Enhancing Single-Frame Supervision for Better Temporal Action Localization

Changjian Chen, Jiashu Chen, Weikai Yang, Haoze Wang, Johannes Knittel,
Xibin Zhao, Steffen Koch, Thomas Ertl, and Shixia Liu

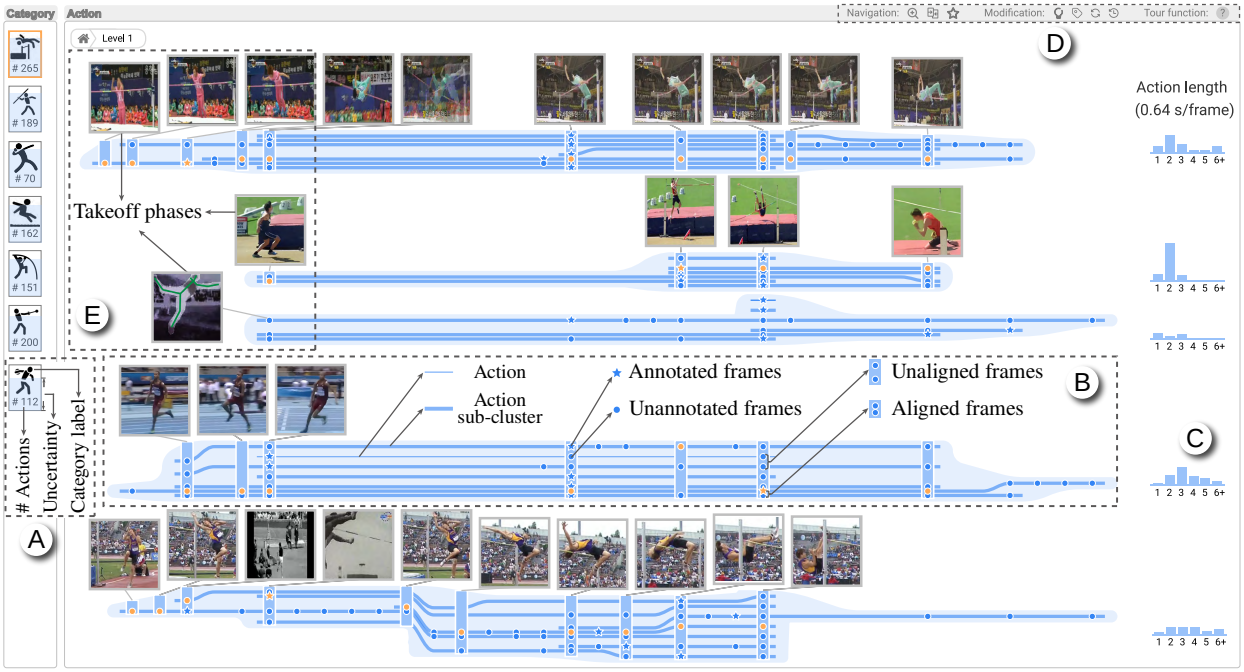

Fig. 1: ActLocalizer: (a) a list to show action categories and their uncertainty; (b) a storyline to show actions and their alignments.

Abstract—*Temporal action localization* aims to identify the boundaries and categories of actions in videos, such as scoring a goal in a football match. Single-frame supervision has emerged as a labor-efficient way to train action localizers as it requires only one annotated frame per action. However, it often suffers from poor performance due to the lack of precise boundary annotations. To address this issue, we propose a visual analysis method that aligns similar actions and then propagates a few user-provided annotations (*e.g.*, boundaries, category labels) to similar actions via the generated alignments. Our method models the alignment between actions as a heaviest path problem and the annotation propagation as a quadratic optimization problem. As the automatically generated alignments may not accurately match the associated actions and could produce inaccurate localization results, we develop a storyline visualization to explain the localization results of actions and their alignments. This visualization facilitates users in correcting wrong localization results and misalignments. The corrections are then used to improve the localization results of other actions. The effectiveness of our method in improving localization performance is demonstrated through quantitative evaluation and a case study.

Index Terms—Temporal action localization, single-frame supervision, storyline visualization

1 Introduction

Temporal actions refer to human motions or human-object interactions that occur in a video, such as scoring a goal in a football match and throwing a javelin [45, 56]. Detecting and analyzing temporal actions is important for a variety of applications, ranging from security surveillance to video moderation and home care [37, 58, 65]. As a result, *temporal action localization* has emerged as a significant research topic in computer vision [56]. It determines the boundaries and categories of actions in videos. In recent years, deep learning models [33, 63] have significantly improved the performance of temporal action localization by exploiting a large number of fully-annotated videos. However, it is time-consuming to obtain these annotations since annotators need to seek back and forth through the videos to identify the precise boundaries of actions. To address this issue, single-frame-oriented temporal action localization methods have been developed, which utilize *single-frame annotations* to train action localizers [30, 37] (Fig. 2(a)). A single-frame annotation of an action consists of the temporal location of one single frame (anchor frame) and its category label (Fig. 2A). Annotators can provide such annotations by watching the video once with some extra

- *C. Chen is with the College of Computer Science and Electronic Engineering, Hunan University. E-mail: changjianchen@hnu.edu.cn.*
- *J. Chen, W. Yang, H. Wang, X. Zhao, and S. Liu are with the School of Software, BNRist, Tsinghua University. S. Liu is the corresponding author. E-mail: {{cjs22, yangwk21, wang-hz22}@mails., zxb, shixia@}tsinghua.edu.cn.*
- *J. Knittel, S. Koch, and T. Ertl are with University of Stuttgart. E-mail: johannes.knittel, steffen.koch, Thomas.Ertl@vis.uni-stuttgart.de.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

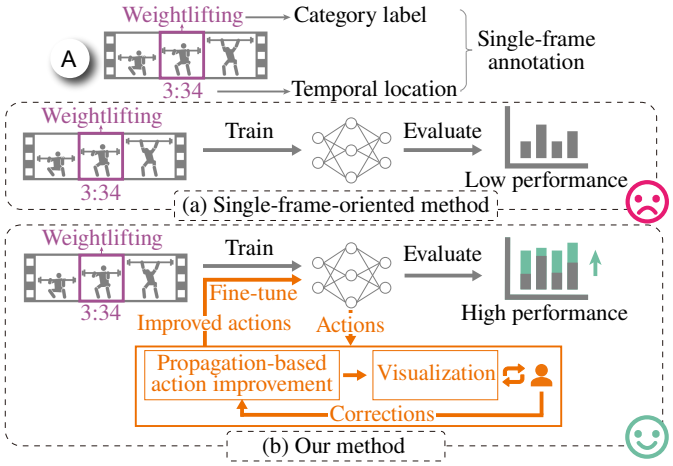

Fig. 2: The comparison between (a) the single-frame-oriented method and (b) the proposed visual analysis method (in orange).

pauses, which largely reduces annotation costs [37]. However, the performance of these methods is generally poor due to the lack of precise boundary annotations and the presence of noisy category labels.

An effective way to boost performance is to integrate humans into the automatic localization process. First, a few imprecise boundaries and noisy category labels of detected actions are corrected by users. Then, the corrections are propagated to similar actions that share common human motions or human-object interactions to improve the localization results. These results are used to fine-tune the localizer for better performance. This largely saves human efforts. However, in the context of action localization, two challenges still persist. First, as the dataset grows in size, manually identifying imprecise boundaries and noisy category labels becomes more challenging. Second, due to the need to consider both frame similarities and inherent temporal relationships between sequential video frames, propagating corrections across similar actions presents a complex task.

To address these challenges, we develop ActLocalizer, a visual analysis method to 1) help users explore the localization results of actions for correction; 2) propagate the corrections to similar actions through the generated alignments to improve localization performance. As shown in Fig. 2(b), ActLocalizer first trains an initial action localizer using the single-frame annotations to detect actions. Then a propagation-based action improvement method is developed to improve the actions. Specifically, similar actions are aligned together by considering both similarities and temporal relationships between frames, which is formulated as a heaviest path problem. Based on the generated alignments, the localization results are improved by propagating the single-frame annotations to similar actions. This is achieved by solving a quadratic optimization problem. As the automatically generated alignments may not accurately match the associated actions and could produce inaccurate localization results, we develop a storyline visualization to explain the actions and their alignments. Users can interactively correct misalignments and wrong localization results. The corrected alignments are incorporated into the heaviest path problem to generate better alignments, and the corrected localization results are propagated through the updated alignments to obtain improved actions. Based on the improved actions, the action localizer is fine-tuned for better performance. The effectiveness of the propagation-based action improvement method and the storyline is demonstrated through quantitative evaluation and a case study. The source codes are available at: http://actlocalizer.thuvis.org/.

In summary, our contributions include:

- **A visual analysis tool** for iteratively improving the performance of action localizers through minimal user corrections on action annotations and alignments.
- **A storyline visualization** for exploring actions, examining alignments, and making necessary corrections.
- **A propagation-based action improvement method** that effectively propagates user corrections to improve action localization results while saving human efforts.

2 Related Work

2.1 Semi-Supervised Temporal Action Localization

Semi-supervised temporal action localization methods can be classified into two groups: boundary-oriented and single-frame-oriented methods.

Boundary-oriented methods utilize both annotated and unannotated videos to train action localizers [24, 53]. They introduce a consistency constraint between unannotated videos and their perturbed counterparts to improve the robustness against perturbations. For example, Ji *et al.* [24] kept the localization results unchanged for unannotated videos when the frames were resampled or some frames were masked. Recently, single-frame-oriented methods have been proposed to reduce annotation costs [30, 37, 58]. Compared to boundary-oriented methods, they achieve better performance with the same number of annotated frames. Compared to boundary-oriented methods, they perform better when utilizing the same number of annotated frames. Their training consists of two phrases. First, an action localizer trained on the single-frame annotations is utilized to detect actions. Second, these actions, which may be imprecise, are used to fine-tune the action localizer. Assuming no irrelevant background frames in videos, Li *et al.* [30] identified boundaries by detecting action changes between consecutive anchor frames. Since the assumption is not always true, this method may mispredict a background frame as an action frame. To address this issue, Ma *et al.* [37] developed SF-Net, which determined boundaries by extending anchor frames to their temporally adjacent frames with high prediction confidence.

Despite the capabilities of single-frame-oriented methods, they can benefit from extra boundary annotations. Thus, we introduce a propagation-based method that improves the localization results of any single-frame-oriented method using these extra boundary annotations.

2.2 Interactive Annotation for Sequence Data

Interactive annotation methods for sequence data can be divided into two groups: direct verification and iterative verification.

Direct verification methods utilize automatic algorithms to generate annotations [11, 26, 43]. Users only need to validate or correct the generated annotations, greatly saving their efforts. For example, Kurzhals *et al.* [26] utilized a video segmentation algorithm to partition eye-tracking data into multiple segments and clustered them. Users can annotate multiple segments of a cluster in one go. However, the accuracy of the automatically generated annotations significantly affects the annotation efficiency. To address this issue, iterative verification methods have been proposed [19, 28, 51, 60]. In these methods, users annotated sequences that were recommended by an active learning model or identified in the visualization. The annotations were then used to fine-tune the model for better performance. This paradigm was also employed by Lekschas *et al.* [28] and Yu *et al.* [60] for identifying sequence data of interest and Tang *et al.* [51] for annotating abnormal videos.

Among these methods, the most relevant one is VideoModerator [51]. It aims to identify the videos with misinformation or offensive content. Initially, a classifier was trained to recommend such videos, The recommended videos were then analyzed and annotated in three coordinated views. These annotations were used to fine-tune the classifier. Since the classifier in VideoModerator cannot be utilized to detect the actions in videos, we have developed ActLocalizer, which effectively detects actions by combining single-frame annotations and a few user corrections (*e.g.*, corrected boundaries). Additionally, ActLocalizer utilized a storyline visualization to aid users in analyzing and correcting actions, ultimately enhancing localization performance.

2.3 Video Visualization

Initial efforts on video visualization focus on visually illustrating the main content of a video by summarizing the representative frames [13, 25, 41, 46, 49] or extracted attributes, including the motions of objects [10, 14] and their trajectories [3, 21, 38]. Recent efforts seek to combine machine learning techniques with interactive visualization for performing video analysis tasks, including exploring the engagement of students in class [61], understanding inherent structures of movies [27, 36], and analyzing presentation techniques in TED talks [55, 62].

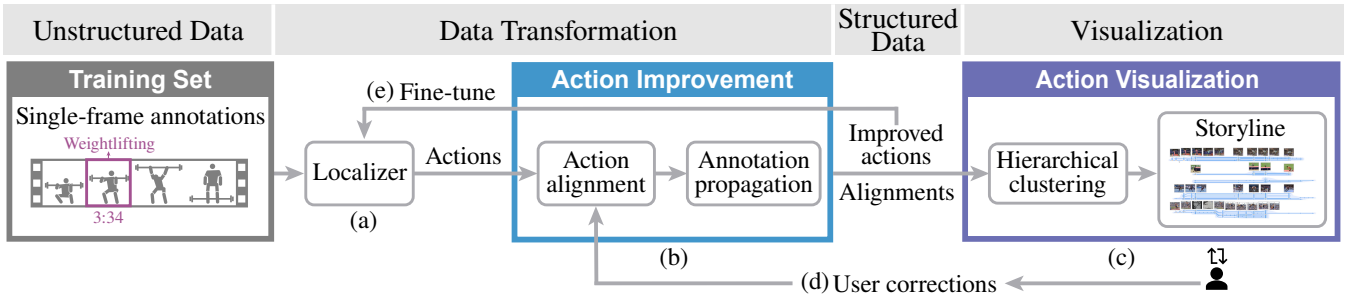

Fig. 3: System overview: given videos and their single-frame annotations, (a) an action localizer is trained; (b) the action improvement module aligns similar actions together for propagating annotations; (c) the visualization module explains the actions and their alignments; (d) users correct misalignments and wrong localization results; (e) the corrections are utilized to fine-tune the localizer.

Orthogonal to these methods, our work focuses on improving the performance of action localizers. For this purpose, we developed a storyline visualization that enables users to easily correct wrong localization results. These corrections are then propagated by the propagation-based action improvement method to boost model performance.

3 Requirement Analysis

We worked closely with four machine learning experts (E_1–E_4) to develop ActLocalizer. None of them are the co-authors of this paper. E_1 is a postdoctoral researcher, and E_2 is a Ph.D. student. They have studied single-frame-oriented temporal action localization methods for over four years. During their research studies, they found that the quality of the annotations limited the performance of the models. Therefore, they wanted to interactively correct a few wrong localization results and propagate them to similar actions for better performance. E_3 and E_4 are two Ph.D. students who applied several single-frame-oriented methods to detect abnormal actions in surveillance videos for a project. These methods did not perform as expected, so they wanted to correct the localization results of some actions to enhance performance.

We distilled the following requirements from four 40-70 minute semi-structured interviews with the experts and literature reviews.

R1 - Explore localization results and identify the wrong ones. In practice, the experts had to examine each action individually to pinpoint the wrong localization results, including imprecise boundaries and noisy category labels. This becomes tedious when dealing with a large number of actions. To make the examination more labor-efficient, they desired an overview of the actions first. Then, they wanted to examine the actions with imprecise boundaries and noisy labels at different levels of detail. E_2 mentioned, "I prefer grouping similar actions together, each accompanied by a few representative frames to provide an overview. The grouping results and the associated representative frames allow me to quickly identify the actions with wrong localization results that require further analysis."

R2 - Correct wrong localization results more efficiently. Upon identifying wrong localization results, the experts required to correct them for better performance. While the experts can correct noisy category labels quickly, adjusting imprecise boundaries can be quite tedious. For example, E_1 said, "When refining the imprecise boundaries for one action, I find myself frequently rewinding the video to locate the precise boundaries, especially when they are unclear or ambiguous." A more labor-efficient way is to recommend several boundary candidates for validation or refinement. Additionally, E_1 suggested displaying boundary candidates in the context of neighboring frames. This would reduce the need for repeated video playback and thus save time and efforts.

R3 - Propagate the the corrected localization results to similar actions. All the experts expressed the need to minimize the number of corrected localization results. E_3 commented, "An labor-efficient way to achieve this is to propagate a few user-corrected localization results to similar actions." He noted that the propagation results depended on the alignments between actions. Existing alignment methods only consider the similarities between the frames [8, 31] and ignore the temporal relationships between successive frames. This frequently leads to numerous misalignments, such as aligning the upward and downward phases of two pull-up actions. Thus, it is desired to align similar actions by considering both similarities and temporal relationships between frames.

R4 - Correct the action alignments for better propagation. Since the alignments are generated automatically, some may not accurately match the associated actions [18]. These misalignments result in wrong propagation and subsequently degrade model performance. Therefore, the experts wanted to examine the alignments between actions and understand how the corrected localization results are propagated through the alignments. With a comprehensive understanding, they aimed to identify and correct a few misalignments. The remaining misalignments are expected to be corrected automatically for saving efforts. For example, E_2 expressed the need for a tool to inspect action alignments, identify misalignments, and correct them for more effective propagation.

4 Design of ActLocalizer

Guided by the requirements, we have developed ActLocalizer to iteratively improve the performance of action localizers. As shown in Fig. 3, the analysis process begins with an input set of videos and their single-frame annotations. Given the unstructured nature of this input, we first transform it into structured actions and alignments that are suitable for visualization. The visualization then facilitates user feedback, which in turn refines the actions and alignments for further analysis. Specifically, in the data transformation phase, an initial action localizer is trained on the input set for detecting the actions (Fig. 3(a)). The **action improvement module** (Fig. 3(b)) first aligns similar actions together. Then it propagates the single-frame annotations to the similar actions (**R3**). When transitioning to the visualization phase, the **action visualization module** organizes the actions hierarchically and explains the actions and their alignments with a storyline (**R1**, Fig. 3(c)). During exploration, users can correct wrong localization results of actions (**R2**) and misalignments between them (**R4**). The corrected alignments are utilized to improve other alignments, and the corrected localization results are propagated through the updated alignments to obtain improved actions (Fig. 3(d)). Based on the improved actions, the action localizer is fine-tuned for better performance (Fig. 3(e)).

4.1 Propagation-based Action Improvement

The propagation-based action improvement includes two main components: action alignment and annotation propagation.

4.1.1 Action Alignment

Given two actions P and Q, the action alignment establishes the temporal correspondence between their frames (Fig. 4). A straightforward solution is to use dynamic time warping [39]. However, it requires the alignments of each frame in one action with at least one frame in another action. In practice, detected actions usually include background frames, which should not be aligned with any action frames. To address this issue, we adopted the method proposed by Tan *et al.* [47], which enables background frames to remain unaligned. This method aligns actions by maximizing the total similarities of the aligned pairs:

$$\max_{z} \sum_{i=1}^{|P|} \sum_{j=1}^{|Q|} z_{ij} s_{ij} \qquad \text{s.t. } z_{ij} \in \{0,\ 1\}. \tag{1}$$

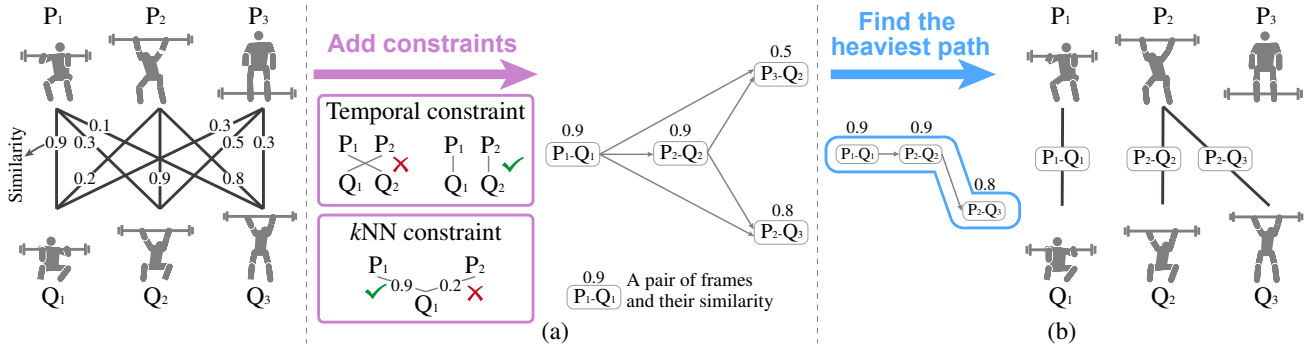

Fig. 4: Action alignment: (a) adding constraints to construct a graph; (b) finding the heaviest path to obtain the alignments.

Here, z_{ij} is a binary variable indicating whether P_i and Q_j are aligned. P_i and Q_j are the i-th and j-th frames of P and Q, respectively. s_{ij} is their cosine similarity, which is a common measure of frame similarity [4]. $|P|$ and $|Q|$ are the numbers of frames in P and Q, respectively.

Since temporal relationships and frame similarities are the most important factors for aligning actions [48], we introduce the temporal and kNN constraints into Eq. (1). In addition, to help correct misalignments more effectively, the must-link/cannot-link constraints are also considered because they are easy for users to provide. Here we only consider the most common and domain-agnostic constraints. Other constraints, such as the interval constraint, can be easily integrated into our method.

- *Temporal constraint.* As the frames in an action are monotonic in time, the aligned frame pairs should maintain such monotonicity (*i.e.*, $\forall(z_{ij}=z_{rt}=1)\wedge(i<r)\Rightarrow j\leq t$).
- *kNN constraint.* The aligned frame pairs should have similar content as they represent the same phases of the corresponding actions. Consequently, each frame within one action is constrained to align with its k nearest frames within another action. We choose kNN due to its simplicity and robustness [7,66]. For each frame, k is adaptively determined by the state-of-the-art method proposed by Zhao *et al.* [64]. k of each frame is adaptively determined by the state-of-the-art method proposed by Zhao *et al.* [64]. It finds the smallest k that yields a sufficiently large averaged prediction confidence for the k nearest frames.
- *Must-link/cannot-link constraints.* The must-link and cannot-link constraints are provided by users through interactions. They constrain which frame pairs should be aligned and which should not.

All these constraints form a directed acyclic graph (Fig. 4(a)). In the graph, each node corresponds to a frame pair satisfying the kNN constraint, with its weight encoding the pair similarity. The edges between the nodes describe the temporal constraint. For example, in Fig. 4(a), the edge from "P_1-Q_1" to "P_2-Q_2" is added because P_1 occurs before P_2 and Q_1 occurs before Q_2. Each path of the graph represents a possible alignment result. The must-link/cannot-link constraints specify the nodes to be included or excluded in the final path. Given this graph, obtaining the optimal alignments (Eq. (1)) while satisfying the constraints is equivalent to finding a path that maximizes node weights. This corresponds to a network heaviest path problem (Fig. 4(b)) and can be solved by dynamic programming.

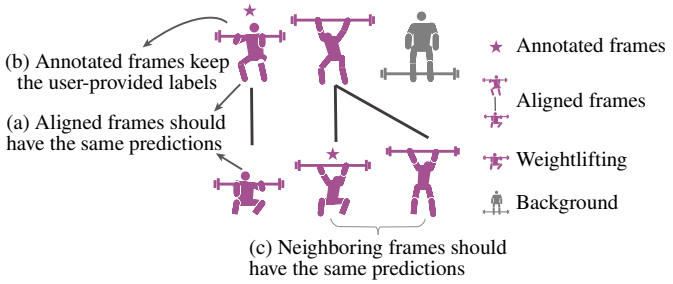

Fig. 5: Annotation propagation: (a) aligned frames should have the same predictions; (b) annotated frames keep the user-provided labels; (c) neighboring frames should have the same predictions.

4.1.2 Annotation Propagation

Annotation propagation improves localization results by propagating single-frame annotations and user corrections through the generated alignments (Fig. 5). This propagation ensures that the aligned frames should have the same predictions (Fig. 5(a)) and the prediction of the annotated frames are consistent with their category labels (Fig. 5(b)). If one boundary of an action is corrected by the user, the frames between its anchor frame and this boundary are also regarded as the annotated frames. In addition, due to the temporal relationships among frames, each unannotated frame should have the same predictions as its temporally closest annotated frame (Fig. 5(c)). Accordingly, annotation propagation is formulated as a quadratic optimization problem:

$$\min_{\mathbf{F}} \sum_{i,j}^{T} z_{ij}\|\mathbf{F}_i-\mathbf{F}_j\|^2+\alpha\sum_{i}^{T}\delta_i\|\mathbf{F}_i-\mathbf{G}_i\|^2+\beta\sum_{i}^{T}\min(\|\mathbf{F}_i-\mathbf{F}_i^c\|^2,\tau). \quad (2)$$

The first term minimizes the prediction difference between the aligned frames. The second term ensures the consistency between the predictions and the category labels for the annotated frames. The third term minimizes the prediction difference between unannotated frames and their temporally closest annotated frames. The weights α and β balance the three terms.

Following the work of Iscen *et al.* [23], the **first term** utilizes the L_2 loss to measure the prediction difference between the aligned frames. T is the total number of frames. $\mathbf{F}_i$ represents the prediction of the i-th frame across different categories. $z_{ij}=1$ indicates the i-th and j-th frames are aligned, and 0 otherwise. The **second term** also uses the L_2 loss to measure the difference between the predictions of the annotated frames and their category labels. $\mathbf{G}_i$ is the category label of the i-th frame. $\delta_i=1$ indicates the i-th frame is annotated, and 0 otherwise. The **third term** utilizes the truncated mean squared error function [16] to measure the prediction consistency between each unannotated frame i and its temporally closest annotated frame $\mathbf{F}_i^c$. The i-th frame is likely to be a background frame if the prediction difference between $\mathbf{F}_i$ and $\mathbf{F}_i^c$ is larger than a threshold τ. Limiting the mean squared error to τ prevents the background frames from being mispredicted as action frames. α, β, and τ are determined by a grid search to balance the magnitude difference among the three terms. The optimization problem Eq. (2) can be solved by the gradient descent method [16].

4.2 Action Visualization

To help users explore and correct a large number of actions and their alignments, we cluster the actions into a hierarchy. Based on the hierarchy, we developed a storyline visualization to explain the actions and their alignments. Several interactions are also provided to help users correct misalignments and wrong localization results.

4.2.1 Hierarchical Action Clustering

In ActLocalizer, an action hierarchy is built by a divisive method, which is one of the most widely used hierarchical clustering methods and fast in computation [44]. The divisive method repeatedly applies a flat clustering algorithm to build the hierarchy in a top-down manner.

We choose K-medoids [42] as the flat clustering algorithm due to its simplicity and robustness to noise [2, 35]. Following the work of Wang *et al.* [54], the action similarities used for clustering are measured by the averaged similarities of aligned frame pairs between two actions. As there is no gold standard for determining the number of clusters [40,52,57], we employ the average silhouette width to evaluate the cluster results because it considers both the cluster compactness and separability, and use a grid search for the best one. Other methods for determining the number of clusters can also be used in our hierarchical action clustering method.

To provide an overview of the action hierarchy, a set of **representative frames** ($\mathcal{U}$) is selected from each displayed cluster [6, 9]. This selection aims to retain the representation quality while minimizing the number of the selected frames:

$$\min_{\mathcal{U}} \sum_{j=1}^{T_c} \min_{i \in \mathcal{U}} d_{ij} + \gamma |\mathcal{U}|. \tag{3}$$

The first term ensures better representativeness by minimizing the sum of the minimum dissimilarities between the selected frames and each unselected one. The second term favors the selection of a small subset of frames. $d_{ij} = 1 - s_{ij}$, where s_{ij} is the cosine similarity between two frames. T_c is the total number of frames in an action cluster. $|\mathcal{U}|$ is the number of the selected frames, and γ is the weight to control the number of the selected frames. According to the study of Elhamifar *et al.* [15], it can be set as $\max_{ij} d_{ij}/\mathrm{M}$ to select around M frames. In our implementation, M is set as 10 due to the space limit. As optimizing Eq. (3) is NP-hard, we employ a state-of-the-art approximate algorithm, the Alternating Direction Method of Multipliers [15,59], to solve it. This method decomposes the optimization problem into a set of simpler sub-problems and optimizes each sub-problem alternatively to obtain the final result.

4.2.2 Visual Design

Previous studies have demonstrated the effectiveness of the storyline metaphor in conveying sequential data and their temporal interconnections [32,50,61]. Therefore, we adopt this metaphor to illustrate actions and their alignments. The visual design consists of two parts: 1) a category list to represent the action categories (Fig. 1(a)); 2) a storyline to illustrate the actions and their alignments (action view, Fig. 1(b)).

In the category list, users examine the overview of the detected actions across different categories and their uncertainty. Each category is represented by a rectangle (Fig. 1A). The simple drawing (*e.g.*, 🏃) denotes the category label, and the number of actions in that category is shown below the drawing. A solid filling style is utilized to encode the uncertainty of the category, which is inversely proportional to the average confidence of all the frames in this category. The greater the filling height, the increased uncertainty. These categories are placed in descending order of uncertainty from top to bottom.

In the action view, the storyline enables users to explore actions at different levels of detail and identify the actions of interest. As shown in Fig. 1B, each contour represents an action cluster, where each thin line (—) represents an action, and each thick line (▬) represents an action sub-cluster. The circles (●) on each line represent the unannotated frames of the action, and the stars (★) represent the annotated frames. The horizontal positions of the frames represent their sequential orders. The vertical distance between the circles/stars at the same horizontal positions indicates whether the associated frames are aligned. The circles/stars are close if their associated frames are aligned; otherwise, they are unaligned. The representative frames of a cluster are displayed above its contour with the associated circles/stars highlighted in orange. The bar chart on the right side of each cluster shows the distribution of the action lengths (Fig. 1C).

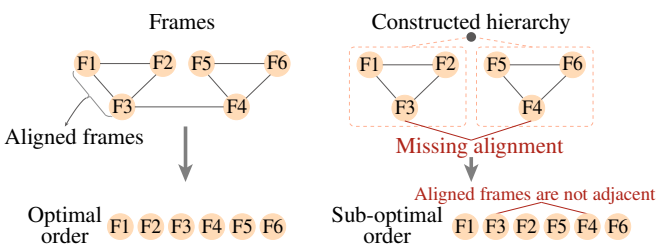

Fig. 6: Some alignments are left out by the constructed hierarchy.

4.2.3 Layout

As the implementation of the category list is easy, here we focus on introducing how to generate the storyline. The state-of-the-art solution, StoryFlow [32], generates legible storylines by placing entities in the same group adjacently while reducing line crossings, line wiggles, and white space. Similar to StoryFlow, the aligned frames are expected to be placed adjacently while satisfying these legibility constraints. However, utilizing StoryFlow directly presents two key issues. First, StoryFlow assumes that each entity has a synchronized timestamp and places the entities with the same timestamp at the same horizontal position. As actions are usually captured at different times, their frames do not have such synchronized timestamps. Second, StoryFlow assumes the existence of a location hierarchy for each timestamp. Based on the hierarchies, StoryFlow reduces line crossings and wiggles while ensuring that entities in the same sub-hierarchy are placed adjacently along the vertical direction. To apply StoryFlow to generate our storyline, we can build hierarchies for each timestamp based on the alignments between frames. However, as some aligned frames may belong to different sub-hierarchies, such hierarchies may overlook these alignments and result in sub-optimal orders. For example, the aligned pair "F3-F4" in Fig. 6 is overlooked by the constructed hierarchy as they belong to two different sub-hierarchies. This subsequently leads to a sub-optimal order where "F3" and "F4" are not placed adjacently.

To address these issues, we develop a layout method that consists of a horizontal placement and a vertical placement (Fig. 7). The horizontal placement aims to place aligned frames with higher similarities at the same horizontal positions (Fig. 7(a)), and the vertical placement seeks to reduce line crossings, line wiggles, and white space (Fig. 7(b)).

Horizontal placement. Mathematically, the horizontal placement is expressed as:

$$\begin{aligned} \max_{x} \quad & \sum_{\mathrm{P},\mathrm{Q} \in \mathcal{A}} \sum_{x(\mathrm{P}_i)=x(\mathrm{Q}_j)} (z_{ij} + \lambda \cdot \mathrm{s}_{ij}) \\ \text{s.t.} \quad & x(\mathrm{P}_i) < x(\mathrm{P}_j) \quad \text{if} \quad i < j, \ \forall\, \mathrm{P} \in \mathcal{A}. \end{aligned} \tag{4}$$

The first term ensures that aligned frames are placed at the same horizontal positions. The second term encourages similar frames to be placed at the same horizontal positions. The constraint guarantees that the frames' horizontal positions are consistent with their sequential orders. $\mathcal{A}$ is a set of actions, $x(\cdot)$ is the horizontal position of a frame, z_{ij} is a binary variable indicating whether frames P_i and Q_j are aligned ($z_{ij} = 1$) or not ($z_{ij} = 0$), s_{ij} is their cosine similarity, and λ is the weight to balance the two terms. Since the primary goal of this step is to place aligned frames at the same positions, the weight of the first term (alignment) should be larger than that of the second term (similarity). Therefore, we set λ as 0.1 in our implementation. Following the work of Feng *et al.* [17], this optimization problem is solved by a greedy strategy. This strategy iteratively determines the horizontal position of the frames in one action at a time, while freezing the frames of previously placed actions.

Vertical placement. The vertical placement consists of three steps: ordering, straightening, and compaction (Fig. 7(b)). The compaction algorithm of StoryFlow is employed to reduce unnecessary white space, so we focus on introducing the first two steps.

Ordering. This step ensures that the aligned frames are placed adjacently along the vertical direction while minimizing line crossings.

$$\min_{\phi} \sum_{\mathrm{P},\mathrm{Q} \in \mathcal{A}} \sum_{x(\mathrm{P}_i)=x(\mathrm{Q}_j)} z_{ij} \cdot \mathbb{I}(|\phi(\mathrm{P}_i) - \phi(\mathrm{Q}_j)| > 1) + \mu C(\phi) \tag{5}$$

The first term ensures that aligned frames are placed in adjacent vertical positions, while the second term penalizes line crossings. Here, $\phi(\cdot)$ denotes the vertical order of a frame among all frames at the same horizontal position and $\mathrm{C}(\phi)$ is the number of line crossings. μ is the weight to balance the two terms, which is set as 3 in our implementation

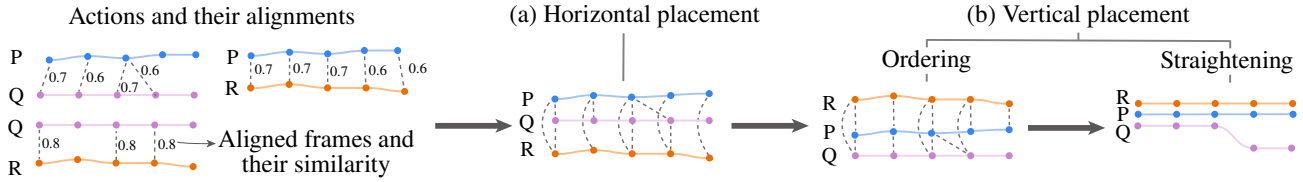

Fig. 7: The storyline layout: (a) horizontal placement places aligned frames with higher similarities at the same horizontal positions; (b) vertical placement places aligned frames adjacently while minimizing line crossings (ordering) and the wiggle number (straightening).

to prioritize the reduction of line crossings. The global optimum of Eq. (5) is obtained using state compression dynamic programming [20].

Straightening. The main goal of straightening is to minimize the number of line wiggles while also preserving the vertical distances between frames based on their alignments. This is formulated as a constrained optimization problem:

$$\begin{aligned} \min_{y} \quad & \sum_{\mathrm{P}\in\mathcal{A}} \sum_{i} \mathbb{I}[y(\mathrm{P}_i) \neq y(\mathrm{P}_{i+1})] \\ & + \mu \sum_{\mathrm{P},\mathrm{Q}\in\mathcal{A}} \sum_{\substack{x(\mathrm{P}_i)=x(\mathrm{Q}_j),\\ |\phi(\mathrm{P}_i)-\phi(\mathrm{Q}_j)|=1}} [|y(\mathrm{P}_i)-y(\mathrm{Q}_j)|-(1-z_{ij})]^2 \qquad (6) \\ \text{s.t.} \quad & y(\mathrm{P}_i) < y(\mathrm{Q}_j) \quad \text{if} \quad \phi(\mathrm{P}_i) < \phi(\mathrm{Q}_j). \end{aligned}$$

The first term penalizes the line wiggles, while the second term guarantees vertical closeness between aligned frames and maintains a specific vertical distance between unaligned frames. $y(\cdot)$ is the vertical position of a frame. The weight μ is used to balance the magnitude difference between the two terms and is determined by a grid search. Eq. (6) can be solved by dynamic programming.

4.2.4 Interactive Exploration and Correction

To help users easily correct misalignments and wrong localization results, ActLocalizer provides two types of interactions: action filtering and interactive correction.

Action filtering. When users identify an action of interest, they can filter this action and its neighbors to better analyze their alignments (Fig. 8). The selected action and its neighbors are placed on the top (Fig. 8(a)). The frames (Fig. 8(b)) and the associated video clip (Fig. 8(c)) of the selected action are also provided to facilitate the examination and analysis in context.

Interactive correction. ActLocalizer allows users to enhance the performance of action localizers by interactively correcting misalignments and wrong localization results.

Correcting misalignments. When users identify misalignments, they can correct them directly in the visualization. Users can drag two frames closer to indicate that they must be aligned and farther apart to indicate that they cannot be aligned. The corrections are converted to the must-link and cannot-link constraints in the alignment algorithm, which are then utilized to update the other alignments between actions.

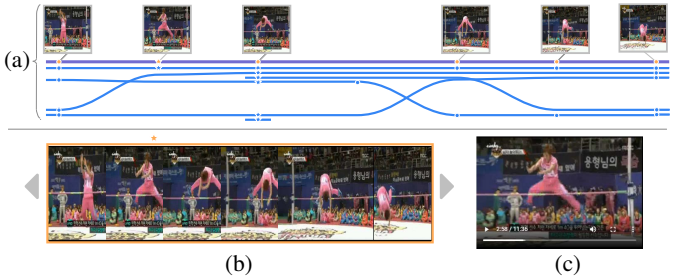

Fig. 8: Selecting an action of interest: (a) the storyline to show the alignments between the selected action and its neighbors; (b) the content of the associated frames; (c) the associated video clip.

Table 1: Performance comparison between our method and SF-Net in terms of the mAP (in %) on two benchmark datasets.

Annotations	2%	5%	10%	20%
SF-Net	**42.70**	**42.91**	**43.08**	**43.49**
+ our method	(**+1.10**)	(**+1.18**)	(**+1.04**)	(**+0.74**)
SF-Net	41.60	41.73	42.04	42.75

(a) **THUMOS14**

Annotations	2%	5%	10%	20%
SF-Net	**32.51**	**32.79**	**33.21**	**33.36**
+ our method	(**+1.03**)	(**+1.18**)	(**+1.25**)	(**+1.03**)
SF-Net	31.48	31.61	31.96	32.33

(b) **BEOID**

Correcting wrong localization results. If users identify wrong localization results, including imprecise boundaries and noisy category labels, they can right-click the frames to annotate the boundaries or click in Fig. 1D to change the category labels. The corrected localization results are then propagated to other similar actions. To reduce user efforts in this correction process, ActLocalizer allows them to annotate a rough boundary and recommends a more precise one based on the rough one. The recommended boundary is obtained by propagating the annotated rough boundary using the propagation method described in Sec. 4.1.2. If the recommended boundary is still imprecise, users can annotate a new one based on the recommendation and iteratively refine it.

5 Evaluation

We conducted quantitative evaluation to evaluate the performance of the propagation-based action improvement method and a case study to demonstrate the effectiveness of ActLocalizer.

5.1 Quantitative Evaluation on Action Localization

Datasets. In this experiment, two widely used datasets are employed. The first dataset, **THUMOS14** [22], contains 200 training videos (63,575 frames) and 213 test videos (70,044 frames) with 20 sports categories. We noticed that some cliff diving actions were annotated as diving. To mitigate this inconsistency, we merged these two categories into a single category, namely "diving." The second dataset, **BEOID** [12], contains 58 videos (6,588 frames) with 30 action categories. They are recorded in six different scenes (*e.g.*, kitchens and gyms). The videos are randomly split into an 80% training set and a 20% test set. Each action in the training sets of both datasets has a single-frame annotation [37].

Experimental settings. Theoretically, our propagation-based action improvement method can enhance the performance of any single-frame-oriented localization method. In this experiment, we selected the state-of-the-art one, SF-Net, as a representative example. The effectiveness of our method is demonstrated by comparing the performance of SF-Net with and without our propagation-based action improvement method. To conduct the experiments while saving user annotation time and efforts, we simulated user-provided boundary annotations by randomly sampling v% ($v \in \{2,\ 5,\ 10,\ 20\}$) of actions and using the ground truth boundaries as the annotations. For SF-Net without our method, the RGB features extracted by Swin Transformer [34] and optical

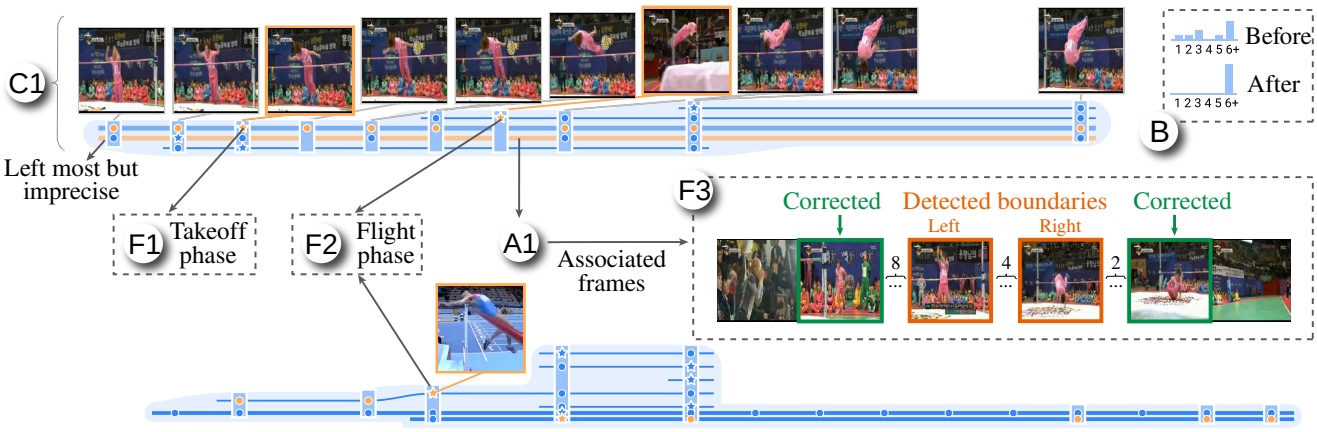

Fig. 9: When analyzing the high jumping category, C1 with more frames on the left side and imprecise boundaries is identified.

flow features extracted by I3D network [5] (a pretrained model) are utilized for training. Based on the extracted features, SF-Net uses both single-frame and user-provided boundary annotations to train an action localizer, which is utilized to detect actions. Then, the detected actions are utilized to fine-tune the action localizer. For SF-Net with our method, the proposed propagation-based action improvement method is added to improve the detected actions by propagating the user-provided boundary annotations to similar actions.

Result. Performance was evaluated by mAP@IoU and mAP@Hit [37], the commonly used measures for temporal action localization. We found that using the mAP@IoU and mAP@Hit scores led to similar conclusions. Therefore, we included only the mAP@IoU scores in the paper and abbreviate them as mAP scores, while the mAP@Hit scores can be accessed in the supplementary material. As shown in Table 1, our method improved the localization performance on the two datasets.

5.2 Case Study

We conducted a case study with E_1 and E_2, the experts consulted for the requirement analysis, to demonstrate the effectiveness of ActLocalizer in improving the performance of action localizers trained on single-frame annotations. The case study was conducted on a subset of **THUMOS14**, consisting of 1,135 actions in seven Track and Field sports action categories. An initial action localizer was trained on the single-frame annotations with SF-Net, which achieved a mAP of **47.47%**. The experts were not satisfied with the performance and wanted to improve it with ActLocalizer. As the experts were not involved in the design phase, we introduced the visual design and interactions of ActLocalizer to them before the case study. It lasted around 15 minutes. During the case study, E_1 focused on improving the performance of the three categories of jumping sports (high jumping, long jumping, and pole jumping), and E_2 focused on the four categories of throwing sports (javelin throwing, shot put, hammer throwing, and discus throwing). Here, we take high jumping and javelin throwing as two examples to illustrate the basic idea. In the case study, we employed the pair analytics protocol [1], which allows experts to concentrate on analytical tasks while we navigate the tool.

5.2.1 High Jumping Action Localization

E_1 started his analysis by examining the uncertainty of the seven action categories (Fig. 1(a)). Among them, the high jumping category (🤸) was at the top in the category list, which indicated it had the highest uncertainty. To investigate the cause of such high uncertainty, E_1 selected this category and switched to the action view.

Identifying wrong localization results (**R1**). In the action view, the high jumping actions were clustered into five groups (Fig. 1(b)). Typically, a high jumping action starts from an approach phase (🏃), followed by a takeoff phase (🤸) and a flight phase (🤸). By checking the representative frames, E_1 noticed that the starting points of the actions in the top three clusters were not the approach phases but the takeoff phases (Fig. 1E). To investigate why the starting points of these clusters were imprecise, he decided to examine them one by one.

Correcting imprecise boundaries (**R3**, **R4**). E_1 first selected the top cluster and zoomed in for further analysis. It contains seven sub-clusters, two of which are shown in Fig. 9. He examined the annotated

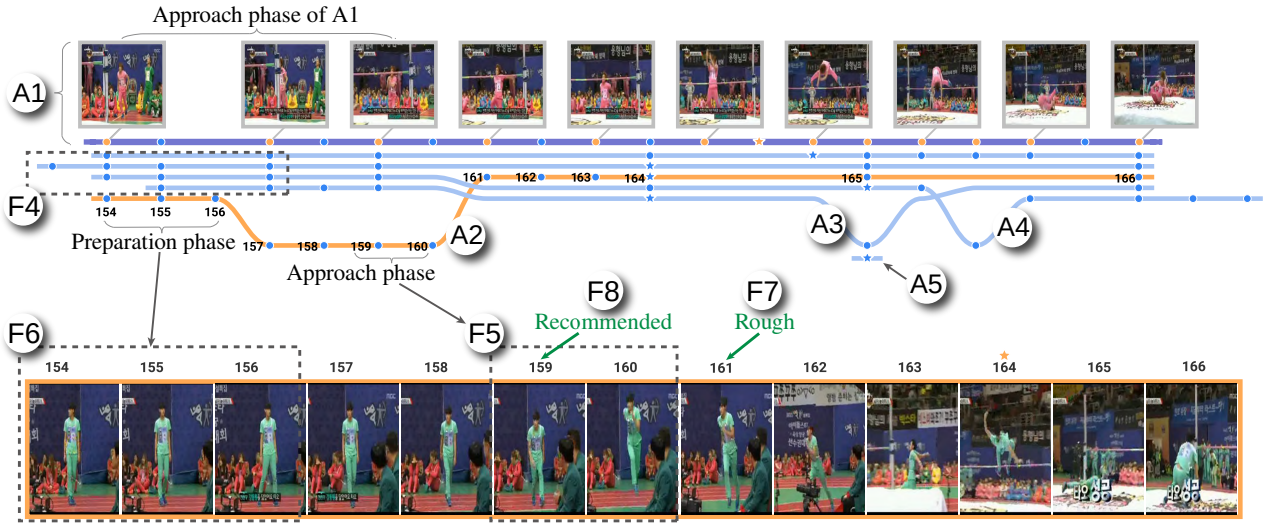

Fig. 10: After correcting the imprecise boundaries of A1 in the high jumping category, more frames of the approach phase are detected. Some unaligned frames are also identified in A1's neighbors (A2–A5).

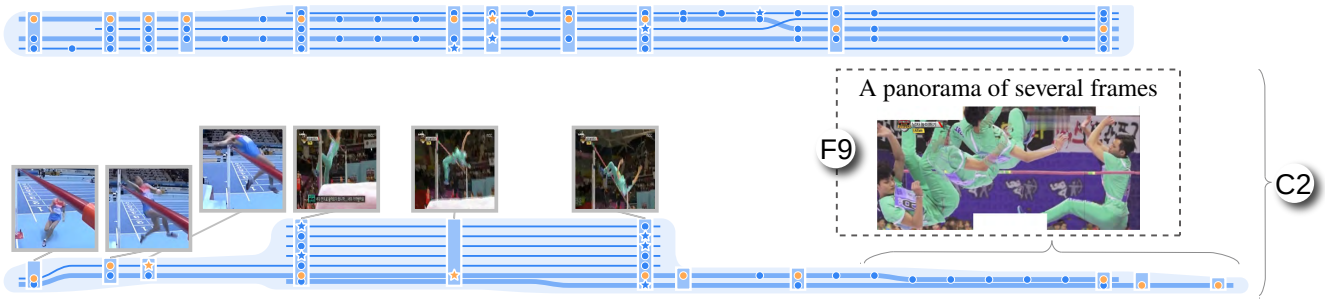

Fig. 11: Further analysis on the high jumping category identifies sub-cluster C2 with more frames on the right side.

frames and their locations in these actions and found that all of them were in the takeoff (Fig. 9F1) or flight (Fig. 9F2) phases. He concluded that the model could not learn to detect the approach phase without any annotated frames from this phase. Thus, E_1 decided to add some annotated frames of the approach phase to the training set. He first analyzed sub-cluster C1 where the left boundaries were the left most of these actions but still imprecise (Fig. 9C1). He randomly selected one action (Fig. 9A1) in this sub-cluster and checked its boundaries. This action was recorded in a TV show. Therefore, the man was not a professional jumper and was using an outdated forward jumping technique. E_1 found that both the left and right boundaries (frames with orange borders in Fig. 9F3) were imprecise. As E_1 quickly identified the boundaries (frames with green borders in Fig. 9F3), he corrected them directly.

The corrected boundaries of A1 were propagated, and the visualization was updated accordingly (Fig. 10). In A1's neighbors, more frames of the approach phase were detected (Fig. 10F4), and their boundaries were corrected. This demonstrated the effectiveness of the developed propagation-based action improvement method.

Correcting misalignments (**R2**, **R3**, **R4**). In the updated visualization, E_1 also observed that some frames of four A1's neighbors (A2–A5 in Fig. 10) were not aligned with A1. To investigate the cause, he decided to examine them one by one.

E_1 first examined the unaligned frames of A2 (Fig. 10F5). Some of these frames belonged to the approach phase but were not aligned with those of A1. Instead, A2's frames left to the approach phase were aligned (Fig. 10F6). These frames belonged to the preparation phase and looked very similar to the approach phase. Such similarity led to the misalignments. To correct the alignments, E_1 decided to align the precise left boundary of A2 with that of A1. However, E_1 would need a lot of time to identify the precise left boundary of A2 because the frames of the preparation and approach phases near the boundary looked similar. To accelerate the annotation process, E_1 used the boundary recommendation function. He first annotated a rough left boundary for this action (Fig. 10F7). Then ActLocalizer recommended a more precise boundary (Fig. 10F8) based on the roughly annotated one. E_1 confirmed that it was the precise boundary and dragged it to be aligned with the left boundary of A1. Similarly, E_1 checked A3, A4, and A5 and corrected their alignments with A1. These corrections were converted to the must-link constraints and utilized to update other alignments.

With the updated alignments, the corrected boundaries were propagated to the similar actions. A1 and its neighbors were well aligned. E_1 was satisfied with this and proceeded to check the actions in sub-cluster C1. According to the representative frames, the approach phases of these actions were correctly detected. This was also verified by the bar chart, which showed that most of the action lengths were longer than before (Fig. 9B). He was satisfied with the localization improvement in this sub-cluster and proceeded to examine the remaining sub-clusters.

Correcting noisy category labels (**R2**). While examining the remaining sub-clusters, E_1 discovered an interesting one, C2 (Fig. 11). Compared to others, the actions in C2 had more frames on the right side. To investigate this, E_1 checked the associated frames of one such action and found that it was a panorama of several frames in a high jumping action (Fig. 11F9). These frames could probably hurt the training process as the optical flow features were used to train the action localizer. These features represent human motions or human-object interactions in videos because they capture the relative motion between the cameras and the scenes [5]. However, for such a panorama, the optical flow features would capture the camera motion instead of the human motion. Thus, it should not be regarded as an action (a human motion or a human-object interaction). E_1 then labeled the associated frames as background frames.

Next, E_1 examined the remaining high jumping action clusters in Fig. 1(b) and corrected misalignments and wrong localization results. In total, five boundaries and 25 alignments of high jumping actions were corrected. These corrections were utilized by the propagation-based action improvement method to obtain improved actions, which were used to fine-tune the action localizer. The mAP was increased from **47.47%** to **47.99%**.

5.2.2 Javelin Throwing Action Localization

After the correction of the high jumping category, the uncertainty of the javelin throwing category (🏃) became the highest. Therefore, E_2 selected this category for further analysis.

Identifying wrong localization results (**R1**). In the action view

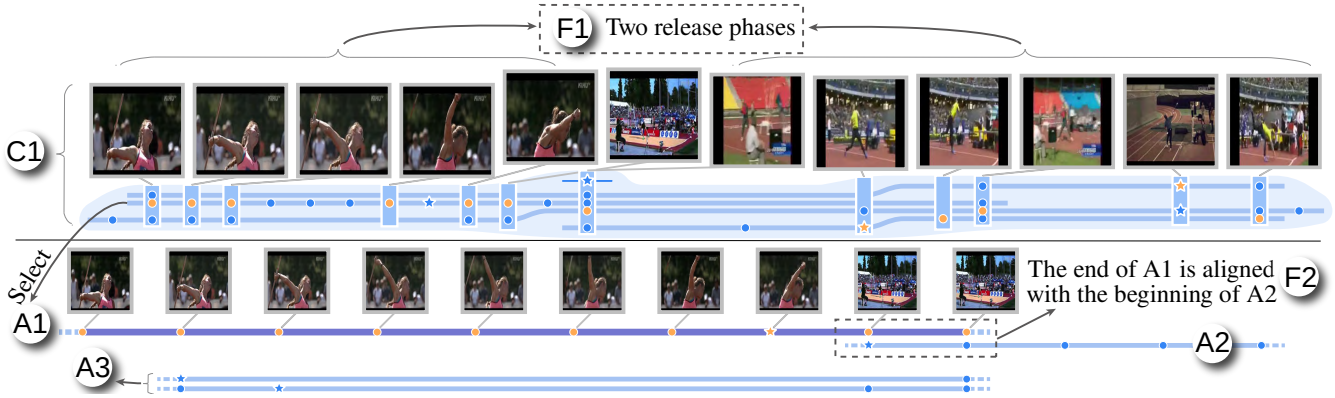

Fig. 12: The action view of the javelin throwing category: (a) cluster C1; (b) examination of A1 selected from cluster C1.

Table 2: The numbers of corrected boundaries and alignments, as well as annotation time comparison between ActLocalizer and the baseline.

Method	# boundaries	# alignments	time	mAP
Baseline	113	0	4.49 h	57.11%
ActLocalizer	34	211	0.99 h	58.29%

(Fig. 12), the representative frames of cluster C1 revealed the presence of two release phases (Fig. 12F1). It is abnormal since the actions in a cluster should be aligned without any repetitive phases. This phenomenon indicated that some actions in this cluster were not aligned properly. To investigate the cause of such misalignments, E_2 selected one action, A1, for further examination.

Correcting misalignments (**R3**, **R4**). By examining the alignments between the selected action A1 and its neighbors, E_2 found that the end of action A1 was aligned with the beginning of action A2 (Fig. 12F2). It was abnormal because the beginnings of two actions should be aligned, as well as their ends. Further investigation revealed that a set of identical frames were detected in both A1 and A2, which caused the abnormal alignments. These abnormal alignments resulted in incorrect localization results by preventing the same action phases being aligned. To address this issue, E_2 corrected the alignments directly by dragging and dropping. Additionally, he examined and corrected the alignments of other neighbors (Fig. 12A3).

E_2 continued to examine and correct the alignments and localization results of other clusters. In total, five boundaries and 21 alignments of javelin throwing actions were corrected. The mAP was improved from **47.99%** to **49.56%**.

5.2.3 Other Action Localization and Post Analysis

Actions in other categories. Similar to the analysis of the high jumping and the javelin throwing categories, E_1 and E_2 analyzed the remaining five categories. 24 more boundaries and 165 more alignments were corrected. The mAP was increased from **49.56%** to **58.29%**. Overall, the experts improved the mAP of all the seven categories from **47.47%** to **58.29%** by correcting a total of 34 boundary annotations and 211 alignments.

Post analysis. After the case study, we conducted a quantitative evaluation to demonstrate the annotation efficiency of ActLocalizer in comparison with a human-in-the-loop method without ActLocalizer. The latter selects the most uncertain actions for users to annotate, which is one of the most widely-used active learning methods to save annotation efforts [29]. We compared the two methods in terms of the number of corrected boundaries and alignments, annotation time, and mAP. The annotation time for the baseline method was estimated based on the previous study of Ma *et al*. [37], which indicated an average of five minutes for annotating action boundaries in a one-minute video. The results summarized in Table 2 show that despite requiring additional must-link/cannot-link constraints, ActLocalizer largely reduces the number of boundaries and annotation time, indicating improved annotation efficiency. For example, ActLocalizer saves 78% annotation time compared with the baseline. Additionally, ActLocalizer achieves a higher mAP as well.

6 Expert Feedback and Discussion

Following the case study, four semi-structured interviews were conducted with the four experts we worked with. As E_3 and E_4 did not participate in the case study, we spent around 20 minutes to introduce the tool and the case study before the interviews. Each interview lasted 30 to 50 minutes. All the experts provided positive feedback on the usability of ActLocalizer. They also pointed out its limitations and suggested directions for future research.

6.1 Usability

Intuitive visual design and simple interactions. All the experts appreciated the intuitive design of the action view. For example, E_2 said, "Using lines to represent the actions and the distance between them to depict the alignments is very intuitive to me, and I quickly became familiar with the design." E_4 appreciated the simple interactions supported by ActLocalizer. For example, he found that correcting alignments was intuitive. It involved simply dragging and dropping operations to indicate which ones should or should not be aligned. "Since this tool does not require any prior knowledge of the underlying model, I believe that practitioners can learn to use it quickly," he commented.

Reducing annotation efforts. All the experts were impressed by a 78% reduction in annotation time. E_1 noted that the boundary recommendation helped save time in annotating imprecise boundaries. He said, "When I am unsure about boundaries, I often turn to the boundary recommendation feature. It largely saves my annotation efforts in the case study." E_2 also said that exploring the actions at different levels of detail saved his efforts in identifying wrong localization results, "It helps me quickly identify wrong localization results at the global level and zoom in for more in-depth analysis."

Generalization to non-single-frame-oriented localization methods. This work focuses on improving the performance of single-frame-oriented temporal action localization methods. However, the experts indicated that our method could be directly utilized to enhance other temporal action localization methods. The propagation-based action improvement method only relies on the predictions of the frames and the similarities between them, which can be obtained from any action localizer. Furthermore, the storyline aims to present the localization results and the alignments between actions, which are independent of the underlying localization methods.

6.2 Limitation

Algorithm scalability. The time complexity of the propagation-based action improvement method is $O(m^2)$, where m is the total number of frames in the training set. For the **THUMOS14** dataset with 63,575 frames, the localization results can be updated within one second upon the user corrections. However, when the number of frames reaches millions, the update could take several seconds or even minutes. Therefore, it is worth investigating how to accelerate the propagation-based action improvement method in the future. One potential solution is to develop an incremental algorithm that only updates the prediction of frames affected by the user corrections.

Non-utilization of multi-modality data. In the current implementation, the action localizer only utilizes image frames. However, videos also contain other modalities, such as audio. They can provide complementary information to improve the localization performance. For example, audio can indicate that two men in a video are conversing even when they have their backs toward the camera. As a result, it would be valuable to study how to integrate multi-modality data into the localization model and utilize complementary information between different modalities to improve performance. Furthermore, in cases where the performance is not satisfactory, it would be beneficial to explore ways of visually illustrating the complementary relationships between modalities. This can assist users in correcting misalignments and localization errors in a more efficient manner.

7 Conclusion

In this paper, we introduce ActLocalizer, a visual analysis method to improve the performance of action localizers trained on single-frame annotations. The key feature of our method is the tight integration of the propagation-based action improvement method with the storyline visualization to facilitate users in interactively correcting misalignments and wrong localization results. The corrected alignments are converted to constraints to derive better alignments for other actions. Based on the improved alignments, the corrected localization results are propagated to similar actions to further enhance localization performance. The effectiveness of ActLocalizer is demonstrated by the reduced annotation efforts in the quantitative evaluation, the presented findings in the case study, and the positive feedback from the experts after the case study.

REFERENCES

[1] R. Arias-Hernandez, L. T. Kaastra, T. M. Green, and B. Fisher. Pair analytics: Capturing reasoning processes in collaborative visual analytics. In *IEEE Hawaii International Conference on System Sciences*, pp. 1–10. Koloa, Kauai, 2011. doi: 10.1109/HICSS.2011.339 7

[2] P. Arora, Deepali, and S. Varshney. Analysis of K-Means and K-Medoids algorithm for big data. *Procedia Computer Science*, 78:507–512, 2016. doi: 10.1016/j.procs.2016.02.095 5

[3] R. P. Botchen, S. Bachthaler, F. Schick, M. Chen, G. Mori, D. Weiskopf, and T. Ertl. Action-based multifield video visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):885–899, 2008. doi: 10.1109/tvcg.2008.40 2

[4] K. Cao, J. Ji, Z. Cao, C.-Y. Chang, and J. C. Niebles. Few-shot video classification via temporal alignment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10618–10627, 2020. doi: 10.1109/cvpr42600.2020.01063 4

[5] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308, 2017. doi: 10.1109/cvpr.2017.502 7, 8

[6] C. Chen, Y. Guo, F. Tian, S. Liu, W. Yang, Z. Wang, J. Wu, H. Su, H. Pfister, and S. Liu. A unified interactive model evaluation for classification, object detection, and instance segmentation in computer vision. *IEEE Transactions on Visualization and Computer Graphics (to be published)*, 2023. doi: 10.1109/tvcg.2023.3326588 5

[7] C. Chen, Z. Wang, J. Wu, X. Wang, L.-Z. Guo, Y.-F. Li, and S. Liu. Interactive graph construction for graph-based semi-supervised learning. *IEEE Transactions on Visualization and Computer Graphics*, 27(9):3701–3716, 2021. doi: 10.1109/tvcg.2021.3084694 4

[8] C. Chen, J. Wu, X. Wang, S. Xiang, S.-H. Zhang, Q. Tang, and S. Liu. Towards better caption supervision for object detection. *IEEE Transactions on Visualization and Computer Graphics*, 28(4):1941–1954, 2022. doi: 10.1109/tvcg.2021.3138933 3

[9] C. Chen, J. Yuan, Y. Lu, Y. Liu, H. Su, S. Yuan, and S. Liu. OoDAnalyzer: Interactive analysis of out-of-distribution samples. *IEEE Transactions on Visualization and Computer Graphics*, 27(7):3335–3349, 2021. doi: 10.1109/tvcg.2020.2973258 5

[10] M. Chen, R. Botchen, R. Hashim, D. Weiskopf, T. Ertl, and I. Thornton. Visual signatures in video visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1093–1100, 2006. doi: 10.1109/tvcg.2006.194 2

[11] Z. Chen, S. Ye, X. Chu, H. Xia, H. Zhang, H. Qu, and Y. Wu. Augmenting sports videos with VisCommentator. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):824–834, 2022. doi: 10.1109/tvcg.2021.3114806 2

[12] D. Damen, T. Leelasawassuk, O. Haines, A. Calway, and W. W. Mayol-Cuevas. You-Do, I-Learn: Discovering task relevant objects and their modes of interaction from multi-user egocentric video. In *Proceedings of the British Machine Vision Conference*, pp. 1–13, 2014. doi: 10.5244/c.28.30 6

[13] G. Daniel and M. Chen. Video visualization. In *Proceedings of the IEEE Visualization Conference*, pp. 409–416, 2003. doi: 10.1109/visual.2003.1250401 2

[14] B. Duffy, J. Thiyagalingam, S. Walton, D. J. Smith, A. Trefethen, J. C. Kirkman-Brown, E. A. Gaffney, and M. Chen. Glyph-based video visualization for semen analysis. *IEEE Transactions on Visualization and Computer Graphics*, 21(8):980–993, 2015. doi: 10.1109/tvcg.2013.265 2

[15] E. Elhamifar, G. Sapiro, and S. S. Sastry. Dissimilarity-based sparse subset selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2182–2197, 2015. doi: 10.1109/tpami.2015.2511748 5

[16] Y. A. Farha and J. Gall. MS-TCN: Multi-stage temporal convolutional network for action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3575–3584, 2019. doi: 10.1109/cvpr.2019.00369 4

[17] D.-F. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisitetto correct phylogenetic trees. *Journal of Molecular Evolution*, 25(4):351–360, 1987. doi: 10.1007/bf02603120 5

[18] Z. Han, X. He, M. Tang, and Y. Lv. Video similarity and alignment learning on partial video copy detection. In *Proceedings of the ACM International Conference on Multimedia*, pp. 4165–4173, 2021. doi: 10.1145/3474085.3475549 3

[19] J. He, X. Wang, K. K. Wong, X. Huang, C. Chen, Z. Chen, F. Wang, M. Zhu, and H. Qu. VideoPro: A visual analytics approach for interactive video programming. *IEEE Transactions on Visualization and Computer Graphics (to be published)*, 2023. doi: 10.1109/tvcg.2023.3326586 2

[20] M. Held and R. M. Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied mathematics*, 10(1):196–210, 1962. doi: 10.1145/800029.808532 6

[21] M. Hoeferlin, B. Hoeferlin, G. Heidemann, and D. Weiskopf. Interactive schematic summaries for faceted exploration of surveillance video. *IEEE Transactions on Multimedia*, 15(4):908–920, 2013. doi: 10.1109/TMM.2013.2238521 2

[22] H. Idrees, A. R. Zamir, Y.-G. Jiang, A. Gorban, I. Laptev, R. Sukthankar, and M. Shah. The THUMOS challenge on action recognition for videos "in the wild". *Computer Vision and Image Understanding*, 155:1–23, 2017. doi: 10.1016/j.cviu.2016.10.018 6

[23] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum. Label propagation for deep semi-supervised learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5070–5079, 2019. doi: 10.1109/cvpr.2019.00521 4

[24] J. Ji, K. Cao, and J. C. Niebles. Learning temporal action proposals with fewer labels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7073–7082, 2019. doi: 10.1109/iccv.2019.00717 2

[25] H.-W. Kang, Y. Matsushita, X. Tang, and X.-Q. Chen. Space-time video montage. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1331–1338, 2006. doi: 10.1109/cvpr.2006.284 2

[26] K. Kurzhals, M. Hlawatsch, C. Seeger, and D. Weiskopf. Visual analytics for mobile eye tracking. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):301–310, 2017. doi: 10.1109/tvcg.2016.2598695 2

[27] K. Kurzhals, M. John, F. Heimerl, P. Kuznecov, and D. Weiskopf. Visual movie analytics. *IEEE Transactions on Multimedia*, 18(11):2149–2160, 2016. doi: 10.1109/tmm.2016.2614184 2

[28] F. Lekschas, B. Peterson, D. Haehn, E. Ma, N. Gehlenborg, and H. Pfister. PEAX: Interactive visual pattern search in sequential data using unsupervised deep representation learning. *Computer Graphics Forum*, 39(3):167–179, 2020. doi: 10.1101/597518 2

[29] D. D. Lewis. A sequential algorithm for training text classifiers: Corrigendum and additional data. *Acm Sigir Forum*, 29(2):13–19, 1995. 9

[30] Z. Li, Y. Abu Farha, and J. Gall. Temporal action segmentation from timestamp supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8365–8374, 2021. doi: 10.1109/cvpr46437.2021.00826 1, 2

[31] S. Liu, C. Chen, Y. Lu, F. Ouyang, and B. Wang. An interactive method to improve crowdsourced annotations. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):235–245, 2019. doi: 10.1109/tvcg.2018.2864843 3

[32] S. Liu, Y. Wu, E. Wei, M. Liu, and Y. Liu. StoryFlow: Tracking the evolution of stories. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2436–2445, 2013. doi: 10.1109/tvcg.2013.196 5

[33] X. Liu, Q. Wang, Y. Hu, X. Tang, S. Zhang, S. Bai, and X. Bai. End-to-end temporal action detection with transformer. *IEEE Transactions on Image Processing*, 31:5427–5441, 2022. doi: 10.1109/tip.2022.3195321 1

[34] Z. Liu, J. Ning, Y. Cao, Y. Wei, Z. Zhang, S. Lin, and H. Hu. Video swin transformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3202–3211, 2021. doi: 10.1109/cvpr52688.2022.00320 6

[35] B. Ma and A. Entezari. Volumetric feature-based classification and visibility analysis for transfer function design. *IEEE Transactions on Visualization and Computer Graphics*, 24(12):3253–3267, 2018. doi: 10.1109/tvcg.2017.2776935 5

[36] C.-X. Ma, J.-C. Song, Q. Zhu, K. Maher, Z.-Y. Huang, and H.-A. Wang. EmotionMap: Visual analysis of video emotional content on a map. *Journal of Computer Science and Technology*, 35(3):576–591, 2020. doi: 10.1007/s11390-020-0271-2 2

[37] F. Ma, L. Zhu, Y. Yang, S. Zha, G. Kundu, M. Feiszli, and Z. Shou. SF-Net: Single-frame supervision for temporal action localization. In *Proceedings of the European Conference on Computer Vision*, pp. 420–437, 2020. doi: 10.1007/978-3-030-58548-8_25 1, 2, 6, 7, 9

[38] A. H. Meghdadi and P. Irani. Interactive exploration of surveillance video through action shot summarization and trajectory visualization. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2119–2128, 2013. doi: 10.1109/tvcg.2013.168 2

[39] M. Müller. Dynamic time warping. *Information Retrieval for Music and*

Motion, pp. 69–84, 2007. doi: 10.1007/978-3-540-74048-3_4 3

[40] P. K. Newby, D. Muller, J. Hallfrisch, N. Qiao, R. Andres, and K. L. Tucker. Dietary patterns and changes in body mass index and waist circumference in adults. *The American Journal of Clinical Nutrition*, 77(6):1417–1425, 2003. doi: 10.1093/ajcn/77.6.1417 5

[41] C. Nguyen, Y. Niu, and F. Liu. Video Summagator: An interface for video summarization and navigation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 647–650, 2012. doi: 10.1145/2207676.2207767 2

[42] H.-S. Park and C.-H. Jun. A simple and fast algorithm for K-medoids clustering. *Expert Systems with Applications*, 36(2, Part 2):3336–3341, 2009. doi: 10.1016/j.eswa.2008.01.039 5

[43] J. Piazentin Ono, A. Gjoka, J. Salamon, C. Dietrich, and C. T. Silva. HistoryTracker: Minimizing human interactions in baseball game annotation. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2019. doi: 10.1145/3290605.3300293 2

[44] C. K. Reddy and B. Vinzamuri. A survey of partitional and hierarchical clustering algorithms. In *Data clustering*, pp. 87–110. Chapman and Hall/CRC, 2018. doi: 10.1201/9781315373515-4 4

[45] W. Sultani, C. Chen, and M. Shah. Real-world anomaly detection in surveillance videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6479–6488, 2018. doi: 10.1109/cvpr.2018.00678 1

[46] M. E. Swift, W. Ayers, S. Pallanck, and S. Wehrwein. Visualizing the passage of time with video temporal pyramids. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):171–181, 2022. 2

[47] H.-K. Tan, C.-W. Ngo, R. Hong, and T.-S. Chua. Scalable detection of partial near-duplicate videos by visual-temporal consistency. In *Proceedings of the ACM International Conference on Multimedia*, pp. 145–154, 2009. doi: 10.1145/1631272.1631295 3

[48] H.-K. Tan, X. Wu, C.-W. Ngo, and W.-L. Zhao. Accelerating near-duplicate video matching by combining visual similarity and alignment distortion. In *Proceedings of the ACM International Conference on Multimedia*, pp. 861–864, 2008. doi: 10.1145/1459359.1459506 4

[49] L. Tan, Y. Song, S. Liu, and L. Xie. ImageHive: Interactive content-aware image summarization. *IEEE Computer Graphics and Applications*, 32(1):46–55, 2011. doi: 10.1109/mcg.2011.89 2

[50] Y. Tanahashi and K.-L. Ma. Design considerations for optimizing storyline visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2679–2688, 2012. doi: 10.1109/TVCG.2012.212 5

[51] T. Tang, Y. Wu, L. Yu, Y. Li, and Y. Wu. VideoModerator: A risk-aware framework for multimodal video moderation in e-commerce. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):846–856, 2022. doi: 10.1109/tvcg.2021.3114781 2

[52] P. Togo, M. Osler, T. Sørensen, and B. Heitmann. Food intake patterns and body mass index in observational studies. *International Journal of Obesity*, 25(12):1741–1751, 2001. doi: 10.1038/sj.ijo.0801819 5

[53] X. Wang, S. Zhang, Z. Qing, Y. Shao, C. Gao, and N. Sang. Self-supervised learning for semi-supervised temporal action proposal. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1905–1914, 2021. doi: 10.1109/cvpr46437.2021.00194 2

[54] Y. Wang, M. Belkhatir, and B. Tahayna. Near-duplicate video retrieval based on clustering by multiple sequence alignment. In *Proceedings of the ACM International Conference on Multimedia*, pp. 941–944, 2012. doi: 10.1145/2393347.2396352 5

[55] A. Wu and H. Qu. Multimodal analysis of video collections: Visual exploration of presentation techniques in ted talks. *IEEE Transactions on Visualization and Computer Graphics*, 26(7):2429–2442, 2020. doi: 10.14711/thesis-991012757568503412 2

[56] H. Xia and Y. Zhan. A survey on temporal action localization. *IEEE Access*, 8:70477–70487, 2020. 1

[57] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005. doi: 10.1109/TNN.2005.845141 5

[58] L. Yang, J. Han, T. Zhao, T. Lin, D. Zhang, and J. Chen. Background-click supervision for temporal action localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):9814–9829, 2022. doi: 10.1109/tpami.2021.3132058 1, 2

[59] W. Yang, X. Ye, X. Zhang, L. Xiao, J. Xia, Z. Wang, J. Zhu, H. Pfister, and S. Liu. Diagnosing ensemble few-shot classifiers. *IEEE Transactions on Visualization and Computer Graphics*, 28(9):3292–3306, 2022. doi: 10.1109/tvcg.2022.3182488 5

[60] Y. Yu, D. Kruyff, J. Jiao, T. Becker, and M. Behrisch. PSEUDo: Interactive pattern search in multivariate time series with locality-sensitive hashing and relevance feedback. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):33–42, 2023. doi: 10.1109/tvcg.2022.3209431 2

[61] H. Zeng, X. Shu, Y. Wang, Y. Wang, L. Zhang, T.-C. Pong, and H. Qu. EmotionCues: Emotion-oriented visual summarization of classroom videos. *IEEE Transactions on Visualization and Computer Graphics*, 27(7):3168–3181, 2021. doi: 10.1109/tvcg.2019.2963659 2, 5

[62] H. Zeng, X. Wang, A. Wu, Y. Wang, Q. Li, A. Endert, and H. Qu. EmoCo: Visual analysis of emotion coherence in presentation videos. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):927–937, 2020. doi: 10.1109/tvcg.2019.2934656 2

[63] C.-L. Zhang, J. Wu, and Y. Li. ActionFormer: Localizing moments of actions with transformers. In *Proceedings of the European Conference on Computer Vision*, pp. 492–510, 2022. doi: 10.1007/978-3-031-19772-7_29 1

[64] P. Zhao and L. Lai. Efficient classification with adaptive KNN. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 11007–11014, 2021. doi: 10.1609/aaai.v35i12.17314 4

[65] T. Zhao, J. Han, L. Yang, and D. Zhang. Equivalent classification mapping for weakly supervised temporal action localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. doi: 10.1109/TPAMI.2022.3178957 1

[66] X. Zhu. *Semi-supervised learning with graphs*. PhD thesis, Carnegie Mellon University, 2005. 4