ISSN 2464-4617 (print)
ISSN 2464-4625 (CD)
Computer Science Research Notes
CSRN 2802
Short Papers Proceedings
http://www.WSCG.eu

Fish Motion Capture with Refraction Synthesis

Klaus Müller, Jan-Marco Hütwohl and Klaus-Dieter Kuhnert
Institute of Real-Time Learning Systems
Department of Electrical Engineering & Computer Science, University of Siegen, Germany
klaus.mueller@uni-siegen.de

Stefanie Gierszewski and Klaudia Witte
Research Group of Ecology and Behavioral Biology
Institute of Biology, University of Siegen, Germany
gierszewski@chemie-bio.uni-siegen.de

ABSTRACT

3D fish animations become more and more popular in fish behavioral research. It empowers the experimenter to design fish stimuli and their specific behavior to the experiment's needs. The fish animation can be done manually or derived from video footage. Especially automatic fish model parameter recovery for 3D animations is not well studied yet. Here we present a novel, flexible method for this purpose. It can be used to recover position, pose, bone rotation and size from single or multiple view and for single or multiple fish. Additionally we implement a novel method to compensate the fish tank's refraction effect and show that this method can decrease the error up to 80 %. We successfully applied the proposed method to two different data sets and recovered fish parameters out of single- and double-view video stream. A video attached to this paper demonstrates the results.

Keywords

motion capture, pose recovery, analysis-by-synthesis, refraction compensation, fish tracking

1 INTRODUCTION

The use of virtual 3D fish stimuli is the current trend in fish behavior research and partly replace the use of fish video or live stimulus fish partly [WGCT17]. In such kind of experiments screens with different 3D fish animations are placed next to a fish tank. Each animation shows different fish (one or several) with different appearance (e.g. skin texture or coloration), size, and morphology or behavior pattern. Inside the real fish tank are one or several test fish, which show their interest to a stimulus by physical presence in front of the corresponding screen. In order to create stimulus animations some open source software tools e.g. *FishSim Animation Toolchain*[1] or *AnyFish*[2] came into the market and help inexperienced users to create photorealistic 3D fish models and animations. The animation part of the stimulus is mostly done manually (or in case of [MSH+17] semi-automatic) since these tools do not provide methods to derive actions and behavioral patterns automatically from video footage.

In this paper we present a novel method which automatically recovers 3D fish model parameters like position, orientation and joint configuration out of single or multiple view video footage by using a model-based analysis-by-synthesis approach [Pop07]. This method was originally applied for pose recovery and tracking of humans [PMBH+10] or for human pose recovery out of a single image [KKTM15]. Especially for the task presented here this method is very promising for two main reasons: first, the time-consuming process of model-creation, which is needed for such a method, can be omitted, since the 3D fish model is already available. Second, fish have a very simple kinematic bone structure, which minimize the risk of misconvergence, what in general can happen by using this method. Here we extended this method by refraction synthesizing, which appears at the air-water border of the fish tank. Additionally we add an occlusion handling for fish. The method is based on single or multiple view silhouettes of live fish, which are approximated by view-depended artificial silhouettes, extracted out of the provided 3D fish model. For approximation we employed a least-squares method. We finally validated the presented method with video footage of single camera and dual camera, showing a single fish or a pair of fish. We annotated a video sequence of 1000 frames manually and compare this dataset to the result of the proposed algorithm (with and without refraction compensation, single and multiple view). It could be shown that the method recovered fish position and pose very precisely. Especially the refraction compensation improved the position recovery significantly. A video showing the results of the method is attached to this paper. In summary, the

[1] https://bitbucket.org/EZLS/fish_animation_toolchain for more information see [MSH+17] and [GMS+17] for its validation

[2] https://github.com/anyFish-Editor/anyFish-2.0 for more information see [VIC+13, IAW+15]

work presented in this paper solves common problems in research on fish behavior and contributes the following features:

- precise transfer of fish movement patterns from video footage to a photo-realistic 3D fish model (as used in skeletal animations) precisely
- high position precision based on refraction compensation by synthesis during optimization
- very flexible: single or multiple view camera setup, single or multiple fish, fast (without refraction compensation) or precise

The paper is divided into six chapters. In chapter 2 we present related work. This is followed by chapter 3 giving information regarding preliminaries. In Chapter 4 the method is described in detail. We present in chapter 5 the results of the method which are finally discussed in chapter 6.

2 RELATED WORK

Most motion capture research has been done and is still going on in the field of human motion capture. There are several different methods available: on the one hand wearable motion trackers are used, which track the position and rotation of single joints (head, arms, legs etc.) (see e.g. [RLS09]). On the other hand there are optical motion capture methods. These are divided in system which use markers mounted to the human body and markerless systems, which use single or multiple RGB-cameras (e.g. [ST02, PMBH+10]) or RGB-depth cameras (see e.g. [SSK+13]). Nowadays there are also deep learning methods used for pose recovery like presented in [CSWS17] or in [WRKS16]. In contrast to human motion capture, fish pose recovery leads to special challenges: firstly, the use of motion trackers or markers for visual pose recovery is very difficult, since wearable motion trackers are not available for small fish or markers can only be fixed under great difficulties to fish. Secondly, RGB-D cameras (e.g. Microsoft Kinect), which pushed the human pose recovery research forward massively, can only be used very limited: such cameras mostly use active light, which brings difficulties regarding reflection and refraction while light travels through different media (e.g. water and air). Due to these facts multiple view camera setups are the most used configuration for 3D tracking and pose recovering of fish. Besides some research in field of fish position tracking in 2D and 3D (for a review see [DDYP13]), there is little research in fish pose-recovery. Takahashi et al. introduce a method to extract fish position and posture from orthogonal video footage. They used a simple 3D fish model, which was projected to the real images. With the help of a brute force, box constrained search algorithm they estimated the model-parameters in a way, that the projection fits best to the recorded fish images. They finally used the gathered motion data to estimate a locomotion model of the fish [THHN00]. Butail and Paley estimated 3D position and shape to analyse fish schooling kinematics [BP10]. They modelled the fish shape as bendable ellipsoid. Based on this model fish pose, position and bending is estimated out of 2D silhouettes with the help of a particle filter. Later on they improved this method and extended the used 3D-model [BP12]. In contrast to the former model, the newer model consisted of estimated cross-sectional ellipses, which were ordered along a three-dimensional midline, describing the bending of the fish body more precisely. They used simulated annealing to match 2D silhouettes to the model and to find the best model parameter set. The cost function is based on the sum of distances between occluding contour points and the model surface. Voesenek et al. used a similar but more precise model with more degrees of freedom regarding fish bending and rotation [VPvL16]. They also approximated 2D silhouettes to a 3D-model, which consists of merged ellipsoids along the longitudinal axis. For finding the optimal model-parameters they re-projected the model to the virtual cameras and calculated a scalar value describing the overlap and used a downhill simplex algorithm for optimization. Besides extraction of fish motion they used the system to derive resultant forces and torques of fish during swimming.

In contrast to the former work, the proposed method differs in the following:

- motion capture for 3D fish animation: this method uses a 3D fish animation model with bones to recover position, pose and bending. The resulting parameter set can directly be used to animate 3D-models
- the proposed method synthesizes the refraction caused by the air-water border
- we use a non-linear least-squares method to approximate the fish position, pose and bending, which uses all silhouette pixels separately for optimization
- the method is very flexible and can be used for single or multiple fish, for single- or multiple camera setups, precise (with refraction compensation) or fast (without refraction compensation)

3 PRELIMINARIES

3.1 Calibration

Since our method is specialized for fish in aquaria, we use an easy and precise calibration method, which was especially developed for this purpose (see [MSKK14]). The method assumes, that camera position and alignment are static in relation to the aquarium. Based on

 ISBN 978-80-86943-41-1

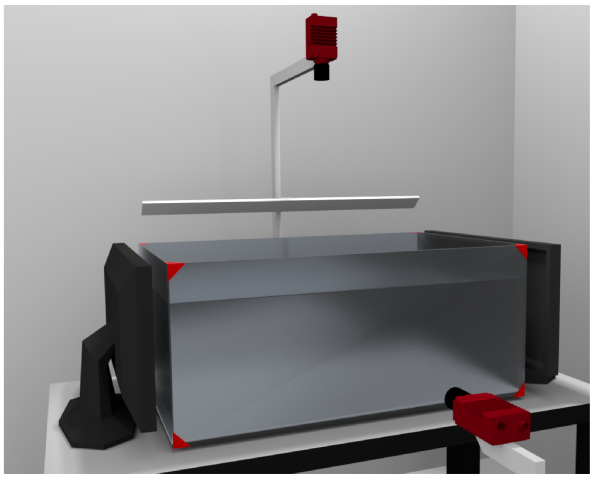

Figure 1: Setup with two cameras. The red corners at the tank are used for automatic-calibration of extrinsic camera-parameters and position and normal calculation of the fish-tank sides.

markers mounted to the corners of the fish-tank, an optimization algorithm estimates the camera parameters with respect to the aquarium. Besides the extrinsic camera-parameter estimation the method also estimates the normals and positions of the tank windows automatically and the normal and the position of the water surface semi-automatically. This information is used for the refraction calculations afterwards.

3.2 Contour retrieving from video

Since the proposed method is specialized for aquaria, we consider scene and cameras as static. This provides the opportunity to use classical background subtraction methods to divide the camera image in background (tank) and foreground (fish). Based on this binary image we extract the silhouette S of all foreground objects. In case of using multiple cameras fish silhouettes originated by mirroring in the tank's glass walls can be detected and eliminated by using epipolar geometry constraint. We also use this constraint to assign silhouettes of fish in multiple views (see also [MSKK14]).

3.3 3D-model

The pose recovery result strongly depends on the 3D fish-model quality: the higher the shape similarity between the 3D-model and the live fish is, the better the model can be approximated. Depending on the fish species, it can be enough to use two different models for male and female, like done in the validation presented here with sailfin mollies (*Poecilia latipinna*). In case of fish species varying strongly within sex it could be necessary to use several different models for female and male, mapping its variation. In general, the fish model has to be implemented a as 3D-mesh. Even thin parts of the fish like fins have to be designed as 3D object in order to retrieve the contour by the proposed algorithm presented in section 4.1. Since the proposed method is based on fish silhouettes, texture and reflection properties of the model can be ignored. In order to deform the fish model, all parts of the mesh whose position and rotation should be recovered by the algorithm have to be connected to the skeleton bones. In the field of computer animation, this type of model presentation and animation is called *skeletal animation*. For the validation presented in this paper we developed fish models with the help of the free available fish designer included in *FishSim Animation Toolchain* [3]. At the moment the toolchain includes five different fish species (the sailfin molly *Poecilia latipinna*, the Atlantic molly *Poecilia mexicana*, the guppy *Poecilia reticulata*, the three-spined stickleback *Gasterosteus aculeatus*, and a chichlid *Haplochromis* spp.). In general it is possible to add new fish species to the toolchain. More information on the toolchain can be found in [MSH+17] and see also [GMS+17] for a validation of the generated fish stimuli in research.

4 POSITION, POSE AND SKELETON RECOVERY

The proposed method aims to approximate the former described 3D-model to the live fish in tank. The approximation is based on silhouettes of live fish, captured from single or multiple cameras. The silhouette of live fish is extracted as described in section 3.2 and is compared to the artificial contour of the 3D-model. In order to create as similar artificial silhouettes as possible it is necessary to adjust and position the virtual cameras exactly like the live ones. In order to do so, we use the estimated calibration information as described in section 3.1. Besides the camera parameters the dimensions of the fish are necessary to approximate the model parameters. Therefore we also implement a method which estimates fish size in a preprocessing step from single- or multiple view video sequences (see section 4.6). With the help of the in 4.1 described method the silhouettes are extracted from the 3D-model and are compared to real silhouettes by an error function. An optimization algorithm, described in section 4.4, optimizes the model parameters in order to minimize the error function (see section 4.3). To lower the risk of converging to a local minimum of the error function, we apply an initialization method described in section 4.5 at the beginning of a video sequence. Since the air-water boarder of the fish tank causes refraction effects, we implement a compensation method in order to increase the accuracy (see section 4.2). The whole workflow is shown figure 2.

[3] `https://bitbucket.org/EZLS/fish_animation_toolchain`

ISBN 978-80-86943-41-1

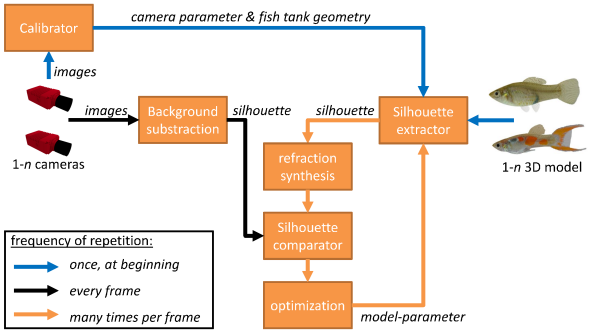

Figure 2: Schematic of the fish recovering system.

4.1 2D silhouette extraction from 3D-model

There are several different ways to extract 2D silhouettes from a 3D-model: one obvious possibility is the use of a render system as included in *FishSim Animation Toolchain*. The silhouette is rendered to a homogeneous surface and can easily be extracted by thresholding. In practice, it turned out that the rasterization which is used by the render system to draw primitives to a pixel-based device or image, is too imprecise since its resolution is limited to the pixel resolution of the underlying device. Especially during the optimization process this fact causes that the algorithm does not converge correctly. For that reason we apply the classical method leaned on [BS00] to retrieve silhouettes from polygonal mesh structures. This method is based on the principle that a silhouette edge will appear if one front-face (front side of mesh-triangle) of two neighboring mesh triangles is visible and the other one is invisible for the camera. In order to find silhouette edges we traverse all k polygon mesh triangles and check if the normal $\vec{n}_k$ of triangle $\triangle ABC_k$ is pointing in the same direction as the viewing vector $\vec{v}$ of the camera and store the result in a new vector $\vec{f}$:

$$f_i = \begin{cases} 1 & \text{if } \vec{v} \circ \vec{n}_i \geq 0 \quad i \in k \\ 0 & \text{if } \vec{v} \circ \vec{n}_i < 0 \quad i \in k \end{cases} \tag{1}$$

In case that f_i and f_j of neighboring triangles $\triangle ABC_i$ and $\triangle ABC_j$ are different the shared edge of both triangles is part of the silhouette. The gathered 3D silhouette points are converted with the help of the camera projection matrix to a 2D pixel coordinates $\tilde{S}$. Depending on the 3D-mesh resolution and the camera distance to the object, it can happen that the distance between two neighboring silhouette pixels is several pixel units. Especially for the comparator algorithm (see 4.3) that searches nearest neighbor pixels between real and virtual silhouette this could cause bigger errors. For that reason, we interpolate silhouette pixels in case that the distance is bigger than one pixel unit between neighboring silhouette pixels. Finally, this method extracts a vector of silhouette pixels $\tilde{S}_X$ out of the 3D-mesh model $M(X)$ which is based on the model parameters X.

4.2 Refraction compensation

Since the proposed method is specialized for pose recovery of fish in aquaria we also address the problem of refraction. Especially for stereo or multiple view camera setups refraction can cause errors of several centimeters [MSKK14] and hinder the mapping of silhouettes from different views. In general, we calculate the pixel ray refraction with the help of Snell's law, which calculates the refraction angle of the ray with the help of media's refractive indexes (n_1, n_2) and the incident angle (angle between surface and ray).

$$\frac{n_1}{n_2} = \frac{sin(\beta)}{sin(\alpha)} \tag{2}$$

In contrast to a ray-tracing approach, in which the optical path of each pixel ray can be calculated step-by-step, we have to go the way around: we start at the 3D coordinate of the silhouette edge and have to find the intersection point with the refraction plane (fish tank plane) in order to calculate the refraction angle and finally the 2D pixel coordinate. At first an additional plane P is calculated, which is defined by the normal vector $\vec{n}$ of the intersection plane I (aquarium window or water surface defined during calibration), the camera position vector $\vec{c}$ and the position vector of a silhouette point $\vec{s}_i$ ($i \in$ all 3D silhouette points). Next, the line of intersection $g(x) : \vec{x} = \vec{o} + x\vec{r}$ between intersection plane I and plane P is calculated. There are several algorithms available to calculate the line of intersection between two planes and therefore it is here not discussed further. The intersection point of the pixel ray lays on this line. With the help of the law of sines we can calculate α and β (see figure 3):

$$sin(\alpha) = \frac{\|(\vec{x} - \vec{c}) \times \vec{n}\|}{\|(\vec{x} - \vec{c})\| \cdot \|\vec{n}\|} \tag{3}$$

$$sin(\beta) = \frac{\|(\vec{x} - \vec{s}_i) \times \vec{n}\|}{\|(\vec{x} - \vec{s}_i)\| \cdot \|\vec{n}\|} \tag{4}$$

In order to find the final intersection point on the line g, we combine the equations (2), (3) and (4) and minimize it:

$$\min_{x \in \mathbb{R}} \frac{\|(g(x) - \vec{c}) \times \vec{n}\|}{\|(g(x) - \vec{c})\| \cdot \|\vec{n}\|} \frac{\|(g(x) - \vec{s}_i)\| \cdot \|\vec{n}\|}{\|(g(x) - \vec{s}_i) \times \vec{n}\|} - \frac{n_1}{n_2} \tag{5}$$

In order to initialize the minimization well, the initial value x_i is defined by calculating the intersection point D (position vector $\vec{d}$) between $\overline{S_iC}$ and intersection plane I. D also lays on g and the factor x_i is calculated as follows:

$$x_i = \begin{cases} \frac{o_x - d_x}{r_x} & \text{if } r_x \neq 0 \text{ else} \\ \frac{o_y - d_y}{r_y} & \text{if } r_y \neq 0 \text{ else} \\ \frac{o_z - d_z}{r_z} \end{cases} \qquad \text{with } [\vec{d}, \vec{o}, \vec{r}] = \begin{pmatrix} [d,o,r]_x \\ [d,o,r]_y \\ [d,o,r]_z \end{pmatrix} \tag{6}$$

ISBN 978-80-86943-41-1

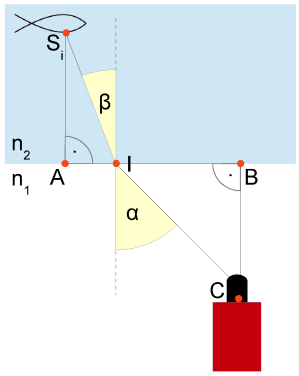

Figure 3: Refraction. A, B and I lie on the intersection plane, which is also the separation plane between air and water. A ray starts at the fish's surface S, intersects in I and gets refracted. Finally it hits the camera center C.

4.3 Contour comparator

To approximate pose and position of fish, we compare the view depending silhouettes of live and virtual fish. This is done by searching the nearest virtual silhouette pixel $v_i \in S_v$ for each real silhouette pixel $r_i \in S_r$ in 2D pixel space. We employ a brute force matching algorithm which searches the nearest neighbor on base of the Euclidean distance. We finally store all distances in order of real pixel silhouettes. In case of a single camera setup the error vector has always the same size as the real silhouette and depends on the model parameters X.

$$e(X) = \begin{pmatrix} e_0(X) \\ e_1(X) \\ \vdots \\ e_n(X) \end{pmatrix} = \begin{pmatrix} \|r_0 - v_i\| \\ \|r_1 - v_j\| \\ \vdots \\ \|r_n - v_k\| \end{pmatrix} \quad v_n \in \tilde{S}_X \quad (7)$$

In case of a multiple camera setup with more than one silhouettes of a single live fish (one silhouette for each camera view), the error vectors for each silhouette are stacked in a new error vector. $e^k(X)$ describes the error vector $e(X)$ of camera k. The final error vector of a multiple view setup has the same size as the sum of all real silhouettes pixels.

$$e(X) = \begin{pmatrix} e^1(X) \\ e^2(X) \\ \vdots \\ e^k(X) \end{pmatrix} \quad (8)$$

4.4 Optimization

In order to approximate the pose and position parameters of the virtual fish we employ a least-squares optimization method (combination of Levenberg-Marquardt method and quasi-newton method [DJGW81]) which optimizes the model parameters by reducing the error-vector described in equation 7. The optimization method uses the derivative of the error function with respect to the parameter vector X. We numerically approximate a derivative vector for each real silhouette pixel r_i as follows:

$$D_i(X) = \begin{pmatrix} \frac{e_i(X_{0+\varepsilon}) - e_i(X_{0-\varepsilon})}{2\varepsilon} \\ \frac{e_i(X_{1+\varepsilon}) - e_i(X_{1-\varepsilon})}{2\varepsilon} \\ \vdots \\ \frac{e_i(X_{j+\varepsilon}) - e_i(X_{j-\varepsilon})}{2\varepsilon} \end{pmatrix} \quad (9)$$

$$with\ X_{0+\varepsilon} = \begin{pmatrix} x_0 + \varepsilon \\ x_1 \\ \vdots \\ x_j \end{pmatrix}$$

In case of a multiple camera setup, the derivative has to be calculated for all silhouette pixels of all views. In case of using a single camera setup it is recommended to use a Kalman filter to stabilize the fish position, pose and bending.

4.5 Initialization

The better the optimization initialization the higher the probability of convergence and the faster the optimization can be finished. Especially for real-time applications fast initialization is very important. For this task, we employ a method which searches an initial model parameter set out of a database based on simple silhouette features like, position of snout, centroids of silhouette quarters or angle of major segment axis. In order to find the fitting model parameter set for the incoming silhouettes, the features are extracted and a brute force matcher searches the nearest neighbor. To speed up the process of pose initialization we use the method described in [MSK16] which defines a feature subset regarding the pose-space location. In order to initialize the position of fish, we calculate the centroid of the live fish silhouette and define the pixel-ray of this centroid pixel with the help of the camera projection matrix. In case of an multiple camera setup, we calculate the rough intersection point of these pixel-rays and use it as initial position. If a single view setup is used, the approximated position is calculated by shifting the model along the centroid pixel ray to the middle of the fish tank. In case of a post processing application, it is also possible to find the initial pose and position manually.

4.6 Estimation of fish-size from single and multiple view imagery

Besides a fish's shape, its correct size is very important for a precise recovery of position and pose of the live

ISBN 978-80-86943-41-1

fish. One option is to measure the size of the fish manually. This can be quite difficult since the fish has to be caught and its body has to be aligned along the measurement tool. An easier option is to use the computer vision system to measure the size of the fish. In the proposed method we also apply the in subsection 4.4 described optimization method in a preprocessing step. To do so we record a short video sequence of the swimming fish and besides the pose and position parameters we also optimize the size in x-, y- and z-direction. Depending on the used model, it is also possible to optimize the scale of the bones in order to adjust the shape of the fish automatically. We average the size parameters over the whole test sequence and use these parameters for the actual recovery process. For a multiple view setup, the size can be approximated fast and precisely. In contrast, in a single-view setup the model-size can not be recovered exactly: the projected size of the silhouette depends on the size of the model as well as on the distance between camera and object. For that reason we use a constrained size optimization, in which the fish position is bounded to the size of the fish tank. In order to get a good result, the recorded fish movement should cover the area in front of the tank's front and back wall.

4.7 Multiple fish and occlusion handling

The method presented in this paper is capable of multiple fish tracking. It is recommended to use a multiple camera setup in order to increase the stability of the system in case of occlusion. As long as no occlusion occurs, every fish can be handled separately according the previous described method. In case of occlusion, we modify the method as follows:

Silhouette mapping

Since the mapping of silhouette and fish is straightforward in case of a single fish (single silhouette to single fish), the problem of silhouette mapping occurs if several fish have to be tracked. In order to find the right silhouette for each fish, we compare the extracted contours of the current image regarding equation 7 with the silhouette of each fish model of the last frame. We assign the extracted contour to the model with the smallest error. This is done for each frame and for each camera view.

2D silhouette retrieving for occluding fish

If two or more fish cover each other in a camera view, the background subtraction method will just provide a single silhouette for these fish. For approximation of the virtual contour to the real silhouette it is necessary to reconstruct the silhouette as good as possible. We do so by creating the silhouette of each involved fish separately and merge these silhouettes together. This results in a single silhouette which consists of all outer silhouette edges.

Optimization in case of occlusion

Due to the fact that more fish are involved in the optimization process we combine all parameter vectors X of the involved fish to a new parameter vector. The contour comparator works the same way as described in section 4.3 except that the silhouette pixels of the combined silhouette are used for the camera view where the occlusion takes place. For the optimization all silhouette pixels of all fish in all camera views were used to approximate the virtual models to the live ones. Tests showed that the combined silhouette of multiple fish brings a higher risk of wrong convergence. For that reason we will check if the involved fish has a separate silhouette in another camera view and push this silhouette twice to the optimization process. By doing so this fish silhouette has a higher impact to the optimization process and the risk of wrong convergence decreases.

4.8 Handling of transparent fish parts

Another difficulty of fish pose recovery is the handling of transparent parts like fins. In our experiments we figured out that especially semi-transparent fins can cause trouble: depending on the fish position and alignment, it could happen that, for the background subtraction system, a fin is visible in some regions of the fish tank and invisible in other regions. This can cause problems for the method presented here since we extract (see chapter 4.1) the outer silhouette of the fish. If for example the caudal fin is not always visible, it will influence the optimization algorithm negatively. In order to handle this problem, we recommend to organize fish parts (e.g. fins) in mesh-groups. If a part is not detected by the background subtraction, it can be easily removed from the model. In case a fin is detected from time to time we extract the silhouette of this fin separately and add it to the total contour. By doing so both contours (with and without fin) are available and the contour comparator searches for the best matching one. This is also shown in figure 4.

5 RESULTS

We compared the results of the here introduced method regarding runtime and precision with a manually annotated dataset. This included the results of single-camera setup, multiple camera setup, with and without refraction compensation. Additionally, we also applied the method to a dataset of two fish including occlusion in one and both camera views.

5.1 Dataset

The dataset consisted of 1000 manually annotated frames which show a single female sailfin molly swimming in a fish tank (26 cm x 18 cm x 17 cm). The fish had a length of approximately 5 cm which corresponds to about 180 to 200 pixels. For annotation we manually

 ISBN 978-80-86943-41-1

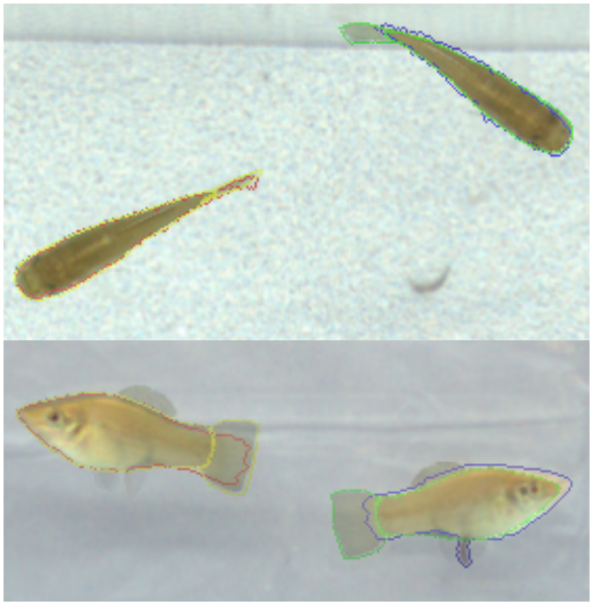

Figure 4: Silhouettes of two fish. Silhouettes of background subtraction is marked blue and red, the virtual silhouettes yellow and green. The quality of the real contour was not stable at all. Especially the transparent fins were sometimes not detected by the background subtraction like in case of the right fish.

adjusted the model parameters of the according 3D fish-model and rendered (raytracing with refraction based on the method validated in [MSKK14]) it frame-by-frame over the video footage of two cameras, observing the fish. The second dataset showed two female sailfin mollies (about 5 cm, approx. 120 to 140 pixels in length), swimming close to each other through a bigger fish tank (60 cm x 30 cm x 30 cm). Since the fins of the used fish are nearly transparent, the background segmentation method sometimes did not detect the whole fin. This caused an imprecise silhouette (see figure 4) and special demands on the proposed method regarding fin pose recovery. Both datasets were recorded by two cameras (Allied Vision Technologies, Prosilia GT1910c) with a resolution of 1920 x 1080 pixels and with a frame rate of 57 frames per second mounted above and in front of the tank. The cameras were synchronized by hardware trigger.

5.2 Model

The used 3D-model was created with the *FishSim Animation Toolchain* and consisted of 946 vertices and 36 bones. For the optimization we just used 17 bones: head, four backbones and twelve bones of the tail (caudal) fin. In order to decrease the model parameters we used a function that approximates the fish bending to a single bending value (see [SMK15]). In total the optimization process comprised six parameters per fish: position (x,y,z), rotation (pitch, yaw) and bending. During size estimation the parameter set was extended by three size parameters. We also tested the method with three and eight different bending parameters.

5.3 Initialization

Before the optimization process started a rough initial position and rotation of fish were found as described in section 4.5. The used database consisted of 32400 parameter sets. The features were extracted in a preprocessing step out of 32400 artificial fish images, showing a single fish (rendered from the used 3D-model) rotated in two degree steps around the main axes. Fish size was estimated out of 100 images from two perspectives (top and front) using the method described in section 4.6. We estimated the size along x-, y- and z- axis of fish separately. We also estimated the size manually. The results of manual and automatic size estimation differed very slightly.

5.4 Refraction compensation

We applied the proposed method to the first dataset, showing a single fish from two perspectives (front and top view), and recovered fish pose, position and bending with and without refraction compensation. The recovery results especially differed in terms of position error. The method using refraction compensation reached a mean error of 1.13 mm (standard deviation 1.00 mm, biggest error 3.9 mm) and the version without compensation reached a mean error of 8.24 mm (std. dev. 4.91 mm, biggest error 19.5 mm). Regarding rotation and bending error the methods differed less, but the version with refraction compensation was always better (see figures 5, 6, 7). This is due to the fact, that the silhouettes of the not refracted version do not fit precisely to each other, especially at the outer borders of the fish tank where the refraction is particularly high.

5.5 Single view vs. dual view

Besides the dual camera setup, we also applied the method with refraction compensation to single camera setup. We used the first dataset and applied the method separately to the top and front camera. As expected, the errors of position, rotation and bending increased. Especially the error for the dimension along the direction of camera view increased strongly (see figures 5, 6, 7). In general, for the here used fish species (sailfin molly) the top view camera delivered better results. This was mainly due to the fact, that the bending can better be observed from top (see figure 4). For experiments with a narrow third dimension (little water or thin tank) this method can be a cost-effective and less computation-intensive alternative to the multiple camera setup.

5.6 Occlusion of multiple fish

We recovered pose and position of two fish out of the second dataset (dual camera setup). The recorded video

ISBN 978-80-86943-41-1

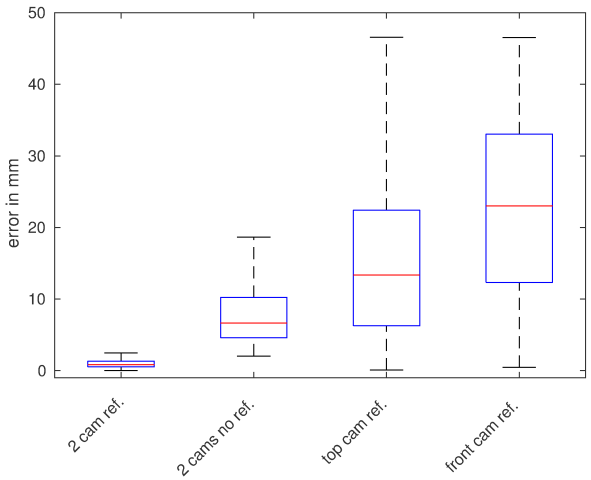

Figure 5: Fish position error split by method of application. The diagram shows box-plots, which indicates the median (red line) the 25th and 75th percentiles(blue box) and the biggest error values (whiskers) not considered outliers. From left to right it shows the position error (in relation to ground truth in mm) 1. using two cameras and refraction compensation 2. two cameras without refraction compensation 3. single top view camera with refraction compensation and 4. single front view camera with refraction compensation.

included several sequences with fish occlusion in one or both views. The algorithm recovered fish position and pose of both reliably. If an occlusion occurred in all views, it can happen that the algorithm will converge to the wrong fish model. This problem can be minimized by applying a kalman-filter to each fish. In figure 8 an occlusion case and the recovered fish positions and poses are shown. Additionally, a video sequence showing occluding situations can be found inter alia in

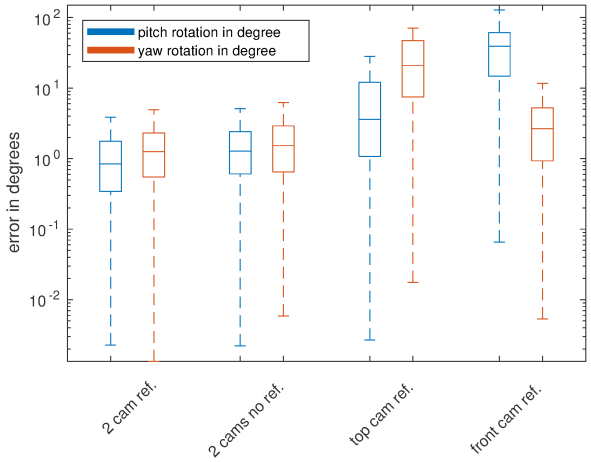

Figure 6: Fish pitch- and yaw-rotation error split by method of application. The diagram shows box-plots of pitch-rotation (blue) and yaw-rotation (red) in degrees. Since the used test fish nearly never rotate around roll-axis, this rotation was neglected.

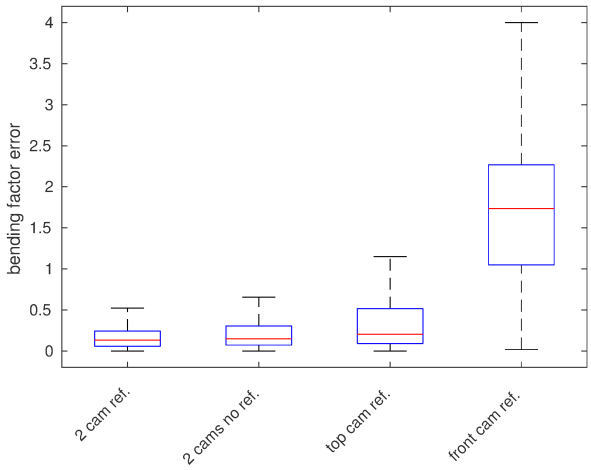

Figure 7: Fish position error split by method of application. The diagram shows box-plots of bending-factor errors. The factor maps fish bending to model parameters.

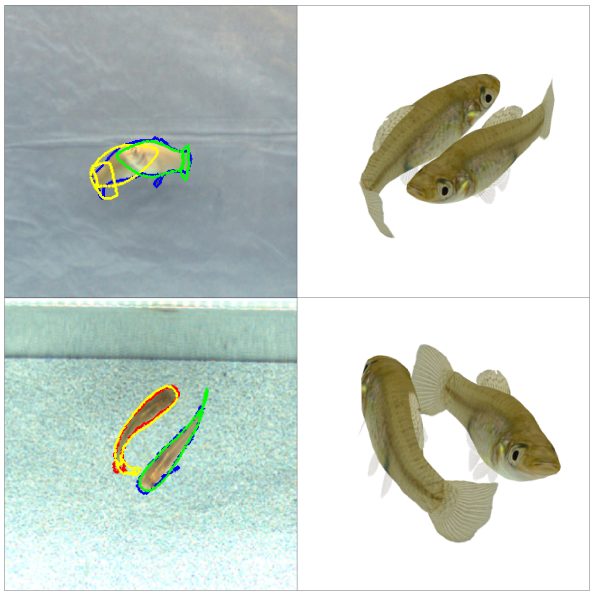

Figure 8: Reconstruction of two fish with occlusion. On the left the capture images with the silhouette overlay (blue and red contour - captured silhouettes; yellow and green contour - silhouettes of virtual fish). On the right, rendered images of the reconstructed fish models captured from two different perspectives.

the supplementary file. In general, in case of occlusion, the recovery method slightly loses accuracy.

5.7 Runtime

The software was tested on a system with Intel I7-3770 (4 x 3.4 GHz) CPU, 16 GB memory and Ubuntu 14.04 operation system. The algorithm was implemented with the help of the following open source libraries: *OpenCV* (version 2.4.12, `https://opencv.org/`) for computer vision tasks, *Dlib* (version 19.2, `http://dlib.net/`) for optimization and the game en-

ISBN 978-80-86943-41-1

Table 1: Algorithm's runtime under different configurations

configuration	runtime per frame / 2 frames in sec. (fps)
single fish, single camera, no refraction compensation	0.07 (14.2)
single fish, single camera, with refraction compensation	0.25 (4)
single fish, two cameras, no refraction compensation	0.14 (7.1)
single fish, two cameras, with refraction compensation	0.5 (2.0)
two fish, two cameras, with refraction compensation	0.7 (1.4)
single fish, two cameras, with refraction compensation and size estimation (three additional parameters)	0.67 (1.49)
single fish, two cameras, with refraction compensation and eight bending parameters	0.92 (1.08)
single fish, two cameras, no refraction compensation and eight bending parameters	0.25 (4)

gine *irrlicht* (version 1.8.1, `http://dlib.net/`) for rendering and silhouette extraction. We measured the mean time which was needed to process one frame (or two frames in case of two cameras). Table 1 gives a rough impression of the computational-intensity of different configurations. The refraction compensation was relative computational-intensive since every silhouette pixel was optimized separately. For runtime improvement it could be interesting to find an analytic solution of equation 5 which substitutes the optimization. In general it can be noted, that increasing the number of cameras and of fish the runtime increases approximately linear. Additionally, with up-to-date hardware, the method can be real-time capable.

6 CONCLUSION

In this work we introduce a new method to approximate 3D fish skeletal model parameters out of single- or multiple view video stream. We propose a new method to synthesize the refraction effect during optimization. We successfully applied the method to two different datasets with different configurations: we extracted model parameters for a one and two fish with and without refraction compensation. We showed that refraction compensation increases the recover accuracy: for position recovery the mean error was reduced by $\sim$85 % for rotation by $\sim$20 % and for bending by $\sim$11 %. We demonstrated that it is possible to recover the 3D-model parameters out of a single view video stream and reduce the runtime at the same time. By doing so it is possible to use the method in real-time application.

ACKNOWLEDGEMENTS

The presented work was developed within the scope of the interdisciplinary, DFG-funded project "virtual fish" (KU 689/11-1 and Wi 1531/12-1) of the Institute of Real-Time Learning Systems (EZLS) and the Research group of Ecology and Behavioral Biology at the University of Siegen.

7 REFERENCES

[BP10] Sachit Butail and Derek A Paley. 3d reconstruction of fish schooling kinematics from underwater video. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2438–2443. IEEE, 2010.

[BP12] Sachit Butail and Derek A Paley. Three-dimensional reconstruction of the fast-start swimming kinematics of densely schooling fish. *Journal of the Royal Society Interface*, 9(66):77–88, 2012.

[BS00] John W Buchanan and Mario C Sousa. The edge buffer: A data structure for easy silhouette rendering. In *Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, pages 39–42. ACM, 2000.

[CSWS17] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, volume 1, page 7, 2017.

[DDYP13] Johann Delcourt, Mathieu Denoël, Marc Ylieff, and Pascal Poncin. Video multitracking of fish behaviour: a synthesis and future perspectives. *Fish and Fisheries*, 14(2):186–204, 2013.

[DJGW81] John E Dennis Jr, David M Gay, and Roy E Walsh. An adaptive nonlinear least-squares algorithm. *ACM Transactions on Mathematical Software (TOMS)*, 7(3):348–368, 1981.

[GMS+17] Stefanie Gierszewski, Klaus Müller, Ievgen Smielik, Jan-Marco Hütwohl, Klaus-Dieter Kuhnert, and Klaudia Witte. The virtual lover: variable and easily guided 3d fish animations as an innovative tool in mate-choice experiments with sailfin mollies-ii. validation. *Current Zoology*, 63(1):65–74, 2017.

 ISBN 978-80-86943-41-1

[IAW+15] Spencer J Ingley, Mohammad Rahmani Asl, Chengde Wu, Rongfeng Cui, Mahmoud Gadelhak, Wen Li, Ji Zhang, Jon Simpson, Chelsea Hash, Trisha Butkowski, et al. anyfish 2.0: an open-source software platform to generate and share animated fish models to study behavior. *SoftwareX*, 3:13–21, 2015.

[KKTM15] Tejas D Kulkarni, Pushmeet Kohli, Joshua B Tenenbaum, and Vikash Mansinghka. Picture: A probabilistic programming language for scene perception. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 4390–4399, 2015.

[MSH+17] Klaus Müller, Ievgen Smielik, Jan-Marco Hütwohl, Stefanie Gierszewski, Klaudia Witte, and Klaus-Dieter Kuhnert. The virtual lover: variable and easily guided 3d fish animations as an innovative tool in mate-choice experiments with sailfin mollies-i. design and implementation. *Current Zoology*, 63(1):55–64, 2017.

[MSK16] Klaus Müller, Ievgen Smielik, and Klaus-Dieter Kuhnert. Optimal feature-set selection controlled by pose-space location. In *VISIGRAPP (4: VISAPP)*, pages 200–207, 2016.

[MSKK14] Klaus Müller, Jens Schlemper, Lars Kuhnert, and Klaus-Dieter Kuhnert. Calibration and 3d ground truth data generation with orthogonal camera-setup and refraction compensation for aquaria in real-time. In *Computer Vision Theory and Applications (VISAPP), 2014 International Conference on*, volume 3, pages 626–634. IEEE, 2014.

[PMBH+10] Gerard Pons-Moll, Andreas Baak, Thomas Helten, Meinard Müller, Hans-Peter Seidel, and Bodo Rosenhahn. Multisensor-fusion for 3d full-body human motion capture. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 663–670. IEEE, 2010.

[Pop07] Ronald Poppe. Vision-based human motion analysis: An overview. *Computer vision and image understanding*, 108(1-2):4–18, 2007.

[RLS09] Daniel Roetenberg, Henk Luinge, and Per Slycke. Xsens mvn: full 6dof human motion tracking using miniature inertial sensors. *Xsens Motion Technologies BV, Tech. Rep*, 2009.

[SMK15] Ievgen Smielik, Klaus Müller, and Klaus-Dieter Kuhnert. Fish motion simulation. In *ESM-European Simulation and Modelling Conference*, pages 392–396, 2015.

[SSK+13] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[ST02] Cristian Sminchisescu and Alexandru Telea. Human pose estimation from silhouettes. a consistent approach using distance level sets. In *10th International Conference on Computer Graphics, Visualization and Computer Vision (WSCG'02)*, volume 10, 2002.

[THHN00] Hiroki Takahashi, Junji Hatoya, Naoki Hashimoto, and Masayuki Nakajima. Animation synthesis for virtual fish from video. In *Proceedings of the 10th ICAT (International Conference on Artificial reality and Telexistence)*, pages 90–97, 2000.

[VIC+13] Thor Veen, Spencer J Ingley, Rongfeng Cui, Jon Simpson, Mohammad Rahmani Asl, Ji Zhang, Trisha Butkowski, Wen Li, Chelsea Hash, Jerald B Johnson, et al. anyfish: an open-source software to generate animated fish models for behavioural studies. *Evolutionary Ecology Research*, 15(3):361–375, 2013.

[VPvL16] Cees J Voesenek, Remco PM Pieters, and Johan L van Leeuwen. Automated reconstruction of three-dimensional fish motion, forces, and torques. *PloS one*, 11(1):e0146682, 2016.

[WGCT17] Klaudia Witte, Stefanie Gierszewski, and Laura Chouinard-Thuly. Virtual is the new reality. *Current Zoology*, 63(1):1–4, 2017.

[WRKS16] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016.

 ISBN 978-80-86943-41-1