# Thread-Safe: Towards Recognizing Human Actions Across Shot Boundaries

Minh Hoai[1,2] and Andrew Zisserman[1]

[1]Visual Geometry Group, Department of Engineering Science,
University of Oxford, Oxford, OX1 3PJ, UK.
[2]Department of Computer Science,
Stony Brook University, Stony Brook, NY 11794, USA.

**Abstract.** We study the task of recognizing human actions in video whilst paying attention to the shot and thread editing structure. Most existing action recognition algorithms ignore this structure, but it is generally present in edited TV and film material.

To this end, we make the following contributions: first, we introduce a new dataset of human actions to study the occurrence/reoccurrence of patterns of human actions in edited TV material; second, we propose composing a video into threads of related shots, removing some of the discontinuities due to shot boundaries; and third, we show the benefits of utilizing video threads in recognizing human actions. The experiments demonstrate that human action retrieval accuracy can be improved using threads.

## 1 Introduction

Humans are the primary focus of many TV shows, and consequently recognizing their actions is important for automated semantic analysis of TV material. This importance is well recognized, and several datasets (e.g.,[1–4]) and many approaches have been proposed (e.g., [1, 2, 4–7]). Existing approaches, however, ignore the structure and discontinuities in edited material. For examples, many algorithms track object patches or compute motion cues across shot boundaries, which are irrelevant to the actual content of an action.

In this paper, we propose to embrace the editing structure when recognizing human actions. We reverse the editing and decompose a video into *threads* [8]. Each thread is an ordered sequence of shots, filming the same scene by the same camera. Recall that a scene is typically filmed by multiple cameras at multiple angles, and a video is composed by cutting and joining video clips from multiple cameras. These video clips are referred to as shots and the transitions between them are shot boundaries.

Fig. 1 shows a typical video sequence and illustrates the importance of considering threads when recognizing human actions. This video sequence depicts a scene of an affectionate kiss. It consists of several shots, which can be connected to form several interleaving threads. Apart from removing abrupt discontinuities due to shot boundaries, threads can be used to separate parts of the video sequence that are irrelevant to the action of interest.

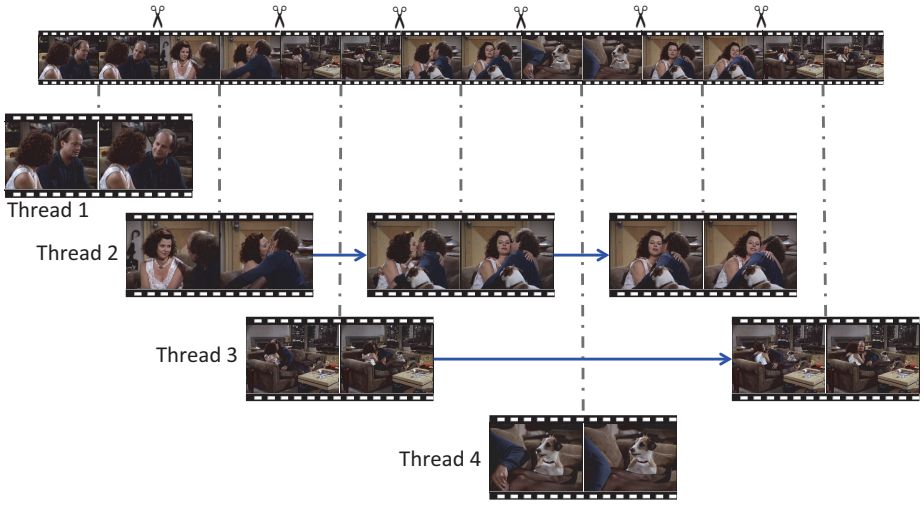


**Fig. 1. A typical scene with shots and threads.** This video sequence of an affectionate kiss consists of several interleaving threads. Thread 1 sets the context for the kiss, while Threads 2 and 3 portray the kiss at different angles. Thread 4 shows a dog being amused by the affection between two people. This thread shows a part of the scene, but it is completely irrelevant to the kissing action.

Video threads and shot grouping have been considered before, but primarily for scene segmentation. Zhai & Shah [9] proposed an MCMC algorithm for clustering shots into scenes. Yeung *et al.* [10] constructed shot connectivity graph and used hierarchical clustering for scene segmentation. Kender & Yeo [11] detected scene boundaries by measuring coherence between shots and taking the local minima. Chasanis *et al.* [12] used sequence alignment and shot threading for scene detection. Cour *et al.* [8] proposed a weakly supervised algorithm that uses screenplays and close captions to parse a movie into a hierarchy of scenes, threads, and shots. Lehane *et al.* [13, 14] considered repeating shots with still cameras and used Finite State Machines to detect dialogues. Pickup & Zisserman [15] used threads to spot visual continuity errors in movies. Tapaswi *et al.* [16] utilized threads for visualizing character interactions. All of these works, however, do not study human actions and the benefits of video threading in recognition.

Unfortunately, no existing datasets for human actions, including [2–4, 17–19], can be used to study the editing structure and the benefits of using threads for recognition, as they lack annotation and contextual surround (video sequences before and after the actions). Therefore, we introduce here a new dataset of human actions. Our dataset contains more than 4000 video samples, divided into shots with annotated occurrences of human actions. The data is extracted

from a large collection of TV series with different types of genres. This dataset is the first of its kind, and this is a contribution of our paper.

## 2 Dataset

We introduce a large dataset of video threads with annotated occurrences of human actions. The dataset consists of video samples for 13 popular actions, extracted from 15 different TV series. The video samples are divided into shots, and Amazon Mechanical Turk (MTurk) is used to verify the occurrence of human actions in the shots. This dataset can be used to study the occurrence/reoccurrence patterns of human actions in edited TV material and the benefits of using video threads to recognize human actions. These tasks can not be performed with existing human action datasets [2–4, 17–19]. The dataset is available at `http://www.robots.ox.ac.uk/~vgg/data/threadsafe`.

### 2.1 Data collection with script mining

The data is extracted from a large collection of edited TV material. This collection consists of 15 different TV series, each with two entire seasons. The TV series are: Frasier, Married With Children, Millennium, Friends, Andromeda, Gilmore Girls, Smallville, Farscape, Seinfeld, Scrubs, Lost, The Big Bang Theory, Star Trek TNG, Desperate Housewives, and Roswell. These TV series cover a wide range of genres, from family and friends to crime investigation and science fiction. There are 658 different episodes. The duration of each episode ranges from 20 to 60 minutes.

To obtain rough locations of human actions, we use video-aligned scripts [20]. Scripts are text documents that contain dialogs and scene descriptions. Scripts are generally available for popular TV shows. All TV series considered in our dataset have scripts, which are publicly available on the Internet. Scripts, however, do not provide time synchronization with the video. Following [20], we resolve this issue by synchronizing script dialogs with subtitles. Subtitles are already synchronized with videos through timestamps, and they can be easily downloaded from the Internet or copied from DVDs. Using dynamic time warping, we match script text with subtitles and transfer the time information from subtitles to scripts.

From the scene descriptions in video-aligned scripts, we collect video samples for 13 actions: answer phone, drive car, eat, fight, get out car, shake hand, hug, kiss, run, sit down, sit up, stand up, and high five. These actions frequently occur in TV shows. They are the superset of the actions considered in Hollywood2 [2] and TVHI [4] datasets, two benchmarks for human action recognition.

To collect video samples from scene descriptions, we build a text search engine. For a particular action, we first identify a set of relevant keywords and phrases. For example, the keywords and phrases for *Shakehand* are: "shakehand", "handshake", "shake * * hand", and "hand * * shake". Here, the "*" is a wild card, and it can be matched to any word. The search engine also supports

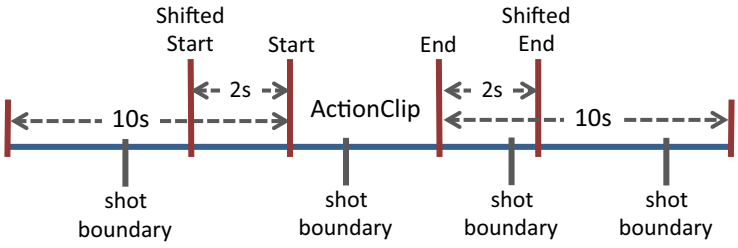


**Fig. 2. Video intervals for annotation**. We ask MTurk workers to annotate the extended video sequence of an action clip. The video sequence is divided into non-overlapping intervals based on shot boundaries and the start and end markings of the action clip. The shot boundaries depicted here are just for illustrative purpose; their actual locations vary.

word stemming, so "shake" is equivalent to "shakes" and "shaking". With wildcard and stemming support, searching for "shake * * hand" will return all the following sentences if they are in the scene descriptions: 1) they shake hands; 2) he shakes her hand; and 3) Leonard, shaking Penny's hand, smiled excitedly. We refer the video samples obtained using this way as ActionClips (ACs).

### 2.2 Annotation verification and refinement

We use MTurk to verify and refine the annotation for the ACs that were automatically obtained using aligned scripts. We also collect annotation for the video sequences that immediately precede or follow the action clips. For each AC, we first extend it by 10 seconds at both ends of the clip, as illustrated in Fig. 2. This extended sequence will be referred to as a Ten second Extended Action Clip (TEAC). We divide the TEAC into non-overlapping intervals based on: 1) shot boundaries; 2) the start and the end of the AC; and 3) the shifted start time (2 seconds earlier) and the shifted end time (2 seconds later) of the AC. This division procedure is illustrated in Fig. 2. Each obtained interval is a video shot or a part of a video shot. Hereafter, we will refer to these intervals simply as shots for brevity. We ask three MTurk workers to identify the shots that visually contain the action of interest.

### 2.3 Dataset statistics and consistency of annotation

Tab. 1 displays dataset statistics and the consistency of annotation. The second and third columns show the number of clips and the number of shots for each of the 13 actions. Altogether, there are around 5000 video samples with 64000 shots. Each *shot* is labeled by three MTurk workers, and the last three columns of Tab. 1 shows the percentage of shots where all three MTurk workers agree. The overall percentage of agreement is 86.3%, and 9.1% of the shots is unanimously marked as containing the action of interest. The percentage of shots

**Table 1. Dataset statistics and annotation consistency.** Each shot is annotated by three MTurk workers. The last three columns show the percentage of shots that receive the same annotation from all MTurk workers. The percentage of unanimous decision is 86.3%.

| | | | %shots with agreed annotation | | |
|---|---|---|---|---|---|
| Action | #clips | #shots | No | Yes | No/Yes |
| AnswerPhone | 237 | 2768 | 62.8% | 25.6% | 88.4% |
| DriveCar | 171 | 2419 | 82.8% | 4.5% | 87.3% |
| Eat | 307 | 3539 | 64.1% | 11.0% | 75.1% |
| Fight | 383 | 6268 | 68.4% | 9.7% | 78.1% |
| GetOutCar | 159 | 2246 | 87.5% | 4.5% | 92.0% |
| Shakehand | 181 | 2090 | 78.0% | 7.8% | 85.8% |
| Hug | 431 | 4831 | 70.6% | 17.4% | 88.0% |
| Kiss | 774 | 8550 | 75.4% | 12.0% | 87.4% |
| Run | 1441 | 20758 | 80.4% | 6.6% | 87.0% |
| SitDown | 459 | 4921 | 83.0% | 6.2% | 89.2% |
| SitUp | 133 | 1608 | 84.5% | 3.0% | 87.5% |
| StandUp | 274 | 3421 | 87.9% | 4.3% | 92.2% |
| Highfive | 53 | 571 | 87.6% | 5.4% | 93.0% |
| Total | 5003 | 63990 | 77.2% | 9.1% | 86.3% |

containing the action of interest varies from action to action. This is because human actions are different and they are portrayed differently in edited TV material. For example, a video sample for AnswerPhone tends to alternate between two threads of two people talking on the phone for an extended period of time. This is why the percentage of AnswerPhone shots in an AnswerPhone sample is relatively high (25.6%). Meanwhile, the action SitUp or StandUp are usually shown briefly. This explains the low percentages of SitUp and StandUp shots.

### 2.4 Temporal extent of actions

Where does an action occur? To answer this question, we report the percentage of times ACs and their surrounding video sequences contain the action of interest. Refer to Fig. 3 and consider an ActionClip (AC). Let PreAC and PostAC be the video sequences obtained by extending the action clip to the previous or next shot boundaries. PreAC2 and PostAC2 are the extension before PreAC and after PostAC, respectively. The occurrence percentage of the actions in these video parts are reported in Tab. 2. Here, a video sequence is believed to contain an action of interest if it contains a shot that is marked to contain the action by at least two MTurk workers.

As can be seen in Tab. 2, the action samples obtained using video-aligned scripts are useful, but noisy. On the one hand, the occurrence percentage for an action inside ACs is high, 52.3%. On the other hand, this action-occurrence

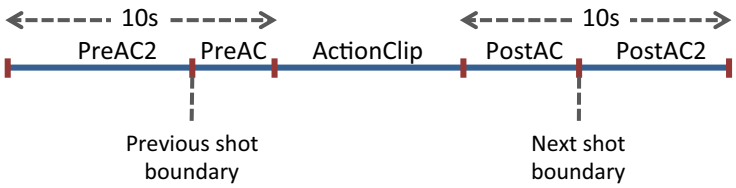


**Fig. 3. Video sequences surrounding an action clip**. Given an ActionClip (AC), PreAC and PostAC are obtained by extending the AC to the closest shot boundaries. PreAC2 and PostAC2 are continual extensions of PreAC and PostAC.

percentage is far from 100%. Furthermore, consider the occurrence percentage for all combined temporal locations (last column of Tab. 2). The mean value is 71.3%, which is essentially the percentage of times a verbally-described action is visually seen. This reflects the nature of scene descriptions in video scripts: many of them are based on inference instead of visualization. For example, an out-of-frame handshake between two people can still be inferred based on other cues such as audio or the greeting scenario.

The AC is the best location to extract an action sample because its action-occurrence percentage is significantly higher than those of other temporal locations. However, AC does not usually contain the entire action as the action percentage for PreAC and PostAC are also significant. Thus, perhaps the AC should be extended to capture the action in its entirety. We propose to consider Extended Action Clip (EAC), obtained by extending the AC to the previous and next shot boundaries, but clipping the extension at 2s. This 2s clipping is to avoid a situation where the previous or next shot boundaries are far away. Tab. 3 shows the action-occurrence percentage for EAC and its preceding and following video sequences. Notably, the action-occurrence percentage for PostEAC is significantly higher than PreEAC. This suggests that the beginning of an action is more precisely aligned with a scene description than the end of the action.

## 3 Video Threads

A video sequence is decomposed into interleaving threads. Each thread is an ordered sequence of video shots filmed from the same camera for the same scene. This section describes a shot boundary detection algorithm and the algorithm for joining shots into threads.

### 3.1 Shot Boundary Detection

Shot boundary detection is a very well studied area [21–23]. Our algorithm is based on several principles suggested in [21, 22] such as temporal discontinuity and adaptive threshold, but it uses more recent visual features namely HOG [24] and SIFT [25]. Based on HOG, the algorithm produces a set of candidate shot

**Table 2. Occurrence percentage for human actions at different temporal locations.** AC is the video sample automatically obtained by video-aligned script. PreAC, PreAC2, PostAC, and PostAC2 are video sequences before or after AC, as depicted in Fig. 3.

| | PreAC2 | PreAC | AC | PostAC | PostAC2 | Anywhere |
|---|---|---|---|---|---|---|
| AnswerPhone | 13.9% | 17.3% | 73.8% | 67.5% | 54.9% | 89.9% |
| DriveCar | 5.8% | 4.1% | 43.9% | 13.5% | 16.4% | 53.8% |
| Eat | 13.0% | 13.0% | 55.0% | 42.3% | 42.0% | 81.4% |
| Fight | 16.2% | 11.2% | 34.5% | 10.2% | 20.4% | 44.4% |
| GetOutCar | 3.1% | 7.5% | 54.7% | 21.4% | 9.4% | 69.2% |
| Shakehand | 6.1% | 18.2% | 49.2% | 23.2% | 11.0% | 70.2% |
| Hug | 11.4% | 22.0% | 68.4% | 40.6% | 27.6% | 87.9% |
| Kiss | 9.3% | 21.7% | 70.9% | 21.2% | 13.4% | 86.6% |
| Run | 13.3% | 12.5% | 44.2% | 18.5% | 20.2% | 61.0% |
| SitDown | 3.9% | 11.3% | 51.4% | 23.7% | 10.9% | 81.0% |
| SitUp | 5.3% | 6.8% | 41.4% | 13.5% | 6.8% | 58.6% |
| StandUp | 9.5% | 15.0% | 50.7% | 13.5% | 11.3% | 75.5% |
| Highfive | 9.4% | 24.5% | 41.5% | 5.7% | 7.5% | 67.9% |
| Mean | 9.2% | 14.3% | 52.3% | 24.2% | 19.4% | 71.3% |

boundaries by thresholding the difference between pairs of consecutive frames. Subsequently, SIFT matching is used to remove false candidates. Evaluated on the TVHI dataset [4], this shot boundary detection algorithm has no false positive and 1 false negative. The details of HOG proposal and SIFT verification are given below.

**Shot boundary proposal using HOG.** For each video frame of a video sequence, our algorithm first normalizes the frame to $128 \times 96$ pixels and computes the HOG feature vector with cell size of 8. Let $\mathbf{h}_i$ be the HOG feature vector for frame $i$, and let $d_i$ be the HOG-difference between frame $i$ and its previous frame: $d_i = ||\mathbf{h}_i - \mathbf{h}_{i-1}||_1$. Centering at $i$, consider the values of HOG-difference in a temporal window around $i$ (i.e., $\{d_{i-5}, \cdots, d_{i+5}\}$), and let $m_i$ and $s_i$ be the mean and standard deviation respectively. We first discard all frames $i$ such that $d_i < threshold$, where $threshold$ is set to be the 98.5 percentile of all HOG-difference values. We further remove all frames where the HOG difference is not significantly higher than the mean value of the HOG differences in the supporting window. Specifically, we remove frame $i$ from the list of shot boundary candidates if $d_i < m_i + 1.5s_i$. This procedure can be performed in real time because: i) HOG feature extraction is fast, and ii) the mean and standard deviation for a sliding window can be computed using convolution.

**False positive removal with SIFT matching.** The set of candidate boundaries returned by the above procedure can still contain false positives due to fast

**Table 3. Action-occurrence percentage for extended action clip.** EAC: obtained by extending the AC to the previous and next shot boundaries, but clipping the extension at 2s. PreEAC: the video sequence right before EAC. PostEAC: the video sequence right after EAC.

| | PreEAC | EAC | PostEAC |
|---|---|---|---|
| AnswerPhone | 15.6% | 82.7% | 71.3% |
| DriveCar | 6.4% | 45.0% | 19.9% |
| Eat | 16.9% | 64.5% | 59.0% |
| Fight | 17.2% | 36.3% | 21.1% |
| GetOutCar | 5.0% | 60.4% | 17.6% |
| Shakehand | 8.3% | 61.3% | 17.1% |
| Hug | 16.2% | 77.3% | 40.4% |
| Kiss | 13.2% | 78.6% | 20.4% |
| Run | 15.1% | 50.7% | 24.4% |
| SitDown | 6.8% | 67.3% | 18.5% |
| SitUp | 6.8% | 49.6% | 12.8% |
| StandUp | 10.6% | 65.0% | 15.0% |
| Highfive | 11.3% | 64.2% | 7.5% |
| Mean | 11.5% | 61.8% | 26.5% |

motion. We address this problem using SIFT matching [25]. Two video frames are considered to be in the same shot if there are at least 40 geometrically valid matches (the horizontal and vertical distance between two matched descriptors must be smaller than a quarter of the frame width and height respectively). SIFT matching is much slower than HOG computation. Fortunately, we only need to perform SIFT matching for a small set of shot boundary candidates.

### 3.2 Joining video shots into threads

Given an ordered sequence of video shots $\mathbf{s}_1, \cdots, \mathbf{s}_k$, we link shots into threads as follows. First, we construct an undirected connectivity graph where each node represents a shot. Two nodes $i$ and $j$ are connected if $0 < i - j < 10$ and the first frame of shot $i$ can be matched with the last frame of shot $j$ (using SIFT matching [25]). We then find all connected components of the graph. Each connected component defines a video thread of shots. The complexity for building the graph and for finding the connected components is $\mathbf{O}(k)$, i.e., linear in the number of shots.

Tab. 4 shows some summary statistics of threads and shots in TEACs. Some actions such as Fight and Run contain more threads than other actions. Compared Columns (A) and (B), it can be seen that not all threads contain an action of interest. In fact, the proportion of threads that contain an action can be small (e.g., GetOutCar and SitDown). This suggests the importance of considering threads for recognizing human actions. Compared Columns (C) and (D), on average, a thread that does not contain an action has fewer shots than a thread

**Table 4. Mean numbers of threads and shots in TEACs**. Column A: mean number of threads. B: mean number of threads containing the action of interest. C: mean number of shots in threads without the action. D: mean number of shots in threads with the action. E: percentage of shots that contain the action in the threads that contain the action.

| | Mean #threads | | Mean #shots | | |
|---|---|---|---|---|---|
| | A | B | C | D | E |
| AnswerPhone | 5.24 | 2.07 | 1.34 | 1.65 | 89.9% |
| DriveCar | 8.09 | 1.78 | 1.32 | 1.32 | 93.8% |
| Eat | 5.02 | 1.73 | 1.38 | 1.56 | 88.6% |
| Fight | 11.91 | 4.87 | 1.31 | 1.45 | 91.9% |
| GetOutCar | 7.39 | 1.21 | 1.36 | 1.43 | 85.0% |
| Shakehand | 4.90 | 1.24 | 1.41 | 1.80 | 73.4% |
| Hug | 5.08 | 1.75 | 1.28 | 1.71 | 80.6% |
| Kiss | 4.78 | 1.54 | 1.36 | 1.72 | 78.8% |
| Run | 8.67 | 2.13 | 1.28 | 1.38 | 88.4% |
| SitDown | 4.74 | 1.12 | 1.34 | 1.53 | 80.8% |
| SitUp | 5.64 | 1.17 | 1.34 | 1.66 | 76.4% |
| StandUp | 5.70 | 1.22 | 1.37 | 1.71 | 76.0% |
| Highfive | 4.56 | 1.25 | 1.30 | 1.89 | 66.4% |
| Mean | 6.29 | 1.77 | 1.34 | 1.60 | 82.3% |

that does. For threads that contain an action, the percentages of shots containing the action are high, varying from 66.4% to 93.8% (last column of Tab. 4).

## 4 Experiments and Analysis

### 4.1 Experimental Setup

**Training and Testing data.** We split video samples into the test and training subsets such that the two subsets do not share samples from the same TV series. We split the TV series into two separate subsets, aiming to have scene and genre diversity in both training and testing sets. In particular, the following TV series are used for training: Frasier, Married With Children, Millennium, Friends, Andromeda, Gilmore Girls, Smallville, and Farscape. In testing, to ensure the correctness of test data, a video sequence (or thread) is considered positive only if all three MTurk workers believe it contains the action. In training, to increase the amount of training data, a video sequence (or thread) is considered as a positive training sample if it is annotated to contain the action by at least two MTurk workers.

**Trajectory features.** The feature representation is based on improved Dense-Trajectory Descriptors (DTDs) [6]. DTD extracts dense trajectories and encodes

gradient and motion cues along trajectories. Each trajectory leads to four feature vectors: Trajectory, HOG, HOF, and MBH, which have dimensions of 30, 96, 108, and 192 respectively. We refer the reader to [6] for more details.

The procedure for extracting DTDs is the same as [6] with two subtle modifications: (i) videos are normalized to have the height of 360 pixels, and (ii) frames are extracted at 25 fps. These modifications are added to standardize the feature extraction procedure across videos and datasets. They do not significantly alter the performance of the action recognition system [26]f.

**Fisher vector encoding.** To encode features, we use Fisher vectors [27]. A Fisher vector encodes both first and second order statistics between the feature descriptors and a Gaussian Mixture Model (GMM). In [6], Fisher vector shows an improved performance over bag of features for action classification. Following [27, 6], we first reduce the dimension of DTDs by a factor of two using Principal Component Analysis (PCA). We set the number of Gaussians to $k = 256$ and randomly sample a subset of 1,000,000 features from the training sets of TVHI and Hollywood2 to learn the GMM. There is one GMM for each feature type. A video sequence is represented by a $2dk$ dimensional Fisher vector for each descriptor type, where $d$ is the descriptor dimension after performing PCA. As in [27, 6], we apply power ($\alpha = 0.5$) and $L_2$ normalization to the Fisher vectors. We combine all descriptor types by concatenating their normalized Fisher vectors, leading to a single feature vector of $109,056$ dimensions.

**Least-Squares SVM.** For classification, we propose to use Least-Squares Support Vector Machines (LSSVM) [28]. LSSVM, also known as kernel Ridge regression [29], has been shown to perform equally well as SVM in many classification benchmarks [30]. LSSVM has a closed-form solution, which is a computational advantage over SVM. Furthermore, once the solution of LSSVM has been computed, the solution for a reduced training set obtaining by removing any training data point can found efficiently. This enables reusing training data for further calibration (e.g., used in [31, 32, 26]). This section reviews LSSVM and the leave-one-sample-out formula.

Given a set of $n$ data points $\{\mathbf{x}_i | \mathbf{x}_i \in \Re^d\}_{i=1}^n$ and associated labels $\{y_i | y_i \in \{1, -1\}\}_{i=1}^n$, LSSVM optimizes the following:

$$\underset{\mathbf{w},b}{\text{minimize}} \;\; \lambda ||\mathbf{w}||^2 + \sum_{i=1}^{n} (\mathbf{w}^T \mathbf{x}_i + b - y_i)^2. \tag{1}$$

For high dimensional data ($d \gg n$), it is more efficient to obtain the solution for $(\mathbf{w}, b)$ via the representer theorem, which states that $\mathbf{w}$ can be expressed as a linear combination of training data, i.e., $\mathbf{w} = \sum_{i=1}^n \alpha_i \mathbf{x}_i$. Let $\mathbf{K}$ be the kernel matrix, $k_{ij} = \mathbf{x}_i^T \mathbf{x}_j$. The optimal coefficients $\{\alpha_i\}$ and the bias term $b$ can be found using closed-form formula: $[\boldsymbol{\alpha}^T, b]^T = \mathbf{M}\mathbf{y}$. Where $\mathbf{M}$ and other auxiliary

variables are defined as:

$$\mathbf{R} = \begin{bmatrix} \lambda\mathbf{K} & \mathbf{0}_n \\ \mathbf{0}_n^T & 0 \end{bmatrix}, \mathbf{Z} = \begin{bmatrix} \mathbf{K} \\ \mathbf{1}_n^T \end{bmatrix}, \mathbf{C} = \mathbf{R} + \mathbf{Z}\mathbf{Z}^T, \mathbf{M} = \mathbf{C}^{-1}\mathbf{Z}, \mathbf{H} = \mathbf{Z}^T\mathbf{M}. \quad (2)$$

If $\mathbf{x}_i$ is removed from the training data, the optimal coefficients can be computed:

$$\begin{bmatrix} \boldsymbol{\alpha}_{(i)} \\ b_{(i)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix} + \left( \frac{[\boldsymbol{\alpha}^T \ b]\mathbf{z}_i - y_i}{1 - h_{ii}} \right) \mathbf{m}_i. \quad (3)$$

Here, $\mathbf{z}_i$ is the $i^{th}$ column vector of $\mathbf{Z}$ and $h_{ii}$ is the $i^{th}$ element in the diagonal of $\mathbf{H}$. Note that $\mathbf{R}, \mathbf{Z}, \mathbf{C}, \mathbf{M}$, and $\mathbf{H}$ are independent of the label vector $\mathbf{y}$. Thus, training LSSVMs for multiple classes is efficient as these matrices need to be computed once. A more gentle derivation of the above formula is given in [33].

### 4.2 The benefits of knowing relevant video threads

Not every thread of a video sequence is relevant for recognizing human actions, as we discussed earlier. This subsection shows the empirical benefits of knowing relevant threads.

Consider the task of classifying whether a video sequence contains an action of interest. We create training data for this experiment by combining EACs of all actions. For a particular action, the positive samples are EACs that are annotated to contain the action, and the negative samples are EACs for other actions. Recall that an EAC is believed to contain the action if it has a shot containing the action (agreed by at least two MTurk workers). Similarly, we create a test set by extracting EACs from testing TV series (which are disjoint from training TV series). However, for testing, we only use EACs that are unanimously agreed to contain the actions by all MTurk workers. This is to ensure the correctness of test data.

We consider several methods, with and without video threading. In all cases the task is to classify whether an EAC contains the action of interest in the test data. However, we change the *representation* of the EAC in both training and testing over the different methods as follows. If video threading is not used, a feature vector (using Fisher vector encoding of dense trajectories) is computed for the entire video sequence, ignoring the discontinuities due to shot boundaries. If video threading is used, an EAC is divided into several threads and a feature vector (using Fisher vector encoding) is computed for each thread independently. The feature vectors of all threads are then aggregated to be the feature vector for the EAC. The task considered in this subsection should not be confused with individual thread classification, which is investigated in the next subsection.

We measure performance using Average Precision (AP), which is an accepted standard for action recognition [2, 4, 34–38]. Tab. 5 compares the performance of using and not using threads. *Clip* is a popular approach [34, 37, 6], which treats an EAC as the whole and extract feature trajectories across shot boundaries. *AllThreads* is the method that decomposes an EAC into threads and computes feature trajectories for each thread separately. Here, we join the shots of a thread

**Table 5. The benefits of discarding irrelevant threads in training and testing.** This shows APs of several methods for classifying video clips. *Clip*: a video clip is treated as the whole, with dense trajectories computed across shot boundaries. *AllThreads*: a video clip is decomposed into threads and dense trajectories and a Fisher Vectors are computed for each thread separately; each clip is then represented by the mean of Fisher Vectors. *AllThreads+Clip*: combining AllThreads and Clip (by concatenating feature vectors). *RelevantThreads*: similar to AllThreads, but assuming we know which threads contain the actions so irrelevant threads can be discarded. These results indicate the importance of finding relevant threads.

| | Ignore threads | Use threads | | |
|---|---|---|---|---|
| Action | Clip | AllThreads | AllThreads+Clip | RelevantThreads |
| AnswerPhone | 22.1 | 21.6 | 23.2 | 31.5 |
| DriveCar | 44.5 | 51.7 | 49.6 | 70.8 |
| Eat | 38.1 | 35.0 | 38.9 | 37.3 |
| Fight | 35.8 | 29.5 | 35.0 | 55.6 |
| GetOutCar | 21.9 | 26.0 | 24.5 | 41.5 |
| Shakehand | 20.9 | 18.9 | 21.2 | 30.6 |
| Hug | 41.2 | 39.3 | 40.6 | 44.2 |
| Kiss | 69.6 | 68.8 | 69.8 | 76.6 |
| Run | 87.7 | 88.3 | 88.8 | 94.9 |
| SitDown | 71.5 | 69.7 | 71.3 | 80.2 |
| SitUp | 16.1 | 14.5 | 13.7 | 12.1 |
| StandUp | 19.1 | 17.8 | 20.1 | 26.6 |
| Highfive | 13.2 | 11.9 | 12.2 | 9.3 |
| mean | 38.6 | 37.9 | 39.1 | 47.0 |

together and compute dense trajectories normally, as explained in Subsection 4.1. For an EAC with multiple threads, we average the feature vectors of all threads and perform L2-normalization. Notably, AllThreads is slightly worse than Clip. This suggests that the shot boundaries may provide some indicative cues toward recognizing human actions in edited material, even though they are not parts of an action. *AllThreads+Clip* is the method that combine both threads and the whole EAC, by concatenating feature vectors computed for both. *RelevantThreads* is the method that only aggregates feature vectors for threads that contain the actions. This leads to huge AP improvement, suggesting the importance of identifying relevant threads.

### 4.3 Video thread classification

The previous subsection shows the benefits of knowing relevant threads. In this subsection, we consider the task of recognizing those threads, as opposed to classifying the whole action clip.

We create the training data for this experiment by combining positive threads of EACs of all actions. For a particular action, the positive samples are threads

**Table 6. Recognizing relevant threads of human actions**. This table shows the APs for action recognition where the testing samples are video threads. *Clip*: training samples are EACs without using threads. *PT*: training samples are threads; positive samples are positive threads extracted inside the EACs. *PT+NITAP*, *PT+NITAN*, *PT+POTAP*: same as PT but with additional training samples. NITAP: use negative threads inside EACs as positive training samples. NITAN: use negative threads inside EACs as negative training samples. POTAP: use positive threads outside EACs as positive training samples.

| | | Training data are video threads | | | |
|---|---|---|---|---|---|
| | | | Using additional training threads | | |
| Action | Clip | PT | PT+NITAP | PT+NITAN | PT+POTAP |
| AnswerPhone | 25.4 | 27.9 | 20.2 | 28.4 | 23.2 |
| DriveCar | 54.7 | 63.5 | 52.8 | 60.5 | 67.5 |
| Eat | 31.0 | 33.4 | 23.3 | 31.4 | 34.5 |
| Fight | 28.9 | 46.3 | 44.5 | 43.0 | 48.7 |
| GetOutCar | 23.0 | 36.5 | 19.3 | 34.6 | 38.1 |
| Shakehand | 21.2 | 28.3 | 21.2 | 28.8 | 26.3 |
| Hug | 38.7 | 43.5 | 41.6 | 39.7 | 44.4 |
| Kiss | 72.0 | 75.2 | 64.7 | 74.2 | 76.2 |
| Run | 92.4 | 93.7 | 85.8 | 93.4 | 94.1 |
| SitDown | 77.7 | 77.8 | 63.9 | 77.1 | 79.0 |
| SitUp | 12.4 | 11.3 | 3.4 | 12.2 | 9.6 |
| StandUp | 22.4 | 24.3 | 11.4 | 22.5 | 25.0 |
| Highfive | 13.8 | 8.4 | 5.0 | 8.3 | 9.3 |
| mean | 39.5 | 43.8 | 35.2 | 42.6 | 44.3 |

that are annotated to contain the action, and the negative samples are positive threads for other actions. In training, a thread is believed to contain the action if it has a shot containing the action (agreed by at least two MTurk workers). Similarly, we create a test set of positive threads of testing EACs. For testing, we only use threads that are unanimously agreed to contain the actions by all MTurk workers. Again, this is to ensure the correctness of test data.

Tab. 6 shows the performance for recognizing relevant action threads. *Clip* is the method in which training samples are the whole EACs without considering threads. As can be seen, Clip performs relatively poorly compared to *PT*. PT is the method in which training samples are threads. *PT+NITAP*, *PT+NITAN*, *PT+POTAP*: are similar to PT, but with additional training samples. PT+NITAP is the method where negative threads inside EACs are mistakenly used as additional positive training samples. As can be seen, a mistake for identifying relevant threads is devastating. This reaffirms the importance for identifying relevant threads. PT+NITAN is the method where negative threads inside EACs are used as additional negative training samples. Surprisingly, this does not improve the performance. This is perhaps due to the importance of

contextual cues: a thread might not portray an action, but still provides discriminative cue for recognizing the action. PT+POTAP is the method where additional positive training samples are positive threads extracted outside EACs (i.e., either PreEACs or PostEACs). PT+POTAP improves the performance of PT; this reemphasizes the importance of having more relevant threads in training data. However, the improvement is slim. This indicates the high degree of similarity between the additional and the original data. Recall that the positive threads outside an EAC may just be the continuation of the positive threads inside.

Several conclusions can be drawn from this experiment and the experiment in Subsection 4.2. First, it is beneficial to consider threads. Second, it is important to identify the relevant threads. Third, it is perhaps not so beneficial to consider additional training threads from action clips (EACs or TEACs) where we already collect some positive training threads.

## 5 Summary

We have considered the task of recognizing human actions in TV material and discussed the problem of ignoring the discontinuity due to shot boundaries. Towards addressing the problem, we have introduced a large dataset with annotated occurrences of human actions in video shots. We used our dataset to study video threads and human actions, and our experiments confirmed the importance of considering and identifying relevant video threads in action recognition.

#### Acknowledgements

This work was supported by the EPSRC grant EP/I012001/1 and a Royal Society Wolfson Research Merit Award.

## References

1. Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2008)
2. Marszalek, M., Laptev, I., Schmid, C.: Actions in context. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2009)
3. Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., Serre, T.: HMDB: A large video database for human motion recognition. In: Proceedings of the International Conference on Computer Vision. (2011)
4. Patron-Perez, A., Marszalek, M., Reid, I., Zisserman, A.: Structured learning of human interactions in TV shows. IEEE Transactions on Pattern Analysis and Machine Intelligence **34** (2012) 2441–2453
5. Hoai, M., Lan, Z.Z., De la Torre, F.: Joint segmentation and classification of human actions in video. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2011)

6. Wang, H., Schmid, C.: Action recognition with improved trajectories. In: Proceedings of the International Conference on Computer Vision. (2013)
7. Hoai, M., Zisserman, A.: Talking heads: Detecting humans and recognizing their interactions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2014)
8. Cour, T., Jordan, C., Milsakaki, E., Taskar, B.: Movie/script: Alignment and parsing of video and text transcription. In: Proceedings of the European Conference on Computer Vision. (2008)
9. Zhai, Y., Shah, M.: Video scene segmentation using markov chain monte carlo. IEEE Transactions on Multimedia **8** (2006) 686–697
10. Yeung, M., Yeo, B.L., Liu, B.: Segmentation of video by clustering and graph analysis. Computer Vision and Image Understanding **71** (1998)
11. Kender, J., Yeo, B.L.: Video scene segmentation via continuous video coherence. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (1998)
12. Chasanis, V.T., Likas, A.C., Galatsanos, N.P.: Scene detection in videos using shot clustering and sequence alignment. IEEE Transactions on Multimedia **11** (2009)
13. B., L., N., O., N., M.: Dialogue sequence detection in movies. In: International Conference on Image and Video Retrieval. (2005)
14. Lehane, B., O'Connor, N.E., Smeaton, A.F., Lee, H.: A system for event-based film browsing. In: Proceedings of International Conference on Technologies for Interactive Digital Storytelling and Entertainment. (2006)
15. Pickup, L., Zisserman, A.: Automatic retrieval of visual continuity errors in movies. In: ACM International Conference on Image and Video Retrieval. (2009)
16. Tapaswi, M., Bauml, M., Stiefelhagen, R.: Storygraphs: Visualizing character interactions as a timeline. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2014)
17. Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: A local svm approach. In: Proceedings of the International Conference on Pattern Recognition. (2004)
18. Gorelick, L., Blank, M., Shechtman, E., Irani, M., Basri, R.: Actions as space-time shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence **29** (2007) 2247–2253
19. Reddy, K.K., Shah, M.: Recognizing 50 human action categories of web videos. Machine Vision and Applications **24** (2012) 971–981
20. Everingham, M., Sivic, J., Zisserman, A.: "hello! my name is ... Buffy" – automatic naming of characters in tv video. In: Proceedings of the British Machine Vision Conference. (2006)
21. Boreczky, J.S., Rowe, L.A.: Comparison of video shot boundary detection techniques. Journal of Electronic Imaging **5** (1996)
22. Lienhart, R.: Comparison of automatic shot boundary detection algorithms. In: SPIE. Volume 3656. (1998)
23. Lienhart, R.: Reliable transition detection in videos: A survey and practitioner's guide. Internation Journal of Image and Graphics **1** (2001) 469–486
24. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2005)
25. Lowe, D.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60** (2004) 91–110

26. Hoai, M., Zisserman, A.: Improving human action recognition using score distribution and ranking. In: Proceedings of the Asian Conference on Computer Vision. (2014)
27. Perronnin, F., Sánchez, J., Mensink, T.: Improving the fisher kernel for large-scale image classification. In: Proceedings of the European Conference on Computer Vision. (2010)
28. Suykens, J.A.K., Vandewalle, J.: Least squares support vector machine classifiers. Neural Processing Letters **9** (1999) 293–300
29. Saunders, C., Gammerman, A., Vovk, V.: Ridge regression learning algorithm in dual variables. In: Proceedings of the International Conference on Machine Learning. (1998)
30. Suykens, J.A.K., Gestel, T.V., Brabanter, J.D., DeMoor, B., Vandewalle, J.: Least Squares Support Vector Machines. World Scientific (2002)
31. Tommasi, T., Caputo, B.: The more you know, the less you learn: From knowledge transfer to one-shot learning of object categories. In: Proceedings of the British Machine Vision Conference. (2009)
32. Hoai, M.: Regularized max pooling for image categorization. In: Proceedings of the British Machine Vision Conference. (2014)
33. Cawley, G.C., Talbot, N.L.: Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. Neural Networks **17** (2004) 1467–1475
34. Vig, E., Dorr, M., Cox, D.: Space-variant descriptor sampling for action recognition based on saliency and eye movements. In: Proceedings of the European Conference on Computer Vision. (2012)
35. Marin-Jimenez, M.J., Yeguas, E., de la Blanca, N.P.: Exploring STIP-based models for recognizing human interactions in TV videos. PRL **34** (2013) 1819–1828
36. Jiang, Y.G., Dai, Q., Xue, X., Liu, W., Ngo, C.W.: Trajectory-based modeling of human actions with motion reference points. In: Proceedings of the European Conference on Computer Vision. (2012)
37. Mathe, S., Sminchisescu, C.: Dynamic eye movement datasets and learnt saliency models for visual action recognition. In: Proceedings of the European Conference on Computer Vision. (2012)
38. Gaidon, A., Harchaoui, Z., Schmid, C.: Recognizing activities with cluster-trees of tracklets. In: Proceedings of the British Machine Vision Conference. (2012)