Animating Animal Motion from Still

Xuemiao Xu[1] Liang Wan[1,2] Xiaopei Liu[1] Tien-Tsin Wong[1] Liansheng Wang[1] Chi-Sing Leung[2]
[1]The Chinese University of Hong Kong [2]City University of Hong Kong

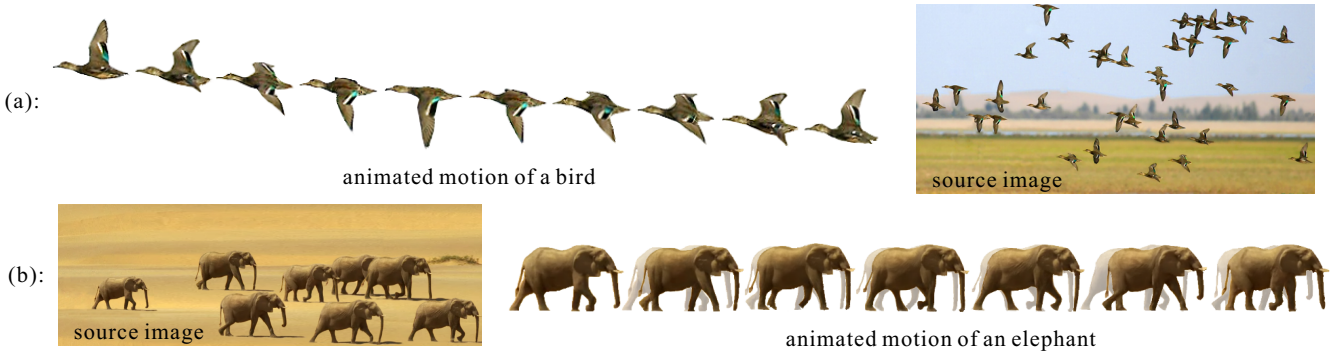


Figure 1: Given still pictures of animal groups, we can infer the motion of animals and perform realistic animations, such as (a) the animated motion of a bird and (b) the animated motion of an elephant. In (b), we also superimpose the adjacent frames to illustrate the motion.

Abstract

Even though the temporal information is lost, a still picture of moving animals hints at their motion. In this paper, we infer motion cycle of animals from the "motion snapshots" (snapshots of different individuals) captured in a still picture. By finding the motion path in the graph connecting motion snapshots, we can infer the order of motion snapshots with respect to time, and hence the motion cycle. Both "half-cycle" and "full-cycle" motions can be inferred in a unified manner. Therefore, we can animate a still picture of a moving animal group by morphing among the ordered snapshots. By refining the pose, morphology, and appearance consistencies, smooth and realistic animal motion can be synthesized. Our results demonstrate the applicability of the proposed method to a wide range of species, including birds, fishes, mammals, and reptiles.

Keywords: still picture, animal group, motion cycle, motion inference, consistency refinement

1 Introduction

Animal groups are ubiquitous in nature. Considering a picture of moving animals of the same species (e.g. Figures 1(a) & (b)), it is not difficult to observe that in one single picture, the animal motion can be encoded in different individuals, due to the motion asynchronism of different individuals in the group. Although individuals may vary morphologically, their motion snapshots suggest how this species moves. This observation motivates us to deduce and synthesize the motion of an animal group from a still picture, by *ordering* and *morphing* the motion snapshots captured in one single picture. To the best of our knowledge, the proposed work is the first attempt to animate a still picture of moving animals.

Previous work on animating a still picture includes physically simulating natural phenomena [Shinya et al. 1999][Chuang et al. 2005], navigating inside the picture by reconstructing scene geometry [Horry et al. 1997][Criminisi et al. 2000][Oh et al. 2001], and deforming static object using shape deformation [Litwinowicz and Williams 1994][Barrett and Cheney 2002] [Igarashi et al. 2005]. However, animating the motion by physical modeling and/or geometry reconstruction may require a lot of user intervention. Unlike the previous methods, we propose an image-based approach which is simpler and does not rely on any physical modeling or reconstruction, as the motion snapshots are already embedded in the picture. The same framework allows us to animate a wide variety of species, from birds to mammals.

The framework of our system (Figure 2) starts by extracting the individuals, or equivalently the motion snapshots, in a single picture. These snapshots form the "key frames" of the motion cycle. However, they are disordered initially. Our key contribution is an optimization that determines the optimal ordering of snapshots in order to reconstruct the motion cycle (Section 3). To achieve this, we first construct a snapshot graph based on the snapshots extracted. Then, an objective function, which maximizes the overall shape distinction of all snapshots and minimizes the shape difference between adjacent snapshots, is used to determine the *motion path* in the graph (corresponds to the motion cycle). To alleviate the pose, morphology, and appearance variation of snapshots along the motion path, we further propose to perform consistency refinement (Section 4). Finally, a smooth motion sequence can be synthesized by morphing among the ordered snapshots (Figure 1). However, certain information, such as motion trajectory and speed, is unavailable from the still (Section 5). Hence, we require the user to draw the motion trajectory for each individual in the picture. The timing of the motion is also provided via a user interface in order to produce realistic motion.

2 Related Work

Animating from a single picture or a sparse set of pictures remains an interesting and challenging problem in computer graphics. Since a still picture lacks the motion information, researchers attempted to introduce prior knowledge to achieve the physical realism. Shinya et al. [1999] employed physically based techniques and image morphing to create 2D animation of plants in an input image. Chuang et al. [2005] simulated natural phenomena relying on physical models. They mainly handled the images containing passive elements with the motion driven by wind, like water, trees, and boats.

ACM Reference Format
Xu, X., Wan, L., Liu, X., Wong, T., Wang, L., Leung, C. 2008. Animating Animal Motion from Still. *ACM Trans. Graph. 27*, 5, Article 117 (December 2008), 8 pages. DOI = 10.1145/1409060.1409070 http://doi.acm.org/10.1145/1409060.1409070.

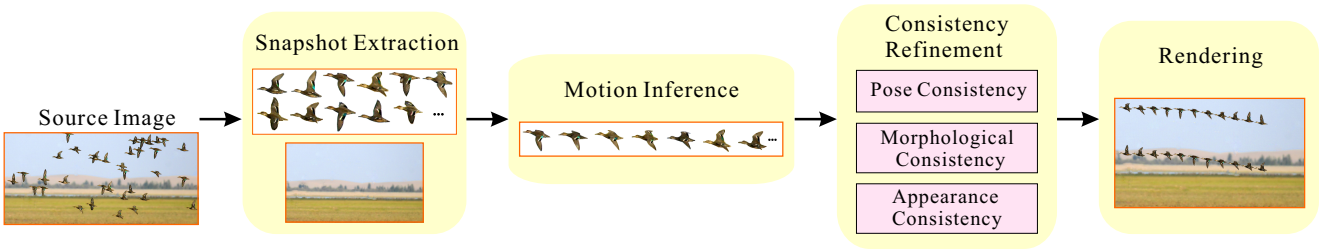


Figure 2: An overview of our system.

Researchers also carried out studies on navigating into the 2D images via scene reconstruction. Horry et al. [1997] let users explore the image in 3D by first obtaining a mesh-based scene model. Their system requires the interactive selection of camera position and vanishing points. Criminisi et al. [2000] computed 3D affine measurements from a single perspective view of a scene. They mainly handled scenes containing planes and parallel lines. Oh et al. [2001] represented a scene as a layered collection of depth images, and developed the editing operations for modifying the shape, color and illumination of the objects.

Other than the physical realism, techniques have been developed to efficiently manipulate the image objects via user control. Litwinowicz and Williams [1994] applied image deformation to create 2D animation based on the key-frame skeletons provided by the user. Ngo et al. [2000] embedded free-form constraints into a graphical model so that users can easily manipulate the rendered image. Barrett and Cheney [2002] triangulated a manually selected object, and then allowed the user to perform interactive editing, such as scaling, stretching, bending, and deleting. Igarashi et al. [2005] presented an interactive system that deforms a 2D shape by moving several vertices as constrained handles. Their work was further improved by Schaefer et al. [2006] to create fast deformation.

Our approach does not rely on physical modeling or geometry reconstruction to simulate complex animal motions. Instead, we make use of the motion snapshots embedded in the image, and order them to synthesize a realistic motion sequence. For motion inference, our work is somewhat related to video texture [Schödl et al. 2000][Soatto et al. 2001][Wang and Zhu 2003], which models the stochastic motion from a video sequence, and the work of [Lin et al. 2007] which synthesizes the partial temporal order of the dynamic motion from a sparse set of images. While Lin et al.'s focus is on natural phenomena, such as water, wind and fire, we emphasize living creatures that exhibit essentially highly regular and periodic motions. In addition, Schindler et al. [2007] inferred the temporal ordering of a collection of images by estimating the 3D structures.

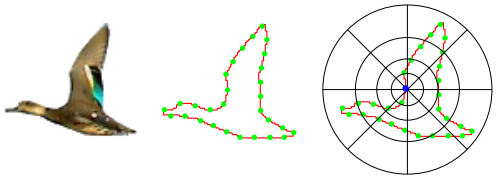
3 Motion Inference

Unlike some natural phenomena such as water and wind, which are stochastically stationary [Chetverikov and Fazekas 2006], animal motion is highly regular and repetitive. Assuming that the input picture contains sufficient snapshots (each corresponds to one individual), we recover the motion cycle by inferring the order of these snapshots with respect to time. To achieve this, we first interactively extract the snapshots using GrabCut [Rother et al. 2004], which provides alpha-matte for each snapshot as well. Occluded individuals are simply discarded. Then, we construct a snapshot graph using a shape similarity metric. Inferring the motion cycle is further formulated as determining the optimal path in the graph, based on an objective function.

3.1 Building Snapshot Graph

The snapshot graph is a complete graph, in which each node represents a snapshot, and the weight of the edge connecting two nodes is the similarity measured between the two corresponding snapshots. An example is shown in Figure 7(a).

The similarity between two snapshots are measured using shape features. The shape of snapshot is represented by a set of discrete points uniformly sampled on the shape contour (Figure 3(b)). We call these points the *contour points*. Then we employ the shape context [Belongie et al. 2002][Mori et al. 2005], a commonly used shape descriptor, to quantify the shape features. This descriptor is invariant to translation, scaling, rotation, and even robust under small geometrical distortion, occlusion and outliers. For each contour point, shape context describes the distribution of the relative positions of all the other points in a spatial histogram. As shown in Figure 3(c), we construct bins that are uniformly distributed in log-polar space, and the number of contour points falling into each bin is one corresponding component in the resulting shape feature vector. The feature vectors of all coutour points are then combined together to describe the shape.



(a) (b) (c)

Figure 3: Shape feature extraction. (a) a snapshot image, (b) the sampled contour points, and (c) the shape context using the log-polar histogram to compute the distribution of the relative positions of all the other contour points to the reference point.

The shape similarity between two snapshots S_k and S_l is further measured by considering the distances from S_k to S_l and from S_l to S_k, respectively [Belongie et al. 2002]. This symmetric measurement is more stable. The formulation is given by

$$D(S_k, S_l) = \frac{1}{M_k} \sum_{i \in M_k} ||\mathbf{f}_i - \mathbf{h}_{j^*}|| + \frac{1}{M_l} \sum_{j \in M_l} ||\mathbf{f}_{i^*} - \mathbf{h}_j||, \quad (1)$$

where M_k (M_l) is the total number of contour points of S_k (S_l); $\mathbf{f}_i$ ($\mathbf{h}_j$) is the shape context feature vector for the i-th (j-th) contour point on S_k (S_l). The j^*-th contour point on S_l corresponds to the i-th contour point on S_k, where $j^* = \text{argmin}_j ||\mathbf{f}_i - \mathbf{h}_j||$. Similarly, $i^* = \text{argmin}_i ||\mathbf{f}_i - \mathbf{h}_j||$. This metric is simple but accurate. Figure 4 shows the query result on snapshot shape using this metric. Obviously, small topological shape changes can be successfully handled.

The above shape similarity metric, however, may find semantically incorrect corresponding points due to the self-occlusion of limbs for some animals as viewed from certain viewpoints. For instance, the elephant snapshots in Figures 5(a) & (c) have very similar silhouettes, but in fact the left and right legs are interchanged. Hence, the contour point i in Figure 5(b) may be incorrectly associated with the contour point j_1^* in Figure 5(d) rather than the semantically correct contour point j_2^*. This problem could be potentially solved by using the technique of human body tracking and reconstruction from single camera [Sminchisescu 2006] [Agarwal and Triggs 2005][Agarwal 2006]. However, most existing techniques adopt learning-based strategies that require a large training set, and

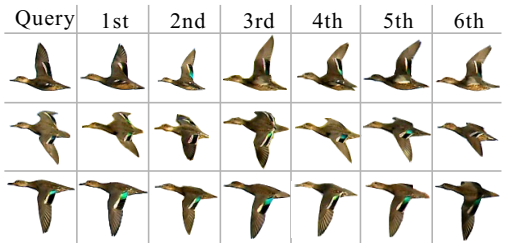


Figure 4: Shortlists of snapshot query. The first column shows the query snapshots, and the rest columns show the closest snapshots with decreasing shape similarity from left to right.

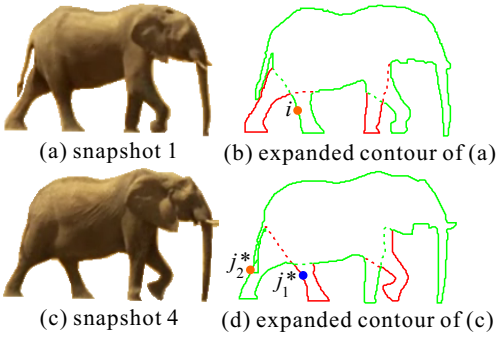


Figure 5: Handling self-occlusion. (a) snapshot 1 and (c) snapshot 4 have very similar shape contours. The contour point i in snapshot 1 may be incorrectly associated with the point j_1^* in snapshot 4. To handle this problem, we manually draw auxiliary contours (in dashed lines) to eliminate the ambiguity, and color-code all the contours to indicate which two legs are on the same side. Finally the corresponding point j_2^* can be located correctly.

the 3D poses they used are even from motion capture. Thus, their works are not applicable in our case. In order to tackle this problem, we introduce auxiliary contours to eliminate the ambiguity on shape contour. Auxiliary contours are assigned interactively by user in the occluded regions, for example, the dashed lines on the boundaries of the elephant legs (Figures 5(b) & (d)). We also color-code the contours to provide the correspondence. Here, we use red and green colors to label legs on different sides. When computing shape distance, we match the points only on the contours with the same color. In this way, we can avoid the shape ambiguity. Note that such auxiliary contours are not always needed. In all our examples, only the elephant example requires this treatment.

3.2 Reconstruction of Motion Cycle

The question now is on how to reconstruct the motion cycle with the snapshot graph. Recall the basic assumption that the input picture contains *sufficient number* of snapshots to form the motion cycle. Intuitively, we want the reconstructed motion cycle to cover *distinct motion states* and represent *smooth movement*. There is no need to make use of all snapshots to form the motion cycle, as some snapshots are too similar to be useful and some are outliers. Instead, we find an optimal path in the snapshot graph, which we call the *motion path*, that corresponds to the motion cycle. To approximate smooth and natural motion, the adjacent snapshots on the motion path should be close in shape. To cover the important motion states, the snapshots on the motion path should be as distinct as possible. Ideally, the snapshots are uniformly sampled from the motion cycle. These requirements lead to our objective for optimizing three factors: local similarity, global distinction, and sampling uniformity.

If we denote the total number of snapshots as N, the motion path of snapshots as an ordered set $\mathcal{I} = \{I_i\}$, where I_i is a node (or snapshot), and the path length (the number of nodes) as $L \leq N$, then the *local similarity* is defined as the mean of all shape distances between adjacent nodes on the path:

$$C_s = \alpha \cdot \mathrm{E}(\{D(I_i, I_{i+1})\}), \tag{2}$$

where $D(I_i, I_{i+1})$ is evaluated by Equation 1, and $\mathrm{E}(\cdot)$ computes the mean value. The normalization factor α is set to be $\max(D(S_k, S_l))$, the maximum distance between two snapshots in the graph. *Sampling uniformity* is measured by the variance of the shape distances between any two adjacent nodes on the path, and is given by

$$C_u = \mathrm{Var}(\{D(I_i, I_{i+1})\}), \tag{3}$$

where $\mathrm{Var}(\cdot)$ evaluates the variance. Finally, *global distinction* is measured by the mean of the distances between any two nodes on the path, which is formulated as

$$C_d = \alpha \cdot \frac{\sum_i \sum_{j, j \neq i} D(I_i, I_j)}{L \cdot (L-1)}. \tag{4}$$

Based on the above three terms, the motion path $\mathcal{I}$ is found by minimizing the following energy function:

$$C(\mathcal{I}) = C_s + \lambda_1 C_u + \lambda_2 (1 - C_d). \tag{5}$$

The coefficients are empirically set as $\lambda_1 = 2.5$ and $\lambda_2 = 0.5$. We solve this discrete optimization problem using simulated annealing on the graph, and the path length L is automatically determined during the optimization process (refer to Appendix A). Note that although we employ the simulated annealing here, other discrete optimization techniques, such as some ML techniques, may also be applicable.

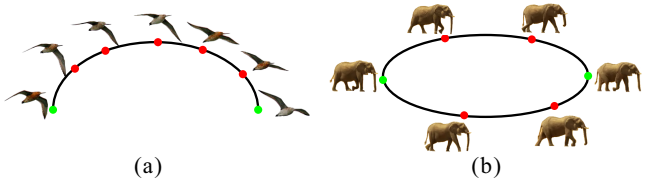


Figure 6: Two types of motions: (a) half-cycle motion and (b) full-cycle motion. In half-cycle motion, the motions of the two half cycles are mirrored, while in full-cycle motion, all the states are completely distinct.

Conceptually, we can classify animal motion into two types: full-cycle motion and half-cycle motion. Full-cycle motion contains completely distinct states in the cycle. The walking of mammals usually belongs to this type. For example, in elephant motion (Figure 6(b)), one pair of diagonal legs moves first in the upper half cycle (from the left to the right green nodes), while another pair of legs moves in the lower half cycle. In contrast, if the motions of two half cycles are mirrored to each other, we call this type of motion the half-cycle motion. One typical example is the flapping of birds (Figure 6(a)), where the motion of up-swing half cycle is mirrored to the motion of down-swing half cycle. So it is obvious that a full-cycle motion corresponds to a closed path in the graph while the half-cycle motion corresponds to an open path.

To determine the motion path, we first automatically determine the two extremal nodes (the green nodes in Figure 6) by finding two nodes with maximal shape distance. For half-cycle motion, the extremal nodes are used as the start and end nodes of the motion path in the snapshot graph. For full-cycle motion, we further constrain that the motion path should pass through the two extremal nodes. Figure 7(a) shows the motion path (the solid black line segments) of seabirds in Figure 11(a). Note that some outlier snapshots, which are substantially different from others, may exist and be selected inappropriately as the extremal nodes. To remove the outliers, we assume that the outlier snapshots should not dominate the set of snapshots, and that the distances between snapshots conform to a Gaussian distribution, $N(\mu, \sigma^2)$. When the minimal distance from one

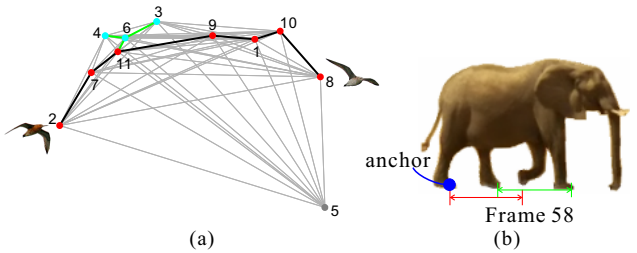


Figure 7: (a) Given the snapshot graph, we search the motion path (the solid black line segments). Node 5 is detected as an outlier. We also connect the rest of the nodes (in blue) to the motion path for later animation by constructing a spanning tree. (b) Anchoring the walking elephant. For frame 58, two diagonal feet are linked with red and green lines. The anchor point is indicated by the blue dot.

snapshot to the other snapshots exceeds the threshold $\delta = \mu + c \cdot \sigma$, this snapshot is removed as an outlier (e.g., the dark gray node in Figure 7(a)). Empirically, we set $c = 2.0$.

It seems that the reconstruction of motion cycle can be solved by taking the shortest path between two extremal nodes. However, finding shortest path only guarantees the smoothness of recovered motion, but will not guarantee sufficient distinction among different snapshots. Hence, some important snapshots may be missed during the motion path reconstruction. Another potential solution is the dimension reduction technique in which the multi-dimensional shape feature vectors are projected onto 1D space for determining the cycle. However, even small errors in the reduction may yield significant errors in the ordering of snapshots. We have tested both the shortest path and dimension reduction approaches, but both give unsatisfactory reconstruction results.

4 Consistency Refinement

As the snapshots are actually different individuals, morphing directly among them may yield unnatural and inconsistent animation results. To allow smooth transition among ordered snapshots, we refine three kinds of consistencies: pose consistency, morphology consistency, and appearance consistency. *For the animated illustration of the consistency issue, readers are kindly referred to the supplementary video.*

4.1 Pose Consistency

Since the individuals are captured in different poses relative to the camera, we first "normalize" them so that they share the same scale and orientation. Although the shape descriptor in previous step is scale, rotation, and translation invariant, the snapshots are not yet normalized. Without loss of generality, the first snapshot in the ordered sequence is used as the reference for normalization. In our image-based system, the normalization can be done by estimating 2D affine transformations between snapshot poses.

To achieve this, we first establish the correspondence between the contour points on every two adjacent snapshots by shape matching [Belongie et al. 2002]. Then, for each contour point q_1 on the first snapshot, its corresponding point q_k on the k-th snapshot can be located according to the correspondence determined from shape matching. The affine transformation $T_{1,k}$ between the first and the k-th snapshot can be solved from the corresponding points using least square.

Due to the non-rigid shape deformation of the captured snapshots, we cannot simply use all the contour points for estimating the affine transformation. Instead, we only utilize the contour points that are relatively *stable* (less deformed during the motion) for estimation, e.g., the contour points on the head and the tail of bird, or the contour points on the head and the back of elephant. To identify those points, users can interactively circle on the first snapshot and obtain a set of stable contour points Q_1. Then, our system automatically identifies the corresponding stable contour points Q_k on the k-th snapshot via the correspondence from shape matching. The affine transformation $T_{1,k}$ is then solved from these corresponding stable contour points. After the estimation, snapshots are "normalized" in terms of pose by applying the affine transformations (Figure 8(b)).

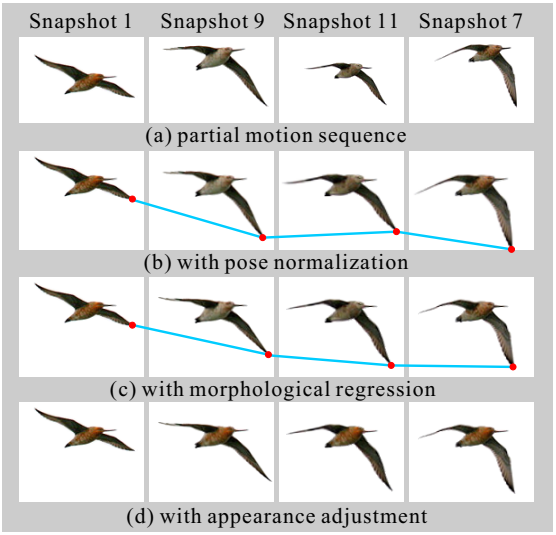


Figure 8: Consistency refinement. (a) shows the first 4 snapshots of the ordered sequence. (b) The snapshot poses are normalized in scale and orientation. (c) Morphology consistency in the sequence is further refined. Note that the perturbation (drawn in the blue line) becomes more smooth after regression. (d) Color variation is also suppressed in the appearance consistency refinement.

4.2 Morphology Consistency

Even after pose normalization, the snapshots may still suffer from local shape inconsistency due to the morphological difference of individuals. Morphing among them may lead to unnatural temporal perturbation as depicted by the bumpy blue line in Figure 8(b). To solve this problem, we apply a regression process on the trajectory of the corresponding contour points to smoothen out the perturbation. Here, we assume that such trajectory conform to a B-spline curve, which is parameterized by

$$P(t) = \sum_{j=1}^{m} \tilde{P}_j b_{j,s}(t), \quad t \in [0,1], \tag{6}$$

where m is the number of control points $\tilde{P}_j$, and $b_{j,s}(t)$ is the j-th B-spline basis function with order s. Specifically, we use one piece of uniform cubic B-spline, and set $m = \min(15, n/2)$, where n is the number of snapshots in the ordered sequence. The B-spline regression can then be formulated as a least square minimization problem:

$$\min \sum_{i=1}^{n} \|P(t_i) - \hat{P}_i\|^2, \tag{7}$$

where $\hat{P}_i$ is the contour point on the i-th snapshot in the ordered sequence, and $t_i \in [0,1]$ is the B-spline parameter value set to the i-th snapshot. To ensure a consistent temporal order of regression, the parameter values t_i should be monotonic increasing ($t_i < t_{i+1}$) with respect to the snapshot order in the sequence. In addition, the differences in value are proportional to the corresponding shape distances, i.e., $t_{i+1} - t_i \propto D(I_i, I_{i+1})$.

The minimization returns optimal control points $\tilde{P}_j$ of the B-spline curve. However, when there are outlier contour points on the trajectory, the B-spline approximation may be erroneous. Hence, iterative re-weighted least square (IRLS) is used for estimation, which

is robust to outliers and does not over-suppress the trajectory. With the estimated B-spline curve, we replace $\hat{P}_i$ with $P(t_i)$. This ensures a smooth transition among snapshots with consistent order (depicted as the blue line in Figure 8(c)). The same regression is applied to all the contour points. Equivalently, we can regard $\hat{P}_i$ as a high dimensional vector containing elements of 2D coordinates of all the contour points during the regression. Although we may use convolution methods to suppress the perturbation, they are not able to remove outliers while maintaining a reasonable trajectory.

So far, we only smoothen the trajectories of contour points on the shape boundary. To smoothly adjust the interior points within the shapes, we apply thin-plate spline (TPS) mapping on each shape according to the correspondence between $\hat{P}_i$ and $P(t_i)$ (refer to Appendix B). Figure 9 superimposes the snapshots to compare the results with and without such B-spline based regression in (b) and (a), respectively. The original bumpy trajectory is now smoothed after the regression.

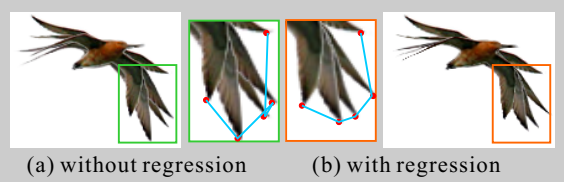


Figure 9: Morphology consistency. (a) shows the superimposed snapshots without regression. The trajectory of contour points (connected with blue line) is bumpy. (b) shows the resulting snapshots with the shape regression. Note that the trajectory is now smoothed.

4.3 Appearance Consistency

The snapshots are now consistent in terms of pose and morphology, but their appearance (color and texture) may still be inconsistent from each other due to the difference of individuality. In order to solve this, we select the first snapshot as the reference and "transfer" the appearance of the reference to all the other snapshots, ignoring the illumination.

Transferring the appearance of the entire body of a snapshot to the others may be erroneous because of the self-occlusion caused by deformation during motion. In order to obtain visually consistent appearance, we first manually mark on the reference snapshot the region that is inconsistent in appearance, denoted as R_1 (enclosed by the green curve in Figure 10). From observation, the region usually occurs at places with small deformation, and hence less chance of being occluded. Therefore, we can warp the region R_1 in the reference snapshot to the region R_k in the k-th snapshot using TPS mapping. The mapping is determined by the corresponding contour points of the two snapshots. Finally, the region R_k (enclosed by the orange curve) is replaced with the warped R_1. The seam on the boundary of replaced region is alleviated by feathering technique. Note that the marked region R_1 is allowed to contain multiple disjoint sub-regions.

5 Results and Discussions

Once motion inference and consistency refinement are done, we can then animate the motion. Here we first demonstrate the applicability of our method on species with substantially different motions. Next, we illustrate how to animate the still images showing different groups of animals. *Readers are referred to the supplementary video for a better animated presentation.*

Rendering To render the animation starting from a still picture, motion trajectory should be first specified by the user for each individual. Then, the animation of each individual is synthesized by applying TPS morphing among all the consistency-refined snapshots.

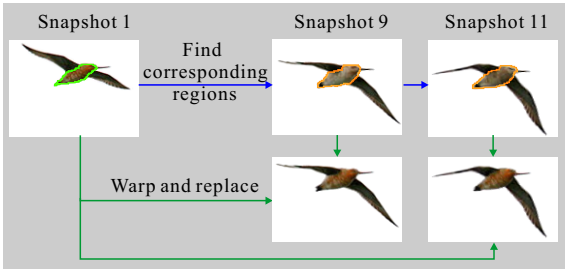


Figure 10: Appearance consistency. The appearance-inconsistent region (enclosed by the green curve) is first manually selected in the reference snapshot. Then the selected region is warped to replace the corresponding region in other snapshots (enclosed by the orange curve).

The TPS morphing is implemented on GPU to reach interactive rate. The final animated sequence is created by blending the animation of each individual with a background image or a video. Motion blur effect is also added in the final animation using the technique of [Brostow and Essa 2001]. In addition, users can further control the animation, such as changing the speed of the motion, adjusting the scale and the orientation of each individual during the motion, in order to refine the animation.

However, certain motion may need additional treatment, for example, the motion of walking, like that of an elephant or a dog. There should be one more constraint applied. Their feet touching on the ground must be anchored on the ground during the motion. Otherwise, odd drifting artifact may occur. Hence, we need to determine the anchor feet for each rendered frame. In the example of elephant in Figure 7(b), we first compute the distance between two diagonal feet, as denoted by D_r (the red line) and D_g (the green line), respectively. Then, for every two adjacent frames k and $k+1$, we compute the changes of D_r and D_g. The diagonal feet with less change are identified in frame k. Then the tip of the back foot (blue dot) of the identified diagonal feet is chosen to anchor the rendered animal in the frame.

Motions of Different Species To verify the effectiveness of our motion inference on different species, we show the inferred motion of birds (Figure 1(a) and Figure 11(d)), elephants (Figure 1(b)), loaches (Figure 11(e)), and tortoises (Figure 11(f)). The motions of birds and loaches are half-cycle while the motions of elephants and tortoises are full-cycle.

Given the input picture of a flock of birds in Figure 1(a), we extract 30 snapshots in total, in which 13 snapshots are used in the reconstructed half-cycle motion. A full motion cycle after morphing among the ordered snapshots is shown on the left side of Figure 1(a). Another example of a flock of birds is shown in Figure 11(a), where 7 out of 11 snapshots are selected to form the half-cycle motion. The swimming motion of loaches (Figure 11(b)) is represented by 9 ordered snapshots. The reconstructed motions for these two examples are demonstrated in Figures 11 (d) and (e), respectively. The input picture of elephant example contains 7 valid snapshots, and 6 out of them are selected to form the full cycle of the motion (Figure 1(b)).

Our method also allows the user to take multiple still pictures of the same species as input pictures, so as to provide sufficient snapshots. The major criterion is that individuals from different pictures are taken with similar viewing perspectives. Figure 11(c) shows the three input pictures for inferring the motion of tortoises.

Timing Statistics The proposed system was implemented on a PC equipped with Xeon(TM) CPU 3.73GHz, 3GB system memory, and nVidia Geforce 8800 GTX GPU with 768 MB video memory.

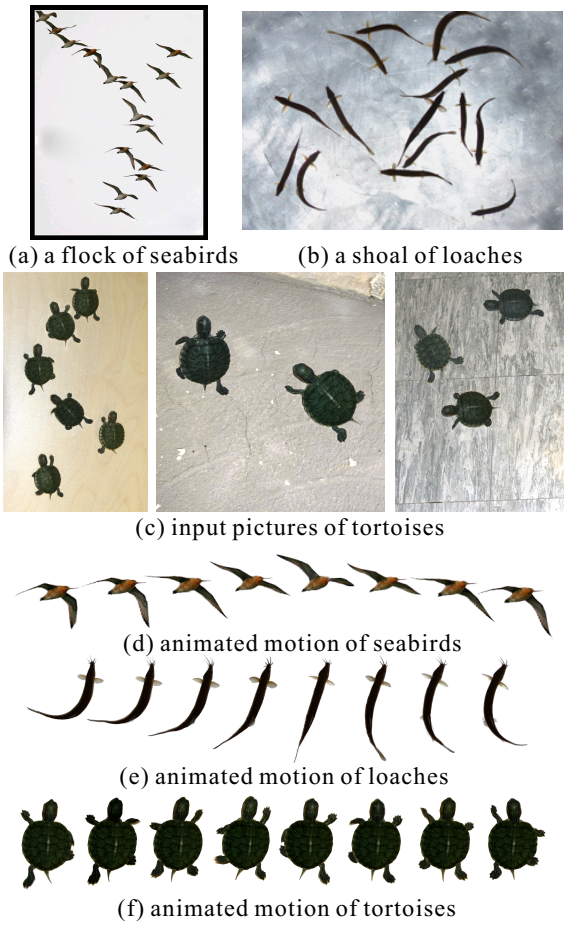


Figure 11: The animated motions from a single or multiple pictures.

Table 1 summarizes the timing statistics of all the examples used in this paper. The processing times for different parts of our system, including the approximate time for user interaction, are listed in different columns in the table. From the table, we can see that the whole processing time can be split into two parts: the time with user interaction involved, and the time for fully automatic processing.

The total time for user interaction (column 7 in Table 1) is not more than 6 minutes for all our examples. Note that the most time-consuming user interaction is the semi-automatic extraction of snapshots. The time for circling the stable contour points and marking the appearance inconsistent regions is usually small as they are only required on the first snapshot. Auxiliary contours are only needed for the example of elephants.

On the other hand, the total time for automatic processing is much smaller. The motion inference is efficient and requires only 7 to 35 seconds, even though we adopted simulated annealing. The column "rendering" only shows the processing time for synthesizing one single individual in a single frame, and the total time for rendering the whole animation sequence depends on both the number of frames and the number of individuals used in the final animation. The fast rendering is achieved via a GPU implementation of TPS mapping.

Animating from Still Finally, we show an application of animating a group of animals starting from their initial poses and positions in the input still picture. To achieve this, we first complete the background after snapshot extraction, using example-based inpainting algorithm [Criminisi et al. 2003]. Then, we create a motion sequence for each individual based on the snapshot graph. Recall that not all the snapshots are selected in the motion path. Thus, the snapshots that are not in the motion path cannot be animated directly. In order to create a motion sequence starting from these snapshots, we need to construct a spanning tree (Figure 7(a)) in the snapshot graph.

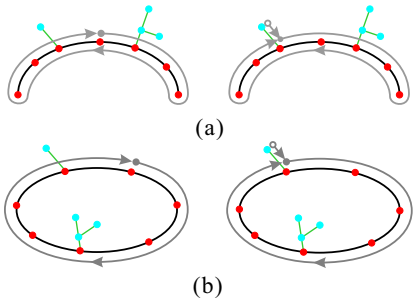


Figure 12: Animating from initial poses of individuals. (a) the spanning tree of half-cycle motion and (b) the spanning tree of full-cycle motion. The left column shows when the node of individual is on the motion path (solid gray node). The right column is the case when node of individuals are not on the motion path (hollow gray node).

The spanning tree is constructed beginning with the motion path. Then, the entire spanning tree is constructed iteratively by selecting the node in the remaining node set (blue dots) with the minimal distance to its nearest node in the current spanning tree. As shown in Figure 7(a), the red nodes are on the motion path and the blue nodes are selected iteratively to form the spanning tree.

Based on the spanning tree, animation can be created for each individual. For half-cycle motion, if the individual is on the motion path (the solid black line segments in Figure 12), we traverse the path back and forth between the two extremal nodes to generate the motion sequence. If the individual is not on the motion path, we first find the shortest path from the initial node (the individual) to the motion path in the spanning tree. Then we traverse along the shortest path to the motion path. Finally we keep traversing back and forth again on the motion path to generate the motion sequence. The two cases are illustrated in Figure 12(a). Full-cycle motion can be achieved similarly, except that the motion path is already a loop (Figure 12(b)).

Once the motion sequence is obtained from the snapshot graph, we perform consistency refinement and TPS morphing to generate the final animation. Figure 13 shows a few frames of animating animals from the input still pictures.

Limitations In our system, all the motion snapshots must be captured in similar viewing perspectives, and there must be sufficient snapshots to cover the important states of the motion cycle. Due to the loss of temporal information and the lack of prior knowledge, the system relies on users to tell whether the animal motion is in half-cycle or full-cycle. In addition, since we use the image-based approach, the current appearance consistency refinement may not be physically correct. Nevertheless, even with these limitations, visually plausible results are obtained.

6 Conclusion

In this paper, we animate a still picture of an animal group. Based on the observation that the motion snapshots of the same species can be captured in a still picture, we propose to infer the motion cycle of the animals from the disordered snapshots. The inference is formulated as finding an optimal path in the snapshot graph. To avoid inconsistencies during the animation, we ensure the pose, morphology, and appearance consistencies. Although we do not use a physical model, convincing animal motions are obtained as demonstrated by the animation results of species with substantially

Figure 13: Animation from the still pictures in Figure 1.

	Total snapshots /snapshots in cycle /snapshot resolution	User Interaction					Automatic		
		Snapshots extraction (A)	Drawing auxiliary contours (B)	Circling stable contour points (C)	Marking appearance inconsistent regions (D)	Total time of user intervention (A)+(B)+(C)+(D)	Motion inference	Consistency refinement	Rendering (single morphed snapshot)
Bird1 (Fig. 1(a))	30 / 13 / 174×174	~5mins		~30s	~30s	~6mins	35s	17s	0.024s
Bird2 (Fig. 11(a))	11 / 7 / 175×175	~2mins		~30s	~30s	~3mins	7s	5s	0.016s
Loach (Fig. 11(b))	15 / 9 / 500×500	~3mins		~30s	~30s	~4mins	13s	158s	0.76s
Tortoise (Fig. 11(c))	10 / 9 / 500×500	~2mins		~30s	~30s	~3mins	13s	108s	0.78s
Elephant (Fig. 1(b))	7 / 6 / 290×290	~2mins	~2mins	~30s	~30s	~5mins	13s	24s	0.31s

Table 1: Timing statistics for all the examples used in this paper. In the second column, the "total snapshots" refers to the number of snapshots extracted from the input picture. The "snapshots in cycle" refers to the number of snapshots selected for reconstructing the motion cycle.

different motion behaviors.

An automatic method for handling self-occlusion problem is worth further investigation. The appearance transfer from one individual to another should also be investigated in the future to improve its robustness. In addition, inference on non-cyclic motion may also be worthwhile for further study.

Appendix A: Motion Cycle Optimization

Algorithm 1 presents the pseudo-code of motion-path optimization. The motion path contains at least two extremal nodes in snapshot graph. During each iteration, we randomly generate a new path in the graph, which is associated with a valid path length L. Based on the energy function (Equation 5), the new path is accepted or rejected according to an annealing strategy [Pepper et al. 2002]. To guarantee a stable convergence, a certain number of iterations (K) is set at temperature T before the next annealing process. In our implementation, we set $T_0 = 50,000$, $K = 500$, $AnnealFactor = 0.992$, and $limit = 0.01$. It takes less than 35 seconds to search the motion path even for the bird example with 30 snapshots (Figure 1(a)).

Appendix B: TPS Mapping

Thin-plate spline (TPS) mapping [Bookstein 1989] is the modeling of deformation of biological shape change. Due to the energy minimization property, TPS introduces less distortion and hence offers high visual quality in image warping. When TPS is applied to image warping, n pairs of corresponding control points between the source image ($\{(x_i, y_i)\}$) and the warped image $\{(x'_i, y'_i)\}$ should be first specified. Then, the TPS mapping is defined as follows,

$$x' = f_x(x,y) = a_1 + a_x x + a_y y + \sum_{i=1}^{n} \alpha_i \phi(||(x_i, y_i) - (x, y)||), \tag{8}$$

$$y' = f_y(x,y) = b_1 + b_x x + b_y y + \sum_{i=1}^{n} \beta_i \phi(||(x_i, y_i) - (x, y)||), \tag{9}$$

where (x, y) is the point in the source image and (x', y') is the corresponding point in the warped image; $\phi(r) = r^2 \log r$ is the radial basis function; $||\cdot||$ is the Euclidean distance operator; a_1, a_x, a_y, α_i, b_1, b_x, b_y, and β_i are the parameters to be determined. To solve the mapping, two sets of extra constraints are enforced: $\sum \alpha_i = \sum \alpha_i x_i = \sum \alpha_i y_i = 0$, and $\sum \beta_i = \sum \beta_i x_i = \sum \beta_i y_i = 0$. Then, we are able to obtain two linear systems. Once the unknowns are determined, we can warp other points in the source image with the mappings $f_x(x, y)$ and $f_y(x, y)$.

An example of image warping is given in Figure 14. Figure 14(a) shows the sampled contour points (blue crosses) in the source image (Figure 14(b)) and their corresponding points (red dots) in the warped image. Figure 14(c) is the result by warping the source image according to the correspondence in (a). It is obvious that TPS mapping not only maintains the shape contour but also creates smooth and consistent warped content in the interior region. Based on the warping, image morphing between two motion snapshots is

(a) (b) (c)

Figure 14: TPS mapping. (a) shows the sampled contour points (blue crosses) in the source image and their corresponding points (red dots) in the warped image. (b) is the source image. (c) shows the result by TPS warping. Note that not only the outline but also the interior region is naturally warped.

then achieved by blending the result from warping one snapshot towards the intermediate position and that from warping the next snapshot towards the intermediate position.

Algorithm 1: Pseudo-code of motion cycle optimization

```
choose a path length L
generate an initial path I_L with length L
choose a temperature T = T_0 > 0
C_old = C(I_L)
Loop ( T > limit )
    Loop (K times)
        choose a new path length L
        generate a new path I_L
        C_new = C(I_L)
        ΔC = C_new - C_old
        if ΔC ≤ 0, accept new path
        else if exp(−ΔC/T) ≤ rand(0, 1), accept new path
        else reject new path
        T = T × AnnealFactor
```

Acknowledgements

We would like to thank all reviewers for their constructive comments and guidance in shaping this paper. This project is supported by the Research Grants Council of the Hong Kong Special Administrative Region, under RGC Earmarked Grants (Project No. CUHK 416806), and research grants from City University of Hong Kong (Project No. 7002108).

References

AGARWAL, A., AND TRIGGS, B. 2005. Monocular human motion capture with a mixture of regressors. In *Proc. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) 2005 - Workshops*, 72.

AGARWAL, A. 2006. Machine learning for image based motion capture. PhD Thesis, Institut National Polytechnique de Grenoble.

BARRETT, W. A., AND CHENEY, A. S. 2002. Object-based image editing. In *Proc. ACM SIGGRAPH 2002*, 777–784.

BELONGIE, S., MALIK, J., AND PUZICHA, J. 2002. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligience 24*, 4, 509–522.

BOOKSTEIN, F. L. 1989. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transaction on Pattern Analysis and Machine Intelligience 11*, 6, 567–585.

BROSTOW, G. J., AND ESSA, I. 2001. Image-based motion blur for stop motion animation. In *Proc. ACM SIGGRAPH 2001*, 561–566.

CHETVERIKOV, D., AND FAZEKAS, S. 2006. On motion periodicity of dynamic textures. In *British Machine Vision Conference (BMVC)*, vol. I, 167–175.

CHUANG, Y.-Y., GOLDMAN, D. B., ZHENG, K. C., CURLESS, B., SALESIN, D. H., AND SZELISKI, R. 2005. Animating pictures with stochastic motion textures. In *Proc. ACM SIGGRAPH 2005*, 853–860.

CRIMINISI, A., REID, I. D., AND ZISSERMAN, A. 2000. Single view metrology. *International Journal of Computer Vision 40*, 2, 123–148.

CRIMINISI, A., PEREZ, P., AND TOYAMA, K. 2003. Object removal by exemplar-based inpainting. In *Proc. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) 2003*, 721–728.

HORRY, Y., ANJYO, K.-I., AND ARAI, K. 1997. Tour into the picture: using a spidery mesh interface to make animation from a single image. In *Proc. ACM SIGGRAPH 1997*, 225–232.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. As-rigid-as-possible shape manipulation. In *Proc. ACM SIGGRAPH 2005*, 1134–1141.

LIN, Z., WANG, L., WANG, Y., KANG, S. B., AND FANG, T. 2007. High resolution animated scenes from stills. *IEEE Transactions on Visualization and Computer Graphics 13*, 3, 562–568.

LITWINOWICZ, P., AND WILLIAMS, L. 1994. Animating images with drawings. In *Proc. ACM SIGGRAPH 1994*, 409–412.

MORI, G., BELONGIE, S., AND MALIK, J. 2005. Efficient shape matching using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 11, 1832–1837.

NGO, T., CUTRELL, D., DANA, J., DONALD, B., LOEB, L., AND ZHU, S. 2000. Accessible animation and customizable graphics via simplicial configuration modeling. In *Proc. ACM SIGGRAPH 2000*, 403–410.

OH, B. M., CHEN, M., DORSEY, J., AND DURAND, F. 2001. Image-based modeling and photo editing. In *Proc. ACM SIGGRAPH 2001*, 433–442.

PEPPER, J. W., GOLDEN, B. L., AND WASIL, E. A. 2002. Solving the traveling salesman problem with annealing-basedheuristics: a computational study. *IEEE Transactions on Systems, Man and Cybernetics, Part A 32*, 1, 72–77.

ROTHER, C., KOLMOGOROV, V., AND BLAKE, A. 2004. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Transaction on Graphics 23*, 3, 309–314.

SCHAEFER, S., MCPHAIL, T., AND WARREN, J. 2006. Image deformation using moving least squares. In *Proc. ACM SIGGRAPH 2006*, 533–540.

SCHINDLER, G., DELLAERT, F., AND KANG, S. B. 2007. Inferring temporal order of images from 3D structure. In *Proc. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) 2007*.

SCHÖDL, A., SZELISKI, R., SALESIN, D. H., AND ESSA, I. 2000. Video textures. In *Proc. ACM SIGGRAPH 2000*, 489–498.

SHINYA, M., AOKI, M., TSUTSUGUCHI, K., AND KOTANI, N. 1999. Dynamic texture: physically based 2D animation. In *Proc. ACM SIGGRAPH 1999*, 239.

SMINCHISESCU, C. 2006. 3D human motion analysis in monocular video techniques and challenges. In *Proc. the IEEE International Conference on Video and Signal Based Surveillance*, 76.

SOATTO, S., DORETTO, G., AND WU, Y. N. 2001. Dynamic textures. In *International Conference on Computer Vision*, 439–446.

WANG, Y., AND ZHU, S.-C. 2003. Modeling textured motion: Particle, wave and sketch. In *Proc. the 9th IEEE International Conference on Computer Vision(ICCV) 2003*, 213–220.