Computers & Graphics 71 (2018) 55–65

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on Cyberworlds 2017

Real-time fish animation generation by monocular camera

Xiangfei Meng[a], Junjun Pan[a,*], Hong Qin[b], Pu Ge[a]

[a] State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, China
[b] Department of Computer Science, Stony Brook University, USA

ARTICLE INFO

Article history:
Received 28 October 2017
Revised 19 December 2017
Accepted 19 December 2017
Available online 27 December 2017

Keywords:
Fish animation
Markerless motion capture
Monocular camera
Motion retargeting
Motion fine tuning

ABSTRACT

Fish animation generation is an interesting topic since it plays an important role in building virtual underwater worlds. Accurate motion capture and flexible retargeting of fish is difficult, in particular with the challenges of underwater marker attachment and feature description for soft bodies. Little research into this problem has been published and real-time fish motion retargeting with a desirable motion pattern remains elusive. Motivated by our goal of achieving high-quality data-driven fish animation with a light-weight, mobile device, this paper develops a novel framework of motion capturing, retargeting, and fine tuning for a fish. We demonstrate a markerless technique for the motion capture of an actual fish using a monocular camera. The elliptical Fourier coefficients are then integrated into the contour-based feature extraction process to analyze fish swimming patterns. This novel approach can obtain motion information in a robust way, utilizing the smooth medial axis as the descriptor for a soft fish body. For motion retargeting, we propose a two-level scheme to transfer the captured motion into new models, such as 2D meshes (with texture) generated from pictures or 3D models designed by artists, regardless of the body geometry and fin proportions amongst various species of fish. Both the motion capture and retargeting processes operate in real time. Hence, the system can obtain video sequences of real fish using a monocular camera and simultaneously create fish animation with variation. In addition, a motion fine tuning method is provided for animators to efficiently refine the retargeted frames in an interactive manner. It can enhance the final output animation to an appropriate level of fidelity.

1. Introduction

Fish animation is an interesting research topic due to its important role in building virtual underwater world. Generally, the motion style in animation tends to be more realistic, if the data-driven animation approach is utilized to provide more accurate motion details. In data-driven approaches, the motion data can be captured from the real world by motion capture (or Mo-cap) technique. Motion capture has been widely used in video game production and film industry for the last two decades, with the goal of creating natural character animation and special effects. Realistic motion data of an actor can be captured by a mo-cap system and then retargeted to new characters, creating animation with the same motion as the actor but with new appearances. Nonetheless, traditional mo-cap systems suffer from noisy motion data resulting from occlusion, interference [1] or skin artifact [2], and specialized high-end devices which are too expensive to afford by the general public [3]. Moreover, it is extremely difficult to attach markers to certain types of characters such as underwater creatures (e.g., fish).

To address the limitations above, we develop a markerless motion capture technique to record the motion of an underwater fish in this paper. Since the most movement of a fish takes place in the horizontal plane, we simply employ a monocular camera to record the fish movement from the top of a fish tank. The swimming fish (i.e., foreground region) is segmented by a background subtraction algorithm. Based on the extracted fish contour, the head and tail can be located successively. However, the accurate medial axis of a fish body serving as a spine is difficult to represent, due to the asymmetric motion of the fish fins. To solve this problem, we employ elliptical Fourier coefficients [4] to convert the fish contour from the original physical domain to the frequency domain, and then, reconstruct it with fewer coefficients. Similar to the low pass filter, it gives rise to a smoothed contour. Then a smooth medial axis, which is considered as the deformable spine, can be generated for the soft fish body. Finally, the remaining feature points can be located based on the medial axis and the original contour. By retargeting such feature points as fish motion to 2D/3D meshes with texture information, the final fish animation can be obtained. With our proposed data-driven method, even the general public

* Corresponding author.
E-mail address: pan_junjun@buaa.edu.cn (J. Pan).

https://doi.org/10.1016/j.cag.2017.12.004
0097-8493/

without specially-trained artistic skills is capable of creating fish animation with low-cost mobile devices, such as a cellphone with a camera.

Motion retargeting technique has been widely studied among characters which can be represented with skeleton models. Motion retargeting of fish, however, is more challenging for the flexible body and variant fin locations. Essentially, fish motion can be decomposed into two parts: the global motion including position and orientation, and the local motion which is the deformation under local coordinates. The estimation and recovery of global motion are straightforward since it consists of a rotation and a translation only. As for the local deformation, simply scaling the motion w.r.t. the skeleton according to the fish body ratio will cause unnatural geometry distortion, especially in the vicinity of the fins area. In this paper, we propose a two-level motion retargeting scheme, in which the local shape transferring process is divided into two steps. At the first step, we only transfer the body motion into the target model, during which the junction points between the fish body and two fish fins are regarded as a part of the fish body and can be accurately retargeted. Afterwards, we transfer the fin motion through the junction points, serving as both local control points and relative positions of fin motion. With this two-level motion retargeting approach, we can properly transfer the motion of a real fish into a target model.

To improve the final output animation to an appropriate level of fidelity, we propose a motion fine tuning approach to interactively editing the retargeted frames. During the refinement stage, each editing operation on a certain frame will influence the neighboring frames. It avoids the tedious frame-by-frame manipulation from users.

In particular, the innovative contributions of our research can be summarized as follows:

- We develop a data-driven based technique for fish animation generation with a low-cost device. Using a monocular camera, the motion sequences of a real fish can be captured and retargeted to a 2D/3D fish or fish-like model in real time. With comprehensive functionalities, our system also supports interactions from users to edit retargeted frames for final animation production.
- The smooth medial axis of a soft fish body is hard to acquire, due to the asymmetry of its fins and contour during fish swimming. We incorporate elliptical Fourier coefficients into the contour-based feature extraction. Then, we reconstruct the contour with low-frequency harmonics, which can effectively attenuate the asymmetry of the fins while preserving the fish global shape. Finally, a preferable medial axis can be extracted.
- We propose a two-level motion retargeting scheme to transfer the motion from an original fish sequence to new models. The major challenge is to handle various body proportions and fin locations. Here, we decompose the local motion retargeting process into two steps. First, the motion of a fish body is transferred to the target model, with the positions of junction points determined. At the second step, the motion of fins can be accurately retargeted with the junction points, serving as both local control points and relative positions of the fin motion.

2. Related work

Motion capture and motion retargeting have been studied widely in computer vision and computer graphics. However, both of them remain open and challenging problems. Given the significant literature in these areas, we focus on the most relevant researches.

2.1. Video-based motion capture

Most video based mo-cap systems employ a number of synchronized cameras to acquire multiple views of the character with markers or sensors, which has been well studied in the existing literature [5]. Motion capture from the video is more challenging for the lack of depth information and occlusion of feature points or markers. Most video-based approaches estimate the pose by searching from the state space [6] or relying on Bayesian filtering with the prior models of dynamics [7] and focus on human or articulated characters only [8–10]. In contrast, our mo-cap system is designed for fish or fish-like creatures.

In the field of motion capture for animals, Ju et al. [11] present a mo-cap system for a bird but their method is marker-based and needs multiple high-speed video cameras. Wilhelms and Gelder [12] propose an interactive video-based mo-cap system for character animation. They identify features and establish feature relationships from frame to frame, but their method is developed only for articulated characters such as horses. Lee et al. [13] present a video-based fish animation generating system which is the most similar work to ours, but their work relies on multi-view cameras and does not take the motion of fins into consideration. Yu and Terzopoulos [14] present a synthetic motion capture to render plenty of fish at an interactive rate. However, the motion data they use is generated from the simulation of the biomechanical model, while we directly use captured videos as a reliable data source.

Video-based motion capture can be regarded as extracting motion from image sequences. There are some classic approaches for point tracking [15] or object tracking [16]. Shi and Tomasi present the famous KLT tracking algorithm [15] to track good features which are selected under the principle that a good feature is the one that can be tracked well. The features selected by KLT tracker are mostly corners while the body of fish is too flexible to obtain stable tracking across a large number of frames. Comaniciu et al. [16] propose the mean-shift tracking approach for real-time non-rigid object tracking, which is used in the mo-cap system presented in [8] to track the motion of a human body. However, the mean-shift algorithm calculates the shift vector based on the histogram of the object region, which makes it hard to track a patch of a fish due to its self-similar texture. In contrast, our contour-based feature extraction avoids the direct tracking to a specific point or patch, thus is able to provide robust motion information.

2.2. Motion retargeting

Motion retargeting is a classic topic which aims to transfer the motion from one character to another, while keeping styles of the original motion. Most existing work is either offline or requires a large database of example motion. An offline method presented by Gleicher [17] uses spacetime constraints on example motion. They considers different segment lengths but identical structure of characters with fewer degrees of freedom (DOF). Similar limitations exist in many other offline methods [18,19]. Shin et al. [20] present an online method using inverse kinematics (IK), but only take identical skeletal topologies into consideration. Kulpa et al. [21] propose an approach to supporting different numbers of bones in limbs by using IK in real time, but only on humanoid topologies. Popović and Witkin [19] applies the principles of physical-based animation and constraint optimization formulation on motion data. But like most other approaches, it retargets the motion to articulated figures only. An interesting method proposed by Bregler [22] is able to retarget the motion from a cartoon character to another. It needs the user to define each key-shape motion to synthesize the final animation. Hornung et al. propose a retargeting approach [23] similar to ours that can animate photos of 2D

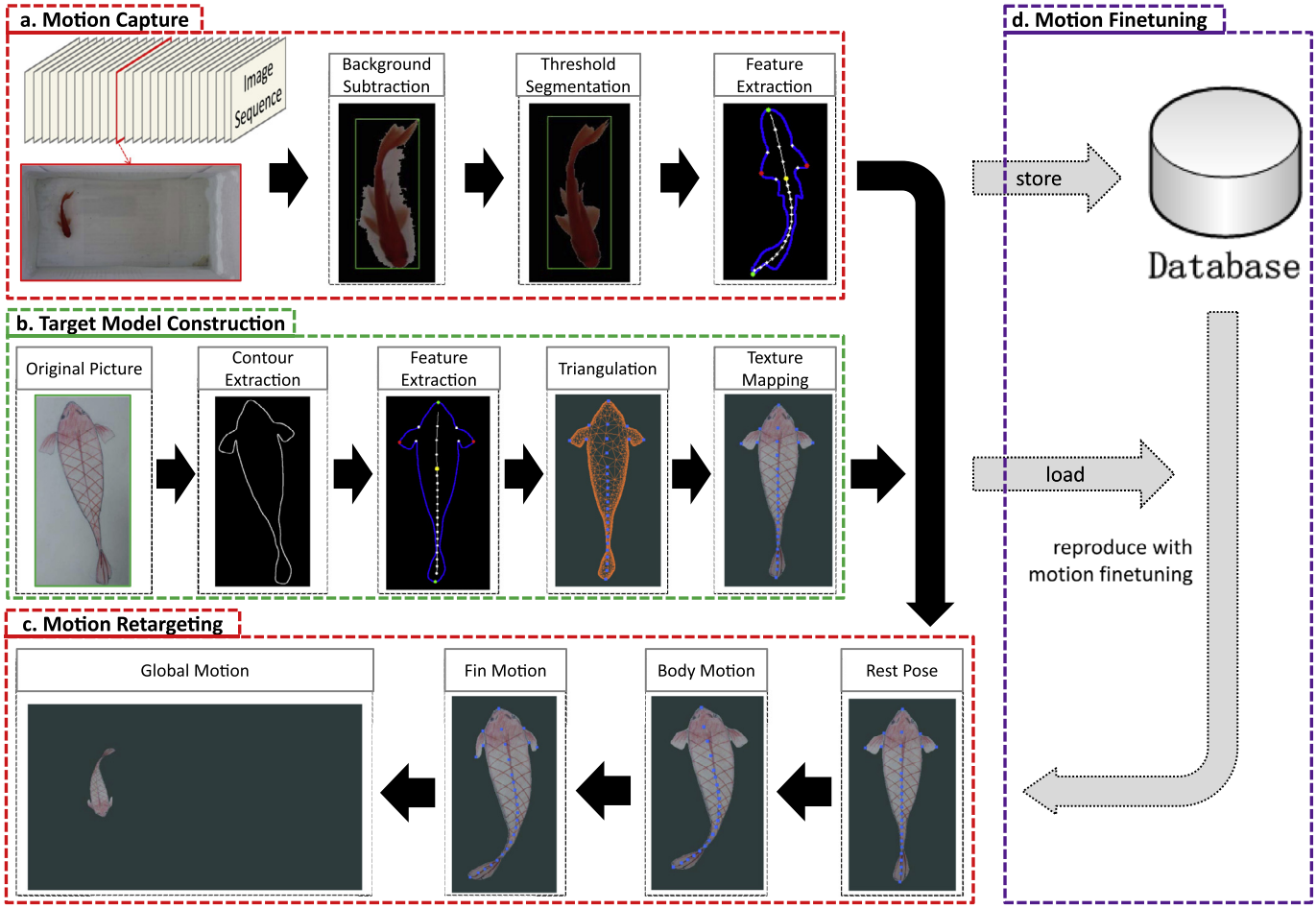

Fig. 1. Flow chart of our technique. (a) The motion of a swimming fish in water is captured. (b) The target model is constructed from a hand-drawn cartoon fish. (c) Body motion, fin motion, and global motion are retargeted sequentially. (d) Animation can be reproduced by the recorded motion and an imported model, along with interactive editing. Both (a) and (c) are executed synchronously in real time. (b) is executed only once. (d) is an interactive post-process for animation production.

characters. They present a solution to reconstruct a projective camera model and a 3D model pose which matches best to the given 2D image. However, the motion types are limited and the animation characters must be humanoid.

Several sketch-based motion editing techniques can also be considered in the scenario of motion retargeting since they hold several high-level shape descriptors to achieve the character posing [24,25]. However, some of them relied on stick figures [26,27]. As the major concern of sketch-based character posing is to describe the motion as abstract shape descriptors, few of them can work in real time [28].

In terms of fish motion retargeting, it's difficult to treat the fish as an articulated character and represent it with a traditional skeleton model, due to the flexibility of the fish body and the variant locations of the fins. The two-level retargeting scheme we propose treats the fish topology as two parts. The body part can deal with the flexibility of fish body, and the fin part is able to handle the variance of fin locations.

3. System overview

Our motion capture and retargeting technique aims at providing a framework to drive a 2D/3D fish or fish-like model to swim lively with the same motion style as the real one in the video. The motion fine tuning technique is a postprocessing tool which allows users to efficiently edit the retargeted frames to output the final animation with a high level of fidelity. Fig. 1 shows the pipeline.

3.1. Motion capture

After acquiring a swimming fish video, we employ an adaptive background subtraction method: ViBe [29] to detect the foreground of each frame. A morphological filter and a connected component filter are applied to remove the salt and pepper noise and the non-significant foreground region, leaving salient connected components as the foreground objects. Once the user selecting a certain object, the system will regard it as the swimming fish and keep tracking on it. While the fish is being tracked, its motion will be continuously recorded by the contour-based feature extraction procedure and then delivered to the motion retargeting module.

3.2. Target model construction

In Fig. 1(b), the target model is constructed from a hand-drawn cartoon fish. The aim of this module is generating a target 2D/3D model with several control points to achieve the animation. This task is processed off-line and only once. Given a picture containing a cartoon fish, we preprocess the image to acquire the contour, then the control points will be extracted by the contour-based feature extraction process. Finally, the fish model is generated by constrained Delaunay triangulation [30]. For 3D models, we take a snapshot on the top view of the fish model, extract the control points from the 2D snapshot (as same as the 2D fish picture), and finally, recover them to the 3D coordinate system.

3.3. Motion retargeting

The recorded motion consists of global motion and local motion, where the former one is represented by a rotation and a translation, and the latter one is represented by the positions of the control points of the fish model. The control points can be divided into two groups: body control points and fin control points. At the first stage, we transfer the local motion to the body control points of the fish model, then deform the model with body control points only. Thus we can obtain the positions of junction points, which are deformed as a local part of the fish body just now. At the second stage, we transfer the local motion to the fin control points with the help of junction points, then deform the fish model with all control points. Afterwards, the final retargeted model is obtained by applying a rotation and a translation according to the global motion. For shape deformation, we choose a moving-least-squares (MLS) based approach [31] £due to its high efficiency for the closed-form solution and low distortion stemming from the least square error.

3.4. Motion fine tuning

In extreme cases, several feature points of the fish may hardly be located accurately. It can cause distortion frames. Besides, animators may want to modify the final output for special purposes, such as exaggerated effects. Here we provide a motion fine tuning function for users to edit consecutive frames interactively. The editing manipulation in a certain frame will influence its neighboring frames. The range of effected neighboring frames depends on both the current frequency of the fish motion and the order of current frame in a consecutive motion sequence.

4. Motion capture

In each frame, the foreground pixels are identified as binary mask by using ViBe [29]. The salt and pepper noise is eliminated through a morphological opening operation on the mask. Then we identify the connected components and remove the components whose areas are too small, leaving the salient components as foreground objects.

The system will start tracking once user selecting one object. The reason why we do not automate this selecting process is that the tracking algorithm needs time to initialize the background model. Besides, there could be multiple objects in the scene at the same time, we can import a simple user interaction to avoid introducing supervised learning to recognize which is a real fish.

As shown in Fig. 1(a), an adaptive threshold image segmentation method [32] is employed to eliminate the noise caused by shadow. After we get an accurate contour of the fish, the motion will be captured by a contour-based feature extraction process.

Here we list the motion attributes (position or orientation) in Table 1 and label them in Fig. 2, where *u_num* and *l_num* are the sampling numbers of the upper segment and lower segment of the medial axis, respectively.

The feature extraction process can be divided into four steps: global motion estimation, head and tail recognition, medial axis extraction, and fin recognition.

4.1. Global motion estimation

The aim of this step is to estimate the center position (**center**) and the orientation (**dir**) of the fish. We use PCA method to process the coordinates of all pixels inside the closed contour to obtain the mean value, eigenvalues and eigenvectors. The mean value that representing the barycenter is assigned to **center** temporarily. As the barycenter will sometimes drift apart from the medial axis when the fish forms “C-shape”, we’ll update the value of **center** later after the medial axis is extracted.

As for orientation estimation, we firstly choose the eigenvector with the larger eigenvalue as **dir**. However, it is possible the negative direction of the fish head. So the direction of the eigenvector will be adjusted by the orientation of the last frame so that the angle of the orientations between two consecutive frames won’t be larger than $\pi/2$.

Table 1
Fish motion attributes.

Attribute	Explanation
center	Fish center
dir	Major direction, representing the fish orientation
head	Fish head
tail	Fish tail
l_fin	The left pectoral fin tip
r_fin	The right pectoral fin tip
jun_tl	The top-left junction point
jun_tr	The top-right junction point
jun_bl	The bottom-left junction point
jun_br	The bottom-right junction point
up_axis$_{\{1,2,\ldots,u\_num\}}$	Sample points in the upper segment of the medial axis, from center towards head
low_axis$_{\{1,2,\ldots,l\_num\}}$	Sample points in the lower segment of the medial axis, from center towards tail

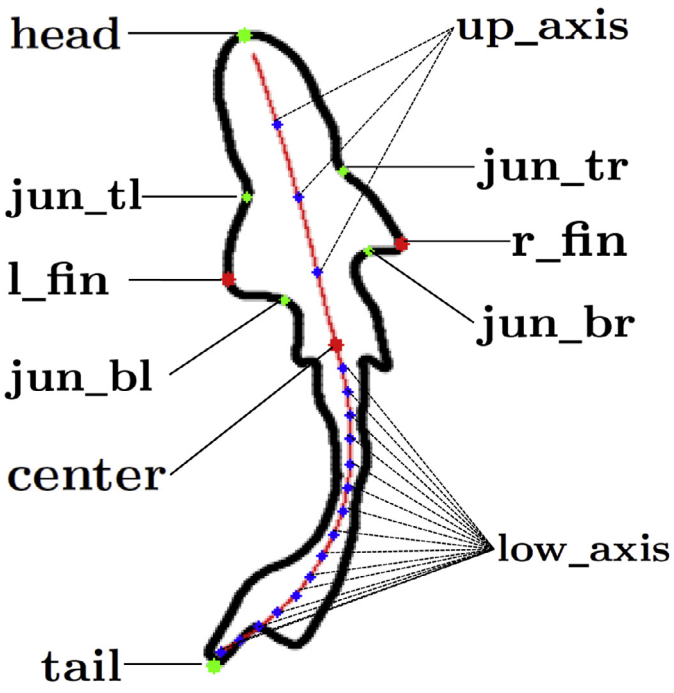

Fig. 2. Motion attributes of a fish. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article).

4.2. Head and tail recognition

The recognition of **head** and **tail** is straightforward. At first, we calculate all the distances between contour points and the barycenter. The point with the largest distance is treated as **tail**. Then we divide the contour point sequence into two equal-length segments such that **tail** is the midpoint of the one, leaving **head** in the other. Therefore we can locate **head** as the point with the largest distance in its own segment. Occasionally, there could be a chance to misidentify **head** and **tail** in reverse order. In case of that, we will use the orientation **dir** as a calibration.

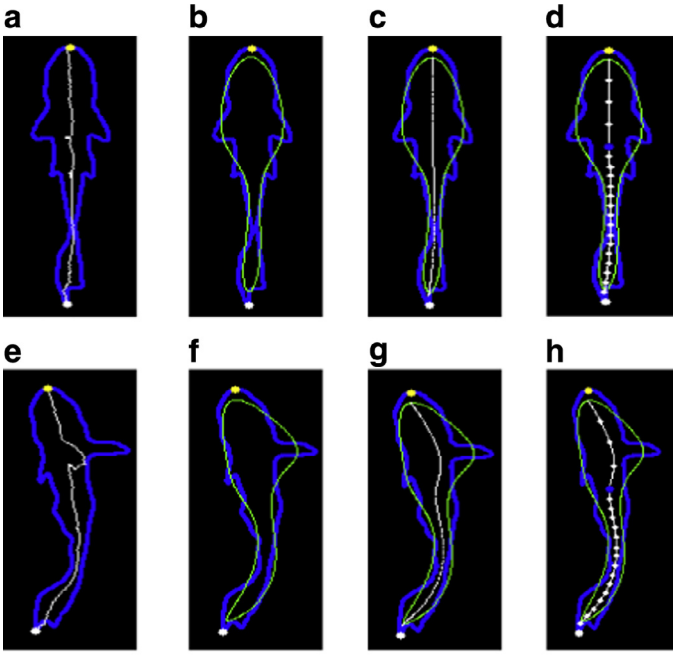

Fig. 3. The functionality of elliptical Fourier coefficients in medial axis extraction. (a) Jagged medial axis extracted directly from the original contour. (b) Smooth contour reconstructed by using elliptical Fourier coefficients. (c) Smooth medial axis extracted from the reconstructed contour. (d) Sampled points in the medial axis. (e) Biased medial axis caused by the asymmetric fins. (f) Asymmetry is largely eliminated by the reconstructed contour. (g) Smooth medial axis with little bias. (h) Sampled points in the medial axis.

For general purpose, considering some species of fish that have two tips in its caudal fin, we will detect whether there is another local max-distance point near **tail** as the other tip. If the other tip is found, **tail** will be updated as the point in the contour segment between the two local max-distance points and having the minimum distance.

4.3. Medial axis extraction

The core part of the motion attributes is the fish medial axis which is drawn in red in Fig. 2, since it is the most important descriptor of the flexible fish body. There exist plenty of medial axis extraction methods [33–37]. However, the time burden of them is too heavy to reach the real-time application. Thus it is preferable for us to develop an efficient and effective medial axis extraction method particularly for fish.

At the beginning, we divide the closed contour into two segments, which is the two point sequences from **head** to **tail** in different ways. Without loss of generality, assume the number of points in the two sequences is $l1$ and $l2$ with $l1 \le l2$. Then the coordinates of the two sequences can be represented by $\mathbf{p}_{\{1,2,\ldots,l1\}}$ and $\mathbf{q}_{\{1,2,\ldots,l2\}}$. And the medial axis can be given by

$$\mathbf{m}_i = \frac{\mathbf{p}_i + \mathbf{q}_{\lfloor \frac{l2}{l1} i \rfloor}}{2}, \quad i = 1, 2, \ldots, l1. \tag{1}$$

However, this intuitive strategy has two problems. As we can see in Fig. 3(a), the fish contour is not smooth enough thus is likely to produce a jagged axis. In Fig. 3(e), the medial axis is pretty biased because the two pectoral fins are asymmetric. Under this circumstance, we believe that most information a contour carries is used to represent the detail, and the body shape can be represented by only a small, and exactly the low-frequency part of the whole contour's information.

The spatial filter could be a choice to smooth the contour to eliminate high-frequency part. However, it's not easy to determine how large the filter template should be. Theoretically, we must know both the amount of the whole information, and the amount of the information we want to keep to describe the fish body. And then we can elaborately design a spatial filter to extract the low-frequency information. Apparently, there are two factors influencing the filter design, and the amount of the whole information is strongly relevant to the size of the contour (which can be variant). Therefore, we employ the elliptical Fourier coefficients technology to transform the contour into the frequency domain, where we only need to care about the amount of information to keep (how many coefficients we select). Once the frequency filter is determined, we can use it permanently as long as the complexity of the fish model is not changed.

Elliptical Fourier coefficients model a closed contour as sums of elliptical harmonics. Each harmonic is described by four coefficients, interpreted geometrically as major axis length, minor axis length and the orientation of the ellipse. As shown in [4], we regard the contour as two periodic functions $x(t)$ and $y(t)$. For example, $x(t)$ can be written as

$$x(t) = a_0 + \sum_{k=1}^{\infty} \left(a_k \cos \frac{2k\pi t}{T} + b_k \sin \frac{2k\pi t}{T} \right), \tag{2}$$

where T is the perimeter of the contour, $t = 2\pi l/T$, l is the arc length from a preset starting point, and the coefficients can be calculated as

$$a_0 = \frac{1}{T} \sum_{p=1}^{K} \frac{\Delta x_p}{2\Delta t_p} (t_p^2 - t_{p-1}^2) + \zeta_p (t_p - t_{p-1}), \tag{3}$$

$$a_k = \frac{T}{2k^2\pi^2} \sum_{p=1}^{K} \frac{\Delta x_p}{\Delta t_p} \left(\cos \frac{2k\pi t_p}{T} - \cos \frac{2k\pi t_{p-1}}{T} \right), \tag{4}$$

$$b_k = \frac{T}{2k^2\pi^2} \sum_{p=1}^{K} \frac{\Delta x_p}{\Delta t_p} \left(\sin \frac{2k\pi t_p}{T} - \sin \frac{2k\pi t_{p-1}}{T} \right), \tag{5}$$

with K being the number of points in the contour, $\Delta x_p = x_p - x_{p-1}$, $\Delta t_p = \sqrt{(\Delta x_p)^2 + (\Delta y_p)^2}$, and

$$\zeta_p = \sum_{j=1}^{p-1} \Delta x_j - \frac{\Delta x_p}{\Delta t_p} \sum_{j=1}^{p-1} \Delta t_j. \tag{6}$$

$y(t)$ can be defined in terms of the coefficients c_0, c_k and d_k similarly.

After obtaining the coefficients from the original contour, we use the first S harmonics to reconstruct a smooth contour according to Eq. (2) with the same point number as the origin. The reconstruction result is shown in Fig. 3(b) and (f) with S being 5. Afterwards, we use the reconstructed contour to update point sequences $\mathbf{p}_{\{1,2,\ldots,l1\}}$ and $\mathbf{q}_{\{1,2,\ldots,l2\}}$. Then the medial axis can be extracted according to Eq. (1), as shown in Fig. 3(c) and (g), which becomes much more smooth. The number of harmonics S is a hyperparameter which is chosen by experience. The convenience of parameters choosing in elliptical Fourier coefficients is that S will remain constant once the desired smoothness is fixed. It means that the variance of the size and the complexity of the original contour can be neglected.

As mentioned in Section 4.1, the barycenter is likely to drift apart from the medial axis when the fish body is really curved. Since we already have the medial axis, we can choose the point among medial axis whose distance is the least to the barycenter, then record its position as the new **center**.

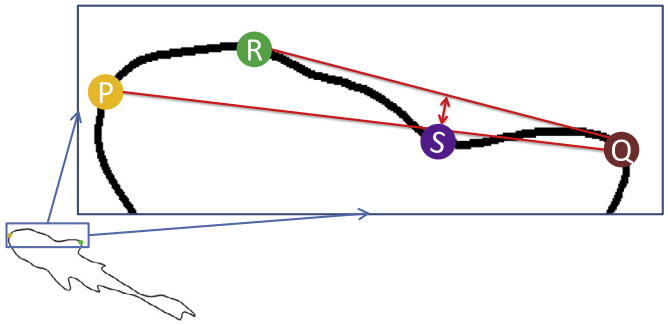

Fig. 4. The recognition process of **jun_tr**. *P* denotes the head, *Q* denotes the right fin tip, *R* is an assistant point, and *S* denotes the recognized **jun_tr**.

Since more feature points are required to represent the curved body deformation in the lower part of the fish, we sample the medial axis with different densities. We uniformly sampled *u_num* points in the segment of the medial axis from center point towards head ($\mathbf{up\_axis}_{\{1,2,\ldots,u\_num\}}$), and *l_num* points in the segment from center point towards tail ($\mathbf{low\_axis}_{\{1,2,\ldots,l\_num\}}$). The result is illustrated as Fig. 3(d) and (h) with *u_num* begin 3 and *l_num* being 15.

4.4. Fin recognition

Besides describing the body shape, the medial axis can offer a great help to locate the fins. We divide the contour into two segments with **head** and **tail** as the breakpoints, then find the point with the largest distance to the medial axis in each segment as the left or right pectoral fin's tip (**l_fin** and **r_fin**). Note that we only extract the pectoral fins of the fish, for the pelvic fin and dorsal fin are usually occluded by the fish body thus can hardly be recognized consistently.

The remaining task is to locate four junction points (Fig. 2). The method to find each junction point is basically the same, so here we only describe how to find the upper junction between right fin and body (**jun_tr**). For the contour segment from **head** to **r_fin** (Fig. 4), supposing *P* represents **head** and *Q* represents **r_fin**, we search a point *R* on the contour segment between *P* and *Q* to ensure $\angle PQR$ is the maximum. Then the position of the point among the segment $\widetilde{QR}$ with the largest distance to the line $\overrightarrow{QR}$ is regarded as **jun_tr**.

5. Motion retargeting

Before motion retargeting, we need to construct a target model from 2D pictures or 3D meshes for retargeting. As shown in Fig. 1(b), the 2D target model is constructed from a hand-drawn fish. We use a Canny edge detector to generate an edge image as binary mask. Control points can be extracted through the contour-based feature extraction process, which is basically the same as illustrated in Section 4. The mesh is generated from the contour by constrained Delaunay triangulation [30]. Finally, the original picture is attached as the texture of the model.

For a 3D fish model, we can directly import the mesh and texture into our system. In order to generate the positions of control points, we use PCA to calculate three principle directions of the vertices. Then we apply a translation and rotation to the model, making it lying on the *x*–*y* plane, with its center at the origin, head pointing to the positive *y*-axis. A 2D snapshot is taken at the top view of the model, from which control points can be extracted with planar coordinates. Then we recover the control points to the 3D coordinate system with $z = 0$.

Algorithm 1 Simplified Curve Skeleton Construction.

Input: Explicit model *E*
Output: Bone length *L*

1: Compute *head_len* and *tail_len* of *L* as the distances between **head**, **tail** and the uppermost, the lowermost ending sample points in the medial axis of E.
2: Compute *up_len* and *low_len* of *L* as the piecewise lengths of the sampled points in the medial axis of E.
3: Compute l_fin_len and r_fin_len of *L* as the length between *E.C.***l_fin** and *E.C.***jun_tl** and the length between *E.C.***r_fin** and *E.C.***jun_tr**
4: $L.body\_len = \sum_{i=1}^{u\_num} up\_len_i + \sum_{i=1}^{l\_num} low\_len_i + L.head\_len + L.tail\_len$

The motion retargeting module consists of three stages – body motion, fin motion and global motion. The first two stages are the core part of the two-level retargeting scheme we propose, and the last stage aims to restore the fish's global motion.

In fact, we have two fish models constructed after target model construction. One is an implicit model which only works during body motion retargeting. The other is an explicit model which is utilized in the fin motion retargeting and displayed on the screen finally. The implicit model only has body control points while the explicit model contains all control points. In addition, the four junction points are added to the implicit model as ordinary vertices, thus can be transformed during body motion retargeting.

For explanation, we define the implicit model *I* and explicit model *E*:

$$I =< V, C >, \tag{7}$$

$$E =< V, C > \tag{8}$$

where *V* denotes the vertices

$$I.V = \{\mathbf{v}; \mathbf{jun\_tl}, \mathbf{jun\_tr}, \mathbf{jun\_bl}, \mathbf{jun\_br}\}, \tag{9}$$

$$E.V = \{\mathbf{v}\} \tag{10}$$

and *C* denotes the control points

$$I.C = \{\mathbf{center}, \mathbf{head}, \mathbf{tail}, \mathbf{up\_axis}, \mathbf{low\_axis}\} \tag{11}$$

$$E.C = \{\mathbf{center}, \mathbf{head}, \mathbf{tail}, \mathbf{up\_axis}, \mathbf{low\_axis}, \mathbf{jun\_tl}, \mathbf{jun\_tr}, \mathbf{jun\_bl}, \mathbf{jun\_br}, \mathbf{l\_fin}, \mathbf{r\_fin}\} \tag{12}$$

and **v** denotes the vertices generated from the triangulation process.

To deform the target fish model, the motion structure of the target fish should be analyzed. Here we use a simplified curve skeleton *L* to represent the structure of the fish model.

$$L = \{body\_len, head\_len, tail\_len, l\_fin\_len, r\_fin\_len, up\_len_{\{1,2,\ldots,u\_num\}}, low\_len_{\{1,2,\ldots,l\_num\}}\} \tag{13}$$

The construction of *L* is described in Algorithm 1, which is executed only once after the target model is constructed.

We choose angles of bones together with **center** and **dir** as the intermediate motion parameters, which transfer between feature points of the actual fish and control points of the fish model. Specifically, for each video frame, we acquire the value of all the motion attributes listed in Table 1, and then calculate an intermediate motion *M*:

$$M = \{\mathbf{center}, \mathbf{dir}, head\_angle, tail\_angle, l\_fin\_angle, r\_fin\_angle, up\_axis\_angle_{1,2,\ldots,u\_num}, low\_axis\_angle_{1,2,\ldots,l\_num}\}; \tag{14}$$

Algorithm 2 Intermediate Motion Calculation.

Input: All motion attributes in one video frame
Output: Intermediate Motion M

1: $M.\mathbf{center} = \mathbf{center}$
2: $M.\mathbf{dir} = \mathbf{dir}$
3: $M.head\_angle = angle(\mathbf{head} - \mathbf{up\_axis}_{u\_num})$
4: $M.tail\_angle = angle(\mathbf{tail} - \mathbf{low\_axis}_{l\_num})$
5: $M.l\_fin\_angle = angle(\mathbf{l\_fin} - \mathbf{jun\_tl})$
6: $M.r\_fin\_angle = angle(\mathbf{r\_fin} - \mathbf{jun\_tr})$
7: $\mathbf{v1} = [\mathbf{center}, \mathbf{up\_aixs}_1, \ldots, M.\mathbf{up\_aixs}_{u\_num-1}]$
8: $\mathbf{v2} = [\mathbf{center}, \mathbf{low\_aixs}_1, \ldots, M.\mathbf{low\_aixs}_{l\_num-1}]$
9: $M.\mathbf{up\_axis\_angle} = angle(\mathbf{up\_axis} - \mathbf{v1})$
10: $M.\mathbf{low\_axis\_angle} = angle(\mathbf{low\_axis} - \mathbf{v2})$

The calculation of M is described in Algorithm 2, which is executed per video frame after the motion attributes are acquired.

5.1. Body motion retargeting

Body motion retargeting is realized by transferring the fish's local motion to the control points of the implicit fish model I, and then applying a deformation on the mesh of I. The details of implicit model deformation are described in Algorithm 3.

Algorithm 3 Implicit Model Deformation.

Input: Implicit model I, bone length L and intermediate motion M
Output: The deformed implicit model I

1: $I.C.\mathbf{center} = \mathbf{0}$
2: Sequentially construct $I.C.\mathbf{up\_axis}$ according to $L.up\_len$ and $M.\mathbf{up\_axis\_angle}$
3: Sequentially construct $I.C.\mathbf{low\_axis}$ according to $L.low\_len$ and $M.\mathbf{low\_axis\_angle}$
4: $I.C.\mathbf{head} = polarcart(L.head\_len, M.head\_angle) + I.C.\mathbf{up\_axis}_{u\_num}$
5: $I.C.\mathbf{tail} = polarcart(L.tail\_len, M.tail\_angle) + I.C.\mathbf{low\_axis}_{l\_num}$
6: Deform $I.V$ according to $I.C$

The function *polarcart*(*len*, *angle*) is used to translate coordinates from polar coordinate system to Cartesian coordinate system. We employ a point-based deformation method [31] to deform the target model under the principle of moving least squares. Its closed-form solution can provide high computation performance. Meanwhile, unlike some other deformation approaches [38,39] which require control points belonging to the existing vertices of the mesh, this method allows control points to be added at any positions, thus provides convenience for our sampled medial axis points (**up_axis** and **low_axis**) serving as control points.

After body motion retargeting, we can obtain an intermediate result of the target fish model (Fig. 5(c)), in which the fish body is deformed but the fins remain the same as the rest pose.

5.2. Fin motion retargeting

Once the implicit model I is deformed, the positions of the four junction points are determined. Then we can retarget the fin motion to the explicit model E with four junction points serving as relative positions. The details are described in Algorithm 4. The result is shown in Fig. 5(d).

5.3. Global motion retargeting

After the first two stages, we are able to retarget the local motion accurately to the fish model. In the end, we need to transform the deformed model E to the world coordinate system with a rotation and a translation. The rotation angle can be directly determined by orientation $M.\mathbf{dir}$, while the translation needs to be adjusted to meet the difference of the scales between the actual fish and the target fish model. Here we use a straightforward way to obtain the position of the fish model E after translation as

$$E.C.\mathbf{center} = \frac{L.body\_len}{init\_len} M.\mathbf{center}, \tag{15}$$

where $init\_len$ is the medial axis length of the actual fish which is recorded in the first tracking frame.

Finally, for the smoothness of the final animation, we linearly interpolate 5 animation frames between two consecutive video frames, and use Kalman filters to estimate the control points' positions in each frame.

Algorithm 4 Explicit Model Deformation.

Input: Explicit model E, bone length L and recorded motion M
Output: The deformed explicit model E

1: Copy the positions of the body control points from $I.C$ to $E.C$.
2: Copy the positions of the deformed four junction points from $I.V$ to $E.C$.
3: $E.C.\mathbf{l\_fin} = polarcart(L.l\_fin\_len, M.l\_fin\_angle) + E.C.\mathbf{jun\_tl}$
4: $E.C.\mathbf{r\_fin} = polarcart(L.r\_fin\_len, M.r\_fin\_angle) + E.C.\mathbf{jun\_tr}$
5: Deform $E.V$ according to $E.C$

6. Motion fine tuning

While the fish animation is generated in real time during the motion capture and retargeting process, we store the recorded motion in a database. Another interface is then provided to reproduce the fish animation using the stored motion data together with a fish model. When the animation is being reproduced, users are allowed to interactively pause the process in one frame, and edit the position of the control points in the fish model to manipulate its deformed shape. Each editing operation to a certain frame will exert influence on its neighboring frames with various extent in unimodal distribution (such as Gaussian distribution), of which the specific distribution parameter depends on both the frequency of the fish motion, and the order of the current frame in its effected frame sequence.

Specifically, the motion fine tuning function should meet the following requirements:

- One editing operation should influence not only the current frame, but also the neighboring frames. Meanwhile, the influenced frames must be limited inside the current effected frame sequence.
- The influence extent should be in unimodal distribution, with the current editing frame as the peak. And the farther the frame distance is, the less the motion will be influenced.

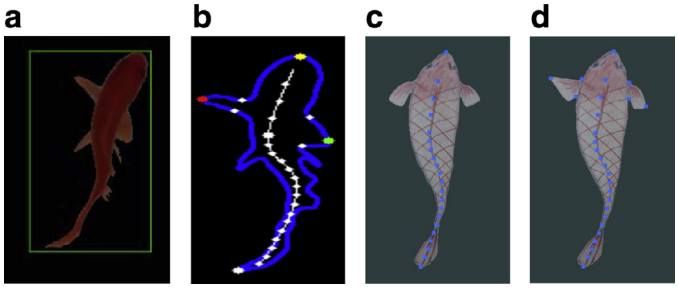

Fig. 5. Our two-level motion retargeting scheme. (a) The tracked fish. (b) Extracted contour and feature points. (c) The deformed implicit model with body control points. (d) The deformed explicit model with all control points.

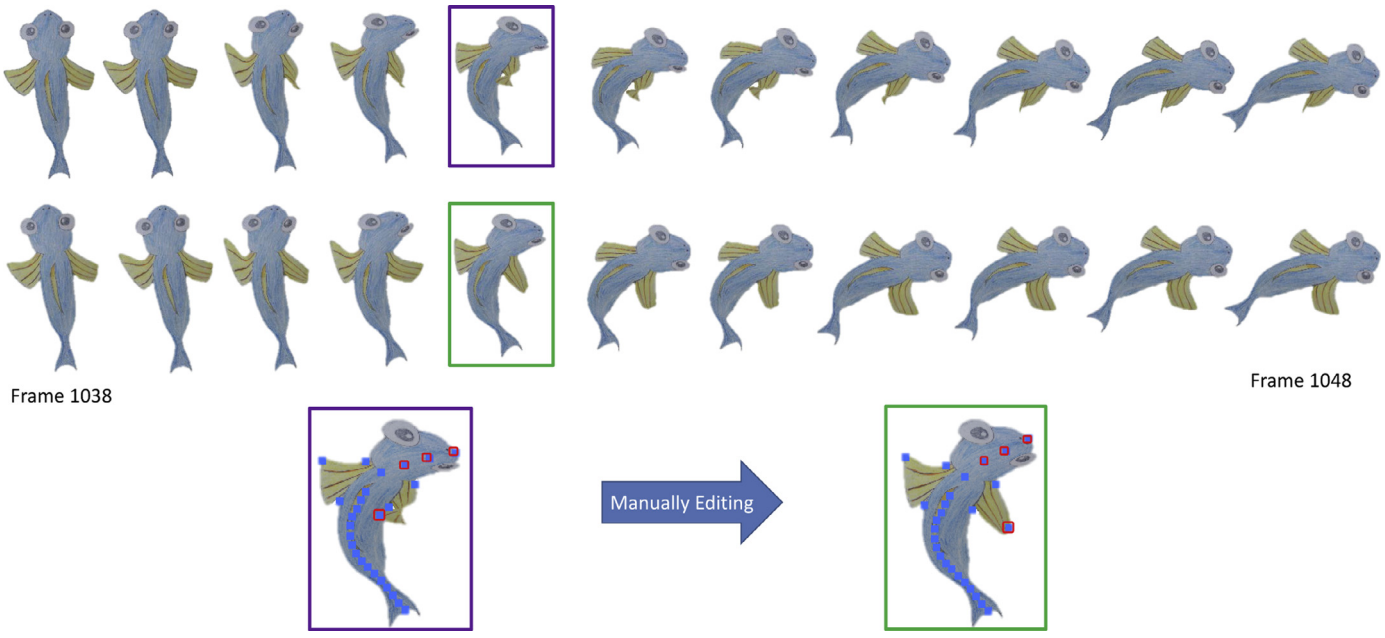

Fig. 6. Motion fine tuning process. The top row contains 11 clipped frames from a sequence window of retargeted animation. The second row shows the same animation frames after motion fine tuning. The bottom row illustrates the specific four control points edited in fine tuning.

- Naturally, if the editing operation tends to adjust the extent of the original motion(e.g., enlarge or narrow the bend angle of the fish tail), the result animation should be the average of the edited value and the original motion. Otherwise, if the editing operation tends to reverse the original motion (e.g., change the bend direction of the fish tail to the opposite side), the result animation should be set by the edited operation directly, as the original motion is incorrect.

6.1. Formalization of motion fine tuning

Since we regard the intermediate motion M as a motion parameter to transfer between feature points of the actual fish and control points of the fish model, we simply save M for each video frame to a data file. For a fish video containing t valid frames, its corresponding motion data can be represented by a motion sequence:

$$\mathbf{M} = \{M_i : 1 \le i \le t\}, \tag{16}$$

where M_i is the motion of the ith frame and defined as same as Eq. (14). To record the modified motion, we construct a new motion sequence

$$\mathbf{N} = \{N_i : 1 \le i \le t\} \tag{17}$$

and initialize it with the original motion sequence **M**.

Assume that an editing operation occurs at frame r on a certain property $M_r.u$. M' denotes the edited motion. Then we update **N** as

$$N_i.u = I(1 - f(i; r, p, q)) * M_i.u + f(i; r, p, q) * M'.u,\ p \le i \le q \tag{18}$$

where I is a symbol variable to indicate whether this editing is an extent adjustment ($I = 1$) or a motion pattern reversal ($I = 0$), $f(x)$ is the unimodal influence function, and p and q are the lower and upper bound of the motion sequence window which $M_r.u$ belongs to.

6.2. Motion sequence window identification

For the rest pose of a fish model, we can acquire an intermediate motion M_0. For $M_r.u$, we define its motion sequence window with a lower bound p and an upper bound q as

$$p = min(i), \forall i \le k \le r, (M_k.u - M_0.u)(M_r.u - M_0.u) \ge 0 \tag{19}$$

$$q = max(j), \forall r \le k \le j, (M_k.u - M_0.u)(M_r.u - M_0.u) \ge 0 \tag{20}$$

To ensure the real-time fine tuning, we support editing all properties of an intermediate motion except **center** and **dir**, due to the computation cost.

6.3. Motion effect range function

The motion effect range function can be defined as unimodal distribution function. Here we use a quadratic form:

$$f(i; r, p, q) = \begin{cases} 1 - \dfrac{(r-i)^2}{(r-p)^2}, & i \le r \\ 1 - \dfrac{(r-i)^2}{(r-q)^2}, & i > r \end{cases}. \tag{21}$$

With this function, we can achieve the unimodal influence for the motion editing.

We choose a sequence of consecutive retargeted animation frames which includes several flawed frames. In general, the distorted frames may come from the wrong location of feature points during motion capture due to the occlusion of the fish, or the distortion of the deformation result. In the frame sequence we chose, the distortion is produced because the target fish has pectoral fins with a considerate large area rather than small fin tips. The distortion is inevitable with the current deformation algorithm when the fin angle is too extreme.

We edit one frame among the segment, and the motion fine tuning tool can apply the editing operation to its neighboring frames. As shown in Fig. 6, the top row shows the original animation frames from frame 1038 to frame 1048 in the original animation sequence. The area around the fish head and the right fin is distorted and need to be refined. We move four control points

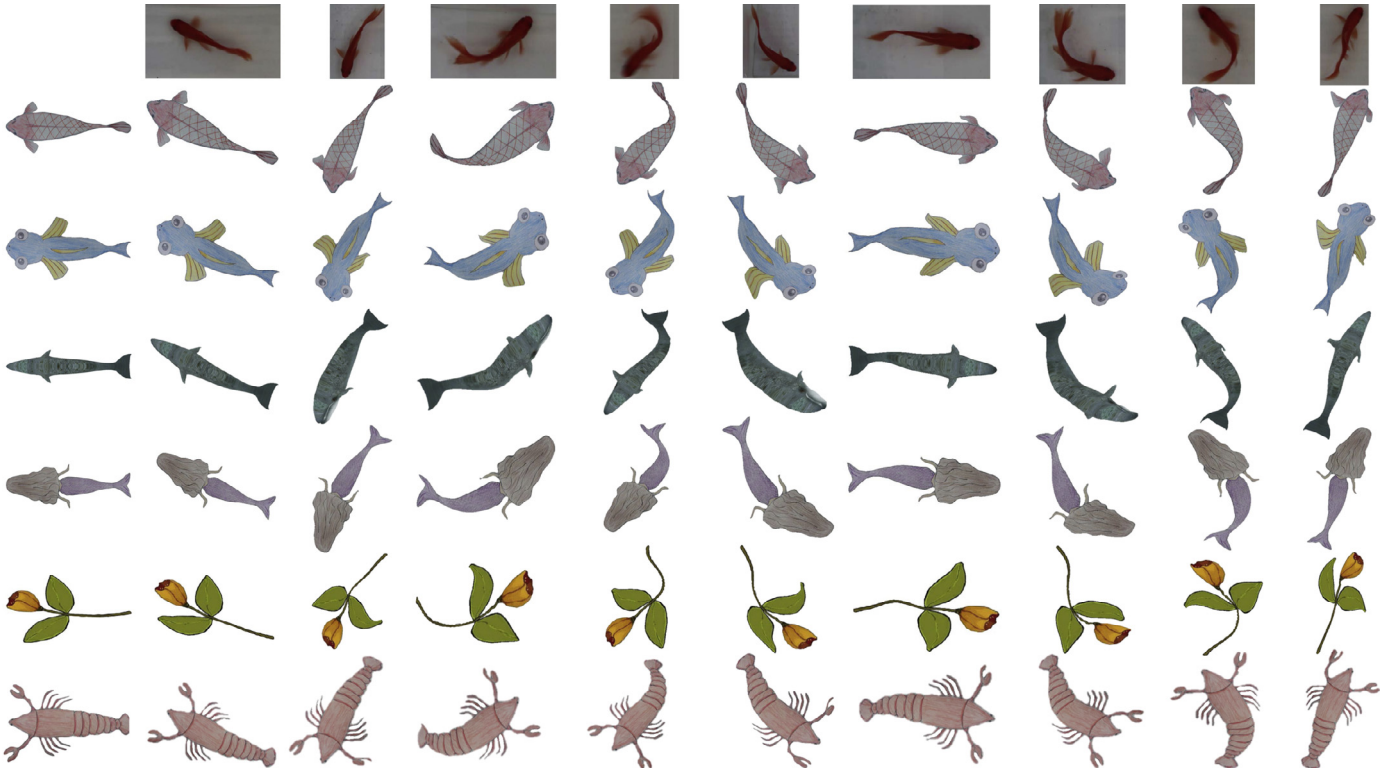

Fig. 7. Sample frames in our experiments. The top row contains nine clipped frames from the video. The remaining five rows are the corresponding animation frames of the target models. The target models from top to bottom are carp, cartoon fish, 3D whale, mermaid, flower, and lobster. The first column on the left shows the rest poses of the target models.

Table 2
The motion fine tuning parameters.

u	I	r	p	q	$M_0.u$	$M_r.u$	$M'.u$
$up\_axis\_angle_2$	1	1042	1040	1049	1.6589	0.9704	1.1062
$up\_axis\_angle_3$	1	1042	1039	1050	1.5965	0.7586	1.0261
$head\_angle$	1	1042	1039	1052	1.4987	0.5493	0.8666
r_fin_angle	1	1042	1040	1132	5.5535	4.1450	5.2623

on frame 1042 to make this frame expended and smooth. Then the motion fine tuning can effect to its neighboring frames automatically. The second row shows the result animation frames in the same frame number as the top row. The only editing occurs on the frame 1402, and the moved control points are marked in the third row. Table 2 shows the detail parameter of Eq. (18). Note that all angle values are represented by radian in Table 2.

7. Experimental results

We have implemented the system using C++ and OpenCV, and all the experiments are run on an I7-3970x CPU (3.5 GHz) with 16 GB RAM. A video of a goldfish is captured by a monocular camera with a resolution of 1920x1088 pixels yet we resize it to 960x544 pixels to process. For 2D animation, we use a carp and a cartoon fish as the target models. For 3D animation, we use a whale as the target model. We also deliver a hand-drawn mermaid and a flower picture to the system, to test the motion retargeting effect of our technique for other fish-like characters.

Fig. 7 shows the results. The performance statistics are listed in Table 3. The memory consumption is in the range of 130 MB to 200 MB proportional to the number of vertices. As for the parameter setting, we set u_num being 3, l_num being 15, and S being 5 to describe the flexible body of a fish. One exceptional case occurs when the flower is treated as the target model. l_num is set to be 25 and S is set to be 8 to keep more information of the lower body of the fish to make the deformation of the flower stem more natural.

The current implementation is not time optimized and the entire pipeline can operate at around 20 FPS. We also compare our technique with others' work. The only research we have found in data-driven fish animation is presented by Lee et al. [13].

Table 3
The experimental performance statistics.

Target model	# Vertices	# Triangles	Tracking			Motion retargeting (ms)	Rendering (ms)	Total time cost (ms)
			Background subtraction	Filter	Feature extraction			
Carp	2445	3827	12.48 ms	9.11 ms	6.34 ms	1.11	10.35	39.39
Cartoon fish	2667	4136				1.23	10.95	40.11
3D Whale	6003	9200				7.78	10.96	46.67
Mermaid	3033	4710				1.45	11.06	40.44
Flower	3492	5431				2.08	10.81	40.82
Lobster	3924	5829				2.02	11.03	40.98

Table 4
Time costs for one-frame processing.

Method	[8]	[9]	[10]	Ours
Camera number	4	3	4	1
Image size	320x240	320x240	250x250	960x544
Average time (ms)	236.5	33	33	41.40

However, they have not shown any convincing figures or tables. Other similar work such as [40,41] concentrates on detection and trajectory tracking which is similar to the global motion estimation task under our framework (their methods can handle multiple fishes yet). Since we hardly find other valid researches for motion capture for fish, we compare our method with several video-based mo-cap systems for human [8–10]. The comparison result is shown in Table. 4 which is made with statistics published by peers. Given the image size and camera number, our time cost is competitive among these video-based mo-cap systems.

8. Conclusion and discussion

In this paper, we have presented a novel framework of creating fish animation by motion capture, retargeting, and fine tuning. The fish motion can be captured by a monocular camera without any marker. We employ elliptical Fourier coefficients to analyze the swimming patterns of a fish. The fish motion is represented in a robust way, with smooth medial axis serving as the descriptor for the soft fish body. A two-level motion retargeting scheme is proposed to properly transfer the captured motion into fish-like models. The system can then drive a static pictured fish to swim with the same motion style of the actual fish in real time. We also provide an interactive tool for motion fine tuning, which is able to edit retargeted frames with high efficiency. Besides, 3D models and fish-like characters can also be animated vividly in this framework.

The proposed method for medial axis extraction can be extended to describe the shape of flexible objects. Instead of the spatial filter, the elliptical Fourier coefficients technology allows users to select the number of remaining harmonics directly according to the amount of information they want to keep, without regard to the noise or detail carried by the original contour. Moreover, once we reconstruct a contour from the frequency domain, the contour itself can be represented by sums of elliptical harmonics, thus is not only continuous but even derivable. This property benefits us a lot when we want to find out more useful feature of the object, for example, finding correspondences across various contours using energy models in differential domain [42], or executing manifold learning on a set of contours to learn a certain kind of object's shape and probability density [43].

The two-level motion retargeting scheme can be considered as a multi-scale skeleton model together with point-based deformation. The fish model is represented by a two-level skeleton (body and fins), and correspondingly, the local motion retargeting is achieved by two steps sequentially. Such a multi-scale scheme greatly enlarges the range of retargeting models – making the motion retargeting process much more easily accessible between models that have similar structures at the corresponding scales.

Nevertheless, our technique still suffers from several limitations. First, our technique can only capture the fish motion from the top view, thus loses the fish's visual information from the side view. Second, the simplified fish model we employ limits the generalization of our system. For example, we cannot model the complex fin motion of a lionfish, or the body motion of a porcupine fish. Finally, the fish needs to move properly in a small aquarium to be tracked.

Our ultimate motivation in this paper is to showcase a cheap yet effective way for the common users to create data-driven fish animation with a low-end mobile device. In the near future, we plan to transplant this system to a cellphone platform, so that any cellphone user could capture a swimming fish in a pool or aquarium and generate a fish character animation in an augmented-reality (AR) environment. The motion fine tuning tool will be extended to edit the depth information of the animation frames when a 3D fish model is imported. To extend our system comprehensively to 3D animation, we can utilize depth estimate technique to fetch the depth information from the video, and 3D deformation algorithm such as [44] to achieve the retargeted animation results.

Acknowledgments

This research is supported by National Natural Science Foundation of China (Nos. 61402025, 61672149, 61532002 and IIS-1715985).

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.cag.2017.12.004.

References

[1] Barca JC, Rumantir G. A modified k-means algorithm for noise reduction in optical motion capture data. In: Proceedings of the international conference on computer and information science; 2007. p. 118–22.
[2] Corazza S, Mndermann L, Chaudhari AM, Demattio T, Cobelli C, Andriacchi TP. A markerless motion capture system to study musculoskeletal biomechanics: visual hull and simulated annealing approach. Ann Biomed Eng 2006;34(6):1019–29.
[3] Oshita M. Motion-capture-based avatar control framework in third-person view virtual environments. In: Proceedings of the ACM sigchi international conference on advances in computer entertainment technology; 2006. p. 2.
[4] Kuhl FP, Giardina CR. Elliptic fourier features of a closed contour. Comput Graph Image Process 1982;18(3):236–58.
[5] Gall J, Stoll C, de Aguiar E, Theobalt C, Rosenhahn B, Seidel HP. Motion capture using joint skeleton tracking and surface estimation. In: Proceedings of the computer vision and pattern recognition. IEEE; 2009. p. 1746–53. doi:10.1109/CVPR.2009.5206755.
[6] Vondrak M, Sigal L, Hodgins J, Jenkins O. Video-based 3D motion capture through biped control. ACM Trans Graph 2012;31(4). doi:10.1145/2185520.2185523.
[7] Urtasun R, Fleet D, Fua P. Gaussian process dynamical models for 3D people tracking. In: Proceedings of the computer vision and pattern recognition; 2006. p. 238–45.
[8] Theobalt C, Magnor MA, Schüler P, Seidel H-P. Combining 2D feature tracking and volume reconstruction for online video-based human motion capture. Int J Image Graph 2004;4(04):563–83.
[9] Michoud B, Guillou E, Bouakaz S. Real-time and markerless 3D human motion capture using multiple views. In: Proceedings of the human motion: understanding, modeling, capture and animation; 2007. p. 88–103.
[10] Shafaei A, Little JJ. Real-time human motion capture with multiple depth cameras. In: Proceedings of the computer and robot vision; 2016. p. 24–31.
[11] Ju E, Won J, Lee J, Choi B, Noh J, Choi MG. Data-driven control of flapping flight. ACM Trans Graph 2013;32(5):151.
[12] Wilhelms J, Gelder AV. Interactive video-based motion capture for character animation. In: Proceedings of the IASTED computer graphics and imaging conference; 2002.
[13] Lee C-N, Hsieh W-C, Zhang-Jian D-J, Yang Y. 3D fish animation with visual learning ability. In: Proceedings of the Asia-Pacific signal and information processing association (APSIPA). IEEE; 2014. p. 1–4. doi:10.1109/APSIPA.2014.7041578.
[14] Yu Q, Terzopoulos D. Synthetic motion capture: implementing an interactive virtual marine world. Vis Comput 1999;15(7):377–94.
[15] Shi J, Tomasi C. Good features to track. In: Proceedings of the computer vision and pattern recognition; 1994. p. 593–600. doi:10.1109/CVPR.1994.323794.
[16] Comaniciu D, Ramesh V, Meer P. Real-Time Tracking of Non-Rigid Objects Using Mean Shift. In: 2000 Conference on Computer Vision and Pattern Recognition (CVPR 2000), 13-15 June 2000, Hilton Head, SC, USA; 2000. p. 2142. doi:10.1109/CVPR.2000.854761.
[17] Gleicher M. Retargetting motion to new characters. In: Proceedings of the computer graphics and interactive techniques. ACM; 1998. p. 33–42. doi:10.1145/280814.280820.
[18] Lee J. A hierarchical approach to interactive motion editing for human-like figures. In: Proceedings of the computer graphics and interactive techniques. ACM; 1999. p. 39–48. doi:10.1145/311535.311539.
[19] Popović Z, Witkin A. Physically based motion transformation. In: Proceedings of the computer graphics and interactive techniques. ACM; 1999. p. 11–20. doi:10.1145/311535.311536.

[20] Shin HJ, Lee J, Shin SY, Gleicher M. Computer puppetry: an importance-based approach. ACM Trans. Graph. 2001;20(2):67–94. doi:10.1145/502122.502123.
[21] Kulpa R, Multon F, Arnaldi B. Morphology-independent representation of motions for interactive human-like animation. Comput Graph Forum 2005;24(3):343–51. doi:10.1111/j.1467-8659.2005.00859.x.
[22] Bregler C, Loeb L, Chuang E, Deshpande H. Turning to the masters: Motion capturing cartoons. ACM Trans Graph 2002;21(3):399–407. doi:10.1145/566654.566595.
[23] Hornung A, Dekkers E, Kobbelt L. Character animation from 2D pictures and 3D motion data. ACM Trans Graph 2007;26(1):1. doi:10.1145/1189762.1189763.
[24] Karpenko OA, Hughes JF. Smoothsketch: 3d free-form shapes from complex sketches. ACM Trans Graph 2006;25(3):589–98. doi:10.1145/1141911.1141928.
[25] Igarashi T, Matsuoka S, Tanaka H. Teddy: A sketching interface for 3d freeform design. In: Proceedings of the 26th annual conference on computer graphics and interactive techniques, SIGGRAPH'99. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.; 1999. p. 409–16. doi:10.1145/311535.311602.
[26] Wei XK, Chai J. Intuitive interactive human-character posing with millions of example poses. IEEE Comput Graph Appl 2011;31(4):78–88. doi:10.1109/MCG.2009.132.
[27] Choi MG, Yang K, Igarashi T, Mitani J, Lee J. Retrieval and visualization of human motion data via stick figures. Comput Graph Forum 2012;31(7pt1):2057–65. doi:10.1111/j.1467-8659.2012.03198.x.
[28] Guay M, Cani M-P, Ronfard R. The line of action: an intuitive interface for expressive character posing. ACM Trans Graph 2013;32(6). doi:10.1145/2508363.2508397. 205:1–205:8.
[29] Barnich O, Van DM. Vibe: A universal background subtraction algorithm for video sequences. IEEE Trans Image Process 2011;20(6):1709–24. doi:10.1109/TIP.2010.2101613.
[30] Shewchuk JR. Delaunay refinement algorithms for triangular mesh generation. Comput Geom 2002;22(1–3):21–74. https://doi.org/10.1016/S0925-7721(01)00047-5.
[31] Schaefer S, Mcphail T, Warren J. Image deformation using moving least squares. ACM Trans Graph 2006;25(3):533–40. doi:10.1145/1141911.1141920.
[32] Otsu N. A threshold selection method from gray-level histograms. IEEE Trans Syst Man Cybern 1979;9(1):62–6.
[33] Foskey M, Lin MC, Manocha D. Efficient computation of a simplified medial axis. In: Proceedings of the eighth ACM symposium on solid modeling and applications, SM '03. New York, NY, USA: ACM; 2003. p. 96–107. doi:10.1145/781606.781623.
[34] Dey TK, Zhao W. Approximating the medial axis from the voronoi diagram with a convergence guarantee. Algorithmica 2004;38(1):179–200. doi:10.1007/s00453-003-1049-y.
[35] Du H, Qin H. Medial axis extraction and shape manipulation of solid objects using parabolic pdes. Proceedings of the ACM symposium on solid modeling and applications; 2004. p. 25–35.
[36] Chaussard J, Couprie M, Talbot H. A discrete λ-medial axis. In: Proceedings of the 15th IAPR international conference on discrete geometry for computer imagery, DGCI'09. Berlin, Heidelberg: Springer-Verlag; 2009. p. 421–33. http://dl.acm.org/citation.cfm?id=1813270.1813312.
[37] Faraj N, Thiery J-M, Boubekeur T. Progressive medial axis filtration. In: Proceedings of the SIGGRAPH Asia 2013 technical briefs, SA '13. New York, NY, USA: ACM; 2013. 3:1–3:4. http://doi.acm.org/10.1145/2542355.2542359.
[38] Igarashi T, Moscovich T, Hughes JF. As-rigid-as-possible shape manipulation. ACM Trans Graph 2005;24(3):1134–41. doi:10.1145/1073204.1073323.
[39] Weng Y, Xu W, Wu Y, Zhou K, Guo B. 2D shape deformation using nonlinear least squares optimization. Vis Comput 2006;22(9):653–60. doi:10.1007/s00371-006-0054-y.
[40] Chuang MC, Hwang JN, Ye JH, Huang SC, Williams K. Underwater fish tracking for moving cameras based on deformable multiple kernels. IEEE Trans Syst Man Cybern Syst 2016;PP(99):1–11. doi:10.1109/TSMC.2016.2523943.
[41] Chuang MC, Hwang JN, Williams K, Towler R. Tracking live fish from low-contrast and low-frame-rate stereo videos. IEEE Trans Circuits Syst Video Technol 2015;25(1):167–79. doi:10.1109/TCSVT.2014.2357093.
[42] Campbell NDF, Kautz J. Learning a manifold of fonts. ACM Trans Graph 2014;33(4). doi:10.1145/2601097.2601212. 91:1–91:11.
[43] Turmukhambetov D, Campbell NDF, Dan BG, Kautz J. Interactive sketch-driven image synthesis. Comput Graph Forum 2015;34(8):130–42. doi:10.1111/cgf.12665.
[44] Lan L, Yao J, Huang P, Guo X. Medial-axis-driven shape deformation with volume preservation. Vis Comput 2017;33(6–8):789–800. doi:10.1007/s00371-017-1401-x.