Unifying Short and Long-Term Tracking with Graph Hierarchies

Orcun Cetintas\textsuperscript{1}\textsuperscript{*} \quad Guillem Brasó\textsuperscript{1,2}\textsuperscript{*} \quad Laura Leal-Taixé\textsuperscript{1,3}\textsuperscript{†}

\textsuperscript{1}Technical University of Munich \quad \textsuperscript{2}Munich Center for Machine Learning

Abstract

Tracking objects over long videos effectively means solving a spectrum of problems, from short-term association for un-occluded objects to long-term association for objects that are occluded and then reappear in the scene. Methods tackling these two tasks are often disjoint and crafted for specific scenarios, and top-performing approaches are often a mix of techniques, which yields engineering-heavy solutions that lack generality. In this work, we question the need for hybrid approaches and introduce SUSHI, a unified and scalable multi-object tracker. Our approach processes long clips by splitting them into a hierarchy of sub-clips, which enables high scalability. We leverage graph neural networks to process all levels of the hierarchy, which makes our model unified across temporal scales and highly general. As a result, we obtain significant improvements over state-of-the-art on four diverse datasets. Our code and models are available at bit.ly/sushi-mot.

1. Introduction

Multi-Object Tracking (MOT) aims to identify the trajectories of all moving objects from a video. It is an essential task for many applications such as autonomous driving, robotics, and video analysis. Tracking-by-detection is a commonly used paradigm that divides the problem into (i) detecting objects at every frame and (ii) performing data association, i.e., linking objects into trajectories.

In the presence of highly accurate object detections, data association happens mostly among detections that are close in time, i.e., short-term association. Simple cues such as position and motion-based proximity \cite{2,3,45,65,67} or local appearance \cite{30,49,52,66} are often enough for accurate association. Different challenges appear in crowded scenes, when objects may be often occluded and not detected for several frames. This forces methods to perform association among detections in distant time frames, i.e., long-term association. This requires specific solutions that build more robust global appearance models \cite{32,35,44}, create motion models capable of long-term trajectory prediction \cite{11,21,37} or bring robustness by performing association across all frames and all trajectories using a graph representation \cite{4,8,14,43,44}.

Due to the different nature of these tasks, solutions used for short-term associations tend to fail in long-term scenarios. In fact, most state-of-the-art trackers use a combination of approaches to track over different timespans and therefore can be considered to be multi-level trackers. Several short-term trackers use independent re-identification (reID) mechanisms for long-term association \cite{2,28,40,52,57}. Analogously, various graph approaches rely on local trackers to perform short-term association \cite{4,15,16}. All of these hybrid multi-level approaches have two main limitations.

The first one is scalability since current methods cannot deal with long videos. As we increase the timespan between detections to be linked, association becomes more ambiguous due to significant appearance changes and large displacements. Hence, local trackers using a handcrafted combination of appearance and motion cues will fail to scale to arbitrary timespans. Graph-based methods are more robust, but association for large timespans entails the creation of very large graphs (even if combined with local methods), which is infeasible both computationally and memory-wise.

\* Equal contribution.
† Currently at NVIDIA.
The second limitation is **generality**. Using different techniques for different timespans requires making strong assumptions about the cues needed at each temporal scale, which limits the applicability of these approaches. For instance, in tracking scenarios where people dress uniformly and frame rate is high, e.g. dancing videos [41], proximity or motion-based local trackers [2,3,65,67] are more reliable than appearance-based trackers. On the other hand, in the presence of heavy camera motion or low frame rates, the performance of the aforementioned trackers degrades significantly, and appearance may become the most reliable cue [30,60]. Overall, these discrepancies lead inevitably to handcrafted solutions for each type of scenario.

In this work, we ask the following question: *can we design a unified method that generalizes to multiple timespans and further scales to long videos?*

We propose a method that processes videos hierarchically: lower levels of our hierarchy focus on short-term association, and higher levels focus on increasingly long-term scenarios. The key differences to existing hybrid multi-level solutions is that we use the same learnable model for all time scales, i.e., hierarchy levels. Instead of hand-crafting different models for different scales, we show that our model can learn to exploit the cues that are best suited for each time-scale in a data-driven manner. Furthermore, our hierarchy allows a finer-grained transition from short to long-term instead of using two distinct stages. Our method targets the two main limitations of previous works: (i) its hierarchical structure makes it highly scalable and enables processing long clips efficiently, and (ii) it is highly general and does not make any assumptions about which cues are best suited for which timespans, but instead allows the model to obtain the necessary cues in a data-driven manner.

We, therefore, obtain a **Strong** tracker, with a **Unified** solution across timespans, and good **Scalability** thanks to its **Hierarchical** nature, and name it **SUSHI**.

At its core, SUSHI is a graph method, but instead of working on a single monolithic graph, we embrace the different nature of data association over different timespans and operate on a hierarchy of graphs. At the lowest level of our hierarchy, nodes represent object detections in nearby frames. We use a graph neural network (GNN) [4,13,39] to process those into short tracklets, and then build new graphs to generate increasingly longer trajectories at every level of our hierarchy. Notably, we use the **same GNN architecture at every level**. Thus we do not make any assumptions about what cues are best for each timespan. We demonstrate the generality of our approach by showing significant identity preservation improvements over the state-of-the-art in several highly diverse benchmarks: up to +4.7 \text{IDF1} on MOT17 [9], +9.1 \text{IDF1} on MOT20 [10], +9.5 \text{IDF1} on DanceTrack [41], and +4.2 \text{IDF1} on BDD [60], therefore setting new state-of-the-art results by a significant margin.

## 2. Related Work

### Short-term tracking

Numerous modern trackers use frame-by-frame online association frameworks [2,3,28,30,45,49,55,62,65–67]. Motion and spatial proximity cues tend to be central components of these trackers. Notable examples include the widespread use of kalman-filter based motion models [3,49,65,66] or frame-by-frame regression-based frameworks [2,28,62,67]. Some trackers further rely on appearance to increase robustness at lower frame-rates or under strong camera movement [30,49,52,55,66]. While having good performance in short-term scenarios, these trackers lack robustness when it comes to long-term identity preservation.

### Graph-based tracking

Graphs are a commonly used framework to model data association. They model nodes as object detections and edges as trajectory hypotheses. In contrast to frame-by-frame trackers, graph-based methods search for **global** solutions to the data association problem over several frames and are therefore more robust. To this end, numerous optimization frameworks have been studied, including network flows [1,64], multi-cuts [44], minimum cliques [61], disjoint paths [15,16,43], and efficient solvers [1,5] have been designed. In our work, we rely on a simplified version of the min-cost flow formulation [4,64], which allows us to avoid expensive optimization and use a small-scale linear program while still taking advantage of graph-based tracking.

### Learning in graph-based tracking

While early graph-based methods focused on obtaining pairwise association costs from learning methods such as conditional random fields [59], or handcrafted models [42], recent approaches focus almost exclusively on deep learning techniques. Notable examples include learning pairwise appearance costs with convolutional networks [20,36,40], or learning track management policies with recurrent models [29,38]. Recently, numerous approaches learn models that natively operate on the graph domain such as graph neural networks (GNNs) [4,8,14,22,26,51] or transformers [68]. While showing promise, current GNN and transformer-based works have an important limitation: they operate over large monolithic graphs of detections, and therefore lack the scalability needed to process long video clips.

### Multi-level hybrid tracking approaches

Multi-level tracking methods are dominated by handcrafted combinations of approaches. Several early tracking works exploited the idea of building tracks hierarchically [7,17,50,56]. They generally did so by generating short tracklets with hand-crafted methods and then merging those within multiple stages involving different optimization techniques [7,50,56] and association cues [17]. In a similar fashion, numerous modern trackers combine several techniques to build tracks in a hierarchical, incremental way, without necessarily relying on graphs [2,4,8,15,16,28,53]. Some examples include
short-term trackers using external networks for reID-based association \cite{2, 28, 52}, and graph-based methods relying on local trackers to either generate tracklets \cite{4} or fill gaps in trajectories \cite{15, 16}. In our work, we show that we do not need to manually combine techniques, and instead, we use a unified GNN-based framework to efficiently perform data association across multiple hierarchical levels.

3. Background

**Tracking by detection.** Our approach follows the tracking-by-detection paradigm. That is, given a video clip, we assume that object detections are computed for every frame, and our task is to perform data association by linking object detections into trajectories. We denote the set of object detections as $\mathcal{O}$. Each object detection $o_i \in \mathcal{O}$ can be identified by its bounding box coordinates, its corresponding image region, and timestamp. The goal of the data association step is to obtain the set of trajectories $\mathcal{T}^*$ that group detections corresponding to the same identity. Each trajectory $T_k$ is given by its set of detections $T_k := \{o_{k_i}\}_{i=1}^{n_k}$, with $n_k$ being the number of object detections, or trajectory length.

**Graph-based tracking.** Our method builds on the commonly used graph-based formulation of \cite{64}, which we briefly review. We model data association with an undirected graph $G = (V, E)$ in which each node corresponds to an object detection, i.e., $V := \mathcal{O}$. Edges represent association hypotheses among objects at different frames $E \subset \{(o_i, o_j) \in V \times V | t_i \neq t_j\}$. Formally, a time-ordered track $T_k = \{o_{k_i}\}_{i=1}^{n_k}$ with $t_{k_i} < t_{k_{i+1}}$, can be represented as a path in $G$ given by its edges $E(T_k) := \{(o_{k_1}, o_{k_2}), \ldots, (o_{k_{n_k-1}}, o_{k_{n_k}})\}$. Therefore, edges $(u, v) \in E$ can be classified into correct hypotheses if $(u, v) \in E(T_k)$ for some $T_k \in \mathcal{T}^*$, in which case we denote $y_{(u, v)} = 1$; or incorrect otherwise, i.e. $y_{(u, v)} = 0$. Given a set of edge predictions or costs, $\{y_{(u, v)}^{\text{pred}}\}_{(u, v) \in E}$, we are aiming to estimate the set of edge labels $\{y_{(u, v)}\}_{(u, v) \in E}$. Final trajectories can be obtained by rounding predictions into binary decisions via discrete optimization or heuristics, and then extracting their corresponding paths in the graph. Overall, this formulation allows casting data association as edge classification.

4. SUSHI

We provide an overview of SUSHI in Fig. 2. It consists of a sequence of jointly trained SUSHI blocks that operate over a video clip. Starting from initial per-frame object detections (referred to as tracklets of length one from now on) each SUSHI block learns to merge tracklets from the previous level into longer ones. To do so, each SUSHI block builds a graph in which the nodes represent tracklets from the previous level and the edges model trajectory hypotheses. Nodes and edges have associated embeddings encoding position, appearance, and motion cues that are propagated across the graph with a GNN. After several message-passing steps, edge embeddings are classified into correct and incorrect hypotheses, yielding a new set of longer tracklets. By hierarchically stacking several SUSHI blocks, tracklets grow progressively into our final output: final tracks spanning over the entire input video clip.

Overall, the GNN in each SUSHI block learns to exploit timespan dependent association cues and, combined, SUSHI blocks enable tracking over long-time horizons efficiently.
4.1. Constructing a hierarchy of tracking graphs

On the limitations of monolithic tracking graphs. Given an input video clip with \( C \) frames, our goal is to enable the association of objects even if they are occluded for long time spans, i.e., up to \( C - 1 \) frames. Given the graph formulation we introduced in Sec. 3, this is only possible if we consider edges in our graph spanning across all possible time distances. Doing so naïvely has two main drawbacks. First, it is prohibitively expensive for long sequences, as it requires either considering a quadratic number of edges or using sophisticated pruning techniques. Secondly, it implies that most edges in the graph will represent incorrect hypotheses. To prove that, observe that each object detection can, at most, be incident to two correct edges (one in the past, and one in the future). Therefore, in a clip with \( n \) object detections, the correct number of hypotheses is at most \( 2n \), while the number of edges can grow up to \( n^2 \). The ratio \( o \left( \frac{n^2}{2n} \right) \) will vanish for sequences consisting of thousands of detections and will cause a severe label imbalance for learning methods operating on those edges.

Building a hierarchical clip partition. Motivated by the aforementioned limitations, we propose a hierarchy of smaller graphs that operate over long video clips instead of a single large monolithic graph. Our hierarchy is based on a recursive partition of the clip into non-overlapping time windows or smaller clips. We illustrate our construction in Figure 3. At each consecutive hierarchy level we only consider edges among tracklets contained in small windows of consecutive frames, which ensures that our tracklet length is relatively uniform at every level. After each consecutive level in our hierarchy, we merge tracklets that are close in time into longer ones. These longer tracklets then become our new set of nodes to be associated at the following hierarchy level. By recursively merging tracklets in nearby frames, we progressively reduce the number of nodes after each hierarchy level. Therefore, at each consecutive level of our hierarchy, we can consider edges spanning across longer timespans without neither prohibitively increasing the edge count, nor incurring a severe label imbalance.

4.2. Learning a unified hierarchical tracker

Overview. In the previous section, we presented a hierarchical graph-based framework to recursively merge short tracklets into longer ones. We use a message-passing GNN to process graphs in our hierarchy and learn to merge tracklets. We refer to each hierarchy level of our model as a SUSHI block. A core feature of our method is that each SUSHI block uses the same architecture, and has access to the same feature sources. This is in contrast to previous work that engineered different solutions for different levels [4, 8, 15, 16]. Therefore, instead of making assumptions on the cues needed to perform association at each timespan, we let SUSHI blocks at each hierarchy learn these from data.

SUSHI blocks. An illustration of a SUSHI block is shown in Figure 2. Inspired by [4], the main idea behind them is: (i) consider a graph with tracklets as nodes as defined in Section 4.1, (ii) propagate node and edge embeddings across the graph via message-passing, and (iii) perform edge classification to merge nodes, i.e., tracklets, into longer tracklets. More specifically, for each graph \( G^l = (V^l, E^l) \) at level \( l \) in our hierarchy, we consider embeddings \( h_v^{(0)} \in \mathbb{R}^{d_v} \) and \( h_{(u,w)}^{(0)} \in \mathbb{R}^{d_e} \) for every node \( v \in V^l \) and edge \((u, w) \in E^l\), with \( d_v \) and \( d_e \) being their respective dimension. Node embeddings are zero-initialized, and edge embeddings are learned from a set of initial association features. The goal of message-passing is to then propagate embeddings across the graph, hereby encoding higher-order contextual information in them. To do so, we follow the time-aware neural message-passing framework of [4] for \( s = 1, \ldots, S \) steps, yielding embeddings \( h_{(u,w)}^{(1)}, \ldots, h_{(u,w)}^{(S)} \). After the last step, edge embeddings are fed to a binary classifier to obtain a score representing whether they represent a correct hypothesis or not, \( y_{(u,v)} = \text{MLP}_\text{class}(h_{(u,v)}^{(S)}) \). After that, predicted edge scores are rounded to obtain binary decisions using a linear program. As an end result, we obtain a longer-spanning set of tracklets from the initial ones in \( V^l \). Further details on our GNN and rounding scheme are provided in the supplementary material.

Edge association cues. We compute input edge embeddings to the GNN in each SUSHI block by feeding an initial vector of concatenated pairwise association features to a light-weight multi-layer perception \( \text{MLP}_{\text{edge}} \). The initial vector computation is a generalization and extension of [4] to tracklets, and based on spatial and motion-based proximity, time distance, and reID embedding-based appearance similarity between nodes. For appearance, we consider the
average embedding vector over all detections in the tracklet which is more robust than that of a single detection. For spatial, size, and time proximity, we use the closest detections in time for each pair of nodes. Explicitly working with tracklets allows us to utilize motion, a strong cue that [4] does not exploit. Given tracklets $T_u$ and $T_v$, we estimate their pixelwise velocities as $v_u$ and $v_v$, respectively. We then forward propagate $u$’s last position and backward propagate $v$’s first position until their middle time point $t^\text{mid} := (t_v - t_u)/2$. Formally, we compute $\text{pos}^\text{fwrd}_u := b_{u,v} + t^\text{mid} v_u$ and $\text{pos}^\text{bwd}_v := b_{v,u} - t^\text{mid} v_v$, to obtain the edge feature $\text{GIoU}(\text{pos}^\text{fwrd}_u, \text{pos}^\text{bwd}_v)$, where $\text{GIoU}$ is the Generalized Intersection over Union score [33]. For further details, we refer to the supplementary material.

Weight sharing. SUSHI blocks use the same GNN architecture at every hierarchy level. We observe that we can share parameters and learnable weights among the GNNs used for each SUSHI block. To do so, we sequentially unfreeze levels from first to last. This ensures that tracklets used in each subsequent level in the hierarchy during training are stable and accurate enough to provide a meaningful signal. Specifically, starting with only the first block being unfrozen, we unfreeze each subsequent level after $M$ training iterations, with $M = 500$ in our implementation. After the GNNs from all levels are unfrozen, they are trained jointly. This allows them to adapt to the particular tracklet statistics produced by their predecessors. For each level, we use a focal loss [25] over the edge classification scores produced and sum losses over all levels to obtain our final loss.

5. Experiments

5.1. Datasets and Metrics

We conduct experiments on four public benchmarks with significantly different properties: MOT17 [9], MOT20 [10], DanceTrack [41], and BDD100K [60].

MOT17. The benchmark includes diverse scenarios with varying camera viewpoints and pedestrian densities. It consists of 14 videos (7 for training, and 7 for testing) with a total of 11235 frames. In the public setting, object detections are provided to emphasize data association performance during evaluation, while in the private setting trackers are allowed to use their own set of detections.

MOT20. The dataset focuses on extremely crowded scenes, with some videos containing over 200 pedestrians per frame on average. It contains 8 videos, a total of 13410 frames with more than 2 million object annotations. Similar to MOT17, it includes public and private settings.

DanceTrack. The benchmark consists of 100 group dancing videos (105855 frames). Scenes from this dataset are moderately crowded and contain persons with highly similar appearances and complex and diverse motion patterns.

BDD100K. The dataset focuses on egocentric vision from a car-mounted camera in highly diverse autonomous driving scenarios. Therefore, large camera motion is prominent. The MOT benchmark contains 2,000 videos with approximately 400K images and 8 object categories.

Metrics. To evaluate several aspects of MOT, we report the widely accepted metrics IDF1 [34], MOTA [18], and recently introduced HOTA [27]. While IDF1 measures identity preservation, MOTA is heavily biased towards trajectory coverage, i.e., detection quality. HOTA finds balance between both aspects, and unifies detection, association and localization performance into a single metric. Since our method focuses on data association, we treat IDF1 and HOTA as main metrics, and also report MOTA and identity switches (ID Sw.) for completeness.

5.2. Implementation Details

Model architecture and training. SUSHI consists of nine SUSHI blocks utilizing MPNTrack [4]’s GNN architecture and shared weights between hierarchy levels. For our reID network we use a pretrained ResNet50-IBN following [8]. We do not fine-tune our reID model on tracking videos, and simply freeze it during training. GNNs across hierarchical levels are trained jointly with a learning rate of $3 \cdot 10^{-4}$ and a weight decay of $10^{-4}$ in batches of 8 clips for 250 epochs. We use focal loss with $\gamma = 1$ and the Adam optimizer [19].

Hierarchy construction and inference. Though our model is capable of processing sequences of arbitrary length, we construct hierarchies spanning a maximum temporal edge distance of 512 frames (approx. 17-36 seconds on MOT17). We experimentally found that this covers the majority of occlusions and long-term association cases across multiple datasets, and is a significant improvement over previous works, which processed clips of up to 15 [4], 32 [68], and 60 frames [15, 16]. Our hierarchy consists of nine levels and we process $2^l$ frames in each hierarchy level $l$. For each graph, we connect each node to its top 10 nearest neighbors based on geometry, appearance, and motion similarity. We process entire videos by feeding overlapping clips of 512 frames to our method in a sliding window fashion. We then merge per-clip tracks into trajectories of arbitrary
length with a simple stitching scheme, following [4]. During inference, we fill trajectory gaps by linear interpolation.

**Runtime.** Inference runs at a competitive runtime of 21fps on MOT17 with given detections.

**Object Detections.** For DanceTrack and BDD, as well as the private settings of MOT17 and MOT20, we obtain object detections from a YOLOX detector [12] trained following [65]. In the public setting of MOT17 and MOT20, for a fair comparison to the state-of-the-art, we use the public detections provided by MOTChallenge and preprocess them with [2] following [4, 8, 14, 15].

### 5.3. Ablation Studies

We conduct the ablation experiments on the MOT17 train set by performing 3-fold cross-validation following [4]. Our ablation studies analyze two main aspects of our method: (i) the advantages of using a unified method across different levels of our hierarchy for data association, and (ii) our model’s ability to scale to long video clips.

**Hybrid vs unified multi-level approaches.** In Table 1 we analyze the performance differences between hybrid multi-level approaches and our unified GNN-based approach. We consider four association time horizons 25, 75, 150 and 512 frames, and analyze the strengths of each approach at each time horizon. We initially consider two local baselines: a framewise hungarian matching tracker (IoU), and a superior regression-based tracker (Tracktor) [2]. As additional baselines, we add long-term association levels using a combination of reID and motion cues (IoU++, Tracktor++). We observe that single-level GNNs show stronger performance at this second level (MPNTrack, MPNTrack++)\(^1\), and using GNNs at both levels achieves the best performance gains at up to 75 frames. This shows that the benefits of being a unified method are also applicable at moderate timespans of up to 75 frames. In the following three rows, we attempt to naively extend hybrid approaches to perform association over longer time horizons of up to 150 frames, and observe no significant improvements, due to the increased difficulty of these scenarios. In contrast, our three-level hierarchy benefits from tracking over increased timespans. Finally, we investigate very long time horizons in the penultimate three rows and observe that MPNTrack++ again fails to show significant improvements while our nine-level hierarchy leads to an even better tracking performance with 77.6 IDF1. Next, we further investigate the need for hierarchy levels.

**The potential of hierarchies over monolithic graphs.** As explained in Sec. 4.1, our graph hierarchy allows us to cover large time spans with a significantly reduced number of edges, while mitigating the label imbalance between correct and incorrect trajectory hypotheses. To quantify this phenomenon, we compute the oracle IDF1 score, i.e., assuming perfect edge classification, of models consisting of a varying number of hierarchical levels over video clips of 512 frames. In Fig. 4a, we report both the overall number of edges needed to achieve each score, and the corresponding ratio of correct edge hypotheses, i.e., labels, encoded as the circle area. For each number of levels (color-coded) each of its data points is obtained by changing the number of nearest neighbors of nodes. We observe a clear trend: hierarchical graphs achieve significantly higher scores for a fixed memory budget. At the same time, they do so with a significantly less severe label imbalance, i.e., a larger circle area, which indicates that learning on them will be significantly easier.

---

**Table 1. Ablation study on hybrid and unified multi-level approaches.**

<table>
<thead>
<tr>
<th>Method</th>
<th>Short (≤ 25 Frames)</th>
<th>Long (75 Frames)</th>
<th>Long++ (150 Frames)</th>
<th>Very Long (512 Frames)</th>
<th>IDF1</th>
<th>HOTA</th>
<th>MOTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>IoU</td>
<td>IoU</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>60.5</td>
<td>55.4</td>
<td>64.9</td>
</tr>
<tr>
<td>Tracktor</td>
<td>Reg.</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>65.6</td>
<td>58.9</td>
<td>66.0</td>
</tr>
<tr>
<td>IoU++</td>
<td>IoU</td>
<td>ReID + Motion</td>
<td>✗</td>
<td>✗</td>
<td>65.1</td>
<td>58.4</td>
<td>65.8</td>
</tr>
<tr>
<td>Tracktor++</td>
<td>Reg.</td>
<td>ReID + Motion</td>
<td>✗</td>
<td>✗</td>
<td>69.2</td>
<td>60.7</td>
<td>66.8</td>
</tr>
<tr>
<td>MPNTrack</td>
<td>Reg.</td>
<td>GNN</td>
<td>✗</td>
<td>✗</td>
<td>71.8</td>
<td>62.8</td>
<td>67.6</td>
</tr>
<tr>
<td>MPNTrack++</td>
<td>Reg.</td>
<td>GNN</td>
<td>✗</td>
<td>✗</td>
<td>73.2</td>
<td>62.9</td>
<td>67.7</td>
</tr>
<tr>
<td>SUSHI (Ours) - 2 Level</td>
<td>Hierarchy of 2 GNN</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>73.5</td>
<td>63.8</td>
<td>67.7</td>
</tr>
<tr>
<td>Tracktor++</td>
<td>Reg.</td>
<td>ReID+Motion</td>
<td>✗</td>
<td>✗</td>
<td>71.2</td>
<td>62.5</td>
<td>66.4</td>
</tr>
<tr>
<td>MPNTrack</td>
<td>Reg.</td>
<td>GNN</td>
<td>✗</td>
<td>✗</td>
<td>71.9</td>
<td>62.8</td>
<td>67.6</td>
</tr>
<tr>
<td>MPNTrack++</td>
<td>Reg.</td>
<td>GNN</td>
<td>✗</td>
<td>✗</td>
<td>73.4</td>
<td>63.7</td>
<td>68.0</td>
</tr>
<tr>
<td>SUSHI (Ours) - 2 Level</td>
<td>Hierarchy of 2 GNN</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>73.8</td>
<td>64.2</td>
<td>68.3</td>
</tr>
<tr>
<td>SUSHI (Ours) - 3 Level</td>
<td>Hierarchy of 3 GNN</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>74.8</td>
<td>64.9</td>
<td>68.3</td>
</tr>
<tr>
<td>MPNTrack++</td>
<td>Reg.</td>
<td>GNN</td>
<td>✗</td>
<td>✗</td>
<td>76.2</td>
<td>65.5</td>
<td>68.8</td>
</tr>
<tr>
<td>SUSHI (Ours) - 3 Level</td>
<td>Hierarchy of 9 GNN</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>77.6</td>
<td>66.7</td>
<td>68.9</td>
</tr>
</tbody>
</table>

---

\(^1\)For MPNTrack++, we enhance MPNTrack with our motion-based edge feature.
(a) Potential of hierarchies over monolithic graphs.  
(b) Effect of clip length and hierarchy levels.  
(c) Importance of features at different levels.

Figure 4. Ablation studies on scalability of our approach and association cues.

It is important to note that the benefits of our hierarchy saturates around nine hierarchy levels, and therefore we use this value in our model. Overall, Fig. 4a, highlights the main two reasons behind the scalability of our hierarchy.

<table>
<thead>
<tr>
<th>Method</th>
<th>Det Ref.</th>
<th>HOTA ↑</th>
<th>MOTA ↑</th>
<th>ID Sw. ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOT17 - Public</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tracktor [2]</td>
<td>Tracktor</td>
<td>55.1</td>
<td>44.8</td>
<td>56.3</td>
</tr>
<tr>
<td>LPT [24]</td>
<td>Tractor</td>
<td>57.7</td>
<td>–</td>
<td>57.3</td>
</tr>
<tr>
<td>MPNTrack [4]</td>
<td>Tractor</td>
<td>61.7</td>
<td>49.0</td>
<td>58.8</td>
</tr>
<tr>
<td>Lif,T [15]</td>
<td>Tractor</td>
<td>65.6</td>
<td>51.3</td>
<td>60.5</td>
</tr>
<tr>
<td>ApLift [16]</td>
<td>Tractor</td>
<td>65.6</td>
<td>51.1</td>
<td>60.5</td>
</tr>
<tr>
<td>GMT [14]</td>
<td>Tractor</td>
<td>65.9</td>
<td>51.2</td>
<td>60.2</td>
</tr>
<tr>
<td>LPC_MOT [8]</td>
<td>Tractor</td>
<td>66.8</td>
<td>51.5</td>
<td>59.0</td>
</tr>
<tr>
<td><strong>SUSHI (Ours)</strong></td>
<td>Tracktor</td>
<td>71.5</td>
<td>54.6</td>
<td>62.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Det Ref.</th>
<th>HOTA ↑</th>
<th>MOTA ↑</th>
<th>ID Sw. ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOT17 - Private</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QDTTrack [31]</td>
<td>✗</td>
<td>66.3</td>
<td>53.9</td>
<td>68.7</td>
</tr>
<tr>
<td>TrackFormer [28]</td>
<td>✗</td>
<td>68.0</td>
<td>57.3</td>
<td>74.1</td>
</tr>
<tr>
<td>MOTR [63]</td>
<td>✗</td>
<td>68.6</td>
<td>57.8</td>
<td>73.4</td>
</tr>
<tr>
<td>PermaTrack [45]</td>
<td>✗</td>
<td>68.9</td>
<td>55.5</td>
<td>73.8</td>
</tr>
<tr>
<td>MeMOT [6]</td>
<td>✗</td>
<td>69.0</td>
<td>56.9</td>
<td>72.5</td>
</tr>
<tr>
<td>GTR [68]</td>
<td>✗</td>
<td>71.5</td>
<td>59.3</td>
<td>75.3</td>
</tr>
<tr>
<td>FairMOT [66]</td>
<td>✗</td>
<td>72.3</td>
<td>59.3</td>
<td>73.7</td>
</tr>
<tr>
<td>GRTU [47]</td>
<td>✗</td>
<td>75.0</td>
<td>62.0</td>
<td>74.9</td>
</tr>
<tr>
<td>CoreTracker [46]</td>
<td>✗</td>
<td>73.6</td>
<td>60.7</td>
<td>76.5</td>
</tr>
<tr>
<td>Unicorn [58]</td>
<td>✗</td>
<td>75.5</td>
<td>61.7</td>
<td>77.2</td>
</tr>
<tr>
<td>ByteTrack fur 65</td>
<td>✗</td>
<td>71.1</td>
<td>62.8</td>
<td>78.9</td>
</tr>
<tr>
<td>ByteTrack [65]</td>
<td>✗</td>
<td>77.3</td>
<td>63.1</td>
<td>80.3</td>
</tr>
<tr>
<td><strong>SUSHI (Ours)</strong></td>
<td>✗</td>
<td>83.1</td>
<td>66.5</td>
<td>81.1</td>
</tr>
</tbody>
</table>

Table 2. Test set results on MOT17 benchmark. Det. Ref. denotes the public detection refinement strategy. As ByteTrack (gray) uses different thresholds for test set sequences and interpolation, we also report scores by disabling these as ByteTrack† (black).

Effect of clip length and hierarchy levels. In Fig. 4b, we visualize the impact on identity preservation (IDF1) when we increase the number of hierarchy levels and the clip length, hence incorporating longer-term association scenarios. As explained in the previous paragraph, increasing the clip size allows the tracker to potentially bridge longer occlusions, but a naive increase in size yields a very large graph with severe label imbalance. This is confirmed in the first column of Fig. 4b: increasing the clip length in a non-hierarchical way, i.e., using a single level, can even harm performance when going beyond moderate lengths (≥ 32 frames). Conversely, given a fixed clip size, increasing the number of hierarchy levels up to nine yields monotonic improvements for all clip lengths. Overall, our hierarchical framework enables processing clip sizes of up to 512 frames, with an overall improvement of +13.4 IDF1 over a naive, i.e., single level, baseline.

Importance of features at different levels. As explained in Sec. 4.2, our GNNs exploit three main feature modalities: position, appearance, and motion. In Fig. 4c, we show our network’s ability to utilize association cues differently at each hierarchy level, that is, over different association timespans. To do so, we report the performance loss obtained from removing edge features from each type over individual levels. Starting from completely removing features from each modality while keeping other modalities intact (first column), we add features of the target type sequentially over consecutive levels starting from the lowest level. We observe that appearance has the largest impact, but it is mostly used at later hierarchy levels since it leads to -3.9 IDF1 when disabled for levels 7-9 compared to -1.2 IDF1 for levels 1-3. Motion seems to have a moderate but uniform impact across levels, and lastly, position information is only relevant for short-term association.

5.4. Benchmark Results

MOT17. Under the public setting, our model outperforms all published work using Tracktor-based preprocessing (Table 2). It is worth noting that all listed methods are graph-based and our approach significantly surpasses their identity preservation performance, as measured by IDF1 and HOTA.
Compared to MPNTrack, we achieve an improvement of 9.8 IDF1 and 5.6 HOTA, despite using their GNN architecture in our SUSHI blocks, which highlights the importance of our graph hierarchy. Analogously, we also obtain significant improvements over hybrid graph-based methods [8, 15, 16]. In the private setting, we significantly improve upon all published methods. While using the same detector as ByteTrack [65], our model improves upon it by 5.8 IDF1 and 3.4 HOTA, and reduces ID switches by 50%.

<table>
<thead>
<tr>
<th>Method</th>
<th>Det Ref.</th>
<th>IDF1 ↑</th>
<th>HOTA ↑</th>
<th>MOTA ↑</th>
<th>ID Sw. ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracktor [2]</td>
<td>Tracktor</td>
<td>52.7</td>
<td>42.1</td>
<td>52.6</td>
<td>1648</td>
</tr>
<tr>
<td>LPT [24]</td>
<td>Tracktor</td>
<td>53.5</td>
<td>–</td>
<td>57.9</td>
<td>1827</td>
</tr>
<tr>
<td>ApLift [16]</td>
<td>Tracktor</td>
<td>56.5</td>
<td>46.6</td>
<td>58.9</td>
<td>2241</td>
</tr>
<tr>
<td>MPNTrack [4]</td>
<td>Tracktor</td>
<td>59.1</td>
<td>46.8</td>
<td>57.6</td>
<td>1210</td>
</tr>
<tr>
<td>LPC_MOT [8]</td>
<td>Tracktor</td>
<td>62.5</td>
<td>49.0</td>
<td>56.3</td>
<td>1562</td>
</tr>
<tr>
<td><strong>SUSHI (Ours)</strong></td>
<td>Tracktor</td>
<td><strong>71.6</strong></td>
<td><strong>55.4</strong></td>
<td><strong>61.6</strong></td>
<td><strong>1053</strong></td>
</tr>
</tbody>
</table>

Table 3. Test set results on MOT20 benchmark. Det. Ref. denotes the public detection refinement strategy. As ByteTrack (gray) uses different thresholds for test set sequences and interpolation, we also report scores by disabling these as ByteTrack† (black).

**MOT20.** In the crowded and challenging scenes of MOT20, we achieve even greater improvements over previous work, which further highlights the scalability benefit of our hierarchical and unified framework (Table 3). In the public setting, we surpass all state-of-the-art approaches, remarkably, by 9.1 IDF1 and 6.4 HOTA while, again, using the same refinement model over public detections. In the private setting, we also advance the state-of-the-art by 4.6 IDF1 and 3.0 HOTA, while again significantly reducing ID switches.

**DanceTrack.** In Table 4, we report a remarkable improvement of 9.5 IDF1 and 9.1 HOTA over state-of-the-art. Given the unique features of this dataset, these results highlight the versatility of our approach to utilize the right cues for different scenarios. This is in contrast with methods like [65] that show strong performance in HOTA at both MOT17 and MOT20, but fall behind other approaches on DanceTrack. Our improvements are consistent across datasets, which demonstrates the generality of SUSHI.

**BDD.** We further demonstrate the versatility of our approach in Table 5 where we report results on the highly diverse BDD. Note that this dataset contains multiple classes, and we simply apply our GNNs across all of them. This naive extension of our approach already achieves significant improvements of +4.2 mIDF1 and +4.9 IDF1 over the state-of-the-art. Overall, these results further consolidate the generality of our approach and its ability to accurately track non-pedestrian classes.

### 6. Conclusion

We have presented SUSHI, a unified method for tracking across multiple timespans. Through our ablation studies, we have shown clear benefits from the two main features of our approach: (i) its unified nature across temporal scales, and (ii) its ability to scale to long video clips. Moreover, our benchmark results have demonstrated our model’s ability to significantly advance state-of-the-art across highly diverse tracking scenarios, hence proving its generality.

We expect SUSHI to inspire future research by questioning the need for engineering timescale-specific solutions for tracking. Lastly, we believe SUSHI makes significant progress towards tackling long-term tracking, and will highlight the potential of graph hierarchies towards this end.
References


[14] Jiawei He, Zehao Huang, Naiyin Wang, and Zhaoxiang Zhang. Learnable graph matching: Constructing graph partitioning with deep feature learning for multiple object tracking. In *CVPR*, pages 5299–5309, 2021. 1, 2, 6, 7


[30] Jiangmiao Pang, Linlu Qiu, Xia Li, Hao Feng Chen, Qi Li, Trevor Darrell, and Fisher Yu. Quasi-dense similarity learn-


