Movie/Script: Alignment and Parsing of Video and Text Transcription

Timothee Cour, Chris Jordan, Eleni Miltsakaki, Ben Taskar

University of Pennsylvania, Philadelphia, PA 19104, USA,
{timothee,wjc,elenimi,taskar}@seas.upenn.edu

Abstract. Movies and TV are a rich source of diverse and complex video of people, objects, actions and locales "in the wild". Harvesting automatically labeled sequences of actions from video would enable creation of large-scale and highly-varied datasets. To enable such collection, we focus on the task of recovering scene structure in movies and TV series for object tracking and action retrieval. We present a weakly supervised algorithm that uses the screenplay and closed captions to parse a movie into a hierarchy of shots and scenes. Scene boundaries in the movie are aligned with screenplay scene labels and shots are reordered into a sequence of long continuous tracks or *threads* which allow for more accurate tracking of people, actions and objects. Scene segmentation, alignment, and shot threading are formulated as inference in a unified generative model and a novel hierarchical dynamic programming algorithm that can handle alignment and jump-limited reorderings in linear time is presented. We present quantitative and qualitative results on movie alignment and parsing, and use the recovered structure to improve character naming and retrieval of common actions in several episodes of popular TV series.

1 Introduction

Hand-labeling images of people and objects is a laborious task that is difficult to scale up. Several recent papers [1, 2] have successfully collected very large-scale, diverse datasets of faces "in the wild" using weakly supervised techniques. These datasets contain a wide variation in subject, pose, lighting, expression, and occlusions which is not matched by any previous hand-built dataset. Labeling and segmenting actions is perhaps an even more painstaking endeavor, where curated datasets are more limited. Automatically extracting large collections of actions is of paramount importance. In this paper, we argue that using movies and TV shows *precisely* aligned with easily obtainable screenplays can pave a way to building such large-scale collections. Figure 1 illustrates this goal, showing the top 6 retrieved video snippets for 2 actions (walk, turn) in TV series LOST using our system. The screenplay is parsed into a temporally aligned sequence of action frames (subject verb object), and matched to detected and named characters in the video sequence. Simultaneous work[3] explores similar goals in a more supervised fashion. In order to enable accurately localized action retrieval, we propose a much deeper analysis of the structure and syntax of both movies and transcriptions.

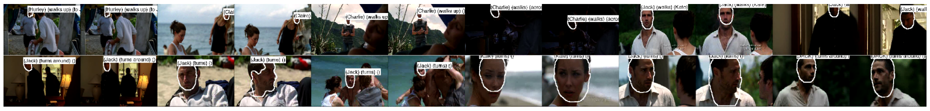


Fig. 1. Action retrieval using alignment between video and parsed screenplay. For each action verb (top: walk, bottom: turn), we display the top 6 retrieved video snippets in TV series LOST using our system. The screenplay and closed captions are parsed into a temporally aligned sequence of verb frames (subject-verb-object), and then matched to detected and named characters in the video sequence. The third retrieval, second row ("Jack turns") is counted as an error, since the face shows Boone instead of Jack. Additional results appear under www.seas.upenn.edu/˜timothee.

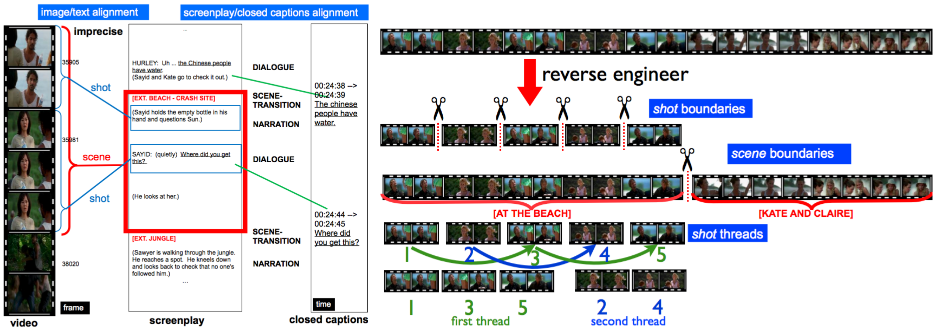
Movies, TV series, news clips, and nowadays plentiful amateur videos, are designed to effectively communicate events and stories. A visual narrative is conveyed from multiple camera angles that are carefully composed and interleaved to create seamless action. Strong coherence cues and continuity editing rules are (typically) used to orient the viewer, guide attention and help follow the action and geometry of the scene. Video shots, much like words in sentences and paragraphs, must fit together to minimize perceptual discontinuity across cuts and produce a meaningful scene. We attempt to uncover elements of the inherent structure of scenes and shots in video narratives. This uncovered structure can be used to analyze the content of the video for tracking objects across cuts, action retrieval, as well as enriching browsing and editing interfaces.

We present a framework for automatic parsing of a movie or video into a hierarchy of shots and scenes and recovery of the shot interconnection structure. Our algorithm makes use of both the input image sequence, closed captions and the screenplay of the movie. We assume a hierarchical organization of movies into shots, threads and scenes, where each scene is composed of a set of interlaced threads of shots with smooth transitions of camera viewpoint inside each thread. To model the scene structure, we propose a unified generative model for joint scene segmentation and shot threading. We show that inference in the model to recover latent structure amounts to finding a Hamiltonian path in the sequence of shots that maximizes the "head to tail" shot similarity along the path, given the scene boundaries. Finding the maximum weight Hamiltonian path (reducible to the Traveling Salesman Problem or TSP) is intractable in general, but in our case, limited memory constraints on the paths make it tractable. In fact we show how to jointly optimize scene boundaries and shot threading in *linear time* in the number of shots using a novel hierarchical dynamic program.

We introduce textual features to inform the model with scene segmentation, via temporal alignment with screenplay and closed captions, see figure 2. Such text data has been used for character naming [4, 5] and is widely available, which makes our approach applicable to a large number of movies and TV series. In order to retrieve temporally-aligned actions, we delve deeper into resolving textual ambiguities with pronoun resolution (determining whom or what 'he', 'she', 'it', etc. refer to in the screenplay) and extraction of verb frames. By detecting and naming characters, and resolving pronouns, we show promising results for more accurate action retrieval for several common verbs. We present quantitative and qualitative results for scene segmentation/alignment, shot segmentation/threading, tracking and character naming across shots and action retrieval in numerous episodes of popular TV series, and illustrate that shot reordering provides much improved character naming.

The main contributions of the paper are: 1) novel probabilistic model and inference procedure for shot threading and scene alignment driven by text, 2) extraction of verb frames and pronoun resolution from screenplay, and 3) retrieval of the corresponding actions informed by scene sctructure and character naming.

The paper is organized as follows. Section 2 proposes a hierarchical organization of movies into shots, threads and scenes. Sections 3 and 4 introduce a generative model for joint scene segmentation and shot threading, and a hierarchical dynamic program to solve it as a restricted TSP variant. Section 5 addresses the textual features used in our model. We report results in section 6 and conclude in section 7.



...

HURLEY: Uh ... the Chinese people have water.
(Sayid and Kate go to check it out.)

[EXT. BEACH - CRASH SITE]

(Sayid holds the empty bottle in his hand and questions Sun.)

SAYID: (quietly) Where did you get this?

(He looks at her.)

[EXT. JUNGLE]

(Sawyer is walking through the jungle. He reaches a spot. He kneels down and looks back to check that no one's followed him.)

...

00:24:38 --> 00:24:39
The chinese people have water.

00:24:44 --> 00:24:45
Where did you get this?

(a) (b)

Fig. 2. (a) Alignment between video, screenplay and closed captions; (b) Deconstruction pipeline.

2 Movie elements: shots, threads, scenes

Movies and TV series are organized in distinctive hierarchical and continuity structures consisting of elements such as scenes, threads and shots. Detecting and recovering these elements is needed for uninterrupted tracking of objects and people in a scene across multiple cameras, recovering geometric relationships of objects in a scene, intelligent video browsing, search and summarization.

Shot boundaries. The aim of shot segmentation is to segment the input frames into a sequence of shots (single unbroken video recordings) by detecting camera viewpoint discontinuities. A popular technique is to compute a set of localized color histograms for each image and use a histogram distance function to detect boundaries [6, 7].

Shot threads. Scenes are often modeled as a sequence of shots represented as letters: **ABABAB** represents a typical dialogue scene alternating between two camera points of view A and B. More complex patterns are usually observed and in practice, the clustering of the shots into letters (camera angles/poses) is not always a very well defined problem, as smooth transitions between shots occur. Nevertheless we assume in our case that each shot in a scene is either a novel camera viewpoint or is generated from (similar to) a previous shot in the scene. This makes weaker assumptions about the scene construction and doesn't require reasoning about the number of clusters. In the example above, the first A and B are novel viewpoints, and each subsequent A and B is generated by the previous A or B. Figure 5 shows a more complex structure.

Scene boundaries. A scene consists of a set of consecutive semantically related shots (coherence in action, location and group of actors is typical). The process of segmenting a video sequence into scenes has received some attention in the video analysis literature [7]. An MCMC based clustering framework is used in [8]. Hierarchical clustering on a shot connectivity graph is proposed in [9]. In [10], the authors detect scene boundaries as local minima of a backward shot coherence measure. As opposed to shot boundaries, which correspond to strong visual discontinuity in consecutive frames, scene boundaries are not detectable from purely local cues: the entire sequence of preceding and following shots must be considered. For example, **ABCBABDEFEABD** shot sequence is one scene, while **ABCBAB DEFEDEF** can be two.

3 A (simple) generative model of movies

To capture the hierarchical and continuity structure, we propose a simple generative model, where scenes are constructed independently of other scenes, while shots within a scene are produced via an interleaved Markov (first order) structure.

We begin with some notation to define our model, assuming the video sequence has already be segmented into shots:

- s_i: i^{th} shot (interval of frames), with $i \in [1, n]$
- b_j: j^{th} scene boundary (index of its *last* shot), with $j \leq m$; $1 \leq b_1 < ... < b_m = n$
- $p_j[i]$: parent generating shot i in scene j (could be NULL), with $j \leq m, i \leq n$.

We assume the shots in a video sequence are generated as follows: first generate the sequence of scene boundaries (b_j), then generate for each scene j a dependency structure p_j defining a Markov chain on shots, and finally generate each shot i given its parent $p_j[i]$. The model is conditioned upon m and n, assumed to be known in advance. This can be represented using the generative model in figure 3. For the **scene boundary model** $P(b)$, we investigate both a uniform model and an improved model, where scene boundaries are informed by the screenplay (see section 5). The **shot threading model** $P(p|b)$ is uniformly distributed over valid Markov chains (shot orderings) on each scene. The **shot appearance model** $P(s_i|s_{p_j[i]})$ is treated next (we set it to uniform for the root of scene j where $p_j[i] =$ NULL). This model encourages (1) smooth shot transitions within a scene and (2) scene breaks between shots with low similarity, since the model doesn't penalize transitions across scenes.

Shot appearance model ($\boldsymbol{P(s_{i'}|s_i)}$). In order to obtain smooth transitions and allow tracking of objects throughout reordered shots, we require that $P(s_{i'}|s_i)$ depends

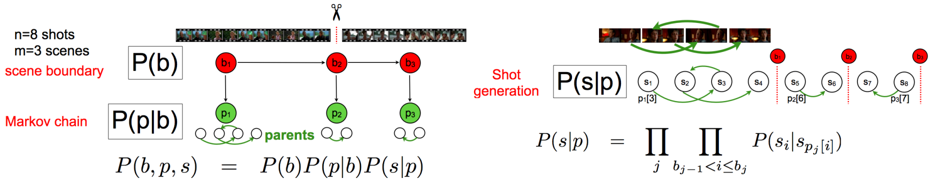


Fig. 3. Graphical model for joint scene segmentation and shot reordering, see text for details.

on the similarity between the *last* frame of shot s_i ($I = s_i^{\text{last}}$) and the *first* frame of shot $s_{i'}$ ($I' = s_{i'}^{\text{first}}$). Treating each shot as a word in a finite set, we parameterize the **shot similarity term** as $P(s_{i'}|s_i) = \exp(-d_{\text{shot}}(s_i, s_{i'}))/\sum_{i''} \exp(-d_{\text{shot}}(s_i, s_{i''}))$ where $d_{\text{shot}}(s_i, s_{i'}) = d_{\text{frame}}(I, I')$ is the chi-squared distance in color histogram between frames I, I'. Note, $d_{\text{shot}}(s_i, s_{i'})$ is *not symmetric*, even though $d_{\text{frame}}(I, I')$ is.

4 Inference in the model

In this section we attempt to solve the MAP problem in figure 3. Let us first consider the simplified case without scene transitions (when $m = 1$). In this case, maximizing the log becomes:

$$\max_{p:\text{Markov Chain}} \sum_i W_{i,p[i]} = \max_{\pi \in \mathcal{P}_{[1,n]}} \sum_t W_{\pi_{t-1},\pi_t} \tag{1}$$

where $W_{ii'} = \log P(s_{i'}|s_i)$ and $\pi \in \mathcal{P}_{[1,n]}$ denotes a permutation of $[1, n]$ defined recursively from the parent variable p as follows: $p[\pi_t] = \pi_{t-1}$, with π_1 indicating the root. This amounts to finding a maximum weight Hamiltonian Path or **Traveling Salesman Problem** (TSP), with π_t indicating which shot is visited at time t on a virtual tour. TSPs are *intractable in general*, so we make one additional assumption restricting the set of feasible permutations.

4.1 Memory-limited TSPs

Given an integer $k > 0$ (memory width), and an initial ordering of shots (or cities by analogy to TSP) $1, ..., n$, we introduce the following limited memory constraint on our hamiltonian path $\pi = (\pi_t)$:

$$\mathcal{P}^k_{[1,n]} = \{\pi \in \mathcal{P}_{[1,n]} : \forall (i, i') i' \geq i + k \Rightarrow \pi_{i'} > \pi_i\} \tag{2}$$

This is illustrated in figure 4 for $k = 2$ ($k = 1$ means π is the identity, and $k = n$ is fully unconstrained). There are two important consequences: (1) the MAP becomes tractable (*linear complexity in* n), and (2) the problem becomes sparse, *i.e.*, we can restrict W.L.O.G. W to be sparse (banded):

$$\pi_t \in [t - (k-1), t + (k-1)] \tag{3}$$

$$W_{ii'} = -\infty \text{ except for } i - (2k-3) \leq i' \leq i + 2k - 1 \tag{4}$$

The first line comes from the pigeonhole principle, and the second one uses the first line: $-(2k-3) \leq \pi_{t+1} - \pi_t \leq 2k-1$. Note, this constraint is natural in a video sequence, as video editing takes into account the limited memory span of humans consisting of a few consecutive shots.

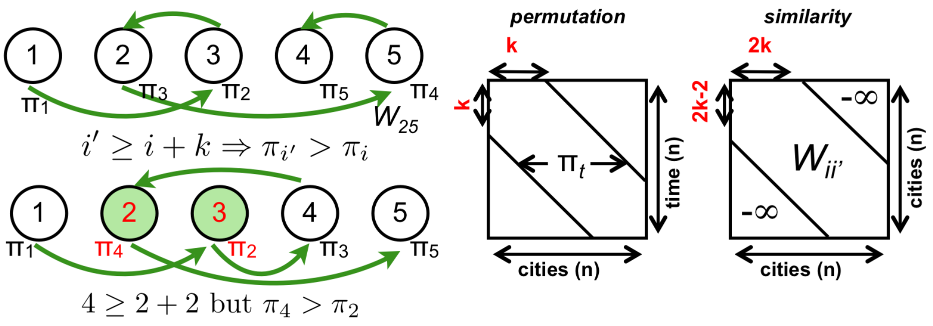


Fig. 4. Top: a feasible solution for the restricted TSP with $k = 2$. Bottom: an infeasible solution, violating the precedence constraint (shaded cities). Middle: the constraint limits the range of the permutation: $\pi_t \in [t-(k-1), t+(k-1)]$. Right: the constraint implies a banded structure on the similarity matrix $W = (W_{ii'})$: $i-(2k-3) \leq i' \leq i+2k-1$.

4.2 Dynamic Programming solution without scene breaks $(P(p, s))$

The solution to the simplified problem without scene breaks (1) under constraint (2) has been addressed in [11] (it dealt with a hamiltonian *cycle* with $\pi_1(1) = 1$, but this is easily adaptable to our case). We summarize the main points below. Let $C_t(S, i')$ be the optimal cost of the paths $\pi \in \mathcal{P}^k_{[1,n]}$ satisfying $\pi_t = i'$ and $\{\pi_1, ..., \pi_{t-1}\} = S$ (set of cities visited before time t). The dynamic programming solution uses the relation:

$$C_t(S, i') = \min_{i \in S} C_{t-1}(S - \{i\}, i) + W_{ii'} \qquad (5)$$

Because of the precedence constraint, the pair (S, i') can take at most $(k+1)2^{k-2}$ possible values at any given time t (instead of $\binom{n-1}{t-1}n$ without the constraint). The idea is to construct a directed weighted graph G^k_n with n layers of nodes, one layer per position in the path, with paths in the graph joining layer 1 to layer n corresponding to feasible hamiltonian paths, and shortest paths joining layer 1 to n corresponding to optimal hamiltonian paths. Since there are at most k incoming edges per node (corresponding to valid transitions $\pi_{t-1} \rightarrow \pi_t$), the total complexity of the dynamic program is $O(k(k+1)2^{k-2} \cdot n)$, exponential in k (fixed) but *linear in* n, see [11] for details.

4.3 Dynamic Programming solution with scene breaks $(\boldsymbol{P(b,p,s)})$

The general problem can be rewritten as:

$$\max_b \sum_j \max_{\pi \in \mathcal{P}^k_{(b_{j-1},b_j]}} \sum_t W_{\pi_{t-1},\pi_t} \tag{6}$$

Naive solution. One can solve (6) as follows: for each interval $\mathcal{I} \subset [1,n]$, pre-compute the optimal path $\pi^*_{\mathcal{I}} \in \mathcal{P}^k_{\mathcal{I}}$ using 4, and then use a straightforward dynamic programming algorithm to compute the optimal concatenation of m such paths to form the optimal solution. Letting $f(k) = k(k+1)2^{k-2}$, the complexity of this algorithm is $O(\sum_{1\leq i\leq i'\leq n} f(k)\cdot(i'-i+1)) = O(f(k)n(n+1)(n+2)/6)$ for the precomputation and $O(mn(n+1)/2)$ for the dynamic program, which totals to $O(f(k)n^3/6)$. The next paragraph introduces our joint dynamic programming over scene segmentation and shot threading, which reduces computational complexity by a factor n (number of shots).

Joint dynamic program over scene breaks and shot threading. We exploit the presence of overlapping subproblems. We construct a *single* tour π, walking over the joint space of shots and scene labels. Our approach is based on the (categorical) **product graph** $G^k_n \times C_m$ where G^k_n is the graph from 4.2 and C_m is the chain graph of order m.

A node $(u,j) \in G^k_n \times C_m$ represents the node $u \in G^k_n$ in the j^{th} scene. Given two *connected* nodes $u = (S,i,t)$ and $u' = (S',i',t+1)$ in G^k_n, there are two types of connections in the product graph. The first connections correspond to shots i, i' both being in the j^{th} scene:

$$(u,j) \rightarrow (u',j), \text{ with weight } W_{ii'} \tag{7}$$

The second connections correspond to a scene transition:

$$(u,j) \rightarrow (u',j+1), \text{ with weight } 0, \tag{8}$$

and only happen when $u = (S,i,t)$ satisfies $\max(i,\max(S)) = t$, to make sure the tour decomposes into a tour of each scene (we can switch to the next scene when the set of shots visited up to time t is exactly $\{1,...,t\}$).

The solution to (6) similarly uses a dynamic program to find the shortest path in $G^k_n \times C_m$ (and backtracking to recover the $\arg\max$). Since there are m times as many nodes in the graph as in G^k_n and at most twice as many incoming connections per node (nodes from the previous scene or from the same scene), the total complexity is: $O(2k(k+1)2^{k-2}mn) = O(2f(k)mn)$.

Comparison. We manually labeled shot and scene breaks for a number of movies and TV series and found that a typical scene contains on average about 11 shots, *i.e.*$m \approx n/11$. So the reduction in complexity between the naive algorithm and our joint dynamic program is: $O(\frac{f(k)n^3/6}{2f(k)mn}) = O(n^2/(12m)) \approx n$, which is a huge gain, especially given typical values of $n = 600$. The resulting complexity is linear in n and m and in practice takes about 1 minute as opposed to 11 hours for an entire episode, given pre-computed shot similarity.

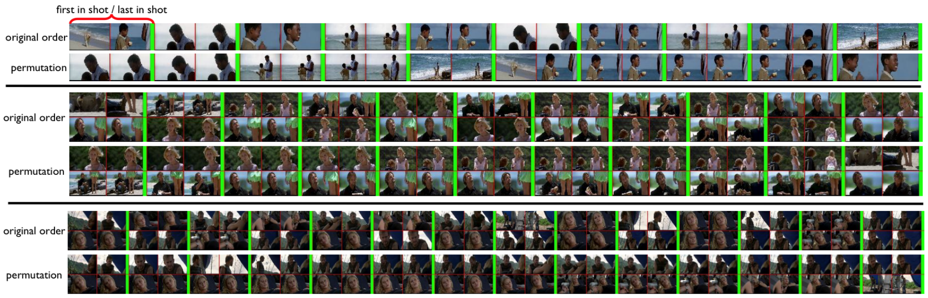


Fig. 5. Shot reordering to recover continuity in 3 scenes of LOST.

5 Scene segmentation via coarse image to text alignment $(P(b))$

We now assume we have some text data corresponding to the movie sequence, and we focus on simultaneously segmenting/threading the video into scenes and aligning the text with the video. The extra text media removes a lot of ambiguity for the scene segmentation and, combined with our model, leads to improved scene segmentation results as we shall see in section (6).

5.1 Text data: screenplay and closed captions

We use two sources of text for our segmentation-alignment problem: the screenplay, which narrates the actions and provides a transcript of the dialogues, and the closed captions, which provide time-stamped dialogues, as in figure 2(a). Both sources are essential since the screenplay reveals speaker identity, dialogues and scene transitions but no time-stamps, and closed captions reveal dialogues with time-stamps but nothing else. The screenplay and the closed captions are readily available for a majority of movies and TV series produced in the US. A similar approach was used in [5] to align faces with character names, with 2 differences: 1) they used the screenplay to reveal the speaker identity as opposed to scene transitions, and 2) subtitles were used instead of closed captions. Subtitles are encoded as bitmaps, thus require additional steps of OCR and spell-checking to convert them to text[5], whereas closed captions are encoded as ASCII text in DVDs, making our approach simpler and more reliable, requiring a simple modification of mplayer (http://www.mplayerhq.hu/).

5.2 Screenplay/closed captions alignment

The alignment between the screenplay and the closed captions is non-trivial since the closed captions only contain the dialogues (without speaker) mentioned in the screenplay, often with wide discrepancies between both versions. We extend the dynamic time warping[12] approach in a straightforward way to time-stamp each element of the screenplay (as opposed to just the dialogues as in [5]). The screenplay is first parsed into a sequence of elements (either NARRATION, DIALOGUE, or SCENE-TRANSITION) using a simple grammar, and the dynamic programming alignment of the words in the screenplay and the closed captions provides a time interval $[T^{\text{start}}(i), T^{\text{end}}(i)]$ for each DIALOGUE element E_i. A NARRATION or SCENE-TRANSITION element E_j enclosed between two DIALOGUE elements E_{i_1}, E_{i_2} is assigned the following conservative time interval: $[T^{\text{start}}(i_1), T^{\text{end}}(i_2)]$.

5.3 Scene segmentation via alignment

We determine the scene boundary term $P(b)$ from section 3 by aligning each SCENE-TRANSITION element mentioned in the screenplay to a scene start. $P(b)$ is uniform among the set of b satisfying the temporal alignment constraints:

$$1 \leq b_1 < ... < b_m = n \tag{9}$$

$$t^{\text{start}}(j) \leq b_{j-1} + 1 \leq t^{\text{end}}(j) \tag{10}$$

where $[t^{\text{start}}(j), [t^{\text{end}}(j)]$ is the time interval of the j^{th} SCENE-TRANSITION element, converted into frame numbers, then to shot indexes.

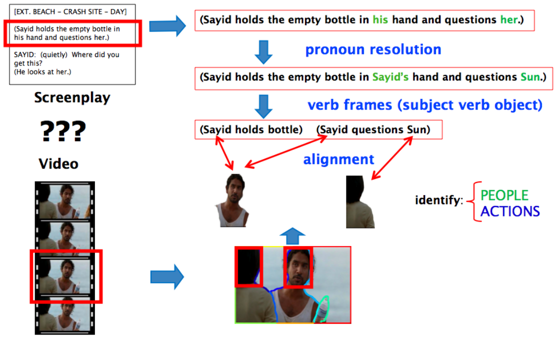
Additional alignment constraints. Close inspection of a large number of screenplays collected for movies and TV series revealed a fairly regular vocabulary used to describe shots and scenes. One such example is FADE IN and FADE OUT corresponding to a transition between a black shot (where each frame is totally black) and a normal shot, and vice versa. Such black shots are easy to detect, leading to additional constraints in the alignment problem, and a performance boost.

5.4 Pronoun resolution and verb frames

Alignment of the screenplay to dialog in closed captions and scene boundaries in the video helps to narrow down the scope of reference for other parts of the screenplay that are interspersed – the narration or scene descriptions, which contain mentions of actions and objects on the screen. In addition to temporal scope uncertainty for these descriptions, there is also ambiguity with respect to the subject of the verb, since personal pronouns (he, she) are commonly used. In fact, our analysis of common screenplays reveals there are more pronouns than occurences of character names in the narrations, and so resolving those pronouns is an important task. We employed a simple, deterministic scheme for pronoun resolution that uses a standard probabilistic context-free parser to analyze sentences and determine verb frames (subject-verb-object) and then scans the sentence for possible antecedents of each pronoun that agree in number and gender, see figure 6. The details of the algorithm are given in *supplemental materials*. Here is an example output of our implementation on a sentence extracted from screenplay narration (pronoun resolution shown in parenthesis): *On the side, Sun watches them. Jin reaches out and touches Sun 's chin, his (Jin's) thumb brushes her (Sun's) lips. She (Sun) looks at him (Jin) and pulls away a little. He (Jin) puts his (Jin's) hand down.*

Output verb frames: *(Sun - watches - something) (Jin - reaches out -) (Jin - touches - chin) (Sun - looks - at Jin) . (Sun - pulls away -) (Jin - puts down - hand).*

We report pronoun resolution accuracy on screenplay narrations of 3 different TV series (about half a screenplay for each), see table 1.


[EXT. BEACH – CRASH SITE – DAY]
(Sayid holds the empty bottle in his hand and questions her.)
SAYID: (quietly) Where did you get this?
(He looks at her.)
Screenplay
???
Video
(Sayid holds the empty bottle in his hand and questions her.)
pronoun resolution
(Sayid holds the empty bottle in Sayid's hand and questions Sun.)
verb frames (subject verb object)
(Sayid holds bottle) (Sayid questions Sun)
alignment
identify: PEOPLE
ACTIONS

total verbs	25,000
distinct verbs	1,000
looks (most common)	2,000
turns	1,100
walks	800
takes	550
climbs	40
kisses	40
total dialogue lines	16,000
distinct speaker names	190
Jack (most common)	2,100

Fig. 6. Left: pronoun resolution and verb frames obtained from the parsed screenplay narrations. Right: statistics collected from 24 parsed screenplays (1 season of LOST).

TV series screenplay	pronoun resolution accuracy	# pronouns	# sentences
LOST	75%	93	100
CSI	76 %	118	250
ALIAS	78%	178	250

Table 1. Pronoun resolution accuracy on screenplay narrations of 3 different TV series.

6 Results

We experimented with our framework on a significant amount of data, composed of TV series (19 episodes from one season of LOST, several episodes of CSI), one feature length movie “The Fifth Element”, and one animation movie “Aladdin”, representing about 20 hours of video at DVD resolution. We report results on scene segmentation/alignment, character naming and tracking, as well as retrieval of query action verbs.

Shot segmentation. We obtain 97% F-score (harmonic mean of precision and recall) for shot segmentation, using standard color histogram based methods.
Scene segmentation and alignment. We hand labeled scene boundaries in one episode of LOST and one episode of CSI based on manual alignment of the frames with the screenplay. The accuracy for predicting the scene label of each shot was 97% for LOST and 91% for CSI. The F-score for scene boundary detection was 86% for LOST and 75% for CSI, see figure 7. We used $k = 9$ for the memory width, a value similar to the buffer size used in [10] for computing shot coherence. We also analyzed the effect on performance of the memory width k, and report results with and without alignment to screenplay in table 2. In comparison, we obtained an F-score of 43% for scene boundary detection using a model based on backward shot coherence [10] uninformed by screenplay, but optimized over buffer size and non-maximum suppression window size.
Scene content analysis. We manually labeled the scene layout in the same episodes of LOST and CSI, providing for each shot in a scene its generating shot (including the special case when this is a new viewpoint). We obtain a precision/recall of 75% for predicting the generating parent shot. See figure 5 for a sample of the results on 3 scenes.

Note, to obtain longer tracks in figure 5, we recursively applied the memory limited TSP until convergence (typically a few iterations).

Character identification on reordered shots. We illustrate a simple speaker identification based on screenplay alignment and shot threading, see figure 8. We use a Viola-Jones[13] based face detector and tracking with normalized cross-correlation to obtain face tracks in each shot. We build a Hiddden Markov Model (HMM) with states corresponding to assignments of face tracks to character names. The face tracks are ordered according to the shot threading permutation, and as a result there are much fewer changes of character name along this ordering. Following [14], we detect on-screen speakers as follows: 1) locate mouth for each face track using a mouth detector based on Viola-Jones, 2) compute a mouth motion score based on the normalized cross correlation between consecutive windows of the mouth track, averaged over temporal segments corresponding to speech portions of the screenplay. Finally we label the face tracks using Viterbi decoding for the Maximum a Posteriori (MAP) assignment (see website for more details). We computed groundtruth face names for one episode of LOST and compared our method against the following baseline that does not use shot reordering: each unlabeled face track (without a detected speaking character on screen) is labeled using the closest labeled face track in feature space (position of face track and color histogram). The accuracy over an episode of LOST is 76% for mainly dialogue scenes and 66% for the entire episode, as evaluated against groundtruth. The baseline model based using nearest neighbor performs at resp. 43% and 39%.

Retrieval of actions in videos. We consider a query-by-action verb retrieval task for 15 query verbs across 10 episodes of LOST, see figure 9. The screenplay is parsed into verb frames (subject-verb-object) with pronoun resolution, as discussed earlier. Each verb frame is assigned a temporal interval based on time-stamped intervening dialogues and tightened with nearby shot/scene boundaries. Queries are further refined to match the subject of the verb frame with a named character face. We report retrieval results as follows: for each of the following action verbs, we measure the number of times (out of 10) the retrieved video snippet correctly shows the actor on screen performing the action (we penalize for wrong naming): close eyes (9/10), grab (9/10), kiss (8/10), kneel (9/10), open (9/10), stand (9/10), cry (9/10), open door (10/10), phone (10/10), point (10/10), shout (7/10), sit (10/10), sleep (8/10), smile (9/10), take breath (9/10). The average is 90/100. Two additional queries are shown in figure 1 along with the detected and identified characters. We created a large dataset of retrieved action sequences combined with character naming for improved temporal and spatial localization, see www.seas.upenn.edu/˜timothee for results and matlab code.

7 Conclusion

In this work we have addressed basic elements of movie structure: hierarchy of scenes and shots and continuity of shot threads. We believe that this structure can be useful for many intelligent movie manipulation tasks, such as semantic retrieval and indexing,

$P(b)$	$k = 1$	$k = 2$	$k = 3$	$k = 9$	$k = 12$
aligned	73/90	77/91	82/96	86/97	88/97
uniform	25/0	45/14	55/0	52/1	-/-
total time (s)	< 0.1	< 0.1	0.1	5	68

Table 2. % F-score (first number) for scene boundary detection and % accuracy (second number) for predicting scene label of shots (on 1 episode of LOST) as a function of the memory width k used in the TSP, and the prior $P(b)$. The case $k = 1$ corresponds to no reordering at all. Line 1: $P(b)$ informed by screenplay; line 2: $P(b)$ uniform; line 3: total computation time.

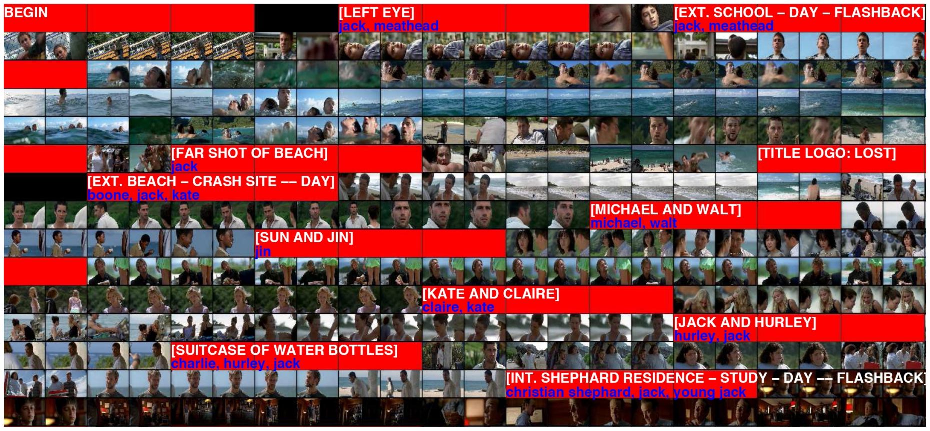


Fig. 7. Movie at a glance: scene segmentation-alignment and shot reordering for an episode of LOST (only a portion shown for readability). Scene boundaries are in red, together with the set of characters appearing in each scene, in blue.

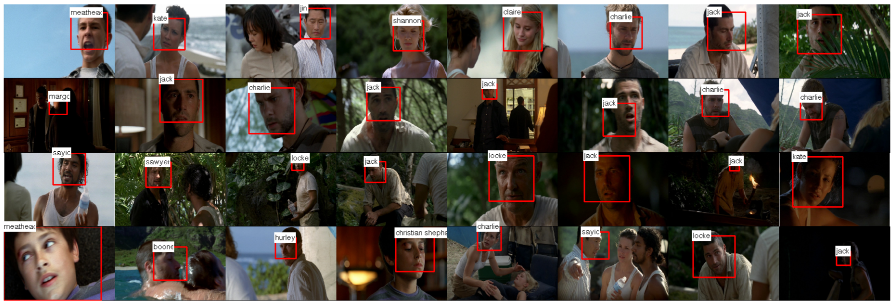


Fig. 8. Character naming using screenplay alignment and shot threading. Top 3 rows: correctly named faces; bottom row: incorrectly named faces. We detect face tracks in each shot and reorder them according to the shot threading permutation. Some face tracks are assigned a name prior based on the alignment between dialogues and mouth motion. We compute a joint assignment of names to face tracks using an HMM on the reordered face tracks.

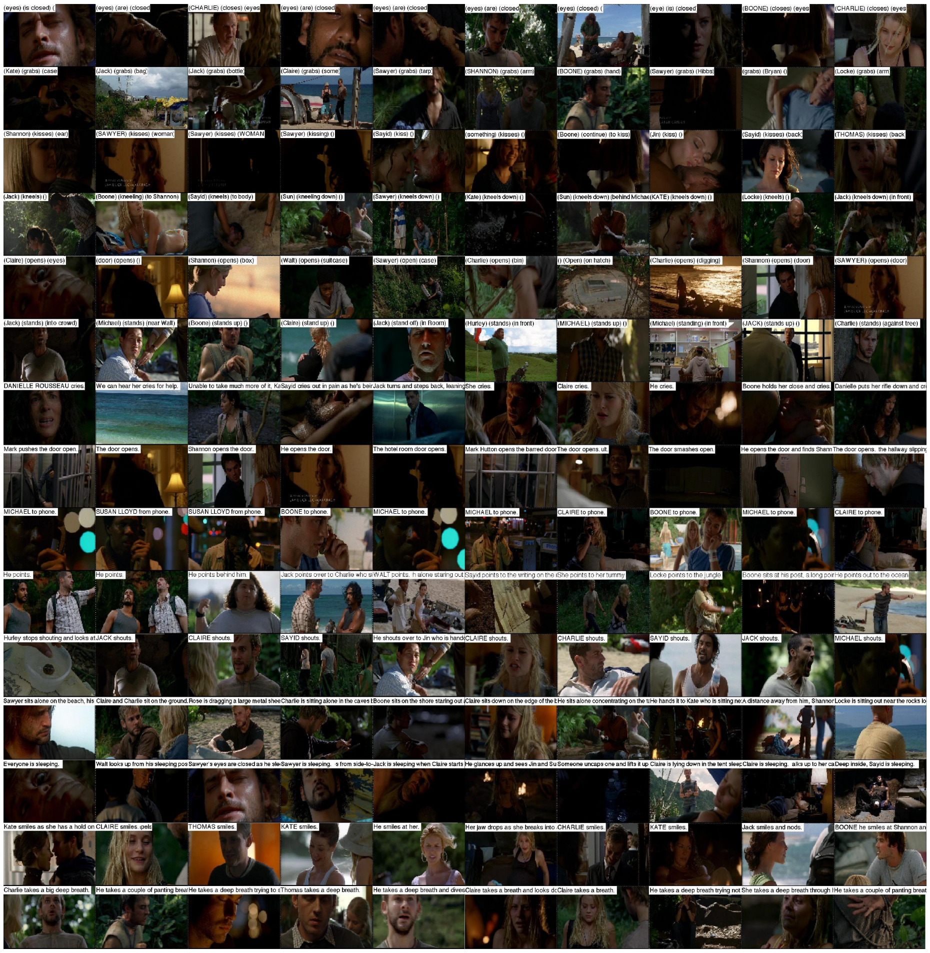

(eyes) (is closed) (
(eyes) (are) (closed
(CHARLIE) (closes) (eyes
(eyes) (are) (closed
(eyes) (are) (closed
(eyes) (are) (closed
(eyes) (closed) (
(eye) (is) (closed
(BOONE) (closes) (eyes
(CHARLIE) (closes) (eyes
(Kate) (grabs) (case
(Jack) (grabs) (bag)
(Jack) (grabs) (bottle)
(Claire) (grabs) (some)
(Sawyer) (grabs) (tarp)
(SHANNON) (grabs) (arm)
(BOONE) (grabs) (hand)
(Sawyer) (grabs) (Hibbs)
(grabs) (Bryan) ()
(Locke) (grabs) (arm
(Shannon) (kisses) (ear)
(SAWYER) (kisses) (woman)
(Sawyer) (kisses) (WOMAN
(Sawyer) (kissing) ()
(Sayid) (kiss) ()
(something) (kisses) ()
(Boone) (continue) (to kiss)
(Jin) (kiss) ()
(Sayid) (kisses) (back)
(THOMAS) (kisses) (back
(Jack) (kneels) ()
(Boone) (kneeling) (to Shannon)
(Sayid) (kneels) (to body)
(Sun) (kneeling down) ()
(Sawyer) (kneels down) ()
(Kate) (kneels down) ()
(Sun) (kneels down) (behind Michae
(KATE) (kneels down) ()
(Locke) (kneels) ()
(Jack) (kneels down) (in front)
(Claire) (opens) (eyes)
(door) (opens) ()
(Shannon) (opens) (box)
(Walt) (opens) (suitcase)
(Sawyer) (open) (case)
(Charlie) (opens) (bin)
() (Open) (on hatch)
(Charlie) (opens) (digging)
(Shannon) (opens) (door)
(SAWYER) (opens) (door)
(Jack) (stands) (into crowd)
(Michael) (stands) (near Walt)
(Boone) (stands up) ()
(Claire) (stand up) ()
(Jack) (stand off) (in Room)
(Hurley) (stands) (in front)
(MICHAEL) (stands up) ()
(Michael) (standing) (in front)
(JACK) (stands up) ()
(Charlie) (stands) (against tree)
DANIELLE ROUSSEAU cries.
We can hear her cries for help.
Unable to take much more of it, Ka
Sayid cries out in pain as he's beir
Jack turns and steps back, leaning
She cries.
Claire cries.
He cries.
Boone holds her close and cries.
Danielle puts her rifle down and cr
Mark pushes the door open.
The door opens.
Shannon opens the door.
He opens the door.
The hotel room door opens.
Mark Hutton opens the barred door
The door opens. ult.
The door smashes open.
He opens the door and finds Shann
The door opens. the hallway slipping
MICHAEL to phone.
SUSAN LLOYD from phone.
SUSAN LLOYD from phone.
BOONE to phone.
MICHAEL to phone.
MICHAEL to phone.
CLAIRE to phone.
BOONE to phone.
MICHAEL to phone.
CLAIRE to phone.
He points.
He points.
He points behind him.
Jack points over to Charlie who si
WALT points. :h alone staring out
Sayid points to the writing on the i
She points to her tummy
Locke points to the jungle
Boone sits at his post, a long poir
He points out to the ocean
Hurley stops shouting and looks at
JACK shouts.
CLAIRE shouts.
SAYID shouts.
He shouts over to Jin who is hand
CLAIRE shouts.
CHARLIE shouts.
SAYID shouts.
JACK shouts.
MICHAEL shouts.
Sawyer sits alone on the beach, his
Claire and Charlie sit on the ground.
Rose is dragging a large metal shee
Charlie is sitting alone in the caves t
Boone sits on the shore staring out a
Claire sits down on the edge of the b
He sits alone concentrating on the ta
He hands it to Kate who is sitting ne:
A distance away from him, Shannor
Locke is sitting out near the rocks lo
Everyone is sleeping.
Walt looks up from his sleeping pos
Sawyer's eyes are closed as he sle
Sawyer is sleeping. :s from side-to-
Jack is sleeping when Claire starts
He glances up and sees Jin and Su
Someone uncaps one and lifts it up
Claire is lying down in the tent sleep
Claire is sleeping. alks up to her ca
Deep inside, Sayid is sleeping.
Kate smiles as she has a hold on
CLAIRE smiles. pels
THOMAS smiles.
KATE smiles.
He smiles at her.
Her jaw drops as she breaks into
CHARLIE smiles.
KATE smiles.
Jack smiles and nods.
BOONE he smiles at Shannon an
Charlie takes a big deep breath.
He takes a couple of panting breat
He takes a deep breath trying to s
Thomas takes a deep breath.
He takes a deep breath and dives
Claire takes a breath and looks dc
Claire takes a breath.
He takes a deep breath trying not
She takes a deep breath through I
He takes a couple of panting breat

Fig. 9. Top 10 retrieved video snippets for 15 query action verbs: close eyes, grab, kiss, kneel, open, stand, cry, open door, phone, point, shout, sit, sleep, smile, take breath. Please zoom in to see screenplay annotation (and its parsing into verb frames for the first 6 verbs).

browsing by character or object, re-editing and many more. We plan to extend our work to provide more fine-grained alignment of movies and screenplay, using coarse scene geometry, gaze and pose estimation.

References

1. Huang, G., Jain, V., Learned-Miller, E.: Unsupervised joint alignment of complex images. In: International Conference on Computer Vision. (2007) 1–8
2. Ramanan, D., Baker, S., Kakade, S.: Leveraging archival video for building face datasets. In: International Conference on Computer Vision. (2007) 1–8
3. Laptev, I., Marszałek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: IEEE Conference on Computer Vision and Pattern Recognition. (2008)
4. Sivic, J., Everingham, M., Zisserman, A.: Person spotting: video shot retrieval for face sets. In: International Conference on Image and Video Retrieval (CIVR 2005), Singapore. (2005)
5. 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)
6. Lienhart, R.: Reliable transition detection in videos: A survey and practitioner's guide. Int. Journal of Image and Graphics (2001)
7. Ngo, C.W., Pong, T.C., Zhang, H.J.: Recent advances in content-based video analysis. International Journal of Image and Graphics **1** (2001) 445–468
8. Zhai, Y., Shah, M.: Video scene segmentation using markov chain monte carlo. IEEE Transactions on Multimedia **8** (2006) 686–697
9. Yeung, M., Yeo, B.L., Liu, B.: Segmentation of video by clustering and graph analysis. Comp. Vision Image Understanding (1998)
10. Kender, J., Yeo, B.: Video scene segmentation via continuous video coherence. In: IEEE Conference on Computer Vision and Pattern Recognition. (1998)
11. Balas, E., Simonetti, N.: Linear time dynamic programming algorithms for new classes of restricted tsps: A computational study. INFORMS Journal on Computing **13** (2001) 56–75
12. Myers, C.S., Rabiner, L.R.: A comparative study of several dynamic time-warping algorithms for connected word recognition. In: The Bell System Technical Journal. (1981)
13. Viola, P., Jones, M.: Robust real-time face detection. International Journal of Computer Vision **57** (2004) 137–154
14. Everingham, M., Sivic, J., Zisserman, A.: Hello! my name is buffy: Automatic naming of characters in tv video. In: BMVC. (2006) III:899