . 47
Figura 21 – Esquema do artigo. . . . . 48
Figura 22 – Ilustração da estrutura de estratégia hierárquica. . . . . 49
Figura 23 – Fluxograma resumo deste trabalho. . . . . 53
Figura 24 – Cenário de aplicação do MEDAVET. . . . . 54
Figura 25 – Visão Geral do MEDAVET. . . . . 55
Figura 26 – Imagens do *dataset UA-DETRAC*. . . . . 56
Figura 27 – Áreas de interesse nas anotações do *dataset* . . . . . 56
Figura 28 – Imagens do *dataset Track 4*. . . . . 57
Figura 29 – Ruídos e cenas noturnas. . . . . 58
Figura 30 – Imagens sem pré-processamento. . . . . 59
Figura 31 – Diagrama de bloco da estabilização da câmera. . . . . 60

Figura 32 – Detecção de objetos usando *YoloV7* com o *dataset UA-DETRAC*. . . . . 61
Figura 33 – Detecção de veículos usando *YoloV7* com o *dataset UA-DETRAC* com as classes selecionadas. . . . . 62
Figura 34 – Fluxograma do Rastreamento de objetos . . . . . 62
Figura 35 – *Matching* em *frames* consecutivos. . . . . 65
Figura 36 – Veículos dentro da região definida via *Convex Hull*. . . . . 66
Figura 37 – Componente de Detecção de Anomalias. . . . . 68
Figura 38 – Processo de construção da *QuadTree* . . . . . 69
Figura 39 – Ilustração da relação temporal e espacial. . . . . 72
Figura 40 – Polígonos tendo *frames* como pontos de referência . . . . . 78
Figura 41 – Resultados dos extratores de características. . . . . 79
Figura 42 – Resultados gerais com outros métodos. . . . . 83
Figura 43 – Anomalias não detectadas. . . . . 85
Figura 44 – Anomalias detectadas . . . . . 85
Figura 45 – Resultados globais de TP, FN e FP. . . . . 86

# LISTA DE TABELAS

Tabela 1 – Comparativo entre os trabalhos correlatos com o presente trabalho. . . . . 50
Tabela 2 – Comparativo dos pesos usando diferentes parâmetros. . . . . 75
Tabela 3 – Avaliação dos parâmetros. . . . . 76
Tabela 4 – Resultados obtidos sem o uso do *Convex Hull* . . . . . 77
Tabela 5 – Resultados obtidos com o uso do *Convex Hull* . . . . . 78
Tabela 6 – Nosso resultado no conjunto de testes *Track 4* . . . . . 86

# LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| AM | Aprendizado de Máquina |
| AP | Aprendizado Profundo |
| CLEAR MOT | *Clear Metrics for Object Tracking* |
| CLIP | *Contrastive Language-Image Pre-training* |
| CNN | *Convolutional Neural Networks* |
| DeepSORT | *Deep Learning-based SORT* |
| Faster R CNN | *Fast Region-based Convolutional Neural Network* |
| FF | *Feedforward Neural Networks* |
| FN | Falsos Negativos |
| FP | Falsos Positivos |
| GFTT | *Good Features to Track* |
| GPT | *Generative Pre-trained Transformer* |
| GT | *Ground Truth* |
| HOG | *Histogramas of Oriented Gradient* |
| Hz | *Number of reasoning frames per second* |
| IA | *Inteligência Artificial* |
| IDS | *ID Switches* |
| IOU | *Intersection Over Union* |
| Mask R-CNN | *Mask Region-based Convolutional Neural Network* |
| ML | *Mostly Lost* |
| MOG2 | *Mixture of Gaussians 2* |
| MOTA | *Multi-Object Tracking Accuracy* |
| MOTP | *Multi-Object Tracking Precision* |
| MT | *Mostly Tracked* |
| NLP | *Natural Language Processing* |
| NMS | *Non maximum suppression* |
| NRMSE | *Normalized Root Mean Suared Error* |
| PCA | *Principal Component Analysis* |
| RBD | Rastreamento Baseado em Detecção |
| RNN | *Recurrent Neural Networks* |
| RSD | Rastreamento Sem Detecção |

SIFT *Scale Invert Feature Transform*

SORT *Simple online and realtime tracking*

SSMI *Structural Similarity Index*

SVM *Support Vector Machine*

UA-DETRAC *University at Albany DETection and tRACking*

WHO *World Health Organization*

YOLO *You Only Look Once*

# SUMÁRIO


1 INTRODUÇÃO . . . . . 23
1.1 Justificação e motivação . . . . . 25
1.2 Objetivos . . . . . 25
1.3 Estrutura do Documento . . . . . 26

2 FUNDAMENTAÇÃO TEÓRICA . . . . . 27
2.1 Visão Computacional . . . . . 27
*2.1.1 Detecção de objetos* . . . . . *30*
*2.1.2 You Only Look Once (YOLO)* . . . . . *32*
*2.1.3 Rastreamento de Objetos* . . . . . *32*
*2.1.3.1 SIFT* . . . . . *34*
*2.1.3.2 Similaridade Estrutural (SSMI)* . . . . . *35*
*2.1.3.3 Representação do vetor denso* . . . . . *38*
2.2 Estruturas de Dados e Algoritmos . . . . . 39
*2.2.1 Grafo Bipartido* . . . . . *39*
*2.2.2 QuadTree* . . . . . *41*
*2.2.3 Matching* . . . . . *42*
*2.2.4 Convex Hull* . . . . . *42*

3 TRABALHOS CORRELATOS . . . . . 45

4 METODOLOGIA . . . . . 53
4.1 *Datasets* . . . . . 55
*4.1.1 UA-DETRAC* . . . . . *55*
*4.1.2 Track 4* . . . . . *57*
4.2 Pré-Processamento . . . . . 59
4.3 Detecção de objetos . . . . . 60
4.4 Rastreamento de objetos . . . . . 62
*4.4.1 Grafos Bipartidos* . . . . . *63*
*4.4.2 Região de Interesse* . . . . . *65*
*4.4.3 Funcionamento do rastreamento* . . . . . *66*
4.5 Detecção de anomalias . . . . . 67
*4.5.1 Relação Espacial* . . . . . *68*
*4.5.2 Relação Temporal* . . . . . *69*

*4.5.3 Funcionamento da Detecção de Anomalias . . . 70*

**5 EXPERIMENTOS E RESULTADOS . . . 73**
**5.1 Detalhes de Implementação . . . 73**
**5.2 Detecção de objetos . . . 73**
**5.3 Rastreamento de objetos . . . 74**
***5.3.1 Métricas de avaliação . . . 74***
***5.3.2 Avaliação dos parâmetros . . . 75***
***5.3.3 Avaliação do Rastreamento em Regiões de Interesse . . . 76***
*5.3.3.1 Rastreamento sem usar o Convex Hull . . . 77*
*5.3.3.2 Rastreamento usando o Convex Hull . . . 77*
***5.3.4 Resultados Gerais do MEDAVET no Rastreamento. . . . 79***
*5.3.4.1 Resultados Gerais dos Extratores de Características . . . 79*
*5.3.4.2 Resultados Gerais com Outros Métodos . . . 80*
**5.4 Detecção de anomalia . . . 82**
***5.4.1 Métricas de avaliação . . . 82***
***5.4.2 Resultados gerais do MEDAVET na detecção de anomalias . . . 84***

**6 CONCLUSÕES . . . 87**
**6.1 Contribuições Intelectuais . . . 88**

**REFERÊNCIAS . . . 89**

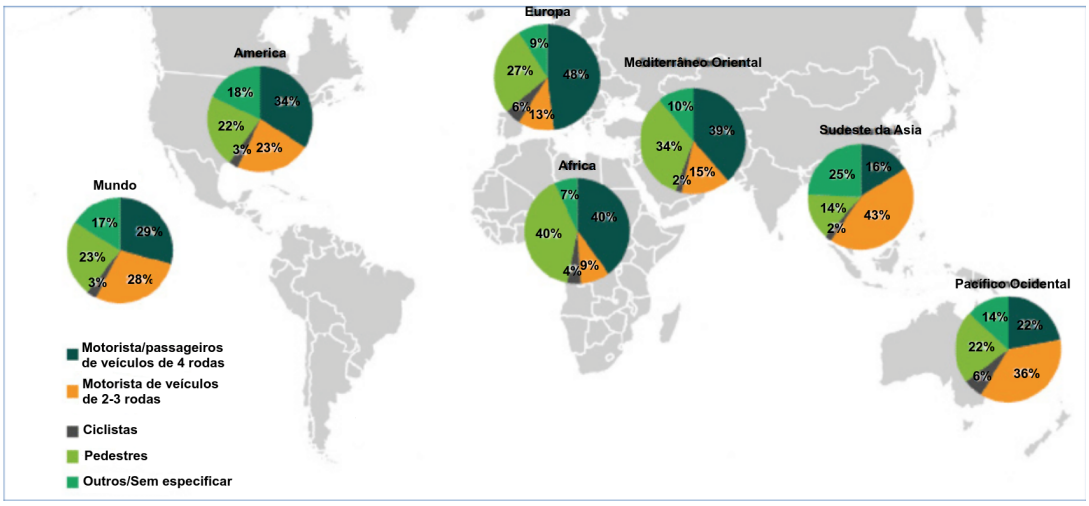
CAPÍTULO

1

# INTRODUÇÃO

A segurança no trânsito é um tema de fundamental importância em todas as partes do mundo. De acordo com dados divulgados no *Global Status Report on Road Safety 2018* da Organização Mundial da Saúde (HEALTH, 2018), a cada ano, cerca de 1.35 milhão de vidas são perdidas devido a acidentes de trânsito no mundo todo. Além dessas tragédias fatais, entre 20 e 50 milhões de pessoas sofrem com sequelas permanentes decorrentes desses acidentes. Alarmantemente, mais da metade (54%) de todas as mortes e lesões no trânsito envolvem usuários vulneráveis, como pedestres, ciclistas, motociclistas e seus passageiros.

Figura 1 – Distribuição de mortes por tipo de usuário das estradas por região da *WHO* no 2018




Fonte: Adaptado de Vanderschuren e Roux (2019).

Os dados apresentados têm gerado uma mobilização global intensa em busca do desenvolvimento de estratégias para aumentar a segurança nas ruas e, consequentemente, reduzir o número de vítimas no trânsito. Um exemplo dessa iniciativa é a Segunda Década de Ação para a Segurança no Trânsito (NATIONS., 2020), estabelecida durante a Assembleia Geral da ONU. Seu objetivo, no período de 2021 a 2030, é a redução de, no mínimo, 50% das lesões e mortes

relacionadas ao trânsito em todo o mundo.

Para assegurar a segurança e o funcionamento eficiente do fluxo de tráfego diário, é importante que as autoridades realizem o monitoramento e a fiscalização das vias, utilizando dados provenientes de câmeras instaladas (FERRANTE *et al.*, 2021). Essas medidas de controle estão diretamente relacionadas à regulamentação dos limites de velocidade, detecção de situações anormais e identificação de veículos parados, entre outras funções, que visam prevenir congestionamentos em rodovias (GOMIDES *et al.*, 2022). Em resumo, é fundamental monitorar, detectar e rastrear o fluxo de veículos para garantir o funcionamento adequado das atividades da sociedade (HUK; KUROWSKI, 2022a).

Sistemas de percepção, que dependem de dados de câmeras, apresentam forte influência de condições adversas (GE *et al.*, 2023). Portanto, apesar dos dados de imagem terem impulsionado avanços significativos no campo da Visão Computacional nos últimos anos, existe uma necessidade urgente de explorar novas estratégias de percepção (LIU *et al.*, 2023).

A visão computacional, um campo de pesquisa na área de inteligência artificial, capacita os computadores a extrair informações significativas a partir de dados de imagens digitais, vídeos e outras entradas visuais, com o propósito de realizar ações ou fornecer recomendações baseadas nesses dados (SHABBIR; ANWER, 2018). Este campo tem demonstrado sua popularidade com o crescente número de aplicações industriais, incluindo, entre outras, visão robótica, interfaces homem-máquina, recuperação de informações, análise de imagens médicas, sistemas de segurança, vigilância de tráfego, entre outros (SANTOS, 2014).

No contexto específico da vigilância de tráfego, a detecção e rastreamento de veículos em movimento representam estágios chave para sistemas de visão computacional (MONTANARI, 2016). É essencial monitorar, detectar e rastrear o fluxo de veículos para garantir o funcionamento ininterrupto das atividades da sociedade (HUK; KUROWSKI, 2022b).

Para controlar o fluxo de veículos, um grande número de câmeras de vigilância tem sido instalado nas rodovias, com o propósito de monitorar e assegurar a segurança das pessoas que trafegam nessas vias (MARTINEZ *et al.*, 2010). Essa extensa rede de câmeras gera uma quantidade considerável de dados de tráfego veicular, levando os pesquisadores a concentrarem seus esforços no desenvolvimento em sistemas de visão computacional, para analisar e compreender esses dados (PAWAR; ATTAR, 2021). Esses dados de vídeo capturados nas rodovias são usados na obtenção de informações de trânsito, permitindo a determinação do número de veículos, suas velocidades, a detecção de anomalias no tráfego e o aviso de possíveis congestionamentos (DJENOURI *et al.*, 2022).

Assim, a análise e detecção de anomalias no tráfego representa uma etapa importante desse processo. Uma anomalia indica que algo não previsto está acontecendo na rodovia, e por tanto, é preciso conseguir identificar as anomalias para realizar uma ação apropriada à situação, e assim por exemplo, evitar possíveis acidentes (ZHANG *et al.*, 2021a). A detecção de

anomalias no tráfego rodoviário é uma tarefa fundamental de visão computacional e desempenha um papel crítico na análise da estrutura de vídeo e na análise do tráfego urbano (ZHAO, 2021). Dada a complexidade das condições de tráfego, a análise do tráfego continua sendo um desafio significativo, devido à complexidade do cenário de tráfego, variações nas orientações dos objetos em movimento, mudanças nas condições climáticas e a presença de objetos não relacionados em segundo plano (PAWAR; ATTAR, 2021).

Ainda em relação ao trânsito veicular, inúmeras situações imprevistas podem surgir, representando desafios para os motoristas. Para auxiliar na identificação dessas situações nas rodovias, recorre-se a sistemas de vigilância de tráfego, sistemas de visão computacional capazes de analisar e determinar, em tempo real, o fluxo veicular. Por meio desses sistemas, as autoridades podem tomar decisões mais rápidas e melhores, permitindo a correção imediata de situações críticas e, ao mesmo tempo, antecipar e evitar possíveis ocorrências semelhantes no futuro (GOMIDES *et al.*, 2022).

## 1.1 Justificação e motivação

Dentro do contexto crítico da segurança viária e da gestão do tráfego rodoviário, este trabalho se propõe a desenvolver um modelo inovador de detecção e rastreamento de veículos, utilizando a visão computacional como base. A motivação subjacente a esta pesquisa reside na necessidade premente de identificar e responder a anomalias de tráfego, proporcionando às autoridades ferramentas mais eficazes para garantir a segurança nas rodovias e otimizar o fluxo veicular.

## 1.2 Objetivos

O objetivo principal deste trabalho é desenvolver um sistema de detecção e rastreamento de veículos utilizando visão computacional para detectar anomalias de tráfego em rodovias. A ideia é analisar o tráfego de veículos a partir de cenas de vídeo de tráfego urbano, detectar veículos e rastreá-los, tanto aqueles em movimento quanto os que estão parados, com o objetivo de detectar anomalias no tráfego da rodovia.

Os objetivos específicos deste trabalho são os seguintes:

1. Realizar a detecção de veículos a partir de cenas de vídeo de tráfego urbano, estabelecendo a base para a coleta de dados de tráfego.

2. Efetuar o rastreamento contínuo de veículos, modelando o tráfego em um grafo bipartido de estrutura dinâmica em áreas de interesse, permitindo uma representação eficaz da dinâmica do tráfego ao longo do tempo.

3. Por fim, detectar anomalias nas rodovias, visando aprimorar a segurança viária e identificar situações incomuns que podem representar riscos ou causar congestionamentos.

Cada um desses objetivos contribui para a consecução do objetivo global deste trabalho: desenvolver um sistema de visão computacional capaz de detectar e rastrear anomalias no tráfego rodoviário, melhorando assim a gestão e segurança do tráfego.

## 1.3 Estrutura do Documento

A estrutura desta monografia está organizada de forma a proporcionar uma compreensão clara e progressiva do projeto.

- O Capítulo 2 faz uma introdução detalhada aos conceitos fundamentais que estabelecem a base para uma compreensão do tema abordado nesta monografia. Neste capítulo, são explicados os principais conceitos, teorias e terminologias para que os leitores se familiarizem com o contexto do projeto, permitindo uma apreciação mais sólida das discussões subsequentes.

- O Capítulo 3 revisa e contextualiza os trabalhos relacionados à proposta deste trabalho. Neste capítulo, são analisados estudos, projetos e abordagens anteriores que compartilham afinidades com a pesquisa em andamento, proporcionando uma visão panorâmica das contribuições e lacunas na área de estudo. Isso ajuda a situar o trabalho atual dentro do contexto mais amplo da pesquisa e a identificar como ele se insere e avança na literatura existente.

- O Capítulo 4 constitui uma peça central ao detalhar a proposta do projeto, a metodologia adotada, o fluxograma de trabalho e todas as etapas essenciais. Aqui, o leitor encontrará uma exposição completa da abordagem adotada, os métodos empregados, bem como os passos específicos que conduzem à realização dos objetivos do projeto.

- Os resultados do modelo desenvolvido são examinados e debatidos no Capítulo 5. Neste capítulo, os leitores encontrarão uma análise detalhada dos dados coletados e dos resultados obtidos, juntamente com discussões que visam explicar o significado e a relevância desses resultados no contexto do projeto.

- No Capítulo 6, encerra-se a monografia com a apresentação das conclusões . Neste capítulo, são apresentadas as conclusões do trabalho, ressaltando as vantagens e desvantagens de nosso método. Além disso, são descritos os possíveis trabalhos futuros e as contribuições intelectuais.

CAPÍTULO

# 2

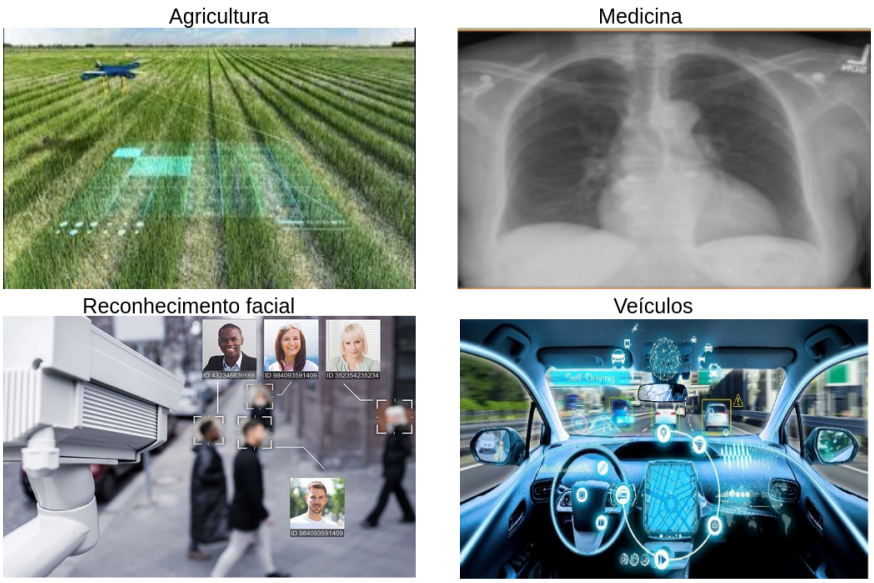
# FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, abordaremos os conceitos fundamentais que fornecem a base necessária para compreender o trabalho. A visão computacional será explicada em detalhes, e os vários componentes que compõem as etapas deste projeto, como detecção de objetos, rastreamento de objetos e detecção de anomalias de tráfego, serão minuciosamente exploradas. Esses conceitos serão apresentados em conformidade com o fluxograma da proposta, como ilustrado nas Figura 23 e Figura 25, conforme descrito no Capítulo 4.

## 2.1 Visão Computacional

A Visão Computacional é o processo de modelagem e replicação da visão humana usando *software* e *hardware* (HUGHES; FOLEY, 2014). Envolve desde o reconhecimento de objetos e caracteres até análise de texto e sentimento (CHARLES *et al.*, 2012). O atual reconhecimento de objetos executa apenas classificações e detecções de objetos simples, como uma banana ou uma bicicleta em uma imagem, mas o potencial da visão computacional é sobre-humano, como, por exemplo, ser capaz de ver claramente no escuro, através de paredes, em longas distâncias e processar todos esses dados rapidamente e em volume maciço (RUIZ; TODT, 2021). Já a visão computacional, em seu sentido mais pleno, está sendo usada na vida cotidiana e nos negócios para conduzir todos os tipos de tarefas, incluindo identificar doenças médicas em raios-x, identificar produtos e onde comprá-los, anúncios dentro de imagens editoriais e entre outros (ELLAHI *et al.*, 2011). Também pode ser usada para digitalizar plataformas de mídia social, afim de encontrar imagens relevantes que não podem ser descobertas por meio de pesquisas tradicionais (TOMIC *et al.*, 2012). A visão computacional é a principal técnica por trás dos veículos autônomos, que devem mudar completamente o mundo como o conhecemos (JIANG *et al.*, 2018). As principais aplicações da área incluem: Reconhecimento facial, Veículos autônomos, Robótica, Reconhecimento de objetos, Jogos e Áreas da saúde (CARVALHO, 2020), como ilustra a Figura 2.

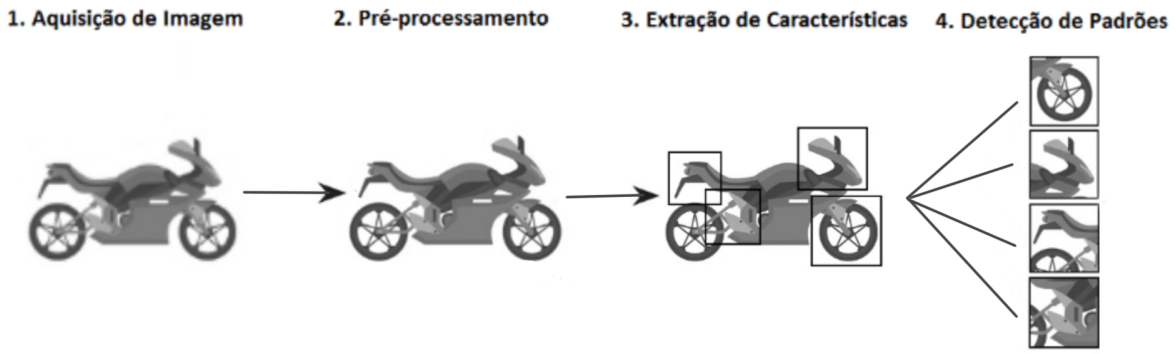
Figura 2 – Exemplo de aplicações da visão computacional.




Fonte: Adaptado de Carvalho (2020).

O desenvolvimento da visão computacional tem as seguintes etapas: aquisição de imagem, pré-processamento, extração de características e detecção de padrões como mostra a Figura 3. A primeira etapa consiste em enviar uma imagem ou um vídeo (sequência de imagens) para o sistema (TOMIC *et al.*, 2012). A segunda etapa consiste no pré-processamento da imagem, que são operações realizadas para realçar objetos, remover os ruídos da imagem, normalizar a imagem, tornando as partes escuras mais claras e as partes muito claras mais escuras e são utilizadas conforme a necessidade do problema que o sistema de visão computacional se propõe a resolver (PEREIRA, 2022). A etapa três, consiste em obter informações que tornam possível classificar ou identificar um objeto. As características de um objeto podem ser classificadas em categorias, sendo as principais: características de cor, de aspecto, de forma, entre outras (RUIZ; TODT, 2021). Por fim, a última etapa, é uma área que estuda técnicas para entender padrões de objetos a fim de classificá-los.

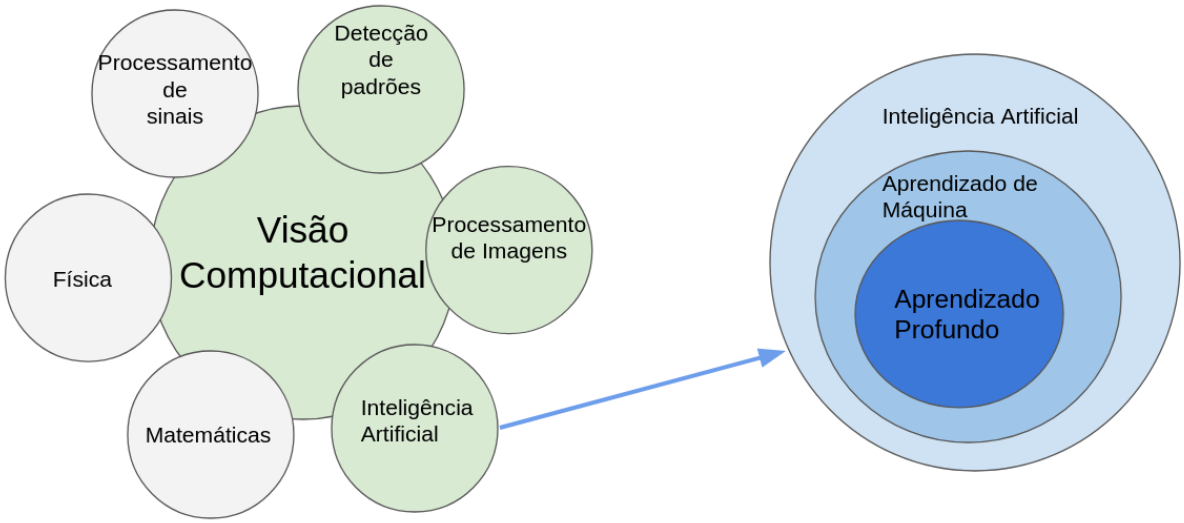
Figura 3 – Etapas de um sistema de Visão Computacional.




Fonte: Viana (2020).

A visão computacional é uma tecnologia que, assim como tantas outras, está sob o guarda-chuva da Inteligência Artificial (IA) (RUIZ; TODT, 2021), como ilustrado na Figura 4. Computadores e máquinas conseguem "enxergar", ou seja, é possível que eles consigam entender o mundo visual, interpretando e extraindo informações das imagens que captam por meio de câmeras e sensores (NUNES, 2023). Para que isso seja possível, a visão computacional emprega outras duas tecnologias que derivam da IA: o *Aprendizado de Máquina (AM)* e o *Aprendizado Profundo (AP)* (HUGHES; FOLEY, 2014). Assim, além de reproduzir determinados aspectos da visão humana, a visão computacional consegue ser ainda mais precisa, uma vez que consegue identificar detalhes que geralmente passariam despercebidos por um observador (NUNES, 2023).

Figura 4 – Visão computacional parte de outras disciplinas




Fonte: Adaptado de Nunes (2023).

A AM é um subcampo da IA que lida com algoritmos de computação que podem ser melhorados via dados de treinamento sem programação explícita. É considerado o caminho mais promissor para alcançar a IA verdadeiramente próxima à humana (DAVIES, 2022). Os algoritmos de AM podem ser classificados, de maneira geral, em três categorias:

- **Aprendizagem supervisionada:** São treinados com *datasets* rotulados, que permitem que os modelos aprendam e se tornem mais precisos ao longo do tempo. Isso permite ao algoritmo aprender as regras que mapeiam entradas e saídas (JURADO-RODRíGUEZ *et al.*, 2022).
- **Aprendizagem não supervisionada:** Procuram padrões em dados não rotulados e também podem encontrar padrões ou tendências que as pessoas não estão procurando explicitamente (DAVIES, 2022).
- **Aprendizagem por reforço:** Treinam as máquinas por meio de tentativa e erro para tomar a melhor ação, estabelecendo um sistema de recompensa. O aprendizado por reforço

pode treinar modelos para jogar ou treinar veículos autônomos para dirigir, informando à máquina quando ela tomou as decisões corretas, o que a ajuda a aprender ao longo do tempo quais ações devem ser tomadas (JURADO-RODRíGUEZ *et al.*, 2022).

A AP é um ramo do AM que usa redes neurais artificiais para se aproximar da inteligência humana. Inspirada pelos neurônios humanos, a AP usa a teoria dos grafos para organizar algoritmos ponderados em camadas de vértices e arestas (Pavan Vadapalli, 2021). Os algoritmos da AP são ótimos em processar dados não estruturados, como imagens ou linguagem (DUAN *et al.*, 2022). Tecnicamente, para ser classificada como profunda, a rede neural precisa conter camadas ocultas entre as camadas de entrada e saída da perceptron, que é a estrutura base de uma rede neural. Essas camadas são consideradas ocultas porque não têm conexão com o mundo exterior (Pavan Vadapalli, 2021). Dentre as principais arquiteturas do Aprendizado Profundo temos 3 categorias específicas:

- ***Feedforward Neural Networks (FF)***: Os dados trafegam em uma direção desde a camada de entrada, passando pelas camadas ocultas, até a camada de saída; todos os vértices estão conectados e os dados nunca retomam o ciclo nas camadas ocultas. A *FF* é usada na compactação de dados e no processamento básico de imagens (JURADO-RODRíGUEZ *et al.*, 2022).

- ***Recurrent Neural Networks (RNN)***: È um tipo de rede *FF* que adiciona um tempo de espera às camadas ocultas, o que permite o acesso à informações anteriores durante uma iteração corrente. Esse *loop* de *FF* se aproxima da memória, o que torna as *RNNs* adequadas para processamento de linguagem. Um bom exemplo disso é o texto preditivo que se baseia em palavras usadas com mais frequência para personalizar as sugestões (Pavan Vadapalli, 2021).

- ***Convolutional Neural Networks (CNN)*** : A *CNN* é uma operação matemática em duas funções que produz uma terceira, descrevendo como uma é modificada pela outra. Usada principalmente para reconhecimento e classificação de imagens, as *CNNs* são os “olhos” da IA. As camadas ocultas em uma *CNN* agem como filtros matemáticos usando a soma ponderada para identificar arestas, cores, contraste e outros elementos de um *pixel* (DAVIES, 2022).

### 2.1.1 Detecção de objetos

Uma das aplicações de grande importância da visão computacional e a IA é a detecção de objetos (FRITZ LABS INCORPORATED, 2022). A detecção de objetos tenta responder à pergunta: Quais objetos estão nesta imagem e onde eles estão? (AZEVEDO, 2022). A detecção de objetos tem como finalidade identificar a presença e a localização de objetos específicos em imagens ou vídeos, o que é essencial em aplicações como carros autônomos, vigilância de

segurança, análise de vídeo e muito mais (MOUTIK *et al.*, 2021). Especificamente, a detecção de objetos desenha *bounding boxes* em torno desses objetos detectados, o que nos permite localizar onde esses objetos estão (ou como eles se movem) em uma determinada cena (SALARI *et al.*, 2022).

Figura 5 – *Bounding boxes* em torno dos objetos detectados


Fonte: Gautam e Singh (2021).

Diversos são os desafios para a detecção, pois podem existir alguns fatores e cenários onde a detecção pode ser comprometida, como, por exemplo, oclusões de ambiente, pouca ou muita iluminação, baixa resolução, cenários noturnos e até a falta de dados de amostra. Existem diversas abordagens para a detecção de objetos já aplicadas na literatura, como o uso de *Histograms of Oriented Gradient (HOG)* , *Support Vector Machine (SVM)* , *Scale Invert Feature Transform (SIFT)* , *Principal Component Analysis (PCA)* . Atualmente, o estado da arte na área de detecção de objetos é com uso de *CNN* (FERRANTE *et al.*, 2022)

Os modelos de AP geralmente tem dois tipos de rede de detecção de objetos: rede de um estágio ou rede de dois estágios. Os detectores de objetos de estágio único produzem previsões de rede para regiões em toda a imagem usando *boxes* de âncora e, em seguida, as previsões são decodificadas para gerar os *bounding boxes* finais para os objetos (DUAN *et al.*, 2022). Enquanto os detectores de dois estágios identificam, no estágio inicial, propostas de regiões ou subconjuntos da imagem que podem conter um objeto e, na segunda etapa, identificam os objetos dentro das propostas de região. As redes de dois estágios podem obter resultados de detecção de objetos muito precisos; no entanto, elas são normalmente mais lentas do que as redes de estágio único (Aditya Sharma, 2022).

Dentro da rede de um estágio temos a arquitetura *You Only Look Once (YOLO)* . Esta

arquitetura é um sistema de código aberto do estado da arte para detecção de objetos em tempo real. Mais detalhes em Subseção 2.1.2.

### 2.1.2 You Only Look Once (YOLO)

O *YOLO* é uma ferramenta da visão computacional para detecção e classificação de objetos em tempo real (SALMAN *et al.*, 2022). Propõe o uso de uma rede neural de ponta a ponta que faz todas as suas previsões de *bounding boxes* e probabilidades de classes, atingindo resultados ótimos, superando outros algoritmos de detecção de objetos em tempo real com uma grande margem (AZAM *et al.*, 2022).

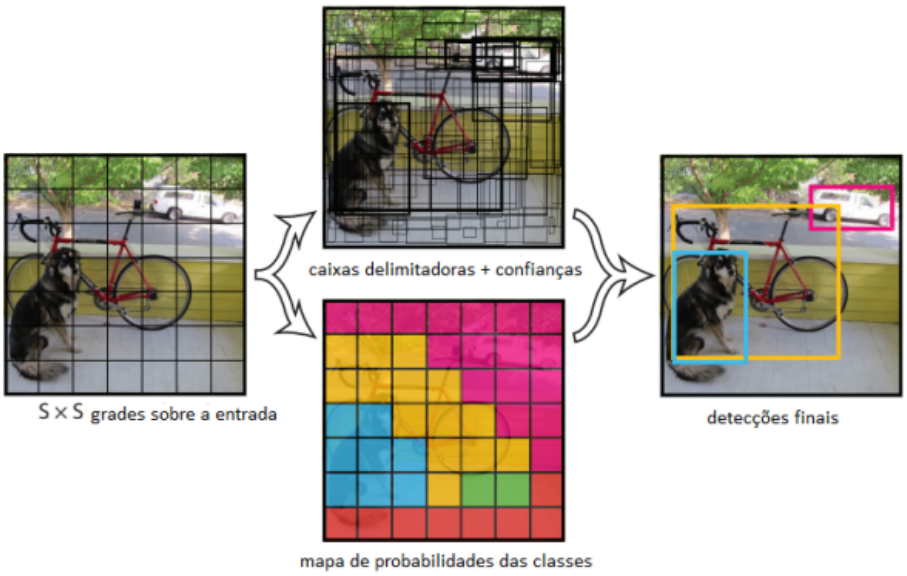
A Figura 6 apresenta o processo geral do algoritmo YOLO. O algoritmo funciona dividindo uma imagem em $N$ grades, cada uma com uma região dimensional de $S \times S$ *pixels*. Cada uma dessas $N$ grades é responsável pela detecção e localização do objeto que contém (SALMAN *et al.*, 2022). As grades predizem as coordenadas do *bounding box* em relação às suas coordenadas de célula, juntamente com o rótulo do objeto e a probabilidade do objeto estar presente na célula (REDMON *et al.*, 2016). Esse processo reduz bastante o tempo de computação, pois tanto a detecção quanto o reconhecimento são manipulados pelas células da imagem, mas ele traz muitas previsões duplicadas devido às várias células prevendo o mesmo objeto com diferentes previsões de *bounding box* (Aditya Sharma, 2022). Para lidar com esse problema o YOLO usa a *Non-Maximum Suppression (NMS)* . No *NMS*, o *YOLO* suprime todos os *bounding boxes* que possuem pontuações de probabilidade mais baixas. O *YOLO* consegue isso analisando primeiro as pontuações de probabilidade associadas a cada decisão e escolhendo a maior. Em seguida, ele suprime os *bounding boxes* com a maior *Intersection over Union* (IOU) com o *bounding box* atual de alta probabilidade. Este passo é repetido até que os *bounding boxes* finais sejam obtidos (REDMON *et al.*, 2016).

### 2.1.3 Rastreamento de Objetos

Outra das aplicações de grande importância da visão computacional e a IA é o rastreamento de objetos. O rastreamento de objetos responde à pergunta: Para onde esses objetos estão indo entre os *frames*? (AZEVEDO, 2022). O rastreamento de objetos é uma área dentro da visão computacional e a IA, no qual pega um conjunto inicial de detecção de objetos e desenvolve uma identificação única para cada uma das detecções iniciais, e em seguida rastreia os objetos detectados à medida que se movem nos *frames* de um vídeo (SYEDRZ, 2020), como ilustra a Figura 7. Em outras palavras, o rastreamento de objetos é a tarefa de identificar automaticamente objetos em um vídeo e interpretá-los como um conjunto de trajetórias com alta precisão (MEEL, 2022).

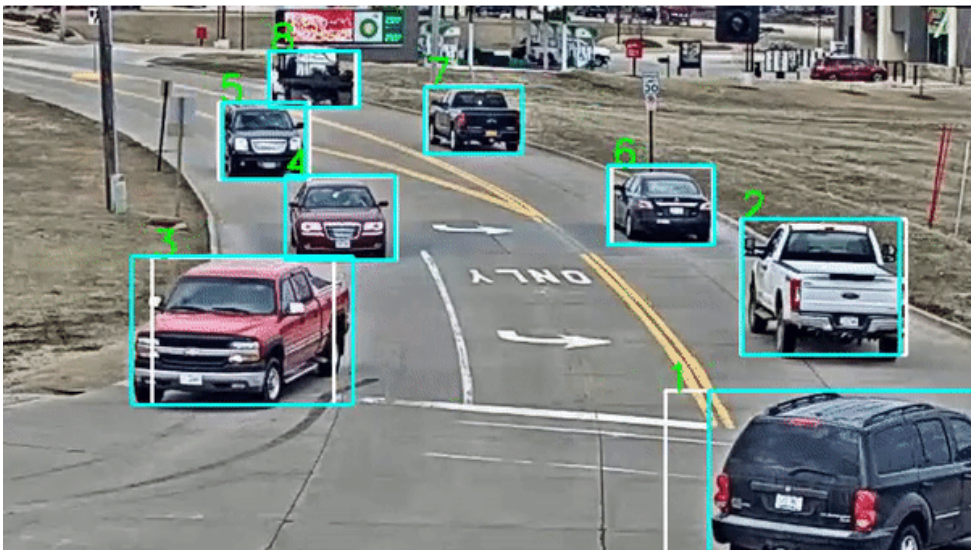
Atualmente, existem duas estruturas principais de rastreamento de objetos: Rastreamento Sem Detecção (RSD) e Rastreamento Baseado em Detecção (RBD) (SYEDRZ, 2020). O RSD

Figura 6 – Processo geral de detecção de objetos da arquitetura YOLO




Fonte: Adaptado de Redmon *et al.* (2016).

Figura 7 – Rastreamento de objetos



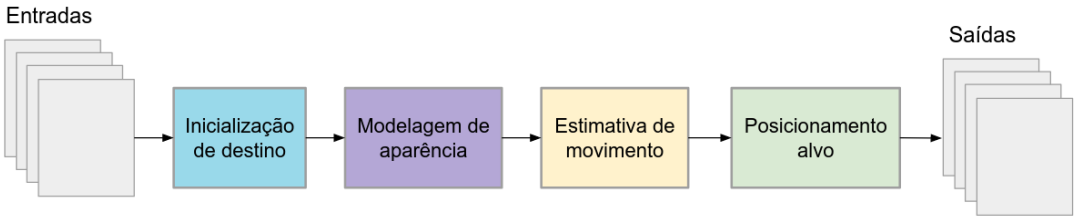
Fonte: Meel (2022).

precisa inicializar manualmente o destino de rastreamento, portanto é aplicável apenas ao rastreamento de um destino especificado e não pode detectar e rastrear automaticamente um novo destino que aparece no processo de monitoramento (MEEL, 2022). O RBD integra detecção e rastreamento e pode detectar automaticamente a aparição de novos objetos ou o desaparecimento de objetos existentes. Assim, o RBD é capaz de atender aos requisitos reais do desaparecimento aleatório de objetos ou da mudança dinâmica de objetos na cena de monitoramento. (YANG *et al.*, 2020).

O rastreamento de objetos tem as seguintes etapas: Inicialização de destino, modelagem

de aparência, estimativa de movimento e posicionamento alvo (AERIAL. . . , 2021), como ilustra a Figura 8. Na etapa de inicialização de destino o objetivo principal é definir o objeto de interesse. Tendo o objeto de interesse são construídos os *bounding boxes* dentro dos *frames* das imagens ou vídeos (Nilesh Barla, 2022). A etapa de modelagem de aparência lida com a modelagem da aparência visual do objeto. Quando o objeto alvo passa por mudanças de cenário, como variações nas condições de iluminação, ângulo, velocidade, etc., pode acontecer a alteração da aparência do objeto e isso pode levar à perda de informação e o algoritmo perder o rastro do objeto. Para isso a modelagem de aparência deve ser utilizada para diminuir as distorções introduzidas quando o objeto alvo se move (MEEL, 2022). A etapa de estimativa de movimento, geralmente infere a capacidade preditiva do modelo para prever com precisão a posição futura do objeto (AERIAL. . . , 2021). Na última etapa de posicionamento do objeto, a estimativa de movimento aproxima a possível região onde o objeto poderia estar presente (Nilesh Barla, 2022).

Figura 8 – Pipeline do rastreamento de objetos




Fonte: Adaptado de Nilesh Barla (2022).

O rastreamento de objetos e o reconhecimento de características são tarefas comuns em visão computacional e processamento de imagens. Existem vários métodos e técnicas para extrair características de uma imagem que podem ser úteis para o rastreamento de objetos. Vamos descrever alguns deles.

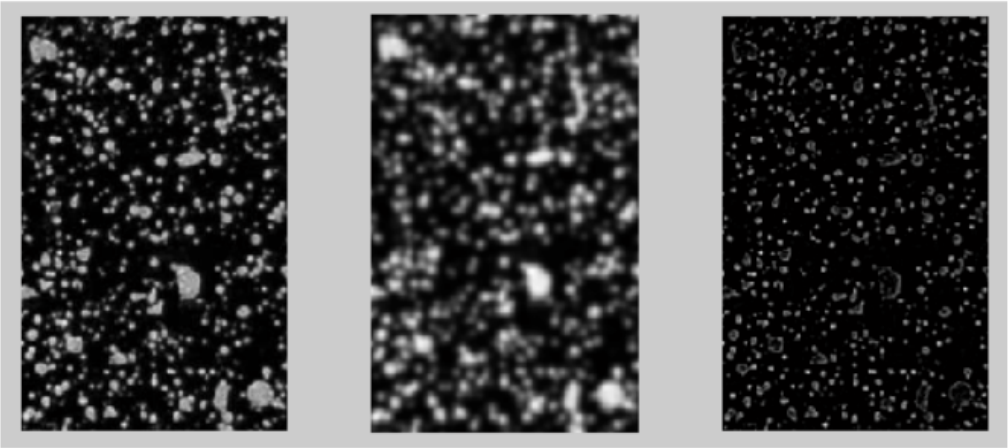
#### 2.1.3.1 SIFT

*SIFT* é um algoritmo de visão computacional publicado por David Lowe em 1999 e patenteado nos EUA pela *University of British Columbia*. *SIFT* é composto por duas partes distintas: o detector e o descritor. O detector *SIFT* é baseado em cálculos de diferença de Gaussianas e o descritor *SIFT* utiliza histogramas de gradientes orientados para descrever a vizinhança local dos pontos de interesse (LOWE, 2004). O algoritmo *SIFT* é executado através de quatro etapas principais: detecção de extremos, localização de pontos-chave, definição da orientação e descrição dos pontos-chave. As duas primeiras descrevem a parte do detector e as duas seguintes descrevem a formação do descritor (CHEUNG; HAMARNEH, 2009).

A Figura 9 descreve a primeira etapa da técnica *SIFT* que consiste em buscar pontos que sejam invariantes à mudanças de escala da imagem, possibilitando a detecção de pontos com a câmera próxima ou distante do objeto de interesse. Tal objetivo é alcançado procurando

características estáveis em diferentes escalas, utilizando uma função chamada de espaço de escala que, neste caso, é a função Gaussiana (CHEUNG; HAMARNEH, 2009).

Figura 9 – Exemplos de aplicação dos filtros. Imagem de padrão de granulados(esquerda), filtro Gaussiano (centro) e resultado da Diferença Gaussiana(direita).



Fonte: Cheung e Hamarneh (2009).

A Figura 10 descreve todos os pontos detectados como extremos, são candidatos a pontos-chave. Deseja-se agora calcular a localização exata destes pontos-chave. O método consiste em ajustar uma função quadrática 3D do ponto de amostragem local, de modo a determinar uma localização interpolada do máximo. Isto é feito utilizando uma expansão de *Taylor* da função Diferença de Gaussiano aplicada à imagem (LOWE, 2004).

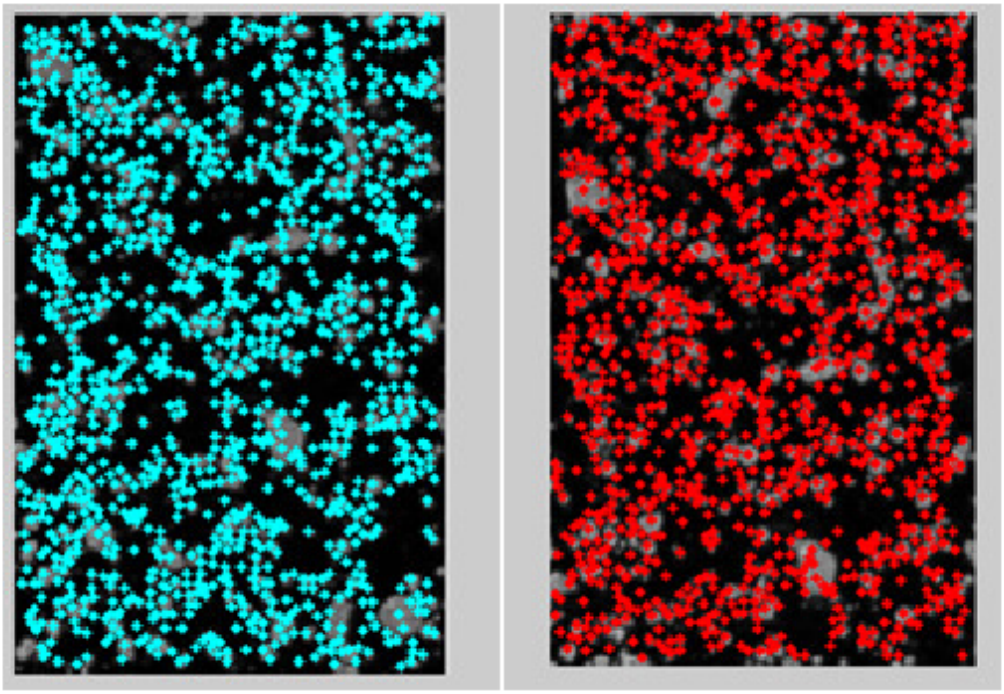
A Figura 11 descreve que a cada ponto-chave é atribuída uma orientação, para se construir descritores invariantes quanto à rotação. Essa invariância é obtida através das características locais da imagem (LOWE, 2004).

A Figura 12 descreve como será atribuído a cada ponto-chave um descritor invariante à iluminação e ponto de vista 3D, tornando-os bem distinguíveis. É importante lembrar que os procedimentos a seguir são feitos com os valores normalizados em relação à orientação e magnitude de gradiente (LOWE, 2004).

#### 2.1.3.2 Similaridade Estrutural (SSMI)

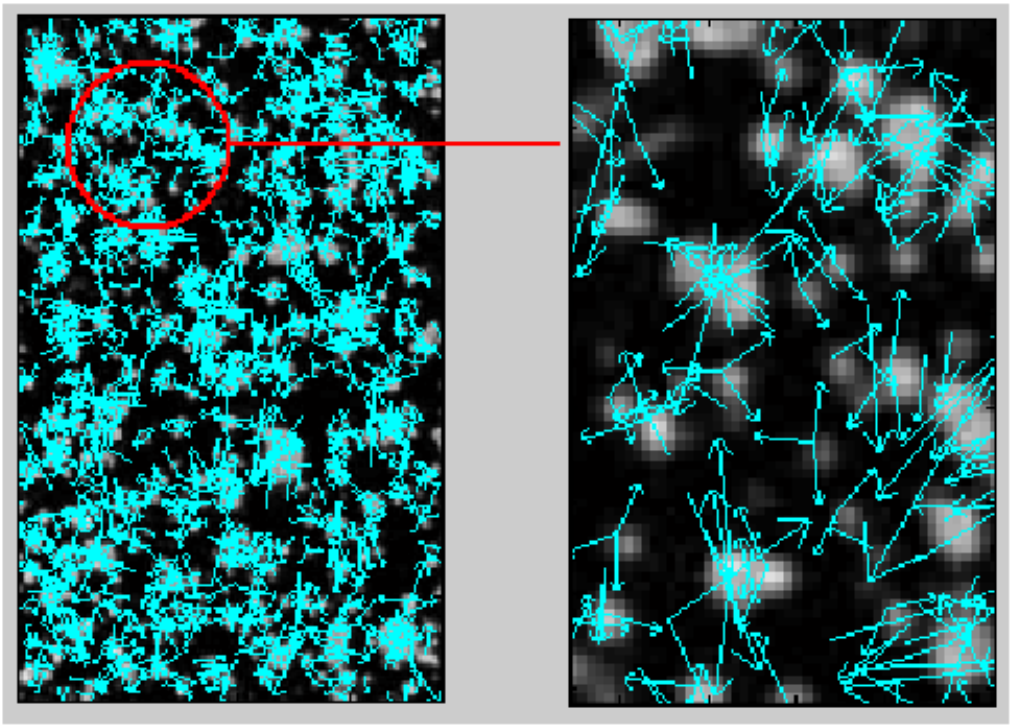
A Similaridade Estrutural (*SSIM*) é usada como uma métrica para medir a similaridade entre duas imagens. A *SSIM* extrai três características principais de uma imagem: luminância, contraste e estrutura (DATTA, 2023). A luminância é medida pela média de todos os valores de pixel. O contraste é medido tomando o desvio padrão (raiz quadrada da variância) de todos os valores de pixel. A comparação estrutural é feita dividindo o sinal de entrada pelo seu desvio padrão para que o resultado tenha desvio padrão unitário, o que permite uma comparação mais robusta (CANDELA[1]; MORABITO, 2023). A comparação entre duas imagens é realizada com

Figura 10 – Pontos-Chaves localizados em duas imagens.



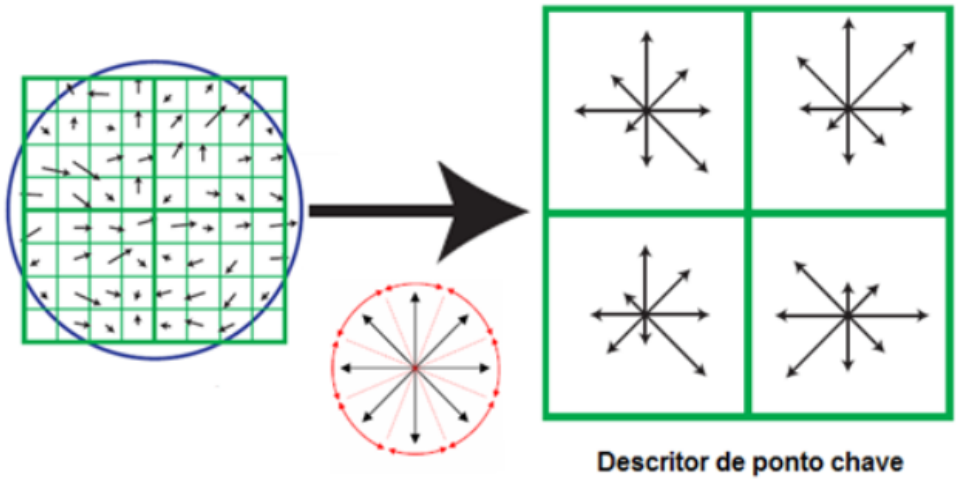
Fonte: Cheung e Hamarneh (2009).

Figura 11 – Atribuição de orientação e magnitude a cada ponto-chave.



Fonte: Cheung e Hamarneh (2009).

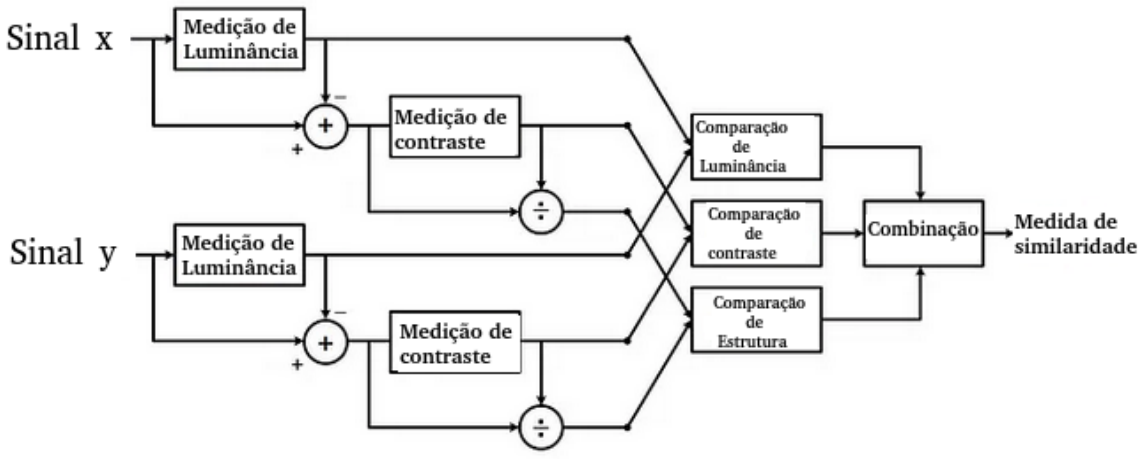
Figura 12 – Construção do descritor para um ponto-chave de 2x2 com 48 elementos.




Fonte: Cheung e Hamarneh (2009).

base nessas 3 características, como ilustra a Figura 13, onde sinal x e sinal y referem-se às imagens de referência e amostra.

Figura 13 – Sistema de Medição de Similaridade Estrutural.




Fonte: Adaptado de Candela[1] e Morabito (2023).

Este sistema calcula o Índice de Similaridade Estrutural entre 2 imagens, que é um valor entre -1 e +1. Um valor de +1 indica que as 2 imagens fornecidas são muito semelhantes ou iguais, enquanto um valor de -1 indica que as 2 imagens fornecidas são muito diferentes. Frequentemente, esses valores são ajustados para ficar na faixa [0, 1], onde os extremos têm o mesmo significado (CANDELA[1]; MORABITO, 2023).

#### 2.1.3.3 Representação do vetor denso

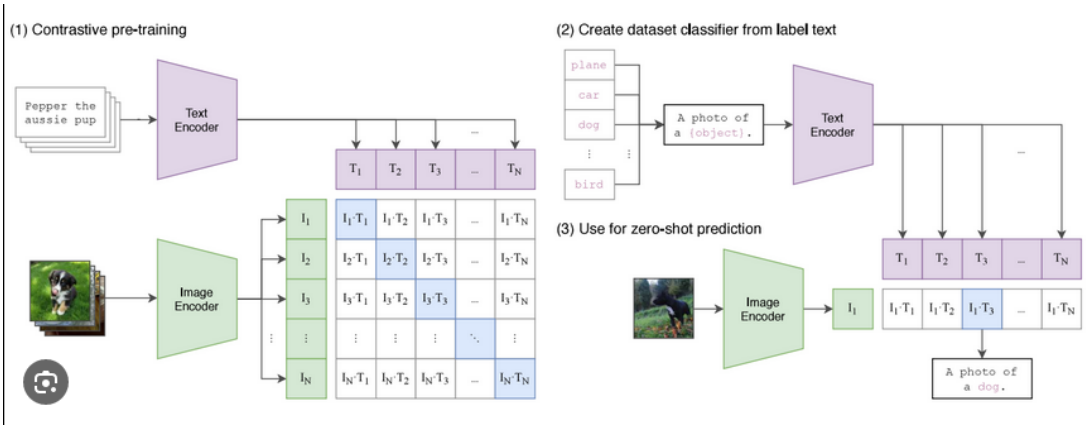
Os vetores densos se referem a vetores numéricos que representam informações de alta dimensionalidade de forma compacta e contínua. Esses vetores são usados em várias aplicações de AP e *Natural Language Processing (NLP)* (ZUO *et al.*, 2022). Aqui estão alguns pontos importantes: representação vetorial, alta dimensionalidade, continuidade, aprendizado de representações e aplicações. Os vetores densos são usados para representar informações complexas, como palavras, frases, documentos, imagens ou qualquer outro tipo de dado . Cada elemento do vetor contém informações relevantes sobre o objeto que está sendo representado. Em muitos casos, os vetores densos têm uma alta dimensionalidade, o que significa que podem ter muitos elementos. Por exemplo, vetores densos usados em *NLP* podem ter centenas ou milhares de dimensões (RANFTL; BOCHKOVSKIY; KOLTUN, 2021). Ao contrário de representações binárias ou esparsas, os vetores densos são contínuos, o que significa que os valores de cada elemento do vetor podem ser qualquer número real. Essa característica permite uma representação mais rica e flexível dos dados. Os vetores densos são frequentemente aprendidos a partir dos dados por meio de técnicas de aprendizado profundo, como redes neurais. Isso permite que o sistema aprenda representações significativas dos dados de forma automatizada (ZUO *et al.*, 2022). Vetores densos são usados em várias aplicações, incluindo tradução automática, análise de sentimentos, recomendação de produtos, classificação de documentos e muito mais. Eles também são comuns em modelos de linguagem, como o *Generative Pre-trained Transformer (GPT)* , onde palavras, frases e documentos são representados por vetores densos (GU *et al.*, 2020).

Existem muitas tecnologias para a construção de vetores densos, desde representações vetoriais de palavras ou frases até mesmo textos e imagens. Algumas das técnicas e abordagens mais comuns incluem: *Word Embeddings* (Incorporação de Palavras), *CNN* e *RNN*, *Transformers*, AP para Visão Computacional, *Cross-media embeddings*. Dentro dos *Transformers* temos o *Contrastive Language-Image Pre-training (CLIP)* , que é uma técnica que envolve uma arquitetura de rede neural profunda (ZUO *et al.*, 2022).

O *CLIP* é um modelo de aprendizado de máquina desenvolvido pela *OpenAI* que se destaca por sua capacidade de entender a relação entre texto e imagens em uma variedade de tarefas. Ele representa uma abordagem inovadora para criar representações densas que cruzam modalidades, ou seja, que associam texto e imagens de maneira significativa (RANFTL; BOCHKOVSKIY; KOLTUN, 2021). O *CLIP* é pré-treinado em uma grande quantidade de dados multimodais que incluem imagens e texto e é projetado para entender as relações semânticas entre essas modalidades. A abordagem central do *CLIP* é o pré-treinamento contrastivo, no qual o modelo aprende a associar pares de imagens e descrições de texto que se referem à mesma coisa, permitindo que ele compreenda a relação entre o conteúdo visual e textual, como ilustra a Figura 14. O *CLIP* usa a arquitetura do *Transformer* para realizar essa tarefa de associação entre texto e imagens, e o pré-treinamento contrastivo ajuda a criar representações densas que

capturam as semelhanças e diferenças entre as duas modalidades. Depois do pré-treinamento, o *CLIP* pode ser afinado para tarefas específicas, como classificação de imagens ou pesquisa de texto por imagens (ZUO *et al.*, 2022).

Figura 14 – O *CLIP* pré-treina um codificador de imagem e um codificador de texto.




Fonte: Ranftl, Bochkovskiy e Koltun (2021).

## 2.2 Estruturas de Dados e Algoritmos

Nesta seção, vamos explicar algumas estruturas de dados para as etapas de rastreamento e detecção de anomalias, bem como os algoritmos que serão utilizados para aprimorar a eficiência e a precisão do processo.

### 2.2.1 Grafo Bipartido

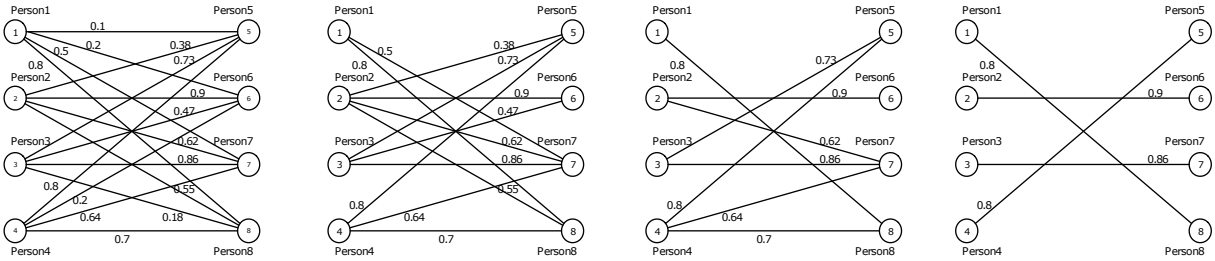
Um grafo é uma estrutura de dados que em matemática consiste em um conjunto de nós e um conjunto de pares ordenados chamados arestas, que são criadas com nós distintos (ZOLA *et al.*, 2022). O grafo é geralmente escrito na forma $G = (V, E)$, onde $V$ representa o conjunto de nós e $E$ representa o conjunto de arestas. Suponhamos que $v_1$ e $v_2$ são nós, então uma aresta é escrita como o par $(v_1, v_2)$, onde a aresta é a linha conectando esses dois pontos. Então dizemos que $v_1$ e $v_2$ são conectados por essa aresta (METCALF; CASEY, 2016). Na visualização do grafo, os nós são pontos enquanto as arestas são linhas conectando dois pontos (PUPYREV *et al.*, 2011).

Um grafo bipartido, também chamado bi-grafo, é um grafo onde os nós podem ser divididos em dois conjuntos disjuntos e nenhuma aresta está conectando os nós no mesmo conjunto (YU *et al.*, 2019). Formalmente podemos definir um grafo bipartido por $G = (U, V, E, \omega)$, onde $U, V$ são conjuntos disjuntos de nós respectivamente, conhecidos como bipartições, $E \subset U \times V$ denota as arestas entre os nós em $U$ e $V$, e $\omega$ denota os pesos nas arestas (CHEN; SUN, 2020).

Um caso especial de grafo bipartido é o grafo bipartido completo. Neste grafo, cada vértice de um conjunto está conectado a todos os vértices de outro conjunto (TAMURA; ITO; ZHOU, 2021). Representamos um grafo bipartido completo por $K_{m,n}$, onde $m$ é o tamanho do primeiro conjunto e $n$ é o tamanho do segundo conjunto. Assim por exemplo, um grafo bipartido completo $K_{3,2}$ tem três nós no primeiro conjunto, dois nós no segundo e cada nó no segundo conjunto está conectado aos nós no primeiro conjunto, então, o número de arestas neste grafo é 6 (METCALF; CASEY, 2016).

A partir de um grafo bipartido completo com pesos, podemos realizar um processo de poda de arestas. O processo de poda consiste em eliminar arestas que não superam um limiar especificado. Nste trabalho se fará uso da poda para obter correspondência dos objetos de *frame* a *frame*, como é explicado no Capítulo 3 de Trabalhos Correlatos. A Figura 15 apresenta um exemplo de um grafo bipartido e o processo de poda. O grafo bipartido aparece junto com a sua matriz de adjacência, que contém os pesos do grafo. Pode se observar que conforme aumenta o limiar (*threshold* em inglês) o número de arestas diminui. Na sequencia de imagens do grafo, o limiar é comparado com os pesos das arestas, e se este é menor que o limiar a aresta é podada.

Figura 15 – Exemplo de grafo bipartido e processo de poda aplicando um limiar.




| | Person 1 | Person 2 | Person 3 | Person 4 |
|---|---|---|---|---|
| Person 5 | 0.1 | 0.38 | 0.73 | 0.8 |
| Person 6 | 0.2 | 0.9 | 0.47 | 0.2 |
| Person 7 | 0.5 | 0.62 | 0.86 | 0.64 |
| Person 8 | 0.8 | 0.55 | 0.18 | 0.7 |

| | Person 1 | Person 2 | Person 3 | Person 4 |
|---|---|---|---|---|
| Person 5 | 0 | 0.38 | 0.73 | 0.8 |
| Person 6 | 0 | 0.9 | 0.47 | 0 |
| Person 7 | 0.5 | 0.62 | 0.86 | 0.64 |
| Person 8 | 0.8 | 0.55 | 0 | 0.7 |

| | Person 1 | Person 2 | Person 3 | Person 4 |
|---|---|---|---|---|
| Person 5 | 0 | 0 | 0.73 | 0.8 |
| Person 6 | 0 | 0.9 | 0 | 0 |
| Person 7 | 0 | 0.62 | 0.86 | 0.64 |
| Person 8 | 0.8 | 0 | 0 | 0.7 |

| | Person 1 | Person 2 | Person 3 | Person 4 |
|---|---|---|---|---|
| Person 5 | 0 | 0 | 0 | 0.8 |
| Person 6 | 0 | 0.9 | 0 | 0 |
| Person 7 | 0 | 0 | 0.86 | 0 |
| Person 8 | 0.8 | 0 | 0 | 0 |

Fonte: Collarana *et al.* (2018).

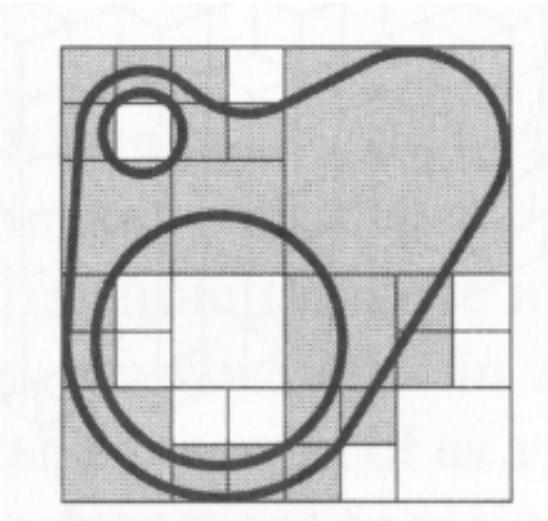
Muitos problemas do mundo real podem ser modelados como um grafo bipartido. Para modelar os grafos bipartidos, a correspondência gráfica bipartida é amplamente utilizada para encontrar uma solução ótima para estes problemas. A correspondência gráfica bipartida é uma técnica de otimização na teoria de grafos, que é usada para encontrar uma correspondência ótima entre os nós dos grafos (CARUSI; BIANCHI, 2019). A correspondência gráfica bipartida tenta encontrar um subconjunto de arestas, de forma que não haja duas arestas em um conjunto que compartilhem um nó (GAO *et al.*, 2013). Em problemas de reconhecimento de padrões, a correspondência gráfica bipartida é usada para verificar a similaridade entre dois grafos (DWIVEDI; SINGH, 2020). A modelagem de grafo bipartido é uma técnica poderosa na

aprendizagem de máquinas e aplicações de mineração de dados. (HASHEMI; DOWLATSHAHI; NEZAMABADI-POUR, 2021). Para o rastreamento de múltiplos objetos, são usados métodos de correspondência gráfica bipartida a ser associada com os objetos em duas estruturas vizinhas ou *frames* vizinhos, onde um nó representa um objeto e as arestas representam os pesos entre cada dois nós dos *frames* vizinhos e os pesos são as similaridades entre os dois objetos em diferentes *frames* (CHEN; LIN; LIU, 2001).

### 2.2.2 QuadTree

Uma *QuadTree* é uma forma de representação bidimensional, criada no final dos anos 60. A ideia fundamental é a capacidade de aproveitamento do conceito de dividir para conquistar de uma subdivisão binária (MIRANDA; SILVA, 2020). Uma *QuadTree* é criada pela subdivisão sucessiva de um plano bidimensional, para que sejam formados quadrantes, como mostrado na Figura 16. Quando uma *QuadTree* é utilizada para representar uma área no plano, cada quadrante pode estar: inteiramente contido nesta área, parcialmente contido, ou estar vazio, de acordo com a interseção do quadrante com a área considerada (MIRANDA; SILVA, 2020). Um quadrante parcialmente contido é subdividido recursivamente em sub quadrantes. Estas subdivisões continuam até que todos os quadrantes sejam homogêneos (inteiramente contidos ou não) ou até que determinado nível de subdivisão predeterminado seja alcançado (LIMA; MOTA; PINTO, 2012). As subdivisões sucessivas podem ser representadas com uma árvore, onde os quadrantes parcialmente contidos são os nós internos e os quadrantes vazios ou completos são as folhas, como ilustra a Figura 17.

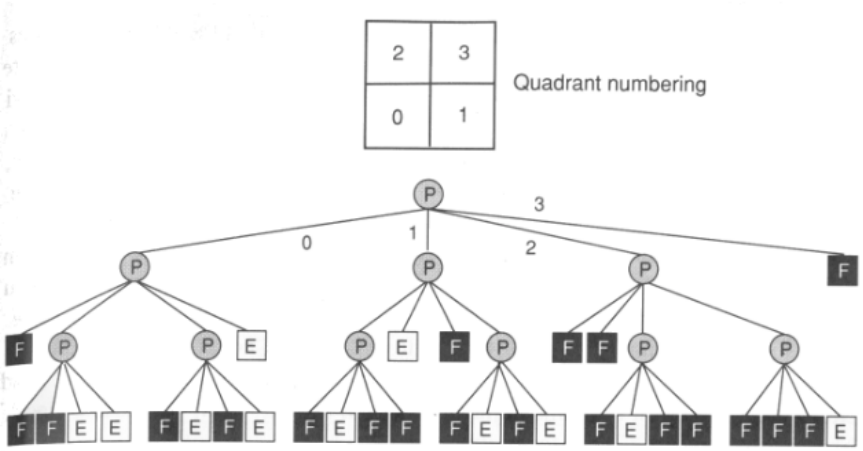
Figura 16 – Representação de um objeto usando *QuadTree*.



Fonte: Miranda e Silva (2020).

Uma *QuadTree* é uma estrutura de dados usada para codificar imagens. A ideia fundamental por trás da ideia de uma *QuadTree* é que qualquer imagem pode ser dividida em

Figura 17 – Estrutura da *QuadTree* para a Figura 16.




Fonte: Miranda e Silva (2020).

quatro quadrantes. Cada quadrante pode ser, recursivamente, dividido em quatro quadrantes. Em uma *QuadTree*, a imagem é representada pelo nó pai, enquanto que os quatro quadrantes são representados pelos quatro nós filhos (em uma determinada ordem) (CHIEN; KANADE, 1989). É claro que se toda a imagem for de uma única cor, ela poderá ser representada por uma *QuadTree* consistindo apenas de um nó. Em geral, uma *QuadTree* precisa ser subdividida em quadrantes se possui pixeis de diferentes cores. Como resultado, uma *QuadTree* não precisa ter uma altura uniforme (não precisa ser uma árvore completa) (LIMA; MOTA; PINTO, 2012).

### 2.2.3 Matching

Um *matching* é um subconjunto de arestas em que nenhum nó ocorre mais de uma vez. O peso de um *matching* é a soma dos pesos de suas arestas. A cardinalidade de um *matching* é o número de arestas combinadas (HIGGOTT, 2022). Entre os diferentes tipos de *matching* temos os algoritmos de peso máximo e os algoritmos de peso mínimo (JINJIANG, 1998). Os algoritmos de peso mínimo substitui os pesos das arestas por 1, mais o peso máximo das arestas, menos o peso da borda original Equação 2.1.

$$minimo = (maximo + 1) - borda \tag{2.1}$$

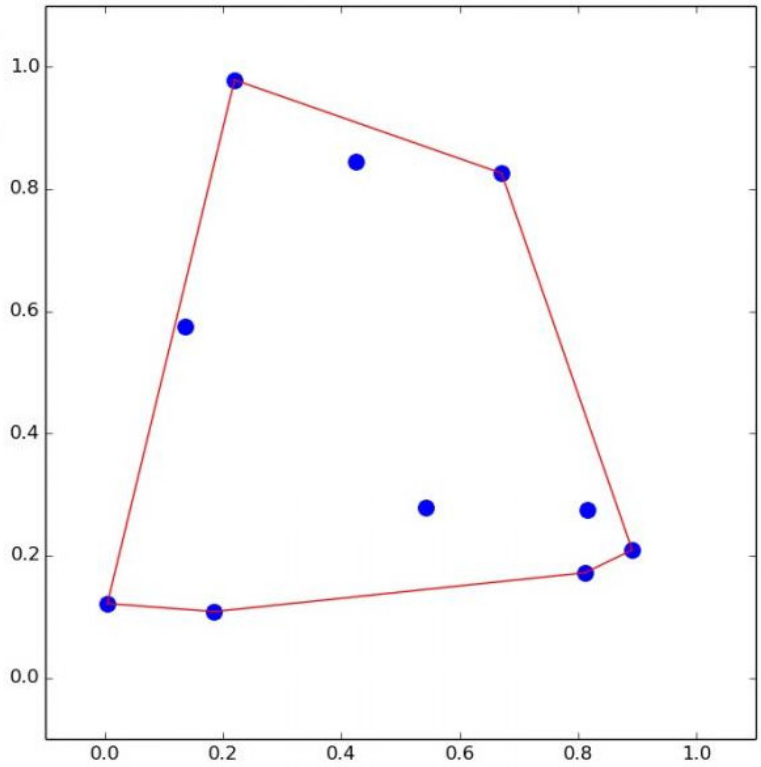
### 2.2.4 Convex Hull

O algoritmo utilizado na operação *Convex Hull*, também chamada de polígono ou envelope convexo, cria o menor polígono que engloba um determinado conjunto de entidades espaciais. Opera somente com uma única camada de entrada por vez. Esta geometria inicial pode ser de qualquer tipo. Este tipo de geometria gerada é útil quando se deseja determinar a área

mínima onde estão inseridos determinados elementos em sua região de estudo (MELKMAN, 1987).

O problema do *Convex Hull* se encaixa na área da geometria computacional. Dado um conjunto de pontos em um espaço, o problema consiste em encontrar o menor número de pontos que gerem um polígono convexo que abrange todos os outros pontos, como ilustrado na Figura 18. A imagem abaixo é um exemplo do *Convex Hull*. Dos vários pontos, somente os pontos mais externos e que formam o menor polígono que "fecham" todos os outros pontos (AMARO, 2023).

Figura 18 – Exemplo de *Convex Hull*




Fonte: Amaro (2023).

O *Convex Hull* pode ser expandido para 3 ou mais dimensões. Portanto, não se limita ao plano. Além disso, ele pode ser aplicado em diversos problemas: *ray tracing*, *videogames*, busca de caminhos em robótica, sistemas de informação geográfica, entre vários outros (SKLANSKY, 1982).

CAPÍTULO

# 3

# TRABALHOS CORRELATOS

O uso de visão computacional para as soluções dos mais diversos problemas é bastante amplo. Para este trabalho, foi feito um estudo de alguns trabalhos correlatos que aplicaram a visão computacional para solucionar problemas relacionados ao tráfego de veículos. O objetivo é fornecer ao leitor uma compreensão mais ampla do contexto e das contribuições deste trabalho em relação aos estudos anteriores. Serão descritos alguns trabalhos recentes que abordam o mesmo tema ou problemas semelhantes, apresentando seus objetivos, métodos e resultados. Ao final, é feita uma comparação com a proposta do presente trabalho, ressaltando as diferenças e contribuições.

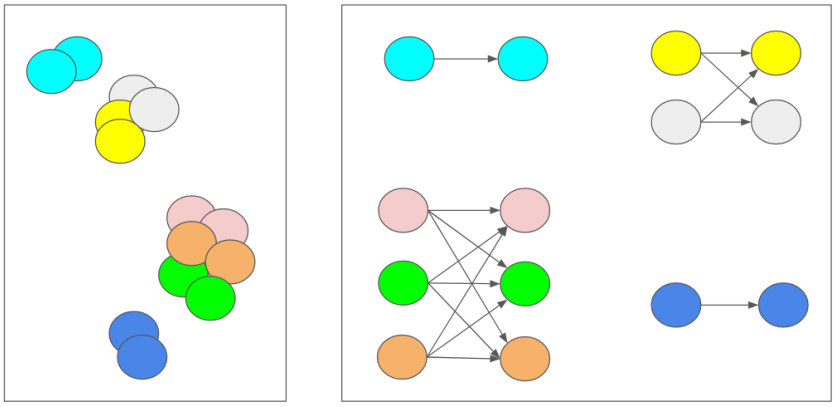
No trabalho de Bafghi e Shoushtarian (2020), o objetivo é apresentar um sistema de rastreamento de vários veículos em uma rodovia usando modelos de aparência e rastreamento visual. O método consiste em três etapas: detecção, modelagem de aparência e rastreamento de veículos. Na primeira etapa, o algoritmo de detecção *Mask Region-based Convolutional Neural Network (Mask R-CNN)* [1] (HE *et al.*, 2017) identifica a presença de veículos em cada *frame* do vídeo. Na segunda etapa, o modelo de aparência *SIFT* (LOWE, 2004) e histogramas de cores são criados com base em suas características visuais a partir de uma imagem de referência de cada veículo. Na terceira etapa, os modelos de aparência são utilizados para identificar e rastrear os veículos detectados em várias imagens subsequentes ao longo do tempo. Para isso, são empregadas técnicas de detecção de borda e correspondência de características visuais, tais como os grafos bipartidos. Afim de obter uma maior precisão na estimativa das posições dos objetos, foi utilizado um modelo de movimento. Com esses dois modelos, de aparência e movimento, os pesos das arestas são encontrados em combinação linear desses dois modelos. O método é avaliado no *dataset UA-DETRAC* [2] e mostra resultados promissores em termos de precisão e eficiência na detecção e rastreamento de múltiplos veículos em movimento na rodovia.

[1] Mais informação em : <https://openaccess.thecvf.com/content_ICCV_2017/papers/He_Mask_R-CNN_ICCV_2017_paper.pdf>

[2] Mais informação em : <https://detrac-db.rit.albany.edu/>

Na Figura 19, é ilustrado o processo de poda de nós utilizando a métrica *IOU*. A poda é uma técnica que consiste em agrupar nós vizinhos em dois *frames* consecutivos, com o objetivo de evitar cálculos desnecessários (NOWOZIN, 2014). Após a poda, o grafo é dividido em componentes desconexos, resultando em um grafo esparsa. Isso contribui para minimizar os custos computacionais envolvidos no processo. Na parte esquerda da imagem, podemos observar os nós vizinhos em dois *frames* consecutivos, enquanto na parte direita da imagem, ocorre a comparação dos *bounding boxes* dos nós usando a métrica *IOU* juntamente com o extrator de características *SIFT*. Essa combinação de métricas permite determinar o peso das comparações entre os nós e selecionar as correspondências com a maior semelhança para realizar o *matching* (BAFGHI; SHOUSHTARIAN, 2020).

Figura 19 – Processo de poda em nós vizinhos



Fonte: Bafghi e Shoushtarian (2020).

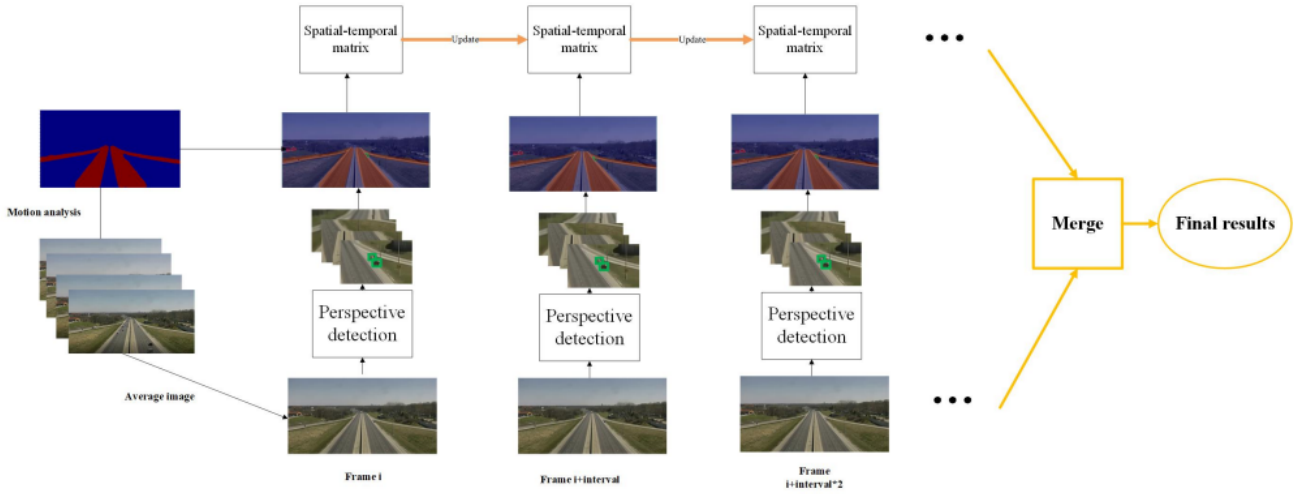
No trabalho de Bai *et al.* (2019) é proposto um sistema de detecção de anomalias que inclui três módulos: Módulo de modelagem de fundo, Módulo de detecção de perspectiva e Módulo de discriminação de matriz espaço-temporal. A modelagem de fundo analisa o fluxo de tráfego para obter os resultados de segmentação não supervisionada da estrada, com base na análise do fluxo de tráfego, que elimina a interferência de veículos fora da estrada. O modelo de perspectiva de detecção obtém o mapa de perspectiva pelo primeiro resultado de detecção, que é feito pelo modelo *Faster R CNN Faster R CNN* [3], e em conjunto a imagem é cortada em escala uniforme para diferentes veículos e re-detecção (FAN; BROWN; SMITH, 2016). Finalmente, são obtidas todas as anomalias construindo uma matriz de informação espacial temporal com os resultados da detecção. Além disso, todas as anomalias são combinadas através do *NMS* e do modelo de re-identificação, incluindo dimensões espaciais e temporais.

A Figura 20 apresenta a arquitetura da estrutura de detecção de anomalias. Considerando que a obtenção de dados normais é mais fácil, o principal objetivo deste artigo é contribuir para a detecção semi-supervisionada de comportamentos anômalos. Para garantir a identificação

[3] Mais informação em : <https://openaccess.thecvf.com/content_iccv_2015/papers/Girshick_Fast_R-CNN_ICCV_2015_paper.pdf>

de todos os veículos no vídeo, é adotado um método de detecção de perspectiva. Essa abordagem visa normalizar objetos próximos e distantes, reduzindo a amplitude das variações e, assim, assegurando uma alta taxa de *recall* para objetos menores à distância. As informações relacionadas à posição e ao tempo dos veículos parados são obtidas por meio do módulo de detecção de perspectiva. É importante ressaltar que nem todos os veículos parados detectados são necessariamente anomalias. Para determinar se um veículo parado representa uma anomalia, são analisadas as informações relacionadas à dinâmica do veículo. Esse processo de análise permite distinguir comportamentos anômalos daqueles que são considerados normais, tornando a detecção de anomalias mais precisa e eficaz.

Figura 20 – Arquitetura da estrutura de detecção de anomalias




Fonte: Bai *et al.* (2019).

No trabalho de Li *et al.* (2020), foi desenvolvido um método de detecção de anomalias de tráfego com base na detecção e rastreamento de veículos. Para a detecção de veículos foi usado o algoritmo *Faster R-CNN* e para o rastreamento foi usado o algoritmo *DeepSORT* [4] para obter a trajetória do veículo. O *DeepSORT* é uma estrutura de rastreamento de objetos e é uma extensão do *Simple online and realtime tracking (SORT)* (HOU; WANG; CHAU, 2019). Com os resultados da detecção e rastreamento foi apresentado o modelo *Mixture of Gaussians 2 (MOG2)* [5], baseado em um modelo de mistura gaussiana , que tem por objetivo remover veículos em movimento e somente analisar os veículos parados (LI *et al.*, 2020). Depois é implementado um novo mecanismo de extração de máscara baseado na diferença de *frames* e na trajetória de rastreamento do veículo para remover as vias secundarias, como estacionamentos e assim evitar falsas detecções. Em seguida é usada uma estrutura de rastreamento de multi granularidade que contém uma ramificação de rastreamento em nível de *box* e uma ramificação de rastreamento em nível de *pixel*. Cada ramificação contribui para capturar abstrações anormais em diferentes

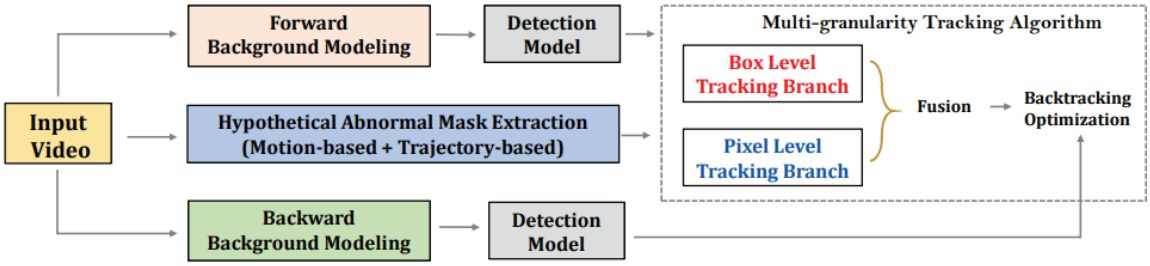
[4] Mais informação em : <https://www.researchgate.net/publication/368646461_Object_Tracking_with_DeepSORT>

[5] Mais informação em: <https://link.springer.com/article/10.1007/s10044-018-0699-y>

níveis de granularidade para modelar conceitos anormais. Finalmente é proposto um método de otimização de fusão de retrocesso e anomalia para refinar as previsões anormais, que pode melhorar significativamente a robustez e a precisão dos resultados.

A Figura 21 ilustra o rastreamento de múltipla granularidade com uma estrutura de componentes modularizados. Essa estrutura abrange a fusão das abordagens de rastreamento em nível de *box* e em nível de *pixel*. Além disso, para aprimorar ainda mais as previsões, são aplicadas técnicas de otimização de retrocesso. Nesse contexto, a combinação de rastreamento em diferentes níveis de granularidade, ou seja, tanto em termos de *bounding boxes* quanto de *pixels*, permite obter uma compreensão mais abrangente do cenário de rastreamento. A otimização de retrocesso aprimora a precisão das previsões, tornando esse sistema de rastreamento mais robusto e eficaz na identificação e rastreamento de objetos em diversas situações. Essa abordagem modularizada e multifacetada é essencial para lidar com desafios complexos de rastreamento em cenários variados.

Figura 21 – Esquema do artigo.

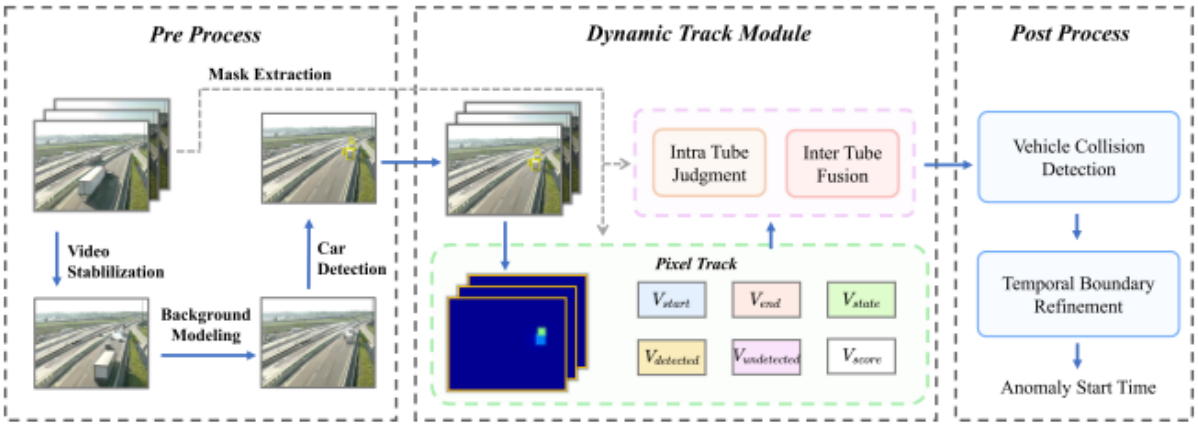



Fonte: Li *et al.* (2020).

No trabalho de Zhao (2021) é proposto uma estrutura simples e eficiente que inclui três etapas: o pré-processamento, um módulo de pista dinâmico e o pós-processamento, conforme apresentado na Figura 22. A etapa de pré-processamento visa gerar anomalias candidatas e é composto de quatro partes: estabilização de vídeo, modelagem de fundo, detecção de veículos e geração de máscara. A estabilização de vídeo tem por objetivo corrigir as oscilações de movimento da câmera que ocorrem no processo de aquisição, através de técnicas de software, como correspondência de pontos com base na *Good Features to Track* (GFTT) (SHI *et al.*, 1994) e é calculado um fluxo óptico esparso para gerar transformações *frame* a *frame*, e assim melhorar a qualidade visual e as aplicações finais, tais como detecção e rastreamento de veículos. Na modelagem de fundo é usada a abordagem de subtração de fundo baseado na *MOG*. O objetivo é comparar a diferença entre os resultados de subtração de fundo para frente e para trás. Os resultados de subtração de fundo para trás tornarão os veículos parados mais claros enquanto a subtração de fundo para frente auxilia na obtenção da hora de início mais precisa da anomalia. Para a detecção de veículos é empregada o *Faster R-CNN*. Na geração de máscara é preciso filtrar os veículos estáticos nas estradas secundárias e estacionamentos para detetar anomalias nas vias primárias e para isso será preciso segmentar as regiões hipotéticas anômalas da máscara, usando

um método de extração de máscaras baseado no movimento e máscara baseada em trajetória. A etapa de rastreamento dinâmico procura e localiza o tempo de início das anomalias, utilizando padrões de movimento do veículo. Finalmente, é utilizado o pós-processamento para afinar o limite temporal das anomalias.

Figura 22 – Ilustração da estrutura de estratégia hierárquica.




Fonte: Zhao (2021).

Os trabalhos descritos acima servem de influência direta para a criação da proposta deste trabalho. A Tabela 1 apresenta as principais comparações com os trabalhos citados. Cada um contribuindo com ideias que podem ser aplicadas para a detecção e rastreamento de objetos e detecção de anomalias onde nossa proposta de trabalho se concentra.

O trabalho de Bafghi e Shoushtarian (2020) apresentou dois métodos distintos para a extração de características de objetos. O primeiro método utilizou o *SIFT* e características de histogramas de cores em cada imagem para avaliar as similaridades entre os objetos dos *frames* vizinhos. Por outro lado, o segundo método empregou características profundas obtidas a partir da rede de detecção de objetos *Mask R-CNN* para atingir o mesmo objetivo. Neste trabalho, tomamos como referência os grafos bipartidos como um guia para o rastreamento de objetos, mas escolhemos o *CLIP* e a *SSMI* para determinar os pesos das conexões. Diferentemente do *SIFT*, que requer histogramas de cores para aprimorar as características dos objetos, o *CLIP* é capaz de extrair características de forma eficaz em diversos cenários, pois foi projetado para compreender e correlacionar informações visuais e linguísticas, permitindo que máquinas processem dados visuais e de texto de maneira conjunta. Essa capacidade de associação entre texto e imagens o torna uma ferramenta valiosa em diversas aplicações. Além disso, a *SSMI* desempenha um papel importante ao comparar e avaliar a qualidade das imagens resultantes de diferentes processos e transformações. Ela oferece uma métrica objetiva para medir a semelhança estrutural entre imagens originais e suas versões modificadas.

O trabalho de Li *et al.* (2020) apresentou uma solução para detectar anomalias de tráfego, utilizando métodos de visão computacional, como a subtração de fundo, segmentação de imagens e componentes modularizados para rastrear veículos em níveis de *box* e *pixels*.

Tabela 1 – Comparativo entre os trabalhos correlatos com o presente trabalho.

| Trabalho | *Dataset* | Técnica de detecção | Técnica de Rastreamento | Técnica de anomalias |
|---|---|---|---|---|
| Bafghi e Shoushtarian (2020) | *UA-DETRAC* | *Mask R-CNN* | Modelo de aparência e rastreamento visual de objetos | |
| Li *et al.* (2020) | *Track 4* | *Faster R-CNN* | | Modelo de rastreamento em nível de *box* e *pixel* |
| Bai *et al.* (2019) | *Track 3* | *Faster R-CNN* | | Modelo de detecção de relacionamento de perspectiva |
| Zhao (2021) | *Track 4* | *Faster R-CNN* | | Modelo de rastreamento dinâmico |
| Este trabalho | *UA-DETRAC e Track 4* | *YoloV7* | Rastreamento de veículos usando áreas de interesse | Detecção de Anomalias em Áreas de Interesse Espaço-Temporais |

Fonte: Elaborada pelo autor.

Neste trabalho, usamos a ideia da subtração de fundo e segmentação para remover veículos em vias secundárias. No entanto, optamos por utilizar o algoritmo *Convex Hull* para gerar áreas tanto de movimento quanto de ausência de movimento. Escolhemos as áreas com movimento para analisar o comportamento dos veículos e identificar possíveis veículos parados nessas vias. Uma distinção importante entre este trabalho e o anterior é que, no nosso método, as áreas de movimento são geradas durante o processo de rastreamento. No trabalho anterior, a subtração de fundo e a segmentação eram realizadas separadamente em dois processos distintos, o que resultava em um maior custo computacional.

No trabalho de Bai *et al.* (2019), foi apresentada uma solução para a detecção de anomalias de tráfego, analisando os eventos de anomalia com base na informação dinâmica

dos veículos. Especificamente, eles utilizaram um total de seis matrizes de informação espaço-temporais para identificar o horário de início e término da anomalia detectada. Essas informações são relacionadas ao nível de *pixel* e estavam associadas ao tempo. No nosso trabalho, adotamos a ideia de incorporar informações relacionadas ao tempo para determinar o período em que um veículo permanece na cena. Para isso, monitoramos se o veículo ainda está presente na cena ou se saiu dela. Se o veículo estiver na cena, calculamos sua velocidade para determinar se ele está parado ou em movimento. Quando a velocidade é baixa, interpretamos que o veículo está parado e, nesse caso, criamos posições hipotéticas para esse objeto, com o objetivo de não perder informações importantes sobre ele, considerando-o como um objeto com possível anomalia. Uma distinção entre o nosso método e o método anterior é que nosso processo incorpora informações temporais usando uma abordagem unificada. Em contrapartida, o método anterior utiliza a modelagem de fundo e a detecção de perspectiva como processos separados para obter informações sobre o início e o fim das possíveis anomalias. Nosso método oferece uma abordagem mais integrada, utilizando informações das trajetórias dos veículos para identificar o início e o fim das possíveis anomalias, resultando em uma análise mais completa e eficiente do comportamento dos veículos em um único processo.

O trabalho de Zhao (2021) apresenta um processo composto por três etapas com o objetivo de detecção de anomalias de tráfego: pré-processamento, rastreamento dinâmico e pós-processamento. No nosso trabalho, nos inspiramos na etapa de pré-processamento, uma vez que enfrentamos desafios relacionados a ruídos nos dados, como instabilidade na câmera e variações na iluminação. No entanto, optamos por abordar a detecção de anomalias de forma diferente, utilizando o rastreamento de objetos em conjunto com uma estrutura de dados conhecida como *QuadTree*, além de uma abordagem temporal. A *QuadTree* é empregada para comparar as posições e características de objetos próximos, enquanto a estrutura temporal analisa se esses objetos se enquadram na categoria de anomalias. Essa abordagem torna nosso método robusto na detecção de anomalias em comparação com o outro método, que requer um pós-processamento adicional para ajustar os limites temporais das anomalias. Em suma, nossa estratégia simplifica o processo de detecção e aprimora a eficácia na identificação de anomalias de tráfego.

Portanto, os trabalhos mencionados acima são de guia principal para este projeto. Foram desenvolvidos a partir de diferentes perspectivas, enriquecendo significativamente o conhecimento necessário para a definição do método adotado nesta pesquisa.

CAPÍTULO

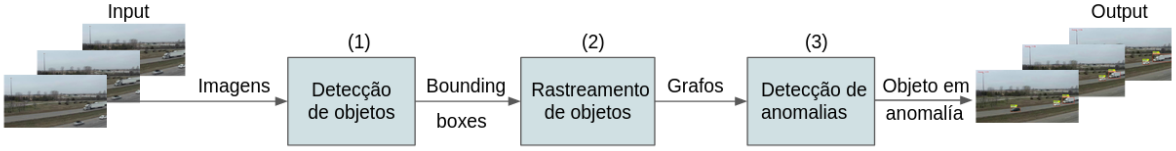
# 4

# METODOLOGIA

Neste capítulo, apresentamos a metodologia utilizada neste trabalho. Descreveremos em detalhes as técnicas que serão empregadas para a detecção e rastreamento de objetos e detecção de anomalias. Além disso, abordaremos os recursos e ferramentas que serão utilizados para alcançar nossos objetivos, oferecendo uma visão completa do processo de desenvolvimento.

Esse processo foi subdividido em três etapas, conforme ilustrado na Figura 23. Na fase inicial, o foco é direcionado para a detecção de objetos (etapa 1), para o qual se utiliza um *dataset* composto por um conjunto de vídeos de tráfego rodoviário. As saídas geradas por essa etapa de detecção servem como entradas para a segunda etapa, que consiste no rastreamento de objetos (etapa 2), que tem como objetivo atribuir um identificador único a cada veículo detectado, viabilizando, assim, o acompanhamento de suas trajetórias. Por fim, adentramos à última etapa, onde efetuamos uma análise de anomalias com base nas trajetórias rastreadas (etapa 3). No contexto deste trabalho, define-se uma anomalia como a situação em que um veículo permanece imóvel por mais de um minuto em uma via principal. Vale destacar que uma via principal é uma rota viária de maior movimentação, projetada para a circulação regular de veículos.

Figura 23 – Fluxograma resumo deste trabalho.




Fonte: Elaborada pelo autor.

Nosso trabalho tem como nome: Mecanismo de Detecção de Anomalias de Veículos no Tráfego - **MEDAVET**. O **MEDAVET** é baseado em detecção e rastreamento de veículos em vias urbanas, visando detectar anomalias no tráfego da rodovia. Nossa pesquisa visa melhorar signifi-

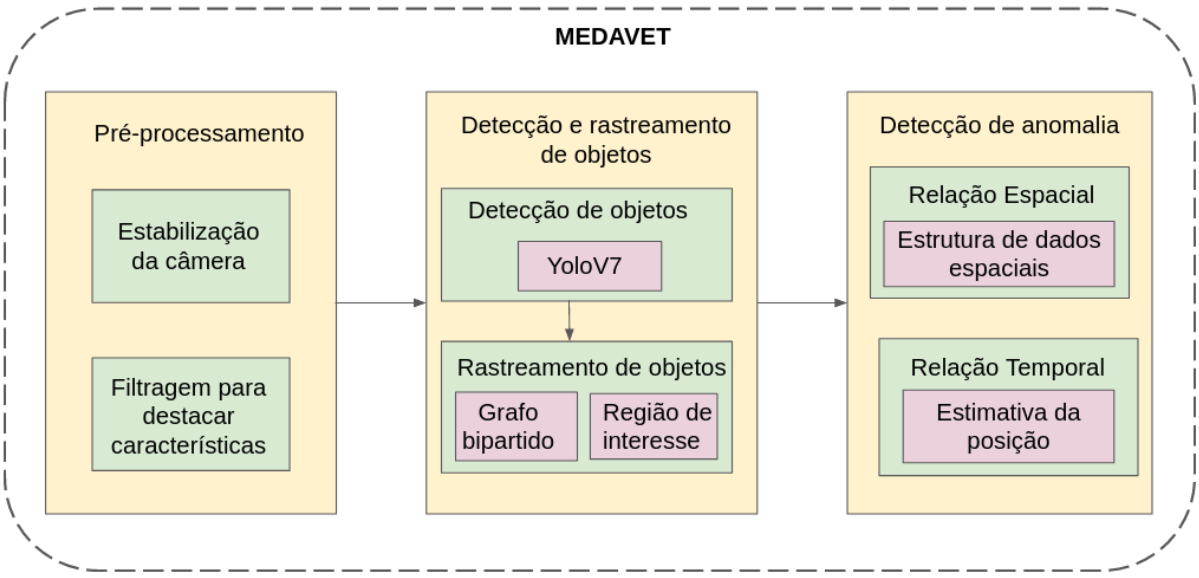
cativamente a segurança rodoviária e a eficiência do tráfego, identificando situações anômalas que podem representar riscos, como congestionamentos prolongados ou incidentes que exigem intervenção rápida. Portanto, nesta pesquisa consideramos que ao longo das rodovias existem câmeras que realizam o seu monitoramento e que as informações captadas são enviadas a uma central a qual realiza o processamento, conforme ilustrado na Figura 24. Assim o **MEDAVET** é executado na central de controle permitindo que um operador possa tomar uma ação em tempo hábil, como acionar uma viatura para verificar o que está acontecendo com o condutor.

Figura 24 – Cenário de aplicação do MEDAVET.

Fonte: Dados da pesquisa.

Neste estudo, implementamos um sistema de monitoramento de veículos em vídeos, que compreende um processo de pré-processamento de vídeo para estabilização de câmera e realce de características visuais. Em seguida utilizamos os *frames* pré-processados para a detecção e rastreamento dos veículos (Figura 25). Para isso, é utilizada uma ferramenta para detecção de objeto que irá detectar os veículos nos *frames* capturados. Portanto um objeto nesse trabalho seria um veículo na imagem. Após a detecção dos objetos utilizamos a teoria de grafos para criar uma estrutura para a representação e análise da trajetória dos veículos, permitindo o rastreamento dos veículos ao longo do tempo. Com base no grafo de saída no componente de detecção e rastreamento de veículos, realizamos a detecção de anomalia, que verificará o tempo que o veículo estará sem mobilidade e o local em que o veículo está parado, para realizar a inferência, se é um comportamento normal ou não. Nas próximas subseções descreveremos mais detalhamento de cada componente do **MEDAVET**.

Figura 25 – Visão Geral do MEDAVET.




Fonte: Elaborada pelo autor.

## 4.1 *Datasets*

Para este trabalho, estamos utilizando dois *datasets* como entrada. O primeiro é o *UA-DETRAC* (WEN *et al.*, 2015), o qual será empregado na etapa de rastreamento. O segundo é o *Track 4* do *NVIDIA AI CITY CHALLENGE 2021* (NAPHADE *et al.*, 2020), que será aplicado para a detecção de anomalias.

### *4.1.1 UA-DETRAC*

O *dataset UA-DETRAC* é um *benchmark* desafiador do mundo real usado para a avaliação de algoritmos de detecção e rastreamento de veículos, composta por 10 horas de vídeos, gravados nas ruas da China, como é ilustado na Figura 26. Este *dataset* inclui um total de 140.000 *frames*, registrados a uma taxa de 25 *frames* por segundo *(fps)* e com resolução de $960 \times 540$ *pixels*. Esse conjunto de dados abrange a presença de 8.250 veículos, distribuídos ao longo de todos os *frames*.

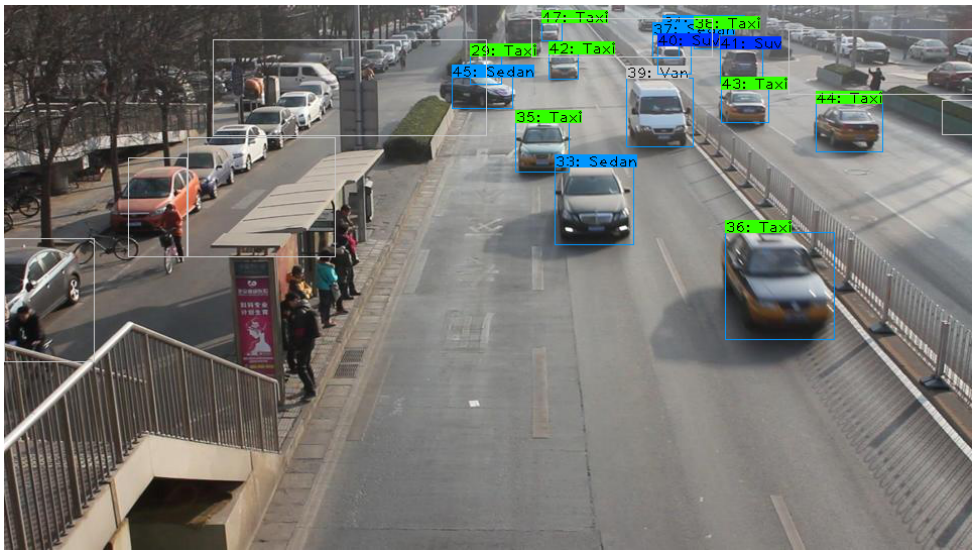
Para realizar o rastreamento neste *dataset*, apenas as áreas de interesse que contêm veículos são consideradas. No entanto, é importante observar que nas áreas ignoradas também podem existir veículos, tanto parados quanto em movimento. Esses veículos não são considerados para o cálculo das métricas de rastreamento, como pode se observar na Figura 27. Essa abordagem se baseia na suposição de que as áreas de interesse são as principais regiões onde a movimentação dos veículos é relevante para o contexto de rastreamento. As áreas ignoradas podem incluir espaços onde os veículos não são de interesse imediato, como áreas de estacionamento ou locais

Figura 26 – Imagens do *dataset UA-DETRAC*.

Fonte: Wen *et al.* (2015).

onde os veículos estão parados por um longo período. A decisão de ignorar essas áreas tem como objetivo otimizar o cálculo das métricas de rastreamento, focando nos veículos que são mais relevantes para a análise. No entanto, é importante ter em mente que, esses veículos em áreas ignoradas não contribuam diretamente para as métricas.

Figura 27 – Áreas de interesse nas anotações do *dataset*




Fonte: Dados da pesquisa.

Enfrentamos vários desafios ao lidar com o *dataset*. Um exemplo notável é a instabilidade na câmera, frequentemente causada por condições climáticas adversas como, por exemplo, o vento. Essa instabilidade resulta em movimentos indesejados da câmera o que, por sua vez, faz com que objetos estáticos pareçam estar em movimento. Essa situação compromete a precisão da detecção e introduz complicações adicionais no processo. Além disso, há outros desafios igualmente relevantes que devemos considerar. Ocorrências como oclusões, variações de escala e de iluminação, densidade de tráfego e flutuações sazonais também influenciam de maneira significativa. Esses fatores, somados, contribuem para que a detecção e rastreamento se tornem uma tarefa verdadeiramente complexa.

Para este *dataset*, não é necessário realizar pré-processamento adicional, uma vez que nosso método implementado provou ser suficiente para lidar com os desafios apresentados. É importante ressaltar que este *dataset* é utilizado exclusivamente na fase de rastreamento, com o objetivo de avaliar o desempenho do nosso algoritmo em comparação com outros métodos (ver Capítulo 5). Posteriormente, nosso algoritmo será empregado na etapa de detecção de anomalias, utilizando o *dataset Track 4*.

### 4.1.2 Track 4

No entanto o *Track 4* é um *dataset* de trânsito de alta densidade. O *dataset* é dividido em conjunto de treinamento e conjunto de testes. O conjunto de treinamento contem 100 vídeos e o conjunto de testes tem 150 vídeos. Cada vídeo tem duração de 15 minutos aproximadamente. Os vídeos são rastreados a 30 *frames* por segundo *(fps)* e tem resolução de 800 $\times$ 410 *pixels*. Os vídeos foram gravados na China em diferentes cenários, como pode se observar na Figura 28

Figura 28 – Imagens do *dataset Track 4*.

Fonte: Naphade *et al.* (2022).

Este *dataset* é usado para detectar anomalias nas vias principais, mas encontramos alguns desafios. Um desafio foi a ausência de dados do detector em alguns *frames*, ou seja, a falta de geração de arquivos com as coordenadas desses *frames*. Para enfrentar essa situação, realizamos ajustes em nosso algoritmo. Quando os arquivos de coordenadas não foram encontrados para determinados *frames*, eles foram excluídos do processo de análise. No entanto, para garantir que

não perdêssemos informações sobre os objetos durante o rastreamento, implementamos uma técnica alternativa. Utilizamos a análise das distâncias entre os centros dos objetos nos *frames* que não dispunham das informações de coordenadas, ou seja, estabelecemos um limiar de distância com base nos *frames* consecutivos e multiplicamos esse limiar pela quantidade de *frames* para os quais os arquivos não foram gerados. Dessa forma, conseguimos extrair características dos objetos com base nas variações de suas posições ao longo do tempo. Essa abordagem nos permitiu continuar a análise mesmo quando os arquivos de dados não estavam disponíveis devido a limitações do detector ou mesmo não tendo veículos na cena na detecção de objetos. Enquanto nos *frames* não enfrentamos problemas, já que eles eram processados sequencialmente, a geração dos arquivos nem sempre era garantida devido às limitações do detector. Essa técnica foi fundamental para garantir que as informações nos arquivos de coordenadas coincidissem com os *frames* analisados, mesmo em cenários de detecção incompleta.

Outro desafio que enfrentamos foi a presença de ruídos, cenas noturnas e até mesmo a ocorrência de *frames* duplicados devido a erros no processamento, o que cria a ilusão de que os veículos estão parados por longos períodos de tempo, como ilustrado na Figura 29. Para garantir o funcionamento eficaz de nosso algoritmo, optamos por implementar uma estratégia de remoção dessas imagens indesejadas, visando a melhoria dos resultados obtidos. Isso nos permitiu aprimorar a qualidade dos dados de entrada e, consequentemente, obter resultados mais confiáveis em nossa análise de detecção de anomalias.

Figura 29 – Ruídos e cenas noturnas.

Fonte: Dados da pesquisa.

Além dos desafios mencionados anteriormente, enfrentamos outros, como o movimento intenso da câmera e a presença de ruídos nas imagens, mesmo durante o dia. Isso ocorreu em parte devido à gravação de alguns vídeos a longa distância, especialmente em condições chuvosas, o que resultou em artefatos de *aliasing* indesejados. A solução para abordar esses problemas envolveu a aplicação de técnicas de estabilização de imagens e a utilização de filtros para aprimorar as características dos veículos. Os detalhes sobre essas técnicas adotadas para lidar com esses desafios estão na Seção 4.2.

## 4.2 Pré-Processamento

O processo de captura de imagens em alguns vídeos do *dataset Track4* frequentemente resultam em ruídos indesejados, que se manifestam na forma de *frames* não detectados, movimentação indesejada da câmera, surgimento de artefatos, entre outros problemas, como ilustrado na Figura 30. Esses ruídos podem prejudicar o desempenho adequado dos veículos, causando interferências no processo de detecção e rastreamento. Isso pode levar a múltiplas detecções e *IDs* atribuídos aos veículos, comprometendo a precisão e a consistência dos resultados. Portanto, é fundamental realizar um processo de pré-processamento para mitigar essas questões. Ao analisar nossos dados, identificamos a necessidade de aplicar técnicas de estabilização de câmera e filtragem de ruído, com o objetivo de realçar as características relevantes nas imagens. É importante ressaltar que essas técnicas serão aplicadas exclusivamente ao *dataset Track 4*, visando melhorar sua usabilidade e confiabilidade para aplicações de detecção e rastreamento de veículos.

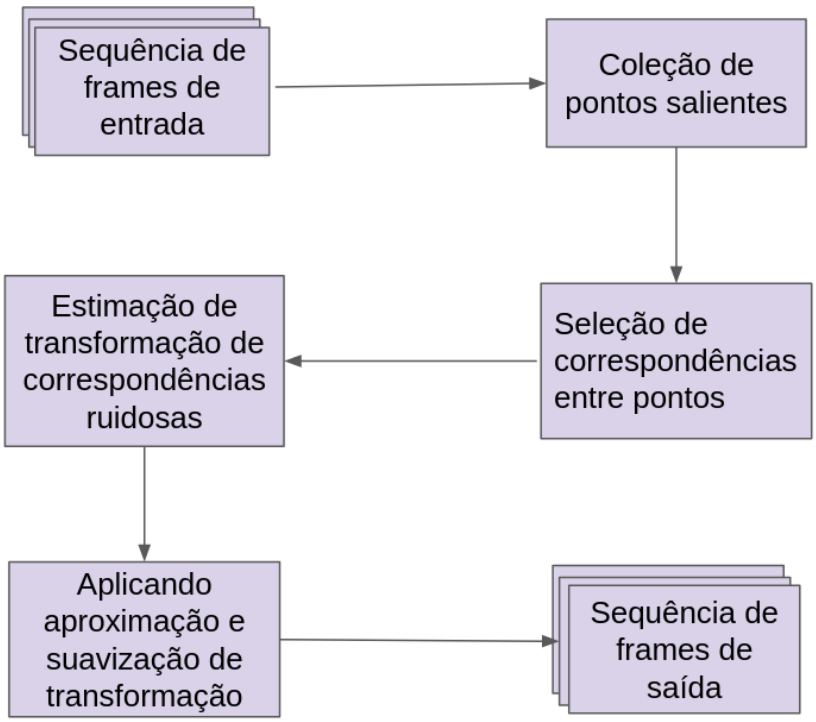
Figura 30 – Imagens sem pré-processamento.


Fonte: Dados da pesquisa.

A estabilização da câmera é um processo importante para melhorar a qualidade visual de vídeos, reduzindo os efeitos indesejados do movimento da câmera. Nesse método, o algoritmo *goodFeaturesToTrack* é empregado para detectar e selecionar pontos característicos de destaque em um *frame*, que servirão como referências para o rastreamento subsequente, como ilustra a Figura 31. O algoritmo *Lucas-Kanade Optical Flow*, por sua vez, é utilizado para rastrear esses pontos característicos nos *frames* subsequentes, permitindo a determinação precisa do deslocamento desses pontos ao longo do tempo. Com base nesses deslocamentos, é possível calcular a transformação rígida (euclidiana), que engloba informações de translação, rotação e escala, para corrigir o movimento da câmera em cada *frame*, resultando em um vídeo final mais suave e estável.

Para filtrar o ruído na trajetória de movimento, aproveitamos a estimativa de movimento

Figura 31 – Diagrama de bloco da estabilização da câmera.




Fonte: Elaborada pelo autor.

entre os *frames* realizada na etapa anterior. Nessa etapa, buscamos determinar a trajetória de movimento acumulando incrementalmente o movimento diferencial estimado entre os *frames* consecutivos. Isso significa somar o movimento entre os *frames* para calcular a trajetória global. O objetivo final é suavizar essa trajetória para torná-la mais estável. Para essa suavização, utilizamos um filtro de média móvel que, como o nome sugere, substitui o valor de uma função em um ponto pela média dos valores de seus vizinhos. Aplicamos essa trajetória suavizada para obter transformações de movimento mais suaves que podem ser aplicadas aos *frames* de vídeo para estabilizá-los. Isso é alcançado encontrando a diferença entre a trajetória suavizada e a trajetória original e, em seguida, adicionando essa diferença às transformações originais. O processo envolve iterar pelos *frames* e aplicar essas transformações, resultando em um vídeo final estabilizado e livre de movimentos indesejados.

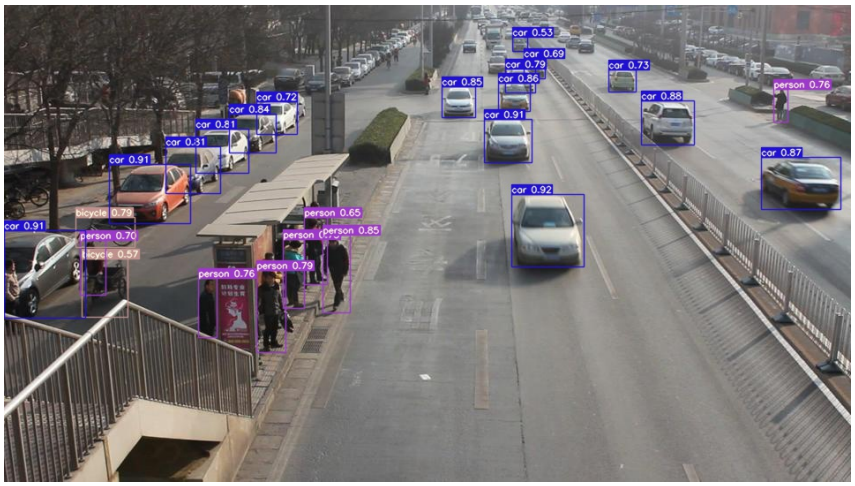
## 4.3 Detecção de objetos

A detecção de objetos desempenha um papel de grande importância na etapa de rastreamento, tornando muito importante a escolha de um detector de objetos de alta qualidade. Nesse contexto, as categorias de métodos existentes para a detecção se dividem principalmente em abordagens clássicas e extratores de objetos baseados em redes neurais profundas. Entre elas, as redes neurais profundas, exemplificadas pelo *Yolo*, emergem como a opção mais precisa e

confiável. A arquitetura *Yolo* se destaca por sua capacidade de detectar objetos com notável precisão em cada *frame*. O que diferencia o *Yolo* é sua abordagem de uma única passagem (ou seja, uma única iteração) na qual a rede neural analisa o *frame* de uma vez, detectando objetos e suas respectivas localizações. O *Yolo* é reconhecido por sua eficiência e rapidez, realizando a detecção em uma única etapa, tornando-o uma escolha eficaz para aplicações de detecção de objetos em tempo real, onde a agilidade e precisão são essenciais.

Nesta primeira etapa, optamos por utilizar a sétima versão do algoritmo *Yolo*, conhecido como *YoloV7*, para realizar a detecção de objetos nos vídeos presentes nos nossos *datasets*, como pode se ver na Figura 32. A escolha do *YoloV7* deve-se à sua implementação de um novo algoritmo de treinamento chamado *CrossEntropyLossWithLogits*, que se destaca por ser mais rápido e eficiente em comparação com os algoritmos utilizados nas versões anteriores do *Yolo*. Essa otimização resulta em um tempo de treinamento significativamente reduzido. Além disso, esta versão incorpora diferentes conjuntos de pesos, os quais foram treinados em um enorme conjunto de dados de imagens e são capazes de detectar uma ampla variedade de objetos, incluindo veículos. Para este projeto, optamos pelo uso do modelo *YoloV7-W6*, uma escolha que se mostrou altamente eficaz em nossa busca por uma detecção precisa de objetos em uma ampla gama de tamanhos, desde objetos pequenos até objetos em grande escala.

Figura 32 – Detecção de objetos usando *YoloV7* com o *dataset UA-DETRAC*.




Fonte: Dados da pesquisa.

Nesse contexto, decidimos focar em cinco categorias específicas dentre as 80 classes disponíveis no *Yolo*. As classes selecionadas correspondem aos números 1, 2, 3, 5 e 7. Essas categorias específicas representam, respectivamente, os objetos “bicicleta”, “carro”, “moto”, “ônibus” e “caminhão”. Ao delimitar nosso foco a essas cinco classes, o algoritmo *YoloV7* será treinado para realizar a detecção e classificação de objetos pertencentes a essas categorias nos vídeos dos *datasets*, como mostra a Figura 33. Esta abordagem visa otimizar o desempenho da detecção de objetos, permitindo que o sistema seja particularmente hábil detectando veículos e

meios de transporte terrestre, essenciais para diversas aplicações, desde sistemas de segurança até soluções de gerenciamento de tráfego.

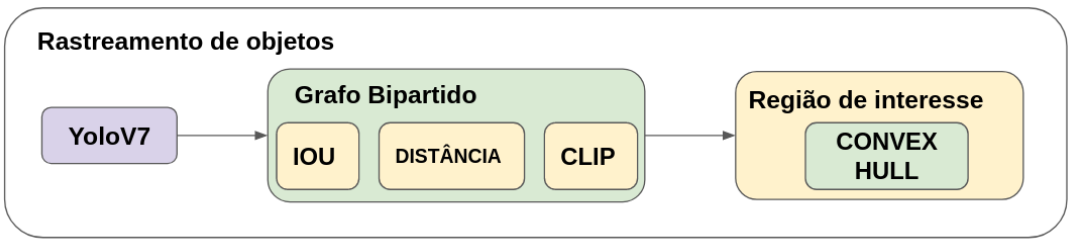
Figura 33 – Detecção de veículos usando *YoloV7* com o *dataset UA-DETRAC* com as classes selecionadas.


Fonte: Dados da pesquisa.

## 4.4 Rastreamento de objetos

Nesta etapa, realizamos o rastreamento utilizando os resultados da detecção de objetos. O propósito principal é atribuir um identificador único a cada objeto detectado, possibilitando assim, a obtenção de suas trajetórias estimadas. Começamos com a criação de grafos bipartidos para conectar as detecções correspondentes entre *frames* consecutivos, permitindo um rastreamento contínuo dos objetos em movimento. Para aprimorar ainda mais a eficiência, delimitamos uma área de interesse em torno dos veículos em movimento e para esse fim usamos o algoritmo *Convex Hull*, como pode se observar na Figura 34. Por fim, o sistema atribui *IDs* aos veículos rastreados e registra suas trajetórias ao longo do tempo. Essas trajetórias fornecem informações valiosas para a análise e compreensão do comportamento dos veículos no contexto do vídeo.

Figura 34 – Fluxograma do Rastreamento de objetos




Fonte: Elaborada pelo autor.

### 4.4.1 Grafos Bipartidos

Optamos por usar grafos bipartidos para rastreamento de objetos com base nas detecções do *Yolo*. A escolha do grafo bipartido se deve à sua capacidade de modelar relações claras entre objetos detectados e os *frames* correspondentes. Isso nos permite rastrear eficientemente os objetos ao longo do tempo e estabelecer conexões entre as detecções em diferentes *frames*.

Um grafo bipartido é um grafo $G = (V, E, \omega)$, onde $V$ representa o conjunto de nós, $E$ as arestas e $\omega$ os pesos entres os nós. De forma geral, o conjunto de nós é dividido em dois conjuntos disjuntos $V1$ e $V2$, de modo que cada aresta liga um nó de $V1$ a um nó de $V2$. Em outras palavras, é um grafo em que não há arestas que conectam dois nós do mesmo conjunto. Neste trabalho $V1$ e $V2$ representam dois *frames* consecutivos que tem como informação os objetos extraídos deles. Esses *frames* são as entradas para o grafo bipartido e os objetos extraídos deles representam os nós de cada um dos *frames*. As arestas são as conexões dos nós de cada *frame*. Como o grafo é bipartido não deve haver nenhuma conexão entre nós de um mesmo *frame* e além disso entre dois nós de dois *frames* consecutivos deve haver só uma aresta de conexão. Os pesos tem uma função muito importante em um grafo bipartido, eles determinam como vai ser a conexão das arestas entre os nós dos *frames* consecutivos.

Para cada vídeo do *dataset*, pegamos *frames* consecutivos e para cada dois *frames* consecutivos criamos um grafo bipartido, de forma que todos os nós no *frame* atual estejam conectados aos nós no *frames* seguinte. Então o conjunto $V$ é dividido em $n$ conjuntos disjuntos, cada um representando um *frame*, e os nós em cada conjunto representam os objetos, que são veículos neste trabalho. Para um melhor entendimento, vamos explicar em termos matemáticos: Seja

$$V = \{V_i \mid i \in \{1, \ldots, n\}, V_i \text{ disjuntos}\} \tag{4.1}$$

onde $V_i$ representa o $i$-ésimo *frame* do conjunto $V$. Então:

$$V_i = \left\{v_j^i \mid j \in \{1, \ldots, p\}\right\} \tag{4.2}$$

é o conjunto de nós do *frame* $i$, onde $v_j^i$ denota o $j$-ésimo nó. Tendo o conjunto de *frames* e o conjunto de nós, o conjunto de arestas pode ser definido como:

$$E = \{(v_j^i, v_k^{i+1}) \mid j \neq k\} \tag{4.3}$$

onde $v_j^i$ representa o $j$-ésimo nó do *frame* $i$ e $v_k^{i+1}$ representa o $k$-ésimo nó do *frame* $i+1$

Neste trabalho, nosso foco está nos veículos, e os pesos desempenham o papel de determinar as arestas apropriadas para conectar os nós de *frames* consecutivos. Os pesos têm

a função de garantir que as arestas conectem apenas veículos com alta similaridade, evitando a sobrecarga do grafo e, consequentemente, cálculos desnecessários. Nosso método consiste em empregar as métricas de *IOU* , a distância entre centros e a extração de características de similaridade. A métrica *IOU* é usada para quantificar o movimento dos veículos de um *frame* para outro. Embora o movimento dos veículos entre *frames* seja pequeno, ele varia dependendo da velocidade de cada veículo. Para capturar diferentes cenários, decidimos incorporar a distância entre os centros dos veículos, permitindo explorar variações nas velocidades de cada um.

Assim as métricas são definidas:

$$IOU(v_j^i, v_k^{i+1}) = \frac{Area((v_j^i) \cap Area(v_k^{i+1})}{Area((v_j^i) \cup Area(v_k^{i+1})} \geq \gamma \tag{4.4}$$

$$dist(v_j^i, v_k^{i+1}) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \leq \sigma \tag{4.5}$$

onde $(x_1, y_1)$ é a posição do centro de $v_j^i$ e $(x_2, y_2)$ é a posição do centro de $v_k^{i+1}$. Os valos $\gamma$ e $\sigma$ estão detalhados no Capítulo 5

Para determinar a similaridade entre os veículos, optamos por utilizar diversos métodos de extração de características para comparar qual se adapta melhor a nossos *datasets*. Entre os extratores temos o *CLIP*, a *SSMI* e o *SIFT*. Mais informação desses extratores no Capítulo 2

A métrica de similaridade que empregamos é definida da seguinte maneira:

$$similarity(v_j^i, v_k^{i+1}) \geq \theta \tag{4.6}$$

Os experimentos para o valor de $\theta$ adequado estão no Capítulo 5

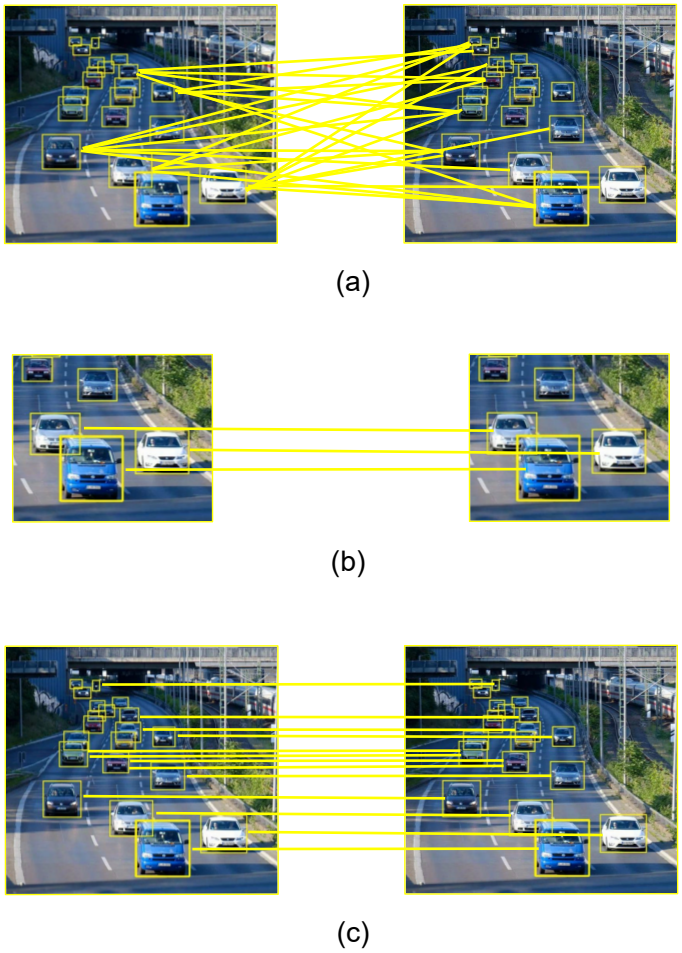
Para calcular os pesos das arestas fazemos a combinação linear da *IOU*, como mostra a Equação 4.4 e o extrator de características, como é mostrado na Equação 4.6, colocando dois parâmetros de medição para que os pesos estejam calibrados entre 0 e 1. (ver Capítulo 5)

$$\omega(v_j^i, v_k^{i+1}) = \alpha * IOU(v_j^i, v_k^{i+1}) + \beta * similarity(v_j^i, v_k^{i+1}) \tag{4.7}$$

Como os pesos podem associar mais de um nó no *frame* atual em relação ao *frame* seguinte, o *matching* desempenha uma tarefa importante. Em nosso contexto de rastreamento de objetos, a necessidade de associar de forma única cada objeto a um identificador específico é de extrema importância. Na Figura 35, ilustramos essa necessidade. Na parte (a) da figura, apresentamos um grafo bipartido no qual os nós do lado esquerdo têm conexões com mais de um nó do lado direito. Já na parte (b) e (c) os objetos da esquerda tem só uma conexão com um único objeto da direita, isso acontece porque usamos o algoritmo *min-weight-matching* de *matching* para a conexão única dos objetos. Isso garante que cada objeto seja rastreado de forma única e eficaz, respeitando o requisito de alocação exclusiva. Dessa forma, o *matching* em

grafos bipartidos se revela essencial em nosso trabalho, permitindo o rastreamento preciso de objetos e evitando associações indesejadas que poderiam comprometer a qualidade do processo de rastreamento.

Figura 35 – *Matching* em *frames* consecutivos.




Fonte: Dados da pesquisa.

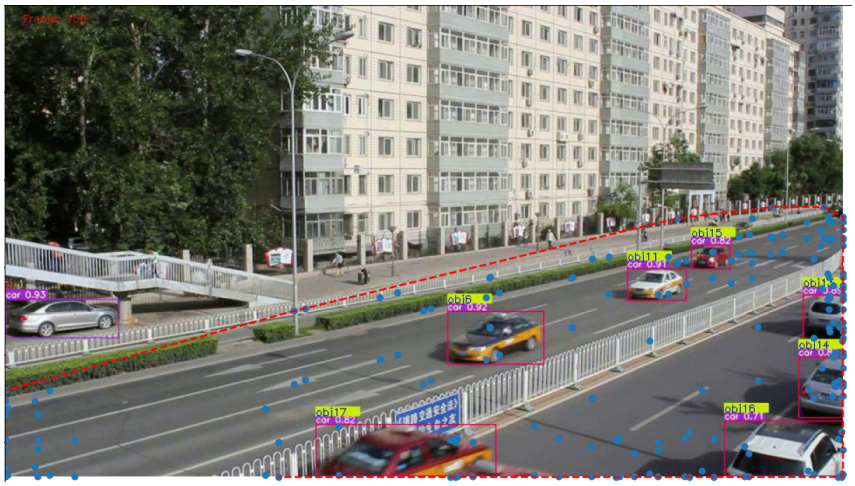
.

### 4.4.2 Região de Interesse

Dentro do contexto deste trabalho, estamos particularmente focados em rastrear áreas de interesse ao longo das vias onde ocorrem acidentes rodoviários, alguns dos quais podem ser causados pelo movimento dos veículos. Nossa área de interesse principal abrange essas áreas específicas de movimentação viária. O objetivo é monitorar continuamente o tráfego nessas regiões para capturar eventos como acidentes, incidentes ou comportamentos de condução que possam levar a situações perigosas. Para demarcar essa área de interesse na imagem, utilizamos o algoritmo *Convex Hull*. Esse algoritmo cria polígonos convexos que envolvem as áreas em movimento, ou seja, as vias da rodovia.

A criação desses polígonos nos permite definir claramente as regiões onde os veículos em movimento estão presentes, como ilustrado na Figura 36. Dessa forma, podemos excluir áreas que se encontram além do acostamento, como postos de gasolina e outros estabelecimentos adjacentes à rodovia, concentrando nossa análise nas áreas diretamente relacionadas ao fluxo de tráfego. Assim, evitar confusões com veículos que se encontram em vias secundárias. Essas vias são estradas que normalmente têm menor capacidade de tráfego e são destinadas a fins específicos, como acessos a estacionamentos.

Figura 36 – Veículos dentro da região definida via *Convex Hull*.



Fonte: Dados da pesquisa.

.

### 4.4.3 Funcionamento do rastreamento

É importante ressaltar que só os *frames* de vídeo do *dataset Track 4* passam por uma etapa de pré-processamento antes de serem passados ao algoritmo *Yolo*. Esse pré-processamento melhora a qualidade das imagens e a localização dos objetos. Após essa preparação, os *frames* são encaminhados para o algoritmo *Yolo*, que é responsável pela detecção dos objetos. A fase de rastreamento começa com a identificação dos objetos em cada *frame*, conforme mostrado no Algoritmo 1. Para cada par de *frames* consecutivos, é criado um grafo bipartido. Os nós de cada *frame* são analisados para verificar se atendem às condições definidas pelo grafo bipartido. Os nós que atendem a essas condições recebem um *ID* único, como ilustrado na linha 6 do Algoritmo 1. Caso contrário, são verificadas as condições de similaridade, *IOU* e distância (linhas 8 a 10). Se essas condições forem atendidas, as informações dos *frames* e dos vértices são atribuídas ao objeto correspondente da lista de objetos (linha 12). Caso contrário, um novo objeto é criado (linha 14). Os valores utilizados na linha 11 foram objetadas através de um conjunto de experimentos descritos na seção Capítulo 5.

Para minimização da quantidade de veículos a ser analisados usamos o algoritmo *Convex Hull* para cercar a área dos veículos em movimento e assim criamos uma nova lista de objetos

que estão na área de interesse (list_IA_obj). Assim se há um veículo e esse veículo está na área de interesse (Linhas 17 a 19) é atribuído um ID para esse veículo, excluindo os veículos ou outros objetos que estão fora dessa área.

**Algoritmo 1** – Rastreamento de veículos

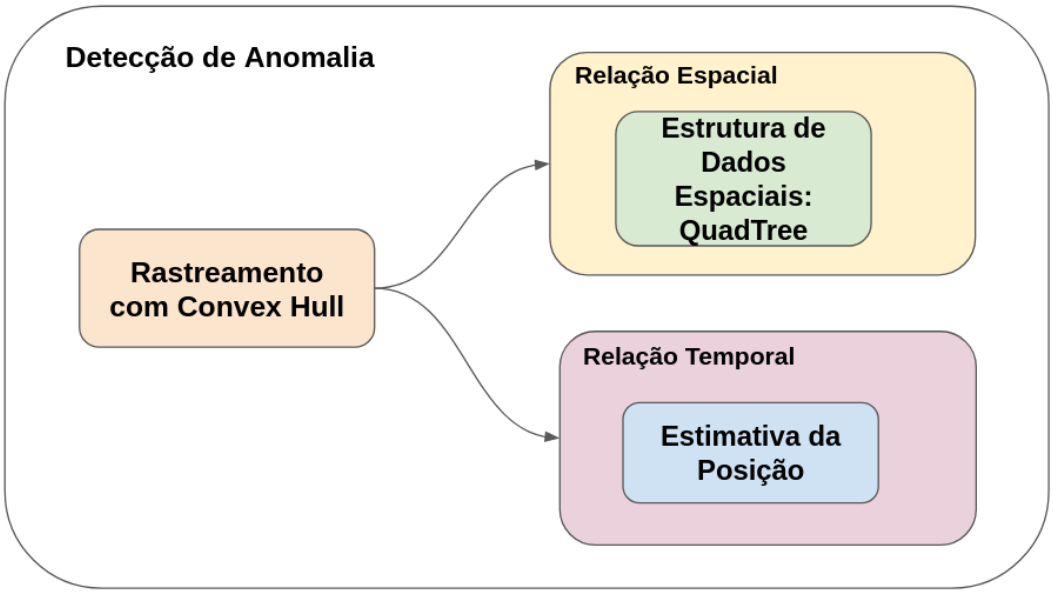
**Entrada:** *frames*
**Saída:** list_IA_obj, trajectory_obj ▷ Objetos com *IDs* únicos nas áreas de interesse

```
1:  i = 0
2:  enquanto i ≤ n_frames − 1 faça
3:      Graf[i] = Get_obj(F_i, F_{i+1})          ▷ Construção do grafo entre dois frames
4:      obj = Graf[i].get_obj                    ▷ Objetos presentes nesse grafo
5:      se Is_vértice (obj_i,obj_{i+1}) então    ▷ Analisa similitude de objetos nos frames
6:          atualizar(list_obj(obj_i,obj_{i+1})  ▷ Se é verdade agrega objetos
7:      senão
8:          IOU = IOU(obj_i,obj_{i+1})           ▷ Caso contrario analisa , 7, 8 e 9
9:          dist = dist(obj_i,obj_{i+1})
10:         sim = sim(obj_i,obj_{i+1})
11:         se IOU ⩾ γ and dist ⩽ σ and sim ⩾ θ então   ▷ Analisa os parametros γ, σ, θ
12:             atualizar(List_obj(obj_i,obj_{i+1}))    ▷ Agrega objetos
13:         senão
14:             list_obj.add(obj)                ▷ Novo IDs
15:         fim se
16:     fim se
17:     se Is_obj(i) and Is_AI(obj) então        ▷ Só veículos no polígono
18:         list_IA_obj.add(obj)
19:     fim se
20: fim enquanto
```

## 4.5 Detecção de anomalias

O componente de detecção de anomalia tem como objetivo detectar veículos parados nas vias principais. Implementamos duas estruturas essenciais: uma voltada para análises espaciais e outra para análises temporais, como é ilustrado na Figura 37. A análise espacial realiza buscas espaciais, permitindo a obtenção das localizações dos objetos, em especial para identificar veículos parados. Por outro lado, a análise temporal lida com desafios como a oclusão de objetos, garantindo que não percamos informações sobre o objeto. A análise temporal possibilita a identificação de objetos que saíram da cena ou estão imobilizados na via por um período prolongado ou mesmo quando o detector não faz a detecção. Essa abordagem combinada nos permite detectar veículos parados nas vias movimentadas, fornecendo uma solução robusta para a identificação de anomalias no tráfego rodoviário.

Figura 37 – Componente de Detecção de Anomalias.



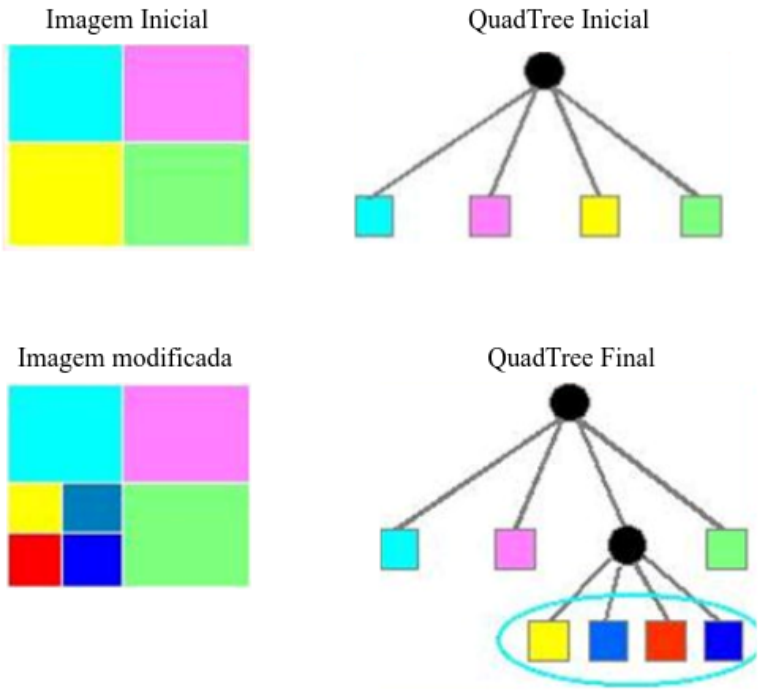

Fonte: Elaborada pelo autor.

### 4.5.1 Relação Espacial

Nesta última etapa, é recebido os dados dos objetos (veículos) dentro de uma área de interesse calculada pelo *Convex Hull*. Entretanto, foi observado que quando o veículo está parado na cena outro veículo pode sobrepor o veículo parado, dificultando assim, uma análise mais precisa. Assim, para o componente espacial utilizamos o método *QuadTree*, tendo como objetivo aprimorar a organização dos dados espaciais dentro da área delimitada. A *QuadTree* realiza uma divisão recursiva da região delimitada em quadrantes menores, criando uma representação hierárquica das informações espaciais, como pode se ver na Figura 38. Essa abordagem permite a análise dos veículos que estão próximos em termos de localização e, ao mesmo tempo, exclui regiões onde a análise não é necessária. Além disso, a *QuadTree* desempenha um papel importante na análise do comportamento dos veículos. Ao agrupar informações espaciais de maneira hierárquica, podemos identificar padrões de movimento, velocidades médias, interações entre veículos e outros aspectos importantes do comportamento veicular.

Assim para a etapa espacial, a *QuadTree* prova uma redução do custo computacional, mas também na análise do comportamento das trajetórias dos veículos, especialmente quando se trata de veículos parados. Ela analisa os objetos do *frame* atual em relação aos objetos próximos nos *frames* anteriores, considerando um raio "r" (ver Capítulo 5) como limite para a comparação com os objetos dos *frames* anteriores. *A QuadTree* é dividida em quatro subárvores, cada uma representando uma região espacial do campo de visão e escolhe a região onde os veículos estão próximos em relação aos veículos que estão sendo analisados. Esta estrutura de dados aprimora o rastreamento dos veículos, sendo usada em conjunto com outra estrutura de dados dedicada à

Figura 38 – Processo de construção da *QuadTree*




Fonte: Elaborada pelo autor.

análise temporal.

### 4.5.2 Relação Temporal

Por outro lado, a estrutura de dados temporais lida com situações em que os veículos podem estar temporariamente fora da cena. Ela realiza a aproximação das posições dos veículos que estavam temporariamente indisponíveis, garantindo que não percamos informações sobre esses veículos. Além disso, analisa o tempo que um veículo está na cena, o que é fundamental para evitar a reatribuição de *IDs* já existentes à novos veículos que entram no campo de visão. Essa abordagem impede que os veículos que estão parados na cena principal sejam erroneamente re-identificados, permitindo-nos manter um registro completo desses veículos e analisar se há ou não anomalias com base no tempo de permanência. Essa análise temporal é fundamental para identificar veículos que podem representar uma situação anômala, como um congestionamento prolongado ou uma parada não programada em uma via movimentada.

Para lidar com o mecanismo temporal, utilizamos a velocidade do veículo em cada *frame*, verificando se o veículo está em movimento ou parado. Caso o veículo esteja parado iniciamos o monitoramento desse objeto, caso ele continue parado em torno de 1800 *frames*, que corresponde a 1 min em que o veículo estará parado na rodovia, esse veículo será classificado como um veículo com comportamento anormal, pois um veículo parado na rodovia pode causar

um acidente grave.

### 4.5.3 Funcionamento da Detecção de Anomalias

O método de datação de anomalia recebe como entrada os rastreamentos dos objetos dentro de uma área de interesse e sua trajetória. Com essas informações, o método de detecção de anomalia utilizará a *QuadTree* para analisar objetos próximos nos *frames* anteriores (linha 3 do Algoritmo 2). Para analisar a proximidade se faz uma análise de semelhanças e critérios de limiares da distância e a *IOU* com objetos anteriores em *frames* anteriores em relação ao *frame* atual, (linhas 4 a 7). Se esses critérios são atendidos, as informações do objeto são atualizadas (linha 8). Entretanto, caso os critérios não forem atendidos, a estrutura de tempo verifica a posição do objeto para determinar se ele ainda está presente na cena. Para isso, construímos um retângulo e verificamos se o objeto está dentro dele. Se não estiver, presumimos que ele deixou a cena (linhas 9 a 12). No entanto, se estiver presente, analisamos sua velocidade nos últimos 100 *frames*. Se a velocidade for muito baixa, quase zero, inferimos que o veículo está parado na cena. Nesse caso, compreendemos que o veículo está oculto, e para não perdermos informações sobre sua trajetória, realizamos estimativas hipotéticas no próximo *frame* (linhas 13 a 14 ). Novamente, utilizamos a *QuadTree* para avaliar a proximidade e semelhança do objeto aproximado com os objetos próximos em *frames* anteriores, mantendo seu mesmo *ID* juntamente com todas as informações anteriores. Por fim, consideramos a quantidade de *frames* em que o veículo esteve na cena. Se essa quantidade for maior que 1800 *frames*, classificamos como anomalia. (linhas 18 a 19)

Para tornar o funcionamento do algoritmo mais compreensível, imagine um conjunto de *frames* consecutivos. No primeiro *frame*, podemos visualizar um veículo parado e outro veículo se sobrepondo ao veículo parado, como pode se ver na Figura 39. O processo de análise começa com o primeiro *frame*, onde utilizamos a *QuadTree* para examinar os objetos próximos nos f*rames* anteriores. A *QuadTree* é uma técnica que divide recursivamente a região da imagem em quadrantes menores, criando uma representação hierárquica das informações espaciais dos objetos. Utilizamos métricas de similaridade, distância e *IOU* para avaliar se os objetos atuais são semelhantes aos objetos dos *frames* anteriores. Se os objetos atuais não apresentam similaridade com os objetos próximos nos *frames* anteriores, passamos a utilizar a estrutura de dados temporal para verificar se os veículos ainda estão na cena. Para fazer isso, definimos um retângulo dentro do *frame*, com margens de -10 pixels em cada lado do retângulo. Se os veículos estiverem dentro desse retângulo, assumimos que ainda estão na cena (*frame* dois e três), mas podem não ter sido rastreados corretamente. Nesse caso, estimamos suas trajetórias calculando suas velocidades nos últimos 100 *frames*. Se a velocidade for muito baixa, indicando que o veículo está praticamente parado, assumimos que ele está oculto na cena. Após a estimativa, aplicamos novamente a *QuadTree*, utilizando as métricas de similaridade, distância e *IOU*, para avaliar a proximidade e semelhança do objeto estimado com os objetos próximos nos *frames* anteriores. Mantemos o

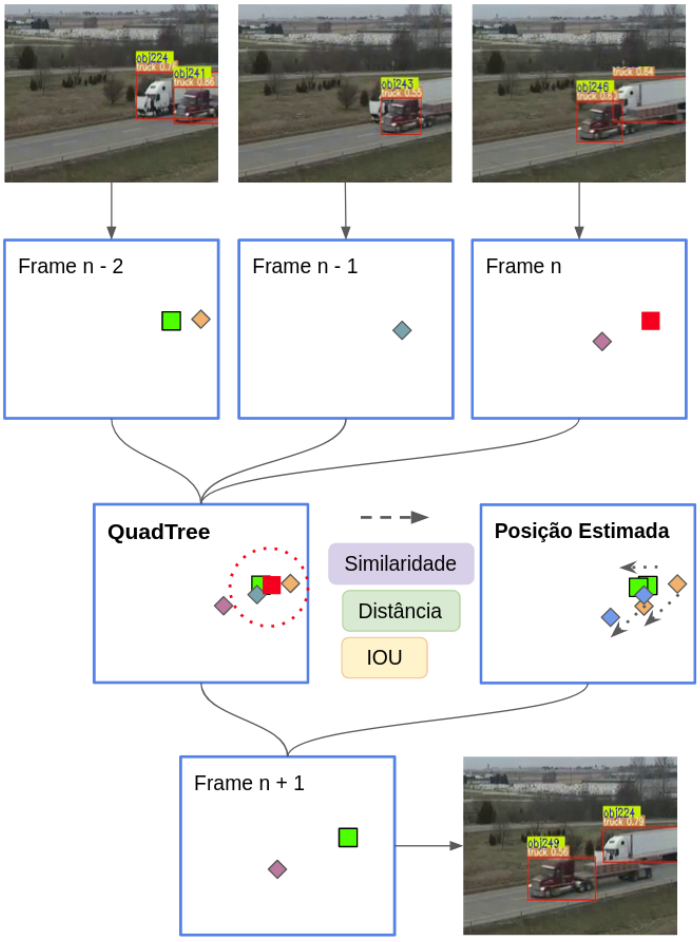
**Algoritmo 2** – Detecção de Anomalias

**Entrada:** list_IA_obj, trajectoria_obj
**Saída:** list_ano

```
1:  para veh in trajectory_obj faça
2:      se veh in list_IA_obj então
3:          quadtree = QuadTree(veh)                ▷ analisa obj prox nos frames anteriores
4:          IOU = IOU(quadtree.obj)
5:          dist = dist(quadtree.obj)
6:          sim = sim(quadtree.obj)
7:          se IOU ⩾ γ and dist ⩽ σ and sim ⩾ θ então    ▷ Analisa critérios de similaridade
8:              update(List_Ia_obj(quadtree.obj))        ▷ Atualiza informações do veículo
9:          senão
10:             se Is_scena(quadree.obj) então           ▷ Veículo não presente na cena
11:                 update(List_Ia_obj(quadtree.obj))    ▷ Deixou a cena
12:             senão
13:                 se Velocity_Zero(quadtree.obj) então b    ▷ Velocidade baixa
14:                     se Is_TheSameID(quadtree.obj) então   ▷ Veículo parado
15:                         timestamp ++                      ▷ Estimativa hipotética no prox frame
16:                     fim se
17:                 fim se
18:             fim se
19:         fim se
20:         se timestamp ==1800 então b    ▷ Veículo na cena por mais de 1800 frames
21:             list_ano.add(quadtree.obj)                    ▷ Anomalia
22:         fim se
23:     fim se
24: fim para
```

mesmo *ID* e todas as informações anteriores do objeto ao longo desse processo (último *frame*). Para classificar a situação como anomalia, contabilizamos o número de *frames* em que o veículo esteve presente na cena. Se essa quantidade for maior que 1800 *frames*, consideramos a situação como uma anomalia. Essa abordagem nos permite identificar situações em que um veículo permanece na cena por um período anormalmente longo, o que pode indicar uma anomalia.

Figura 39 – Ilustração da relação temporal e espacial.




Fonte: Elaborada pelo autor.

CAPÍTULO

# 5

# EXPERIMENTOS E RESULTADOS

Neste capitulo, descreveremos em detalhes a condução dos experimentos, apresentando informações específicas sobre o método proposto para a detecção e rastreamento de veículos e detecção de anomalias de veículos. Este método incorpora uma abordagem inovadora que combina algoritmos de visão computacional, aprendizado de máquina e aprendizado profundo para realizar as tarefas de detecção de veículos, acompanhamento de sua trajetória e identificação de comportamentos anômalos, contribuindo assim para um ambiente mais seguro e eficiente no contexto de transporte e mobilidade.

## 5.1 Detalhes de Implementação

A implementação deste trabalho foi realizada na linguagem de programação Python, utilizando a versão 3.11, e executada em uma máquina com o sistema operacional Linux (Ubuntu 20.04). O hardware utilizado para suportar o desenvolvimento inclui um processador Intel Core i7 de 8 núcleos, 16GB de memória RAM e uma placa NVidia RTX-3060 com 12GB de memória de vídeo. Esta configuração de *hardware* proporcionou um desempenho eficiente durante todas as fases de implementação do trabalho, garantindo a capacidade de processamento necessária para as tarefas deste trabalho.

## 5.2 Detecção de objetos

Para a detecção de objetos com o algoritmo *YoloV7*, avaliamos quatro conjuntos de pesos distintos: *YoloV7-D6*, *YoloV7-E6*, *YoloV7-E6E* e *YoloV7-W6*. Essas escolhas foram feitas com base em sua diversidade de tamanhos e precisões. *YoloV7-D6* é uma boa escolha para dispositivos com recursos limitados, enquanto *YoloV7-E6E* é a melhor escolha para máxima precisão. *YoloV7-E6* e *YoloV7-W6* são bons equilíbrios entre precisão e recursos. No entanto, os pesos em geral têm algumas limitações, como a incapacidade de detectar objetos que estão

parcialmente ocultos ou em movimento rápido. O peso *YoloV7-D6* tem o menor número de parâmetros e, portanto, é o mais rápido e eficiente em termos de recursos. No entanto, também é o menos preciso dos quatro modelos. O peso *YoloV7-E6* é mais preciso que o *YoloV7-D6*, mas também é mais lento e exige mais recursos. O *YoloV7-E6E* é o modelo mais preciso dos quatro modelos, mas também é o mais lento e exige os recursos mais altos. Emfim o *YoloV7-W6* é um bom equilíbrio entre precisão e recursos. É mais preciso que *YoloV7-D6*, mas não tão preciso quanto *YoloV7-E6E*. Também é mais rápido e eficiente em termos de recursos do que *YoloV7-E6E*. A melhor escolha para a detecção de veículos para este trabalho é o *YoloV7-W6*. É preciso o suficiente para detectar objetos com precisão, mas também é rápido e eficiente em termos de recursos.

Após variarmos os parâmetros de confiança e *IOU* e analisarmos os resultados detalhados. Para escolher os valores ideais de *IOU* e confiança, fizemos experimentos com diferentes valores. A partir destes testes, escolhemos quatro *frames* do vídeo 1 do conjunto de treinamento do *dataset UA-DETRAC* para uma comparação detalhada com os pesos selecionados. Optamos por testar valores de *thresholds* variando entre 0,5 e 0,7, uma faixa que oferece medidas razoáveis para obter resultados de qualidade. Ficou evidente que obteríamos resultados mais sólidos com um *IOU* de 0,5 e uma confiança de detecção de 0,5, como pode-se ver na Tabela 2. Com base nos resultados obtidos, nossa atenção se voltou para a utilização do peso *YoloV7-W6*, junto aos parâmetros mencionados anteriormente. Esta escolha se deu pela capacidade desse conjunto de parâmetros e peso em fornecer mais detecções, ao mesmo tempo em que diminuiu o tempo necessário para as inferências. As medições de tempo foram realizadas abrangendo tanto o carregamento dos pesos quanto a inferência nos *frames* selecionados. Essa abordagem permitiu uma avaliação completa do desempenho do sistema.

## 5.3 Rastreamento de objetos

### 5.3.1 Métricas de avaliação

As métricas de avaliação têm a finalidade de medir o desempenho na análise de sistemas de rastreamento de objetos. Neste trabalho, concentramos nossa atenção em duas áreas essenciais: rastreamento multi-objeto e estimativa de velocidade do veículo. Para a avaliação do rastreamento de objetos, utilizamos as métricas *Clear Metrics for Object Tracking (CLEAR MOT)*, conforme proposto por (BERNARDIN; STIEFELHAGEN, 2008), que tem diversos aspectos do desempenho. A métrica principal que empregamos é a *Multi-Object Tracking Accuracy (MOTA)*, que é fundamental para a avaliação geral do desempenho do rastreamento de múltiplos objetos. A *MOTA* leva em consideração o número de falsos positivos *(FP)*, falsos negativos *(FN)* e *ID Switches (IDS)* em relação ao *Ground Truth (GT)* . Em essência, a *MOTA* quantifica quão bem nosso sistema de rastreamento está detectando e seguindo corretamente os objetos em comparação com as anotações reais. Um valor mais alto de *MOTA* indica um desempenho de

Tabela 2 – Comparativo dos pesos usando diferentes parâmetros.

| **Pesos** | ***IOU*** | **Conf** | ***Frame*** | | | | **Tempo** |
|---|---|---|---|---|---|---|---|
| | | | **100** | **200** | **300** | **400** | |
| *YoloV7-D6* | 0.7 | 0.7 | 12 | 11 | 15 | 11 | 1.825s |
| *YoloV7-E6* | 0.7 | 0.7 | 10 | 10 | 13 | 10 | 1.880s |
| *YoloV7-E6E* | 0.7 | 0.7 | 8 | 11 | 11 | 11 | 1.951s |
| *YoloV7-W6* | 0.7 | 0.7 | 10 | 12 | 11 | 11 | 1.467s |
| *YoloV7-D6* | 0.5 | 0.7 | 12 | 11 | 15 | 11 | 1.927s |
| *YoloV7-E6* | 0.5 | 0.7 | 10 | 10 | 13 | 10 | 1.812s |
| *YoloV7-E6E* | 0.5 | 0.7 | 8 | 11 | 11 | 11 | 1.966s |
| *YoloV7-W6* | 0.5 | 0.7 | 10 | 12 | 11 | 11 | 1.488s |
| *YoloV7-D6* | 0.5 | 0.5 | 17 | 14 | 17 | 15 | 2.298s |
| *YoloV7-E6* | 0.5 | 0.5 | 17 | 16 | 18 | 16 | 1.880s |
| *YoloV7-E6E* | 0.5 | 0.5 | 14 | 13 | 16 | 14 | 1.967s |
| *YoloV7-W6* | 0.5 | 0.5 | 19 | 16 | 19 | 16 | 1.515s |

Fonte: Elaborada pelo autor.

rastreamento superior, com menos erros de detecção e identificação de objetos. A fórmula para calcular *MOTA* é a seguinte:

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + IDS_t)}{\sum_t GT_t} \tag{5.1}$$

Além disso, utilizamos outras métricas como a *Multi-Object Tracking Precision (MOTP)*, que caracteriza o desalinhamento entre os *bounding boxes* anotados e os *bounding boxes* previstos. O *Mostly Tracked (MT)* e *Mostly Lost (ML)* são empregados para avaliar a continuidade da trajetória dos objetos rastreados. Além disso, os *IDS* são usados para medir a qualidade do rastreamento em termos de identificação dos objetos. Para avaliar o desempenho computacional, também levamos em consideração o *Number of reasoning frames per second (Hz)*, o que descreve a frequência ou taxa na qual os *frames* de raciocínio são processados por segundo, e a unidade de medida dessa taxa é o *Hertz*.

### 5.3.2 Avaliação dos parâmetros

Para determinar os valores adequados para os parâmetros do nosso método, fizemos uma série de experimentos, explorando diferentes valores de similaridade, distância e *IOU*, como pode se ver na Tabela 3. Começamos com configurações fixas, mantendo um valor alto

e constante para a similaridade (Sim) e um valor baixo e constante para a distância (Dist), enquanto variamos os valores da *IOU* nas três primeiras linhas. No entanto, nas três últimas linhas, decidimos não fixar valores específicos para esses parâmetros, permitindo-nos observar o impacto considerável da distância, especialmente quando a *IOU* é reduzida devido ao tamanho considerável de alguns veículos e sua alta velocidade. Os resultados mais promissores de nossos experimentos são destacados na última linha da tabela. Notavelmente, observa-se que os FN e os *IDS* são menores em comparação com as outras configurações. No entanto, é importante notar que os FP são ligeiramente mais altos que na penúltima linha, com uma diferença mínima de apenas 4, fazendo que a *MOTA* tenha melhor resultado em comparação com os demais. Em geral, o objetivo é minimizar FP, FN e *IDS*, enquanto maximizamos a pontuação *MOTA*. Para nossos testes, selecionamos o vídeo 2 do conjunto de testes, que consiste em um total de 1120 *frames*. Optamos por utilizar a *SSMI* como nosso extrator de características, principalmente devido ao seu menor tempo de execução. Os pesos do grafo bipartido foram gerados de acordo com a Equação 4.7, onde $\alpha$ e $\beta$ são parâmetros que ajustam os pesos para garantir que permaneçam no intervalo entre zero e um. Neste trabalho, determinamos que os valores adequados para $\alpha$ e $\beta$ são, respectivamente, 0.4 e 0.6.

Tabela 3 – Avaliação dos parâmetros.

| **Parâmetros** | | | **Resultados** | | | |
|---|---|---|---|---|---|---|
| **Sim** | ***IOU*** | **Dist** | **FP** $\downarrow$ | **FN** $\downarrow$ | ***IDS*** $\downarrow$ | ***MOTA*** $\uparrow$ |
| 0.9 | 0.9 | 5 | 35 | 2346 | 27 | 0.7% |
| 0.9 | 0.8 | 5 | 276 | 485 | 200 | 60.4% |
| 0.9 | 0.7 | 5 | 390 | 122 | 11 | 78.4% |
| 0.8 | 0.6 | 10 | 395 | 126 | 7 | 78.2% |
| 0.7 | 0.5 | 20 | 378 | 129 | 5 | 78.9% |
| 0.6 | 0.4 | 30 | 382 | 125 | 4 | 78.9% |

Fonte: Elaborada pelo autor.

### 5.3.3 Avaliação do Rastreamento em Regiões de Interesse

Nosso objetivo principal é operar em áreas de interesse, e para atingir esse propósito, empregamos o algoritmo *Convex Hull*. Para um melhor entendimento, dividimos esta seção em duas partes: Rastreamento sem usar o *Convex Hull* e Rastreamento usando o *Convex Hull* para fazer um comparativo dos resultados. Realizamos os experimentos com o vídeo 1 do conjunto de treinamento, usando como extrator de características a *SSMI*, porque é mais rápido computacionalmente.

#### 5.3.3.1 Rastreamento sem usar o Convex Hull

Na primeira etapa do rastreamento, o objetivo foi atribuir um *ID* a todos os veículos presentes na cena, como o *dateset* em questão só contém informação das áreas de interesse. No entanto, ao não utilizar o algoritmo *Convex Hull*, o processo de rastreamento considerou todos os veículos presentes na cena, o que resultou em resultados inferiores aos esperados, como pode-se ver na tabela 4.

Tabela 4 – Resultados obtidos sem o uso do *Convex Hull*

| **Total de *frames*** | **FP** ↓ | **FN** ↓ | ***IDS*** ↓ | ***MOTA*** ↑ | ***MOTP*** ↑ | **Tempo** |
|---|---|---|---|---|---|---|
| 664 | 5023 | 1354 | 102 | 15% | 85% | 984s |

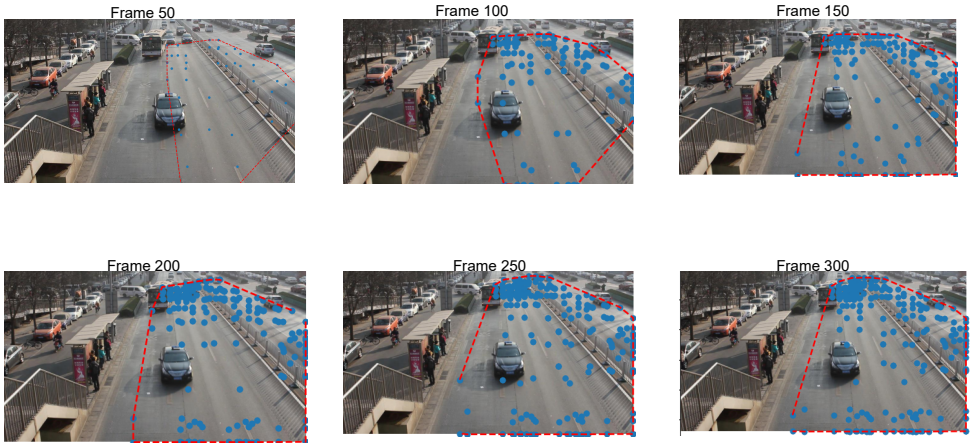
Fonte: Elaborada pelo autor.

#### 5.3.3.2 Rastreamento usando o Convex Hull

Na segunda etapa do rastreamento, nosso objetivo foi trabalhar com áreas em movimento e para isso usamos o algoritmo *Convex Hull*. Para gerar polígonos na cena, fizemos experimentos usando diferentes *frames* como referência. Isso significa que, para gerar um polígono, foi necessário identificar veículos em movimento, e para determinar se os veículos estavam em movimento, realizamos experimentos usando diferentes *frames* como pontos de referência. Inicialmente, começamos a partir do *frame* 50 e geramos um polígono, embora essa abordagem não abrangesse toda a área em movimento. Prosseguimos realizando experimentos com *frames* subsequentes, até alcançarmos o *frame* 300. Foi no *frame* 300 que conseguimos gerar um polígono que englobava toda a área em movimento, como pode ser observado na Figura 40. Essa abordagem dinâmica nos permitiu adaptar a geração de polígonos de acordo com a evolução das áreas em movimento na cena, garantindo uma cobertura abrangente das regiões relevantes para o rastreamento de veículos. Os resultados dessas avaliações estão detalhados na Tabela 5.

É importante destacar que obtivemos resultados mais satisfatórios em relação à métrica *MOTA* nos *frames* de referência 250 e 300, atingindo 73%. Essa diferença de 16%, 4%, 5% e 1% em relação aos *frames* de referência 50, 100, 150 e 200, respectivamente, acontece porque quanto maior é o *frame* de referência há mais informação dos veículos em áreas de interesse, mas também é necessário analisar até que quantidade de *frames* de referência a informação atinge valores ótimos. No que diz respeito ao *MOTP*, os resultados são comparáveis, isso acontece porque nos experimentos usamos o mesmo detector de objetos que faz com que as coordenadas dos *bounding boxes* do *GT* sejam próximas com às coordenadas dos *bounding boxes* do método proposto, fazendo com que os resultados sejam similares. Em termos de FN, os *frames* de referência 250 e 300 são menores em relação aos outros *frames* de referência. Essa redução nos FN contribuiu para um desempenho global mais preciso no rastreamento de objetos. Só em

Figura 40 – Polígonos tendo *frames* como pontos de referência




Fonte: Dados da pesquisa.

.

termos de FP e *IDS* nos *frames* de referência 250 e 300 não foram os esperados. No entanto o resultado geral foi ótimo.

Tabela 5 – Resultados obtidos com o uso do *Convex Hull*

| ***Frames* de referência** | **FP ↓** | **FN ↓** | ***IDS* ↓** | ***MOTA* ↑** | ***MOTP* ↑** | **Tempo** |
|---|---|---|---|---|---|---|
| 50 | 141 | 3168 | 18 | 57% | 85% | 1550s |
| 100 | 160 | 2206 | 34 | 69% | 85% | 1300s |
| 150 | 181 | 2220 | 36 | 68% | 85% | 1326s |
| 200 | 408 | 1638 | 82 | 72% | 84% | 1225s |
| 250 | 452 | 1566 | 84 | 73% | 84% | 1167s |
| 300 | 403 | 1576 | 82 | 73% | 84% | 1148s |

Fonte: Elaborada pelo autor.

Finalmente, os resultados da Tabela 4 são inferiores em relação à Tabela 5 em termos de FP, FN, *IDS*, *MOTA* e *MOTP*. Essa disparidade se deve ao fato de que na Tabela 4 não utilizamos o algoritmo *Convex Hull*, ou seja, o rastreamento incluiu áreas sem movimento, o que resultou em resultados abaixo do esperado. Isso ocorre porque o *dataset UA-DETRAC* se concentra em áreas em movimento (áreas de interesse). Nesse contexto, optamos por realizar experimentos gerais com *frames* de referência de 300 para cada vídeo no conjunto de teste, como detalhado na próxima subseção, Subseção 5.3.4.

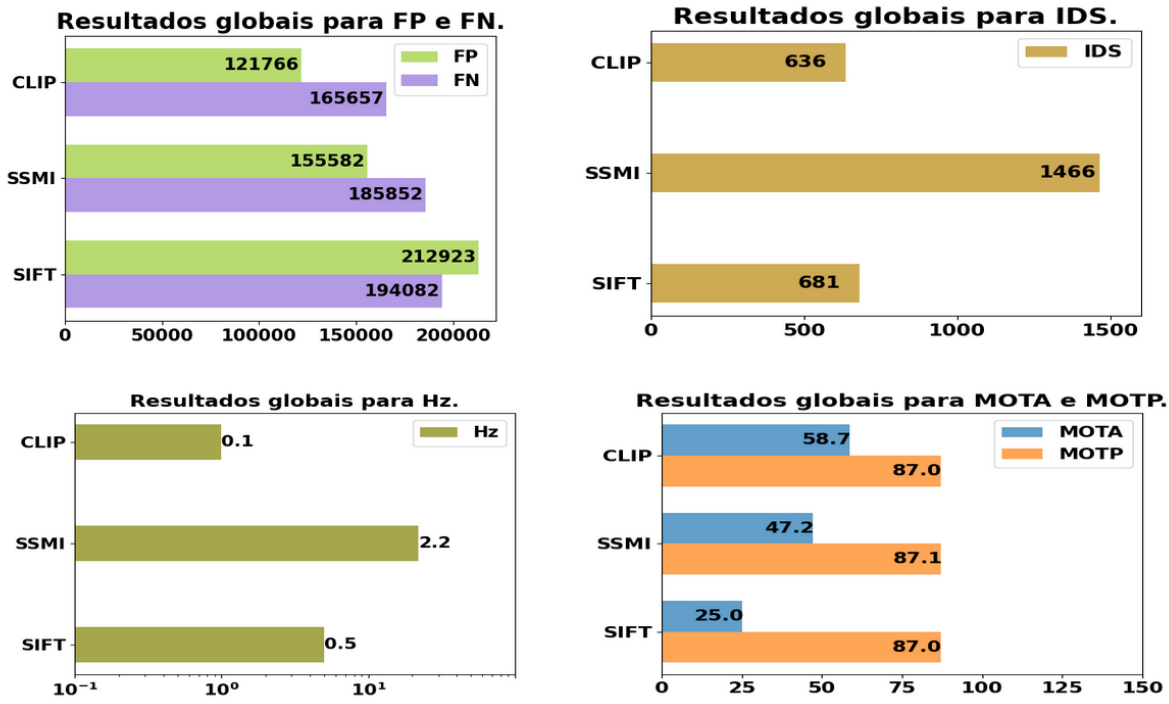
### 5.3.4 Resultados Gerais do MEDAVET no Rastreamento.

Nesta subseção, apresentaremos os resultados obtidos com os três extratores de características, destacando suas vantagens e desvantagens. Em seguida, analisaremos os resultados gerais, com foco no extrator *CLIP*, e compararemos o desempenho com outros métodos da literatura. Durante essa análise comparativa, também examinaremos as vantagens e desvantagens de cada um desses métodos.

#### 5.3.4.1 Resultados Gerais dos Extratores de Características

Os experimentos realizados no conjunto de testes *UA-DETRAC*, utilizando os extratores de características *CLIP*, *SSMI* e *SIFT*, claramente demonstraram a superioridade do *CLIP* em relação à *SSMI* e ao *SIFT*. Essa superioridade se reflete em melhorias significativas nas métricas, incluindo a redução de FP, FN e *IDS*, além de um desempenho superior em termos de precisão do rastreamento *MOTP* e métrica global de rastreamento *MOTA*. Os resultados desses experimentos estão detalhados na Figura 41.

Figura 41 – Resultados dos extratores de características.




Fonte: Elaborada pelo autor.

Essa vantagem do *CLIP* é atribuída à sua capacidade única de analisar a estrutura semântica de cada objeto, tornando-o notavelmente mais robusto em comparação aos outros extratores. Especificamente, em relação ao *MOTA*, o *CLIP* superou à *SSMI* em 11.5% e ao *SIFT* em 33.7%, demonstrando sua eficácia na tarefa de rastreamento. É importante observar que, em relação ao *MOTP*, o *CLIP* apresenta um desempenho comparável aos outros dois extratores. Isso

ocorre porque o *MOTP* avalia a diferença espacial entre os *bounding boxes* dos dados reais e aqueles gerados pelo método proposto, e o algoritmo *Yolo* desempenha um papel fundamental em padronizar e fornecer essas informações, permitindo uma comparação justa entre os extratores.

É importante mencionar que, em termos de *Hz*, a *SSMI* e o *SIFT* superaram ao *CLIP*. Isso ocorre porque o *CLIP*, devido à sua análise semântica detalhada, é mais intensivo em termos computacionais. No entanto, considerando a importância da precisão e do desempenho do rastreamento, a escolha do *CLIP* como extrator de características é justificada.

#### 5.3.4.2 Resultados Gerais com Outros Métodos

A avaliação do desempenho do rastreamento em comparação com outros modelos da literatura é importante para determinar a eficácia e a relevância de nosso método. A seguir, apresentaremos uma breve descrição de sete modelos identificados na literatura, a fim de estabelecer uma base comparativa sólida. Em seguida, realizaremos uma análise comparativa entre esses modelos, destacando suas principais características, pontos fortes e limitações. Isso nos permitirá avaliar o posicionamento de nosso modelo em relação aos existentes.

O *SORT* (BEWLEY *et al.*, 2016) é um algoritmo no campo de rastreamento de objetos em tempo real. Ele se baseia no uso da difusão de partículas, uma abordagem probabilística, para estimar a posição e rastrear objetos em sequências de vídeo. Uma das vantagens do *SORT* é sua simplicidade, o que o torna eficiente para uso em tempo real. Ele é particularmente útil em cenários onde é necessário rastrear objetos em movimento em vídeo, como em sistemas de vigilância, veículos autônomos e análise de vídeo. O *SORT* pode não ser tão preciso em situações desafiadoras, como quando os objetos estão muito próximos uns dos outros ou quando ocorrem oclusões.

O *High-speed tracking-by-detection without using image information (IOU)* (BOCHINSKI; EISELEIN; SIKORA, 2017) é um algoritmo de rastreamento de objetos notável por sua capacidade de rastrear objetos de alta velocidade com alta precisão, tudo isso sem a necessidade de processar informações de imagem. Em vez disso, ele se baseia em um modelo de movimento simples, fazendo suposições sobre como os objetos se deslocam. O sucesso do *IOU* se deve à sua eficiência e capacidade de lidar com objetos de alta velocidade. Como ele não depende da análise de imagens de vídeo, é computacionalmente eficiente e adequado para cenários onde a velocidade de processamento é crítica. No entanto, vale ressaltar que o *IOU* pode não ser tão robusto em cenários onde as suposições de movimento em linha reta e velocidade constante não se aplicam. Também pode ser menos eficaz em cenários com oclusões frequentes ou quando objetos mudam abruptamente de direção.

O *Confidence-Based Data Association and Discriminative Deep Appearance Learning for Robust Online Multi-Object Tracking (CMOT)* (BAE; YOON, 2018) é um algoritmo de rastreamento de objetos online que se destaca por sua robustez e eficácia, especialmente em cenas complexas com múltiplos objetos e de aparência semelhante. Ele faz uso de técnicas

de aprendizado profundo para discriminar objetos que compartilham características visuais próximas e divide o problema de rastreamento de objetos em subproblemas menores com base na confiança dos rastros. Isso permite que o algoritmo rastreie objetos com precisão e eficiência, mesmo em cenas complexas com muitos objetos.

O *Joint detection and tracking in videos with identification features (model2)* (MUNJAL *et al.*, 2020) é um algoritmo de detecção e rastreamento conjunto de objetos em vídeos usando características de identificação. O algoritmo usa um modelo de aprendizado profundo para detectar e rastrear objetos simultaneamente. O modelo de aprendizado profundo é treinado em um conjunto de dados de vídeos que contém objetos rotulados com características de identificação, como cor, textura e forma. O algoritmo melhora o desempenho de ambas as tarefas (detecção e rastreamento) em relação à abordagens convencionais. Isso ocorre porque o algoritmo usa as características de identificação para associar as detecções de objetos aos rastros de objetos com precisão. O algoritmo também é eficiente e robusto a oclusão, variações de iluminação e mudanças de aparência dos objetos. Isso ocorre porque o modelo de aprendizado profundo é treinado em um conjunto de dados de vídeos desafiador que contém uma grande variedade de objetos e condições.

O *Towards real-time multi-object tracking (JDE)* (WANG *et al.*, 2020) é um algoritmo de rastreamento de múltiplos objetos em tempo real que se destaca por sua capacidade de realizar tanto a detecção quanto o rastreamento de objetos de forma simultânea. Ele utiliza uma rede neural profunda para aprender essas tarefas de maneira conjunta, o que resulta em um rastreamento preciso e eficiente, mesmo em cenários complexos com a presença de vários objetos.

O *Fairmot: On the fairness of detection and re-identification in multiple object tracking (FairMOT)* (ZHANG *et al.*, 2021b) é um algoritmo de rastreamento de múltiplos objetos que trata a detecção e a re-identificação de forma equilibrada. Isso é importante porque, em abordagens convencionais, a detecção tende a ter um desempenho melhor do que a re-identificação, pois é uma tarefa mais fácil. O desequilíbrio de tarefas pode levar a um desempenho geral inferior do algoritmo de rastreamento. O algoritmo FairMOT usa uma rede neural única para realizar as tarefas de detecção e re-identificação simultaneamente. Isso evita o problema de desequilíbrio de tarefas e permite que o algoritmo obtenha resultados de última geração em conjuntos de dados desafiadores de rastreamento de objetos. O algoritmo FairMOT também é eficiente e robusto a oclusão, variações de iluminação e mudanças de aparência dos objetos. Isso ocorre porque a rede neural única é treinada em um conjunto de dados de vídeos diversificado e desafiador.

*ECCNet: Efficient chained centre network for real-time multi-category vehicle tracking and vehicle speed estimation (ECCNet)* (YU *et al.*, 2022) é um um avançado algoritmo de rastreamento em tempo real de veículos multi-categoria e estimativa de velocidade. O *ECCNet* utiliza uma rede neural profunda eficiente e é composto por três módulos principais: detecção, rastreamento e estimativa de velocidade. Sua estrutura encadeada permite a reutilização eficiente

de recursos do mapa de características, reduzindo o custo computacional e aprimorando o desempenho em situações desafiadoras, como oclusões e variações de iluminação. Este sistema oferece uma solução robusta e eficaz para aplicações que exigem o rastreamento preciso de veículos em tempo real, representando um avanço significativo no campo de visão computacional.

A comparação experimental detalhada no conjunto de testes do *dataset UA-DETRAC*, conforme resumida na Figura 42, demonstra que a nossa abordagem de rastreamento supera significativamente diversos métodos existentes. O *MEDAVET* atinge um *MOTA* de 58.7%, superando consideravelmente métodos como *SORT*, *IOU*, *CMOT*, *Model2*, *JDE*, *FairMOT* e *ECCNet* em 42.3%, 39.3%, 46.1%, 3.6%, 34.2%, 27% e 3.2%, respectivamente. Isso se deve ao excelente desempenho do *CLIP* na extração de características e à sua capacidade de trabalhar eficazmente em áreas de interesse. O *MEDAVET* apresenta um desempenho comparável ao *ECCNet* e ao *Model2* em relação ao *MOTP*. Isso ocorre porque esses três modelos utilizam métodos avançados de detecção, resultando em *bounding boxes* de detecção que estão próximos aos objetos reais. O *CMOT* e o *FairMOT* superam o *MEDAVET* em termos de *IDS*. Isso se deve ao fato de esses métodos empregarem modelos de re-identificação que podem associar detecções de objetos com base em características únicas, diferenciando objetos semelhantes mesmo em cenários complexos. O *MEDAVET*, por outro lado, utiliza um método de associação baseado em detecção, que pode resultar em erros de associação em situações desafiadoras. O *MEDAVET* é superado apenas pelo *Model2* e *ECCNet* em relação ao *MT* e pelo *Model2* em relação ao *ML*. Isso indica que o *MEDAVET* é altamente competitivo em termos de manter o rastreamento da maioria dos objetos, mas ainda há espaço para melhorias em relação aos objetos perdidos. O *MEDAVET* é superado em termos de *Hz* por outros modelos devido à complexidade computacional do *CLIP* e à análise detalhada da estrutura semântica. É importante destacar que, em aplicações onde a precisão é fundamental, a escolha do *MEDAVET* é justificada, mesmo que seja menos rápido em comparação com outros modelos.

## 5.4 Detecção de anomalia

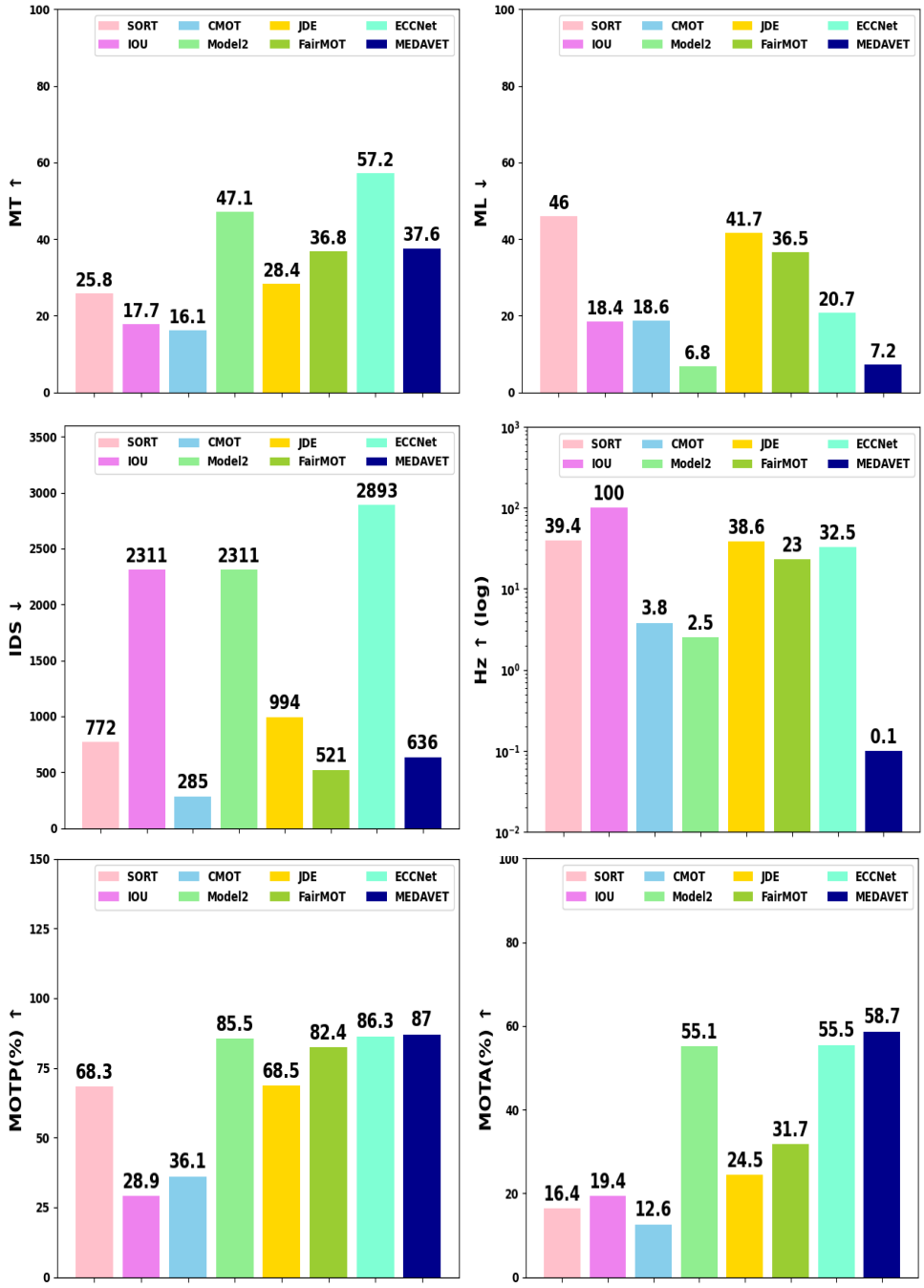
### 5.4.1 Métricas de avaliação

Para avaliar o desempenho da detecção de anomalias no conjunto de testes é usada a métrica $S4$, que é a combinação de duas métricas: a pontuação $F1$ e o *Normalized Root Mean Suared Error (NRMSE)*.

$$S4 = F1 \times (1 - NRMSE) \tag{5.2}$$

A pontuação $F1$ é a média harmônica de *recall* e precisão. Especificamente, uma detecção de TP é considerada a anomalia correta dentro de dez segundos de uma anomalia real (antes ou depois). Um FN é uma anomalia real que nosso algoritmo não consegue prever corretamente.

Figura 42 – Resultados gerais com outros métodos.




Fonte: Elaborada pelo autor.

Um FP representa a anomalia prevista mas não é uma anomalia real. A pontuação $F1$ é resumida por:

$$F1 = \frac{2\text{TP}}{2\text{TP} + \text{FN} + \text{FP}} \tag{5.3}$$

O *NRMSE* denota o erro temporal do tempo previsto (por nosso método) e o tempo de anomalia real ($GT$) para todas as previsões TP. O *NRMSE* emprega uma normalização max-min

com um valor máximo de 300 e um valor mínimo de 0. Resumindo, o *NRMSE* é definido da seguinte forma:

$$\text{NRMSE} = \frac{\min\left(\sqrt{\frac{1}{\text{TP}}\sum_{i=1}^{\text{TP}}\left(t_i^p - t_i^{gt}\right)^2}, 300\right)}{300}, \tag{5.4}$$

onde $t_i^{gt}$ denota o horário de início da anomalia do *GT* e $t_i^p$ é o horário de início previsto proposto por nosso método.

### 5.4.2 Resultados gerais do MEDAVET na detecção de anomalias

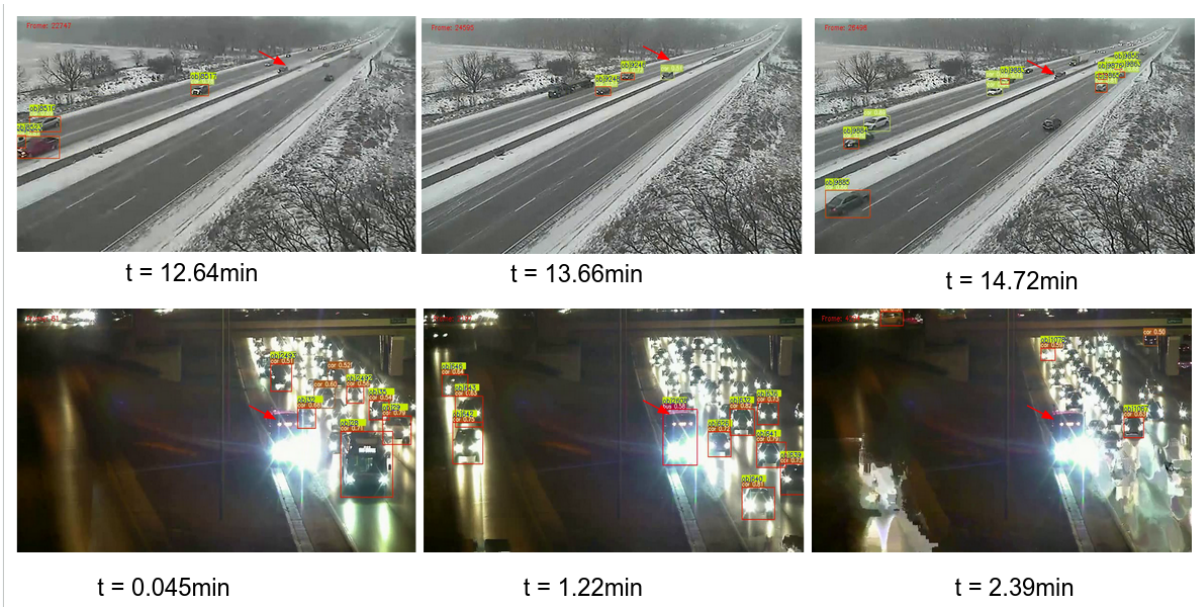
O algoritmo identifica a maior quantidade das anomalias presentes nos 150 vídeos do conjunto de testes e fornece o horário de início e fim da anomalia, além da pontuação de confiança.

As anomalias não identificadas são atribuídas a cenários desafiadores nos vídeos, como condições climáticas adversas, como neblina, ou períodos noturnos, nos quais as capacidades do detector são limitadas. Além disso, a distância entre a câmera e os veículos, juntamente com o tamanho reduzido dos veículos, contribuem para a não detecção constante desses veículos. Na Figura 43 apresentamos uma ilustração visual desses cenários. Nos primeiros três *frames* do vídeo 45 do conjunto de testes, o detector não consegue identificar os veículos durante todo o tempo, resultando na perda de informações e, consequentemente, na não detecção da anomalia. Nos três *frames* seguintes do video 41 também do conjunto de testes, o cenário noturno torna ainda mais desafiadora a detecção de veículos que estão parados por longos períodos, contribuindo igualmente para a não detecção das anomalias.

A Figura 44 apresenta *frames* do vídeo 43 do conjunto de testes, permitindo a visualização e ilustração das anomalias discutidas em nosso estudo. Para distinguir os períodos de parada dos veículos, adotamos um sistema de sinalização em cores. Inicialmente, um sinal de alarme é exibido em verde, indicando que o veículo está parado por um minuto. Após esse período, o alarme muda para a cor amarela e permanece nessa cor por dois minutos. Finalmente, ele se torna vermelho para indicar que o veículo está parado por mais de três minutos. A mudança de cores ao longo do tempo tem o propósito de informar que quanto mais prolongada for a parada, maior será o risco de acidentes.

Em contexto deste trabalho, a análise de TP, FN e FP tem muita relevância na avaliação do desempenho do sistema. A Figura 45 ilustra os resultados de todo o conjunto de testes em relação a essas métricas. Um TP neste cenário indica a concordância entre os dados reais e os dados previstos, seja para anomalias ou não anomalias. Neste contexto, encontramos um total de 198 TP. Em contrapartida, os FN indicam que, embora as anomalias estejam presentes nos dados reais, os dados previstos não as identificam corretamente, ou então, o veículo que apresenta anomalias nos dados reais não coincide com as previsões, resultando em 13 FN. No que diz

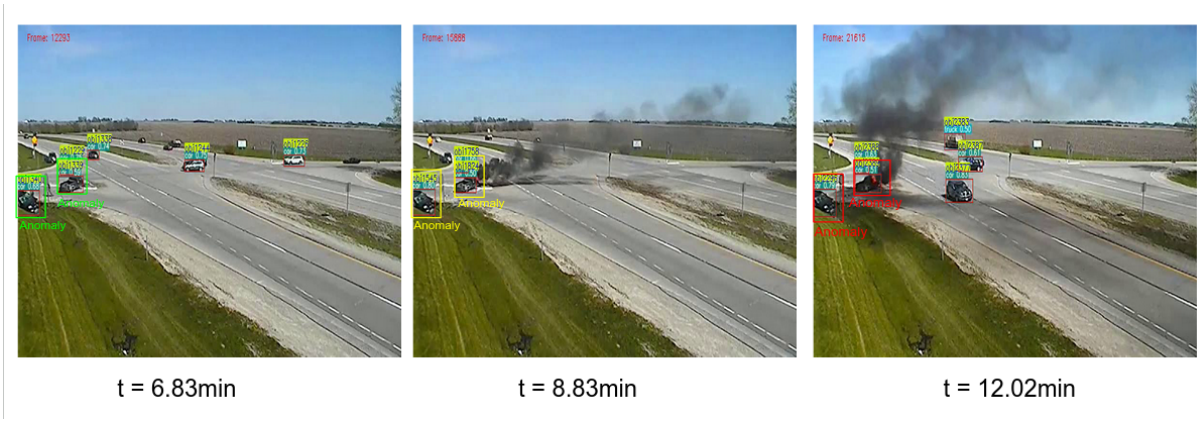
Figura 43 – Anomalias não detectadas.




Fonte: Dados da pesquisa.

.

Figura 44 – Anomalias detectadas



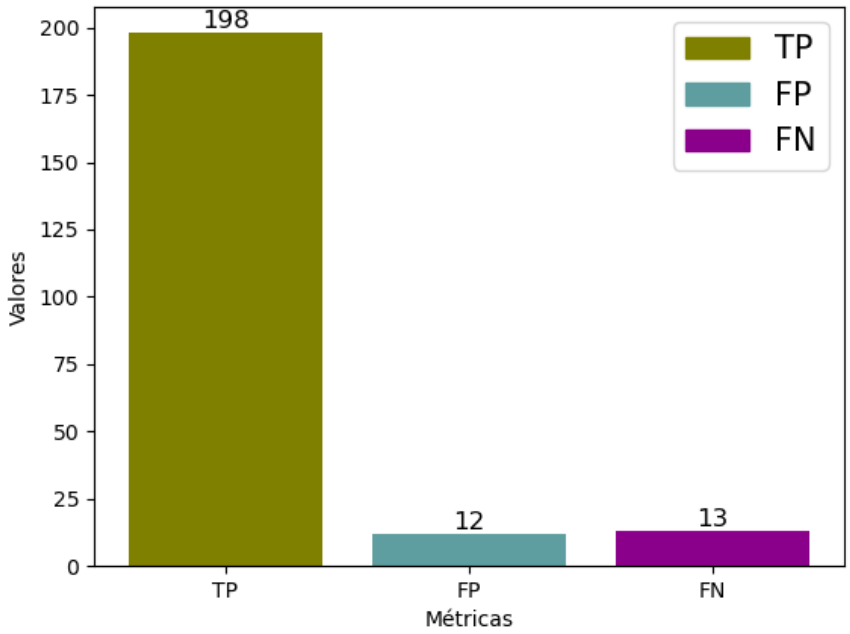

Fonte: Dados da pesquisa.

.

respeito aos FP, eles ocorrem quando as previsões indicam anomalias que não estão presentes nos dados reais, e neste caso, encontramos 12 FP. A presença de FN e FP é indesejável em qualquer cenário, porém, se fosse necessário escolher entre os dois, os FP são a opção menos prejudicial. Isso ocorre porque um FP levaria a um alarme de anomalia, que pode resultar em perda de tempo para o pessoal de assistência em acidentes ao se deslocar para o local, mas, em última análise, não haveria uma situação de risco real. No entanto, o pior cenário seria um FN, pois neste caso, haveria uma anomalia real, mas o algoritmo não a detectaria, o que poderia levar a acidentes graves, incluindo o risco de perda de vidas caso não sejam prestados os devidos socorros.

Avaliamos o desempenho do nosso método no conjunto de teste do *Track 4* do *NVIDIA AI CITY CHALLENGE 2021*. Como evidenciado na Tabela 6, obtivemos uma pontuação global

Figura 45 – Resultados globais de TP, FN e FP.




Fonte: Elaborada pelo autor.

Tabela 6 – Nosso resultado no conjunto de testes *Track 4*

| **S4** | **F1** | **RMSE** |
|---|---|---|
| 0.7845 | 0.8571 | 25.432 |

Fonte: Elaborada pelo autor.

de 0,7845 na métrica $S4$, acompanhada por uma sólida pontuação $F1$ de 85,71%. Além disso, o erro no tempo de início foi de 25,432 segundos, o que ressalta a robustez do nosso método proposto.

CAPÍTULO

# 6

# CONCLUSÕES

Esta dissertação de mestrado propôs um sistema de detecção e rastreamento de veículos, visando a identificação de anomalias no tráfego das vias urbanas, com o objetivo de contribuir para a prevenção e mitigação dos frequentes acidentes que ocorrem diariamente nas ruas da cidade.

O **MEDAVET** obteve resultados aceitáveis na etapa de rastreamento, superando significativamente alguns métodos da literatura. Dos três extratores que escolhemos, o extrator *CLIP* se destacou ao oferecer os melhores resultados, devido à sua capacidade de compreender a relação semântica entre as imagens. No entanto, é importante notar que o *CLIP* apresenta um custo computacional significativamente mais alto, o que impede o rastreamento em tempo real.

Na etapa de rastreamento, empregamos o algoritmo *Convex Hull* com o propósito de agrupar áreas de interesse, neste caso, veículos em movimento. Dado que o *dataset UA-DETRAC* opera em áreas de interesse e nosso objetivo é a detecção de anomalias, especificamente veículos parados, optamos por utilizar o *Convex Hull* nessa fase. Isso garante que o rastreamento seja exclusivamente aplicado a veículos em movimento, evitando a necessidade de realizar processos adicionais que seriam requeridos com outros métodos de rastreamento.

Para lidar com veículos que permanecem parados por longos períodos nas vias em movimento e que podem ser temporariamente ocultos, empregamos a estrutura de dados *QuadTree* juntamente com a estrutura temporal. Essa abordagem permitiu agrupar esses veículos e estimar suas posições eficazmente, evitando a perda de informações sobre potenciais anomalias. Essa combinação resultou em uma detecção robusta de anomalias.

Para trabalhos futuros, há algumas áreas de melhoria. Primeiramente, é necessário aprimorar a detecção de objetos, especialmente em cenários de baixa luminosidade, quando a imagem é pequena e distante da câmera, e em presença de ruídos. Isso envolveria o desenvolvimento de técnicas mais robustas de processamento para lidar com esses desafios.

Além disso, um trabalho futuro importante seria a otimização do algoritmo **MEDAVET** para torná-lo mais adequado para o processamento em tempo real. Isso poderia incluir a otimização de código ou uso de *hardware* mais poderoso. O objetivo é tornar o sistema mais responsivo e ágil, permitindo a detecção e rastreamento de anomalias em tempo real, o que seria crucial para a segurança viária e a prevenção de acidentes.

## 6.1 Contribuições Intelectuais

As contribuições intelectuais deste projeto incluem um artigo aceito e um artigo em submissão.

O primeiro artigo foi aceito na conferência internacional: *The Southwest Symposium on Image Analysis and Interpretation (SSIAI)*, a qual é dedicada a todos os aspectos da análise computacional e interpretação de imagens e vídeos. O objetivo principal deste artigo foi apresentar a detecção e rastreamento de veículos em áreas de interesse. Isso foi realizado através da aplicação do algoritmo *Convex Hull* e incluiu uma análise comparativa com diferentes extratores de características e diversas técnicas de rastreamento descritas na literatura.

- *Detection and tracking of multiple vehicles using semantic information*

O segundo artigo foi submetido ao *Journal of Internet Services and Applications*. Este artigo tem como objetivo apresentar a pesquisa completa, que se concentra na detecção de anomalias por meio da detecção e rastreamento de objetos. Este trabalho representa uma contribuição integral ao campo da detecção de anomalias e oferece uma visão abrangente da pesquisa realizada.

- *MEDAVET: Traffic Vehicle Anomaly Detection Mechanism based on spatial and temporal structures in vehicle traffic*

# REFERÊNCIAS


Aditya Sharma. **Understanding a Real-Time Object Detection Network: You Only Look Once (YOLOv1)**. 2022. Disponível em: <https://pyimagesearch.com/2022/04/11/understanding-a-real-time-object-detection-network-you-only-look-once-yolov1/>. Citado nas páginas 31 e 32.

AERIAL Infrared Object Tracking via an improved Long-term Correlation Filter with optical flow estimation and SURF matching. **Infrared Physics Technology**, v. 116, p. 103790, 2021. ISSN 1350-4495. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1350449521001626>. Citado na página 34.

AMARO, J. **3D-CSD+: Extração de características 3D baseada em grafos**. Tese (Doutorado) — Universidade de São Paulo, 2023. Citado na página 43.

AZAM, B.; KHAN, M. J.; BHATTI, F. A.; MAUD, A. R. M.; HUSSAIN, S. F.; HASHMI, A. J.; KHURSHID, K. Aircraft detection in satellite imagery using deep learning-based object detectors. **Microprocessors and Microsystems**, v. 94, p. 104630, 2022. ISSN 0141-9331. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0141933122001673>. Citado na página 32.

AZEVEDO, P. **Object Tracking State of the Art 2022**. 2022. Disponível em: <https://medium.com/@pedroazevedo6/object-tracking-state-of-the-art-2022-fe9457b77382>. Citado nas páginas 30 e 32.

BAE, S.-H.; YOON, K.-J. Confidence-based data association and discriminative deep appearance learning for robust online multi-object tracking. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 40, n. 3, p. 595–610, 2018. Citado na página 80.

BAFGHI, F.; SHOUSHTARIAN, B. Multiple-vehicle tracking in the highway using appearance model and visual object tracking. In: **2020 International Conference on Machine Vision and Image Processing (MVIP)**. [s.n.], 2020. p. 1–6. Disponível em: <https://ieeexplore.ieee.org/abstract/document/9116905>. Citado nas páginas 45, 46, 49 e 50.

BAI, S.; HE, Z.; LEI, Y.; WU, W.; ZHU, C.; SUN, M.; YAN, J. Traffic anomaly detection via perspective map based on spatial-temporal information matrix. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops**. [s.n.], 2019. Disponível em: <https://openaccess.thecvf.com/content_CVPRW_2019/papers/AI%20City/Bai_Traffic_Anomaly_Detection_via_Perspective_Map_based_on_Spatial-temporal_Information_CVPRW_2019_paper.pdf>. Citado nas páginas 46, 47 e 50.

BERNARDIN, K.; STIEFELHAGEN, R. Evaluating multiple object tracking performance: The clear mot metrics. **EURASIP Journal on Image and Video Processing**, v. 246309, p. 1–10, 2008. Disponível em: <https://jivp-eurasipjournals.springeropen.com/articles/10.1155/2008/246309>. Citado na página 74.

BEWLEY, A.; GE, Z.; OTT, L.; RAMOS, F.; UPCROFT, B. Simple online and realtime tracking. In: IEEE. **2016 IEEE international conference on image processing (ICIP)**. [S.l.], 2016. p. 3464–3468. Citado na página 80.

BOCHINSKI, E.; EISELEIN, V.; SIKORA, T. High-speed tracking-by-detection without using image information. In: IEEE. **2017 14th IEEE international conference on advanced video and signal based surveillance (AVSS)**. [S.l.], 2017. p. 1–6. Citado na página 80.

CANDELA[1], F.; MORABITO, F. C. Ssmi-cnn. In: SPRINGER NATURE. **Applied Intelligence and Informatics: Second International Conference, AII 2022, Reggio Calabria, Italy, September 1–3, 2022, Proceedings**. [S.l.], 2023. p. 293. Citado nas páginas 35 e 37.

CARUSI, C.; BIANCHI, G. Scientific community detection via bipartite scholar/journal graph co-clustering. **Journal of Informetrics**, v. 13, n. 1, p. 354–386, 2019. ISSN 1751-1577. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1751157718304073>. Citado na página 40.

CARVALHO, E. E. d. **Introdução à Visão Computacional**. 2020. Citado nas páginas 27 e 28.

CHARLES, P. K.; HARISH, V.; SWATHI, M.; DEEPTHI, C. A review on the various techniques used for optical character recognition. **International Journal of Engineering Research and Applications**, v. 2, n. 1, p. 659–662, 2012. Citado na página 27.

CHEN, H.-T.; LIN, H.-H.; LIU, T.-L. Multi-object tracking using dynamical graph matching. In: **Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001**. [s.n.], 2001. v. 2, p. II–II. ISSN: 1063-6919. Disponível em: <https://ieeexplore.ieee.org/document/990962?reload=true>. Citado na página 41.

CHEN, Z.; SUN, A. Anomaly detection on dynamic bipartite graph with burstiness. In: IEEE. **2020 IEEE International Conference on Data Mining (ICDM)**. 2020. p. 966–971. Disponível em: <https://ieeexplore.ieee.org/abstract/document/9338258>. Citado na página 39.

CHEUNG, W.; HAMARNEH, G. *n*-sift: *n*-dimensional scale invariant feature transform. **IEEE Transactions on Image Processing**, IEEE, v. 18, n. 9, p. 2012–2021, 2009. Citado nas páginas 34, 35, 36 e 37.

CHIEN, C.; KANADE, T. Distributed quadtree processing. In: SPRINGER. **Symposium on Large Spatial Databases**. [S.l.], 1989. p. 213–232. Citado na página 42.

COLLARANA, D.; GALKIN, M.; LANGE, C.; SCERRI, S.; AUER, S.; VIDAL, M.-E. Synthesizing knowledge graphs from web sources with the minte

Θ

framework. In: SPRINGER. **International Semantic Web Conference**. [S.l.], 2018. p. 359–375. Citado na página 40.

DATTA, P. All about structural similarity index (ssim): Theory+ code in pytorchg. 2020. **Available also from: https://medium. com/srm-mic/all-about-structural-similarity-indexssim-theory-code-in-pytorch-6551b455541e.[Online]. Accessed**, v. 20, 2023. Citado na página 35.

DAVIES, E. Chapter 1 - the dramatically changing face of computer vision. In: DAVIES, E.; TURK, M. A. (Ed.). **Advanced Methods and Deep Learning in Computer Vision**. Academic Press, 2022, (Computer Vision and Pattern Recognition). p. 1–91. ISBN 978-0-12-822109-9. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780128221099000102>. Citado nas páginas 29 e 30.

DJENOURI, Y.; BELHADI, A.; CHEN, H.-C.; LIN, J. C.-W. Intelligent deep fusion network for urban traffic flow anomaly identification. **Computer Communications**, Elsevier, v. 189, p. 175–181, 2022. Citado na página 24.

DUAN, R.; DENG, H.; TIAN, M.; DENG, Y.; LIN, J. Soda: A large-scale open site object detection dataset for deep learning in construction. **Automation in Construction**, v. 142, p. 104499, 2022. ISSN 0926-5805. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0926580522003727>. Citado nas páginas 30 e 31.

DWIVEDI, S. P.; SINGH, R. S. Error-tolerant approximate graph matching utilizing node centrality information. **Pattern Recognition Letters**, v. 133, p. 313–319, 2020. ISSN 0167-8655. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167865520300970>. Citado na página 40.

ELLAHI, A.; KHALIL, M. S.; AKRAM, F. *et al.* Computer users at risk: Health disorders associated with prolonged computer use. **Journal of Business Management and Economics**, Citeseer, v. 2, n. 4, p. 171–182, 2011. Citado na página 27.

FAN, Q.; BROWN, L.; SMITH, J. A closer look at faster r-cnn for vehicle detection. In: IEEE. **2016 IEEE intelligent vehicles symposium (IV)**. [S.l.], 2016. p. 124–129. Citado na página 46.

FERRANTE, G. S.; NAKAMURA, L. H. V.; ANDRADE, F. R. H.; FILHO, G. P. R.; GRANDE, R. E. D.; MENEGUETTE, R. I. Brazilian road's animals (bra): An image dataset of most commonly run over animals. In: **2022 35th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)**. [S.l.: s.n.], 2022. v. 1, p. 246–251. Citado na página 31.

FERRANTE, G. S.; RODRIGUES, F. M.; ANDRADE, F. R.; GOULARTE, R.; MENEGUETTE, R. I. Understanding the state of the art in animal detection and classification using computer vision technologies. In: IEEE. **2021 IEEE International Conference on Big Data (Big Data)**. [S.l.], 2021. p. 3056–3065. Citado na página 24.

FRITZ LABS INCORPORATED. **Object Detection Guide | Fritz AI**. 2022. Disponível em: <https://www.fritz.ai/object-detection/>. Citado na página 30.

GAO, L.; WANG, Y.; TANG, Z.; LIN, X. Newspaper article reconstruction using ant colony optimization and bipartite graph. **Applied Soft Computing**, v. 13, n. 6, p. 3033–3046, 2013. ISSN 1568-4946. Swarm intelligence in image and video processing. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494612003122>. Citado na página 40.

GAUTAM, A.; SINGH, S. Deep learning based object detection combined with internet of things for remote surveillance. **Wireless Personal Communications**, Springer, v. 118, n. 4, p. 2121–2140, 2021. Citado na página 31.

GE, D.-y.; YAO, X.-f.; XIANG, W.-j.; CHEN, Y.-p. Vehicle detection and tracking based on video image processing in intelligent transportation system. **Neural Computing and Applications**, Springer, v. 35, n. 3, p. 2197–2209, 2023. Citado na página 24.

GOMIDES, T. S.; ROBSON, E.; MENEGUETTE, R. I.; SOUZA, F. S. de; GUIDONI, D. L. Predictive congestion control based on collaborative information sharing for vehicular ad hoc networks. **Computer Networks**, Elsevier, v. 211, p. 108955, 2022. Citado nas páginas 24 e 25.

GU, Y.; DING, Z.; WANG, S.; ZOU, L.; LIU, Y.; YIN, D. Deep multifaceted transformers for multi-objective ranking in large-scale e-commerce recommender systems. In: **Proceedings of the 29th ACM International Conference on Information & Knowledge Management**. [S.l.: s.n.], 2020. p. 2493–2500. Citado na página 38.

HASHEMI, A.; DOWLATSHAHI, M. B.; NEZAMABADI-POUR, H. A bipartite matching-based feature selection for multi-label learning. **International Journal of Machine Learning and Cybernetics**, v. 12, n. 2, p. 459–475, fev. 2021. ISSN 1868-808X. Disponível em: <https://doi.org/10.1007/s13042-020-01180-w>. Citado na página 41.

HE, K.; GKIOXARI, G.; DOLLÁR, P.; GIRSHICK, R. Mask r-cnn. In: **Proceedings of the IEEE international conference on computer vision**. [S.l.: s.n.], 2017. p. 2961–2969. Citado na página 45.

HEALTH, S. D. of. Global status report on road safety 2018. **WHO**, World Health Organization, 2018. Citado na página 23.

HIGGOTT, O. Pymatching: A python package for decoding quantum codes with minimum-weight perfect matching. **ACM Transactions on Quantum Computing**, ACM New York, NY, v. 3, n. 3, p. 1–16, 2022. Citado na página 42.

HOU, X.; WANG, Y.; CHAU, L.-P. Vehicle tracking using deep sort with low confidence track filtering. In: IEEE. **2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)**. [S.l.], 2019. p. 1–6. Citado na página 47.

HUGHES, J. F.; FOLEY, J. D. **Computer graphics: principles and practice**. [S.l.]: Pearson Education, 2014. Citado nas páginas 27 e 29.

HUK, K.; KUROWSKI, M. Innovations and new possibilities of vehicle tracking in transport and forwarding. **Wireless Networks**, Springer, v. 28, n. 1, p. 481–491, 2022. Citado na página 24.

____. Innovations and new possibilities of vehicle tracking in transport and forwarding. **Wireless Networks**, Springer, v. 28, n. 1, p. 481–491, 2022. Citado na página 24.

JIANG, C.; QI, S.; ZHU, Y.; HUANG, S.; LIN, J.; YU, L.-F.; TERZOPOULOS, D.; ZHU, S.-C. Configurable 3d scene synthesis and 2d image rendering with per-pixel ground truth using stochastic grammars. **International Journal of Computer Vision**, Springer, v. 126, n. 9, p. 920–941, 2018. Citado na página 27.

JINJIANG, Y. Induced matching extendable graphs. **Journal of Graph Theory**, Wiley Online Library, v. 28, n. 4, p. 203–213, 1998. Citado na página 42.

JURADO-RODRíGUEZ, D.; JURADO, J. M.; PáDUA, L.; NETO, A.; MUñOZ-SALINAS, R.; SOUSA, J. J. Semantic segmentation of 3d car parts using uav-based images. **Computers Graphics**, v. 107, p. 93–103, 2022. ISSN 0097-8493. Disponível em: <https://www.sciencedirect.com/science/article/pii/S009784932200125X>. Citado nas páginas 29 e 30.

LI, Y.; WU, J.; BAI, X.; YANG, X.; TAN, X.; LI, G.; WEN, S.; ZHANG, H.; DING, E. Multi-granularity tracking with modularlized components for unsupervised vehicles anomaly detection. In: **2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)**. [s.n.], 2020. p. 2501–2510. Disponível em: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9150675>. Citado nas páginas 47, 48, 49 e 50.

LIMA, M. P. G. de; MOTA, G. L. A.; PINTO, P. E. D. Visualização da manipulação de dados em point quadtrees. **Cadernos do IME-Série Informática**, v. 34, p. 7–21, 2012. Citado nas páginas 41 e 42.

LIU, B.; HAN, C.; LIU, X.; LI, W. Vehicle artificial intelligence system based on intelligent image analysis and 5g network. **International Journal of Wireless Information Networks**, Springer, v. 30, n. 1, p. 86–102, 2023. Citado na página 24.

LOWE, G. Sift-the scale invariant feature transform. **Int. J**, v. 2, n. 91-110, p. 2, 2004. Citado nas páginas 34, 35 e 45.

MARTINEZ, F. J.; TOH, C.-K.; CANO, J.-C.; CALAFATE, C. T.; MANZONI, P. Emergency services in future intelligent transportation systems based on vehicular communication networks. **IEEE Intelligent Transportation Systems Magazine**, IEEE, v. 2, n. 2, p. 6–20, 2010. Citado na página 24.

MEEL, V. **Object Tracking in Computer Vision (Complete Guide)**. 2022. Disponível em: <https://viso.ai/deep-learning/object-tracking/>. Citado nas páginas 32, 33 e 34.

MELKMAN, A. A. On-line construction of the convex hull of a simple polyline. **Information Processing Letters**, Elsevier, v. 25, n. 1, p. 11–12, 1987. Citado na página 43.

METCALF, L.; CASEY, W. Chapter 5 - Graph theory. In: METCALF, L.; CASEY, W. (Ed.). **Cybersecurity and Applied Mathematics**. Boston: Syngress, 2016. p. 67–94. ISBN 9780128044520. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780128044520000051>. Citado nas páginas 39 e 40.

MIRANDA, A. C. de O.; SILVA da. Quadtrees e aplicações à geração de malhas. 2020. Citado nas páginas 41 e 42.

MONTANARI, R. **Detecção e classificação de objetos em imagens para rastreamento de veículos**. Tese (Doutorado) — Universidade de São Paulo, 2016. Citado na página 24.

MOUTIK, O.; TIGANI, S.; SAADANE, R.; CHEHRI, A. Hybrid deep learning vision-based models for human object interaction detection by knowledge distillation. **Procedia Computer Science**, v. 192, p. 5093–5103, 2021. ISSN 1877-0509. Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 25th International Conference KES2021. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050921020263>. Citado na página 31.

MUNJAL, B.; AFTAB, A. R.; AMIN, S.; BRANDLMAIER, M. D.; TOMBARI, F.; GALASSO, F. Joint detection and tracking in videos with identification features. **Image and Vision Computing**, Elsevier, v. 100, p. 103932, 2020. Citado na página 81.

NAPHADE, M.; WANG, S.; ANASTASIU, D. C.; TANG, Z.; CHANG, M.-C.; YANG, X.; ZHENG, L.; SHARMA, A.; CHELLAPPA, R.; CHAKRABORTY, P. The 4th ai city challenge. In: **The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops**. [S.l.: s.n.], 2020. p. 2665–2674. Citado na página 55.

NAPHADE, M.; WANG, S.; ANASTASIU, D. C.; TANG, Z.; CHANG, M.; YAO, Y.; ZHENG, L.; RAHMAN, M. S.; VENKATACHALAPATHY, A.; SHARMA, A.; FENG, Q.; ABLAVSKY, V.; SCLAROFF, S.; CHAKRABORTY, P.; LI, A.; LI, S.; CHELLAPPA, R. The 6th ai city challenge. In: **2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition**

**Workshops (CVPRW)**. [S.l.]: IEEE Computer Society, 2022. p. 3346–3355. Citado na página 57.

NATIONS., U. Seventy-fourth session of the united nations general assembly: Improving global road safety. **United Nations General Assembly**, United Nations General Assembly, 2020. Citado na página 23.

Nilesh Barla. **The Complete Guide to Object Tracking [+V7 Tutorial]**. 2022. Disponível em: <https://www.v7labs.com/blog/object-tracking-guide,https://www.v7labs.com/blog/object-tracking-guide>. Citado na página 34.

NOWOZIN, S. Optimal decisions from probabilistic models: the intersection-over-union case. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2014. p. 548–555. Citado na página 46.

NUNES, V. d. S. Tópicos em visão computacional: uma revisão sistemática com aplicações em economia 4.0. Serra, 2023. Citado na página 29.

Pavan Vadapalli. **Ultimate Guide to Object Detection Using Deep Learning [2022]**. 2021. Disponível em: <https://www.upgrad.com/blog/ultimate-guide-to-object-detection-using-deep-learning/>. Citado na página 30.

PAWAR, K.; ATTAR, V. Deep learning based detection and localization of road accidents from traffic surveillance videos. **ICT Express**, Elsevier, 2021. Citado nas páginas 24 e 25.

PEREIRA, K. O. P. Uso de visão computacional para reconhecimento de imagens de frutas em imagens rgb. 2022. Citado na página 28.

PUPYREV, S.; NACHMANSON, L.; BEREG, S.; HOLROYD, A. E. Edge routing with ordered bundles. In: SPRINGER. **International Symposium on Graph Drawing**. [S.l.], 2011. p. 136–147. Citado na página 39.

RANFTL, R.; BOCHKOVSKIY, A.; KOLTUN, V. Vision transformers for dense prediction. In: **Proceedings of the IEEE/CVF international conference on computer vision**. [S.l.: s.n.], 2021. p. 12179–12188. Citado nas páginas 38 e 39.

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. You only look once: Unified, real-time object detection. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. Citado nas páginas 32 e 33.

RUIZ, D. V.; TODT, E. Beyond observation: an approach for objectnav. **arXiv preprint arXiv:2106.11379**, 2021. Citado nas páginas 27, 28 e 29.

SALARI, A.; DJAVADIFAR, A.; LIU, X.; NAJJARAN, H. Object recognition datasets and challenges: A review. **Neurocomputing**, v. 495, p. 129–152, 2022. ISSN 0925-2312. Disponível em: <https://www.sciencedirect.com/science/article/pii/S092523122200039X>. Citado na página 31.

SALMAN, M. E.; Çakirsoy Çakar, G.; AZIMJONOV, J.; KöSEM, M.; CEDIMOğLU İsmail H. Automated prostate cancer grading and diagnosis system using deep learning-based yolo object detection algorithm. **Expert Systems with Applications**, v. 201, p. 117148, 2022. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417422005425>. Citado na página 32.

SANTOS, E. d. **O uso de visão computacional para o controle de um manipulador robótico**. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2014. Citado na página 24.

SHABBIR, J.; ANWER, T. Artificial intelligence and its role in near future. **arXiv preprint arXiv:1804.01396**, 2018. Citado na página 24.

SHI, J. *et al.* Good features to track. In: IEEE. **1994 Proceedings of IEEE conference on computer vision and pattern recognition**. [S.l.], 1994. p. 593–600. Citado na página 48.

SKLANSKY, J. Finding the convex hull of a simple polygon. **Pattern Recognition Letters**, Elsevier Science Inc. New York, NY, USA, v. 1, n. 2, p. 79–83, 1982. Citado na página 43.

SYEDRZ, R. **Object Tracking in Deep Learning – Deep Machine Learning AI**. 2020. Disponível em: <http://deepmachinelearningai.com/object-tracking-in-deep-learning/>. Citado na página 32.

TAMURA, Y.; ITO, T.; ZHOU, X. Approximability of the independent feedback vertex set problem for bipartite graphs. **Theoretical Computer Science**, v. 849, p. 227–236, 2021. ISSN 0304-3975. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0304397520306101>. Citado na página 40.

TOMIC, T.; SCHMID, K.; LUTZ, P.; DOMEL, A.; KASSECKER, M.; MAIR, E.; GRIXA, I. L.; RUESS, F.; SUPPA, M.; BURSCHKA, D. Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue. **IEEE robotics & automation magazine**, IEEE, v. 19, n. 3, p. 46–56, 2012. Citado nas páginas 27 e 28.

VANDERSCHUREN, M.; ROUX, D. Road safety comparison in south africa -how do the different provinces compare? In: . [S.l.: s.n.], 2019. Citado na página 23.

VIANA, S. **O pipeline de Visão Computacional**. 2020. Disponível em: <https://suzana-svm.medium.com/o-pipeline-de-visao-computacional-com-python-opencv-adc70112f5ee>. Citado na página 28.

WANG, Z.; ZHENG, L.; LIU, Y.; LI, Y.; WANG, S. Towards real-time multi-object tracking. In: SPRINGER. **European Conference on Computer Vision**. [S.l.], 2020. p. 107–122. Citado na página 81.

WEN, L.; DU, D.; CAI, Z.; LEI, Z.; CHANG, M.; QI, H.; LIM, J.; YANG, M.; LYU, S. DETRAC: A new benchmark and protocol for multi-object detection and tracking. **arXiv CoRR**, abs/1511.04136, 2015. Disponível em: <https://detrac-db.rit.albany.edu/Tracking>. Citado nas páginas 55 e 56.

YANG, B.; TANG, M.; CHEN, S.; WANG, G.; TAN, Y.; LI, B. A vehicle tracking algorithm combining detector and tracker. **EURASIP Journal on Image and Video Processing**, v. 2020, n. 1, p. 17, abr. 2020. ISSN 1687-5281. Disponível em: <https://doi.org/10.1186/s13640-020-00505-7>. Citado na página 33.

YU, C.; YANG, J.; JIANG, S.; ZHANG, Y.; LI, H.; DU, L. Eccnet: Efficient chained centre network for real-time multi-category vehicle tracking and vehicle speed estimation. **IET Intelligent Transport Systems**, Wiley Online Library, v. 16, n. 11, p. 1489–1503, 2022. Citado na página 81.

YU, Q.; CHEN, J.; DU, Y.; SUI, J.; DAMARAJU, E.; TURNER, J. A.; van Erp, T. G.; MACCIARDI, F.; BELGER, A.; FORD, J. M.; MCEWEN, S.; MATHALON, D. H.; MUELLER, B. A.; PREDA, A.; VAIDYA, J.; PEARLSON, G. D.; CALHOUN, V. D. A method for building a genome-connectome bipartite graph model. **Journal of Neuroscience Methods**, v. 320, p. 64–71, 2019. ISSN 0165-0270. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0165027019300895>. Citado na página 39.

ZHANG, X.; ZHENG, Y.; ZHAO, Z.; LIU, Y.; BLUMENSTEIN, M.; LI, J. Deep learning detection of anomalous patterns from bus trajectories for traffic insight analysis. **Knowledge-Based Systems**, Elsevier, v. 217, p. 106833, 2021. Citado na página 24.

ZHANG, Y.; WANG, C.; WANG, X.; ZENG, W.; LIU, W. Fairmot: On the fairness of detection and re-identification in multiple object tracking. **International Journal of Computer Vision**, Springer, v. 129, p. 3069–3087, 2021. Citado na página 81.

ZHAO, Y. **Good Practices and A Strong Baseline for Traffic Anomaly Detection**. [s.n.], 2021. Disponível em: <https://arxiv.org/pdf/2105.03827.pdf>. Citado nas páginas 25, 48, 49, 50 e 51.

ZOLA, F.; SEGUROLA-GIL, L.; BRUSE, J. L.; GALAR, M.; ORDUNA-URRUTIA, R. Network traffic analysis through node behaviour classification: a graph-based approach with temporal dissection and data-level preprocessing. **Computers & Security**, Elsevier, v. 115, p. 102632, 2022. Citado na página 39.

ZUO, S.; XIAO, Y.; CHANG, X.; WANG, X. Vision transformers for dense prediction: A survey. **Knowledge-Based Systems**, Elsevier, v. 253, p. 109552, 2022. Citado nas páginas 38 e 39.


ICMC
USP
SÃO CARLOS