

**UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO**



JOÃO MANOEL MACEDO MOREIRA DA COSTA

**DETECÇÃO E RASTREAMENTO DE VEÍCULOS: UMA
ABORDAGEM POR VISÃO COMPUTACIONAL E DEEP
LEARNING**

VITÓRIA-ES

AGOSTO/2022

João Manoel Macedo Moreira da Costa

DETECÇÃO E RASTREAMENTO DE VEÍCULOS: UMA ABORDAGEM POR VISÃO COMPUTACIONAL E DEEP LEARNING

Parte manuscrita do Projeto de Graduação do aluno João Manoel Macedo Moreira da Costa, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Vitória-ES

Agosto/2022

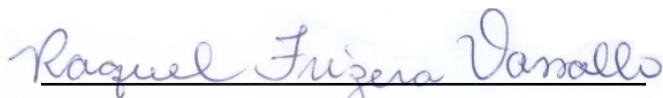
João Manoel Macedo Moreira da Costa

DETECÇÃO E RASTREAMENTO DE VEÍCULOS: UMA ABORDAGEM POR VISÃO COMPUTACIONAL E DEEP LEARNING

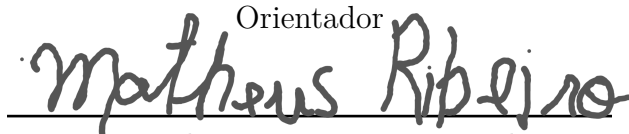
Parte manuscrita do Projeto de Graduação do aluno João Manoel Macedo Moreira da Costa, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Aprovado em 18 de agosto de 2022.

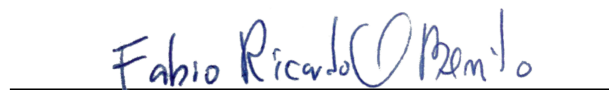
COMISSÃO EXAMINADORA:



Profa. Dra. Raquel Frizera Vassallo
Universidade Federal do Espírito Santo
Orientador




Msc. Matheus Vieira Lessa Ribeiro
Universidade Federal do Espírito Santo
Coorientador



Prof. Msc. Fabio Ricardo Oliveira Bento

Instituto Federal do Espírito Santo



Prof. Dr. Jorge Leonid Aching Samatelo

Universidade Federal do Espírito Santo
Examinador



Eng. Augusto Abling
Universidade Federal do Espírito Santo
Examinador

Vitória-ES
Agosto/2022

AGRADECIMENTOS

Aos meus pais, pelo apoio e dedicação a mim, não só durante toda a graduação, como durante toda a minha vida.

À minha namorada por toda compreensão e apoio durante toda a graduação.

A minha orientadora Raquel Frizera Vassallo, e aos meus coorientadores, Matheus Vieira Lessa Ribeiro e Fabio Ricardo Oliveira Bento, por toda ajuda, orientação e dedicação durante o desenvolvimento deste trabalho.

À banca examinadora pela aceitação do convite e pelo tempo investido para leitura e avaliação desse trabalho.

RESUMO

Devido ao rápido crescimento metropolitano, soluções para a identificação de veículos em vias têm sido de extrema importância para o planejamento e manutenção do tráfego urbano. No entanto, ainda que diferentes estudos e aprimoramentos tenham sido realizados, essa temática ainda carece de soluções para auxiliar nesse planejamento. Enquanto isso, com o aumento da qualidade e queda no custo de câmeras de vídeo, além do aumento na capacidade de processamento dos computadores, soluções baseadas em visão computacional e *deep-learning* (tradução livre, aprendizado profundo), têm se mostrado cada vez mais promissoras na tarefa de detectar e rastrear objetos. Nesse sentido, este trabalho busca desenvolver um sistema capaz de realizar a detecção e o rastreamento de veículos em vias públicas. A avaliação do desempenho do sistema foi realizada com métricas associadas à acurácia e precisão do conjunto detector-rastreador, além do tempo de execução do sistema quando aplicado ao *dataset* (tradução livre, conjunto de dados) UA-DETRAC. A fim de se obter um sistema capaz de realizar detecções em tempo real e com alto desempenho nas métricas citadas anteriormente, foram analisados diferentes modelos do método de detecção de objetos YOLOv5 (do inglês, *You Only Look Once*), treinada com a base pública MS COCO. Do mesmo modo, optou-se por avaliar o desempenho desses detectores operando em conjunto com rastreadores conhecidos na comunidade científica, como por exemplo, o SORT (do inglês, *Simple Online Realtime Tracker*), e o Deep SORT. Por fim, será analisada uma solução própria baseada no SORT, alterando seu espaço de estados e a métrica de associação aplicada. Como resultado, chegou-se a um conjunto detector-rastreador capaz de alcançar um HOTA de 50,033% quando avaliado no *dataset* UA-DETRAC.

Palavras-chave: Mobilidade Urbana; Detecção de Objetos; Rastreamento de Objetos; *Deep-learning*.

ABSTRACT

Due to the rapid metropolitan growth, solutions for the re-identification of vehicles on roads have been of utmost importance for the planning and maintenance of urban traffic. However, even though many studies and improvements have been carried out, this topic still lacks solutions to support this planning. Meanwhile, with the increase in quality and decrease in the cost of video cameras, besides the increase in the processing capacity of computers, solutions based on computer vision and deep learning have shown to be increasingly effective in detecting and tracking objects. In this context, this work aims to develop a system capable of detecting and tracking vehicles on public roads. The system performance was evaluated using metrics associated with the accuracy and precision of the detector-tracker set, as well as the system execution time when applied to the UA DETRAC dataset. To obtain a system capable of performing detections in real-time and with high performance in the metrics previously mentioned, different models of the YOLOv5 (You Only Look Once) object detection method were analyzed and trained with the public MS COCO database. In the same way, it was decided to evaluate the performance of these detectors operating in conjunction with trackers known in the scientific community, such as SORT (Simple Online Realtime Tracker), and Deep SORT. Finally, a custom solution based on SORT will be analyzed, changing its state space and the association metric applied. As a result, a detector-tracker set achieved an average HOTA of 50.033% when evaluated on the UA-DETRAC dataset.

Keywords: Urban Mobility; Object Detection; Object Tracking; *Deep-learning*.

LISTA DE FIGURAS

Figura 1	–	Representação das camadas existentes nas CNNs	18
Figura 2	–	Obtenção de Propostas de Regiões <i>via Selective Search</i>	19
Figura 3	–	<i>Pipeline</i> de detecção da R-CNN	20
Figura 4	–	<i>Pipeline</i> de detecção da <i>Fast R-CNN</i>	21
Figura 5	–	Rede unificada da <i>Faster R-CNN</i>	22
Figura 6	–	Detector de objetos SSD. (a) Imagem com <i>bounding-boxes</i> representando as detecções obtidos pelo detector, (b) Mapa de <i>features</i> 8x8, (c) Mapa de <i>features</i> 4x4	23
Figura 7	–	Modelo de detecção do detector de objetos YOLO	24
Figura 8	–	Exemplificação algoritmo de <i>Non-maximal Suppression</i> : (a) detecção sem a aplicação do algoritmo e (b) detecção com a aplicação do algoritmo	26
Figura 9	–	Diagrama UML simplificado da arquitetura desenvolvida para visualização de imagens	38
Figura 10	–	Diagrama UML da arquitetura desenvolvida para detecção de objetos	39
Figura 11	–	Diagrama UML da arquitetura desenvolvida para o rastreamento de objetos	40
Figura 12	–	Obtenção da velocidade estimada aplicada ao conjunto de dados UA-DETRAC.	43
Figura 13	–	Processo de contagem de veículos aplicado ao conjunto de dados UA-DETRAC.	43
Figura 14	–	Processo de obtenção da densidade de via aplicado ao conjunto de dados UA-DETRAC.	44
Figura 15	–	Exemplo de imagens pertencentes ao conjunto de dados UA-DETRAC	46
Figura 16	–	Demonstração visual do conceito de erro de detecção	47
Figura 17	–	Demonstração visual do conceito de erro de localização	48
Figura 18	–	Demonstração visual do conceito de erro de associação	49
Figura 19	–	Demonstração visual dos conceitos de VP, FP e FN	50
Figura 20	–	Demonstração visual dos conceitos de AVP, AFP e AFN	53
Figura 21	–	Comparação AssA vs DetA	56
Figura 22	–	Comparação HOTA vs IDF1	57
Figura 23	–	Comparação HOTA vs MOTA	58

LISTA DE TABELAS

Tabela 1 – Resultados da análise de desempenho do rastreamento de carros no conjunto de dados UA-DETRAC.	55
--	----

LISTA DE ABREVIATURAS E SIGLAS

COCO	<i>Comon Object in Context</i>
CNN	<i>Convolutional Neural Network</i>
CV	<i>Computer Vision</i>
DL	<i>Deep Learning</i>
DETRAN	Departamento de Trânsito
ES	Espírito Santo
FN	<i>Falso Negativo</i>
FP	<i>Falso Positivo</i>
FPS	<i>Frames Per Second</i>
GPU	<i>Graphics Processing Unit</i>
GT	<i>Ground-truth</i>
HOTA	<i>High Order Tracking Accuracy</i>
IOU	<i>Intersection Over Union</i>
ML	<i>Machine Learning</i>
mAP	<i>Mean Average Precision</i>
MS	<i>Microsoft</i>
MOT	<i>Multiple Object Tracking</i>
MVC	<i>Model-View-Controller</i>
RCNN	<i>Region-based Convolutional Neural Network</i>
RoI	<i>Region of Interest</i>
RPN	<i>Region Proposal Network</i>
SORT	<i>Simple Online and Realtime Tracker</i>
SOT	<i>Single Object Tracking</i>

SSD	<i>Single Shot Detector</i>
SVM	<i>Support Vector Machine</i>
UFES	Universidade Federal do Espírito Santo
UML	<i>Unified Modeling Language</i>
VC	Visão Computacional
VP	Verdadeiro Positivo
YOLO	<i>You Only Look Once</i>

SUMÁRIO

1	INTRODUÇÃO	12
1.1	Apresentação	12
1.2	Trabalhos Relacionados	14
1.3	Objetivos	15
1.4	Estrutura do Trabalho	16
2	REFERENCIAL TEÓRICO	17
2.1	Introdução	17
2.2	Detecção de Objetos	17
2.2.1	<i>Convolutional Neural Networks</i>	17
2.2.2	R-CNN	18
2.2.3	<i>Fast R-CNN</i>	21
2.2.4	<i>Faster R-CNN</i>	22
2.2.5	SSD	23
2.2.6	YOLO	24
2.3	Rastreamento de Múltiplos Objetos	26
2.3.1	Filtro de Kalman	26
2.3.2	SORT	31
2.3.3	<i>Deep SORT</i>	33
3	PROPOSTA	37
3.1	Arquitetura do Sistema	37
3.2	Detecção de Objetos	38
3.3	Rastreamento de Objetos	40
3.3.1	SORT	40
3.3.2	<i>Deep SORT</i>	41
3.3.3	Implementação Própria	41
3.4	Extração de Informações para Controle de Tráfego	42
3.4.1	Estimativa da Velocidade	42
3.4.2	Contagem de Veículos	43
3.4.3	Densidade de Vias	44
4	RESULTADOS	45
4.1	Introdução	45
4.2	Recursos Computacionais	45
4.3	Avaliação dos Resultados	47

4.3.1	Métricas	47
4.3.1.1	Conceitos Básicos	47
4.3.1.2	Métricas de Associação	50
4.3.1.3	Métricas de Identificação	51
4.3.1.4	Métricas de Avaliação de Alta Ordem	51
4.3.1.5	Avaliação no conjunto de dados UA-DETRAC	54
4.4	Resumo	58
5	CONCLUSÃO E TRABALHOS FUTUROS	59
5.1	Conclusão	59
5.2	Temas a serem pesquisados	60
	REFERÊNCIAS	61

1 INTRODUÇÃO

1.1 Apresentação

Garantir a qualidade da mobilidade urbana tem sido um desafio enfrentado por pesquisadores e gestores de centros urbanos nas últimas décadas. De modo especial, no que se refere a questões envolvendo transporte, o crescimento da frota de veículos, a falta de infra-estrutura urbana e as limitações existentes nas soluções para o planejamento e controle do tráfego têm resultado em uma série de danos ao meio ambiente e à população no geral (MAGAGNIN, 2008).

Segundo Maga e Hass (1960), as emissões veiculares são uma das principais causas dos problemas relacionados à poluição do ar. Dados retirados da pesquisa de Araújo (ARAÚJO, 2016), referente às emissões médias de poluentes na rede municipal da Grande Vitória, mostram a participação significativa das emissões veiculares se comparados às demais fontes de poluentes, como por exemplo, o setor industrial minero-siderúrgico, o setor de logística e outros.

Dados do Departamento de Trânsito do Espírito Santo (DETRAN-ES, 2016; DETRAN-ES, 2020), mostram que entre os anos de 2009 a 2019, a frota veicular particular cresceu em 74,77% no estado, compondo, ao final de 2019, cerca de 1.844.605 veículos. Esse crescimento, além de agravar a emissão de poluentes de origem veicular, resulta em um aumento proporcional da complexidade na gestão do tráfego urbano.

Nesse contexto, surge a necessidade de serem exploradas alternativas para a gestão eficiente da mobilidade urbana. Paralelamente, fatores como o aumento considerável da quantidade de câmeras instaladas e a redução do custo associado a este tipo de *hardware*, contribuem para a viabilidade de utilizar tais sistemas para obter informações do tráfego urbano (SILVA, 2017).

Valente (2019) reitera sobre a utilização de câmeras para esse fim, ao lembrar que, atualmente, diversos centros urbanos já possuem sistemas de monitoramento com câmeras de alta resolução para auxiliar na segurança e no trânsito.

Dessa forma, com base nos fatos apresentados anteriormente, a proposta deste trabalho é abordar o problema do tráfego urbano, aplicando técnicas de detecção e rastreamento de objetos baseadas em Visão Computacional (VC)¹.

¹ Visão Computacional é uma área da ciência responsável pela forma como um computador pode processar o ambiente à sua volta, extraindo informações significativas a partir de imagens capturadas

Nesse contexto, a detecção de objetos pode ser entendida como a tarefa da VC que trata da detecção de instâncias visuais de objetos de determinada classe (como por exemplo, humanos, animais, ou veículos) em imagens digitais (ZOU et al., 2019). Este trabalho propõe a utilização do detector de objetos *You Only Look Once* - YOLO (REDMON et al., 2016).

Com relação ao rastreamento de objetos, essa atividade pode ser definida como a tarefa de estimar a trajetória de um objeto em uma sequência de imagens (YILMAZ; JAVED; SHAH, 2006). Nesse sentido, a tarefa de rastreamento é dividida entre o rastreamento de um único objeto (SOT, do inglês *single object tracking*) e o rastreamento de múltiplos objetos (MOT, do inglês *multiple object tracking*).

Segundo Ciaparrone et al. (2020), no SOT as características do objeto que se deseja rastrear são conhecidas de modo que o rastreador deve, além de coletar a posição e as dimensões do objeto, diferenciá-lo de outros objetos semelhantes. Ao contrário do SOT, o MOT realiza a detecção e classificação de um grupo de objetos em uma ou mais categorias (como por exemplo, carros, pedestres e animais), sem nenhum conhecimento prévio das características ou quantidade desses objetos. Para isso, além da posição e dimensões desses objetos, no MOT o rastreador deve atribuir a cada detecção um identificador (ID) único de forma a distingui-lo de outras detecções de mesma classe.

Como esta pesquisa busca avaliar o desenvolvimento de um sistema de detecção e rastreamento de objetos aplicado ao tráfego urbano, foram avaliados apenas algoritmos de MOT. Dos métodos aplicados temos, o Filtro de Kalman (KALMAN, 1960) (*i*), o *Simple Online and Realtime Tracking* - SORT (BEWLEY et al., 2016) (*ii*) e o *Simple Online and Realtime Tracking with a Deep Association Metric* - Deep SORT (WOJKE; BEWLEY; PAULUS, 2017) (*iii*).

Assim, este trabalho busca analisar a problemática de mobilidade urbana através do desenvolvimento de um sistema capaz de localizar, classificar e rastrear veículos em vias públicas. Para isso foram verificados diferentes modelos do *framework* de detecção de objetos YOLOv5 combinados com os algoritmos de rastreamento, Filtro de Kalman, SORT e Deep SORT. Por fim, foi analisado o desempenho dos conjuntos detector-rastreador no conjunto de dados UA-DETRAC ², apresentada por Wen et al. (2020) que será melhor detalhada na Subseção 4.2.

por câmeras de vídeo, sensores, *scanners* entre outros dispositivos (MILANO; HONORATO, 2010)

² Disponível em <<https://detrac-db.rit.albany.edu>>

1.2 Trabalhos Relacionados

Segundo Zou et al. (2019), a detecção de objetos é uma importante área da visão computacional que tem como objetivo, fornecer ao computador a capacidade de localizar e classificar objetos em vídeos ou imagens. Com o desenvolvimento de novas técnicas de *Deep Learning*, foi possível expandir a aplicação da visão computacional para diversas aplicações da vida real, como por exemplo, a robótica, controle de veículos autônomos, análise de imagens médicas entre outras. No tópico da mobilidade urbana, a detecção de objetos tem sido aplicada para diversas tarefas, como por exemplo a detecção de veículos, ciclistas e pedestres em imagens (KHOSHELHAM, 2021).

Cortez (2022), aplica técnicas de VC e *Deep Learning* - DL (tradução livre, aprendizado profundo) para a detecção de veículos, combinadas a técnicas clássicas de *background subtraction* (tradução livre, subtração de fundo) para desenvolver um sistema de controle de tráfego inteligente. O sistema busca trazer mais inteligência aos semáforos de tempo fixo, possibilitando o controle dinâmico do tempo de acionamento das luzes, a partir da análise em tempo real do fluxo de veículos. Além do controlador semaforico, neste trabalho também é desenvolvido um sistema *web* capaz de realizar o gerenciamento dos controladores, além de coletar estatísticas sobre o tráfego. No desenvolvimento dessa solução foram aplicados os algoritmos de detecção de objetos YOLOv3, YOLOv4 e MobileNetV2, em que, através do sistema *web* citado anteriormente, o funcionário responsável pela programação de um dado controlador poderia optar por qual algoritmo aplicar em um dado semáforo. Para o rastreamento de objetos, foi desenvolvida uma solução própria baseada na utilização da distância euclidiana como método de associação.

Em um outro estudo, Silva (2017) propõe o desenvolvimento de um modelo eficiente e de baixo custo computacional, capaz de realizar a coleta de dados de trânsito de forma automática e em tempo real. No modelo em questão, focou-se na robustez do modelo contra diferentes situações de oclusão e iluminação. Para a realização da detecção de objetos, o modelo em questão busca uma solução de baixo custo computacional, aplicando técnicas de subtração de fundo combinadas com Funções de Haar e Árvores de Decisão. Para o rastreamento de objetos, é aplicado o método de Senior et al. (2006), que realiza a associação das detecções entre *frames*, utilizando como métrica a menor distância entre centroides das detecções no instante de tempo atual e anterior. Quanto a extração de dados, o sistema desenvolvido é capaz de coletar informações como a velocidade média estimada das detecções, a quantidade de veículos que atravessaram uma determinada área de interesse e o sentido dos veículos na via.

Por outro lado Jaswanth, Karri e Venkataraman (2020), busca por soluções para o problema da mobilidade urbana, aplicando técnicas de detecção que compõem o estado da arte em conjunto de técnicas clássicas para o rastreamento. Entretanto, além de não terem sido utilizadas técnicas de rastreamento envolvendo redes neurais convolucionais (CNN, do inglês *convolutional neural network*) e *deep learning*, métricas valiosas para o estudo e controle do tráfego urbano não foram exploradas, como por exemplo, a taxa de ocupação de vias, a velocidade média e quantidade de veículos.

De mesmo modo Wang, Ozcan e Sharma (2017), utilizam técnicas do estado da arte para a detecção de veículos. Entretanto, o estudo foca na utilização de detectores de dois estágios, e assim como no trabalho realizado por Jaswanth, Karri e Venkataraman (2020), variáveis de controle do tráfego urbano não são exploradas.

Embora Chowdhury, Biplob e Uddin (2018) implemente a verificação e cálculos de taxa de ocupação de vias, neste trabalho a detecção é realizada por meio de abordagens clássicas da visão computacional. Do mesmo modo, para o rastreamento, a abordagem utilizada também foi clássica, sendo implementado o rastreamento de veículos por meio do filtro de Kalman.

Apesar de todos esses esforços, do ponto de vista da VC, ainda há muito o que se explorar sobre a problemática do tráfego urbano. Nesse sentido, este trabalho busca investigar o comportamento de rastreadores de objetos pouco explorados, como por exemplo, o SORT e o *Deep SORT*, analisando combinações desses métodos com o detector de objetos YOLO. Por fim, este trabalho também busca contribuir com a utilização de métricas de relevância para o controle do tráfego urbano, como por exemplo a velocidade estimada dos veículos e a densidade de vias na aplicação a ser desenvolvida.

1.3 Objetivos

Objetivo Geral

- O objetivo dessa pesquisa aplicada é desenvolver um sistema capaz de realizar a detecção e o rastreamento de veículos em vias públicas através das imagens geradas pelas câmeras de videomonitoramento.

Objetivos Específicos

Os objetivos específicos definidos a fim de alcançar o objetivo geral, são listados a seguir:

- Desenvolver um sistema capaz de realizar a detecção em tempo real de veículos em imagens e vídeos captados em vias públicas;
- Desenvolver um sistema para rastrear veículos detectados em imagens sequenciais;
- Extrair informações de velocidade média estimada, taxa de ocupação e quantidade de veículos em imagens e vídeos captados em vias públicas;
- Avaliar a eficiência do sistema na base UA-DETRAC;
- Investigar a eficiência dos rastreadores utilizados no sistema e compará-los.

1.4 Estrutura do Trabalho

O presente trabalho está estruturado da seguinte maneira:

- **Introdução:** este capítulo tem como objetivo contextualizar o tema deste trabalho, apresentando suas justificativas, trabalhos relacionados e objetivos;
- **Referencial Teórico:** este capítulo dedica-se a explicar a fundamentação teórica necessária para a realização deste trabalho;
- **Proposta:** este capítulo é dedicado à apresentação da solução proposta para o problema em estudo;
- **Resultados:** neste capítulo são apresentadas as métricas de desempenho e os resultados obtidos após os experimentos;
- **Conclusão e Projetos Futuros:** no capítulo final deste trabalho são apresentadas as conclusões e os trabalhos futuros.

2 REFERENCIAL TEÓRICO

2.1 Introdução

Neste capítulo serão explicados os conceitos teóricos necessários para a construção deste trabalho. O capítulo se inicia revisando o conceito de *Convolutional Neural Networks* - CNNs (tradução livre, redes neurais convolucionais) e em seguida, busca explorar a sua aplicação, analisando alguns métodos de detecção de objetos que aplicam essas redes. Por fim, são explorados os conceitos de rastreamento de objetos, analisando o processo de rastreamento através do Filtro de Kalman, SORT e *Deep SORT*.

2.2 Detecção de Objetos

2.2.1 *Convolutional Neural Networks*

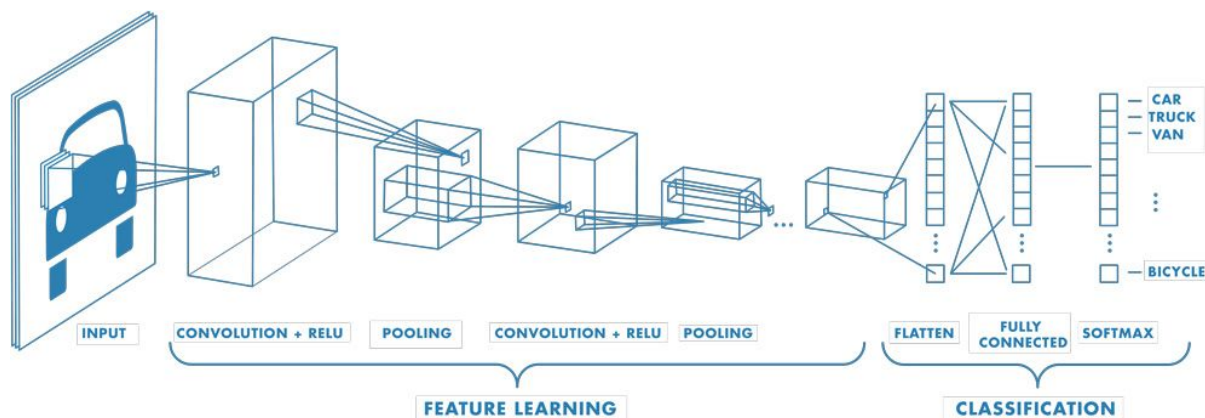
As CNNs são arquiteturas biologicamente inspiradas capazes de serem treinadas e aprenderem representações invariantes à escala, translação, rotação e transformações afins (SOUZA et al., 2020). Segundo Voulodimos et al. (2018), as CNNs são compostas principalmente por três tipos de camadas: (i) a camada de convolução, (ii) a camada de *pooling* e (iii) as camadas *fully connected* (tradução livre, totalmente conectadas). Essas três camadas, quando combinadas, formam uma estrutura de classificação de objetos ilustrada pela Figura 1.

- **Camada de convolução (*convolution layer*):** A camada de convolução utiliza diversos *kernels* (tradução livre, filtros) para convoluir uma imagem ou *feature maps* (tradução livre, mapas de características), gerando no processo novos *feature maps*;
- **Camada de *pooling* (*pooling layer*):** A camada de *pooling*, por sua vez, é responsável por reduzir a dimensionalidade (altura x largura) da entrada, realizando um processo de *downsampling* (tradução livre, subamostragem) dos dados e, consequentemente, a perda de parte da informação transmitida para camada seguinte. No entanto, em alguns casos, essa perda pode ser benéfica para a rede, uma vez que essa perda reduz o esforço computacional necessário por parte das camadas seguintes,

além de evitar o *overfitting* (tradução livre, sobreajuste) da rede (VOULODIMOS et al., 2018);

- **Camadas totalmente conectadas (*fully connected layers*):** A *fully connected layer* é responsável pelo processamento em alto nível da rede neural, isto é, por realizar tarefas como por exemplo, a classificação de objetos. Para isso, esta camada possui uma série de nós ou unidades interconectadas, de modo que o mapa de características recebido por esta camada passa por uma série de funções de ativação, para que, ao final da camada, essas características sejam convertidas em um vetor de características unidimensional, de tamanho igual a quantidade de classes a serem classificadas, e onde cada posição do vetor corresponde ao conjunto de características fornecidas à rede para a classe em questão.

Figura 1 – Representação das camadas existentes nas CNNs



Fonte: MathWorks (2022).

A fim de compor o sistema de detecção, foram analisados diferentes métodos de detecção de objetos. Esta seção busca comparar esses sistemas, apontando suas evoluções e analisando seu desempenho.

2.2.2 R-CNN

Segundo Girshick et al. (2014), a *Region-based Convolutional Network* (R-CNN, do inglês, Redes Neurais Convolucionais Baseadas em Região) é um detector de dois estágios compostos por três módulos:

- **Módulo principal:** O módulo principal é responsável por obter as propostas de região através do método de busca seletiva (do inglês, *selective search*), ilustrado

pela Figura 2. Segundo Uijlings et al. (2013), o método de busca seletiva define as propostas de região através dos seguintes procedimentos:

1. É realizada a sobre-segmentação da imagem, ou seja, sua divisão em pequenos grupos;
2. É feito o agrupamento dos dois grupos de maior similaridade dentro de uma vizinhança, formando então um novo grupo e recalculando a similaridade entre esse grupo e a nova vizinhança;
3. O item dois é então repetido iterativamente, reduzindo o critério de similaridade para a junção dos grupos, até que reste um único grupo na imagem;
4. Por fim, todos os grupos obtidos são verificados como candidatos a objetos.

Figura 2 – Obtenção de Propostas de Regiões *via Selective Search*

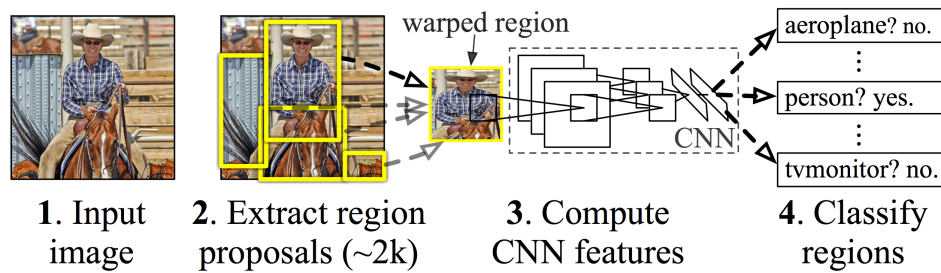


Fonte: Uijlings et al. (2013).

- **Módulo secundário:** Por sua vez, o módulo secundário é composto por uma CNN que extrai um vetor de características de dimensão fixa de cada uma dessas regiões.
- **Módulo terciário:** Por fim, o módulo terciário recebe esse vetor de características e os aplica a um conjunto de *Support Vector Machines* - SVMs (tradução livre, máquinas de vetores-suporte) (HEARST et al., 1998), realizando a classificação das regiões propostas.

A Figura 3 resume os módulos apresentados anteriormente através de um *pipeline*, de modo que: a segunda etapa do *pipeline* representa a obtenção das propostas de região, a terceira

Figura 3 – Pipeline de detecção da R-CNN



Fonte: Girshick et al. (2014).

representa a extração de características dessas regiões e por fim, a quarta representa a utilização das SVMs para a classificação das regiões propostas.

Segundo Girshick (2015), os principais problemas da R-CNN são:

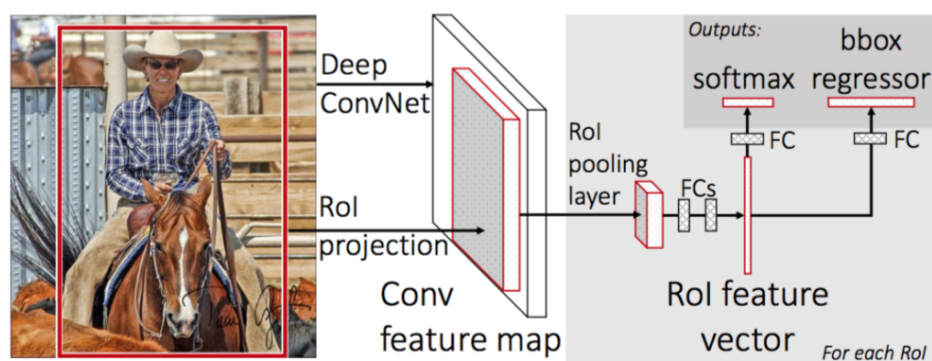
- O treinamento é um pipeline multiestágio: a R-CNN primeiro aperfeiçoa uma CNN sobre as propostas de objetos através de uma função de perda logarítmica. Em seguida, ela ajusta SVMs para as *features* (tradução livre, características) da CNN. Essas SVMs atuam como detectores, substituindo o classificador *softmax* aprendido durante o aperfeiçoamento. No terceiro estágio do treinamento, os regressores das *bounding-boxes* (tradução livre, caixas delimitadoras) são aprendidos;
- O treinamento é custoso em relação ao uso de memória e tempo de processamento: para SVMs e para o treinamento do regressor das *bounding-boxes*, *features* são extraídas a cada proposta de objeto de cada imagem e armazenadas na memória física. Dessa forma, para redes profundas, como por exemplo a VGG-16 (SIMONYAN; ZISSERMAN, 2014), o processo de treinamento leva até 2.5 dias com as 5 mil imagens do conjunto de dados VOC07 (EVERINGHAM et al., 2007).
- A detecção dos objetos é lenta em comparação a outros modelos de detecção: para uma unidade de processamento gráfico (GPU, do inglês *graphics processing unity*), a detecção com a estrutura de rede convolucional VGG-16 leva em média 47 segundos por imagem.

Dessa forma, a aplicação da R-CNN no contexto do projeto proposto torna-se não aconselhável, uma vez que demanda por um algoritmo de detecção que seja capaz de ser aplicado em tempo real.

2.2.3 Fast R-CNN

Como solução proposta para o problema destacado anteriormente, surge a *Fast R-CNN* (tradução livre, R-CNN Rápida) (GIRSHICK, 2015) ilustrada pela Figura 4. Ainda segundo Girshick (2015), a rede da *Fast R-CNN* recebe como entrada uma imagem e um conjunto de *regions of interest* - RoI (tradução livre, regiões de interesse). A rede então processa a imagem através de uma série de camadas de convolução e *max pooling*, extraindo o mapa de características da imagem fornecida. Em seguida, cada RoI é agrupado a um mapa de características e então fornecido a uma rede *fully connected* a fim de se obter o vetor de características. Por fim, essa rede se ramifica em duas saídas: a primeira que produz a classificação do vetor de características através da função de ativação *softmax*, retornando a probabilidade associada a cada detecção para as K classes disponíveis. E a segunda, que retorna um conjunto $K \times 4$, onde essas 4 colunas codificam as posições das *bounding-boxes* das K classes detectadas.

Figura 4 – Pipeline de detecção da *Fast R-CNN*



Fonte: Girshick (2015).

Ainda segundo Girshick (2015), a *Fast R-CNN* tem como principais contribuições:

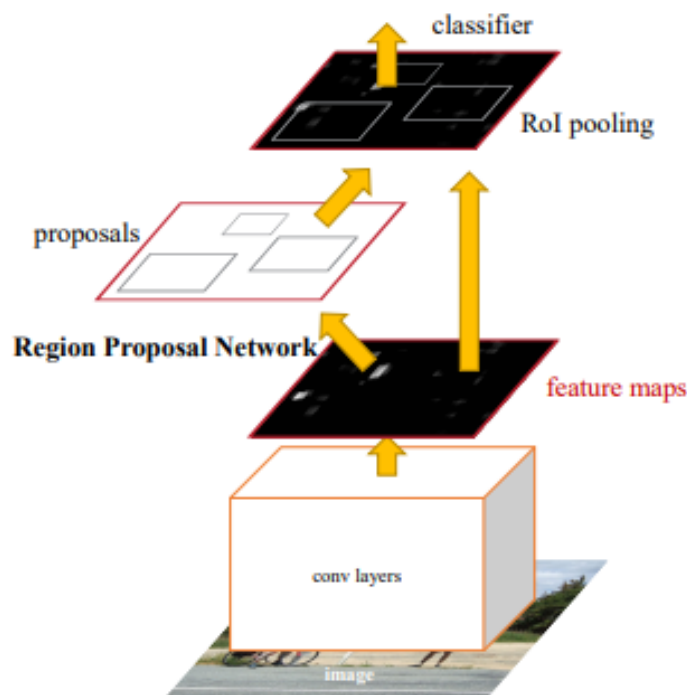
- O treinamento é feito em um pipeline de estágio único, ou seja, a *Fast R-CNN* possibilita o treinamento simultâneo do detector e do regressor de *bounding-boxes*;
- Valores mais altos na métrica de desempenho mAP se comparada à R-CNN no conjunto de dados VOC07, aumentado de 62,9% para 68,8%, em que mAP se refere ao valor médio da precisão de um detector;
- A detecção é mais rápida se comparada a R-CNN: para uma GPU Nvidia K40, a detecção com a estrutura de rede convolucional VGG-16 leva em média 0,15 segundos por imagem.

Dessa forma, apesar de ser consideravelmente mais rápida que a R-CNN, a aplicação da *Fast* R-CNN não é aconselhada, uma vez que o tempo médio entre detecções continua elevado para uma aplicação em tempo real. Isso decorre do fato da velocidade de detecção dessa rede estar limitada pelo tempo gasto na obtenção das regiões de interesse (ZOU et al., 2019).

2.2.4 *Faster* R-CNN

Segundo Adarsh, Rathi e Kumar (2020), a *Faster* R-CNN é uma variante da *Fast* R-CNN (GIRSHICK, 2015), no entanto, com um menor tempo de detecção. A principal diferença deste modelo é a aplicação de uma técnica chamada rede de proposta de região (RPN, do inglês *region proposal network*) (REN et al., 2015). Esta técnica tem como função acelerar a obtenção das propostas de regiões, substituindo o algoritmo do *selective search*. A arquitetura proposta pela *Faster* R-CNN pode ser melhor entendida através da Figura 5.

Figura 5 – Rede unificada da *Faster* R-CNN



Fonte: Ren et al. (2015).

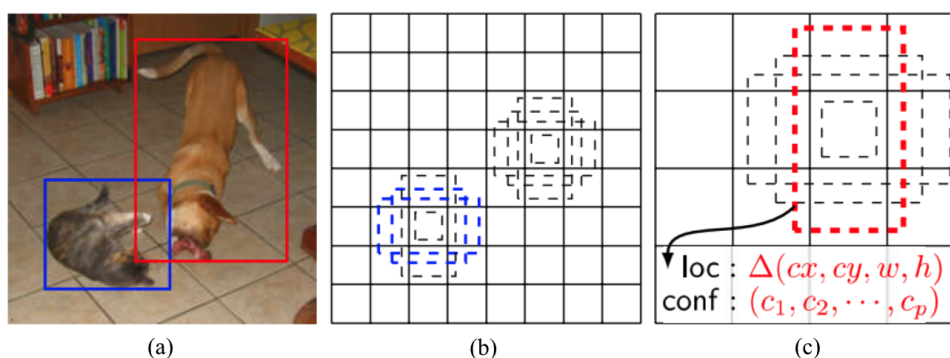
Segundo Ren et al. (2015), para uma estrutura de rede convolucional VGG-16, a *Faster* R-CNN leva em média 198ms para realizar, tanto a proposta de regiões, quanto detecção de objetos. Comparativamente, a *Fast* R-CNN necessita de em média 1.830ms para realizar

a mesma operação. Essa diferença, ainda que expressiva, não é suficiente para a aplicação deste detector em problemas que demandem por soluções em tempo real como é o caso deste projeto. Dessa forma, apesar das RPN melhorarem o tempo médio de detecção dos objetos em uma imagem, esse tempo ainda é grande se comparado aos de detectores de único estágio, como o YOLO e o SSD.

2.2.5 SSD

O detector de disparo único (SSD, do inglês *single shot detector*), é um detector de estágio único capaz de obter altas taxas de detecção e ainda ser significativamente mais preciso que outros modelos do mesmo tipo (LIU et al., 2016). Este detector reconstrói a estrutura da VGG-16, descartando as redes *fully connected* (ALBAWI; MOHAMMED; AL-ZAWI, 2017) e as substituindo por CNNs. A Figura 6 exemplifica o processo de detecção realizado pelo SSD.

Figura 6 – Detector de objetos SSD. (a) Imagem com *bounding-boxes* representando as detecções obtidos pelo detector, (b) Mapa de *features* 8x8, (c) Mapa de *features* 4x4



Fonte: Liu et al. (2016).

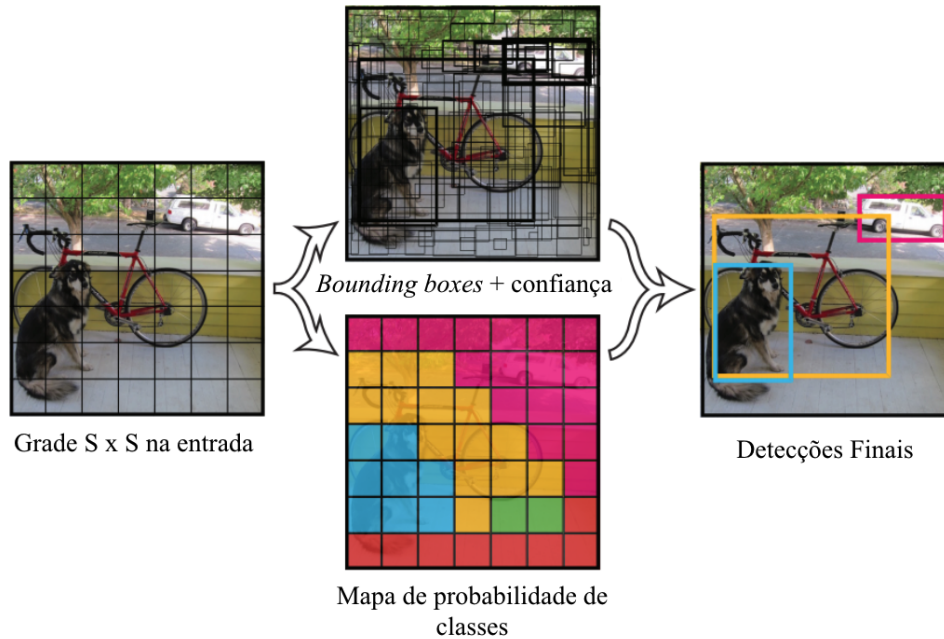
Nota: Modificado pelo autor.

Como citado anteriormente, alguns dos destaques desse detector são seus altos valores de mAP, acima de 70%, e alta velocidade de detecção, tendo em média de 16ms entre as detecções para o conjunto de dados MS COCO (LIN et al., 2014) e utilizando uma GPU *Titan X*. Nesse sentido o SSD torna-se uma opção viável para solucionar o problema da pesquisa, uma vez que atinge altas velocidades de detecção enquanto mantém resultados competitivos de mAP.

2.2.6 YOLO

Segundo Redmon et al. (2016), o YOLO é um detector de estágio único que utiliza *framework* Darknet (REDMON, 2013–2016) para realizar a extração de *features* e é pré-treinado no *dataset* ImageNet (RUSSAKOVSKY et al., 2015). O grande diferencial entre este detector e os demais baseados em propostas de região é que, diferentemente de outros métodos de detecção, como por exemplo, a R-CNN e suas variações, a YOLO aplica uma única rede neural a toda a imagem. Dessa forma, conforme ilustrado pela Figura 7, a rede aplicada divide a imagem em uma grade $S \times S$, onde cada célula dessa grade é responsável apenas pela predição dos objetos contidos nela. Assim, cada uma dessas células faz a previsão de B *bounding-boxes* e de seus respectivos intervalos de confiança.

Figura 7 – Modelo de detecção do detector de objetos YOLO



Fonte: Redmon et al. (2016).

Nota: Traduzido pelo autor.

Esses valores de confiança ($\text{Pr}(\text{Object})$) refletem o quão certa a rede neural acredita estar sobre a *bounding-box* B conter um objeto, e o quão precisa ela acredita ser a detecção desse objeto. A confiança da detecção pode ser definida pela Equação 2.1:

$$\text{Pr}(\text{Object}) \cdot \text{IoU}_{\text{pred}}^{\text{truth}} \quad (2.1)$$

Onde, novamente, $\text{Pr}(\text{Object})$ se refere a probabilidade da célula conter determinado

objeto, e a $\text{IoU}_{\text{pred}}^{\text{truth}}$, é a *intersection over union* - IoU (tradução livre, interseção sobre união) entre a *bounding-box* de *ground truth* (tradução livre, verdade fundamental) e a *bounding-box* de predição.

Cada *bounding-box* consiste em 5 previsões: x, y, w, h e o intervalo de confiança. Onde o par (x, y) corresponde as coordenadas do centro da *bounding-box* relativa à delimitação de determinada célula. A largura e a altura da detecção em relação ao tamanho da imagem, expressas pelo w e h , (do inglês, *width* e *height*), e o valor da confiança representa a IoU entre as *bounding-boxes* prevista e de *ground-truth* ($\text{Pr}(\text{Object}) \cdot \text{IoU}_{\text{pred}}^{\text{truth}}$) (REDMON et al., 2016).

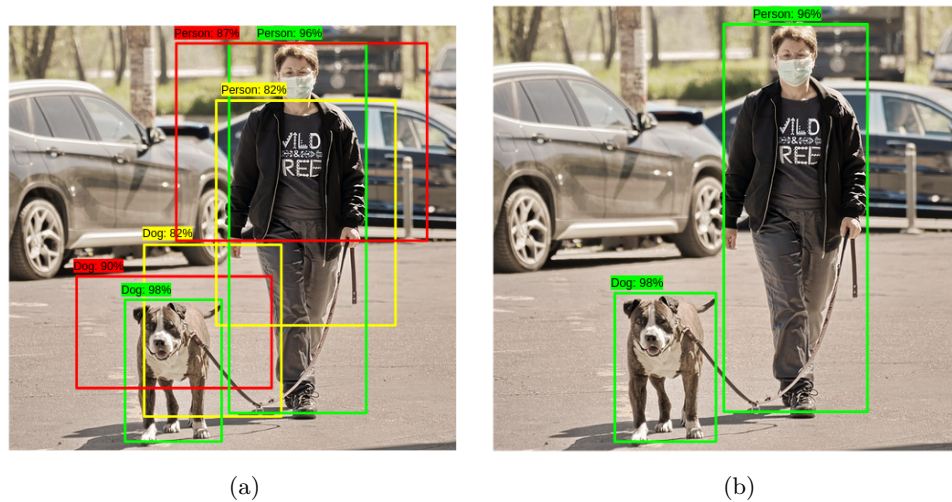
Como a YOLO modela a detecção como um problema de regressão, cada célula também possui um *appearance vector* para cada classe (C) de objetos a ser detectado. Dessa forma as previsões passam a ser codificadas como um vetor com dimensão igual ao valor descrito pela Equação 2.2:

$$S \times S \times (B \cdot 5 + C) \quad (2.2)$$

Por fim, como são definidas diversas *bounding-boxes*, é necessário aplicar o processo de *Non-maximal Suppression* (tradução livre, Supressão do Não Máximo). Esse algoritmo ordena as detecções de forma decrescente de acordo com seus respectivos intervalos de confiança. Se duas detecções tiverem sobreposição superior a 50% na mesma classe, apenas a de maior confiança é mantida, assim removendo possíveis detecções duplicadas (REDMON et al., 2016). A Figura 8 exemplifica a aplicação do algoritmo.

Dentre as principais características deste sistema, destaca-se o seu baixo tempo entre detecções. A exemplo da YOLOv4 (BOCHKOVSKIY; WANG; LIAO, 2020), esses valores chegam a aproximadamente 15ms para uma GPU Tesla V100 no *dataset* MS COCO. Nesse sentido o YOLO torna-se recomendável para solucionar o problema da pesquisa, uma vez que atinge altas velocidades de detecção se comparado a outros métodos (BOCHKOVSKIY; WANG; LIAO, 2020). Assim, pelo seu desempenho em sistemas de tempo real e pela maior acurácia quando comparada a outros métodos de detecção, optou-se por utilizar a YOLO para compor o sistema de detecção. Neste trabalho será utilizada a YOLOv5, treinada com a base de imagens MS COCO em 80 classes distintas.

Figura 8 – Exemplificação algoritmo de *Non-maximal Suppression*: (a) detecção sem a aplicação do algoritmo e (b) detecção com a aplicação do algoritmo



Fonte: Singh (2020).

2.3 Rastreamento de Múltiplos Objetos

Segundo Luiten et al. (2021), o MOT pode ser definido como a tarefa de rastrear múltiplos objetos em uma sequência de imagens ou vídeo, associando os objetos rastreados ao longo do tempo de acordo com sua identidade. Conforme visto na Subseção 1.2, algoritmos de MOT tem sido de extrema importância para a construção de sistemas cada vez mais eficientes na coleta de informações valiosas para o controle do tráfego, como por exemplo, a velocidade estimada de veículos, a densidade de vias e a quantidade de veículos. Dessa forma, esta seção busca revisar os algoritmos que serão aplicados neste trabalho, fornecendo o embasamento teórico necessário para a replicação dos resultados obtidos.

2.3.1 Filtro de Kalman

Segundo Yilmaz (2006), no campo do rastreamento de objetos, o filtro de Kalman é utilizado para implementar métodos de rastreamento por pontos, estimando o estado de um sistema linear através de uma distribuição gaussiana. Esse filtro é composto por duas etapas, previsão e atualização. A etapa de previsão utiliza o modelo de estado para prever o novo estado das variáveis. Enquanto o de atualização utiliza cálculos estatísticos para estimar o novo estado do objeto.

Pegoretti et al. (2021) propõem um melhor entendimento do algoritmo através do seguinte exemplo, considere um veículo se movendo em um plano de coordenadas (u, v) . Considere que a velocidade do veículo é constante e sua aceleração é nula em qualquer ponto do

plano (u, v) . Para cada instante de tempo k , a medição da posição pode ser representada através da Equação 2.3.

$$\mathbf{z}_k = \begin{bmatrix} u_k \\ v_k \end{bmatrix} \quad (2.3)$$

Dada a informação da posição a cada instante de tempo, e sabendo-se que o veículo segue a uma velocidade constante, podemos definir um modelo em espaço de estados para representar a dinâmica do veículo a cada instante de tempo k . Dessa forma, a posição e a velocidade do veículo são dadas por 2.4 e 2.5 respectivamente.

$$x_k = x_{k-1} + \Delta T \cdot \dot{x}_{k-1} \quad (2.4)$$

$$\dot{x}_k = \dot{x}_{k-1} \quad (2.5)$$

Onde, k e $k - 1$, representam, respectivamente, os estados de tempo atual e anterior. Assim, o vetor de estados \mathbf{x}_k que descreve a posição e a velocidade estimada do objeto no plano (u, v) , é dado pela Equação 2.6.

$$\mathbf{x}_k = \begin{bmatrix} u_k \\ \dot{u}_k \\ v_k \\ \dot{v}_k \end{bmatrix} \quad (2.6)$$

Em que u_k e v_k correspondem às coordenadas do veículo nos eixos horizontal e vertical respectivamente, com \dot{u}_k e \dot{v}_k representando as velocidades nesses mesmos eixos. Assim, podemos modelar o espaço de estados do sistema proposto através da Equação 2.7.

$$\begin{bmatrix} u_k \\ \dot{u}_k \\ v_k \\ \dot{v}_k \end{bmatrix} = \begin{bmatrix} 1 & \Delta T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta T \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_{k-1} \\ \dot{u}_{k-1} \\ v_{k-1} \\ \dot{v}_{k-1} \end{bmatrix} \quad (2.7)$$

Assumido que o modelo está suscetível a perturbações aleatórias no sistema (\mathbf{s}_k), podemos representá-lo por meio da Equação 2.8.

$$\mathbf{x}_k = \mathbf{F} \cdot \mathbf{x}_{k-1} + \mathbf{s}_k \quad (2.8)$$

Em que a Equação 2.9, representa a matriz de transição de estados (\mathbf{F}) do sistema.

$$\mathbf{F} = \begin{bmatrix} 1 & \Delta T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta T \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.9)$$

De mesmo modo, supondo que no instante k exista um ruído aleatório na medição (w_k), temos que a saída do sistema, ou seja, a saída no mesmo instante k , é dada pela Equação 2.10 abaixo.

$$\mathbf{z}_k = \mathbf{H} \cdot \mathbf{x}_k + \mathbf{w}_k \quad (2.10)$$

Onde a Equação 2.11, representa o modelo de observação do sistema.

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (2.11)$$

Após a definição do modelo do espaço de estados, podemos seguir com a execução do algoritmo de forma iterativa para obter novas estimativas da posição e velocidade do objeto a cada novo instante k . Para isso, primeiro é executada a etapa de predição, representada pelas Equações 2.12 e 2.13 abaixo:

$$\tilde{\mathbf{x}}_k = \mathbf{F} \cdot \hat{\mathbf{x}}_{k-1} \quad (2.12)$$

$$\tilde{\mathbf{P}}_k = \tilde{\mathbf{F}} \cdot \hat{\mathbf{P}}_{k-1} \cdot \mathbf{F}^T + \hat{\mathbf{Q}} \quad (2.13)$$

Onde $\tilde{\mathbf{x}}_k$ é o vetor de estado predito no instante atual (k) e, $\hat{\mathbf{x}}_{k-1}$, representa o vetor de estado estimado no instante de tempo anterior ($k-1$). De mesmo modo, $\tilde{\mathbf{P}}_k$ representa a matriz de covariância do estado predito no instante atual (k) e $\hat{\mathbf{P}}_{k-1}$ é a matriz de covariância do estado estimada no instante anterior ($k-1$). Assim, essas matrizes de covariância modelam as incertezas associadas às previsões e estimativas do vetor de estado. Em particular, a matriz de ruído do processo ($\hat{\mathbf{Q}}$) pode ser descrita pela Equação 2.14 abaixo:

$$\hat{\mathbf{Q}} = \begin{bmatrix} \Delta T^4/4 & \Delta T^3/2 & 0 & 0 \\ \Delta T^3/2 & \Delta T^2 & 0 & 0 \\ 0 & 0 & \Delta T^4/4 & \Delta T^3/2 \\ 0 & 0 & \Delta T^3/2 & \Delta T^2 \end{bmatrix} \sigma_s^2 \quad (2.14)$$

Onde, σ_s^2 é um parâmetro definido pelo projetista que modela a incerteza do modelo escolhido para a representação do sistema. Dessa forma, quanto maior o valor de σ_s^2 , mais impreciso é o modelo projetado, de modo que o vetor de estado predito $\tilde{\mathbf{x}}_k$ será menos relevante na obtenção do estado estimado $\hat{\mathbf{x}}_k$, determinado durante a etapa de atualização. Como condições iniciais do sistema ($k=0$) para os cálculos da etapa de predição, temos as Equações 2.15 e 2.16 abaixo:

$$\hat{\mathbf{P}}_0 = \mathbf{I} \quad (2.15)$$

$$\hat{\mathbf{x}}_0 = \mathbf{H}^T \cdot \mathbf{z}_0 \quad (2.16)$$

Em que \mathbf{z}_0 denota a posição inicial medida para o objeto.

Durante a etapa de atualização, isto é, após feita a predição do estado atual, é calculado o erro \mathbf{e}_k , ou seja, a diferença entre a medição das variáveis controladas do sistema (\mathbf{z}_k) obtidas através de algum sensor (no caso do sistema de rastreamento de objetos proposto, através do método de detecção de objetos YOLO), e as estimativa do estado anterior ($\mathbf{H} \cdot \tilde{\mathbf{x}}_k$). Assim, o erro associado a medição pode ser descrito através da Equação 2.17 abaixo:

$$\mathbf{e}_k = \mathbf{z}_k - \mathbf{H} \cdot \tilde{\mathbf{x}}_k \quad (2.17)$$

Em sequência, são calculados o ganho de Kalman (\mathbf{K}_k), o vetor de estados estimado ($\hat{\mathbf{x}}_k$), e a matriz de covariância estimada ($\hat{\mathbf{P}}_k$), através das Equações 2.18, 2.19 e 2.20 respectivamente.

$$\mathbf{K}_k = \tilde{\mathbf{P}}_k \cdot \mathbf{H}^T \cdot (\mathbf{H} \cdot \tilde{\mathbf{P}}_k \cdot \mathbf{H}^T + \hat{\mathbf{W}})^{-1} \quad (2.18)$$

$$\hat{\mathbf{x}}_k = \tilde{\mathbf{x}}_k + \mathbf{K}_k \cdot \mathbf{e}_k \quad (2.19)$$

$$\hat{\mathbf{P}}_k = \tilde{\mathbf{P}}_k - \mathbf{K}_k \cdot \mathbf{H} \cdot \tilde{\mathbf{P}}_k \quad (2.20)$$

Onde a covariância do ruído de observação (\mathbf{W}), é descrito pela Equação 2.21 abaixo:

$$\hat{\mathbf{W}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \sigma_w^2 \quad (2.21)$$

Sendo σ_w^2 , um parâmetro definido pelo projetista, a fim de aferir o nível de incerteza associado às medições do sistema (no caso do sistema proposto, a incerteza associada a medição da posição do veículo).

Finalmente, a estimativa da posição do objeto no instante de tempo k ($\hat{\mathbf{z}}_k$), podem ser obtidas através da equação 2.22 abaixo:

$$\hat{\mathbf{z}}_k = \mathbf{H} \cdot \hat{\mathbf{x}}_k \quad (2.22)$$

Onde, para o sistema em questão, $\hat{\mathbf{z}}_k$, denota o vetor contendo a posição estimada do objeto no instante k . É importante ressaltar que durante o processo de rastreamento de objetos, as posições medidas (\mathbf{z}_k) do objeto podem não ser obtidas devido a algum problema no detector, como por exemplo, oclusões ou falhas na detecção. Nesses casos, os cálculos para atualização do estado são realizados parcialmente, de modo que apenas a etapa de predição é realizada, uma vez que não é possível obter o erro (\mathbf{e}_k), já que não se tem o valor do vetor de posições (\mathbf{z}_k). Dessa forma, é considerado que o estado estimado é igual ao predito, isto é, conforme a Equação 2.23 abaixo:

$$\hat{\mathbf{x}}_k = \tilde{\mathbf{x}}_k \quad (2.23)$$

2.3.2 SORT

Segundo Bewley et al. (2016), o SORT trata-se de um algoritmo de rastreamento focado na realização do MOT em tempo real, buscando tratar de forma eficiente, o problema da associação *frame a frame* de detecções. Para isso, o SORT combina os algoritmos do filtro de Kalman e algoritmo Húngaro (do inglês, *Hungarian Algorithm*), para realizar a previsão da movimentação dos objetos e a associação dessas previsões com as novas detecções.

Essa construção minimalista traz benefícios ao algoritmo, como por exemplo, uma taxa de atualização de até 260 Hertz (Hz), isto é, uma velocidade de execução até 20 vezes maior que outros rastreadores que compõem o estado da arte (BEWLEY et al., 2016). Outra vantagem desta arquitetura, é que apesar de ter uma maior velocidade, sua acurácia se mantém similar a esses outros métodos.

Em contrapartida, por aplicar uma arquitetura de rastreamento simples, este algoritmo é mais suscetível a problemas de re-identificação de objetos se comparada a outros métodos de rastreamento, o que pode limitar sua aplicabilidade caso essa re-identificação seja relevante no problema em questão.

De modo a resumir sua operação, podemos dividir o funcionamento deste algoritmo em quatro etapas:

- **Detecção:** a qualidade da detecção é um dos fatores chave para se obter um alto desempenho durante o rastreamento de objetos, o SORT aplica a *Faster R-CNN* como *framework* de detecção. No entanto, apesar deste método de detecção fornecer como saída tanto a *bounding-box* quanto o *appearance vector* associado a essa detecção, o SORT realiza o rastreamento utilizando apenas as informações da posição horizontal (u) e vertical (v), a área (s), e suas respectivas velocidades (\dot{u}), (\dot{v}) e (\dot{s}), além do *aspect ratio* (a) (tradução livre, proporção) da *bounding box* dessa detecção para realizar o seu rastreamento.
- **Predição:** o sistema é então modelado considerando que a velocidade observada durante o deslocamento entre dois *frames* é aproximadamente constante, o que é razoável dado a premissa de que este método é aplicável a sistemas de tempo real,

logo o espaço de tempo entre dois *frames* é demasiado pequeno. Assim, o vetor de estado de um alvo é dado pela Equação 2.24:

$$\mathbf{x} = [u, v, s, a, \dot{u}, \dot{v}, \dot{s}]^T \quad (2.24)$$

E assim temos que a matriz de transição de estados (\mathbf{F}), e a matriz de observação (\mathbf{H}) são, respectivamente iguais as Equações 2.25 e 2.26 respectivamente:

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 0 & 0 & \Delta T & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \Delta T & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \Delta T \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.25)$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad (2.26)$$

Dessa forma, tanto o estado predito ($\tilde{\mathbf{x}}_k$), quanto o estado estimado ($\hat{\mathbf{x}}_k$) num dado instante k , são obtidos de forma otimizada através das equações de estado do filtro de Kalman.

- **Associação:** para realizar a associação das detecções e os alvos (do inglês, *targets*) rastreados, para cada uma das detecções obtidas em um dado instante de tempo k , o SORT compara a informação geométrica da *bounding-box* da detecção com a dos alvos existentes no instante anterior $k - 1$, calculando a IoU entre esses dois valores. Dessa forma, é então calculada a matriz de custo \mathbf{C}_{cost} de dimensões $\mathcal{T} \times \mathcal{D}$, onde \mathcal{D} corresponde ao número de detecções no instante k , e \mathcal{T} ao número de alvos no instante $k - 1$, onde o custo ($c_{i,j}$) associado ao alvo (i) e a detecção (j) é dado pela Equação 2.27 abaixo:

$$c_{i,j} = 1 - \frac{\text{área}_{i \cap j}}{\text{área}_i + \text{área}_j - \text{área}_{i \cap j}} \quad (2.27)$$

Em seguida, a associação entre os pares de alvos e detecções é resolvida através do algoritmo Húngaro (KUHN, 1955, apud BEWLEY et al., 2016), de modo que os pares são divididos em 3 vetores que correspondem às seguintes categorias: alvos com detecções associadas, alvos sem detecções associadas e detecções sem alvos associados.

Com essas informações, o algoritmo agora é capaz de realizar a atualização, remoção ou criação de novos alvos.

- **Atualização, criação e remoção de alvos:** após a etapa de associação, são aplicadas diferentes rotinas para cada um dos vetores obtidos.

Para cada alvo sem uma detecção associada, é incrementada uma propriedade nesse alvo que contabiliza o número de vezes que esse alvo não pôde ser associado a uma detecção. Se esse valor for maior que um mínimo determinado pelo usuário, esse alvo é então removido. Caso contrário, nenhuma outra alteração é feita para esse alvo.

Para cada alvo com uma detecção associada, é feita a atualização do estado desse alvo no instante de tempo k , com base nas medições da detecção associada a esse alvo.

Por fim, para cada detecção sem um alvo associado, é criado um novo alvo com os parâmetros dessa detecção.

2.3.3 *Deep* SORT

Segundo Wojke, Bewley e Paulus (2017), o *Deep* SORT é um algoritmo para MOT, que incrementa o SORT, substituindo as métricas aplicadas à etapa de associação, com a combinação das métricas de movimentação e aparência, da detecção. Dessa forma, o *Deep* SORT aumenta a robustez contra oclusões e perdas de detecções, uma vez que o rastreador agora é capaz de re-identificar os alvos a partir de suas características visuais (*appearance vector*), extraídas através de uma CNN.

Ainda segundo Wojke, Bewley e Paulus (2017), a fim de entender melhor o funcionamento desse algoritmo, podemos dividi-lo nas seguintes etapas:

- **Detecção:** assim como no SORT, durante a etapa de detecção são extraídas as informações de posição (*bounding boxes*) e confiança das detecções. No entanto, a fim de obtermos informações sobre a aparência dos objetos detectados, é integrado a essa etapa um *appearance descriptor* (tradução livre, descritor de aparência), isto é, uma CNN responsável pela extração do vetor de aparência (\mathbf{r}) de cada uma das detecções, onde $\|\mathbf{r}\| = 1$. Portanto, a entrada desta CNN é a imagem gerada pelo bounding-box.
- **Predição:** de forma similar, o *Deep* SORT realiza a modelagem em espaço de estados considerando que a velocidade de deslocamento entre dois *frames* é aproximadamente constante. No entanto, diferentemente do SORT, o vetor de estados é definido

utilizando as informações da posição horizontal (u) e vertical (v) da detecção, o *aspect ratio* (a) (tradução livre, proporção de tela), a altura (h) da *bounding box*, e suas respectivas velocidades (\dot{u} , \dot{v} , \dot{a} e \dot{h}), em coordenadas da imagem. Dessa forma cada alvo é associado ao seguinte vetor de estados (\mathbf{x}), representado pela Equação 2.28 abaixo:

$$\mathbf{x} = [u, v, a, h, \dot{u}, \dot{v}, \dot{a}, \dot{h}]^T \quad (2.28)$$

Onde o *aspect ratio* (a) é dado pela Equação 2.29:

$$a = \frac{w}{h} \quad (2.29)$$

Em que w se refere à largura da *bounding box*.

Portanto, temos que a matriz de transição de estados (\mathbf{F}) e a matriz de observação (\mathbf{H}), são dadas pelas Equações 2.30 e 2.31 respectivamente:

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 0 & 0 & \Delta T & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \Delta T & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \Delta T & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \Delta T \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.30)$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad (2.31)$$

Assim, de modo similar ao que foi visto na Subseção 2.3.2, tanto o estado predito ($\tilde{\mathbf{x}}_{\mathbf{k}}$), quanto o estado estimado ($\hat{\mathbf{x}}_{\mathbf{k}}$) são obtidos de forma otimizada através das equações de estado do filtro de Kalman, 2.12 e 2.19, respectivamente.

- **Associação:** diferentemente do SORT, o *Deep SORT* busca resolver o problema da associação através da combinação de métricas associadas à movimentação e aparência das detecções. A informação, relacionada à movimentação, é aplicada através da distância de Mahalanobis (MAESSCHALCK; JOUAN-RIMBAUD; MASSART, 2000) quadrada entre o estado predito pelo filtro de Kalman e as novas detecções, descrita pela Equação 2.32 abaixo:

$$d^{(1)}(i, j) = (\mathbf{d}_j - \mathbf{y}_i)^T \mathbf{S}_i^{-1} (\mathbf{d}_j - \mathbf{y}_i) \quad (2.32)$$

Onde a projeção em espaço de estados do i -ésimo alvo é denotada por $(\mathbf{y}_i, \mathbf{S}_i)$, e a j -ésima *bounding-box* por \mathbf{d}_j . A distância de Mahalanobis leva em conta a incerteza da estimativa através da medição do desvio padrão das *bounding-boxes* da detecção e do alvo. Assim, é possível excluir associações improváveis limitando à distância de Mahalanobis a valores dentro do intervalo de confiança de 95% ($t^{(1)}$), calculada através da inversa da distribuição Qui-quadrado (χ^2). Essa verificação pode ser denotada pela Equação 2.33 abaixo:

$$b^{(1)}(i, j) = 1[\mathbf{d}_{(i,j)}^1 \leq t^{(1)}] \quad (2.33)$$

Em que o indicador $b^{(1)}(i, j)$ é 1, se a distância de Mahalanobis entre o i -ésimo alvo e a j -ésima detecção for menor ou igual a um limiar $t^{(1)}$. Onde, para o espaço de estados proposto, temos que $t^{(1)} = 9,4877$.

Por outro lado, a fim de incrementar a informação relacionada à aparência das detecções à lógica de associação, é criado um conjunto (\mathcal{R}_t) contendo os $L_t = 100$ últimos *appearance vectors* associados a cada alvo (t). Esse conjunto pode ser descrito pela Equação 2.34 abaixo:

$$\mathcal{R}_t = \{\mathbf{r}_t^{(i)}\}_{t=1}^{L_t} \quad (2.34)$$

Em seguida, é calculada a menor distância cosseno entre a j -ésima detecção e o i -ésimo alvo, descrita pela Equação 2.35:

$$d^{(2)}(i, j) = \min\{1 - \mathbf{r}_j^T \mathbf{r}_t^{(i)} | \mathbf{r}_t^{(i)} \in \mathcal{R}_i\} \quad (2.35)$$

Combinadas, essas métricas se complementam por servirem a diferentes aspectos do problema de associação. A distância de Mahalanobis, por exemplo, é útil para a tarefa de associação dado um curto intervalo de tempo. Por outro lado, o uso de informações relacionados a aparência da detecção se mostra útil na re-identificação de objeto e em cenários de oclusão.

Por fim, para a construção da matriz de custo, as métricas citadas anteriormente são combinadas através da soma ponderada. Dessa forma, os elementos da matriz de custo pode ser descritos pela Equação 2.36 abaixo:

$$c_{i,j} = \lambda d^{(1)}(i, j) + (1 - \lambda) d^{(2)}(i, j) \quad (2.36)$$

Onde o $c_{i,j}$ representa o custo associado entre um alvo (i) e uma detecção (j) e $d^{(1)}$ e $d^{(2)}$ representam, respectivamente, a distância de Mahalanobis e a distância cosseno entre o mesmo par de alvo e detecção. Por fim λ , representa um hiperparâmetro determinado pelo usuário, para determinação do peso associado a cada distância.

Dessa forma, o *Deep SORT* busca uma maior resiliência contra oclusões através da combinação de métricas associadas à posição e aparência dos objetos detectados. No entanto, a aplicação de uma CNN para a extração do vetor de aparência resulta em um maior custo computacional se comparado ao SORT. Ainda assim, o *Deep SORT* se mostra aplicável para o cenário de detecção em tempo real, alcançando uma média de 40 FPS nos experimentos realizados por Wojke, Bewley e Paulus (2017), quando aplicado ao conjunto de dados do MOT16 (MILAN et al., 2016) com a GPU Nvidia GeForce GTX 1050.

3 PROPOSTA

Este capítulo se dedica a fornecer os detalhes de implementação do sistema proposto. As seções estão divididas de forma a viabilizar a reprodução dos resultados obtidos, ressaltando as particularidades encontradas nas etapas de implementação da: (i) arquitetura do sistema, (ii) rotinas de detecção de objetos, (iii) rotinas de rastreamento de objetos, (iv) sistemas para extração de informações relevantes para o controle de tráfego, e (v) sistema de avaliação.

3.1 Arquitetura do Sistema

Segundo Burbeck (1992), a arquitetura *Model-View-Controller* - MVC (tradução livre, modelagem-visualização-controle), realiza a divisão de uma aplicação em três camadas, a camada de modelagem (do inglês, *model layer*), a camada de apresentação (do inglês, *view layer*) e a camada de controle (do inglês, *control layer*), em que essas camadas possuem as seguintes finalidades:

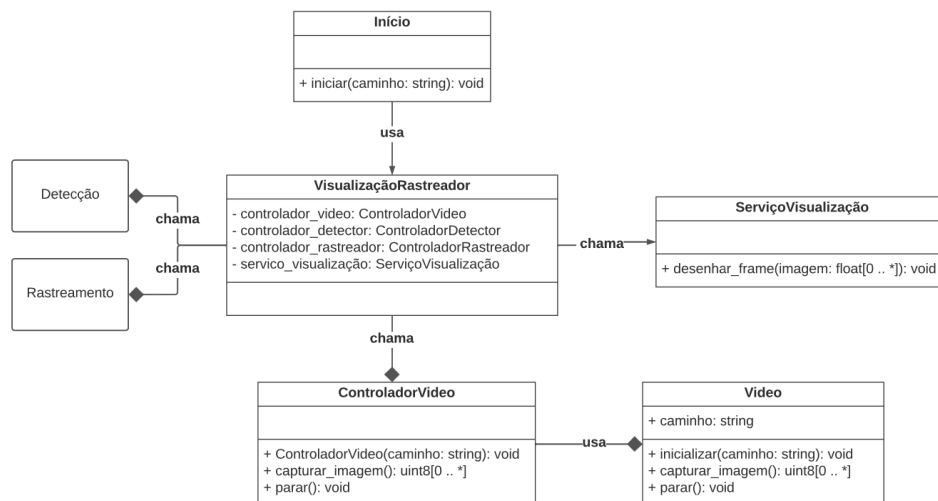
- **Camada de Modelagem:** realiza a descrição das regras de negócio do sistema através de modelos ou entidades, que por sua vez, são responsáveis por manipular os dados da aplicação. Dessa forma, a camada de modelagem é responsável por manipular as entradas fornecidas por usuários ou outras entidades, convertendo-as em informações úteis ao sistema.
- **Camada de Apresentação:** gerencia a saída gráfica da aplicação, ou seja, fornece ao usuário o retorno visual gerado por suas interações, como por exemplo a apresentação de um gráfico ou texto explicativo.
- **Camada de Controle:** possibilita a troca de informações entre as camadas anteriores, realizando eventuais conversões entre os dados. Essa camada é responsável pelo desacoplamento entre as regras de negócio do sistema e sua interface gráfica.

Dessa forma, por facilitar a implementação de múltiplos algoritmos, bem como sua reutilização em diferentes aplicações, optou-se por aplicar a arquitetura MVC para o desenvolvimento deste sistema.

3.2 Detecção de Objetos

Por se tratar de um método de detecção de objetos rápido e de fácil implementação, optou-se por utilizar o YOLOv5¹ (JOCHER, 2020), pré-treinado no conjunto de dados MS COCO (LIN et al., 2014) em 80 classes. De modo a compor os sistema de detecção, inicialmente foi desenvolvida uma rotina para a visualização de imagens e vídeos através da biblioteca de visão computacional, *OpenCV* (BRADSKI, 2000). A Figura 9 ilustra o funcionamento dessa rotina através de um diagrama feito com a linguagem de modelagem unificada² (do inglês *Unified Modeling Language* - UML).

Figura 9 – Diagrama UML simplificado da arquitetura desenvolvida para visualização de imagens



Fonte: Produção do próprio autor.

Nota: Para a simplificação do diagrama, os fluxos de rastreamento e detecção de veículos foram omitidos.

Para a implementação do detector, antes foram criados as entidades de *Detecção* e *Detector*, essas entidades tem como objetivo garantir o desacoplamento da *framework* além de possibilitar a conversão dos dados de detecção para os algoritmos de rastreamentos e serviços implementados.

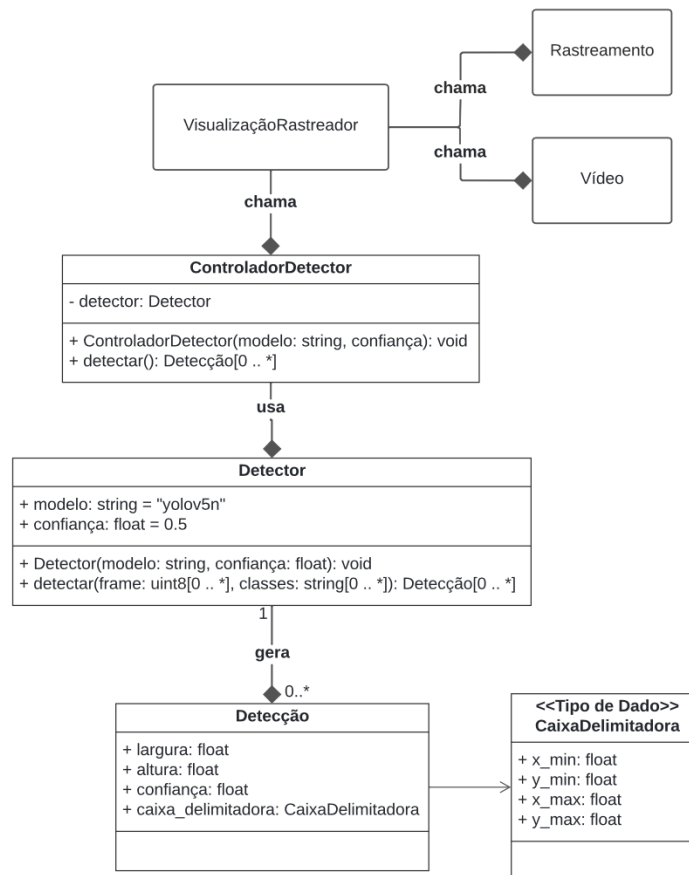
Quando instanciado, o *Detector*, recebe como parâmetros o tipo do modelo da YOLOv5 que se deseja aplicar (dentre yolov5n a yolov5x) e um valor real representando o limiar para filtragem das detecções. Essa classe também expõe um método *detectar*, capaz de

¹ Disponível em <<https://github.com/ultralytics/yolov5>>

² A UML é uma linguagem-padrão para a elaboração da estrutura de projetos de software. Essa linguagem utiliza de série de elementos gráficos, como por exemplo, retângulos, setas e outros, para descrever os elementos de uma aplicação, bem como suas interações, facilitando a visualização, documentação, especificação e construção de artefatos de sistemas de *software* (BOOCH, 2006)

retornar uma lista de **Detecções**, em que cada instância de **Detecção** possui informações referentes à segmentação semântica do objeto detectado, a confiança da detecção e sua *bounding box*. O diagrama simplificado da rotina de detecção de objetos pode ser visto na Figura 10.

Figura 10 – Diagrama UML da arquitetura desenvolvida para detecção de objetos



Fonte: Produção do próprio autor.

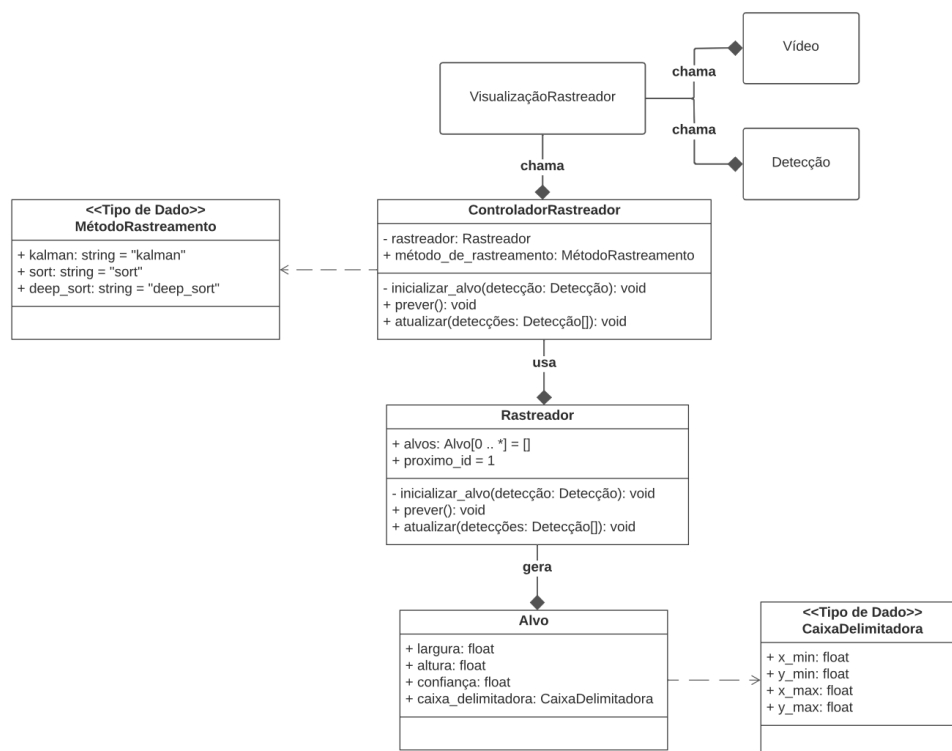
Nota: Para a simplificação do diagrama, os fluxos de visualização dos *frames* e rastreamento dos veículos foram omitidos.

Essa rotina também é capaz de receber um parâmetro opcional **classes**, usado para limitar quais tipos de objetos serão detectados. Dessa maneira, durante a realização dos experimentos, as classes de detecção foram limitadas a classes de veículos, como por exemplo, carros, ônibus, caminhões e vans.

3.3 Rastreamento de Objetos

Para o desenvolvimento das rotinas de rastreamento, e prevendo eventuais diferenças entre os algoritmos de rastreamento, foram criadas as entidades de **Rastreador** e **Hipótese**. Essas entidades têm como objetivo, abstrair a passagem de dados do detector para o algoritmo de rastreamento escolhido, fazendo as conversões necessárias através de métodos específicos. Para isso, quando instanciado, o **Rastreador** recebe a informação do algoritmo de rastreamento a ser aplicado, dentre, o SORT e o *Deep* SORT. De forma simplificada, esta entidade também expõe os métodos **prever** e **atualizar**, referentes às etapas de predição e atualização descritas na Seção 2.3. Para um melhor entendimento do fluxo realizado, o diagrama UML dessa rotina pode ser encontrado na Figura 11.

Figura 11 – Diagrama UML da arquitetura desenvolvida para o rastreamento de objetos



Fonte: Produção do próprio autor.

Nota: Para a simplificação do diagrama, os fluxos de visualização dos *frames* e detecção dos veículos foram omitidos.

3.3.1 SORT

Para a utilização do algoritmo de rastreamento SORT, foi desenvolvido um adaptador responsável pela conversão dos valores de entrada e a saída deste algoritmo, para o formato

utilizado pelo restante da aplicação.

Como exemplo das conversões realizadas por este adaptador, temos a alteração no formato das *bounding boxes* obtidas pelo sistema de detecção, possibilitando que a detecção seja aplicada corretamente sem grandes alterações no algoritmo.

3.3.2 *Deep* SORT

De mesmo modo, para a implementação do *Deep* SORT, foi necessário o desenvolvimento de um adaptador, que permite a conversão dos dados de rastreamento e detecção para o formato usado pelo algoritmo.

Como citado na Subseção 2.3.3, este algoritmo de rastreamento utiliza um extrator de características para realizar a re-identificação de objetos. Por conta disso, foram utilizados pesos pré-treinados³ no conjunto de dados NVIDIA AI *City Challenge* 2019 (NAPHADE et al., 2019; TANG et al., 2019), voltada para a detecção de veículos.

Para a aplicação desses pesos no sistema desenvolvido, foi necessária a refatoração da classe `Detector`, possibilitando que estes fossem passados como um parâmetro opcional ao construtor da classe. Dessa forma, durante a chamada do método `detectar`, o extrator de características é aplicado a cada região detectada, possibilitando a extração do vetor de características.

3.3.3 Implementação Própria

Além dos algoritmos SORT e *Deep* SORT, também foi feita uma implementação própria baseada no algoritmo SORT. Essa implementação buscou ser ainda mais simples que o SORT, realizando o rastreamento utilizando apenas as informações da posição horizontal (u) e vertical (v), bem como as velocidades nesses eixos, \dot{u} e \dot{v} , respectivamente. Assim, nessa implementação, o vetor de estados de um alvo pode ser descrito pela Equação 3.1 abaixo:

$$\mathbf{x} = [u, v, \dot{u}, \dot{v}]^T \quad (3.1)$$

³ Disponível em <https://github.com/abhyantrika/nanonets_object_tracking>

E dessa forma, a seguinte matriz de transição de estados (\mathbf{F}) e observação (\mathbf{H}), são descritas pelas equações 3.2 e 3.3 respectivamente:

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta T & 0 \\ 0 & 1 & 0 & \Delta T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.2)$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.3)$$

Outra alteração foi quanto à utilização da distância euclidiana como métrica de associação, substituindo a IoU no cálculo da matriz de custo (\mathbf{C}_{cost}). Dessa forma, os elementos da matriz de custo podem ser descritos através da Equação 3.4 abaixo:

$$c_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (3.4)$$

Onde, $c_{i,j}$ representa o custo (ou distância euclidiana) associado entre o i -ésimo alvo e a j -ésima detecção.

3.4 Extração de Informações para Controle de Tráfego

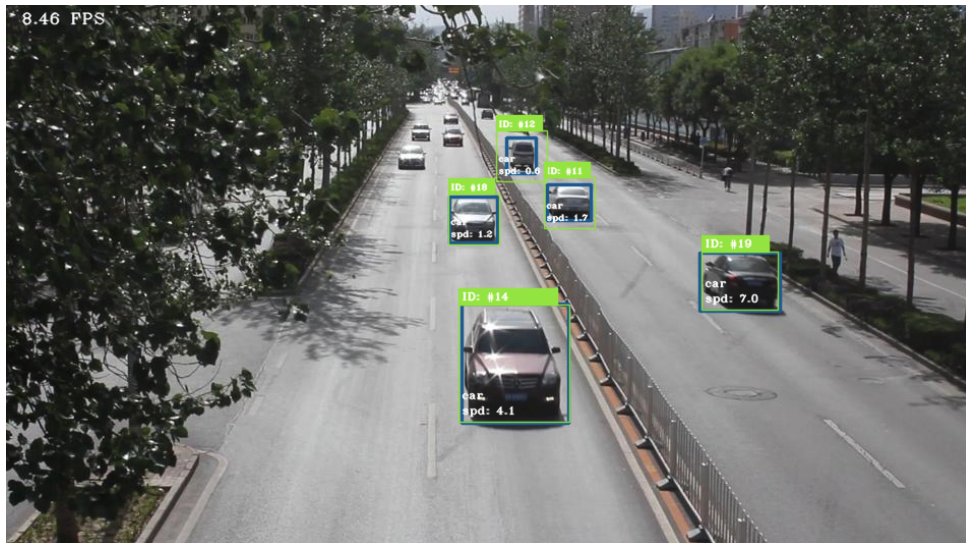
3.4.1 Estimativa da Velocidade

Para a obtenção da estimativa da velocidade foram utilizadas as velocidades em pixel por segundo, ilustradas através da Figura 12 e obtidas através das medições dos estados pelos rastreadores aplicados. Dessa forma, a velocidade em pixel (s) pode ser calculada conforme a Equação 3.5 abaixo:

$$s = \sqrt{\dot{u}^2 + \dot{v}^2} \quad (3.5)$$

Onde, \dot{u} e \dot{v} representam, respectivamente as velocidades nos eixos horizontal e vertical.

Figura 12 – Obtenção da velocidade estimada aplicada ao conjunto de dados UA-DETRAC.

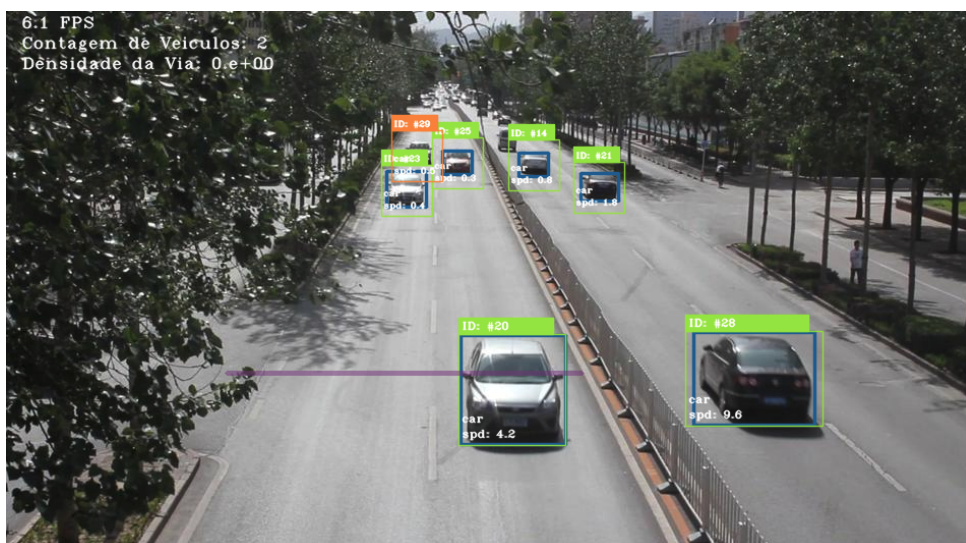


Fonte: Produção do próprio autor.

3.4.2 Contagem de Veículos

Para a realização da contagem de veículos foi adicionada uma funcionalidade de linha de contagem, em que após a seleção de dois pontos quaisquer em um dado *frame*, é adicionada uma linha à imagem, de modo que assim que um veículo cruzar essa linha será acrescido uma unidade ao contador de veículos. Este processo pode ser exemplificado através da Figura 13 abaixo:

Figura 13 – Processo de contagem de veículos aplicado ao conjunto de dados UA-DETRAC.



Fonte: Produção do próprio autor.

3.4.3 Densidade de Vias

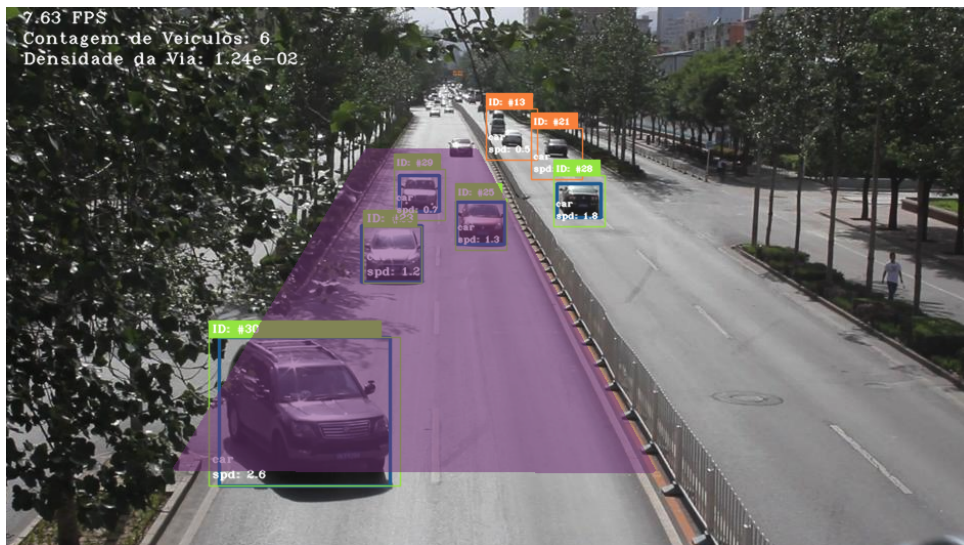
Para se obter a densidade de via, assumiu-se que a trepidação nas filmagens seria desprezível, e então foi desenvolvida uma rotina que possibilita a delimitação de uma região no vídeo, dada a seleção de três ou mais pontos. Em posse desses pontos, a densidade da via pode ser calculada conforme a Equação 3.6 abaixo:

$$d = \frac{n_{\text{carros}}}{L_{\text{via}}} \quad (3.6)$$

Onde, a densidade da via (d) é dada pela razão entre o número de carros (n_{carros}) na via e a extensão da via (L_{via}), calculados *frame a frame*.

A Figura 14, ilustra os resultados obtidos através da aplicação dessa rotina, em que a área em roxo representa a região selecionada para o cálculo da densidade da via.

Figura 14 – Processo de obtenção da densidade de via aplicado ao conjunto de dados UA-DETRAC.



Fonte: Produção do próprio autor.

4 RESULTADOS

4.1 Introdução

No presente capítulo são apresentados os resultados obtidos após a implementação do sistema. O capítulo inicia descrevendo os recursos computacionais aplicados no desenvolvimento deste trabalho, a arquitetura do sistema proposto, as métricas utilizadas para avaliar o desempenho do conjunto rastreador e detector, e os resultados obtidos. Por fim, é feita uma comparação entre os resultados deste trabalho com outras propostas encontradas durante a revisão da literatura.

4.2 Recursos Computacionais

Recursos de *Software*

Para a visualização das cenas e operações de visão computacional, foi utilizada a biblioteca de código aberto OpenCV (BRADSKI, 2000). As etapas de detecção e extração de *features* foram realizadas utilizando o *PyTorch* (PASZKE et al., 2017), *software* de código aberto desenvolvido pela *Meta AI*, destinado à construção de modelos de aprendizado profundo. Por fim, para a análise de resultados foi utilizada a ferramenta para avaliação de métricas de MOT, *TrackEval*¹ (LUITEN, 2020), modificado para possibilitar a avaliação no conjunto de dados UA-DETRAC.

Recursos de *Hardware*

O *hardware* necessário para o desenvolvimento deste trabalho se restringe a um computador com capacidade de processamento suficiente para a execução dos detectores e rastreadores de objetos analisados. Desta forma, o trabalho e experimentos realizados foram desenvolvidos em máquina pessoal com a seguinte configuração: (*i*) sistema operacional MacOS, distribuição Monterey versão 12.4; (*ii*) processador Intel Core i9 8-Core, 2.3GHz; (*iii*) memória RAM de 16 GB 2400 MHz DDR4; (*iv*) unidade de armazenamento de 500GB (disco rígido); (*v*) placa de vídeo Radeon Pro 560X, com 4 GB de memória dedicada.

Conjunto de dados UA-DETRAC

¹ Disponível em <<https://github.com/JonathonLuiten/TrackEval>>

Para a validação dos resultados de detecção e rastreamento, optou-se pela utilização do conjunto de dados UA-DETRAC (WEN et al., 2020). O DETRAC é composto por mais de 100 vídeos selecionados em mais de 10 horas de gravações, representando diversos cenários de tráfego, como por exemplo, rodovias, faixas de segurança e cruzamentos, localizados em 24 pontos diferentes das cidades de Pequim e Tanjin, China. Os vídeos foram gravados em diferentes condições de iluminação e ângulos de câmera, em uma taxa de quadros de 25 FPS e com uma resolução de 960×540 pixels. Este conjunto possui no total mais de 140.000 frames, totalizando 1,21 milhões de objetos rotulados, contendo características como, tipo, velocidade, *bounding box* e outros. As Figuras 15a a 15f exemplificam os diversos cenários avaliados pelo conjunto de dados.

Figura 15 – Exemplo de imagens pertencentes ao conjunto de dados UA-DETRAC



Fonte: Wen et al. (2020).

4.3 Avaliação dos Resultados

Esta seção é dedicada a detalhar as etapas de avaliação do sistema implementado. As subseções foram divididas de modo a ressaltar as particularidades encontradas durante a execução dos experimentos, dando ênfase: (i) às métricas aplicadas, (ii) ao validador escolhido e (iii) aos resultados obtidos.

4.3.1 Métricas

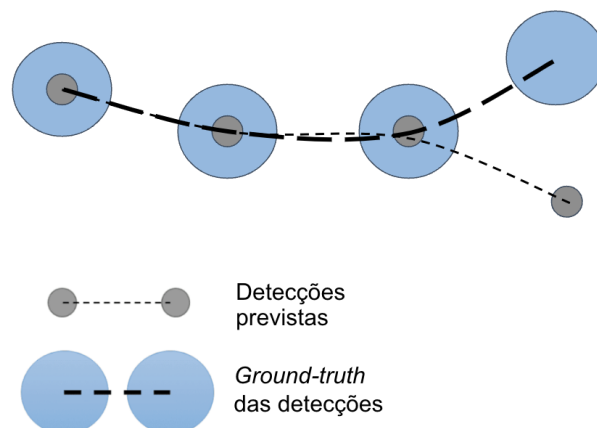
A fim de garantir um melhor entendimento dos resultados obtidos, nesta subseção será feita a descrição das métricas empregadas.

4.3.1.1 Conceitos Básicos

Segundo Leichter e Krupka (2013), os erros associados à classificação de um conjunto de predições e o *ground-truth* (GT) pode ser dividido em três categorias: (i) erros de detecção, (ii) erros de localização e (iii) erros de associação. Segundo Luiten et al. (2021), esses erros podem ser descritos como:

- **Erros de Detecção:** Ocorrem quando o detector realiza detecções que não existem no GT, ou quando o detector falha ao não conseguir realizar uma detecção existente no GT. Esse tipo de erro pode ser melhor entendido através da Figura 16 abaixo:

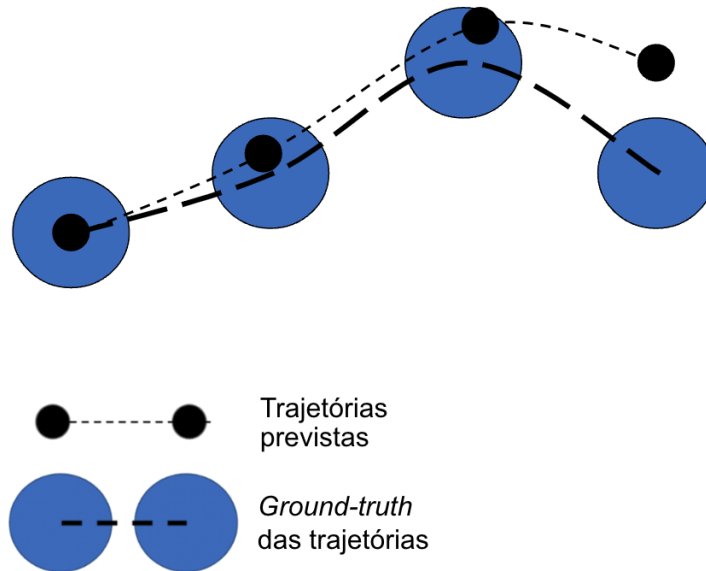
Figura 16 – Demonstração visual do conceito de erro de detecção



Fonte: Produção do próprio autor.

- **Erros de Localização:** Ocorrem quando as previsões feitas pelo rastreador não sobrepõem corretamente o seu GT. A Figura 17 ilustra esse tipo de erro:

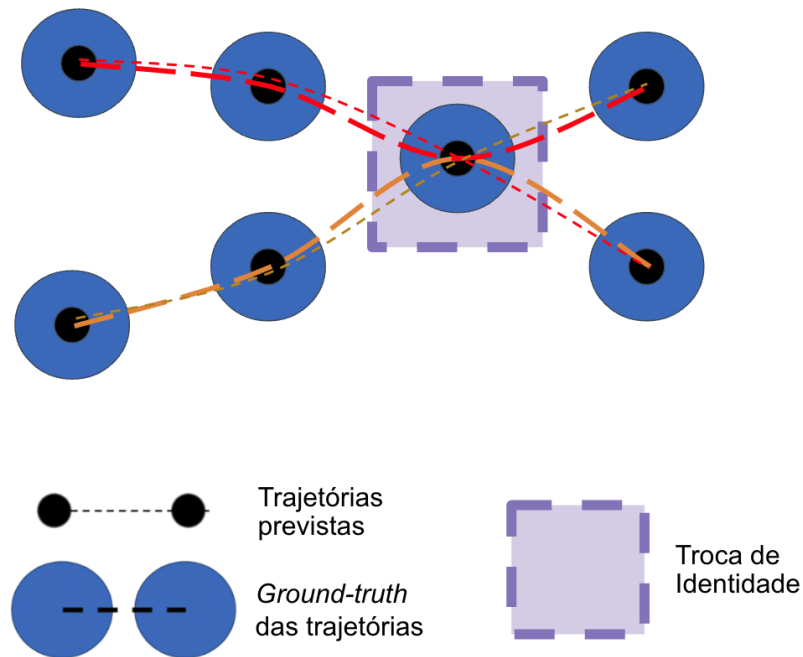
Figura 17 – Demonstração visual do conceito de erro de localização



Fonte: Produção do próprio autor.

- **Erros de Associação:** Ocorrem quando o rastreador associa a mesma identidade a duas previsões que possuem GTs diferentes, ou ainda, quando o rastreador associa duas identidades diferentes a duas previsões que deveriam ter o mesmo GT. Esse tipo de erro pode ser melhor entendido através da Figura 18 abaixo:

Figura 18 – Demonstração visual do conceito de erro de associação



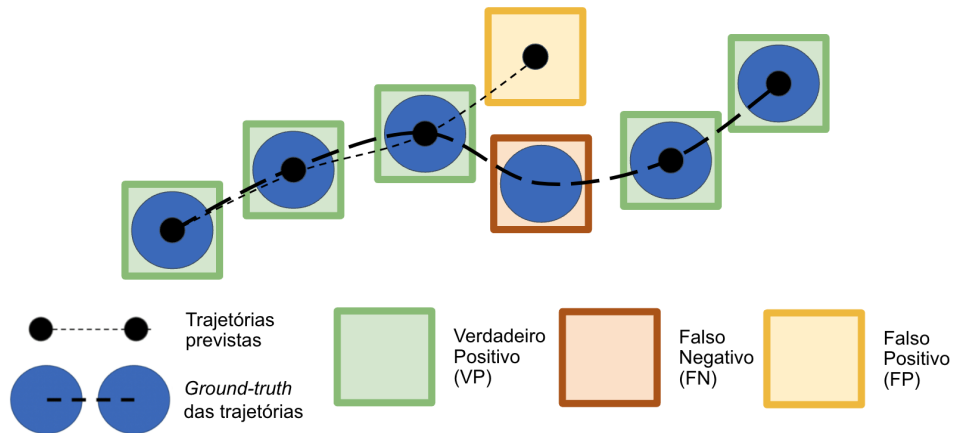
Fonte: Produção do próprio autor.

Dessa forma, podemos classificar a associação entre os valores previstos e GT como:

- **Verdadeiro Positivas (VP)**: Quando há uma associação correta entre uma previsão e um valor de GT.
- **Falso Positivas (FP)**: Por sua vez, uma associação é dito FP sempre que tivermos uma previsão sem um valor de GT associado.
- **Falso Negativas (FN)**: Por fim, uma associação é dita FN quando um valor de GT não tiver nenhuma previsão associada.

Os tipos de associação explicados anteriormente podem ser ilustrados através da Figura 19 abaixo:

Figura 19 – Demonstração visual dos conceitos de VP, FP e FN



Fonte: Produção do próprio autor.

Outro conceito relevante na análise da acurácia de um detector são as *identity switches* - (IDSW) (tradução livre, trocas de identidade). No contexto do MOT, uma IDSW ocorre quando um rastreador erroneamente troca a identidade de um objeto, ou ainda, quando um rastreador perde um alvo, reiniciando o seu rastreamento com uma identidade diferente.

A fim de estimar a resiliência dos rastreadores quanto aos três tipos de erros citados anteriormente, diversas métricas foram propostas e validadas em trabalhos anteriores. Das métricas utilizadas neste trabalho temos:

4.3.1.2 Métricas de Associação

Segundo Milan et al. (2016), a fim de mensurar a resiliência do rastreador quanto a possíveis erros de detecção e associação, é utilizada a função de avaliação *Multiple Object Tracking Accuracy* (MOTA), descrita pela Equação 4.1 abaixo:

$$\text{MOTA} = 1 - \frac{\sum_t (\text{FN}_t + \text{FP}_t + \text{IDSW}_t)}{\sum_t \text{GT}_t} \quad (4.1)$$

Em que, t indica o *frame* em que a métrica está sendo avaliada.

De forma complementar, a fim de avaliar o desempenho do rastreador, quanto a possíveis erros de localização em um *frame* t , é utilizada a função de pontuação *Multiple Object Tracking Precision* (MOTP), descrita pela Equação 4.2 abaixo:

$$\text{MOTP} = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t} \quad (4.2)$$

Em que c representa a quantidade de VPs no frame t e $d_{t,i}$ representa a sobreposição entre a *bounding box* de um alvo i com seu respectivo *ground truth*.

4.3.1.3 Métricas de Identificação

Segundo Luiten et al. (2021), para avaliar a associação entre as predições feitas pelo rastreador e o GT, são utilizadas as funções de avaliação Precisão, *Recall* e IDF1. Essas funções podem ser calculadas conforme as Equações 4.3, 4.4 e 4.5 abaixo:

$$\text{Precisão} = \frac{\text{VP}}{\text{VP} + \text{FP}} \quad (4.3)$$

$$\text{Recall} = \frac{\text{VP}}{\text{VP} + \text{FN}} \quad (4.4)$$

$$\text{IDF1} = \frac{\text{VP}}{\text{VP} + 0,5 \cdot \text{FN} + 0,5 \cdot \text{FP}} \quad (4.5)$$

4.3.1.4 Métricas de Avaliação de Alta Ordem

A fim de combinar os diferentes aspectos analisados durante a avaliação de um rastreador, Luiten et al. (2021) definem a métrica de *Higher Order Tracking Accuracy* - HOTA (tradução livre, acurácia de rastreamento de alta ordem). O HOTA, divide a tarefa de avaliar um rastreador em três tarefas, (*i*) localização, (*ii*) detecção e (*iii*) associação.

- **Localização:** A tarefa de localização busca medir o alinhamento espacial entre uma previsão e um GT. Para avaliar este alinhamento, é utilizada a interseção sobre união da região delimitada pelo GT e a predição, conforme visto na Equação 4.6 abaixo:

$$\text{Loc-IoU} = \frac{\text{área}_{\text{predição}} \cap \text{área}_{\text{ground truth}}}{\text{área}_{\text{predição}} \cup \text{área}_{\text{ground truth}}} \quad (4.6)$$

Dessa forma, a acurácia da localização (LocA) pode ser calculada como a Equação 4.7 abaixo:

$$\text{LocA} = \frac{1}{|\text{VP}|} \sum_{c \in \text{VP}} \text{Loc-IoU}(c) \quad (4.7)$$

Onde c é um valor VP de interesse.

- **Detecção:** Por sua vez, a tarefa da detecção busca medir o alinhamento entre os conjuntos de detecções realizadas e o GT de detecções. No entanto, como uma mesma detecção pode intersectar mais de um GT, é necessário utilizar o algoritmo Húngaro para realizar essa associação, através de um limiar de localização (α). Essa métrica pode ser entendida através da Equação 4.8 abaixo:

$$\text{Det-IoU} = \frac{\text{VP}}{\text{VP} + \text{FP} + \text{FN}} \quad (4.8)$$

Para calcular a acurácia associada à detecção, basta calcular o Det-IoU sobre todo o conjunto de dados. Dessa forma a acurácia associada a detecção (DetA) pode ser descrita através da Equação 4.9 abaixo:

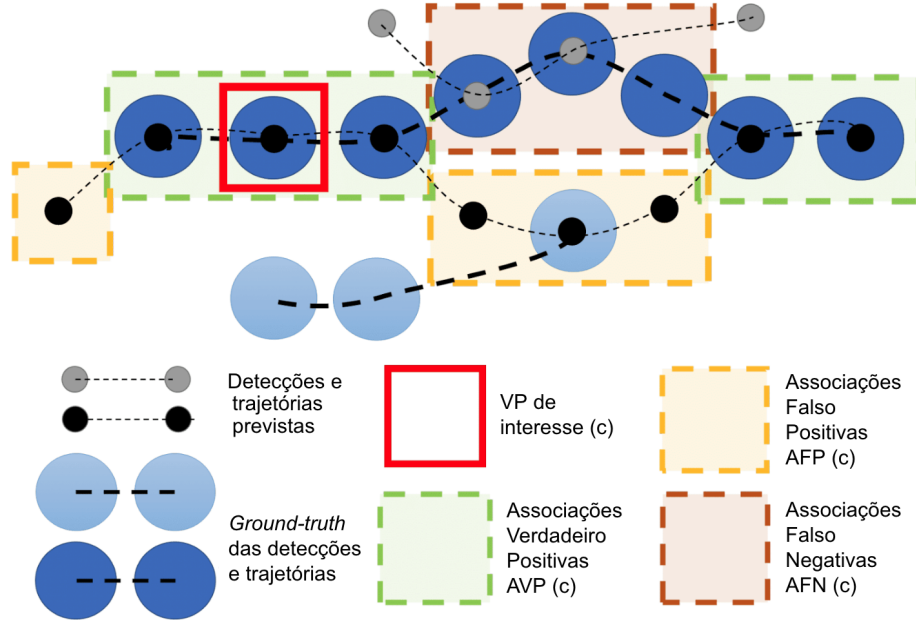
$$\text{DetA} = \text{Det-IoU} = \frac{\text{VP}}{\text{VP} + \text{FP} + \text{FN}} \quad (4.9)$$

- **Associação:** Por fim, a tarefa de associação busca medir o quão bem o rastreador consegue associar os dados da detecção a uma mesma identidade. Novamente, esta métrica pode ser representada através da interseção sobre união das associações conforme a Equação 4.10 abaixo:

$$\text{Ass-IoU}(c) = \frac{\text{AVP}}{\text{AVP} + \text{AFP} + \text{AFN}} \quad (4.10)$$

Em que, as Associações Verdadeiro Positivas (AVP), representam o conjunto de VPs que possuem a mesma identidade que um VP de interesse c . As Associações de Falso Positivos (AFP), são o conjunto de detecções que possuem a mesma identidade que c , mas que foram associadas incorretamente a um GT de identidade diferente da esperada, ou que não foram associadas a nenhum GT. Por fim, todas as demais detecções compõem o conjunto das Associações Falso Negativas (AFN). A Figura 20 ilustra os conceitos de AVP, AFP e AVP explicados anteriormente.

Figura 20 – Demonstração visual dos conceitos de AVP, AFP e AFN



Fonte: Luiten et al. (2021).

Nota: Traduzido pelo autor.

Assim, a acurácia média das associações (AssA) pode ser calculada através das Equações 4.11 e 4.12 abaixo:

$$\text{AssA} = \frac{1}{|\text{VP}|} \sum_{c \in \text{VP}} \text{Ass-IoU}(c) \quad (4.11)$$

$$\text{AssA} = \frac{1}{|\text{VP}|} \sum_{c \in \text{VP}} \frac{\text{AVP}}{\text{AVP} + \text{AFP} + \text{AFN}} \quad (4.12)$$

Essas métricas podem então ser combinadas no HOTA através das Equações 4.13 e 4.14 a seguir:

$$\text{HOTA}_\alpha = \sqrt{\text{DetA}_\alpha \cdot \text{AssA}_\alpha} \quad (4.13)$$

$$\text{HOTA} = \int_{0 < \alpha \leq 1} \text{HOTA}_\alpha \quad (4.14)$$

Onde α representa o limiar aplicado ao algoritmo Húngaro na associação dos dados de detecção e suas respectivas *ground-truths*.

4.3.1.5 Avaliação no conjunto de dados UA-DETRAC

A fim de comparar as combinações de detectores e rastreadores propostos, utilizou-se da rotina de avaliação de MOT, *TrackerEval*² modificada para suportar o conjunto de dados UA-DETRAC. Na Tabela 1 é possível visualizar os resultados obtidos na classificação dessas combinações:

² Disponível em <<https://github.com/JonathonLuiten/TrackEval>>

Tabela 1 – Resultados da análise de desempenho do rastreamento de carros no conjunto de dados UA-DETRAC.

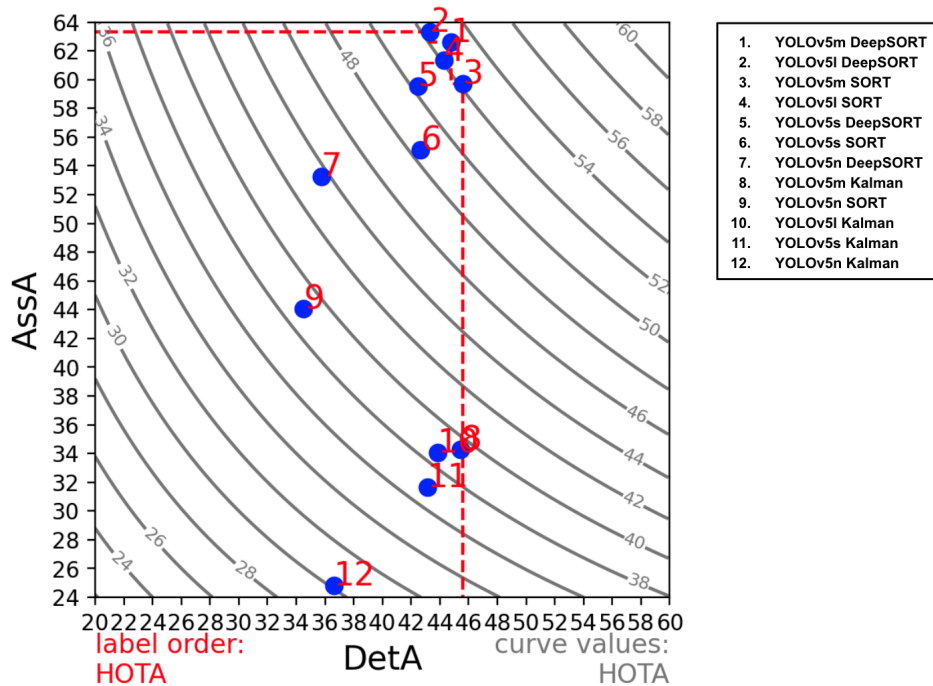
Modelo	VP	FP	FN	MOTA	MOTP	IDF1	DetA	AssA	HOTA	Execução (s)
YOLOv5n + Kalman	178243	244558	497111	35,863	82,120	32,462	36,611	24,808	29,897	4817,349
YOLOv5n + SORT	232754	145694	442600	35,730	81,272	44,174	34,499	44,066	38,681	4799,639
YOLOv5n + <i>Deep</i> SORT	290477	121102	384877	37,100	80,269	53,449	35,740	53,281	43,316	8669,427
YOLOv5s + Kalman	247569	332785	427785	39,100	83,588	39,431	43,108	31,663	36,689	11529,750
YOLOv5s + SORT	335852	203715	339502	41,641	82,606	55,288	42,624	55,115	48,181	11455,226
YOLOv5s + <i>Deep</i> SORT	382062	191250	293292	40,679	81,536	61,195	42,483	59,571	50,033	17646,324
YOLOv5m + Kalman	279083	358654	396271	39,992	84.864	42,508	45,452	34,281	39,172	48712,959
YOLOv5m + SORT	383519	221775	291835	43,545	83,560	59,895	45,586	59,712	51,808	48614,408
YOLOv5m + <i>Deep</i> SORT	416503	218859	258851	41,289	82,560	63,554	44,756	62,628	52,585	52277,146
YOLOv5l + Kalman	277680	391016	397674	34,706	85,326	41,320	43,830	34,057	38,376	71365,624
YOLOv5l + SORT	390721	246745	284633	38,988	84,005	59,524	44,282	61,323	51,765	71288,628
YOLOv5l + <i>Deep</i> SORT	419759	248062	255595	36,147	83,005	62,503	43,280	63,342	52,024	74536,075

Nota: O rastreador representado por Kalman se refere ao algoritmo explicado na Subseção 3.3.3.

Através dos resultados apresentados anteriormente, é possível perceber uma melhora na acurácia do detector conforme são aplicados modelos mais densos, como por exemplo, os modelos *m* e *l* da YOLOv5. No entanto, como visto na Tabela 1, esse aumento é acompanhado de um maior custo computacional, de forma que, para o *hardware* utilizado, essas redes tornam-se desaconselháveis para aplicações em tempo real.

Além disso, também é possível perceber maiores valores de rastreamentos verdadeiro positivos por parte das combinações que aplicam o *Deep SORT* como algoritmo de rastreamento. Este resultado já era esperado uma vez que, como discutido na Subseção 2.3.3, por utilizar tanto da informação espacial, quanto da aparência dessas detecções, este algoritmo, além de aumentar a resiliência contra oclusões, possibilita a re-identificação de alvos, reduzindo IDSs e FPs no processo. A Figura 21 ilustra essa observação de modo que, conforme abordado na Subsubseção 4.3.1.4, a métrica AssA mede a acurácia da associação do rastreador, ou seja, sua proficiência em relacionar um conjunto de detecções, aos seus GTs correspondentes.

Figura 21 – Comparação AssA vs DetA

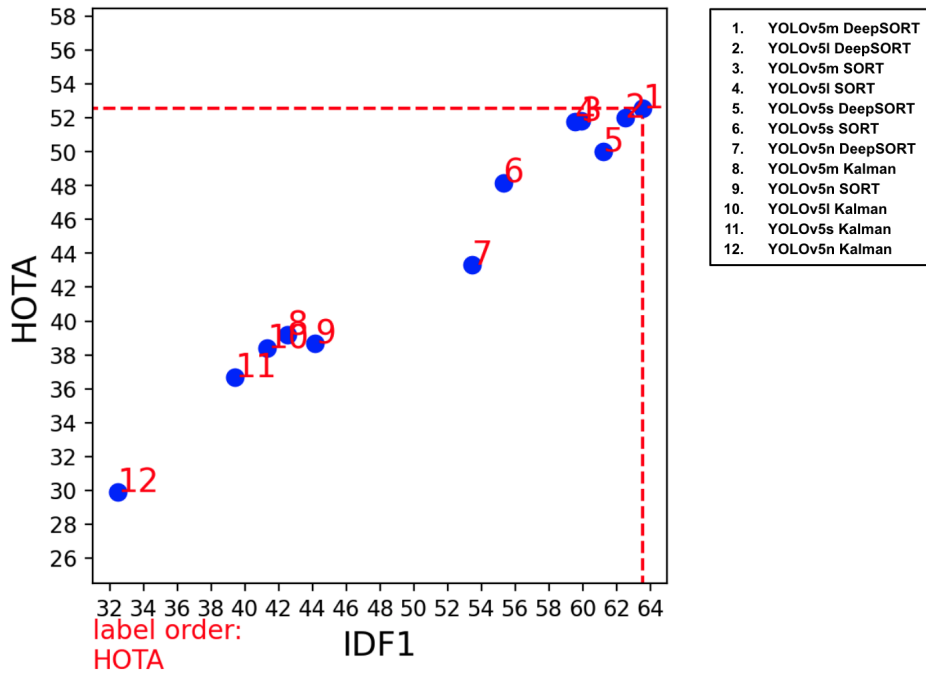


Fonte: Produção do próprio autor.

Outro detalhe importante que pode ser percebido através da análise dos resultados é como as métricas de IDF1 e MOTA priorizam, respectivamente, a associação em detrimento da detecção e a detecção em detrimento da associação. Essa observação pode ser melhor ilustrada através das Figuras 22 e 23, de modo que no primeiro caso, os conjuntos de

rastreadores que aplicam o *Deep SORT* estão localizados mais à direita do gráfico, isto é, possuem maiores valores de IDF1.

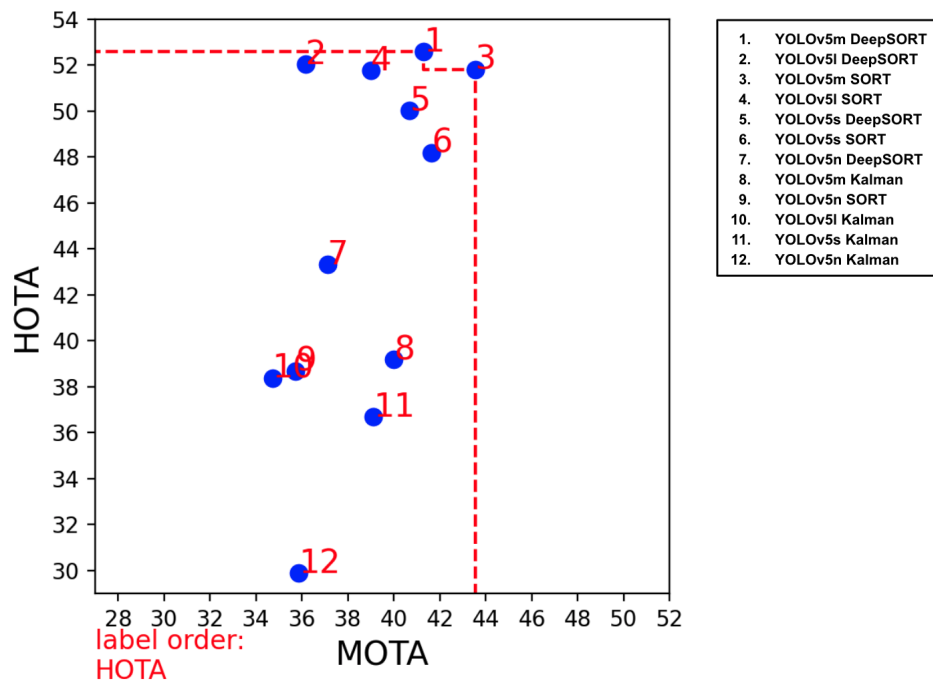
Figura 22 – Comparação HOTA vs IDF1



Fonte: Produção do próprio autor.

De forma similar, para o segundo caso, os conjuntos de rastreadores que utilizam modelos mais denso da YOLOv5 são localizados mais à direita do gráfico, isto é, possuem maiores valores de MOTA.

Figura 23 – Comparação HOTA vs MOTA



Fonte: Produção do próprio autor.

4.4 Resumo

Neste capítulo foram apresentados os resultados obtidos com a aplicação do sistema proposto no conjunto de dados UA-DETRAC. O capítulo seguinte aborda as conclusões e possíveis temas a serem pesquisados futuramente a partir deste trabalho.

5 CONCLUSÃO E TRABALHOS FUTUROS

5.1 Conclusão

O objetivo principal deste trabalho foi desenvolver um sistema capaz de realizar o rastreamento de veículos em vias públicas, extraindo métricas associadas ao controle de tráfego. Para isso, foram explorados diferentes modelos do detector de estágio único YOLOv5 (JOCHER, 2020), pré treinado no conjunto de dados MS COCO (LIN et al., 2014), combinadas aos algoritmos de rastreamento em MOT, SORT (BEWLEY et al., 2016) e *Deep SORT* (WOJKE; BEWLEY; PAULUS, 2017).

O sistema implementado foi avaliado no conjunto de dados UA-DETRAC (WEN et al., 2020), que consistem em imagens de vídeo monitoramento de rodovias para a detecção de veículos (carros, caminhões, ônibus e vans). Este trabalho permitiu: (i) comprovar a viabilidade da utilização dos algoritmos de rastreamento SORT e *Deep SORT*, na composição de um sistema de rastreamento de veículos em tempo real, (ii) avaliar a aplicabilidade do detector de objetos YOLO para o problema de detecção de veículos e (iii) comprovar a aplicabilidade de técnicas de visão computacional para a extração de informações relevantes ao controle de tráfego.

Através dos resultados coletados, é possível observar a influência do número de camadas da rede no rastreamento de objetos onde, quanto maior esse número, maiores as quantidades de detecções verdadeiro positivas e maior a acurácia do rastreador. Essa conclusão já era esperada uma vez que a estimativa de estados do detector está intimamente ligada à covariância daquela medição. Dessa forma, um detector com um bom desempenho consequentemente irá proporcionar melhores resultados para o rastreador.

Por fim, também foi possível observar um aumento na precisão do rastreador ao utilizar o algoritmo *Deep SORT*. Este aumento era previsto pois, como discutido na Subseção 2.3.3, o *Deep SORT* combina informações espaciais coletadas no SORT, com características visuais dos objetos, extraídas a partir de um extrator de características. Essa combinação além de fornecer mais robustez ao detector, possibilita a re-detecção de objetos, reduzindo potenciais trocas de ID, comuns no SORT. No entanto, como esperado, a aplicação de um extrator de características, ou seja, de uma rede neural para extração de informações visuais das imagens, aumenta o custo computacional do algoritmo. Todavia, conforme observado na Tabela 1, esse aumento não inviabiliza o uso desse rastreador em aplicações

em tempo real.

Dessa forma, após essa análise, e comparando os dados obtidos, optou-se pela utilização do rastreador de objetos *Deep SORT* em conjunto do modelo *s* do detector de objetos YOLOv5, visto que essa combinação é capaz de manter, tanto uma boa taxa de *frames* por segundo, quanto bons valores de acurácia (HOTA de 50,033 e MOTP de 81,536) se comparado aos demais resultados obtidos. Caso a aplicação em questão necessite de uma maior velocidade no rastreamento, e ainda conte com limitações de *hardware*, recomenda-se o uso de um conjunto rastreador-detector de menor peso computacional, como por exemplo, a YOLOv5n em conjunto do rastreador SORT. Por fim, caso a aplicação conte com mais poder computacional, recomenda-se o uso da YOLOv5m em conjunto do rastreador *Deep SORT*.

5.2 Temas a serem pesquisados

Como trabalhos futuros:

- Verificar a viabilidade de outros detectores de objetos, como por exemplo a *Faster R-CNN* (GIRSHICK, 2015) e o SSD (LIU et al., 2016), analisando os resultados obtidos;
- Verificar a possibilidade de explorar outras métricas de associação além do IoU;
- Analisar o desempenho do sistema proposto quando aplicado a *hardwares* embarcados;
- Considerar a otimização do código desenvolvido através de técnicas de *Tiny Machine Learning* (tradução livre, aprendizado de máquina pequeno);
- Realizar o treinamento de modelos pré-treinados da YOLOv5 no conjunto de dados de treino da UA-DETRAC, aplicando o algoritmo de *backpropagation* e analisar os resultados obtidos.

REFERÊNCIAS

- ADARSH, P.; RATHI, P.; KUMAR, M. Yolo v3-tiny: Object detection and recognition using one stage improved model. In: IEEE. 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS). [S.l.], 2020. p. 687–694. Citado na página 22.
- ALBAWI, S.; MOHAMMED, T. A.; AL-ZAWI, S. Understanding of a convolutional neural network. In: IEEE. 2017 international conference on engineering and technology (ICET). [S.l.], 2017. p. 1–6. Citado na página 23.
- ARAUJO, I. B. d. Quantificação, espacialização e especificação das emissões atmosféricas de origem veicular na região metropolitana da Grande Vitória. Quantificação, espacialização e especificação das emissões atmosféricas de origem veicular na região metropolitana da Grande Vitória, Universidade Federal do Espírito Santo, 2016. Citado na página 12.
- 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 3 vezes nas páginas 13, 31 e 59.
- BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. Yolov4: Optimal speed and accuracy of object detection. 2020. Citado na página 25.
- BOOCH, G. UML: guia do usuário. [S.l.]: Elsevier Brasil, 2006. Citado na página 38.
- BRADSKI, G. The OpenCV Library. Dr. Dobb's Journal of Software Tools, 2000. Citado 2 vezes nas páginas 38 e 45.
- BURBECK, S. Applications programming in smalltalk-80 (tm): How to use model-view-controller (mvc). Smalltalk-80 v2, v. 5, p. 1–11, 1992. Citado na página 37.
- CHOWDHURY, M. F.; BIPLOB, M. R. A.; UDDIN, J. Real time traffic density measurement using computer vision and dynamic traffic control. In: IEEE. 2018 Joint 7th International Conference on Informatics, Electronics & Vision (ICIEV) and 2018 2nd International Conference on Imaging, Vision & Pattern Recognition (icIVPR). [S.l.], 2018. p. 353–356. Citado na página 15.
- CIAPARRONE, G.; SÁNCHEZ, F. L.; TABIK, S.; TROIANO, L.; TAGLIAFERRI, R.; HERRERA, F. Deep learning in video multi-object tracking: A survey. Neurocomputing, Elsevier, v. 381, p. 61–88, 2020. Citado na página 13.
- CORTEZ, D. E. d. S. Desenvolvimento de um sistema de controle de tráfego inteligente baseado em visão computacional. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2022. Citado na página 14.
- DETRAN-ES. Relatório Anual de Estatística de Trânsito – 2009. [S.l.], 2016. Disponível em: <https://detran.es.gov.br/Media/detran/Estatistica/Frota/frota_2009.pdf>. Citado na página 12.

DETRAN-ES. Relatório Anual de Estatística de Trânsito – 2019. [S.l.], 2020. Disponível em: <<https://detran.es.gov.br/Media/detran/Estatistica/Frota/FROTA-2019.pdf>>. Citado na página 12.

EVERINGHAM, M.; GOOL, L. V.; WILLIAMS, C. K. I.; WINN, J.; ZISSERMAN, A. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. 2007. [Http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html](http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html). Citado na página 20.

GIRSHICK, R. Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision. [S.l.: s.n.], 2015. p. 1440–1448. Citado 4 vezes nas páginas 20, 21, 22 e 60.

GIRSHICK, R.; DONAHUE, J.; DARRELL, T.; MALIK, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2014. p. 580–587. Citado 2 vezes nas páginas 18 e 20.

HEARST, M. A.; DUMAIS, S. T.; OSUNA, E.; PLATT, J.; SCHOLKOPF, B. Support vector machines. IEEE Intelligent Systems and their applications, IEEE, v. 13, n. 4, p. 18–28, 1998. Citado na página 19.

JASWANTH, N.; KARRI, A. P.; VENKATARAMAN, H. Perceptual modelling of unconstrained road traffic scenarios with deep learning. In: IEEE. 2020 10th International Conference on Advanced Computer Information Technologies (ACIT). [S.l.], 2020. p. 811–814. Citado na página 15.

JOCHER, G. ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements. Zenodo, 2020. <<https://github.com/ultralytics/yolov5>>. Disponível em: <<https://doi.org/10.5281/zenodo.3908559>>. Citado 2 vezes nas páginas 38 e 59.

KALMAN, R. E. A new approach to linear filtering and prediction problems. 1960. Citado na página 13.

KHOSHELHAM, K. Computer vision techniques for urban mobility. In: SMART PARKING IN FAST-GROWING CITIES. [S.l.: s.n.], 2021. p. 62–76. Citado na página 14.

LEICHTER, I.; KRUPKA, E. Monotonicity and error type differentiability in performance measures for target detection and tracking in video. IEEE transactions on pattern analysis and machine intelligence, IEEE, v. 35, n. 10, p. 2553–2560, 2013. Citado na página 47.

LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P.; ZITNICK, C. L. Microsoft coco: Common objects in context. In: SPRINGER. European conference on computer vision. [S.l.], 2014. p. 740–755. Citado 3 vezes nas páginas 23, 38 e 59.

LIU, W.; ANGUELOV, D.; ERHAN, D.; SZEGEDY, C.; REED, S.; FU, C.-Y.; BERG, A. C. Ssd: Single shot multibox detector. In: SPRINGER. European conference on computer vision. [S.l.], 2016. p. 21–37. Citado 2 vezes nas páginas 23 e 60.

LUITEN, A. H. J. TrackEval. 2020. <<https://github.com/JonathonLuiten/TrackEval>>. Citado na página 45.

LUITEN, J.; OSEP, A.; DENDORFER, P.; TORR, P.; GEIGER, A.; LEAL-TAIXÉ, L.; LEIBE, B. Hota: A higher order metric for evaluating multi-object tracking. International journal of computer vision, Springer, v. 129, n. 2, p. 548–578, 2021. Citado 4 vezes nas páginas 26, 47, 51 e 53.

MAESSCHALCK, R. D.; JOUAN-RIMBAUD, D.; MASSART, D. L. The mahalanobis distance. Chemometrics and intelligent laboratory systems, Elsevier, v. 50, n. 1, p. 1–18, 2000. Citado na página 34.

MAGA, J. A.; HASS, G. C. The development of motor vehicle exhaust emission standards in california. Journal of the Air Pollution Control Association, Taylor & Francis, v. 10, n. 5, p. 393–414, 1960. Citado na página 12.

MAGAGNIN, R. C. Um sistema de suporte à decisão na internet para o planejamento da mobilidade urbana. Tese (Doutorado) — Universidade de São Paulo, 2008. Citado na página 12.

MATHWORKS. What is a convolutional neural network? What is a Convolutional Neural Network? 2022. <<https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html>>. Citado na página 18.

MILAN, A.; LEAL-TAIXÉ, L.; REID, I.; ROTH, S.; SCHINDLER, K. Mot16: A benchmark for multi-object tracking. 2016. Citado 2 vezes nas páginas 36 e 50.

NAPHADE, M.; TANG, Z.; CHANG, M.-C.; ANASTASIU, D. C.; SHARMA, A.; CHELLAPPA, R.; WANG, S.; CHAKRABORTY, P.; HUANG, T.; HWANG, J.-N.; LYU, S. The 2019 ai city challenge. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops. [S.l.: s.n.], 2019. p. 452–460. Citado na página 41.

PASZKE, A.; GROSS, S.; CHINTALA, S.; CHANAN, G.; YANG, E.; DEVITO, Z.; LIN, Z.; DESMAISON, A.; ANTIGA, L.; LERER, A. Automatic differentiation in pytorch. 2017. Citado na página 45.

PEGORETTI, H. F. et al. Rastreamento de múltiplos objetos utilizando filtro de kalman. Blumenau, SC, 2021. Citado na página 26.

REDMON, J. Darknet: Open Source Neural Networks in C. 2013–2016. <<http://pjreddie.com/darknet/>>. Citado na página 24.

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. [S.l.: s.n.], 2016. p. 779–788. Citado 3 vezes nas páginas 13, 24 e 25.

REN, S.; HE, K.; GIRSHICK, R.; SUN, J. Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, v. 28, 2015. Citado na página 22.

- RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATY, A.; KHOSLA, A.; BERNSTEIN, M. et al. Imagenet large scale visual recognition challenge. International journal of computer vision, Springer, v. 115, n. 3, p. 211–252, 2015. Citado na página 24.
- SENIOR, A.; HAMPAPUR, A.; TIAN, Y.-L.; BROWN, L.; PANKANTI, S.; BOLLE, R. Appearance models for occlusion handling. Image and vision computing, Elsevier, v. 24, n. 11, p. 1233–1243, 2006. Citado na página 14.
- SILVA, L. F. V. d. Extraindo dados de tráfego a partir de vídeos em tempo real. Dissertação (Mestrado) — Brasil, 2017. Citado 2 vezes nas páginas 12 e 14.
- SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. 2014. Citado na página 20.
- SINGH, A. Selecting the right bounding box using Non-Max suppression (with implementation). 2020. Disponível em: <<https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/>>. Citado na página 26.
- SOUZA, V.; SILVA, L.; SANTOS, A.; ARAÚJO, L. Análise comparativa de redes neurais convolucionais no reconhecimento de cenas. Anais do Computer on the Beach, v. 11, n. 1, p. 419–426, 2020. Citado na página 17.
- TANG, Z.; NAPHADE, M.; LIU, M.-Y.; YANG, X.; BIRCHFIELD, S.; WANG, S.; KUMAR, R.; ANASTASIU, D.; HWANG, J.-N. Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [S.l.: s.n.], 2019. Citado na página 41.
- UIJLINGS, J. R.; SANDE, K. E. V. D.; GEVERS, T.; SMEULDERS, A. W. Selective search for object recognition. International journal of computer vision, Springer, v. 104, n. 2, p. 154–171, 2013. Citado na página 19.
- VALENTE, J. Aumento do monitoramento traz debate sobre modernização e privacidade. EBC, 2019. Disponível em: <<https://agenciabrasil.ebc.com.br/geral/noticia/2019-09/aumento-do-monitoramento-traz-debate-sobre-modernizacao-e-privacidade>>. Citado na página 12.
- VOULODIMOS, A.; DOULAMIS, N.; DOULAMIS, A.; PROTOPAPADAKIS, E. Deep learning for computer vision: A brief review. Computational intelligence and neuroscience, Hindawi, v. 2018, 2018. Citado 2 vezes nas páginas 17 e 18.
- WANG, S.; OZCAN, K.; SHARMA, A. Region-based deformable fully convolutional networks for multi-class object detection at signalized traffic intersections: Nvidia aicity challenge 2017 track 1. In: IEEE. 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCOM/IOP/SCI). [S.l.], 2017. p. 1–4. Citado na página 15.

WEN, L.; DU, D.; CAI, Z.; LEI, Z.; CHANG, M.-C.; QI, H.; LIM, J.; YANG, M.-H.; LYU, S. Ua-detrac: A new benchmark and protocol for multi-object detection and tracking. Computer Vision and Image Understanding, Elsevier, v. 193, p. 102907, 2020. Citado 3 vezes nas páginas 13, 46 e 59.

WOJKE, N.; BEWLEY, A.; PAULUS, D. Simple online and realtime tracking with a deep association metric. In: IEEE. 2017 IEEE international conference on image processing (ICIP). [S.l.], 2017. p. 3645–3649. Citado 4 vezes nas páginas 13, 33, 36 e 59.

YILMAZ, A.; JAVED, O.; SHAH, M. Object tracking: A survey. Acm computing surveys (CSUR), Acm New York, NY, USA, v. 38, n. 4, p. 13–es, 2006. Citado na página 13.

ZOU, Z.; SHI, Z.; GUO, Y.; YE, J. Object detection in 20 years: A survey. 2019. Citado 3 vezes nas páginas 13, 14 e 22.