Bayesian Tracking of Video Graphs Using Joint Kalman Smoothing and Registration

Aditi Basu Bal[1], Ramy Mounir[2], Sathyanarayanan Aakur[3], Sudeep Sarkar[2], and Anuj Srivastava[1]

[1] Florida State University, Tallahassee, FL 32309, USA
[2] University of South Florida, Tampa, FL 33620, USA
[3] Oklahoma State University, Stillwater, OK 74078
ab18z@fsu.edu, {ramy, sarkar}@usf.edu, saakurn@okstate.edu, anuj@stat.fsu.edu

Abstract. Graph-based representations are becoming increasingly popular for representing and analyzing video data, especially in object tracking and scene understanding applications. Accordingly, an essential tool in this approach is to generate statistical inferences for graphical time series associated with videos. This paper develops a Kalman-smoothing method for estimating graphs from noisy, cluttered, and incomplete data. The main challenge here is to find and preserve the registration of nodes (salient detected objects) across time frames when the data has noise and clutter due to false and missing nodes. First, we introduce a quotient-space representation of graphs that incorporates temporal registration of nodes, and we use that metric structure to impose a dynamical model on graph evolution. Then, we derive a Kalman smoother, adapted to the quotient space geometry, to estimate dense, smooth trajectories of graphs. We demonstrate this framework using simulated data and actual video graphs extracted from the Multiview Extended Video with Activities (MEVA) dataset. This framework successfully estimates graphs despite the noise, clutter, and missed detections.

Keywords: motion tracking, graph-representations, video graphs, quotient metrics, Kalman smoothing, nonlinear manifolds

1 Introduction

Graph-based representations in videos provide a convenient setting to represent and analyze higher-level structures of scenes. Graphs help focus us on information of interest while discarding irrelevant items for the given task. Graphs are also helpful in capturing spatiotemporal interactions between nodes (humans, objects, etc.) in video sequences while reducing the effects of noise, clutter and (appearance/scene) variability. This representation facilitates the modeling and testing of statistical variability across multiple observations within and across observation classes. The promise of graphical representations, especially in performing statistical analysis, is fueling interest in developing novel mathematical and statistical techniques for handling graph data [22,21,16,6,5,15].

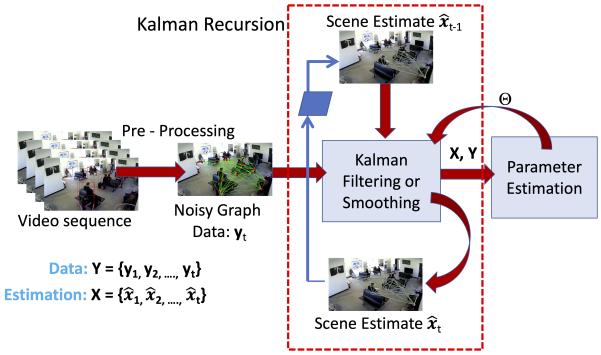

Fig. 1: An overview of time-series analysis for estimating graphs in video frames.

This paper advances the use of graphical methods for handling video streams in computer vision and artificial intelligence. It represents the detections at any time – **objects as nodes** and **their interactions as edges** in a graph. As the objects and interactions change over time, one obtains a time series of graphs representing an evolving scene. We focus on the problem of analyzing these *dynamical graphs or time-series of graphs*, where each time point represents a graph extracted from a frame (or a small set of consecutive frames). In this setup, the attributes associated with nodes and edges are typically real-valued vectors obtained by embedding concepts, and their co-occurrences in some large Euclidean space (as observed in Visual Genome [26], Action Genome [23], and ConceptNet [28,37], to name a few). The nodes' attributes are often descriptive of the objects and their spatiotemporal locations in the scenes. The edge attributes are designed to capture pairwise relational properties of objects. The sequence of graphs as extracted video data forms our observed time series.

Time-series analysis of graphs, especially when dealing with noisy and cluttered data, requires novel tools. In the current context, corrupted data results from limitations in the pre-processing steps. These steps result in random perturbations in edge-node attributes, false detection of spurious concepts, and missed nodes when concepts go undetected. The goal is to use this data to estimate the underlying time series of true graphs. We aim to use model-based estimation, equipped with dynamical models and likelihood functions, to help overcome these data limitations and generate robust inferences. The classical toolsets in (Euclidean) time-series analysis are filtering, smoothing, prediction, and estimation. Can we directly apply them to the graph data? The answer is no. The main issue with graphs is that ordering nodes (or objects) in a graph is arbitrary. In other words, one can randomly re-order the nodes in a graph and still preserve the scene description, although it may change its mathematical representation. This motivates a representation space that is naturally a quotient space under re-ordering of nodes. Consequently, the graph space $\mathcal{G}$ is a non-Euclidean space where the past Euclidean filtering techniques do not apply directly.

This paper develops a Kalman filtering and smoothing approach adapted to the non-Euclidean geometry of $\mathcal{G}$ to perform Bayesian inference. Fig. 1 provides an overview of the proposed framework. Using deep learning techniques (detailed in Supplementary), we obtain noisy graph data $\mathbf{y}_t$ at each time t. Then we apply a combination of Kalman estimation and maximum-likelihood estimation of the system parameters Θ to obtain the scene estimate $\hat{\mathbf{x}}_t$. This Bayesian estimate uses system dynamics and the data likelihood (while registering nodes across time points) to form optimal estimates.

Contributions. The main contributions of this paper are three-fold: (i) We introduce a novel framework for Bayesian estimation/tracking using Kalman filtering and smoothing for time-series of unregistered graphs. A graph-formulation provides joint inferences on the detected objects (nodes) and their relationships; (ii) We formulate estimation in the quotient space of graphs modulo the node-registration group. Incorporating registration inside filtering allows for optimal matching and tracking of nodes over time and handles the problem of missed and false nodes in noisy and cluttered data; and (iii) We demonstrate the effectiveness of the proposed approach through extensive experimental results on both simulated and real datasets. This is the first paper of its kind that formulates a time-series estimation on the quotient space of graphs.

2 Related Work

Graph-based representations are commonly used in the modeling and prediction of human trajectory [3,19,20,25,34,31,44]. In many cases, graphs are used to model the human-human and human-objects interactions in the context of trajectory and behavior prediction. Another common application of graph-based representations is Multiple Object Tracking (MOT), where the goal is to match detected objects across frames [4,27,47,48] and assign a registration for this graph matching problem. Graphs have also been used for modeling human activity [2,1] in videos by constructing graphs that exploit semantic and commonsense knowledge [37]. In such approaches, the graph representations are constrained by context and physical laws. Some other tasks, *e.g.*, video object segmentation [29,45], utilize graph representations to extract relevant information from neighboring frames in a video by employing an attentive mechanism.

Time series prediction over graphs has not been explored to a great extent in prior literature. The most common use of graphs in time series prediction [7,49,11,36,43] has been for modeling the relationships between the variables in multivariate time series data such as traffic forecasting [51] and action and gesture recognition [50,41]. There have been few works that address the prediction of actual graph structures over time using Gaussian process regression [32], manifolds-based prediction [33], Kalman and other filtering methods applied to graph or network data [35,10,24]. However, these methods typically assume graphs with fully registered nodes and do not account for clutter.

3 Background: Graph Representation & Quotient Metric

This section describes the necessary background for understanding the proposed approach. Specifically, we lay out the mathematical framework needed for analyzing unregistered graphs as elements of a quotient space [21,22,16].

Representing Graphs. We represent a graph G of n nodes by two sets of variables: an *adjacency matrix* and a *node attribute vector*. First, we define an adjacency matrix $A \in \mathbb{R}^{n\times n\times p}$ to capture the edge attributes, where an element $A_{ij} \in \mathbb{R}^p$ represents the interaction between node i and j. We assume that $\{A_{ij}\}$ is a symmetric matrix with diagonal elements being zero and has only $k_n = n(n-1)/2$ degrees of freedom in $\mathbb{R}^p$. (In case the interactions are directional, one will need to keep the full matrix A with $k_n = n^2$.) Hence, we can represent A by a long vector of its upper triangular sub-matrix: $\mathbf{w} \in \mathbb{R}^{k_n\times p}$. Note that there is a one-to-one relationship between the adjacency matrix A and the vector $\mathbf{w}$. Let $\phi(A) = \mathbf{w}$ denote the mapping of the adjacency matrix into a vector form; then $\phi^{-1}(\mathbf{w}) = A$ maps the vector back to the adjacency matrix. Second, we define a node attribute vector $\mathbf{v} \in \mathbb{R}^{n\times m}$ such that each element $\mathbf{v}_i \in \mathbb{R}^m$ denotes the attributes of the i-th node.

Metric Structure. We define a joint edge-node representation $X = (\mathbf{w}, \mathbf{v}) \in \mathcal{X}$, with the representation space given by $\mathcal{X}_n \doteq (\mathbb{R}^{k_n\times p} \times \mathbb{R}^{n\times m})$. $\mathcal{X}_n$ is a Euclidean space with the standard Euclidean metric. The distance on $\mathcal{X}$ is given by: for any $X^{(1)} \equiv (\mathbf{w}^{(1)}, \mathbf{v}^{(1)}), X^{(2)} \equiv (\mathbf{w}^{(2)}, \mathbf{v}^{(2)})$, define:

$$d_x(X^{(1)}, X^{(2)}) = \sum_{k=1}^{k_n} \|\mathbf{w}_k^{(1)} - \mathbf{w}_k^{(2)}\|^2 + \lambda \sum_{i=1}^{n} \|\mathbf{v}_i^{(1)} - \mathbf{v}_i^{(2)}\|^2, \tag{1}$$

where $\lambda > 0$ is the relative weight of the second term and d_x is a weighted combination of the corresponding metrics on both the edge and node attributes. The set of all graphs with different number of nodes is given by the union $\mathcal{X} = \cup_{n=1}^{\infty} \mathcal{X}_n$.

Graph Matching and Comparison. A critical issue in comparing graphs is that the size and ordering of nodes can be arbitrary in given data. Consequently, matching nodes across graphs becomes an intermediate problem when comparing different video frames. The computation of metric d_x requires registration of nodes between $X^{(1)}$ and $X^{(2)}$, which may not be known beforehand. Furthermore, the two graphs may have a different number of nodes (due to missing or false nodes), making node matching more difficult. To handle this *registration problem*, we introduce the action of the permutation group on $\mathcal{X}$ as follows. Let $\mathcal{P}_n$ denote the set of all $n \times n$ permutation matrices: each $P \in \mathcal{P}_n$ is an $n \times n$ matrix with only one 1 in each column and row and all other entries are 0. The mapping $\mathbf{v} \mapsto P\mathbf{v}$ permutes the elements of $\mathbf{v}$ according to the elements of P, changing the ordering of nodes in the graph. (Note that the ordering within the node attributes remains unchanged.) The corresponding adjacency

Algorithm 1: Kalman Filtering for Graph Time Series Prediction

Input : Observed Graph Sequence: $\mathbf{Y} = \{\mathbf{y}_t : t = 1, 2, \ldots, T\}$
Output: Estimated Graph Sequence: $\hat{\mathbf{X}} = \{\hat{\mathbf{x}}_t : t = 1, 2, \ldots, T\}$

for $t = 2$ *to* T **do**
  `/* Registration Step (Performed only once for a dataset.) */`
  $P^* = \underset{P \in \mathcal{P}_n}{\operatorname{argmin}} \quad d_x((\mathbf{y}_{t-1}^{(e)}, \mathbf{y}_{t-1}^{(n)}), (P \star \mathbf{y}_t^{(e)}, P\mathbf{y}_t^{(n)}))$
  $(\mathbf{y}_t^{(e)}, \mathbf{y}_t^{(n)}) \overset{\text{set}}{=} (P^* \star \mathbf{y}_t^{(e)}, P\mathbf{y}_t^{(n)})$
end
Initialize $\hat{\mathbf{x}}_{1|1} = \mathbf{y}_1$, $K_{1|1} = I$
for $t = 1$ *to* T **do**
  `/* Prediction Step */`
  $\hat{\mathbf{x}}_{t+1|t} = B^{(l)}\hat{\mathbf{x}}_{t|t}$
  $K_{t+1|t} = B^{(l)}K_{t|t}B^{(l)\prime} + Q^{(l)} \quad Q^{(l)} = C^{(l)}C^{(l)\prime}$
  `/* Update Step */`
  $S_t = W_t E^{(l)} K_{t+1|t} E^{(l)\prime} W_t' + W_t \Lambda^{(l)} W_t'$
  $G_t = K_{t+1|t} E^{(l)\prime} W_t' S_t^{-1}$
  $\hat{\mathbf{x}}_{t+1|t+1} = \hat{\mathbf{x}}_{t+1|t} + G_t(W_t\mathbf{y}_{t+1} - W_t E^{(l)} \hat{\mathbf{x}}_{t+1|t})$
  $K_{t+1|t+1} = K_{t+1|t} - G_t W_t E^{(l)} K_{t+1|t} \quad \Lambda^{(l)} = F^{(l)} F^{(l)\prime}$
end

matrix changes according to $A \mapsto PAP^T$, and the representation $\mathbf{w}$ becomes $\phi(P(\phi^{-1}(\mathbf{w})P^T)$. We will use $P \star \mathbf{w}$ to denote this transformed $\mathbf{w}$. Together, the action of $\mathcal{P}_n$ on $\mathcal{X}_n$ is given by: $(P, X) = (P, (\mathbf{w}, \mathbf{v})) = (P \star \mathbf{w}, P\mathbf{v}) \in \mathcal{X}$. The graph space is then defined as the quotient space $\mathcal{G}_n \doteq \mathcal{X}_n / \mathcal{P}_n$. Elements of this quotient space are the permutation orbits of a graph: $[X] = \{(P \star \mathbf{w}, P\mathbf{v} | P \in \mathcal{P}_n\}$. The distance between any two graphs of size n is given by:

$$d_g([X^{(1)}], [X^{(2)}]) = \min_{P \in \mathcal{P}_n} d_x((\mathbf{w}^{(1)}, \mathbf{v}^{(1)}), (P \star \mathbf{w}^{(2)}, P\mathbf{v}^{(2)})) \ . \tag{2}$$

The optimization over $P \in \mathcal{P}_n$ is precisely the graph matching problem that has received a lot of attention in the literature [13,30,42]. We presently use the *Umeyama* approach laid out in [16] for this optimization and graph matching since it allows us to handle graphs of varying sizes seamlessly. The full graph space is given by the union: $\mathcal{G} = \cup_{n=1}^{\infty} \mathcal{G}_n$.

When comparing graphs of two different sizes, say n_1 and n_2, we introduce null nodes to make them size $n_1 + n_2$ each and then apply the above setup. Null nodes are used only for matching purposes and are assigned attributes carefully to reach a desired result (see [16] for details). A matching of a real node in one graph to a null node in the other graph implies a birth or a removal of a node from the representation.

Algorithm 2: Kalman Smoothing for Handling Missing or Noisy Input

Input : Estimated Graphs Sequence: $\hat{\mathbf{X}} = \{\hat{\mathbf{x}}_t : t = 1, 2, \ldots, T\}$
Output: Smoothed Graph Sequence: $\tilde{\mathbf{X}} = \{\tilde{\mathbf{x}}_t : t = 1, 2, \ldots, T\}$

```
for t = T − 1 to 1 do
    H_t = K_{t|t} B^{(l)} K^{-1}_{t+1|t}
    x̃_{t|T} = x̂_{t|t} + H_t (x̃_{t+1|T} − x̂_{t+1|t})
    K̃_{t|T} = K_{t|t} + H_t (K̃_{t+1|T} − K_{t+1|t}) H'_t
end
```

4 Bayesian Time-Series Analysis of Graphs

Problem Statement: In this work, we aim to model the evolution of graph structures using a time-series model and then estimate the true time-series from noisy data using Kalman smoothing [17]. Specifically, our goal is to estimate $\mathbf{X} = \{\mathbf{x}_t \in \mathcal{X} : t = 1, 2, \ldots, T\}$, given a possibly corrupted or noisy observations $\mathbf{Y} = \{\mathbf{y}_t : t = 1, 2, \ldots, T\}$. This also involves estimating model parameters $\Theta = \{B^{(l)}, C^{(l)}, E^{(l)}, F^{(l)} : l = e, n\}$ from the observed data. Here $l = e$ indexes the edges and $l = n$ indexes the nodes. To this end, we specify two models – one for capturing dynamics underlying the process generating graphs (S) and the other for the observations (O). Formally, we define these models as

$$\text{S}: \begin{cases} \mathbf{w}_{t+1} = & B^{(e)}\mathbf{w}_t + C^{(e)}\mathbf{u}_t^{(e)} & \in \mathbb{R}^{k_n \times p}, \\ \mathbf{v}_{t+1} = & B^{(n)}\mathbf{v}_t + C^{(n)}\mathbf{u}_t^{(n)} & \in \mathbb{R}^{n \times m} \end{cases} \tag{3}$$

$$\text{O}: \begin{cases} \mathbf{y}_{t+1}^{(e)} = & P_{t+1} \star (W_{t+1} E^{(e)} \mathbf{w}_{t+1}) + F^{(e)} \boldsymbol{\epsilon}_{t+1}^{(e)} & \in \mathbb{R}^{k_n \times p}, \\ \mathbf{y}_{t+1}^{(n)} = & P_{t+1} W_{t+1} E^{(n)} \mathbf{v}_{t+1} + F^{(n)} \boldsymbol{\epsilon}_{t+1}^{(n)} & \in \mathbb{R}^{n \times m} \end{cases} \tag{4}$$

where the superscripts (e) and (n) denote the models for edges and nodes respectively; $P_{t+1} \in \mathcal{P}_n$ is a random permutation matrix; B, C, E, F are unknown coefficient matrices that are estimated from the data. The quantities $\mathbf{u}_t^{(e)}, \mathbf{u}_t^{(n)}$ denote random perturbations in the system dynamics and are modeled with independent standard normal components. We assume the measurement noise $\boldsymbol{\epsilon}_t^{(e)}, \boldsymbol{\epsilon}_t^{(n)}$ to be multivariate normal with mean zero and identity covariances. Also, W_{t+1} is a matrix that encodes the dropping of actual nodes or adding false nodes; W_{t+1} is arbitrary but known for use in estimation and tracking. For brevity, we will use $\mathbf{x}_t = \{\mathbf{w}_t, \mathbf{v}_t\}$ to denote the underlying graph at time t and $\mathbf{y}_t = \{\mathbf{y}_t^{(e)}, \mathbf{y}_t^{(n)}\}$ as its noisy observation.

Since the dynamical model is a linear system with Gaussian distributions, the estimates are readily available in close form. The problem of estimating $\mathbf{X}$, given $\mathbf{Y}$ and Θ, can be solved using either the Kalman filter (forward pass) or smoother (forward and backward pass). The problem of finding Θ, given $\mathbf{Y}$ and $\mathbf{X}$, is relatively straightforward using maximum-likelihood estimation (MLE). Applying these two solutions iteratively, we apply a well-known EM-type algorithm for estimating $\mathbf{X}$ from $\mathbf{Y}$.

Algorithm 3: Parameter Estimation for the Kalman Smoother

Input : Observed and Estimated Graph Sequences: $\{\mathbf{Y}, \hat{\mathbf{X}}\}$
Output: Estimated Parameters: $\Theta = \{\hat{E}, \hat{\Lambda}, \hat{B}, \hat{Q}\}$

$\hat{E}^{(l)} = (\sum_{t=1}^{T} \mathbf{y}_t \mathbf{x}_t')(\sum_{t=1}^{T} \mathbf{x}_t \mathbf{x}_t')^{-1}$

$\hat{\Lambda}^{(l)} = \frac{1}{T-1} \sum_{t=1}^{T} (\mathbf{y}_t - \hat{E}^{(l)} \mathbf{x}_t)(\mathbf{y}_t - \hat{E}^{(l)} \mathbf{x}_t)'$

$\hat{B}^{(l)} = (\sum_{t=1}^{T} \mathbf{x}_t \mathbf{x}_{t-1}')(\sum_{t=1}^{T} \mathbf{x}_{t-1} \mathbf{x}_{t-1}')^{-1}$

$\hat{Q}^{(l)} = \frac{1}{T-1} \sum_{t=1}^{T} (\mathbf{x}_t - \hat{B}^{(l)} \mathbf{x}_{t-1})(\mathbf{x}_t - \hat{B}^{(l)} \mathbf{x}_{t-1})'$

Focusing on estimating the unknown state variable $\mathbf{X}$ using the data $\mathbf{Y}$ (and a current estimate of Θ) we seek the posterior distribution – $f(\mathbf{x}_{t+1} | \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_t, \mathbf{y}_{t+1})$. Given the linear-Gaussian nature of Eqns 3,4, the posterior distribution at each time is a multivariate normal distribution. To characterize this posterior, one needs only to evaluate the associated mean $\hat{\mathbf{x}}_{t+1|t+1}$ and covariance $\hat{K}_{t+1|t+1}$. Kalman filter provides a recursive formula to compute these quantities at time $t+1$, as a function of estimates at time t and the new data $\mathbf{y}_{t+1}$. We adopt and apply these expressions for the representation spaces of graphs and obtain filtering results. Algorithm 1 outlines this recursion. An important distinction from classical filtering is that not all nodes are visible in the data at all times. Sometimes some nodes are undetected, while other times, some extraneous nodes are added. To deal with this issue, we follow two steps. Firstly, we introduce a known matrix W_t whose rows are a subset of the rows of identity I depending on the elements that are missing in the observation $\mathbf{y}_t$. If no elements are missing in $\mathbf{y}_t$, then W_t is identity. Secondly, we utilize an optimization over $\mathcal{P}_n$, *i.e.,* graph matching, which registers the observed graphs at successive times. While Kalman Filter provides a forward pass through the temporal graph data, the results can be further improved by performing an additional backward pass. In this process, one uses the full data (over the full interval) to improve estimates. Algorithm 2 outlines steps for implementing this smoother.

Parameter Estimation. So far, we have discussed estimation of $\mathbf{X}$ given $\mathbf{Y}$ (the extracted graphs) and Θ (the system parameters). However, in the real data, Θ is unknown and needs to be estimated itself. Given $\mathbf{X}$ (or rather its estimate) and $\mathbf{Y}$, one can estimate Θ using a maximum-likelihood criterion that results in closed-form expressions. These expressions are given in Algorithm 3. To put these pieces together, one initializes $\hat{\Theta}$ and iterates between the estimation of $\mathbf{X}$ and Θ using Algorithms (1, 2) and 3. This results in the estimated values of the graph sequence and the system parameters, as well as the registration of nodes between successive graphs.

5 Experimental Evaluation

5.1 Data, Metrics and Baselines

Data. For evaluating the proposed approach, we devise three experimental settings with two kinds of data - synthetic and real-world video data. Since there are no existing comparable benchmarks, we develop a synthetic dataset for simulating the problem of predicting the future states of a visual-semantic sequence such as graphs [26] or spatial-temporal graphs [46]. We consider two scenarios - without and with observation noise/clutter. The former refers to a setting where each frame in a long video is devoid of any degradation, such as object detection failures. All objects are correctly detected at all times, and no new object enters the scene. The latter refers to a more realistic scenario, where objects may enter and/or exit the scene, and some may even go undetected in certain frames. We also demonstrate that our approach can work well with real-world scenarios by evaluating on a complex activity detection dataset with multiple actors from surveillance videos. Due to space constraints, we briefly describe the data in detail below. More details are presented in the supplementary.

Synthetic Data 1 - Data with Registered Nodes: For evaluating in the *ideal* observation setting, we generate an ordered sequence of fully connected graphs with n=10 nodes observed over t=5000 time points. The node attributes $\mathbf{v}_t$ denote the position coordinates of objects in the scene ($m = 2$) while the edge attributes $\mathbf{w}_t$ are scalar ($p = 1$) and signify relationships between connecting nodes. To simulate a graph time series, we initialize $\mathbf{v}_0$ randomly on a unit circle and $\mathbf{w_0}$ using a uniform distribution. We set the model parameters in the system dynamics (S) to: $B^{(e)} = 0.95I_{k_n}, Q^{(e)} = I_{k_n}, \Lambda^{(e)} = 0.01I_{k_n}, E^{(e)} = 0.001M + I_{k_n}$, $M \in \mathbb{R}^{k_n \times k_n}$ whose elements are i.i.d. standard normal, $B^{(n)} = I_{2n}$, $Q^{(n)} = 100I_{2n}$ $\Lambda^{(n)} = 10I_{2n}$ $E^{(n)} = 3M + I_{2n}$, $M \in \mathbb{R}^{2n \times 2n}$ whose elements are i.i.d, $\mathcal{N}(0, 0.01)$. Recall $k_n = n(n-1)/2$.

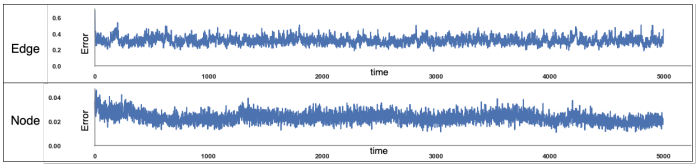

Fig. 2: Evolution of $\|\mathbf{w}_t - \mathbf{w}_{t+1}\|$ (top) and $\|\mathbf{v}_t - \mathbf{v}_{t+1}\|$ (bottom) versus t in a sample simulation.

Fig. 2 shows the $\mathbb{L}^2$ norm between successive edge weights $\|\mathbf{w}_t - \mathbf{w}_{t+1}\|$ (top) and the node attributes $\|\mathbf{v}_t - \mathbf{v}_{t+1}\|$ (bottom). The large values indicate a significant changes in graphs over times. Similarly, Fig. 3 shows $\|\mathbf{w}_t - \mathbf{y}_t^{e)}\|$ (top) and $\|\mathbf{v}_t - \mathbf{y}_t^{n)}\|$ (bottom) over time. Once again, large values indicate a large level of noise in the observations. All baselines are trained with the first 4500 graphs and evaluated on the last 500 graphs.

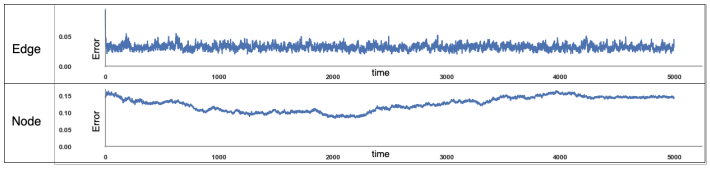

Fig. 3: Evolution of $\|\mathbf{w}_t - \mathbf{y}_t^{e)}\|$ (top) and $\|\mathbf{v}_t - \mathbf{y}_t^{n)}\|$ (bottom) in a sample run.

Synthetic Data 2 - Data with Missed and False Nodes: For creating second synthetic data with missing or false nodes, we start as above with graphs with n=10 that are fully connected for t up to 5000. Then we randomly select some time points and randomly remove 2−3 of the nodes in the graphs at each of the selected instants. This modification simulates missed detections and/or the exit and entry of objects as time progresses. To aid the registration process and simulate feature embeddings for objects in a video clip, we define the node attributes here to be the position coordinates concatenated with a simulated feature vector that is distinct for each node.

Real-world Video Data. For evaluation with real-world video data, we use a subset of the Multiview Extended Video with Activities (MEVA) [12] dataset, a large-scale human activity detection benchmark. It consists of over 9300 hours of scripted scenarios and spontaneous background activities from indoor and outdoor viewpoints. For our experiments, we use a scene from an indoor bus station where the actors are continuously moving in and out of the camera field of view while performing different activities. The bus station scene is 5 minutes long (9000 frames). Each frame is represented by a graph, where each node is a person detection, and their node embeddings represent visual features. Specifically, we apply a pretrained object detector (DeTR [8]) on each frame and characterize the interactions and activities in the scene as an undirected graph. Each node $v_i \in V$ denotes a unique detection, where the node embeddings are the features vector extracted from the last layer of the transformer decoder. The edges are constructed using relative distances between detections after the homography transformation matrix transforms the input to a top view.

Metrics. We use the $\mathbb{L}^2$ norm of the difference between the predicted and actual system node and edge vectors to evaluate the quality of the predicted graphs. This metric is used in defining the components of d_x in Eqn. 1 and is a natural choice of quantifying estimation performance.

Baselines. For a fair comparison with prior work on time series prediction, we devise three types of models: (i) multi-epoch training models, (ii) online training models, and (iii) static or reactive prediction models. For multi-epoch training models, we use the following baselines as SOTA techniques: 1-step feed-forward network (FFN) [14], recurrent neural networks [39,18], gated recurrent units (GRU) [9], transformers [40], seq2seq [38] with RNNs and GRUs. For online training models, we train an RNN and LSTM model in online fashion, *i.e.*, one graph at a time for 1-step prediction trained for 1 epoch continually. For static predictors, we predict either (i) the previous observation, (ii) the first

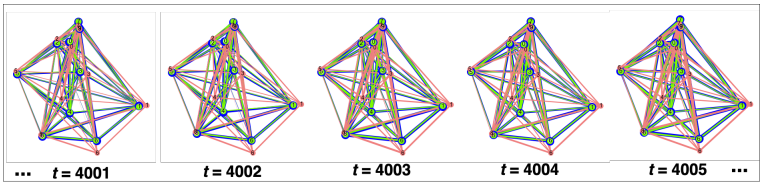

Fig. 4: For each t, we show $\mathbf{x}_t$ (blue), $\mathbf{y}_t$ (coral), and $\hat{\mathbf{x}}_t$ (green). The thickness of the edges are proportional to the edge values.

observation, or (iii) the median of three successive graphs (current, previous, and the next). (For median filtering, we perform node registration using the well-known Hungarian algorithm.) Hyperparameters were chosen using a grid search and optimized using the Adam optimizer.

5.2 Evaluation on Synthetic Data 1

We apply Algorithm (1,2) and 3 to the simulated data and obtain estimates $\hat{\mathbf{X}}$ from the given data $\mathbf{Y}$. Figure 4 shows an illustrative example of the estimation results. It shows five timeframes of system dynamics (blue) and its noisy observations (coral). The observed graphs are slightly distorted versions of the original ones with respect to both edge weights and node positions. Next, we run several iterations of Kalman smoother, and the estimated graph sequence is shown in green color.

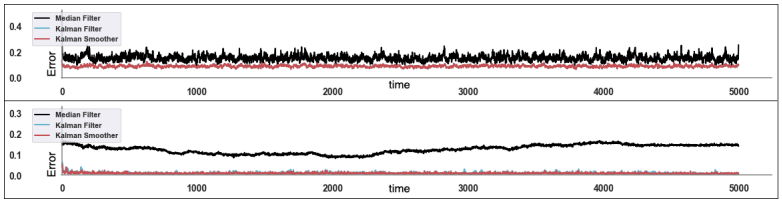

Fig. 5: Plots of $\|\mathbf{x}_t - \hat{\mathbf{x}}_t\|$ versus t for the edges (top) and nodes (bottom) for Synthetic Dataset 1.

To visualize estimation error over the full interval, we present error plots in Figure 5. Here, the y-axis represent $\|\mathbf{w}_t - \hat{\mathbf{w}}_t\|$ and $\|\mathbf{v}_t - \hat{\mathbf{v}}_t\|$ for (a) Edges and (b) Nodes respectively while the x-axis represents time t. For comparison, we observe that the Kalman smoother (red) estimation performs slightly better than that obtained from Kalman filter (only forward pass) (blue) and much better than the median filter (black).

Finally, we compute the performance summary over multiple simulated time series and compare our method with alternative ideas described earlier (under Baselines). These results are presented in Table 1. As the table shows, the estima-

Approach	Synthetic Dataset 1		Synthetic Dataset 2	
	Prediction Errors		Prediction Errors	
	Nodes	Edges	Nodes	Edges
Multi-Epoch Training				
1-Step FFN	0.623 ± 0.0555	1.107 ± 0.084	0.198 ± 0.014	0.766 ± 0.090
RNN	0.303 ± 0.0239	0.496 ± 0.061	0.165 ± 0.019	0.472 ± 0.067
GRU	0.402 ± 0.0264	0.575 ± 0.062	0.164 ± 0.010	0.493 ± 0.076
Seq2Seq-RNN	0.470 ± 0.0274	0.747 ± 0.069	0.199 ± 0.025	0.771 ± 0.091
Seq2Seq-GRU	0.521 ± 0.0343	0.690 ± 0.074	0.165 ± 0.015	0.518 ± 0.064
Transformer	0.769 ± 0.0145	1.003 ± 0.015	0.284 ± 0.016	1.032 ± 0.030
Online Training				
RNN	0.465 ± 0.089	0.932 ± 0.103	0.215 ± 0.031	0.876 ± 0.082
GRU	0.460 ± 0.065	0.877 ± 0.097	0.225 ± 0.033	0.847 ± 0.090
Kalman Filter	$\mathbf{0.009 \pm 0.003}$	$\mathbf{0.091 \pm 0.007}$	$\mathbf{0.008 \pm 0.004}$	$\mathbf{0.084 \pm 0.026}$
Kalman Smoother	$\mathbf{0.008 \pm 0.002}$	$\mathbf{0.091 \pm 0.007}$	$\mathbf{0.006 \pm 0.003}$	$\mathbf{0.085 \pm 0.020}$
Static Prediction				
Median Filter	0.146 ± 0.009	0.153 ± 0.025	0.123 ± 0.002	0.156 ± 0.032
Previous observation	0.998 ± 0.000	0.713 ± 0.015	0.999 ± 0.000	0.712 ± 0.022
First observation	0.998 ± 0.000	1.017 ± 0.056	0.999 ± 0.000	1.038 ± 0.056

Table 1: Quantitative evaluation on Synthetic Datasets 1 and 2. Table reports mean and std. deviation of the $\mathbb{L}^2$ errors, for edge and node estimates.

tion errors are substantially lower for the Kalman inference relative to baselines using multi-epoch training, online training, or simply using a static prediction.

5.3 Evaluation on Synthetic Data 2

In this experiment, we consider graph data with missing and spurious nodes inserted randomly at an arbitrary time. Consequently, one needs a *graph matching* step to match and track nodes across times. Every observed graph $\mathbf{y}_t$ is registered to the one before it, as stated in Algorithm 1. We use the *Umeyama* algorithm involving both edge and node attributes described in [16] for this purpose. The simulated graph time series after registration is shown in Figure 6 at five consecutive time points. At t=239, $\mathbf{x}_t$ has 10 nodes whereas $\mathbf{y}_t$ has 7 connected nodes and 3 more that are not connected to the rest of the graph. The latter three are *null nodes* which were added to $\mathbf{y}_{239}$ to facilitate the registration process and have degree 0. The graphs at t=239 represent a situation where three objects went undetected. The *null nodes* are assigned the position coordinates of the corresponding nodes they were matched to in the previous time point.

Figure 6 shows the estimated graphs in green. The *null nodes* that appear isolated in $\mathbf{y}_{239}$ are connected in $\hat{\mathbf{x}}_{239}$ by weighted edges estimated by the Kalman smoother. The error plots in Fig 7 show high accuracy for the Kalman smoother. The spikes here correspond to errors at time points with missed or false nodes. In spite of these spikes, the Kalman smoother (red) performs better than the Kalman filter (only forward pass) (blue) and the median filter (black). This result emphasizes that the graph Kalman filter-smoother framework can detect-track objects even when missing in some frames. Table 1 presents a more comprehen-

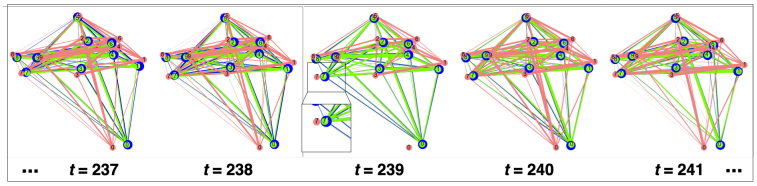

Fig. 6: Illustration of handling missed object detection. (Same color scheme as earlier.)

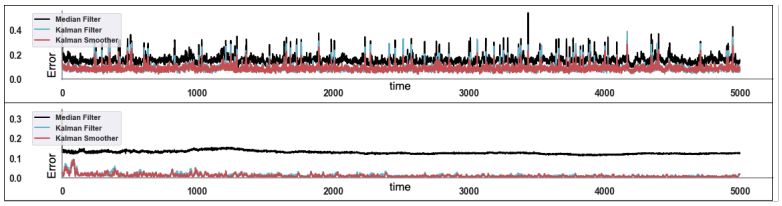

Fig. 7: Plots of the error $\|\mathbf{x}_t - \hat{\mathbf{x}}_t\|$ versus t for edges (top) and nodes (bottom) for Synthetic Dataset 2.

sive study of estimation error for a number of alternative methods. Once again, we see a clear superiority of Kalman smoother over other methods.

5.4 Real-world Evaluation

Next, we apply this framework to tracking human subjects in a bus station scene introduced earlier. To focus on evaluation, we consider three short segments of the full video sequence. Fig 8 shows 4 frames from an event where a subject enters the bus station and eventually exits it. To compare estimation errors for

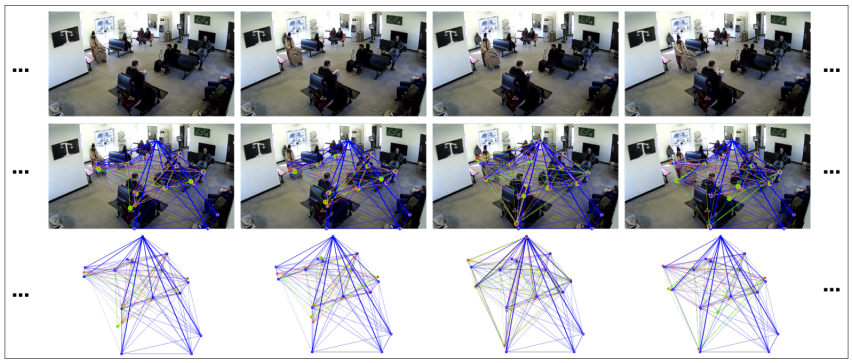

Fig. 8: Top: Four frames (one-second spacing) showing a subject entering. Middle: $\mathbf{x}_t$, $\mathbf{y}_t$, $\hat{\mathbf{x}}_t$ overlaid. Bottom: these graphs alone.

different methods, we have manually generated the *ground truth* $\mathbf{X}$. It represents each manually detected subjects in each frame of the sequence. This manual annotation results in bounding boxes of true subjects using the Davinci Resolve video editor. This software allows for manually annotating some keyframes and linearly interpolating the bounding boxes (in positions and aspect ratios) to fill in between. We treat the resulting sequence as the ground truth $\mathbf{X}$.

To aid the registration of the observed series $\{\mathbf{y}_t\}$, we combine several types of node attributes: (a) top-view node position coordinates obtained from homography transformation of the input camera view, (b) eight principal components of the 256 length node embeddings explaining 85.3% of the variation, and (c) four coordinates of the bounding box of the detected object, resulting in a $m = 14$-length attribute vector for each node. The edge weights in this setup come from Euclidean distances between the nodes.

With this setup, we apply Algorithms (1,2) and 3 to get estimates of the underlying system graphs $\hat{\mathbf{X}}$ as well as the model parameters $\hat{\Theta}$. Note that results are sensitive to initialization of the system parameters. For each video clip, we measure the estimation error using three items: (a)$\|\mathbf{x}_t - \hat{\mathbf{x}}_t\|$, (b) the average count of noisy or spurious detections (present in only $\mathbf{y}_t$) and (c) the average count of missed detections (present in only $\mathbf{x}_t$). The $\mathbb{L}^2$ norm estimation errors in Fig 9 indicate consistently high prediction accuracy for Kalman smoother. The

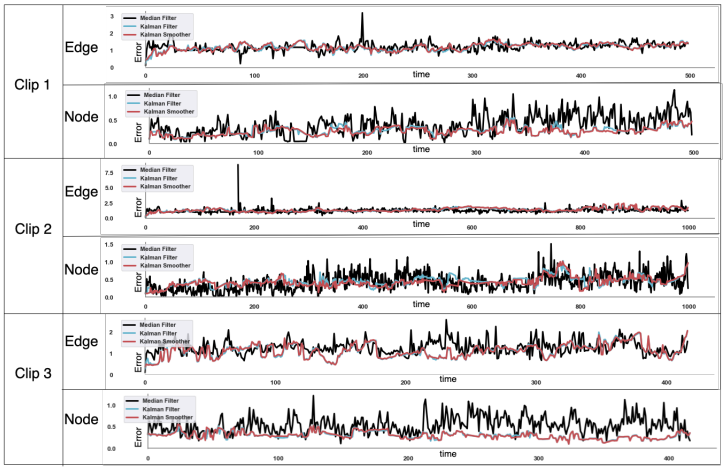

Fig. 9: Plots of the error $\|\mathbf{x}_t - \hat{\mathbf{x}}_t\|$ for edges and nodes in three video clips from *real-world* observations (MEVA dataset).

average count of false and missed detections in Table 2 shows that the Kalman filter-smoother framework performs better at eliminating noise and retrieving missed detections. Fig 10 highlights some key features of the results obtained.

Video Clips	Rate of False Detections			Rate of Missed Detections		
	KS	KF	MF	KS	KF	MF
clip 1 ($t = 0$ to 500)	0.762	0.721	4.118	2.427	2.387	5.914
clip 2 ($t = 0$ to 1000)	0.956	0.997	4.498	3.024	3.065	6.738
clip 3 ($t = 782$ to 1200)	0.885	0.909	4.690	3.556	3.580	7.599

Table 2: Average count of nodes present in $\hat{\mathbf{x}}_t$ but not in $\mathbf{x}_t$ (left) and average count of nodes present in $\mathbf{x}_t$ but not in $\hat{\mathbf{x}}_t$ (right) for 3 video clips using Kalman smoother (KS), Kalman filter (KF) and median filter (MF)

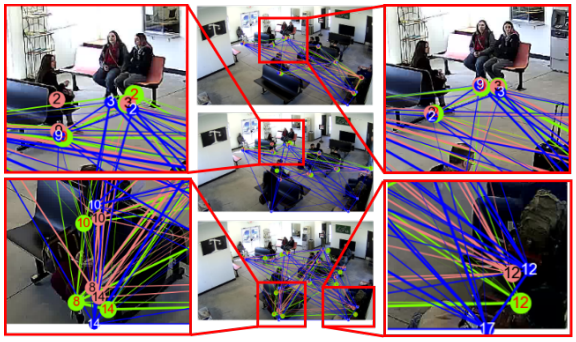

Fig. 10: Examples of successful (top) and failed (bottom) node placements given incorrect data: node 2 (top left), node 9 (top right), node 8 (bottom left) and node 17 (bottom right).

6 Discussion

This paper develops a Kalman smoother approach, adapted to the quotient space geometry of graph representations, to track objects of interest in video data. Specifically, it incorporates a node-registration step at each time to maintain tracks despite having noise and clutter in the scene. Utilizing the dynamics of system evolution, along with a likelihood model, this Bayesian framework helps overcome the issues raised by missed and false detections in the pre-processing step. This framework understandably outperforms several learning-based and some basic online ideas. To the best of our knowledge, this is the first paper to formulate and apply classical filtering techniques to quotient spaces of graphs. While this approach can handle a small number of missed or false detections, it fails when this number grows large. The system dynamics, or temporal smoothing, of estimated states, can't make up for this large data corruption. One needs additional information to help track such objects.

Acknowledgements. This research was supported in part by the US National Science Foundation grants 1955154, IIS 2143150, IIS 1955230, CNS 1513126, and IIS 1956050.

References

1. Aakur, S., de Souza, F.D., Sarkar, S.: Going deeper with semantics: Video activity interpretation using semantic contextualization. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 190–199. IEEE (2019)
2. Aakur, S.N., de Souza, F.D.M., Sarkar, S.: Generating open world descriptions of video using common sense knowledge in a pattern theory framework. Quarterly of Applied Mathematics (2019)
3. Adeli, V., Ehsanpour, M., Reid, I.D., Niebles, J.C., Savarese, S., Adeli, E., Rezatofighi, H.: TRiPOD: Human trajectory and pose dynamics forecasting in the wild. CoRR **abs/2104.04029** (2021), `https://arxiv.org/abs/2104.04029`
4. Brasó, G., Leal-Taixé, L.: Learning a neural solver for multiple object tracking. CoRR **abs/1912.07515** (2019), `http://arxiv.org/abs/1912.07515`
5. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond Euclidean data. IEEE Signal Processing Magazine **34**(4), 18–42 (2017)
6. Calissano, A., Feragen, A., Vantini, S.: Populations of unlabeled networks: Graph space geometry and geodesic principal components (2020)
7. Cao, D., Wang, Y., Duan, J., Zhang, C., Zhu, X., Huang, C., Tong, Y., Xu, B., Bai, J., Tong, J., et al.: Spectral temporal graph neural network for multivariate time-series forecasting. Advances in Neural Information Processing Systems **33**, 17766–17778 (2020)
8. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European Conference on Computer Vision. pp. 213–229. Springer (2020)
9. Che, Z., Purushotham, S., Cho, K., Sontag, D., Liu, Y.: Recurrent neural networks for multivariate time series with missing values. Scientific Reports **8**(6085) (2018)
10. Chen, F., Chen, Z., Biswas, S., Lei, S., Ramakrishnan, N., Lu, C.T.: Graph convolutional networks with kalman filtering for traffic prediction. In: In 28th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '20) (2020)
11. Cheng, D., Yang, F., Xiang, S., Liu, J.: Financial time series forecasting with multi-modality graph neural network. Pattern Recognition **121**, 108218 (2022)
12. Corona, K., Osterdahl, K., Collins, R., Hoogs, A.: MEVA: A large-scale multiview, multimodal video dataset for activity detection. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 1060–1068 (January 2021)
13. Gold, S., Rangarajan, A.: A graduated assignment algorithm for graph matching. IEEE Transactions on pattern analysis and machine intelligence **18**(4), 377–388 (1996)
14. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), `http://www.deeplearningbook.org`
15. Guo, X., Bal, A.B., Needham, T., Srivastava, A.: Statistical shape analysis of brain arterial networks (BAN). Annals of Applied Statistics **16**(2), 1130–1150 (2022)
16. Guo, X., Srivastava, A., Sarkar, S.: A quotient space formulation for statistical analysis of graphical data. Journal of Mathematical Imaging and Vision **63**, 735–752 (March 2021)
17. Haykin, S.: Kalman filtering and neural networks, vol. 47. John Wiley & Sons (2004)

18. Hewamalage, H., Bergmeir, C., Bandara, K.: Recurrent neural networks for time series forecasting: Current status and future directions. International Journal of Forecasting **37**(1), 388–427 (2021)
19. Huang, Y., Bi, H., Li, Z., Mao, T., Wang, Z.: Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6272–6281 (2019)
20. Ivanovic, B., Pavone, M.: The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2375–2384 (2019)
21. Jain, B.J.: On the geometry of graph spaces. Discrete Applied Mathematics **214**, 126–144 (2016)
22. Jain, B.J.: Statistical graph space analysis. Pattern Recognition **60**, 802–812 (2016)
23. Ji, J., Krishna, R., Fei-Fei, L., Niebles, J.C.: Action genome: Actions as compositions of spatio-temporal scene graphs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10236–10247 (2020)
24. Knyazev, A., Malyshev, A.: Accelerated graph-based nonlinear denoising filters. Procedia Computer Science **80**, 607–616 (2016)
25. Kosaraju, V., Sadeghian, A., Martín-Martín, R., Reid, I., Rezatofighi, S.H., Savarese, S.: Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. arXiv preprint arXiv:1907.03395 (2019)
26. Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.J., Shamma, D.A., et al.: Visual genome: Connecting language and vision using crowdsourced dense image annotations. International Journal of Computer Vision **123**(1), 32–73 (2017)
27. Li, J., Gao, X., Jiang, T.: Graph networks for multiple object tracking. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) (March 2020)
28. Liu, H., Singh, P.: Conceptnet—a practical commonsense reasoning tool-kit. BT Technology Journal **22**(4), 211–226 (2004)
29. Lu, X., Wang, W., Danelljan, M., Zhou, T., Shen, J., Gool, L.V.: Video object segmentation with episodic graph memory networks. CoRR **abs/2007.07020** (2020), `https://arxiv.org/abs/2007.07020`
30. Lyzinski, V., Fishkind, D.E., Fiori, M., Vogelstein, J.T., Priebe, C.E., Sapiro, G.: Graph matching: Relax at your own risk. IEEE transactions on pattern analysis and machine intelligence **38**(1), 60–73 (2016)
31. Mohamed, A., Qian, K., Elhoseiny, M., Claudel, C.: Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14424–14432 (2020)
32. Paaßen, B., Göpfert, C., Hammer, B.: Time series prediction for graphs in kernel and dissimilarity spaces. Neural Processing Letters **48**(2), 669–689 (2018)
33. Rudi, A., Ciliberto, C., Marconi, G., Rosasco, L.: Manifold structured prediction. Advances in Neural Information Processing Systems **31** (2018)
34. Salzmann, T., Ivanovic, B., Chakravarty, P., Pavone, M.: Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16. pp. 683–700. Springer (2020)
35. Shi, L.: Kalman filtering over graphs: Theory and applications. IEEE Transactions on Automatic Control **54**(9), 2230–2234 (2009)

36. Song, C., Lin, Y., Guo, S., Wan, H.: Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 914–921 (2020)
37. Speer, R., Chin, J., Havasi, C.: Conceptnet 5.5: An open multilingual graph of general knowledge. In: Thirty-first AAAI conference on artificial intelligence (2017)
38. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. CoRR **arXiv:1409.3215** (2014)
39. Tealab, A.: Time series forecasting using artificial neural networks methodologies: A systematic review. Future Computing and Informatics Journal **3**(2), 334–340
40. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. arXiv:1706.03762 (2017)
41. Vázquez-Enríquez, M., Alba-Castro, J.L., Docío-Fernández, L., Rodríguez-Banga, E.: Isolated sign language recognition with multi-scale spatial-temporal graph convolutional networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3462–3471 (2021)
42. Vogelstein, J.T., Conroy, J.M., Lyzinski, V., Podrazik, L.J., Kratzer, S.G., Harley, E.T., Fishkind, D.E., Vogelstein, R.J., Priebe, C.E.: Fast approximate quadratic programming for graph matching. PLOS one **10**(4), e0121002 (2015)
43. Wang, C., Gao, D., Qiu, Y., Scherer, S.: Lifelong graph learning. In: 2022 Conference on Computer Vision and Pattern Recognition (CVPR) (2022)
44. Wang, C., Cai, S., Tan, G.: Graphtcn: Spatio-temporal interaction modeling for human trajectory prediction. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 3450–3459 (2021)
45. Wang, W., Lu, X., Shen, J., Crandall, D.J., Shao, L.: Zero-shot video object segmentation via attentive graph neural networks. CoRR **abs/2001.06807** (2020), `https://arxiv.org/abs/2001.06807`
46. Wang, X., Gupta, A.: Videos as space-time region graphs. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 399–417 (2018)
47. Wang, Y., Kitani, K., Weng, X.: Joint object detection and multi-object tracking with graph neural networks. In: 2021 IEEE International Conference on Robotics and Automation (ICRA). pp. 13708–13715. IEEE (2021)
48. Weng, X., Wang, Y., Man, Y., Kitani, K.M.: Gnn3dmot: Graph neural network for 3d multi-object tracking with 2d-3d multi-feature learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6499–6508 (2020)
49. Wu, Z., Pan, S., Long, G., Jiang, J., Chang, X., Zhang, C.: Connecting the dots: Multivariate time series forecasting with graph neural networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 753–763 (2020)
50. Yan, S., Xiong, Y., Lin, D.: Spatial temporal graph convolutional networks for skeleton-based action recognition. In: Thirty-second AAAI Conference on Artificial Intelligence (2018)
51. Yu, B., Yin, H., Zhu, Z.: Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In: IJCAI (2018)