LEARNING FOR MULTI-VIEW TRACKING

Faculté Informatique et Communications
École Polytechnique Fédérale de Lausanne

Thèse de Master de

Ivan Vrkic

Supervisée par

Prof. Pascal Fua, CVLAB, EPFL
Dr. Martin Engilberge, CVLAB, EPFL

Lausanne, EPFL, 2023

To my family, for their unconditional love and support

Acknowledgements

First and foremost, I owe a debt of gratitude to *Martin*. His guidance has been invaluable, and his support has become the backbone of this work. He constantly reminded me to start small and keep iterating.
I extend my thanks to Prof. *Fua* and the entire CVLAB community – *Vidit*, *Hieu*, *Nicolas*, *Yann*, *Benoit*, *Soumava*, *Olivier* and others. Our lunches, outings, and your inclusiveness and support created an environment where I felt welcomed and at home.
To my office mates *Aoxiang*, *Corentin*, *Deniz*, *Malo*, and *Zhi Chen* – our technical and non-technical banter was both informative and entertaining. You, among others, made every day at the office a learning experience.
The team at Invision AI has been nothing short of incredible. *Julien*, thank you for introducing me to the nuances of Google coding practices. A big shoutout to *Carlos* and *Cloé*; you were always supportive, and I'll never forget the croissant breaks that brought moments of joy.
A special mention goes to *Udaranga* for opening the doors to CVLAB, enabling me to work on the semester research project, and offering great career advice.
To my kitchen mates, *Eva* and *Martin* – the dinners we had together were always special. Similarly, *Luca* and *Lorenzo*, our math-heavy projects will remain memorable.
I'm thankful to *Klara* for being a great study buddy, and my Viennese friends – *Michael K.* and *Tahel*. I was always looking forward to lunches and discussions at the BC terrace.
Bettina, *Benjamin*, *Lukas*, *Max*, *Tin*, *Tom* and *Valentin* – your presence always added zest to my stay in Vienna. *Michael R.*, your startup ideas never fail to inspire.
And to everyone else I've had the pleasure of meeting, who took the time to discuss my ideas and share their expertise – your input has been invaluable.
Most importantly, my deepest gratitude goes out to my family and friends in Zagreb. Your love and belief in me have been the key of all my endeavors.
I have grown a lot as person during my stay in Lausanne, and I owe a significant part of this growth to every individual mentioned above and many more who have touched my life. Thank you.

Lausanne, July 12, 2023 I. V.

Abstract

In response to the growing trend towards end-to-end learning, we propose a novel framework advancing towards an end-to-end multi-camera multi-object tracking (MC-MOT) solution that addresses challenges like occlusions, viewpoint variations, and illumination changes. Although current strategies treat detection, feature extraction, association, and trajectory generation as separate stages, our approach fuses these stages, marking a stride towards an end-to-end approach.

Our method introduces a mechanism for neural association that effectively links object identities across frames and cameras, adeptly handling object appearance and disappearance. This mechanism keeps track of object identities across camera views, reducing identity switches and track losses found in prior techniques. Our method is capable of performing online, meaning it can process and update tracking information as new data is received, making it suitable for applications requiring immediate action.

Through experimentation, we demonstrate that our system achieves performance levels similar to those of existing state-of-the-art (SOTA) multi-camera multi-object tracking (MC-MOT) techniques, both in accuracy and robustness, while being able to perform online. We also introduce a benchmark dataset with a tool used to create it to assess our MC-MOT framework's capabilities. For other multi-camera systems, we suggest using object appearance through a Multi-View Retrieval setup.

While fully realizing end-to-end learning remains an open challenge in MC-MOT, the proposed method's integrated nature marks a promising step towards such a paradigm in future MC-MOT research.

Keywords: end-to-end learning, multi-camera multi-object tracking (MC-MOT), neural association, graph neural networks, online tracking, benchmark dataset, multi-view retrieval

Résumé

En réponse à la tendance croissante vers l'apprentissage de bout en bout, nous proposons un nouveau cadre d'apprentissage qui progresse vers une solution de suivi multi-objets et multi-caméras de bout en bout (MC-MOT) qui relève des défis tels que les occultation, les variations de point de vue et les changements d'illumination. Contrainement aux stratégies actuelles qui traitent la détection, l'extraction de caractéristiques, l'association et la génération de trajectoires comme des étapes disparates, notre approche fait converger ces étapes, marquant une avancée significative vers l'apprentissage de bout en bout.

Notre méthode introduit un mécanisme d'association neuronale qui associe efficacement les identités des objets à travers les images et les caméras, en gérant adéquatement l'apparition et la disparition des objets. Notre méthode est capable de fonctionner en ligne, ce qui signifie qu'elle peut traiter et mettre à jour les informations de suivi à mesure que de nouvelles données sont reçues. Ce mécanisme gère les identités des objets, réduisant ainsi les changements d'identité et les pertes de pistes constatées dans les techniques antérieures.

Grâce à l'expérimentation, nous démontrons que notre système atteint des niveaux de performance similaires à ceux des techniques de suivi multicaméras multi-objets (MC-MOT) de pointe existantes (SOTA), tant en termes de précision que de robustesse, tout en étant capable de fonctionner en ligne. Nous présentons également un ensemble de données de référence avec un outil de création pour évaluer les capacités de notre cadre MC-MOT. Pour d'autres systèmes multi-caméras, nous suggérons d'utiliser l'apparence des objets par le biais d'une configuration de récupération multi-vues.

Bien que la réalisation complète de l'apprentissage de bout en bout reste un défi ouvert dans le MC-MOT, la nature intégrée de la méthode proposée marque une étape prometteuse vers un tel paradigme dans la recherche future sur le MC-MOT.

Contents

1 Introduction

Recently, a shift from traditional, algorithm-based programming to neural software has been taking place. Instead of manually coding software using explicit rules in traditional languages, the future involves accumulating training datasets and crafting objectives to train neural networks. In this context, the learned parameters of the neural network effectively become the software itself, ready for deployment.

For example, image classification has evolved from using hard-coded algorithms to employing convolutional neural networks that learn to recognize patterns. This approach replaces manual feature engineering, resulting in an optimization process that outperforms human capabilities, and signifies a transition towards end-to-end learning (Krizhevsky et al., 2012). Currently, the focus is on designing the overall system and objectives, while machine learning algorithms determine specific computations. Consequently, a significant proportion of *code* is now represented by the weights in neural networks.

This shift has broad implications, as seen in fields like multi-object tracking, which increasingly uses neural solvers, aiming for for end-to-end learning (Brasó and Leal-Taixé, 2019). Another important area where this shift can make an impact is in multi-view multi-object tracking. Multi-view multi-object tracking, as the name suggests, entails tracking multiple objects across numerous camera views. This task is challenging due to the diverse perspectives, varying lighting conditions, object occlusions, and unpredictable object motions in blind spots that often characterizes the environment of multiple cameras. These challenges often lead to difficulties in distinguishing and re-identifying individuals across different views, contributing to inaccuracies in tracking.

In thesis we aim to explore and address the challenges of multi-view multi-object tracking, with an emphasis on creating an efficient neural solution for tracking that uses object appearance.

1.1 Scope of the Work

This section outlines the progression of work undertaken during this thesis. The initial aim of this work was to integrate appearance features into the tracking framework developed by Ali et al. (2023) at Invision AI[1]. The tracking framework was based solely on geometric principles for associating tracklets across multiple camera views with a probabilistic solver. The efforts to improve the tracking led to the development of the neural association algorithm. The detailed sequence of work conducted is outlined as follows:

Incorporation of Appearance Features. The initial objective of this work was to improve the tracking cluster by incorporating appearance features. The idea was to enrich the geometric basis of the tracking framework by Ali et al. (2023), offering a potential for more robust tracking associations.

Dataset Creation for Appearance Feature Evaluation. In alignment with the objective of incorporating appearance features into the tracking cluster, we created a dedicated dataset. This dataset was developed to enable testing and evaluation of the improvements made through the integration of appearance features.

Retrieval (ReID) Model Training and Integration. To achieve the incorporation of appearance features into the tracking framework, we use a ReID model. This trained ReID model was adapted to operate within the multi-view context of the tracking framework, thereby integrating appearance information with geometric tracking. We show the results in the proposed Multi-View Retrieval setup.

Multi-View Framework with Neural Association. Following the successful integration of appearance features through the ReID model, the focus shifted to further improving the tracking results. To achieve this, we developed an online neural tracking framework that works in a multi-view setting. This approach builds upon the improvements made through appearance integration and single-view tracking approaches (Brasó and Leal-Taixé, 2019) to further optimize the tracking in a multi-view setting.

1.2 Research Contribution

We propose a novel framework that ties together different steps of MC-MOT, moving towards an all-in-one learning solution. We present a mechanism for neural association in a multiview setting, to keep track of object identities better, reducing mistakes and lost tracks. We show through experiments that our system yields competitive performance when compared to existing MC-MOT methods, while being able to perform online. Alongside this, we propose a new benchmark dataset and an accompanying tool for its creation[2] to test the capabilities of the proposed MC-MOT tracking framework. For other multi-camera systems that utilize standard approaches, we propose using object appearance with a Multi-View Retrieval setup. We provide a promising direction for future research to develop a fully all-in-one MC-MOT solution.

[1] https://invision.ai/

[2] https://github.com/ivanvrkic/multicam-gt

1.3 Thesis Structure

Following this introduction, Chapter 2 gives some background of the field required for understanding the concepts used throughout the the thesis, readers familiar with the field may skip this chapter. We provide an overview of the benchmark dataset in Chapter 3. Chapter 4 explains our method for learning appearance in a multi-view setting. Chapter 5 goes into detail on how we designed and built the neural graph association algorithm for our multi-view tracking tool, and how we tested its performance. Chapter 6 wraps up the work with a summary of what we found and ideas for future research in multi-camera multi-object tracking.

2 Background

The following sections offer an introduction to basic machine learning concepts that we use throughout the thesis such as supervised learning, and delve into more advanced topics like neural networks that operate on graphs for understanding the neural association algorithm. Given the vastness of the field, this overview is by no means exhaustive, and it omits other topics to maintain thesis relevance. For deeper understanding, further exploration in machine learning literature and resources is recommended, such as Goodfellow et al. (2016); Bishop (2006).

2.1 Introduction to Machine Learning

Machine learning focuses on discovering an appropriate model f within a given set of models $\mathscr{F}$, for solving some task.

A set of parametric models that learn the target mapping consists of functions represented as $\mathscr{F} = \{f_{\theta \in \Theta}\}$, where the function $f_\theta(\mathbf{x})$ is $f_{\mathscr{F}}(\theta, \mathbf{x})$ for a certain function that maps the parameter space Θ and domain $\mathbb{D}$ to output space $\mathbb{O}$, with $\theta \in \Theta$ and $\mathbf{x} \in \mathbb{D}$. The function $f_{\mathscr{F}}$ incorporates the prior knowledge used to create our model set, while θ identifies a specific member of this set and will be learned. The layout of $f_{\mathscr{F}}$ and consequently that of $\mathscr{F}$ determines the nature of solutions that can be obtained, restricting the potential behaviors that the model can learn.

Learning signifies the procedure of trying to identify the most appropriate member f^* from a set of models $\mathscr{F}$ to solve a particular task. For a parametric set of models, this equates to finding the optimal parameter value $\theta^* \in \Theta$. As the solution is often not closed-form, this generally involves an iterative optimization process aimed at reducing some measure of discrepancy, also referred to as a loss function, between the output of the model and the desired output. If we consider $\mathscr{X}$ as a distribution over $\mathbb{D}$ and $\mathscr{L}$ as a suitable loss function, we can express this as:

$$f^* := \arg\min_{f \in \mathscr{F}} \mathbf{E}_{\mathbf{x} \sim \mathscr{X}}(\mathscr{L}(\mathbf{x}, f)) \tag{2.1}$$

Here, $\mathbf{E}_{\mathbf{x}\sim\mathscr{X}}(\mathscr{L}(\mathbf{x}, f))$ represents the expected loss that we see if we sample inputs $\mathbf{x}$ from the distribution $\mathscr{X}$ and evaluate the model's predictions for those inputs. However, in real-world scenarios, we have of set observations $\mathscr{D} = \{\mathbf{x}_i\}_{i=1}^N$ where N is the set size and each $\mathbf{x}_i \in \mathscr{D}$ is drawn from $\mathscr{X}$, commonly referred to as the training dataset. We therefore do not have access to the empirical distribution $\mathscr{X}$ Then, the *empirical* error, which is an estimate of the expected loss, can be written as:

$$f_E^* := \arg\min_{f\in\mathscr{F}} \frac{1}{N}\sum_{i=1}^{N} \mathscr{L}(\mathbf{x}_i, f) \tag{2.2}$$

2.1.1 Supervised Learning

The training approach can vary based on the specific task. Commonly, these strategies can be classified into supervised learning and unsupervised learning.

In case of supervised learning, samples $\mathbf{x} \in \mathbb{D}$ are expressed as

$$\mathbf{x} = (\mathbf{u}, \mathbf{y}) \in (\mathbb{U}, \mathbb{O}) \tag{2.3}$$

In this context, $\mathbb{U}$ and $\mathbb{O}$ denote possible values for $\mathbf{u}$ and $\mathbf{y}$, respectively, with $\mathbf{u}$ representing the input data and $\mathbf{y}$ representing the target data. Here we usually assume $\mathbb{U}$ to be a high dimensional Euclidean space: $\mathbb{U} = \mathbb{R}^d$. In these cases, the objective is to determine the genuine conditional $\pi(\mathbf{y} \mid \mathbf{u})$ based on the data set $\mathscr{D} = \{(\mathbf{u}_i, \mathbf{y}_i)\}_{i=1}^N$ where each pair $(\mathbf{u}_i, \mathbf{y}_i)$ is a random, independent, and identically distributed sample drawn from the joint distribution $\mathscr{X}(\mathbf{u}, \mathbf{y})$. From a probabilistic perspective, we may think of the model f_θ as conditional probability density function $p_\theta(\mathbf{y} \mid \mathbf{u})$.

Classification

In classification, the objective is to label each input sample $\mathbf{u}$. A standard example is the MNIST dataset, in which the input samples are images displaying a single digit. The task required by this data set is to identify the digit in the current image and output the corresponding tag y. Hence, the labels are $\mathbb{O} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

Binary classification refers to a task where we want to predict one of two possible outcomes. For example, predicting whether an email is spam or not spam. $\mathbb{O}$ contains exactly two elements. We can interpret the output as a probability of success for a binary outcome (i.e., Bernoulli distribution). The binary cross-entropy, a metric for distribution similarity, between conditional Bernoulli distribution and the true conditional distribution $\mathscr{X}(y \mid \mathbf{u})$ is utilized as our loss:

$$\begin{aligned} \mathscr{L}_{\text{BCE}}(y, \mathbf{u}, f_\theta) &= -y \log p_\theta(y \mid \mathbf{u}) - (1-y) \log p_\theta(1-y \mid \mathbf{u}) \\ &= -y \log f_\theta(\mathbf{u}) - (1-y) \log\big(1 - f_\theta(\mathbf{u})\big) \end{aligned} \tag{2.4}$$

Multi-class classification, on the other hand, applies when there are more than two potential mutually exclusive labels. A standard strategy here is to use one-hot encoding for the output. Meaning, the output from the model is a vector of dimension k, where k denotes the total number of potential classes, i.e. $\hat{\mathbf{y}} \in \mathbb{R}^k$. Let $\mathbb{O} = \{d_1, d_2, \ldots, d_k\}$ be the set of all dimensions, each representing a class. If p_{d_i} denotes the probability of a given input belonging to class d_i, then the vector representation of the output $\hat{\mathbf{y}}$ is:

$$\hat{\mathbf{y}} = [p_{d_1}, p_{d_2}, \ldots, p_{d_k}] \tag{2.5}$$

The label vector $\mathbf{y} \in \mathbb{R}^k$ is a one-hot encoded vector representing the actual class of the input. If the input belongs to class d_i, then:

$$\mathbf{y} = [0, 0, \ldots, 1, \ldots, 0] \tag{2.6}$$

Here, 1 is at the i-th position. Here the negative log likelihood serves as a loss metric, and it simplifies to the binary case's cross-entropy when there are only two classes. The negative log likelihood's equation is:

$$\mathscr{L}_{\text{NLL}}\left(\mathbf{y}, \mathbf{u}, f_\theta\right) = \sum_i -\mathbf{y}_i \log p_\theta\left(\mathbf{y}_i \mid \mathbf{u}\right) = \sum_i -\mathbf{y}_i \log f_\theta(\mathbf{u})_i \tag{2.7}$$

Here, $p_\theta\left(\mathbf{y}_i \mid \mathbf{u}\right)$ denotes the probability of example $\mathbf{u}$ being in class i.

Multi-label classification involves making several independent predictions for the same data point. This approach is seen in scenarios such as predicting a movie category, where several possible categories need to be predicted for the same movie. Here, each decision is treated as a classification task, and our objective or loss is the mean of each random variable.

Regression

The output space $\mathbb{O}$ is continuous in regression tasks. We usually take a simplified covariance matrix $\Sigma = \sigma^2 I$, where σ is kept constant and the mean is denoted by f_θ. Hence,

$$p_\theta(\mathbf{y} \mid \mathbf{u}) = \mathscr{N}\left(\mathbf{y}; f_\theta(\mathbf{u}), \Sigma\right) \tag{2.8}$$

The commonly used loss in this case is the mean square error (MSE) criterion:

$$\mathscr{L}_{\text{MSE}}\left(\mathbf{y}, \mathbf{u}, f_\theta\right) = \left(\mathbf{y} - f_\theta(\mathbf{u})\right)^2 \tag{2.9}$$

2.1.2 Unsupervised Learning

In an unsupervised setting, no labels $\mathbf{y}$ are associated with inputs $\mathbf{u}$. The aim is to uncover hidden patterns within the dataset. Several algorithms fall into this category, with some examples like clustering and dimensionality reduction.

Clustering, seeks to segment the data into clusters based on the similarity among samples. One example is the k-Means algorithm (Lloyd, 1982).

Techniques for dimensionality reduction, like principal component analysis (PCA) or UMAP (McInnes et al., 2020), seek to simplify input data by reducing its dimensionality while preserving the most important features.

2.2 Neural Networks

Neural Networks (NNs) have layers of neurons: the input layer, hidden layers, and the output layer. Data is passed to the input layer $\mathbf{u}$, processed in hidden layers through intermediate calculations, and the results are produced by the last layer $\hat{\mathbf{y}} \in \mathbb{R}^k$.

The model parameters - the weights and biases, are learned through an iterative optimization process. The goal of the model is to produce an output $\hat{\mathbf{y}}$ similar to or equal to the target $\mathbf{y}$ given an input $\mathbf{u}$. This could mean aligning the output $\hat{\mathbf{y}}$ with a characteristic of a probability distribution $p(\mathbf{y}|\mathbf{u})$. The method to measure the difference during this optimization is model-dependent and task-dependent. In this model, the input layer is represented as $\mathbf{u}$, i.e. $\mathbf{h}^{(0)} = \mathbf{u}$.

We have a series of hidden layers denoted as $\mathbf{h}^{(l)}, 1 \leq l \leq L$, where L is the count of hidden layers, each performing a nonlinear transformation of the input, which can be represented as:

$$\mathbf{h}^{(l)} = \sigma^{(l)}\left(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right) \tag{2.10}$$

at layer l. Here, $\sigma^{(l)}$ indicates a fixed nonlinearity, such as a sigmoid or ReLU function. The matrix $\mathbf{W}^{(l)}$ indicates the connection between layers l and $l-1$. Bias is linked to each layer and models the average activation of a specific unit, denoted as $\mathbf{b}^{(l)}, 1 \leq l \leq L$. The weights and biases in this case are the model parameters.

The output $\hat{\mathbf{y}}$ can be obtained by:

$$\hat{\mathbf{y}} = \sigma^{(L)}\left(\mathbf{W}^{(L)}\mathbf{h}^{(L-1)} + \mathbf{b}^{(L)}\right) \tag{2.11}$$

If the network has more than one hidden layer, we often call it *deep*. One key notion is that as we move towards these deeper layers in the network, the hidden units have the ability to discover more intricate and complex features compared to those located in the initial layer.

2.2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are commonly utilized for image processing tasks, with their main component being the convolutional layer. Given an input tensor $\mathbf{X}^{(l-1)}$ with dimensions of width W, height H, and $C^{(l-1)}$ channels, the convolution operation at l-th layer

performs an update at position (i, j) as:

$$\mathbf{x}_{ij}^{(l)} = \sum_{(\Delta_1,\Delta_2)\in\mathscr{K}} \mathbf{W}_{\Delta_1,\Delta_2}^{(l)} \mathbf{x}_{\Delta_1+i,\Delta_2+j}^{(l-1)} + \boldsymbol{b}^{(l)} \tag{2.12}$$

Here, $\mathbf{x}_{(\cdot,\cdot)}^{(l-1)} \in \mathbb{R}^{C^{(l-1)}}$ is a channel vector that indexes the first two dimensions in input tensor $\mathbf{X}^{(l-1)}$. Similarly, $\mathbf{W}_{(\cdot,\cdot)}^{(l)}$ is the channel weight matrix that indexes the first two dimensions in weight tensor $\mathbf{W} \in \mathbb{R}^{k\times k\times C^{(l)}\times C^{(l-1)}}$ at l-th layer and $\boldsymbol{b} \in \mathbb{R}^{C^{(l)}}$ represents the bias. The set $\mathscr{K}$ represents all changes in the spatial domain when using a kernel of size k and is defined as $\mathscr{K} = F(k) \times F(k)$ where $F(k) = \{-\left\lfloor \frac{k}{2} \right\rfloor, \cdots, \left\lfloor \frac{k}{2} \right\rfloor\}$. In essence, CNNs use the convolution process to recognize different features in images, ranging from basic edges to intricate patterns.

2.2.2 Graph Neural Networks

Conventional deep learning methods do not apply readily when working with graphs. For instance, Convolutional Neural Networks (CNNs) work best with grid-structured data, such as images, while recurrent neural networks (RNNs) are more suited to sequentially structured data, such as text. These structures are not invariant to permutations but are assumed to be invariant to translation. Thus, to enable deep learning on general graphs, a new type of deep learning architecture needs to be defined.

A naive approach to apply deep learning to graphs is to use the graph's adjacency matrix $\mathbf{A}$ as input to a neural network. For instance, an entire graph's embedding could be created by flattening the adjacency matrix $f_{flatten}(\mathbf{A})$ such that $f_{flatten} : \mathbb{R}^{|\mathscr{V}|\times|\mathscr{V}|} \rightarrow \mathbb{R}^{|\mathscr{V}|\cdot|\mathscr{V}|}$ and feeding it into a Multi-Layer Perceptron (MLP) to get a global graph representation:

$$\mathbf{z}_g = \text{MLP}(f_{flatten}(\mathbf{A})), \tag{2.13}$$

However, this approach is based on the arbitrary order of nodes in the adjacency matrix, which is an issue. Thus, neural networks for graphs should be permutation invariant $f\left(\mathbf{PAP}^\top\right) = f(\mathbf{A})$ or equivariant $f\left(\mathbf{PAP}^\top\right) = \mathbf{P}f(\mathbf{A})$ with $\mathbf{P}$ being the permutation matrix.

Let $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ indicate a graph composed of a set of nodes $\mathscr{V}$ and a set of edges $\mathscr{E}$. A GNN can be structured such that it takes a graph $\mathscr{G}$ as input, where nodes are associated with feature vectors $\mathbf{x}_i$ where $i \in \mathscr{V}$ and edges can also have feature vectors $\mathbf{x}_{ij}$ where $i, j \in \mathscr{V} \times \mathscr{V}$. The graph structure then dictates the subsequent message passing updates to obtain updated hidden edge and node representations, denoted with $\mathbf{h}_i$ and $\mathbf{h}_{ij}$ respectively, as follows (Gilmer et al., 2017):

$$\begin{aligned} \mathbf{h}_{ij}^{(l)} &= f_{\mathscr{E}}\left(\mathbf{x}_{ij}, \quad \mathbf{h}_i^{(l-1)}, \mathbf{h}_j^{(l-1)}\right) \\ \mathbf{h}_i^{(l)} &= f_{\mathscr{V}}\left(\mathbf{x}_i, \quad \mathbf{h}_i^{(l-1)}, \sum_{j\in\mathscr{N}_i} \mathbf{h}_{ji}^{(l)}\right) \end{aligned} \tag{2.14}$$

We can set initial hidden representation of the node to be $\mathbf{h}_i^{(0)} = \mathbf{x}_i$ here. This flavor of GNN is called the *message-passing neural network* and can be considered as a general model. This network may be used for node- and edge-level tasks. If we want to classify the whole graph (e.g. classification for chemical compounds), the final output of the GNN is aggregated into a global representation $\mathbf{h}_g = \sum_{i \in \mathcal{V}} \mathbf{h}_i$ or $\mathbf{h}_g = \frac{\sum_{i \in \mathcal{V}} \mathbf{h}_i}{|\mathcal{V}|}$.

Models like graph attention networks (Veličković et al., 2018) fit within framework of Equation 2.14. The transformer architecture (Vaswani et al., 2017) can be viewed within the context of MPNN when the MPNN represents a fully connected graph. There are models which are more expressive than MPNN (Battaglia et al., 2018).

Graph Convolutional Networks

This section provides an overview of the graph convolutional network (GCN), a subclass of message-passing networks, introduced by Kipf and Welling (2017). We consider $\mathbf{h}_i^{(l)} \in \mathbb{R}^{d^{(l)}}$ as the hidden representation vector for node $i \in \mathcal{V}$ at the l-th layer. The GCN model performs a single message passing step at layer l as follows:

$$\mathbf{h}_i^{(l)} = \sigma^{(l)} \left(\mathbf{W}_{\text{self}}^{(l)} \mathbf{h}_i^{(l-1)} + \sum_{j \in \mathcal{N}_i} c_{ij} \mathbf{W}_{\text{neigh}}^{(l)} \mathbf{h}_j^{(l-1)} \right), \tag{2.15}$$

Here, $\sigma^{(l)}$ refers to an element-wise non-linearity, e.g. ReLU. $\mathbf{W}_{\text{self}}^{(l)} \in \mathbb{R}^{d^{(l)} \times d^{(l-1)}}$ and $\mathbf{W}_{\text{neigh}}^{(l)} \in \mathbb{R}^{d^{(l)} \times d^{(l-1)}}$ represent weight matrices at layer l, while $\mathcal{N}_i$ is in the neighborhood of node i. The term $c_{ij} = 1/\sqrt{D_{i,i} D_{j,j}}$ is a constant used for normalization based on the degree of node i, represented by $D_{i,i}$. We can represent this in the matrix form as:

$$\mathbf{H}^{(l)} = \sigma^{(l)} \left(\mathbf{H}^{(l-1)} \mathbf{W}_{\text{self}}^{(l)\top} + \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}} \mathbf{H}^{(l-1)} \mathbf{W}_{\text{neigh}}^{(l)\top} \right), \tag{2.16}$$

Here, $\mathbf{D}$ is the diagonal node degree matrix. The sum over neighboring nodes is replaced by multiplying with the normalized adjacency matrix $\mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$. $\mathbf{H}^{(l)} \in \mathbb{R}^{N \times d^{(l)}}$ is the activation matrix at layer l, where $\mathbf{H}^{(0)} = \mathbf{X}$, with $\mathbf{X} \in \mathbb{R}^{N \times F}$ represents the node feature matrix with node feature of length F.

Overfitting may occur with the model in Equation 2.16 if there is not enough labeled nodes, but can be rectified using a single weight matrix as:

$$\mathbf{H}^{(l)} = \sigma^{(l)} \left(\mathbf{H}^{(l-1)} \mathbf{W}^{(l)\top} + \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}} \mathbf{H}^{(l-1)} \mathbf{W}^{(l)\top} \right), \tag{2.17}$$

Kipf and Welling (2017) have shown that renormalizing the adjacency matrix with identity matrix additionally boosts the performance empirically when using a single paramater model.

This single parameter variant of GCN can thus be written as follows:

$$\mathbf{H}^{(l)} = \sigma^{(l)}\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)^\top}\right) \tag{2.18}$$

Here, $\tilde{D}_{i,i} = \sum_j A_{ij} + 1$.

For example, in task where we have to classify *nodes*, we may use a two layer renormalized single parameter model:

$$\hat{\mathbf{Y}} = f_\theta(\mathbf{X},\mathbf{A}) = \text{softmax}\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})\tilde{\mathbf{D}}^{-\frac{1}{2}}\,\text{ReLU}\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{W}^{(1)^\top}\right)\mathbf{W}^{(2)^\top}\right). \tag{2.19}$$

Here, $\mathbf{W}^{(1)} \in \mathbb{R}^{F\times d^{(1)}}$ represents the weights for hidden layer of dimension $d^{(1)}$. $\mathbf{W}^{(2)} \in \mathbb{R}^{d^{(1)}\times k}$ is an output weight matrix where k represents the number of classes. The elements of $\tilde{\mathbf{D}}^{-\frac{1}{2}}$ may be computed before the forward pass. The softmax operator is applied row-wise, we can define it as:

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{2.20}$$

The denominator sums over all entries in the row.

2.2.3 Universal Approximation of Neural Networks

The Universal Approximation Theorem (UAT) provides a justification for using neural networks as learning models. The UAT states that a continuous function $f : \mathbb{D} \to \mathbb{O}$, mapping from the unit hypercube in $\mathbb{D}$ to an output space in $\mathbb{O}$, can be approximated with arbitrary precision by a neural network with one hidden layer (Hornik et al., 1989).

Proof by Hornik et al. (1989) verifies the existence of the approximation without providing a method to obtain the specific model parameters. In contrast, constructive proofs, which provide a method for obtaining a solution, exist for MLPs with two hidden layers (Scarselli and Chung Tsoi, 1998).

However, these proofs suggest that even for simple functions, a large number of hidden units may be needed, potentially rendering the model impractical due to computational inefficiency or overfitting risks. So, although a neural network can theoretically mimic any function, the required number of hidden units may limit its practical use.

This limitation brings us to the question of network architecture. While multi-layer feed-forward networks are confirmed universal approximators, their architecture development remains largely empirical. The Universal Approximation Theorem does not address the networks' learnability, despite various architectures potentially reaching identical optimal approximations under suitable parameters.

This expansive parameter space represents a significant challenge for deep learning practitioners. Deep learning uses gradient descent optimization, but it doesn't ensure convergence

to global minima when applied to non-convex functions, which highlights the importance of network architecture. This architecture shapes the parameter space landscape, with different ones creating diverse terrains.

Network architecture can boost learnability by affecting convergence speed, network stability, initial condition robustness. Theoretically modeling these features is a challenge, with most proposed methods assessed on empirical data. Consequently, the field of deep learning primarily uses an empirical paradigm that incorporates effective priors to adapt to new data.

2.3 Optimization

Previously, we saw how the process of training neural networks can be viewed as a problem of optimization. Our goal is to find the best suitable function, denoted f^*, from a set of potential functions $\mathscr{F}$. The optimization approach can sometimes be misleading due to the issue of overfitting, as we often minimize a surrogate loss, i.e. the empirical loss on some specific training dataset. However, it is important to note that our main goal is not to find a model that minimizes this surrogate loss, but one that reduces the expected loss. It is important to note that, for neural networks, this loss is typically nonconvex, making it difficult to analyze mathematically the convergence properties of this optimization method. The key is usually to reduce the said empirical loss through an optimization method. In the following sections, we will briefly cover some of these gradient-based optimization methods commonly used in learning.

2.3.1 Gradient Descent

Gradient descent (GD) is a method that uses first-order derivatives of a differentiable loss function $\mathscr{L}$ quantifies the user-specified difference between the model's output and the labels based on the model parameters θ. This function is usually non-convex. The procedure is iterative, at every timestamp t, given the value of the present parameter θ_t, a Taylor series of first-order loss function $\mathscr{L}$ around θ_t is evaluated. We see that going in the negative direction of the gradient decreases this Taylor series' value. Assuming that the step size is sufficiently minor, the approximation of $\mathscr{L}$ remains valid, and it will reduce the actual function $\mathscr{L}$ in the process. We denote the gradients via the Jacobian row vector

$$\nabla \mathscr{L} = \left[\frac{\partial \mathscr{L}}{\partial \theta_1}, \frac{\partial \mathscr{L}}{\partial \theta_2}, ..., \frac{\partial \mathscr{L}}{\partial \theta_{n_\theta}} \right] \tag{2.21}$$

The pseudocode for implementing the gradient descent algorithm is outlined below. Each gradient at t is multiplied by the learning rate $\eta_t \in \mathbb{R}_+$ to adjust the step size.

In practice, a more efficient method involves approximating the step $\Delta\theta$ that Gradient Descent takes using only a single or a small number of data examples, which improves the speed of convergence (Lecun et al., 1998). This adaptation gives rise to Stochastic Gradient Descent

Algorithm 1 Gradient Descent

Set θ_0 = INIT()
for $t = 1...T$ **do**
  Set $\Delta\theta = \mathbf{0}$
  for each $\mathbf{x} \in \mathscr{D}$ **do**
    Update $\Delta\theta = \Delta\theta - (\nabla\mathscr{L}(\mathbf{x}, f_{\theta_t}))^\top$
  end for
  Update $\theta_{t+1} = \theta_t + \eta_t \Delta\theta$
end for

(SGD) or Mini-batch Gradient Descent (MGD). In this context, a *mini-batch* indicates a set of examples from the training set that contribute to the gradient calculation. We outline the algorithm for SGD below.

Algorithm 2 Stochastic Gradient Descent

Set θ_0 = INIT()
for $t = 1$ to T **do**
  Set $\mathbf{x}_i$ = random sample $\in \mathscr{D}$
  Set $\Delta\theta = (\nabla\mathscr{L}(\mathbf{x}_i, f_{\theta_t}))^\top$
  Update $\theta_{t+1} = \theta_t - \eta_t \Delta\theta$
end for

These stochastic techniques rely on gradient estimates that generally point towards the desired direction, thus allowing faster convergence towards a local minimum and help escape undesired narrow minima.

Mini-batch GD offers an advantage over SGD by minimizing the variation in gradient estimates through the use of multiple examples at almost no additional overhead through the use of libraries for vectorization. This reduction in variance allows a larger learning rate η, which again is arguably leading to faster convergence.

One additional variant involves the introduction of momentum (Nocedal and Wright, 2006), which uses a moving average of the gradients. Other methods focus mainly on customizing the learning rate, sometimes assigning a unique learning rate to each parameter (Schaul and LeCun, 2013). Commonly used optimizer nowadays is Adam (Kingma and Ba, 2014).

3 Dataset

Multi-camera tracking in fields like autonomous driving and surveillance requires quality datasets. Many existing datasets fall short in areas such as view overlap, recording duration, or they deal with low-resolution videos and staged setups.

Addressing these shortcomings, we present the INVISION dataset. Captured at a busy train station, it features eight synchronized cameras with overlapping views, showcasing real-world challenges like occlusions and blind spots. It includes over 300,000 annotated bounding boxes and represents more than 150 individuals, making it ideal for deep learning detectors and trackers.

Further sections will delve into dataset specifics, calibration, and the annotation process. We will use this dataset to evaluate the performance of frameworks using object appearance in a multi-view setting.

3.1 Related Work

In the multi-camera, multi-object tracking domain, several data sets have served as a basis for research. One of the earliest was the EPFL dataset (Fleuret et al., 2008; Berclaz et al., 2011), a

Figure 3.1: Views from eight synchronized cameras and their respective annotations.

collection of Laboratory, Campus, and Terrace sequences, along with a Passageway sequence. These datasets, spanning from 2.5 to 20 minutes, covered both indoor and outdoor environments and showcased features such as actor tracking, calibrated cameras, overlapping views, albeit with a limited number of distinct ID annotations.
The Issia Soccer dataset (D'Orazio et al., 2009) deviated from the actor-tracking approach and, instead, focused on tracking non-actors across a 2-minute, high-definition soccer game recorded by six calibrated cameras. The Apidis Basket dataset (Vleeschouwer et al., 2008) followed a similar trajectory.
The PETS2009 dataset (Ferryman and Shahrokni, 2009) has 3 sequences of varying tracking difficulty. It has been widely despite calibration inaccuracies and synchronization issues, indicating the demand for better data sets.
The KITTI dataset (Geiger et al., 2012) introduced sequences with car and pedestrian annotations captured via mobile stereo cameras. In NLPR MCT series (Chen et al., 2016) and USC Campus dataset (Kuo et al., 2010a) the cameras have non-overlapping views and lack calibration. The CamNeT dataset (Zhang et al., 2015a) integrated elements from previous datasets and introduced a blend of indoor and outdoor scenes captured by 5-8 cameras, but it still lacked calibration.
The SALSA dataset (Alameda-Pineda et al., 2015) utilized multiple static cameras, although its long-duration video primarily featured static pedestrians, which is not a challenging tracking problem.
The EPFL-RLC dataset (Chavdarova and Fleuret, 2017) featured high-resolution, high frame rate video from three calibrated static cameras and overlapping views. Still, only the final 300 frames included ground truth annotations.
Similarly, the WILDTRACK dataset (Chavdarova et al., 2018) was recorded with seven static cameras with overlapping views, but ground truth annotations were only available for the last 400 frames.

Table 3.1: Existing datasets for multi-camera multi-object tracking

Dataset	Author	Duration	Cams	Actors	Overlap	Blind Spots	Calibrated	Resolution	FPS	Scene	Mobile/Static
Laboratory	Fleuret et al. (2008)	2.5 min	4	Yes	Yes	No	Yes	320 x 240	25	Indoor	S
Campus	Fleuret et al. (2008)	5.5 min	3	Yes	Yes	No	Yes	320 x 240	25	Outdoor	S
Terrace	Fleuret et al. (2008)	3.5 min	4	Yes	Yes	No	Yes	320 x 240	25	Outdoor	S
Passageway	Berclaz et al. (2011)	20 min	4	Yes	Yes	No	Yes	320 x 240	25	Mixed	S
Issia Soccer	(D'Orazio et al., 2009)	2 min	6	No	Yes	No	Yes	1920 x 1080	25	Outdoor	S
Apidis Basket.	(Vleeschouwer et al., 2008)	1 min	7	No	Yes	No	Yes	1600 x 1200	22	Indoor	S
PETS2009	(Ferryman and Shahrokni, 2009)	1 min	8	Yes	Yes	No	Yes	768 x 576	7	Outdoor	S
NLPR MCT 1	(Chen et al., 2016)	20 min	3	No	No	Yes	No	320 x 240	20	Mixed	S
NLPR MCT 2	(Chen et al., 2016)	20 min	3	No	No	Yes	No	320 x 240	20	Mixed	S
NLPR MCT 3	(Chen et al., 2016)	4min	4	Yes	Yes	Yes	No	320 x 240	25	Indoor	S
NLPR MCT 4	(Chen et al., 2016)	25 min	5	Yes	Yes	Yes	No	320 x 240	25	Mixed	S
Dana36	Per et al. (2012)	N/A	36	Yes	Yes	Yes	No	2048 x 1536	N/A	Mixed	S
USC Campus	(Kuo et al., 2010a)	25 min	3	No	No	Yes	No	852 x 480	30	Outdoor	S
CamNeT	(Zhang et al., 2015a)	30 min	8	Yes	Yes	Yes	No	640 x 480	25	Mixed	S
KITTI	(Geiger et al., 2012)	7 min	4	No	NA	-	Yes	1392 x 512	10	Outdoor	M
SALSA	(Alameda-Pineda et al., 2015)	60 min	4	No	Yes	No	No	1024 x 768	15	Indoor	S
EPFL-RLC	(Chavdarova and Fleuret, 2017)	2 min	3	No	Yes	No	Yes	1920 x 1080	60	Indoor	S
WILDTRACK	(Chavdarova et al., 2018)	60 min	7	No	Yes	No	Yes	1920 x 1080	60	Outdoor	S
Ours		8h	8	No	Yes	Yes	Yes	1920 x 1080	15	Indoor	S

3.2 Acquisition and Calibration

The setup consists of 8 static cameras at busy train station with resolution of 1920×1080 pixels recording at 15 FPS.
The station was recorded using a GoPro Max 360 camera. The recording was done by walking through the station to capture all the necessary details. The recorded visuals were then reconstructed using Structure from Motion (SfM) software, namely OpenSfM and COLMAP. These tools converted the 360-degree videos into a 3D model.

To calibrate the cameras, for each camera view, correspondences between the 3D model and the view were manually identified. Floor tiling was also used as a guide to help identify straight lines. The camera-specific intrinsic matrix $\mathbf{K}_c$ can be written as:

$$\mathbf{K}_c = \begin{bmatrix} f_{x_c} & s_c & u_{0_c} \\ 0 & f_{y_c} & v_{0_c} \\ 0 & 0 & 1 \end{bmatrix}$$

Here, f_{x_c} and f_{y_c} are the focal lengths along the X and Y axes for camera c, s_c is the skew factor, and u_{0_c} and v_{0_c} are the coordinates of the principal point for camera c. The extrinsic matrix $\mathbf{E}_c$ represents the pose of camera c in the world coordinates:

$$\mathbf{E}_c = \begin{bmatrix} \mathbf{R}_c & \mathbf{t}_c \\ \mathbf{0} & 1 \end{bmatrix}$$

Here, $\mathbf{R}_c$ is a 3x3 rotation matrix representing the orientation of the camera c, and T_c is a 3x1 translation vector representing the position of the camera c in the world. Furthermore, we consider distortion parameters that contribute to the intrinsic matrix for each camera c. For simplicity, we only take into account the first two radial distortion parameters k_{1_c} and k_{2_c}, and assume that the tangential distortion parameters are zero. Therefore, our distortion coefficients for camera c become $\mathbf{D}_c = [k_{1_c}, k_{2_c}, 0, 0, 0]$. Given a dataset $\mathscr{D}_c$ containing N corresponding projections in the view of camera c, specifically: $\mathscr{D}_c = \{\mathbf{p}_i^c\}_{i=1}^N$, where $\mathbf{p}_i^c \in \mathbb{R}^2$. We aim to determine projection matrix $\mathbf{P}_c$, whose elements are included in $\{\mathbf{K}_c, \mathbf{E}_c, \mathbf{D}_c\}$ and the $3D$ points $\mathscr{M}_c = \{\mathbf{m}_i\}_{i=1}^N, \mathbf{m}_i \in \mathbb{R}^3$ in such a way that:

$$\mathbf{K}_c^*, \mathbf{E}_c^*, \mathbf{D}_c^* := \underset{\mathbf{D}_c, \mathbf{K}_c, \mathbf{E}_c, \mathscr{M}}{\operatorname{argmin}} \sum_{i=1}^{N} \left| \mathbf{p}_i^c - \mathbf{P}_c \mathbf{m}_i \right|_2^2, \tag{3.1}$$

Here, $|\cdot|_2$ denotes the Euclidean norm, and $\mathbf{P}_c$ is the projection matrix for camera c whose parameters are contained in $\{\mathbf{K}_c, \mathbf{E}_c, \mathbf{D}_c\}$. This formulation represents the optimization problem for each individual view, with its own set of intrinsic and extrinsic parameters, which are optimized to minimize the re-projection error of the points in that view. We use Ceres C++

library to jointly optimize intrinsic and extrinsic parameters from Eq. 3.1 for each camera c.

3.3 Annotation

In our approach, we adjust the position, dimensions, and orientation of a 3D bounding box so that its projections accurately correspond across all views with the object being marked. To gain correspondence across views, we reproject the base of the annotated bounding boxes onto the world ground plane. We provide the code for the tool[1].

Given the homogeneous pixel coordinates $\tilde{\mathbf{u}}_i$ of the object's base point in the view from camera i, along with the intrinsic matrix $\mathbf{K}_i$, rotation matrix $\mathbf{R}_i$, and translation vector $\mathbf{t}_i$ of camera i, we compute an intermediate ray direction $\mathbf{l} = \mathbf{R}_i^\top \cdot \mathbf{K}_i^{-1} \cdot \tilde{\mathbf{u}}_i$ and the camera's position in world coordinates $\mathbf{c}_i = -\mathbf{R}_i^\top \cdot \mathbf{t}_i$. The ray from the optical center of camera i through the pixel's image coordinate in 3D space is given by $\mathbf{Q}_i(\lambda) = \mathbf{c}_i + \lambda \cdot \mathbf{l}_i$, where z is the third component of the respective vectors. By setting $z = 0$, the world point is found by scaling $\mathbf{l}$ by the factor $\lambda = -c_{i_z} / l_z$, ensuring that the ray intersects the ground plane at the correct point. The world point $\mathbf{p}$, reflecting the intersection of a ray with a plane in 3D space, is given by:

$$\mathbf{p} = \mathbf{c}_i - \mathbf{l} \cdot \frac{c_{i_z}}{l_z} \tag{3.2}$$

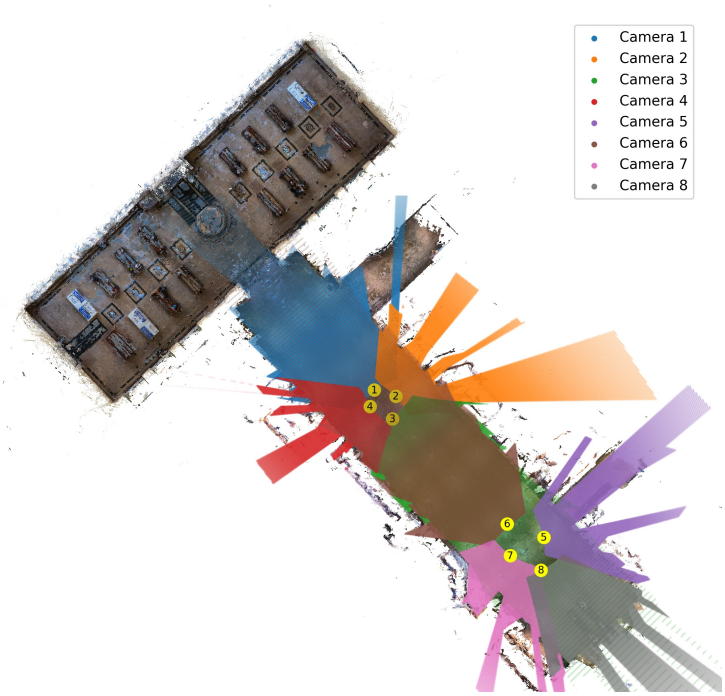
Given the reprojected ground point, the projection to the 2D image plane in homogeneous coordinates can be written for any other camera view j:

$$\tilde{\mathbf{u}}_j = \mathbf{K}_j \cdot (\mathbf{R}_j \cdot \mathbf{p} + \mathbf{t}_j) \tag{3.3}$$

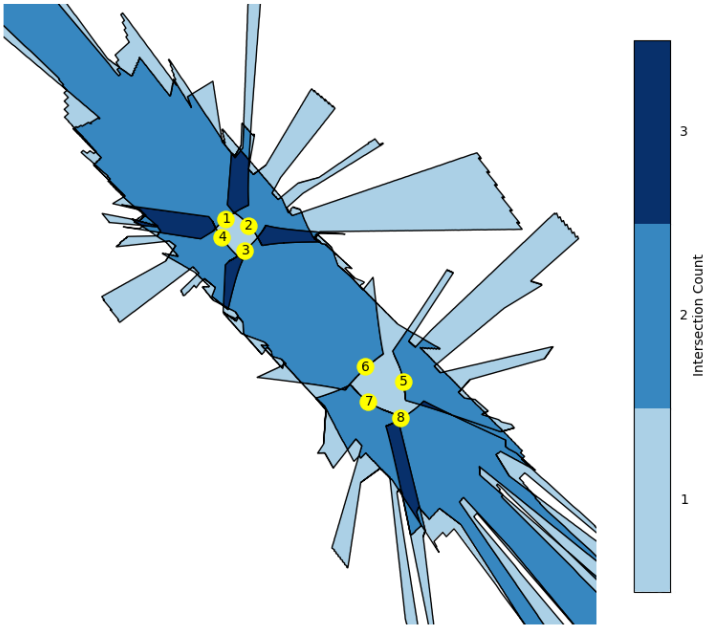
Here, $\tilde{\mathbf{u}}_j$ represents the projected coordinates in the 2D image plane corresponding to the view from camera j.

By using the accurate camera calibration and the precise ground plane information, the projected bounding boxes across the multiple views can be aligned accurately. This reprojection to the world ground helps overcome the inherent ambiguity and potential occlusion issues when annotating in a single view. This process is similar to the adjustment of a 3D position as proposed by Chavdarova et al. (2018) but now incorporates the adjustment of orientation and dimensions. Specifically, we represent the object using its dimensions: height, width, and length, along with the position of the base in the world (x, y, z), and a rotation θ around the z-axis. In the procedure of converting a cuboid from the ground to the world coordinate system, we define the eight vertices of the cuboid based on these dimensions. We store the transformation to the world coordinate system in a single cuboid-to-world transformation matrix $\mathbf{T}_{c2w}$

[1] https://github.com/ivanvrkic/multicam-gt



(a) Bird's-eye view (BEV) of the cameras and their respective fields of view.



(b) Overlap between the camera views.

Figure 3.2: Bird's-eye view (BEV) of the cameras and their overlap.

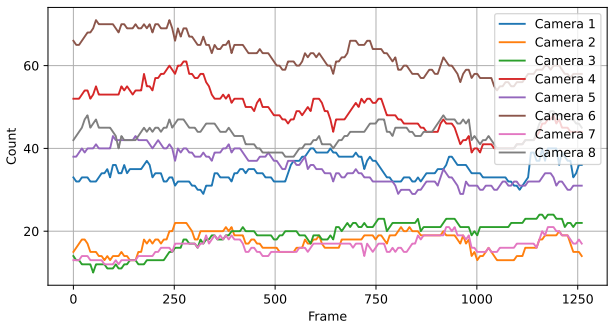


Figure 3.3: Pedestrian count per frame.

Cam	Pedestrian Count	Mean
1	43890	34.64
2	21868	17.26
3	23961	18.91
4	62132	49.04
5	44786	35.35
6	79254	62.55
7	20685	16.33
8	55307	43.65

Figure 3.4: Annotation statistics for each camera.

$$\mathbf{T}_{c2w} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & x \\ \sin(\theta) & \cos(\theta) & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

The matrix $\mathbf{T}_{c2w}$ encapsulates both the rotation around the z-axis and the translation by the world point (x, y, z). With this transformation matrix, we obtain the transformed vertices of the cuboid in the world coordinate system.
The 3D points that have been transformed can be subsequently projected onto the image plane, resulting in the 2D representation of the object's bounding box from different camera perspectives.

3.4 Dataset Statistics

We denote a frame I^t as set of images from the cameras $\{I_i^t\}_{i=1}^8$ at timestep t. We provide manual annotations for 1274 frames. Distribution of people across frames is shown in Figure 3.3. Single view statistics are shown in Figure 3.4. This results to a total of 351883 detections.

4 Learning Appearance

Appearance learning plays an important role in image retrieval to establish correspondence between images from different viewpoints. Given a reference image, the goal of a image retrieval system is to identify another image containing the same object but viewed from an alternate perspective. This approach has applications in domains such as product retrieval for online shopping (Cai et al., 2020; Liu et al., 2012), localization for deliveries (Zhu et al., 2018), and management in urban planning (Tang et al., 2019, 2020).

In the context of tracking objects over time and across varied camera configurations, appearance is crucial. This becomes important in applications that include autonomous vehicles and robotics. However, leveraging appearance for consistent and accurate tracking has its set of challenges. First, there is the possibility of misidentifying objects that have similar appearances. Factors like lighting variations, background dynamics, changes in camera positions and angles, occlusions, or other external elements can impact the consistent perception of an object's appearance across separate captures. Creating representations that are resistant to these intraclass changes is important. Efforts to address this range from improving visual features to in-depth metric learning. Another problem is annotation, as it requires significant resources when working with large datasets with bounding boxes and object associations. In addition, the number of training images for each ID of the object is often limited, which affects the learning of the variance of the object.

Image retrieval (or re-identification in this context) and multi-object tracking (MOT – or MC-MOT if we are working with multiple cameras) stand out as different methods with specific purposes and evaluation metrics. While retrieval system focuses on ranking distances relative to a particular query, MOT determines if two images are the same or not. They are judged by distinct standards: retrieval system examines the accuracy of ranking, and MOT evaluates the precision of classification. They require different learning approaches.

In calibrated multi-camera setup with overlapping views, capturing objects from various angles can overcome limitations like occlusions or variable lighting. Using multiple crops of the same object from different angles, we propose a multi-view retrieval framework that offers a more comprehensive representation of objects compared to traditional single-crop methods.

Figure 4.1: **UMAP visualization of learned model embeddings for INVISION dataset.** Semantically similar features cluster closely together.

4.1 Related Work

Early studies emphasized the role of human appearance using color histograms and texture patterns (Chen et al., 2011a; Kuo et al., 2010b; Gilbert and Bowden, 2006). Enhanced discrimination is achieved using saliency data (Martinel et al., 2014) or specialized characteristics for different attributes of pedestrians (Kuo et al., 2010b).

State-of-the-art techniques use deep learning (He et al., 2015; Liu et al., 2021). Recent works employ pedestrian attributes for multitask learning (Wang et al., 2018; Lin et al., 2017; Su et al., 2016) and use pedestrian alignment and local spatial cues (Yi et al., 2014; Sun et al., 2017; Li et al., 2017), some integrating pose estimation (Zheng et al., 2017a; Suh et al., 2018; Su et al., 2017) and human parsing (Kalayeh et al., 2018) for enhanced matching. In an effort to avoid model overfitting and improve results, data enhancement is used (Zhong et al., 2017). One area of research is to use a GAN to generate new samples (Zheng et al., 2017b).

Loss functions play an important role in retrieval solutions. Most of the methods in the field are based on deep metric learning (Liu et al., 2016b; Jun et al., 2019; Diao et al., 2021), where deep learning models transform images into vector representations. When ranking, the query embedding is compared with the gallery embedding to find the most similar matches. Earlier methods, such as those in (Zhang et al., 2017), used cross-entropy loss, considering appearance learning as a multi-class classification problem. Nowadays, there is a shift towards ranking losses. Schroff et al. (2015) introduce the triplet loss which directly optimizes the embedding. Hermans et al. (2017) demonstrated that triplet loss can be used for end-to-end learning when combined with hard example mining. Although triplet loss outperforms many methods, it faces challenges with computationally intensive hard negative sampling, potential suboptimal minima, and susceptibility to outliers and noisy labels (Do et al., 2019; Yuan et al., 2019; Zhang et al., 2020b; Zhou et al., 2017).

4.2 Multi-View Retrieval

Given a multi-camera setup, we represent object i as a set of detections $\mathscr{X}_i = \{\mathbf{x}_j^i\}_{j=1}^k$, where k is the number of cameras from which the object is visible, and each detection $\mathbf{x}_j^i \in \mathbb{R}^d$. The instance set, which contains m objects, is given by $\mathscr{I} = \{\mathscr{X}_1, \mathscr{X}_2, \ldots, \mathscr{X}_m\}$. The associated labels for these samples are in the label vector $\boldsymbol{y} \in \{1, 2, \ldots, C\}^m$ where C is the number of IDs.

Each detection $\mathbf{x}_j^i \in \mathscr{X}_i$, is mapped to an l-dimensional space unit sphere by the function $f_{\boldsymbol{\theta},\kappa} : \mathbb{R}^d \rightarrow S^l$. Here, f is a neural network with parameters $\boldsymbol{\theta}$ and κ is a L2 normalization layer. Considering two camera data sets, $\mathscr{X}_a$ and $\mathscr{X}_b$, we construct a set of similarity scores, $\mathscr{S}_{ab}$, containing the pairwise dot products between their members after they are passed through the neural network. Specifically, for each $\mathbf{x}_i^a$ in $\mathscr{X}_a$ and $\mathbf{x}_j^b$ in $\mathscr{X}_b$, a score is calculated:

$$S_{ij} = f_{\boldsymbol{\theta},\kappa}(\mathbf{x}_i^a)^\top f_{\boldsymbol{\theta},\kappa}(\mathbf{x}_j^b) \tag{4.1}$$

Figure 4.2: **Proposed Multi-View Retrieval Results.** Since we are working with calibrated multi-camera setup, one object can be seen from from multiple views, thereby enhancing the results. In this figure a query is seen from 3 different views. First retrieved result is seen from 3 different views, whereas the second result is seen from 2 different views. False positives are shown in red.

Thus, $\mathscr{S}_{ab}$ contains $a \times b$ scores and can be represented as $\mathscr{S}_{ab} = \{S_{11}, S_{12}, \ldots, S_{ab}\}$. The overall similarity between $\mathscr{X}_a$ and $\mathscr{X}_b$ is derived from $\mathscr{S}_{ab}$ as:

$$\text{Similarity}(\mathscr{X}_a, \mathscr{X}_b) = \bigoplus(\mathscr{S}_{ab}) \tag{4.2}$$

Here $\bigoplus$ represents some permutation-invariant operation such as sum, mean, or maximum.

4.2.1 Object Position and Dimension

Let us denote the camera's intrinsic matrix as $\mathbf{K}$ and its extrinsic parameters as rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$. Given the image points at the bottom (u_b, v_b) and top (u_t, v_t) of the bounding box, we can back-project these points to 3D rays in the camera-centric coordinates as:

$$\mathbf{p}_c^b = \mathbf{K}^{-1} \begin{bmatrix} u_b \\ v_b \\ 1 \end{bmatrix}, \qquad \mathbf{p}_c^t = \mathbf{K}^{-1} \begin{bmatrix} u_t \\ v_t \\ 1 \end{bmatrix}. \tag{4.3}$$

For simplicity, we assume that the distortion coefficients are 0 here. A point $\mathbf{p}_c$ in the camera's frame and its corresponding point $\mathbf{Q}(\lambda)$ in the world frame are related by:

$$\mathbf{Q}(\lambda) = \mathbf{R}^{-1}(\lambda \mathbf{p}_c - \mathbf{t}) \tag{4.4}$$

The bottom of the bounding box, denoted as $\mathbf{Q}_b$, is considered as a reference point located on the ground surface, where the elevation or height (in terms of the z-coordinate) is defined as 0.

From which, λ_b can be derived as:

$$\lambda_b = \frac{\mathbf{R}_3^{-1} \cdot \mathbf{t}}{\mathbf{R}_3^{-1} \cdot \mathbf{p}_c^b} \tag{4.5}$$

Here, $\mathbf{R}_3^{-1}$ indexes the third row in $\mathbf{R}^{-1}$. Plugging in λ_b into the ray Equation 4.4 we get the base point in the world frame. Considering $\mathbf{Q}_t$ as the top of the bounding box and being vertically aligned with $\mathbf{Q}_b$, then $\mathbf{Q}_{2_x} = \mathbf{Q}_{1_x}$ and $\mathbf{Q}_{2_y} = \mathbf{Q}_{1_y}$. Using these relations we get:

$$\lambda_t = \frac{\mathbf{R}_1^{-1}(\lambda_b \cdot \mathbf{p}_c^b)}{\mathbf{R}_1^{-1} \cdot \mathbf{p}_c^t} \tag{4.6}$$

Height is then given by:

$$h = \mathbf{Q}_{t_z} = \mathbf{R}_3^{-1}(\frac{\mathbf{R}_1^{-1}(\lambda_b \cdot \mathbf{p}_c^b)}{\mathbf{R}_1^{-1} \cdot \mathbf{p}_c^t}\mathbf{p}_c^t - \mathbf{t}) \tag{4.7}$$

4.2.2 Crop Extraction

The extraction process for image crops is carried out so that the bounding box defined by the height of the object h and a fixed factor m_l lies in the plane perpendicular to the camera ray projected on the ground that pierces the base of the object. This is carried out through a combination of geometric transformations.

We calculate the camera position in the object coordinates $\mathbf{c}_o$ as:

$$\mathbf{c}_o = (-\mathbf{R}^\top \cdot \mathbf{t}) - \mathbf{b}_w \tag{4.8}$$

where $\mathbf{R}^\top$ is the transpose of the camera rotation matrix, and $\mathbf{t}$ is the camera translation vector, and $\mathbf{b}_w$ is the base point of the object in world coordinates. $-\mathbf{R}^\top \cdot \mathbf{t}$ represents the camera position in the world coordinates. We assume flat ground, i.e. $z = 0$ for all object base points. Next, we find a vector (denoted $\mathbf{v}$) perpendicular to the camera ray piercing the object base. This is computed as a cross product of $\mathbf{c}_o$ and the unit vector of the z-axis:

$$\mathbf{v} = \mathbf{c}_o \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.9}$$

This vector is normalized and scaled by a fixed factor m_l :

$$\mathbf{v}_l = m_l \cdot \frac{\mathbf{v}}{||\mathbf{v}||} \tag{4.10}$$

We use $m_l = 0.3$ (30cm) in our experiments. A z-margin m_z is introduced to account for

calibration error for the four corners of the bounding box in world coordinates (denoted as $\mathbf{p}_w^{\text{lb}}$, $\mathbf{p}_w^{\text{lt}}$, $\mathbf{p}_w^{\text{rb}}$, and $\mathbf{p}_w^{\text{rt}}$ for the left bottom, left top, right bottom, and right top point, respectively). We use $m_z = 0.1$:

$$\begin{aligned} \mathbf{p}_w^{\text{lt}} &= \mathbf{v}_l + \mathbf{b}_w + \begin{bmatrix} 0 \\ 0 \\ m_z + h \end{bmatrix}, \quad \mathbf{p}_w^{\text{rt}} = -\mathbf{v}_l + \mathbf{b}_w + \begin{bmatrix} 0 \\ 0 \\ m_z + h \end{bmatrix} \\ \mathbf{p}_w^{\text{lb}} &= \mathbf{v}_l + \mathbf{b}_w + \begin{bmatrix} 0 \\ 0 \\ -m_z \end{bmatrix}, \qquad \mathbf{p}_w^{\text{rb}} = -\mathbf{v}_l + \mathbf{b}_w + \begin{bmatrix} 0 \\ 0 \\ -m_z \end{bmatrix} \end{aligned} \tag{4.11}$$

where h is the size of the detected object along the Z-axis, i.e. height. The bounding box in world coordinates is defined as $\mathbf{P}_w = [\mathbf{p}_w^{\text{lb}}, \mathbf{p}_w^{\text{rb}}, \mathbf{p}_w^{\text{rt}}, \mathbf{p}_w^{\text{lt}}]$, which is then converted into a homogeneous representation in the image coordinates.

$$\tilde{\mathbf{U}} = \mathbf{K} \cdot [\mathbf{R}|\mathbf{t}] \cdot \tilde{\mathbf{P}}_w^{\top} \tag{4.12}$$

where $\mathbf{K}$ is the camera matrix, $\mathbf{R}$ is the camera rotation matrix, and $\mathbf{t}$ is the camera translation vector. We normalize the homogeneous representation of the points $[x', y', w']$ in $\tilde{\mathbf{U}}$ to get the representation $\mathbf{U}$ in Cartesian coordinates, i.e., $[x', y'] = [x'/w', y'/w']$.

To get the final representation of the crop, we apply perspective warping of the bounding box in the image plane represented by $\mathbf{U}$ using a 3x3 homography transformation matrix $\mathbf{H}$. We find $\mathbf{H}$ such that the points in $\mathbf{U}$ are transformed into known points in the crop of fixed size $\mathbf{U}_{crop}$, i.e. $\tilde{\mathbf{U}}_{crop} = \mathbf{H} \cdot \tilde{\mathbf{U}}$

With this, we can successfully extract the region of interest from the image, making it suitable for subsequent analyses. An example implementation in C++ using Eigen (Guennebaud et al., 2010) and OpenCV (Bradski, 2000) is provided in the Appendix A.2.

4.2.3 Feature Extraction

We use Swin Transformer (Liu et al., 2021) as the primary feature extraction backbone. The Swin Transformer captures both local and global features due to its hierarchical structure and shifted window mechanism, making it effective for retrieval system tasks. We also use ResNet (He et al., 2015) as a baseline.

We use a surrogate classification loss related to identity and circle loss to directly optimize the embedding. The identity classification loss is described as:

$$\mathscr{L}_{\text{ID}} = -\sum_{c=1}^{C^{\text{ID}}} y_c \log\left(\hat{y}_c\right)$$

(a) Failure case of standard single-view retrieval approach.

(b) Results of the proposed approach.

Figure 4.3: Comparison of the single-view and multi-view setup.

where C^{ID} is the total number of identity classes.

We use circle loss (Sun et al., 2020) as it is a more general formulation of triplet loss (Schroff et al., 2015). Given a detection of some object i, i.e. $\mathbf{x}_j^i \in \mathcal{X}_i$, and its embedding $f_{\boldsymbol{\theta}}(\mathbf{x}_j^i)$, we consider there to be K within-class similarity scores and L between-class similarity scores related to embedding $f_{\boldsymbol{\theta}}(\mathbf{x}_j^i)$, represented as $\{s_p^k\}_{k=1}^K$ and $\{s_n^l\}_{l=1}^L$ respectively. We use the simplified two-margin variant defined as:

$$\mathcal{L}_{\text{circle}} = \log\left[1 + \sum_{i=1}^{K}\sum_{j=1}^{L} \exp\left(\gamma\left(\left[s_n^j + m_n\right]_+ (s_n^j - m_n) - \left[2 - (m_p + s_p^i)\right]_+ (s_p^i - m_p)\right)\right)\right] \tag{4.13}$$

where γ is the scale factor, while m_n and m_p indicate the negative and positive margin, respectively. The collective objective function used during the training process is:

$$\mathcal{L} = \lambda_i \mathcal{L}_{\text{ID}} + \lambda_c \mathcal{L}_{\text{circle}} \tag{4.14}$$

We augment our data using Random Erasing (Zhong et al., 2017), making the model resilient against occlusions and diversifying feature focus. During training, we introduce linear warm-up to stabilize early training epochs by gradually adjusting the learning rate.

4.3 Experiments

Our approach was implemented in PyTorch. For reproducibility, seeds were fixed across all runs. We use mean, max and median as an aggregator for similarity in Eq. 4.2. We use Rank@k to quantify the proportion of correct top-k matches. We focus on the proposed INVISION dataset (Chapter 3) for evaluation. We use Swin and ResNet backbones pretrained on ImageNet (Deng et al., 2009). We train the models on a 3150 frame sequence using noisy labels from Ali et al. (2023), and evaluate the the model on a non-overlapping annotated data containing 1274 frames discussed in Chapter 3. We denote circle loss and random erasing as CL and RE, respectively. We introduce temporal consistency (TC) such that $t_i < t_j$ when comparing objects i and j, applicable in tracking. To make the results comparable with a multi-view setting, we select only the top-scoring crop for each object at a given time in a single-view setting and we also ignore results from the same timestamp and object as the query.

4.3.1 Results and Ablation

We present the results in Table 4.1. The basic ResNet model without temporal consistency (TC) and Multi-View Retrieval (MVR) records a Rank@1 score of 0.475. Adding TC to the ResNet boosts the Rank@1 score to 0.514. With MVR, different aggregators (mean, median, maximum) affect performance. Maximum consistently performs best for ResNet configurations. Adding Circle Loss (CL) and Random Erasing (RE) improves performance. Swin combined with RE, CL, MVR, and TC achieves the highest Rank@1 score of 0.973. In comparable configurations, the Swin backbone outperforms ResNet.

Name	MVR	TC	AGG	Rank@1	Rank@5	Rank@10
ResNet			N/A	0.475145	0.573410	0.621195
ResNet		✓	N/A	0.513455	0.593570	0.682340
ResNet	✓		MEDIAN	0.713333	0.773333	0.800000
ResNet	✓	✓	MEDIAN	0.660000	0.806667	0.860000
ResNet	✓		MEAN	0.846667	0.926667	0.953333
ResNet	✓		MAX	0.846667	0.953333	0.973333
ResNet	✓	✓	MAX	0.913333	0.963333	0.993333
ResNet+CL	✓	✓	MAX	0.926667	0.976667	0.993333
Swin+CL	✓	✓	MAX	0.946667	0.983333	0.993333
Swin+RE	✓	✓	MAX	0.96667	0.986667	0.993333
Swin+RE+CL	✓	✓	MAX	**0.973333**	**0.986667**	**0.993333**

Table 4.1: Rank@k performance on the INVISION dataset. MVR denotes the proposed Multi-View Retrieval approach, while TC denotes a flag enforcing temporal consistency.

4.3.2 Discussion

Single-view capture methods can struggle with occlusions and variable lighting, as shown in Figure 4.3. Using multiple crops of an object from different angles, can avoid the limitations of traditional single-crop methods. Our approach, grounded in basic camera geometry, provides a more robust representation of objects without added complexity. We show the effectiveness and robustness of the approach through experiments on a multi-camera dataset.

5 Multi-View Tracking

Multi-camera multi-object tracking (MC-MOT) has applications in robotics, surveillance, augmented reality, and sports analytics. However, it's challenged by occlusions, viewpoint variations, and changes in illumination. Traditional methods handle MC-MOT by breaking down the process into separate stages: detection, feature extraction, association, and trajectory generation.

There's now a shift towards combining these stages for an end-to-end solution. This is where Graph Neural Networks (GNNs) become relevant due to their ability to process structured data, making it possible to *learn* the inter-camera and intra-camera associations. However, the diversity in data from different camera views requires a specialized network. Heterogeneous GNNs, designed to process varied data types, offer a potential solution for MC-MOT.

In this chapter, we present a framework that uses heterogeneous GNNs for MC-MOT, aiming to address the challenges and streamline the process in an online manner.

5.1 Related Work

This summary reviews prior research in individual detection, methods for multiple views, machine learning applications in tracking, and various optimization strategies in MC-MOT tracking, contrasting these with our neural graph based approach.

5.1.1 Object Detection

Before deep learning, traditional methods were predominant Felzenszwalb et al. (2010) and were utilized in MOTChallenge sequences (Leal-Taixé et al., 2015). However, with the introduction of the recent MOT challenges, deep learning-based detectors demonstrated improved accuracy (Ren et al., 2016; Liu et al., 2016a; Ge et al., 2021).

5.1.2 Single-Camera Tracking

In single-camera multi-object tracking (MOT), initial methods utilized Kalman filtering (Mittal et al., 2003). These faced drift errors, prompting a shift to tracking-by-detection techniques that trace objects across video frames using a global function (Andriluka et al., 2008). Merging Kalman filtering with these techniques enhanced tracking (Bewley et al., 2016; Zhang et al., 2021). Traditional strategies employed hand-crafted appearance features (Fleuret et al., 2008; Milan et al., 2014). Various algorithms are used for data association, from Conditional Random Fields to Network Flow Programming and binary quadratic optimization (Lafferty et al., 2001; Milan et al., 2013; Shitrit et al., 2014; Tang et al., 2015, 2017; Dehghan et al., 2015; Zamir et al., 2012; Henschel et al., 2016). Graph based approaches include K-Shortest Paths (KSP) and Successive Shortest Paths (SSP) (Berclaz et al., 2011; Pirsiavash et al., 2011). Emphasis in the field has moved towards end-to-end systems integrating detection and data association (Schulter et al., 2017; Zhang et al., 2020a; Bergmann et al., 2019), with a pivot towards neural graph-based methods in recent research (Brasó and Leal-Taixé, 2019)

5.1.3 Multi-Camera Tracking

Partially overlapping views can merge their data (Zhang et al., 2015b). For systems with fully overlapping views, the challenge is how to fuse the information (Kamal et al., 2013). In known camera layouts, spatial relationships are mapped to the world frame (Chen et al., 2011b; Zhang et al., 2015a). For unknown layouts, estimation is done via video, either by tracking identities or comparing camera pairs (Wan and Li, 2013; Cai and Medioni, 2014).

Plethora of research uses single-camera trajectories as input directly, evaluating associations across cameras without taking into account the trajectory errors from the tracker (Bredereck et al., 2012; Chen et al., 2011b, 2015; Zhang et al., 2015a; Wen et al., 2017; Lan et al., 2020; Nguyen et al., 2021). Formulations of the problem include Minimum Cost Bipartite Matching (Cai and Medioni, 2014) and Multi-Label Markov Random Fields (Lan et al., 2020).

Another area of research in multi-view tracking is to estimate accurate 2D or 3D occupancy maps to account for different perspectives and object occlusions. Methods range from foreground-background distinctions (Mittal and Davis, 2002), ground plane homographs (Khan and Shah, 2006), to integrating CNNs with other algorithms (Ong et al., 2020; You and Jiang, 2020a) for enhanced multi-camera detections.

5.2 Method

5.2.1 Problem Definition

In a system with C cameras, each camera detects objects across its video frames. This can be represented by a collection of object detections $\mathcal{O} = \{\mathcal{O}_c\}_{c=1}^{C}$, with $\mathcal{O}_c = \{o_i^c\}_{i=1}^{n^c}$, where n^c is the number of detections from the c-th camera. Each detection $o_i^c \in \mathcal{O}_c$ includes the bounding

box pixels, coordinates in the image frame, coordinates in world frame, and the timestamp, denoted as $o_i^c = (a_i^c, p_i^c, w_i^c, t_i^c)$ respectively.

For each camera, a trajectory is a series of detections that follow an object over time. It's defined for the c-th camera as $T_i^c = \{o_{i_j}^c\}_{j=1}^{n_i^c}$, with n_i^c being the count of detections that make up trajectory i for the c-th camera.

Two detections $o_{i_j}^{c_1}$ and $o_{i_k}^{c_2}$ of object i from cameras c_1 and c_2 are considered to be the same if their timestamps match, i.e. $t_{i_j}^{c_1} = t_{i_k}^{c_2}$. We define the union of two trajectories $T_i^{c_1}$ and $T_i^{c_2}$ where $c_1 \neq c_2$ for object i as :

$$T_i^{c_1} \cup T_i^{c_2} := \{o_{i_j}^{c_1} \in T_i^{c_1}\} \cup \{o_{i_k}^{c_2} \in T_i^{c_2} \mid \nexists o_{i_j}^{c_1} \in T_i^{c_1} \wedge t_{i_j}^{c_1} = t_{i_k}^{c_2}\} \tag{5.1}$$

Trajectory of object i across all cameras can be then represented as $T_i^* = \bigcup_{c=1}^{C} T_i^c$. The goal is to find the best set of trajectories across all cameras $\mathcal{T}_* = \{T_i^*\}_{i=1}^{M}$ where M is the total number of unique objects detected across all cameras.

5.2.2 Heterogeneous Graph Formulation

We define a heterogeneous graph (HG) as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \phi_n, \phi_e)$, where $\mathcal{V} = \bigcup_{c=1}^{C} \mathcal{V}_c$ is the unified set of vertices, composed of C distinct node sets $\mathcal{V}_c = \{i^c\}_{i=1}^{n^c}$, each corresponding to a different camera c, and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. Each node $i \in \mathcal{V}$ has an associated feature vector $\mathbf{x}_i$ derived from detection $o_{k_i} \in \mathcal{O}$ of object k and each edge $(i, j) \in \mathcal{E}$ has a feature vector $\mathbf{x}_{ij}$. Nodes are associated with a node type mapping function $\phi_n : \mathcal{V} \to \mathcal{C}_n$, where $\mathcal{C}_n = \{r_n^c\}_{c=1}^{C}$ is the node type set such that $\phi_n(i^c) = r_n^c$ for $i^c \in \mathcal{V}_c$.

The edge set is defined as $\mathcal{E} = \mathcal{E}^s \cup \mathcal{E}^t \cup \mathcal{E}^v$, representing spatial, temporal, and view dimensions respectively. Edges are associated with a type mapping function $\phi_e : \mathcal{E} \to \mathcal{C}_e$, where $\mathcal{C}_e = \{r_e^s, r_e^t, r_e^v\}$ denotes the edge type set.

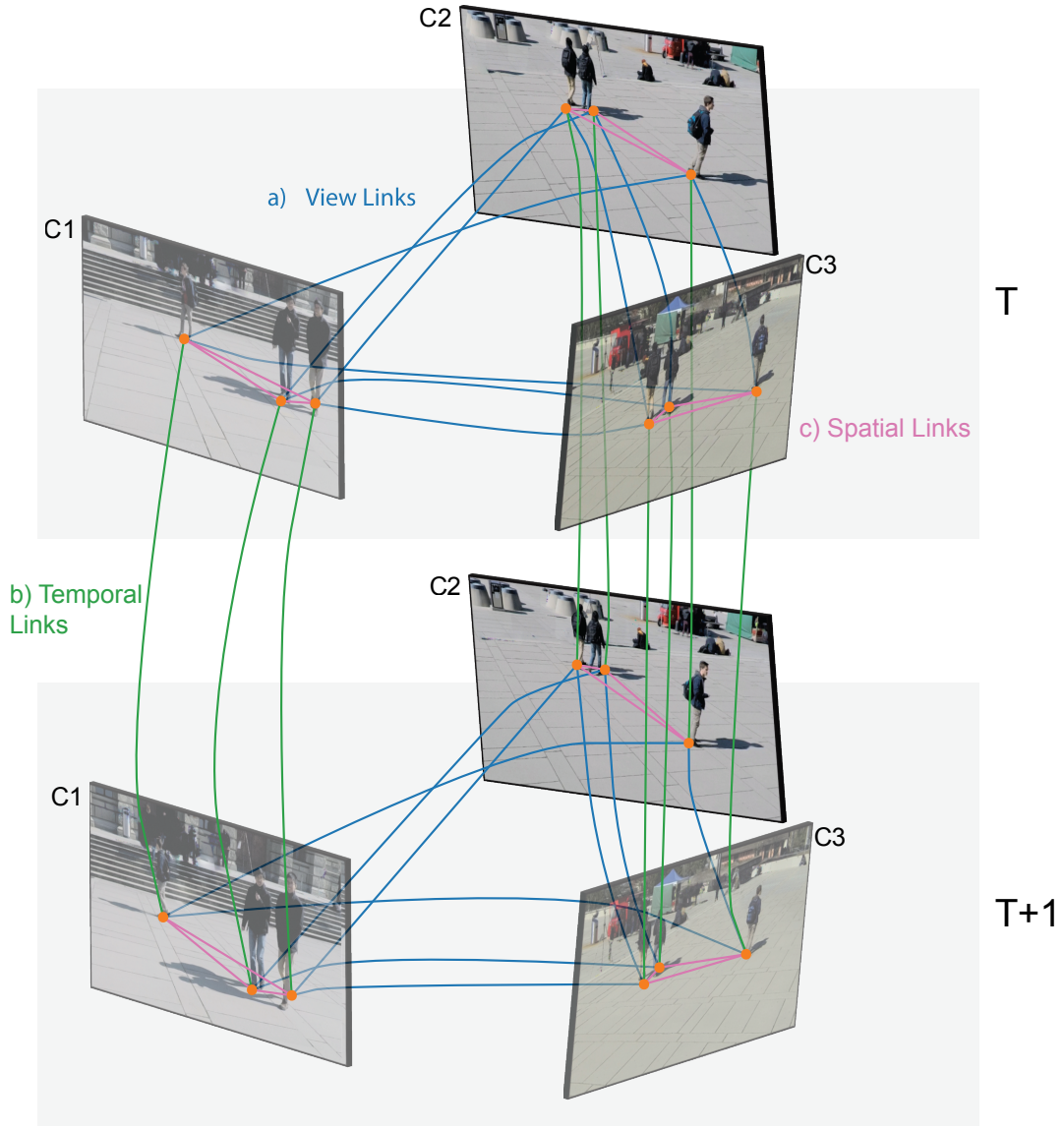


Figure 5.1: **Multi-View Graph Construction.** **a) View Links** connect identical detections across views at the same timepoint. **b) Temporal Links** connect detections in the same view at different timepoints, forming a fully connected graph (only ground truth connections shown for clarity). **c) Spatial Links** connect detections to nearby ones in the same view at the same timepoint.

Temporal Linking

Within the defined heterogeneous graph, we define the temporal edge subset $\mathscr{E}^t$ as:

$$\mathscr{E}^t := \{(i,j) \mid (i,j) \in \mathscr{V} \times \mathscr{V} \wedge j \neq i \wedge t_i < t_j \wedge \phi_n(i) = \phi_n(j)\} \tag{5.2}$$

Here $\phi_e(i,j) = r_e^t$ for every $(i,j) \in \mathscr{E}^t$. The task can now be translated into a graph-based problem, where the aim is to partition the nodes within each $\mathscr{V}_c$ into distinct clusters, where each cluster represents a trajectory. We represent it as an edge classification task, where the labels are defined as set of edges that form a trajectory. Formally, we aim label the edges using a function $f_t^{clf} : \mathscr{E}^t \rightarrow \{0,1\}$ defined as:

$$f_t^{clf}\left(i,j\right) := \begin{cases} 1, & \text{if } (i,j) \in T_k^* \wedge \exists T_k^* \in \mathscr{T}^* \wedge \phi_n(i) = \phi_n(j) \\ 0, & \text{else} \end{cases} \tag{5.3}$$

The system assumes a synchronized and a calibrated setup. The representation of this problem includes the following components: a) **Node Feature Matrix** $\mathbf{X} \in \mathbb{R}^{|\mathscr{V}| \times F_n}$ containing specific features per node of length F_n. b) **Adjacency Matrix** $\mathbf{A}^t \in \mathbb{R}^{|\mathscr{V}| \times |\mathscr{V}|}$ representing the connectivity of the graph c) **Edge Feature Matrix** $\mathbf{Z}^t \in \mathbb{R}^{|\mathscr{E}| \times F_e}$ representing features of edges of length F_e. d) **Prediction Target** $\hat{\mathbf{Y}}^t \in \mathbb{R}_{[0,1]}^{|\mathscr{E}^t| \times 1}$ signifying the likelihood of object association.

To parallelize temporal message passing across multiple graphs representing different cameras, we represent the setup for C cameras as:

$$\mathbf{A}^t = \begin{bmatrix} \mathbf{A}_1^t & & \\ & \ddots & \\ & & \mathbf{A}_C^t \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_C \end{bmatrix}, \quad \mathbf{Z}^t = \begin{bmatrix} \mathbf{Z}_1^t \\ \vdots \\ \mathbf{Z}_C^t \end{bmatrix}, \quad \mathbf{Y}^t = \begin{bmatrix} \mathbf{Y}_1^t \\ \vdots \\ \mathbf{Y}_C^t \end{bmatrix} \tag{5.4}$$

Here, $\mathbf{A}_c^t$, $\mathbf{X}_c$, $\mathbf{Z}_c^t$, and $\mathbf{Y}_c^t$ correspond to the adjacency matrix, node features, edge features, and target features for camera c, respectively. As $\mathbf{A}^t$ is sparse, only non-zero entries can be stored, creating a sparse connectivity matrix $\mathbf{E} \in \mathbb{N}^{2 \times |\mathscr{E}^t|}$. The application of temporal parallelization technique offers: 1. **Consistency**: Individual graphs are isolated, allowing temporal message passing scheme to function unmodified. 2. **Efficiency**: The method avoids unnecessary computational or memory overhead without needing node or edge feature padding. We only additionally store node vector $\mathbf{m}$ containing node mappings $\phi_n(i)$ for every $i \in \mathscr{V}$ to easily index components of individual cameras.

View (Inter-Camera) Linking

The inter-camera context captures relationships between detections of the same object across different cameras. This connection is done by constructing edges that link corresponding

detections.

The coordinates of each detection in the world coordinates for camera c are denoted as $\mathbf{S}_c \in \mathbb{R}^{|\mathcal{V}_c| \times 3}$. Given C cameras and a set of detections for each camera, we construct the inter-camera edges using a k-nearest neighbors (k-NN) graph, where $k = x \cdot (C-1)$ and x is a hyperparameter. The k-NN graph can be constructed for each camera by:

$$\mathbf{K}_c := \text{kNN}(\mathbf{S}_c, \mathbf{S}_{-c}, k) \tag{5.5}$$

where $\mathbf{S}_{-c}$ represents the features from all cameras except c, $\mathbf{K}_c \in \mathbb{N}^{|\mathcal{V}_c| \times k}$, represents the k-NN indices matrix for camera c, where each row contains the indices of the k nearest detections from other cameras to the corresponding detection in camera c.

Now, the inter-camera edges $\mathcal{E}^v$ can be defined as:

$$\mathcal{E}^v := \{(i,j) \mid (i,j) \in \mathcal{V} \times \mathcal{V} \wedge j \in \mathbf{K}_{c,i} \wedge t_i = t_j \wedge \phi_n(i) \neq \phi_n(j)\} \tag{5.6}$$

Here $\mathbf{K}_{c,i}$ indexes k nearest detections for the i-th detection in camera c, and $\phi_e(i,j) = r_e^v$ for every $(i,j) \in \mathcal{E}^v$. We represent the inter-camera edges as the edge connectivity matrix $\mathbf{E}^v \in \mathbb{N}^{2 \times |\mathcal{E}^v|}$ containing pairs of linked detection indices across different cameras. and the corresponding edge feature matrix can be computed based on the features of connected detections. The goal is to label the edges using a function $f_v^{clf} : \mathcal{E}^v \rightarrow \{0,1\}$ with the value being 1 if there is a connection between the object across cameras at a specific timepoint, and 0 otherwise. Formally,

$$f_v^{clf}\left(i,j\right) := \begin{cases} 1, & \text{if } (i,j) \in T_k^* \wedge \exists T_k^* \in \mathcal{T}^* \wedge \phi_n(i) \neq \phi_n(j) \\ 0, & \text{else} \end{cases} \tag{5.7}$$

Spatial Linking

The spatial linking within each camera view takes into account the spatial relationships between different detections captured by the same camera.
For each camera c, we consider pairs of nodes $\{\mathbf{s}_i, \mathbf{s}_j\} \in \mathbf{S}_c$, where $i, j \in \mathcal{V}_c \times \mathcal{V}_c$, $i \neq j$, and $t_i^c = t_j^c$. The Euclidean distance d_{ij} between detections i and j within camera c at time t is computed as:

$$d_{ij} = |\mathbf{s}_i - \mathbf{s}_j|_2, \tag{5.8}$$

where $|\cdot|_2$ denotes the L_2 norm. If this computed distance d_{ij} is less than a spatial threshold γ (hyperparameter), we establish a link between detections i and j such that $\phi_e(i,j) = r_e^s$. Formally, the set of spatially linked detections within camera c can be defined as:

$$\mathcal{E}^s := \{(i,j) \mid (i,j) \in \mathcal{V} \times \mathcal{V} \wedge i \neq j \wedge t_i = t_j \wedge d_{ij} < \gamma \wedge \phi_n(i) = \phi_n(j)\}, \tag{5.9}$$

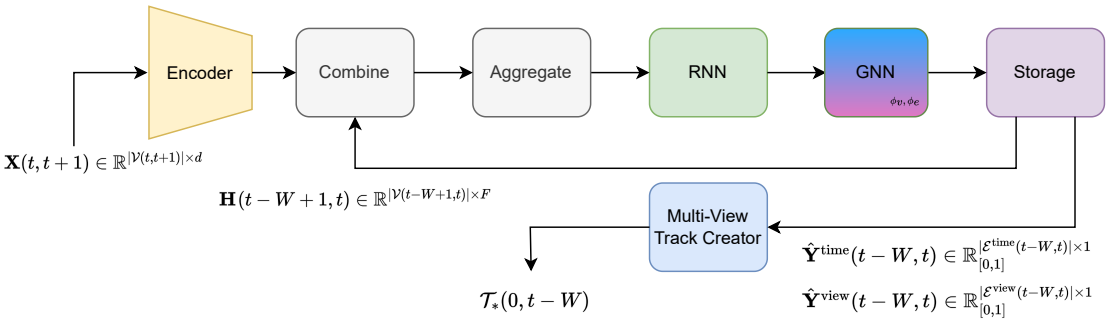


Figure 5.2: **Simplified schema of the proposed approach.** We update the graph at every timestamp, feeding its encoded dynamics into the multi-view track creator. The framework's output is a matrix that determines active temporal and view (inter-camera) edges. Track predictions are made in the form of a directed graph with soft temporal edge weights, indicating potential links between consecutive frames. Active inter-camera edges are used to merge tracks across cameras.

We represent the spatially linked detection indices within each camera view as $\mathbf{E}^s \in \mathbb{N}^{2 \times |\mathcal{E}^s|}$, containing pairs of spatially-linked detection indices.

5.2.3 Rolling Window Formulation

We use a rolling window of size W for our graph to keep it online, always covering a span from $t-W$ to t at any any given point t. Formally, $\mathcal{G}(t-W, t) = (\mathcal{V}(t-W, t), \mathcal{E}(t-W, t), \phi_n, \phi_e)$. Moving forward, for nodes active within the rolling window, we define a node set $\mathcal{V}(t-W, t)$, such that $\mathcal{V}(t-W, t) := \{i \mid t-W \leq t_i \leq t\}$. Similarly, the set of edges within this timeframe is defined as $\mathcal{E}(t-W, t) := \{(i, j) \mid t-W \leq t_i, t_j \leq t\}$. Node set $\mathcal{V}(t-W, t)$ and edge set $\mathcal{E}(t-W, t)$ have associated node feature matrix $\mathbf{X}(t-W, t) \in \mathbb{R}^{|\mathcal{V}(t-W,t)| \times F_n}$ and edge feature matrix $\mathbf{Z}(t-W, t) \in \mathbb{R}^{|\mathcal{E}(t-W,t)| \times F_e}$, respectively.

5.2.4 Message Passing

Initial Representation

We encode each detection from the set of detections and represent it as the initial node feature:

$$\mathbf{x}_i = \left[x, y, z, x_1, y_1, w, h\right] \in \mathbb{R}^7 \tag{5.10}$$

$$\mathbf{h}_i^{(0)} = \text{MLP}^{enc}(\mathbf{x}_i) \tag{5.11}$$

where x, y, z represent the 3D object position, known in world coordinates due to a calibrated system. The dimensions of the 2D bounding box are denoted by x_1, y_1, w, h. Similarly, the

initial embedding of a temporal edge between nodes i and j is initialized as:

$$\mathbf{h}_{ij}^{(0)} = \mathbf{h}_{i}^{(0)} - \mathbf{h}_{j}^{(0)} \tag{5.12}$$

Temporal Message Passing

The embedding of the edge between nodes i and j is updated based on its previous embedding and the previous embeddings using edge-specific MLP:

$$\mathbf{h}_{ij}^{(l)} = \text{MLP}_{\phi_e(i,j)}\left(\left[\mathbf{h}_{ij}^{(l-1)}, \mathbf{h}_{j}^{(l-1)}, \mathbf{h}_{i}^{(l-1)}\right]\right), \quad \text{s.t. } \phi_e(i,j) = r_e^t \tag{5.13}$$

Node i is updated by its neighboring edges using a learnable update function:

$$\mathbf{h}_i' = \text{RNN}\left(\mathbf{h}_i, \quad \mathbf{h}_i + \sum_{j \in \mathcal{N}_{r_e^v}(i)} \mathbf{h}_{ij}\right) \tag{5.14}$$

where $\mathcal{N}_{r_e^v}(i) = \left\{j \mid (i,j) \in \mathcal{E} \wedge \phi_e(i,j) = r_e^v\right\}$ represents neighborhood of node i with temporal links. Here RNN implements a learnable recurrent module such as long short-term memory network (LSTM) (Hochreiter and Schmidhuber, 1997) or gated recurrent unit (GRU) (Cho et al., 2014).

Spatial and View Message Passing

We follow Veličković et al. (2018) and and update the embedding of node i by its neighboring nodes connected by a view or spatial edge at l-th layer as:

$$\mathbf{h}_i^{(l)} = \mathbf{W}^{(l)}\mathbf{h}_i^{(l-1)} + \sum_{j \in \mathcal{N}_{r_e^{vs}}(i)} \alpha_{ij}^{(l-1)} \mathbf{W}^{(l)} \mathbf{h}_j^{(l-1)} \tag{5.15}$$

where $\mathbf{W} \in \mathbb{R}^{d^{(l)} \times d^{(l-1)}}$ is the weight matrix and $\mathcal{N}_{r_e^{vs}}(i) = \{j \mid (i,j) \in \mathcal{E} \wedge \phi_e(i,j) \in r_e^{vs} = \{r_e^v, r_e^s\}\}$ represents neighborhood of node i with spatial and view links.

The relation-specific attention coefficient is computed as:

$$\alpha_{ij}^{(l)} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top \mathbf{W}_{\phi_e(i,j)}\left[\mathbf{h}_i^{(l)} - \mathbf{h}_j^{(l)}\right]\right)\right)}{\sum_{k \in \mathcal{N}(i)} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top \mathbf{W}_{\phi_e(i,k)}\left[\mathbf{h}_i^{(l)} - \mathbf{h}_k^{(l)}\right]\right)\right)} \tag{5.16}$$

where $\mathcal{N}(\cdot)$ represents the neighborhood of a node, and $\mathbf{W}_{\phi_e(i,j)}$ is the edge-specific transformation matrix for nodes i and j where $\phi_e(i,j) \in \{r_e^v, r_e^s\}$.

5.2.5 Edge Classification

For temporal edge classification, the output is derived using a simple one layer MLP with temporal edge embedding. In the case of view edge classification, we use MLP on the concatenated node embeddings.

$$\hat{y}^t_{ij} = \sigma\left(\mathbf{W}^{clf}_{\phi_e(i,j)}\mathbf{h}^{(l)}_{ij} + \mathbf{b}^{clf}_{\phi_e(i,j)}\right) \quad \text{s.t. } \phi_e(i,j) = r^t_e \tag{5.17a}$$

$$\hat{y}^v_{ij} = \text{MLP}^{clf}_{\phi_e(i,j)}\left(\left[\mathbf{h}^{(l)}_i, \mathbf{h}^{(l)}_j\right]\right) \quad \text{s.t. } \phi_e(i,j) = r^v_e \tag{5.17b}$$

5.2.6 Loss Definition

We use a binary cross-entropy loss (BCE) for each temporal edge as the temporal loss, with the ground truth indicating if the edge is part of the trajectory. For the view loss, we consider if the detections in different views are identical using BCE. The cumulative training loss in a multi-view setting is then:

$$\mathscr{L} = \lambda_t \mathscr{L}_{\text{temporal}} + \lambda_v \mathscr{L}_{\text{view}} \tag{5.18}$$

5.3 Experiments

This section provides a overview of our evaluation setting, the datasets, the metrics, and the nature of the input detections used to construct the graph. Following this, Section 5.3.1 presents our online tracking results, and offers a comparison with current state-of-the-art benchmarks. In Section 5.3.2, we talk the rationale behind our design choices and explain the advantages of our contributions through ablation studies.

Datasets

We evaluate our approach on Wildtrack (Chavdarova et al., 2018) and PETS2009 Ferryman and Shahrokni (2009). We follow the official train/test splits unless stated otherwise. Wildtrack has 7 overlapping cameras with last 400 frames annotated, capturing 313 pedestrians in total. PETS2009 includes three sequences varying in density. These sequences are known have poor quality and camera calibration problems. We evaluate our method on the first sequence (S2-L1).

Evaluation Metrics

Based on Bernardin and Stiefelhagen (2008), we utilize these metrics: a) MOTA: Evaluates tracking accuracy, accounting for false negatives (FN), false positives (FP), and identity switches (IDs). b) MOTP: Assesses precision by penalizing differences between true positives and actual objects. c) IDF1: Measures alignment between predicted and actual trajectories using the F1 score. d) MT: Tally of objects tracked for at least 80% of their duration. e) ML:

Counts objects missing for 80% or more of their lifespan. f) We also record the number of ID switches.

Detections

We use the detections provided by Zhou et al. (2020) to construct the heterogeneous graph in our experiments for Wildtrack.

5.3.1 Results

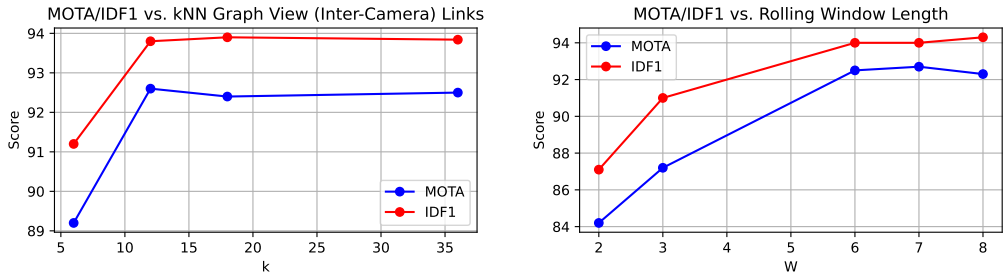
Table 5.1 details the performance metrics of multiple tracking methods on the Wildtrack dataset. LMGP (Nguyen et al., 2021), an offline method, achieves the highest MOTA at 97.1%, IDF1 at 98.2%, MT at 97.6%, and the lowest ML at 1.3%. Our online method registers a MOTA of 92.7, IDF1 of 94.1, ML of 3.4, and MT of 96.2, placing it second on this benchmark. The performance on the PETS2009 dataset, specifically the S2-L1 sequence, is shown in Table [reference missing, presumed 5.2]. The LMGP method achieves a MOTA of 97.8, MOTP of 82.4, MT of 100, and has 2 identity switches. In comparison, our approach achieves a MOTA of 96.2, MOTP of 80.3, MT of 100%, and registers 3 identity switches. It follows MLMRF (Lan et al., 2020) in terms of overall performance.

Method	MOTA ↑	IDF1 ↑	ML ↓	MT↑
GLMB-YOLOv3 (Ong et al., 2020)	69.7	74.3	21.6	79.5
GLMB-DO (Ong et al., 2020)	70.1	72.5	22.8	93.6
DMCT Stack (You and Jiang, 2020b)	74.6	81.9	4.9	65.9
DO+KSP+ptrack (Chavdarova et al., 2018)	72.2	78.4	-	-
MVFlow (Engilberge et al., 2023)	91.3	93.5	-	-
LMGP (Nguyen et al., 2021)	97.1	98.2	1.3	97.6
Ours	92.7	94.1	3.4	96.2

Table 5.1: Performance on the Wildtrack compared to SOTA baselines. The best result is shown in red and the second best in blue.

Sequence	Method	MOTA ↑	MOTP ↑	MT ↑	ML ↓	IDs ↓
S2-L1	HJMV (Hofmann et al., 2013)	91.7	79.4	94.7	0.0	45
	MVFLow (Engilberge et al., 2023)	93.3	64.0	-	0.0	-
	STVH (Wen et al., 2017)	95.1	79.8	100.0	0.0	13
	MLMRF (Lan et al., 2020)	96.8	79.9	100.0	0.0	2
	LMGP	97.8	82.4	100.0	0.0	2
	Ours	96.2	80.3	100.0	0.0	3

Table 5.2: Performance on the PETS2009 compared to SOTA baselines. The best result is shown in red and the second best in blue.



(a) Performance w.r.t. kNN View Link Graph on Wildtrack (C=7). We use $k = x \cdot (C-1)$ with $x \in \{1,2,3,6\}$.

(b) Performance w.r.t. Rolling Window Length on Wildtrack.

Figure 5.3: Performance w.r.t. View Links and Rolling Window Length

5.3.2 Ablation

We conducted an ablation study on the Wildtrack dataset to assess the significance of individual components in our multi-camera tracking framework, as shown in Table 5.3. Eliminating spatial linking (with $\gamma = 0$) results in a MOTA of 90.8% and IDF1 of 93.4%. Without view linking ($k = 0$), the performance declines more notably, with MOTA dropping to 87.7% and IDF1 to 89.8%. While both components contribute to the overall performance, the absence of view linking results in a more significant performance drop. We conclude that view linking plays an important role and contributes to the effectiveness of our tracking approach.

Dataset	Method	MOTA ↑	IDF1 ↑	ML ↓	MT↑
Wildtrack	w/o spatial linking ($\gamma = 0$)	90.8	93.4	5.3	94.9
	w/o view linking ($k = 0$)	87.7	89.8	12.4	93.6
	Ours (all components)	92.7	94.1	3.4	96.2

Table 5.3: Ablation of heterogeneous graph formulation.

5.3.3 Discussion

In testing, our system achieves competitive performance while being able to perform online. The top-performing LMGP is an *offline* approach that relies on single-camera tracklets, post-processed without accounting for errors stemming from imprecise single-camera trajectories. Our approach directly addresses this challenge, suggesting that it can potentially surpass current top score with further refinement, which we'll try address in future research. Notably, our method also offers real-time potential for many useful applications, including human robot interaction and autonomous driving. These advancements can also support industries relying on precise motion analysis, such as sports analytics and filmmaking. However, while the advancements offer potential benefits, they also raise concerns related to privacy and misuse, requiring strong regulatory frameworks and ethical considerations.

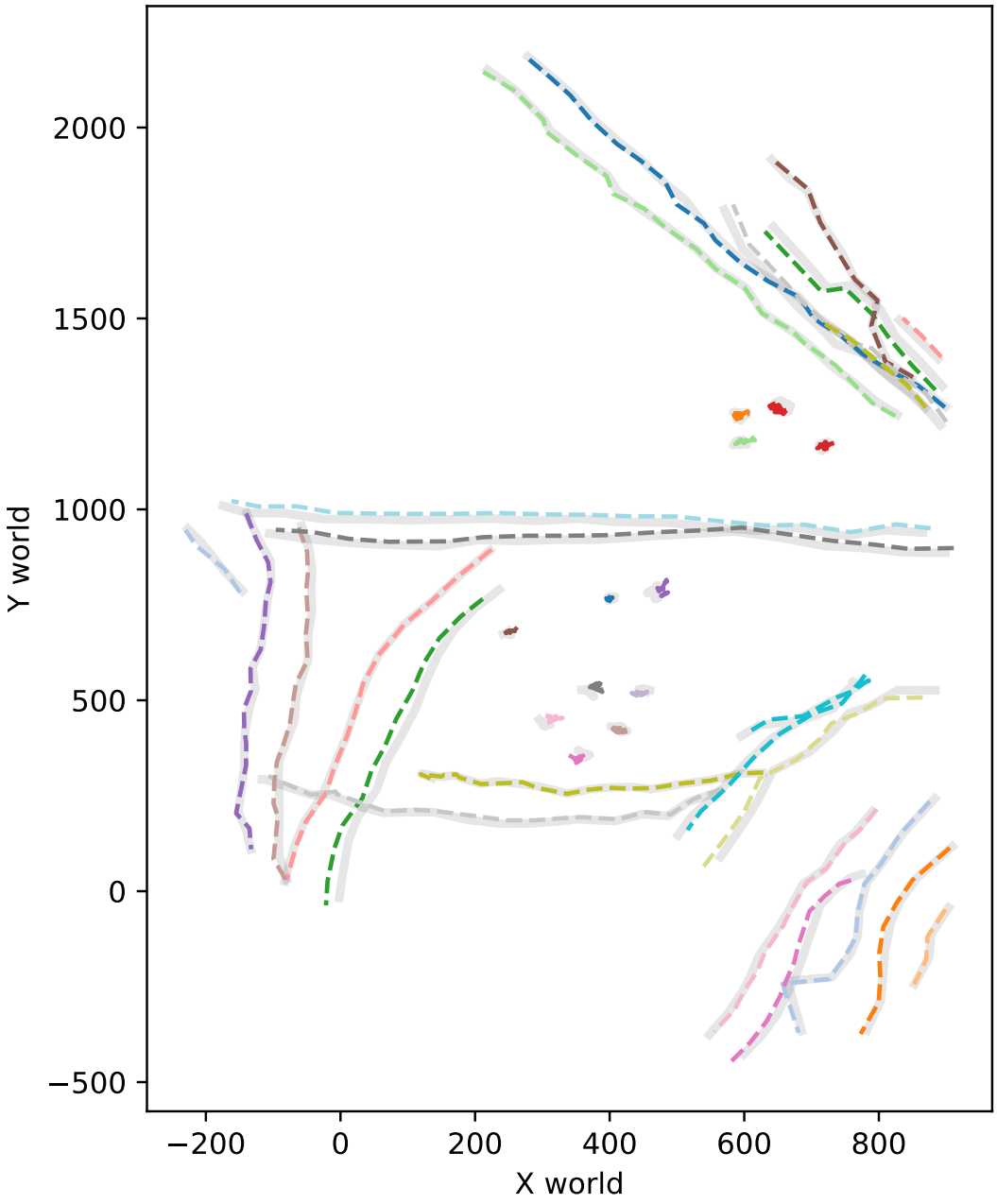


Figure 5.4: **Qualitative results of the proposed approach on sample Wildtrack sequence.** Ground truth is in grey, while dotted lines of various colors represent predicted trajectories for different pedestrians.

6 Conclusion

We have proposed a framework for multi-camera multi-object tracking (MC-MOT) in line with the move towards end-to-end learning, converging the process of trajectory creation. We have introduced a neural association mechanism that works in a multi-view setting, making it easier to link object identities across different frames and cameras. This reduces common problems like identity switches and track losses. In testing, our system achieves competitive performance while being able to perform online. The integrated nature of our approach suggests that it can potentially surpass current state-of-the-art results (SOTA) with further refinement, as SOTA methods rely on single camera tracking results. We also present a benchmark dataset, along with the tool we used for its creation, to evaluate the performance of MC-MOT frameworks. For traditional multi-camera tracking setups, we present a simple approach for using appearance in a multi-view setting. One interesting area of research would be to employ a model similar to NRI (Kipf et al., 2018) to adjust the detections, but we leave this as future work. We are still on the journey to a fully end-to-end MC-MOT solution, but our current progress indicates a promising direction.

A Code Snippets

Listing A.1: Definitions in C++

```
struct AP_Pose{
  double matrix[12];  // 3x4 column-major matrix, i.e. Eigen::
    AffineCompact3d
};

// OpenCV camera calibration model.
struct AP_Calibration
{
  double fx, fy, cx, cy;
  double k1, k2, p1, p2;
  AP_Pose pose;
};

struct AP_Detection
{
  float boundingBox[4];  // [x0, y0, x1, y1]
  // Cuboid local coordinate system is as follows:
  // z = 0 is on the ground
  // x,y,z = 0 is the center of the bottom facet
  // [- dimensions[0]/2, dimensions[0]/2] is the range of the cuboid
    along axis x
  // [- dimensions[1]/2, dimensions[1]/2] is the range of the cuboid
    along axis y
  // [0, dimensions[2]] is the range of the cuboid along axis z

  // the cuboidToWorld matrix convert from these local coordinates to
    world coordinates.
  AP_Pose cuboidToWorld;

  float dimensions[3];// length, width, height, in this order.
};
```

Listing A.2: Crop Extraction in C++

```
cv::Mat getCrop(const AP_Detection& detection, const AP_Calibration&
  calib, const cv::Mat& frame){
  Eigen::AffineCompact3d cuboidToWorld;
  cuboidToWorld.matrix() = Eigen::Map<const Eigen::Matrix<double, 3, 4>>(
    detection.cuboidToWorld.matrix);
  Eigen::AffineCompact3d poseMatrix;
  poseMatrix.matrix() = Eigen::Map<const Eigen::Matrix<double, 3, 4>>(
    calib.pose.matrix);
  Eigen::Vector3d cuboidWorld = cuboidToWorld.translation();
  Eigen::Vector3d camWorld = -poseMatrix.linear().transpose() *
    poseMatrix.translation();
  Eigen::Vector3d cameraCuboid = camWorld - cuboidWorld;
  cameraCuboid[2] = 0; // set z to 0
  Eigen::Vector3d vNorm = 0.3 * cameraCuboid.cross(Eigen::Vector3d(0, 0,
    1)).normalized(); // scale by 30cm

  double zMargin = 0.1; //use 10cm Z margin
  Eigen::Vector3d leftPointBottomWorld = vNorm + cuboidWorld + Eigen::
    Vector3d(0, 0, -zMargin);
  Eigen::Vector3d leftPointTopWorld = vNorm + cuboidWorld + Eigen::
    Vector3d(0, 0, zMargin + detection.dimensions[0]);
  Eigen::Vector3d rightPointBottomWorld = -vNorm + cuboidWorld + Eigen::
    Vector3d(0, 0, -zMargin);
  Eigen::Vector3d rightPointTopWorld = -vNorm + cuboidWorld + Eigen::
    Vector3d(0, 0, zMargin + detection.dimensions[0]);
  Eigen::Matrix<double, 3, 4> pointsMatrix;
  pointsMatrix << leftPointBottomWorld, rightPointBottomWorld,
    rightPointTopWorld, leftPointTopWorld;

  cv::Mat cameraMatrix = (cv::Mat_<double>(3, 3) << calib.fx, 0, calib.cx
    , 0, calib.fy, calib.cy, 0, 0, 1);
  Eigen::Matrix<double, 3, 3> cameraMatrixEigen;
  cv::cv2eigen(cameraMatrix, cameraMatrixEigen);

  Eigen::Matrix<double, 3, 4> pointsCam = poseMatrix*pointsMatrix;
  if ((pointsCam.row(2).array() <= 0).any()){return cv::Mat();}
  Eigen::Matrix<double, 3, 4> pointsCam2d = cameraMatrixEigen*pointsCam;
  Eigen::Matrix<double, 2, 4> points2d = pointsCam2d.topRows(2).array() /
     pointsCam2d.row(2).array().replicate(2,1);

  cv::Mat srcPoints = (cv::Mat_<float>(4, 2) << points2d(0,0), points2d
    (1,0), points2d(0,1), points2d(1,1), points2d(0,2), points2d(1,2),
    points2d(0,3), points2d(1,3));
  cv::Mat dstPoints = (cv::Mat_<float>(4, 2) << 0, cropHeight-1,
    cropWidth-1, cropHeight-1,  cropWidth-1, 0, 0, 0);
  cv::Mat M = cv::getPerspectiveTransform(srcPoints, dstPoints);

  cv::Mat warped;
  cv::warpPerspective(frame, warped, M, cv::Size(cropWidth, cropHeight));
  return warped;}
```

Listing A.3: Appearance Feature Extraction of Detections in C++

```
void extractFeatures(const std::vector<cv::Mat>& crops, const std::vector
  <int>& indices, AP_Descriptor** descriptors){
  torch::Tensor image_tensors = cvMatsToTensor(crops);  // Convert the
    input image to a LibTorch tensor
  if (torch::cuda::is_available()) { image_tensors = image_tensors.to(at
    ::kCUDA); } // Move the tensor to the GPU
  torch::Tensor output_tensors = torch_module.forward({image_tensors}).
    toTensor();  // Forward pass with TorchScript model
  for (int i = 0; i < output_tensors.size(0); ++i)
  {
      // Extract the descriptor for the current image
      torch::Tensor output_tensor = output_tensors[i];
      AP_Descriptor descriptor = tensorToDescriptor(output_tensor);
      int descIndex = indices[i];
      descriptors[descIndex] = new AP_Descriptor(descriptor);
  }
}
```

Listing A.4: Feature Comparison using Cosine Similarity in C++

```
static double compareDescriptors(const AP_Descriptor* a, const
  AP_Descriptor* b){
  if (!a || !b) { return 0; }
  return double(a->vector.dot(b->vector) / (a->vector.norm() * b->vector.
    norm()));
}
```

Bibliography

Alameda-Pineda, X., Staiano, J., Subramanian, R., Batrinca, L. M., Ricci, E., Lepri, B., Lanz, O., and Sebe, N. (2015). SALSA: A novel dataset for multimodal group behavior analysis. *CoRR*, abs/1506.06882.

Ali, K., Pilet, J. V., and Becker, C. J. (2023). Video-based tracking systems and methods. United States, Status: Pending, Current Assignee: Invision Ai Inc, Application filed: 2021-07-05, Publication: 2023-03-02.

Andriluka, M., Roth, S., and Schiele, B. (2008). People-tracking-by-detection and people-detection-by-tracking. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V. F., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gülçehre, Ç., Song, H. F., Ballard, A. J., Gilmer, J., Dahl, G. E., Vaswani, A., Allen, K. R., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M. M., Vinyals, O., Li, Y., and Pascanu, R. (2018). Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261.

Berclaz, J., Fleuret, F., Turetken, E., and Fua, P. (2011). Multiple object tracking using k-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1806–1819.

Bergmann, P., Meinhardt, T., and Leal-Taixé, L. (2019). Tracking without bells and whistles. *CoRR*, abs/1903.05625.

Bernardin, K. and Stiefelhagen, R. (2008). Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008.

Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. (2016). Simple online and realtime tracking. *CoRR*, abs/1602.00763.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Brasó, G. and Leal-Taixé, L. (2019). Learning a neural solver for multiple object tracking. *CoRR*, abs/1912.07515.

Bredereck, M., Jiang, X., Körner, M., and Denzler, J. (2012). Data association for multi-object tracking-by-detection in multi-camera networks. In *2012 Sixth International Conference on Distributed Smart Cameras (ICDSC)*, pages 1–6.

Cai, Y. and Medioni, G. (2014). Exploring context information for inter-camera multiple target tracking. In *IEEE Winter Conference on Applications of Computer Vision*, pages 761–768.

Cai, Z., Zhang, J., Ren, D., Yu, C., Zhao, H., Yi, S., Yeo, C. K., and Loy, C. C. (2020). Messytable: Instance association in multiple camera views. *CoRR*, abs/2007.14878.

Chavdarova, T., Baque, P., Bouquet, S., Maksai, A., Jose, C., Bagautdinov, T., Lettry, L., Fua, P., Gool, L. V., and Fleuret, F. (2018). Wildtrack: A multi-camera hd dataset for dense unscripted pedestrian detection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5030–5039, Los Alamitos, CA, USA. IEEE Computer Society.

Chavdarova, T. and Fleuret, F. (2017). Deep multi-camera people detection. *CoRR*, abs/1702.04593.

Chen, K.-W., Lai, C.-C., Lee, P.-J., Chen, C.-S., and Hung, Y.-P. (2011a). Adaptive learning for target tracking and true linking discovering across multiple non-overlapping cameras. *IEEE Transactions on Multimedia*, 13:625–638.

Chen, K.-W., Lai, C.-C., Lee, P.-J., Chen, C.-S., and Hung, Y.-P. (2011b). Adaptive learning for target tracking and true linking discovering across multiple non-overlapping cameras. *IEEE Transactions on Multimedia*, 13(4):625–638.

Chen, W., Cao, L., Chen, X., and Huang, K. (2016). An equalised global graphical model-based approach for multi-camera object tracking.

Chen, X., An, L., and Bhanu, B. (2015). Multitarget tracking in nonoverlapping cameras using a reference set. *IEEE Sensors Journal*, 15(5):2692–2704.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Dehghan, A., Assari, S. M., and Shah, M. (2015). Gmmcp tracker: Globally optimal generalized maximum multi clique problem for multiple object tracking. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4091–4099.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Diao, H., Zhang, Y., Ma, L., and Lu, H. (2021). Similarity reasoning and filtration for image-text matching. *CoRR*, abs/2101.01368.

Do, T., Tran, T., Reid, I. D., Kumar, B. G. V., Hoang, T., and Carneiro, G. (2019). A theoretically sound upper bound on the triplet loss for improving the efficiency of deep distance metric learning. *CoRR*, abs/1904.08720.

D'Orazio, T., Leo, M., Mosca, N., Spagnolo, P., and Mazzeo, P. (2009). A semi-automatic system for ground truth generation of soccer video sequences. In *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 559–564.

Engilberge, M., Liu, W., and Fua, P. (2023). Multi-view tracking using weakly supervised human motion prediction. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1582–1592.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645.

Ferryman, J. and Shahrokni, A. (2009). Pets2009: Dataset and challenge. In *2009 Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 1–6.

Fleuret, F., Berclaz, J., Lengagne, R., and Fua, P. (2008). Multicamera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):267–282.

Ge, Z., Liu, S., Wang, F., Li, Z., and Sun, J. (2021). YOLOX: exceeding YOLO series in 2021. *CoRR*, abs/2107.08430.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gilbert, A. and Bowden, R. (2006). Tracking objects across cameras by incrementally learning inter-camera colour calibration and patterns of activity. In *European Conference on Computer Vision*.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. *CoRR*, abs/1704.01212.

Goodfellow, I. J., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge, MA, USA. http://www.deeplearningbook.org.

Guennebaud, G., Jacob, B., et al. (2010). Eigen v3. http://eigen.tuxfamily.org.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Henschel, R., Leal-Taixé, L., Rosenhahn, B., and Schindler, K. (2016). Tracking with multi-level features. *CoRR*, abs/1607.07304.

Hermans, A., Beyer, L., and Leibe, B. (2017). In defense of the triplet loss for person re-identification. *CoRR*, abs/1703.07737.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Hofmann, M., Wolf, D., and Rigoll, G. (2013). Hypergraphs for joint multi-view reconstruction and multi-object tracking. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3650–3657.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Jun, H., Ko, B., Kim, Y., Kim, I., and Kim, J. (2019). Combination of multiple global descriptors for image retrieval. *CoRR*, abs/1903.10663.

Kalayeh, M. M., Basaran, E., Gokmen, M., Kamasak, M. E., and Shah, M. (2018). Human semantic parsing for person re-identification. *CoRR*, abs/1804.00216.

Kamal, A. T., Farrell, J. A., and Roy-Chowdhury, A. K. (2013). Information consensus for distributed multi-target tracking. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2403–2410.

Khan, S. M. and Shah, M. (2006). A multiview approach to tracking people in crowded scenes using a planar homography constraint. In *European Conference on Computer Vision*.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. (2018). Neural relational inference for interacting systems.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA. Curran Associates Inc.

Kuo, C.-H., Huang, C., and Nevatia, R. (2010a). Inter-camera association of multi-target tracks by on-line learned appearance affinity models. In *Proceedings of the 11th European Conference on Computer Vision: Part I*, ECCV'10, page 383–396, Berlin, Heidelberg. Springer-Verlag.

Kuo, C.-H., Huang, C., and Nevatia, R. (2010b). Inter-camera association of multi-target tracks by on-line learned appearance affinity models. In *European Conference on Computer Vision*.

Lafferty, J., Mccallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. pages 282–289.

Lan, L., Wang, X., Hua, G., Huang, T. S., and Tao, D. (2020). Semi-online multi-people tracking by re-identification. *International Journal of Computer Vision*, 128:1937 – 1955.

Leal-Taixé, L., Milan, A., Reid, I., Roth, S., and Schindler, K. (2015). MOTChallenge 2015: Towards a benchmark for multi-target tracking. *arXiv:1504.01942 [cs]*. arXiv: 1504.01942.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Li, W., Zhu, X., and Gong, S. (2017). Person re-identification by deep joint learning of multi-loss classification. *CoRR*, abs/1705.04724.

Lin, Y., Zheng, L., Zheng, Z., Wu, Y., and Yang, Y. (2017). Improving person re-identification by attribute and identity learning. *CoRR*, abs/1703.07220.

Liu, S., Song, Z., Liu, G., Xu, C., Lu, H., and Yan, S. (2012). Street-to-shop: Cross-scenario clothing retrieval via parts alignment and auxiliary set. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3330–3337.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016a). SSD: Single shot MultiBox detector. In *Computer Vision – ECCV 2016*, pages 21–37. Springer International Publishing.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. *CoRR*, abs/2103.14030.

Liu, Z., Luo, P., Qiu, S., Wang, X., and Tang, X. (2016b). Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1096–1104.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137.

Martinel, N., Micheloni, C., and Foresti, G. L. (2014). Saliency weighted features for person re-identification. In *ECCV Workshops*.

McInnes, L., Healy, J., and Melville, J. (2020). Umap: Uniform manifold approximation and projection for dimension reduction.

Milan, A., Roth, S., and Schindler, K. (2014). Continuous energy minimization for multitarget tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(1):58–72.

Milan, A., Schindler, K., and Roth, S. (2013). Detection- and trajectory-level exclusion in multiple object tracking. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3682–3689.

Mittal, A. and Davis, L. S. (2002). M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene using region-based stereo. In *European Conference on Computer Vision*.

Mittal, A., Zhao, L., and Davis, L. (2003). Human body pose estimation using silhouette shape analysis.

Nguyen, D. M. H., Henschel, R., Rosenhahn, B., Sonntag, D., and Swoboda, P. (2021). LMGP: lifted multicut meets geometry projections for multi-camera multi-object tracking. *CoRR*, abs/2111.11892.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, New York, NY, USA, 2e edition.

Ong, J., Vo, B.-T., Vo, B.-N., Kim, D. Y., and Nordholm, S. E. (2020). A bayesian filter for multi-view 3d multi-object tracking with occlusion handling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:2246–2263.

Pirsiavash, H., Ramanan, D., and Fowlkes, C. C. (2011). Globally-optimal greedy algorithms for tracking a variable number of objects. *CVPR 2011*, pages 1201–1208.

Ren, S., He, K., Girshick, R., and Sun, J. (2016). Faster r-cnn: Towards real-time object detection with region proposal networks.

Scarselli, F. and Chung Tsoi, A. (1998). Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results. *Neural Networks*, 11(1):15–37.

Schaul, T. and LeCun, Y. (2013). Adaptive learning rates and parallelization for stochastic, sparse, non-smooth gradients.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823.

Schulter, S., Vernaza, P., Choi, W., and Chandraker, M. (2017). Deep network flow for multi-object tracking.

Shitrit, H. B., Berclaz, J., Fleuret, F., and Fua, P. V. (2014). Multi-commodity network flow for tracking multiple people. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36:1614–1627.

Su, C., Li, J., Zhang, S., Xing, J., Gao, W., and Tian, Q. (2017). Pose-driven deep convolutional model for person re-identification. *CoRR*, abs/1709.08325.

Su, C., Zhang, S., Xing, J., Gao, W., and Tian, Q. (2016). Deep attributes driven multi-camera person re-identification. *CoRR*, abs/1605.03259.

Suh, Y., Wang, J., Tang, S., Mei, T., and Lee, K. M. (2018). Part-aligned bilinear representations for person re-identification. *CoRR*, abs/1804.07094.

Sun, Y., Cheng, C., Zhang, Y., Zhang, C., Zheng, L., Wang, Z., and Wei, Y. (2020). Circle loss: A unified perspective of pair similarity optimization. *CoRR*, abs/2002.10857.

Sun, Y., Zheng, L., Yang, Y., Tian, Q., and Wang, S. (2017). Beyond part models: Person retrieval with refined part pooling. *CoRR*, abs/1711.09349.

Tang, S., Andres, B., Andriluka, M., and Schiele, B. (2015). Subgraph decomposition for multi-target tracking. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5033–5041. IEEE.

Tang, S., Andriluka, M., Andres, B., and Schiele, B. (2017). Multiple people tracking by lifted multicut and person re-identification. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3701–3710.

Tang, Z., Naphade, M., Birchfield, S., Tremblay, J., Hodge, W., Kumar, R., Wang, S., and Yang, X. (2020). PAMTRI: pose-aware multi-task learning for vehicle re-identification using highly randomized synthetic data. *CoRR*, abs/2005.00673.

Tang, Z., Naphade, M., Liu, M., Yang, X., Birchfield, S., Wang, S., Kumar, R., Anastasiu, D. C., and Hwang, J. (2019). Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. *CoRR*, abs/1903.09254.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks.

Vleeschouwer, C., Chen, F., Delannay, D., Parisot, C., Chaudy, C., Martrou, E., and Cavallaro, A. (2008). Distributed video acquisition and annotation for sport-event summarization.

Wan, J. and Li, L. (2013). Distributed optimization for global data association in non-overlapping camera networks. In *2013 Seventh International Conference on Distributed Smart Cameras (ICDSC)*, pages 1–7.

Wang, J., Zhu, X., Gong, S., and Li, W. (2018). Transferable joint attribute-identity deep learning for unsupervised person re-identification. *CoRR*, abs/1803.09786.

Wen, L., Lei, Z., Chang, M.-C., Qi, H., and Lyu, S. (2017). Multi-camera multi-target tracking with space-time-view hyper-graph. *International Journal of Computer Vision*, 122:313–333.

Yi, D., Lei, Z., and Li, S. Z. (2014). Deep metric learning for practical person re-identification. *CoRR*, abs/1407.4979.

You, Q. and Jiang, H. (2020a). Real-time 3d deep multi-camera tracking. *ArXiv*, abs/2003.11753.

You, Q. and Jiang, H. (2020b). Real-time 3d deep multi-camera tracking. *CoRR*, abs/2003.11753.

Yuan, Y., Chen, W., Yang, Y., and Wang, Z. (2019). In defense of the triplet loss again: Learning robust person re-identification with fast approximated triplet loss and label distillation. *CoRR*, abs/1912.07863.

Zamir, A., Dehghan, A., and Shah, M. (2012). Gmcp-tracker: Global multi-object tracking using generalized minimum clique graphs. *ECCV*, 7573.

Zhang, S., Staudt, E., Faltemier, T., and Roy-Chowdhury, A. K. (2015a). A camera network tracking (camnet) dataset and performance baseline. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 365–372.

Zhang, S., Zhu, Y., and Roy-Chowdhury, A. (2015b). Tracking multiple interacting targets in a camera network. *Computer Vision and Image Understanding*, 134:64–73. Image Understanding for Real-world Distributed Video Networks.

Zhang, X., Luo, H., Fan, X., Xiang, W., Sun, Y., Xiao, Q., Jiang, W., Zhang, C., and Sun, J. (2017). Alignedreid: Surpassing human-level performance in person re-identification. *CoRR*, abs/1711.08184.

Zhang, Y., Sun, P., Jiang, Y., Yu, D., Yuan, Z., Luo, P., Liu, W., and Wang, X. (2021). Bytetrack: Multi-object tracking by associating every detection box. *CoRR*, abs/2110.06864.

Zhang, Y., Wang, C., Wang, X., Zeng, W., and Liu, W. (2020a). A simple baseline for multi-object tracking. *CoRR*, abs/2004.01888.

Zhang, Z., Lan, C., Zeng, W., Chen, Z., and Chang, S. F. (2020b). Rethinking classification loss designs for person re-identification with a unified view. *ArXiv*, abs/2006.04991.

Zheng, Z., Zheng, L., and Yang, Y. (2017a). Pedestrian alignment network for large-scale person re-identification. *CoRR*, abs/1707.00408.

Zheng, Z., Zheng, L., and Yang, Y. (2017b). Unlabeled samples generated by GAN improve the person re-identification baseline in vitro. *CoRR*, abs/1701.07717.

Zhong, Z., Zheng, L., Kang, G., Li, S., and Yang, Y. (2017). Random erasing data augmentation. *CoRR*, abs/1708.04896.

Zhou, S., Wang, J., Wang, J., Gong, Y., and Zheng, N. (2017). Point to set similarity based deep feature learning for person re-identification. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5028–5037.

Zhou, X., Koltun, V., and Krähenbühl, P. (2020). Tracking objects as points. *ECCV*.

Zhu, P., Wen, L., Bian, X., Ling, H., and Hu, Q. (2018). Vision meets drones: A challenge. *CoRR*, abs/1804.07437.