Keyframe-Based Video Object Deformation

Paper ID: 000

Abstract

This paper presents a novel deformation system for video objects. The system is designed to minimize the amount of user interaction, while providing flexible and precise user control. It has a keyframe-based user interface. The user only needs to manipulate the video object at some keyframes. Our algorithm will smoothly propagate the editing result from the keyframes to the rest frames and automatically generate the new video object. The algorithm is able to preserve the temporal coherence as well as the shape features of the video objects in the original video clips. We demonstrate the potential of our system with a variety of examples.

1. Introduction

Video editing is the process of re-arranging or modifying segments of video to form another piece of video. It has long been used in film industry to produce studio-quality motion pictures. With the advent of recent interactive systems to cutout dynamic foreground objects from a video sequence [4, 10, 14, 19], nowaday video editing can be easily performed at object level. A example operation is video object cut and paste, which is widely used in movie production to seamlessly integrate the video object into new scene for special effects.

Recently, many research efforts have been devoted to video object editing. Non-photorealistic rendering (NPR) technique has been generalized to render video objects in cartoon-look style [1, 21]. For the editing of motion, Liu et al. [15] presented a motion magnification technique to amplify the small movements of vide objects. Jue et al. [20] present to use cartoon animation filter to exaggerate the animation of video objects to create stretch and squash effect. All these techniques are very useful to generate new video objects with different styles or motions, and therefore achieve interesting video effects. However, to the best of our knowledge, few work has been done on interactive editing of the shape of video objects, or video object deformation.

In this paper, we present a novel system for video object deformation. Our system has the following features:

- ◇ **Keyframe-based editing:** Our system has a keyframe-based user interface. The user only needs to manipulate the video object at some keyframes. At each keyframe, the user deforms the 2D shapes in the same way as traditional image deformation. Our algorithm will smoothly propagate the deformation result from the keyframes to the rest frames and automatically produce the new video object. In this way, it is able to minimize the amount of user interaction, while providing flexible and precise user control.
- ◇ **Temporal coherence preserving:** Our algorithm can preserve the temporal coherence of the video object in the original video clip.
- ◇ **Shape feature preserving:** Our algorithm is effective in preserving shape features of the video object while generating visually pleasing deformation.

To develop such a system, we need to address the following challenges.

First, we need a shape representation for video objects. Unlike static image object, video object is dynamic in nature. Therefore, its shape is changing with time. We first use the recent interactive video cutout tool [14] to extract the foreground video objects. Then a keyframe-based contour tracking technique [1] is employed to get the boundary Bezier curves of the video object. The shape of the video object is subsequently represented as the region bounded by these boundary curves.

Secondly, to preserve the temporal coherence of video objects, the keyframe editing results need to be smoothly propagated from keyframes to the rest frames such that the whole video object can be deformed consistently. At each keyframe, the user manipulates the deformation handles to deform the shape. We use a least-squares handle propagation algorithm to smoothly propagate the handle editing. Then 2D shape deformation can be applied to each frame given the propagated handle constraints. This handle propagation algorithm can elegantly preserve the temporal coherence because it tries to minimize an energy function that represents the temporal properties of original temporal handle trajectory.

Finally, although the nonlinear 2D shape deformation algorithm described in [22] can preserve shape features and achieve physically plausible results in the interactive keyframe editing, it may generate unsatisfactory results for the rest frames with the propagated handle constraints (see Fig. 4). This is because the deformation solver may stick into a bad local minimum due to the abrupt change of handle positions in these rest frames, while the handle positions are smoothly changed in keyframe editing. Inspired by the subspace method in [8], we propose a dimension reduction technique by projecting the deformation energy onto the subspace formed by the control points of boundary Bezier curves. Performing energy minimization in this subspace greatly improves the stability and convergence of the nonlinear deformation process, and thus leads to much better deformation results.

The remainder of this paper is organized as follows. The following section reviews related work. Section 3 briefly describes the main steps of our video object deformation system. Algorithm details of each step are presented in section 4. In section 5, we introduce how we edit video objects with self-occlusion. Experimental results are shown in Section 6, and the paper concludes with some discussion of future work in Section 7.

2. Related Work

This paper is inspired by a lot of previous works on video object cutout, animation editing and 2D shape editing. We discuss here only those most related works.

Video Object Cutout The most common method for video object cutout might be blue screen matting [18]. However, it is restricted to controlled studio environments. Video object cutout in natural scene is a more challenging task. Image segmentation techniques, such as mean shift and graph cut, have been generalized for video cutout. In [14, 19], a video is over-segmented first and then 3D graph cut is invoked to label the pixels of video to be foreground or background. Since the graph cut is built on the regions from over-segmentation, per-pixel level postprocessing is necessary to get high quality matte. Boundary tracking can also be used to extract video object in natural scene [6, 11], which is also called "rotoscoping". Recently, Agarwala et al. [1] have combined space-time optimization with user guidance for rotoscoping. In contrast, Chuang et al. [4] proposed to track trimap, instead of boundary, to generate high quality video mattes.

Image and Video Object Editing Object level editing gains its popularity because it allows the user to manipulate the image or video at a more meaningful level. Barrett et al. [3] proposed an object-based image editing system, which allows the user to animate the static object in image. A more recent work drives the 2D character in image with 3D motion data [7]. Compared with image object, video object has an additional temporal dimension, which leads to much increased data complexity and new technical challenges. Video tooning [21] generalized NPR to video object to create cartoon-style animation by solving the spatial-temporal coherence problem with mean shift segmentation and mean shift guided interpolation. Agarwala et al. [1] also presented a NPR system for video object. However, their algorithm is based on the roto-curves from tracking. While these two methods focus on changing the style of video object, Liu et al. [15] presented a motion magnification technique to visualize the small movement of video objects, which they call motion layer. To extract motion layers, they first cluster feature points into correlated motions, and then preform segmentation to group pixels into motion layers. More recently, Jue et al [20] propose to use cartoon animation filter to the temporal trajectory to centroid and outline vertices of the video object to create squash and stretch effect. Our video object deformation system provides a more general way to manipulate the shape of video objects.

2D Shape Deformation with Feature Preservation Recent 2D shape deformation algorithms aim to produce visually pleasing results with simple operations and to provide interactive feedback to users. Igrashi et al. [9] developed an interactive system that allows the user to deform a 2D triangular shape by manipulating a few points. To make the deformation *as-rigid-as-possible* [2], they presented a two-step linearization algorithm to minimize the distortion of each triangle. However, it might cause unnatural deformation results due to its linear nature. On the other hand, the algorithm based on moving least squares [17] does not require a triangular mesh and can be applied to general images. The 2D shape deformation algorithm in [22] solved the deformation using nonlinear least-squares optimization. It tries to preserve two geometric properties of 2D shapes: the Laplacian coordinates of the boundary curve of the shape and local areas inside the shape. The resulting system is able to achieve physically plausible deformation results and runs in real time. We therefore use this algorithm for keyframe shape editing.

Our work is also inspired by the recent research trend on generalizing static mesh editing techniques to the editing of mesh animation data [12, 23]. We try to generalize static image object editing techniques to deform video object. This brings several technical challenges as we mentioned in the last section.

3. System Overview

As illustrated in Figure 1, our video object deformation system performs the following steps in a typical editing session:

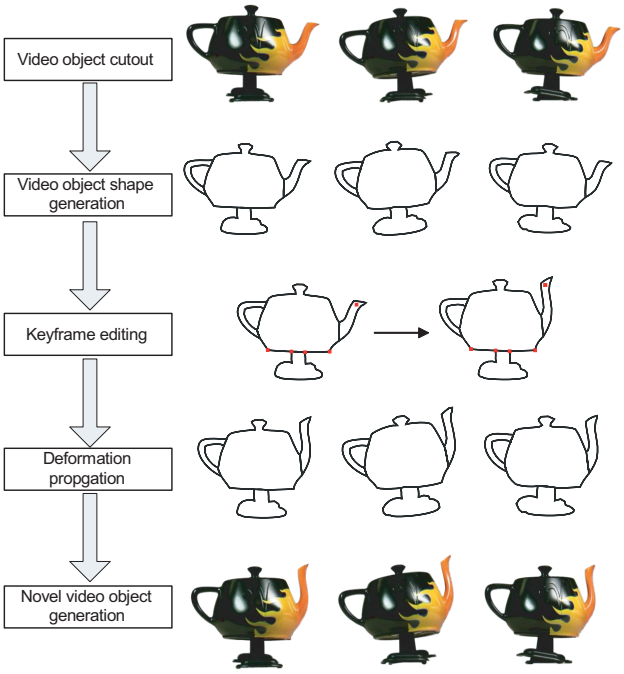

Figure 1. System flowchart. (a) Input video object. (b) Tracking result, note only the boundaries of the region of editing interest are precisely tracked. (c)Interactive keyframe deformation (d) Resulting video object.

1. **Video Object Cutout**: Given an input video, we first extract the foreground object from the video using the segmentation based video cutout technique [14]. The result is a sequence of foreground images with alpha channels indicating the opacity of each pixel.

2. **Video Object Shape Generation**: The key-frame based rotoscoping technique in [1] is used to track the contours of the video object in the foreground images. We first manually draw several Bezier curves along the boundary of the video object at some keyframes. Then the tracking algorithm will locate the optimal position of the curves in the rest frames. These Bezier curves are then organized into a boundary polygon and our algorithm automatically inserts a set of points into the interior region of the polygon and generates a 2D graph by connecting the vertices of the boundary polygon and the inside points (see Section 4.1 for details). Finally, we get a sequence of 2D shapes: $\{S^i, i = 1, ..., N\}$.

3. **Keyframe Editing**: The user can deform the 2D shape S^k at any frame k. The edited frame will subsequently become a keyframe. During deformation, the user only needs to specify deformation handles $\{H_j^k, j = 1, ..., l\}$ at frame k, and then drags the handles to new positions $\hat{H}_j^k$. The 2D shape deformation algorithm [22] is used to automatically deform the shape S^k into $\hat{S}^k$.

4. **Deformation Propagation**: With the user edited handle $\hat{H}_j^k$ at keyframe k, a handle propagation algorithm computes the new position of handle $\hat{H}_j^i$ at each frame i by using a least squares optimization. Then a shape deformation algorithm using dimension reduction is applied at each frame i to meet the new positional constraints from handles $\hat{H}_j^i$ and simultaneously generate the deformed 2D shape $\hat{S}^i$.

5. **New Video Object Generation**: Using the foreground images as textures, we can render the new 2D shapes $\{\hat{S}^i, i = 1, ..., N\}$ to produce the new foreground images, which is a novel video object ready for integration into any new background image or video.

Note that the contour tracking in Step 2 is a challenging task in computer vision because of the complexity of motion. By applying it to foreground images only, we are able to reduce the difficulty greatly. Our system also supports the deformation of video objects with self-occlusions. To achieve this, we allow to model the shape of video object with multiple polygons and the polygons may occlude each other. Please see Section 5 for details.

4. Video Object Deformation

4.1. Video Object Shape Representation

The output of the contour tracking algorithm is the positions of the Bezier curves at each frame. We still need to make use of these curves to construct a 2D shape representation for the video object. Instead of asking the user to do this frame by frame tediously, we first build a 2D graph for the video object with some user interaction at the first frame. Then the algorithm will automatically transfer the graph to the rest frames.

As shown in Figure 2(a), after tracking, the boundary of the walking teapot at the first frame is represented in Bezier curves. In this paper, we use cubic Bezier curves:

$$\mathbf{g}(t) = \mathbf{p}_0(1-t)^3 + 3\mathbf{p}_1 t(1-t)^2 + 3\mathbf{p}_2 t^2(1-t) + \mathbf{p}_3 t^3, \quad (1)$$

where $\mathbf{p}_i$ is the control points, and $t \in [0, 1]$ is a scalar parameter.

The Bezier curves are then automatically discretized into connected line segments by uniformly sampling the parameter t (Figure 2(b)).

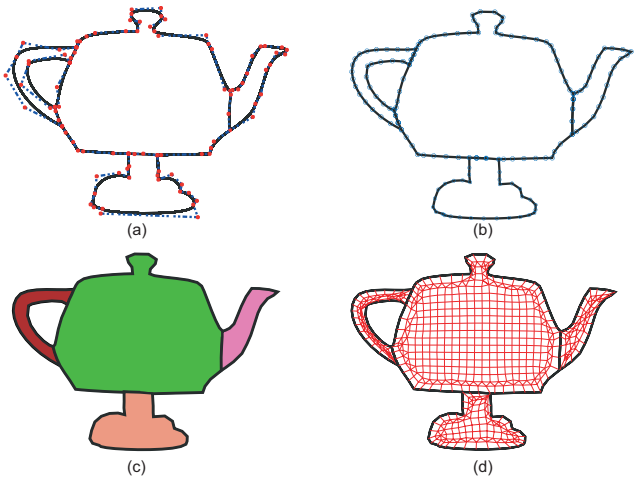

Figure 2. Creation of 2D Graph. (a) 2D shape in bezier curves. The dots in red represent the control points. (b) Discretization of bezier curves. The sampled points are represented in blue dot. (c) Organized polygons (filled in different colors) (d) Final 2D Graph.

To define the interior region of the shape, the user needs to specify how the line segments are arranged into polygons. In Figure 2(c), we organize the line segments into four polygons. In this way, we can manipulate the four components of the teapot: the handle, the body, and feet and the mouth. A few interior points are then inserted into each polygon and then connected to form 2D graphs (Figure 2(d)).

Formally, we can denote the 2D graph as $\{\mathbf{V}^0, \mathbf{E}^0\}$, where $\mathbf{V}^0$ is the set of n vertices in the graph, and $\mathbf{E}^0$ is the set of edges. $\mathbf{V}^0$ consists of two parts: $\mathbf{V}_p^0$, which contains m vertices on the shape boundary, and $\mathbf{V}_g^0$ contains $n-m$ interior vertices of the graph. $\mathbf{V}_g^0$ can be easily represented as a linear combination of $\mathbf{V}_p^0$ by using mean value coordinates [5] computed for each polygon:

$$\mathbf{V}_g^0 = \mathbf{M}_p \mathbf{V}_p^0, \tag{2}$$

where $\mathbf{M}_p$ is a matrix of mean value coordinates.

The transferring algorithm is relatively simple. For each frame i, since each Bezier curve has its corresponding curve in the first frame, we can easily get the boundary polygons (i.e., $\mathbf{V}_p^i$) through sampling the Bezier curves with the same parameters t as in the first frame. Then we directly copy the interior points and edge information from the 2D graph in the first frame, and calculate the interior vertex positions $\mathbf{V}_g^i$ using the same mean value coordinates $\mathbf{M}_p$ computed in the first frame:

$$\mathbf{V}_g^i = \mathbf{M}_p \mathbf{V}_p^i. \tag{3}$$

As a result, video object shape is represented as a sequence of 2D graphs $\{\mathbf{V}^i, \mathbf{E}^i\}, i = 1, ..., N$. These graphs differ in positions, but have the same topology. Now the user can deform the shape at any keyframe using the algorithm in [22].

4.2. Handle Editing Propagation

During keyframe editing, the user selects some vertices as deformation handles and drags them to new positions. The updated positions of the handles will serve as positional constraints to drive the 2D shape deformation algorithm. Therefore, a handle editing propagation algorithm is necessary to smoothly propagate the handle editing result from keyframes to the rest frames such that the 2D shape at each frame can be deformed properly. In order to preserve the temporal properties of the handle in the original sequence of 2D graphs, we formulate the handle propagation as a least squares optimization problem as in [23].

Since a handle may contain multiple vertices, we define a single local coordinate frame for each handle to force all vertices inside this handle to move together rigidly, which prevents severe distortion of the overall shape of the handle. Formally, let us denote a user specified handle i at frame k as $H_i^k = \{\mathbf{c}_i^k, \mathbf{F}_i^k, \{\mathbf{v}_{i_j}^k | j = 0, ..., n_i - 1\}\}$, where $\{v_{i_j}^k | j = 0, ..., n_i - 1\}$ is the set of vertices inside this handle, $\mathbf{c}_i^k$ and columns of $\mathbf{F}_i^k$ define the center and axes of the local coordinate frame. $\mathbf{c}_i^k$ can be the centroid of the vertices and the initial axes can be defined arbitrarily since we only consider the rotation between the initial and altered axes. Note that once a handle is specified at some keyframe by the user, we can easily get a set of corresponding handles in all frames by using the same indices of the vertices inside the handle. The center of all the corresponding handles $\mathbf{c}_i^k, k = 1, ..., N$ forms a curve in temporal domain, and a local coordinate axes $\mathbf{F}_i^k$ is associated with each center.

The temporal properties of the handle is defined as the rotation invariant coordinate of each center in its neighboring local coordinate system. Precisely, for each center $\mathbf{c}_i^k$, we can compute its rotation invariant coordinate using the following formula in the original animation:

$$\mathbf{c}_i^k - \mathbf{c}_i^l = \mathbf{F}_i^l \mathbf{d}_i^{k \to l}, l \in N^k, \tag{4}$$

where N^k represents the index set of the immediate neighbors of $\mathbf{c}_i^k$, $|N^k| \leq 2$, and $\mathbf{d}_i^{k \to l}$ represents the so-called rotation invariant coordinate of $\mathbf{c}_i^k$. It is obvious that $\mathbf{d}_i^{k \to l}$ is the local coordinate of $\mathbf{c}_i^k$ in its immediate neighboring local coordinate frames.

During handle propagation, we try to preserve the $\mathbf{d}_i^{k \to l}$, since it represents the temporal properties computed from the original video. Therefore, we formulate the following

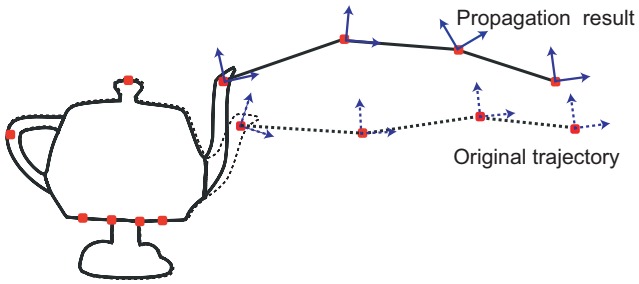

Figure 3. Handle editing propagation. The deformed shape and handle propagation result are represented in solid lines, while the dash lines represents the original shape and original temporal trajectory of handle. Note the frame axes (represented by arrow lines) are associated with handles at each frame.

quadratic energy for handle propagation:

$$\sum_k \sum_{l \in N^k} \|\hat{\mathbf{c}}_i^k - \hat{\mathbf{c}}_i^l - \hat{\mathbf{F}}_i^l \mathbf{d}_i^{k \to l}\|^2 \tag{5}$$

where $\hat{\mathbf{c}}_i^l$ represents the target position of the handle i at frame l, and $\hat{\mathbf{F}}_i^l$ represents the target local coordinate frame axes. Since $\mathbf{d}_i^{k \to l}$ remains the same value computed from Equation (4), the target positions and local coordinate frame axes solved from Equation (5) will have the same temporal properties as those in the input 2D shape animation.

To make Equation (5) a linear least squares problem, we simply interpolate corresponding handle rotations at keyframes over the rest nodes on the temporal curve to get the orientations of all the new local coordinate frames, $\{\hat{\mathbf{F}}_i^k\}$. Handle rotations are represented as unit quaternions, and the logarithm of the quaternions are interpolated using Hermite splines [13]. The user can optionally designate an influence interval for a keyframe to have finer control over the interpolation.

Once these new local frames are known, the equations in (5) over all unconstrained handle centers give rise to an overdetermined linear system and can be solved using least squares minimization. The new world coordinates of the vertices inside each handle at an intermediate frame can be obtained by maintaining their original local coordinates in the new local coordinate frame at that handle.

4.3. 2D Shape Deformation Using Dimension Reduction

Once the handle editing is propagated to all frames, we are ready to independently deform the 2D shape at each frame using the nonlinear shape deformation algorithm of [22]. However, although the algorithm in [22] works well in interactive keyframe editing, it may produce bad results for the rest frames due to the abrupt change of handle positions in these rest frames. Inspired by the subspace method in [8], we propose a dimension reduction technique by projecting the deformation energy onto the subspace formed by the control points of the boundary Bezier curves. Performing energy minimization in this subspace greatly improves the stability and convergence of the nonlinear deformation process, and thus leads to much better deformation results. In the following, we will first briefly describe the deformation energy, and then introduce how it can be represented as a function of the control points of the Bezier curves only. To facilitate discussion, we omit the frame index in the following equations.

The deformation energy in [22] can be written as:

$$\|\mathbf{LV} - \delta(\mathbf{V})\|^2 + \|\mathbf{MV}\|^2 + \|\mathbf{HV} - e(\mathbf{V})\|^2 + \|\mathbf{CV} - \mathbf{U}\|^2, \tag{6}$$

where $\mathbf{V}$ is the point positions of the 2D graph, $\|\mathbf{LV} - \delta(\mathbf{V})\|^2$ is the energy term for Laplacian coordinates preservation, $\|\mathbf{MV}\|^2 + \|\mathbf{HV} - e(\mathbf{V})\|^2$ corresponds to local area preservation, and $\|\mathbf{CV} - \mathbf{U}\|^2$ represents the position constraints from the handles. Please refer to [22] for details on how to compute each term. Note that $\mathbf{U}$ contains the target positions of the handles. In keyframe editing, $\mathbf{U}$ is changed smoothly because the user moves the mouse continuously. In the rest frames, $\mathbf{U}$ is computed from handle propagation and may change abruptly.

The above energy can be simplified into the following formula:

$$\min_{\mathbf{V}} \|\mathbf{AV} - \mathbf{b}(\mathbf{V})\|^2 \tag{7}$$

where:

$$\mathbf{A} = \begin{pmatrix} \mathbf{L} \\ \mathbf{M} \\ \mathbf{H} \\ \mathbf{C} \end{pmatrix}, \mathbf{b}(\mathbf{V}) = \begin{pmatrix} \delta(\mathbf{V}) \\ 0 \\ e(\mathbf{V}) \\ \mathbf{U} \end{pmatrix}.$$

$\mathbf{V}$ consists of two parts: $\mathbf{V}_p$ and $\mathbf{V}_g$. Since $\mathbf{V}_p$ is sampled from the Bezier curves according to Equation (1), it can be represented as a linear combination of the control points of the Bezier curves:

$$\mathbf{V}_p = \mathbf{BP}, \tag{8}$$

where $\mathbf{P}$ is the control point postions and $\mathbf{B}$ is the parameter matrix that maps $\mathbf{P}$ to $\mathbf{V}_p$.

Recall that the interior points $\mathbf{V}_g$ are computed from $\mathbf{V}_p$ using mean value coordinates (Equation (3)), we have:

$$\mathbf{V}_g = \mathbf{M}_p \mathbf{V}_p = \mathbf{M}_p \mathbf{BP}. \tag{9}$$

Therefore, the point positions $\mathbf{V}$ can be represented as a linear combination of the control points:

$$\mathbf{V} = \begin{pmatrix} \mathbf{V}_p \\ \mathbf{V}_g \end{pmatrix} = \begin{pmatrix} \mathbf{B} \\ \mathbf{M}_p \mathbf{B} \end{pmatrix} \mathbf{P} = \mathbf{WP}. \tag{10}$$

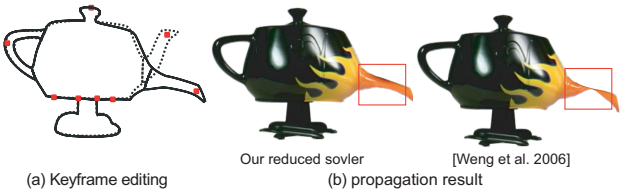

Figure 4. Comparison between the nonelinear 2D shape deformation solver and our dimension reduced solver. (a) The deformed shape at keyframe with the original shape represented in dash lines. (b) propagation result at one frame. The nonlinear 2D shape deformation in [22] causes serious distortion at the mouth of teapot , while result from our dimension reduced solver is natural.

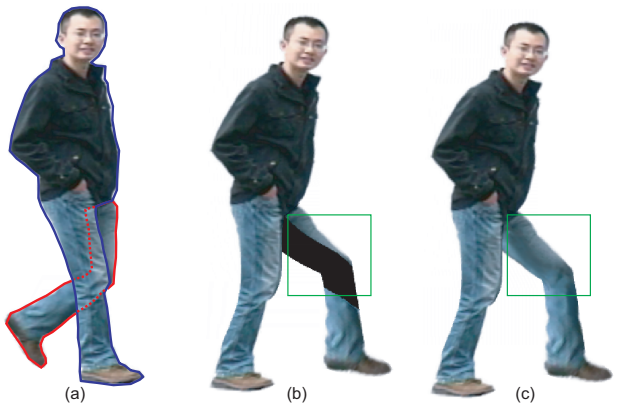

Figure 5. Occlusion example. (a) The rest shape. Lines indicates the polygons we choose for this occlusion example. (b) The editing result without inpainting. Black area indicates the originally occluded region. (c) The editing result with inpainted texture.

Replacing $\mathbf{V}$ with $\mathbf{WP}$ in Equation (7), we get:

$$\min_{\mathbf{P}} \|\mathbf{AWP} - \mathbf{b}(\mathbf{WP})\|^2 \tag{11}$$

This is a nonlinear least squares problem since $\mathbf{b}$ is a nonlinear function dependent on the unknown control point positions. It can be solved using the inexact iterative Gauss-Newton method as described in [8]. Precisely, the inexact Gauss-newton method converts the nonlinear least squares problem in Equation (11) into a linear least squares problem at each iteration step:

$$\min_{\mathbf{P}^{k+1}} \|\mathbf{AWP}^{k+1} - \mathbf{b}(\mathbf{WP}^k)\|^2, \tag{12}$$

where $\mathbf{P}^k$ is the control point positions solved from the k-th iteration and $\mathbf{P}^{k+1}$ is the control point postions we want to solve at iteration $k+1$. Since $\mathbf{b}(\mathbf{WP}^k)$ is known at the current iteration, Equation (12) can be solved through a linear least squares system:

$$\mathbf{P}^{k+1} = (\mathbf{W}^T\mathbf{A}^T\mathbf{AW})^{-1}\mathbf{W}^T\mathbf{A}^T\mathbf{b}(\mathbf{WP}^k) = \mathbf{Gb}(\mathbf{WP}^k). \tag{13}$$

Note that $\mathbf{G} = (\mathbf{W}^T\mathbf{A}^T\mathbf{AW})^{-1}\mathbf{W}^T\mathbf{A}^T$ only depends on the 2D graph before deformation and is fixed during deformation. It can be precomputed before deformation.

The model reduction we use in Equation (10) is based on the matrices $\mathbf{B}$ and $\mathbf{M}_p$. Both have nice properties: each component of the matrices are positive and the sum of each row of the matrices equals to one. Therefore, this dimension reduction greatly reduces the nonlinearity of $\mathbf{b}$ according to the analysis in [8]. Hence the stability of the inexact Gauss-Newton solver is improved significantly.

Figure 4 shows a comparison between the nonlinear deformation solver in [22] and our improved solver. Due to the abrupt change of the handle position, the solver in [22] produces unnatural deformation results, while our solver generates satisfactory result. Please see the companion video for an animation comparison.

5. Handling Video Object with Self-Occlusion

Video objects in real-life often have complex topology. Self-occlusions frequently occur when one part of the object occlude another, especially in articulated video objects. Figure 5 illustrates a simple example. One leg of the character is occluded by by his another leg. It makes the animation of his legs difficult.

To enable editing of video objects with complex topology, our system allows to model the shape of the video object with multiple polygons. These polygons may occlude each other. For each polygon, a depth value is assigned by the user to determine its rendering order. With this setting, the user is able to manipulate the meaningful parts of video object after generating the interior graph for each polygon. However, we still need to solve the following two problems.

First, once the polygons are deformed, some originally occluded regions in the video object may become exposed. However, there is no texture information for these regions in the extracted foreground images. To solve this incomplete texture problem, we adopt existing video inpainting technique [16] to automatically generate textures for these occluded region. Figure 5(c) shows the inpainting result. Note the occluded region of the right leg now is filled with inpainted texture.

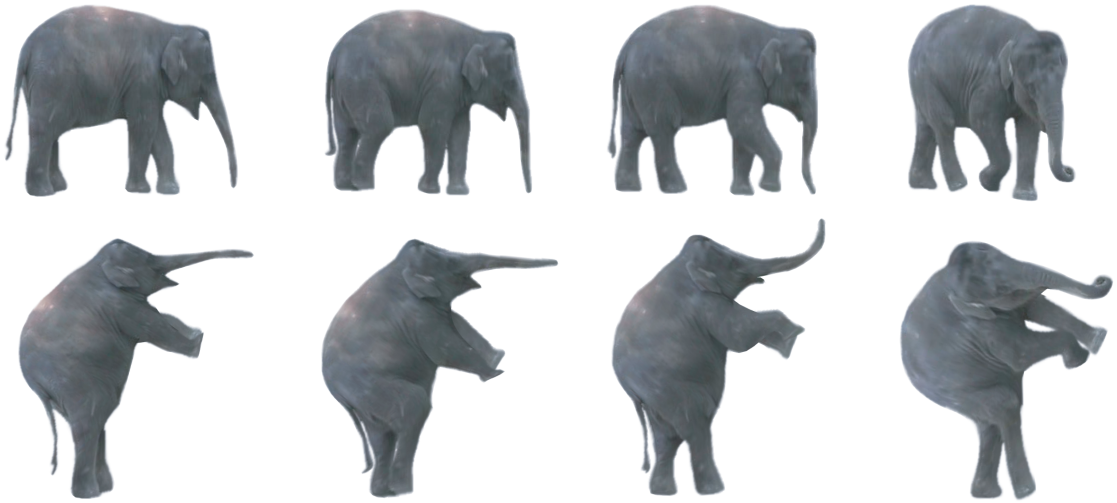

Figure 6. Editing of an elephant video object: from plane walking to stand walking. Top row:original video object. Bottom row:editing result.

Video object	Frames	Bezier curves	Graph vertices	Solving time	Keyframes	Propagation time
Teapot	73	12	698	2.93ms	1	4.69s
Elephant	90	10	1269	4.93ms	7	10.04s
Walking man	61	14	658	2.577ms	6	3.32s
Flame	100	2	519	1.36ms	4	2.998s
Fish	100	5	365	1.44ms	10	3.241s

Table 1. Statistics and timings. Solving time means the time for each iteration. Keyframes indicates the number of frames edited to achieve the result, and the propagation time means the time required to propagate the keyframe editing to the entire sequence.

Secondly, the contour tracking algorithm may output unexpected results when occlusions occur. Although there exist some tracking algorithms that can handle self-occlusions [24], currently we simply decide the positions of the Bezier curves in the occluded regions by interpolating the positions from neighboring keyframes. If the simple interpolation cannot generate satisfactory results, the user can manually adjust the positions of the Bezier curves.

6. Experimental Results

We have implemented our system on a 3.7Ghz PC with 1GB of memory. An editing session of our system has two stages: keyframe editing and deformation propagation. In keyframe editing, our system runs in real-time due to the high speed of our 2D shape deformation solver. In deformation propagation, we propagate the handle editing results from keyframes to the rest frames and then perform offline computation to solve Equation (11) for every frame. The statistics and timing of the two stages are listed in table 1 for the editing results presented in this paper. Please also see the companion video for all deformation results.

Figure 7 demonstrates that our system is very easy to use. In this editing example, the user only needs to set the first frame as keyfame, deforms it into the desired shape, and specifies the influence area of the keyframe to be the entire sequence, our system will automatically generate a novel teapot walking sequence (see the accompanying video for the editing process).

Two more complicated results with self-occlusions are shown in Figure 6 and Figure 8. In Figure 6, an elephant is made stand-up. The large scale deformation in this example is made possible by the power of our dimension reduced 2D shape deformation solver. Figure 8 demonstrates that our system is capable of editing complex motion, like human walking. The input video object is a man walking on the ground, and we make it walk upstairs.

The candle flame in Figure 9 exhibits highly nonrigid motions. After editing, all subtle motions of the flame are well preserved. This clearly demonstrates that our system can preserve the temporal coherence of the video object in the original video clip. In Figure 10, the motion of a swim-

ming fish video object is exaggerated to create motion magnification like effect.

7. Conclusion and Future Work

We have presented a novel deformation system for video objects. The system is designed to minimize the amount of user interaction, while providing flexible and precise user control. It has a keyframe-based user interface. The user only needs to deform the video object into desired shape at some keyframes. Our algorithm will smoothly propagate the editing result from the keyframes to the rest frames and automatically generate the new video object. The algorithm can preserve the temporal coherence as well as the shape features of the video objects in the original video clips.

Although our deformation system can generate some interesting results, it is still immature and has several restrictions. First, the handle propagation algorithm requires the 2D shapes of video object to have the same topology, which greatly restricts the motion complexity of the video object. Our method will not work if the shape boundary of video object experiences topology change in motion. Secondly, the 2D handle editing result propagation algorithm does not take the perspective projection effect into consideration, it may cause undesirable deformation results.

References

[1] A. Agarwala, A. Hertzmann, D. H. Salesin, and S. M. Seitz. Keyframe-based tracking for rotoscoping and animation. *ACM Trans. Graphics*, 23(3):584–591, 2004.

[2] M. Alexa, D. Cohen-Or, and D. Levin. As-rigid-as-possible shape interpolation. In *SIGGRAPH 2000 Conference Proceedings*, pages 157–164, 2000.

[3] W. A. Barrett and A. S. Cheney. Object-based image editing. *ACM Transactions on Graphics*, 21:777–784, 2002.

[4] Y.-Y. Chuang, A. Agarwala, B. Curless, D. H. Salesin, and R. Szeliski. Video matting of complex scenes. *ACM Trans. Graphics*, 21(3):243–248, 2002.

[5] M. S. Floater. Mean value coordinates. *Comp. Aided Geom. Design*, 20(1):19–27, 2003.

[6] M. Hoch and P. C. Litwinowicz. A semi-automatic system for edge tracking with snakes. *International Journal of Computer Vision*, 1(4):321–331, 1996.

[7] A. Hornung, E. Dekkers, and L. Kobbelt. Character animation from 2d pictures and 3d motion data. *ACM Transactions on Graphics*, 26:1–9, 2007.

[8] J. Huang, X. Shi, X. Liu, K. Zhou, L. Wei, S. Teng, H. Bao, B. Guo, and H.-Y. Shum. Subspace gradient domain deformation. *ACM Trans. Graphics*, 25(3):1126–1134, 2006.

[9] T. Igarashi, T. Moscovich, and J. F. Hughes. As-rigid-as-possible shape manipulation. *ACM Trans. Graphics*, 24(3):1134–1141, 2005.

[10] N. Joshi, W. Matusik, and S. Avidan. Natural video matting using camera arrays. *ACM Trans. Graphics*, 25(3):779–786, 2006.

[11] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contor models. *International Journal of Computer Vision*, 1(4):321–331, 1987.

[12] S. Kircher and M. Garland. Editing arbitrarily deforming surface animations. *ACM Trans. Graphics*, 25(3):1098–1107, 2006.

[13] D. H. U. Kochanek and R. H. Bartels. Interpolating splines with local tension, continuity, and bias control. In *SIGGRAPH 1984 Conference Proceedings*, pages 33–41, 1984.

[14] Y. Li, J. Sun, and H.-Y. Shum. Video object cut and paste. *ACM Trans. Graphics*, 24(3):595–600, 2005.

[15] C. Liu, A. Torralba, W. T. Freeman, F. Durand, and E. H. Adelso. Motion magnification. *ACM Transactions on Graphics*, 24(3):321–331, 2005.

[16] K. A. Patwardhan, G. Sapiro, and M. Bertalmio. Video inpainting of occluding and occluded objects. In *Proceedings of IEEE International conference on Image Processing*, pages 69–72, 2005.

[17] S. Schaefer, T. McPhail, and J. Warren. Image deformation using moving least squares. *ACM Trans. Graph.*, 25(3):533–540, 2006.

[18] A. R. Smith and J. F. Blinn. Blue screen matting. In *SIGGRAPH 1996 Conference Proceedings*, pages 259–268, 1996.

[19] J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. *ACM Trans. Graphics*, 24(3):585–594, 2005.

[20] J. Wang, S. Drucker, M. Agrawala, and M. F. Cohen. The cartoon animation filter. *ACM Trans. Graphics*, 25(3):1169–1173, 2006.

[21] J. Wang, Y. Xu, H.-Y. Shum, and M. F. Cohen. Video tooning. *ACM Trans. Graphics*, 23(3):574–583, 2004.

[22] Y. Weng, W. Xu, Y. Wu, K. Zhou, and B. Guo. 2d shape deformation using nonlinear least squares optimization. *The Visual Computer*, 22(9-11):653–660, 2006.

[23] W. Xu, K. Zhou, Y. Yu, Q. Tan, Q. Peng, and B. Guo. Gradient domain editing of deforming mesh sequence. In *SIGGRAPH 2007 Conference Proceedings*, to appear.

[24] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. 38(4):1–45, 2006.

Figure 7. Editing of teapot. (a) Keyframe editing. Left is the rest shape. (b) Propagation result. Only one keyframe is edited in this result.

Figure 8. Walking editing. (a) 2 frames in original video. (b) The editing result.

Figure 9. Editing of a flame video object. Top row: Original video object. Bottom row: Editing result.

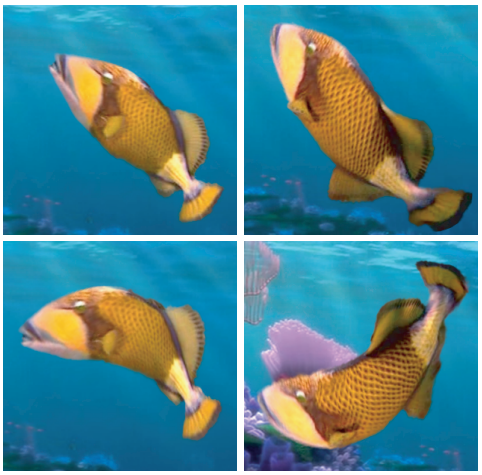

Figure 10. Editing of a fish video object. Top row: Original video object. Bottom row: Editing result.