UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

Isabella Stefanny Fernandes de Andrade

Rastreamento de Pedestres 3D Multi-Câmera Usando Redes Neurais de Grafos

Recife

2022

Isabella Stefanny Fernandes de Andrade

Rastreamento de Pedestres 3D Multi-Câmera Usando Redes Neurais de Grafos

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Universidade Federal Rural de Pernambuco – UFRPE

Departamento de Computação

Curso de Bacharelado em Ciência da Computação

Orientador: João Paulo Silva do Monte Lima

Recife

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

D278r de Andrade, Isabella Stefanny Fernandes
Rastreamento de pedestres 3D multi-câmera usando redes neurais de grafos / Isabella Stefanny Fernandes de Andrade. - 2022.
43 f. : il.

Orientador: Joao Paulo Silva do Monte Lima.
Inclui referências.

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal Rural de Pernambuco, Bacharelado em Ciência da Computação, Recife, 2022.

1. Rastreamento. 2. Pedestres. 3. Redes neurais. 4. Multi-câmera. I. Lima, Joao Paulo Silva do Monte, orient. II. Título

CDD 004

MINISTÉRIO DA EDUCAÇÃO E DO DESPORTO
UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO (UFRPE)
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

http://www.bcc.ufrpe.br

Ficha de Aprovação do Trabalho de Conclusão de Curso

Trabalho defendido por **Isabella Stefanny Fernandes de Andrade** às 10 horas do dia 27 de maio de 2022, na sala de reunião online http://meet.google.com/ohj-gfdu-fie , como requisito para conclusão do curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco, intitulado **Rastreamento de Pedestres 3D Multi-Câmera Usando Redes Neurais de Grafos**, orientado pelo Prof. **João Paulo Silva do Monte Lima** e aprovado pela seguinte banca examinadora:

João Paulo Silva do Monte Lima

DC/UFRPE

Filipe Rolim Cordeiro

DC/UFRPE

Aos meus pais que sempre apoiaram a minha educação.

Agradecimentos

Agradeço aos meus pais, Kátia e Arlindo, que sempre me deram todo o suporte necessário para que eu tivesse uma boa educação e respeitaram todas as minhas escolhas.

Agradeço imensamente aos meus amigos Clara e Yves que me proporcionaram muitos momentos de descontração principalmente quando saíamos para jogar, mas também quando me confortavam em momentos difíceis.

Agradeço a todos os colegas da faculdade com quem estudei junto, particularmente a Heriberto que me ajudou em praticamente metade das disciplinas do curso.

Agradeço a todos do departamento de computação, sobretudo aos professores Ana Paula, Leandro, Suzana e Péricles pelo aprendizado oferecido e pela dedicação em metodologias de ensino, e à Sandra que sempre se dispôs a ajudar os alunos.

Agradeço a todos com quem tive contato durante a minha pesquisa e especialmente ao meu orientador João Paulo, que me proporcionou a oportunidade de aprender um assunto interessante e atual, e me guiou com excelência em meio a um assunto ao qual eu não tinha muito conhecimento.

“Quando eu era uma criança,
Meu coração era sempre um mar sem fim
Mas agora apenas traços permanecem naquele lugar tênue

O som da minha respiração, que era preenchido com entusiasmo
O vento que soprava sobre minha cabeça
Eu quero me tornar uma onda e correr para qualquer lugar
Com um pouco de medo, se eu abrir meus olhos lentamente

Eu farei com que cada momento nesse mundo
Se torne um presente deslumbrante
E acho que a versão de mim que ficava duvidando de si mesma
Poderá finalmente encontrar as respostas”
(Lee Ji-eun)

Resumo

Rastrear a posição de pedestres ao longo do tempo através de imagens de câmeras é um tópico de pesquisa em visão computacional em ascensão. No cenário multi-câmera, as pesquisas são mais recentes ainda. Muitas soluções utilizam redes neurais supervisionadas para resolver esse problema, o que pode exigir um esforço muito grande para anotar os dados além de muito tempo gasto para treinar a rede. Os objetivos deste trabalho são: desenvolver variações de algoritmos de rastreamento de pedestres sendo desejável dispensar a necessidade de possuir dados anotados; e comparar os resultados obtidos através de métricas de acurácia. Este trabalho propõe, portanto, uma abordagem para rastrear pedestres no espaço 3D em ambientes multi-câmera utilizando a arquitetura de rede neural *Message Passing Neural Network* inspirada em grafos. Avaliamos a solução utilizando a base de dados WILDTRACK e um método de detecção generalizável, conseguindo 77,1% de MOTA ao treinar com dados obtidos de um algoritmo de rastreamento generalizável. O algoritmo consegue realizar o rastreamento a uma taxa de 40 quadros por segundo.

Palavras-chave: rastreamento, pedestres, redes neurais, multi-câmera.

Abstract

Tracking the position of pedestrians over time through camera images is a rising computer vision research topic. In multi-camera settings, the researches are even more recent. Many solutions use supervised neural networks to solve this problem, which can require a lot of effort to annotate the data in addition to a lot of time spent to train the network. The goals of this work are: develop variations of pedestrian tracking algorithms, being desirable to avoid the need to have annotated data; and compare the results obtained through accuracy metrics. Therefore, this work proposes an approach for tracking pedestrians in 3D space in multi-camera environments using the Message Passing Neural Network framework inspired by graphs. We evaluated the solution using the WILDTRACK dataset and a generalizable detection method, reaching 77.1% of MOTA when training with data obtained by a generalizable tracking algorithm. The algorithm can track at a 40 frames per second rate.

Keywords: tracking, pedestrians, neural networks, multi-camera.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

MFT	Model-free-tracking
TBD	Tracking-by-detection
MPNN	Message Passing Neural Network
MOTA	Multiple Object Tracking Accuracy
MOTP	Multiple Object Tracking Precision
Re-ID	Reidentificação de pessoas
ReLU	Unidade linear retificada
MLP	Perceptron Multicamadas

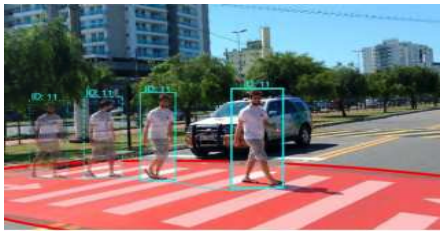
Sumário

1 Introducão

O rastreamento de pedestres é um problema de visão computacional que consiste em localizar e atribuir uma identidade para cada pessoa ao longo de um vídeo ou uma sequência de imagens. Esse tema de pesquisa recebe muita atenção por ser uma das tarefas do sistema de percepção presente em veículos autônomos (BADUE et al., 2021), podendo auxiliar na análise de comportamento de indivíduos e na vigilância por vídeo (ZHANG; YU; YU, 2018), entre outras aplicações. A Figura 1 ilustra alguns exemplos.

(a) O rastreamento pode gerar dados úteis para identificar padrões de rotas de pedestres. A imagem mostra uma cena retirada da base de dados Pedestrian Walking Route que possui anotações sobre caminhos que pedestres fazem frequentemente. Fonte: Yi, Li & Wang (2015) (YI; LI; WANG, 2015).

(b) A evacuação de pessoas pode ser identificada utilizando algoritmos de rastreamento de pedestres. A imagem mostra uma cena onde existe uma evacuação retirada da base de dados PETS 2009. Fonte: Ferryman & Shahrokni (2009) (FERRYMAN; SHAHROKNI, 2009).



(c) Ilustração de um pedestre sendo rastreado ao atravessar uma rua na frente de um carro. Fonte: Sarcinelli *et al.* (2019) (SARCINELLI et al., 2019).

Figura 1 – Exemplos de aplicação do rastreamento de pedestres.

As duas formas mais comuns para começar a rastrear os pedestres são *model-free-tracking* (MFT) e *tracking-by-detection* (TBD) (SUN et al., 2020). No MFT, é necessário que os pedestres do primeiro quadro sejam indicados manualmente, e então esses pedestres são rastreados nos próximos quadros, o que torna esse método limitado por não lidar com a variação do número de pedestres ao longo do tempo. Em contrapartida, no TBD os pedestres são detectados independentemente em cada quadro,

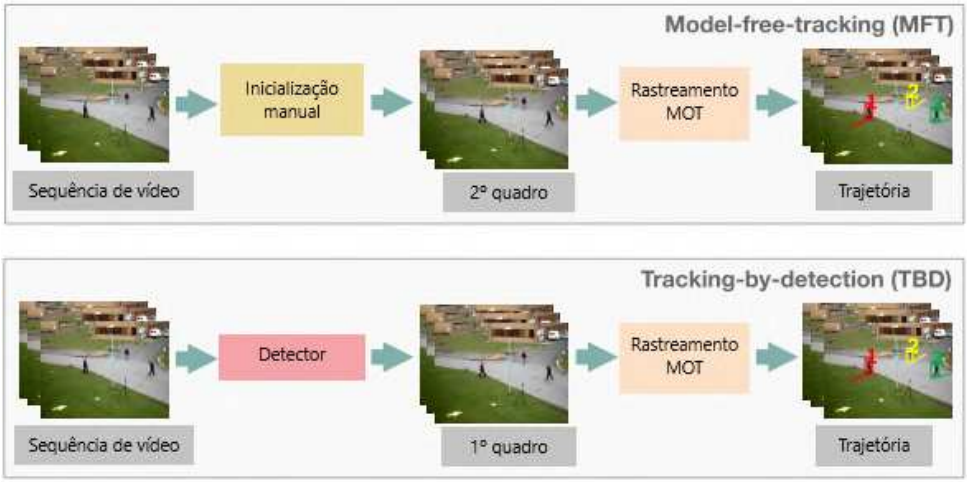


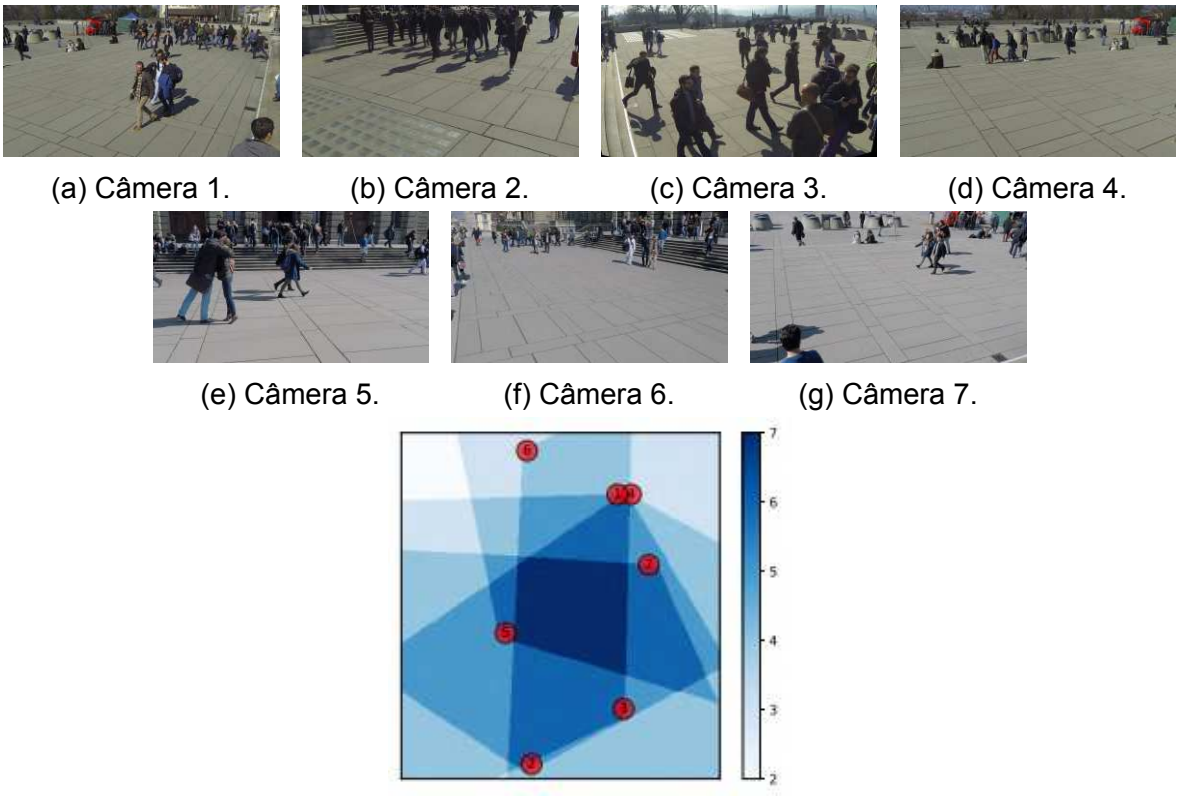
Figura 2 – Ilustração dos tipos de rastreamento mais comuns. Fonte: Sun *et al.* (2020) (SUN et al., 2020). Tradução nossa.

e depois é criada uma relação de identidade entre as pessoas de quadros diferentes de forma a traçar as trajetórias dos pedestres. A Figura 2 apresenta visualmente as diferenças entre o MFT e o TBD.

A precisão do rastreamento baseado em TBD depende fortemente da qualidade das detecções (HENSCHEL; ZOU; ROSENHAHN, 2019), porém o desempenho dos detectores tem melhorado significativamente como resultado dos desafios anuais de detecção de múltiplos objetos realizados em bases de dados como a MS-COCO (LIN et al., 2014). Por exemplo, o MMDet venceu o desafio de 2018 atingindo 48,60% de precisão na detecção de pedestres (CHEN et al., 2019), sendo 2,3% superior ao PANet que venceu o desafio em 2017 (LIU et al., 2018), reforçando a tendência no uso de TBD para o rastreamento de pedestres.

Outro item que diferencia as abordagens de rastreamento de pedestres se baseia na quantidade de câmeras utilizadas, podendo ser uma ou múltiplas câmeras. Um exemplo de método que utiliza apenas uma câmera é o proposto por Brasó & Leal-Taixé (2020) (BRASÓ; LEAL-TAIXÉ, 2020), que obteve 58,8% de precisão no rastreamento na base de dados MOT17 (MILAN et al., 2016). Enquanto isso, os métodos que utilizam múltiplas câmeras lidam melhor com oclusões, aumentando a confiabilidade das soluções (CHAVDAROVA; FLEURET, 2017).

Buscando incentivar as pesquisas que utilizam essa abordagem, Chavdarova *et al.* (2018) tornaram pública a base de dados WILDTRACK, que possui sequências de 400 imagens com anotações das posições reais dos pedestres provenientes de 7 câmeras diferentes (CHAVDAROVA et al., 2018), e é desafiadora por conter um grande número de pessoas. A Figura 3 mostra as vistas das 7 câmeras e a sobreposição delas. O algoritmo proposto por Brasó & Leal-Taixé (2020), quando utilizado apenas na primeira câmera do WILDTRACK, possui um desempenho de 45,5% de precisão, um



(a) Câmera 1. (b) Câmera 2. (c) Câmera 3. (d) Câmera 4.

(e) Câmera 5. (f) Câmera 6. (g) Câmera 7.

(h) Sobreposição das câmeras.

Figura 3 – Vistas sincronizadas das 7 câmeras presentes na base de dados WILDTRACK e a sua sobreposição. Fonte: Chavdarova *et al.* (2018) (CHAVDAROVA et al., 2018).

pouco inferior ao do MOT17 por lidar com muitas oclusões e não explorar as vantagens do ambiente multi-câmera.

Contudo, com diferentes fontes de imagens a complexidade para reidentificar os pedestres e o poder computacional necessário também crescem. Muitas soluções foram desenvolvidas para funcionar de forma *offline*, em que o algoritmo só consegue processar uma sequência de imagens que já esteja completa, inviabilizando aplicações em tempo real que possibilitam interações à medida que as imagens são obtidas. Um exemplo disso é uma das soluções do estado da arte proposta por Vo *et al.* (2020), que utiliza técnicas de aprendizagem profunda e requer treinamento na base de dados em que será aplicada, o que também pode consumir muito tempo (VO et al., 2020).

As etapas que são utilizadas para realizar o rastreamento baseado em TBD podem ser divididas em quatro categorias: i) aplicar métodos de associação entre as detecções; ii) incluir técnicas de visão computacional que já têm sido estudadas para resolver outros problemas que podem ser relacionados ao rastreamento; iii) usar aprendizagem profunda para extrair características de pedestres que podem ser comparadas ou até realizar o rastreamento inteiro com modelos *end-to-end*; iv) usar diferentes tipos de dados para agregar mais informações sobre os pedestres (SUN et al., 2020).

Após desenvolver uma solução, também é importante realizar uma comparação com outros trabalhos. A avaliação comparativa de algoritmos de rastreamento de múltiplos objetos é considerada uma tarefa difícil devido aos seus resultados diversos (BERNARDIN; STIEFELHAGEN, 2008), que podem incluir falhas, confusões, falsos positivos e falsos negativos para cada objeto. Por outro lado, é uma missão indispensável entender o quão efetivo é um algoritmo e poder compará-lo com outros (LEAL-TAIXÉ et al., 2017).

No entanto, é fundamental analisar a literatura existente sobre técnicas de avaliação para compreender suas fragilidades e potencialidades, bem como experimentá-las no cenário específico de rastreamento de pedestres.

1.1 Objetivos do Trabalho

Esse trabalho propõe a implementação e o refinamento de variações de algoritmos para rastrear pedestres no espaço 3D se baseando na técnica TBD proposta por Brasó & Leal-Taixé (2020), modificando para que esta seja utilizada no ambiente multi-câmera. Serão utilizadas as detecções obtidas pela solução multi-câmera de Lima *et al.* (2021), que é generalizável por não precisar de treinamento na base de dados utilizada (LIMA et al., 2021). Além disso, também será realizada uma avaliação comparativa entre as variações implementadas utilizando métricas de acurácia. Portanto, os objetivos deste trabalho são:

- Geral: Desenvolver um algoritmo de rastreamento de pedestres 3D utilizando as detecções de Lima *et al.* (2021).
- Específicos:
  1. Desenvolver uma versão multi-câmera do algoritmo proposto por Brasó & Leal-Taixé (2020) baseado em TBD;
  2. Realizar avaliações comparativas entre variações do algoritmo implementado levando-se em conta métricas de acurácia.

1.2 Visão Geral

Os próximos capítulos desse documento estão divididos na estrutura a seguir. No Capítulo 2 os trabalhos relacionados são apresentados. No Capítulo 3 alguns conceitos teóricos que foram utilizados no trabalho são explicados. No Capítulo 4 a metodologia da pesquisa é especificada. No Capítulo 5 os resultados obtidos são relatados e discutidos. Por fim, no Capítulo 6 o trabalho é concluído resumindo os principais pontos e sugerindo trabalhos futuros.

2 Trabalhos Relacionados

O rastreamento de pedestres é um problema de rastreamento de múltiplos objetos (LUO et al., 2020) estudado em visão computacional e tem aplicações em diversos contextos, como vigilância e direção autônoma. Este trabalho tem como objetivo contribuir para o rastreamento 3D da posição de pedestres a partir de múltiplas câmeras considerando medidas de qualidade, visando avaliar com precisão os algoritmos atuais desta área e suas variações.

A base de dados WILDTRACK (CHAVDAROVA et al., 2018) é considerada desafiadora pela ocorrência de muitas oclusões, contudo isso também a torna semelhante a casos reais. Ela possui imagens de sete câmeras diferentes com visão sobreposta e anotações das localizações dos pedestres em 400 quadros a uma taxa de dois quadros por segundo.

O algoritmo K-Shortest Paths (KSP) (BERCLAZ et al., 2011) utilizou um mapa de probabilidade de ocupação para criar um grafo acíclico direcionado e encontrar as trajetórias mais prováveis dos pedestres. O KSP também supõe que a qualquer momento podem aparecer novos pedestres ou algum pedestre pode sair da cena, por isso ele conecta um vértice chamado v_{fonte} a todas as posições e conecta todas as posições a um vértice chamado v_{saida}. O grafo do KSP pode ser observado na Figura 4. A técnica ptrack, por outro lado, foi utilizada para aprender padrões de comportamento e pode ser usada em conjunto com diferentes algoritmos de rastreamento para melhorar seu desempenho (MAKSAI et al., 2017). Chavdarova *et al.* (2018) utilizaram os dois métodos em conjunto para obter um resultado satisfatório no WILDTRACK. No entanto, essa solução funciona apenas de maneira *offline*, não atendendo às necessidades de aplicações de tempo real (CHAVDAROVA et al., 2018).

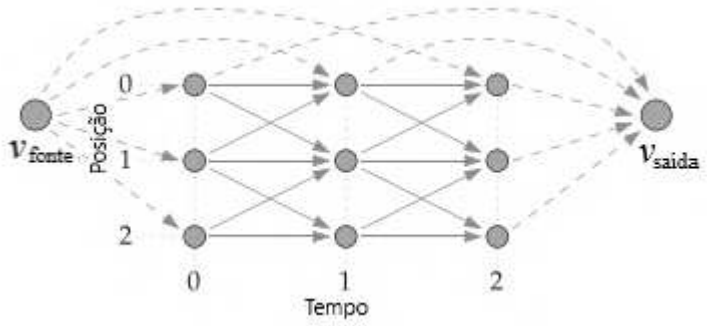


Figura 4 – Ilustração de um grafo acíclico direcionado do KSP contendo três posições durante três quadros e os vértices v_{fonte} e v_{saida}. Fonte: Berclaz *et al.* (2011) (BERCLAZ et al., 2011). Tradução nossa.

Muitos trabalhos recentes para esse tipo de problema utilizam aprendizagem profunda para rastrear pedestres. Vo *et al.* (2020) treinaram um descritor da aparência dos pedestres de forma não supervisionada e o utilizaram para fazer o rastreamento (VO et al., 2020).

You & Jiang (2020) construíram uma arquitetura *end-to-end* que usa uma rede neural para estimar as localizações dos pedestres e utilizaram as localizações como entrada para o rastreador (YOU; JIANG, 2020). O rastreamento é feito majoritariamente pela própria localização dos pedestres, porém também foi demonstrado que incluir características visuais como histogramas de cor trouxe benefícios para essa solução.

Brasó & Leal-Taixé (2020) utilizaram redes neurais que passam mensagens em grafos, chamadas *Message Passing Neural Networks* (MPNN), de forma adaptada ao problema de rastrear pedestres (BRASÓ; LEAL-TAIXÉ, 2020). Essa rede foi treinada com as características da caixa que delimita o pedestre detectado, como posição e tamanho, e com vetores de características visuais obtidos por uma rede neural, e rastreia os pedestres considerando apenas uma câmera. A Figura 5 apresenta os passos desse modelo.

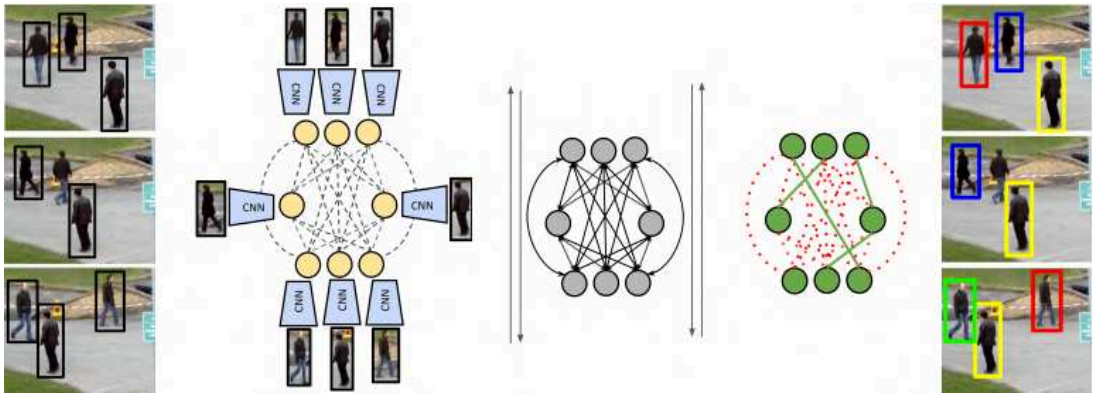


Figura 5 – Modelo proposto por Brasó & Leal-Taixé (2020). Primeiro as detecções são utilizadas como entrada, e então o modelo extrai características visuais de cada detecção e constrói um grafo. Depois as características são compartilhadas pelo grafo, e em seguida a rede realiza a classificação das conexões para determinar quais são de um mesmo pedestre. Fonte: Brasó & Leal-Taixé (2020) (BRASÓ; LEAL-TAIXÉ, 2020).

Esses algoritmos precisam de treinamento na base de dados em que serão utilizados, por isso caso haja uma mudança de cenário é realizado um novo treinamento para que a solução continue obtendo um resultado satisfatório. Porém, esse processo costuma ou precisar de muitos dados anotados ou consumir muito tempo, tornando desejável o desenvolvimento de um algoritmo generalizável. O algoritmo é generalizável se não precisar ser treinado novamente em cada base de dados que será utilizado, ou seja, mesmo que ele seja aplicado em um novo domínio ele continua funcionando sem ter sido treinado nesse domínio.

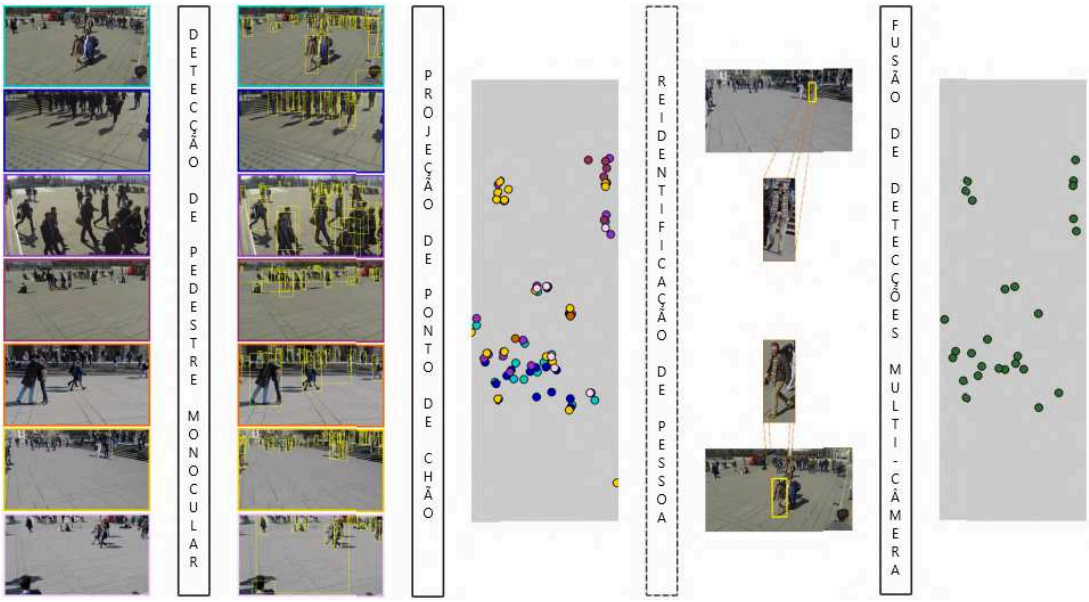


Figura 6 – Solução generalizável de Lima *et al.* (2021) para detectar pedestres. Ao receber as imagens como entrada, os pedestres e seus pontos de chão são detectados em cada câmera, depois os pontos de chão são projetados no plano de chão e o algoritmo realiza a fusão para obter um ponto final para cada pedestre. Fonte: Lima *et al.* (2021) (LIMA et al., 2021). Tradução nossa.

O trabalho de Lima *et al.* (2021) é uma solução generalizável que utiliza detecções próximas de pedestres em diferentes câmeras para construir um grafo e realizar a fusão, obtendo assim a coordenada 3D de cada pedestre (LIMA et al., 2021). O processo que esse trabalho utiliza é ilustrado na Figura 6. Contudo, essa solução não mantém uma relação temporal entre as detecções.

Lyra *et al.* (2022) propuseram um algoritmo de rastreamento *online* generalizável que cria um grafo bipartido entre detecções de quadros consecutivos (LYRA et al., 2022). Cada aresta possui como peso a distância entre as duas detecções e as detecções são conectadas a um mesmo pedestre através de algoritmos clássicos de associação de grafos. O peso da aresta pode utilizar diferentes informações, como a distância métrica e a distância visual das detecções, porém essas informações são relacionadas de forma artesanal, quando o computador poderia estar aprendendo a como relacionar da melhor forma. Além disso, durante o rastreamento o algoritmo utiliza o filtro de Kalman para prever a posição em que os pedestres que não foram encontrados no quadro atual estariam. A Figura 7 mostra as etapas desse trabalho.

Portanto, a contribuição esperada do presente trabalho é criar um algoritmo que associe as detecções obtidas por Lima *et al.* (2021) (LIMA et al., 2021) de um mesmo pedestre através de vários quadros, de forma a estabelecer a trajetória de pedestres no plano 3D, utilizando técnicas recentes de aprendizagem profunda. Em particular, foi implementado um algoritmo baseado no trabalho de Brasó & Leal-Taixé

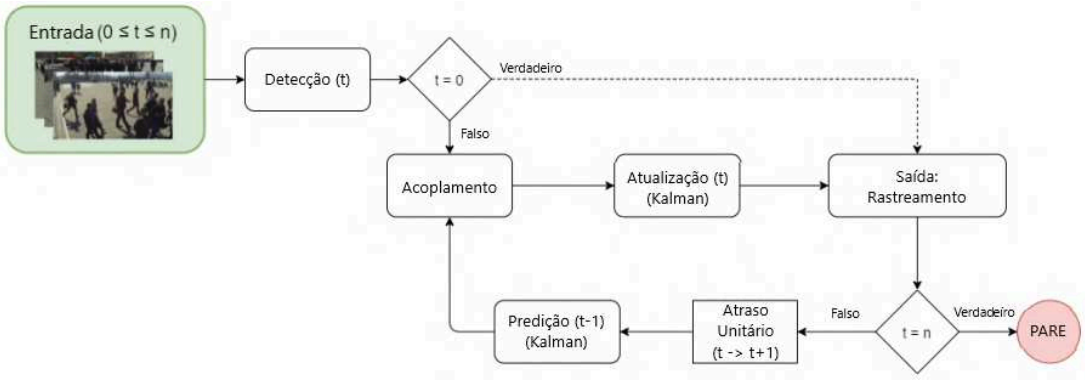


Figura 7 – Rastreamento realizado por Lyra *et al.* (2022), onde *t* é um instante de tempo e *n* é o número de quadros no vídeo. Fonte: Lyra *et al.* (2022) (LYRA et al., 2022). Tradução nossa.

(2020) (BRASÓ; LEAL-TAIXÉ, 2020), porém a rede neural foi adaptada para utilizar as detecções 3D obtidas em um ambiente multi-câmera para aproveitar as vantagens deste.

Quanto à avaliação, Bernardin & Stiefelhagen (2008) propuseram as métricas *Multiple Object Tracking Accuracy* (MOTA) e *Multiple Object Tracking Precision* (MOTP), que ajudam a comparar a precisão de localização de objetos, a precisão de reconhecer configurações de objetos e sua capacidade de rastrear objetos ao longo do tempo (BERNARDIN; STIEFELHAGEN, 2008). Essas métricas são generalizáveis e podem ser aplicadas em uma variedade de cenários.

Luo *et al.* (2020) resumiu os métodos da literatura para avaliar o rastreamento de múltiplos objetos, incluindo uma lista detalhada de métricas divididas em detecção e rastreamento (LUO et al., 2020). Nesse estudo foi concluído que a métrica MOTA ainda é a medida mais amplamente aceita para o rastreamento de múltiplos objetos. Além disso, Leal-Taixé *et al.* (2017) analisou o contraste entre a visão humana e a computacional, criando uma avaliação com ambas, e MOTA foi considerada a métrica que mais se alinha com a visão humana (LEAL-TAIXÉ et al., 2017).

3 Fundamentação Teórica

Na literatura existem diferentes técnicas que podem ser utilizadas para o rastreamento de pedestres. Neste trabalho, foram exploradas técnicas relacionadas a aprendizagem profunda para extrair características e associar as detecções, e esse capítulo explica elas nas seções 3.1 e 3.2 respectivamente.

3.1 Reidentificação de Pessoas

Associar uma pessoa através de vistas de câmeras em diferentes locais e tempos é uma tarefa conhecida como reidentificação de pessoas (Re-ID). Independente das diferenças na iluminação ou no fundo das imagens, um sistema computacional de Re-ID pretende reconhecer indivíduos automaticamente, sendo potencialmente uma forma mais econômica e precisa do que um trabalho manual (GONG et al., 2014).

Uma técnica recente de Re-ID se baseia em utilizar aprendizagem profunda para extrair características da imagem do pedestre. Dada uma caixa limitante de um pedestre, a rede neural devolve um vetor numérico que representa as características que a rede aprendeu sobre aquele pedestre. A Figura 8 apresenta uma visão geral sobre como esse vetor é obtido.

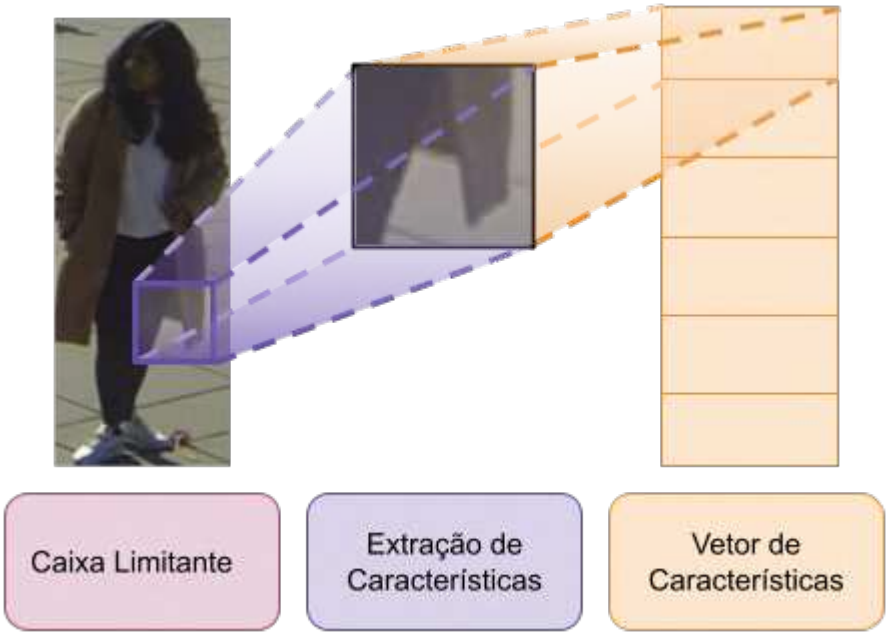


Figura 8 – Demonstração de como as características de um pedestre são obtidas. Fonte: elaborado pela autora.

Ao obter esse vetor para cada pedestre em cada câmera, é possível calcular a distância entre a aparência de cada detecção e a utilizar como uma informação a mais para determinar a probabilidade de aquelas detecções pertencerem à mesma pessoa.

3.2 Redes Neurais

De maneira sucinta, as redes neurais são modelos matemáticos que, a partir de dados de entrada, aplicam um conjunto de funções para extrair outras informações.

Porém, a sua característica mais importante é que elas são inspiradas pelo funcionamento do cérebro humano. As redes neurais são formadas por vários neurônios que atuam em conjunto para resolver um problema, e para isso elas devem ser treinadas para aprender as funções mais adequadas, aquelas que irão fornecer o resultado mais próximo do desejado.

O treinamento pode ser realizado de forma supervisionada ou não-supervisionada. No primeiro caso, a saída desejada para cada entrada é conhecida, portanto o treinamento ocorre de forma a aproximar o resultado obtido do resultado correto. Já no segundo caso, não há uma saída conhecida para as entradas, e o algoritmo se preocupa em aprender como agrupar os dados que são mais parecidos.

Este trabalho foca na aprendizagem supervisionada para realizar a associação entre pedestres. Nas próximas subseções serão apresentados os modelos de redes neurais que serão utilizados.

3.2.1 Perceptron Multicamadas

O modelo perceptron com apenas um neurônio é o modelo mais simples no aprendizado de máquina, e será introduzido para auxiliar na definição do perceptron multicamadas.

Dada uma sequência $x_1, x_2, ..., x_n$ de n entradas, existe uma sequência de pesos $w_1, w_2, ..., w_n$ e uma sequência de deslocamentos $b_1, b_2, ..., b_n$, que levam as entradas até o neurônio de saída através da equação

$$z = \sum_{i=0}^{n} w_i x_i + b_i, \tag{3.1}$$

onde cada entrada é multiplicada pelo seu peso e somada ao seu deslocamento, e após obter esse resultado para todas as entradas eles são somados gerando apenas um resultado z.

Após esse neurônio, o resultado z ainda passa por uma função de ativação para

produzir a predição final $\hat{y}$, de acordo com a equação

$$\hat{y} = f_{act}(z), \tag{3.2}$$

em que alguns exemplos de função f_{act} são a função sigmoide, caso o objetivo seja uma classificação binária, e a função linear, caso o objetivo seja uma classificação multi-classe. A Figura 9 resume o processamento do perceptron de um neurônio.

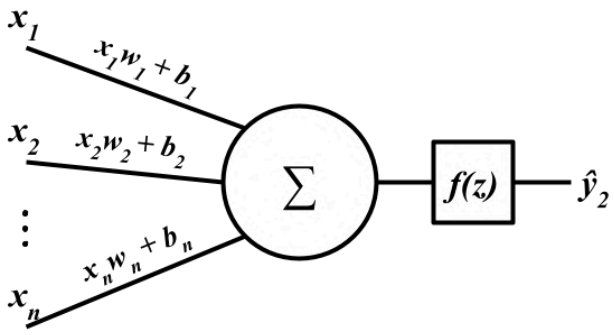


Figura 9 – Imagem ilustrativa do perceptron de um neurônio. Fonte: elaborado pela autora.

Outro exemplo de função de ativação muito utilizada nos trabalhos mais recentes é a unidade linear retificada (*rectified linear unit* - ReLU), definida pela equação

$$f_{act}(z) = max(0, z), \tag{3.3}$$

onde os valores negativos são substituídos por zero e os valores maiores ou iguais a zero permanecem os mesmos.

Como esse algoritmo é baseado na aprendizagem supervisionada, a saída correta y pode ser utilizada para calcular o erro da saída $\hat{y}$ obtida pelo algoritmo. Técnicas como gradiente descendente são aplicadas para encontrar os parâmetros w e b que tornam a função da Equação 3.1 mais próxima da função que separa o conjunto de dados.

Contudo, essa função é linear, e por isso ela só consegue descrever um conjunto de dados que seja linearmente separável, o que não é suficiente para resolver problemas complexos em um cenário real. Como demonstrado na Figura 10, ao tentar separar um conjunto de itens verdes de um conjunto de itens vermelhos apenas traçando uma reta, alguns itens são classificados incorretamente pois os dados não são lineares.

Uma solução para isso é adicionar mais camadas ao perceptron, criando assim o perceptron multicamadas (*multi-layer perceptron* - MLP). Cada camada pode conter vários neurônios em paralelo, e a saída de uma camada é utilizada como entrada da

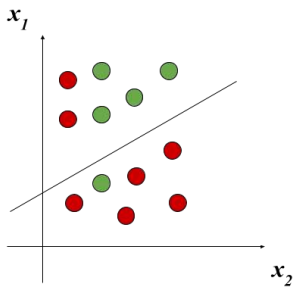


Figura 10 – Gráfico que demonstra a tentativa de utilizar uma função linear para separar um conjunto de dados não-lineares. Fonte: elaborado pela autora.

próxima, processo chamado de *feedforward*. A primeira camada é chamada de camada de entrada, a última camada é chamada de camada de saída, e todas as camadas entre elas são chamadas de camadas escondidas.

Na Figura 11 é possível ver como exemplo um MLP com uma camada escondida de três neurônios e uma camada de saída de dois neurônios. Além disso, cada camada possui sua própria função de ativação.

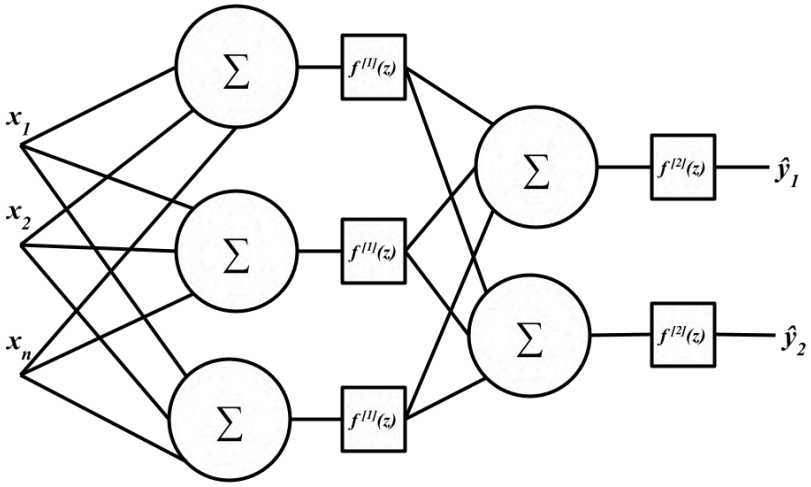


Figura 11 – Imagem ilustrativa de um perceptron multicamadas. Fonte: elaborado pela autora.

O aprendizado no MLP é chamado de *backpropagation*, pois o erro é propagado de trás para frente, começando na camada de saída e indo de camada em camada até chegar na camada escondida inicial.

3.2.2 Rede Neural de Grafos

O grafo é uma estrutura de dados que pode ser utilizada para representar dados que possuem conexões entre si. Ele é composto por vértices e arestas, em que cada

aresta conecta dois vértices. Tanto os vértices quanto as arestas podem ter propriedades. Por exemplo, os vértices podem ter características associadas, e as arestas podem representar a distância entre os vértices que estão em suas extremidades. Na Figura 12 um grafo é ilustrado como exemplo. Ele possui três vértices representando cidades, onde cada vértice possui como propriedade a sua população e as arestas entre os vértices possuem como propriedade a distância entre essas cidades.

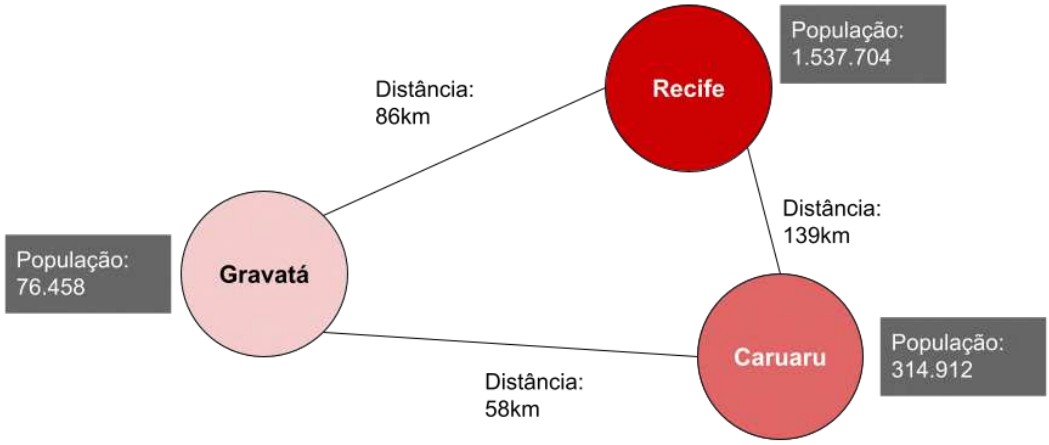


Figura 12 – Exemplo de grafo mostrando a relação entre três cidades. Fonte: elaborado pela autora.

Essa estrutura possibilita o uso de algoritmos que incluam na descrição de objetos informações sobre seu contexto, o que motiva o desenvolvimento de redes neurais específicas para esse fim. A arquitetura *Message Passing Neural Network* (MPNN) é um exemplo de como as redes neurais podem maximizar o potencial dos grafos.

Na rede MPNN, cada vértice v_i tem um estado oculto h_i, que é inicializado com alguma característica que ele possua. Em seguida, para cada vértice uma função de agregação é utilizada para sumarizar o estado oculto de todos os vértices vizinhos e de suas arestas:

$$m_i^l = \sum_{j \in N(i)} f_{agg}(h_i^{l-1}, h_j^{l-1}, e_{(i,j)}^{l-1}), \tag{3.4}$$

em que h_i^{l-1} e h_j^{l-1} são estados ocultos de dois vértices v_i e v_j conectados pela aresta $e_{(i,j)}^{l-1}$, e a função é chamada para cada vértice j que pertence aos vizinhos $N(i)$, para no fim realizar a soma.

O resultado é uma mensagem que é utilizada para atualizar o estado oculto do vértice v_i dessa forma:

$$h_i^l = f_{update}(h_i^{l-1}, m_i^l), \tag{3.5}$$

onde f_{update} pode ser por exemplo a função soma ou média, e o estado h_i^l é atualizado com o seu estado passado h_i^{l-1} e a mensagem m_i^l.

A Figura 13 resume esse processo utilizando como exemplo o grafo da Figura 12. Supondo que fosse importante agregar informações sobre a população ao redor de Recife, a função f_{agg} obtém a população de uma cidade ligada a Recife relativa à distância ao dividir a quantidade de habitantes pela distância, e soma à população de Recife. Isso acontece para cada nó conectado a Recife, e a média desses resultados é a mensagem m_i^l. Então, a função f_{update} calcula a média entre a população de Recife e a mensagem m_i^l, e atualiza o valor da população de Recife com esse resultado. A função média foi utilizada para facilitar o entendimento, porém ao utilizar MPNN é comum incluir funções que podem aprender a minimizar o erro, como MLP.

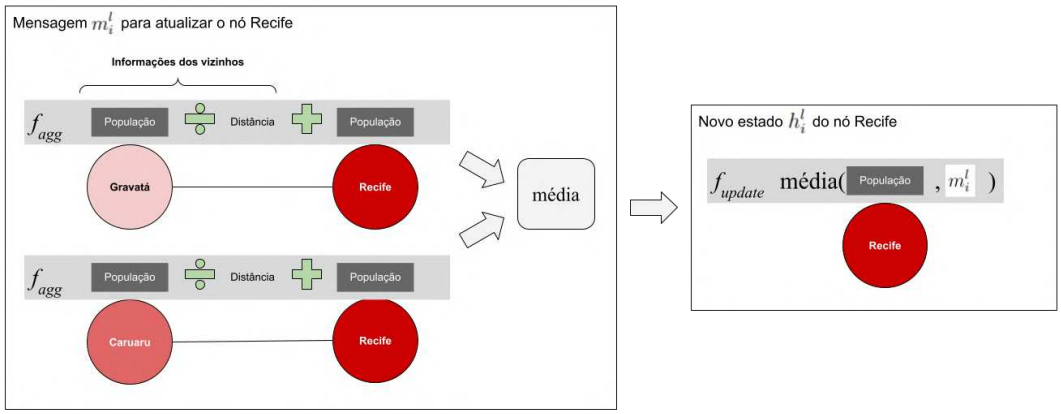


Figura 13 – Processo de obter informações dos nós vizinhos de Recife para atualizar o valor da população de Recife levando em conta a população de cidades ao redor. Fonte: elaborado pela autora.

Esses passos são repetidos L vezes, em que a cada vez a variável l é alterada. As equações 3.4 e 3.5 só consideram l e $l-1$ para atualizar o contexto, porém como o estado de l é sempre atualizado pelo estado de $l-1$, a mensagem é propagada pelo grafo de maneira eficiente em todos os vértices que possuam no máximo a distância L.

3.2.3 Rede Neural Convolucional

As imagens costumam ser representadas na forma de matrizes no computador. No caso de imagens coloridas, podem existir três matrizes representando cada canal entre vermelho (*red* - R), verde (*green* - G) e azul (*blue* - B) por exemplo. Cada elemento da matriz possui um número indicando a intensidade da cor naquela posição. A junção do elemento de cada matriz que está na mesma posição é chamada de pixel. A Figura 14 mostra um exemplo.

Diferentemente de dados lineares, em imagens existe uma dependência entre um pixel e os que estão ao seu redor. Portanto, para processar uma imagem, em vez de achatar as suas matrizes e formar um vetor que pode ser utilizado em camadas

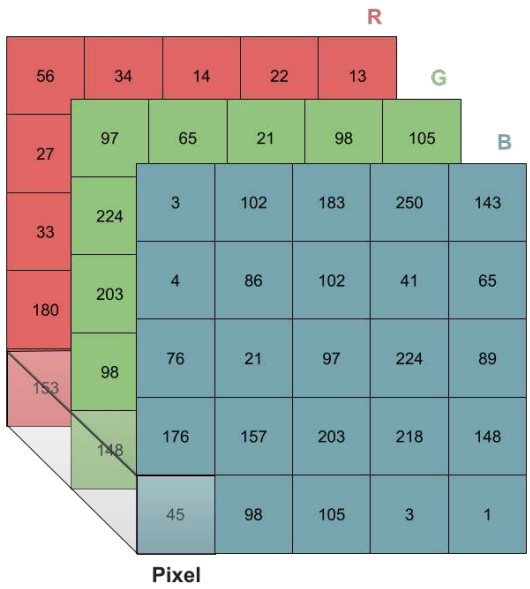


Figura 14 – Ilustração de como uma imagem colorida é representada no espaço RGB. Fonte: elaborado pela autora.

lineares como as descritas na MLP, é interessante manter o contexto daqueles pixels durante o processamento.

As redes neurais convolucionais conseguem preservar o contexto dos pixels com um custo eficiente ao aplicar filtros sobre regiões de pixels. Um filtro é uma matriz que possui um peso em cada posição. O filtro é colocado inicialmente no canto superior esquerdo de cada canal, e a rede neural calcula o produto entre cada elemento que esteja na mesma posição (x, y) no canal da imagem e no filtro, e em seguida faz a soma. Essa operação é repetida em cada canal da imagem e os resultados são somados para obter o primeiro elemento da matriz de saída do filtro convolucional.

A rede convolucional desloca o filtro para a direita e realiza o mesmo procedimento. A quantidade de posições que o filtro é deslocado por vez pode ser chamada de passo da rede convolucional e é ajustável. Quando o filtro chega ao fim de uma linha, ele volta para a esquerda e é deslocado para baixo com o mesmo passo. A Figura 15 ilustra como um filtro 3x3 seria deslocado no canal R durante quatro deslocamentos com passo de convolução 1. Adicionalmente, a rede pode incluir um preenchimento ao redor da imagem antes de filtrar, o que mudaria o tamanho da matriz de saída da convolução.

Cada camada convolucional pode ter vários filtros para aprender características diferentes, em que cada filtro gera uma camada de saída, portanto a profundidade da saída da rede aumenta conforme filtros são adicionados.

Em conjunto com camadas convolucionais, é comum utilizar outros tipos de camada como *max pooling* e *average pooling*. Esses dois tipos de camada também

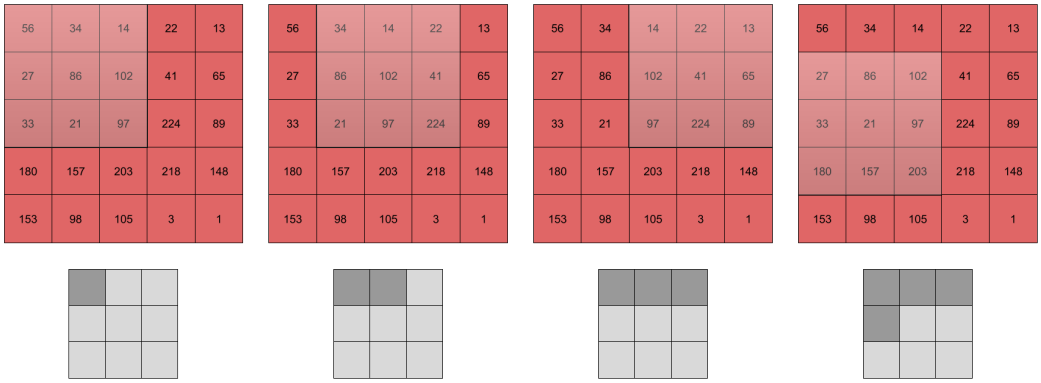


Figura 15 – Demonstração de como um filtro 3x3 é deslocado quatro vezes com passo de convolução 1, transformando a matriz original (em cima) na matriz de saída (embaixo). Fonte: elaborado pela autora.

possuem um filtro e são similares, porém no *max pooling* a rede seleciona o pixel que possui o maior valor entre os pixels que estão dentro do filtro, e no *average pooling* a rede faz a média entre os valores de todos os pixels contidos no filtro. Essas camadas reduzem a dimensão da imagem, o que também reduz o poder computacional exigido.

Após algumas camadas de convolução, *max pooling* e/ou *average pooling*, a camada *global average pooling* pode ser utilizada para extrair as características finais em um vetor linear, onde para cada característica se calcula a média.

4 Rastreador de Pedestres Proposto

Neste capítulo apresentamos em detalhe a abordagem utilizada para rastrear os pedestres. Na Seção 4.1 falamos sobre como as detecções são obtidas e sobre a arquitetura de rede neural utilizada. Já na Seção 4.2 especificamos como foi realizado o treino e a validação.

4.1 Detecções e Detalhes de Implementação

O método utilizado para rastreamento é o TBD, que foi introduzido no Capítulo 1. Esse método pode ser dividido em duas etapas, em que a primeira se baseia em utilizar um algoritmo de detecção para extrair a caixa limitante correspondente a cada pedestre em cada câmera, e a segunda corresponde à associação das detecções ao longo do tempo.

O algoritmo utilizado para realizar a detecção foi proposto por Lima *et al.* (2021), que além de utilizar a biblioteca AlphaPose (LI et al., 2018) para extrair pontos-chave do corpo humano e YOLOv3 (REDMON; FARHADI, 2018) para detectar a caixa limitante de cada pessoa, realiza a fusão dos pedestres detectados em diferentes câmeras para determinar a sua coordenada no espaço 3D considerando um plano de chão (LIMA et al., 2021).

Sendo assim, apesar de os pedestres serem detectados em várias câmeras, a entrada para o algoritmo deste trabalho consiste de apenas uma localização para cada pedestre, não sendo necessário se preocupar com a associação de dados vindos de diferentes câmeras. Além disso, o algoritmo utilizado possui bom desempenho mesmo sendo generalizável, o que possibilita que esse trabalho também evolua para uma generalização.

As detecções representam a localização de cada pedestre em cada instante de tempo t, porém não estabelecem uma relação de identidade entre os pedestres de diferentes tempos. O intuito deste trabalho é relacionar as detecções do mesmo pedestre ao longo de uma sequência de imagens, conseguindo assim traçar a sua trajetória no espaço.

Para isso, utilizamos a rede neural proposta por Brasó & Leal-Taixé (2020) que se baseia na arquitetura MPNN (BRASÓ; LEAL-TAIXÉ, 2020). Ela foi proposta para realizar o rastreamento de pedestres considerando apenas uma câmera, e nesse trabalho foi estendida para utilizar múltiplas câmeras.

Essa rede possui ao todo duas MLP *encoder*, quatro MLP *update* e uma MLP

classifier. Cada MLP possui n camadas lineares, e cada uma é seguida de uma camada de normalização e uma camada de ativação. A função de ativação utilizada é a ReLU.

A MLP do tipo *encoder* serve para processar e comprimir os dados de entrada. A MLP *encoder* de arestas possui duas camadas escondidas com 18 neurônios e uma camada de saída com 16 neurônios. Enquanto isso a MLP *encoder* de vértices possui uma camada escondida com 128 neurônios e uma camada de saída com 32 neurônios.

A MLP *update* é atualizada durante o treinamento para aprender os pesos que convergem para resultados próximos do correto. A MLP *update* de arestas possui uma camada escondida com 80 neurônios e uma camada de saída com 16 neurônios. Existem duas MLP *update* para obter os atributos dos nós do passado e do futuro separadamente, e elas possuem uma camada escondida com 56 neurônios e uma camada de saída com 32 neurônios. Uma MLP *update* atualiza os nós com apenas uma camada de 32 neurônios, para concatenar as informações obtidas dos nós do passado e do futuro e transformar no tamanho dos atributos dos nós.

A MLP *classifier* produz uma saída numérica utilizada para classificação. Ela possui uma camada escondida com 8 neurônios e uma camada de saída com 1 neurônio.

A Figura 16 ilustra a arquitetura utilizada. Tanto nós quanto arestas possuem atributos, sendo que para os nós são utilizadas características visuais da caixa limitante extraídas utilizando a rede ResNet50 (HE et al., 2016) pré-treinada na base de dados ImageNet (DENG et al., 2009).

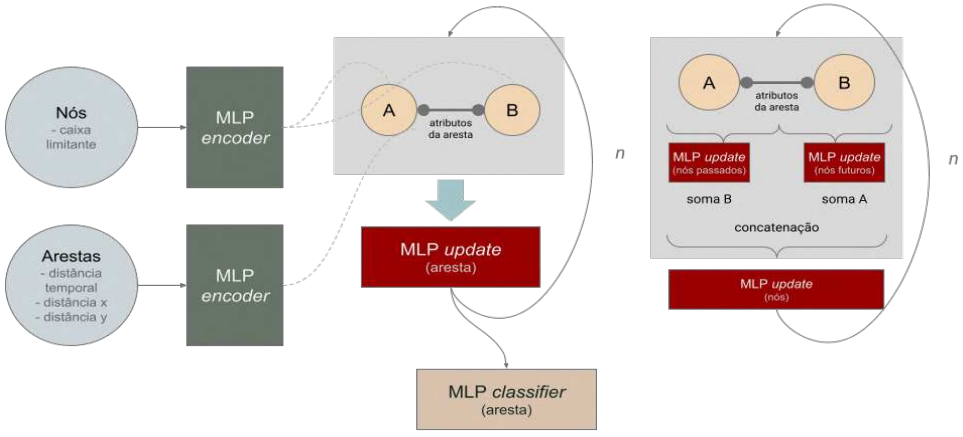


Figura 16 – Arquitetura da rede neural. Fonte: elaborado pela autora.

A rede ResNet50 possui uma camada convolucional, uma camada de *max pooling* e quatro camadas com 3, 4, 6 e 3 blocos respectivamente. Em seguida há uma camada de *global average pooling*. A arquitetura utilizada é descrita com mais detalhes na Tabela 1.

	Tipo	Filtro	Passo	Preenchimento
	Convolucional (64 filtros)	7x7	2	3
	Norm + ReLU			
	Max Pooling	3x3	2	1
3x	Convolucional (64 filtros)	1x1	1	
	Norm + ReLU			
	Convolucional (64 filtros)	3x3	1	1
	Norm + ReLU			
	Convolucional (256 filtros)	1x1	1	
	Norm			
	Convolucional (256 filtros)	1x1	1	
	+ Norm			
	(apenas no 1º bloco)			
	ReLU			
4x	Convolucional (128 filtros)	1x1	1	
	Norm + ReLU			
	Convolucional (128 filtros)	3x3	1	1
			(2 no 1º bloco)	
	Norm + ReLU			
	Convolucional (512 filtros)	1x1	1	
	Norm			
	Convolucional (512 filtros)	1x1	2	
	+ Norm			
	(apenas no 1º bloco)			
	ReLU			
6x	Convolucional (256 filtros)	1x1	1	
	Norm + ReLU			
	Convolucional (256 filtros)	3x3	1	1
			(2 no 1º bloco)	
	Norm + ReLU			
	Convolucional (1024 filtros)	1x1	1	
	Norm			
	Convolucional (1024 filtros)	1x1	2	
	+ Norm			
	(apenas no 1º bloco)			
	ReLU			
3x	Convolucional (512 filtros)	1x1	1	
	Norm + ReLU			
	Convolucional (512 filtros)	3x3	1	1
	Norm + ReLU			
	Convolucional (2048 filtros)	1x1	1	
	Norm			
	Convolucional (2048 filtros)	1x1	1	
	+ Norm			
	(apenas no 1º bloco)			
	ReLU			
	Global Average Pooling			

Tabela 1 – Descrição da arquitetura utilizada para extrair características visuais.

Para as arestas são utilizadas as distâncias entre quadros, entre as coordenadas X e entre as coordenadas Y. Também foram feitos testes utilizando a distância entre as características visuais, que são obtidas de forma semelhante às características dos nós, porém após a camada de *global average pooling* há duas camadas lineares com 1024 e 256 neurônios respectivamente, cada uma seguida de normalização e ReLU. Esses atributos são utilizados como entrada para a MLP *encoder*.

Como os nós precisam utilizar uma caixa limitante e nesse trabalho temos várias, elas foram concatenadas para serem utilizadas como apenas uma entrada para a rede neural. Contudo, a rede neural precisa de uma entrada de tamanho fixo, e o pedestre pode aparecer em diferentes quantidades de câmeras, por isso o tamanho da entrada foi fixado no tamanho máximo considerando o total de câmeras e foi utilizado um vetor de zeros em substituição à imagem nas câmeras que o pedestre não foi detectado. Além disso, cada caixa limitante é redimensionada para 128x64 antes de serem concatenadas.

Em seguida, para cada par de nós A e B que são conectados por uma aresta, os seus atributos e os atributos da aresta são concatenados e utilizados como entrada para a MLP *update* de arestas. Inicialmente esses atributos são a saída da MLP *encoder*, porém essa MLP *update* atualiza os atributos da aresta, assim conseguindo propagar as informações dos nós para as arestas.

De forma semelhante, cada par de nós A e B conectados por uma aresta é utilizado para atualizar os atributos dos nós. Porém, essa atualização considera a relação de passado e futuro entre eles. Primeiro os atributos dos nós do passado são concatenados com os atributos da aresta e utilizados como entrada para a MLP *update* dos nós do passado, e a saída é somada aos atributos dos nós do futuro. Depois, os atributos dos nós do futuro são concatenados com os atributos da aresta e utilizados como entrada para a MLP *update* dos nós do futuro, e sua saída é somada aos atributos dos nós do passado. Dessa forma, a rede é treinada diferenciando as informações do passado e do futuro dos pedestres. Os resultados do passado e do futuro são concatenados e utilizados como entrada para uma última MLP *update*, que atualiza os atributos dos nós.

Essa fase de *update* pode ser repetida várias vezes, e é a parte que representa o processo da MPNN de passar mensagens, pois as informações obtidas pelos nós e pelas arestas são compartilhadas e misturadas.

Posteriormente, os atributos das arestas são utilizados como entrada da MLP *classifier*, e a sua saída é utilizada como parâmetro da função sigmóide para realizar a classificação binária da aresta entre 0 (inativa) ou 1 (ativa), o que vai determinar se aquelas detecções são realmente da mesma pessoa.

4.2 Treino e Validação

Para calcular o quanto predições erradas deveriam ser penalizadas, a função de perda *binary cross entropy* foi utilizada. Para cada passagem l, a penalização da predição $\hat{y}_{(i,j)}$ feita para a aresta (i, j) é calculada, e em seguida as penalizações de todas as arestas são somadas, como é descrito na equação

$$loss^{(l)}(\hat{y}, y) = -\sum_{(i,j)\in E} w \cdot y_{(i,j)} \cdot log(\hat{y}^{(l)}_{(i,j)}) + (1 - y_{(i,j)}) \cdot log(1 - \hat{y}^{(l)}_{(i,j)}), \tag{4.1}$$

em que E é o conjunto de arestas e w é um peso que ajuda a balancear a perda quando o número de rótulos ativos e inativos é muito diferente. Esse peso é calculado ao dividir o número de rótulos inativos pelo número de rótulos ativos, pois, dessa forma, quando existem mais rótulos inativos a penalização de uma predição errada para um rótulo ativo é maior. Além disso, a função logarítmica negativa possui valores maiores quando está próxima de zero, portanto quando o rótulo $y_{(i,j)}$ é 1, a perda utiliza -$log(\hat{y}^{(l)}_{(i,j)})$ porque se $\hat{y}^{(l)}_{(i,j)}$ for zero a penalização será maior. Caso o rótulo seja 0, a perda é calculada com -$log(1 - \hat{y}^{(l)}_{(i,j)})$.

Em seguida, a penalização de todas as passagens l é somada. Como esse valor equivale a perda de todas as arestas, ele ainda é dividido pelo número de arestas para obter a perda média entre uma predição $\hat{y}_{(i,j)}$ e um rótulo $y_{(i,j)}$, como descrito na equação

$$loss(\hat{y}, y) = \frac{1}{|E|} \sum_{l=l_0}^{L} loss^{(l)}. \tag{4.2}$$

O treino e a validação foram realizados de duas formas. Primeiro, para realizar a comparação dos resultados obtidos com diferentes configurações, a rede foi treinada utilizando 70% dos quadros presentes na base de dados como treinamento, 20% como validação e 10% como teste.

Depois foi realizada a avaliação *10-fold-cross-validation* utilizando a configuração que obteve melhor resultado durante o experimento anterior. O *10-fold-cross-validation* consiste em dividir a base de dados em 10 conjuntos de dados de tamanho igual, executar o treino 10 vezes e em cada execução utilizar um conjunto diferente como validação e os outros como treino.

Além disso, o treino foi realizado executando 25 épocas com o otimizador Adam para cada experimento, e a taxa de aprendizado, que é utilizada para regular o quanto os valores de w_i e b_i da equação 3.1 serão corrigidos por época, foi mantida como $l_r = 10^{-3}$.

Os experimentos foram feitos utilizando uma máquina que possui o processador Intel Xeon @ 2.20GHz, 26GB de RAM e placa de aceleração gráfica NVIDIA Tesla P100 com 16GB de memória.

5 Resultados e Discussão

Após o desenvolvimento de uma tecnologia, é necessário realizar experimentos e comparar o desempenho dela com as soluções existentes. Neste capítulo são apresentadas na Seção 5.1 a base de dados e as métricas utilizadas, depois relatamos os experimentos com o detector de pedestres na Seção 5.2 e os experimentos com as anotações verdadeiras na Seção 5.3.

5.1 Base de Dados e Métricas

A base de dados utilizada nos experimentos foi a WILDTRACK, que possui 7 câmeras com vistas sobrepostas em uma área aberta com um grande fluxo de pessoas. Ela possui anotações verdadeiras que indicam para cada pedestre o seu número de identificação, o par de coordenadas (x, y) que representa a posição 3D do pedestre, e a sua caixa limitante $(x_{left}, y_{top}, x_{width}, y_{height})$ em cada câmera ao longo de 400 quadros.

A principal métrica utilizada é o MOTA, que resume a relação entre os erros e o total de detecções através da equação:

$$MOTA = 1 - \frac{FP + FN + MM}{OBJ} \tag{5.1}$$

onde FP é a quantidade de falsos positivos, FN é a quantidade de falsos negativos, MM é a quantidade de trocas de id (*mismatches*) e OBJ é a quantidade de objetos que existem nas anotações verdadeiras. De forma simplificada, os três tipos de erros são somados e divididos pela quantidade de pedestres, resultando na taxa de erro por anotação.

A segunda métrica utilizada é o MOTP, que calcula a precisão do rastreamento:

$$MOTP = 1 - \frac{d_{err}}{n_{matches}} \tag{5.2}$$

onde d_{err} é a soma das diferenças métricas entre a posição rastreada e a localização real, e $n_{matches}$ é o número de casos em que ocorreu uma correspondência entre o rastreamento e a anotação verdadeira.

5.2 Experimentos com Detector de Pedestres

Utilizar as detecções obtidas através de um algoritmo de detecção permite que a solução proposta nesse trabalho seja avaliada da forma que seria utilizada na prática.

Sendo assim, realizamos experimentos treinando a rede neural com as anotações verdadeiras e também treinando com o resultado do rastreamento proposto por Lyra *et al.* (2022), não precisando das anotações verdadeiras (LYRA et al., 2022). Enquanto isso, no conjunto de teste utilizamos as detecções de Lima *et al.* (2021) nos dois casos (LIMA et al., 2021).

As Tabelas 2 e 3 mostram os resultados obtidos treinando com as anotações verdadeiras e com o rastreamento de Lyra *et al.* (2022) respectivamente. A coluna "Quadros" se refere à quantidade de quadros que é considerada na montagem do grafo, sendo que quando são 15 quadros o algoritmo só funcionaria de forma *offline* visto que utiliza tanto quadros anteriores como quadros futuros, e a intenção de utilizar 2 quadros é verificar o desempenho do algoritmo caso ele seja *online*. Também avaliamos o desempenho do algoritmo adicionando aos atributos das arestas a distância entre os vetores de Re-ID.

É possível observar que em ambos os casos os melhores resultados foram obtidos ao utilizar 15 quadros e não utilizar a distância visual entre as detecções, sendo que treinando com as anotações verdadeiras a solução proposta atingiu 76,6% de MOTA e treinando com o rastreamento de Lyra *et al.* (2022) conseguimos 77,1% de MOTA.

Além disso, o treinamento tem duração aproximada de 3 a 6 minutos ao todo, enquanto a inferência processa cerca de 40 quadros por segundo. Porém, para utilizar as características visuais é necessário processar as detecções antes para obter os vetores de Re-ID, e isso torna o algoritmo mais lento, chegando a 0,6 quadros por segundo. Portanto, também realizamos testes utilizando as coordenadas x e y como atributos dos nós, e conseguimos obter os mesmos resultados com mais eficiência.

Quadros	Distância Métrica	Distância Visual	MOTA	MOTP
2	Sim	Não	75,2%	90,7%
2	Sim	Sim	73,7%	80,5%
15	Sim	Não	**76,6%**	**90,9%**
15	Sim	Sim	75,3%	86,6%

Tabela 2 – Resultados obtidos no conjunto de testes após treinamento com anotações verdadeiras. Os melhores resultados estão em negrito.

Quadros	Distância Métrica	Distância Visual	MOTA	MOTP
2	Sim	Não	75,8%	90,6%
2	Sim	Sim	75,8%	90,5%
15	Sim	Não	**77,1%**	**90,7%**
15	Sim	Sim	76,6%	89,5%

Tabela 3 – Resultados obtidos no conjunto de testes após treinamento com rastreamento de Lyra *et al.* (2022). Os melhores resultados estão em negrito.

Considerando a melhor configuração, realizamos um experimento seguindo o padrão *10-fold-cross-validation*, em que foi realizada uma média dos resultados obtidos nos 10 treinos por iteração, e o melhor MOTA foi alcançado na época 24 com 62,3%. A evolução do MOTA pode ser observada na Figura 17.

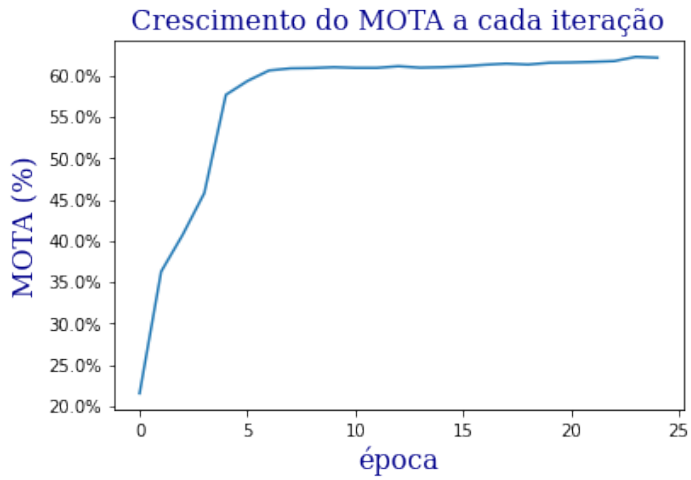


Figura 17 – MOTA obtido a cada iteração durante o *10-fold-cross-validation*. Fonte: elaborado pela autora.

Observando três quadros consecutivos da inferência obtida após treinar com a melhor configuração anterior, analisamos o resultado de forma qualitativa destacando quatro pedestres. A Figura 18 mostra as 7 câmeras e o plano de chão correspondentes aos quadros 363, 364 e 365, com os pedestres 009, 014, 019 e 024 coloridos de amarelo, verde claro, verde escuro e azul claro respectivamente. Nessas imagens, é possível ver que além dos pedestres serem reidentificados em diferentes câmeras, também continuam a ser rastreados mesmo quando possuem oclusão severa, por exemplo, na câmera 5 (imagem do meio), inicialmente o pedestre 024 é visto no quadro 363, não é visto no quadro 364, e volta a aparecer no quadro 365, mas tem um rastreamento estável pois continua a ser visto em outras câmeras.

O pedestre 009 está ao lado dos pedestres 006 e 011 e é possível olhar com clareza na câmera 3 que o pedestre 006 é o que segura uma bolsa. Na câmera 5, a pessoa com bolsa está ocludindo o pedestre 009, mas notamos que tanto o pedestre 009 como o 011 continuam a ser rastreados de forma estável também.

Os pedestres 014 e 019 são mais exemplos de pedestres que continuam sendo rastreados mesmo sofrendo oclusões parciais ou severas em algumas câmeras pois estão sendo vistos de outras câmeras. No plano de chão a trajetória dos pedestres pode ser vista de maneira mais precisa, o pedestre 009 está indo do meio em direção à direita, o 019 também, o 014 está no meio e o 024 está do lado oposto ao 009, mostrando que o plano de chão é coerente em relação às câmeras.

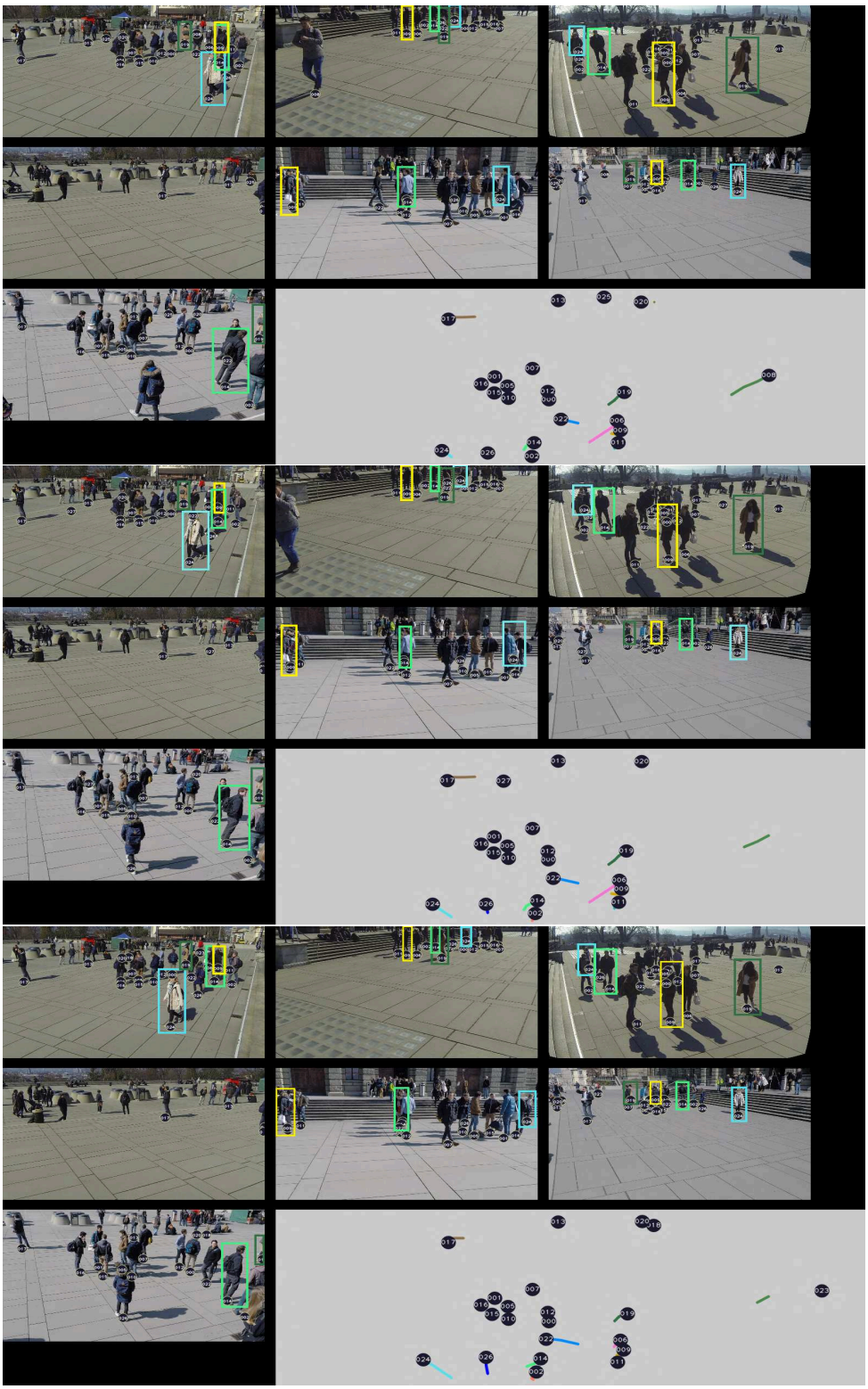


Figura 18 – Imagens das 7 câmeras e do plano de chão dos quadros 363, 364 e 365 destacando os pedestres 009, 014, 019 e 024 de amarelo, verde claro, verde escuro e azul claro respectivamente. As câmeras são consideradas numeradas da esquerda para a direita e de cima para baixo. Fonte: elaborado pela autora.

Além disso, apesar do resultado geral ser bom, ainda existem erros no rastreamento. Por exemplo, apenas no quadro 364 aparece a detecção 027, e ao ser projetada nas câmeras 3, 6 e 7 constatamos que aponta para um espaço vazio, sendo um falso positivo.

5.3 Experimentos com Anotações Verdadeiras

Como a qualidade do rastreamento TBD depende da qualidade das detecções, experimentamos testar a rede neural com as anotações verdadeiras também. A Tabela 4 exibe os resultados, em que o maior MOTA é 99,9% com a mesma configuração que os experimentos anteriores.

Esse resultado demonstra o quanto a qualidade das detecções reflete no rastreamento, chegando a ser 22,9% melhor do que o melhor resultado dos experimentos anteriores e sendo quase perfeito.

Quadros	Distância Métrica	Distância Visual	MOTA	MOTP
2	Sim	Não	98,6%	99,9%
2	Sim	Sim	98,9%	99,9%
15	Sim	Não	**99,9%**	**100%**
15	Sim	Sim	99,7%	100%

Tabela 4 – Resultados obtidos no conjunto de testes de anotações verdadeiras. Os melhores resultados estão em negrito.

5.4 Comparação com Trabalhos Relacionados

Considerando 10% da base de dados WILDTRACK para teste, apresentamos na Tabela 5 a comparação com outros algoritmos de rastreamento.

Técnica	MOTA	MOTP
Chavdarova *et al.* (2018)	72,2%	60,3%
You & Jiang (2020)	74,6%	78,9%
Vo *et al.* (2020)	75,8%	-
Lyra *et al.* (2022)	**77,1%**	**96,4%**
Nossa solução	**77,1%**	90,7%

Tabela 5 – Comparação da nossa solução com os trabalhos relacionados.

Os melhores resultados obtidos são similares aos resultados de Lyra *et al.* (2022), que podem ser visualizados com mais detalhes na Tabela 6 (LYRA et al., 2022). A solução de Lyra *et al.* (2022) utiliza um algoritmo determinístico que possui o resultado fixo de 77,1% de MOTA, enquanto a nossa solução obteve resultados que oscilam entre

76,9% e 77,1%. Porém, a nossa solução consegue rastrear a uma taxa de 40 quadros por segundo, enquanto a de Lyra *et al.* (2022) rastreia 20 quadros por segundo. O trabalho de You & Jiang (2020) rastreia 15 quadros por segundo e as outras soluções são *offline*.

Técnica	Detecções	MOTA	MOTP
Lyra *et al.* (2022)	Detector	**77,1%**	**96,4%**
Nossa solução	Detector	**77,1%**	90,7%
Lyra *et al.* (2022)	Anotações Verdadeiras	98,9%	98,7%
Nossa solução	Anotações Verdadeiras	**99,9%**	**100%**

Tabela 6 – Comparação da nossa solução com Lyra *et al.* (2022).

Como a base de dados WILDTRACK também possui anotações sobre as caixas limitantes de cada câmera, realizamos um experimento utilizando o algoritmo original de Brasó & Leal-Taixé (2020) para rastrear apenas as detecções 2D da primeira câmera do WILDTRACK, e o MOTA foi de 45,5%, demonstrando que o uso de múltiplas câmeras proposto neste trabalho produziu uma melhora significativa nesse tipo de ambiente.

6 Conclusões e Trabalhos Futuros

Neste capítulo fazemos um resumo dos resultados obtidos nesse trabalho na Seção 6.1 e falamos brevemente sobre possíveis tópicos para serem estudados após esse trabalho na Seção 6.3.

6.1 Conclusões

Neste trabalho propusemos uma nova abordagem para o rastreamento 3D de pedestres em ambientes multi-câmera. A abordagem utiliza a arquitetura de rede neural MPNN para associar as detecções que pertencem a um mesmo pedestre e traçar a sua trajetória espaço-temporal. Realizando experimentos na base de dados WILDTRACK, mostramos que a técnica atinge até 77,1% de MOTA ao ser treinada com o resultado do rastreamento de Lyra *et al.* (2022), e 62,3% de MOTA no *10-fold-cross-validation*. Além disso, o tempo necessário para rastrear os pedestres é de 40 quadros por segundo, sendo competitivo em relação a outras soluções.

6.2 Contribuições

O algoritmo proposto para rastrear pedestres pode ser utilizado em diversas aplicações úteis para a sociedade, como a análise de comportamento de indivíduos e a vigilância por vídeo citados no Capítulo 1. Durante o desenvolvimento, também contribuímos para a publicação Generalizable Online 3D Pedestrian Tracking with Multiple Cameras (LYRA et al., 2022) na Conferência Internacional sobre Teoria e Aplicações de Visão Computacional (VISAPP).

6.3 Trabalhos Futuros

Os resultados obtidos considerando apenas 2 quadros são inferiores aos obtidos com 15 quadros porque ocorrem mais trocas de identidade, portanto seria interessante estudar como diminuir essas trocas para que o desempenho *online* seja tão bom quanto *offline*.

Para mais, esse trabalho avaliou o uso de uma possível abordagem para treinar a rede neural sem a necessidade de anotações verdadeiras, porém existem várias técnicas de treinamento não-supervisionado que poderiam ser experimentadas neste cenário.

Referências

BADUE, C. et al. Self-driving cars: A survey. *Expert Systems with Applications*, Elsevier, v. 165, p. 113816, 2021. Citado na página 12.

BERCLAZ, J. et al. Multiple object tracking using k-shortest paths optimization. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 33, n. 9, p. 1806–1819, 2011. Citado 2 vezes nas páginas 7 e 16.

BERNARDIN, K.; STIEFELHAGEN, R. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, Springer, v. 2008, p. 1–10, 2008. Citado 2 vezes nas páginas 15 e 19.

BRASÓ, G.; LEAL-TAIXÉ, L. Learning a neural solver for multiple object tracking. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2020. p. 6247–6257. Citado 5 vezes nas páginas 7, 13, 17, 19 e 28.

CHAVDAROVA, T. et al. Wildtrack: A multi-camera hd dataset for dense unscripted pedestrian detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2018. p. 5030–5039. Citado 4 vezes nas páginas 7, 13, 14 e 16.

CHAVDAROVA, T.; FLEURET, F. Deep multi-camera people detection. In: IEEE. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. [S.l.], 2017. p. 848–853. Citado na página 13.

CHEN, K. et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. Citado na página 13.

DENG, J. et al. Imagenet: A large-scale hierarchical image database. In: IEEE. *2009 IEEE conference on computer vision and pattern recognition*. [S.l.], 2009. p. 248–255. Citado na página 29.

FERRYMAN, J.; SHAHROKNI, A. Pets2009: Dataset and challenge. In: IEEE. *2009 Twelfth IEEE international workshop on performance evaluation of tracking and surveillance*. [S.l.], 2009. p. 1–6. Citado na página 12.

GONG, S. et al. The re-identification challenge. In: *Person Re-Identification*. [S.l.]: Springer, 2014. p. 1–20. Citado na página 20.

HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778. Citado na página 29.

HENSCHEL, R.; ZOU, Y.; ROSENHAHN, B. Multiple people tracking using body and joint detections. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. [S.l.: s.n.], 2019. p. 0–0. Citado na página 13.

LEAL-TAIXÉ, L. et al. Tracking the trackers: an analysis of the state of the art in multiple object tracking. *arXiv preprint arXiv:1704.02781*, 2017. Citado 2 vezes nas páginas 15 e 19.

LI, J. et al. Crowdpose: Efficient crowded scenes pose estimation and a new benchmark. *arXiv preprint arXiv:1812.00324*, 2018. Citado na página 28.

LIMA, J. P. et al. Generalizable multi-camera 3d pedestrian detection. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2021. p. 1232–1240. Citado 5 vezes nas páginas 7, 15, 18, 28 e 34.

LIN, T.-Y. et al. Microsoft coco: Common objects in context. In: SPRINGER. *European conference on computer vision*. [S.l.], 2014. p. 740–755. Citado na página 13.

LIU, S. et al. Path aggregation network for instance segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2018. p. 8759–8768. Citado na página 13.

LUO, W. et al. Multiple object tracking: A literature review. *Artificial Intelligence*, Elsevier, p. 103448, 2020. Citado 2 vezes nas páginas 16 e 19.

LYRA, V. et al. Generalizable online 3d pedestrian tracking with multiple cameras. 2022. Citado 6 vezes nas páginas 7, 18, 19, 34, 37 e 39.

MAKSAI, A. et al. Non-markovian globally consistent multi-object tracking. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2017. p. 2544–2554. Citado na página 16.

MILAN, A. et al. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016. Citado na página 13.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. *arXiv*, 2018. Citado na página 28.

SARCINELLI, R. et al. Handling pedestrians in self-driving cars using image tracking and alternative path generation with frenét frames. *Computers & Graphics*, Elsevier, v. 84, p. 173–184, 2019. Citado na página 12.

SUN, Z. et al. A survey of multiple pedestrian tracking based on tracking-by-detection framework. *IEEE Transactions on Circuits and Systems for Video Technology*, IEEE, v. 31, n. 5, p. 1819–1833, 2020. Citado 4 vezes nas páginas 7, 12, 13 e 14.

VO, M. P. et al. Self-supervised multi-view person association and its applications. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, 2020. Citado 2 vezes nas páginas 14 e 17.

YI, S.; LI, H.; WANG, X. Understanding pedestrian behaviors from stationary crowd groups. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 3488–3496. Citado na página 12.

YOU, Q.; JIANG, H. Real-time 3d deep multi-camera tracking. *arXiv preprint arXiv:2003.11753*, 2020. Citado na página 17.

ZHANG, X.; YU, Q.; YU, H. Physics inspired methods for crowd video surveillance and analysis: a survey. *IEEE Access*, IEEE, v. 6, p. 66816–66830, 2018. Citado na página 12.