High Performance Video Condensation System

Jianqing Zhu; Shikun Feng; Dong Yi; Shengcai Liao, *Member, IEEE*; Zhen Lei; Stan Z. Li, *Fellow, IEEE*

***Abstract*—Video synopsis or condensation, is a smart solution for fast video browsing and storage. However, most of the existing methods work offline, where two main phases are required. The first phase is to prepare tubes and background images. The second phase is to rearrange tubes and stitch them into backgrounds. However, with a long video sequence, the first phase is memory consuming for data storage, and the second phase is computationally expensive to rearrange all tubes simultaneously. In order to overcome these problems, we propose a high performance video condensation system based on an online content-aware framework. The online framework transforms the optimization problem of tube rearrangement into a stepwise optimization problem. Therefore, it can condense video with much less memory and higher speed than the offline framework. With the aid of this transformation, the proposed system can process input videos and produce condensed videos simultaneously, thus it is suitable for real-time endless surveillance videos. Meanwhile, the online mechanism allows users directly visit the condensation video that has been generated. Moreover, the content-aware mechanism makes the proposed system able to automatically determine the duration of a condensed video. Finally, the proposed system uses GPU and multi-core techniques to improve the speed. Extensive experiments that validate the high efficiency of the system are presented.**

***Index Terms*—online background generation, moving object segmentation, GPU acceleration, video condensation system, video storage, video surveillance.**

I. Introduction

IN the past decade, there is an explosive growth of surveillance video data in the world. This situation brings about great demands for fast video browsing and storage technologies in public security field. However, how to fast browse and effectively extract useful information from the huge video data still remain challenging problems.

The easiest approaches about efficient browsing include fast forwarding [1] and video skimming [2]. In those methods, videos are fast browsed by skipping several frames between selected frames. However, skipped frames may cause important contents missing. The adaptive methods of skipping frames [3, 4] are thus proposed. Such methods skip frames in periods of low activity, while keeping frames in periods of high activity. A survey of fast video browsing is presented in [5]. Other approaches are video abstraction [6, 7], which use key frames as a synopsis to represent an original video. However, this key frame representation may lose the dynamic effect of a video sequence. A survey of video abstraction is given in [8]. Overall, the smallest processing unit of the above approaches is an entire frame which means that they only condense original videos in time domain, but neglect redundances in spatial domain. Therefore, they can not achieve high condensation ratio.

Stan Z. Li is the corresponding author. The authors are from Center for Biometrics and Security Research and National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, 95 Zhongguancun East Road, Beijing 100190, China. E-mail: {jqzhu, szli}@cbsr.ia.ac.cn

Alternatively, the space-time video montage [9] analyzes both the spatial and temporal information distribution of an original video. By taking the visually informative space-time portion as the smallest processing unit, it packs all these portions together to maximize the visual information of a condensed video. However, the video condensed by this method has obvious seams and information loss.

The ribbon carving based method [10] considers a ribbon as the smallest processing unit. The so-called ribbon can be thought as a flexible frame without activity. This method repeatedly removes ribbons until there is no ribbon in an original video. However, its condensation ratio is low, and may fail when adjacent objects having different speeds and directions. Moreover, it always creates vertical or horizontal visible seams in condensed videos.

A significant progress in this field is video synopsis [11–13]. The goal of video synopsis is to produce a shortened video sequence by condensing an original video in temporal and spatial domains. As a tube (tube is a frame-sequence of an object) based approach, video synopsis enables users to browse a day-long video recording in just a few minutes by creating a summary of all activities. As shown in Fig. 1, the video synopsis framework includes the online phase and the response phase. The online phase is firstly performed to record background images and extract tubes from an original video using foreground segmentation and tracking algorithms. Therefore, this online phase is actually a preprocessing step used to collect tubes and backgrounds. In the response phase, an energy function is minimized to determine the play time of objects (tube rearrangement) in synopsis video and time-lapse background video is constructed, then the objects are stitched into the selected backgrounds to generate a synopsis video. Because the first phase is performed on the whole input video,

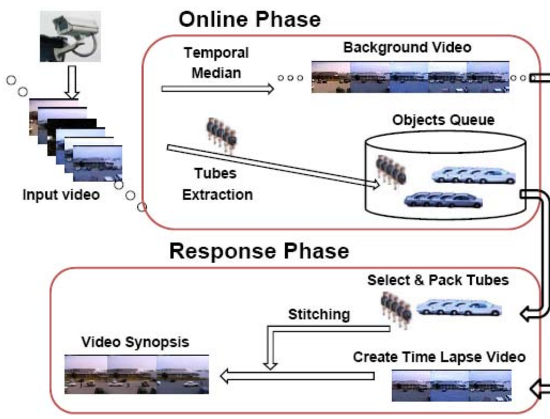


Fig. 1. The framework of video synopsis [11]. The image is derived from [11]. It is essentially an offline framework because it relies on a preprocessing step performed in the first phase.

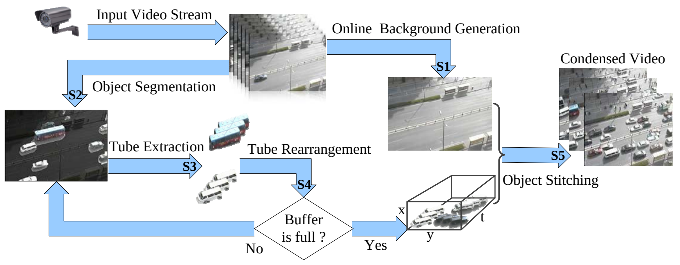


Fig. 2. The online content-aware video condensation framework. This framework does not need any preprocessing step. Based on a memory buffer, it can parallelly perform S4 and S5, thus can process an input video and generate the condensed video simultaneously.

therefore it is essentially an offline processing framework.

Following the video synopsis framework, several improved methods are proposed in [14–16]. These methods have achieved some improvements with respect to speed or tube rearrangement effect. However, this two-phase offline framework still has some drawbacks when it handles a very long video sequence:

- It needs huge memory to store all tubes and backgrounds first or have to delete some objects.
- Its processing speed is low, because it deals with all data in one time.
- Its query efficiency is low, because it must perform tube rearrangement and object stitching algorithms to generate the corresponding synopsis video for each query.
- The duration of a synopsis video is determined manually rather than by the content of an input video, which is impractical because users may not know the activity density of an input video beforehand.

In order to overcome the drawbacks of huge memory cost and slow speed when the offline framework deals with long videos, the most direct way is to manually divide a long input video into several short sequences, and then use the offline framework to process each short sequence. However, in practice, this method may face two problems: (1) An object's trajectory may be divided into different sequences which will degrade user experience; (2) Different sequences usually have different activity density, which causes a headache for users to determine an adequate length for each of the synopsis video.

In this paper, a high performance video condensation system is proposed to overcome the aforementioned drawbacks. The paper is built upon our preliminary work: online content-aware video condensation framework reported in [17]. The online content-aware video condensation framework transforms the optimization problem of tube rearrangement in traditional video synopsis approaches into a stepwise optimization problem. The main novelty of this framework is the online processing manner, which is able to keep processing a long input video, while at the same time incrementally produce the condensed video. As shown in Fig. 2, our online framework only includes one phase, which does not need a preprocessing step used to prepare tubes and background images.

In order to construct a more practical video condensation system, a number of techniques are introduced in this paper to enhance the system on speed and memory consumption, including:

- *An Online Background Generation Method*. This method generates a time-lapse background image by averaging frames in a time interval and updates it over time. The memory cost of this method is low and the produced background can reflect the background changes over time.
- *A Faster Moving Object Segmentation Method*. The scale invariant local ternary pattern (SILTP) feature based background subtraction algorithm [18] is applied to achieve effective moving object segmentation.
- *A Multi-thread Implementation Framework*. The online content-aware video condensation framework is divided into tube generation, tube rearrangement and object stitching modules, which are parallelly implemented.
- *An Effective Memory Buffer Design*. The memory buffer is based on the producer-consumer model used to control the memory balance between different multi-thread modules of the system;
- *GPU and Multi-core Acceleration Strategies*. The GPU and multi-core techniques are used to accelerate the processing speeds of SILTP based moving object segmentation and object stitching, respectively.

The advantages of the high performance video condensation system will be validated by experiments, which are summarized as follows:

- *Fast Speed.* On an 8 cores @ 2.66 GHz computer with a GPU (Nvidia GeForce GTX 285), the system processing speed achieves 530 to 660 fps for videos with 320×240 resolution, and even for high resolution (704×576) videos, it achieves 100 fps.
- *Low Memory Cost.* The system can online processing videos, which does not require huge memory to preserve all tubes and backgrounds.
- *High Condensation Ratio.* The system can achieve a higher condensation ratio than the ribbon carving based method [10].
- *High Query Efficiency.* In each query, the system allows users to directly get the condensed videos that have been generated rather than executing tube rearrangement and object stitching each time.
- *Ability to Process Endless Video.* The system can process input videos and produce condensed videos simultaneously, thus it is able to deal with real-time endless surveillance video.
- *Adaptivity.* The system can produce a condensed video with adaptive condensation ratio. This is more reasonable than setting by manual as in [11].
- *Compatibility.* The system is able to condense videos in offline or online modes.

The rest of this paper is organized as follows. Section II describes the details of the online content-aware video condensation framework, including online background generation, moving object segmentation, sticky tracking, tube rearrangement and object stitching, respectively. Section III introduces the high performance system, including software design and acceleration strategies. Section IV presents experimental results to show the superiority of the proposed system. Section V concludes this paper.

II. Online Content-aware Video Condensation Framework

In this section, an online content-aware video condensation framework is introduced. Then, the details of the online content-aware framework are introduced, including online background generation, moving object segmentation, sticky tracking, tube rearrangement and object stitching.

A. From Offline To Online

The smallest processing unit of video synopsis is tube. A tube is defined as the frame-sequence of the same object in a video. The main idea of video synopsis is to remove activity-less frames, and rearrange tubes in video frames to make objects that appear sequently in an original video can appear simultaneously in the shortened video. As illustrated in Fig. 1, the video synopsis framework includes two phases. The online phase is first performed in real time during video capturing and recording. The response phase is performed following a user query to generate synopsis videos. The online phase can be seen as a preprocessing step for an original video to prepare tubes and backgrounds which must be performed first, and then a synopsis video is produced in the respond phase which needs a user query. Consequently, it is essentially an offline framework because a synopsis video is produced until a preprocessing step was performed in the first phase.

Let us focus on the response phase where tubes are rearranged in temporal domain. Assuming that tubes and backgrounds have been prepared, then the problem of tube rearrangement can be viewed as a constrained optimization problem. Tube rearrangement is to reassign the start-time label of each tube so that multiple tubes originally appearing in different times can be displayed simultaneously under some constraints. The constraints proposed in [11] include keeping maximum activity, consistency with background, keeping chronological order and avoiding collisions (occlusions) between tubes. The optimization problem can be formulated as minimizing the following energy function:

$$E(\ell) = \sum_{i \in \boldsymbol{Q}} E_u(\ell_i) + \alpha \sum_{i,j \in \boldsymbol{Q}} E_p(\ell_i, \ell_j), \tag{1}$$

where $\boldsymbol{Q}$ denotes the whole tube set, and ℓ_i denotes the start-time label of tube i which takes a value from the time label set:

$$\mathcal{L}_{\text{Offline}} = \{1, \cdots, M\}, \tag{2}$$

where M denotes the number of frames in a condensed video and is actually set by users. E_u and E_p denote the unary and pairwise energy functions, respectively. α is used to control the weight of the pairwise energy function E_p. Specific formulations of E_u and E_p will be discussed later. We can find that, the optimization of Eq. (1) involves all tubes in one time. Minimizing such problem is very time-consuming when $|\boldsymbol{Q}|$ or M is huge. Besides that, it requires much room to prepare all tubes and backgrounds. For example, the memory cost of a video sequence with 10 hours duration (30fps $\times$ 3600s $\times$ 10 $= 1,080,000$ frames, 320 $\times$ 240 pixel resolution, 3 channels) is $1,080,000 \times 320 \times 240 \times 3 \times 8\text{bit} \approx$ 232 GB. Assuming that the foreground pixel ratio is 1% and backgrounds are recorded with a 10-frame interval, therefore, the offline framework must cost $232 \times 1\% = 2.32$ GB to save tubes and $232 \times 0.1 = 23.2$ GB to save backgrounds, respectively. In other words, the offline framework has the drawbacks of slow speed and huge memory cost when it deals with long videos.

The online content-aware video condensation framework is used to overcome the drawbacks in the offline framework. As illustrated in Fig. 2, the online framework only includes one phase, rather than two phases. It consists of five main steps:

S1: Background image generation: generate a background image using the online background generation method.
S2: Moving object segmentation: segment moving objects using the scale invariant local ternary pattern (SILTP) feature based background substraction algorithm [18].
S3: Tube extraction: extract tubes using the sticky tracking algorithm.
S4: Tube rearrangement: decide optimal start-time labels of tubes using the online tube filling algorithm and push the rearranged tubes into a memory buffer.
S5: Object stitching: If the buffer is full, stitch tubes in this buffer into the background image (S1) using the modified Poisson image editing method [11].

Note that, based on a memory buffer, the online framework is able to parallelly perform S4 and S5 which can process an input video and generate the condensed video simultaneously. Therefore, it is a real online video condensation framework.

The online property of the online content-aware framework is first discussed, and the content-aware property will be discussed later. The basic idea of the online framework is to transform the optimization problem of Eq. (1) into a stepwise optimization problem of Eq. (3), which only involves a subset of the whole tube set; that is, it determines the start-time label of each tube in the subset one by one. For the current tube i, its start-time label ℓ_i is calculated by minimizing the following energy function:

$$E(\ell_i) = E_u(\ell_i) + \alpha \sum_{j \in \boldsymbol{Q'}} E_p(\ell_i \mid \ell_j), \tag{3}$$

where $\boldsymbol{Q'} \subset \boldsymbol{Q}$ denotes the subset of processed tubes; $\ell_{(\cdot)}$ takes a value from a much smaller label set:

$$\mathcal{L}_{\text{Online}} = \{1, \cdots, n\}, \tag{4}$$

where n denotes the number of frames in a temporary condensation space, with $n \ll M$. ℓ_j is the known start-time label of the processed tube j. The optimization of Eq. (3) deals with a tube subset at each step, which is a good approximation to the optimal solution of Eq. (1). However, compared with Eq. (1), the label set of Eq. (3) is much smaller ($n \ll M$) and the number of involved tubes is fewer ($|Q'| < |Q|$), which significantly reduces the time cost of tube rearrangement. Moveover, there is no need to store all tubes $\boldsymbol{Q}$ in memory but just a subset $\boldsymbol{Q'}$. Therefore, the optimization of Eq. (3) can achieve high speed with low memory cost.

B. Online Background Generation

A condensed video sequence consists of tubes and backgrounds; that is, each frame in a condensed video sequence is generated by stitching moving objects into a background image. In practice, the number of frames in a condensed video M is much smaller than the number of frames in the corresponding original video N ($M \ll N$). Therefore, a background selection mechanism is needed in video condensation application.

The selected background images should meet two properties [12]: (1) Property-I, it should reflect background changes over time, such as the alternation of day and night. (2) Property-II, it should be related with video activities; that is, backgrounds containing more moving objects are preferred. In fact, the two properties are conflicting. In order to address this trade-off, [12] combines two temporal histograms with a weight to meet the two properties. However, this method is an offline approach which needs to store all N backgrounds first, requiring huge storage space. In order to reduce storage cost, the online principal background selection (OPBS) method is proposed in [19]. As an online version of background selection method, the OPBS method needs to store n backgrounds ($n < M \ll N$, a typical value is $n = 500$).

In this paper, an online background generation method is proposed to coordinate with the online framework. It generates a background image by averaging frames in a time interval (in our experiments, set this time interval to be 3,000 frames) and updates it over time. The advantage of this method is that it only needs to store one background image.

The background image generated by this simple method still meets the two properties under the online content-aware video condensation framework. Because it is updated over time, therefore it naturally meet Property-I. As discussed above, the tube rearrangement is realized in a stepwise way, thus the corresponding condensed video is produced in a stepwise way too. In each step, a set of tubes are stitched into a generated background image. In the period which contains more activities, the online content-aware framework is more likely to trigger stitching operation (this is called content-aware ability of the online framework which will be discussed later) to produce stitched frames that in a condensed video, thus background images that appear in the period with more activities are more often to be used in the condensed video. Therefore, it also meets Property-II.

C. Moving Object Segmentation

Because the smallest processing unit of video condensation is tube, the moving object segmentation must be performed before tube extraction. In [11, 17], a graph cut based object segmentation method is used to get a smooth segmentation of moving objects. The graph cut based object segmentation method used in [11, 17] is a simplification of background cut [20]. In [17], the color-based unary term is the color difference between the current image and the estimated background image and the background image is produced by using the mixture of Gaussian (MoG) [21] method. Since the graph cut based object segmentation method is computationally expensive, the multi-thread graph cut method is proposed in [17] to accelerate the processing speed.

TABLE I
THE ACCURACY COMPARISON BETWEEN THE SILTP [18] AND GRAPH CUT (GC) BASED [17] OBJECT SEGMENTATION METHODS. THE TEST VIDEO SEQUENCES COME FROM CHANGE DETECTION 2014 [22].

Sequence	Recall(%)		Precision(%)		F-score(%)	
	SILTP	GC	SILTP	GC	SILTP	GC
highway	**95.87**	95.65	**59.03**	56.81	**73.07**	71.28
PETS2006	**97.08**	91.59	**54.19**	35.18	**69.56**	50.83
pedestrians	**95.22**	86.67	37.72	**46.68**	54.04	**60.68**
peopleInShade	**98.96**	97.64	**55.68**	46.90	**71.26**	63.37
Average	**96.78**	92.89	**51.66**	46.39	**66.98**	61.54

In this work, the scale invariant local ternary pattern (SILTP) feature based background subtraction algorithm [18] is applied to achieve effective moving object segmentation. As reported in [18], the SILTP feature based background subtraction method can achieve better segmentation results than the mixture of Gaussian (MoG) approach [21], while the processing speed of SILTP is comparable with that of MoG.

Here, we also report a comparison of the SILTP [18] and the graph cut based [17] object segmentation methods. The accuracy comparisons are shown in Table I, where Recall=$TP/(TP+FN)$, Precision=$TP/(TP+FP)$ and F$-$score=2 $\cdot$ Recall $\cdot$ Precision/(Recall + Precision). TP, FP and FN are true positives (true foreground pixels), false positives and false negatives (false background pixels), respectively.

From Table I, we can find that both SILTP and graph cut based object segmentation methods achieve high recall rates. This illustrates they can completely segment moving objects. Moreover, the precisions of SILTP based method are higher than that of the graph cut based method in most cases, therefore, the SILTP based method has better F-score performance. However, the precision rates of the two methods are relatively low. This is resulted from a conservative background decision threshold setting, which favors more foreground pixels so that the extracted objects will be complete for video condensation. The segmentation results of SILTP and graph cut based methods are shown in Fig. 3, where we can see that both of the two methods can completely segment moving objects. However, the graph cut based algorithm used in [17] sometimes causes obvious under-segmentation because the color difference based unary term is sensitive to illumination variations, while the SILTP based method has a better segmentation due to its illumination invariant nature in design. Furthermore, we propose a GPU accelerated SILTP based moving object segmentation method, which will be described in Section III-C1, with a processing speed comparison to [17].

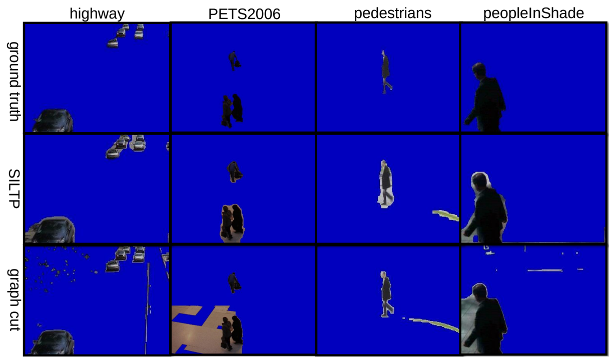


Fig. 3. The segmentation results of SILTP and graph cut based methods.

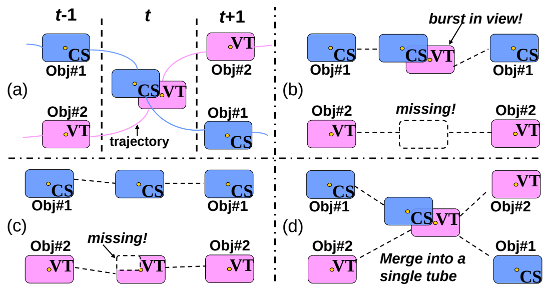


Fig. 4. Occlusion between two objects.

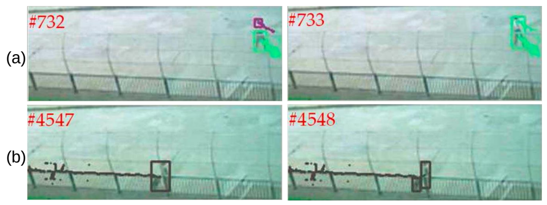


Fig. 5. Sticky tracking can amend poor object segmentation for the case of over-segmentation, with a surveillance video from [10]. If two or more parts have the same color, they are considered as a single tube.

D. Sticky Tracking

If moving objects have been segmented, a tracking algorithm is applied to “connect” the same object appearing in different frames for tube extraction. Many tracking methods [23, 24] have been proposed, however, those methods may not be entirely suitable for video condensation. The following example illustrates the problems. As illustrated in Fig. 4(a) with two objects, where Obj$\#2$ is occluded by Obj$\#1$ at frame t. Fig. 4(b) shows the result of a common blob tracking method [24]: Two tubes are generated, but both are not good enough. Because the merged blob is matched to Obj$\#1$ at frame t, something not belonging to Obj$\#1$ will burst into view in a condensed video. Meanwhile, no blob is matched to Obj$\#2$, Obj$\#2$ will disappear abruptly and then appear again in the view. As a result, both tubes will cause blinking effect, thus deteriorating user experience. Fig. 4(c) is the result of an ideal tracking method that produces the most accurate result. However, such an optimal tracker still can not completely suitable for video condensation application: A part of Obj$\#2$ is lost due to the occlusion at frame t, which also causes blinking effect.

The sticky tracking strategy is used to reduce blinking effect in a condensed video for better visual effects. It is based on the following idea: if occlusions happen to two or more tubes, they will be merged into a single tube, as if they are sticking together in Fig. 4(d). Note that the goal of sticky tracking is very different with that of traditional tracking methods. The key point is to launch merging before matching. That is, if two or more tubes in the object list are matched to the same blob by the nearest object center distance at frame t, the two tubes will be merged and treated as one tube from frame t on.

In addition to reducing the blinking effect caused by occlusions between objects, sticky tracking presents another advantage, being able to amend poor object segmentation (under-segmentation and over-segmentation). The case of under-segmentation can be treated as object occlusion, which can be solved by sticky tracking. Fig. 5 shows some sticky tracking results for the other case of over-segmentation. In Fig. 5(a), the head and the body of the same person were considered as two objects due to the over-segmentation at frame 732. When the segmentation becomes correct at frame 733, the head and body are merged into a single tube by using sticky tracking. In Fig. 5(b), one person with a suitcase splits into two objects at frame 4548 due to occlusions by the fence, and sticky tracking still successfully considers them as the same object.

Moreover, sticky tracking has an ability to keep the chronological order of objects when they are close to each other (e.g. taking a conversation): If the distance between objects is less than a threshold (10 pixels in our experiments), they will be merged into one tube by sticky tracking, therefore, their chronological order will be naturally preserved.

E. Tube Rearrangement

The role of tube rearrangement is to decide each tube's start-time label. The online content-aware tube filling algorithm is the core of tube rearrangement. The main idea of online tube filling comes from the Tetris game, suggesting to deal with tubes one by one, rather than all in one time as in [11]. In Tetris, a player is encouraged to manipulate the tetris to create a horizontal line without gaps, then such line can be cleared. If the player is smart enough, the game can go on forever. Similarly, the online tube filling algorithm treats tubes as tetris, and regards a 3D condensation space as the playing field of video condensation. If the playing field is saturated, the rearranged tubes are pushed into the object stitching stage and so the current playing field is cleared. In the following, our job is to design a smart player for the tube filling game.

1) Objective Function Construction: In Eq. (1) and (3), the objective functions consist of unary and pairwise energy functions. In [11], the unary energy function includes activity cost (the penalty of losing tubes in a condensed video) and background inconsistency cost (the penalty of the inconsistency among tubes and background images), while the pairwise energy function includes collision cost (the penalty of occlusion between tubes) and temporal consistency cost (the penalty of the temporal inconsistency between tubes). Based on these four cost terms, it is time consuming to find an optimal solution for Eq. (3); therefore, according to the characteristics of the online content-aware framework, we define a simplified objective function:

$$E(\ell_i) = \sum_{j \in \boldsymbol{Q'}} E_c(\ell_i \mid \ell_j), \tag{5}$$

where $E_c(\cdot|\cdot)$ represents the collision cost between two tubes, and its definition will be discussed later.

Firstly, we can find that Eq. (5) does not include the unary energy function $E_u(\cdot)$, where it is set as $E_u(\cdot) \equiv 0$. The motivations are follows: (1) Considering the activity cost is to punish the case that some tubes disappear in a condensed video, therefore, if all tubes are forcibly stitched into the condensed video, there is no need to consider the activity cost. (2) The background inconsistency among tubes and background images is not significant in the online framework. Because the background image generation and tube rearrangement both are performed in online way that the time difference

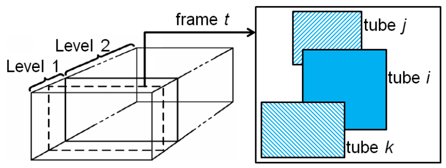


Fig. 6. Collision situations of tube i, j and k at frame t.

between rearranged tubes and background image is not too large, therefore, there is no need to consider the background inconsistency cost.

Secondly, we can find that only the collision cost is taken into account in Eq. (5). The motivation is that the temporal inconsistency between tubes is not remarkable in the online framework where the tube rearrangement is performed in a stepwise way; that is, rearranged tubes are extracted from a short interval at each step. Besides that, the start-time label set $\mathcal{L}_{\text{Online}}$ is small so that it can not produce serious temporal inconsistency between two tubes.

In [11], the collision cost is defined as the volume of two tubes' space-time overlap weighted by their activity measures. However, this cost function is insufficient because it prone to ignore small sized tubes. Since a small tube contributes tiny penalty in the overall energy function, it may be completely occluded by other tubes. To overcome this drawback, a new collision cost will be necessary to give high penalty for this case.

Assuming a current tube i and a rearranged tube $j \in \boldsymbol{Q'}$ are placed at the location ℓ_i and ℓ_j, respectively. Then, the collision cost function $E_c(\ell_i \mid \ell_j)$ is defined as:

$$E_c(\ell_i|\ell_j) = \sum_{t \in t_i \cap t_j} e^t(i,j), \tag{6}$$

$$e^t(i,j) = \begin{cases} s^t_{i,j}, & \frac{s^t_{i,j}}{I^t(i,j)\cdot a^t_j + (1-I^t(i,j))\cdot a^t_i} < \beta, \\ \kappa \cdot a^t_i, & \text{otherwise}, \end{cases} \tag{7}$$

where $t_i \cap t_j$ is the temporal intersection of tube i and j in the condensation space; $e^t(i,j)$ denotes the collision cost between tube i and a rearranged tube j at frame t; a^t_i and a^t_j denote the area of tube i and j at frame t, respectively; $s^t_{i,j} \in \{0, \min(a^t_i, a^t_j)\}$ denotes the area intersection between tube i and j at frame t; $\beta \in [0,1]$ denotes the maximal tolerable occlusion ratio; $I^t(i,j)$ is an indicator used to designate the depth ordering of tube i and j at frame t: if tube i is closer to the camera than tube j, which means tube i may occlude tube j, then $I^t(i,j) = 1$, otherwise, $I^t(i,j) = 0$. As shown in Fig. 6, the tube i occludes tube j, while it is occluded by tube k at frame t. In this case, $I^t(i,j) = 1$ and $I^t(i,k) = 0$. Such depth ordering of tubes, which determines the relationship of occlusion, can be obtained by a simple "ground plane" heuristic [12]. See [25] for more details about the depth ordering.

With Eq. (7), no matter the size of tube i, once the occlusion ratio of the corresponding slice t of tube i is higher than β which indicates the rearrangement of tube i will cause serious occlusion, then it should be heavily penalized, and the strength of penalty depends on the coefficient κ.

2) Two-Level Condensation Space: The two-level condensation space is used as the playing field of the tube filling game. As shown in Fig. 7, it includes two level containers with different size: the size of the first level condensation space $L1$ is $w \times h \times n$, while the size of the second level condensation space $L2$ is $w \times h \times \infty$, where w and h denote the width and height of the input video frames, and n (same as in Eq. (4)) denotes the number of frames in the $L1$ condensation space. The tube set $\boldsymbol{Q'}$ in Eq. (3) is made of tubes in the $L1$ condensation space.

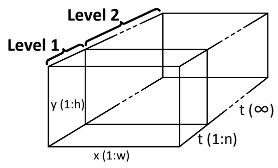


Fig. 7. Illustration of the two-level condensation space.

The $L1$ condensation space is the real playing field of the tube filling game. That is to say, the start-time label of an incoming tube should be confined in the $L1$ condensation space. Besides, the function of the $L2$ condensation space is to receive the tail of the incoming tube if $L1$ condensation space can not hold the whole tube.

As the $L1$ condensation space receives more and more tubes, it will be full at some point, just like creating "a horizontal line of blocks without gaps" in the Tetris game. At this time, the content of the $L1$ condensation space is stitched to a background image for producing condensed video, then the $L1$ condensation space will be cleared and the first n frames of $L2$ condensation space will be pushed into the $L1$ condensation space. That is to say, the tube set $\boldsymbol{Q'}$ is set to $\boldsymbol{Q''}$ at this time, where $\boldsymbol{Q''}$ is the tube tails whose start-time labels have been set as 1. This mechanism is able to keep the chronological order of tubes even they may not be filled into the same $L1$ condensation space in one time.

Note that, both $L1$ and $L2$ condensation spaces are logical spaces rather than real physical memory spaces; that is, rearranged tubes are stored in the rearranged tube buffer which will be introduced in Section III-B.

3) Online Content-aware Tube Filling: There are two main tasks for the online content-aware tube filling algorithm: (1) deciding each tube's optimal location in the $L1$ condensation space; (2) deciding whether the $L1$ condensation space is saturated. Regarding the first task, a greedy optimization method is used to decide each tube's optimal location. Regarding the second task, tubes with high collision ratio are saved in a temporary list, once the length of the temporary list surpasses a threshold, the $L1$ condensation space is considered as saturated.

Based on Eq. (5), Eq. (6) and Eq. (7), the optimal location $\mathscr{L}(i)$ of tube i is found by greedy search, as follows:

$$\mathscr{L}(i) = \arg\min_{\ell_i} \sum_{j \in \boldsymbol{Q'}} E_c(\ell_i \mid \ell_j), \tag{8}$$

where $\ell_{(\cdot)} \in \{1, 2, ..., n\}$. The optimal location is found according to the locally optimal choice at each greedy search

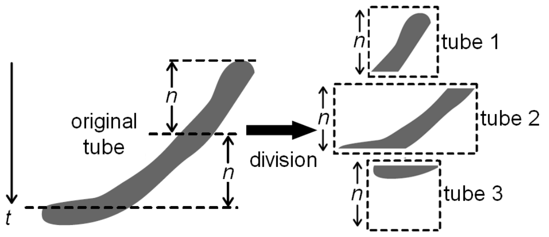


Fig. 8. Tube Division.

step with the hope of making the $L1$ condensation space full with least collision at last.

A content-aware mechanism is further designed to accomplish the second task. The following criterion is used to decide whether tube i can not be filled into the $L1$ condensation space:

$$\mathcal{CR}_i\Big(\mathscr{L}(i)\Big) > \tau, \tag{9}$$

where τ is the maximum tolerable threshold and the collision ratio $\mathcal{CR}_i\Big(\mathscr{L}(i)\Big)$ of tube i is defined as:

$$\mathcal{CR}_i(\mathscr{L}(i)) = \sum_{j\in \boldsymbol{Q'}} E_c(\mathscr{L}(i) \mid \ell_j) \Big/ \sum_{t\in t_i} a_i^t, \tag{10}$$

where a_i^t denotes the area of tube i at frame t and t_i denotes the frame length of tube i. The tube i can not be filled into the $L1$ condensation space if its collision ratio $\mathcal{CR}_i\Big(\mathscr{L}(i)\Big)$ larger than the maximum tolerable threshold τ, then such tube will be added to a temporary list $\boldsymbol{D}$, and the $L1$ condensation space is assumed to be saturated once the length of this temporary list reaches to the maximum temporary list size m. The benefit of this content-aware mechanism is that the duration of a condensed video is determined by the content of the input video, rather than by users as in [11].

4) Tube Division: Tube division is used to produce the dynamic stroboscopic effect [11]: the same object from different frames in an original video may be displayed at the same frame in the corresponding condensed video. It is simply achieved by dividing a tube into several smaller tubes, and the size of each one after division is guaranteed to be smaller or equal to the number of frames n in the $L1$ condensation space. See Fig. 8 for an example.

Tube division may cause the blinking effect and is not favorable in the online video condensation algorithm. However, with it the online video condensation algorithm can achieve higher condensation ratio.

5) Computational Cost Analysis : Assume that the computational cost of Eq. (6) is c; the start-time label set of online and offline tube filling are $\{1, 2, ..., n\}$ and $\{1, 2, ..., M\}$, respectively; the $L1$ condensation space saturated count is k, then we have $M = kn$; there are P tubes to be rearranged. Furthermore, we assume the $L1$ condensation space includes p tubes at each saturated moment, thus we have $P = kp$. With greedy search (Eq. (8)), the computational cost of online and offline tube filling is $k\cdot n\frac{(p-1)p}{2}c$ and $M\frac{(P-1)P}{2}c$, respectively. Therefore, their ratio is:

$$\frac{k\cdot n\frac{(p-1)p}{2}c}{M\frac{(P-1)P}{2}c} = \frac{k\cdot n\frac{(p-1)p}{2}c}{kn\frac{(kp-1)kp}{2}c} \approx \frac{1}{k^2} \tag{11}$$

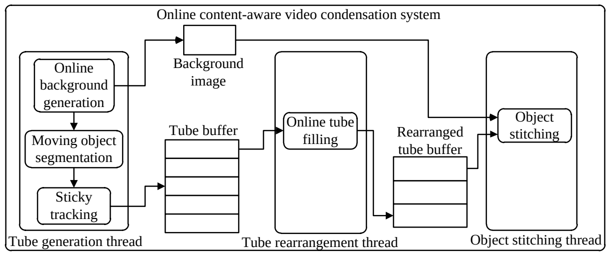


Fig. 9. The structure of the online content-aware video condensation system.

F. Object Stitching

The rearranged tubes are stitched into backgrounds using the modified Poisson image editing [11] according to the corresponding start-time labels. See [11] for more details about the modified Poisson image editing.

III. High Performance Video Condensation System

Based on the online content-aware video condensation framework, a high performance video condensation system is developed. The system is implemented with multi-threading technique. In the following section: The system structure is first described. Then, several optimizing strategies are introduced, including buffer design and acceleration strategies. Finally, a tube reallocation method is proposed to further improve tube rearrangement without decelerating the processing speed.

A. Multi-thread Based System

The online content-aware video condensation framework includes three primary modules: tube generation module including online background generation, moving object segmentation and sticky tracking, tube rearrangement module and object stitching module. As shown in Fig. 9, these modules are parallelled performed by the tube generation thread, the tube rearrangement thread and the object stitching thread, respectively. The relationship between the tube generation thread and the tube rearrangement thread can be seen as the relationship between producer and consumer. The tube generation thread just like a tube producer that pushes tubes into a tube buffer, while the tube rearrangement thread like a consumer that gets out tubes from the tube buffer for online tube filling. The detail of this producer-consumer parallel model is introduced in [26]. Similarly, there is also a producer-consumer relationship between the tube rearrangement thread and the object stitching thread. However, in this case, the tube rearrangement thread is a producer and the object stitching thread is a consumer.

Note that, the system is compatible with offline video synopsis [11–13] by a slight adjustment. Specially, one can trigger the tube generation thread first to extract all tubes and backgrounds from an input video sequence, then trigger the tube rearrangement thread to rearrange all tubes, at last trigger the object stitching thread to generate a condensed video. Note that, tubes are still rearranged by using Eq. (8), however, the system must prepare all tubes and backgrounds existing in the input video sequence in this case.

TABLE II
SPEED COMPARISON AMONG CPU-SILTP, SINGLE THREAD GRAPH CUT (STGC), MULTI-THREAD GRAPH CUT (MTGC) [17] AND GPU-SILTP BASED OBJECT SEGMENTATION METHODS.

Sequence	Resolution	CPU-SILTP (fps)	STGC (fps)	MTGC (fps)	GPU-SILTP (fps)
Outdoor [17]	320×240	73.9	29.9	147.6	1036.2
Park1 [17]	352×288	53.6	25.2	106.8	833.6
Street	704×576	13.7	5.1	21.2	292.2

B. Buffer Design

As shown in Fig. 9, there are two memory buffers in the proposed system. The tube buffer used between the tube generation thread and the tube rearrangement thread is a FIFO (First In First Out) list. The element of this FIFO buffer is a tube that waiting to be rearranged. The rearranged tube buffer used between the tube rearrangement thread and the object stitching thread is also a FIFO list. In order to avoid the efficiency deterioration caused by the frequent data interaction between the tube rearrangement thread and the object stitching thread, the element of the rearranged tube buffer is designed to store multiple rearranged tubes. Adjusting the length of the two buffers can balance the speed and memory usage of the proposed system.

C. Acceleration Strategies

In Fig. 9, moving object segmentation and object stitching are the most time consuming steps. In order to achieve faster processing speed, the GPU based moving object segmentation and the multi-core based object stitching are proposed.

1) GPU Based Moving Object Segmentation: Traditional background subtraction methods, such as the mixture of Gaussian approach [21], are usually based on the color of pixels, and each pixel is processed independently. This makes the mixture of Gaussian approach a highly parallelable algorithm, which can be easily accelerated by GPU. However, the SILTP [18] pattern of each pixel is related to its 4-neighborhood, therefore SILTP is harder to be parallelized on GPU. However, there are many tricks introduced in [27] can be used to parallelize the SILTP method. The SILTP is optimized in three aspects: pinned memory, memory coalescing and asynchronous execution.

The bandwidth between host memory and device memory is usually a bottleneck in GPU computation. As suggested in [27], pinned memory can help to improve the speed of data transfer between host memory and device memory, hence, the pinned memory is used. Furthermore, using pinned memory allow us to launch the SILTP algorithm asynchronously. In CUDA, when 16 continuous threads in the same block access continuous global memory, all individual transfers will be combined into a single transfer automatically, which is called memory coalescing [27]. To take this advantage, the distribution of the neighborhood pixels to the threads is designed from a same block as possible. In the calculation of SILTP pattern for each pixel, it always access the 4-neighborhood in a block. However, the 4 pixels are not continuous in global memory. Therefore, they are first fetched from global memory to local shared memory, and then the pattern is calculated based on the values in shared memory.

Asynchronous execution is a famous way to improve the performance of CUDA programs. The pipeline of CUDA is usually divided into three steps: (1) memory copy from host to device; (2) kernel execution; (3) memory copy from device to host. In synchronous execution mode, if the step (1) or (3) are running, the kernel is idle. In contrast, when the kernel is running, the memory bus is idle. To parallelize the memory copy and kernel execution, double buffer and two CUDA streams are used, one for memory copy and the other for kernel execution. When the kernel is processing the image data in the first buffer, the copy stream is used to transfer new image data into the second buffer. When the kernel is finished, the pointers of buffer are swaped and the kernel is restarted.

We compared the speeds of the CPU-SILTP based, graph cut based [17] and GPU-SILTP based object segmentation methods at an 8-cores @ 2.66 GHz computer with Nvidia GeForce GTX 285 (the same device used in other experiments of this paper). The results are shown in Table II. The CPU-SILTP method is implemented in single thread mode. Both the single thread (STGC) and multi-thread graph cut (MTGC) based methods are evaluated, and the number of threads used in MTGC is set to be 8, as in [17]. From Table II, we can see that the speed of CPU-SILTP is about 1 time faster than STGC. Moreover, we can find that the speed-up ratio between GPU-SILTP and CPU-SILTP consistently surpasses 10 and increases with pixel resolution. Finally, we can see that the speed-up ratio between GPU-SILTP and MTGC consistently surpasses 7 and also increases with pixel resolution.

2) Multi-core Based Object Stitching: The modified Poisson editing method [11] is used to achieve the smooth stitching and it can be seen as a problem of solving the following linear equations [11]:

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{12}$$

where $\mathbf{A}$ is a large, sparse and known $p \times p$ matrix, and p denotes the processing pixel number; the column vector $\mathbf{x}$ denotes the p unknown pixel values, and $\mathbf{b}$ is a known column vector for the Poisson equation. Therefore, our goal is to find a fast solution for Eq. (12).

Konstantinidis et al. [28] proposed a parallel Red-Black Successive Over-Relaxation method to fast solve Eq. (12), based on GPU with CUDA. The key idea of this parallel method is to divide the unknown variables $\mathbf{x}$ into a red set and a black set according to their coordinates, then parallelly update the values of each set in turn at each iteration with GPU. However, the GPU resource has been assigned to accelerate the moving object segmentation as mentioned before, therefore, to avoid the competition for the GPU resource, the multi-core parallel technique is used. Thus, the OpenMP [29] programing is used to parallelly update the values of each set in turn at each iteration. Table III shows the comparison of stitching time between solving Eq. (12) with and without OpenMP. It can be found that the speed-up increases with the processing pixel number. This inspires us to combine object stitching jobs of several frames as a whole linear equation to be solved.

TABLE III
STITCHING TIME COMPARISON BETWEEN WITH AND WITHOUT OPENMP.

Pixel Num.	Iter.	OMP. (ms)	No OMP. (ms)	Speed-Up
5994 × 3	5000	725.13	202.27	3.6
13229 × 3	5000	1608.44	350.57	4.6
56899 × 3	5000	6868.79	1319.49	5.2
82925 × 3	5000	22590.12	1808.46	12.5

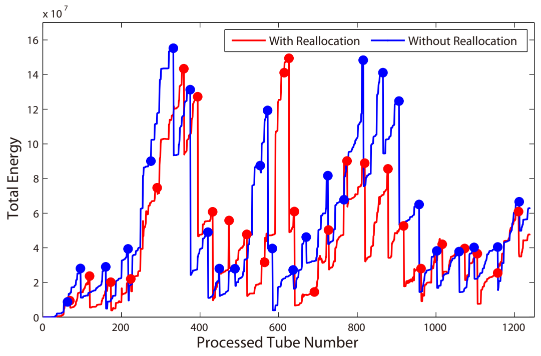


Fig. 10. Comparison between the online video condensation with and without tube reallocation for a typical tube filling process. The dots in the figure denote the moment when the $L1$ condensation space is full.

D. Tube Reallocation

Rearranged tubes can be reallocated to further improve the solution for the optimization problem of Eq. (5). The idea is to reassign optimal start-time labels of the tube set $\boldsymbol{Q}' - \boldsymbol{Q}''$, where $\boldsymbol{Q}'$ is a set including rearranged tubes in current stepwise and $\boldsymbol{Q}''$ is a set including tube tails of the previous stepwise. A tube tail is often caused by the fact that the frame length of the tube is longer than the frame length of the $L1$ condensation space (see Fig. 8). In this case, the start-time label of $\boldsymbol{Q}''$ must be set as 1 to keep chronological order of tubes. Therefore, tube reallocation only process tubes that from $\boldsymbol{Q}' - \boldsymbol{Q}''$.

In practice, the system randomly selects a tube from $\boldsymbol{Q}' - \boldsymbol{Q}''$ and treats it as an incoming tube, then recalculates it's optimal start-time label with Eq. (8). Because there doesn't exist tubes to be processed all the time, tube reallocation can be applied to the online video condensation at idle time. As a result, the proposed system can obtain a better solution without decelerating the processing speed. Fig. 10 shows a comparison of a typical process of the online video condensation with and without tube reallocation, where the same parameters are applied. We can find that before being full, the $L1$ condensation space accepts more tubes by the tube reallocation, which suggests that tube reallocation is helpful for improving condensation ratio.

IV. EXPERIMENTS AND ANALYSIS

The proposed system was evaluated with extensive experiments. Nine surveillance video sequences taken from indoor and outdoor scenes are used. The details of the running environment and the system setup are listed in Table IV and Table V, respectively. During the whole evaluation process, the proposed system does not do any down sample operation for input videos. In the following, the performance of the proposed system is summarized in five aspects: speed, condensation ratio, content-aware ability, memory usage and visual quality. The results and condensed videos are reported on a project website: http://www.cbsr.ia.ac.cn/users/jqzhu/hpvcs.htm.

TABLE IV
THE DETAILS OF THE RUNNING ENVIRONMENT.

Operating System	Window Server 2008 R2 Enterprise 64 bit
Hardware	CPU: 2.66 GHz 8 cores, GPU: Nvidia GeForce GTX 285
Compiler	Microsoft Visual Studio 2010

TABLE V
THE SYSTEM SETUP.

Parameter	value
the frame length n of the $L1$ condensation space (Fig. 7)	500
the maximal tolerable threshold τ (Eq. (9))	1.0
the maximum temporary list size m (II-E3)	16
the length of tube buffer (Fig. 9)	200
the length of rearranged tube buffer (Fig. 9)	1

A. Speed

The results of speed are summarized in Table VI. The speed decreases with the increase of the pixel resolution. For the video sequence Overpass [10] with the smallest pixel resolution, the processing speed of the proposed system achieves 995.00 fps. For those video sequences with 320×240 resolution, the processing speed ranges from 531.99 to 662.57 fps. For those video sequences with 352×288 resolution, the processing speed is about 390 fps. The processing speed of the proposed system is about 5 times as faster as that reported in [17]. Even for those high resolution (704×576) video sequences, the proposed system still has 100 fps that is 3 times faster than real-time (25fps). Besides, as shown in Fig. 11, the time cost of the whole process is mostly determined by the tube generation thread. This result demonstrates that the time cost of the tube arrangement thread and the object stitching thread is well hidden, thus our multi-thread based system is a very effective parallel system.

In order to show the speed advantage of online tube filling, the time cost of the online and offline tube filling algorithms was compared under the same hardware condition. Both of these two tube filling algorithms use greedy search (Eq. (8)) and the frame number of condensed video used in offline tube filling is equal to that in online tube filling. From Fig. 12, it can be found that the online tube filling is faster than offline tube filling. Especially when the number of frames in an input video is huge (Outdoor, Street and T-junction cases in Fig. 12), the running time of the online tube filling is only 6.53% to 20.10% of the offline tube filling. Because the longer the input video is, the larger the saturated count k of the $L1$ condensation space will be, resulting in the larger speed-up ratio between the online and the offline tube filling according to Eq. (11).

TABLE VI
THE RESULTS OF THE ONLINE CONTENT-AWARE VIDEO CONDENSATION SYSTEM ON NINE SURVEILLANCE VIDEO SEQUENCES. CR, AOMU AND POMU ARE ABBREVIATIONS OF CONDENSATION RATIO, AVERAGE OF MEMORY USAGE AND PEAK OF MEMORY USAGE, RESPECTIVELY.

Video	Resolution	#Frame	Tube Num.	Speed (fps)	CR	AoMU (MB)	PoMU (MB)
Overpass [10]	320 × 120	23950	60	995.00	23.85	91.27	133.00
Exit [17]	320 × 240	81538	361	608.17	22.15	186.13	276.00
Garden [17]	320 × 240	33826	142	662.57	35.06	58.55	81.00
Outdoor [17]	320 × 240	138556	1181	531.99	8.20	475.47	733.00
Park1 [17]	352 × 288	10221	44	390.43	9.26	92.29	111.00
Passage	352 × 288	51040	20	390.36	34.96	134.75	316.00
Street	704 × 576	100113	521	100.39	15.83	777.30	1155.00
Staircase	704 × 576	46108	495	100.62	10.25	310.44	732.00
T-junction	704 × 576	600001	1418	106.71	36.76	513.67	1366.00

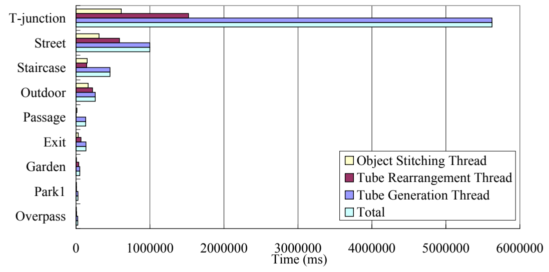


Fig. 11. Time cost of the three threads on nine video sequences.

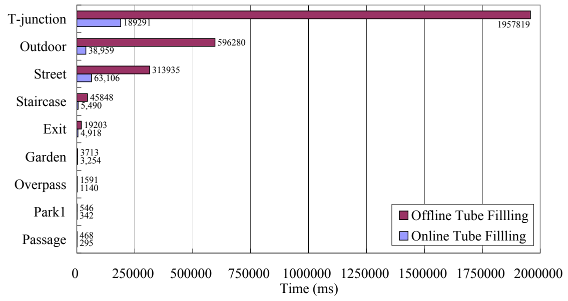


Fig. 12. Running time comparison between the online and the offline tube filling.

B. Condensation Ratio

Condensation ratio is defined as the ratio between the number of frames in an input video and the number of frames in the condensed video. The results of condensation ratio are summarized in Table VI. In Table VI, the lowest condensation ratio is 8.20 while the highest one is 36.76. Because most existing methods have not reported condensation ratio results, we only compare our method with the ribbon carving based method [10]. The proposed method has higher condensation ratio on the Overpass sequence, about 8 times (the condensation ratio reported in [10] is lower than 3 for the Overpass sequence). Moreover, the condensation ratio of the propose method can be easily adjusted by setting the maximum temporary list size m to different values. Comparing Fig. 13 (b), (c) and (d), we can find that our method can produce condensed videos with denser activities than the ribbon carving based method [10].

C. Content-aware Ability

The nine surveillance videos were processed automatically without any manual intervention. As shown in Table VI, the condensation ratio varies with different input video sequences, which demonstrates that our system has a content-aware ability to adaptively control condensation ratio. Fig. 14 intuitively exhibits the content-aware ability of our system. As shown in Fig. 14, from 3×10^5 to $4 \times 10^5-\mathrm{th}$ frame, there are more activities in this period and the $L1$ condensation space is saturated more frequently. Besides that, in this period, more background images are used for object stitching which is the desiring property of the background image selection. The content-aware ability is more desirable and reasonable than setting a fixed condensation ratio as in video synopsis [11], because users usually do not know the activity density of an input video beforehand.

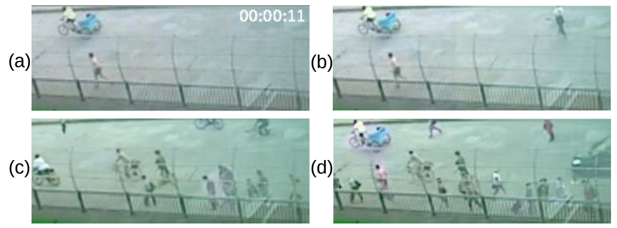


Fig. 13. Comparison of the proposed method vs. the ribbon carving based method [10]. (a) One frame from Overpass video. (b) Results by [10]. (c) and (d) Results by the proposed method with the maximum temporary list size $m = 8$ and $m = 16$, respectively.

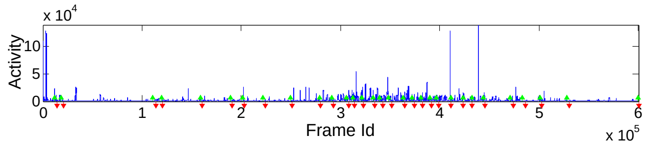


Fig. 14. Illustration of the content-aware ability in the T-junction video sequence condensation process. The blue line represents the activity at each frame. Each red nabla in the figure denotes the moment that the $L1$ condensation space is full and green delta denotes the updating moment of the background image used for object stitching.

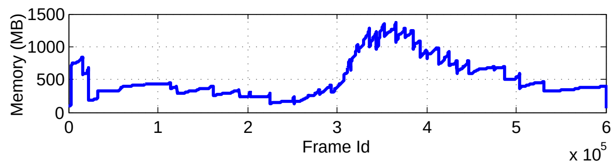


Fig. 15. Memory usage status in the condensation process of T-junction.

D. Memory Usage

The results of memory usage are summarized in Table VI. In Table VI, even for those high resolution (704×576) input video sequences, the memory usage peak of our system is lower than 1.5 GB. Additionally, for the T-junction sequence with $600,001$ frames, its memory usage status over time is plotted in Fig. 15. Combining Fig. 14 and 15, we can find two situations: (1) the memory usage is low and changes smoothly from 1×10^5 to $3 \times 10^5-\mathrm{th}$ frame where the activity is sparse; (2) the memory usage firstly increases and then decreases from 3×10^5 to $4 \times 10^5-\mathrm{th}$ frame where the activity is dense. This is because the online processing mechanism make the system use less memory in the activity sparse period and the two producer-consumer parallel models (see Fig. 9) make our system is able to limit the memory usage when the consumption speed lower than the production speed in the activity dense period. Therefore, our system is suitable for endless input videos.

E. Visual Quality

For object-level recall rate, both the proposed system and our previous work [17] are able to condense all moving objects into the condensed video due to the online content-aware strategy. However, the method in [11] has a chance of discarding some moving objects in the synopsis video due to fixed synopsis video length and occlusion conflict. For pixel-level recall rate, as shown in Table I, the SILTP based moving object segmentation method used in our system achieves higher recall rate than the graph cut based object segmentation method used in [17], thus the recall rate of the proposed system is higher than that in [17]. Note that [11] used a similar graph cut based object segmentation method as in [17].

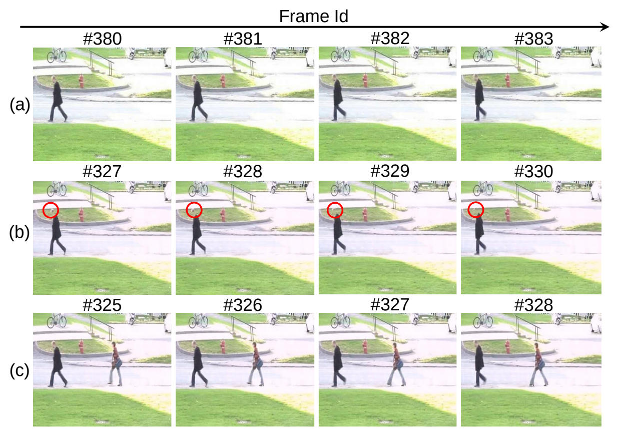


Fig. 16. Visual quality comparison on the pedestrians [22] sequence. (a) Input video. (b) Condensed video produced by [11] and red circles point out the missing parts. (c) Condensed video produced by the proposed method.

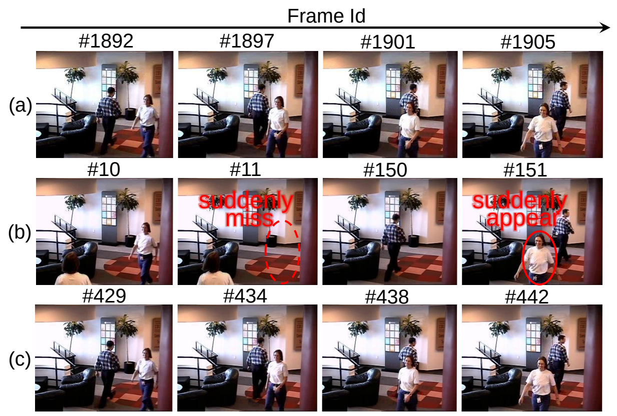


Fig. 17. Visual quality comparison on the IndoorGTTest1 [30] sequence. (a) Input video. (b) Condensed video produced by [11]. (c) Condensed video produced by the proposed method.

We further made a visual quality comparison between the proposed method and the video synopsis method [11] on two public sequences, pedestrians [22] and IndoorGTTest1 [30], as shown in Fig. 16 and Fig. 17. The IndoorGTTest1 [30] sequence was captured in indoor scene, while the pedestrians [22] was captured in outdoor scene with a poorer imaging quality. The numbers of frames in the condensed videos produced by the two methods were set equal. Because the video synopsis method [11] uses a color and graph-cut based moving object segmentation method, it has a chance of over-segmentation when the color of a moving object is similar to background. From Fig. 16, we can see that the man (the color of the head region is similar to background) in the condensed video is more completely segmented by the texture feature (SILTP) based method than that of [11]. Furthermore, from Fig. 17, we can find that the proposed system can correctly keep the chronological order of moving objects thanks to the sticky tracking strategy, while the method of [11] has a blinking effect (moving objects suddenly appear or miss in two consecutive frames, see Fig. 17 (b)) due to tracking failure. Therefore, the propose method achieves better visual effect than [11].

V. Conclusion

A high performance video condensation system based on an online content-aware video condensation framework has been proposed in this paper. The online framework can process input videos and produce condensed videos simultaneously, with much less memory and higher speed than the offline framework. Besides that, the online framework can automatically determine the duration of a condensed video. The high performance video condensation system is designed by using the multi-threading technique. The proposed system further applies GPU and multi-core techniques to accelerate the processing speed. The extensive experiments have shown the superiorities of the proposed system.

VI. Acknowledgement

This work was supported by the Chinese National Natural Science Foundation Projects #61105023, #61103156, #61105037, #61203267, #61375037, #61473291, National Science and Technology Support Program Project #2013BAK02B01, Chinese Academy of Sciences Project No. KGZD-EW-102-2, and AuthenMetric R&D Funds.

References

[1] D. O'callaghan and E. L. Lew, "Method and apparatus for video on demand with fast forward, reverse and channel pause," 1995, US Patent 5,477,263.
[2] M. A. Smith, "Video skimming and characterization through the combination of image and language understanding techniques," in *CVPR*, 1997, p. 775.
[3] N. Petrovic, N. Jojic, and T. S. Huang, "Adaptive video fast forward," *Multimedia Tools and Applications*, vol. 26, no. 3, pp. 327–344, 2005.
[4] B. Höferlin, M. Höferlin, D. Weiskopf, and G. Heidemann, "Information-based adaptive fast-forward for visual surveillance," *Multimedia Tools and Applications*, vol. 55, no. 1, pp. 127–150, 2011.
[5] M. M. Yeung and B.-L. Yeo, "Video visualization for compact presentation and fast browsing of pictorial content," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 7, no. 5, pp. 771–785, 1997.
[6] X. Zhu, X. Wu, J. Fan, A. K. Elmagarmid, and W. G. Aref, "Exploring video content structure for hierarchical summarization," *Multimedia Systems*, vol. 10, no. 2, pp. 98–115, 2004.
[7] C. Kim and J.-N. Hwang, "An integrated scheme for object-based video abstraction," in *Proceedings of the eighth ACM international conference on Multimedia*, 2000, pp. 303–311.
[8] Y. Li, T. Zhang, and D. Tretter, "An overview of video abstraction techniques," *HP Laboratories Palo Alto*, 2001.
[9] H.-W. Kang, X.-Q. Chen, Y. Matsushita, and X. Tang, "Space-time video montage," in *CVPR*, 2006, pp. 1331–1338.
[10] Z. Li, P. Ishwar, and J. Konrad, "Video condensation by ribbon carving," *IEEE Trans. on Image Processing*, vol. 18, pp. 2572–2583, November 2009.
[11] Y. Pritch, A. Rav-Acha, and S. Peleg, "Nonchronological video synopsis and indexing," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 30, no. 11, pp. 1971–1984, 2008.

[12] Y. Pritch, A. Rav-Acha, A. Gutman, and S. Peleg, “Webcam synopsis: Peeking around the world,” in *ICCV*, 2007, pp. 1–8.

[13] A. Rav-Acha, Y. Pritch, and S. Peleg, “Making a long video short: Dynamic video synopsis,” in *CVPR*, 2006, pp. 435–441.

[14] Y. Pritch, S. Ratovitch, A. Hendel, and S. Peleg, “Clustered synopsis of surveillance video,” in *IEEE International Conference on Advanced Video and Signal Based Surveillance*, 2009, pp. 195–200.

[15] M. Xu, S. Z. Li, B. Li, X. Yuan, and S. Xiang, “A set theoretical method for video synopsis,” in *Multimedia Information Retrieval*, 2008, pp. 366–370.

[16] S. Kasamwattanarote, N. Cooharojananone, S. Satoh, and R. Lipikorn, “Real Time Tunnel Based Video Summarization Using Direct Shift Collision Detection,” *Advances in Multimedia Information Processing-PCM 2010*, pp. 136–147, 2010.

[17] S. Feng, Z. Lei, D. Yi, and S. Z. Li, “Online content-aware video condensation,” in *CVPR*, 2012, pp. 2082–2087.

[18] S. Liao, G. Zhao, V. Kellokumpu, M. Pietikainen, and S. Z. Li, “Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes,” in *CVPR*, 2010, pp. 1301–1306.

[19] S. Feng, S. Liao, Z. Yuan, and S. Z. Li, “Online principal background selection for video synopsis,” in *IEEE Conf. on International Conference on Pattern Recognition*, 2010, pp. 17–20.

[20] J. Sun, W. Zhang, X. Tang, and H.-Y. Shum, “Background cut,” in *ECCV*, 2006, pp. 628–641.

[21] C. Stauffer and W. E. L. Grimson, “Adaptive background mixture models for real-time tracking,” in *CVPR*.

[22] *A video database for testing change detection algorithms*. Online, 2014, http://www.changedetection.net.

[23] A. Yilmaz, O. Javed, and M. Shah, “Object tracking: A survey,” *ACM Computing Surveys*, vol. 38, no. 4, p. 13, 2006.

[24] T. Yang, S. Z. Li, Q. Pan, and J. Li, “Real-time multiple objects tracking with occlusion handling in dynamic scenes,” in *CVPR*, 2005, pp. 970–975.

[25] G. Brostow and I. Essa, “Motion based decompositing of video,” in *ICCV*, 1999, p. 8.

[26] W. Stallings, *Operating Systems: Internals and Design Principles (7-th Edition)*. Prentice Hall, March 10, 2011.

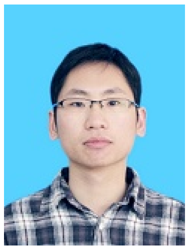
[27] C. NVIDIA, “C best practices guide,” *NVIDIA, Santa Clara, CA*, 2012.

[28] E. Konstantinidis and Y. Cotronis, “Accelerating the red/black sor method using gpus with cuda,” in *Parallel Processing and Applied Mathematics*. Springer, 2012, pp. 589–598.

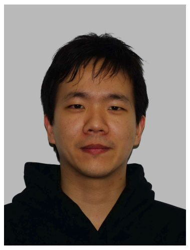
[29] B. Chapman, G. Jost, and R. Van Der Pas, *Using OpenMP: portable shared memory parallel programming*. The MIT Press, 2008, vol. 10.

[30] L. M. Brown, A. W. Senior, Y.-l. Tian, J. Connell, A. Hampapur, C.-F. Shu, H. Merkl, and M. Lu, “Performance evaluation of surveillance systems under varying conditions,” in *Proceedings of IEEE PETS Workshop*. Citeseer, 2005, pp. 1–8.

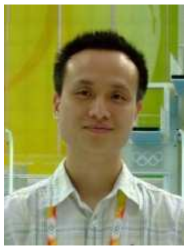
Jianqing Zhu received the B.S. degree in communication engineering and the M.S. degree in communication and Information Systems from the School of Information Science and Technology, Huaqiao University, Xiamen, China, in 2009 and 2012. Currently, he is pursuing the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China. His current research interests include computer vision and pattern recognition, with a focus on video analysis and video surveillance.



Shikun Feng received the B.S. degreein computer science and technology from the School of Computer, Wuhan University, Wuhan, China in 2009, and received the M.S. degree in pattern recognition and intelligent system from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2012. His current research interests include computer vision, pattern recognition, recommendation system and user modeling.



Dong Yi received the B.S. degree in electronic engineering in 2003, the M.S. degree in communication and information system in 2006 from Wuhan University, and the PhD degree in pattern recognition and intelligent systems from CASIA. His research areas are unconstrained face recognition, heterogeneous face recognition and deep learning. He developed the face biometric modules and systems for the immigration control projects and 2008 Beijing Olympic Games.



Shengcai Liao received the B.S. degree in mathematics and applied mathematics from the Sun Yat-Sen University in 2005 and the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences (CASIA) in 2010. He was a Post Doctoral Fellow in the Dept. of Computer Science and Engineering, Michigan State University during 2010-2012. He is currently an Assistant Professor in CASIA. His research interests include face recognition and video analysis.

Zhen Lei received the B.S. degree in Automation from University of Science and Technology of China, in 2005 and the PhD degree from Institute of Automation, Chinese Academy of Sciences in 2010, where he is currently an associate professor. His research interests are in computer vision, pattern recognition, image processing and face recognition in particular. He has published over 80 papers in international journals and conferences. He serves as an area chair of International Joint Conference on Biometrics, 2014, IAPR/IEEE International Conference on Biometric, 2015 and IEEE International Conference on Automatic Face and Gesture Recognition, 2015.

Stan Z. Li received the B.Eng. degree from Hunan University, Changsha, China, the M.Eng. degree from the National University of Defense Technology, China, and the Ph.D. degree from Surrey University, Surrey, U.K. He is currently a Professor and the Director of Center for Biometrics and Security Research, Institute of Automation, Chinese Academy of Sciences. He worked at Microsoft Research Asia as a researcher from 2000 to 2004. Prior to that, he was an Associate Professor at Nanyang Technological University, Singapore. His research interest includes pattern recognition and machine learning, image and vision processing, face recognition, biometrics, and intelligent video surveillance. He has published over 200 papers in international journals and conferences, and authored and edited eight books.