Object Virtual Viewing
Using Adaptive Tri-View Morphing

Pin Chatkaewmanee
Matthew N. Dailey

Computer Science and Information Management
School of Engineering and Technology
Asian Institute of Technology
P.O. Box 4, Klong Luang, Pathumthani, 12120
Thailand
st102175@ait.ac.th, mdailey@ait.ac.th

March 12, 2014

Abstract
This paper proposes a new technique for generating an arbitrary virtual view of an object of interest given a set of images taken from around that object. The algorithm extends Xiao and Shah’s tri-view morphing scheme to work with wide baseline imagery. Our method performs feature detection and feature matching across three views then blends the real views into a virtual view. Tri-view morphing by itself is realistic when occlusion across the three views is minimal, but when it is applied to cases of more complex objects and wide baselines, occlusions lead to significant artifacts. We propose a new adaptive algorithm to solve these problems
by 1) segmenting the views into object and background, 2) obtaining fine-grained correspondences across the three views, 3) constructing, when a border point in one view is occluded in one or two of the other views, a virtual correspondence for that point, and 4) synthesizing novel views using barycentric interpolation and automatic elimination of occluded polygons. The result is a system allowing smooth and realistic animation of the virtual object over arbitrary viewing paths.

1 Introduction

Robust reconstruction of three dimensional models from two dimensional images is one of the great open problems of computer vision [1].

Some of the best work to date performs structure from motion using a hand-held camera and continuous capture [2,3], reconstruction using structured lighting with multiple cameras in fixed positions [4,5], three dimensional (3D) modeling of modifiable building interior floor plans for virtual reality [6] incorporating both simple planar geometry [7] and unstructured polygonal geometry [8].

With the advent of large camera arrays [25], it is now possible to use image-based techniques to render high-quality virtual views based on texture without explicit 3D object models.

In some applications, such as morphing one object into another in a movie scene or rotating a virtual object around some axis, we may only want to see a virtual view in which the camera position interpolates two actual pictures of the same object. This is called image morphing [9,10]. Two-view morphing is the special case of image morphing to produce a smooth transition between two different objects or two views of the same object. Following Xiao and Shah [11], we can write the two-view morphing method as
\[ v_{\text{syn}} = (1 - c)v_1 + cv_2, \]  

(1)

where \( v_1 \) and \( v_2 \) are the color or position of a pixel in input view 1 and 2, respectively. \( v_{\text{syn}} \) is the color or position of the pixel in the resulting blended virtual view, and \( c \) is a control variable varying between 0 (\( v_{\text{syn}} \) is identical to \( v_1 \)) and 1 (\( v_{\text{syn}} \) is identical to \( v_2 \)).

The special case of view morphing [12–14] interpolates between different views of the same object, so that synthetic views appear as if rendered from a novel camera position. Xiao and Shah’s tri-view morphing [11, 15, 16] method improves upon simple two-view morphing by changing from linear interpolation to barycentric interpolation with three variables to control the barycentric magnitudes of the three input views, as follows:

\[ v_{\text{syn}} = c_1v_1 + c_2v_2 + c_3v_3. \]  

(2)

In the equation, \( c_1, c_2, \) and \( c_3 \) vary between 0 and 1 with \( c_1 + c_2 + c_3 = 1 \), \( v_{\text{syn}} \) is the position or color in the synthetic view, and \( v_1, v_2, \) and \( v_3 \) are the corresponding position and color in view 1, 2 and 3, respectively.

Tri-view morphing works very well when nearly all feature points appear in all three input views and object border features in one view are not occluded in other input images.

We propose to address the issue of tri-view morphing over all possible views of a complex object by using much wider baselines along with more knowledge of the 3D structure of the object under view. Our method, wherever possible, automatically detects regions that are occluded in one view and only selects views in which the region is visible when synthesizing virtual views. The result is a system allowing smooth and realistic animation of the virtual object over
arbitrary viewing paths.

The main contribution of the paper is thus to generalize image-based rendering algorithms to wider baselines than has been previously possible. Our method neither performs compute-intensive explicit 3D object modeling nor requires a dense camera array to generate synthetic views. Instead, the method obtains correspondences between triangles over three views and performs barycentric view morphing, automatically eliminating the triangles corresponding to hidden surfaces in each view.

2 Adaptive tri-view morphing

We developed the adaptive tri-view morphing algorithm with two design criteria in mind: computational efficiency and photorealistic quality. We first avoid computationally intensive processing by performing image-based rendering rather than detailed 3D modeling and texture mapping. To achieve photorealistic quality without large computational demands, we extend the traditional method with the capability to automatically filter out triangles representing hidden surfaces, preventing these surfaces from creating artifacts during the creation of a synthetic view. This allows the algorithm to achieve superior realism when synthesizing portions of the view near the object border or hidden surfaces.

The input is two or more sets of images, taken around an object placed on a flat surface. Fig. 1 shows two sets of images, each containing four views of an object. The user can then select a set of three views forming a triangle within which any view can be synthesized as shown in Fig. 1(b).
Figure 1: Image capture positions and tri-view sets. (a) Camera positions at four azimuths and two inclinations with respect to an object of interest. (b) Eight sets of three views.
2.1 Boundary and feature detection

We assume that the object is first placed on a matte background (for an explanation, see Fig. 1(b) view 6) and that the user manually selects an arbitrary point on the background. We calculate the average color $C_b$ in a $5 \times 5$ neighborhood of the selected point and then perform a flood fill operation from that point. A neighboring pixel is included in the flood fill if its red, green, and blue (RGB) color vector lies within distance $D_0$ of $C_b$, measured by Euclidean distance. The result is an initial binary background mask, as shown in Fig. 2(a) ($D_0 = 50$).

To detect the object boundary, we first use morphological opening to delete small objects followed by morphological closing to fill gaps in the foreground mask boundary and small holes in the foreground [22–24]. Then we eliminate any remaining holes in the foreground, as shown in Fig. 2(b).

We find the boundary features by expanding the boundary region by a customizable width and then selecting feature points that are located in the boundary region. The boundary features are on the dashed line in Fig 2(d).

Our feature detection algorithm looks for local maxima of the Harris “cornerness” criterion but rather than applying the criterion to the whole image, we only apply it to those points detected as edge pixels by the Canny edge detector [21]. Finding local Harris maxima only on Canny edges emphasizes finding reliable features on shape or texture discontinuities. The result of Canny-Harris features are shown in Fig. 2(c). The result after filtering out features in the background is shown in Fig. 2(d).

2.2 Feature point matching

To describe our algorithm, we let the set of feature points in view $i$ be $F_i = \{p_{i,1}, \ldots, p_{i,n_i}\}$. To find the point in view $k$ corresponding to point $p_{i,k}$ in view
Figure 2: Boundary and feature point detection. (a) Initial foreground mask. (b) Object boundary obtained from (a) by morphological operations. (c) Feature points found by extracting Canny edges and Harris corners. (d) Results after background feature point elimination.
Figure 3: Canny-Harris feature point matching. (a) One view of an object with 25×25 pixel region around feature point (yellow box) to be detected in other views. (b) Second view with corresponding region highlighted (yellow box). (c) Image patch for feature point in part (a) with transformed versions of corresponding region in second view. The smallest SSD error is highlighted with a red box.

In the process, we assume that the neighborhood around the corresponding feature points are related by rotation and scaling. We use the standard sum-squared distance (SSD) measure of neighborhood similarity along with a search over matching scales $s$ and matching angles $\alpha$, as shown in Fig. 3. The smallest error match is selected if the error value is less than a user-specified threshold.

After combining consistent correspondences across views 1, 2, and 3, we obtain a set of triples of indices $C = \{(q_1, r_1, s_1), (q_2, r_2, s_2), \ldots\}$ where each triple $(q, r, s)$ indexes points in $F_1$, $F_2$, and $F_3$, respectively, indicating that point $p_{1,q}$ was found to correspond to point $p_{2,r}$, and that point $p_{2,r}$ was found to correspond to point $p_{3,s}$. That is, both $p_{1,q}$ and $p_{3,s}$ correspond to the
same point $p_{2,r}$ in view 2.

2.3 Epipolar geometry estimation

We use the Affine Scale Invariant Feature Transform (ASIFT) [26, 27] to get a set of putative correspondences and a much more accurate estimate of the fundamental matrices than would be possible from the initial Canny-Harris feature point correspondence algorithm. We then use the accurate epipolar geometry estimated from ASIFT correspondences to refine the set of matching Canny-Harris feature points. This procedure is described in Section 2.4.

2.4 Hidden feature estimation

In two-view matching, a visible feature point corresponding to an occluded point in the other view may be either within the object boundary or interior to the object boundary.

In three-view matching, a set of three corresponding points is simply a combination of two two-view correspondences, as shown in Fig. 4(a). Across all three views, we can classify detected feature points into three categories:

- **Free features** are detected in only one view and hidden or not detected in the other two views.

- **Two-view corresponding features** are detected across two views and hidden or not detected in a third view.

- **Three-view corresponding features** are detected and in correspondence across all three views.

Accurate synthesis requires that we accurately estimate correspondences across all three views for border points and points near occlusions. Interior
Figure 4: Feature estimation. (a) 1) Free features (red crosses) have no correspondences in other views. Green circle gives example. 2) Two-view corresponding features (red crosses with only a yellow or green square in view 2) are invisible or not in correspondence in one of the three views. Red and yellow circles give examples. 3) Three-view corresponding features (red crosses with both yellow and green square in view 2) are matched across all three views. Blue circles give examples. (b) Minimal estimation of hidden features. See red dashed circles for examples. (c) Full estimation of all hidden features across all three views.
points on planar surfaces can safely be ignored. For points with hidden correspon-
dences near occlusion boundaries, the procedure depends on whether the point in question is a free feature or a two-view correspondence. Fig. 4 shows an example of a three-view correspondence, an interior free feature, an interior two-view correspondence, and a boundary two-view correspondence. We use a user-supervised procedure to automatically estimate hidden points where possible and request user assistance where necessary. First, as already discussed, we estimate the fundamental matrix for each of the three image pairs using the correspondences obtained from ASIFT and the Canny-Harris features already matched. The fundamental matrices can then be used to constrain or identify hidden point locations corresponding to the free features and two-view corresponding features. In the case of two-view corresponding features, the single corresponding hidden point in the third view can be identified as the intersection of the epipolar lines for the two corresponding points in the first two views. In the case of free features, the corresponding hidden points in the other two views can be constrained to a line, and the user can be requested to manually estimate the position of the corresponding hidden point along the epipolar line. We use this process, and every time the user supplies a new correspondence, we reestimate the related fundamental matrix. This reestimation procedure continues to improve the accuracy of the fundamental matrices, especially near the object boundaries. A detailed example is given in Section 3.

2.5 Boundary conversion

In the next step, we will perform boundary estimation, triangulate the interior of the boundary in a reference view, and then apply the resulting triangulation to the other two views. In the case of an object with a convex boundary, we could simply use the well-known Delaunay method to find a set of triangles interior to
the object’s border. On the other hand, for non-convex shapes, we cannot use the Delaunay triangulation directly because the convex hull of the interior points of a non-convex shape might cross the object’s border, combining object texture and background texture in the same triangle. To avoid this problem, when we need to triangulate a non-convex shape, we will first convert the non-convex region into a set of convex subregions using Algorithm 1.

**Algorithm 1 NONCONVEXTOTRANSFORM**

**Input:** $P$, a clockwise sequence of border points.

**Output:** $C$, a set of convex border point sequences.

$C \leftarrow \emptyset$

$n \leftarrow |P|$

while $n \geq 3$ do

{Find the first clockwise sequence of border points}

$i \leftarrow 1, j \leftarrow 2, k \leftarrow 3$

$m \leftarrow n$

while $\triangle(p_i, p_j, p_k)$ is counter-clockwise and $m > 0$ do

$i \leftarrow j, j \leftarrow k, k \leftarrow (k \mod n) + 1$

$m \leftarrow m - 1$

end while

if $m = 0$ then

  return $C$;

end if

{Adjust $P$ so that index $i$ is first}

$P \leftarrow (p_i, p_{i+1}, \ldots, p_n) \parallel (p_1, p_2, \ldots, p_{i-1})$

{Find maximum-length convex set beginning with $p_1$}

$T \leftarrow (p_1, p_2)$

$i \leftarrow 3$

while $i \leq n$ and $T \parallel (p_i)$ is convex do

  $T \leftarrow T \parallel (p_i)$

  $i \leftarrow i + 1$

end while

{Add convex set $T$ to $C$}

$C \leftarrow C \cup \{T\}$

{Remove points from $P$}

$P \leftarrow (p_n, p_{n-1}, \ldots, p_i)$

$n \leftarrow |P|$

end while

return $C$;
2.6 Triangulation across three views

Now we specify the triangulation method in detail. Suppose we have a set of corresponding points across all three views, with the hidden points marked. Next, we must find a consistent triangulation of the points that are visible in all three views such that all the triangles are interior to the object boundary while maximizing the area of the triangulated region.

Since we have correspondences between views 1 and 2 as well as between views 2 and 3, we choose view 2 as the reference view. To find the boundary of the points that are visible across all three views, we first compute the Delaunay triangulation of the view 2 points that are visible in both view 1 and view 3. An example of the results of this step is shown in Fig. 5(b). Since for non-convex shapes, the convex hull of the points in correspondence might not lie entirely interior to the object boundary, we next eliminate any edges crossing the boundaries of the object and any edges exterior to the boundary (Fig. 5(c)). We next run Algorithm 1 to break the visible boundary into convex subregions. We then add and triangulate all visible points with correspondences that are hidden in some other view. For points outside the visible boundary, we add all triangles that can be added without an edge crossing the boundary. For interior points, we obtain the triangles using the Delaunay method restricted to the local convex region previously calculated. We next obtain a sequence of boundary points visible across all three views (the dashed line in Fig. 5(d)) and a triangulation of all the visible points in each view such that corresponding points across different views are triangulated the same way.

As a last step, we triangulate hidden feature points in every view based on the previously estimated hidden point positions, then for each view, we add additional triangles based on the new corresponding points according to Algorithm 2. There are two main cases for each point: whether it is interior to
Figure 5: Example of triangulation method for non-convex shapes. (a) Example set of boundary and interior points in view 2. ○ points represent boundary features; × points represent features visible in all three views. (b) Results of Delaunay triangulation of the visible points (marked by × in panel (a)). (c) Edges remaining after rejecting Delaunay edges which cross or are outside of the object boundary. (d) Final boundary of object region (dashed line) after adding one new edge.
Figure 6: Adding new triangles for hidden points with correspondences in other views. (a) An interior point (×) visible in the current view that is hidden in one or more of the other views. (b) Final boundary after retriangulation based on the new point. (c) Triangles considered for addition based on the new exterior point (×). (d) Final boundary of the object, which is changed by adding one exterior point.
an existing triangle (e.g., the cross point in Fig. 6(a)) or exterior to all existing triangles (e.g., the cross point in Fig. 6(c)).

Algorithm 2: AddTriangles

**Input:** Set of existing triangles $T$; Set of all object boundary points $B$; Point under consideration $p$.

**Output:** New set of triangles $T'$.

1. Initialize output set to the input set $T' \leftarrow T$
2. Calculate the boundary of $T' \leftarrow C$
3. If $p$ is interior to $\triangle t_i$ in $T$ then
   - Add three triangles $T' \leftarrow T' \cup \{ \triangle(t_{i,1}, t_{i,2}, p), \triangle(t_{i,2}, t_{i,3}, p), \triangle(t_{i,3}, t_{i,1}, p) \}$
   - Remove old triangle $T' \leftarrow T' - \triangle t_i$
4. Else (p is exterior to all triangles in $T$)
   - For all $i$, $1 \leq i \leq |C|$
     - $\triangle t_i \leftarrow \triangle(c_{i}, p, c_{(i \mod |C| + 1)})$
     - If no edge of $\triangle t_i$ crosses any edge in $B$ or $C$ then
       - $T' \leftarrow T' \cup \{ \triangle t_i \}$
   - End if
   - End for
5. End if
6. Return $T'$

2.7 Synthetic view blending

To construct synthetic views, we first allow the user to select a position $\vec{v}_s$ in a triangular area using a graphical user interface (GUI) application and calculate the barycentric coefficients $c_1$, $c_2$, and $c_3$ for $\vec{v}_s$ (see Equation 2).

In order to blend the views to create the synthetic view, suppose that $p_{1,i}$, $p_{2,i}$, and $p_{3,i}$ are corresponding features in the three views, with $i = 1, 2, \ldots, n$ where $n$ is the number of corresponding features. We will use $p^*_i$ to denote the corresponding location in the synthetic view, and we will also use the set of triangles $T$ output by the previous (three-view triangulation) step. The procedure is as follows:
1. Calculate virtual points $p^*_i$ in the synthetic view using the barycentric view magnitudes $c_1, c_2, \text{and } c_3$.

2. Filter out the triangles from $T$ which are invisible in the synthetic view using the ordering of the vertices to get result set $N$.

3. Calculate the barycentric magnitudes and colors for each point in each triangle in $N$.

Our method allows for automatic transition between image triples as needed to obtain smooth morph sequences incorporating more than three images. We maintain the relationships between each triple of images and automatically disregard the unneeded image in the old triple and incorporate the needed image in the new triple. A schematic of the jump from one image triple to another is shown in Fig. 7(a), and four morph images from such a sequence for the book shape are shown in Fig. 7(b)–(c).

### 3 Results

In this section, we provide experimental result for a simple shape (the book from the previous section) and a complex shape (a Vietnamese toothpick box). We will demonstrate that our method provides photorealistic blending of images acquired at large baselines with significant distortion and occlusion of image regions between views. Standard morphing methods are incapable of properly handling regions that are visible in the synthetic view but hidden in one or more views being blended, leading to significant artifacts with wide-baseline imagery. Our method obtains a very accurate estimate of the epipolar geometry and provides special handling to ignore triangles that are hidden in some of the views being blended, leading to more photorealistic synthesis. This allows us to
Figure 7: Example results of automatic transition between image triples to create a long smooth morph sequence. (a) Jumping between image triples. (b) Synthetic views based on image triple (1, 5, 2). (c) Synthetic views based on image triple (5, 2, 6).
achieve superior realism when synthesizing portions of the view near the object border or interior hidden surfaces.

For the book shape, as shown in Fig. 1, we acquired eight images, four images at each of two heights, and grouped the images into triples. We then paired the images in each triple as shown in Fig. 1(b). Across all 8 views, we obtained 437 Canny-Harris points including 79 points on the borders.

In the feature correspondence step, we obtained 143 pairwise point correspondences across eight pairs of views after matching and outlier rejection using random sample consensus (RANSAC) for fundamental matrix estimation. Of these 143 correspondences, 24 were incorrect but consistent with the epipolar geometry. We manually removed these incorrect correspondences to obtain 119 correct pairwise correspondences. For every feature point that should have been matched with a corresponding feature point that was not correctly matched in the previous step, we scanned for the missing feature point near the appropriate epipolar line, requesting the user to confirm each correspondence manually.

In the hidden feature estimation step, we added 36 hidden features with their correspondences to obtain a total of 188 three-view correspondences across all eight image triples.

After triangulation of the visible features, over all 8 images, we obtained 245 triangles from the points visible across all three views. After adding triangles for the hidden points as described in Section 2.6, we obtained a total of 287 triangles. Sample blending results are shown for the book shape in Fig. 8(a).

To compare our results against those that would be produced by a standard view morphing method such as that of Xiao and Shah (2004), we used the same barycentric morphing algorithm used to produce Fig. 8(a) with the triangles obtained using Xiao and Shah’s dense trinocular stereo correspondence algorithm. The results for the same book views are shown in Fig. 8(b). Xiao and
Figure 8: Sample morph results for book shape. (a) Two sample synthetic views. (b) Resulting syntheses for the same views as in (a), without the user interface for manual addition of feature points near the image boundary.

Shah’s system requires that the user correct any occlusion problems by manually specifying point correspondences near the occlusions. Without this manual intervention, surfaces that are hidden in one of the three views lead to artifacts (in the case of the book shape, missing surfaces).

For the complex non-convex toothpick box model, we acquired 12 images at each of two heights, rather than eight images at each height as for the book object. We performed the same steps of automatic feature correspondence estimation, manual removal of incorrect correspondences, and completion of miss-
Figure 9: Sample results for a complex shape (a toothpick box). (a) Results after triangulating only the features visible in all three views. (b) Results after add correspondences for all hidden feature points. (c) Sample synthetic views. (d) Resulting synthetic views using the Xiao and Shah algorithm for the same barycentric magnitudes as in (c), without the user interface for manual addition of feature points near the image boundary.
ing correspondences to obtain the triangulations of visible features shown in Fig. 9(a). After adding the hidden points needed to complete each three-view correspondence, we obtained complete triangulations as shown in Fig. 9(b). Synthetic views after blending are shown in Fig. 9(c), in comparison to the results of Xiao and Shah’s method, shown in Fig. 9(d).

For videos exhibiting continuous changes of barycentric magnitudes across every triple in 360° sweeps around the objects, please see http://www.cs.ait.ac.th/~mdailey/Morph/.

4 Conclusion

In this paper, we have proposed and evaluated a method for automated barycentric interpolation between three views of the same object. By using multiple sets of three views, animations of synthesized views of objects from arbitrary points of view can be constructed. Our method is based on Xiao and Shah’s three-view morphing algorithm but achieves improved results in cases of complex objects with wide baselines between the views.

The small baseline assumption is relevant when we need to perform 3D object reconstruction from a sequence of video frames and the aim is to build a real 3D object model. In wide baseline view morphing of complex objects, however, occlusions are much larger, so a technique that is aware of point visibility in the virtual view is required. We solve the problem by obtaining correspondences across all three views for a set of triangles that covers the entire object in all three views. Occluded triangle vertices (hidden points) are estimated automatically when possible and with user assistance when not possible. Once a complete set of correspondences is obtained, triangles that are visible in all three original views can be blended directly. Visibility is easy to determine, because a visible triangle with vertices in a clockwise ordering will map to a triangle
with vertices in a counter-clockwise ordering when the triangle is not visible. Triangles that are visible in some views but occluded in the desired virtual view can be automatically detected and ignored during the blending process.

With this work, the principle of view morphing using barycentric interpolation is applicable not only to simple scenes or narrow baselines but also to wide baseline views of complex objects.

Another important issue that arises in view morphing is estimation of the fundamental matrices or trifocal tensor. If the object in question has insufficient texture, the epipolar geometry will be insufficiently accurate, causing blending artifacts. Our method allows for user assistance in specifying corresponding hidden points only until the epipolar geometry is accurate enough to allow precise, automatic identification of hidden point locations.

We provide a detailed evaluation of the method on both simple and complex objects, and we find that the method is much better able than methods not explicitly handling occlusion regions to deal with large occlusions occurring due to wide baselines. The results confirm that adaptive tri-view morphing performs better than existing methods with wide baselines for both simple shapes and complex shapes. The lack of distortion in our results shows that the method is able to appropriately select the triangle texture to display from barycentric interpolation across three views without 3D model information. Brief animations at the Web site mentioned in Section 3 demonstrate the ability of the method to blend continuously both within triplets of views and between neighboring triplet views.

There are a few limitations to our method. First, as already discussed, we require initial user assistance during the hidden feature point estimation stage. We also assume that the scene consists of only a single object without holes. Finally, our method will not perform well on objects composed of smooth curved
textureless surfaces.

In future work, we hope to eliminate some of these limitations, e.g., allowing synthesis of more than one object in arbitrary viewing conditions. This would require accurate modeling of the geometry of the scene at acquisition time and appropriate transformation of the coordinate systems in which the objects are shown. Hardware for automatic acquisition of images including a tiltable turntable would be useful and would invite exploration of active methods to select the most efficient set of images to acquire.

References


25


