Time-Line Editing of Objects in Video

Shao-Ping Lu, Song-Hai Zhang, Jin Wei, Shi-Min Hu, *Member, IEEE,* and Ralph R Martin

Abstract—We present a video editing technique based on changing the time-lines of individual objects in video, which leaves them in their original places but puts them at different times. This allows the production of *object-level* slow motion effects, fast motion effects, or even time reversal. This is more flexible than simply applying such effects to whole frames, as new relationships between objects can be created. As we restrict object interactions to the same *spatial* locations as in the original video, our approach can produce high-quality results using only *coarse* matting of video objects. Coarse matting can be done efficiently using automatic video object segmentation, avoiding tedious manual matting. To design the output, the user interactively indicates the desired new life-spans of objects, and may also change the overall running time of the video. Our method rearranges the time-lines of objects in the video whilst applying appropriate object interaction constraints. We demonstrate that, while this editing technique is somewhat restrictive, it still allows many interesting results.

Index Terms—Object-level motion editing, Foreground/background reconstruction, Slow motion, Fast motion, Time reversal.

1 Introduction

Visual special effects can make movies more entertaining, allowing the impossible to become possible, and bringing dreams, illusions, and fantasies to life. Special effects are an indispensable post-production tool to help convey a director's ideas and artistic concepts.

Time-line editing during post-production is an important strategy to produce special effects. *Fast-motion* is a creative way to indicate the passage of time. Accelerated clouds, city traffic or crowds of people are often depicted in this way. Slowing down a video can enhance emotional and dramatic moments: for example, comic moments are often more appealing when seen in *slow-motion*. However, time-line editing is normally applied to entire *frames*, so that the whole scene in a section of video undergoes the same transformation of time coordinate. Allowing time-line changes for *individual objects* in a video has the potential to offer the director much more freedom of artistic expression, and allows new relationships between objects to be constructed. Several popular films have used such effects: for example, characters move while time stands still in the film 'The Matrix'. Usually, such effects are captured on the set.

The time-lines of individual objects in video may be changed by cutting, transforming and pasting them back into the video during post-production. This requires fine-scale object matting, as in general new object interactions may occur within the space-time video volume: objects may newly touch or overlap in certain frames, or an occlusion of one object by another may no longer happen. Typical video object segmentation and composition methods still need intensive user interaction, especially in regions where objects interact. On the other hand, automatic or semi-automatic tracking approaches, such as mean shift tracking [1], particle filtering [2] and space-time optimization [3], can readily provide coarse matting results, e.g. a bounding ellipse that contains the tracked object together with some background pixels. We take advantage of such methods to provide an easy-to-use object-level time-line editing system. Our key idea is to retain and reuse the original interactions between moving objects, and the relationships between moving objects and the background. In particular, moving objects and are kept at their original spatial *locations*, as are the original object interactions, but these may occur at a different *time*. The user can specify a starting and ending time for the motion of each object, and a speed function (constant by default) in that interval, subject to these constraints.

We use an optimization method to adjust video objects' time-lines to best meet user specified requirements, while satisfying constraints enforcing object interactions to remain at their original spatial positions. This optimization process is fast, taking just a second to perform for a dozen objects over 100 frames, allowing interactive editing of video. The user can clone objects, speed them up or slow then down, or even reverse time for objects, to achieve special effects in video.

- *S.-P Lu, S.-H Zhang and J. W are with TNList, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China.
E-mail: lushaoping@gmail.com, shz@tsinghua.edu.cn, jweimaniac@gmail.com.*
- *S.-M Hu (Corresponding author) is with TNList, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China.
E-mail: shimin@tsinghua.edu.cn.*
- *R.R. Martin is with the School of Computer Science and Informatics, Cardiff University, Cardiff, Wales CF24 3AA, UK.
E-mail: ralph@cs.cf.ac.uk.*

Fig. 1. Time-line editing of video objects puts them in the same places but at different times, resulting in new temporal relationships between objects. Top: original video, bottom: with slower cat. Left: trajectories of the cat and the woman in the video.

2 RELATED WORK

Video editing based on object matting and compositing is often used in digital media production. Schodl and Essa [4] extract video objects using blue screen matting, and generate new videos by controlling the trajectories of the objects and rendering them in arbitrary video locations. Video matting and compositing entail a tedious amount of user interaction to extract objects, even for a short video [5], [6], [7], [8]. A variety of approaches can be used to alleviate this, such as contour tracking [9], optical flow assisted Bayesian matting [10], 3D graph cut [11], mean shift segmentation [12], and local classifiers [13]. Even so, current methods still require intensive user interaction to perform *accurate* video object matting, and cannot handle objects lacking clear shape boundaries such as smoke, or objects with motion blur. Although path arrangement has been extensively considered in 3D animation, such as group motion editing [14], it cannot be directly used in video object path editing due to the difficulty of object extraction and compositing. In contrast, our system avoids the need for accurate object matting as moving objects are always placed at their original locations, albeit at different times, finessing the compositing problem. Even a bounding ellipse provided by straightforward tracking of the object can provide adequate matting results.

Various approaches have been devised to provide temporal analysis and editing tools for video. For example, video abstraction [15], [16] allows fast video browsing by automatically creating a shortened version which still contains all important events. A common approach to representing video evolution over time is to use key frame selection and arrangement [17], but, being frame-based, it does not permit manipulation of individual objects. Video montage [18] is similar to video summarization, but extracts informative spatio-temporal portions of the input video and fuses them in a new way into an output video volume. Barnes *et al*. [19] visualize the video in the style of a tapestry without hard borders between frames, providing spatial continuity yet also allowing continuous zoomin to finer temporal resolutions. Again, the aim is automatic summarization, rather than user-controlled editing. Video condensation based on seam carving [20], [21], [22] is another a flexible approach for removing frames to adjust the overall length of a video. The above methods generally handle information at the frame, or pixel level, whereas our tools allow the user to modify objects, which allows more flexible rearrangement of video content.

Goldman *et al*. [2] advocate that object tracking, annotation and composition can lead to enriched video editing applications. Methods in [23], [24] enable users to navigate video using computed timelines of moving objects and other interesting content. Liu *et al*. [25] present a system for magnifying micro-repetitive motions by motion based pixel clustering and inpainting. Recent work [26] provides a video mesh structure to interactively achieve depth-aware compositing and relighting. Scholz *et al*. [27] present a fine segmentation and composition framework to produce object deformation and other editing effects, allowing spatial changes to objects; it requires intensive user interaction as well as foreground extraction and 3D inpainting. Rav-Acha *et al*. [28], [29] introduce a dynamic scene mosaic editing approach to generate temporal changes, but, to avoid artifacts, require that the moving objects should not interact. Our method analyzes and records object interactions, and avoids artifacts in the output by constraining the kinds of editing allowed. We provide a visual interface for efficient manipulation of objects' life-spans and speeds in the video volume.

Object-based video summarization methods also exist, rearranging objects into an image or a short video in the style of a static or moving storyboard [30], [31]. Goldman *et al*. [32] present a method for visualizing short video clips as a single image, using the visual language of storyboards. Pritch *et al*. [33]

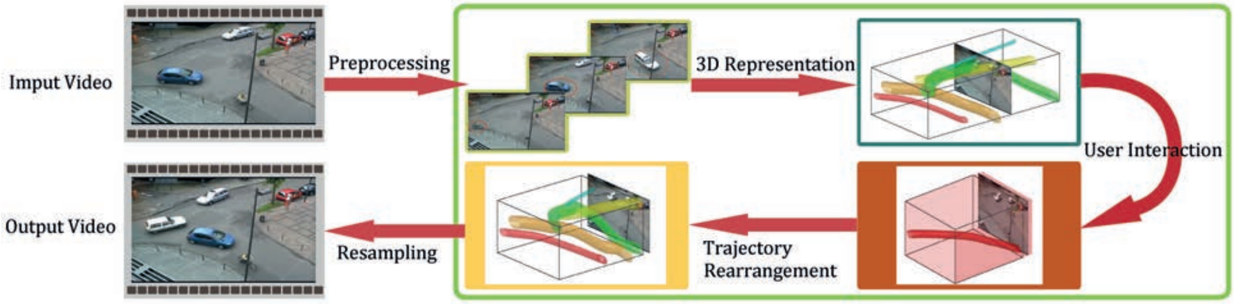


Fig. 2. Pipeline. Preprocessing includes panoramic background construction and coarse extraction of moving objects. Video tubes are constructed for each extracted object. The user specifies starting times and speed for edited objects. Trajectories of video objects are optimized to preserve original interactions between objects, and relationships to the background. Resampling of objects from suitable frames produces the output.

shorten an input video by simultaneously showing several actions which originally occurred at different times. These techniques represent activities as 3D objects in the space-time volume, and produce shortened output by packing these objects more tightly along the time axis while avoiding object collisions. These approaches shift object interactions through time while keeping their spatial locations intact to avoid visual artifacts. We use the same approach of keeping object interactions at the same spatial locations, optimizing objects' locations in time to best meet the user's requirements whilst also satisfying constraints. Our method not only allows video to be condensed, but also allows the user to determine the life-spans of individual objects, including changing their starting times, speeding them up or slowing them down, or even reversing their time-lines.

3 Approach

Our system allows the user to edit objects' time-lines, and produces artifact-free video output while only needing *coarse* video object matting (see the bottom of Figure 4). Our approach relies on reinserting each object in the output at the *same place* as before, with the same background, but at a *different time*. Carefully handling object interactions is the key to our approach. When one object partly or wholly occludes another, or their coarse segmentations overlap, any *changes* to their interaction will *prevent* direct composition of these objects with the background. We thus disallow such changes: output is produced using an optimization approach which imposes two hard constraints, while also best satisfying the (possibly conflicting) user requests:

- Moving objects *must* remain in their original *spatial* positions (and orientations) and can only be transformed to a new *time*.
- Interacting objects (i.e. objects which are very close or overlapping) *must* still interact in the same way, at the same relative speed, although maybe at a different time.
- The user *may* specify new starting and ending times for objects, as well as a speed function within that duration; weights priorities such requirements for different objects.
- Certain frames for an object *may* be marked as important, with a greater priority of selection in the output.
- The user *may* lengthen or shorten the entire video.

We allow the background to move (pan), in which case keeping objects and their interactions at the same spatial positions does not mean at the same pixel coordinates, but at the same location *relative* to a global static background for the scene. Thus, our method builds a panoramic background for all frames and coarsely extracts *tubes* representing the spatio-temporal presence of each moving object in the video. Object extraction is performed using an interactive keyframe-interpolation system, which coarsely determines a bounding ellipse in each frame for each moving object, forming a *tube* in video space-time. After detecting all bounding ellipses in each frame, SIFT features are extracted from the remaining background in each frame, and we follow the approach in [34] to register frames to generate the panoramic background image. We extract SIFT features for all images and also use optical flow to provide correspondences between adjacent frames. RANSAC is then used to match all frames to a base frame and compute a perspective matrix for each frame. The homography parameters obtained from the above approach are used to transform each frame and its bounding ellipses to global coordinates. The bounding ellipses are labeled, and may be adjusted on key frames by the user if necessary. After interpolation to other frames, the resulting ellipses can also be manually adjusted if poor results are caused by problems with interpolation or homography parameters. To perform coarse matting, we *directly* extract the difference between each aligned image and the background image inside each object's ellipse to produce that object's alpha information for the current frame.

Having determined the background and moving objects, video object trajectory rearrangement involves two steps: adjusting the shapes of the video tubes within the video volume, and resampling the tubes at new times. The basic principle used in the first step is that all objects should follow their original spatial pathways but at different times to before. Initially, the user sets a new time-line for each object to be changed, including its starting and ending time, and its speed function. These user-selected time-lines may conflict with the interaction constraints, so we optimize the time-lines for all objects to best meet the user's input requests while strictly preserving the nature and spatial locations of object interactions. We also take into account any requested change in overall video length. This optimization may be weighted to indicate that some tubes are more important than others. The result is a new video tube for the optimized life-span of each object; this may be shorter or longer than the original.

The new video tubes are now resampled, and stitched with the background to produce the overall result. Resampling is done by means of a per-object frame selection process which takes into account any user-prioritized frames that should be preferentially kept. As objects still appear in their original spatial locations, and have the same relationships with the background and other objects (but at different times), the main visual defects which may arise are due to illumination changes over time. Alpha matting with illumination adjustment solves this problem to a large degree.

We next define our notation. We suppose the input video has N frames, and the chosen number of output frames is M. The spatio-temporal track of a moving object is a tube made up of pixels with spatial coordinates (x, y) in frames at time t. See the top-left of Figure 3. The purple tube representing one object interacts with another gold tube, green dots marking the beginning and end of the interaction. In such cases, we subdivide these tubes into *sub-tubes* at the start and end interaction points as shown at the bottom-left of Figure 3. A point at (x, y) in frame t which belongs to sub-tube i is denoted by $p_i(x, y, t)$. Sub-tubes are given an index i; all sub-tubes for a given object have consecutive indices. For example, if there were only 2 tubes, with 3 and 4 parts respectively, their sub-tubes would have indices $0, 1, 2$ and $3, 4, 5, 6$. We use t_{is} and t_{ie} to represent the start and end times of each sub-tube. The second sub-tube of the purple tube (see the bottom-left of Figure 3) is an *interaction sub-tube* shared with the gold tube. Within such a sub-tube, both interacting objects must retain their original relative temporal relationship, so that they interact in the same way, ensuring that the original frames still represent a valid appearance for the interaction.

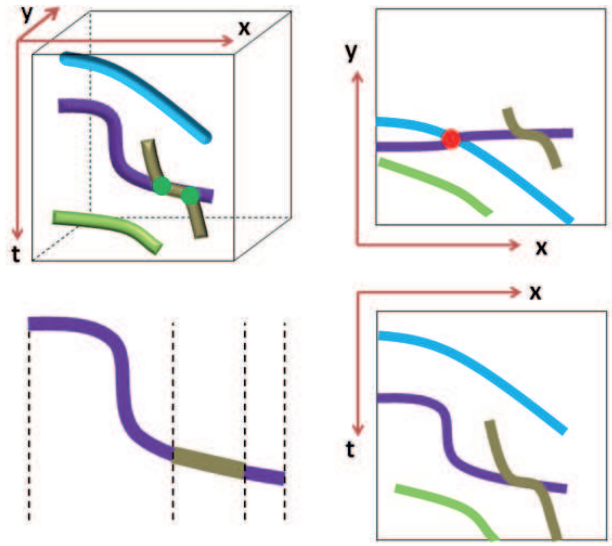


Fig. 3. The video volume. Top-left: original trajectories, with start and end of an interaction marked by green dots. Bottom-left: object trajectories are subdivided into sub-tubes at these points. The second sub-tube in the purple trajectory interacts with the gold trajectory. Right: trajectories mapped onto x-y and x-t planes. Note that the red dot is not a real interaction event.

The top-right of Figure 3 shows all tubes mapped onto the x-y plane. *Potential interaction points* like the red circle are not an actual intersection in the video volume, but have the potential to become one if object tubes were adjusted independently. When optimizing the new object tubes, we do so in a way which avoids the possibility of a potential interaction becoming an actual interaction.

Our video editing process rearranges object time-lines using affine transformations of time for each sub-tube. First, however all user-defined speed functions are applied as pre-mapping functions which adjust the trajectories of the tubes while keeping the life-spans. Thus, each sub-tube i is transformed to a new output sub-tube i' given by

$$p_{i'}(x, y, t) = p_i(x, y, A_i t + B_i), \tag{1}$$

where A_i determines time scaling and B_i determines time shift. We find A_i and B_i for each sub-tube by seeking a solution which is close as possible to the user's requests for time-line modification while meeting the hard constraints.

3.1 Optimization

Determining appropriate affine transformations of time is done by taking into account the considerations described below; appropriate scaling is applied if the overall video length is to be changed. These requirements may conflict, so we seek an overall solution which is the best compromise to meeting them all. For brevity, we ignore cases in which objects are to

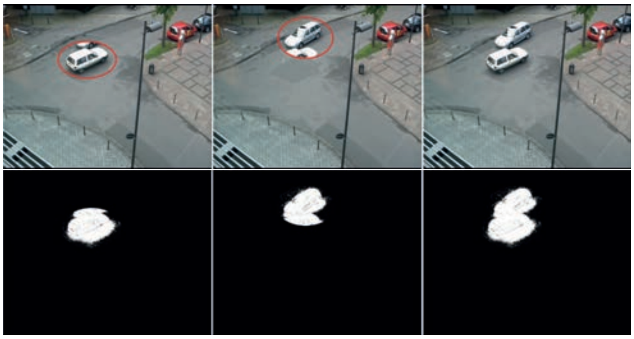


Fig. 4. Original interaction preservation. Left, center: the sergmented regions for two cars, taken from the original frames, shown on the global background. Right: composed result containing both cars simultaneously. Note that their original interaction is preserved. Bottom: corresponding matting results.

be reversed in time; these can easily be handled by straightforward modifications.

Duration: Durations of life-spans should remain unchanged for unedited objects. Edited objects should have new life-spans with lengths as close as possible to those specified by the user.

Temporal location: The life-spans of unedited objects should *start* and *end* as near as possible to their original starting and ending times, with time progressing uniformly between start and finish. Edited objects should start and end as close as possible to user specified times, with time progressing uniformly in between. For objects with a user specified speed function, the new space-time distribution of the tube is applied after mapping the original tube by the speed function.

To meet the first requirement, we define an energy term $E_D(i)$ whose effect is to enforce the appropriate life-span for each sub-tube:

$$E_D(i) = ||\frac{H_i - A_i L_i}{M}||. \tag{2}$$

Here H_i is the desired life-span of sub-tube i, and $L_i = (t_{ie} - t_{is})$ is its original life-span. For edited objects, H_i is given by the desired life-span of its parent tube as determined by the user, while it is set to ML_i/N for unedited objects.

To meet the second requirement, we define an energy $E_L(i)$ which penalizes displacement of sub-tube i from its desired position in time; we do so by considering the time at which each frame of the object should occur:

$$E_L(i) = \frac{1}{(t_{ie} - t_{is})} \sum_{t=t_{is}}^{t_{ie}} ||(\frac{A_i t + B_i}{M} - \frac{t}{N})||. \tag{3}$$

(This equation can be simplified to avoid the need for explicit summation).

These two terms are combined in an overall energy function to balance these requirements; a per-tube

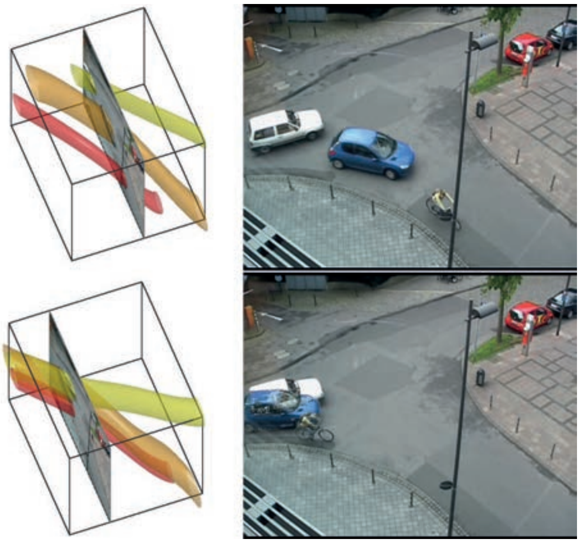


Fig. 5. New interaction prevention. Top: if new interactions are prevented, cars' tubes remain separate. Bottom: otherwise, unwanted interactions may arise between tubes, resulting in blue and white cars overlapping in an unrealistic manner.

weight w_i allows the user to indicate the importance of meeting the requirements for each subtube—tubes with higher weight should more closely meet the user's requirements:

$$E = \sum_i w_i(\lambda E_D(i) + E_L(i)), \tag{4}$$

where λ controls relative importance of these requirements, and is set to 2.5 by default.

Before we can minimize this energy, we also apply several hard constraints, as described shortly. This leads to a non-linear convex optimization problem whose unknowns are A_i and B_i. We use CVX [35] to efficiently find an optimal solution.

3.2 Constraints

When minimizing the energy, several constraints must be imposed in addition to those described earlier.

Affine parameters: Affine parameters can only take on certain values if each object is to fit into the target number of output frames. Temporal scaling and shifting parameters must thus satisfy

$$\begin{aligned} \frac{\max(N, M)}{L_i} \geq \quad & A_i \quad \geq 0, \\ \max(N, M) \geq \quad & B_i \quad \geq \min(-N, -M). \end{aligned}$$

Tube continuity: Consecutive sub-tubes belonging to the same tube must remain connected. Continuity between the end of sub-tube i and the start of sub-tube j requires

$$A_i t_{ie} + B_i + 1 = A_j(t_{ie} + 1) + B_j. \tag{5}$$

Original interaction preservation: To preserve original interaction points at which different objects start to interact, relevant object sub-tubes for the interacting objects must connect in space-time. If one object's sub-tube i starts at time t_k and interacts with sub-tube j of another object trajectory, preservation of the initial interaction point under the affine transformation requires that

$$A_i t_k + B_i = A_j t_k + B_j. \tag{6}$$

An example preserving interaction between two cars is shown in the top row of Figure 4.

New interaction prevention: To prevent *potential* interaction points between different object tubes from becoming *real* interaction points, we should ensure that different objects go through them at *different* times. These times should be sufficiently distinct that we can rely on coarse object matting when placing objects in the final output. We thus impose the following temporal separation constraint. If sub-tube i and sub-tube j share a potential interaction point, we require

$$||(A_i t_i + B_i) - (A_j t_j + B_j)|| > \varepsilon \tag{7}$$

where t_i and t_j are the corresponding times for each object. In our implementation, we set ε to between 5 and 10, taking into account both the sizes of the objects and the speeds at which they are moving, ensuring at least 5 frames separation as a safety margin (as the objects may be sampled differently—see Section 3.3). Figure 5 shows the undesirable results that can happen if this constraint is not added.

3.3 Object Resampling

Having determined the affine temporal scaling for each sub-tube, we separately *resample* each transformed sub-tube to give the object's appearance in each output frame. We use uniform resampling with user defined weighting curves to produce the output frames. While *interpolation* between appropriate input samples would seem an alternative and perhaps more intuitive solution, it can introduce artifacts for various reasons, as explained in e.g. [36], and more importantly, is incompatible with our coarse matting approach.

Clearly, output samples will not necessarily fall precisely at transformed input samples. Thus, instead we *select* input frames for each object to generate the final output. Given a sequence of input samples, and times at which output samples are required, we use the input sample occurring at the time nearest to that of the desired output sample. Other work has also used frame based resampling [31], [37].

3.4 User Interface

Our interface offers various controls for object timeline editing (see the supplemental video), including both overall video length (M), and for moving objects, specifying the start and end time (the t_{is} and t_{id} values), the life-span (H_i), and graphical entry of their speed function (resampling weighting). A default resampling weight of 1 is used; values are allowed to range from 0 to 5. The user may edit the weights using the speed curve to give desired resampling weights for each frame. The user may also specify time reversal for objects, object cloning and object deletion. The interface also allows sub-tubes and output frames to be marked as having greater importance (w_i) in the output. During editing, unedited objects which interact with edited objects are also adjusted to ensure interactions are preserved. For example, if the user clones an object, any other interacting objects will also be cloned. If this is not desired, the user should carefully limit the portion of the object's life which is cloned to that part without interactions with other objects.

3.5 Performance

Our system allows objects to be extracted without laborious interaction, and provides real-time interactive control and visualization of the editing results. The most time consuming step in our system is the preprocessing used to construct the panoramic background image and interactively extract the moving objects. The user marks a bounding ellipse on key frames for each moving object; the system takes about 0.3s per frame to track each object in a complex scene. Background construction takes 0.1s per frame. Although the preprocessing needs hundreds of seconds, it is executed just once, and after that the user is free to experiment with many different rearrangements of the video. As shown in Table 1, once the user has set parameters, optimization takes under 1 second in our examples, mostly to solve the energy Equation 4 to find optimal object rearrangements. This time depends mainly on the number of constraints, which in turn is determined by the number of object interactions. Overall, we can readily achieve realtime performance for user interaction on a typical PC with an Intel 2.5Ghz Core 2 Duo CPU and 2GB memory. To further improve performance, we merge video sub-tubes with start or end points less than 5 frames apart, marking the result as an intersection sub-tube. Table 1 shows timings for all examples in the paper, with numbers of objects and corresponding relation constraints.

4 Results and Discussion

We have tested our video editing algorithm with many examples, producing a variety of visual effects which demonstrate its usefulness, and that would be difficult or tedious to obtain by other means. Figures 1 and 8 show examples of object level fast and slow motion effects and the corresponding original videos. In Figure 1 the cat moves slower. A faster cat is shown in the supplemental material. In order to preserve the

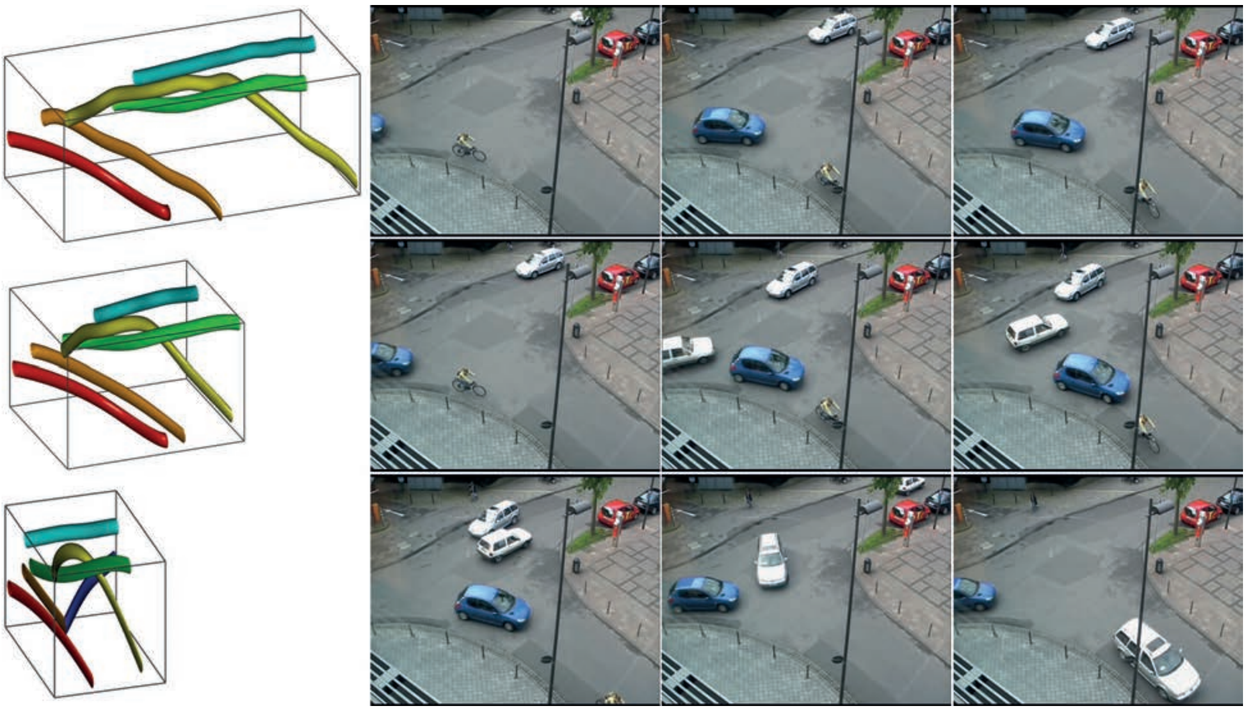


Fig. 6. Video editing. Top: input video. Middle: summarization into half the original time. Bottom: editing to fast forward and then reverse the blue car. Frames are shown at equal time separations in each case. Note that lifespans of unedited objects are preserved to the exent possible.

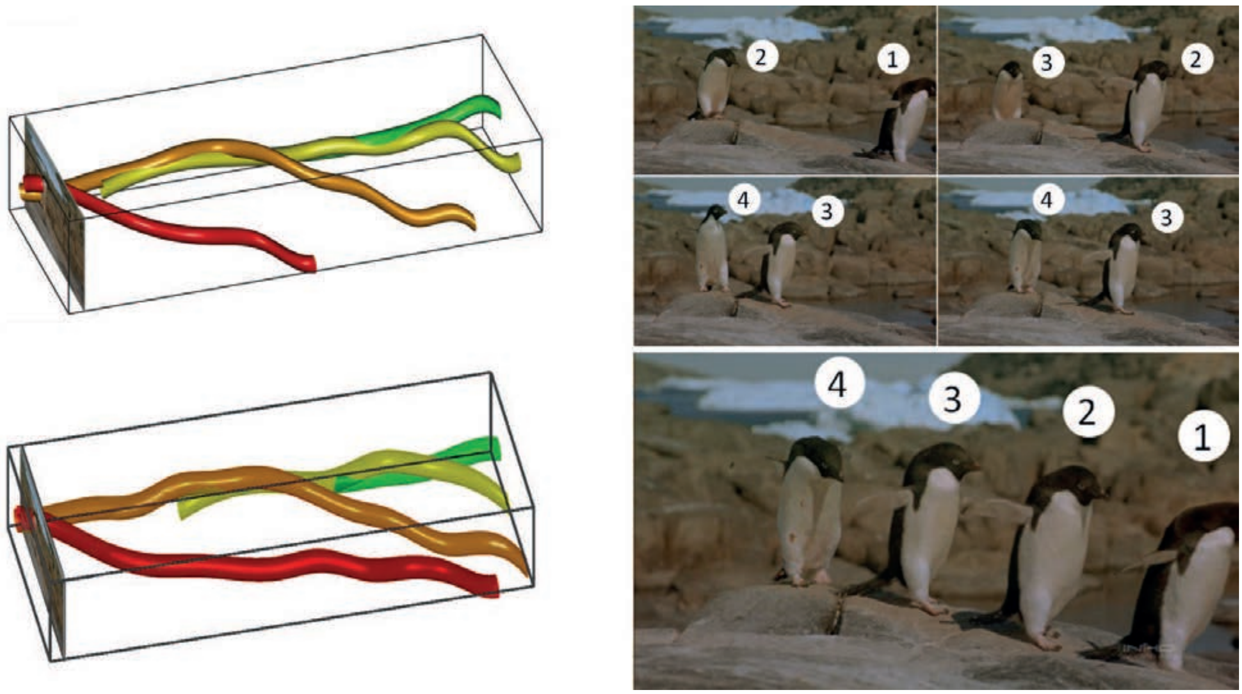


Fig. 7. Video object rearrangement. Top right: four penguins appear pairwise in the original video. Bottom right: an output frame after interactively rearranging the penguins to appear together. Left: corresponding tubes.

original spatial location of the interaction between the cat and the girl (see the fourth column), the faster moving character enters the scene at a later time. In Figure 8, we have shortened the entire video and interactively rearranged the cars to have approximately the same speed, and to be approximately equally spaced. In Figure 7 we have moved the time-spans of the penguins to make them appear in the scene simultaneously. Figure 9 shows object cloning and reversal examples in which we make three copies of

TABLE 1
Performance of the system.

Video Clip	Fig. 1	Fig. 6	Fig. 7	Fig. 8	Fig. 9
Frames	150	270	240	160	230
Width	960	720	960	1280	640
Height	540	560	540	720	480
Objects	2	5	4	10	3
Sub-tubes	5	7	8	16	10
Preprocessing	225s	570s	300s	760s	695s
Optimization	0.48s	0.71s	0.66s	0.85s	0.7s

a girl on a slide; we also make the girl go up the slide rather than down. Unlike [30], which leads to ghosting, our method automatically arranges tubes to prevent object overlaps. (Corresponding videos are available in the supplement).

While such applications are the main target of our approach, other less obvious effects can also be achieved. For example, we can selectively shorten a video to a user-specified length (see Figure 6) by setting object lifetimes to either their original lengths, or the desired total length, whichever is smaller. Our approach differs from previous approaches to video summarization which either produce a summary no shorter than the longest lifetime of a single object, or, for shorter results, unnaturally cut the video tubes for objects and move parts of them in time (e.g. Pritch's method in [33]). Instead, we can speed objects up to reduce the overall time. The bottom row of Figure 6 further shows object fast forward, reverse motion and object duplication, as well as local speed editing.

As Figure 5 shows, if object inter-relationships are ignored when moving objects in time, unwanted overlaps may arise between objects originally crossing the same region of space at different times. By preventing new object interactions, we avoid such collisions between object trajectories in the output video. Unlike Rav-Acha *et al*'s method [28], [29], our algorithm preserves real interactions, allowing effective editing of objects while avoiding visual artifacts at interactions. In Figure 6, more objects are shown per frame as the overall video time is reduced. Objects follow their original paths, but spatial relationships are also well preserved. We also note that each sub-tube (not tube) is adjusted in terms of time scale and offset to meet the users desired object time-lines. It would be extremely difficult and tedious for the user to manually adjust time-lines so as to preserve existing interactions and prevent new interactions, especially for multiple objects.

We use ellipses for masking for simplicity of implementation and ease of manipulation. The user can easily draw an ellipse in key frames to initiate object tracking. Furthermore, when tracking is inaccurate in non-key frames, the user can quickly manually correct the ellipse: a little additional user interaction can overcome minor failures in tracking. This avoids the cost and difficulty of implementation of highly sophisticated tracking methods, which still are not guaranteed to always work. Exactly how coarse matting is done is unimportant—the key idea is that accurate matting is not needed when the background is robustly reconstructed and objects retain their original locations.

In practice, it is not always necessary to extract *all* moving objects, provided that the ones of interest (e.g. football players) consistently occupy a different spatial area of the video to the others (e.g. spectators), so that the two groups do not interact. In this case a moving background can be used. It too must be resampled if a different length video is required, using a similar approach to that for object resampling.

5 LIMITATIONS

Although we have obtained encouraging results for many applications of our video object editing framework, our approach can provide poor quality results in certain cases. Our method is appropriate for video for which a panoramic background can be readily constructed and video objects can be tracked (as individuals, or suitable groups). In such cases, temporal adjustment and rearrangement at the object-level makes it possible to produce special visual effects. Clearly, our system can break down if there is a failure to track and extract foreground objects or the background. Less obviously, if the user places unrealistic or conflicting requirements on the rearranged objects, this may result in an unsolvable optimisation problem; this may also happen if a scene is very complex and there is insufficient freedom meet all of a user's seemingly plausible requests. Finally, if large changes are made to the lifetimes of object sub-tubes, motion of objects in the output video may appear unnatural due to use of a frame-selection process. Widely different changes to lifetimes of adjacent sub-tubes for a single object may also result in unnatural accelerations or decelerations. We now discuss these issues further.

Complex backgrounds and camera motions: Our method may work poorly in the presence of background change (e.g. in lighting, even if background objects remain static), and errors in background reconstruction. Each frame in the output video includes moving objects and the panoramic background, and their composition is performed according to the alpha values obtained by coarse matting. We note that the coarse matting includes part of the background as well as the moving object, so if the background changes noticeably, visible artifacts may arise due to composition of the elliptical region with the background. Furthermore, good video composition results rely on successful background reconstruction. The panoramic background image is generated under an assumption of a particular model of camera motion, which may not be accurate; even if it is, a single static

Fig. 8. Video object rearrangement. Top: two frames of the original video. Bottom: two output frames after interactively rearranging the cars to be approximately equally spaced.

Fig. 9. Video object reversal and cloning. First row: girl on slide cloned. Second row: girl going up the slide.

image may not exist for complex camera motions, e.g. due to parallex effects. Our method thus shares common limitations with several other papers with respect to handling complex backgrounds and camera motions [27], [30]. Robust camera stabilization and background reconstruction for more general camera motions are still challenging topics in computer vision [38], [39]. Our method can potentially benefit from advances in those areas.

Time-line conflicts: Preserving original interactions and preventing new ones are imposed as constraints during video editing. If the user manipulates objects inconsistently, this may lead to visual artifacts; it may even lead to an unsolvable optimization problem if there are many complex interactions and insufficient freedom to permit the desired editing operations. To avoid artifacts, and gain extra freedom, the user may resolve conflicts by moving or trimming parts of objects' time-lines to produce the desired result, or even delete whole objects. For example, in the time reversal example shown, we trimmed the last part of the girl's tube to ensure the problem was solvable.

Implausible speeds: Our algorithm focuses on preserving interrelations between paths in the time dimension. If many objects in the video have the potential to intersect (i.e. to cross a shared spatial location at different times), and certain objects are weighted for preservation, other objects can suffer unnatural accelerations or temporal jittering. This could perhaps be mitigated by using a more sophisticated resampling method (as in [31]) or content-aware interpolation. The former would reduce, but not completely eliminate, motion jitter. Simple 2D interpolation can often produce visual artifacts even with accurate motion estimation, as objects have 3D shapes; such artifacts may be no more acceptable to the viewer than minor motion jitter. An alternative solution to alleviate such artifacts would introduce motion-blur (e.g. using simple box filtering of adjacent frames after realigning object's centroids [40]) for fast moving objects. Such cases could also be handled better if spatial adjustments were allowed as well as temporal ones during video tube optimization, but doing so is incompatible with our framework based on coarse segmentation and matting. Indeed, allowing spatial editing would give a much more flexible system overall. Nevertheless, it may be possible to be a little more flexible without offering full generality of spatial adjustment. If we were to restrict spatial changes to locations with similar, constant coloured, backgrounds for example, we might be able to still use coarse matting, perhaps using some combination of video inpainting and graphcut matching to find an optimal new location.

Our method also may produce unnatural results if the output video is excessively stretched (or compressed) in time, due to the use of frame selection: frames would be repeated, and motion would tend to jump. Again, an interpolation scheme of some kind could overcome this issue. A further problem which may arise is sudden changes in speed between adjacent sub-tubes, resulting in implausibly large accelerations or decelerations. This could be solved by introducing a higher order smoothing term into our optimization framework, or even by constructing a new speed-aware optimization scheme with larger freedom. Currently, sub-tube motion rearrangement assumes an affine transformation with a time shift and scaling, giving little freedom to edit the speed in the presence of higher order constraints. Speed-oriented modeling could be used to precisely edit the motion at frame-level, but would require more complicated user interaction. In the current system, we have preferred simplicity over intricate control, but accept that for some applications, detailed control would be desirable.

6 Conclusions

We have presented a novel realtime method for modifying object motions in video. The key idea in our algorithm is to keep object interactions at the same spatial locations with respected to the background while modifying the interaction times. This allows us to avoid the need for precise matting, reducing the need for much tedious user interaction. We optimize object trajectories to meet user requests concerning temporal locations and speeds of objects, while at the same time including constraints to preserve interrelations between individual object trajectories.

Acknowledgments

We would like to thank all the anonymous reviewers for their valuable comments. This work was supported by the National Basic Research Project of China (Project Number 2011CB302205), the Natural Science Foundation of China (Project Number 61120106007 and 60970100), and a UK EPSRC Travel Grant.

References

[1] G. R. Bradski, "Computer vision face tracking for use in a perceptual user interface," *Intelligence Technology Journal*, vol. 2, pp. 12–21, 1998.

[2] D. B. Goldman, C. Gonterman, B. Curless, D. Salesin, and S. M. Seitz, "Video object annotation, navigation, and composition," in *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*, Oct. 2008, pp. 3–12.

[3] X. L. K. Wei and J. X. Chai, "Interactive tracking of 2D generic objects with spacetime optimization," in *Proceedings of the 10th European Conference on Computer Vision: Part I*, 2008, pp. 657–670.

[4] A. Schodl and I. A. Essa, "Controlled animation of video sprites," in *ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2002, pp. 121–127.

[5] S. Yeung, C. Tang, M. Brown, and S. Kang, "Matting and compositing of transparent and refractive objects," *ACM Transactions on Graphics*, vol. 30, no. 1, p. 2, 2011.

[6] K. He, C. Rhemann, C. Rother, X. Tang, and J. Sun, "A global sampling method for alpha matting," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2011, pp. 2049–2056.

[7] Y. Zhang and R. Tong, "Environment-sensitive cloning in images," *The Visual Computer*, pp. 1–10, 2011.

[8] Z. Tang, Z. Miao, Y. Wan, and D. Zhang, "Video matting via opacity propagation," *The Visual Computer*, pp. 1–15, 2011.

[9] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, pp. 321–331, 1988.

[10] Y.-Y. Chuang, A. Agarwala, B. Curless, D. H. Salesin, and R. Szeliski, "Video matting of complex scenes," *ACM Transactions on Graphics*, vol. 21, pp. 243–248, Jul. 2002.

[11] Y. Li, J. Sun, and H.-Y. Shum, "Video object cut and paste," *ACM Transactions on Graphics*, vol. 24, pp. 595–600, Jul. 2005.

[12] J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen, "Interactive video cutout," *ACM Transactions on Graphics*, vol. 24, pp. 585–594, Jul. 2005.

[13] X. Bai, J. Wang, D. Simons, and G. Sapiro, "Video snapcut: robust video object cutout using localized classifiers," *ACM Transactions on Graphics*, vol. 28, pp. 70:1–70:11, Jul. 2009.

[14] T. Kwon, K. H. Lee, J. Lee, and S. Takahashi, "Group motion editing," *ACM Transactions on Graphics*, vol. 27, no. 3, pp. 80:1–80:8, Aug. 2008.

[15] Y. Li, T. Zhang, and D. Tretter, "An overview of video abstraction techniques," HP Laboratory, Tech. Rep. HP-2001-191, 2001.

[16] B. T. Truong and S. Venkatesh, "Video abstraction: A systematic review and classification," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 3, pp. 1–37, 2007.

[17] C. W. Ngo, Y. F. Ma, and H. J. Zhang, "Automatic video summarization by graph modeling," in *Proceedings of the Ninth IEEE International Conference on Computer Vision*, 2003, pp. 104–109.

[18] H. W. Kang, X. Q. Chen, Y. Matsushita, and X. Tang, "Space-time video montage," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006, pp. II: 1331–1338.

[19] C. Barnes, D. B. Goldman, E. Shechtman, and A. Finkelstein, "Video tapestries with continuous temporal zoom," *ACM Transactions on Graphics*, vol. 29, pp. 89:1–89:9, Jul. 2010.

[20] Z. Li, P. Ishwar, and J. Konrad, "Video condensation by ribbon carving," *IEEE Transactions on Image Processing*, vol. 18, pp. 2572–2583, 2009.

[21] K. Slot, R. Truelsen, and J. Sporring, "Content-aware video editing in the temporal domain," in *Proceedings of the 16th Scandinavian Conference on Image Analysis*, 2009, pp. 490–499.

[22] B. Chen and P. Sen, "Video carving," in *Eurographics'08, Short Papers.*, 2008.

[23] S. Pongnumkul, J. Wang, G. Ramos, and M. F. Cohen, "Content-aware dynamic timeline for video browsing," in *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, 2010, pp. 139–142.

[24] T. Karrer, M. Weiss, E. Lee, and J. Borchers, "Dragon: A direct manipulation interface for frame-accurate in-scene video navigation," in *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, Apr. 2008, pp. 247–250.

[25] C. Liu, A. Torralba, W. T. Freeman, F. Durand, and E. H. Adelson, "Motion magnification," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 519–526, Jul. 2005.

[26] J. Chen, S. Paris, J. Wang, W. Matusik, M. Cohen, and F. Durand, "The video mesh: A data structure for image-based video editing," in *Proceedings of IEEE International Conference on Computational Photography*, 2011, pp. 1–8.

[27] V. Scholz, S. El-Abed, H.-P. Seidel, and M. A. Magnor, "Editing object behaviour in video sequences," *Computer Graphics Forum*, vol. 28, no. 6, pp. 1632–1643, 2009.

[28] A. Rav-Acha, Y. Pritch, D. Lischinski, and S. Peleg, "Evolving time fronts: Spatio-temporal video warping," Hebrew University, Tech. Rep. HUJI-CSE-LTR-2005-10, Apr. 2005.

[29] ——, "Dynamosaicing: Mosaicing of dynamic scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 10, pp. 1789–1801, Oct. 2007.

[30] C. D. Correa and K.-L. Ma, "Dynamic video narratives," *ACM Transactions on Graphics*, vol. 29, pp. 88:1–88:9, Jul. 2010.

[31] E. P. Bennett and L. McMillan, "Computational time-lapse video," *ACM Transactions on Graphics*, vol. 26, pp. 102 – 108, Jul. 2007.

[32] D. B. Goldman, B. Curless, D. Salesin, and S. M. Seitz, "Schematic storyboarding for video visualization and editing," *ACM Transactions on Graphics*, vol. 25, pp. 862–871, Jul. 2006.

[33] Y. Pritch, A. Rav-Acha, and S. Peleg, "Nonchronological video synopsis and indexing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 1971–1984, Nov. 2008.

[34] M. Brown and D. G. Lowe, "Recognising panoramas," in *Proceedings of the Ninth IEEE International Conference on Computer Vision*, 2003, pp. 1218–1227.

[35] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 1.21," http://cvxr.com/cvx, Dec. 2010.

[36] Y. Weng, W. Xu, S. Hu, J. Zhang, and B. Guo, "Keyframe based video object deformation," in *International Conference on Cyberworlds*, 2008, pp. 142–149.

[37] K. Peker, A. Divakaran, and H. Sun, "Constant pace skimming and temporal sub-sampling of video using motion activity," in *IEEE International Conference on Image Processing*, 2001, pp. 414–417.

[38] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala, "Subspace video stabilization," *ACM Transactions on Graphics*, vol. 30, no. 1, p. 4, 2011.

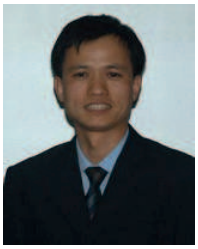
[39] Z. Farbman and D. Lischinski, "Tonal stabilization of video," in *ACM Transactions on Graphics*, vol. 30, no. 4. ACM, 2011, p. 89.

[40] A. Finkelstein, C. E. Jacobs, and D. H. Salesin, "Multiresolution video," in *Computer Graphics (Proceedings of SIGGRAPH 96)*, 1996, pp. 281–290.

Shao-Ping Lu is a Ph.D. candidate at Department of Computer Science and Technology in Tsinghua University. His research interests include image and video process. He is a student member of ACM and CCF.

Song-Hai Zhang obtained his Ph.D. in 2007 from Tsinghua University. He is currently an associate professor of computer science at Tsinghua University, China. His research interests include image and video processing, geometric computing.

Jin Wei is currently a Research Assistant at the Department of Computer Science and Technology, Tsinghua University. He received his MS degree in Computer Science from Tsinghua University and BS degree from Huazhong University of Science and Technology. His research interests include digital geometry modeling and processing, video processing and computational camera.



Shi-Min Hu Shi-Min Hu received the PhD degree from Zhejiang University in 1996. He is currently a professor in the Department of Computer Science and Technology at Tsinghua University, Beijing. His research interests include digital geometry processing, video processing, rendering, computer animation, and computer-aided geometric design. He is associate Editor-in-Chief of The Visual Computer(Springer), and on the editorial boards of Computer-Aided Design and Computer & Graphics (Elsevier). He is a member of the IEEE and ACM.

Ralph R Martin obtained his PhD in 1983 from Cambridge University. Since then he has been at Cardiff University, as Professor since 2000, where he leads the Visual Computing research group. His publications include over 200 papers and 10 books covering such topics as solid modeling, surface modeling, reverse engineering, intelligent sketch input, mesh processing, video processing, computer graphics, vision based geometric inspection and geometric reasoning. He is a Fellow of the Learned Society of Wales, the Institute of Mathematics and its Applications, and the British Computer Society. He is on the editorial boards of "Computer Aided Design", "Computer Aided Geometric Design", "Geometric Models", the "International Journal of Shape Modeling", "CAD and Applications", and the "International Journal of CAD/CAM".