Detecting and Removing Visual Distractors for Video Aesthetic Enhancement

Fang-Lue Zhang, *Member, IEEE,* Xian Wu, Rui-Long Li,
Jue Wang, *Senior Member, IEEE,* Zhao-Heng Zheng, Shi-Min Hu, *Senior Member, IEEE*

Abstract—Personal videos often contain visual distractors, which are objects that are accidentally captured that can distract viewers from focusing on the main subjects. We propose a method to automatically detect and localize these distractors through learning from a manually labeled dataset. To achieve spatially and temporally coherent detection, we propose extracting features at the Temporal-Superpixel (TSP) level using a traditional SVM-based learning framework. We also experiment with end-to-end learning using Convolutional Neural Networks (CNNs), which achieves slightly higher performance than other methods. The classification result is further refined in a post-processing step based on graph-cut optimization. Experimental results show that our method achieves an accuracy of 81% and a recall of 86%. We demonstrate several ways of removing the detected distractors to improve the video quality, including video hole filling; video frame replacement; and camera path re-planning. The user study results show that our method can significantly improve the aesthetic quality of videos.

Index Terms—video distractor, machine learning, visual quality

1 Introduction

In recent years, with the rapid development of digital technology, the public can capture and produce high-quality visual media content, such as photos, videos, digital art and newsletters. Producing such things is no longer limited to professionals who have access to complex and expensive equipment. Therefore, the media content created by all users often has diverse visual qualities. Internet multimedia services, such as on-line video services [1] and multimedia transportation [2] want to automatically assess the aesthetic quality of their acquired materials. The users of capture devices also want to improve their results but often do not know how. Thus, researchers in the multimedia community have proposed intelligent methods to evaluate the visual quality of variant media, such as images [3][4][5], videos [6], paintings [7] and webpages [8][9]. They both utilize state-of-the-art techniques, such as deep learning [4], to rate aesthetic quality and investigate how the intrinsic features contribute to the final perceptual quality [10][11].

For capturing both photos and videos, professionals always carefully plan the scene and exclude all unwanted objects. In contrast, it is hard for amateur users to avoid capturing unintended objects, especially in videos, because users cannot have total control of all dynamic objects. Therefore, it is quite common that a personal video unintentionally captures some other objects that are visually distracting. We call such objects *visual distractors*. These distractors often draw viewers' attention away from the main characters and can make the video unpleasant to watch. The mechanism of visual attention is also brought into sharp focus by the multimedia community [12]. They try to find the connection between the visual saliency and subjective comfort [13] and use it to improve the quality of the generated media content [14]. Additionally, the community provides cues on how to detect the distractors, but these methods cannot be directly used to distinguish the distracting objects from the main objects that should be kept in videos.

Recently, Zhang et al. [15] proposed a method of multi-objective camera path optimization that can simultaneously stabilize the video camera path while cropping out distractors. Distractor detection is done in a trivial way in this work. It uses a heuristic thresholding approach on the local motion contrast of temporal super-pixels extracted from the video. However, the reasons why the human visual system identifies some video objects as distractors could be far more complex. They may involve many factors such as the color distributions of the scene, the motion of the camera, the motion consistency among moving objects, the spatial location of objects, and the length of each object appearing in the video. As we will show later, a simple heuristic method based on individual features is insufficient to achieve high accuracy for video distractor detection.

In this paper, we develop a machine learning-based approach for detecting and localizing visual distractors in video. Instead of imposing heuristic hypotheses on the distractors, we propose to learn a distractor detector from a manually labeled dataset. This is similar to the work of Fried et al. [16], where a LASSO-based learning system was proposed to detect visual distractors in images. However, applying image distractor detection frame-by-frame or on a few key frames does not yield good video distractor detection, given that motion and temporal continuity play a critical role in video saliency analysis. For instance, in an interview video where the camera is static, a person dancing in the remote background may not be a distractor if we only look at a single frame, but it could be quite distracting when watching the full video. In general, video distractor detection requires exploring spatio-temporal features and thus is not a trivial extension of image distractor detection.

- *Fang-Lue Zhang is with the School of Engineering and Computer Science, Victoria University of Wellington, New Zealand.*
- *Xian Wu, Rui-Long Li and Zhao-Heng Zheng are with TNList, Tsinghua University.*
- *Jue Wang is with Megvii (Face++) Research US.*
- *Shi-Min Hu is with TNList, Tsinghua University and Cardiff University. Shi-Min Hu is the corresponding author.*

We explore two different approaches for learning a good distractor classifier from the video distractor dataset we collect. Given that video content has high inter-frame consistency, we first decompose a video into Temporal-Superpixels (TSPs) [17] and extract color, motion, and spatio-temporal distribution features from them. We then use these features to train a traditional SVM classifier. We also experiment with a recent end-to-end deep neural network model SegNet [18], which uses the video frames and feature maps as the multi-channel data layer. Finally, a graph-cut-based post-processing step is applied to refine the detection results. Experimental results show that both learning methods achieve very similar performances, and the best F-score is 83.8% on the test dataset. Finally, we demonstrate three different approaches for removing the distractors from the video, including video hole filling; video frame replacement and camera path replanning.

Our main technical contributions include:

- A new video distractor dataset with manually labeled masks of distractors on each frame.
- New machine learning approaches for learning a distractor classifier.
- Exploration of various approaches for removing distractors for video enhancement.

2 Related work

Much work has been devoted to the visual aesthetic evaluation and measurements of videos. Hammeed et al.[2] and Jang et al. [6] put great efforts into converting problems of subjective video quality assessment into computational ones. Scott et al. improved the computational model by considering personality and cultural influences[19]. Computational quality evaluation models have shown effectiveness in applications such as online video recommendations [1], communications [2] and video editing [20]. With the deepening research, content-based features were introduced into the video quality evaluation models [21]. Then the content-based methods to improve the aesthetic quality were also proposed. Xiang et al. [22] showed how to utilize the video retargeting method to improve the composition of video contents. Temporal information [23] and camera stability [24] were also considered to improve video aesthetic quality. More recently, Lu et al. proposed a spatio-temporally consistent color and structure optimization method[25]. However, in spite of being important factors in video aesthetic quality, the detection of distractors, and their uses to improve video quality have been minimally explored.

Our work is closely related to saliency detection. Based on the theoretical supports from visual psychology, Yu at al.[13] proposed a computational model to obtain object-based saliency. For real-time detection, rule-based methods similar to in [26] are proposed and have achieved good performance. To further improve the precision, machine learning methods have also been applied to predict salient image regions. In [27], initial salient regions or seeds are extracted using models trained on color-space features. Liu et al. [28] processed multiple manual labels to train a CRF model to predict saliency. More recently, a deep neural network improved the capability of the learned models in salient object detection [29][30]. Researchers have started to investigate how to utilize the correlation among the salient regions from different images to improve the detection accuracy [31]. However, these methods are constrained to a single image.

The visual attention mechanism is different for videos because of the dynamic contents. Kim et al.[32] showed that motion features also play an important role (along with appearance features) in the video saliency. In addition, Rudoy et al. [33] found that temporal redundancy makes the video saliency sparsely distributed. Thus, video saliency detection methods heavily rely on both motion features and temporal coherency. Zhai and Shah [34] use spatio-temporal cues to calculate the saliency map on each frame. Rahtu et al. [35] and Liu et al. [36] employed optimization frameworks such as CRFs to detect saliency using energy minimization, since they can be applied to data that are multi-dimensionally correlated. There are also methods that are designed for special video data. With the fast development of the capabilities of multimedia hardware, HDR videos are becoming available to common users. Dong et al. [14] proposed a method to find salient objects in the wide luminous range of HDR videos. To facilitate the saliency analysis in compressed videos, Fang et al. [37] utilized the feature sets extracted from the compressed domain to detect the salient regions. In general, video saliency analysis provides useful cues to the task of distractor detection. However, on its own, it is insufficient to solve the problem, given that not all visually salient objects are distractors.

A similar task to saliency detection is distinctive video object extraction. Tang et al. [38] proposed a method to automatically annotate discriminative objects in weakly labeled videos. Jain et al. [39] represent discriminative video objects at the patch level. Segmentation masks of the extracted objects can be tracked and refined in other frames by the method proposed in [40] and [41]. These methods provide important cues to find objects that can attract the visual attention. The segmentation of moving objects and their dense masks is another related topic. In addition to directly using motion-based dense features such as optical flow for segmentation in the spatio-temporal domain[42], researchers also perform long term motion analysis on tracked sparse feature points to extract the dynamic foreground objects [43][44]. However, these methods cannot tell whether the foreground objects are visually unpleasant.

Abnormal event detection in video is another related topic. Earlier approaches [45] were mostly semi-supervised. Recently, sparse representation was introduced to represent local events for automatic abnormal detection [46]. Adam et al. [47] proposed a system that integrated decisions from multiple cameras. These methods are designed to detect unusual actions of the main objects, and cannot be directly used to detect distractors.

Convolutional neural networks (CNNs) have achieved state-of-the-art performance in many computer vision tasks, including video applications such as large-scale classification [48] and action recognition [49]. By jointly encoding spatio-temporal information in the learning process, 3D convolutional networks [50] have achieved good performance in semantic video short classification. Other works use 2D CNN structures to perform recognition and detection tasks in video by fine-tuning the networks using video frames [51], or by combining video frames and optical flow maps as the input layer [42][52]. Gkioxari and Malik [53] proposed an action tube detection method that learns bounding boxes of actions frame-by-frame using a 2D CNN structure. In this work we propose a method of training CNNs for directly generating pixel-level video distractor maps.

3 The Dataset

To the best of our knowledge, there is no existing video dataset that contains labeled video distractors. Fried et al. [16] provided a distractor dataset for still images only. Existing video saliency datasets or eye-tracking databases [54] contain the labeling of the spatial and temporal parts of a video that will gain viewers' attentions when watching it, but both the main subjects and the distractors can have high visual saliency. On the other hand, even if a video contains only one object, the object could still be a distractor. For instance, if the intention of a video is to capture a static scene, a pedestrian that incidentally passes by should be treated as a distractor. Therefore, the saliency measurement and eye-tracking data are not directly related to the occurrences of distractors.

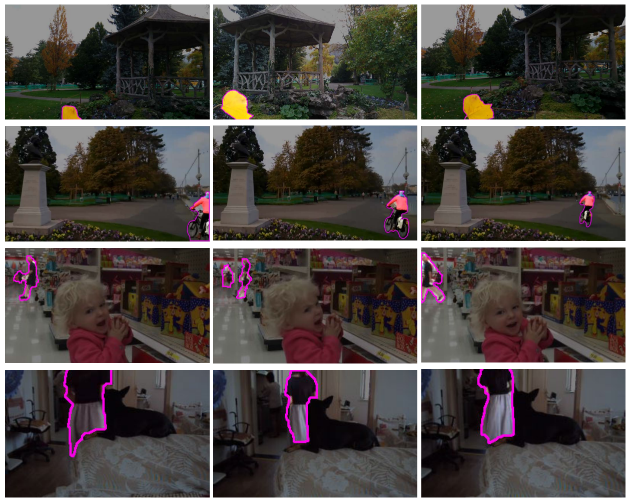


Fig. 1. Manually labeled distractors in our dataset. For each distracting object, an accurate distractor mask is provided on each frame it appears.

3.1 Data collection

In videos that contain multiple objects, both the main object and the distractor can have high saliency compared to the background. Thus, we construct a video dataset that contains manually labeled video distractors. We have collected approximately 2000 personal videos from the internet. Ten participants (5 males and 5 females, aging from 18 - 30) were asked to watch each video, and decide if there were visually intruding distractors that do not appear to be intentionally captured. If such an object exists, they were requested to roughly mask out the distracting objects on one keyframe. There were 102 videos that contained labeled distractors and were used to form our dataset. We then recruited another set of volunteers to create the distractor masks in each video. This was done using the interactive RotoBrush tool in Adobe After Effects, which is an efficient implementation of Video SnapCut [55]. The whole dataset contains over 60,000 video frames. Some labeled examples are shown in Fig. 1.

The reasons for discarding all the videos which do not have distractors are as follows: 1) This dataset is designed to perform the specific task of distractor detection. Because the "distracting" is a relative feeling about objects, only in the videos which contain both main objects and distracting objects, we can get the useful negative and positive samples to train the learning model. 2) The distractors usually exist for a shorter time, and take smaller areas than the main subjects. If we add all these videos in our dataset, there will be 95% of videos containing only negative samples. It is normally not recommended in any machine learning models.

3.2 Data processing

The distractor mask contains per-pixel labels of 0 or 1, indicating whether each pixel belongs to a distractor. However, features extracted at the pixel level are too local to encode the characteristics of distractors. Pixel-level tracking is also erroneous using optical flow. To extract meaningful features, we use the Temporal-Superpixel (TSP) [17] as our basic representation unit, which captures both spatial and temporal local coherence. The input video is decomposed into a set of TSPs. When extracting the TSPs, we resize the videos so that the smaller dimension is 200 pixels. Our dataset contains 0.75 million TSPs in all, including 70,000 positive TSP samples. For the i-th TSP $\mathcal{P}_i$, we denote its covered region on the t-th frame as $\mathcal{P}_i^t$, the starting frame of $\mathcal{P}_i$ as s_i, and the ending frame as e_i. Its temporal length is $l_i = e_i - s_i$.

Next, we assign distractor labels to TSPs according to manually-labeled masks. To decide whether a TSP belongs to a distractor, on each frame in the lifespan of $\mathcal{P}_i$, we calculate the proportion of pixels inside it that have label 1. If more than $0.75 \times l_i$ frames whose distractor proportion is larger than 0.5, we label $\mathcal{P}_i$ as distractor.

4 Feature extraction

As discussed earlier, objects with high visual saliency are not necessarily distractors, but distractors typically have higher saliency than the background of the video, so that they can attract viewers' attentions. Distractors also tend to have different motion characteristics compared to the main objects and background. Moreover, as discovered in previous works [16], [56], objects belonging to some special categories may draw people's attention more easily in videos, such as human faces and cars. Considering all these factors, we extract features related to visual saliency, motion features and object detection labels as part of the overall feature descriptor.

Different from previous works, to preserve the temporal coherency of the detection result, we extract all features at the TSP level. As mentioned earlier, visual saliency is an important indicator for detecting distractors. Given that our feature extraction uses TSP as the basic detection unit, we first propose a method that can estimate visual saliency for TSPs. We then introduce the other features that are used to learn the prediction model.

4.1 TSP saliency

Previous video saliency methods [33], [37] can generate frame-based saliency maps that are temporally smooth. However, for our task, besides rough distribution maps of saliency, we want to know how distinctive each TSP element is. Using the post-processed

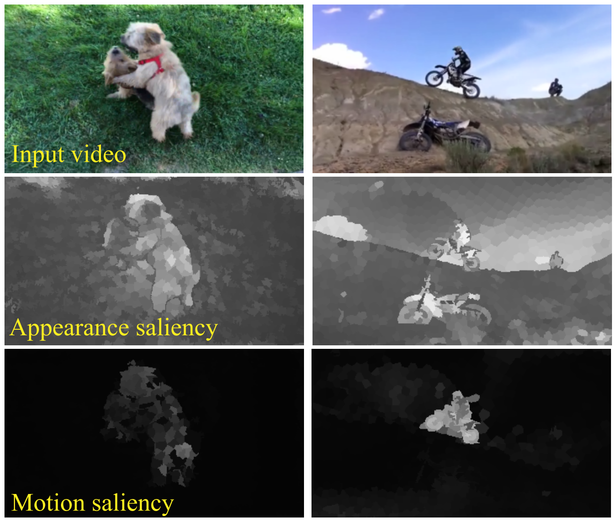


Fig. 2. Some results of TSP-saliency.

results of the frame saliency maps to get each TSP's saliency, may cause errors especially at the boundary regions of moving objects. Therefore, we need a method to directly calculate each TSPs saliency value. We propose a method that considers TSP saliency in both the appearance and motion spaces. Specifically, we use the differences of motion and appearance features between the target TSP and all others to measure its saliency:

$$\begin{aligned} S_a(\mathcal{P}_i) &= \sum_{j=1}^{n} l_j e^{-\frac{d(i,j)}{\sigma^2}} D_a(i,j) \\ S_m(\mathcal{P}_i) &= \sum_{j=1}^{n} l_j e^{-\frac{d(i,j)}{\sigma^2}} D_m(i,j) \end{aligned} \tag{1}$$

where l_j is the length of the lifespan of $\mathcal{P}_j$, $d(i,j)$ is the Euclidean distance in 3D space of (x,y,t), $\sigma = 0.5$. Here, following the common strategy used in the previous saliency extraction method [26], we use $d(i,j)$ to make those spatially and temporally further TSPs contribute less to the current TSP's saliency value. Compared with the uniform weighted scheme, it can achieve better results in the appearance space and in the motion saliency estimation [26]. For example, when the camera is zooming in on a video, the motion vectors of the background TSPs at different parts will have varying orientation. If we use uniform weights for all the TSPs, the background TSPs will also have very high motion saliency value. For motion saliency, the feature distance $D_m(i,j)$ is computed as the L_2 distance between the two average motion vectors of two TSPs. For appearance distance $D_a(i,j)$, it is computed as the L_2 distance between the two average colors in Lab space. $d(i,j)$ and t_j are used to control the contribution of each TSP: the TSPs with longer lifespan and smaller spatial-temporal distance to $\mathcal{P}_j$ will have larger contribution to the calculating of the saliency value. Some example saliency results are shown in Fig. 2.

4.2 Feature set

By visually examining the distractors in the collected dataset, we have the following observations about their characteristics. (1) Visual saliency: Distractors typically have high visual saliency and are thus easily noticed when watching videos. (2) Lifespan: The lifespans of distractors are usually shorter compared to the main objects in a video. (3) Motion contrast: Most distractors have quite different moving speeds or directions compared to the main objects in the scenes. (4) Spatial distribution: Normally, the distractors appear and disappear near the border of video frames. (5) Semantic information: Some specific objects can more easily draw attention, such as human faces and cars. Based on these observations, we extract the following set of features:

1) Visual saliency. We extract both the motion and appearance saliency of each TSP according to Eqn. 1.
2) Lifespan. Denote the frame number of the whole video as L, we use the following features for $\mathcal{P}_i$:
$$t_i^1 = l_i/L, t_i^2 = s_i/L, t_i^3 = e_i/L,$$
where l_i, s_i and e_i are the lifespan, starting and ending frames of the TSP, respectively.
3) Motion. We extract the following motion features for each TSP:
a) Moving distance in its lifespan:
$$m_i^1 = p_i^s - p_i^e,$$
where p_i^s and p_i^e are the starting and ending center positions of $\mathcal{P}_i$, respectively.
b) Moving speed, calculated as the offset of the TSP's center position between adjacent frames.
c) Relative speed, calculated as dividing the average moving speed by the background moving speed. We use the maximum, minimum, mean and median values in both b) and c) as features.
d) The spatial variance of the center trajectory, to describe if the movement is stable and regular. It is calculated as:
$$d_i = \sum_{t \in l_i} (p_i^t - \overline{p_i})^2.$$
4) Spatial distribution. We compute the normalized distance between the center of the TSP and the center of the frame. We then use the maximum, minimum, mean and median value of it during the lifespan of the TSP as features.
5) Semantic labeling. The object detection results of "car", "person" and "face" using methods mentioned in [56] are used as features. If in the lifespan of a TSP, it is detected as within one of those objects, the label will be set to 1 in that dimension.

Inspired by previous work in image and video saliency detection, we also add appearance features. First, each feature is calculated on every frame. For every TSP, we then use the max, min, mean and median values of that feature in its covered region as features:

- Direction controllable pyramids in 3 directions and 4 scales [56].
- RGB values, RGB probabilities, and the probability of each color, computed from 3D color histograms of the image filtered by a median filter at 6 different scales [56].
- Horizon detection results using the method in [56].

By combining all above feature descriptors together, we extract a 177 dimensional feature vector for each TSP.

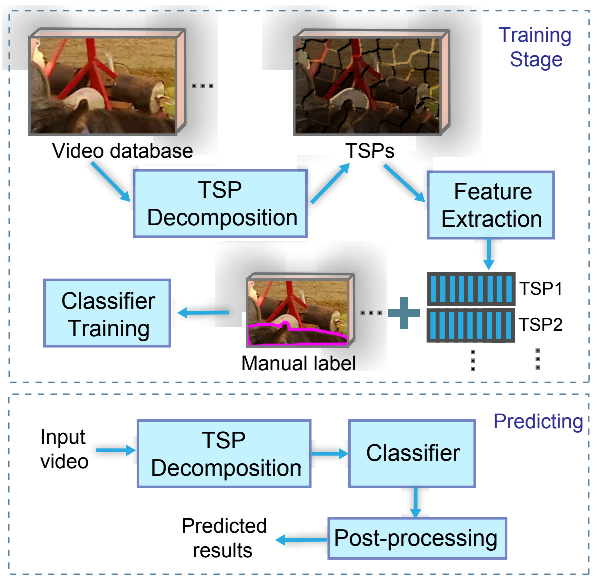


Fig. 3. Overview of TSP-level learning and predicting framework.

5 Training

We explore two different learning and classification models including the TSP-level supporting vector machine (SVM) classifier and the pixel-level end-to-end CNN.

5.1 TSP-level SVM

First, we train an SVM classifier with RBF kernel directly on the extracted features from labeled TSPs, as shown in Fig. 3. We randomly split the dataset into a training set that contains 80% videos and a test set with the remaining ones. All the TSPs with positive distractor labels in the training set are used as positive samples. We then randomly choose the same number of negative TSPs from the training videos to make a balanced training sample set. When training the RBF-SVM model, we use grid search to get the optimized parameters γ and C. We follow the method introduced in [57] to do the 5-fold cross validation. On the training data for each iteration, we randomly pick a subfolder as validation data, and then we perform a grid search on the model trained by other training data. In the classification stage, the input video is first decomposed into TSPs, and each is then tested by the trained SVM model.

5.2 Pixel-level CNN

Most existing deep neural networks are designed for processing images, especially in end-to-end learning tasks such as detection and segmentation [18][58]. Tran et al. [50] proposed a 3D convolutional network for video classification. However, this structure does not produce the pixel-level labeling that is required in our task. Inspired by the work of [42], which uses original frames and corresponding optical flow maps as input layers for the CNN, we combine video frames with extracted feature maps as input layers for our task. We choose to use SegNet [18], the state-of-the-art image segmentation network to predict a distractor map in every video frame.

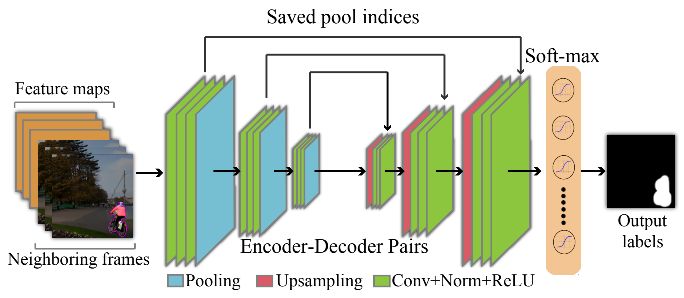


Fig. 4. The structure of the SegNet model. Encoder-decoder pairs at different scales are trained. Our multi-channel input layer contains both video frames and TSP-level feature maps.

As shown in Fig. 4, the SegNet model is composed of several decoding and encoding layers to capture features at different spatial scales. The data layer supports multi-channel inputs for images. We can use the neighboring frames as different channels to the current frame, which helps capture appearance and local motion characteristics of distractors. To encode long-range temporal features, we generate several pixel-wise feature maps as extra channels by propagating the TSP-level feature values down to pixels. The final input layers contain 12 channels with a size of 200×200 pixels, which are:

- 5 grayscale frames centered at the current frame (5 channels).
- The lifespan length of the TSP that each pixel belongs to (1 channel).
- Average motion speed of each TSP that each pixel belongs to (2 channels).
- The spatial distance to the center of the frame (1 channel).
- Pixel-level saliency (2 channels, including the motion and appearance saliency).
- The object detection labels at pixel-level (1 channel).

One problem of our training dataset is the unbalanced numbers of positive and negative samples. For a more balanced input, we only use video frames that contain distractors when training the neural network. We use 5 encoder-decoder pairs in our experiment. Most parameters are set as with the defaults in [18]. For the i-th encoder-decoder pair, we use 32×2^i kernels. We chose a batch size of 8. We use the SGD method as our training algorithm, with 150K epochs. The trained Caffe model of this 5-layer SegNet for distractor detection is included in the supplementary materials.

Note that the features are still extracted from the full-length video. In the prediction stage, the input frames are first decomposed into TSPs to generate multi-channel feature maps. Given a test video, we use videos from $8 \times 8 = 64$ overlapping sliding windows as inputs for the SegNet model. Each window has a size of 240×160 and centers from (120, 80) to (360, 240). The final label for each pixel is achieved by voting. If the pixel is detected as the distractor in more than half of the windows including it, it will be marked as a distractor. To maintain spatio-temporal coherency, we assign distractor labels to TSPs if more than 50% of its pixels are marked as a distractor. Using sliding windows, the noises in detecting results can be reduced by averaging the possibilities from different windows. Thus, we can reduce the number of false-positive results compared with just using the full-frame to test. In our tests, we find it achieves a similar recall but an 1-2% improvement in accuracy. Therefore, we choose the sliding-window strategy when testing our model. In our experiments, 4×4 windows have a lower accuracy by 0.5%; while 16×16 windows obtain a similar performance with 4 times the computation costs. Thus we choose 8×8 windows as our default window setting.

5.3 Post-processing

The prediction result from the classifiers are further refined in a post-processing step. A distractor usually contains a set of TSPs, and one would expect strong spatial labeling continuity. Neighboring TSPs with similar motion and appearance features should have the same label. Based on this observation, we employ a graph-cut based method to further improve the detection results.

We build a graph where each TSP is a node. If a pair of TSPs $(\mathcal{P}_i, \mathcal{P}_j)$ are spatially adjacent in more than 50% of their lifespans, we mark them as neighbors and add a link between them, as shown in Fig. 5. The set of all such neighboring pairs is denoted as $\mathcal{N}$. It is a graph-based optimization problem. So we follow the objective definition in [59] to solve our problem. Each TSP is given a label of 0 (non-distractor) and 1 (distractor) to minimize the following graph energy:

$$E(L) = \sum_{\mathcal{P}_i \in \Omega} D_{\mathcal{P}_i}(L_i) + \sum_{(\mathcal{P}_i, \mathcal{P}_j) \in \mathcal{N}} T_{(i,j)}(L_i, L_j), \quad (2)$$

where Ω is the set of all the TSPs, L_i is the label of $\mathcal{P}_i$. The data term $D_{\mathcal{P}_i}$ is derived from the classification confidence of that sample. For SVM-based TSP level method, the confidence can be measured as the distance to the classification boundary of the RBF-SVM:

$$D_{\mathcal{P}_i}(L_i) = |\sum_{j}^{N} \alpha_j K(x_j, x_i) + b|^{-1} * |L_i^o - L_i|, \quad (3)$$

where L_i^o is the prediction label provided by the classifier, $K(x_j, x_i)$ is the RBF-kernel function, x_j is the $j-th$ supporting vector and α_j is the weight. For results from SegNet, we use the proportion of the pixels that are labeled as distractor in a TSP as its confidence value. By doing so, TSPs with lower classification confidences will have lower cost to switch labels.

The coherence term $T(i,j)$ in Eqn. 2 encourages neighboring TSPs that are similar to have the same label. Here we use the appearance and motion features to measure the similarity as:

$$T_{(i,j)}(L_i, L_j) = \begin{cases} 0, & \text{if } L_i = L_j \\ \|\mathcal{F}_i - \mathcal{F}_j\|^{-1}, & \text{otherwise} \end{cases} \quad (4)$$

Where we use $\mathcal{F}_i$ to represent the feature vector of a TSP. It is worth mentioning that since there are connected edges between TSPs with different lifespans, the distance measurements are not metrics. We thus use alpha-beta swap to solve the graph cuts problem as proposed in [59]. As Fig. 6 shows, graph cuts optimization removes errors caused by noisy TSPs, and at the same time recovers missing parts of distractors, leading to more accurate distractor labeling.

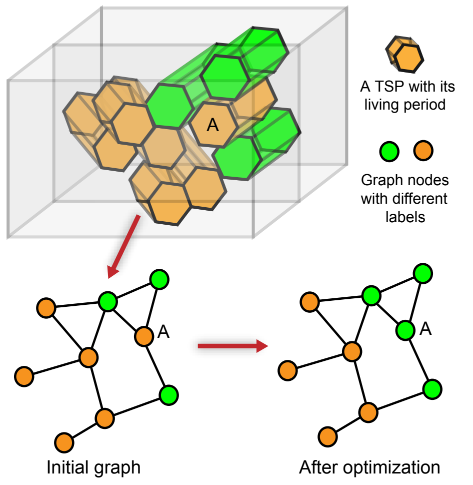

A TSP with its living period
Graph nodes with different labels
A
A
A
Initial graph
After optimization

Fig. 5. Illustration of using graph-cut optimization for detection refinement. Each node represents a TSP. Node A is initially predicted as non-distractor. Since it has similar features to its neighboring nodes, its label changes after optimization.

6 Experimental Results

6.1 Distractor prediction

We use 80% of the videos in our datasets to train the classifier. This contains approximately 60,000 positive TSP samples and 48,000 frames. The remaining 20% of videos are used as the test dataset. When testing, we use all the positive and negative samples in these testing videos. Note that there is a serious imbalance in the numbers of positive and negative samples. Normally, 85%-90% of TSPs in a video are negative samples, and only approximately 10%-15% of TSPs are distractors. This suggests that in our experiments, recall is more important than accuracy when measuring the performance. For example, even a classifier marks all the input testing samples as negative, the accuracy will be above 0.85, but the recall is 0. On the other hand, if the accuracy is too low, there will be too many false alarms, and the post-processing can minimally improve the final results. Thus, to evaluate the performance using a more balanced strategy, we use the F-score as the first measurement. Then we compare their recalls if they have same F-scores.

We conduct a 5-fold cross-validation, and the results are shown in Tab. 1. The SegNet model achieves a slightly better performance than SVM with a similar recall and a higher accuracy. We suspect that with more training data, the SegNet model would eventually perform better than the SVM model. The graph-cut optimization in the post processing step also significantly improves the detection results.

Post-process (Original)	SVM	SegNet
Recall	0.863(0.842)	0.859 (0.841)
Accuracy	0.806(0.721)	0.814 (0.731)
F-score	0.834(0.777)	0.838 (0.782)

TABLE 1
Performance of our method on our dataset.

In terms of computational efficiency, our SegNet model runs on a computer with Intel i7 CPU and an NVidia GTX980 display card, which takes 28 hours for training and 30 minutes for examining all

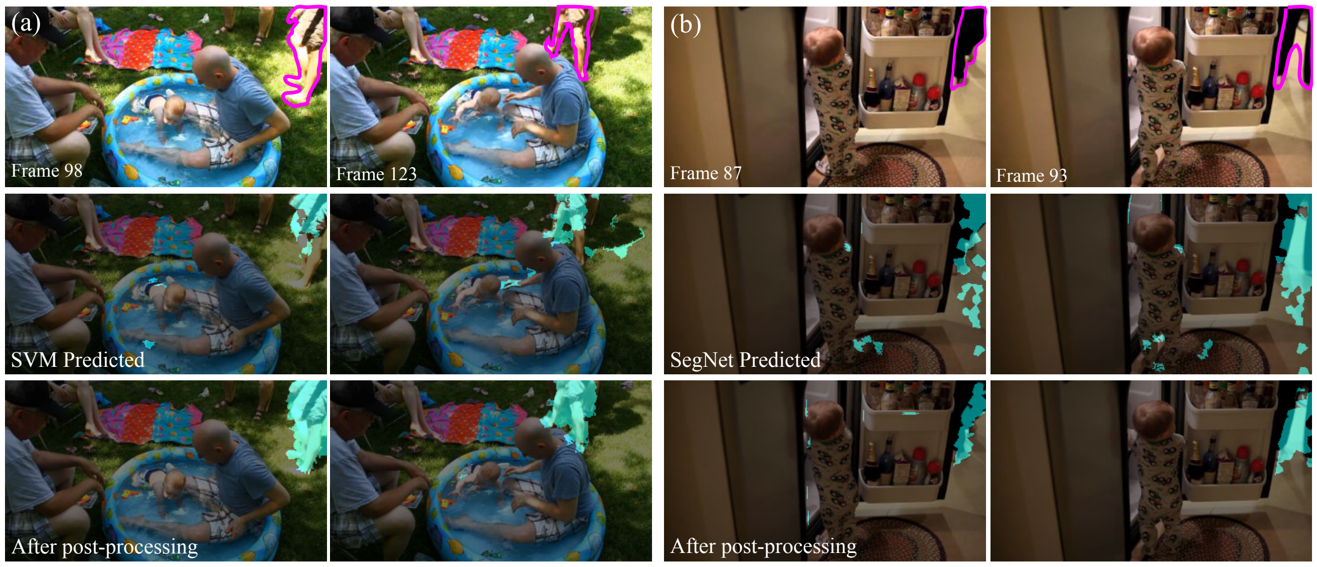


Fig. 6. Some post-processing results. The detected distractors before and after post-processing are marked in blue. The manually labeled ground-truth are marked in pink.

testing video frames. We use LibSVM [60] to train the SVM classifier, which takes about 50 minutes for learning. The prediction time is about 0.5 seconds per frame with pre-segmented TSPs. For a video resized to 320×240, the average TSP segmentation time is about 30 minutes using the code provided in [17].

6.2 Comparisons and discussion

6.2.1 *TSP-level classification*

In addition to SVM, we have also tested a number of other classifiers such as Random Forest, Bayesian Network, and Adaboost. We also varied the size of the training and testing datasets to examine the sensitivity of different methods to the training data size. For fair comparisons, we use the same training and testing datasets for all methods. The results are shown in Tab. 2. Although Adaboost yields the highest recall, its accuracy is also the lowest, indicating that its detection result is quite noisy. It also has a lower F-score. In the end we choose to use SVM given its overall good performance.

We conduct a more detailed comparison between SVM and LASSO [16], which was previously used for image distractor detection. For LASSO, we use different thresholds for deciding whether a TSP is a distractor. We also adjust the parameters in SVM to produce results with different recall and precision values. The performance curves are shown in Fig. 7. Considering that recall is more important for our task, we thus choose SVM as our classifier.

In the experiments reported in Tab. 2, we also test different sizes of training sets when learning the classifiers, to see the data quality of our dataset. The results suggest that the overall performance of each method improves with more training data when the training dataset is small. When the dataset size reaches 60%, the improvement becomes much smaller and the performances of different methods become stable.

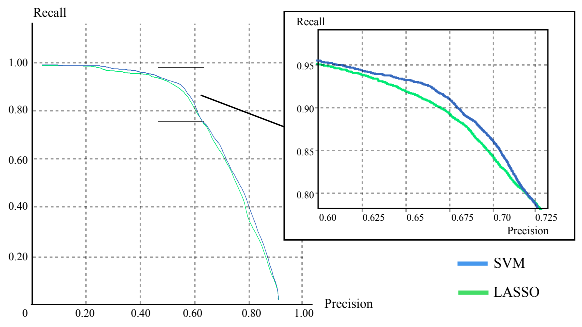


Fig. 7. Recall and precision curves of LASSO and SVM.

Classifier	data(%)	20	40	60	80
Adaboost	Recall	0.71	0.76	0.89	0.92
	Accuracy	0.64	0.68	0.67	0.66
	F-score	0.67	0.71	0.76	0.77
Bayes	Recall	0.52	0.67	0.63	0.71
	Accuracy	0.67	0.71	0.71	0.72
	F-score	0.58	0.68	0.67	0.72
Lasso	Recall	0.67	0.63	0.63	0.69
	Accuracy	0.64	0.72	0.74	0.73
	F-score	0.65	0.67	0.68	0.70
Random Forest	Recall	0.55	0.73	0.73	0.72
	Accuracy	0.75	0.79	0.80	0.81
	F-score	0.63	0.75	0.76	0.76
SVM	Recall	0.48	0.71	0.85	0.84
	Accuracy	0.71	0.68	0.70	0.72
	F-score	0.57	0.69	0.76	0.78

TABLE 2
Performance of different methods using datasets with different sizes.

6.2.2 *Experiments on CNN*

We have experimented with different training strategies for the SegNet model, as summarized in Tab. 3. The results suggest that only using video frames in the input layer generates worse results. As mentioned earlier, this is because long-term temporal

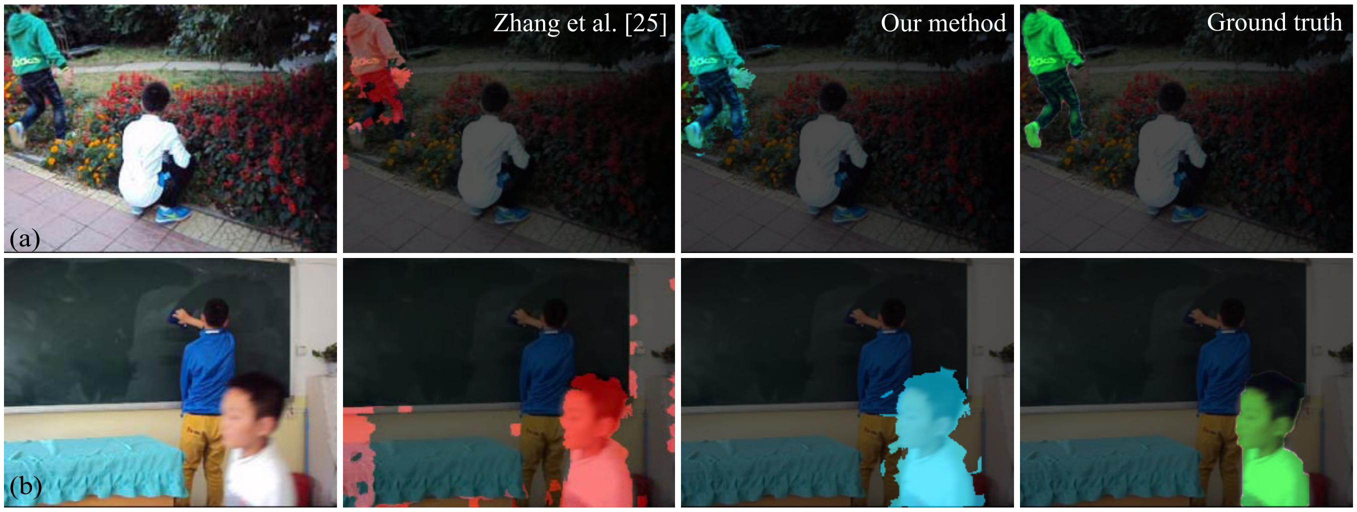


Fig. 8. Comparison with the rule-based method in [15]. The detected TSPs belonging to distractors are highlighted in red (Zhang et al.[15]) and blue (our method), respectively. The ground truth masks are shown in green.

information is not captured in the input data. The feature maps thus play an important role in distractor detection. On the other hand, having too little (0/3) or too many (7) frames in the input layer does not improve the detection, which suggests that local motion information can be well captured in 5 frames.

	Recall	Accuracy	F-score
Only 3 frames	0.812	0.471	0.642
Only 5 frames	0.823	0.584	0.701
Only 7 frames	0.816	0.592	0.707
Only feature maps	0.742	0.783	0.762
3 frames + feature maps	0.764	0.697	0.734
5 frames + feature maps	0.841	0.731	0.782
7 frames + feature maps	0.750	0.681	0.716

TABLE 3
Performance changes of our CNN model by using different training data.

6.2.3 *Compare with rule-based method and image-based method*

Zhang et al. [15] proposed a rule-based distractor detection method. In this method, the decision is made by checking the TSP's saliency value and position. If a TSP has high saliency and is near the frame boundary, it will be labeled as a distractor. As shown in the example in Fig. 8(a), this method will miss those distractors with relatively low saliency thresholds. However, if the threshold is lower, there will be many false alarms as shown in Fig. 8(b). Our learning-based method can utilize more clues for distractor detection, which will avoid ad-hoc parameter tweaking. The performances of the two methods for the examples shown in Fig. 8 are reported in Tab. 4. We also show the comparison results on the full dataset. Considering near 90% of the TSPs in input videos are non-distractors, recall is much more important when evaluating performance. It suggests that our method outperforms the rule-based method on recall with an acceptable accuracy. We also compared our method with the image distractor detection method proposed in [16]. However, since the image-based method cannot utilize the motion information, which has shown importance in the video distractor detection, it fails to successfully extract the distracting object, based on the lowest recall.

Input	Data	Rule-based	Image-based	Ours
(a)8082 TSPs	Accuracy	0.82	0.86	0.79
	Recall	0.37	0.18	0.97
(b)7195 TSPs	Accuracy	0.79	0.85	0.81
	Recall	0.51	0.26	0.86
Full (728K TSPs)	Accuracy	0.76	0.87	0.86
	Recall	0.47	0.24	0.81

TABLE 4
(a)(b)Quantitative results on the two examples in Fig. 8. The numbers of TSPs in those two videos are also shown. The comparison results for the full dataset are shown at the bottom. videos.

6.3 Feature importance

To explore how much each feature contributes to the detection results, we conduct a series of experiments where we use different feature combinations in each. We group the features into the following categories: 1. Visual saliency features (VS); 2. Spatial and temporal positions (POS); 3. Motion features (MOV); 4. Appearance features (APP); 5. Object detection labels (OBJ). In each experiment, we use 4 out of these 5 groups of features to train our classifier and test the performance. The results are shown in Tab. 5.

The results show that the spatial-temporal distribution of objects (POS group) plays an important role in the distractor prediction. Removing these features leads to the largest drops in both recall and accuracy. This is intuitively natural given that most of the distractors are in the border region of video frames and have relatively short lifespans. The appearance features are important to distinguish the distractors from the main objects. The object detection features also have significant contribution in detecting distractors. They carry semantic-level information related to visual attention, which can help to extract distractors like the van in Fig. 13(a).

Interesting, when removing the motion-based features, the accuracy decreased by 3.2%, but the recall increases. That is because

Removed		Performance	Decrease(%)
VS	Recall	0.827	1.26
	Accuracy	0.707	1.94
POS	Recall	0.772	8.31
	Accuracy	0.694	3.74
MOV	Recall	0.871	-3.44*
	Accuracy	0.698	3.19
APP	Recall	0.765	9.14
	Accuracy	0.710	1.53
OBJ	Recall	0.781	7.24
	Accuracy	0.716	0.69

TABLE 5
Performance of different feature sets in which one group of features are removed.

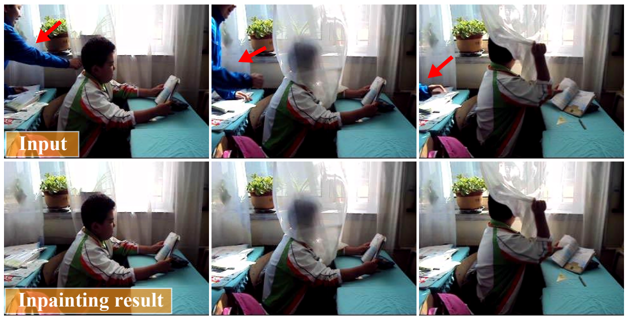


Fig. 10. Removing distractors (highlighted by red arrows) by inpainting.

in some videos, the distractors do not have significant motion, thus motion features tend to group them with the background. But the decreased accuracy shows that the motion features can help avoid more false alarms. Considering the imbalanced data distribution, we choose to keep the motion features. Finally, we find that visual saliency features only improve the results marginally. That is because the distractors and main objects both have high saliency, thus they are not quite separable in saliency space.

7 Distractor Removal

Given the video distractor detection results, we propose different methods to remove or exclude distractors from an input video to improve its aesthetic quality.

7.1 Frame replacement

Distractors could seriously deteriorate an interview video. Inspired by the recent tools for editing interview videos [61], we use our approach to detect and remove the frames containing distractors from an interview video, and then generate smooth transitions to fill the temporal gaps. Given the frames indices with distractors, we use Berthouzoz et al.'s method [61] to find seamless hidden transition frames by building a transition graph. The frames without distractors in the original video are used to form the optimal path in which the body and facial components have the smoothest in between changes. Different from this method, we further consider the case where the camera is moving while recording the interview video. We estimate the homography transformations between frames by tracking feature points in the background, and align video frames before processing. An example is shown as

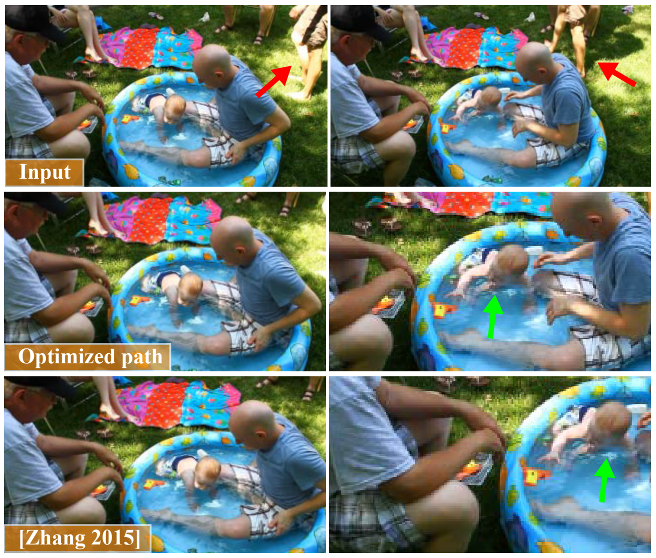


Fig. 11. Re-planning the camera path to exclude distractors (highlighted by red arrows) from the final video frames.

in Fig. 9. Compared with inpainting, replacing frames can avoid hole filling artifacts when a large region of the foreground object is occluded by the distractor, such as the middle frame in Fig. 9. This method is limited to long interview videos where the foreground and background are relatively stable.

7.2 Inpainting

A straightforward way to remove the detected distractors is to apply video completion, or hole filling methods in regions that are detected as distractors. Similar to previous image hole filling systems, in order to avoid masking errors, we expand the distractor mask to produce a larger region to fill using the method in [62]. Such an example is shown in Fig. 10. This method usually works well when the distractor is relatively small and similar textures can be found in other parts of the video frame to fill in the holes.

7.3 Camera Path re-planning

Zhang et al. [15] propose a camera path re-planning method to exclude the distractors from the final video frames. We further extend this idea by constraining the position of the main object when re-planning the camera path. The camera path is represented by a sequence of homography matrices between frames. The objective of the original L1-optimization framework in [15] is to produce a new homography matrix sequence that are smooth, while ensuring that the distractors are excluded. Please refer to the original paper for more details. Because their approach does not consider the final position of the main subject, it often becomes too close to the border after cropping, as shown in Fig. 11. To avoid this problem, we add an extra energy term on the most salient object (excluding the distractors) to constrain it to be at a better position. According to cinemagraphy rules, the main object should lie in the intersection points of 4 one-third lines. Thus, we add slack variables on each frame to optimize the final distance with the nearest intersection:

$$(-\theta_x^t, -\theta_y^t)^T < P^t c^t - q^t < (\theta_x^t, \theta_y^t)^T \quad (5)$$

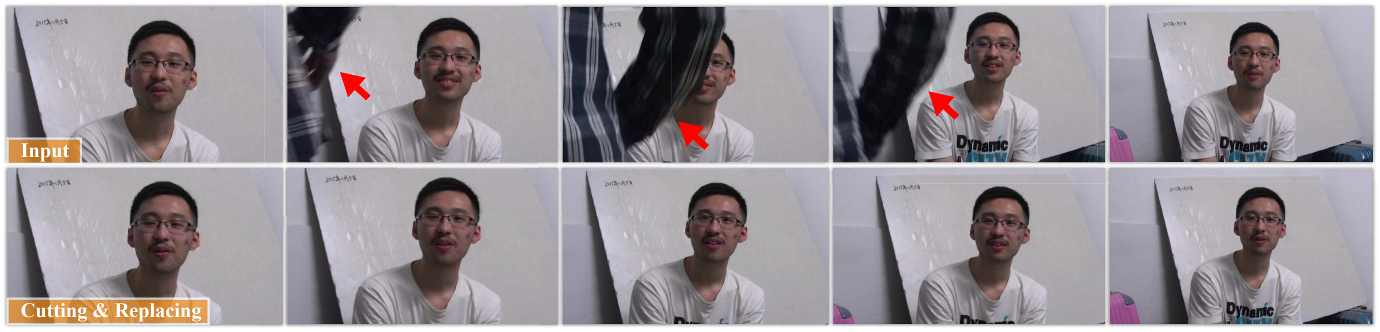


Fig. 9. Replacing video frames that contain distractors (highlighted by red arrows) in an interview video.

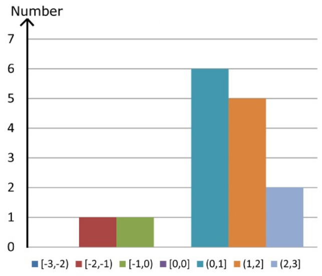


Fig. 12. Distribution of average scores for the aesthetic quality comparison between original video and our improved version.

Score	A. Replacement	B. Inpainting	C. Re-planning	All
Mean	1.48	0.69	1.13	1.09
Std	0.30	1.14	0.76	0.74
T-Test	A vs. B	A vs. C	B vs. C	All*
P-Value	0.036	0.58	0.048	0.0027
H	1	0	1	1

TABLE 6
Statistics of the user study results. Paired-samples t-tests are performed between each pair of the 3 removal methods. $H = 1$ indicates that the scores are different at significance level $\alpha = 0.05$. * This is the result of one-sample t-test.

where P^t is the transformation to be optimized for frame t, c^t is the center of the most salient object, q^t is the nearest intersection point. Slack variables θ will be minimized in the L1-optimization framework (see [15]). In the example shown in Fig. 11, the baby is detected as the main object in the video (shown by the green arrow). Our path re-planning method generates a cropped video with a better composition than the original method of [15].

7.4 User Study and Discussion

A user study was performed to evaluate our proposed distractor detection and removal method, to see whether it can improve the video aesthetic quality. For the method of frame replacement, since it only works on interview videos, we first randomly selected 5 out of the 16 interview videos in the dataset. Then for the completion and path re-planning, we randomly choose 10 videos out of the other 86 videos. Then fifteen pairs of videos were prepared. Each pair contains the original video and the result after distractor removal. We invited twenty participants, with no expertise in photography and visual psychology. The original and result videos were randomly placed next to each other for each pair when they were played for the participants. We did not tell them anything about the operation we performed to change the video. To eliminate bias, the participants were selected from different age and gender groups. There were 10 males and 10 females in both age 18-30 and age 31-45 groups. They were asked to assign an integer rank from -3 to 3, to indicate how much one was more pleasing than the other. They were asked to assign a 0 if they think the qualities were equal. A positive score meant the video on the right was more appealing than the one on the left and vice versa.

The user study outcome is shown in Fig. 12, where we show the distribution of the average scores of 15 video pairs after changing the orders by putting the original video on the left. It can be seen that our method effectively improves the aesthetic quality, with 86.7% percent of videos judged to be improved to differing extents. The statistics are shown in Tab. 6. The mean of the scores is 1.09. Since the scores are given for each video pair, to test whether the scores show a significant difference in terms of the visual quality, we perform one sample t-test. Here, we assume that the average score of the video pairs with equivalent quality should be zero. Then the one sample t-test shows that the improvement is significant.

There are two typical failure cases where the users give negative scores for our results. One is the result from the optimal path planning, where the video is over-cropped due to the large size of the distractor. The second failure case is from the inpainting method. There are noticeable artifacts in the inpainting area for a shaking video. From the bottom table of Tab. 6, we find that the inpainting method gets significantly lower scores than the others. That is because the noticeable artifacts in some inpainting examples are more unpleasant. In contrast, the replacement is only performed on interview videos. The replanning method is based on warping, so they hardly introduce new artifacts if the camera is not very shaky. Therefore, the methods of replacement and path re-planning are recommended to remove distractors.

We may also fail to improve the visual quality because of changing the original semantic structure. Like the examples in Fig. 8 (right) and 10, although the ”distractors” are unintentionally captured, they actually have semantic connections with the behaviors of the main objects. If we just remove them, the semantic structure will not be preserved. To determine whether they have semantic relationships with the behaviors of the main objects, we need better perceptual models to describe the features. It is another limitation of our method and the important future work of distractor detection.

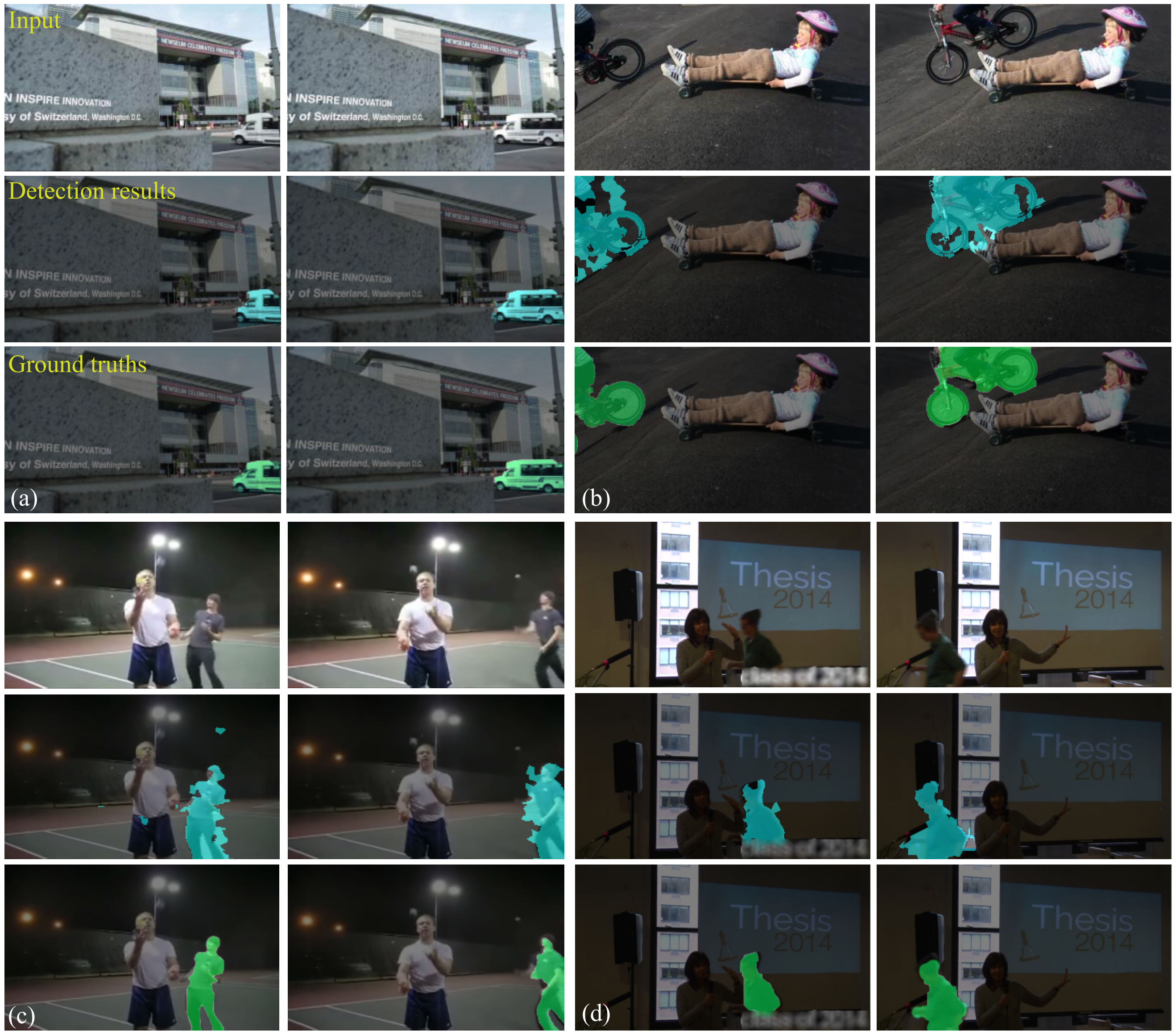


Fig. 13. More distractor detection results. Detected distractors are highlighted in blue. The ground truth are shown in green.

8 Conclusion

In this paper, we have proposed a learning-based method to detect visual distractors in videos. To learn a classifier, we also build a dataset that contains manually labeled distractors. The experimental results show that our learned classifier achieves high precision and recall for this dataset, and the results can be further improved in a post-processing step. This method can be applied for aesthetic video enhancement and in intelligent video editing tools to improve the visual quality of videos.

As a future work, we plan to explore more features for improving the performance, especially those features related to long-range temporal coherence. We also plan to collect more data containing video distractors in order to create a better prediction model that can cover more types of distractors. Distractor detection also has the potential to improve many important video processing tasks even when distractors are hard to remove. For example, by just using non-distractor regions or objects, we can obtain a more robust camera path for video deblurring and stabilization, and make content-based video retrieval more accurate. A better aesthetic quality evaluation model can be established considering distractors.

Acknowledgments

This work was supported by Research Establishment Grant of Victoria University of Wellington (Project No. 8-1620-216786-3744) the Natural Science Foundation of China (Project No. 61521002, 61133008), Research Grant of Beijing Higher Institution Engineering Research Center, and Tsinghua University Initiative Scientific Research Program.

References

[1] L. Sun, X. Wang, Z. Wang, H. Zhao, and W. Zhu, “Social-aware video recommendation for online social groups,” *IEEE Transactions on Multimedia*, vol. 19, no. 3, pp. 609–618, 2017.

[2] A. Hameed, R. Dai, and B. Balas, “A decision-tree-based perceptual video quality prediction model and its application in fec for wireless multimedia communications,” *IEEE Transactions on Multimedia*, vol. 18, no. 4, pp. 764–774, April 2016.

[3] R. Datta, D. Joshi, J. Li, and J. Wang, “Studying aesthetics in photographic images using a computational approach,” *ECCV*, pp. 288–301, 2006.

[4] X. Lu, Z. Lin, H. Jin, J. Yang, and J. Z. Wang, “Rating image aesthetics using deep learning,” *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 2021–2034, Nov 2015.

[5] C. Li, A. C. Loui, and T. Chen, “Towards aesthetics: A photo quality assessment and photo selection system,” in *ACM MM*. ACM, 2010, pp. 827–830.

[6] S. Jang and J. S. Lee, “On evaluating perceptual quality of online user-generated videos,” *IEEE Transactions on Multimedia*, vol. 18, no. 9, pp. 1808–1818, Sept 2016.

[7] C. Li and T. Chen, “Aesthetic visual quality assessment of paintings,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 3, no. 2, pp. 236–252, 2009.

[8] O. Wu, Y. Chen, B. Li, and W. Hu, “Evaluating the visual quality of web pages using a computational aesthetic approach,” in *ACM international conference on Web search and data mining*. ACM, 2011, pp. 337–346.

[9] O. Wu, H. Zuo, W. Hu, and B. Li, “Multimodal web aesthetics assessment based on structural svm and multitask fusion learning,” *IEEE Transactions on Multimedia*, vol. 18, no. 6, pp. 1062–1076, June 2016.

[10] Z. Wu, Y. Huang, and L. Wang, “Learning representative deep features for image set analysis,” *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 1960–1968, Nov 2015.

[11] Q. Li, W. Lin, J. Xu, and Y. Fang, “Blind image quality assessment using statistical structural and luminance features,” *IEEE Transactions on Multimedia*, vol. 18, no. 12, pp. 2457–2469, Dec 2016.

[12] H. Kim and S. Lee, “Transition of visual attention assessment in stereoscopic images with evaluation of subjective visual quality and discomfort,” *IEEE Transactions on Multimedia*, vol. 17, no. 12, pp. 2198–2209, Dec 2015.

[13] J. G. Yu, G. S. Xia, C. Gao, and A. Samal, “A computational model for object-based visual saliency: Spreading attention along gestalt cues,” *IEEE Transactions on Multimedia*, vol. 18, no. 2, pp. 273–286, Feb 2016.

[14] Y. Dong, M. T. Pourazad, and P. Nasiopoulos, “Human visual system-based saliency detection for high dynamic range content,” *IEEE Transactions on Multimedia*, vol. 18, no. 4, pp. 549–562, April 2016.

[15] F.-L. Zhang, J. Wang, H. Zhao, R. R. Martin, and S.-M. Hu, “Simultaneous camera path optimization and distraction removal for improving amateur video,” *IEEE Transactions on Image Processing*, vol. 25, p. 99, 2015.

[16] O. Fried, E. Shechtman, D. B. Goldman, and A. Finkelstein, “Finding distractors in images,” in *IEEE CVPR*, 2015, pp. 1703–1712.

[17] J. Chang, D. Wei, and J. W. Fisher, “A video representation using temporal superpixels,” in *IEEE CVPR*, 2013, pp. 2051–2058.

[18] V. Badrinarayanan, A. Handa, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling,” *arXiv preprint arXiv:1505.07293*, 2015.

[19] M. J. Scott, S. C. Guntuku, W. Lin, and G. Ghinea, “Do personality and culture influence perceived video quality and enjoyment?” *IEEE Transactions on Multimedia*, vol. 18, no. 9, pp. 1796–1807, Sept 2016.

[20] J.-H. Choi and J.-S. Lee, “Automated video editing for aesthetic quality improvement,” in *ACM MM*. ACM, 2015, pp. 1003–1006.

[21] L. Anegekuh, L. Sun, E. Jammeh, I. H. Mkwawa, and E. Ifeachor, “Content-based video quality prediction for hevc encoded videos streamed over packet networks,” *IEEE Transactions on Multimedia*, vol. 17, no. 8, pp. 1323–1334, Aug 2015.

[22] Y.-Y. Xiang and M. S. Kankanhalli, “Video retargeting for aesthetic enhancement,” in *ACM MM*, 2010, pp. 919–922.

[23] M. L. Gleicher and F. Liu, “Re-cinematography: Improving the camerawork of casual video,” *ACM TOMM*, vol. 5, no. 1, p. 2, 2008.

[24] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum, “Full-frame video stabilization with motion inpainting,” *IEEE TPAMI*, vol. 28, no. 7, pp. 1150–1163, 2006.

[25] S. P. Lu, B. Ceulemans, A. Munteanu, and P. Schelkens, “Spatio-temporally consistent color and structure optimization for multiview video color correction,” *IEEE Transactions on Multimedia*, vol. 17, no. 5, pp. 577–590, May 2015.

[26] M.-M. Cheng, G.-X. Zhang, N. J. Mitra, X. Huang, and S.-M. Hu, “Global contrast based salient region detection,” in *IEEE CVPR*, 2011, pp. 409–416.

[27] J. Kim, D. Han, Y.-W. Tai, and J. Kim, “Salient region detection via high-dimensional color transform,” in *IEEE CVPR*, 2014, pp. 883–890.

[28] T. Liu, Z. Yuan, J. Sun, J. Wang, N. Zheng, X. Tang, and H.-Y. Shum, “Learning to detect a salient object,” *IEEE TPAMI*, vol. 33, no. 2, pp. 353–367, 2011.

[29] R. Zhao, W. Ouyang, H. Li, and X. Wang, “Saliency detection by multi-context deep learning,” in *IEEE CVPR*, 2015, pp. 1265–1274.

[30] G. Li and Y. Yu, “Deep contrast learning for salient object detection,” in *IEEE CVPR*, 2016, pp. 478–487.

[31] D. Zhang, D. Meng, and J. Han, “Co-saliency detection via a self-paced multiple-instance learning framework,” *IEEE TPAMI*, vol. 39, no. 5, pp. 865–878, 2017.

[32] H. Kim, Y. Kim, J.-Y. Sim, and C.-S. Kim, “Spatiotemporal saliency detection for video sequences based on random walk with restart,” *IEEE TIP*, vol. 24, no. 8, pp. 2552–2564, 2015.

[33] D. Rudoy, D. B. Goldman, E. Shechtman, and L. Zelnik-Manor, “Learning video saliency from human gaze using candidate selection,” in *IEEE CVPR*, 2013, pp. 1147–1154.

[34] Y. Zhai and M. Shah, “Visual attention detection in video sequences using spatiotemporal cues,” in *ACM MM*, 2006, pp. 815–824.

[35] E. Rahtu, J. Kannala, M. Salo, and J. Heikkilä, “Segmenting salient objects from images and videos,” in *ECCV*, 2010, pp. 366–379.

[36] T. Liu, N. Zheng, W. Ding, and Z. Yuan, “Video attention: Learning to detect a salient object sequence,” in *IEEE ICCV*, 2008, pp. 1–4.

[37] Y. Fang, W. Lin, Z. Chen, C.-M. Tsai, and C.-W. Lin, “A video saliency detection model in compressed domain,” *IEEE TCSVT*, vol. 24, no. 1, pp. 27–38, 2014.

[38] K. Tang, R. Sukthankar, J. Yagnik, and L. Fei-Fei, “Discriminative segment annotation in weakly labeled video,” in *IEEE CVPR*, 2013, pp. 2483–2490.

[39] A. Jain, A. Gupta, M. Rodriguez, and L. S. Davis, “Representing videos using mid-level discriminative patches,” in *IEEE CVPR*, 2013, pp. 2571–2578.

[40] D. Zhang, O. Javed, and M. Shah, “Video object segmentation through spatially accurate and temporally dense extraction of primary object regions,” in *IEEE CVPR*, 2013, pp. 628–635.

[41] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. V. Gool, M. Gross, and A. Sorkine-Hornung, “A benchmark dataset and evaluation methodology for video object segmentation,” in *IEEE CVPR*, 2016, pp. 724–732.

[42] K. Fragkiadaki, P. Arbelaez, P. Felsen, and J. Malik, “Learning to segment moving objects in videos,” in *IEEE CVPR*, 2015, pp. 4083–4090.

[43] P. Ochs, J. Malik, and T. Brox, “Segmentation of moving objects by long term video analysis,” *IEEE TPAMI*, vol. 36, no. 6, pp. 1187–1200, 2014.

[44] F.-L. Zhang, X. Wu, H.-T. Zhang, J. Wang, and S.-M. Hu, “Robust background identification for dynamic video editing,” *ACM Transactions on Graphics*, vol. 35, no. 6, p. 197, 2016.

[45] D. Zhang, D. Gatica-Perez, S. Bengio, and I. McCowan, “Semi-supervised adapted hmms for unusual event detection,” in *IEEE CVPR*, vol. 1, 2005, pp. 611–618.

[46] C. Lu, J. Shi, and J. Jia, “Abnormal event detection at 150 fps in matlab,” in *IEEE ICCV*, 2013, pp. 2720–2727.

[47] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz, “Robust real-time unusual event detection using multiple fixed-location monitors,” *IEEE TPAMI*, vol. 30, no. 3, pp. 555–560, 2008.

[48] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, “Large-scale video classification with convolutional neural networks,” in *IEEE CVPR*, 2014, pp. 1725–1732.

[49] L. Wang, Y. Qiao, and X. Tang, “Action recognition with trajectory-pooled deep-convolutional descriptors,” in *IEEE CVPR*, 2015, pp. 4305–4314.

[50] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3d convolutional networks,” in *IEEE ICCV*, 2015, pp. 4489–4497.

[51] C. Gan, N. Wang, Y. Yang, D.-Y. Yeung, and A. G. Hauptmann, “Devnet: A deep event network for multimedia event detection and evidence recounting,” in *IEEE CVPR*, 2015, pp. 2568–2577.

[52] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, “Beyond short snippets: Deep networks for video classification,” in *IEEE CVPR*, 2015, pp. 4694–4702.

[53] G. Gkioxari and J. Malik, “Finding action tubes,” in *IEEE CVPR*, 2015, pp. 759–768.

[54] P. K. Mital, T. J. Smith, R. L. Hill, and J. M. Henderson, “Clustering of gaze during dynamic scene viewing is predicted by motion,” *Cognitive Computation*, vol. 3, no. 1, pp. 5–24, 2011.

[55] X. Bai, J. Wang, D. Simons, and G. Sapiro, “Video snapcut: robust video object cutout using localized classifiers,” *ACM Transactions on Graphics (TOG)*, vol. 28, no. 3, p. 70, 2009.

[56] T. Judd, K. Ehinger, F. Durand, and A. Torralba, “Learning to predict where humans look,” in *IEEE ICCV*, 2009, pp. 2106–2113.

[57] C.-W. Hsu, C.-C. Chang, C.-J. Lin *et al.*, “A practical guide to support vector classification,” 2003.

[58] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *IEEE CVPR*, 2015, pp. 3431–3440.

[59] A. Delong, A. Osokin, H. N. Isack, and Y. Boykov, “Fast approximate energy minimization with label costs,” *IJCV*, vol. 96, no. 1, pp. 1–27, 2012.

[60] C.-C. Chang and C.-J. Lin, “Libsvm: A library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

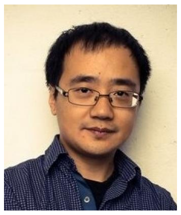
[61] F. Berthouzoz, W. Li, and M. Agrawala, “Tools for placing cuts and transitions in interview video,” *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, p. 67, 2012.

[62] A. Newson, A. Almansa, M. Fradet, Y. Gousseau, and P. Pérez, “Video inpainting of complex scenes,” *SIAM Journal on Imaging Sciences*, vol. 7, no. 4, pp. 1993–2019, 2014.

Fang-Lue Zhang is a lecturer at Victoria University of Wellington. He received his doctoral degree from Tsinghua University in 2015 and bachelor degree from Zhejiang University in 2009. His research interests include image and video editing, computer vision and computer graphics. He is a member of ACM and IEEE.

Xian Wu is a phd student at the Tsinghua University. He received his BS degree from the Tsinghua University in 2015. He is currently interested in computer vision, image and video processing and computer graphics.

Rui-Long Li is a master student at the Tsinghua University. He received his BS degree from the Tsinghua University in 2016. He is currently interested in image and video analysis and processing.



Jue Wang is now at Megvii (Face++) Research US. Before that he has been a Principle Research Scientist at Adobe Research for 9 years. He received his B.E. (2000) and M.Sc. (2003) from Department of Automation, Tsinghua University, and his Ph.D (2007) in Electrical Engineering from the University of Washington. He received Microsoft Research Fellowship and Yang Research Award from University of Washington in 2006. His research interests include image and video processing, computational photography, computer graphics and vision. He is a senior member of IEEE and a member of ACM.

Zhao-Heng Zheng is an undergraduate student at the Tsinghua University. He is currently interested in computer graphics, including image/video processing and animation.

Shi-Min Hu is currently a professor in the department of Computer Science and Technology, Tsinghua University, Beijing. He received the PhD degree from Zhejiang University in 1996. His research interests include digital geometry processing, video processing, rendering, computer animation, and computer-aided geometric design. He has published more than 100 papers in journals and refereed conference. He is Editor-in-Chief of Computational Visual media, and on editorial board of several journals, including IEEE Transactions on Visualization and Computer Graphics, Computer Aided Design and Computer & Graphics.