Feature-Independent Action Spotting Without Human Localization, Segmentation or Frame-wise Tracking*

Chuan Sun†, Marshall Tappen‡, Hassan Foroosh†
†Department of EECS, Division of Computer Science, University of Central Florida, USA
‡Amazon.com, Seattle, WA 98109
csun@eecs.ucf.edu, tappenm@amazon.com, foroosh@cs.ucf.edu

Abstract

In this paper, we propose an unsupervised framework for action spotting in videos that does not depend on any specific feature (e.g. HOG/HOF, STIP, silhouette, bag-of-words, etc.). Furthermore, our solution requires no human localization, segmentation, or framewise tracking. This is achieved by treating the problem holistically as that of extracting the internal dynamics of video cuboids by modeling them in their natural form as multilinear tensors. To extract their internal dynamics, we devised a novel Two-Phase Decomposition (TP-Decomp) of a tensor that generates very compact and discriminative representations that are robust to even heavily perturbed data. Technically, a Rank-based Tensor Core Pyramid (Rank-TCP) descriptor is generated by combining multiple tensor cores under multiple ranks, allowing to represent video cuboids in a hierarchical tensor pyramid. The problem then reduces to a template matching problem, which is solved efficiently by using two boosting strategies: (1) to reduce search space, we filter the dense trajectory cloud extracted from the target video; (2) to boost the matching speed, we perform matching in an iterative coarse-to-fine manner. Experiments on 5 benchmarks show that our method outperforms current state-of-the-art under various challenging conditions. We also created a challenging dataset called Heavily Perturbed Video Array (HPVA) to validate the robustness of our framework under heavily perturbed situations.

1. Introduction

The aim of action spotting is to spatiotemporally detect and localize a given query action within a larger search video. The intra-class variance and scene clutter make action spotting difficult. Some previous works combine tracking and classification for action localization, or treat action spotting and recognition in a joint manner [7, 16]. In terms of feature dependency, we observe several drawbacks hindering the performance of existing techniques. For example, *tracking-based* methods [1, 9] must track multiple body parts or joints and classify actions based on stable motion trajectories. But the tracker initialization and its robustness often impede a fully automatic operation, and manual intervention is therefore inevitable. *Contour/silhouette based* methods [12, 20] seek to extract features from the 3D space-time body shape. But it is often hard to perform robust segmentation for complex videos. For *optical flow-based* methods [2, 10], the dense flow estimates are unreliable when the scene is under camera motion, such as zooming, pan, or translation. *Space-time interest points based* methods [8, 21] are usually adopted to construct global bag-of-words descriptors. Although showing merits, they may fail to detect interest points within shadows, along object occluding boundaries, or in highly dynamic clutters.

A natural way to represent a video is to holistically encode its internal dynamics in a three-way tensorial form, rather than commonly-used vectorization. The multilinear tensor analysis, subsuming conventional linear analysis as a special case, emerges as a unifying powerful mathematical framework suitable for addressing problems in many fields [26]. In computer vision, the past decades witnessed many successful applications in areas such as face recognition [26], action recognition [25], action categorization and detection [15], etc.

Our motivation is to spot actions in a data-driven manner without relying on commonly-used features, aiming to avoid the problems pointed out above. In a nutshell, we treat all video cubiods involved as three-way tensors. We propose a new multilinear tensor decomposition called Two-Phase Decomposition (TP-Decomp) tailored for action spotting, by combining the Tucker decomposition with CANDECOMP/PARAFAC (CP for short) decomposition [17] in a natural yet effective way. We then establish a Rank-based Tensor Core Pyramid (Rank-TCP)

*This work was supported in part by the National Science Foundation under grants IIS-1212948 and IIS-091686.

descriptor using multiple tensor cores under multiple ranks, which is basically a new tensor-based hierarchical video representation. At the final template matching stage, we adopt two effective boosting strategies that requires no human localization, segmentation, or frame-wise tracking.

2. Two-Phase Decomposition (TP-Decomp)

The Tucker and CP decomposition are two powerful techniques that decompose tensors onto modes [17]. However, they have very different characteristics[1], and have been mostly treated as two distinct algorithms independently applied to various fields [3, 22, 26].

We establish a procedure called TP-Decomp that combines these two powerful techniques in a natural yet very effective way. Its resulting vectors shows remarkably good performance in terms of robustness and discriminative ability.

2.1. TP-Decomp procedure

Given a query video $\mathcal{Q} \in \mathbb{R}^{I\times J\times K}$ and a large target video $\mathcal{S}$, where I and J correspond to the spatial dimension of $\mathcal{Q}$, and K corresponds to the temporal dimension. The objective of action spotting is to find the best match for $\mathcal{Q}$ out of all sub-volumes of $\mathcal{S}$. We formulate this problem in a tensor-based framework. For the preliminaries of tensor definitions and operations, please refer to [17].

Since query video $\mathcal{Q}$ is a 3-way tensor, we first apply the Tucker decomposition to $\mathcal{Q}$, resulting in a smaller-sized tensor $\mathcal{G}$ and three factor matrices

$$\mathcal{Q} \approx \mathcal{G} \times_1 A \times_2 B \times_3 C = \sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{r=1}^{R} g_{pqr} a_p \circ b_q \circ c_r, \quad (1)$$

where $A \in \mathbb{R}^{I\times P}$, $B \in \mathbb{R}^{J\times Q}$, and $C \in \mathbb{R}^{K\times R}$ are the orthogonal factor matrices. The $\overline{\times}_i$ operator denotes the multiplication between a tensor and a vector in mode-i of that tensor, whose result is also a tensor, namely, $\mathcal{A} = \mathcal{B}\overline{\times}_i\alpha \Longleftrightarrow (\mathcal{A})_{jk} = \sum_{i=1}^{I}\mathcal{B}_{ijk}\alpha_i$.

The first-phase operator F_{tk} is defined as follows.

Definition 1: The F_{tk} operator is a mapping that transforms the input tensor $\mathcal{Q} \in \mathbb{R}^{I\times J\times K}$ into a tensor $\mathcal{G} \in \mathbb{R}^{R\times R\times R}$, namely, $F_{tk}(\mathcal{Q}) = \mathcal{G} = \mathcal{Q}\times_1 A^T \times_2 B^T \times_3 C^T$, where $R < \min\{I, J, K\}$, and A^T, B^T, C^T are transposes of the factor matrices in Eq.(1).

Recall that the CP decomposition factorizes a 3-order tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ into a sum of component rank-1 tensors $\mathcal{X} \approx \sum_{r=1}^{R} u_r \circ v_r \circ w_r$, where R is a positive integer and $u_r \in \mathbb{R}^I$, $v_r \in \mathbb{R}^J$, and $w_r \in \mathbb{R}^K$ for $r = \{1, ..., R\}$. If $P = Q = R = 1$, then the CP decomposition degenerates to the rank-1 decomposition.

We define the second-phase operator F_{cp} as follows.

[1]The Tucker algorithm is a form of higher-order PCA that decomposes a tensor into a core tensor multiplied by a matrix along each mode; while the CP decomposition factorizes a tensor into a sum of component rank-1 tensors [17]

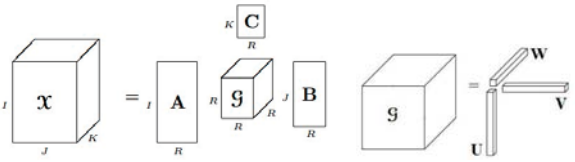

Figure 1. The illustration of Tucker decomposition followed by CP decomposition.

Definition 2: The F_{cp} operator is a mapping that transforms a cubical $\mathcal{G} \in \mathbb{R}^{R\times R\times R}$ into a quadruple, namely, $F_{cp}(\mathcal{G}) = [\![\lambda_R; U_R, V_R, W_R]\!]$, where λ is a scalar, and U, V, W are three vectors of sizes $1 \times R$.

We emphasis that, it is this second phase operator F_{cp}, which operates directly on $\mathcal{G}$, that distinguishes our TP-Decomp from conventional approaches. Schematically, the transformation flow is as follows:

$$\mathcal{Q} \xrightarrow{F_{tk}} \mathcal{G}_R \xrightarrow{F_{cp}} [\![\lambda_R^{\mathcal{Q}}; U_R^{\mathcal{Q}}, V_R^{\mathcal{Q}}, W_R^{\mathcal{Q}}]\!].$$

For notation purpose, we define a mapping function Ξ that maps the input volume $\mathcal{Q}$ to a quadruple

$$\Xi : \mathbb{R}^{I\times J\times K} \longrightarrow \{\mathbb{R}; \mathbb{R}^{1\times I}, \mathbb{R}^{1\times J}, \mathbb{R}^{1\times K}\}.$$

2.2. Theoretical insights

We now elaborate the theoretical insights of TP-Decomp. In F_{tk}, the $\mathcal{G} \in \mathbb{R}^{P\times Q\times R}$ is called the tensor core. Under a proper rank R, $\mathcal{G}$ can be considered as a compressed version of $\mathcal{Q}$ for 2 reasons: (1) The dimension of $\mathcal{G}$ could be much smaller than $\mathcal{Q}$. (2) The core is intuitively analogous to the diagonal singular value matrix in matrix SVD [2]. Hence, it encodes both the data variation and internal dynamics across its 3 orthonormal basis. In terms of action spotting, F_{tk} eliminates large amount of redundancy inherited in $\mathcal{Q}$, and retains, irrespective of $\mathcal{Q}$'s appearance, only the critical motion variation and internal dynamics in $\mathcal{G}$.

The essence of F_{cp} in the second phase is to directly apply rank-1 CP decomposition to $\mathcal{G}$ itself. This step is effective because F_{cp} is theoretically guaranteed by the fact that, for a 3-way tensor, the property of *rotational uniqueness* holds for CP decomposition [18], i.e, there is *one and only one* possible combination of rank-one tensors that sums to $\mathcal{G}$.

Thus, decomposing the core itself yields even more compact vectors than independently applying either the Tucker or the CP algorithm. The resulting rank-1 vectors U_R, V_R, and W_R are equal-sized, unique, and carry key information from the core. The λ_R in quadruple is of little significance since it is a scalar that can be absorbed into the factor vectors.

2.3. Properties

TP-Decomp has three good properties. We mainly focus on the robustness of TP-Decomp stated in Property 3.

[2]Note, however, that pure analogy between matrix SVD and tensor SVD is not well-established because high-order SVD shows far more complicated behavior than the matrix SVD [26].

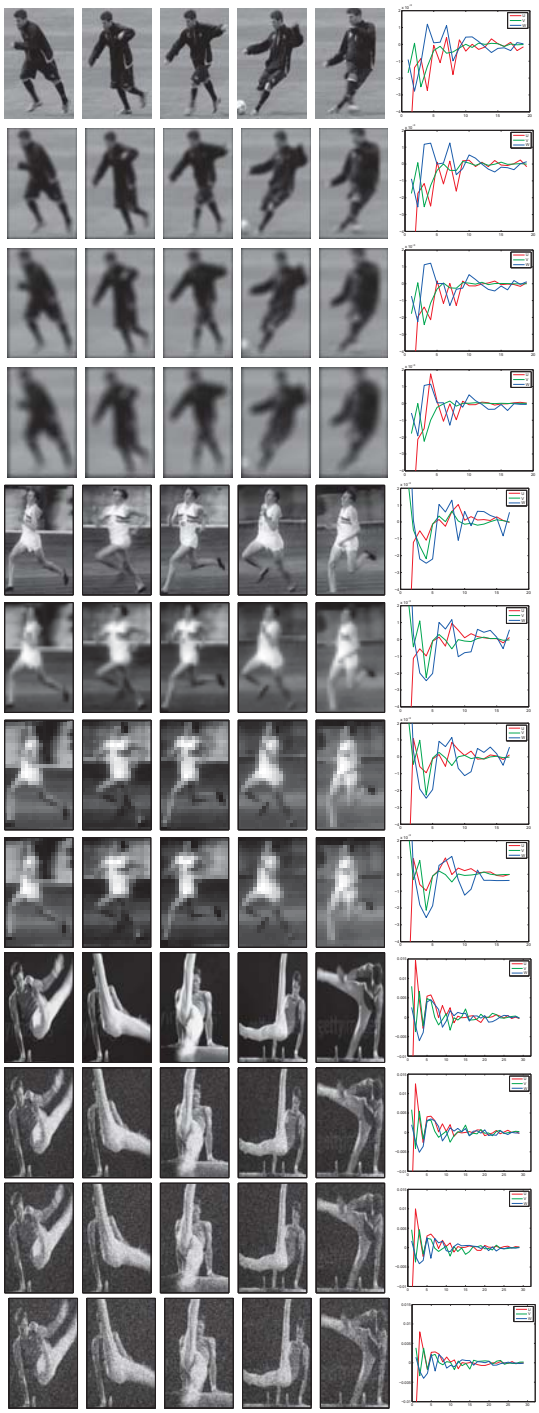

Figure 2. Comparison of the resulting quadruples for *kicking, running* and *benchswing* actions under three types of operations: *Gaussian blurring, down-sampling*, and *noise perturbation*. For Gaussian blurring, the $2nd-4th$ rows correspond to different blur amount. For scaling, the $6th-8th$ rows correspond to scaling ratios of $10\%, 8\%$, and 7%, respectively. For noise perturbation, the $10th-12th$ rows correspond to noise variance values of $0.04, 0.07$, and 0.1, respectively. All curves in the last column show the dynamics of the resulting vectors U, V, and W.

Property 1: The TP-Decomp produces very compact representations. By contrast, the dimensionality is reduced from $I \times J \times K$ for $\mathcal{Q}$, to R^3 for $\mathcal{G}$, and finally to $3R$ for $\Xi_R^{\mathcal{Q}}$, namely $IJK \rightarrow R^3 \rightarrow 3R$.

Property 2: The TP-Decomp is invariant to spatiotemporal dimension permutation for query video. Holistically treating video cuboid as tensor, we use a uniform rank R for both spatial and temporal dimensions. By rotational uniqueness property [18], when permutating the spatiotemporal dimension of a query video, both the tensor core and the factored matrices remain invariant. This leads to stable quadruples in F_{cp}. Practically, this property is useful to spot the cubiods that are rotated, mirrored, or transposed along certain axis in the target video.

Property 3: Under proper rank R, the TP-Decomp is robust to Gaussian blurring, downscaling, and noise perturbation. What happens when the target video is degraded by intensive blurring, low resolution down-sampling, or drastic noise perturbation? Many approaches will possibly fail to answer this due to feature dependency. For example, heavy Gaussian blurring removes high frequency intensities in image. It may affect optical flow extraction that highly depends on pixel gradients. The low resolution down-sampling removes intermediate pixels in frames. It may affects the HOG/HOF features whose statistical performance almost entirely depends on redundant sub-image patches. The heavy noise perturbation adds random contaminations in video frames. It may affect the effective extraction of the silhouettes or STIP features in space-time. In addition, bag-of-features or bag-of-words paradigms, which rely on statistical clustering of sufficient descriptors, may also be hard to classify the codebooks under heavily contaminated situations.

For illustration purposes, we tested TP-Decomp on 3 action classes: *kicking* (KTH), *running* (UCF Sports), and *benchswing* (UCF Sports), as shown in Fig.2. We applied TP-Decomp on $\mathcal{Q}$ under rank $R = 20$, and plotted the resulting triplets as curves. For the Gaussian blurring degradation, under various smoothing kernels, we found that the resulting quadruple of *kicking* largely captures the action dynamics, with merely slight fluctuations in quadruple. For down-sampling, we downscaled the *running* video into various low resolution ones. Even when over 90% of the frame pixels were missing, the *running* dynamics was still well-preserved. For noise perturbation, we contaminated the *benchswing* with Gaussian noises. Even when the frames were heavily contaminated, the resulting triplets remained resilient to noise.

To fully investigate how the TP-Decomp performs for action spotting under heavy perturbation, we create a challenging dataset called Heavily Perturbed Video Array (HPVA). We will describe it in detail in experiment section.

Algorithm 1: Rank-TCP descriptor construction

input : A query video $\mathcal{Q} \in \mathbb{R}^{I \times J \times K}$ where I and J correspond to spatial dimension of frames, and K corresponds to temporal dimension

output: Tuple $\{\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}\}$.

Initialize $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$ to empty vectors;

for *index* e *from* 2 *to* 7 **do**

  Rank $R = 2^e$;

  if $R \leq \min\{I, J, K\}$ **then**

    Apply F_{tk} operator to $\mathcal{Q}$ using rank R;

    Compute core $\mathcal{G}_R = \mathcal{Q} \times_1 A^T \times_2 B^T \times_3 C^T$;

    Apply F_{cp} operator to $\mathcal{G}_R$;

    Compute quadruple $[\![\lambda_R; U_R, V_R, W_R]\!]$;

    Concatenate U_R to the rear of $\boldsymbol{U}$;

    Concatenate V_R to the rear of $\boldsymbol{V}$;

    Concatenate W_R to the rear of $\boldsymbol{W}$;

  end

end

3. Rank-TCP descriptor

3.1. Motivation

The core dimension is solely decided by the rank R, yet the final quadruple is solely decided by the core. Hence, the rank R is the ultimate factor that affects both core and quadruple dynamics.

However, there is no prior on which R leads to good representation for action dynamics in core/quadruple, because determining tensor rank is NP-hard [11], and the behavior of higher-order SVD is far beyond well-understood [26]. In the extreme cases, for instance, if R is too small ($R \leq 3$), the core will be too small to capture sufficient information, whereas if R is too large, the F_{tk} operator may become undefined by *Lemma 1*; and the F_{cp} operator may not be unique by *Lemma 2*, as shown in the Appendix.

For this reason, we experimentally validate how ranks affect the TP-Decomp on a realistic big dataset called CCWebVideo, as stated in Section 5. Our results on five classes (2471 videos) show that, combining quadruples from multiple ranks outperform that of individual ranks (Fig.5). This is what motivates us to establish a Rank-based Tensor Core Pyramid to fully characterize an action.

3.2. Construction procedure

We establish a new tensorial coarse-to-fine pyramid using multiple cores under multiple ranks. Smaller ranks correspond to cores of smaller size, lying above the larger ones in the pyramid. Under each candidate rank R, we apply the TP-Decomp on the query tensor $\mathcal{Q}$, yielding its corresponding quadruple $\Xi_R^{\mathcal{Q}} = [\![\lambda_R^{\mathcal{Q}}; U_R^{\mathcal{Q}}, V_R^{\mathcal{Q}}, W_R^{\mathcal{Q}}]\!]$.

Cores with lower ranks coarsely encode the dynamics in $\mathcal{Q}$, whereas cores with higher ranks encode dynamics of $\mathcal{Q}$ more precisely. But in terms of computational burden, the larger the R, the longer the TP-Decomp takes. Practically, we consider the following candidate ranks

$$R = \{4, 8, 16, 32, 64, 128\}.$$

This choice stems from practical concerns. All 6 ranks are unevenly distributed in a sense that we lean towards lower rank spectrum while not losing higher ranks. This choice can weight the computational burden of TP-Decomp.

For the 6 pyramid layers, there are totally $6 \times 3 = 18$ vectors generated. At the ith layer, the size of the quadruple is $3 * 2^i + 1$. Our final feature descriptor is the concatenation of the quadruple in all pyramid layers. We concatenate the U, V, W vectors at each level, yielding three long vectors $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$, namely

$$\begin{cases} \boldsymbol{U} = [U_4, U_8, ..., U_{128}] \\ \boldsymbol{V} = [V_4, V_8, ..., V_{128}] \\ \boldsymbol{W} = [W_4, W_8, ..., W_{128}]. \end{cases}$$

The triplet $\{\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}\}$, with each element of size $\sum_i 2^i = 252, i = [2, \cdots, 7]$, will be fed into the final template matching phase. If a candidate rank R is larger than the size of query tensor, namely $R > \min\{I, J, K\}$, then we let R_{max} be the largest candidate rank, which is smaller than the query tensor size. Then each element in the triplet $\{\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}\}$ is of size $\sum_i 2^i, i = \{2, \cdots, log_2 R_{max}\}$.

Experimental results in CCWebVideo dataset shows that combining multiple ranks outperforms single rank representations (Fig.5). Intuitively, by combining multiple cores, the pyramid achieves a rich and redundant representation for video.

The procedure to build the Rank-TCP descriptor is shown in **Algorithm 1**. The storage space saved in TP-Decomp is from $I \times J \times K$ to $3R+1$. For Rank-TCP descriptor, the space complexity of the final quadruple is $\sum_{e=2}^{\min\{I,J,K\}} (3 * 2^e + 1)$. In the case that $\min\{I, J, K\} \geq 128$, the total size for the pyramid is 756.

4. Boosting strategies for matching

Although costly, template matching can avoid problematic preprocessing operations in localization, tracking, and segmentation [14]. The computational burden can be reduced by various techniques such as branch-and-bound [31] or voting algorithms [29].

In our framework, two strategies are adopted to boost the template matching: (1) to reduce the space complexity, we employ the dense trajectory in [13], and prune the search space using cues derived from trajectories. (2) to reduce the time complexity, we use a coarse-to-fine strategy assisted by the Rank-TCP descriptor.

4.1. Trajectory-assisted space reduction

Motion is the most reliable and informative source of information for action analysis [27]. The method using

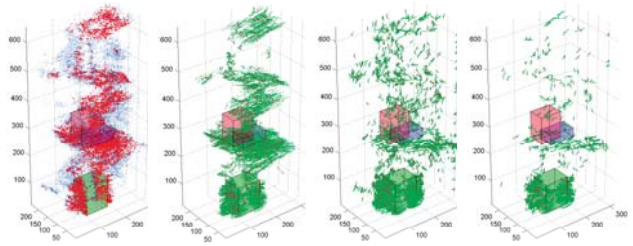

Figure 3. ($1st$) The point cloud formed by the dense trajectories extracted from a video of MSR I dataset. Red points map to top trajectories by length. ($2nd$) Top trajectories thresholded by the mean length of all trajectories. ($3rd$) Top trajectories thresholded by the mean unsigned total curvature of all trajectories. ($4th$) Our filtered trajectories. The red dots denote the average locations of trajectories. The red, green, and blue boxes denote the ground truth volume for *clapping, waving*, and *boxing*, respectively (zoom in for better view)

dense trajectories to compute local descriptor in [27] is one of the state-of-the-art approaches for action recognition. The work in [13] further improves the work of [27] and shows high reliability and robustness in handling motion.

Given a target video, we first extract its dense trajectories by the method in [13] using default parameters. Instead of being used in codebooks for k-means clustering as in [13, 27], the dense trajectories are treated as reliable cues to prune the huge matching space. Our underlying intuition of this trajectory-assisted matching agrees with that of [19]: despite the huge number of candidate cuboids needing search, only very few contain the true motion of interest (MOI). Instead of exhaustively evaluating them all, we target only the best few.

Inspecting thoroughly, we observe the motion annotated by trajectories can be roughly divided into 8 categories: (a) limb movement around MOI, such as the hands in handwaving, the head and feet in jumping, etc. (b) camera motion, such as zoom, pan, translation, vibration, etc. (c) non-MOI motion inside blobs. (d) patch motion inside MOI, such as motion caused by appearance (e.g. cloth) in bending or jumping, non-limb body region motion, etc. (e) background clutter, such as cars on street, crowd, remote or close pedestrians, etc. (f) foreground motion, such as a pedestrian moving from left to right across the scene, etc. (g) local motion propagation, such as a fountain in background, etc. (h) regional random movement of noisy patches.

Out of those 8 categories, we are especially interested in the first. The category (b) is effectively canceled out by method in [13], and few trajectories are responsible for it. Further observation reveals that, short trajectories stem from $C_{short} = \{c, d, e\}$, long trajectories stem from $C_{long} = \{a, d, e, f\}$, curved trajectories stem from $C_{curved} = \{a, e, g, h\}$, and relatively straight trajectories stem from $C_{straight} = \{c, d, e, f\}$. Notice that

$$C_{long} \bigcap C_{curved} = \{a\}.$$

This observation suggests that, a long curved trajectory indicates a possible MOI around this trajectory. This inspires us that, the length and the curvature of trajectory is possibly a very good cue to spot the MOI.

In practice, given a trajectory denoted by $T = \{p_i\}, i = 1, ..., n$ and a neighborhood m, we calculate its tangent orientation based discrete curvature [4] at point p_i by

$$k(p_i) = \frac{\angle(p_{i-m}p_i, p_ip_{i+m})}{|p_{i-m}p_i| + |p_ip_{i+m}|}.$$

Since curvature is signed, the total curvature of a S-shaped trajectory is possibly zero. We then use unsigned total curvature $\kappa = \sum |k(p_i)|$ to reflect the total "bendness" of a trajectory. Along with its total length $l = \sum |p_{i+1} - p_i|$, we define

$$\varsigma = \kappa l$$

as our metric to measure the trajectories, and define the threshold as $\theta = mean(\varsigma_j), j = 1, ..., t$, where t is the total number of extracted trajectories. This metric is unsupervised, simple, yet powerful enough to prune a significant number of irrelevant trajectories, as shown in Fig.3. The mean position of all survived trajectories are used as search locations in template matching.

4.2. Coarse-to-fine matching

The trajectory-assisted strategy prunes a considerable amount of search locations. Let the set of survived search locations be $L = \{L_i\}, i = 1..n$. Due to the coarse-to-fine structure in Rank-TCP descriptor, we can further accelerate the matching in an iterative coarse-to-fine manner.

Since the descriptor with lower rank coarsely represent the candidate volume, we first match all locations in L using the lowest rank $R = 4$. Within the resulting 3D score map, we set a loose threshold $\theta_{R=4}$, filtrating all matched candidate cuboids below this threshold. For the survived candidates $C_{R=4}$, we apply the second round of matching under a higher rank $R = 8$. Under a proper second threshold $\theta_{R=8}$, a smaller portion of the survived candidates from last round, denoted by $C_{R=8}$, survives, and participates in the next level where $R = 16$. We iterate this process m times ($m <= 6$). Under lower ranks ($R = \{4, 8\}$), there could be numerous false alarms, because lower ranks correspond to far more cuboids needing match. As rank grows ($R = \{16, 32, 64, 128\}$), many false alarms will be filtered out. To ensure true positives (locations around MOI) are not falsely filtered out, the thresholds under lower ranks, i.e., $\theta_{R=4}$ and $\theta_{R=8}$, will not be set too tight; but as rank grows, more strict thresholds will be enforced to eliminate false positives. In practice, this strategy can reduce $15\% - 30\%$ matching time.

4.3. Matching measure

To define the similarity measure between the query tensor $\mathcal{Q}$ and candidate sub-volume $\mathcal{V}$ in search video $\mathcal{S}$, we

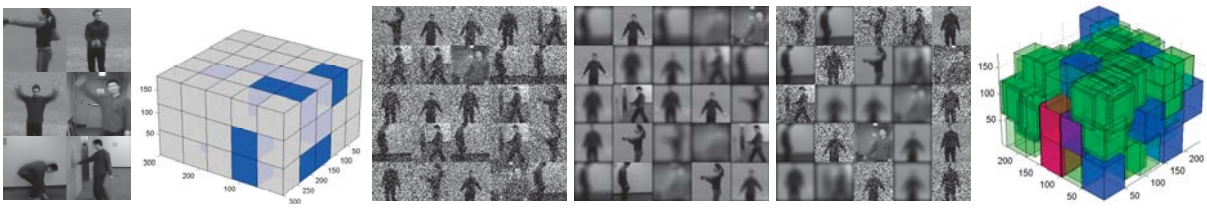

Figure 4. Our HPVA dataset. $(1st)$ The 6 seed videos. We use *boxing, clapping, waving* as queries, and the rest as outliers. $(2nd)$ The $3D$ tiled cuboid grid of size $300 \times 300 \times 180$. Each cell is filled with a random seed video. $(3rd)$ One frame of a video grid perturbed by random heavy noise; $(4th)$ One frame of a video grid affected by random degree of blurring; $(5th)$ A random mixture of noise and bluring perturbation; $(6th)$ The exhaustive matching on a randomly generated grid. Blue/red cuboids denote the ground truth position of queries (eg. *boxing, clapping, waving*), while green cuboids denote sliding windows

extract the Rank-TCP descriptor out of both $\mathcal{Q}$ and $\mathcal{V}$, resulting in two quadruples

$$\begin{cases} \Xi_R^{\mathcal{Q}} = [\![\lambda_R^{\mathcal{Q}}; U_R^{\mathcal{Q}}, V_R^{\mathcal{Q}}, W_R^{\mathcal{Q}}]\!] \\ \Xi_R^{\mathcal{V}} = [\![\lambda_R^{\mathcal{V}}; U_R^{\mathcal{V}}, V_R^{\mathcal{V}}, W_R^{\mathcal{V}}]\!]. \end{cases}$$

We define the matching measure function as $D(\Xi_R^{\mathcal{Q}}, \Xi_R^{\mathcal{V}}) = \phi(U_R^{\mathcal{Q}}, U_R^{\mathcal{V}}) + \phi(V_R^{\mathcal{Q}}, V_R^{\mathcal{V}}) + \phi(W_R^{\mathcal{Q}}, W_R^{\mathcal{V}})$, where the function ϕ is the Euclidean distance measure function.

We define the similarity score between $\mathcal{Q}$ and $\mathcal{V}$ under rank R as $\xi_R = -\log(D(\Xi_R^{\mathcal{Q}}, \Xi_R^{\mathcal{V}}))$. The score is inversely proportional to the measure function D. A high score means the two tensors are similar while a low score implies dissimilarity. The peaks of the similarity score across the 3D score volume indicate potential match locations.

5. Experimental evaluation

We extensively experiment on 5 benchmarks. For CCWebVideo, the retrieval requires no window sliding, since we holistically match query and target. For the remaining 4 datasets, following [14], we match at a single scale instead of multiple scales since actor sizes are stable. We use exhaustive search as our baseline denoted by "RTCP-Exha", and the trajectory-assisted matching is denoted by "RTCP-Traj".

CCWebVideo Dataset This huge dataset [28] contains 24 set of 13129 videos. For each seed video, there are hundreds of different versions, with considerable amount of intra-class variation (such as photometric variations, lighting change, unrelated frames, text overlay, etc.). We use this dataset to verify how our Rank-TCP descriptor performs given huge intra-class variance. We select 5 classes, totally 2471 videos. Since action spotting is analogous to video retrieval, we treat the seed as query video, and spot all videos that matche the seed. We test 5 descriptors, of which 4 use a single rank $(4, 8, 16, 32)$, and 1 combines these multiple ranks.

HPVA	Noise	Blur	Mixed
RTCP-Exha (baseline)	0.44	0.37	0.21

Table 2. Average precision for HPVA results

	CCWebVideo	CMU	MSR I	MSR II	HPVA
RTCP-Exha (baseline)	23	18	15	n/a	0.8
RTCP-Traj wo/ c2f	n/a	2.0	2.3	7.2	n/a
RTCP-Traj w/ c2f	n/a	1.2	1.5	6.3	n/a

Table 3. Time (hours) spent in 3 template matching scenarios for 5 benchmarks. "c2f" means using coarse-to-fine matching strategy

CMU Action Spotting Dataset The CMU action dataset [14] consists of 5 action classes (48 videos, 6 subjects): *jumping jacks*, *pickup*, *push button*, *one-handed wave*, and *two-handed wave*. Videos are recorded in crowded environments such as streets, restaurants, and bus stop. We compare with 3 baseline methods: the holistic flow [23], the part-based shape plus flow [14], and the space-time oriented energy measurements [7].

MSR Action Dataset I This dataset [31] contains 3 classes (16 video sequences, 10 subjects). The video sequences and has in total 63 actions: 14 hand clapping, 24 hand waving, and 25 boxing. Each sequence contains multiple types of actions. There are both indoor and outdoor scenes with clutter and moving backgrounds. The **MSR Action Dataset II** is an extended version of the MSR I. It contains 3 classes (54 video sequences): hand waving, hand clapping, and boxing. In total 203 action instances. Since instances of a query often do not begin and end at the same time span [14], we preserve only the best in each frame.

Heavily Perturbed Video Array (HPVA) Dataset We created a new challenging dataset to test the robustness of our framework under heavily perturbed situations. We choose 6 seed videos (*boxing, clapping, waving,* 1-*handed wave, pickup, pushbutton*), each of size $60 \times 60 \times 60$. We form a $300 \times 300 \times 180$ 3D *tiled cuboid grid* in space-time, as shown in Fig.4 (Each cell contains a random seed). Since down-sampling can be regarded as a variation of Gaussian blurring, we apply one of the two operators, F_{noise} or F_{blur}, on seeds. The degree of F_{noise} and F_{blur} are randomly specified, and which cell maps to which seed is also randomly generated. We created 3 such grids in HPVA, of which 2 are enforced by F_{noise} and F_{blur}, respectively, and the other one by mixing such 2 operations. In other words, our benchmark has three videos ($300 \times 300 \times 180$). The first is under randomly perturbed noise. The second is degraded by random intensive blurring. The last is a random

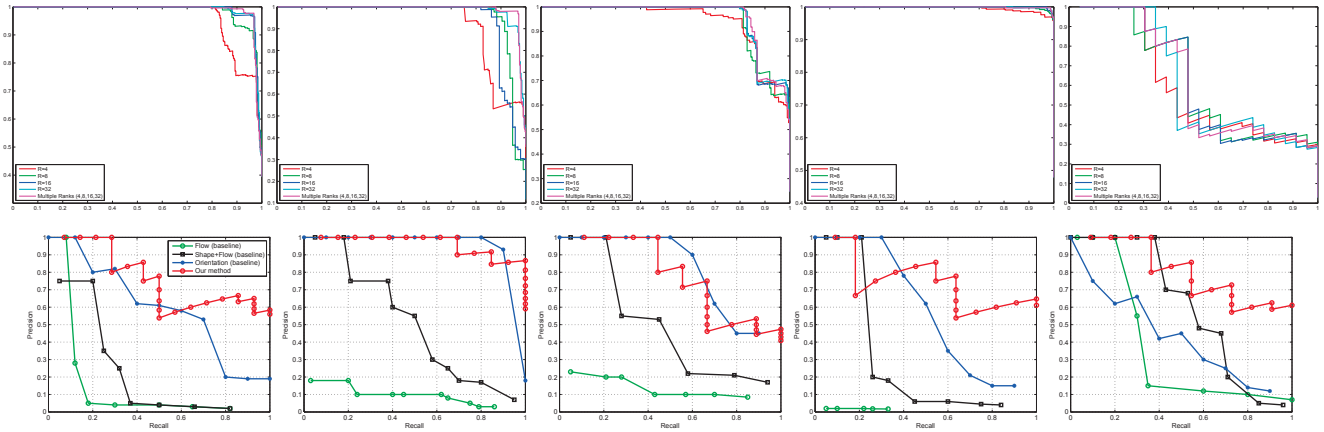

Figure 5. The Precision-Recall Curve for CCWebVideo ($1st$ row) and CMU Action dataset ($2nd$ row). For CCWebVideo dataset, each figure denotes the comparison results between using a single rank and multiple ranks. From left to right, the action classes are: *The lion sleep tonight, Evolution of dance, Fold shirt, I will survive Jesus*, and *Little superstar*. Note that our PR curves for *Little superstar* is more jagged than others because there is only 59 ground truths available out of 377 videos (84% are outliers). For CMU Action, we compare our method with 3 baseline methods, namely, the holistic flow [23], the part-based shape plus flow [14], and the spacetime oriented energy measurements [7]. The 5 subfigures correspond to *jumping jacks*, *pickup*, *push button*, *one-handed wave*, and *two-handed wave*, respectively. Red curves correspond to our proposed method (better be viewed by zooming in and in color)

CCWebVideo	"The lion sleep tonight"	"Evolution of dance"	"Fold shirt"	"I will survive Jesus"	"Little superstar"
Total video #	792	483	436	416	377
Outlier #	458	361	253	29	318
Outlier %	58%	75%	58%	7%	84%
Wu [28]	0.95	0.90	0.86	0.88	0.78
Song [24]	0.94	0.79	0.92	0.94	0.94
RTCP-Traj	**0.97**	**0.94**	**0.93**	**0.98**	**0.81**

Dataset	CMU					MSR I			MSR II		
Actions	jjacks	pickup	pushbutton	1-h wave	2-h wave	clapping	waving	boxing	clapping	waving	boxing
Video instance #	16	20	14	18	34	14	24	25	51	71	81
Deerpanis [7]	0.50	0.95	0.80	0.55	0.48	-	-	-	-	-	-
Ke [14]	0.30	0.45	0.50	0.40	0.60	-	-	-	-	-	-
Yuan [31]	-	-	-	-	0.75	-	-	-	-	-	-
Yu [30]	-	-	-	-	-	-	-	-	0.30	0.84	0.60
Boyraz [5]	-	-	-	-	-	-	-	-	0.35	0.40	0.50
RTCP-Exha (baseline)	0.55	0.59	0.35	0.40	0.57	0.19	0.43	0.25	-	-	-
RTCP-Traj	**0.75**	**0.92**	**0.74**	**0.69**	**0.78**	**0.63**	**0.82**	**0.70**	**0.57**	**0.73**	**0.64**

Table 1. Average Precision (AP) comparison for CCWebVideo, CMU, MSR I, and MSR II datasets

mixture of noise and blurring. We choose *boxing, clapping, waving* as queries and the rest as outliers.

5.1. Analysis

Average precision (AP) For CCWebVideo, despite the large amount of outliers and intra-class variations, the PR curves of our spotted (or retrieved) results show remarkably high precisions. The precision of higher ranks (32) is often higher than lower ranks $(4, 8)$. The combination shows highest AP amongst all 5 cases. For 4 out of 5 seeds, our AP outperform the state-of-the-art (Table 1). The results for 3 out of 5 actions in CMU data outperforms previous works. Higher ranks indeed lead to richer representations, and combining multiple ranks is better than a single one.

Using RTCP-Exha, our AP on CMU data outperform 2 out of 5 actions. Using RTCP-Traj, our AP improves for all 5 actions. Especially, we observe that AP of *pickup* action increases 35% compared to baseline. This action involved a large body motion when the person's upper body approaches to the ground. By inspecting the filtered trajectories of *pickup*, we observed that a considerable amount of irrelevant trajectories were filtered out, leading to largely reduced number of false alarms.

The MSR I and MSR II are challenging because of their long durations and dynamic backgrounds. Some previous works did not provide AP per action, so they are unavailable in Table 1. Note that, we observe the hands of the *clapping* in some videos often overlap the inner region of human body, and lead to some level of intensity confusions. Thus for *clapping*, our AP is relatively low using exhaustive matching without pruning. Because of the matching time ($>$ 40 hours), the exhaustive matching

results for MSR II is unavailable in our test. Our reported trajectory-assisted matching for MSR II took about 6.3 hours with coarse-to-fine boosting (Table 3). Overall, our AP is comparable to that of [30]. Across all test scenarios, RTCP-Traj largely outperforms RTCP-Exha, showing that our trajectory-assisted pruning is indeed effective.

We perform only exhaustive matching on HPVA, because the extracted trajectories on HPVA can hardly represent motion dynamics due to heavy contamination in videos. The results are shown in Table 2. Due to the robustness of TP-Decomp stated in Section 2, our method still spotted many true positives, regardless of the randomness and contamination introduced in HPVA.

Matching time In practice, we parallelized template matching on a cluster using multiple cores. We compare the time spent in matching in Table 3. Along with the AP analyzed above, our trajectory-assisted space reduction is an effective way to reduce search space and boost the spotting precision. The adaptation of coarse-to-fine (c2f) can reduce around $15\% - 30\%$ of the matching time on average compared to when c2f is not used.

6. Conclusions

We propose an action spotting framework that is feature-independent and does not rely on human localization, segmentation, or frame-wise tracking. We start by treating all involved video cuboids as multilinear tensors, and theoretically and experimentally show that, the internal dynamics of an action can be effectively encoded by our new Two-Phase Decomposition technique. We further verify that combining multiple cores under multiple ranks lead to enhanced performance compared to single rank. This inspired us to devise hierarchical rank-based descriptors to fully represent action dynamics. We boost the costly template matching by two strategies, which reduce the size of search space and the matching time. The experimental results on 5 benchmarks, including our newly-created HPVA dataset, show that our framework is very effective in spotting actions under various challenging conditions. We conclude that, (1) Our TP-Decomp method yields compact, discriminative, and robust features. (2) Rank-TCP is effective in yielding richer and reliable representations. (3) Filtering out irrelevant outliers in matching volume, targeting only the best few, indeed leads to largely boosted speed and enhanced precisions. (4) A robust descriptor that preserves action dynamics is critical for spotting actions under heavily perturbed situations.

Appendix

Lemma 1: For a 3-order tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$, only the following weak upper bound on its maximum rank is known [18]: $rank(\mathcal{X}) \leq min\{IJ, IK, JK\}$.

Lemma 2: For a 3-order tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$, the CP decomposition is generically unique if the following two conditions hold [6]: (1) $R \leq K$; (2) $R(R-1) \leq I(I-1)J(J-1)/2$, where R is the rank of $\mathcal{X}$.

References

[1] S. Ali, A. Basharat, and M. Shah. Chaotic invariants for human action recognition. In *ICCV*, pages 1–8. IEEE, 2007.

[2] S. Ali and M. Shah. Human action recognition in videos using kinematic features and multiple instance learning. *PAMI*, 32(2):288–303, 2010.

[3] B. W. Bader, M. W. Berry, and M. Browne. Discussion tracking in enron email using parafac. *Survey of Text Mining II*, pages 147–163, 2008.

[4] J. R. Bennett and J. S. Mac Donald. On the measurement of curvature in a quantized environment. *IEEE Trans on computers*, 24(8):803–820, 1975.

[5] H. Boyraz, M. F. Tappen, and R. Sukthankar. Localizing actions through sequential 2d video projections. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 34–39. IEEE, 2011.

[6] L. De Lathauwer. A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization. *SIAM Journal on Matrix Analysis and Applications*, 28(3):642–666, 2006.

[7] K. G. Derpanis, M. Sizintsev, K. Cannons, and R. P. Wildes. Efficient action spotting based on a spacetime oriented structure representation. In *CVPR*, pages 1990–1997. IEEE, 2010.

[8] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, pages 65–72. IEEE, 2005.

[9] C. Fanti, L. Zelnik-Manor, and P. Perona. Hybrid models for human motion recognition. In *CVPR*, volume 1, pages 1166–1173. IEEE, 2005.

[10] A. Fathi and G. Mori. Action recognition by learning mid-level motion features. In *CVPR*, pages 1–8. IEEE, 2008.

[11] J. Håstad. Tensor rank is np-complete. *Journal of Algorithms*, 11(4):644–654, 1990.

[12] Y. Hu, L. Cao, F. Lv, S. Yan, Y. Gong, and T. S. Huang. Action detection in complex scenes with spatial and temporal ambiguities. In *ICCV*, pages 128–135. IEEE, 2009.

[13] M. Jain, H. Jégou, P. Bouthemy, et al. Better exploiting motion for better action recognition. In *CVPR-International Conference on Computer Vision and Pattern Recognition*, 2013.

[14] Y. Ke, R. Sukthankar, and M. Hebert. Event detection in crowded videos. In *ICCV*, pages 1–8. IEEE, 2007.

[15] T.-K. Kim and R. Cipolla. Canonical correlation analysis of video volume tensors for action categorization and detection. *PAMI*, 31(8):1415–1428, 2009.

[16] J. Knopp, M. Prasad, G. Willems, R. Timofte, and L. Van Gool. Hough transform and 3d surf for robust three dimensional classification. *ECCV*, pages 589–602, 2010.

[17] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[18] J. B. Kruskal. Rank, decomposition, and uniqueness for 3-way and n-way arrays. *Multiway data analysis*, pages 7–18, 1989.

[19] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by subwindow search. In *CVPR*, pages 1–8. IEEE, 2008.

[20] Z. Lin, Z. Jiang, and L. S. Davis. Recognizing actions by shape-motion prototype trees. In *ICCV*, pages 444–451. IEEE, 2009.

[21] K. Rapantzikos, Y. Avrithis, and S. Kollias. Dense saliency-based spatiotemporal feature points for action recognition. In *CVPR*, pages 1454–1461. IEEE, 2009.

[22] A. Shashua and A. Levin. Linear image coding for regression and classification using the tensor-rank principle. In *CVPR*, volume 1, pages I–42. IEEE, 2001.

[23] E. Shechtman and M. Irani. Space-time behavior-based correlation-or-how to tell if two underlying motion fields are similar without computing them? *PAMI*, 29(11):2045–2056, 2007.

[24] J. Song, Y. Yang, Z. Huang, H. T. Shen, and R. Hong. Multiple feature hashing for real-time large scale near-duplicate video retrieval. In *ACM MM*, pages 423–432, 2011.

[25] C. Sun, I. Junejo, and H. Foroosh. Action recognition using rank-1 approximation of joint self-similarity volume. In *ICCV*, pages 1007–1012. IEEE, 2011.

[26] M. Vasilescu and D. Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. *ECCV*, pages 447–460, 2002.

[27] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *IJCV*, pages 1–20, 2013.

[28] X. Wu, A. G. Hauptmann, and C.-W. Ngo. Practical elimination of near-duplicates from web video search. In *ACM MM*, pages 218–227, 2007.

[29] A. Yao, J. Gall, and L. Van Gool. A hough transform-based voting framework for action recognition. In *CVPR*, pages 2061–2068. IEEE, 2010.

[30] G. Yu, N. A. Goussies, J. Yuan, and Z. Liu. Fast action detection via discriminative random forest voting and top-k subvolume search. *Multimedia, IEEE Transactions on*, 13(3):507–517, 2011.

[31] J. Yuan, Z. Liu, and Y. Wu. Discriminative subvolume search for efficient action detection. In *CVPR*, pages 2442–2449. IEEE, 2009.