Scene Consistency Representation Learning for Video Scene Segmentation Supplementary Materials

Haoqian Wu[1,2,3,4*], Keyu Chen[2*], Yanan Luo[2], Ruizhi Qiao[2], Bo Ren[2],
Haozhe Liu[1,3,4,5], Weicheng Xie[1,3,4†], Linlin Shen[1,3,4]
[1] Computer Vision Institute, Shenzhen University [2] Tencent YouTu Lab
[3] Shenzhen Institute of Artificial Intelligence and Robotics for Society
[4] Guangdong Key Laboratory of Intelligent Information Processing [5] KAUST
wuhaoqian2019@email.szu.edu.cn {yolochen, ruizhiqiao, timren}@tencent.com
luoyanan93@gmail.com haozhe.liu@kaust.edu.sa {wcxie, llshen}@szu.edu.cn

1. Additional Illustrations

1.1. Frame, Shot, Clip and Scene

Fig. 1 shows the connection between **frame**, **shot**, **scene** and **video**. More specifically, a **shot** contains only continuous frames taken by the camera without interruption, and a **scene** is composed of successive shots and describes a same short story. Typically, a consecutive shot sequence of arbitrary length can be treated as a **clip**.

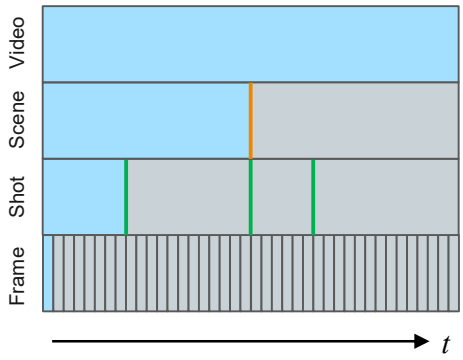


Figure 1. **The illustration of the connection between frame, shot, scene and video.** The green and orange lines represent *Shot* and *Scene Boundaries*, respectively.

1.2. Shot Sampling Strategy

Video data is highly redundant because there are many repetitive frames in chronological order, and we follow the sampling strategy in [1, 2] for long-term videos. More concretely, the video sequence is sliced according to the shot boundaries that are determined by the transition of the visual modality. Based on the beginning and ending positions of shots, a fixed number of $N = 3$ frames are selected and treated as the original feature of one shot, where the starting, middle and ending frames of the shot are sampled.

2. Algorithm Details

2.1. Representation Learning Stage

In the Representation Learning Stage, the proposed SSL algorithm can be summarized in Algorithm 1.

Algorithm 1 SSL for Video Scene Segmentation.

Input: Training samples X
Require: Initialized encoders $f(\cdot \mid \theta_Q)$ and $f(\cdot \mid \theta_K)$; Initialized memory bank $Queue$; Augmentation operations $Aug_Q(\cdot)$ and $Aug_K(\cdot)$; Number of iterations n_{iter}

for $i = 1$ to n_{iter} **do**
  Obtain augmented training samples Q, K by $Q = Aug_Q(X), K = Aug_K(X)$;
  Obtain mapping function $MAP(\cdot)$;
  Obtain positive pairs $\{q, k^+\}$ by Eq. (1);
  Detach samples k^+ from Calculation Graph;
  Obtain negative samples k^- from $Queue$;
  Calculate contrastive loss L_{con} by Eq. (7) or (8);
  Perform backpropagations for L_{con};
  Update encoder $f(\cdot \mid \theta_Q)$ by gradient descent;
  Update encoder $f(\cdot \mid \theta_K)$ by momentum update;
  Enqueue the positive samples k^+ to $Queue$;
end for

Output: Query encoder $f(\cdot \mid \theta_Q)$.

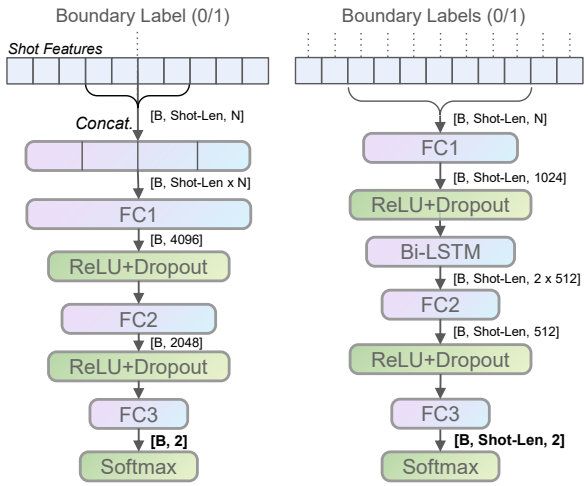
2.2. Video Scene Segmentation Stage

The Boundary based model (*i.e.*, MLP protocol [2]) and Boundary free model (*i.e.*, Bi-LSTM protocol introduced by us) are presented in Fig. 2.

*Equal Contribution †Corresponding Author

For the MLP protocol [2], we use SGD as the optimizer, and the weight decay is 1e-4 and the SGD momentum is 0.9. In the training stage, we use a mini-batch size of 128 and dropout rate of 0.4 for FC layers. Besides, we train for 200 epochs with the learning rate multiplied by 0.1 at 50, 100 and 150 epochs.

For the Bi-LSTM protocol, we use SGD as the optimizer, and the weight decay is 1e-4 and the SGD momentum is 0.9. In the training stage, we use a mini-batch size of 32 and dropout rate of 0.7 for FC layers (except for the last layer), the Bi-LSTM was implemented using the LSTM module in PyTorch [3], which includes two layers with 512 hidden units together with a dropout layer with the dropout probability of 0.6 between them. Besides, we train for 200 epochs with the learning rate multiplied by 0.1 at 160, 180 epochs. In the inference stage, in order to make each shot aggregate as much information as possible from the adjacent shots, in each inference batch, we use the middle portion of the model output sequence as the scene boundary. More specifically, for the input shot feature sequence, *i.e.* $[S_0, S_1, \cdots, S_{Shot-Len\ -1}]$, we adopt the subsequence, *i.e.* $[Y_{\lceil Shot-Len/4 \rceil}, \cdots, Y_{\lceil 3*Shot-Len/4 \rceil\ -1}]$ of the corresponding output sequence, *i.e.* $[Y_0, Y_1, \cdots, Y_{Shot-Len\ -1}]$ as the prediction result. Additionally, the first and last shot features are used to pad the beginning and ending of the shot feature sequence, respectively.



(a) Boundary based Model (b) Boundary free Model

Figure 2. The illustration of Boundary based and Boundary free models, where B, N and $Shot\text{-}Len$ represent the batch size, dimension of the feature and length of shots processed within a batch.

3. Additional Implementation Details

3.1. Datasets

The details of BBC [4], OVSD [5], MovieNet [6] and MovieScene [1] datasets are shown in Tab. 1. The scalability of BBC [4] and OVSD [5] is much smaller than that of MovieNet [6] and MovieScene [6]. BBC [4] has 5 different annotations and the number of scenes is averaged, as shown in Tab. 1. The ground truth Shot Detection result of OVSD [5] is unavailable from its official website and [1], thus, we provide our implementation results on Shot Detection in the Tab. 1. It is worth noting that ***movies in the MovieScene [1] are all included in MovieNet [6].***

Table 1. Details of BBC/OVSD/MovieNet/MovieScene datasets.

Datasets	**#Video**	**Time(h)**	**#Shot**	**#Scene**
BBC [4]	11	9	4,900	547
OVSD [5]	21	10	9,377	607
MovieScene-150 [1]	150	297	270,450	21,428
MovieScene-318 [1]	318	601	503,522	41,963
MovieNet [6]	1,100	3,000		

To show the difference of the 5 annotation results, we present the average (mean), minimum (min), maximum (max) and standard deviation (std.) of the number of scenes for each video in BBC [4] with each annotation in Tab. 2.

Table 2. Number of scenes in each video of BBC dataset.

Video Names	#Scene			
	mean	**min**	**max**	**std.**
From Pole to Pole	47.8	23	65	14.4
Mountains	47.6	36	62	9.8
Ice Worlds	50.6	33	69	12.1
Great Plains	52.0	30	74	14.1
Jungles	47.4	25	59	11.8
Seasonal Forests	53.4	33	71	12.1
Fresh Water	55.2	37	70	10.5
Ocean Deep	47.8	29	67	12.7
Shallow Seas	49.2	33	66	12.0
Caves	45.4	22	63	14.1
Deserts	50.8	26	64	13.7
All Videos Avg.	547			

3.2. Backbones

For the visual modality, ResNet50 [7] is used as the encoder with the same modification [2] that the input channel number of first convolution layer is changed from 3 to 9. As for audio modality, we adopt the same backbone used in [1].

3.3. Choice of Hyperparameters

Two hyperparameters are introduced in the Sec. 3.1.2 of the main body of this work, *i.e.* number of cluster ($\#class$) for Scene Consistency Selection strategy and length of continuous shots (ρ) for Scene Agnostic Clip-Shuffling. We study the sensitivity of the proposed algorithm against these two hyperparameters in Tab. 3. MLP [2] protocol on the MovieScene-318 dataset is used in this experiment.

Table 3. AP results for different settings of hyperparameters. The **bolded** and underlined values stand for the optimal and suboptimal performances, respectively.

ρ / $\#class$	**16**	**24**	**32**
16	51.80	**53.74**	53.22
24	52.64	<u>53.62</u>	53.13
32	52.68	52.91	53.05

3.4. Data Augmentation Details

We follow the data augmentation operation used in the [8], *i.e.* random cropping, flipping, color distortion, Gaussian bluring. A PyTorch-like pseudo code for the data augmentations, *i.e., Asymmetric Augmentation mentioned in Sec. 3.1.2 of the body of this work*, is presented as follows:

```
import torchvision.transforms as transforms
normalize = transforms.Normalize(
    mean=[0.485, 0.456, 0.406],
    std=[0.229, 0.224, 0.225])
# augmentation for key encoder
augmentation_key_encoder = [
    transforms.ToPILImage(),
    transforms.RandomResizedCrop(224, scale=(0.2,
     1.)),
    transforms.RandomApply([
        transforms.ColorJitter(0.4, 0.4, 0.4,
    0.1)], p=0.5),
    transforms.RandomGrayscale(p=0.2),
    transforms.RandomApply([GaussianBlur([.1,
    2.])], p=0.5),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    normalize
    ]
# augmentation for query encoder
augmentation_query_encoder = [
    transforms.ToPILImage(),
    transforms.RandomResizedCrop(224, scale=(0.2,
     1.)),
    transforms.RandomApply([GaussianBlur([.1,
    2.])], p=0.5),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    normalize
    ]
```

Listing 1. A PyTorch-like pseudo code for the data augmentation.

4. Additional Results

4.1. Results on BBC/OVSD Datasets

Since the training/validation/testing datasets of BBC [4]/OVSD [5] are not available and the scale of these two datasets is very small compared to MovieNet [6] dataset, we apply the model trained on MovieNet [6] onto BBC [4] and OVSD [5] to study the generalization abilities of the algorithms without the finetuning, the results are shown in Tab. 4 and Tab. 5.

Tab. 4 shows that the proposed method outperforms ShotCoL [2] by a large margin of 13.27 in terms of AP on OVSD [5].

Table 4. AP results on OVSD dataset.

Methods	**AP**
ShotCoL [2]	25.53
SCRL	**38.80**

We conduct experiments of 5 different annotators on BBC [4] and show the average performances in Tab. 5, where the proposed method outperforms the compared method by a margin of 2.20 in terms of AP.

Table 5. AP results on BBC dataset. **A. *i*** stands for the i-th annotation and **Avg.** represents the average of the results of 5 different annotators.

Methods	**A. 1**	**A. 2**	**A. 3**	**A. 4**	**A. 5**	**Avg.**
ShotCoL [2]	29.90	30.81	31.45	26.45	21.27	27.98
SCRL	32.45	32.54	33.27	28.36	24.27	**30.18**

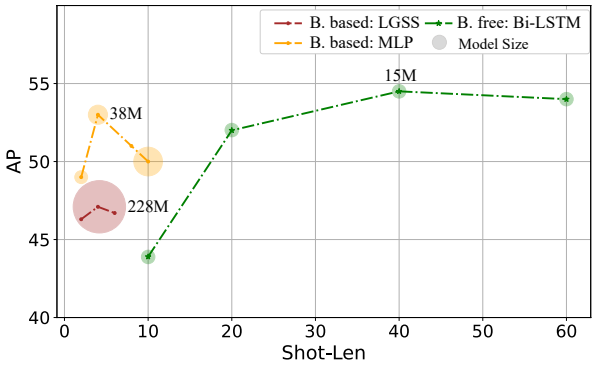


Figure 3. AP results on the MovieScene-318 dataset and model size of Boundary based and Boundary free models, where **B.** stands for **Boundary** and $Shot\text{-}Len$ represents the length of shots processed within a batch.

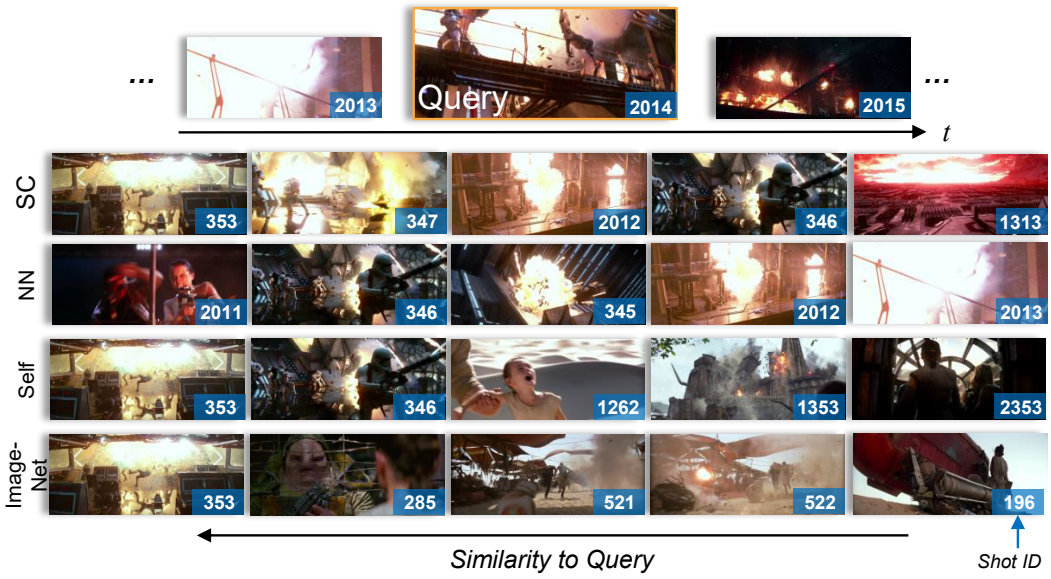


Figure 4. **The visualization results of shot retrieval**. Compared with the other methods, the results of ***SC*** appear more consistent in terms of the the semantic information, *i.e.*, scenes with explosions.

4.2. Results under MLP/Bi-LSTM protocols

To study the superiority of the introduced evaluation protocol for the task of *Video Scene Segmentation*, AP results together with the model size of Boundary based (*i.e.,* LGSS [1] and MLP [2] protocol) and Boundary free (*i.e.,* introduced Bi-LSTM protocol) models are shown in Fig. 3.

As shown in Fig. 3, although performances of all models are associated with the length of the shots, *i.e., Shot-Len*, the Boundary based model achieves the best performance only when the *Shot-Len* is relatively small (and the optimal *Shot-Len* is less than the average number of shots per scene, *i.e.* 12). As *Shot-Len* becomes larger, the performance decreases and the model size increases. By contrast, the Boundary free model produces less inductive bias and takes the shot features as the unit of basic temporal input, hence, it is able to model representations of longer shots, while achieving better performances when *Shot-Len* takes a value in the approximate range and a model with the same size is employed.

4.3. Visualization

4.3.1 Shot Retrieval

An additional result of Shot Retrieval, which is introduced in the Sec. 4.4 of the main body of this work, is given in the Fig. 4.

4.3.2 Scene Boundary

To study the practical performance of our approach for the *Video Scene Segmentation* task, we visualize the GT/Prediction scene boundaries in Fig. 5. For simple scenes in Fig. 5 (a1)/(a2), the proposed method easily identifies these scenes and gives correct predictions of the scene boundary. As shown in Fig. 5 (b1)/(b2)/(c1)/(c2), there are also bad cases where the proposed method fails to distinguish between the segmentation points of a shot and a scene, and for these cases, it may be confusing to identify whether these shots belong to the same scene or not, from the visual modality.

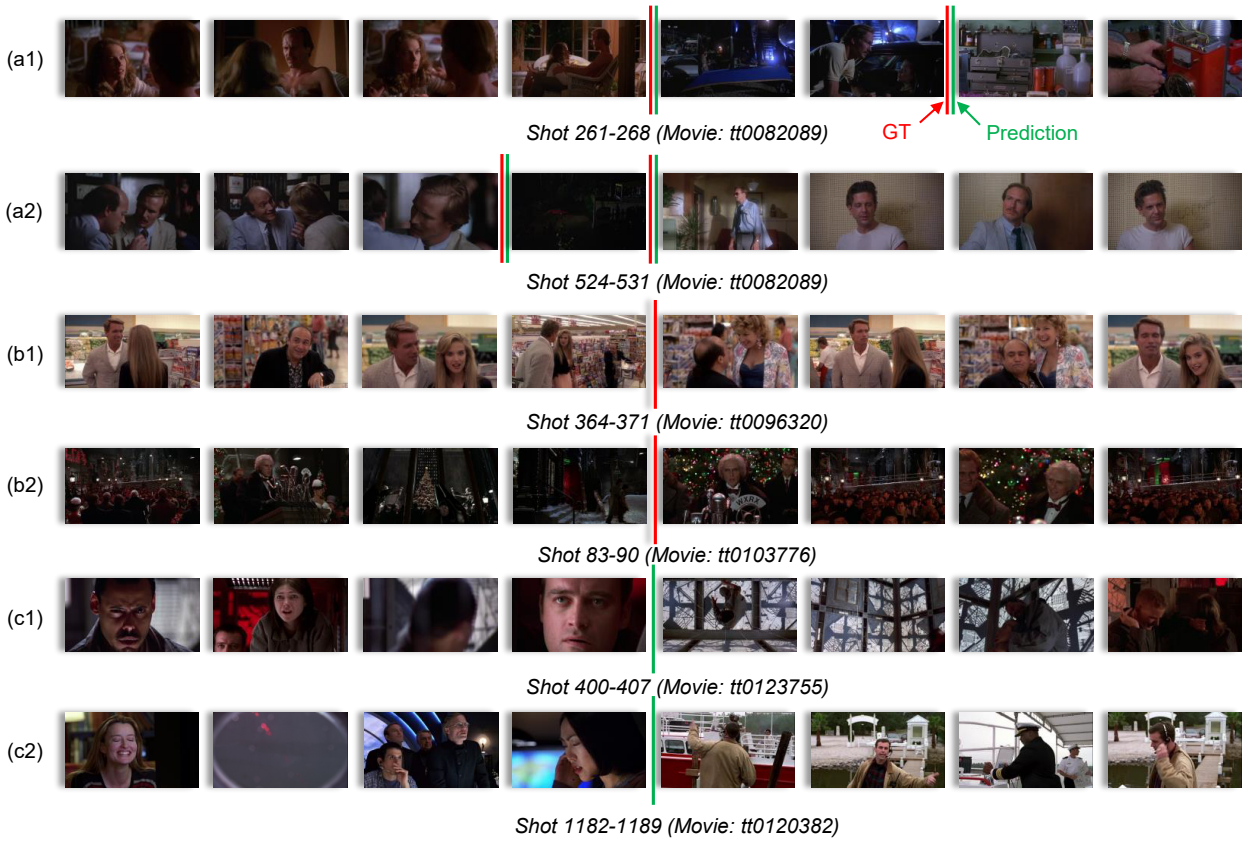


Figure 5. The ground truth (GT) and prediction scene boundaries are presented in this figure, where the middle frame of each shot is visualized. Fig. (a1)/(a2), Fig. (b1)/(b2) and Fig. (c1)/(c2) show the prediction cases of true positive, false negative and false positive, respectively.

References

[1] Anyi Rao, Linning Xu, Yu Xiong, Guodong Xu, Qingqiu Huang, Bolei Zhou, and Dahua Lin. A local-to-global approach to multi-modal movie scene segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10146–10155, 2020. 1, 2, 4

[2] Shixing Chen, Xiaohan Nie, David Fan, Dongqing Zhang, Vimal Bhat, and Raffay Hamid. Shot contrastive self-supervised learning for scene boundary detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9796–9805, 2021. 1, 2, 3, 4

[3] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019. 2

[4] Lorenzo Baraldi, Costantino Grana, and Rita Cucchiara. A deep siamese network for scene detection in broadcast videos. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1199–1202, 2015. 2, 3

[5] Daniel Rotman, Dror Porat, and Gal Ashour. Optimal sequential grouping for robust video scene detection using multiple modalities. *International Journal of Semantic Computing*, 11(02):193–208, 2017. 2, 3

[6] Qingqiu Huang, Yu Xiong, Anyi Rao, Jiaze Wang, and Dahua Lin. Movienet: A holistic dataset for movie understanding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*, pages 709–727. Springer, 2020. 2, 3

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 3