Learning Spatiotemporal Graphs of Human Activities

William Brendel and Sinisa Todorovic
Oregon State University, Corvallis, OR 97331, USA,
brendelw@onid.orst.edu, sinisa@eecs.oregonstate.edu

Abstract

Complex human activities occurring in videos can be defined in terms of temporal configurations of primitive actions. Prior work typically hand-picks the primitives, their total number, and temporal relations (e.g., allow only followed-by), and then only estimates their relative significance for activity recognition. We advance prior work by learning what activity parts and their spatiotemporal relations should be captured to represent the activity, and how relevant they are for enabling efficient inference in realistic videos. We represent videos by spatiotemporal graphs, where nodes correspond to multiscale video segments, and edges capture their hierarchical, temporal, and spatial relationships. Access to video segments is provided by our new, multiscale segmenter. Given a set of training spatiotemporal graphs, we learn their archetype graph, and pdf's associated with model nodes and edges. The model adaptively learns from data relevant video segments and their relations, addressing the "what" and "how." Inference and learning are formulated within the same framework – that of a robust, least-squares optimization – which is invariant to arbitrary permutations of nodes in spatiotemporal graphs. The model is used for parsing new videos in terms of detecting and localizing relevant activity parts. We outperform the state of the art on benchmark Olympic and UT human-interaction datasets, under a favorable complexity-vs.-accuracy trade-off.

1. Introduction

Interpreting videos that show interesting, complex human activities (e.g., human interactions, sports) typically requires reasoning about the spatiotemporal arrangement of relevant activity parts at multiple scales. Without such a reasoning these activities could be easily confused, since they generally have very similar photometric and motion properties, at both local and global scales. For example, two videos of the *long jump* and *triple jump* might be easily confused if we do not account for their different temporal configurations, since both share *running*, *hopping*, and *jumping*. Also, without considering multiple scales, it would be difficult to distinguish between the *3-man-block* and *short-seam-pass* in volleyball, since they share the same primitive actions of each player (e.g., jumping, spiking) and the same temporal ordering of the primitives, at the finest scale, but differ in interactions between the players, at a coarser scale.

To enable such reasoning, graphical models have been used with great success to concisely capture the *structure* of an activity in terms of the hierarchy and spatiotemporal arrangement of its subactivities [1]. For example, activity structure has been modeled by HMMs [26], dynamic Bayesian nets [25], prototype trees [13], spatiotemporal graphs [14], context-free (AND-OR) grammars [10, 9], CRFs [16], and compilations of first-order logic to graphical models [24, 3]. These approaches, however, typically define an activity in terms of pre-selected primitive actions, and manually specify their space-time relationships. A few methods learn relevant activity parts from data (e.g., [16]), but they fix their total number, and allow only the relation followed-by. More formally, they typically pre-specify the number of random variables (nodes) representing primitive actions, and their statistical dependences (edges), referred to as the model structure. Due to this heuristic model specification, in training, significant resources could be wasted on learning hand-picked parts and relations which may not be the most relevant for representing and recognizing the activity. These issues have recently been addressed by learning relevant contextual relations between individual actions of people in a group activity [11]. However, their model encodes a fixed number of primitive actions.

In this paper, we seek to learn **what** activity parts and their spatiotemporal relations should be captured to represent complex human activities, and **how** relevant they are for enabling efficient inference in realistic videos. This advances prior work that typically ignores the "what" question. The goal of our learning is twofold. We learn the structure of the activity model, and the pdf's associated with nodes and edges of the model. This model is then used for **parsing** new videos in terms of localizing relevant activity parts, present at multiple scales.

To address the "what", we partition training videos of

a given activity class into spatiotemporal tubes at multiple scales, and then discover similar and frequently repeating tube configurations. The tubes provide rich visual cues for recognition, since they represent spatiotemporal extents of moving objects in the video. Different hierarchical, spatial, and temporal relationships can be easily established between 2D+t tubes, as illustrated in Fig. 1. For example, a tube occupied by a basketball player holding the ball can be viewed as a parent of two children tubes corresponding to the player and the ball. Also, temporal relations (e.g., before, overlap) can be easily encoded between the tubes.

The goal of our learning will be to identify the most relevant tubes and their most relevant relations for representing the activity. We expect that videos of the same class will give rise to many similar configurations of similar 2D+t tubes. The other tubes are likely to belong to random background video parts. The discovered similar tube configurations will define the structure of our activity model, and allow for robust probabilistic learning of pdf's associated with nodes and edges of our model.

When a new image is encountered, its spatiotemporal graph is parsed using the learned activity model. To account for potential errors in extracting video tubes, our inference is made invariant to arbitrary permutations of nodes in spatiotemporal graphs. We formulate both inference and learning within a unified framework. This has many advantages, including that we ensure a principled handling of the "what," since our activity model is learned using the same objective as that employed for video parsing.

Related work – Volumetric video representations have been used for exemplar based activity recognition [8, 13]. We are not aware of any work on learning their structural model. Learning the model structure has been addressed in the context of Bayesian networks [6]. These methods commonly use greedy searches that incrementally modify the model structure by adding (or deleting) and appropriately re-connecting random variables. They typically make the restrictive assumption that correspondences between data and random variables of the model are given. The problem that we address here is more difficult, because our video representations are graphs with unknown correspondences to the nodes and edges of our graphical model. These correspondences must be estimated by graph matching. The most related to ours is recent work on 2D shape recognition based on learning structural archetypes of graphs – such as, e.g., super-graph [4], mixture of trees [23], and generative Delaunay graph [22]. These approaches typically make restrictive assumptions (e.g., model edges are independent and have Bernoulli distribution [22]), and cannot handle weights associated with both nodes and edges. We also extend the tree-union models for object recognition and texture analysis, presented in [21, 2], by accommodating arbitrary permutations of nodes in our spatiotemporal graphs.

Contributions – To our knowledge, this is the first volumetric-based approach to activity recognition that seeks to learn the structure and pdf's of activities from videos. We formulate learning and inference within a unified framework, that of a robust least-squares optimization, and show that it can be reduced to the quadratic assignment problem (QAP). Finally, we present a new, fast, multiscale approach to spatiotemporal video segmentation.

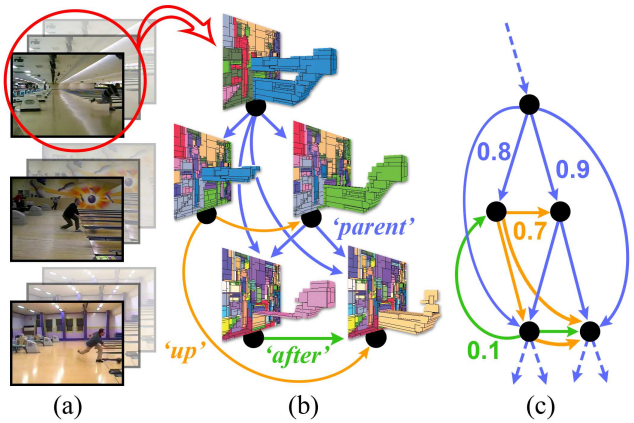

Figure 1. Our Steps 1–2: (a) Videos are represented by spatiotemporal graphs. (b) Nodes represent 2D+t tubes, and directed edges capture hierarchical, temporal, and spatial relationships between the tubes. (c) The video graphs are used to learn a graph model.

In the sequel, Sec. 2 gives an overview of our approach; Sec. 3–5 formalize the graph model and its learning and inference; Sec. 6 specifies the spatiotemporal segmentation; and Sec. 7 presents our experimental evaluation.

2. Overview

Our approach consists of three steps, illustrated in Fig. 1.

Step 1: Feature extraction. Given a video, we use a multiscale, spatiotemporal segmentation to obtain homogeneous subvolumes (tubes) of the video's space-time (2D+t) volume. Homogeneity is defined in terms of both pixel intensity and motion properties, at multiple scales. The resulting tubes are organized in a weighted directed graph, referred to as spatiotemporal graph. Nodes represent the tubes, and directed edges encode their three types of relationships. Hierarchical (ascendant-descendant) edges capture the nesting of smaller tubes within larger ones. Temporal edges represent the Allen's relations between time intervals associated with the tubes (e.g., before, meet). Spatial edges capture spatial layout relations between the tubes (e.g., left, up). The number of nodes and their connectivity, referred to as graph structure, are data-driven. The graph structure is specified by three distinct adjacency matrices, one for each edge type. Weights are associated with both nodes and edges. Node weights (descriptors) capture photometric and motion properties of the corresponding tubes.

Edge weights encode the strength of the corresponding hierarchical, temporal, and spatial relationships.

Step 2: Learning. Given a set of training spatiotemporal graphs of an activity class, we learn their weighted least-squares graph model. The model has three distinct adjacency matrices, one per each edge type. Each is learned as the closest matrix to the corresponding adjacency matrices of training graphs, in the weighted least-squares sense, under a matrix permutation. The weighted least squares is also used to learn descriptor vectors associated with model nodes, under a permutation of nodes in training graphs. Our formulation addresses the well-known instability of low-level segmentation algorithms, including ours, producing spurious nodes and edges in the spatiotemporal graphs.

Step 3: Recognition and Segmentation. A new video is represented by the spatiotemporal graph. The video is parsed by matching its graph with the closest activity model in the weighted least squares sense, under a arbitrary permutation of their adjacency matrices.

3. The Graph Model

This section specifies our graph model. We begin by introducing some notation. A weighted directed graph is a tuple $G = (V, \{\mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_L\}, \mathbf{F})$, where V is a set of nodes, $n = |V|$, and $\mathbf{F}$ is a matrix of d-dimensional descriptors associated with n nodes, $\mathbf{F} \in [0,1]^{n \times d}$. The set of $n \times n$ adjacency matrices $\{\mathbf{A}_l : \mathbf{A}_l \in [0,1]^{n \times n},\ l = 1, ..., L\}$, compactly represent L types of directed edges in G. For example, $(\mathbf{A}_l)_{ij} = 0.7$ means that nodes i and j are in relationship l (e.g., "ascendant-descendant"), with strength 0.7.

The graph model, $\mathcal{G} = (\mathcal{V}, \{\mathcal{A}_l : l = 1, ..., L\}, \mathcal{F})$, can be learned from a given set of graphs $\{G_k : k = 1, ..., K\}$. We expect that only a subset of nodes and edges in $\{G_k\}$ are relevant. We assume that each $\mathcal{A}_l$ is $m \times m$ matrix, where $m = \max_k n_k$. To learn $\mathcal{G}$, we need to find correspondences between nodes and edges of $\{G_k\}$ and $\mathcal{G}$. These correspondences can be encoded by the permutation matrix, $\mathbf{P} \in \{0,1\}^{n \times m}$, which has exactly one 1 in each row, and each of n columns ($n \leq m$), and 0's elsewhere. Multiplying $\mathbf{P}$ with $\mathcal{A}_l$ produces a permutation in the rows and columns of $\mathcal{A}_l$. This leads to a generative probabilistic model:

$$\forall l,\ \mathbf{A}_l = \mathbf{P}\mathcal{A}_l\mathbf{P}^{\mathrm{T}} + \eta_{\mathcal{A}_l}, \qquad \mathbf{F} = \mathbf{P}\mathcal{F} + \eta_{\mathcal{F}}, \quad (1)$$

where $\eta_{\mathcal{A}_l}$ and $\eta_{\mathcal{F}}$ are stochastic, capturing natural variations of the activity class. We define $\eta_{\mathcal{A}_l}$ and $\eta_{\mathcal{F}}$ as zero-mean Gaussian noise.

Note that (1) defines the generative process of sampling activity occurrences. Similar to HMMs, the generative sampling traverses through different types of spatiotemporal and hierarchical relationships, $\mathcal{A}$, and mixes them in the resulting instance, $\mathbf{A}$. As different activity styles can be defined by different graph connectivities, our model is capable of encoding many different styles and natural variations.

Also note that we do not require one-to-one correspondence between the model and instances. Since the adjacency matrices $\{\mathbf{A}_l\}$ of video graphs are smaller than the matrices $\{\mathcal{A}_l\}$ of the model, the permutation matrix $\mathbf{P}$ must have many all-zero columns. Therefore, many of the model nodes and edges are not matched to an instance graph.

4. Learning

Our goal is to learn the model, $\mathcal{G} = (\{\mathcal{A}_l\}, \mathcal{F})$, from K training graphs, $\{G_k = (\{\mathbf{A}_{kl}\}, \mathbf{F}_k) : k = 1, ..., K\}$, so as to minimize $\mathcal{G}$'s variance under permutations $\{\mathbf{P}_k : k = 1, ..., K\}$. This formulation naturally lends itself to the least-squares optimization.

We expect that our training graphs will be characterized by many spurious nodes and edges. Since the least squares is sensitive to outliers, we here use the weighted least squares, which estimates and downweights outliers in the training data. Let $\mathcal{W}_{\mathcal{F}}$ denote an $m \times d$ matrix of node weights which estimate if the corresponding nodes of training graphs are outliers. Also, let $\{\mathcal{W}_{\mathcal{A}_l} : l = 1, ..., L\}$ denote $m \times m$ matrices of edge weights estimating if the corresponding edges of training graphs are outliers. Then, learning can be formulated as

$$\begin{aligned} \min_{\substack{\{\mathcal{A}_l, \mathcal{W}_{\mathcal{A}_l}\} \\ \mathcal{F}, \mathcal{W}_{\mathcal{F}}, \{\mathbf{P}_k\}}} & \sum_{k=1}^{K} \left(\frac{\alpha}{2} \sum_{l=1}^{L} \left\| \mathcal{W}_{\mathcal{A}_l} \circ (\mathbf{P}_k^{\mathrm{T}} \mathbf{A}_{kl} \mathbf{P}_k - \mathcal{A}_l) \right\|_2^2 \right. \\ & \left. + \frac{(1-\alpha)}{2} \left\| \mathcal{W}_{\mathcal{F}} \circ (\mathbf{P}_k^{\mathrm{T}} \mathbf{F}_k - \mathcal{F}) \right\|_2^2 \right) \\ \text{s.t.} \quad & \forall\, k,\ \mathbf{P}_k \mathbf{P}_k^{\mathrm{T}} = \mathbf{I},\ (\mathbf{P}_k)_{ij} \in \{0,1\} \end{aligned} \quad (2)$$

where $\circ$ denotes the element-wise product, $\alpha \in [0,1]$ weights the relative significance of nodes vs. edges.

The convex optimization problem in (2) is hard to solve. We resort to an iterative procedure that consists of the following three steps: (1) Given $\{\mathbf{P}_k\}$, $\{\mathcal{W}_{\mathcal{A}_l}\}$, and $\mathcal{W}_{\mathcal{F}}$, we find $\mathcal{G}$; (2) Given $\mathcal{G}$ and $\{\mathbf{P}_k\}$, we estimate $\{\mathcal{W}_{\mathcal{A}_l}\}$, $\mathcal{W}_{\mathcal{F}}$; and (3) Given $\mathcal{G}$, $\{\mathcal{W}_{\mathcal{A}_l}\}$, and $\mathcal{W}_{\mathcal{F}}$, we compute $\{\mathbf{P}_k\}$. Initially, all elements of $\{\mathcal{W}_{\mathcal{A}_l}\}$ and $\mathcal{W}_{\mathcal{F}}$ are set to 1, and all elements of $\{\mathbf{P}_k\}$ are randomly set to 1 or 0, such that $\forall k$, $\mathbf{P}_k\mathbf{P}_k^{\mathrm{T}} = \mathbf{I}$. The steps are iterated until convergence, i.e., when changes of the objective in (2) become less than $\epsilon = 10^{-3}$. Below, we explain each step.

4.1. Estimating the Model

Given permutations $\{\mathbf{P}_k\}$ and outlier weights $\{\mathcal{W}_{\mathcal{A}_l}\}$ and $\mathcal{W}_{\mathcal{F}}$, we find $\mathcal{G}$. Let $\mathcal{L}$ denote the objective of (2). From $\frac{\partial \mathcal{L}}{\partial \mathcal{A}_l} = 0$ and $\frac{\partial \mathcal{L}}{\partial \mathcal{F}} = 0$, we have:

$$\forall l, \mathcal{A}_l = \tfrac{1}{K} \textstyle\sum_{k=1}^{K} \mathbf{P}_k^{\mathrm{T}} \mathbf{A}_{kl} \mathbf{P}_k,\ \ \mathcal{F} = \tfrac{1}{K} \textstyle\sum_{k=1}^{K} \mathbf{P}_k^{\mathrm{T}} \mathbf{F}_k. \quad (3)$$

4.2. Estimating the Outlier Weights

To compute $\{\mathcal{W}_{\mathcal{A}_l}\}$ and $\mathcal{W}_{\mathcal{F}}$, we use the standard Huber's M-estimation. Specifically, the M-estimation speci-

fies that the outlier node and edge weights should be inversely proportional to the standard deviation of the corresponding nodes and edges in the training data. Thus, given correspondences between model nodes i, and nodes i' in the training set, and their associated descriptor vectors $(\mathcal{F})_{ib}$ and $(\mathbf{F}_k)_{i'b}$, where $b = 1, ..., d$, we compute $(\mathcal{W}_\mathcal{F})_{ib} = 1/\text{STD}(\{(\mathbf{F}_k)_{i'b} : k = 1, ..., K\})$. Similarly, given correspondences between model edges (i, j) and edges (i', j') in the training set, we compute $\forall l$, $(\mathcal{W}_{\mathcal{A}_l})_{ij} = 1/\text{STD}(\{(\mathbf{A}_k)_{i'j'} : k = 1, ..., K\})$.

4.3. Graph Matching

Given $\mathcal{G}$ and $\{\mathcal{W}_{\mathcal{A}_l}\}$ and $\mathcal{W}_\mathcal{F}$, we wish to compute permutation matrices, $\mathbf{P}_k$, of each training graph, G_k, $k = 1, ..., K$, that jointly minimize the weighted least squares in (2). Estimating $\{\mathbf{P}_k\}$ amounts to finding a subgraph isomorphism between $\mathcal{G}$ and each G_k. Below, we show that this problem can be reduced to the Quadratic Assignment Problem (QAP). We begin by re-writing the convex optimization of (2) in a more suitable form.

Define auxiliary matrices $\forall l, k$, $\widehat{\mathcal{W}}_{kl} = \mathbf{P}_k \mathcal{W}_{\mathcal{A}_l} \mathbf{P}_k^{\text{T}}$ and $\widehat{\mathcal{W}}_k = \mathbf{P}_k \mathcal{W}_\mathcal{F}$. To compute $\widehat{\mathcal{W}}_{kl}$ and $\widehat{\mathcal{W}}_k$, we use $\{\mathbf{P}_k\}$ estimated in the previous iteration. Also, define auxiliary matrices $\tilde{\mathcal{A}}_l = \mathcal{W}_{\mathcal{A}_l} \circ \mathcal{A}_l$, $\tilde{\mathbf{A}}_{kl} = \widehat{\mathcal{W}}_{kl} \circ \mathbf{A}_{kl}$, $\tilde{\mathcal{F}} = \mathcal{W}_\mathcal{F} \circ \mathcal{F}$, and $\tilde{\mathbf{F}}_k = \widehat{\mathcal{W}}_k \circ \mathbf{F}_k$. Then, the first and second terms in (2) can be more conveniently written as $\mathcal{W}_{\mathcal{A}_l} \circ (\mathbf{P}_k^{\text{T}} \mathbf{A}_{kl} \mathbf{P}_k - \mathcal{A}_l) = \mathbf{P}_k^{\text{T}} \tilde{\mathbf{A}}_{kl} \mathbf{P}_k - \tilde{\mathcal{A}}_l$, and $\mathcal{W}_\mathcal{F} \circ (\mathbf{P}_k^{\text{T}} \mathbf{F}_k - \mathcal{F}) = \mathbf{P}_k^{\text{T}} \tilde{\mathbf{F}}_k - \tilde{\mathcal{F}}$.

Since for any matrix $\mathbf{M}$ we have $\|\mathbf{M}\|^2 = \text{Tr}(\mathbf{M}^{\text{T}}\mathbf{M})$, and permutation matrices satisfy $\mathbf{P}_k \mathbf{P}_k^{\text{T}} = \mathbf{I}$, it is straightforward to show that the formulation of (2) reduces to the following K independent problems:

$$\max_{\mathbf{P}_k} \ \alpha \sum_{l=1}^{L} \text{Tr}(\mathbf{P}_k^{\text{T}} \tilde{\mathbf{A}}_{kl}^{\text{T}} \mathbf{P}_k \tilde{\mathcal{A}}_l) + (1-\alpha)\text{Tr}(\tilde{\mathcal{F}} \tilde{\mathbf{F}}_k^{\text{T}} \mathbf{P}_k) \quad \text{s.t.} \quad \mathbf{P}_k \mathbf{P}_k^{\text{T}} = \mathbf{I}, \ (\mathbf{P}_k)_{ij} \in \{0, 1\}. \tag{4}$$

The problem in (4) is the NP-hard QAP. The major difficulty comes from the quadratic constraint $\mathbf{P}_k \mathbf{P}_k^{\text{T}} = \mathbf{I}$, $(\mathbf{P}_k)_{ij} \in \{0, 1\}$. Next, we describe how to handle this constraint.

4.4. Vector Optimization Formulation

This section presents our approach to solving the NP-hard QAP, given by (4). Existing work (e.g., [15]) typically relaxes a QAP to a semi definite program (SDP). In our case, this would amount to relaxing $0 \leq (\mathbf{P}_k)_{ij} \leq 1$, and replacing the constraint $\mathbf{P}_k \mathbf{P}_k^{\text{T}} = \mathbf{I}$ with $\|\mathbf{P}_k\|_\infty = 1$. However, an SDP requires that the quadratic term in a QAP be defined by a positive semidefinite (PSD) matrix. We cannot meet this requirement, because our adjacency matrices in (4) are not symmetric, and thus not PSD.

Instead of using the common SDP relaxation, we directly seek a solution of (4) by reformulating the constraint $\mathbf{P}\mathbf{P}^{\text{T}} = \mathbf{I}$ into an equivalent, but more convenient quadratic constraint $\boldsymbol{x}^{\text{T}}\mathbf{B}\boldsymbol{x} = 0$. Here, $\boldsymbol{x}$ is an $nm \times 1$ concatenation vector of all columns of $\mathbf{P}$, $\boldsymbol{x} = \text{vec}(\mathbf{P})$. Also, the $nm \times nm$ matrix $\mathbf{B}$ encodes the basic property of the permutation matrix to have exactly one entry 1 in each row and each column, and 0's elsewhere. The computation of elements of $\mathbf{B}$ has a closed form, as follows. Note that $\boldsymbol{x}^{\text{T}}\mathbf{B}\boldsymbol{x} = \sum_{u=1}^{nm}\sum_{v=1}^{nm}(\boldsymbol{x})_u(\boldsymbol{x})_v(\mathbf{B})_{uv}$, so elements $(\mathbf{B})_{uv}$ should be set to 1, whenever setting vector elements $(\boldsymbol{x})_u = (\boldsymbol{x})_v = 1$ could violate the basic property of the permutation matrix, and thus produce $\boldsymbol{x}^{\text{T}}\mathbf{B}\boldsymbol{x} \neq 0$. That is, we set $(\mathbf{B})_{uv} = 1$ whenever $(\boldsymbol{x})_u$ and $(\boldsymbol{x})_v$ correspond to the elements of $\mathbf{P}$ that share the same row or column; and $(\mathbf{B})_{uv} = 0$ otherwise.

Unlike the original constraint on permutation matrices $\mathbf{P}\mathbf{P}^{\text{T}} = \mathbf{I}$, the equivalent quadratic constraint $\boldsymbol{x}^{\text{T}}\mathbf{B}\boldsymbol{x} = 0$ allows us to efficiently solve (4) using standard optimization software tools (e.g., cvx). Below, we give an equivalent formulation of (4) that immediately follows from replacing $\mathbf{P}_k$ with $\boldsymbol{x}_k$=vec($\mathbf{P}_k$), and the constraint $\mathbf{P}_k\mathbf{P}_k^{\text{T}} = \mathbf{I}$ with $\boldsymbol{x}_k^{\text{T}}\mathbf{B}_k\boldsymbol{x}_k = 0$. Define $\mathbf{Q}_k = -\sum_{l=1}^{L}\left(\tilde{\mathcal{A}}_l^{\text{T}} \otimes \tilde{\mathbf{A}}_{kl}^{\text{T}}\right)$, where $\otimes$ is the Hadamard product, and $\boldsymbol{c}_k = \text{vec}(-\tilde{\mathbf{F}}_k\tilde{\mathcal{F}}^{\text{T}})$. Then, the problem (4) can be equivalently written as

$$\min_{\boldsymbol{x}_k} \ \alpha\boldsymbol{x}_k^{\text{T}}\mathbf{Q}_k\boldsymbol{x}_k + (1-\alpha)\boldsymbol{c}_k^{\text{T}}\boldsymbol{x}_k \quad \text{s.t.} \ \boldsymbol{x}_k^{\text{T}}\mathbf{B}_k\boldsymbol{x}_k = 0, \ \mathbf{1}^{\text{T}}\boldsymbol{x}_k = n_k, \ (\boldsymbol{x}_k)_i \in \{0, 1\} \tag{5}$$

The equivalence between the two optimization problems is more formally stated in the following theorem.

Theorem: *The optimization problems (4) and (5) are equivalent when the constraint $\mathbf{P}\mathbf{P}^T = \mathbf{I}$ of (4) is replaced with the constraints $\boldsymbol{x} = vec(\mathbf{P})$, $\boldsymbol{x}^T\mathbf{B}\boldsymbol{x} = 0$, and $\mathbf{1}^T\boldsymbol{x} = n$.*

The problem in (5) is our final formulation. It can be readily solved by standard software tools. We use publicly available cvx (http://cvxr.com/cvx/).[1] The cvx tool is suitable for our purposes, because it makes use of the sparsity of $\mathbf{Q}$ and $\mathbf{B}$ in (5). Note that the large size of $\mathbf{Q}$ and $\mathbf{B}$ effectively does not increase complexity, because most elements of $\mathbf{Q}$ and $\mathbf{B}$ are zero. In our experiments, cvx can easily handle (5), even when the input graphs have 2000+ nodes (see Sec. 7 for running times).

After finding vectors $\{\boldsymbol{x}_k\}$ from (5), they are mapped to the corresponding permutation matrices $\{\mathbf{P}_k\}$. This concludes the three steps of our iterative learning of $\mathcal{G}$. The learned outlier weights $\{\mathcal{W}_{\mathcal{A}_l}\}$ and $\mathcal{W}_\mathcal{F}$ can be used for a robust model selection, i.e., the selection of model nodes and edges with small variance in the training data. We expect that at least 10% of nodes and edges in the training data belong to the background. Thus, we discard 10% of

[1] The objective and constraint of (5) can be easily convexified by subtracting from the main diagonal of $\mathbf{Q}$ and $\mathbf{B}$ their respective minimum eigenvalues. A lower bound of the minimum eigenvalue can be found efficiently using the Gerschgorin circle theorem.

the learned model nodes i and edges (i, j) with the lowest weights $\|(\mathcal{W}_{\mathcal{F}})_i\|_2$ and $(\mathcal{W}_{\mathcal{A}_l})_{ij}$.

5. Inference

Given a new video, its spatiotemporal graph, G, is matched to the available set of graph models $\{\mathcal{G}_r\}$. G is assigned the class label of the closest model, in the weighted least squares sense. Similar to (4), using the notation and definitions from Sec. 4.3, we formulate inference as

$$\min_r \max_{\mathbf{P}} \alpha \sum_{l=1}^{L} \mathrm{Tr}(\mathbf{P}^{\mathrm{T}} \tilde{\mathbf{A}}_l^{\mathrm{T}} \mathbf{P} \tilde{\mathcal{A}}_{rl}) + (1-\alpha)\mathrm{Tr}(\tilde{\mathcal{F}}_r \tilde{\mathbf{F}}^{\mathrm{T}} \mathbf{P})$$
$$\text{s.t. } \mathbf{P}\mathbf{P}^{\mathrm{T}} = \mathbf{I},\ (\mathbf{P})_{ij} \in \{0,1\} \qquad (6)$$

As in Sec. 4.4, we reformulate (6) into an equivalent, but simpler quadratic program, similar to (5). Then, we use the cvx software tool to efficiently solve it.

6. Extracting the Spatiotemporal Graph

The literature presents many successful approaches to spatiotemporal video segmentation, including those based on tracking interest points [7], clustering pixels from all frames [5], and variational estimation of active surfaces [17]. Their code is typically not publicly available. Since our focus in this paper is not on the problem of low-level video segmentation, we specify and use a simple, blocky segmenter, which can process long videos in nearly real time, and produce spatiotemporal graphs of sufficient accuracy to robustly learn the graph models of human activities.

We initially oversegment the video into space-time blocks with data-driven shapes and sizes. A block is a homogeneous group of pixels from a few consecutive frames, where variations of photometric and motion properties (e.g., color, optical flow) within the block are smaller than variations of its surround. We then agglomeratively group similar, adjacent blocks into a hierarchy of clusters, where each cluster represents a blocky spatiotemporal tube, as illustrated in Fig. 2. While the extracted tubes are blocky, their 2D+t shapes are able to capture the motions and spatial extent of the corresponding objects in the video (see Sec. 7). The tubes are used to build the video spatiotemporal graph.

Oversegmentation: The video is recursively split top-down into 2D+t blocks along the x, y, and t axes. The splitting is done greedily by selecting block b whose partition into subblocks b_1 and b_2 maximally increases the compression gain relative to the other descendants. The compression gain is defined as $\gamma_{b_1 b_2} = |\epsilon_{b_1} + \epsilon_{b_2} - \epsilon_b|$, where the compression error $\epsilon_b = \sum_{p \in b} \|\boldsymbol{f}_p - \overline{\boldsymbol{f}_b}\|^2$ is a sum of Euclidean distances of pixels' feature vectors $\boldsymbol{f}_p$ from the mean $\overline{\boldsymbol{f}_b}$. $\boldsymbol{f}_p$ consists of HSV color values, and Lucas-Kanade optical flow at pixel p. We use integral volumes to efficiently compute $\gamma_{b_1 b_2}$ in $\mathcal{O}(1)$ when searching for the optimal split of a visited block along one of x, y, and t axes. Finding the best split in each iteration takes at most $\mathcal{O}$(width+height+length) of the video. The recursion ends when the splitting yields two children blocks whose volumes are smaller than a certain size (we set 10 frames). Finally, we take the smallest blocks as our result of video oversegmentation.

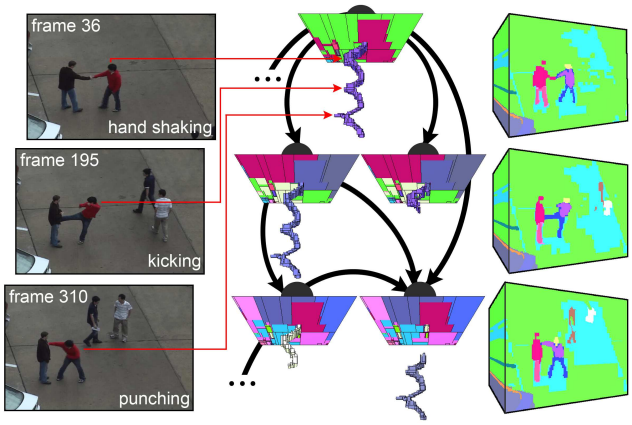

Figure 2. Multiscale spatiotemporal segmentation. The red arrows show corresponding video parts and their volumetric representations. The shape of the tube accurately captures the actor's trajectory. We show only a part of the extracted spatiotemporal graph with only hierarchical relations between 2D+t tubes, for clarity. One level of segmentation is shown on the right.

Agglomerative clustering: Analogously to finding MSER regions, we agglomeratively merge neighboring (touching) and most similar video blocks of the aforementioned oversegmentation. Similarity, i.e., distance between blocks is computed as $\Delta = |\epsilon_{b_1 \cup b_2} - \epsilon_{b_1} - \epsilon_{b_2}|$, and recorded in each merging iteration τ. The merging stops when variations of the gradient $g^{(t)} \approx |\Delta^{(\tau+1)} - \Delta^{(\tau)}|$ become sufficiently large. At that moment, all mergers differ significantly in their compression errors, and thus are taken to represent 2D+t tubes of a plausible spatiotemporal video segmentation. Our experiments agree with the well-known practice with MSER regions that gradient changes are typically small except at a few critical iteration steps, which can be robustly identified. The agglomerative merging of blocks is continued until the next significant jump in the gradient values. This gives multiscale 2D+t tubes at different levels of homogeneity. The tubes are organized in the spatiotemporal graph. Depending on a video, the resulting graph may have 1000–2000 nodes. Note that the number of nodes does not directly depend on the length, but contents of the video.

Features of graph nodes: Each 2D+t tube is described by a descriptor vector representing a concatenation of four types of 10-bin histograms of: (i) 2D areas of the tube's intersections with video frames, normalized relative to the corresponding 2D areas of the parent tube, for scale invariance; (ii) contrasts $\sigma_i = \epsilon - \epsilon_i$ between compression errors of the tube and its neighboring (touching) tubes i, for illumination

invariance; and (iii)-(iv) Lucas-Kanade optical flows along x and y axes, computed relative to the corresponding optical flows of the parent tube, for invariance to camera motion.

Features of graph edges: The 2D+t tubes are connected with following three types of directed edges: (i) hierarchical – *ascendant*, *descendant*; (ii) temporal – *before*, *after*, *overlap*, *meet*; and (iii) spatial – *top*, *bottom*, *left*, and *right*. Each edge is characterized by a strength of the corresponding relationship. The strength of hierarchical edges is estimated as the ratio of ascendant and descendant volumes. The strength of temporal edges is computed as the number of frames relative to the video's length. The strength of spatial edges takes binary values for present or absent.

7. Results

Different aspects of our approach are evaluated on both synthetic data, and real benchmark datasets.

Synthetic data. These experiments serve to test our graph matching (i.e., the core of our graph learning and inference algorithms) in the face of different types and levels of noise. Given a model graph with 2000 nodes and 10000 edges from all the three edge types, we create 100 new graphs by randomly (i) removing or adding nodes and edges, and (ii) changing descriptors associated with nodes and edges. Then, we match the model and resulting graphs, as in Eq. (6). Error is estimated as the percentage of wrongly matched pairs of nodes out of a total number of matches (note that our recall is always 1). Error is then averaged over 100 distinct models with the same number of nodes and edges, but with different connectivity. Fig. 3 shows our results when the synthetic graphs are generated by randomly: (a) Permuting a percentage of rows and columns of the model adjacency matrix; (b) Removing a percentage of edges from (or adding new edges to) the model; when edges are removed (or added), their corresponding elements of the model adjacency matrix are set to zero (or probabilistically sampled from the uniform distribution in [0,1]); (c) Removing a percentage of nodes from (or adding new nodes to) the model by deleting (or inserting) new rows and columns from (in) the mode adjacency matrix; elements of the inserted rows and columns of the resulting adjacency matrices are probabilistically sampled from the uniform distribution in [0,1]); and (d) Adding uniform noise from interval $[0, \eta]$, $0 \leq \eta \leq 1$, to descriptors associated with model nodes. In all these cases, we observe generous degradation of performance when noise levels increase. Next, we also test sensitivity to a specific choice of parameter α, in Fig. 3e, when the synthetic graphs are generated from the model by randomly permuting a 50% of model nodes, and removing 20% of model nodes, and adding uniform noise with η=0.2 to the node features. Fig. 3e shows that the optimal value of α has a relatively wide margin around 0.3, which we use in our experiments with real data. Fig. 3f shows our running times of graph matching when the number of nodes in the synthetic graphs changes by randomly removing (or adding) nodes from (to) the model, as in (c).

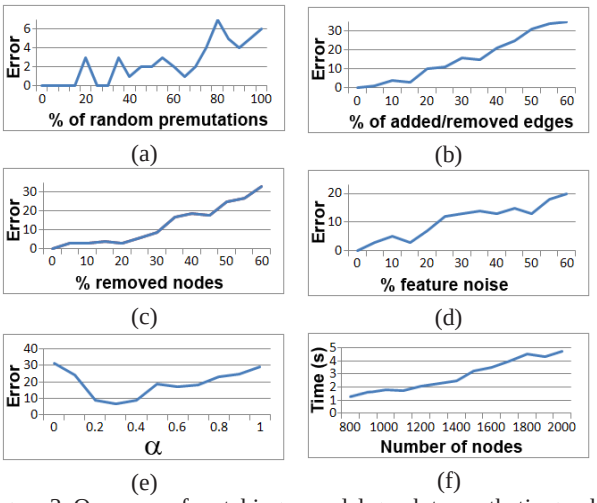

Figure 3. Our error of matching a model graph to synthetic graphs generated from the model by randomly: (a) permuting rows and columns of the model adjacency matrix; (b) removing or adding edges; (c) removing or adding nodes; (d) adding uniform noise from interval $[0, \eta]$, $0 \leq \eta \leq 1$ to node features; (e) 50% of rows and columns of the model adjacency matrix are permuted, 20% of model nodes are removed, and uniform noise with η=0.2 is added to node features; (f) Running times of our graph matching for the same setting as in (c).

Real data. Activity recognition is evaluated on UT interaction dataset [18], Olympic sports dataset [16], and Wiezmann dataset [8]. UT dataset contains 60 videos showing six types of two-person interactions: hand-shaking, hugging, kicking, pointing, punching, and pushing. The dataset is composed of 10 sets, where each set contains videos of a pair of different persons performing all six interactions. This dataset is challenging, because it involves complex, structured interactions between people that are composed of a number of distinct, non-periodic, atomic-level actions, including stretch-arm, withdraw-arm, stretch-leg, lower-leg, and lean-forward. Also, the UT videos show multiple co-occurring activities of interest, and thus evaluate if we can parse even non-prominent activity instances. For training and testing we use the same set-up as in [18] – namely, 20% of the data is used for training and 80% for testing. The Olympic sports dataset [16] consists of 50 YouTube videos for each of 16 activity classes. Each activity is performed only by a single subject, and represents a temporal sequence of primitive actions (e.g., running, jumping, landing, and standing-up). As in [16], we use 80% of videos from the dataset for training, and the rest for testing. Challenges of this dataset arise from low resolution, background clutter, and complex sequence of primitive actions. In training, for both datasets, we only have access to the activity-class label

	Our	nCuts [20]
Accuracy	77.3%	78.7%
Running Time	14.2s	243.6s

Table 1. Average classification accuracy, and average running time per video for computing the spatiotemporal graph on the Olympic Sports Dataset [16].

	hand shaking	hugging	kicking	pointing	punching	pushing
Our	**81.7%**	**89.6%**	**68.6%**	**66.4%**	**84.5%**	**82.7%**
[18]	75%	87.5%	62.5%	50%	75%	75%

Table 2. We outperform the approach of [18], in terms of average classification accuracy on six human-human interactions from the UT dataset, by 8.1% on average.

Sport class	Our	[16]	[12]
high-jump	75.8%	68.9%	52.4%
long-jump	78.6%	74.8%	66.8%
triple-jump	69.7%	52.3%	36.1%
pole-vault	85.5%	82.0%	47.8%
gymnastics-vault	89.4%	86.1%	88.6%
shot-put	65.9%	62.1%	56.2%
snatch	72.1%	69.2%	41.8%
clean-jerk	86.2%	84.1%	83.2%
javelin-throw	77.8%	74.6%	61.1%
hammer-throw	79.4%	77.5%	65.1%
discus-throw	62.2%	58.5%	37.4%
diving-platform	89.9%	87.2%	91.5%
diving-springboard	82.2%	77.2%	80.7%
basketball-layup	79.7%	77.9%	75.8%
bowling	78.7%	72.7%	66.7%
tennis-serve	63.8%	49.1%	39.6%
Average classification accuracy	77.3%	71.1%	62.0%

Table 3. We outperform both [16] and [12] in terms of average classification accuracy on the Olympic sports dataset.

of the entire video. We do not require any human trackers and detectors.

Blocky vs. Ncuts: We evaluate our activity recognition on the Olympic Sports dataset, when the videos are segmented by our blocky spatiotemporal segmentation, and by the 2D+t Ncuts approach [5]. The initial Ncut segments are agglomeratively merged, using the same procedure as ours (see Sec. 6) to build the hierarchical graph. We vary the number of initial Ncut segments, 100:100:500, and report the best activity recognition results in Table 1. As can be seen, despite blockiness, our approach gives similar performance to that or Ncuts, while our computation of the spatiotemporal graph is 20 times faster.

Relevance of capturing activity structure: We use all three datasets to evaluate the influence of our choice of parameter α on activity recognition. On the Wiezmann dataset, our average classification accuracy is 98.2% for the optimal value of $\alpha = 0.1$. This value is lower than the one obtained on the synthetic data in Fig. 3e. This suggests that for simple actions, such as those in the Wiezmann dataset, appearance and motion properties associated with nodes of our activity model are more important than the structure of the model for recognition. On more complex Olympic sports and UT videos, we find the optimal values of $\alpha = 0.3$ and $\alpha = 0.4$, respectively. This suggests that the structure of the model becomes increasingly important for recognizing more complex activities.

The value of hierarchical edges: When all hierarchical edges are removed from the spatiotemporal graphs and the model, and the other edges kept intact, our average classification accuracy drops: (i) from 98.2% to 93.2% on the Wiezmann dataset; from 77.3% to 71.4% on the Olympic sports dataset; and from 78.9% to 70.1% on the UT dataset. This suggests that the hierarchical relationships between video subvolumes provide important contextual information for recognition.

Comparison: Tables 2 and 3 show that we outperform the state of the art on UT and Olympic sports datasets.

Localization: Our inference localizes spatiotemporal tubes in the video by matching the video's graph with the graph model. The UT dataset is annotated with bounding boxes around a group of people performing the activity in each frame. We compute the localization accuracy on the UT dataset as a ratio of intersection and union of the ground truth bounding boxes and the tubes matched by our algorithm. The average localization accuracy is 78%.

Qualitative results: The UT and Olympic datasets do not provide ground-truth segmentation masks of actors performing the activity, and the Weizmann dataset is too trivial. Therefore, we are not in a position to quantitatively evaluate our blocky spatiotemporal segmentation. Fig 4 shows and example which illustrates that our inference simultaneously provides segmentation, i.e., localization of 2D+t tubes occupied by the activity. As can be seen, while matching the model graph with the spatiotemporal graph of this video, in inference, the weights are correctly localized as relevant video parts for activity recognition.

Typical failure cases: The proposed spatiotemporal segmentation produces many spurious tubes for video parts corresponding to structured, spatially repeating backgrounds (e.g., city scenes with similar buildings), and dynamic textures (e.g., crowds of people). Since such spurious tubes have high frequency and similarity across the video, by definition of dynamic texture, our weighted least-squares formulation typically fails to exclude them from the activity model. As a result, we may learn dynamic textures in the background as relevant activity parts.

Implementation: On average, matching the model with about 1000 nodes to a spatiotemporal graph with about 2000+ nodes, in inference given by (6), takes less than 10s in MATLAB on a 2.66GHz, 3.49GB RAM PC.

8. Conclusion

We have presented a new, volumetric-based approach to activity recognition and video parsing. Our approach automatically learns the structure of complex human activities, in terms of relevant subactivities and their hierarchical,

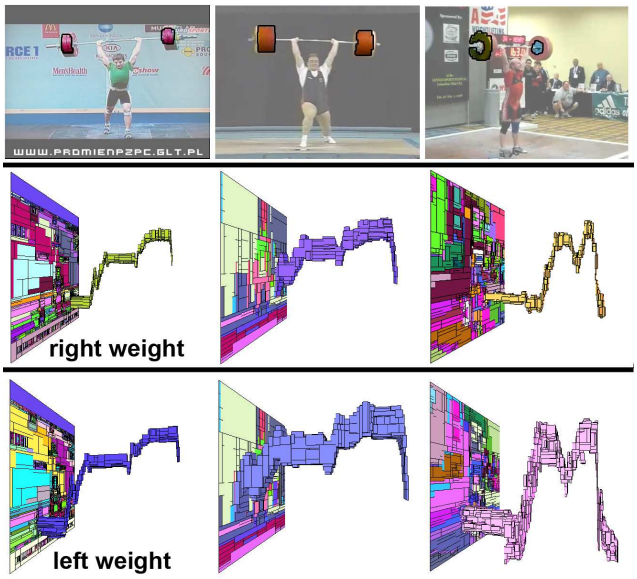

Figure 4. Example of graph alignment on clean-jerk videos from the Olympic Sports Dataset [16]. Weight tubes are matched correctly.

temporal, and spatial relations. We have formulated inference and learning of a structural activity model within the same framework, that of weighted least-squares. Our learning is efficient, allowing for fast training (in seconds) from videos which are segmented into more than 2000 space-time tubes at multiple scales. The presented experimental results demonstrate that our activity recognition downgrades gracefully in the face of increasing noise levels. Under reasonable running times, our recognition rates on benchmark datasets outperform the state of the art. We believe that this is because existing work manually specifies relevant activity parts and their temporal relations.

References

[1] J. Aggarwal and M. Ryoo. Human activity analysis: A review. *ACM Comput. Surv.*, 43:16:1–16:43, 2011. 1

[2] N. Ahuja and S. Todorovic. Extracting texels in 2.1D natural textures. In *ICCV*, 2007. 2

[3] M. Albanese, R. Chellappa, N. Cuntoor, V. Moscato, A. Picariello, V. S. Subrahmanian, and O. Udrea. PADS: A probabilistic activity detection framework for video data. *IEEE TPAMI*, 32:2246–2261, 2010. 1

[4] H. Bunke, P. Foggia, C. Guidobaldi, and M. Vento. Graph clustering using the weighted minimum common supergraph. In *GbRPR*, 2003. 2

[5] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nystrom method. *IEEE TPAMI*, 26(2):214–225, 2004. 5, 7

[6] N. Friedman and D. Koller. Being Bayesian about Bayesian network structure: A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50(1–2):95–125, 2003. 2

[7] D. B. Goldman, C. Gonterman, B. Curless, D. Salesin, and S. M. Seitz. Video object annotation, navigation, and composition. In *UIST*, pages 3–12, 2008. 5

[8] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. *IEEE TPAMI*, 29(12):2247–2253, 2007. 2, 6

[9] A. Gupta, P. Srinivasan, J. Shi, and L. Davis. Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos. In *CVPR*, 2009. 1

[10] R. Hamid, S. Maddi, A. Bobick, and I. Essa. Structure from statistics: Unsupervised activity analysis using suffix trees. In *ICCV*, 2007. 1

[11] T. Lan, Y. Wang, W. Yang, and G. Mori. Beyond actions: Discriminative models for contextual group activities. In *NIPS*, 2010. 1

[12] I. Laptev, M. Marszalek, C. Schmid, B. Rozenfeld, I. Rennes, I. I. Grenoble, and L. L. B. Learning realistic human actions from movies. In *CVPR*, 2008. 7

[13] Z. Lin, Z. Jiang, and L. S. Davis. Recognizing actions by shape-motion prototype trees. In *ICCV*, 2009. 1, 2

[14] X. Liu, L. Lin, S. C. Zhu, and H. Jin. Trajectory parsing by cluster sampling in spatiotemporal graph. In *CVPR*, 2009. 1

[15] A. Nemirovski. Sums of random symmetric matrices and quadratic optimization under orthogonality constraints. *Math. Program.*, 109:283–317, 2007. 4

[16] J. C. Niebles, C.-W. Chen, , and L. Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *ECCV*, 2010. 1, 6, 7, 8

[17] M. Ristivojevic and J. Konrad. Space-time image sequence analysis: object tunnels and occlusion volumes. *IEEE Trans. Image Processing*, 15(2):364 –376, 2006. 5

[18] M. S. Ryoo and J. K. Aggarwal. Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities. In *ICCV*, 2009. 6, 7

[19] J. Shi, S. Belongie, T. Leung, and J. Malik. Image and video segmentation: The normalized cut framework. In *ICIP*, pages 943–947, 1998. 7

[20] S. Todorovic and N. Ahuja. Unsupervised category modeling, recognition, and segmentation in images. *IEEE TPAMI*, 30(12):1–17, 2008. 2

[21] A. Torsello. An importance sampling approach to learning structural representations of shape. *CVPR*, 2008. 2

[22] A. Torsello and E. R. Hancock. Learning shape-classes using a mixture of tree-unions. *IEEE TPAMI*, 28(6):954–967, 2006. 2

[23] S. D. Tran and L. S. Davis. Event modeling and recognition using markov logic networks. In *ECCV*, 2008. 1

[24] T. Xiang and S. Gong. Beyond tracking: Modelling activity and understanding behaviour. *IJCV*, 67(1):21–51, 2006. 1

[25] J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden markov model. In *CVPR*, 1992. 1