StoryGraphs: Visualizing Character Interactions as a Timeline

Makarand Tapaswi Martin Bäuml Rainer Stiefelhagen
Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany
{makarand.tapaswi, baeuml, rainer.stiefelhagen}@kit.edu

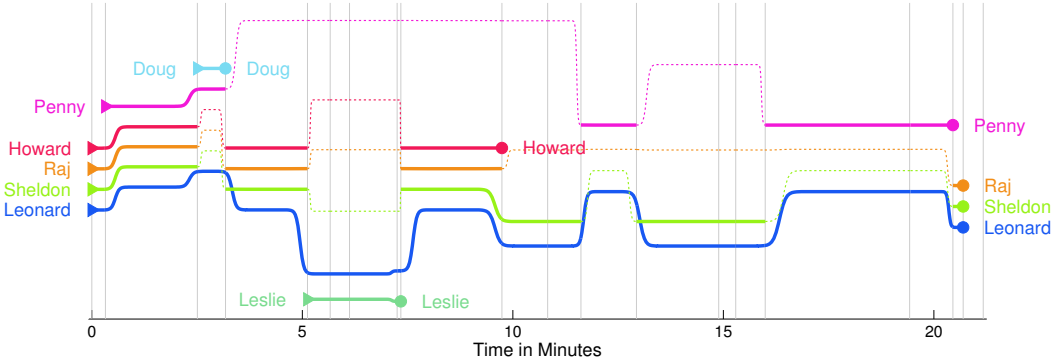

Figure 1: StoryGraph from The Big Bang Theory for season 01 episode 03 (BBT-3). This figure is best viewed in color.

Abstract

We present a novel way to automatically summarize and represent the storyline of a TV episode by visualizing character interactions as a chart. We also propose a scene detection method that lends itself well to generate oversegmented scenes which is used to partition the video. The positioning of character lines in the chart is formulated as an optimization problem which trades between the aesthetics and functionality of the chart. Using automatic person identification, we present StoryGraphs for 3 diverse TV series encompassing a total of 22 episodes. We define quantitative criteria to evaluate StoryGraphs and also compare them against episode summaries to evaluate their ability to provide an overview of the episode.

1. Introduction

Inspired by http://xkcd.com/657, we present a technique to automatically visualize the interactions between characters within a TV episode. All stories are essentially based on interactions between the characters [8, 16, 20], and therefore key parts of the storyline can be conveyed by graphically visualizing people-people interactions.

We call these charts *StoryGraphs* and an example is shown in Fig. 1. Vertical gray lines represent scene boundaries and each horizontal line represents a unique character from the story. A solid line indicates that the character is on-screen during that scene, while a dotted line means he/she is off-screen. The start (▶) and end of the line (●) indicate the first and last appearance of the character in that episode. Finally, the vertical spacing between the lines indicates which characters appear together, providing an overview of character interactions, and flow among them.

For example, we see in Fig. 1 that the first 2 minutes of the episode essentially involve all 5 primary characters from episode 3 of The Big Bang Theory. This is followed by a short scene between Doug, Penny and Leonard. At minute 5–7, we see an interaction between Leonard and Leslie which is the only appearance of Leslie in this episode. Minute 12 and 17–20 are scenes involving Leonard and Penny, while minute 13–16 involve Leonard and Sheldon.

Since StoryGraphs are based on the on-screen presence of characters, they are a direct application of the work in the vision community related to person identification in TV series [2, 5, 9]. StoryGraphs can be used in applications such as *smart browsing* and *video retrieval*. For example, the video player's seek-bar can be augmented by the chart, thus providing a sneak-preview of the video (while not revealing major spoilers) along with fast access to scenes involving interactions between the viewers favorite characters.

StoryGraphs also facilitate the search for specific situations in the story. For example, answering a query such as "Leonard meets Leslie at the lab" is easily achieved by manually looking at Fig. 1 and jumping to minutes 5–7 of the video. Harder queries such as "Leonard asks Penny out on a date" are localized to minute 12 or 17–20 of the video thus massively reducing search time.

Apart from the above applications in the context of multimedia data, the StoryGraphs themselves are an approach to visualize a set of people-people interactions. In particular, they can be used to generate similar charts for meeting room recordings to analyze the nature of the discussion (lecture / group discussion / etc.), list the primary speakers and collaborations; or to obtain a quick overview for crisis control room analysis.

Contributions The contributions of this paper are:

- We present an automatic approach to visualize character interactions within videos as timeline charts.
- In order to segment the video based on the content we propose a scene detection algorithm. Our method has the property to easily produce oversegmented scenes which are used in the StoryGraphs.
- We formulate the positioning of character-lines in the StoryGraph as an optimization problem trading off different aspects of the generated chart.

We evaluate the StoryGraphs on three TV series – The Big Bang Theory, Buffy the Vampire Slayer and Game of Thrones – and show promising results.

The rest of the paper is organized as follows. As a precursor to the automatic generation of StoryGraphs, we first need to track and identify the people (Sec. 3.1) and group them within blocks of time. We design these blocks to lie at scene boundaries obtained by an over-segmented scene-detection algorithm discussed in Sec. 2. We formulate the StoryGraph layout as an optimization problem in Sec. 3 and finally evaluate scenes and StoryGraphs in Sec. 4.

1.1. Related Work

We will discuss some work related to video representation and scene detection. Note that the focus of this paper is StoryGraphs, for which scene detection is only a precursor.

Scene Detection Scene change detection in movies and TV series is a well studied area of research. Rasheed and Shah [15] use color and motion-based shot similarity cues and obtain scenes by recursively applying normalized cuts on the affinity graph. Chasanis *et al*. [3] first create groups of shots using spectral clustering followed by a sequence alignment method. They detect a scene change when there is a substantial difference between overlapping windows of shots. Liang *et al*. [12] rely on external information such as scripts to perform scene detection by using an HMM to map the scene structure of the script onto the movie.

The most related work is by Han and Wu [10]. They model the problem as finding scene boundaries and evaluate the similarity of shots within a scene using normalized cut scores. They use dynamic programming (DP) to optimize boundary placement and consider three boundaries (two scenes) at a time. Our contribution to this domain is the explicit incorporation of shot threading via SIFT matches and the DP method to find optimal sets of shots. In contrast to [10] we do not need to look at two scenes at a time.

Video Representation While there has been a lot of work in the area of video summarization, representation, and abstraction, most of it is based on low-level summarization methods and usually creates still-image summaries [6, 18] or moving-image summaries [14]. The work by Smith and Kanade [17] is among the first attempts to use image understanding with language to produce video skims.

More recently, there have been some methods which use person identities to generate summaries rather than represent the video by its appearance. Tsoneva *et al*. [20] use character names from scripts and propose an importance function to rate content. Sang and Xu [16] first segment the video into scenes using the script followed by sub-story discovery and character interaction analysis to obtain a movie attraction score – a measure they use to generate summaries.

The work by Ercolessi *et al*. [7] is probably most related. They use plot de-interlacing to develop a web-based visualization framework which provides an overview of the TV episode. In a timeline chart, they show a few frames of each scene and place scenes which belong to the same sub-story together on the same row. They automatically mine sub-stories [8] within an episode using cues from color histograms, speaker diarization and automatic speech recognition. Note that unlike our method, they neither depict which characters appear when, nor show their interactions.

To the best of our knowledge, we present for the first time an automatic approach to visualize character interactions within videos as a timeline.

2. Video Analysis and Scene Detection

We first briefly discuss some pre-processing steps in order to prepare the video data for generating StoryGraphs.

Shot Boundaries Similar to [19] we detect shot boundaries using a normalized version of the *Displaced Frame Difference* (DFD) [21] that captures the difference between consecutive motion compensated frames. We will assume that all scene boundaries only occur at shot boundaries.

Shot Threading We perform shot threading [3, 4], *i.e*. link shots which appear similar (such as alternating shots in dialogs) to each other using SIFT matching [13]. Shot threads constitute an important cue of our scene detection as they typically involve the same set of characters at the same location, and thus should belong to the same scene. We compute similarity between the last frame of every shot and the first frame of $R = 25$ subsequent shots and set a threshold on the number of SIFT matches to decide whether two shots are part of a thread. Fig. 2 shows an example of such a matching scheme.

Figure 2: Top 40 SIFT matches on the last frame of shot **55** (frame 005474) and first frame of shot **57** (frame 005519) for BBT-1.

Shot Representation Each shot s is represented by the mean RGB color histogram, denoted as H_s, and computed over all its frames and stored in $6 \times 6 \times 6$ bins. We define the distance between a shot s and a set of shots $\mathcal{P}$

$$\mathcal{C}(s, \mathcal{P}) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \|H_s - H_p\|^2 \quad (1)$$

and normalize it using a sigmoid to yield a color-based shot similarity score $\phi(\mathcal{C}) \in [0, 1]$.

2.1. Scene Detection

A scene in video data typically refers to a group of shots that stays coherent in location, people and general appearance. In this section, we propose a method that can compute the optimal scene change locations for a given number of scenes or automatically determine the total number of scenes. Unlike other methods [15] which split the scenes hierarchically to reach the desired number of scenes or until a stopping criteria is met, we use dynamic programming (DP) to obtain an efficient and optimal solution which maximizes the overall objective.

As discussed in [10], the problem of finding scene change boundaries involves (i) measuring the coherence of a scene, *i.e.*, similarity of shots within that scene and (ii) searching for the appropriate boundary positions.

Let N_{sc} be the number of scenes. We formulate the problem as optimally grouping all the shots into sets of scenes S_i. Note that all scenes together $\bigcup_i S_i$ correspond to the set of all shots, and a shot is only assigned to one scene, *i.e.* $S_i \cap S_j = \{\}$. To obtain this optimal grouping, we solve

$$\mathcal{S}^* = \underset{S_i}{\operatorname{argmax}} \sum_{i=1}^{N_{sc}} \sum_{s \in S_i} \alpha(|\mathcal{P}|)(\phi(\mathcal{C}(s, \mathcal{P})) + T_{s,\mathcal{P}}) \quad (2)$$

where $\phi(\mathcal{C}(s, \mathcal{P}))$ is the color-based shot similarity score between shot $s \in S_i$ and the shots that precede it $\mathcal{P} \in S_i$. $T_{s,\mathcal{P}}$ captures whether there is a thread between s and any shot from $\mathcal{P}$ and $\alpha(n) = 1 - 0.5\,(n/N_l)^2$ is a decay factor which prevents the scene from growing too large.

We essentially perform an exhaustive search (Eq. 2) through all combinations of sets of shots which can be efficiently implemented as finding the optimal path through a cube of size $N_{sc} \times N_{sh} \times N_l$ – number of scenes, shots and layers respectively. N_l represents the maximum number of shots that can be assigned to one scene. By forcing that scenes should start at layer one, the third dimension of the cube allows us to easily determine the set of previous shots that are assigned to the same scene $\mathcal{P}$ and to incorporate the decay. One extreme case is when all shots form their own scene, *i.e.* $N_{sc} = N_{sh}$. However, in TV series data, this is highly impractical and to improve speed we restrict the number of scenes $N_{sc} < N_{sh}/5$.

Dynamic Programming In the forward computation pass of the DP, we add a shot to the same scene by

$$D_{i,j,k} = D_{i,j-1,k-1} + \alpha(k)(\phi(\mathcal{C}(j, \mathcal{P})) + T_{j,\mathcal{P}}) \quad (3)$$
$$\forall (i, j, k) \in [1..N_{sc}, 1..N_{sh}, 2..N_l]$$

where $\mathcal{P} = [j-k+1, \ldots, k-1]$ denotes the set of previous shots assigned to the same scene. $T_{j,\mathcal{P}} = 1$ when j forms a thread with any shot from $\mathcal{P}$.

We start a new scene at layer one ($k = 1$), and are likely to create a scene boundary when the shot appears different from the set of previous shots, and is not part of a thread.

$$D_{i,j,1} = \max_k \left[D_{i,j-1,k} + \beta(k)(1 - \phi(\mathcal{C}(k, \mathcal{P})) + \hat{T}_{k,\mathcal{P}}) \right] \quad (4)$$

where $\hat{T}_{k,\mathcal{P}} = 1$ when the shot k *does not* form a shot thread with any from $\mathcal{P}$. $\beta(k) = 1 - \alpha(k)$ plays the opposite role and prevents scenes from being too short. Note that we can move from any layer k to $k = 1$ which incorporates the potential to start a new scene at every shot.

After we compute the cube D, the optimal scene change locations are obtained via backtracking. If we are given the number of scenes N^*_{sc}, we start backtracking from the point $\max_k D_{N^*_{sc},N_{sh},k}$ *i.e.* the last column of row N^*_{sc} in the cube (after taking max across all layers). When the number of scenes needs to be determined, we use the scores $\forall i, \max_k D_{i,N_{sh},k}$ *i.e.* the last column (one for each row / scene) and detect the elbow point on the curve.

Although we perform an exhaustive search, the time it takes to compute the cube including color similarity scores is approximately only $1/8$ of the episode duration.

3. StoryGraphs

3.1. Person identification

Prior to determine the positions of the character lines in the StoryGraphs, we need to know which characters appear in every scene. We detect and track faces using a particle filter approach as discussed in [2] and extract Discrete Cosine Transform (DCT) coefficients on aligned faces for each track. Due to unavailability of transcripts (for a part of our data set), we are unable to perform fully automatic person identification using methods such as [2, 9]. Nevertheless, we train multinomial logistic regression classifiers by setting aside 20% of the tracks for each character – a close approximation of the number of tracks that are labeled via

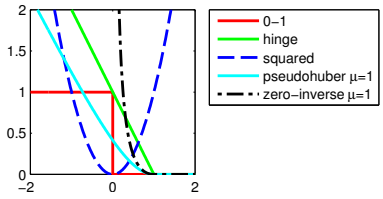

Figure 3: Visualization of loss functions used in our formulation. Note that the zero-inverse loss $\mathcal{Z}(x,1)$ is defined only on $x>0$.

transcripts (*c.f.* speaker-assignment recall [2, 9]). Our models are trained on all named characters, including minor cast to which the viewers can associate a name.

3.2. Loss functions

Generation of the StoryGraphs deals with the trade off between aesthetics and their functionality that characters which interact should occur nearby. We capture this property in an objective function consisting of four terms. The first loss function component (i) proximity, ensures that the lines of characters that interact, or co-occur within the same scene are drawn close to each other. Simultaneously, characters which do not occur in the scene are pushed apart. We capture the aesthetics of the graph in three loss functions where (ii) straight lines are preferred; (iii) lines should not overlap; and (iv) the number of crossings should be small.

Let x_t^c denote the coordinate at which the line for character c will be placed during the scene (time period) t. The total number of characters is denoted by N_C and the number of time periods (scenes) is $N_T = N_{sc}$. For brevity, let us also denote the number of pair-wise combinations of the characters by N_P, where $N_P = N_C \cdot (N_C-1)/2$.

(i) Proximity The proximity loss function $L_p^{(1)}$

$$L_p^{(1)} = \frac{1}{N_P N_T} \sum_{c_i,c_j,t} p_{c_i,c_j,t} \cdot (x_t^{c_i} - x_t^{c_j})^2 \qquad (5)$$

is responsible for pulling lines of characters that co-occur close together. On the other hand the loss function $L_p^{(2)}$

$$L_p^{(2)} = \frac{1}{N_P N_T} \sum_{c_i,c_j,t} \mathbb{1}\{p_{c_i,c_j,t}=0\} \cdot (x_t^{c_i} - x_t^{c_j})^2 \qquad (6)$$

pushes away lines of characters that do not appear. $p_{c_i,c_j,t} \in [0,1]$ is the normalized co-occurrence score between any two characters c_i and c_j at time t.

The combined proximity loss $L_p = L_p^{(1)} - L_p^{(2)}$ and it's minimization forces the lines for character c_i and c_j to come closer / move apart depending on whether they both appear. While ideally we would like to minimize $|x_t^{c_i} - x_t^{c_j}|$, we use the squared loss (Fig. 3-blue) to keep the function differentiable with respect to x_t^c.

$$\frac{\partial L_p^{(1)}}{\partial x_t^c} = \frac{1}{N_P N_T} \sum_{c'} 2 \cdot p_{c,c',t} \cdot (x_t^c - x_t^{c'}). \qquad (7)$$

(ii) Straight lines The straight line loss function L_l tries to keep the lines straight, and is represented for each character c by the squared difference between one coordinate x_t^c with respect to $\mu_{\setminus x_t^c} = \frac{1}{N_T-1}\sum_{q\neq t} x_q^c$, the mean of all other time coordinates

$$L_l = \frac{1}{N_C N_T} \sum_{c,t} (x_t^c - \mu_{\setminus x_t^c})^2 \qquad (8)$$

and has a gradient with respect to x_t^c given by

$$\frac{\partial L_l}{\partial x_t^c} = \frac{1}{N_C N_T} 2(x_t^c - \mu_{\setminus x_t^c}). \qquad (9)$$

(iii) Minimum separation While the proximity loss $L_p^{(1)}$ pulls lines closer, the minimum separation loss L_s ensures that the lines do not collapse or overlap. This is necessary to ensure readability of the StoryGraphs and is enforced strictly by using a smooth differentiable modification of $1/x$ going to $0, \forall x \geq \mu$ (Fig. 3-black)

$$\mathcal{Z}(x,\mu) = \begin{cases} \frac{1}{x} \cdot (\sqrt{1+(x-\mu)^2} - 1) & 0 < x < \mu \\ 0 & x \geq \mu. \end{cases} \qquad (10)$$

The loss is then given by

$$L_s = \frac{1}{N_P N_T} \sum_{c_i,c_j,t} \mathcal{Z}((x_t^{c_i} - x_t^{c_j})^2, \mu_s) \qquad (11)$$

where μ_s is the desired minimum separation between all lines on the chart. The gradient can be computed as

$$\frac{\partial L_s}{\partial x_t^c} = \frac{1}{N_P N_T} \sum_{c'} 2 \cdot \mathcal{Z}'((x_t^c - x_t^{c'})^2, \mu_s) \cdot (x_t^c - x_t^{c'}). \qquad (12)$$

(iv) Crossings Finally, we wish to minimize the number of crossing between lines of different characters. A line crossing occurs iff the product of coordinates between a pair of characters across consecutive time periods is less than 0, $(x_t^{c_i} - x_t^{c_j})(x_{t+1}^{c_i} - x_{t+1}^{c_j}) < 0$.

While the number of crossings can be counted using a 0-1 loss (Fig. 3-red) this is neither differentiable, nor convex. Thus, we replace it by a differentiable version of the hinge loss, namely the pseudo-Huber loss [11] (Fig. 3-cyan) defined on all x

$$\mathcal{H}(x,\mu) = \begin{cases} \sqrt{1+(x-\mu)^2} - 1 & x < \mu \\ 0 & x \geq \mu. \end{cases} \qquad (13)$$

The crossings loss is then formulated as

$$L_c = \frac{1}{N_P N_T} \sum_{c_i,c_j,t} \mathcal{H}((x_t^{c_i} - x_t^{c_j})(x_{t+1}^{c_i} - x_{t+1}^{c_j}), \mu_c) \qquad (14)$$

where we choose $\mu_c = 0$, and the gradient is

$$\frac{\partial L_c}{\partial x_t^c} = \frac{1}{N_P N_T} \sum_{c'} \Big[(x_{t-1}^c - x_{t-1}^{c'}) \mathcal{H}'((x_{t-1}^c - x_{t-1}^{c'})(x_t^c - x_t^{c'})) + (x_{t+1}^c - x_{t+1}^{c'}) \mathcal{H}'((x_{t+1}^c - x_{t+1}^{c'})(x_t^c - x_t^{c'})) \Big]. \quad (15)$$

Combined Loss The final objective is expressed as the weighted combination of the above four losses

$$\mathcal{L}(\mathbf{x}) = w_p L_p(\mathbf{x}) + w_l L_l(\mathbf{x}) + w_s L_s(\mathbf{x}) + w_c L_c(\mathbf{x}) \quad (16)$$

and the optimal positioning of all lines is obtained via

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \mathcal{L}(\mathbf{x}). \quad (17)$$

The final drawing consists of three parts. We (i) sketch scene boundaries; (ii) draw character lines in the central portion of the scene using the coordinates; and (iii) generate smooth transitions across scenes using the sigmoid.

3.3. Implementation Details

The optimization is performed using Matlab's `fmincon` and x_t^c is constrained as $1 \leq x_t^c \leq N_C$.

Co-occurrence computation For each scene (time period) we first count the number of frames for which each character appears. We threshold this value to reduce the influence of person-id errors on the StoryGraph generation. The character-pair co-occurrence score $p_{c_i,c_j,t}$ is then given by the geometric mean of the frame counts and is finally normalized using hysteresis thresholding.

Initialization The objective for the optimization $\mathcal{L}$ is not convex. Therefore, a good initial guess about the coordinate positions for characters is preferred. We use the negated co-occurrence scores as distance and perform agglomerative clustering on the characters and use the optimal leaf order [1] as our initial set of coordinates.

Presence During optimization, we consider the loss functions for a single character (straight line loss) or a character pair (the other 3 losses) only from the first (▶) to the last (●) appearance of the character in the episode.

4. Evaluation and Results

We present results on scene detection, person identification, and StoryGraphs generation on 3 diverse TV series[1]:

(i) The Big Bang Theory: (BBT, S01, E01–E06, 20min) is a traditional sitcom with a short character list in which most characters appear simultaneously. The cast in the six episodes consists of 11 named people (5 major, 6 minor).

[1] cvhci.anthropomatik.kit.edu/projects/mma

		BBT-1	BBT-2	BF-1	BF-2	GOT-1	GOT-2
σ(s)	DP	**18.46**	**7.95**	**13.41**	**12.00**	**12.61**	**17.05**
	[15]	21.38	16.15	19.52	16.34	21.63	29.52
R_{30}	DP	**75.00**	**90.91**	**84.85**	**96.67**	**88.57**	**87.10**
	[15]	62.50	81.82	72.73	86.67	80.00	77.42

Table 1: Scene detection performance with over-segmentation (OS). Note the higher recall which is important for StoryGraphs.

		BBT-1	BBT-2	BF-1	BF-2	GOT-1	GOT-2
σ(s)	DP	**24.91**	**27.79**	**15.04**	**16.34**	**18.55**	**33.36**
	[15]	25.85	39.25	25.62	22.23	42.84	50.74
R_{30}	DP	**62.50**	**63.64**	**81.82**	**90.00**	**74.29**	**77.42**
	[15]	56.25	54.55	63.64	76.67	57.14	64.52

Table 2: Scene detection performance when the number of predicted scenes is exactly the same as number of annotated scenes.

(ii) Buffy the Vampire Slayer: (BUFFY, S05, E01–E06, ∼40min) is a TV series with a serialized format, *i.e.* each episode is a self-contained story which also contributes to a larger storyline of a season. The cast in the first six episodes consists of 27 named people (10 major, 17 minor).

(iii) Game of Thrones: (GOT, S01, E01–E10, ∼55min) is a fantasy-drama TV series with an inclination towards high-quality production which resembles movies. The cast for season 1 (excluding extras) consists of 66 people.

4.1. Scene Detection Performance

In this section, we compare our method (Sec. 2.1 DP) to [15] on 2 episodes from each TV series. We measure the quality of scene detection using two criteria. (i) σ(s): the mean absolute deviation in seconds from annotated scene-boundary locations for the set of computed scenes (lower is better) (ii) R_{30}: recall at 30 seconds [15], the percentage of annotated scene boundaries which lie within 30s of the predicted boundary (higher is better).

Table 1 shows results for the case of over-segmented scene (OS) detection (one scene per minute) used in the StoryGraph generation and Table 2 shows results when the number of automatically detected scenes is set to be the same as the number of scenes annotated using ground-truth. Note that OS scenes have a higher recall due to larger chance of finding one of the scene boundaries at the ground truth location. For StoryGraphs, we use OS since it ensures that we find more scene changes, while only slightly (1.2x) increasing computational load. Fig. 4 shows an example of detected scene changes on one episode of Buffy.

4.2. PersonID Performance and Impact

We encounter large variations in the number of tracks, from 5-10 for minor characters all the way up to a thousand for major ones. This adversely impacts identification accuracy (number of correctly identified tracks / total number of tracks) even after appropriately weighting classes with

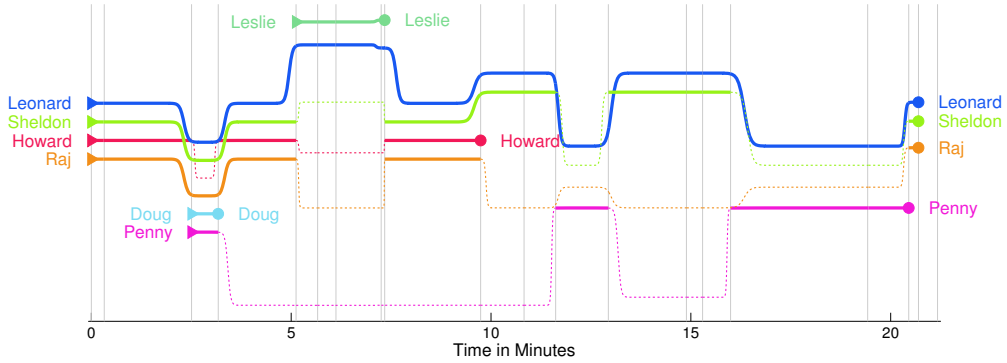

Figure 5: StoryGraph generated on BBT-3 using automatic person identities. Compare against Fig. 1 which uses ground truth person identities. See Sec. 4.2 for a short description. This figure is best viewed in color.

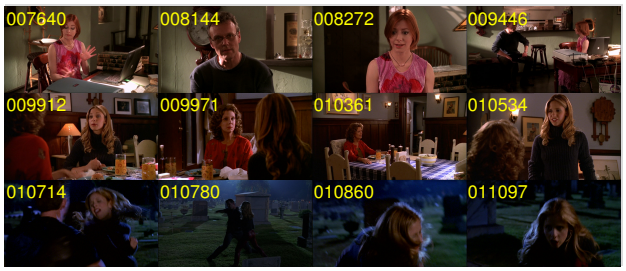

Figure 4: Example of scene change detection from BF-1. Each row indicates a different scene. We show the mid-frame (number on the image) from 4 shots for each scene.

	BBT (6)	BUFFY (6)	GOT (10)
#Characters	11	27	66
Mean Accuracy	92.36%	78.12%	75.25%
SG Presence Error Count	33	260	680
SG Presence Error Fraction	4.85%	8.08%	4.24%

Table 3: Face track identification accuracy and impact on StoryGraph line drawing presence averaged across all episodes. The number of episodes is indicated in brackets.

less number of samples during training. Table 3 (top-half) presents the average identification accuracy for our data.

The impact of person identification on StoryGraphs can be quantified by counting the number of times a character is classified as present in a scene while he is not, and vice versa. The absolute count of errors and the fraction (with respect to the complete chart size is $N_C \times N_{sc}$) is presented in Table 3 (bottom-half). Note that person-id errors do not imply a commensurate SG presence error and therefore do not drastically affect the StoryGraphs visualization.

We also qualitatively compare StoryGraphs for BBT-3 generated using groundtruth (Fig. 1) vs. automatic (Fig. 5) person identification. The main difference is Penny being classified as not-present during minute 1-3, and thus her line starts a little later along with Doug. We also see that Sheldon and Raj are misclassified during the short scene with Doug. The rest of the StoryGraph stays unchanged.

For other charts (see supplementary material for comparative charts on all episodes), we observe that most differences can be attributed to erroneous classification of minor characters, or incorrect presence classification.

4.3. StoryGraph Quality

We define five measures to quantify the nature of a StoryGraph: (i) normalized maximum coordinate movement over time for each line, averaged across characters

$$\text{Move} = \frac{1}{N_C} \sum_{c} \left(\max_t x_t^c - \min_t x_t^c \right). \quad (18)$$

(ii) worst case separation between any two characters who are present during the scene averaged across all scenes

$$\text{MaxSep} = \frac{1}{N_T} \sum_{t} \max_{c_i} \left[\min_{c_j, j \neq i} (|x_t^{c_i} - x_t^{c_j}|) \right]. \quad (19)$$

(iii) number of times MaxSep is more than twice the minimum separation $\mu_s = 0.3$, denoted by "Sep $> 2\mu_s$".
(iv) number of times MaxSep is smaller than half the minimum separation, denoted by "Sep $< 0.5\mu_s$".
(v) and finally, the number of crossings

$$\#\text{Cross} = \sum_{c_i, c_j, t} \mathbb{1}\{(x_t^{c_i} - x_t^{c_j})(x_{t+1}^{c_i} - x_{t+1}^{c_j}) < 0\}. \quad (20)$$

Results for StoryGraphs from all episodes are presented in Table 4 . We observe that lines tend to be straighter (small Move score) for GOT in comparison to BBT and BUFFY due to the higher weight w_l and different groups of characters chosen from a smaller pool. A lower MaxSep is desirable as it indicates that the proximity pull loss works well. Due to the larger cast size for BUFFY and GOT, we observe that few lines stay separated by more than twice the minimum separation (Sep $> 2\mu_s$). We also see that lines are separated by at least $0.5\mu_s$, a result of minimum separation with very few occasional violations. Finally with a few exceptions GOT-1 the number of crossings in relation to the graph size is typically small.

	The Big Bang Theory (BBT)						Buffy the Vampire Slayer (BUFFY)						Game of Thrones (GOT)									
	E01	E02	E03	E04	E05	E06	E01	E02	E03	E04	E05	E06	E01	E02	E03	E04	E05	E06	E07	E08	E09	E10
Move	0.32	0.34	0.19	0.31	0.22	0.30	0.46	0.29	0.21	0.28	0.15	0.39	0.30	0.21	0.21	0.18	0.17	0.12	0.20	0.19	0.18	0.19
MaxSep	0.45	0.67	1.06	1.72	0.87	0.90	0.30	0.34	0.43	0.90	0.51	0.34	0.34	0.43	0.81	0.61	0.72	0.68	0.36	1.01	0.42	0.54
Sep $> 2\mu_s$	4	12	11	11	10	9	0	2	7	27	13	1	1	6	31	11	25	14	1	34	6	8
Sep $< 0.5\mu_s$	1	1	1	1	1	1	1	0	1	1	1	0	1	3	2	3	4	3	3	2	4	2
#Cross	0	1	0	0	0	0	83	43	19	46	8	128	212	66	134	44	44	26	116	60	44	26

Table 4: StoryGraph quality over all episodes from our data set. The StoryGraphs are generated using automatic person identification.

4.4. StoryGraph in relation to the Story

As a way to evaluate the video representation ability of StoryGraphs, we compare them to the crowd-sourced episode summaries commonly found on Wikipedia or fan-websites. To facilitate easy comparison, we provide the plot synopsis (or links) for all episodes of BBT, BUFFY and GOT in the supplementary material. In Fig. 6, we see the StoryGraph for BUFFY-2 along with a short comparison to the actual storyline of the episode. Fig. 7 shows a StoryGraph from Game of Thrones episode 6. We refer to the sub-stories in the caption as a reference for the reader.

5. Conclusion

We present an automatic method to generate representations for character interactions in TV episodes. Characters are indicated by lines, solid (or dashed) when the actor is visible on-screen (or not). The placement of the lines is posed as an optimization problem which trades off between aesthetics and representative power of the StoryGraphs. A scene detection method is first used to oversegment the video into coherent parts. We evaluate the StoryGraphs on a data set consisting of 22 episodes from 3 diverse TV series and show good objective performance and ability to present an overview of the episode.

Acknowledgments We thank Andrew Zisserman for discussions on the scene detection algorithm; the reviewers for valuable feedback on improving the paper; and Daniel Koester for comments on the presentation of StoryGraphs. This work was realized as part of the Quaero Program, funded by OSEO, French State agency for innovation. The views expressed herein are the authors' responsibility and do not necessarily reflect those of OSEO.

References

[1] Z. Bar-Joseph, D. K. Gifford, and T. S. Jaakkola. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics*, 17 Suppl 1:S22–S29, Jan. 2001.

[2] M. Bäuml, M. Tapaswi, and R. Stiefelhagen. Semi-supervised Learning with Constraints for Person Identification in Multimedia Data. In *CVPR*, 2013.

[3] V. T. Chasanis, A. C. Likas, and N. P. Galatsanos. Scene Detection in Videos Using Shot Clustering and Sequence Alignment. *IEEE Transactions on Multimedia*, 11(1):89–100, 2009.

[4] T. Cour, C. Jordan, E. Miltsakaki, and B. Taskar. Movie/script: Alignment and parsing of video and text transcription. In *ECCV*, 2008.

[5] T. Cour, B. Sapp, A. Nagle, and B. Taskar. Talking Pictures: Temporal Grouping and Dialog-Supervised Person Recognition. In *CVPR*, 2010.

[6] D. DeMenthon, V. Kobla, and D. Doermann. Video summarization by curve simplification. In *ACM Multimedia*, 1998.

[7] P. Ercolessi, H. Bredin, and C. Sénac. StoViz : Story Visualisation of TV Series. In *ACM Multimedia*, 2012.

[8] P. Ercolessi, S. Christine, and H. Bredin. Toward Plot De-Interlacing in TV Series using Scenes Clustering. In *Content-Based Multimedia Indexing*, 2012.

[9] M. Everingham, J. Sivic, and A. Zisserman. Hello ! My name is ... Buffy Automatic Naming of Characters in TV Video. In *British Machine Vision Conference*, 2006.

[10] B. Han and W. Wu. Video Scene Segmentation using a Novel Boundary Evaluation Criterion and Dynamic Programming. In *International Conference on Multimedia and Expo*, 2011.

[11] P. J. Huber. Robust Estimation of a Location Parameter. *Annals of Statistics*, 53:73–101, 1964.

[12] C. Liang, Y. Zhang, J. Cheng, C. Xu, and H. Lu. A Novel Role-Based Movie Scene Segmentation Method. In *Pacific Rim Conference on Multimedia*, 2009.

[13] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60(2):91–110, 2004.

[14] S. Pfeiffer, R. Lienhart, S. Fischer, and W. Effelsberg. Abstracting digital movies automatically. *J. of Visual Communication and Image Representation*, 7(4):345–353, 1996.

[15] Z. Rasheed and M. Shah. Detection and representation of scenes in videos. *IEEE Transactions on Multimedia*, 7(6):1097–1105, Dec. 2005.

[16] J. Sang and C. Xu. Character-based Movie Summarization. In *ACM Multimedia*, 2010.

[17] M. Smith and T. Kanade. Video Skimming and Characterization through the Combination of Image and Language Understanding Techniques. In *CVPR*, 1997.

[18] S. W. Smoliar and H. J. Zhang. Content-based video indexing and retrieval. *IEEE Multimedia*, 1(2):62–72, 1994.

[19] M. Tapaswi, M. Bäuml, and R. Stiefelhagen. Story-based Video Retrieval in TV series using Plot Synopses. In *ACM International Conference on Multimedia Retrieval*, 2014.

[20] T. Tsoneva, M. Barbieri, and H. Weda. Automated summarization of narrative video on a semantic level. In *International Conference on Semantic Computing*, 2007.

[21] Y. Yusoff, W. Christmas, and J. Kittler. A Study on Automatic Shot Change Detection. *Multimedia Applications and Services*, 1998.

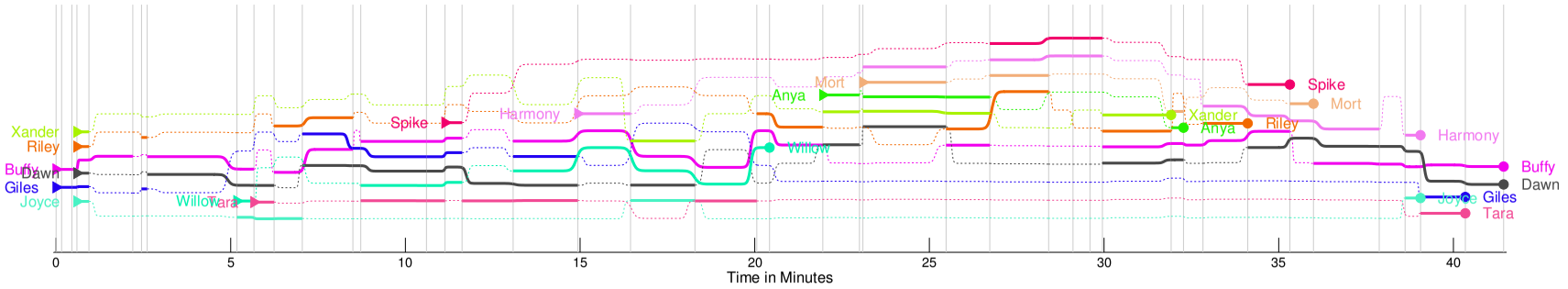

Figure 6: StoryGraph for season 5, episode 2 of Buffy the Vampire Slayer generated using automatic person identification. We briefly compare the representation of the episode to the plot synopsis http://en.wikipedia.org/wiki/Real_Me_(Buffy_the_Vampire_Slayer). For brevity, we will refer to minutes A to B as (mA-B). The episode starts with Buffy and Giles in training (m1). We then see Dawn annoy Buffy while Joyce asks her to take her shopping (m3-6). Buffy has to cancel his plans with Riley (m6-7) as Giles drives off with the two sisters. They meet with Willow and Tara and discover that the owner of the magic shop is killed (m9-11). Tara consoles Dawn as she is kept out of the gang (m12) while the others discuss future plans (m13-15). As Joyce leaves home, Xander keeps Dawn company (m17) while Buffy rants to Riley (m21). Harmony and her gang visit Buffy's home only to find that she is not there (m22-25). On reaching home, Buffy is informed that Dawn invited Harmony to come into the house (m25-27) and she goes out in search of Harmony (m35). Harmony, after her failed plan, later encounters Spike (m28-30). In the meanwhile, Mort kidnaps Dawn and brings her to the lair (m33). We see a fight between Harmony's minions and Buffy to rescue her sister Dawn (m36-38) after which we cut to the shop where Dawn is writing about Buffy in her diary (m40-41). There are some person identification induced errors, mainly between Mort and Riley and Joyce and Buffy at the beginning.

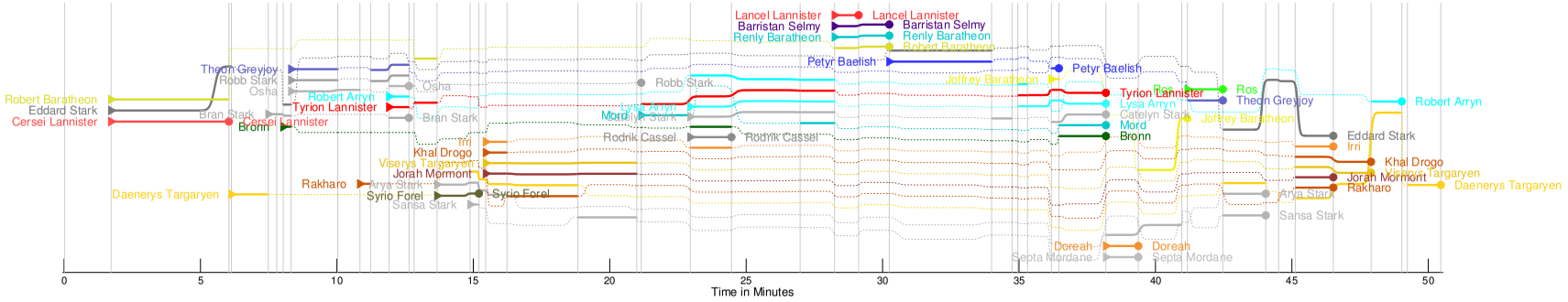

Figure 7: Example StoryGraph generated using automatic person identification for Game of Thrones episode 6. Note the clear demarcation between groups of characters from different places in the fantasy world. For brevity, we will refer to minutes A to B as (mA-B). We see scenes from King's Landing @ m1-6, m12-15, m27-35, m42-45 involving Robert Baratheon, Cersei Lannister, Eddard, Arya and Sansa Stark and Petyr Baelish; from the Eyrie @ m21-m27 and m35-38 involving Catelyn Stark, Rodrick Cassel, Lysa and Robert Arryn, and the imprisonment of Tyrion Lannister; from Winterfell @ m8-12 primarily involving Bran and Robb Stark and Theon Greyjoy; and finally Beyond the Narrow Sea @ m15-20 and m45-50 involving Targaryens, Khal Drogo and Jorah Mormont.