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

KAROLYNE PEREIRA SANTOS

DETECÇÃO E RASTREAMENTO DE OBJETOS PARA MOBILIDADE URBANA USANDO IMAGENS DE 360°

VITÓRIA
2022

KAROLYNE PEREIRA SANTOS

DETECÇÃO E RASTREAMENTO DE OBJETOS PARA MOBILIDADE URBANA USANDO IMAGENS DE 360°

Parte manuscrita do Projeto de Graduação da aluna **Karolyne Pereira Santos**, 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.

Orientadora: Dra. Raquel Frizera Vassallo

VITÓRIA
2022

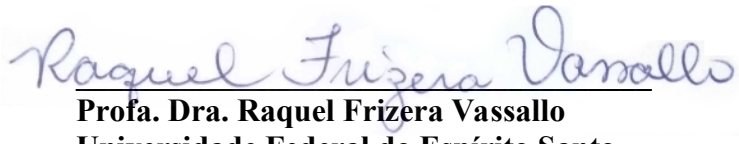
KAROLYNE PEREIRA SANTOS

DETECÇÃO E RASTREAMENTO DE OBJETOS PARA MOBILIDADE URBANA USANDO IMAGENS DE 360°

Parte manuscrita do Projeto de Graduação da aluna **Karolyne Pereira Santos**, 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.

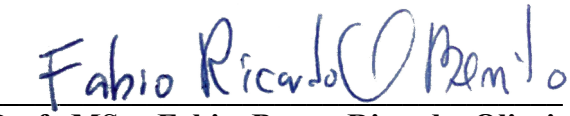
Aprovada em 28 de março de 2022.

COMISSÃO EXAMINADORA:



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

Prof. MSc. Bruno Neves Amigo
IFES - Guarapari
Examinador



Prof. MSc. Fabio Bento Ricardo Oliveira Bento
IFES - Guarapari
Examinador

Aos meus pais e minha irmã.

AGRADECIMENTOS

Agradeço primeiramente a Deus, que derramou sobre a minha vida sua misericórdia e me permitiu chegar até aqui.

Aos meus pais, Zilmar e Elzita, e minha irmã Nathália por todo amor e por sempre acreditarem que eu seria capaz, mesmo quando eu não acreditava. Agradeço a toda a minha família pelo auxílio e orações.

Aos meus amigos que estiveram comigo nesses anos, aguentaram todos os surtos e compartilharam tantos jogos de bisca. Em especial o Kadadarick, Fernanda e Marcos.

A minha psicóloga Letícia Itaborahy pelas intervenções e conselhos.

Agradeço também a professora Raquel, por aceitar me orientar e por toda a paciência, apoio e ajuda. Ao Laboratório VISIO e todos os integrantes que contribuíram de alguma forma.

A todos os professores e servidores do Departamento de Engenharia Elétrica.

Agradeço ao Rafael, Nathália, Joabe e Andressa pelo auxílio na gravação dos vídeos.

A Arthur, Daniel Jesus, Daniel Fonseca, Erick, Nathália e Rafael que acompanharam cada momento e sofrimento ao longo da execução deste trabalho.

RESUMO

Com o crescimento do fluxo de veículos nos centros urbanos, torna-se cada vez mais necessário realizar o monitoramento de veículos e pedestres, visando o levantamento de dados e estatísticas. Dado esse contexto, este trabalho tem por objetivo a criação de um sistema que realiza a detecção e o rastreamento de veículos, ciclistas e pedestres em tempo real em imagens de uma câmera 360º através de técnicas de Aprendizado Profundo aplicadas a uma Rede Neural Convolucional (CNN). Para tal solução foi utilizado o YOLOv5 treinado com a base pública MS COCO, o qual é um método de detecção de objetos que realiza de forma rápida e eficiente essa tarefa, combinado com o DeepSORT, que é um modelo de Aprendizado de Máquina para rastreamento de objetos que usa filtro de Kalman. A solução proposta neste trabalho funciona em tempo real com latência inferior a 300 ms. Através do seu emprego obteve-se uma precisão do sistema de 43,58%.

Palavras-chave: Detecção de objetos. Visão Computacional. Aprendizado Profundo

ABSTRACT

With the growth of the traffic in urban centers, the need to monitor vehicles and pedestrians, aiming at collecting data and statistics. Given this context, this work aims to create a system that performs the detection and tracking of vehicles, cyclists and pedestrians in real time in images taken from a 360º camera through Deep Learning techniques applied to a Convolutional Neural Network (CNN). For such a solution, YOLOv5 was trained with the MS COCO public dataset, which is an object detection method that quickly and efficiently performs this task, combined with DeepSORT, which is a Machine Learning model for tracking objects using Kalman filter. The solution proposed in this work runs in real-time with a latency lower than 300 ms. Through its use, a system precision of 43,58% was obtained.

Keywords: Object detection. Computer Vision. Deep Learning.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

CNN	*Convolutional Neural Network*
FPS	*Frames Per Second*
IEEE	*Institute of Electrical and Electronics Engineers*
IoU	*Intersection Over Union*
MOTA	*Multiple Object Tracking Accuracy*
R-CNN	*Region-based Convolutional Neural Network*
RPN	*Region Proposal Network*
RoI	*Region of Interest*
SORT	*Simple Online and Realtime Tracking*
SSD	*Single Shot Multibox Detect*
UFES	Universidade Federal do Espírito Santo
USB	*Universal Serial Bus*
UVC	USB *Video Class*
YOLO	*You Only Look Once*

SUMÁRIO

1 INTRODUÇÃO

Nos anos recentes, a implementação de câmeras em ambientes urbanos vêm sendo uma oportunidade para o desenvolvimento de novas soluções que superam os desafios na mobilidade urbana e sistemas de transporte (KHOSHELHAM, 2021). Como a principal tecnologia para gerenciamento moderno de trânsito, a detecção de veículos é a base para a realização de funções importantes, tal como a medição e estatísticas de trânsito, fluxo e densidade de tráfico, localização e rastreamento de veículos para controle de ambientes restritos entre outros (LU et al, 2018).

De acordo com Venables (2019, tradução nossa) "a visão computacional é um campo da inteligência artificial que treina computadores para interpretar e entender o mundo visual." Esta possui grande importância na área de sistemas de segurança e vigilância. As imagens de câmeras são utilizadas nas diversas aplicações de visão computacional, seja com identificação facial, detecção de objetos e anomalias (NOGUEIRA, 2018).

Segundo Khoshelham (2021), a detecção de objetos é a combinação das tarefas de localizar e classificar um ou mais objetos. Algumas abordagens de detecção de objetos são baseadas em *Region Proposal Network* - RPN (em português, Redes de Propostas Regional) incluindo *Faster Region based Convolution Network*, *Faster*-RCNN, em português Rápida Região baseada em Rede Neural Convolucional (REN et al., 2015) e *Mask* RCNN (HE et al., 2017), *Single Shot Multibox Detector*, SSD, em português Detector Múltiplo de Disparo Único (LIU et al., 2016) e regressão utilizando *bouding boxes* (em português, caixas delimitadoras) com diferentes versões do *You Only Look Once*, YOLO (REDMON et al., 2016).

Este trabalho propõe o desenvolvimento de um sistema capaz de localizar, classificar e rastrear objetos em tempo real voltado para a temática de mobilidade urbana utilizando como dispositivo de aquisição de informações uma câmera com ângulo de 360°. Mais especificamente um modelo que combine o detector de objetos YOLOv5 e DeepSORT, tendo a Ricoh Theta V como câmera 360º.

2 JUSTIFICATIVA

As técnicas de visão computacional utilizam câmeras para analisar e extrair informações visuais de cenas do mundo real podendo ser usadas em diversas aplicações, como, por exemplo, detecção de objetos, reconhecimento facial, monitoramento de tráfego, detecção de acidentes, interpretação da movimentação de pedestres, auxílio a pessoas com necessidades especiais, segurança pública, iluminação inteligente baseada no rastreamento de pedestres ou ciclistas, contagem de veículos, indicação de vagas disponíveis em vias públicas e até controle de veículos autônomos.

O campo de pesquisa da detecção de objetos tem se destacado nos recentes anos com o desenvolvimento do aprendizado profundo (do inglês, *deep learning*) com ferramentas mais rápidas e poderosas que resolveram os problemas existentes nas tradicionais arquiteturas (ZHAO et al., 2019). A temática de mobilidade urbana, mais especificamente a localização e classificação de veículos, ciclistas e pedestres tem crescido devido ao progresso de importantes aplicações da visão computacional: cidades inteligentes (MONTEMAYOR; PANTRIGO; SALGADO, 2015) e carros autônomos (KHOSHELHAM, 2021).

Com o aumento do número de automóveis trafegando pelos centros urbanos, torna-se cada vez mais necessário o controle dos veículos que passam por determinados lugares e, uma maneira eficaz de se empregar tal controle é a fiscalização de quais veículos passaram por determinados locais (NORVIG; RUSSEL, 2014).

O trabalho em questão utilizará uma câmera 360º, tornando possível a realização de uma leitura mais completa do espaço, pois o campo de visão será aumentado. Assim, com uma única ou poucas câmeras será possível monitorar uma grande área.

3 OBJETIVOS GERAL E ESPECÍFICOS

3.1 Objetivo Geral

O objetivo geral deste trabalho é desenvolver um sistema capaz de detectar, rastrear e classificar objetos em tempo real nas imagens de uma câmera 360°.

3.2 Objetivos Específicos

Os objetivos específicos são:

- Tratar as imagens em 360º para obter imagens panorâmicas;
- Verificar as detecções de veículos, ciclistas e pedestres;
- Rastrear em tempo real os objetos detectados.

4 REFERENCIAL TEÓRICO

4.1 Visão Computacional para Detecção de Objetos

Segundo Milano e Honorato (2010), a visão computacional é a tecnologia que se concentra em como um computador enxerga o meio à sua volta, extraindo significativas informações a partir de imagens capturadas por câmeras de vídeo, sensores e *scanners*, entre outros dispositivos. Isso permite que os computadores identifiquem, manipulem e processem objetos em imagens e vídeos buscando fazê-lo de maneira semelhante aos seres humanos.

A seguir são apresentados algumas das funcionalidades comuns na maioria dos sistemas de detecção e localização de objetos, conforme indicado por Rehem e Trindade (2009):

- Aquisição de imagens;
- Pré-processamento;
- Extração de característica;
- Detecção e segmentação;
- Processamento de alto nível.

Aquisição de imagens é feita a partir de um sensor que captura a imagem e um elemento capaz de digitalizar o sinal do sensor. Dependendo do tipo do sensor, o resultado pode variar entre uma imagem bidimensional, uma cena tridimensional ou ainda uma sequência de imagens. Antes de um método de visão computacional ser aplicado em uma imagem, é necessário realizar o seu pré-processamento para melhorar a sua qualidade e assegurar-se de que esta satisfaz as condições do método. Em seguida é feita a extração de características matemáticas que compõem uma imagem.

A detecção e a segmentação se referem ao processo de dividir a imagem digital em um conjunto de *pixels*, com o objetivo de mudar a representação de uma imagem para localização da região ou objeto de interesse. Por fim, é realizado o processamento de alto nível, que inclui a verificação da satisfação dos dados, a estimativa de parâmetros sobre a imagem e a classificação dos objetos detectados em diferentes categorias.

4.2 Aprendizado Profundo

Deep Learning, ou aprendizado profundo é a mais efetiva abordagem de *Machine Learning* (em português, aprendizado de máquina). Os métodos de *deep learning* tiveram um significativo papel no avanço e bom desempenho de uma variedade de aplicações da área da ciência e segurança, que incluem classificação de imagens biológicas, visão computacional, detecção de câncer, processamento de linguagem natural, detecção de objetos, reconhecimento facial, reconhecimento de escrita à mão, reconhecimento de fala, cidades inteligentes entre outros (DARGAN et al., 2020).

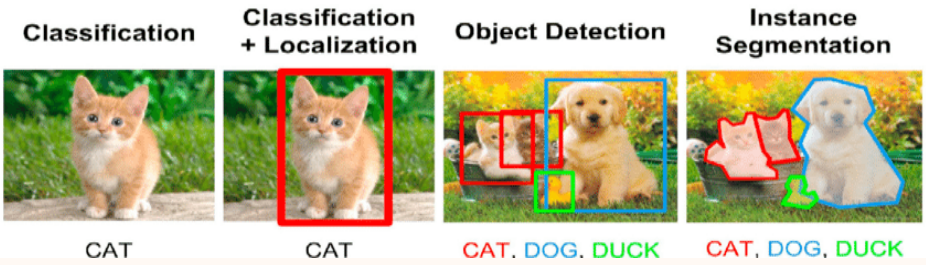
O aprendizado profundo permite que modelos computacionais de multiprocessamento de camadas aprendam e representem dados com múltiplos níveis de abstração, o que implica a captura de estruturas intrínsecas de dados em larga escala. É uma rica família de métodos, abrangendo redes neurais, modelos probabilísticos de hierarquia e uma variedade de algoritmos de aprendizado supervisionados e não supervisionados (VOULODIMOS et al, 2018).

O aprendizado profundo impactou quase todas as facetas de visão computacional que dependem de *machine learnig*. Classificação, localização e segmentação de imagem, foram todas alteradas desde o último ressurgimento em redes neurais e *deep learning*. Não é diferente na detecção de objetos (ROSEBROCK, 2017).

4.3 Detecção de objetos

Visão computacional e *Deep learning* podem ser aplicados na análise de um ou mais objetos em uma imagem. A classificação de uma imagem quanto ao seu tipo é o processo mais simples de predição. Melhorando esse processo pode-se ter a localização exata de determinada característica ou objeto na imagem. Para múltiplos objetos tem-se como método a detecção de objetos, do inglês *object detection*. Existe ainda a segmentação, que permite detectar os exatos pixels que compõem o objeto. A Figura 1 exemplifica essas diferenças (SHARMA, 2019).

Figura 1 – Comparação entre os tipos de objetivos na visão computacional



Fonte: Khan, Al-Habsi (2019).

A detecção de objetos é o problema mais desafiador no campo da visão computacional. Seu objetivo é localizar diferentes objetos em uma cena através das *bouding boxes* (em português, caixas delimitadoras), ou coordenadas cartesianas, e atribuir uma classe de rótulo associado à *bouding box* (SHAFIEE et al., 2017).

Foram identificados alguns trabalhos interessantes relacionados à detecção e rastreamento de veículos e pedestres em imagens. Mandal e outros (2020), Doshi e Yilmaz (2020) e Aboahl e outros (2021) utilizaram o YOLO para a detecção de anomalias no trânsito. Lu e outros (2018) realizaram a detecção de veículos em imagens aéreas usando o YOLOv3. Doan e Truong (2020) desenvolveram um modelo que utiliza o YOLOv4 e DeepSORT para a detecção e contagem em tempo real de veículos. Zhao e Chen (2019) usaram o YOLO para detecção em tempo real de pedestres.

Diferentes métodos têm sido desenvolvidos com o objetivo de minimizar o esforço computacional. Alguns destes são apresentados nas seções seguintes.

4.3.1 R-CNN

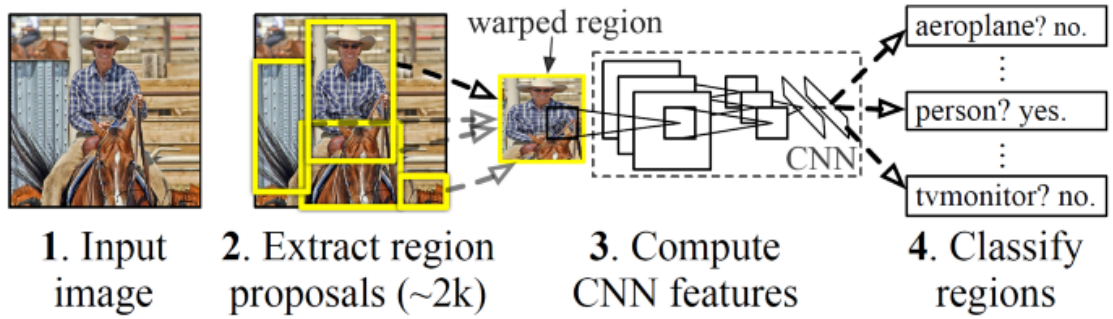
Region-based Convolutional Network (R-CNN, em português, Redes Convolucionais Baseadas em Região) é um método que combina propostas de regiões com CNNs (CHOUDHURY, 2020).

R-CNN é composto por três principais módulos. O módulo principal extrai cerca de 2000 propostas de regiões usando um algoritmo chamado *Selective Search* - SS, em português

pesquisa seletiva, que utiliza cores, texturas, iluminação entre outras características para identificar a possível localização dos objetos e quais partes de uma imagem são mais prováveis de conter um objeto. O segundo módulo é uma CNN aplicada em cada região resultante da pesquisa seletiva. O módulo final classifica cada região com base no *Suport Vector Machine*, SVM, em português, Máquina de Vetores de Suporte (GIRSHICK et al., 2014).

A Figura 2 apresenta um resumo das etapas do R-CNN, onde obtém uma imagem de entrada, extrai cerca de 2.000 propostas de regiões de baixo para cima, calcula recursos para cada região proposta usando uma grande CNN e, em seguida, classifica cada região usando a classe SVMs.

Figura 2 – Arquitetura da R-CNN



Fonte: Girshick et al. (2014).

É necessário um grande investimento de tempo para treinar a rede para a classificação de 2.000 regiões propostas por imagem. Por levar cerca de 47 segundos para cada imagem de teste, tal método não pode ser implementado em tempo real. Por fim, não ocorre aprendizado na fase da pesquisa seletiva, o que pode gerar terríveis candidatos às regiões propostas (GANDHI, 2018).

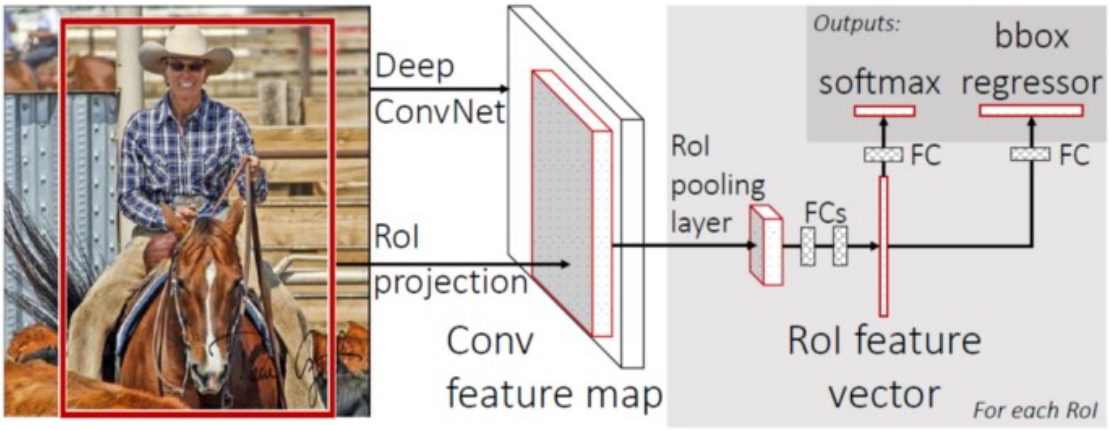
Apesar da fase de treinamento e teste serem lentas, R-CNN foi um método essencial para pavimentar o caminho para o desenvolvimento de importantes métodos melhorados.

4.3.2 *Fast* R-CNN

Fast R-CNN, em português, Rápida Rede Convolucional Baseada em Região, foi desenvolvida com a intenção de diminuir significantemente o tempo de treinamento em comparação com a R-CNN, rodando a rede neural uma vez em toda a imagem (GIRSHICK, 2015).

A Figura 3 apresenta a arquitetura da *Fast* R-CNN, na qual uma imagem de entrada e várias *Region of Interest* (RoI, em português Região de Interesse) são inseridas em uma rede convolucional. Cada RoI é agrupado em um mapa de recursos de tamanho fixo e, em seguida, mapeado para um vetor de recursos por camadas totalmente conectadas. A rede tem dois vetores de saída por RoI: probabilidades *softmax* e deslocamentos de regressão de *bounding boxes* que são a classe e a posição do objeto, respectivamente. Isso torna o Fast R-CNN mais preciso do que o R-CNN original. A arquitetura é treinada de ponta a ponta com uma perda de multitarefa (GIRSHICK, 2015).

Figura 3 – Arquitetura da Fast R-CNN



Fonte: Girshick (2015).

4.3.3 *Faster* R-CNN

Ambos os algoritmos acima (R-CNN e *Fast* R-CNN) usam SS para descobrir as regiões propostas, que é um processo lento e demorado, afetando o desempenho da rede. Por isso, Ren e outros (2017) propuseram um algoritmo de detecção de objetos que elimina o algoritmo de pesquisa seletiva e permite que a rede conheça as regiões propostas.

Ele consiste em dois módulos: uma CNN chamada *Region Proposal Network* (RPN, em português, Rede de Propostas de Região), e o detector *Fast* R-CNN. Os dois módulos são mesclados em uma única rede e treinados de ponta a ponta. Usar o RPN não apenas reduz o tempo de proposta de região de 2s para 10ms por imagem, mas também permite que o estágio de proposta de região compartilhe camadas com os estágios de detecção seguintes, causando uma melhoria geral na representação de recursos (REN et al., 2017).

4.3.4 YOLO

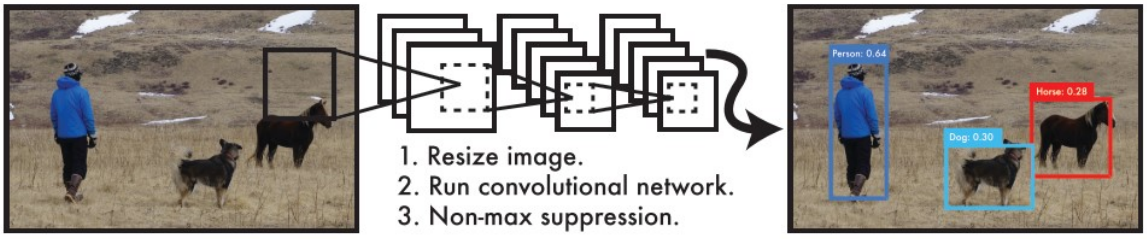
Em 2016 foi apresentado a *You Only Look Once* (em português, Você Só Olha Uma Vez), comumente chamada de YOLO, um sistema que se destaca na detecção de objetos pela sua velocidade na detecção em tempo real que mitigou os problemas de complexidade computacional associados a R-CNN (SHAFIEE et al., 2017).

De acordo com Nogueira (2018) a YOLO divide a imagem em uma grade de S x S onde, para cada célula da grade, é predito apenas um objeto e uma quantidade limitada de *bouding boxes* (B), tendo cada uma um vetor de confiança em cada classe (C) de objetos a ser detectado. Um dos principais conceitos da YOLO é construir uma rede para reduzir o volume de entrada de acordo com a Equação (1).

$$S \times S \times (B * 5 + C) \tag{1}$$

Como pode ser visto na Figura 4, o YOLO redimensiona a imagem de entrada, executa uma única rede convolucional na imagem e limita as detecções resultantes pela confiança do modelo. Esse modelo tem vários benefícios em relação aos métodos tradicionais de detecção de objetos. Processar imagens com YOLO é simples e direto (REDMON et al., 2016).

Figura 4 - Sistema de detecção da YOLO

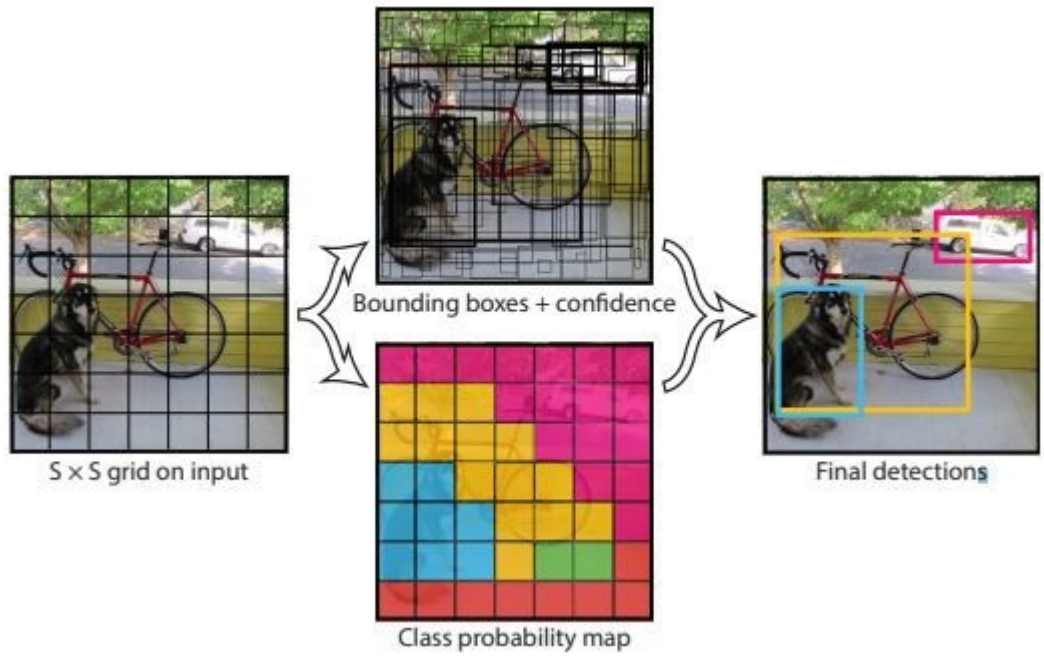


Fonte: Redmon et al. (2016).

Cada *bouding box* consiste em 5 predições: x, y, w, h e confiança. As coordenadas (x,y) representam o centro da caixa relativa à delimitação da célula. A largura e altura da imagem são expressas pelo w, h (do inglês, *width* e *height*) e a confiança na predição se refere a interseção sobre união (do inglês, *Intersection Over Union* – IoU) entre a caixa de predição e a área da verdadeira caixa (REDMON et al., 2016).

Como são definidas diversas caixas por células, é necessário aplicar o processo de *Non-maximal Suppression* (em português, Supressão do Não Máximo) que ordena as predições com base em suas notas de confiança e começa com as primeiras detecções de maior confiança, ignorando as detecções que possuem sobreposição maior que 50% na mesma classe. Assim os objetos não são detectados mais de uma vez com várias *bounding boxes*, como exemplificado na Figura 5 (REDMON et al., 2016).

Figura 5 – Estrutura da YOLO



Fonte: Redmon et al. (2016).

Pelo seu desempenho em sistemas de tempo real e rapidez na detecção, neste trabalho será utilizada a YOLOv5 que é treinada com a base de imagens do MS COCO sendo capaz de detectar e classificar 80 classes distintas. É constituído por três componentes (LIU et al, 2021):

- *Backbone*: para extrair características semânticas. YOLOv5 mantêm a estrutura parcial da versão anterior, CSPDarknet53, e aprende da estrutura de rede do CSPNet.
- *Neck*: para fundir características multiníveis. YOLOv5 usa a estrutura FPN-PAN, a estrutura CSP2 desenvolvida pela CSPNet, e PANet.
- *Head*: para a localização das *bouding boxes* e classificação dos objetos.

4.4 Rastreamento em Tempo Real

Simple Online and Realtime Tracking (SORT, em português Rastreamento Simples Online e em Tempo Real) é um método que implementa o filtro de Kalman e vincula dados *frame* a *frame*, usando o algoritmo húngaro para calcular o valor do grau de sobreposição do *frame*. Entretanto, SORT é menos efetivo em objetos que ficam ocultos por um momento. Para sanar essa limitação, foi proposto o DeepSORT (DOAN; TRUONG, 2020).

O filtro de Kalman desempenha um papel importante no DeepSORT. Ele usa detecções disponíveis e as previsões anteriores para chegar a uma estimativa melhor sobre o estado atual do objeto. Para cada detecção cria-se um rastreador que contém todas as informações de estado necessárias. Ele também possui um parâmetro que exclui as detecções bem-sucedidas realizadas há muito tempo, pois esses objetos teriam deixado a cena (MAIYA, 2019).

Os autores do DeepSORT decidiram usar a distância Mahalanobis para incorporar as incertezas do filtro de Kalman e o algoritmo húngaro para resolver o problema de rastreamentos perdidos (WOJKE; BEWLEY; PAULUS, 2017).

O DeepSORT utiliza *Matching Cascade* que é dedicado a resolver os problemas de oclusão do objeto por um longo período. Contra intuitivamente, quando dois rastreamentos competem pela mesma detecção, a distância de Mahalanobis favorece maior incerteza, porque reduz efetivamente a distância em desvios padrões de qualquer detecção em relação à média da trajetória projetada. Por isso, apresentou-se o *Matching Cascade* onde é priorizada a *bouding box* com o tempo de desaparecimento mais curto (WOJKE; BEWLEY; PAULUS, 2017).

Na Figura 6 está descrito o algoritmo de correspondência. Como entrada, é fornecido o conjunto de índices de *track* (T) e detecção (D), bem como a idade máxima (A_{max}). Nas linhas

1 e 2 calcula-se a matriz de custo de associação e a matriz de associações admissíveis. Em seguida, itera-se sobre a idade da *track* n. Na linha 6 seleciona-se o subconjunto de *tracks* (T_n) que não foram associadas a uma detecção nos últimos n frames. Nas linhas 8 e 9 atualiza-se o conjunto de detecções e detecções perdidas, que é retornado após a conclusão na linha 11. O *matching cascade* prioriza as *tracks* com menores idades, ou seja, aquelas que foram vistas mais recentemente.

Figura 6 – *Matching Cascade*

Listing 1 Matching Cascade

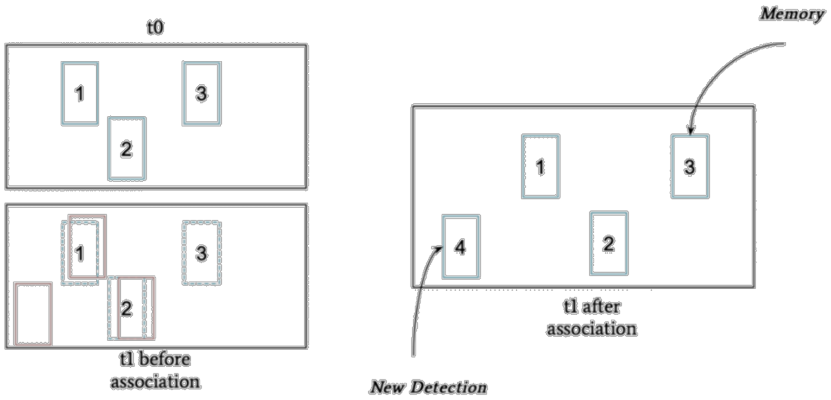
Input: Track indices $\mathcal{T} = \{1, \ldots, N\}$, Detection indices $\mathcal{D} = \{1, \ldots, M\}$, Maximum age $A_{\max}$

1: Compute cost matrix $\boldsymbol{C} = [c_{i,j}]$ using Eq. 5
2: Compute gate matrix $\boldsymbol{B} = [b_{i,j}]$ using Eq. 6
3: Initialize set of matches $\mathcal{M} \leftarrow \emptyset$
4: Initialize set of unmatched detections $\mathcal{U} \leftarrow \mathcal{D}$
5: **for** $n \in \{1, \ldots, A_{\max}\}$ **do**
6: Select tracks by age $\mathcal{T}_n \leftarrow \{i \in \mathcal{T} \mid a_i = n\}$
7: $[x_{i,j}] \leftarrow \text{min\_cost\_matching}(\boldsymbol{C}, \mathcal{T}_n, \mathcal{U})$
8: $\mathcal{M} \leftarrow \mathcal{M} \cup \{(i,j) \mid b_{i,j} \cdot x_{i,j} > 0\}$
9: $\mathcal{U} \leftarrow \mathcal{U} \setminus \{j \mid \sum_i b_{i,j} \cdot x_{i,j} > 0\}$
10: **end for**
11: **return** $\mathcal{M}, \mathcal{U}$

Fonte: Wojke, Bewley e Paulus (2017).

Na Figura 7, tem-se em t0 3 objetos com identificações 1, 2 e 3, em t1 tem-se também 3 objetos, mas estes não são os mesmos: 1 e 2 foram detectados novamente, mas 3 de t0 não foi detectado, incluiu-se também um novo objeto. Após a associação, considera-se que: o objeto 3 é um objeto de memória, uma detecção perdida, e será removido se não for identificado novamente por alguns *frames*. E objeto 4 é um novo objeto.

Figura 7 – Funcionamento do sistema de rastreamento



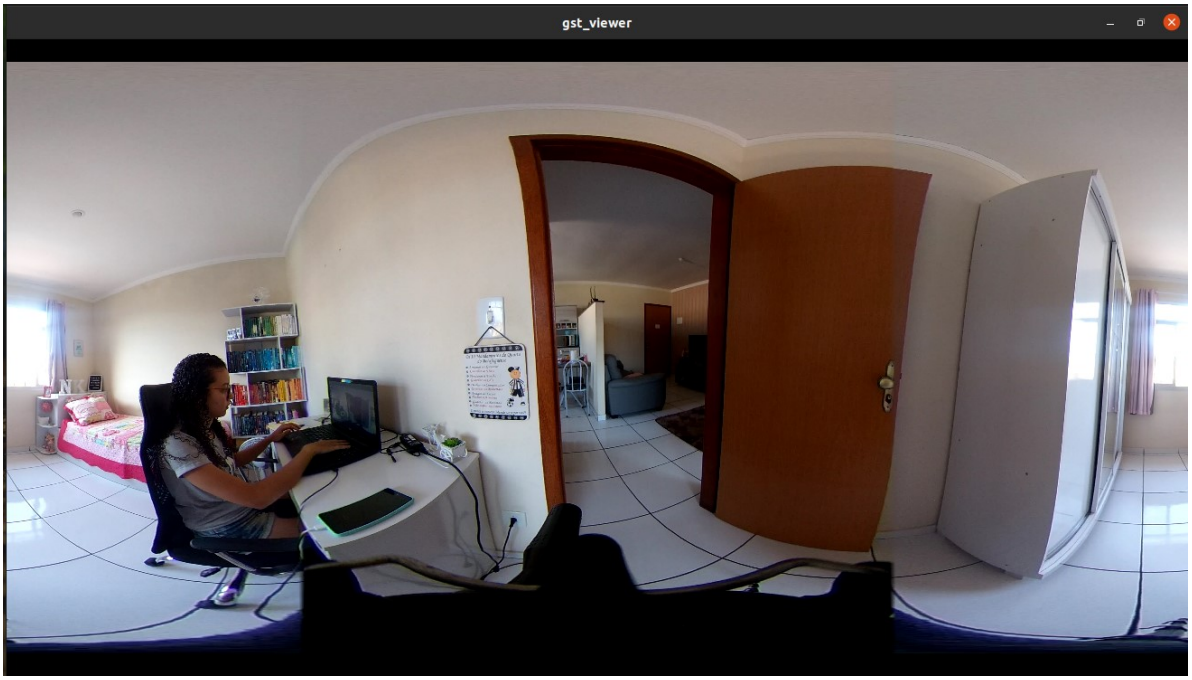
Fonte: Think Autonomous (2019).

5 METODOLOGIA E ETAPAS DE DESENVOLVIMENTO

Quanto à sua natureza, este trabalho se classifica como uma pesquisa aplicada, pois seu objetivo final foi gerar uma ferramenta para a detecção e o rastreamento de objetos em tempo real. Do ponto de vista dos procedimentos técnicos adotados, possui caráter experimental, onde o objeto de estudo foi a análise de imagens para realizar a detecção através de técnicas de Visão Computacional. Classifica-se como uma pesquisa explicativa no que diz respeito aos objetivos, baseando-se na análise de diferentes vídeos gerados.

O fabricante da câmera *Ricoh Theta* possui *firmware* para a transferência e conversão de arquivos compatíveis com os sistemas operacionais *Windows* e *Mac.* Entretanto, neste projeto utilizou-se o sistema operacional *Linux* sendo necessário um método para tornar possível a planificação das imagens em 360º. Usou-se a biblioteca *libuvc theta,* o *driver libuvc theta sample*. Junto com o módulo *v4l2loopback*, para realizar a transmissão em tempo real da imagem da câmera *Ricoh Theta* V com o sistema operacional *Linux*, através do cabo USB, e com a câmera no modo *live* permitindo visualizar as imagens já planificadas como pode ser visto na Figura 8.

Figura 8 – Imagem planificada da câmera 360°



Fonte: Produção do próprio autor.

O *v4l2loopback* é um módulo *kernel* que cria dispositivos V4L2 *loopback*, ou seja, esse módulo permite criar "*virtual video devices*" (do inglês, dispositivos de vídeo virtual). Sendo possível a utilização da câmera como um dispositivo de vídeo virtual, o passo seguinte foi compilar e instalar a biblioteca *libuvc theta* e o d*river libuvc theta sample*. O primeiro é uma biblioteca para dispositivos de vídeo USB, construída sobre o *libusb* que possibilita o controle de dispositivos de vídeo USB, exportando a interface USB *Video Class* (UVC) padrão. O *libucv theta*, junto com o *driver libuvc theta sample* e o módulo *v4l2loopback,* permitem usar a *Ricoh Theta* V e visualizar as imagens em tempo real, com máxima resolução de 3840x1920 *pixels* e latência inferior a 300 ms.

Para a detecção de objetos usou-se o modelo YOLOv5s (YOLOv5 *small*), escolhido por ser menor e mais rápido. Tal modelo é pré-treinado no *dataset* MS COCO em 80 classes. Para este trabalho foram selecionadas 6 classes de interesse, sendo elas: pessoa, carro, moto, bicicleta, ônibus e caminhão. Antes do método ser aplicado, como forma de pré-processamento da imagem foi necessário o redimensionamento do *frame* de 3840x1920 para 640x320.

Após a localização e classificação, os objetos detectados são enviados para que o DeepSORT realize o rastreamento deles, retornando uma *bounding box* com a classe, identificação de cada objeto e a confiança da detecção. Utilizou-se o DeepSORT com pesos pré-treinados, e apresentados no Quadro 1.

Quadro 1 – Configuração do DeepSORT

Maximum distance	0.2
Maximum confidence	0.3
NMS Maximum overlap	0.5
Maximum IOU distance	0.7
Maximum Age	70
Number of Init	3
NM Budget	100

Fonte: Produção do próprio autor.

6 EXPERIMENTOS E RESULTADOS

6.1 Coleta de Dados

Para a validar a detecção e rastreamento de carros, pessoas, motos, bicicletas, ônibus e caminhões foram utilizados vídeos gravados em diferentes localidades do município de Vila Velha, Espírito Santo. Os vídeos foram feitos com a câmera posicionada em um tripé, conforme descritos no Quadro 2.

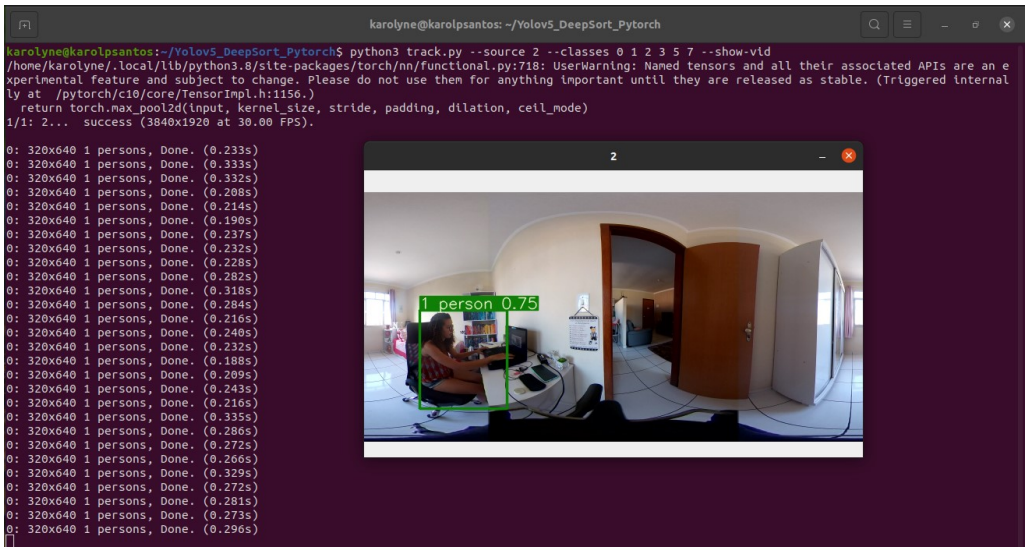
Quadro 2 – Informações sobre os vídeos

Vídeos	Endereço	Duração
1	Rua Santa Terezinha, 450, Glória, Vila Velha – ES	00:01:16
2	Av. Jerônimo Monteiro, 18, Olaria, Vila Velha – ES	00:01:18
3	Rua Gastão Roubach, Praia da Costa, Vila Velha – ES	00:01:29
4	Rua Sergipe, 305, Praia da Costa, Vila Velha – ES	00:01:10

Fonte: Produção do próprio autor.

Para a avaliar o funcionamento do sistema em tempo real foram realizados experimentos na casa da autora, devido à necessidade de a câmera estar conectada ao computador por meio do cabo USB. Na Figura 9 é mostrada a interface do sistema de detecção e rastreamento, onde podem ser vistas as *bounding boxes* com a identificação, classe do objeto e a confiança da detecção.

Figura 9 – Interface do sistema.

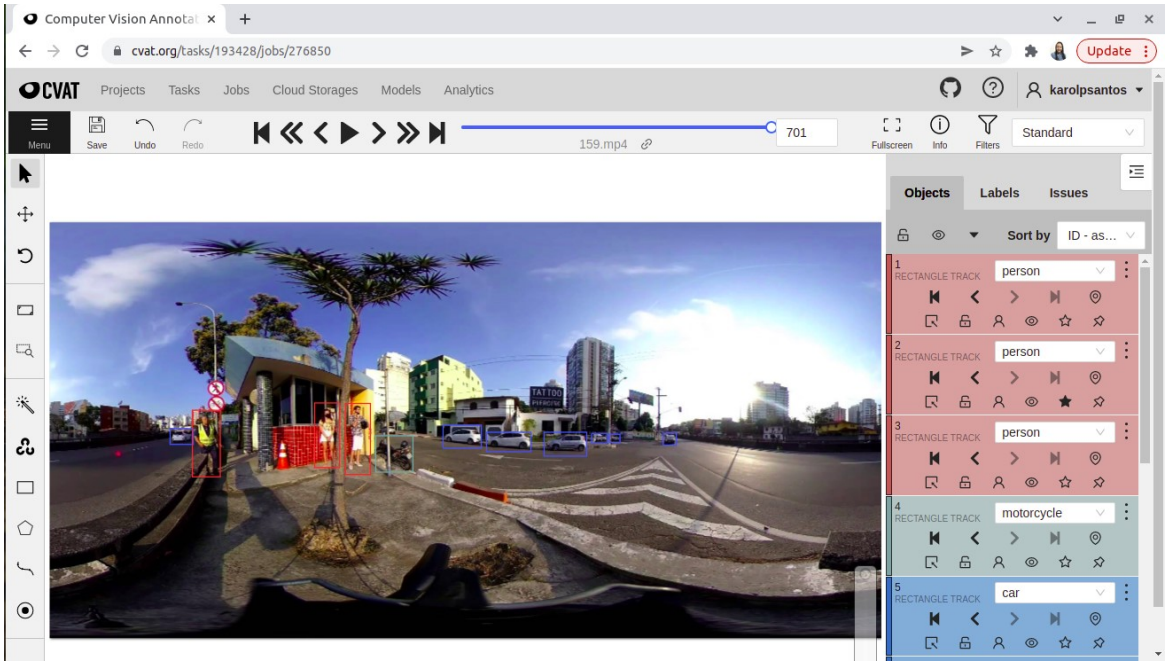


Fonte: Produção do próprio autor.

6.2 Rotulação dos Dados

Para garantir a confiabilidade dos resultados rotulou-se cada vídeo, *frame* por *frame*, através da *Computer Vision Annotation Tool* (CVAT), uma ferramenta de anotação de imagem e vídeo gratuita, de código aberto, baseada na web, usada para rotular dados para algoritmos de visão computacional. A Figura 10 apresenta a interface da ferramenta.

Figura 10 – Interface do CVAT



Fonte: Produção do próprio autor.

Os vídeos foram gravados em 30 FPS, dessa forma, para otimizar o trabalho de rotulação, cada vídeo foi amostrado para 10 FPS utilizando o FFmpeg, um programa em linha de comando composto de uma coleção de software livre e bibliotecas de código aberto usado para converter áudio e vídeo em diferentes formatos.

6.3 Equipamentos Utilizados

Os experimentos foram realizados em uma máquina contendo sistema operacional Linux Ubuntu 20.04 com as seguintes especificações:

- Processador: Intel Core i5 8ª geração;
- Memória RAM: 8 GB;
- Placa de Vídeo: AMD Radeon 520 2GB;

- Armazenamento: HD 1 TB.

A câmera possui as seguintes especificações:

- Fabricante: Ricoh;
- Modelo: Theta V;
- Lentes: 2 lentes com campo de visão 180°;
- Resolução: 4K;
- Processador: Qualcomm's Snapdragon
- Armazenamento: 19 GB;
- Dimensões: 45,2 mm x 130,6 mm x 22,9 mm;
- Peso: 121 g.

6.4 Métricas

Para que possa ser feita a compreensão e análise dos dados obtidos, é importante selecionar as métricas que serão avaliadas. A precisão está relacionada à quantidade de dados que são realmente positivos de todos os selecionados. Já o *recall* diz qual a porcentagem de dados é realmente positiva comparado com a quantidade real dos classificados como positivos. O *F1-score* une a precisão e o *recall* (PÁDUA, 2020).

Para uma visualização mais clara do desempenho da detecção temos a matriz de confusão. Cada entrada em uma matriz de confusão denota o número de previsões feitas pelo modelo onde classificou as classes correta ou incorretamente (MOHAJON, 2020).

Os verdadeiros positivos (VP) são as detecções corretamente classificadas como positivas, falsos positivos (FP) as detecções incorretamente consideradas como positivas, falsos negativos (FN) as detecções que não foram feitas, *mismatch error* (IDSW, do inglês erro de compatibilidade) o número de identidades trocadas e *ground truth* (GT) que são as rotulações ou identificações reais. Além disso, o índice *t* representa o índice do *frame*.

De acordo com Bernardin e Stiefelhagen (2008), *Multiple Object Tracking Accuracy* (MOTA) mede a acurácia geral do rastreador e da detecção, e é dado pela Equação (2).

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + IDSW_t)}{\sum_t GT_t} \quad (2)$$

O valor da precisão é dado pela Equação (3).

$$Precisão = \frac{VP}{VP + FP} \quad (3)$$

O valor do *recall* é dado pela Equação (4).

$$Recall = \frac{VP}{VP + FN} \quad (4)$$

O valor do *F1-score* é dado pela Equação (5).

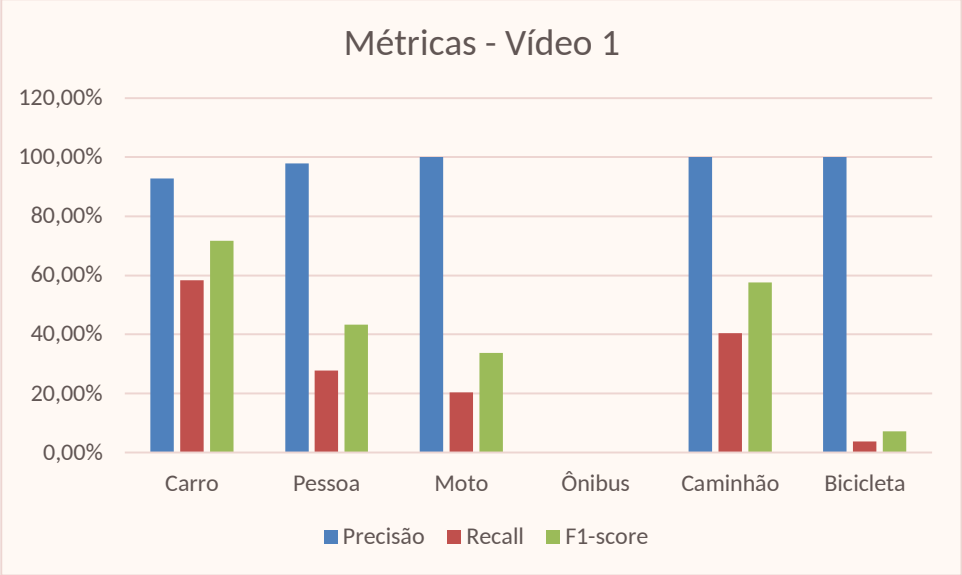
$$F1 - score = \frac{2 \times precisão \times recall}{precisão + recall} \quad (5)$$

6.5 Análise dos Resultados para Detecção

Foram analisados os vídeos após os 10s iniciais, pois a câmera era ajustada nesse tempo inicial. Calcularam-se as métricas para cada classe de detecção, sendo avaliado manualmente *frame* por *frame* em cada vídeo. Do Gráfico 1 ao Gráfico são mostrados os valores encontrados de precisão, *recall* e *F1-score* para cada vídeo.

No Gráfico 1 pode-se observar que não foram detectados ônibus no vídeo 1 e por isso as métricas não foram calculadas para essa classe. O YOLO foi capaz de detectar todas as classes com ótima precisão e poucas detecções erradas uma vez que o menor valor apresentado foi da ordem de 92,28% para a classe carro. Os valores de *recall* calculados demonstram que muitas detecções foram perdidas, principalmente para os objetos que ficavam mais distantes da lente da câmera, especificamente nesse caso, bicicletas e motos.

Gráfico 1 – Precisão, *recall* e *F1-score* para o vídeo 1



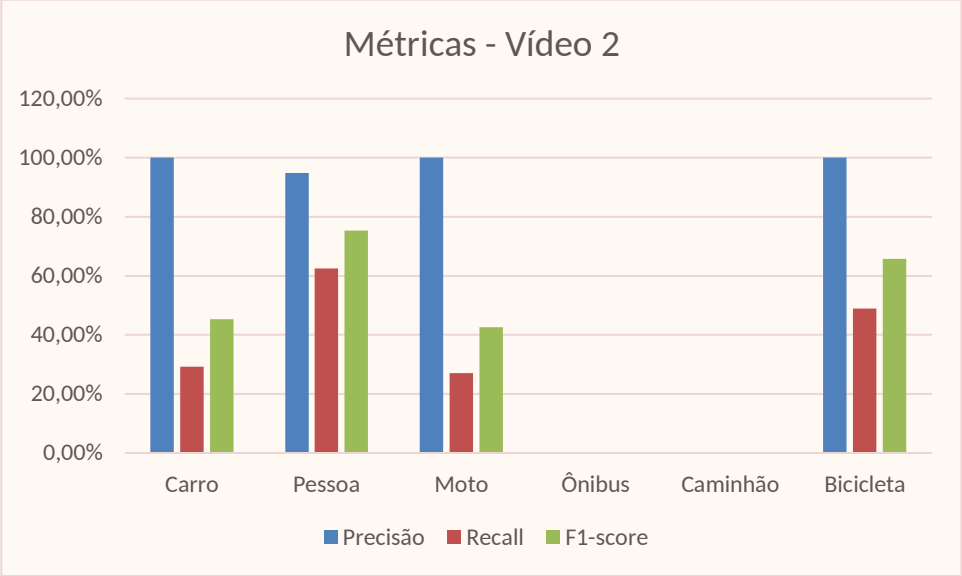
Fonte: Produção do próprio autor.

No Gráfico 2 tem-se para a classe ônibus 32 detecções erradas e 4 detecções erradas para a classe caminhão, apresentando precisão 0 em ambos os casos. Tais erros para a classe ônibus devem-se principalmente pela distorção da imagem quando planificada, o que fez com que o detector apontasse um prédio como ônibus. Todos os caminhões detectados foram carros detectados de forma errada, o que ocorreu quando o objeto se afastava da câmera. Não ocorreram detecções perdidas por isso não se calculou o *recall* e *F1-score* para essas classes.
O YOLO foi capaz de detectar as outras classes com boa precisão sendo o menor valor para a classe pessoa com 94,77%. Muitas detecções foram perdidas, principalmente para a classe carro e moto, como mostrado na Figura 11, por isso os valores de *recall* calculados foram menores que 50%, com exceção da classe pessoa.

Figura 11 – Moto não detectada no vídeo 2

Fonte: Produção do próprio autor.

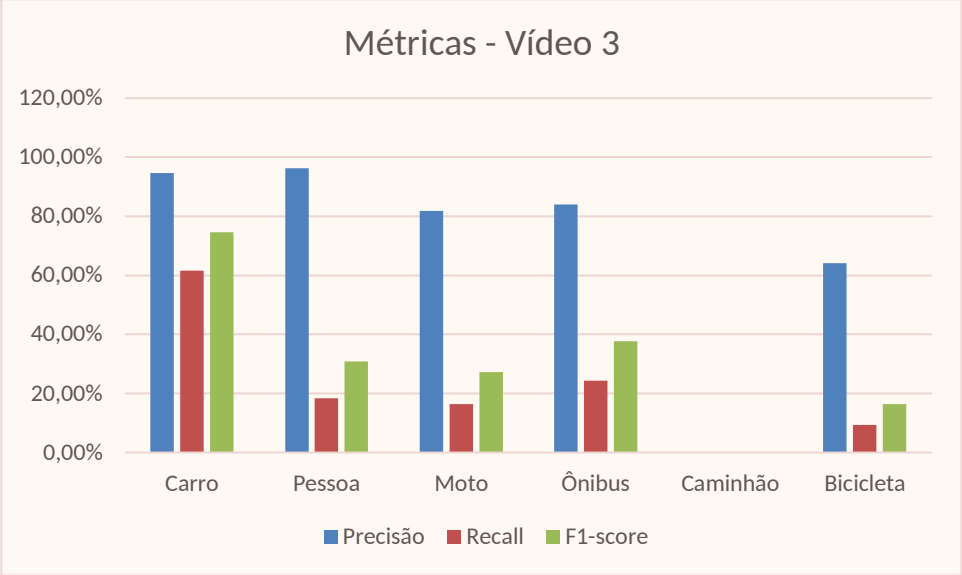
Gráfico 2 – Precisão, *recall* e *F1-score* para o vídeo 2



Fonte: Produção do próprio autor.

Para o vídeo 3, como mostrado no Gráfico 3, o YOLO apresentou uma maior dificuldade de localizar e classificar as classes em comparação aos outros vídeos, obtendo menor precisão e *recall* e, portanto, menor *F1-score*. Isso ocorre principalmente nas classes pessoa, moto, ônibus e bicicleta. Assume-se que tal dificuldade foi causada pelo posicionamento da câmera em relação à rua. Não foram detectados caminhões no vídeo 3 e por isso as métricas não foram calculadas para essa classe.

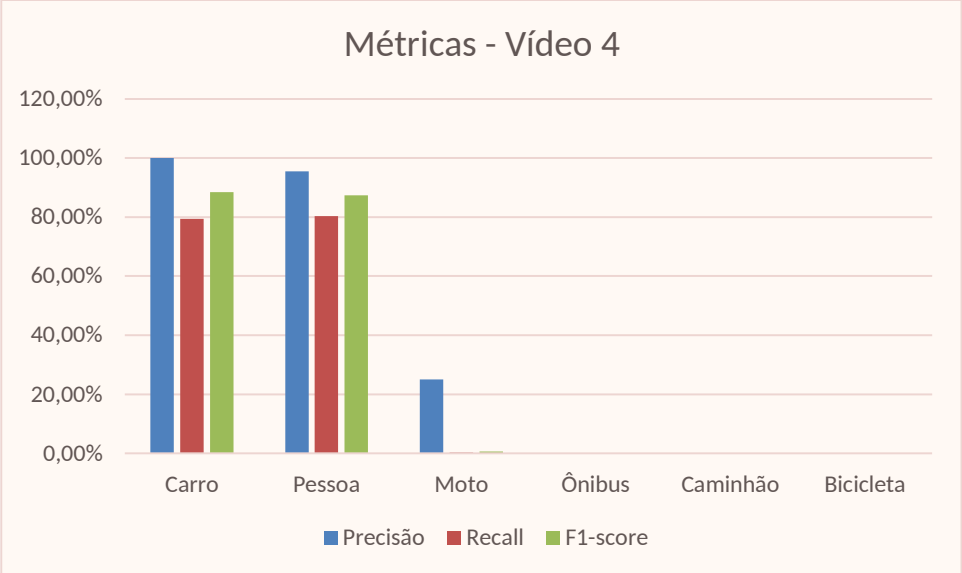
Gráfico 3 – Precisão, *recall* e *F1-score* para o vídeo 3



Fonte: Produção do próprio autor.

Pode-se ver no Gráfico 4 que não foram detectados ônibus, caminhão e bicicleta, por isso as métricas não foram calculadas para essas classes. O YOLO foi capaz de detectar as classes carro e pessoa com ótima precisão, *recall* e *F1-score*, o que leva a conclusão de que ocorreram poucas detecções erradas e perdidas. Houve maior dificuldade em detectar a classe moto, devido à distância entre o objeto e a lente da câmera, como pode ser visto na Figura 12.

Gráfico 4 – Precisão, *recall* e *F1-score* para o vídeo 4



Fonte: Produção do próprio autor.

Figura 12 – Motos não detectadas no vídeo 4

Fonte: Produção do próprio autor.

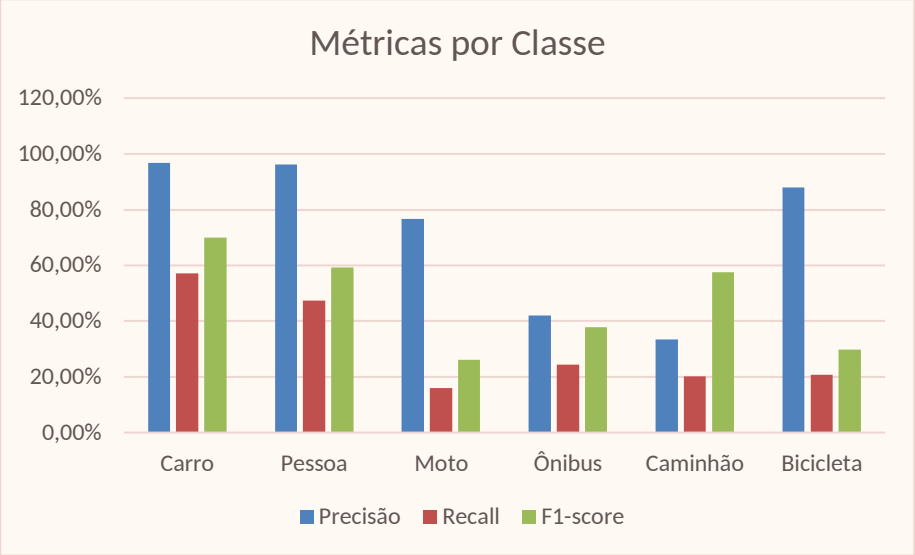
A média de todos os vídeos, para cada métrica, por classe é apresentada no Gráfico 5 e no Quadro 3. A média para a detecção considerando todas as classes foi 72,17% de precisão, 30,95% de *recall* e 46,74% de *F1-score*.

Quadro 3 – Média por classe

Classe	Precisão	Recall	F1-score
Carro	92,82%	57,14%	69,99%
Pessoa	96,14%	47,31%	59,25%
Moto	76,70%	16,04%	26,09%
Ônibus	42%	24,32%	37,72%
Bicicleta	88,03%	20,71%	29,81%
Caminhão	33,33%	20,71%	57,58%

Fonte: Produção do próprio autor.

Gráfico 5 – Precisão, *recall* e F1-*score* para cada classe



Fonte: Produção do próprio autor.

O Quadro 4 apresenta os valores das detecções feitas em comparação com o *ground truth* (objetos rotulados nos vídeos).

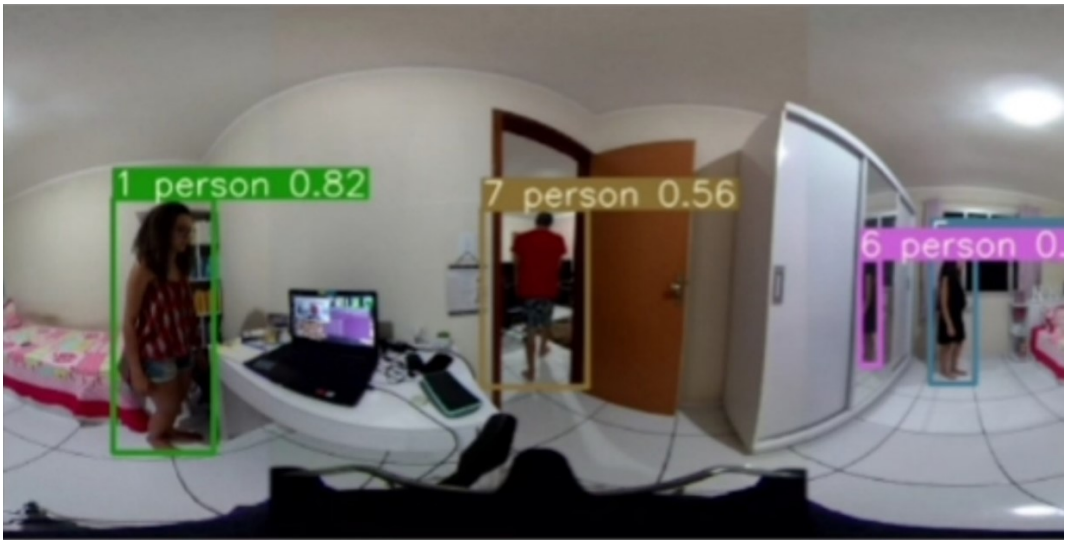
Quadro 4 – Valores totais da detecção

Classes	Vídeo 1		Vídeo 2		Vídeo 3		Vídeo 4	
	Real	Detectado	Real	Detectado	Real	Detectado	Real	Detectado
Carro	5010	3015	2540	743	5028	3163	4763	3779
Pessoa	8690	2458	4071	2598	6216	1182	2159	1750
Moto	113	23	59	16	798	154	664	8
Ônibus	0	0	0	32	271	75	0	0
Bicicleta	1060	40	47	23	838	117	0	0
Caminhão	94	38	0	4	251	0	0	0

Fonte: Produção do próprio autor.

A Figura 13 mostra um *frame* onde ocorre variação na confiança da detecção de acordo com a distância do objeto. Para a pessoa mais próxima da lente a confiança foi de 82% e 56% para a mais distante da lente da câmera. No sistema em tempo real o valor do FPS foi de 3,33.

Figura 13 – Variação na confiança da detecção em um mesmo *frame*



Fonte: Produção do próprio autor.

6.6 Análise dos Resultados para Detecção e Rastreamento

Após a classe ser detectada atribui-se a ela uma identificação. Entretanto, como a câmera é composta por duas lentes com 180° de campo de visão em cada uma, o DeepSORT muda a identificação de um mesmo objeto quando ele passa de uma lente para a outra. Isso pode ser visto na Figura 14, onde o mesmo ônibus recebe duas identificações diferentes.

Figura 14 – Diferentes identificações de um mesmo objeto

Fonte: Produção do próprio autor.

Calculou-se a matriz de confusão para cada vídeo, como pode ser visto nas Figura 15 a Figura 21. Para a matriz de confusão do vídeo 1 observa-se que, para a classe carro 3,57% nos erros de detecção representam que carros foram confundidos com bicicletas, 7,14% com caminhão

e 3,57% com outro objeto (uma carrocinha nesse caso). Nos objetos classificados como a classe caminhão, metade delas foram erros, com carros confundidos com caminhões.

Figura 15 – Matriz de Confusão do Vídeo 1

Vídeo 1		Classe Real						
		Carro	Pessoa	Moto	Bicicleta	Ônibus	Caminhão	Outro
Predição	Carro	85,71%	0	0	3,57%	-	7,14%	3,57%
	Pessoa	0	98,08%	0	1,92%	-	0	0
	Moto	0	0	100,00%	0	-	0	0
	Bicicleta	0	0	0	100,00%	-	0	0
	Ônibus	0	0	0	0	-	0	0
	Caminhão	50,00%	0	0	0	-	50,00%	0

Fonte: Produção do próprio autor.

Na Figura 17, onde se vê a matriz de confusão do vídeo 2, observa-se que, para a classe ônibus e caminhão todas as detecções e rastreamento foram erradas. Para ônibus em 50% dos casos, o detector apontou que um prédio era um ônibus. Acredita-se que a razão para isso está associada à distorção da imagem 360° quando transformada em panorâmica, mostrado na Figura 16. Todos os caminhões detectados foram carros detectados de forma errada, o que ocorria principalmente quando o objeto se afastava da câmera, como na Figura 18.

Figura 16 – Detecção incorreta da classe ônibus no vídeo 2

Fonte: Produção do próprio autor.

Figura 17 – Matriz de Confusão do Vídeo 2

Vídeo 2		Classe Real						
		Carro	Pessoa	Moto	Bicicleta	Ônibus	Caminhão	Outro
Predição	Carro	100,00%	0	0	0	-	-	0
	Pessoa	20,00%	66,67%	0	0	-	-	13,33%
	Moto	0	0	100,00%	0	-	-	0
	Bicicleta	0	0	0	100,00%	-	-	0
	Ônibus	50,00%	0	0	0	-	-	50,00%
	Caminhão	100,00%	0	0	0	-	-	0

Fonte: Produção do próprio autor.

Figura 18 – Carro detectado como caminhão

Fonte: Produção do próprio autor.

Para a matriz de confusão do vídeo 3, Figura 19, observa-se que, para a classe carro 36,84% nos erros de detecção são oriundos da confusão de carros com motos, 5,26% com bicicleta e 5,26% com outros objetos, como visto na Figura 20. Vale destacar que, 66,67% dos ônibus foram carros detectados e rastreados de forma errada.

Figura 19 – Matriz de Confusão do Vídeo 3

Vídeo 3		Classe Real						
		Carro	Pessoa	Moto	Bicicleta	Ônibus	Caminhão	Outro
Predição	Carro	52,63%	0	36,84%	5,26%	0	0	5,26%
	Pessoa	5,26%	84,21%	2,63%	0	0	0	0
	Moto	0	23,53%	76,47%	0	0	0	0
	Bicicleta	36,36%	0	0	54,55%	0	9,09%	0
	Ônibus	66,67%	0	0	0	33,33%	0	0
	Caminhão	0	0	0	0	0	0	0

Fonte: Produção do próprio autor.

Figura 20 - Barraca confundida com carro

Fonte: Produção do próprio autor.

Na Figura 21, onde se vê a matriz de confusão do vídeo 4, 50% das motos detectadas e rastreadas eram, na verdade, carros.

Figura 21 – Matriz de Confusão do Vídeo 4

Vídeo 4		Classe Real						
		Carro	Pessoa	Moto	Bicicleta	Ônibus	Caminhão	Outro
Predição	Carro	98,08%	0	1,92%	-	-	-	0
	Pessoa	0	80,00%	20,00%	-	-	-	0
	Moto	50,00%	0	50,00%	-	-	-	0
	Bicicleta	0	0	0	-	-	-	0
	Ônibus	0	0	0	-	-	-	0
	Caminhão	0	0	0	-	-	-	0

Fonte: Produção do próprio autor.

Os dados analisados pelo sistema nos vídeos podem ser vistos nos Quadro 5 ao Quadro 8, nos quais são calculados o MOTA, precisão, *recall* e *F1-score* em cada vídeo. Em todos os vídeos foram realizadas contagens manuais, verificando a quantidade de objetos de cada classe presente, contabilizando cada nova identificação. Dessa forma, pode-se observar variações em cada classe em relação à quantidade real, mas vale destacar que acontecem situações em que o mesmo objeto recebe mais de uma identificação devido à passagem de uma lente para a outra.

O MOTA é calculado nos casos em que o GT é diferente de zero e varia de valores negativos a 1. Se seu valor for 1 o sistema tem 100% de acurácia, se for próximo ou menor que zero, então a acurácia do sistema é baixa. O MOTA médio foi de -1,12426 para a classe carro,

0,40371 para pessoa, 0,121324 para moto, -1 para ônibus, -0,525 para bicicleta e -0,5 para caminhão.

Para o cálculo de precisão considerou-se como falso positivo quando o mesmo objeto recebia mais de uma identificação. A precisão média foi de 34,72% para a classe carro, 66,33% para pessoa, 57,56% para moto, 16,67% para ônibus, 73,74% para bicicleta e 12,50% para caminhão. Somando todas as precisões médias e dividindo pelo número de classes tem-se uma precisão total para o sistema de 43,58%.

O *recall* e *F1-score* médio foi de 98% e 48,09% para a classe carro, 83,63% e 71,73% para a classe pessoa, 66,67% e 56,61% para moto, 100% e 50% para ônibus, 80% e 73,53% para bicicleta e 100% e 40% para caminhão. Os altos valores de *recall* indicam que o sistema perdeu poucas detecções, sendo capaz de detectar e rastrear quase todos os objetos rotulados. As médias finais podem ser vistas no Quadro 9.

Quadro 5 – Eficácia do sistema no vídeo 1

Vídeo 1						
Classe	Número real	Identificado pelo sistema	MOTA	Precisão	Recall	F1-score
Carro	28	88	-1,28571	27,27%	100%	42,86%
Pessoa	52	57	0,884615	89,47%	100%	94,44%
Moto	2	2	1	100%	100%	100%
Ônibus	0	0	-	-	-	-
Bicicleta	5	3	-2	66,67%	100%	40%
Caminhão	1	4	0,2	25%	40%	50%

Fonte: Produção do próprio autor.

Quadro 6 – Eficácia do sistema no vídeo 2

Vídeo 2						
Classe	Número real	Identificado pelo sistema	MOTA	Precisão	Recall	F1-score
Carro	25	36	0,4	63,89%	92%	75,41%
Pessoa	15	29	-0,33333	34,48%	90,91%	50%
Moto	3	2	0	50%	33,33%	40%
Ônibus	0	2	-	0%	-	-
Bicicleta	1	1	1	100%	100%	100%
Caminhão	0	1	-	0%	-	-

Fonte: Produção do próprio autor.

Quadro 7 – Eficácia do sistema no vídeo 3

Vídeo 3						
Classe	Número real	Identificado pelo sistema	MOTA	Precisão	Recall	F1-score
Carro	19	83	-2,48211	12,05%	100%	21,51%
Pessoa	59	38	0,813559	84,21%	86,49%	85,33%
Moto	17	43	-0,76471	30,23%	100%	46,43%
Ônibus	1	3	-1	33,33%	100%	50%
Bicicleta	8	11	0,375	54,55%	100%	70,59%
Caminhão	0	0	1	-	-	-

Fonte: Produção do próprio autor.

Quadro 8 – Eficácia do sistema no vídeo 4

Vídeo 4						
Classe	Número real	Identificado pelo sistema	MOTA	Precisão	Recall	F1-score
Carro	52	143	-0,76923	35,66%	100%	52,58%
Pessoa	8	7	0,25	57,14%	57,14%	57,14%
Moto	4	2	0,25	50%	33,33%	40%
Ônibus	0	0	-	-	-	-
Bicicleta	0	0	-	-	-	-
Caminhão	0	0	-	-	-	-

Fonte: Produção do próprio autor.

Quadro 9 – Médias finais considerando todos os vídeos

Classe	MOTA	Precisão	Recall	F1-score
Carro	-1,12426	34,72%	98,00%	48,09%
Pessoa	0,40371	66,33%	83,63%	71,73%
Moto	0,121324	57,56%	66,67%	56,61%
Ônibus	-1	16,67%	100,00%	50,00%
Bicicleta	-0,525	73,74%	80,00%	73,53%
Caminhão	-0,5	12,50%	100,00%	40,00%

Fonte: Produção do próprio autor.

A Figura 22 mostra a detecção e rastreamento de 3 pessoas. Na Figura 23 pode-se observar uma falha na detecção, onde ocorre uma oclusão de uma pessoa. Nesse exato *frame* não é possível rastrear esse objeto, mas como mostrado na Figura 24, no momento seguinte, o rastreador recupera a identificação do objeto.

Figura 22 – *Frame* detectando e rastreando

Fonte: Produção do próprio autor.

Figura 23 – *Frame* com uma pessoa oclusa

Fonte: Produção do próprio autor.

Figura 24 – Rastreador recuperando a identificação do objeto que estava ocluso

Fonte: Produção do próprio autor.

7 CONCLUSÕES

Neste trabalho foi apresentado um sistema em tempo real para detecção e rastreamento de objetos para mobilidade urbana nas imagens de uma câmera 360°, com latência de aproximadamente 300 ms e 3,33 FPS para o hardware utilizado.

Utilizou-se como metodologia para a aquisição de imagens a biblioteca *libuvc theta,* o *driver libuvc theta sample* e o módulo *v4l2loopback*. O YOLOv5 foi usado para realizar as detecções das classes pessoa, carro, moto, bicicleta, ônibus e caminhão além do DeepSORT para o rastreamento dos objetos.

A técnica foi avaliada através de métricas e constatou-se que a precisão média para a detecção considerando todas as classes foi 72,17%. Dessa forma, comprovou-se que é possível utilizar uma câmera de 360°, de maneira que o campo de visão foi aumentado e a ferramenta foi capaz de realizar a detecção.

O MOTA médio foi de -1,12426 para a classe carro, 0,40371 para pessoa, 0,121324 para moto, -1 para ônibus, -0,525 para bicicleta e -0,5 para caminhão. Pelos valores de MOTA encontrados, tem-se que o sistema tem baixa acurácia de rastreamento. Acredita-se que a razão para isso está associada principalmente à troca de identificação que o objeto sofre quando passa de uma lente para a outra. Sugere-se como trabalho futuro uma forma de manter a identificação independente do deslocamento do objeto. Entretanto, como ponto positivo, o sistema foi capaz de rastrear objetos que anteriormente estavam oclusos, mesmo considerando a distorção das imagens planificadas.

Considerando a detecção e o rastreamento a precisão média foi de 34,72% para a classe carro, 66,33% para pessoa, 57,56% para moto, 16,67% para ônibus, 73,74% para bicicleta e 12,50% para a classe caminhão. Somando todas as precisões médias e dividindo pelo número de classes tem-se uma precisão total para o sistema de 43,58%.

Acredita-se que o sistema apresentaria resultados melhores se os vídeos usados para os testes fossem feitos em um mesmo local e com maior duração. Sugere-se como trabalho futuro a realização de um treinamento do YOLO com imagens 360° panorâmicas ou parte do vídeo,

para que o detector consiga localizar e classificar com maior precisão os objetos de interesse. Importante ressaltar que, no caso de se utilizar um vídeo para treinamento e teste, a parte usada para treinamento deverá ser diferente da parte utilizada para teste. Isto garante uma análise de desempenho válida, uma vez que a rede não será exposta a nenhuma informação visual na etapa de treinamento que esteja presente na etapa de teste.

Neste trabalho utilizou-se o YOLO e DeepSORT pré-treinados. Como trabalho futuro, sugere-se ainda a alteração dos hiperparâmetros da rede e o teste de outros rastreadores, como por exemplo, o FairMOT (ZHANG et al., 2021).

REFERÊNCIAS BIBLIOGRÁFICAS

ABOAH, A.; SHOMAN, M.; MANDAL, V.; DAVAMI, S.; ADU-GYAMFI, Y.; SHARMA, A. A Vision-based System for Traffic Anomaly Detection using Deep Learning and Decision Trees. **ArXiv**, v. abs/2104.06856, abril 2021. Disponível em: https://arxiv.org/ftp/arxiv/papers/2104/2104.06856.pdf. Acesso em: 10 de ago. de 2021.

BERNARDIN, K.; STIEFELHAGEN, R. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. **EURASIP Journal on Image and Video Processing**, [*s. l.*], v. 2008, 246309 (2008), maio 2008. Disponível em: https://doi.org/10.1155/2008/246309. Acesso em: 22 de set. de 2021.

CHOUDHURY, A. **Top 8 Algorithms for Object Detection**. 2020. Disponível em: https://analyticsindiamag.com/top-8-algorithms-for-object-detection/. Acesso em: 17 de fev. de 2022.

DARGAN, S.; KUMAR, M.; AYYAGARI, M. R.; KUMAR, G. A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning. **ARCH COMPUTAT METHODS ENG 27**, p. 1071-1092, set. 2020. Disponível em: https://doi.org/10.1007/s11831-019-09344-w. Acesso em: 10 de ago. de 2021.

DOAN, T. -N.; TRUONG, M. -T. Realtime Vehicle Detection and Counting Based on YOLO and DeepSORT. *In*: 12TH INTERNATIONAL CONFERENCE ON KNOWLEDGE AND SYSTEMS ENGINEERING (KSE), 2020, Can Tho, Vietnã. **Proceedings** [...]. Can Tho, Vietnã: IEEE, 2020, p. 67-72. Disponível em: https://doi.org/10.1109/KSE50997.2020.9287483. Acesso em: 13 de ago. de 2021.

DOSHI, K.; YILMAZ, Y. Fast Unsupervised Anomaly Detection in Traffic Videos. *In*: IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2020, Seattle. **Proceedings** [...]. Seattle: IEEE, 2020, p. 2658-2664. Disponível em: https://doi.org/10.109/CVPRW50498.2020.00320. Acesso em: 12 de ago. de 2021.

GANDHI, R. **R-CNN, Fast R-CNN, Faster R-CNN, YOLO – Object Detection Algorithms**. 2018. Disponível em: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e/. Acesso em: 17 de fev. de 2022.

GIRSHICK, R.; DONAHUE, R.; DARRELL, T.; MALIK, J. Rich feature Hierarchies for accurate object detection and semantic segmentation. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2014, [s. l.]. **Proceedings** […]. [S. l.]: IEEE, 2014. p. 580-587. DOI: 10.1109/CVPR.2014.81. Acesso em: 20 de fev. de 2022.

GIRSHICK, R. Fast R-CNN. *In*: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, 2015, [s. l.]. **Proceedings** […]. [S. l.]: IEEE, 2015. p. 1440-1448. DOI: 10.1109/ICCV.2015.169. Acesso em: 20 de fev. de 2022.

HE, K.; GKIOXARI, G.; DOLLÁR, P.; GIRSHICK, R. Mask R-CNN. *In*: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, 2017, Veneza. **Proceedings**

[…]. Veneza: IEEE, 2017. p. 2961-2969. Disponível em: https://doi.org/10.1109/ICCV.2017.322. Acesso em: 12 de ago. de 2021.

KHAN, A. I.; AL-HABSI, S. Machine Learning in Computer Vision. *In*: International Conference on Computational Intelligence and Data Science, 2019, Gurgaon. **Proceedings** [...]. Gurgaon: Editora, 2019. p. 15-15. Disponível em: https://www.researchgate.net/publication/340681873_Machine_Learning_in_Computer_Vision. Acesso em: 11 de ago. de 2021.

KHOSHELHAM, K. Computer Vision Techniques for Urban Mobility. In: WINTER, S.; GOEL, S. **Smart Parking in Fast-Growing Cities.** 1. ed. Wien: TU Wien Academic Press, 2021. p. 61-76. Disponível em: https://doi.org/10.34727/2021/isbn.978-3-85448-045-7. Acesso em: 10 de ago. de 2021.

LIU, W.; ANGUELOV, D.; ERHAN, D.; SZEGEDY, C.; REED, S.; FU, C. -Y.; BERG, A. C. SSD: Single Shot Multibox Detector. *In*: EUROPEAN CONFERENCE ON COMPUTER VISION, 2016, Amsterdã. **Proceedings** […]. Amsterdã: Springer, 2016. p. 21-37. Disponível em: https://doi.org/10.1007/978-3-319-46448-0_2. Acesso em: 11 de ago. de 2021.

LIU, W.; WANG, Z.; ZHOU, B.; YANG, S.; GONG, Z. Realtime Signal Light Detection based on YOLOv5 for Railway. **IOP CONFERENCE SERIES: EARTH AND ENVIRONMENTAL SCIENCE**, v. 769, n. 3, p. [S.p.], maio 2021. Disponível em: https://iopscience.iop.org/article/10.1088/1755-1315/769/4/042069. Acesso em: 14 de ago. de 2021.

LU, J. Y.; MA, C.; LI, L.; XING, X. Y.; ZHANG, Y.; WANG, Z. G.; XU, J. W. A Vehicle Detection Method for Aerial Image Based on YOLO. **Journal of Computer and Communications**, [S.l.], v. 6, n. 11, p. 98-107, nov. 2018. Disponível em: https://doi.org/10.4236/jcc.2018.611009. Acesso em: 14 de ago. de 2021.

MANDAL, V.; MUSSAH, A. R.; JIN, P.; ADU-GYAMFI, Y. Artificial Intelligence-Enabled Traffic Monitoring System. **Sustainability**, v. 12, n. 21, p. [S.p.], nov. 2020. Disponível em: https://doi.org/10.3390/su12219177. Acesso em: 12 de ago. de 2021.

MAYA, S. **DeepSORT: Learning to Track Custom Objects in a Video**. 2019. Disponível em: https://nanonets.com/blog/object-tracking-deepsort/#traditional-methods. Acesso em: 15 de março de 2022.

MILANO, D.; HONORATO, L. B. **Visão Computacional**. 2010. Artigo – Universidade Estadual de Campinas. Limeira, São Paulo, 2010. Disponível em: https://docplayer.com.br/3058305-Visao-computacional-danilo-de-milano-luciano-barrozo-honorato-unicamp-universidade-estadual-de-campinas-ft-faculdade-de-tecnologia.html. Acesso em: 20 de jun. de 2021.

MONTEMAYOR, A. S.; PANTRIGO, J. J.; SALGADO, L. Special Issue on Real-Time Computer Vision in Smart Cities. **Journal of Real-Time Image Processing,** 10(4), p. 723-724, 2015.

NOGUEIRA, A. B. **Análise de viabilidade no uso de Deep Learning para contagem de pessoas com câmeras de segurança**. 2018. 78 f. Trabalho de Conclusão de Curso (Graduação em Engenharia de Controle e Automação) – Curso de Engenharia de Controle e Automação, Departamento de Automação e Sistemas, Universidade Federal de Santa Catarina, Florianópolis, 2018.

NORVIG, P.; RUSSELL, S. **Inteligência Artificial: Tradução da 3ª Edição**. Elsevier Brasil, 2014.

PÁDUA, M. **Machine Learning – Métricas de avaliação: Acurácia, Precisão e Recall, F1-score**. 2020. Disponível em: https://medium.com/@mateuspdua/machine-learning-m%C3%A9tricas-de-avalia%C3%A7%C3%A3o-acur%C3%A1cia-precis%C3%A3o-e-recall-d44c72307959. Acesso em: 17 de set. de 2021.

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. You only look once: Unified, real-time object detection. *In*: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2016, [s. l.]. **Proceedings** […]. [S. l.]: IEEE, 2016. p. 779-788. Disponível em: https://www.cvfoundation. org/openaccess/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016 _paper.html. Acesso em: 11 de ago. de 2021.

REHEM, A.; TRINDADE, F. H. V. **Técnicas de Visão Computacional para Rastreamento de Olhar em Vídeos**. 2009. Disponível em: http://almerindo.devin.com.br/index.php?option=com_content&view=article&id=78%3Atecnicas-de-computacao-visual-para-rastreamento-de-olhar-em-videos&catid=43%3Atrabalhos-de-alunos&Itemid=86&showall=1 Acesso em: 20 de jun. de 2021.

REN, S.; HE, K.; GIRSHICK, R.; SUN, J. Faster R-CNN: Towards Realtime Object Detection with Region Proposal Networks. *In*: IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE LEARNING, 2015, Montreal. **Proceedings** [...]. Montreal: IEEE, 2015, vol. 39. p. 1137-1149. Disponível em: https://arxiv.org/pdf/1506.01497.pdf. Acesso em: 13 de ago. de 2021.

RICOH API. **Libuvc Theta.** 2015. Disponível em: https://github.com/ricohapi/libuvc-theta. Acesso em: 28 de março de 2022.

RICOH API. **Libuvc Theta Sample.** 2020. Disponível em: https://github.com/ricohapi/libuvc-theta-sample. Acesso em: 28 de março de 2022.

ROSEBROCK, A. **Deep Learning for Computer Vision with Python**. 1. ed. [S.l]: PyImageSearch.com, 2017. Disponível em: https://bayanbox.ir/view/5130918188419813120/Adrian-Rosebrock-Deep-Learning-for.pdf. Acesso em: 10 de ago. de 2021.

SHARMA, P. **Image Classification vs Object Detection vs Image Segmentation**. 2019. Disponível em: https://medium.com/analytics-vidhya/image-classification-vs-object-detection-vs-image-segmentation-f36db85fe81. Acesso em: 12 de ago. de 2021.

SHAFIEE, M. J.; CHYWL, B; LI, F.; WONG, A. **Fast YOLO: A Fast You Only Look Once System for Realtime Embedded Object Detection in Video**. 2017. Disponível em: https://arxiv.org/abs/1709.05943v1. Acesso em: 13 de ago. de 2021.

THINK AUTONOMOUS. **Computer Vision for Tracking**. 2019. Disponível em: https://www.thinkautonomous.ai/blog/?p=computer-vision-for-tracking. Acesso em: 15 de março de 2022.
THORNTON, A; HU, A. **V4l2loopback.** 2010. Disponível em: https://github.com/umlaeute/v4l2loopback. Acesso em: 28 de março de 2022.

ULTRALYTICS. **YOLOv5**. 2021. Disponível em: https://github.com/ultralytics/yolov5. Acesso em: 28 de março de 2022.

VENABLES, M. **An overview of Computer Vision**. 2019. Disponível em: https://towardsdatascience.com/an-overview-of-computer-vision-1f75c2ab1b66. Acesso em: 10 de ago. de 2021.

VOULODIMOS, A.; DOULAMIS, N.; DOULAMIS, A.; PROTOPAPADAKIS, E. Deep Learning for Computer Vision: A Brief Review. **COMPUTATIONAL INTELLIGENCE AND NEUROSCIENCE**, v. 2018, p. 13, fev. 2018. Disponível em: https://doi.org/10.1155/2018/7068349. Acesso em: 12 de ago. de 2021.

ZHANG, Y.; WANG, C.; WANG, X.; ZENG, W.; LIU, W. FairMOT: On the Fairness of Detection and Re-Identification in Multiple Object Tracking. **International Journal of Computer Vision**, [S.l.], v. 6, n. 129, p. 3069-3087, oct. 2021. Disponível em: https://doi.org/10.1007/s11263-021-01513-4. Acesso em: 30 de março de 2022.

ZHAO, C.; CHEN, B. Realtime Pedestrian Detection Based on Improved YOLO Model. *In*: 11TH INTERNATIONAL CONFERENCE ON INTELLIGENT HUMAN-MACHINE SYSTEMS AND CYBERNETICS (IHMSC), 2019, Hancheu, China. **Proceedings** [...]. Hancheu, China: IEEE, 2019, p. 25-28. Disponível em: https://doi.org/10.1109/IHMSC.2019.10101. Acesso em: 14 de ago. de 2021.

ZIQIANG, P. **DeepSORT Pytorch**. 2020. Disponível em: https://github.com/ZQPei/deep_sort_pytorch. Acesso em: 28 de março de 2022.

ZHAO, Z.; ZHENG, P.; XU, S.; WU, X. Object Detection with Deep Learning: A Review. **IEEE TRANSCTIONS ON NEURAL NETWORKS AN LEARNING SYSTEMS**, v. 30, n. 11., p. 3212-3232, jan. 2019. Disponível em: https://doi.org/10.1109/TNNLS.2018.2876865. Acesso em: 13 de ago. de 2021.

WOJKE, N.; BEWLEY, A.; PAULUS, D. Simple Online and Realtime Tracking with a Deep Association Metric. *In*: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2017, Beijing. **Proceedings** […]. Beijing: IEEE, 2017. p. 3645-3649. Disponível em: http://dx.doi.org/10.1109/icip.2017.8296962. Acesso em: 14 de ago. de 2021.