Motion Imitation with a Handheld Camera

Guofeng Zhang[1], Hanqing Jiang[1], Jin Huang[1], Jiaya Jia[2], Tien-Tsin Wong[2], Kun Zhou[1], and Hujun Bao[1]

[1]State Key Lab of CAD&CG, Zhejiang University
{zhangguofeng, jianghq, hj, kunzhou, bao}@cad.zju.edu.cn

[2]The Chinese University of Hong Kong
{leojia, ttwong}@cse.cuhk.edu.hk

Abstract—In this paper, we present a novel method to extract motion of a dynamic object from a video that is captured by a handheld camera, and apply the motion to a 3D character. Unlike the motion capture techniques, neither special sensors/trackers nor a controllable environment is required. Our system significantly automates motion imitation which is traditionally conducted by professional animators via manual key-framing. Given the input video sequence, we track the dynamic reference object to obtain trajectories of both 2D and 3D tracking points. With them as constraints, we then transfer the motion to the target 3D character by solving an optimization problem to maintain the *motion gradients*. We also provide a user-friendly editing environment for user to fine-tune the motion details. As casual videos can be used for imitation, our system therefore greatly increases the supply source of motion data. Examples of imitating various types of animal motions (that are hard to motion-capture) are shown to demonstrate the effectiveness of our system.

Index Terms—Motion imitation, motion gradient, mesh deformation, depth recovery, motion tracking.

1 Introduction

BESIDES the appearance, the motion of synthetic 3D characters is an important visual cue to increase the rendering realism. Imitating motion of real humans and animals is common in film and game productions. Such realistic motion is traditionally achieved by either motion capture (with specialized equipments) or keyframe-based pose editing (requiring highly skillful animators).

The motion capture technique records the movements of an actor using trackers together with the acquisition device. It tracks the marked points (normally at the joints of an articulated object) over time, usually in a carefully controlled environment. Although methods, such as the one of Vlasic et al. [1], were proposed to alleviate the high configuration requirement, motion capture is still difficult, if not impossible, for wild (e.g. a running lion) and small animals (e.g. a tiny salamander). Animators solve this problem by keyframing the motion by hand. However, if the character undergoes complex motion with many subtle details, manual keyframing could be very time-consuming and labor intensive.

In this paper, we present a novel system to significantly automate this manual editing process by first acquiring the reference motion information of an animal (or a human) from a video sequence, and then applying the acquired motion to a 3D character. The input video can be simply taken by a handheld camera. Neither special hardware nor a controlled environment is required. The top row of Fig. 1 shows a set of example video frames.

The main difficulty of recovering high-quality 3D motion from an ordinary video is the lack of dense depth information to constrain the estimated character pose. This paper presents a novel method to make use of both the limited depth information computed with multi-view geometry and sparse 2D motion tracks estimated from a monocular video to represent character motion. These two groups of data together help define *motion gradients*, which capture the essence of object movements among frames. In the motion transfer step, motion gradient can even compensate moderately divergent shapes of the source and target characters.

Another contribution of our method is to accomplish 3D animation by solving a non-rigid deformation problem with the space-time constraints from the extracted 2D and 3D tracks. It is notable that many motion retargeting techniques [2], [3] are skeleton-based. They only transfer articulated motion, but not the surface deformation as in our case. Such complex non-rigid deformation is vital for high-quality motion imitation, which is however difficult to model via articulated motion transfer.

Our system also provides powerful motion editing ability for user to fine tune the motion-retargeted result. The user can add, remove, and modify control points in both the video motion estimation and the 3D character animation phases. These modifications are combined with automatically refined constraints to produce the desired deformation. As a result, our motion imitation system no longer requires highly skilled animators, or high acquisition cost as in conventional motion capture. Besides, the supply source of motion information is significantly expanded with the wide availability of low-cost digital video cameras and videos. In contrast to inter-frame interpolation typically adopted in the keyframing approaches, our method generates motion in between user-edited frames by solving an optimization problem with regard to the extracted 2D and 3D tracks. Hence the resultant character animation in general is more natural, without the need of defining dense keyframes.

2 Related Work

As our method involves procedures of motion acquisition from video and motion retargeting to 3D objects, we briefly review the related work in these areas.

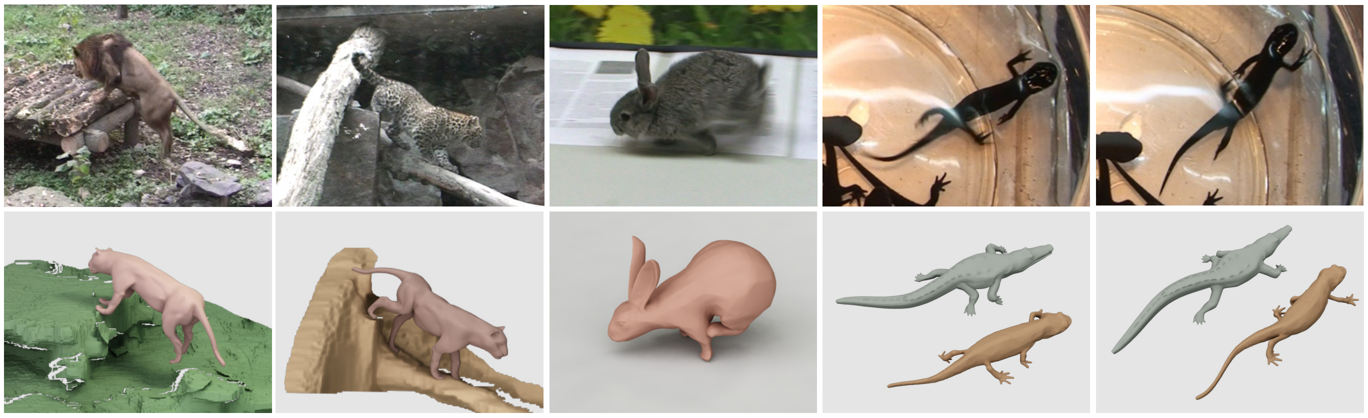


Fig. 1. Snapshots of the input video frames (top) and the animated characters (bottom) from our system.

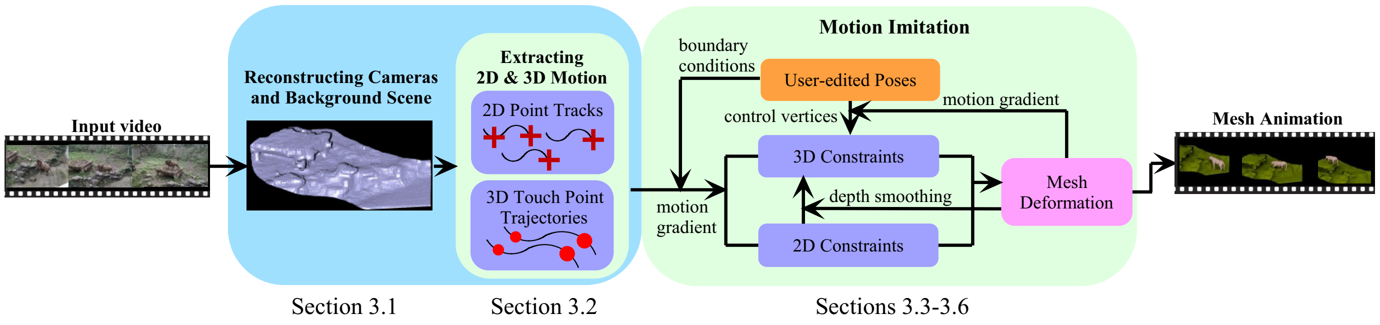


Fig. 2. System overview.

2.1 3D Reconstruction from Images/Videos

Extraction of 3D information from an image sequence of a static scene has been extensively studied in [4], [5], [6]. Traditional multi-view stereo methods [6] aim to automatically recover the dense 3D models from multiple images. In [7], [8], interactive image-based modeling methods were proposed and tailored for the recovery of specific types of static objects, such as trees, vehicles, or urban buildings. All these methods are limited to static scene as dynamic objects do not satisfy the multi-view geometry. By making use of multiple synchronized cameras, methods of [9], [10], [11] can be applied to dynamic 3D models recovery or 3D motion capture.

Non-rigid structure-from-motion (NRSFM) methods [12], [13] can be used to reconstruct non-rigid scenes from a monocular video. They generally assume that the 3D deforming object can be modeled as a linear combination of a series of basis shapes, which is insufficient for constructing high-quality models for complex motion with significant (non-linear) occlusions (see the lion example in this paper). In contrast, our method can handle such occlusion and eliminate visual artifacts by solving a non-rigid deformation problem with the space-time constraints generated from the extracted 2D and 3D motion tracks. It can even moderately tolerate the discrepancy between the reference and target shapes, which is intractable for existing NRSFM methods.

2.2 Vision-based Motion Capture

Typical motion capture consists of hardware sensors/trackers and a camera to collect the motion data. It has been widely adopted in film industry for capturing realistic human motion. However, the specialized hardware is usually expensive. To reduce the hardware requirement, video-based motion capture solutions [14], [15], [16] were proposed, based on computer vision techniques, to create motion data using the limited information provided by a video. However, these techniques are typically limited to tracking simple human motion (e.g. walking), where the acquired motion information is rough. As studied by Gleicher and Ferrier [14], even using many strong priors (e.g., using learned motion models to restrict or predict likely poses), the results from the state-of-the-art vision methods cannot overtake the ones obtained from optical tracking systems, and are difficult to meet the production quality. Existing methods mainly focus on tracking the motion of articulated skeletons.

Sand et al. [17] proposed a full-body motion capture system, which can acquire the deformable human geometry from the silhouettes captured by one or more cameras. The motion of the skeleton is required to be determined first. In comparison, we propose tracking feature points in a monocular video to constrain the surface deformation, without skeleton. So our method can be easily applied to a wide range of characters, including human and animals.

2.3 Mesh Deformation for Retargeting

In 3D deformation, the representative work includes skeleton subspace deformation (SSD) [18], free form deformation (FFD) [19], multi-resolution technique [20], [21], and gradient domain methods [22], [23], [24], [25], [26], [27], [28]. For animation retargeting, Sumner and Popovi [29] transferred the deformation of a source triangular mesh onto a target. Zhou et al. [24] demonstrated the application of retargeting the cartoon animation to 3D models by applying the graph Laplacian mesh deformation technique. They used 3D curve constraints, where the influence weight of the control curves should be carefully tuned by the user and the depth information should be manually assigned (or set almost constant). Therefore, it is difficult to retarget complex 3D motions, such as the ones shown in this paper. In addition, they did not address the problem caused by the shape difference between the video objects and the target 3D models. With the skeleton and key poses of a model as input, Bregler et al. [30] proposed applying the affine deformation from 2D cartoon animations to 2D drawings and 3D shapes. Favreau et al. [31] proposed animating 3D animal models from existing live video sequences. However, this method also requires the skeleton and key poses of the model as input, and assumes the animation has a cyclic motion. In summary, most of the above methods ignore the potentially useful depth information available in the video, probably due to the difficulty of accurate depth recovery.

3 Motion Imitation

The system overview is shown in Fig. 2. The key idea is to infer the trajectories of both 2D and 3D feature points from the input video and apply them to comprehensive motion imitation. The 2D motion track refers to planar pixel shift in the video frames, and hence is projective. The 3D motion track, with depth information, is obtained using multi-view geometry and structure-from-motion (SFM) over multiple frames. With the extracted motion data, we transfer them to a 3D character by maintaining the *motion gradient* with a set of 2D and 3D constraints.

Our system consists of three main phases. Given an input video, if the camera moves, the depths of the static scene are recovered by multi-view geometry. This step is mostly automatic where a small amount of user intervention is on roughly masking out the foreground object in a sparse set of keyframes. Then a complete 3D background scene is produced by pixel reprojection. If no depth information can be recovered from the video, our system still works, but with the trade-off of making some depth assumption or increasing user interaction for model pose adjustment. In the second phase, user selects 2D and 3D key points. The corresponding motion tracks are then extracted from the input video. Finally, in the last phase, based on these 2D and 3D motion tracks, the motion is transferred to the target 3D character with progressive refinement.

3.1 Camera Pose and Background Depth Estimation

Given an input video sequence containing n frames, we estimate camera pose $\mathbf{C}_t$ and recover the corresponding depth map for frame t. In our system, the SFM method of Zhang et al. [32] is used to recover the parameter set $\mathbf{C} = \{\mathbf{K}_t, \mathbf{R}_t, \mathbf{T}_t\}$, where $\mathbf{K}_t$ is the intrinsic matrix, $\mathbf{R}_t$ is the rotation matrix, and $\mathbf{T}_t$ is the translation vector. With these estimated parameters, we then roughly mask out the foreground dynamic object (the reference object) using the lasso tool. The multi-view stereo method of Zhang et al. [33] is used to recover the view-dependent, dense depth maps of the static background.

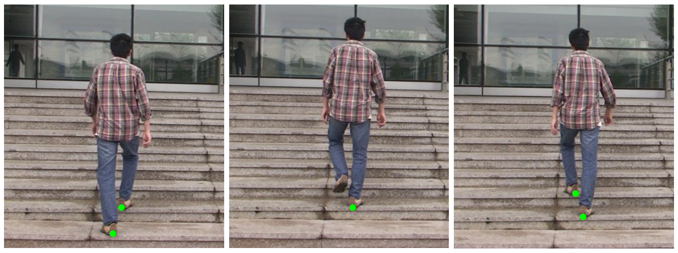


Fig. 3. The soles of the feet (green points) touches the stair from time to time.

Missing pixels, after removal of foreground object, are inferred from the temporally neighboring frames by color and depth projection based on the estimated camera poses. For acceleration, we estimate depth maps only for a sparse set of frames. They are completed and triangulated to construct a 3D background model. The depth information is used in following steps to help generate 3D motion constraints.

3.2 Extraction of 2D and 3D Motion Tracks

We extract sparse feature tracks from the dynamic video object, and use them to animate a 3D character. The user first selects key points in the first frame on the character e.g. the leg and head points shown in Fig. 13. Then an interactive point tracking method described in the Appendix is employed to track the movements of these points in the successive frames and form *motion tracks*. Each of them includes a set of points $\mathcal{X}_t^i$ where t and i index the images and tracks respectively.

However, the obtained 2D motion tracks are not adequate to constrain the 3D motion imitation, due to the lack of necessary depth information. To address this problem, we propose tracking and determining the 3D coordinates of a special type of surface points, called *motion anchors*. A motion anchor refers to a surface point that touches the static background from time to time. A typical example is the sole of the foot, as illustrated in Fig. 3. When a motion anchor contacts the ground, it should have the same depth with the ground point and its 3D coordinate at this moment can be determined. With this observation, we allow user to freely define a frame set Ω and manually label anchor points in Ω. Then the same tracking procedure described above is employed to track the anchors in all frames.

We denote the 3D coordinate of a touch point on ground as $\tilde{X}_k$ where k indexes frames and $k \in \Omega$, as shown in Fig. 4. Our objective is to solve for the complete 3D trajectory $\mathcal{M}_t$, where $t = 0, ..., n-1$, to capture the motion details of the key points. Given the camera parameters estimated in the first step

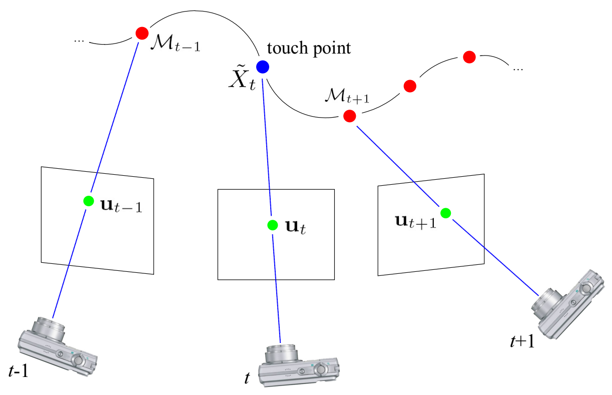


Fig. 4. A 3D trajectory $\{\mathcal{M}_0, \mathcal{M}_1, ..., \mathcal{M}_{n-1}\}$ is obtained for points possibly touching the static background. $\mathcal{M}$ projects to $\mathbf{u}$ in the video frames. The blue point denotes the anchor $\tilde{X}_t$ in frame t.

(Section 3.1), each 3D point in $\mathcal{M}_t$ projects to $\mathbf{u}_t$ in the image plane, as shown in Fig. 4. The depth of $\mathcal{M}_t$, denoted as $z_{\mathcal{M}_t}$, is frame-dependent with respect to the camera parameters. Thus, estimating the 3D position $\mathcal{M}_t$ is equivalent to computing the depth $z_{\mathcal{M}_t}$. Given the camera parameters, projection position $\mathbf{u}_t$ and depth value $z_{\mathcal{M}_t}$, the 3D position $\mathcal{M}_t$ can be expressed as

$$\mathcal{M}_t = \mathbf{R}_t^\top (z_{\mathcal{M}_t} \mathbf{K}_t^{-1} \mathbf{u}_t) - \mathbf{R}_t^\top \mathbf{T}_t. \tag{1}$$

We define a few constraints as follows to estimate $z_{\mathcal{M}}$ in all frames using an optimization method. First, we require $\mathcal{M}_k = \tilde{X}_k$ for all $k \in \Omega$. It is equivalent to minimizing

$$E_1 = \sum_{t \in \Omega} ||z_{\mathcal{M}_t} - z_{\tilde{X}_t}||^2. \tag{2}$$

In addition, we use the following temporal smoothness terms to regularize the solution:

$$\begin{aligned} E_2 \quad = \quad & \sum_{t=0}^{n-2} ||z_{\mathcal{M}_t} - z_{\mathcal{M}_{t+1}}||^2 + \\ & \sum_{t=0}^{n-3} ||2 z_{\mathcal{M}_{t+1}} - z_{\mathcal{M}_t} - z_{\mathcal{M}_{t+2}}||^2. \end{aligned} \tag{3}$$

E_2 minimizes the integration of the first and second derivatives to obtain C0- and C1-continuity. E_2 can also be defined as other energy functions that encourage piecewise smoothness or occasional discontinuities. We use Eq. (3) because we found that the depth of a moving body point in general does not change abruptly in consecutive frames even for the challenging examples shown in this paper. With these two constraints, we solve for $\mathcal{M}$ by minimizing the energy

$$E_D(z_{\mathcal{M}}) = E_1 + \beta E_2, \tag{4}$$

where β is the smoothness weight, and is set to 0.0001 in our experiments. E_D is a quadric energy function and has a closed form solution.

Selecting motion anchors can be done quickly in our system. User performs simple mouse click to indicate the contact points in sparse frames. Then the 3D positions are automatically computed by optimizing depths. This process only takes $1 \sim 2$ seconds for each frame in our experiments. As it is possible to find multiple motion anchors, we use j to index the trajectories and denote the j-th trajectory as $\mathcal{M}^j$.

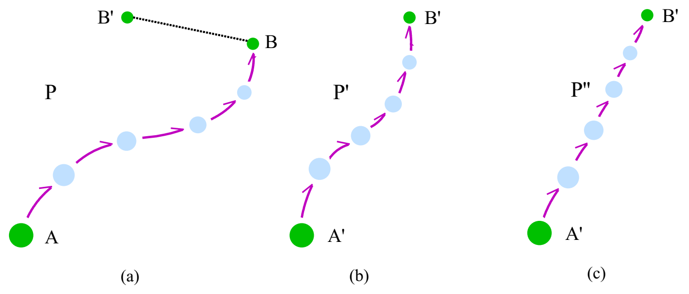


Fig. 5. Motion track transfer. A source point P moves from position A to B in a motion track, shown in (a). To retarget this motion to between endpoints A' and B', we preserve the motion gradient and take A' and B' ($A', B' \in \Psi$) as new boundary conditions to solve a linear system. The transferred motion track is show in (b). The source motion style is naturally preserved in the target. However, using a naïve motion interpolation as shown in (c) fails to capture important details in the curved motion.

3.3 Motion Track Transfer

The above method estimates a set of 2D and 3D motion tracks. Their absolute positions cannot be directly used to animate a 3D character because the reference and target shapes may not be exactly the same. Fig. 7 shows one example of transferring the motion of a man to an armadillo model. Note that their relative lengths of legs and body are quite different. To compensate this discrepancy, we optimize a group of position and projection constraints with *motion gradient*. The notion of *motion gradient* was originally used in the area of optical flow estimation and contour tracking [34]. In this paper, it is defined as the 2D/3D position displacement in consecutive frames.

We first adjust the scale and orientation of the target model and make it approximately aligned with the input video object in the first frame. This process is demonstrated in our supplementary video. We transfer the motion information from tracks $\mathcal{X}$ and $\mathcal{M}$ to the target model as $\hat{\mathcal{X}}$ and $\hat{\mathcal{M}}$ using the motion gradient. It constrains that the motion displacements between source $\mathcal{X}^i$ (or $\mathcal{M}^j$ respectively) and target $\hat{\mathcal{X}}^i$ (or $\hat{\mathcal{M}}^j$) are similar, and is defined as

$$\begin{aligned} \hat{\mathcal{X}}_{t+1}^i - \hat{\mathcal{X}}_t^i \quad &= \quad \mathcal{X}_{t+1}^i - \mathcal{X}_t^i, \\ \hat{\mathcal{M}}_{t+1}^j - \hat{\mathcal{M}}_t^j \quad &= \quad \mathcal{M}_{t+1}^j - \mathcal{M}_t^j. \end{aligned} \tag{5}$$

To obtain a unique solution for (5), we define the Dirichlet boundary condition on a few user manipulated frames (the set is denoted as Ψ). These frames are not continuous. For each frame $t_k \in \Psi$, $\hat{\mathcal{X}}^i$ is set corresponding to an adjusted 2D position $\mathbf{p}_{t_k}$ in frame t_k. We express this condition as

$$\hat{\mathcal{X}}_{t_k}^i = \mathbf{p}_{t_k}. \tag{6}$$

Combining (5) and (6), we solve a linear system and obtain new tracks $\hat{\mathcal{X}}$, which have similar motion as $\mathcal{X}$. The 3D tracks $\hat{\mathcal{M}}$ can be constructed in a similar way. Fig. 5 shows one example. Compared to the naïve motion interpolation, the above system optimizes motion tracks between the reference and target models with sparse point constraints, and hence provides moderate tolerance of shape difference.

3.4 Target Mesh Animation

With the estimated $\hat{\mathcal{X}}$ and $\hat{\mathcal{M}}$, we deform the target model by minimizing a detail-preserving energy similar to the one in [26]. Suppose the deformed model in frame t contains vertices V_t^i and S_t^j that correspond to $\hat{\mathcal{X}}_t^i$ and $\hat{\mathcal{M}}_t^j$ respectively. We construct the following two groups of constraints.

In any frame t, due to the enforced correspondences between the result vertex S_t^j and the 3D point $\hat{\mathcal{M}}_t^j$, we express this condition as

$$S_t^j = \hat{\mathcal{M}}_t^j \tag{7}$$

for all j and t. Each mesh vertex S_t^j in the target 3D model is anchored with a 3D position $\hat{\mathcal{M}}_t^j$.

For the 2D motion tracks, with the similar correspondences, it is required that the coordinate (u_t^i, v_t^i) of $\hat{\mathcal{X}}_t^i$ maps to the camera projection of the result vertex V_t^i in 2D, that is

$$(u_t^i, v_t^i, 1)^\top \sim \mathbf{K}_t(\mathbf{R}_t V_t^i + \mathbf{T}_t)$$

using the estimated camera parameters $\mathbf{K}_t$, $\mathbf{R}_t$, and $\mathbf{T}_t$ in each frame. Denoting $F_t = \mathbf{K}_t\mathbf{R}_t$, $H_t = \mathbf{K}_t\mathbf{T}_t$, the above equations can be rewritten as

$$\begin{aligned}(u_t^i F_t[3] - F_t[1])V_t^i &= H_t[1] - u_t^i H_t[3],\\ (v_t^i F_t[3] - F_t[2])V_t^i &= H_t[2] - v_t^i H_t[3],\end{aligned} \tag{8}$$

for all i and t, where $F_t[s]$ denotes the s-th row of the matrix F_t, and $H_t[s]$ denotes the s-th element of the vector H_t.

Combining (7) and (8), we construct a linear system

$$CU = p, \tag{9}$$

such that U is an unknown vector containing all S and V. C and p are a matrix and a vector, respectively, constructed from (7) and (8). Finally, with a conventional detail preserving function $G(U)$, we minimize the following energy for mesh deformation:

$$E(U) = G(U) + \lambda||CU - p||^2, \tag{10}$$

where λ is a weight and $G(U)$ is a non-linear surface detail preserving energy. It is defined as $G(U) = \|LU - \hat{\delta}(U)\|^2$, where L is a Laplace matrix and $\hat{\delta}(U)$ is the differential coordinate. Their definitions are the same as the one proposed in [26]. Energy $G(U)$ measures the change of the mean curvature normal under a local frame, which reflects the local distortion of the model. Optimizing $E(U)$ helps distribute the distortion over the deformed mesh smoothly. The final objective function is solved using an inexact Gauss-Newton method. In each iteration, the following linear system is solved as a least square problem:

$$AU^{k+1} = b(U^k), \tag{11}$$

where $A = L^\top L + \lambda C^\top C$, and $b(U^k) = L^\top \hat{\delta}(U^k) + \lambda C^\top p$. U^k denotes the value of U in iteration k.

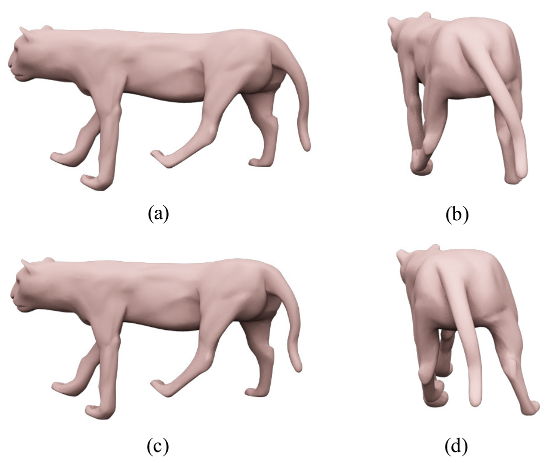


Fig. 6. Depth ambiguity of the 2D constraints. (a-b) One pose of the tiger model from two different views. (c-d) Another pose from the two views. Their frontal views (a) and (c) look identical. But the poses differ in (b) and (d), from a side view.

Note that the deformation produced from this step is by optimizing $E(U)$ for each frame independently. It has a chance to be temporally discontinuous. So we describe in the following section a depth smoothing step to solve this problem.

3.5 Depth Smoothing

One cause of the aforementioned problem is that the 2D motion tracks in image plane have ambiguity in finding corresponding depths. One example is shown in Fig. 6 where two poses look identical from one view ((a) & (c)). But they are dissimilar from another ((b) & (d)) because of different depth assignments for the key points. We thus propose regularizing the deformation by smoothing depths in multiple frames.

The deformation process described in Section 3.4 outputs a depth $\tilde{z}_t^i$ for each key point $\hat{\mathcal{X}}_t^i$. In this step, we refine them by solving the function

$$\begin{aligned}\min \sum_{t,i} ||z_t^i - \tilde{z}_t^i||^2 + \alpha \sum_{t,i} ||z_t^i - z_{t+1}^i||^2 + \\ \alpha \sum_{t,i} ||2z_t^i - z_{t-1}^i - z_{t+1}^i||^2,\end{aligned} \tag{12}$$

where α is the smoothness weight. The data term $||z_t - \tilde{z}_t||^2$ requires that the depth estimate z_t is similar to $\tilde{z}_t^i$, and the terms $||z_t^i - z_{t+1}^i||^2$ and $||2z_t - z_{t-1} - z_{t+1}||^2$ are the first-order and second-order smoothness constraints, respectively. It forms a least square problem and thus can be easily solved. After refining z, with the depth information, all 2D motion tracks are upgraded to 3D. They are taken back into (10) to solve for a refined deformation using the same method described in Section 3.4. The original 2D motion tracks are not used here, as the upgraded 3D motion tracks already contain the corresponding constraints.

3.6 User Control with Two-Pass Propagation

Our system also provides user with tools for conveniently modifying the model. User can iteratively fine tune the defor-

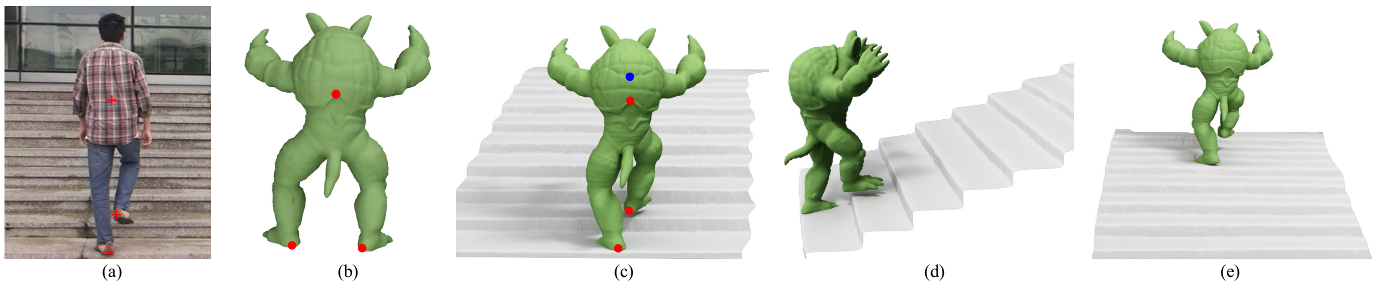


Fig. 7. "Go Upstairs" example. (a) The starting frame of the input video with the tracked points (the red crosses). The 3D trajectories of the two points on the heel are estimated by solving (4). (b) The armadillo model. The red dots are the key points that correspond to the ones in (a), labeled by the user. (c) The starting frame in the computed animation. An extra control point, shown in blue, is added by the user for pose adjustment. (d) The side-view of the character in (c). The pose in frame 80 is shown in (e).

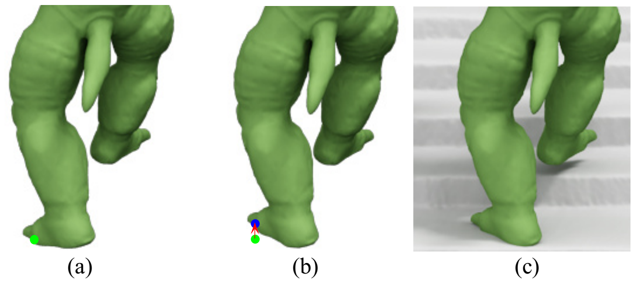


Fig. 8. Local pose tuning with extra control vertices. (a) shows the initial character pose in one frame. The feet penetrates the stairs. We select a control vertex (i.e., the green point) on the tiptoe for local adjustment. (b) and (c) show the illustration without/with stairs.

mation result until satisfied, by manipulating control vertices. The control vertices can be corresponding to the tracked features in the video, or not. Whenever user adds, deletes, or moves the control vertices in the selected frames, our system automatically propagates the modification to other frames. Together with the estimated trajectory and track information from the video, the deformation of the target model is refined. This strategy always outperforms interpolation-based keyframing in terms of the interaction proficiency and result quality. Here we demonstrate how the system refines the deformation when the user moves a vertex v on the mesh in frame t. Other operations such as insertion and deletion of a control vertex work in the similar way.

After changing the position of v in frame t (as shown in Fig. 9(c)), we immediately re-deform the mesh in frame t with this newly added 3D position constraint. Vertex v and frame t are also added to the key point set and the user-editing frame set Ψ (defined in Section 3.3) respectively.

If vertex v is not mapped to a motion track, we need to estimate its motion gradient first. We propose a *two-pass propagation* method to accomplish this task. In the first pass, the positions of the old key points (without including v) in Ψ are taken as boundary conditions. The motion transfer algorithm described in Sections 3.3 and 3.4 is performed to adjust the character poses in the neighboring frames. After the

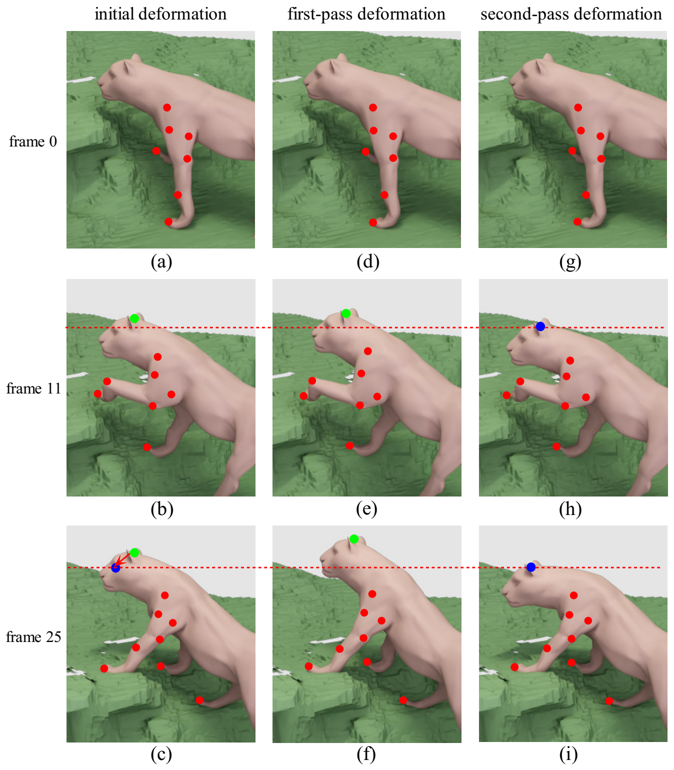


Fig. 9. Two-pass deformation refinement. (a-c) Initial deformation in three frames. Key points are shown in red. User selects one more point (the green one in frame 25) on the model, and moves it to a desired position (shown in blue) for pose adjustment. (d-f) The first-pass deformation result. Although the green point still deviates from the user assigned position in frame 25, its motion gradient is estimated. With the control points as the boundary condition in the edited frames, natural deformation in all frames is yielded by the second-pass deformation as shown in (g-i).

first-pass deformation, the motion track of vertex v is obtained to compute the motion gradients of v in neighborhood frames. Then, in the second pass, we include the 3D constraint of v, and re-deform the mesh sequence. The two-pass deformation naturally propagates the user modification in frame t to the neighboring ones. Note if the motion gradient of v is known beforehand, the deformation propagation can be done in a single pass. But in this case the inference will be significantly dependent of the input where any visual artifacts could lead to unnatural results. In contrast, our multi-pass strategy mitigates this type of influence and thus is more robust.

Fig. 9 demonstrates the effect of this two-pass propagation. The complete sequence is included in the supplementary video[1] (between $2'42''$ and $3'03''$). Figs. 9(d)-(f) show the deformation generated in the first pass for frames 0, 11, and 25. Vertex v (Fig. 9(f)) still deviates from the user-assigned position because it is not used as a motion track constraint in this step. Nevertheless, from the deformed mesh sequence, we can estimate a 3D trajectory for v in all frames. So in the second pass, we take v as a track point for final optimization, faithful to the user modification. We denote the trajectory of v as $\hat{\mathcal{M}}^K$. It is combined with all other tracks to control the deformation using the algorithm described in Sections 3.3 and 3.4. The final deformation result is shown in Figs. 9(g)-(i). It not only contains a new control vertex in frame t, but also has a natural transition among frames.

3.7 System Summary

Fig. 7 shows a working example demonstrating the procedure of our motion transfer. Three feature points are initially tracked on the man in the video – two on the feet and one on the back. Since the two points on the heel touch the stairs in several frames, we recover their 3D coordinates and form trajectories by solving (4) as described in Section 3.2. These points are used to construct motion gradients, which compensate the possible shape deviations between the source and target objects, and facilitate the motion transfer (Section 3.3). Then we label the corresponding key vertices on the target armadillo surface, and adjust its pose in the starting frame. The poses in the following frames can be automatically computed as described in Sections 3.4 and 3.5. Finally, noticing that the feet in some frames penetrate the stairs, we select one more control vertex on the tiptoe and adjust its position in a few frames, as shown in Fig. 8. This modification is automatically propagated to other frames to create natural animation. The pose of frame 80 is shown in Fig. 7(e). Readers are referred to our supplementary video for the illustration.

It should be noted that our progressive pose editing is quite different from traditional keyframing approaches. The latter models poses for a set of frames independently, which requires talent and experience of an artist to envision the naturalness of the character motion in multiple frames. In comparison, our method compensates the character shape discrepancy using motion tracks and gradients. It appropriately adapts the motion of a source video character to the target.

Moreover, keyframing typically requires manipulation on a large number of keyframes for precisely describing motion details, while our method only requires to edit a significantly smaller portion of frames, thank to the desired constraints and optimization. The two-pass propagation strategy always outperforms pose interpolation with the same amount of user input and keyframes. Fig. 12 shows a comparison. The complete sequence is included in our supplementary video (between $3'04''$ and $3'22''$).

1. The supplementary video can be found from the following site: http://www.cad.zju.edu.cn/home/gfzhang/projects/imitation/

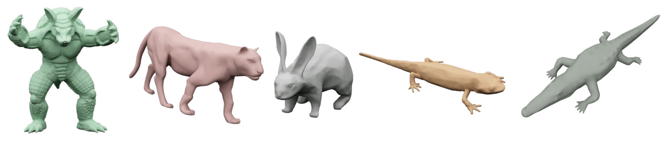


Fig. 10. The collected 3D models to animate, including armadillo, tiger, rabbit, salamander, and crocodile.

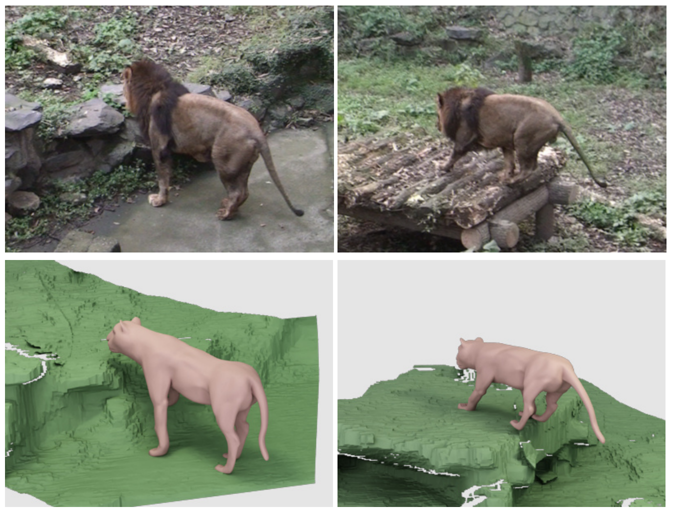


Fig. 11. Lion example. Top row shows two selected frames from the input video. Bottom row shows the corresponding "motion imitation" result on a tiger character.

4 Experimental Results

We have tested the proposed method with several challenging examples where the input videos are taken by a handheld camera. The captured animals include lion, cheetah, rabbit, and salamander. The targeted 3D characters are armadillo, tiger, rabbit, crocodile and salamander (Fig. 10). The results are computed on a desktop computer with a 4-core Xeon 2.0GHz CPU. Table 1 lists the statistics for different examples present in this paper. Complete results are demonstrated in our supplementary video.

Our system can be divided into a few unsupervised operations that include SFM and multi-view stereo (MVS), and phases requiring simple user interactions for interactive feature tracking (IFT) and pose editing. Table 1 lists the running time in different stages. The implementation details on interactive feature tracking are described in Appendix. To process the "Go Upstairs" sequence with 81 frames, our SFM only takes about 3 minutes. The interactive feature tracking takes 5 minutes to track points to obtain a set of 2D motion tracks. For correcting the drifted features, the user only needs to manipulate two frames, then the in-between feature positions can be automatically re-estimated. After obtaining the 2D motion tracks, we

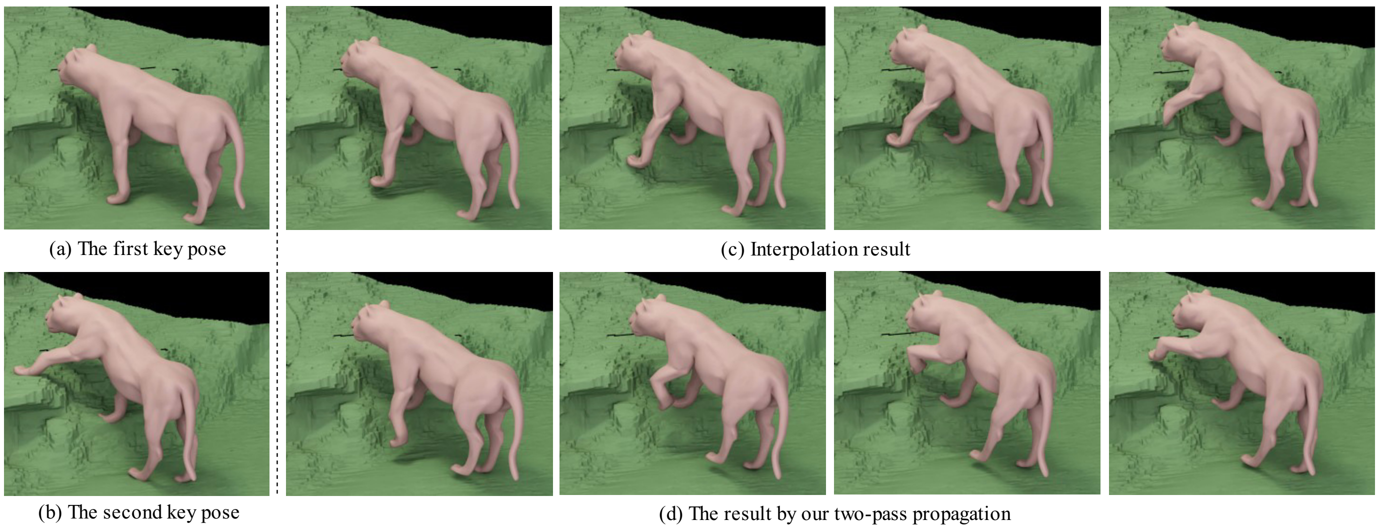


Fig. 12. Comparison of our method and the interpolation-based keyframing. (a) and (b) show two key poses, in between which we infer the others automatically. (c) The interpolated poses by a mesh morphing technique based on differential coordinates [35]. (d) The poses obtained using our two-pass propagation. The comparison shows that our method can naturally preserve subtle motion details.

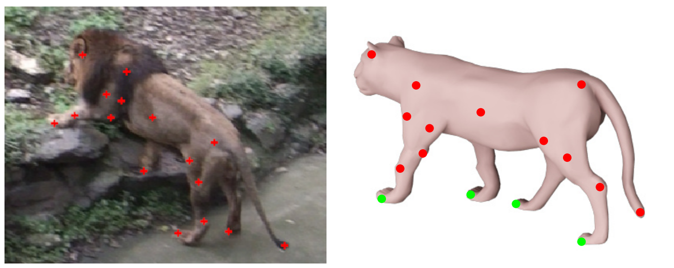


Fig. 13. 16 tracked points in the video (left image) and their corresponding 3D vertices on the tiger model (right image).

TABLE 1
The statistics of examples present in the supplementary video and this paper.

Sequences	Go Upstairs	Lion	Cheetah	Salamander	Rabbit
frames	81	240	150	100	50
3D model	armadillo	tiger	tiger	crocodile/ salamander	rabbit
# mesh vertices	6002	2507	2507	5002/10002	5002
# tracked feature points	3	16	13	10	9
# Pose-edited frames	7	18	20	11/21	3
# SFM (min)	3	8	6	–	3
# MVS (min)	40	90	60	–	–
# IFT (min)	5	90	60	90	20
# Pose Editing (min)	10	180	210	90/210	5

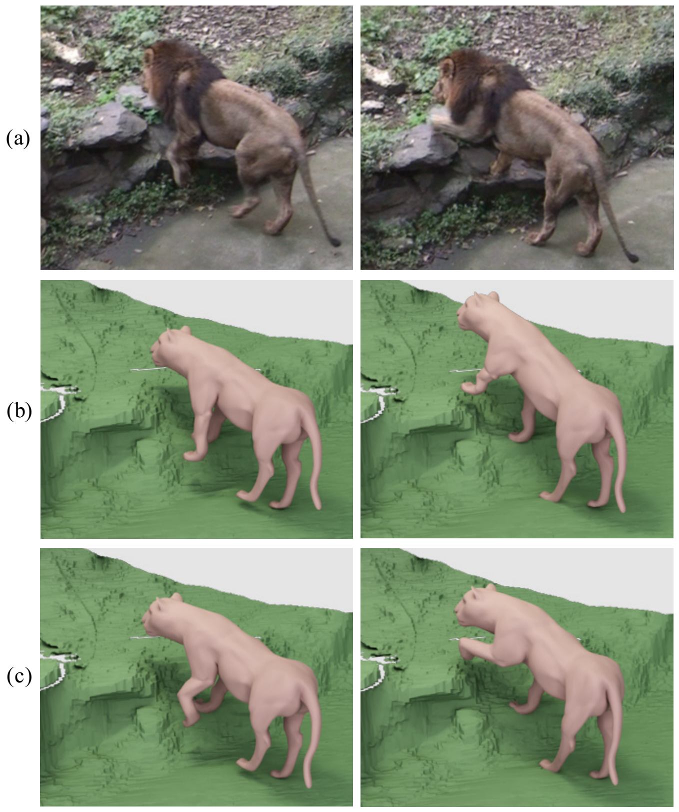


Fig. 14. Deformation with/without 2D motion tracks. (a) Two frames extracted from the reference video. (b) Deformation result only using 3D motion anchors. (c) Deformation result using both 3D and 2D motion tracks. 2D tracks on the legs and body help faithfully transfer the non-rigid deformation details from the reference character to the target.

select special points that contact with the ground as motion anchors. This operation is only performed on the visible points in a few frames. Selected contact points do not need much accuracy because motion tracks after all are optimized by preserving motion gradient with boundary conditions. Our progressive pose adjustment is also very efficient. It is about 10 minutes for “Go Upstairs” example. It should be noted that pose editing refers to the total computation needed for generating the mesh animation, which not only counts user interaction, but also contains re-computation of the character poses in multiple frames (as described in Sections 3.4-3.6). Note that the latter takes the majority of the time. The manual intervention involves inserting and moving control vertices in sparse keyframes for pose adjustment by simple mouse click and dragging (Please refer to our supplementary video for more details). Each time after user manipulates a few points in one frame, s/he can choose to propagate the edit to other frames, by re-executing the automatic animation steps, to see how the resulted sequence looks like. The propagation process includes re-optimizing 2D/3D motion constraints (preserving motion gradients) and re-deforming the mesh subsequence.

In the lion example shown in Fig. 11, the input sequence contains 240 frames. The lion motion involves rock climbing and jumping onto the wooden platform. They are very complex for motion transfer. The muscle on the leg has large non-rigid deformation. We select 16 points on the lion for tracking, as shown in Fig. 13 – four points on the claws, and the other 12 points are on the body and legs. Since the points on the claws touch the background scene in several frames, we recover their 3D positions by solving Eq. (4). With the tracked key points, we transfer the motion of a lion to a tiger model with 2,507 mesh vertices. Only using the 3D motion anchors on the claws cannot naturally transfer the non-rigid deformation on the legs and body, as shown in Fig. 14(b). In comparison, by utilizing the 2D tracks on the legs and body, we faithfully preserve deformation details, as shown in Fig. 14(c). Note that this type of detail preservation would be very difficult for the skeleton-based methods since body deformation is highly non-rigid in general. We include the complete sequence comparison in the supplementary video (between $3'23''$ and $3'40''$).

We edited 18 frames in total for progressive pose adjustment for the lion sequence. Compared to traditional keyframing, our local editing is much more straightforward because user does not need to be concerned about the motion continuity and subtle detail preservation in multiple frames. All modifications are automatically propagated to other frames to avoid the jittering artifacts. The deformation time is approximately linear to the number of the mesh vertices. For a mesh with 2507 vertices, the function construction time with matrix factorization for Eq. (10) is 0.2 second in a single thread; solving Eq. (11) in each iteration takes about 0.02 second. For deformation in each frame, $5 \sim 10$ iterations are sufficient in our experiments. Note that matrix A does not need to be reconstructed if the manual control vertices are not added or deleted during the course of interactive pose adjustment. So the system feedback to user interaction is almost in real-time. The manual pose refinement for each user-modified frame typically requires a few minutes, depending on the motion complexity and the desired animation quality. For most pose adjustment, it requires only a few clicks and drags. After deformation, we insert the animated tiger into the background scene, and render it as shown in Fig. 11.

Figs. 16 and 17 show other animation results. For the salamander example shown in Fig. 16, the camera does not move. So the 3D information cannot be recovered from the video. We track 10 points on the salamander, on the head, back, legs and tail respectively, as shown in Fig. 15. By assuming the point at the back has constant depth, we recover the 3D trajectory of this point in the sequence, and then use it as a 3D position constraint. Other 9 points are used for 2D projection constraints. For the rabbit example (Fig. 17), since the desk is planar, we select the recovered 3D points on the desk to fit the desk plane. In addition, the right hand side of the rabbit cannot be observed. We resolve this ambiguity by assuming the right legs undergo the same motion as the left ones.

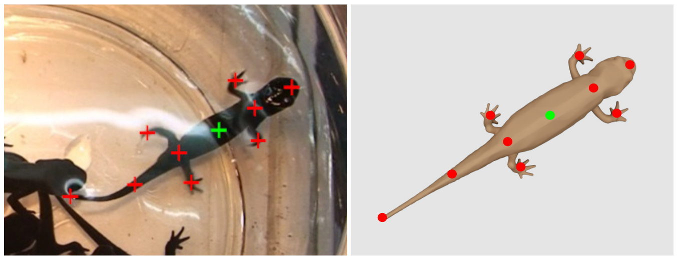


Fig. 15. 10 tracked points and their corresponding 3D vertices are shown in the extracted frame and the salamander model respectively. We estimate the 3D trajectory of the point at the back, and then use it as a 3D position constraint. Other 9 points are used for 2D projection constraints.

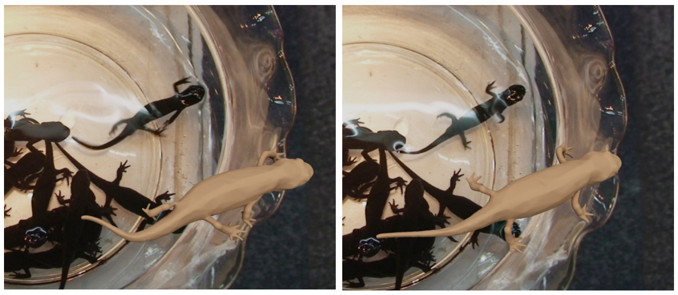


Fig. 16. Salamander example. Two frames selected from the input video show how the animation is retargeted to the 3D model.

In discussion, the amount of user interaction and the number of pose-edited frames mainly depend on the complexity of appearance and motion, and even the mesh quality. As shown in Table 1, in processing the “Go Upstairs” and rabbit examples, our interactive pose editing is rather efficient, only requiring 10 and 5 minutes, respectively. For the cheetah and salamander examples, since the motion is much more complex, more user interactions are required and denser frames are need to be edited. The interactive pose-editing time for the rabbit example is only 0.1 minute/frame. It increases to 1.4 minutes/frame for the cheetah example. Our interactive feature tracking is also directly related to the complexity of appearance and motion. For the salamander sequence, the selected key points are textureless and there are serious reflections, translucency, and fast motion, which make the feature tracking extremely

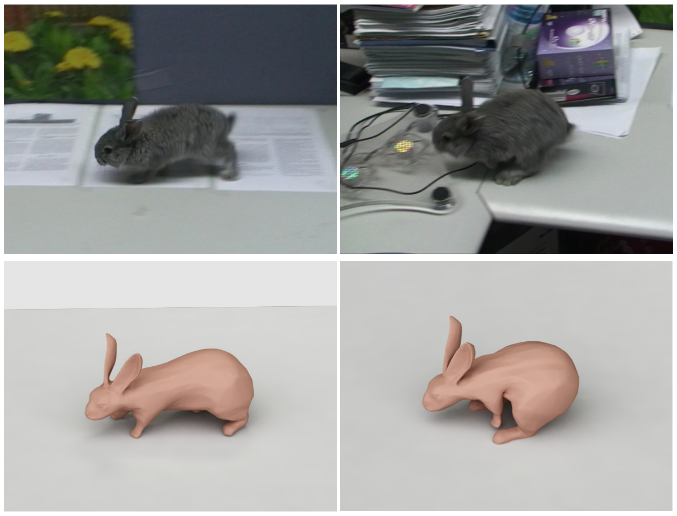


Fig. 17. Rabbit example. Top row shows the original frames from the input video. Bottom row shows the motion retargeted result on a rabbit model.

challenging. Compared to the "Go Upstairs" sequence, the average tracking time increases from 1.2 seconds/frame to 5.4 seconds/frame for each point.

The amount of user interaction also depends on the required mesh quality. For the salamander example, we transfer the 3D motion to the crocodile and salamander models. Between them, the salamander model contains more mesh vertices and many slim triangles around the tiny legs, which makes mesh deformation more challenging. Therefore, more user interactions are required for locally adjusting the leg poses.

5 Conclusions and Discussion

We have presented a comprehensive system capable of properly "extracting" motion from a dynamic object in a monocular video and retargeting it to a 3D character. The motion data are described as a few sparse key points tracked in this sequence. To obtain necessary 3D motion constraints, our method first recovers the camera parameters and the static background. Then we look for contact points between the dynamic object and static background so as to infer the corresponding 3D trajectories in the whole sequence. Our system significantly expands the number and variety of the sources of motion data, and can be appropriately used to estimate the motion of the small-scale and wild animals that are difficult to wear trackers for a motion capture system. Our method also saves animators from tedious and time-consuming manual keyframing.

Our method can preserve certain fine details of motion. The results included in the paper and in the supplementary video have demonstrated the effectiveness of our method. Taking the lion model as an example, subtle non-rigid deformation of the leg muscle is faithfully transferred to the target character. The ability comes from the detail-preserving energy function. The extracted 2D tracks are also quite useful to describe motion details. Increasing the number of feature tracks can help preserve even more of them.

If the motion details cannot be observed from the video due to frequent or consistent occlusion, there is basically no way to obtain sufficient visual information and accordingly the motion data. Currently, we use the temporal smoothing and symmetry constraints to alleviate this problem. We believe with multiple videos captured from different views, this problem could be better addressed.

In addition, it is possible that the input video contains dynamic background. In this extreme case, we may still be able to obtain partial motion estimate by either increasing user intervention to adjust the character poses or making depth assumptions. If the body shapes of the reference and target characters differ too much, it requires more effort to manipulate the motion data, as our system requires control points. We believe this problem can possibly be solved by first transferring our tracked motion data from the video object to an appearance-similar 3D character, and then applying the mesh deformation transfer technique [29] to animate the target 3D character.

Appendix: Interactive Point Tracking On Video

Automatically extracting long and accurate feature tracks from a video is very challenging due to possible occlusions and viewpoint/appearance changes, as described in Section 3. We propose a simple and yet very effective interactive approach to solve this problem. It can yield instant feedback and has no specific preprocessing requirement. High accuracy can also be ensured.

For a track $\{\mathcal{X}_t^i\}$ where i and t index the track and frame respectively, we allow the user to manually correct the drifted features. This process only needs to be done in the user selected frames. Then our system automatically solves for the remaining feature positions. Suppose L and R are two such frames that user operates. Features in frame t, where $L < t < R$, are estimated by solving a function involving three terms. They respectively represent the matching coherence, appearance smoothness, and motion smoothness.

The matching term $e(\mathcal{X}_t^i)$ encodes the local appearance similarity between the corresponding points in multiple frames. It is measured in local windows W centered at these points, as shown in Fig. 18. We denote the window for $\{\mathcal{X}_t^i\}$ as W_t^i and copy the colors of all pixels in W_t^i to vector $\mathbf{p}(\mathcal{X}_t^i)$ in a scanline order. $e(\mathcal{X}_t^i)$ is defined as

$$\begin{aligned} e(\mathcal{X}_t^i) &= w(t)\frac{||\mathbf{p}(\mathcal{X}_t^i)-\mathbf{p}(\mathcal{X}_L^i)||^2}{|W|} + \\ & (1-w(t))\frac{||\mathbf{p}(\mathcal{X}_t^i)-\mathbf{p}(\mathcal{X}_R^i)||^2}{|W|}, \end{aligned} \tag{13}$$

where $w(t) = (R-t)/(R-L)$ is a weight function to balance the appearance similarities with regard to $\mathcal{X}_L^i$ and $\mathcal{X}_R^i$ respectively based on a distance metric. $|W|$ is the size of the window.

The appearance smoothness term is defined as the appearance distance between temporally adjacent $\mathbf{p}(\mathcal{X}_t^i)$ and $\mathbf{p}(\mathcal{X}_{t+1}^i)$; and the motion smoothness term measures the

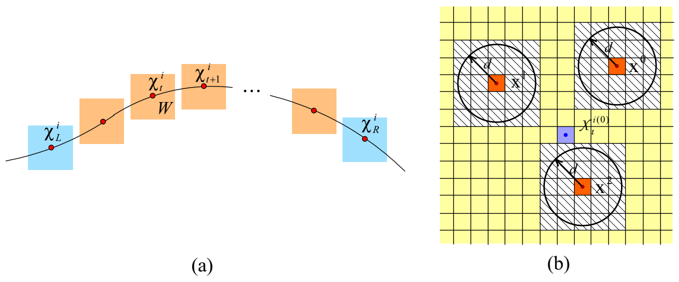


Fig. 18. Interactive tracking with DP optimization. (a) The track points in multiple frames form a single chain. The blue rectangles denote frames L and R that user operates. $\mathcal{X}_t^i$, for all $L < t < R$, is the position to be estimated in the intermediate frames. The local window centered at $\mathcal{X}_t^i$ describes the point appearance. (b) Candidate pruning for local windows centered at $\mathcal{X}_t^{i(0)}$ for different i. $\mathbf{x}^0$, $\mathbf{x}^1$ and $\mathbf{x}^2$ are the selected candidates, among which minimum distance d is enforced.

Algorithm 1 Candidate Pruning

```
1) Sort all pixels in a local window, centered at $\mathcal{X}_t^{i(0)}$,
   with respect to the cost $e(\mathcal{X}_t^i)$. The re-ordered pixels
   are denoted as $\{\mathbf{x}^k\}_{k=1,\ldots,N}$.
2) Define $\{V(\mathbf{x}^k)\}_{k=1,\ldots,N}$ as boolean variables, which are
   initialized to zeros. The set of position candidates is
   denoted as $C(\mathbf{x})$. It is initially empty.
3) For $i = 1, \ldots., N$,
       if $V(\mathbf{x}^k) = 0$ & $|C(\mathbf{x})| < 20$, add $\mathbf{x}^k$ to $C(\mathbf{x})$
           for each pixel $\mathbf{y}$ satisfying $||\mathbf{x}^k - \mathbf{y}|| < d$,
               $V(\mathbf{y}) = 1$.
```

position similarity between adjacent $\mathcal{X}_t^i$ and $\mathcal{X}_{t+1}^i$. The final objective function combines all these terms:

$$\begin{aligned} E(\mathcal{X}^{L\to R}) &= \sum_{t=L}^{R} e(\mathcal{X}_t^i) + \lambda_1 \sum_{t=L}^{R-1} \frac{||\mathbf{p}(\mathcal{X}_t^i) - \mathbf{p}(\mathcal{X}_{t+1}^i)||^2}{|W|} \\ &\quad + \lambda_2 \sum_{t=L}^{R-1} ||\mathcal{X}_t^i - \mathcal{X}_{t+1}^i||^2, \qquad (14) \end{aligned}$$

where λ_1 and λ_2 are two cost weights, and are set to 1.0 and 0.1 respectively.

Optimization We now describe the method to minimize energy $E(\mathcal{X}^{L\to R})$. As illustrated in Fig. 18(a), since the points in a track form a single chain, we can use dynamic programming (DP) for optimization. Given m nodes and N candidates for each node, the complexity of DP is $O(mN^2)$. If N is large, the optimization will be slowed down. In our system, we compute the initial estimate of $\mathcal{X}^i$, denoted as $\mathcal{X}^{i(0)}$, by linearly interpolating $\mathcal{X}_L^i$ and $\mathcal{X}_R^i$. Based on the observation that the true position of $\mathcal{X}_t$ is generally in the neighborhood of these estimates, we introduce an effective pruning algorithm (Algorithm 1) to dramatically accelerate DP.

We first select a reasonable number (20 in our experiments) of candidates within the circular local window centered at each $\mathcal{X}_t^{i(0)}$. These candidates produce small costs in $e(\mathcal{X}_t^i)$ and are not close to each other because our algorithm enforces the minimal distance criteria, as illustrated in Fig. 18(b). With the small number of candidates for each feature, DP is performed to efficiently find the global optimum. For further acceleration, we employ a coarse-to-fine optimization scheme [36] with a Gaussian pyramid. The initial local search radius r is set to 10, and the initial distance d, defined in Algorithm 1, is set to 3. Both r and d are gradually reduced in iterations. Four passes are sufficient to find accurate match positions. In our experiments, tracking a point in 50 frames only takes around 4 seconds, or equivalently 12fps in speed, sufficient for the interactive operations. Compared to the optimization method of Buchanan and Fitzgibbon [37], our method does not need to perform feature search in the whole image and has no preprocessing. It hence yields very high efficiency.

Acknowledgments

The authors would like to thank the associate editor and all the reviewers for their constructive comments to improve the manuscript. Thanks to Zilong Dong and Lei Jiang for their enormous help to implement auxiliary tools and prepare video. Thanks to Michael S. Brown for the video narration. This work is supported by the 973 program of China (No. 2009CB320801), NSF of China (No. 60633070 and 60903135), and the Research Grants Council of the Hong Kong Special Administrative Region, under General Research Fund (Project No. CUHK 412307 and 417107).

References

[1] D. Vlasic, R. Adelsberger, G. Vannucci, J. Barnwell, M. H. Gross, W. Matusik, and J. Popovic, “Practical motion capture in everyday surroundings,” *ACM Trans. Graph.*, vol. 26, no. 3, p. 35, 2007.
[2] M. Gleicher, “Retargeting motion to new characters,” in *SIGGRAPH*, 1998, pp. 33–42.
[3] C. Hecker, B. Raabe, R. W. Enslow, J. DeWeese, J. Maynard, and K. van Prooijen, “Real-time motion retargeting to highly varied user-created morphologies,” *ACM Trans. Graph.*, vol. 27, no. 3, 2008.
[4] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.
[5] M. Pollefeys, L. J. V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch, “Visual modeling with a hand-held camera.” *International Journal of Computer Vision*, vol. 59, no. 3, pp. 207–232, 2004.
[6] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, “A comparison and evaluation of multi-view stereo reconstruction algorithms,” in *CVPR (1)*, 2006, pp. 519–528.
[7] P. Tan, G. Zeng, J. Wang, S. B. Kang, and L. Quan, “Image-based tree modeling,” *ACM Trans. Graph.*, vol. 26, no. 3, p. 87, 2007.
[8] A. van den Hengel, A. R. Dick, T. Thormählen, B. Ward, and P. H. S. Torr, “Videotrace: rapid interactive scene modelling from video,” *ACM Trans. Graph.*, vol. 26, no. 3, p. 86, 2007.
[9] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. A. J. Winder, and R. Szeliski, “High-quality video view interpolation using a layered representation,” *ACM Trans. Graph.*, vol. 23, no. 3, pp. 600–608, 2004.
[10] Y. Furukawa and J. Ponce, “Dense 3D motion capture from synchronized video streams,” in *CVPR*, 2008.
[11] D. Bradley, T. Popa, A. Sheffer, W. Heidrich, and T. Boubekeur, “Markerless garment capture,” *ACM Trans. Graph.*, vol. 27, no. 3, 2008.
[12] L. Torresani, A. Hertzmann, and C. Bregler, “Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 5, pp. 878–892, 2008.
[13] V. Rabaud and S. Belongie, “Re-thinking non-rigid structure from motion,” in *CVPR*, 2008.
[14] M. Gleicher and N. J. Ferrier, “Evaluating video-based motion capture,” in *CA*, 2002, pp. 75–80.

[15] T. B. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Computer Vision and Image Understanding*, vol. 104, no. 2-3, pp. 90–126, 2006.

[16] R. Poppe, "Vision-based human motion analysis: An overview," *Computer Vision and Image Understanding*, vol. 108, no. 1-2, pp. 4–18, 2007.

[17] P. Sand, L. McMillan, and J. Popovic, "Continuous capture of skin deformation," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 578–586, 2003.

[18] J. P. Lewis, M. Cordner, and N. Fong, "Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation," in *SIGGRAPH*, 2000, pp. 165–172.

[19] T. Ju, S. Schaefer, and J. Warren, "Mean value coordinates for closed triangular meshes," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 561–566, 2005.

[20] D. Zorin, P. Schröder, and W. Sweldens, "Interactive multiresolution mesh editing," in *SIGGRAPH*, 1997, pp. 259–268.

[21] S. Kircher and M. Garland, "Editing arbitrarily deforming surface animations," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1098–1107, 2006.

[22] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum, "Mesh editing with poisson-based gradient field manipulation," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 644–651, 2004.

[23] Y. Lipman, O. Sorkine, D. Levin, and D. Cohen-Or, "Linear rotation-invariant coordinates for meshes," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 479–487, 2005.

[24] K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, and H.-Y. Shum, "Large mesh deformation using the volumetric graph laplacian," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 496–503, 2005.

[25] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or, "A sketch-based interface for detail-preserving mesh editing," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 1142–1147, 2005.

[26] J. Huang, X. Shi, X. Liu, K. Zhou, L.-Y. Wei, S.-H. Teng, H. Bao, B. Guo, and H.-Y. Shum, "Subspace gradient domain mesh deformation," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1126–1134, 2006.

[27] O. K.-C. Au, C.-L. Tai, L. Liu, and H. Fu, "Dual laplacian editing for meshes," *IEEE Trans. Vis. Comput. Graph.*, vol. 12, no. 3, pp. 386–395, 2006.

[28] W. Xu, K. Zhou, Y. Yu, Q. Tan, Q. Peng, and B. Guo, "Gradient domain editing of deforming mesh sequences," *ACM Trans. Graph.*, vol. 26, no. 3, p. 84, 2007.

[29] R. W. Sumner and J. Popovic, "Deformation transfer for triangle meshes," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 399–405, 2004.

[30] C. Bregler, L. Loeb, E. Chuang, and H. Deshpande, "Turning to the masters: motion capturing cartoons," in *SIGGRAPH*, 2002, pp. 399–407.

[31] L. Favreau, L. Revéret, C. Depraz, and M.-P. Cani, "Animal gaits from video: Comparative studies," *Graphical Models*, vol. 68, no. 2, pp. 212–234, 2006.

[32] G. Zhang, X. Qin, W. Hua, T.-T. Wong, P.-A. Heng, and H. Bao, "Robust metric reconstruction from challenging video sequences," in *CVPR*, 2007.

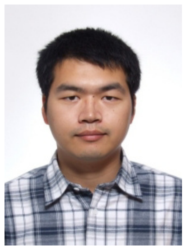
[33] G. Zhang, J. Jia, T.-T. Wong, and H. Bao, "Consistent depth maps recovery from a video sequence," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 6, pp. 974–988, 2009.

[34] N. Ray and S. T. Acton, "Motion gradient vector flow: an external force for tracking rolling leukocytes with shape and size constrained active contours," *IEEE Trans. Med. Imaging*, vol. 23, no. 12, pp. 1466–1478, 2004.

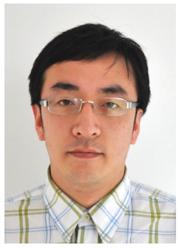
[35] M. Alexa, "Differential coordinates for local mesh morphing and deformation," *The Visual Computer*, vol. 19, no. 2-3, pp. 105–114, 2003.

[36] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani, "Hierarchical model-based motion estimation," in *ECCV*, 1992, pp. 237–252.

[37] A. Buchanan and A. W. Fitzgibbon, "Interactive feature tracking using k-d trees and dynamic programming," in *CVPR (1)*, 2006, pp. 626–633.

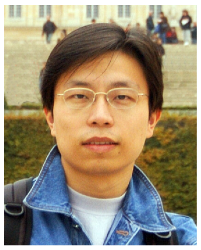


Guofeng Zhang received the BS and PhD degrees in Computer Science from Zhejiang University in 2003 and 2009, respectively. He is currently a Postdoc at State Key Laboratory of CAD&CG, Zhejiang University. His research interests include camera tracking, 3D reconstruction, augmented reality, video segmentation and editing. He is a member of IEEE.



Hanqing Jiang received the BS degree in computer science from Zhejiang University, P.R. China, in 2006. He is currently working toward the PhD degree in computer science at the State Key Laboratory of CAD&CG, Zhejiang University. His main research interests include video segmentation and 3D modeling.

Jin Huang is currently an associated professor in the state key laboratory of CAD&CG at Zhejiang University, P.R.China. He received his PhD. degree in Computer Science Department from Zhejiang University in 2007 with Excellent Doctoral Dissertation Award of CCF (China Computer Federation). His research interests include geometry processing and physically-based simulation. He has served as reviewer for ACM SIGGRAPH, EuroGraphics, Pacific Graphics, TVCG etc.



Jiaya Jia received the PhD degree in computer science from the Hong Kong University of Science and Technology in 2004. He joined the Department of Computer Science and Engineering, Chinese University of Hong Kong, in September 2004, where he is currently an assistant professor. His research interests include vision geometry, image/video editing and enhancement, and motion estimation. He has served on the program committees of ICCV, CVPR, ECCV, and ACCV. He served as co-chair of the interactive computer vision workshop 2007 (in conjunction with ICCV07). He is a member of IEEE.

Tien-Tsin Wong received the B.Sci., M.Phil., and PhD. degrees in computer science from the Chinese University of Hong Kong in 1992, 1994, and 1998, respectively. Currently, he is a Professor in the Department of Computer Science & Engineering, Chinese University of Hong Kong. His main research interest is computer graphics, including computational manga, image-based rendering, natural phenomena modeling, and multimedia data compression. He received IEEE Transactions on Multimedia Prize Paper Award 2005 and Young Researcher Award 2004.

Kun Zhou is a Cheung Kong Distinguished Professor in the Computer Science Department of Zhejiang University, and a member of the State Key Lab of CAD&CG, where he leads the Graphics and Parallel Systems Group. Prior to joining Zhejiang University in 2008, Dr. Zhou was a Leader Researcher of the Internet Graphics Group at Microsoft Research Asia. He received his BS degree and PhD degree in computer science from Zhejiang University in 1997 and 2002, respectively. His research interests include shape modeling/editing, texture mapping/synthesis, real-time rendering and GPU parallel computing.

Hujun Bao received the BS and PhD degrees in applied mathematics from Zhejiang University in 1987 and 1993, respectively. Currently, he is a Professor and the director of State Key Laboratory of CAD&CG at Zhejiang University. His main research interest is computer graphics and computer vision, including realtime rendering technique, geometry computing, virtual reality and 3D reconstruction.