Unsupervised Temporal Commonality Discovery

Wen-Sheng Chu, Feng Zhou, and Fernando De la Torre

Robotics Institute, Carnegie Mellon University

Abstract. Unsupervised discovery of commonalities in images has recently attracted much interest due to the need to find correspondences in large amounts of visual data. A natural extension, and a relatively unexplored problem, is how to discover common semantic temporal patterns in videos. That is, given two or more videos, find the subsequences that contain similar visual content in an unsupervised manner. We call this problem Temporal Commonality Discovery (TCD). The naive exhaustive search approach to solve the TCD problem has a computational complexity quadratic with the length of each sequence, making it impractical for regular-length sequences. This paper proposes an efficient branch and bound (B&B) algorithm to tackle the TCD problem. We derive tight bounds for classical distances between temporal bag of words of two segments, including ℓ_1, intersection and χ^2. Using these bounds the B&B algorithm can efficiently find the global optimal solution. Our algorithm is general, and it can be applied to any feature that has been quantified into histograms. Experiments on finding common facial actions in video and human actions in motion capture data demonstrate the benefits of our approach. To the best of our knowledge, this is the first work that addresses unsupervised discovery of common events in videos.

Key words: Temporal bag of words, branch and bound, temporal commonality discovery.

1 Introduction

Unsupervised discovery of visual patterns in images has been a long standing computer vision problem driven by applications to cosegmentation [8,15,20], learning grammars of images [34], detecting irregularity [6] and automatic tagging [23]. Although recently there has been several work on unsupervised discovery of visual patterns in images, a relatively unexplored problem in computer vision is to discover common temporal patterns among video sequences. For instance, given two or more videos, finding the segments that contain common human actions. Fig. 1 illustrates the main problem addressed in this paper. Given two videos from "*As Good As It Gets*" and "*Indiana Jones And The Last Crusade*", this paper proposes an unsupervised method to find multiple subsequences that share similar semantic contents (*e.g.*, *Kissing* or *Handshake*). Through the paper, we will refer to this problem as Temporally Commonality Discovery (TCD).

Recall that TCD is a fully unsupervised problem, so no prior knowledge is provided—we do not know what the commonalities are, how many there are and where they start and end. A naive method to find desired pair(s) of common

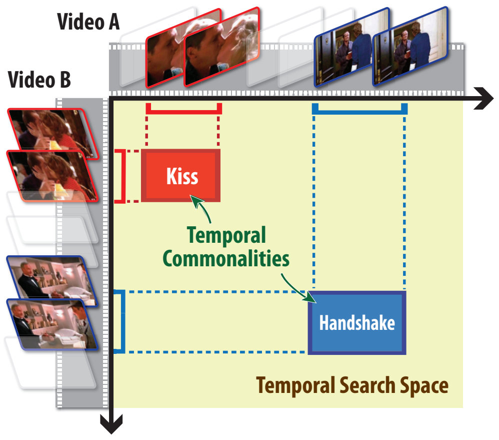


Fig. 1. Temporal Commonality Discovery (TCD). Given two videos from the movies "*As Good As It Gets*"(top) and "*Indiana Jones And The Last Crusade*"(left), how to discover in an unsupervised manner the common actions between them? In this case our algorithm found segments of *Kissing* and *Handshaking* as common actions between both videos. Note that the common segments can have different lengths.

subsequences would be the *sliding window* approach, *i.e.*, exhaustively search all possible pairs of subsequences and select the pair(s) with the highest score(s). However, the complexity of this approach scales quadratically with the length of both sequences, $\mathcal{O}(m^2n^2)$, for two sequences of length m and n. For instance, in the case of two sequences with 200 and 300 frames, there are more than *three billion* possible matchings that need to be computed at different lengths and locations. Therefore, the naive approach is computationally prohibitive for reasonable length sequences.

Inspired by [13,32] that used the branch and bound (B&B) algorithm to efficiently search for optimal image patches or video volumes, we propose to adopt B&B for searching simultaneously over all possible segments in each video sequence (see Fig. 1). Two are the main contributions of this study: (1) Introduce the new problem of unsupervised TCD. While there exist studies that address commonality discovery in images [8,15,20,30], to the best of our knowledge there is little work that tackles unsupervised search of commonalities in video sequences. Also, note that there are several studies that address the problem of event detection or sequence labeling of human actions in video (*e.g.*, [12,27,32]). However, unlike TCD, those studies require learning a set of classifiers from training data. (2) Formulate the TCD as an integer optimization problem and propose an efficient B&B algorithm that finds the globally optimal solution. We derive new tight bounds for ℓ_1, intersection and χ^2 distances between histograms, allowing the B&B scheme to discard large portion of the search space. Experimental validation on standard datasets for finding similar facial expressions in video and human actions in motion capture data illustrates the benefits of our approach.

2 Related Work

Our work is inspired by recent success on using B&B with Support Vector Machines (SVM) for efficient template matching. Lampert *et al.* [13] proposed Efficient Subwindow Search (ESS) to find the optimal subimage that maximizes the prediction score of a pre-trained SVM classifier. Hoai *et al.* [12] combined a multiclass SVM with Dynamic programming for efficient temporal classification and segmentation. Yuan *et al.* [32] generalized Lampert's 4-D search to the 6-D Spatio-Temporal Branch-and-Bound (STBB) by incorporating time, to search for spatio-temporal volumes. However, unlike TCD, these approaches are supervised and require a training stage.

Recently, there have been interests in temporal clustering algorithms for unsupervised discovery of human actions. Wang *et al.* [30] exploited deformable template matching of shape and context in static images to discover action classes. Si *et al.* [25] learned an event grammar by clustering event co-occurrence into a dictionary of atomic actions. Zhou *et al.* [33] combined spectral clustering and dynamic time warping to cluster time series, and applied it to learn taxonomies of facial expressions. Turaga *et al.* [28] used extensions of switching linear dynamical systems for clustering human actions in video sequences. However, if we cluster two sequences that only have one segment in common, previous methods for clustering time series will likely need many clusters to find the common segments. In our case, TCD discovers only similar segments and avoids the need for clustering all the video that is computationally expensive and prone to local minima. Another unsupervised technique related to TCD is motif detection [18,19]. Time series motif algorithms find repeated patterns within a single sequence. Minnen *et al.* [18] discovered motifs as high-density regions in the space of all subsequences. Mueen and Keogh [19] further improved the motif discovery problem using an online technique, maintaining the exact motifs in real-time performance. Nevertheless, these work detects motifs within only one sequence, but TCD considers two (or more) sequences. Moreover, it is unclear how these technique can be robust to noise.

The longest common subsequence (LCS) [10,17,21] is also related to TCD. The LCS problem consists on finding the longest subsequence that is common within a set of sequences (often just two) [21,31]. Closer to our work is the algorithm for longest consecutive common subsequence (LCCS) [31] that finds the longest contiguous part of original sequences (*e.g.*, videos). However, different from TCD, these approaches have a major limitation in that they find only identical subsequences, and hence are not robust to noise that is typical in realistic videos.

3 Unsupervised TCD

3.1 Problem Formulation

In the following, we will assume that at least one commonality exists among a pair of time series (*e.g.*, two video sequences), represented as matrices $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_m]$ and $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_n]$ (see notation[1]). We formulate the TCD problem as the integer programming over two integer intervals $[b_1, e_1] \subseteq [1, m]$ and

[1] Bold capital letters denote a matrix $\mathbf{X}$, bold lower-case letters a column vector $\mathbf{x}$. $\mathbf{x}_i$ represents the i^{th} column of the matrix $\mathbf{X}$. All non-bold letters represent scalars.

$[b_2, e_2] \subseteq [1, n]$:

$$\begin{aligned} \min_{b_1,e_1,b_2,e_2} \quad & d\left(\varphi_{\mathbf{A}[b_1,e_1]}, \varphi_{\mathbf{B}[b_2,e_2]}\right), \\ \text{s.t.} \quad & e_i - b_i \geq \ell, \forall i \in \{1, 2\}, \end{aligned} \tag{1}$$

where $\mathbf{A}[b_1, e_1] = [\mathbf{a}_{b_1}, \ldots, \mathbf{a}_{e_1}]$ denotes the subsequence of $\mathbf{A}$ that begins from frame b_1 and ends in frame e_1 (similar for $\mathbf{B}[b_2, e_2]$). $\varphi_{\mathbf{x}}$ is a feature mapping for a sequence $\mathbf{x}$ (see Sec. 3.3 for details), $d(\mathbf{y}, \mathbf{z})$ is a distance measurement between two feature vectors $\mathbf{y}$ and $\mathbf{z}$, and ℓ controls the minimal length for each subsequence to avoid the trivial solution of both lengths being zero.

Given a sequence pair $\mathbf{A}$ and $\mathbf{B}$, the goal of TCD is to find the two most common intervals $[b_1, e_1]$ and $[b_2, e_2]$, such that problem (1) is minimized. Note that, as illustrated in Fig. 1, the discovered sequences $\mathbf{A}[b_1, e_1]$ and $\mathbf{B}[b_2, e_2]$ can have different lengths, thus we don't assume a fixed length for discovered sequences. A naive approach for solving (1) is to search over all possible locations for (b_1, e_1, b_2, e_2). However, it leads to an algorithm with computational complexity $\mathcal{O}(m^2n^2)$, which is prohibitive for regular videos with hundreds or thousands of frames. To cope with this problem, this paper proposes a B&B algorithm to efficiently find the global optimal solution to (1).

3.2 Optimization by Branch and Bound (B&B)

With a proper bounding function, the B&B framework is significantly more efficient than exhaustive approaches. In this section, we leverage B&B to efficiently find the global solution for problem (1).

Problem interpretation: To have a better understanding of the search space, we first re-formulate the problem (1) as the problem of searching over two sequence's timelines (as illustrated in Fig. 1). A rectangle r in the search space indicates one candidate solution (b_1, e_1, b_2, e_2) for (1). This candidate solution would match a segment in video $\mathbf{A}$ beginning at b_1 and ending at e_1 with another segment in video $\mathbf{B}$ beginning at b_2 and ending at e_2. To allow a more efficient representation for searching, we parameterize each step as searching over sets of candidate solutions. That is, we search over intervals instead of individual value for each parameter. Each parameter interval corresponds to a rectangle set in the search space, *i.e.*, $\mathcal{R} = [B_1, E_1, B_2, E_2]$, where $B_i = [b_i^{lo}, b_i^{hi}]$ and $E_i = [e_i^{lo}, e_i^{hi}]$ $(i = 1$ and $2)$ indicate tuples of parameters ranging from frame lo to frame hi.

The B&B algorithm: Algorithm 1 summarizes the proposed TCD procedure. We use a priority queue $\mathcal{Q}$ to maintain the search process. Each state in $\mathcal{Q}$ contains a rectangle set $\mathcal{R}$, its upper bound $u(\mathcal{R})$ and lower bound $l(\mathcal{R})$. Each iteration starts by selecting the rectangle set $\mathcal{R}$ from the top state, which is defined as the state containing the maximal lower bound (recall that lower bounds can be negative; see Sec. 3.3 for details of the bounds). Then in the *branch step*, each rectangle set is split by its largest interval into two disjoint subsets. For example, suppose E_2 is the largest interval, then $\mathcal{R} \to \mathcal{R}' \cup \mathcal{R}''$ where $E_2' := [e_2^{lo}, \lfloor \frac{e_2^{lo}+e_2^{hi}}{2} \rfloor]$ and $E_2'' := [\lfloor \frac{e_2^{lo}+e_2^{hi}}{2} \rfloor + 1, e_2^{hi}]$. In the *bound step*,

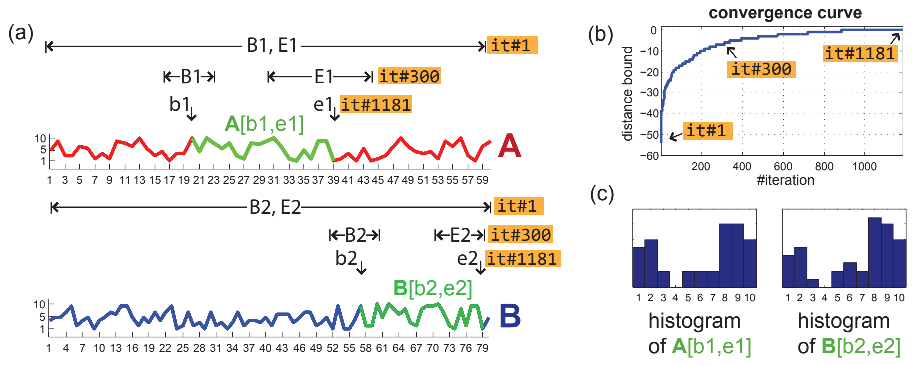


Fig. 2. An example of TCD for two synthetic 1-D time series (best viewed in color). Note that in this case when $\ell = 20$, a naive sliding window approach needs more than *5 million* evaluations while the proposed B&B method takes only 1181 to converge. (a) Search ranges at iterations (`it`) #1, #300 and #1181 over sequences **A** and **B**. Commonalities $\mathbf{A}[b_1, e_1]$ and $\mathbf{B}[b_2, e_2]$ are discovered at convergence (#1811). (b) Convergence curve of the lower bound. (c) Histograms of the discovered commonalities.

we calculate the bounds for the lowest dissimilarity for each rectangle set, and then update new rectangle sets and bounds into $\mathcal{Q}$. The algorithm terminates when $\mathcal{R}$ contains a unique entry. Fig. 2 shows an example of TCD for discovering commonality between two 1-D sequences. Despite that in the worst case the complexity of B&B can be still $\mathcal{O}(m^2n^2)$, we will experimentally show that in general B&B is much more efficient than the naive search.

Note that the optimal discovered sequences can be of length greater than ℓ. To show an example, consider two 1-D sequences $\mathbf{A} = [1, 2, 2, 1]$ and $\mathbf{B} = [1, 1, 3]$. Suppose we use ℓ_1 distance, set the minimal length $\ell = 3$, and represent their 3-bin histograms as $\varphi_{\mathbf{A}[1,4]} = [2, 2, 0]$, $\varphi_{\mathbf{A}[1,3]} = [1, 2, 0]$ and $\varphi_{\mathbf{B}} = [2, 0, 1]$. Hereby we can conclude by showing the distances: $d_{\ell_1}(\varphi_{\mathbf{A}[1,4]}, \varphi_{\mathbf{B}}) = 3 < 4 = d_{\ell_1}(\varphi_{\mathbf{A}[1,3]}, \varphi_{\mathbf{B}})$.

Differences from ESS [13] and STBB [32]: Although the proposed B&B algorithm is in spirit similar to ESS and STBB, it has three essential differences from these methods: (1) Different search space. ESS and STBB search over spatial coordinates of an image or spatio-temporal volumes in a video, while TCD searches over beginning and ending positions of the segments in two sequences. (2) ESS and STBB are supervised techniques and seek for highly confident regions according to a pre-trained SVM classifier; TCD is unsupervised. (3) We introduce new bounding functions for the B&B framework that guarantee efficiency and optimality for the TCD problem. Moreover, ESS and STBB consider only upper bounds, while TCD incorporates both upper and lower bounds and hence is able to prune the priority queue for accelerating the search.

Algorithm 1: Temporal Commonality Discovery

```
input  : Feature lists for a sequence pair $\mathbf{A}, \mathbf{B}$; minimal length $\ell$
output: Optimal rectangle $r^*$ in the temporal search space
1 Initialize $\mathcal{Q} \leftarrow$empty priority queue;
2 Initialize $\mathcal{R} \leftarrow [1,m] \times [1,m] \times [1,n] \times [1,n]$;
3 while Size of $\mathcal{R}$ is not 1 do
4     Split one interval into two disjoint sets $\mathcal{R} \rightarrow \mathcal{R}' \cup \mathcal{R}''$ (branch step);
5     Compute bounds in Sec. 3.3 for two new intervals $\mathcal{R}'$ and $\mathcal{R}''$ (bound step);
6     Push both $\mathcal{R}'$ and $\mathcal{R}''$ into $\mathcal{Q}$, ordered by bounds;
7     Pop the top state $\mathcal{R}$ from $\mathcal{Q}$;
8 end
9 Assign the optimal rectangle $r^* \leftarrow \mathcal{R}$;
```

3.3 Construction of a Bounding Function

Representation of signals: Throughout the paper we will use the Bag of Temporal Words (BoTW) model [26,32] to represent video segments. Observe, that any features that can be discretized into histograms can fit into our framework. In BoTW the codebook is built using a clustering method (*e.g.*, k-means) to group similar feature vectors. Each frame is then quantized according to the k-entry dictionary. The histogram for a given sequence is then built by accumulating individual frame histograms. We represent the feature mapping $\varphi_{\mathbf{A}[b_1,e_1]}$ in (1) as the histogram of temporal words for the subsequence in the interval $[b_1, e_1]$. Another notable benefit of the histogram representation is that it allows for fast recursive computation using the concept of *integral image* [29]. That is, for frame t, we accumulate the sum of $\varphi_{\mathbf{A}[1,t]}$ of the histograms up to t. Using this structure, we can efficiently compute the histogram for any subsequence $\mathbf{A}[t_1, t_2]$ as $\varphi_{\mathbf{A}[t_1,t_2]} = \varphi_{\mathbf{A}[1,t_2]} - \varphi_{\mathbf{A}[1,t_1-1]}$.

Properties of bounding functions: Recall that $\mathcal{R}$ is a rectangle set and $r \equiv (b_1, e_1, b_2, e_2)$ a rectangle in the temporal search space representing two subsequences $\mathbf{A}[b_1, e_1]$ and $\mathbf{B}[b_2, e_2]$. We denote $d(r) = d(\varphi_{\mathbf{A}}, \varphi_{\mathbf{B}})$ as the distance between their histograms $\varphi_{\mathbf{A}}$ and $\varphi_{\mathbf{B}}$. The smaller the value of $d(\varphi_{\mathbf{A}}, \varphi_{\mathbf{B}})$, the more likely the sequences share commonalities. To harness the B&B framework, we need to find an upper bound $u(\mathcal{R})$ and a lower bound $l(\mathcal{R})$ that satisfy the three properties:

(a) $u(\mathcal{R}) \geq \max_{r \in \mathcal{R}} d(r)$,
(b) $l(\mathcal{R}) \leq \min_{r \in \mathcal{R}} d(r)$,
(c) $u(\mathcal{R}) = d(r) = l(\mathcal{R})$, if r is the only element in $\mathcal{R}$.

Properties (a) and (b) ensure that $u(\mathcal{R})$ and $l(\mathcal{R})$ appropriately bound all candidate solutions in $\mathcal{R}$ from above and from below, whereas (c) guarantees the algorithm to converge to the optimal solution. As shown in problem (1) our goal is to minimize a distance function. Hence $u(\mathcal{R})$ in this case is not relevant for the minimization. However, we can use $u(\mathcal{R})$ to prune the priority queue for speeding our search, *i.e.*, eliminate any state $\mathcal{S}$ that satisfies $l(\mathcal{S}) > u(\mathcal{R})$ [3].

Bounding individual histogram bins: Suppose $\mathbf{A}^+$ and $\mathbf{A}^-$ are the longest possible and shortest possible subsequence of $\mathbf{A}$ for a given rectangle set $\mathcal{R}$. We denote their K-bin *unnormalized* histograms as $\varphi_{\mathbf{A}^+} = \{h_1^+, \dots, h_K^+\}$ and $\varphi_{\mathbf{A}^-} = \{h_1^-, \dots, h_K^-\}$. Let $r \in \mathcal{R}$ be a rectangle in the search space representing two subsequences $\mathbf{A}[b_1, e_1]$ and $\mathbf{B}[b_2, e_2]$ with histograms $\{h_1, \dots, h_K\}$ and $\{k_1, \dots, k_K\}$. Considering both histograms of $\mathbf{A}^+, \mathbf{A}^-$ and $\mathbf{B}^+, \mathbf{B}^-$, we can represent the range for the b^{th} histogram bins as

$$0 \le h_b^- \le h_b \le h_b^+, \; 0 \le k_b^- \le k_b \le k_b^+. \tag{2}$$

For *normalized* histograms, we use the fact that $|\mathbf{A}^-| < |\mathbf{A}[b_1, e_1]| < |\mathbf{A}^+|$ and $|\mathbf{B}^-| < |\mathbf{B}[b_2, e_2]| < |\mathbf{B}^+|$, where $|\mathbf{X}| = \sum_{b=1}^{K} \varphi_{\mathbf{X}b}$ is summation over histogram bins of a sequence $\mathbf{X}$. Then we can rewrite (2) for the ranges of normalized bins $\widehat{h}_b = h_b / |\mathbf{A}[b_1, e_1]|$ and $\widehat{k}_b = k_b / |\mathbf{B}[b_2, e_2]|$:

$$0 \le \frac{h_b^-}{|\mathbf{A}^+|} \le \widehat{h}_b \le \frac{h_b^+}{|\mathbf{A}^-|}, \; 0 \le \frac{k_b^-}{|\mathbf{B}^+|} \le \widehat{k}_b \le \frac{k_b^+}{|\mathbf{B}^-|}. \tag{3}$$

Bounding distance between histograms: With the per-bin bounds, we show in the following exemplar constructions of bounds between histograms, *i.e.*, ℓ_1, intersection, and χ^2 distance, which have been widely applied to many tasks such as objection recognition [9,13] and action recognition [7,11,14,16,22].

1) Bounding ℓ_1 distance: Applying the operators min/max on (2), we get

$$\begin{aligned} \min(h_b^-, k_b^-) &\le \min(h_b, k_b) \le \min(h_b^+, k_b^+), \\ \max(h_b^-, k_b^-) &\le \max(h_b, k_b) \le \max(h_b^+, k_b^+). \end{aligned} \tag{4}$$

Reordering both the above inequalities, we obtain the upper bound u_b and lower bound l_b for the b^{th} histogram bin:

$$\begin{aligned} l_b &= \max(h_b^-, k_b^-) - \min(h_b^+, k_b^+) \\ &\le \max(h_b, k_b) - \min(h_b, k_b) = |h_b - k_b| \\ &\le \max(h_b^+, k_b^+) - \min(h_b^-, k_b^-) = u_b. \end{aligned} \tag{5}$$

Summing all the histogram bins, we obtain the bounds of the ℓ_1 distance for two unnormalized histograms $\varphi_{\mathbf{A}}, \varphi_{\mathbf{B}}$:

$$l_{\ell_1}(\mathcal{R}) = \sum_{b=1}^{K} l_b \le \underbrace{\sum_{b=1}^{K} |h_b - k_b|}_{d_{\ell_1}(\varphi_{\mathbf{A}}, \varphi_{\mathbf{B}})} \le \sum_{b=1}^{K} u_b = u_{\ell_1}(\mathcal{R}). \tag{6}$$

Similarly, we can obtain the bounds for normalized histograms $\widehat{\varphi}_{\mathbf{A}}, \widehat{\varphi}_{\mathbf{B}}$ by the same operations as (4) and (5):

$$\widehat{l}_{\ell_1}(\mathcal{R}) = \sum_{b=1}^{K} \widehat{l}_b \le d_{\ell_1}(\widehat{\varphi}_{\mathbf{A}}, \widehat{\varphi}_{\mathbf{B}}) \le \sum_{b=1}^{K} \widehat{u}_b = \widehat{u}_{\ell_1}(\mathcal{R}), \tag{7}$$

where

$$\widehat{l}_b = \max(\frac{h_b^-}{|\mathbf{A}^+|}, \frac{k_b^-}{|\mathbf{B}^+|}) - \min(\frac{h_b^+}{|\mathbf{A}^-|}, \frac{k_b^+}{|\mathbf{B}^-|}), \tag{8}$$

$$\text{and} \quad \widehat{u}_b = \max(\frac{h_b^+}{|\mathbf{A}^-|}, \frac{k_b^+}{|\mathbf{B}^-|}) - \min(\frac{h_b^-}{|\mathbf{A}^+|}, \frac{k_b^-}{|\mathbf{B}^+|}). \tag{9}$$

2) Bounding intersection distance: Given two normalized histograms $\widehat{\varphi}_\mathbf{A} = \{\widehat{h}_1, \dots, \widehat{h}_K\}$ and $\widehat{\varphi}_\mathbf{B} = \{\widehat{k}_1, \dots, \widehat{k}_K\}$, we define their intersection distance by the Hilbert space representation [24]:

$$d_\cap(\widehat{\varphi}_\mathbf{A}, \widehat{\varphi}_\mathbf{B}) = -\sum_{b=1}^{K} \min(\widehat{h}_b, \widehat{k}_b). \tag{10}$$

By (3) and (4), we can find its lower bound and upper bound:

$$l_\cap(\mathcal{R}) = -\sum_{b=1}^{K} \min(\frac{h_b^+}{|\mathbf{A}^-|}, \frac{k_b^+}{|\mathbf{B}^-|}) \quad \text{and} \quad u_\cap(\mathcal{R}) = -\sum_{b=1}^{K} \min(\frac{h_b^-}{|\mathbf{A}^+|}, \frac{k_b^-}{|\mathbf{B}^+|}). \tag{11}$$

3) Bounding χ^2 distance: The χ^2 distance has been proven to be a good metric to measure distance between two histograms for TCD due to its simplicity and efficiency. The χ^2 distance is defined as

$$d_{\chi^2}(\widehat{\varphi}_\mathbf{A}, \widehat{\varphi}_\mathbf{B}) = \sum_{b=1}^{K} \frac{(\widehat{h}_b - \widehat{k}_b)^2}{\widehat{h}_b + \widehat{k}_b}. \tag{12}$$

Incorporating the bounds $\widehat{l}_b$ and $\widehat{u}_b$ for normalized histograms in (8) and the inequalities in (3), we obtain the lower bound and upper bound for d_{χ^2} by summing throughout all histogram bins:

$$l_{\chi^2}(\mathcal{R}) = \sum_{b=1}^{K} \frac{(\max(0, \widehat{l}_b))^2}{h_b^+/|\mathbf{A}^-| + k_b^+/|\mathbf{B}^-|} \quad \text{and} \quad u_{\chi^2}(\mathcal{R}) = \sum_{b=1}^{K} \frac{\widehat{u}_b^2}{h_b^-/|\mathbf{A}^+| + k_b^-/|\mathbf{B}^+|}. \tag{13}$$

The derived lower and upper bounds clearly satisfy the bounding properties (a) and (b). To show how property (c) holds, one can consider the case that $\mathcal{R}$ contains only one rectangle r. Take the d_{ℓ_1} for example, when $r \in \mathcal{R}$ is the unique rectangle, we have $h_b^+ = h_b = h_b^-$ and $k_b^+ = k_b = k_b^-$, and thus Eq. (5) becomes $u_b = |h_b - k_b| = l_b$. Hereby we obtain $l_{\ell_1}(\mathcal{R}) = d_{\ell_1}(\varphi_\mathbf{A}, \varphi_\mathbf{B}) = u_{\ell_1}(\mathcal{R})$. One can show property (c) holds for other distances in a similar manner.

4 Extensions to Multiple TCD and Video Indexing

In the following we show how a simple modification of our proposed algorithm can be applied to multiple TCD and video indexing.

Discover multiple commonalities: For realistic sequences that often contain more than one commonality, we can discover multiple commonalties by applying Algorithm 1 repeatedly. Every time Algorithm 1 returns an optimal rectangle in the temporal search space that represents the best match. Once a commonality is found, we remove the corresponding rectangle from the search space and then begin over the search process to find the next best. The process continues until a desired number of rectangles have been retrieved or the returned matching distance $d(\cdot,\cdot)$ is greater than some threshold, which depends on the desire of applications.

Note that our implementation is different from the conventional multiple-object detection tasks [13]. In object detection, the whole spatial region is removed to search for the next object. In our case, we can not remove all the time-segments for both time sequences because we might miss some commonality at the same location. Instead, we position those rectangles to the bottom of the priority queue by imposing a large penalty to their scores. Using this strategy, we are able to handle *many-to-many* mapping, *i.e.*, $\mathbf{A}[b_1, e_1]$ can match multiple subsequences in $\mathbf{B}$ and vice versa.

Video indexing: A simple modification of the proposed B&B algorithm could be useful for efficient searching for a video with similar content. That is, given a query video, how to efficiently search for common subsequences in a longer video. Let $\mathbf{Q}$ be the query sequence we want to find in the target video $\mathbf{T}$. We can modify (1) by fixing one of the pairwise sequences:

$$\min_{b_t, e_t} \quad d\left(\varphi_{\mathbf{T}[b_t,e_t]}, \varphi_{\mathbf{Q}}\right) \quad \text{s.t.} \quad e_t - b_t \geq \ell. \tag{14}$$

The problem now becomes simpler but it still is an integer programming. Nevertheless, Algorithm 1 can be applied again to find the optimal match efficiently. Searching for multiple segments can also be done as discussed above. Note that we do not claim that this indexing algorithm is state-of-the-art. We just want to illustrate the versatility of our approach.

5 Experimental Results

We evaluated our approach on two experiments. First, we discovered common facial events in the RU-FACS database [5]. Second, we found multiple common human actions in CMU-Mocap dataset [1]. The code is available at `http://www.humansensing.cs.cmu.edu/software/tcd.html`.

5.1 Common Facial Events Discovery

This experiment evaluates the capability of our algorithm to find similar facial events in the RU-FACS database [5]. The RU-FACS database consists of digitized video and manual coding of 34 young adults. They were recorded during an interview of approximately 2 minutes duration in which they lied or told the truth in response to an interviewer's questions. Pose orientation was mostly frontal with moderate out-of-plane head motions. We selected the Action Unit (AU) 12 (*i.e.*, smile) from 15 subjects that had most occurrence of this facial AU. We collected 100 segments containing one AU-12 and other AUs, resulting in $4,950$ video sequence pairs with different subjects.

We represented features as the distances between the height of lips and teeth, angles for mouth corners and SIFT descriptors in the points tracked by Active Appearance Models (AAM) [33] (see Fig. 4(a) for an illustration). We built a 1,000-entry codebook on a random subset of 50,000 feature vectors (see Sec. 3.3).

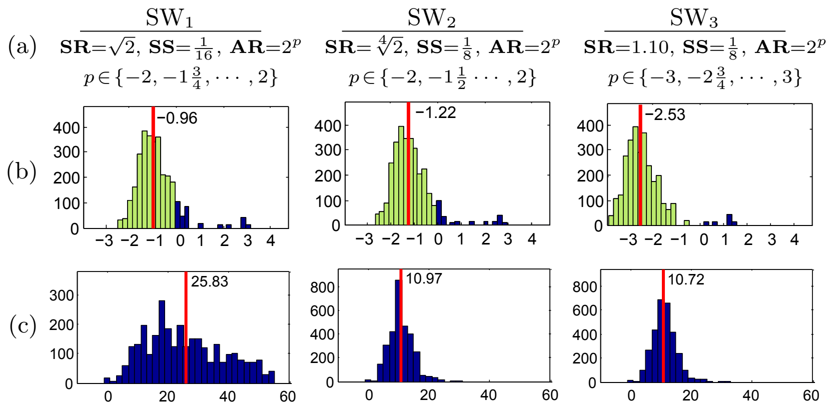
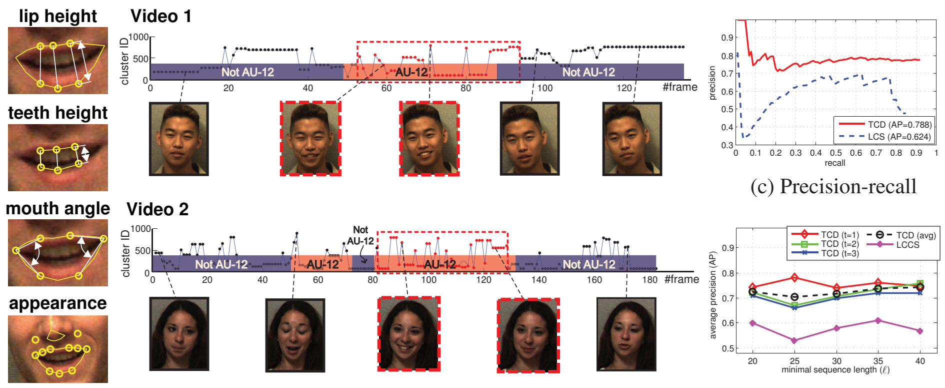


Fig. 3. Efficiency comparison between TCD and the naive sliding window (SW) approach. (a) Parameters for each SW_i: size-ratio (SR), stepsize (SS), and aspect ratios (AR) as 2^p. (b) Histograms ratio of the number of evaluation $\log \frac{n^{\mathrm{TCD}}}{n^{\mathrm{SW}_i}}$. (c) Histograms of difference between resulting distances $d(r^{\mathrm{SW}}) - d(r^{\mathrm{TCD}})$.

Efficiency comparison with the naive sliding window: This experiment evaluates the increase in speed in comparison with the naive sliding window (SW) approach. In the standard SW approach there are three parameter settings to improve efficiency [29]. We denote the parameters as SW_i $(i = 1, 2, 3)$; see Fig. 3(a) for detailed settings. The size-ratio (**SR**) refers to the window scaling factors, the stepsize (**SS**) is the window offset, and the ratio of the window width to its height is the aspect ratio (**AR**). We refer to [29] for more details about the parameters. Recall the lengths of two sequences are m,n and the minimal length for each sequence is ℓ. We fixed the maximal and the minimal rectangle sizes for SW to be $(m \times n)$ and $(\ell\sqrt{\mathbf{AR}} \times \frac{\ell}{\sqrt{\mathbf{AR}}})$, respectively.

To be independent of a particular implementation, we measured the *discovery speed* as the number of evaluations that TCD and SW need to compute the bounding functions. The number of evaluations are referred as n^{TCD} and n^{SW_i} $(i=1, 2, 3)$. Fig. 3(b) shows the histograms of the log ratio for $n^{\mathrm{TCD}}/n^{\mathrm{SW}_i}$. Light green bars show that TCD requires less evaluations than SW, while dark blue bars indicate the opposite. Red vertical line indicates the average ratio. The smaller the ratio value, the less times TCD has to evaluate the distance bounds. Although SW was parameterized by standard settings [13,29] to search only a subset of the search space, TCD searches the entire space yet still performs on average 6.18 times less evaluations than SW.

In order to evaluate the *discovery quality*, we also compared the difference between the distances measured by TCD and SW, *i.e.*, $d(r^{\mathrm{SW}}) - d(r^{\mathrm{TCD}})$. The larger the difference the worse results SW got. Fig. 3(c) shows the histograms of these differences. One can observe that the differences are always greater than or equal to zero. This is because our method always finds the global optimum. Recall that SW, depending on the parameter settings, only does a partial search, hence it is not surprising that it often performs worse than our method.

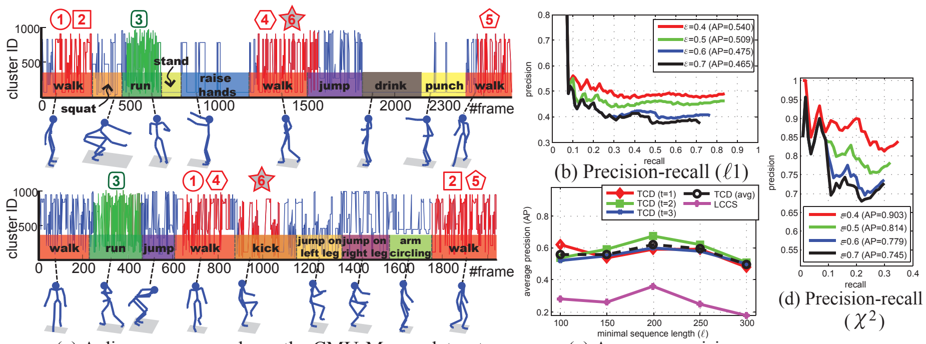


(a) (b) A discovery example on RU-FACS (c) Precision-recall (d) Average precision

Fig. 4. (a) Facial features extracted from the tracked points as in [33]. (b) An example of common discovered facial events (indicated by dashed-line rectangles). (c)(d) Accuracy evaluation on precision-recall and average precision (AP).

Accuracy evaluation: Because the problem of TCD is relatively new in computer vision, to the best of our knowledge there are no baselines we could compare to. Hence, for a baseline comparison, we selected the state-of-the-art method in longest common consecutive subsequence matching (LCCS) [31]. Observe that when the feature representation for each frame was quantized into a temporal word, the unsupervised TCD problem can be naturally interpreted as an LCCS. For fair comparisons with the LCCS that uses a 0-1 distance, in this experiment we used ℓ_1 distance. The minimal subsequence length ℓ was fixed to the same value for both LCCS and TCD. To evaluate the performance, we measured the *overlap score* between the ground truth and the discovered segments, as usually used in object detection tasks [9]: $\text{overlap}(r, g) = \frac{\text{area}(r \cap g)}{\text{area}(r \cup g)}$, where r is the rectangle in the search space representing a discovered commonality, and g is the ground true rectangle indicating the correct match. The higher the overlap score, the better the algorithm discovered the commonality. We consider that a rectangle is correct if the overlap score is greater than a threshold ε (here $\varepsilon = 0.5$). Fig. 4(b) shows an example of a correct discovery. We evaluated the event-level accuracy as precision and recall.

Fig. 4(c) plots the precision-recall curves for the first output result of TCD and LCCS. We computed the average precision (AP) [9] and found TCD outperforms LCCS by 15%. Compared to LCCS that finds identical subsequences, TCD considers a histogram appearing in two sequence, it is more robust to deal with uncertainty in noisy signals such as videos. Fig. 4(d) shows the average precision of our approach under different parameters. We varied the minimal sequence length ℓ in $\{20, 25, \ldots, 40\}$, and examined the AP of the t^{th} result individually. As can be observed from the averaged AP (black dashed line), our method is more robust across different settings of ℓ and t. As a result, TCD performed on average 16% better than LCCS in discovering the common AU-12.



(a) A discovery example on the CMU-Mocap dataset (b) Precision-recall ($\ell 1$) (c) Average precision (d) Precision-recall (χ^2)

Fig. 5. (a) An example of the top six discovered common motions. The numbers indicate discovered commonalities. Note that we shaded the star (number 6) to indicate an incorrect discovery that matched *walk* and *kick*. (b)(c) Precision-recall and average precision on ℓ_1 distance. (d) Precision-recall on χ^2 distance.

5.2 Multiple Common Motions Discovery in Motion Capture Data

In the second experiment we used the CMU-Mocap dataset [1] to demonstrate the ability to discover *multiple* common actions (as discussed in Sec. 4). We selected Subject 86 that contains 15 long sequences. Each sequence contains thousands of frames and up to 10 actions (out of a total of 25 human actions) such as walking, jumping, punching, etc. See Fig. 5(a) for an example. Each sequence ranges from 100 to 300 frames each action. Then we randomly selected 45 pairs of sequences (each having up to 10 actions) and discovered common actions among each pair. We downsampled each sequence by a factor of 4 to make it 30 fps, resulting in a set of sequences with 1,200∼2,600 frames. Note that this experiment is much more challenging than the previous one due to longer sequences and more complicated actions. In this case, we excluded SW for comparison because it needs 10^{12} evaluations which is impractical.

Each human motion was represented as the root position, orientation, and 29 relative joint angles. In order to provide a continuous representation, the 3-D Euler angles were transformed to 3-D quaternions. Following [4], we represented 10 joints as a 30-dimensional feature vector of 3-D quaternions for each frame.

We determined a correct discovery if its overlap score is greater than a threshold ε. Fig. 5(a) illustrates the first six common motions discovered by TCD. A failure discovery is shown in the shaded number 6. Fig. 5(b) shows the precision-recall curve for different values of ε. Using the naive ℓ_1 distance, the curve decreases about 10% AP when the overlap score ε raises from 0.4 to 0.7, which implies that we can obtain higher quality results without losing too much precision. For comparison with the baseline LCCS approach, Fig. 5(c) shows their APs over various ℓ on the n^{th} discovered result. LCCS performed poorly to obtain long common subsequences since in this experiment human motions have more variability than just one facial event (*e.g.*, AU-12). On the other hand, TCD utilized histogram representation, and thus allowed more tolerance in analogy with BoW in the context of object recognition. One can observe that AP drops with increasing ℓ since the common actions in this database can have very short distance, *e.g.*, *jump* and *squad*. Moreover, to demonstrate the generalization performance of our method, we also evaluated the χ^2 distance and plotted the precision-recall curve in Fig. 5(d). The bounds for χ^2 distance were discussed in Sec. 3.3. Although the Mocap dataset is very challenging in terms of various motions and diverse sequence lengths, our approach with χ^2 performed 30% better than ℓ_1 and LCCS. It shows χ^2 is a more powerful measurement for histograms than ℓ_1. Overall, using the χ^2 measurement and $\varepsilon = 0.5$, our algorithm achieved 81% precision.

6 Discussion and Future Work

This paper introduced the new problem of TCD, to find temporal commonalities between sequences in an unsupervised manner. We have shown that the proposed B&B algorithm can efficiently find a global optimal solution for TCD. We presented results in discovering common facial events and human actions. It is important to observe that our method can be applied to any features that can be quantified into histograms. Although this work has shown better performance than baseline methods, more research can be done to speed up the search process, *e.g.*, [2]. Currently, we are also looking to derive tight bounds for other metrics between temporal segments such as dynamic time warping.

Acknowledgements. This work was supported by the National Science Foundation (NSF) under Grant No. RI-1116583 and CPS-0931999. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

References

1. http://mocap.cs.cmu.edu/. 9, 12
2. S. An, P. Peursum, W. Liu, and S. Venkatesh. Efficient subwindow search with submodular score functions. In *CVPR*, 2011. 13
3. V. Balakrishnan, S. Boyd, and S. Balemi. Branch and bound algorithm for computing the minimum stability degree of parameter-dependent linear systems. *International Journal of Robust and Nonlinear Control*, 1(4):295–317, 1991. 6
4. J. Barbič, A. Safonova, J. Y. Pan, C. Faloutsos, J. K. Hodgins, and N. S. Pollard. Segmenting motion capture data into distinct behaviors. In *Proc. of Graphics Interface*, 2004. 12
5. M. S. Bartlett, G. C. Littlewort, M. G. Frank, C. Lainscsek, I. R. Fasel, and J. R. Movellan. Automatic recognition of facial actions in spontaneous expressions. *Journal of Multimedia*, 1(6):22–35, 2006. 9
6. O. Boiman and M. Irani. Detecting irregularities in images and in video. In *ICCV*, 2005. 1
7. W. Brendel and S. Todorovic. Learning spatiotemporal graphs of human activities. In *ICCV*, 2011. 7
8. W.-S. Chu, C.-P. Chen, and C.-S. Chen. Momi-cosegmentation: Simultaneous segmentation of multiple objects among multiple images. In *ACCV*, 2010. 1, 2
9. M. Everingham, A. Zisserman, C. I. Williams, and L. Van Gool. The PASCAL visual object classes challenge 2006 results. In *2th PASCAL Challenge*, 2006. 7, 11

10. D. Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge Univ Press, 1997. 3
11. D. Han, L. Bo, and C. Sminchisescu. Selection and Context for Action Recognition. In *ICCV*, 2009. 7
12. M. Hoai, Z. zhong Lan, and F. De la Torre. Joint segmentation and classification of human actions in video. In *CVPR*, 2011. 2, 3
13. C. Lampert, M. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *PAMI*, 31(12):2129–2142, 2009. 2, 5, 7, 9, 10
14. I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. 7
15. H. Liu and S. Yan. Common visual pattern discovery via spatially coherent correspondences. In *CVPR*, 2010. 1, 2
16. J. Liu, M. Shah, B. Kuipers, and S. Savarese. Cross-view action recognition via view knowledge transfer. In *CVPR*, 2011. 7
17. D. Maier. The complexity of some problems on subsequences and supersequences. *Journal of the ACM*, 25(2):322–336, 1978. 3
18. D. Minnen, C. Isbell, I. Essa, and T. Starner. Discovering multivariate motifs using subsequence density estimation. In *AAAI*, 2007. 3
19. A. Mueen and E. Keogh. Online discovery and maintenance of time series motifs. In *KDD*, 2010. 3
20. L. Mukherjee, V. Singh, and J. Peng. Scale invariant cosegmentation for image groups. In *CVPR*, 2011. 1, 2
21. M. Paterson and V. Dančík. Longest common subsequences. *Mathematical Foundations of Computer Science 1994*, 841:127–142, 1994. 3
22. S. Sadanand and J. J. Corso. Action bank: A high-level representation of activity in video. In *CVPR*, 2012. 7
23. G. Schindler, P. Krishnamurthy, R. Lublinerman, Y. Liu, and F. Dellaert. Detecting and matching repeated patterns for automatic geo-tagging in urban environments. In *CVPR*, 2008. 1
24. B. Scholkopf. The kernel trick for distances. In *NIPS*, 2001. 8
25. Z. Si, M. Pei, B. Yao, and S. Zhu. Unsupervised learning of event and-or grammar and semantics from video. In *ICCV*, 2011. 3
26. J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003. 6
27. K. Tang, L. Fei-Fei, and D. Koller. Learning latent temporal structure for complex event detection. In *CVPR*, 2012. 2
28. P. Turaga, A. Veeraraghavan, and R. Chellappa. Unsupervised view and rate invariant clustering of video sequences. *CVIU*, 113(3):353–371, 2009. 3
29. P. Viola and M. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004. 6, 10
30. Y. Wang, H. Jiang, M. S. Drew, Z. Li, and G. Mori. Unsupervised discovery of action classes. In *CVPR*, 2006. 2, 3
31. Y. Wang and S. Velipasalar. Frame-level temporal calibration of unsynchronized cameras by using Longest Consecutive Common Subsequence. In *ICASSP*, 2009. 3, 11
32. J. Yuan, Z. Liu, and Y. Wu. Discriminative video pattern search for efficient action detection. *PAMI*, 33(9):1728–1743, 2011. 2, 3, 5, 6
33. F. Zhou, F. De la Torre, and J. F. Cohn. Unsupervised discovery of facial events. In *CVPR*, 2010. 3, 9, 11
34. S. Zhu and D. Mumford. A stochastic grammar of images. *Foundations and Trends in Computer Graphics and Vision*, 2(4):259–362, 2006. 1