Unwind: Interactive Fish Straightening

Francis Williams[1], **Alexander Bock**[2], **Harish Doraiswamy**[1], **Cassandra Donatelli**[3], **Kayla Hall**[4], **Adam Summers**[4], **Daniele Panozzo**[1], **Cláudio T. Silva**[1]
[1]New York University; [2]Linköping University; [3]Tufts University; [4]University of Washington
[1]{francis.williams,harishd,panozzo,csilva}@nyu.edu; [2]alexander.bock@liu.se;
[3]cassandra.donatelli@tufts.edu; [4]{kchall8,fishguy}@uw.edu

ABSTRACT

The *ScanAllFish* project is a large-scale effort to scan all the world's 33,100 known species of fishes. It has already generated thousands of volumetric CT scans of fish species which are available on open access platforms such as the Open Science Framework. To achieve a scanning rate required for a project of this magnitude, many specimens are grouped together into a single tube and scanned all at once. The resulting data contain many fish which are often bent and twisted to fit into the scanner. Our system, *Unwind*, is a novel interactive visualization and processing tool which extracts, unbends, and untwists volumetric images of fish with minimal user interaction. Our approach enables scientists to interactively unwarp these volumes to remove the undesired torque and bending using a piecewise-linear skeleton extracted by averaging isosurfaces of a harmonic function connecting the head and tail of each fish. The result is a volumetric dataset of a individual, straight fish in a canonical pose defined by the marine biologist expert user. We have developed Unwind in collaboration with a team of marine biologists: Our system has been deployed in their labs, and is presently being used for dataset construction, biomechanical analysis, and the generation of figures for scientific publication.

Author Keywords

CT Scan Data, Volumetric Deformation, Interactive System

CCS Concepts

•**Human-centered computing** $\rightarrow$ **Human computer interaction (HCI);** *Visual analytics; Visualization toolkits;*

INTRODUCTION

New tools often lead to scientific discoveries, and this is particularly true for new 3D imaging technology, which has helped advance many scientific areas. The availability of 3D imaging scanners has resulted in tens of thousands of large datasets to be analyzed. Our work is centered on the *ScanAllFish* and *oVert* projects, which are large-scale efforts to (CT) scan all the world's known species of fishes and other vertebrates [13, 54]. The Micro-CT scan allows scientists to determine the skeletal structure of a variety of fishes, opening the doors to a better understanding of relationships among skeletal elements and the degree of skeletal mineralization, as well as enabling population-wide studies that were previously impossible. Both volumetric and surface renderings are useful for making quantitative measures of skeletal parameters that are used to build evolutionary trees and demonstrate the directional variation of morphology over evolutionary time [19, 24, 49].

The extracted skeletal geometry can be used to make physical models of function and support the understanding of swimming motions by combining finite element modeling and computational fluid dynamics. Finally, scans allow researchers, curators, and scientific communicators to make 3D printed replica of these fishes for expositions and museum archives. The huge number of fish, their variety, and the scanning techniques involved cause unique challenges. As previously described in Bock [2], fishes are packed together for scanning purposes, with multiple fishes being placed inside a single scanning chamber. Every fish is twisted in a different way, they have different sizes and shapes, and those that are too long are *curled up* to fit into the scanner. This method of packing multiple fishes together allows for rapid scanning of multiple species, but causes difficulty in analyzing the raw volumes. In fact, there are two fundamental problems that hamper effective use of the data: (1) the difficulty of separating each fish into its own volume and (2) dealing with fishes that are bent and twisted in different poses, making side by side comparisons impossible.

While the first problem can be addressed using existing segmentation techniques [2], the second problem is the challenge that we address in this paper: we propose an interactive pipeline to reverse this undesired deformation, restoring the original shape of the fish into a canonical straight pose, and thus facilitating the analysis and visualization of these valuable datasets.

Ideally, fish straightening would be performed completely automatically. Unfortunately, this requires the detection and measurement of the distortion that each exemplar underwent during the packing in the CT scanner. This information is impossible to acquire with a CT scan, since it requires the measurement of the volumetric stresses in the exemplar itself. Custom interactive tools therefore play an important role in helping users effectively guide this straightening process. While existing off-the-shelf software support deforming 3D

CHI'20, April 25–30, 2020, Honolulu, HI, USA

ISBN 978-1-4503-6708-0/20/04...$15.00
DOI: https://doi.org/10.1145/3313831.XXXXXXX

volumes, these approaches require users to work directly in 3D making its usage time consuming even for experienced users, let alone our target users who are not familiar with 3D modelling software. Hence, it is essential to optimize user interactions so as to not overburden the users in their workflow. In fact, minimizing human effort for various tasks has recently gained traction in the design of user interfaces. For example, Hong et al. [20] showed that designing an interface with minimal user-selectable information was most effective in the context of understanding accessibility in cartographic visualization; Ono et al. [39] designed an interface to track baseball plays that reduces the annotation burden on the user. Similarly, Choi et al [8] proposed an approach that automatically emphasizes words within a document and prividing recommendations in order to reduce the burden on users labeling documents.

Following along the above strategy, we design Unwind, a user-driven, interactive volumetric straightening system that provides a single interface to start working directly with the original CT data, and enables the marine biologist to quickly and efficiently process the twisted volumes into clean data in a canonical straight pose. The user interaction is divided in two phases: (1) a selection phase, where the user picks a pair of 3D points to identify the extrema of the spine of a fish, from which the system automatically extracts a skeleton and an initial approximation of the straightened fish; and (2) a refinement (or tuning) phase, where the user navigates the 2D cross-section of the fish and fine-tunes the deformation by specifying additional rotations required to eliminate the torque and bending in the fish. The system is based on a simple but novel deformation method which is specifically designed for undoing the bent and twist introduced during the packing of multiple fishes, and that can be efficiently implemented on a GPU to ensure an interactive volumetric rendering of the undeformed dataset. Unwind is already in use in the labs participating in the ScanAllFish project.

The contributions of this paper can be summarized as follows:

- The design of an interactive tool allowing users to process a CT image, extracting individual fishes, and straightening them. This tool allows a marine biologist to process a dataset in 6 minutes on average.
- A simple deformation algorithm that enables an intuitive and easy user interaction by allowing users to interact with 2D slices instead of the complete 3D volume.
- A preliminary user evaluation, comparing the time and quality obtained by 10 users processing a representative dataset.
- A demonstration of the effectiveness of Unwind through expert feedback on processing 18 fishes.
- A reference system implementation.

RELATED WORK

Our approach combines techniques from geometry processing, to estimate the initial deformation, with rendering and deformations techniques developed within the visualization community. Here, we give an overview of the most closely related works, and we refer to [3, 23] for a complete overview.

Skeletonization via Discrete Maps.

We review here the skeletonization works applicable in our setting, and we refer an interested reader to [51] for a detailed overview.

Our skeleton construction is based on a harmonic volumetric parametrization [53, 30] constructed from a pair of user-provided landmarks. The isosurfaces of the scalar function are averaged to find points in the center of the fish: this construction is inspired by the hexahedral method proposed in [18] and the tubolar parametrization proposed in [33].

It is important to observe that the parametrization induced by these functions is not bijective [42], and it is thus not a proper foliation [4]: however, this is not a problem in our case since we use it only to compute an approximate skeleton. We opted for this skeleton extraction procedure since it allows users to intuitively and interactively control the skeleton, which is mandatory to make our system able to process challenging datasets. For a complete overview of skeletonization techniques, we refer an interested reader to [51].

Volumetric Deformation.

This problem has been heavily studied in both in the context of (1) *volumetric parametrization*, where a energy minimizing, quasi-static solution is found via numerical optimization, (2) in *physical simulation*, where the focus is on modeling dynamics effects, and (3) in *space warping techniques*, where a explicit reparametrization of the space is used to warp an object. Since we are not interested in dynamics (we only need to deform the volume once), we only review the parametrization and free form deformation literature, and we refer an interested reader to [55, 34, 38, 46, 22] for an overview of dynamic physical deformation techniques.

Volumetric Parametrization.

A convex approximation of the space of bounded distortion (and thus inversion-free) maps has been proposed in [31, 25], allowing to efficiently generate these maps both in 2 and 3 dimensions. These methods do not require a starting point, but they might fail to find a valid solution in challenging cases. A different approach, guaranteed to work but requiring a valid map has been proposed in [21, 15]: the idea is to evolve the initial map to minimize a desired cost function, while never leaving the valid space of locally injective maps. Many variants of this construction have been proposed, either enriching existing deformation energies with a barrier function [43], or directly minimizing energies that diverge when elements degenerate [47]. Specific numerical methods to minimize these energies have been proposed, including coordinate descent [21, 17], quasi-newton approaches [47, 26, 41, 45, 9], and Newton [43] methods. A last category of methods [16, 40] produces an initial guess separating all triangles and rotating them into the UV space, and then stitches them together using Newton descent. However, all these methods are computationally intensive, and not suitable for interactively deforming high-resolution CT scans.

Space Warping.

Closed-form volumetric deformations have been defined using lattices [44, 1, 10] or other parametrizations [6]. While not directly minimizing for geometric distortion, they have the major advantage of being directly usable in a volumetric rendering pipeline, enabling to render in real-time the deformed volume. Our approach is also directly usable in a real-time volumetric rendering pipeline and uses a keyframed skeletal parametrization to define the deformation.

Volume wires [52] uses a skeleton to define a free form deformation, parametrizing it with values attached to the skeleton itself. Volume wires relies on a computationally intensive evaluation which prevents its use in a real-time rendering systems. Our method shares the idea of using a skeleton to parametrize the deformation, while providing detailed deformation control using keyframes, an algorithm to automatically estimates an initial deformation, and being specifically tailored to be used in an interactive volumetric rendering system.

Interactive Applications.
Many variants of the previous volumetric deformation techniques have been used in interactive applications in the visualization community: since a complete overview is beyond the scope of this work, we limit our review to the most closely related works, and we refer an interested reader to the surveys by Sun *et al* [50] for visual analytics, and Liu *et al.* [32] for information visualization techniques.

Closely related to this work, Correa *et al.* [12] introduced an interactive visual approach to deform images as well as volumetric data based on a set of user-defined control points. Here, the deformation is controlled based on the movement of these control points. Even though their formulation introduced distortions in other regions of the data, since their focus was on illustrative applications and volume exploration and visualization, such distortions were acceptable since they were occluded in the visualization. On the other hand, our goal is to generate data that will be further analyzed by the marine biologists. It is therefore necessary that such distortions are avoided. Such distortions are common in other approaches as well that perform volume deformation with the focus on exploration and/or animations [11, 27].

There have also been visual approaches that target generating illustrations with the focus on medical data [35, 29], in particular, generating views when the covering surface is “cut open". Since these approaches distort the data that is deformed, they are not suitable for our work. Nakao *et al.* [36, 37] proposed an interactive volume deformation, also catered towards medical applications, which deforms the volume based on a proxy geometry that approximated the volume. However, the proxy geometry itself is computed in a preprocessing phase, which makes the combined pipeline not interactive.

Directly related to CT scans of fishes, in our previous work we proposed TopoAngler [2] that combines a topology based approach with a visual framework to help users segment fishes from the CT data. TopoAngler is used as a preprocessing step in this work, to extract a segmented fish (Section 5).

UNWIND: DESIGN OVERVIEW

Processing one CT dataset containing a packed set of fishes requires multiple steps, which are currently only possible by using different software packages, and that require conversions and manual processing to be combined: (1) to process the scanned images into a 3D volume format (and optionally subsample the volume), (2) to segment and export individual fishes from the 3D volume, and (3) to deform the fishes into a canonical pose. While straightening the individual fishes is desirable, given that there exists no off-the-shelf tool to accomplish this, it was not possible for the users to do this. Our goal in the design of Unwind is to provide a single, efficient, and intuitive tool to process the CT data once they obtain it from the scanning software, enabling non-expert users to process the massive amount of data which is acquired daily in marine biology labs. To accomplish this, we divide the entire process into four stages:

1. Load CT data: The user can directly load the output from the CT scanner, which is then subsampled to fit in the video memory of the workstation to ensure an interactive preview of the deformation.

2. Segment and extract a single fish: For the initial segmentation, we decided to integrate the functionality from the open sourced TopoAngler [2] due to three reasons: (1) it is widely used by marine biologists, being the only tool specifically designed for this tasks; (2) the user interaction is intuitive, requiring only a single parameter accompanied with a few clicks from the user to select the required subvolumes and, (3) it is interactive, providing a live preview even on complex volume scans of multiple fish. This segmentation approach first generates a hierarchical segmentation based on the join tree of the input data [5], and then allows the user to interactively choose the simplification to be performed and select the segmented sub-volumes corresponding to the fish. We refer the reader to [2] for a detailed description of this algorithm.

3. Estimate the straightened volume: In this stage, our goal is to semi-automatically estimate the straightened volume using user input. Since the spine (or the mathematical skeleton) of the fish traces out a curve in space, it was a natural decision to define our deformation as a warping of a curve in space. This requires to compute the spine geometry followed by defining the deformation based on the warping that straightens it. While automatically extracting the spine for a given fish would be ideal, existing skeletonization methods are not easily adaptable in our setting given the varied shapes and sizes different fishes take. Therefore, we decided to employ a minimalistic user assisted approach, wherein we require the user to select the two extremal end points of the fish, which are then used to find the fish skeleton. Note that the alternative of allowing the user to manually specify the 3D curve corresponding to the spine is an arduous task requiring the interaction with a 3D volume.

4. Refine the straightening. Since our deformation occurs along a curve in space, it was very natural to allow the user to interact with 2D cross sections along that curve. Once the approximate spine is computed, we define a set of coordinate frames along this curve, which is then used by the user to refine the deformation. This process mainly involves the user aligning the coordinate frames to generate an accurate spine from the estimated curve. Using such an interface was inspired

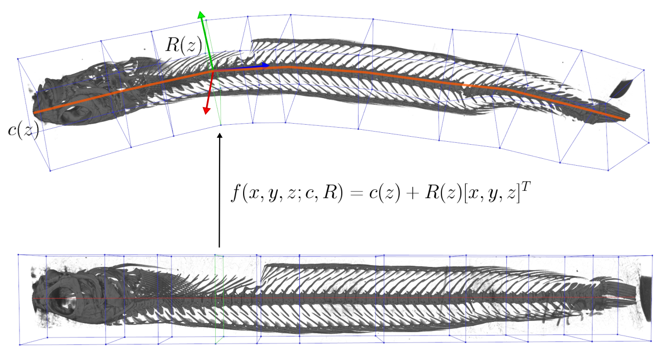

Figure 1. An illustration of our volumetric deformation: The density at a point (x,y,z) in the straight volume is determined by the cross sectional plane $R(z)$ centered at $c(z)$, where $c(z)$ is a piecewise linear curve sweeping the spine of the fish and $R(z)$ is a frame centered at $c(z)$ defining the orientation of the fish at a given position. $R(z)$ and $c(z)$ are specified at a set of *keyframes* (the blue squares) and interpolated linearly in between.

by popular video editing software in which users can edit a set of keyframes. The simplicity of editing in 2D combined with real time 3D visualization of the deformation makes for an easy-to-use tool allowing extremely precise control over the deformation. Not only is this deformation transformation simple mathematically (thus allowing interactivity), but users could also easily understand this procedure simply by using the software without requiring a mathematical explanation.

The next two sections focuses on the third and fourth steps of the above workflow, and describes in detail the user interface of Unwind. We would like to note that the described system was designed over multiple iterations spanning over a year based on constant feedback from our collaborators (who were also the initial users). We discuss this process after the description of the user interface.

CYLINDRICAL DEFORMATION

Without loss of generality, we assume that fishes are straightened one at a time. If more than one fish is present in the scanned volume, we isolate individual fishes using TopoAngler [2]. The key idea in our approach is to identify a deformation function f, which transforms an axis aligned bounding box (which will contain the straight fish), into a deformed version of the fish. This function, being the inverse of the deformation that the fish underwent, will be then used to recover the straight fish.

More concretely, we are interested in a mapping, f, which deforms a straight cylindrical region V_{straight} into a deformed one V_{twist} such that the central axis of the V_{straight} is mapped to a curve $c(t)$, which is aligned with the body of the twisted fish. Furthermore, for every z coordinate in V_{straight}, we define a rotation matrix $R(z)$ which maps points off the central axis to points in V_{twist}. The set of rotation matrices R captures both the “bends" that the fish underwent as well as the “twists" (or torsion) in the fish. Figure 1 illustrates one such deformation function f.

Thus, to parameterize f, we require a space curve $c(t) = [c_x(t), c_y(t), c_z(t)]^T$ and a continuous field of rotation matrices $R(t) = \left[u(t)^T, v(t)^T, n(t)^T\right]^T \in SO^3$, allowing us to write:

$$f(x,y,z;c,R) = c(z) + x \cdot u(z) + y \cdot v(z) \tag{1}$$

Intuitively, the direction along n in R points along the skeleton (central axis) of the fish, while the u and v directions define the right and up directions of the fish respectively. Thus, n allows us to undo any bending in the fish while u and v allow us to remove torsion.

In our setting, we use an arclength parameterized piecewise linear curve for $c(t)$. Note that the arclength parametrization ensures that the mapping will be close to an isometry independently on the speed of the parametrization of c. This is important since it allows the user to freely change the parametrization speed by adding additional keyframes, without introducing unwanted distortion (Section 5). Specifically, $c(t)$ can be parameterized by vertices $e_1, \dots e_m \in \mathbb{R}^3$. Letting $d(e_i) = \sum_{j<i} ||e_{j+1} - e_j||_2$, we can write $c(t)$ explicitly as:

$$c(t) = \lambda(t) e_{k(t)} + (1 - \lambda(t)) e_{k(t)+1} \tag{2}$$

where

$$k(t) = \underset{i,(d(e_i)<t)}{\arg\min}\,(t - d(e_i)) \tag{3}$$

$$\lambda(t) = \frac{t}{e_{k(t)} - d(e_{k(t)+1})} \tag{4}$$

To define $R(t)$, we define orthonormal coordinate frames $R_1 = (u_1, v_1, n_1)^T, \dots, R_m = (u_m, v_m, n_m)^T \in SO^3$ at each vertex e_i of c. For $R(t)$ to be continuously defined at all points along c, we identify the unit normals, n_i, with points on a sphere and spherically interpolate the n directions between adjacent R_i, R_{i+1}:

$$R(t) = \text{SLERP}(n_{k(t)}, n_{k(t)+1}, \lambda(t)) R_{k(t)} \tag{5}$$

As we show next, the parameters of the deformation function f are initially estimated by our system, and optionally interactively refined by the user in a second stage to compute the final straightened fish.

INTERACTIVE FISH STRAIGHTENING

Unwind uses the cylindrical deformation (Equation 1) to assist users in straightening deformed fishes. Once the individual fish is isolated from the input CT data, the remaining stages of the process can be further divided into the following 4 steps:

1. Compute the harmonic function used for estimating the deformation function f.
2. Estimate the parameters of the deformation function f.
3. (Optional) Refine the parameters of the deformation function f.
4. Export the straightened fish.

We now describe in detail each of these steps and the associated visual interface of our system. Each step comprises of its own set of visualization widgets, and the user can move forward and back between the different steps. The entire workflow is illustrated in the accompanying video. Note that the

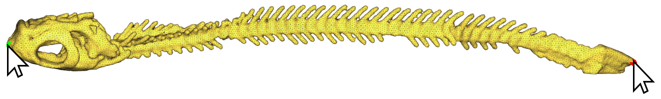
Figure 2. **The user selects two extrema on the endpoints of the fish by clicking on the extracted tetrahedral mesh.**

user input is used to guide this process during the different steps.

Compute Harmonic Function

We first approximate the fish by a smooth curve in order to estimate the parameters of the deformation function f. This curve is computed as the set of centroids of level sets of an harmonic scalar field defined on the volume, following a technique similar to [18].

Discretization.

Different techniques could be used to compute the harmonic function, and we opted for a finite element method due to its efficiency, simplicity, and robustness. While it is possible to use directly the voxel grid as a space discretization, this would be prohibitively expensive on the high resolution CT scan. Downsampling the image is a possibility, but it would lose the high-frequency details and risk to lead to disconnected components in the thin regions of the fish. We therefore opt for an adaptive tetrahedral mesh, generated using an implementation of Isosurface Stuffing [28], which strikes a good balance between boundary approximation and computational efficiency. While unlikely, it is possible that the generated tetrahedral mesh is made of multiple disconnected components, either due to lack of resolution, or due to the TopoAngler segmentation. To ensure there is only a single connected component, we inflate the voxel grid until all the connected components are merged using [7].

User-Provided Extrema.

The extrema of the harmonic function, used as boundary conditions, are provide by the user with an end-point selection widget allowing the user to select points on the segmented fish (see Figure 2). These points corresponds to the head and tail of the fish, making it straightforward for the user do this selection.

The two endpoints are then used to compute a discrete harmonic function with Dirichlet boundary conditions setting the head vertex v_{head} and the tail vertex v_{tail} as a source and sink:

$$(Lu)_i = \begin{cases} 0 & \text{if } u_i \neq v_{\text{head}}, u \neq v_{\text{tail}} \\ 1 & \text{if } u_i = v_{\text{head}} \\ -1 & \text{if } u_i = v_{\text{tail}} \end{cases} \tag{6}$$

Here, L is the discrete Laplace-Beltrami Operator [3], and u is a scalar field defined at each vertex of the tetrahedral mesh. While solutions to Equation 6 produce fields whose level sets trace out a reasonable skeleton approximation, the spacing between level sets is not uniform. To remedy this issue, we resample the curve using a fixed spacing between vertices in ambient space. To ensure the resampling process does not discard details, we use a sampling width which is half the size of the smallest segment in the curve traced out by the level sets of the harmonic solution.

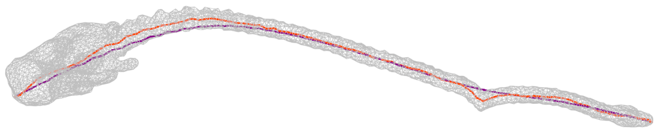
Figure 3. **Estimated skeletons using the centroids of level sets: The *red curve* shows the piece-wise linear curve where the vertices are 100 centroids of the level sets uniformly sampled on the scalar function u (Equation 6 defined on the tetrahedral mesh. The *purple curve* is the result of applying 50 smoothing steps the red curve.**

Estimating Parameters $c(t)$ and $R(t)$

To generate an initial estimate for the piecewise linear curve, $c(t)$, we sample k level sets of v at isovalues uniformly spread on $[-1,1]$. We then compute the centroids, $c_1,\dots,c_k \in \mathbb{R}^3$, of these level sets. While the piecewise linear curve whose vertices are the c_i's traces a curve approximating the bend of the fish, the curve itself might be noisy due to the complex boundary geometry. We thus apply s iterations of Laplacian smoothing (replacing every vertex with the average of its 2 neighbours) to smooth the curve. Figure 3 compares two curves before and after smoothing. Specifically, if c_{i-1}, c_i, and c_{i+1} are consecutive vertices of the curve, one iteration of smoothing can be written as:

$$\text{SMOOTH}(c_i) = c'_i = \frac{c_{i-1} + c_{i+1}}{2} \tag{7}$$

Once we have vertices c'_i, we then compute orthogonal coordinate frames $R'_1,\dots R'_k$, where $R'_i = (u'_i, v'_i, n'_i)^T \in \mathbb{R}^{3\times 3}$. First we compute n'_i at each of the c'_i using central differences on the interior and one sided differences at the boundary:

$$n'_i = \begin{cases} \frac{c'_{i+1}-c'_{i+1}}{\|c'_{i+1}-c'_{i+1}\|_2} & \text{if } 1 < i < k \\ \frac{c'_{i+1}-c'_{i+1}}{\|c'_{i+2}-c'_{i}\|_2} & \text{if } i = 1 \\ \frac{c'_{i+1}-c'_{i+1}}{\|c'_{i}-c'_{i-2}\|_2} & \text{if } i = k \end{cases} \tag{8}$$

We then compute u'_i and v'_i by projecting the x and y axes into the plane defined by n'_i. If such a projection degenerates, we repeat this procedure with the x and z axes as well as the y and z axes (one of them has to succeed since the skeleton is not degenerate by construction):

$$u'_i = \hat{x} - \hat{x}^T n_i * n_i, \tag{9}$$
$$v'_i = \hat{y} - \hat{y}^T n_i * n_i. \tag{10}$$

The result is a piecewise linear curve with vertices $c'_1,\dots c'_k$ and orthonormal bases $R'_1,\dots R'_k$ at each vertex. Note that the number of level set samples, k, and smoothing iterations, s are user-tunable parameters. The default is $k = 100$ and $s = 50$ and our users did not change them in any of their experiments.

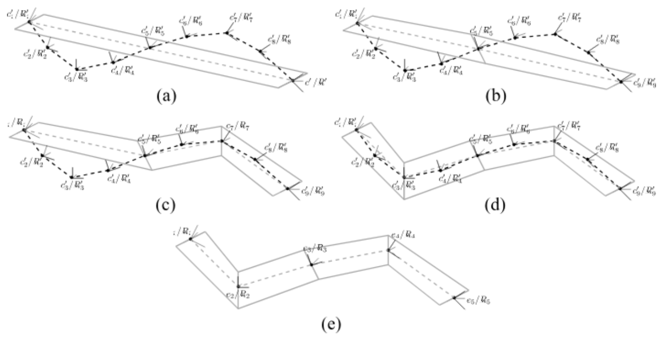

Figure 4. Computing a minimal set of parameters by subdivision: The black dotted line illustrates the initial estimated parameters c'_i and R'_i, using the method described in Section 5.2. We compute the gray prism by connecting two squares on the $u'_1 - v_1\prime$ and $u'_9 - v'_9$ planes. We progressively subdivide the prism at vertices c'_i until the subdivided prisms fully contains all the c'_i. This refinement procedure yields a new, reduced set of parameters $e_1, \ldots e_5$ and $R_1, \ldots, R_5$.

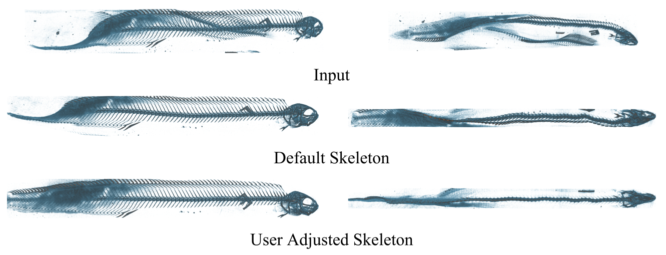

Figure 5. The deformation automatically estimated from the input (top) captures the majority of the distortion (middle) and can be further refined adding additional keyframes (bottom).

Computing a Minimal Set of Parameters.
Having a large numbers of parameters can become cumbersome to a user when refining the deformation (see Section 5.3 below). Thus, while we could use the parameters $c'_1, \ldots c'_k$ and $R'_1, \ldots R'_k$ for the deformation, we opt instead to compute a minimal set of parameters $e_1, \ldots, e_m$ and $R_1, \ldots R_m$, $R_i = (u_i, v_i, n_i)^T$, which agree with the c'_i's and R'_i's.

To compute these new parameters, we select a radius r and construct a prism whose bases are squares with side lengths $2r$. Each base is centered at c'_1 and c'_k and is oriented to lie in the planes n'_1 and n'_k with the sides aligned with u'_1, v'_1 and u'_k, v'_k. Figure 4(a) shows a 2D illustration of the initial configuration for an example curve.

Then, while the prism does not fully contain the vertices, $c'_1, \ldots c'_k$, we subdivide it by first choosing a vertex c'_{mid} and frame R'_{mid}, and then splitting a prism into two with a base centered at c'_{mid} and aligned with R'_{mid}. Figure 4(b)–(e) illustrates this subdivision procedure.

The resulting e_i and R_i are the vertices and coordinate frames of the subdivision location used to construct the prism. The radius hyperparameter, r is user selectable: A larger radius will yield a coarser approximation, and a smaller radius will yield a finer one. By default we set r to 10 voxel-widths, and our users did not adjust it in any of their experiments.

Figure 5 shows an example of the initial estimated deformation on a fish scan.

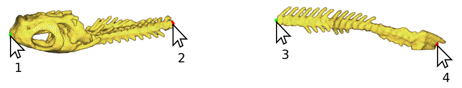

Figure 6. In cases where the segmentation outputs disconnected components, we recover a curve approximating the skeleton by selecting endpoints on each connected component. In the figure, the user selects points 1, 2, 3, and 4 and the system estimates a curve between 1 and 2 and another between 3 and 4 which are joined by a line connecting 2 and 3.

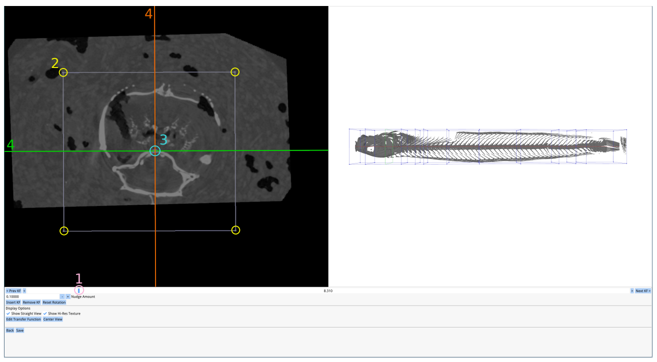

Figure 7. Our user interface for performing interactive deformations.

Handling Disconnected Components.
There are cases where the components corresponding to a fish are significantly far apart that the dilation operation performed when identifying the sub-volume corresponding to the fish is not sufficient to merge the two components (see Section 7.2 for details). To handle such a scenario, we allow the user to select the end points of the different components in order, compute the harmonic function and estimate the deformation function parameters for each component, and use the ordered end points to merge the different curves into a single one (Figure 6).

Deformation Refinement

We allow the user to interactively edit and refine the deformation parameters. The user interface for this step comprises 3 widgets (Figure 7): a *2D view* for editing in a cross section (Figure 7, left), a *3D view* showing the effect of the edits in real time (Figure 7, right), and a *control widget* providing buttons and sliders to help the user perform the deformation (Figure 7, bottom).

Control Widget.
This widget provides the user with a slider (Figure 7, 1) that controls the parameter t along the parametric curve $c(t)$. Users can use this to move along this curve to identify locations to edit the deformation. Buttons at either end of the slider allow the user to skip to parameters corresponding to the parameter vertices, e_i. The widget also provides options for the user to add new vertices, e_i and frames, R_i, as well as delete existing vertices and key frames. Alternatively, any interaction in 2D view will automatically add a new vertex and key frame along the curve. Additionally, this widget enables the user to view and change the transfer function of the rendered volume and includes controls to recenter the camera and toggle between the straight and deformed views.

2D View.
This view shows a cross section of the volume in the plane orthogonal to $n(t)$ corresponding to the currently selected parameter t in the control widget. Visual cues are overlaid over this cross section to allow user modify the different deformation parameters. These cues include:

1. A box corresponding to the region of space in the input which will be deformed to generate the output (Figure 7, 2). This represents the the prism base corresponding to the current parameter vertex e_i. The user can adjust the size of the bounding prism, V_{cage}, by dragging the corners of this box.

2. A point corresponding to the position of $c(t)$ in the 2D plane orthogonal to $n(t)$ (Figure 7, 3). Dragging this point allows the user to change change the position of vertices on the parametric curve.

3. The directions $u(t), v(t)$ in the 2D plane. The user can also rotate the $u(t)$ and $v(t)$ vectors around $n(t)$ by holding shift and dragging (Figure 7, 4). This feature allows the user to align $u(t)$ and $v(t)$ with the principal directions of the fish, thus allowing for the removal of any torsion which may be present in the twisted input.

3D View.
Depending on the option selected in the control widget, the 3D view visualizes in real time, either the straightened fish or the deformed bounding cage and curve $c(t)$. The former option allows the user to receive real-time feedback on how their edits affect the output, while the latter view allows the user to visualize how well their curve approximates the skeleton of the fish. The straightened volume is obtained by sampling the deformation function f after each update to its parameters. This is accomplished by mapping a regular lattice (which is used for the volume rendering) to the input volume which is then sampled using trilinear interpolation. To provide real time feedback at interactive rates, we perform this sampling using the fragment shader as part of the rendering pipeline.

Real-Time Rendering and Export
Our system, similarly to existing volume deformation pipelines [6], enables real-time rendering of the warped volume, allowing the user to instantaneously see the result of their actions. To render deformations in real time, our system computes a straight volume by evaluating $f(x,y,z)$ (Equation 1) along cross sections in z. We implement this evaluation in an OpenGL shader which renders cross sections along z into a volume texture. The texture size is determined by the prisms described in Section 5.2.

Once the user is satisfied with modeled deformation, the straightened volume can be exported onto disk. We use the same procedure for exporting as we do for real time rendering. To preserve the correct dimensions during export, the depth (z-direction) of the volume is set to the arclength of the linear curve $c(z)$. The width and height (x and y directions) are set to maintain the same aspect ratio as the prisms. The user can optionally edit the size of the exported volume.

In addition to exporting the volume, our application allows the user to save a session to disk and reload it later for further editing or inspection. This feature is important to ensure *provenance*, enabling to store a direct mapping between the straightened fish and the original RAW volumetric dataset.

DESIGN PROCESS
Once the problem (straightening scans of fish) was identified, we examined example scans and analyzed the existing workflow used by the marine biologists for manipulating CT scans. While there are manual deformation systems built into several commercial programs (e.g., Amira, Aviso) which have been used in the past (e.g., straightening the deformed jaw of a megamouth shark [14]), these methods require users to manually place and adjust reference points in 3D to perform the necessary deformation. To quote one of our collaborator, when asked how effective these tools were for straightening CT scans, his response was: *"I would say it simply cannot be done. A poor job took hours and hours over several days when I worked on the megamouth shark"*.

Moreover, the gold standard for 3D data acquisition demands that a museum specimen be imaged. That means the shape of the fish is set by the fixative used when the specimen was collected. So, fish are always, whether scanned singly or in groups, scanned with bent spines. Manual, physical straightening of the fish before scanning is difficult, time consuming, and can lead to damage to the specimen, and it is thus not a viable option.

So, our next step was to consider the adaptability of existing free form deformation based approaches to perform the deformation. This requires users to manually deform a lattice bounding the fish directly in 3D. This was an onerous task even for an expert savvy with 3D modelling tools and it requires a high learning curve. We therefore decided to build a custom tool for this purpose with aim of making the straightening process easy for our target users. In particular, our goal was to design an interface that requires minimal and simple user interaction using metaphors that the users were already familiar with.

The first version of our tool used a skeletal-based deformation performed by setting the length of the spine and the number of skeleton vertices. Here the user selects two endpoints on a segmented mesh, inputs a number of skeleton vertices, and the software computed a deformation using a volumetric extension of ARAP [48] to map the automatically computed skeleton to a straightened mesh. While this approach had only a few user inputs, it often failed if the skeleton extraction was imperfect. It also required the segmented mesh to include minute details of the fish which could not be easily obtained through Topoangler.

So, in the next iteration, we tried using a cage based deformation that used the above estimated skeleton to derive the initial bounding cage. The user then had to manipulate cage vertices to get the deformation. Again, depending on the quality of the segmented mesh obtained from Topoangler, the initial cage would often be far from the desired cage and hence required several interactions from the user to rectify it. Moreover, manipulating individual skeleton vertices was not only cumbersome, taking a lot of user time, but it also involved a high learning curve especially for users not familiar with 3D modeling tools.

To overcome the problems caused by the coarse segmentation, we decided to use our curve-based approach, which requires computing only the main spine of the fish. After observing the current tools used by the biologists for segmentation, we noticed they were split into a 2D editing widget for selecting a boundary along a cross section and a 3D widget for selecting an axis aligned cross section along x, y or z axis. Given this familiarity, we decided to use a 2D cross section view as well to help users adjust the alignment of the estimated spine with the actual spine and used a 3D view to show the results of the adjustments in real time. This design also reduced the user interactions, and removed unnecessary input such as cage vertices. Furthermore, this design was also robust in the sense that even in rare cases when the skeleton estimation is far from the actual spine of the fish, the user can easily recover by fixing the spine vertices.

The above version of the tool was deployed at our collaborators' labs during which time we were actively collecting feedback and fine tuning the system. In particular, as more people started using the tool for straightening different fish scans, certain important shortcomings were noticed that had to be fixed. In particular, (1) there were cases where the segmented fish consisted of disconnected segments, which had to be fixed (as described earlier); (2) the users requested the ability to rotate keyframes along a second axis (in the initial version one could only rotate about the normal tangent to the skeleton curve). This was necessary to correct minor warping that was sometimes caused in the output volume; and (3) Since the estimated skeleton tracks the spine, in some cases when the straightened volume was exported, fleshy parts of the bits near the endpoints of the spine were missing. To overcome this, an option was added to pad the exported volumes with additional keyframes at the endpoints.

RESULTS

The results reported in this Section were generated over a period of 8 weeks in the *Friday Harbour Laboratory* (FHL) in the Biology Department at *University of Washington*. Users used Unwind to straighten fishes on a workstation with a Intel Xeon CPU E5-1607 v4 @ 3.10GHz, 32 GB RAM and a NVIDIA GTX-1080-Ti GPU. In the supplemental material, we include raw screencasts of the editing sessions for all the results presented in the paper. Our reference implementation and one sample dataset can be downloaded from our GitHub website https://github.com/fwilliams/unwind: binaries are also provided for Windows and Linux.

Synthetic Evaluation.

We performed a synthetic experiment to verify that our system works on analytic deformations. In Figure 8, we show how a cylinder whose medial axis is a sinusoid can be deformed back to a straight line. We use a different number of keyframes, to show the effect of the refinement on the final deformation. We also compute the L^2 distances (normalized by the length of

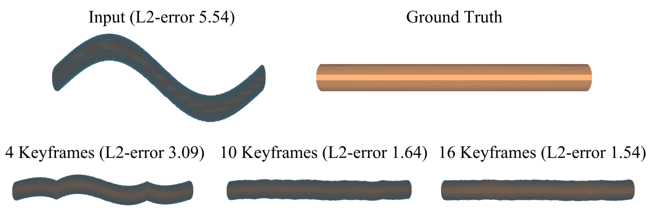
Input (L2-error 5.54)
Ground Truth
4 Keyframes (L2-error 3.09)
10 Keyframes (L2-error 1.64)
16 Keyframes (L2-error 1.54)

Figure 8. Straightening a synthetic example of a cylinder bent into the shape of a sine wave. As the user adds new keyframes, the result more closely approximates the ground truth cylinder.

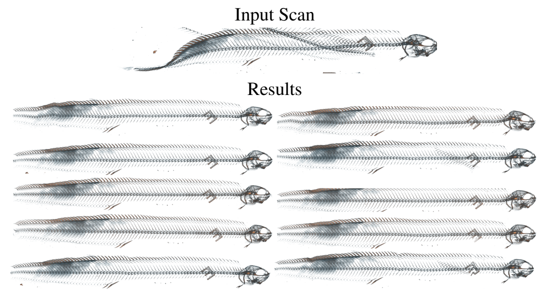

Figure 9. The same fish straightened by 10 different users.

Phase	Average Time	%
File Selection	0m19s	5.3%
Data Loading and Contour Tree	1m27s	24.4%
TopoAngler Segmentation	0m29s	8.1%
Mesh Extraction	0m03s	0.1%
Endpoint Selection	0m07s	1.9%
Straightening	3m30s	58.8%
User Interaction	4m27s	74.8%
Processing	1m31s	25.2%
Total	**5m57s**	100%

Table 1. Average time taken by one expert user took to straighten 15 fishes. The top part of the table shows the time taken in each phase as well as loading times between phases. The bottom part shows how much of the time was spent by the user and now much time the system spent performing computations.

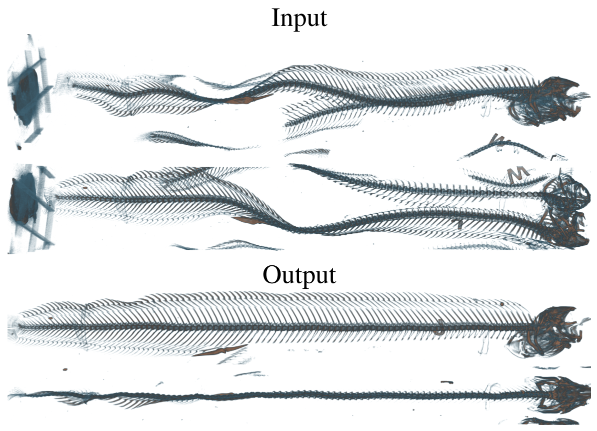

Figure 10. This scan was purposefully twisted to evaluate the performance of our system. In spite of the fact that the scan contains large twists as well as multiple fishes, our system produces a straightened volume for a single exemplar.

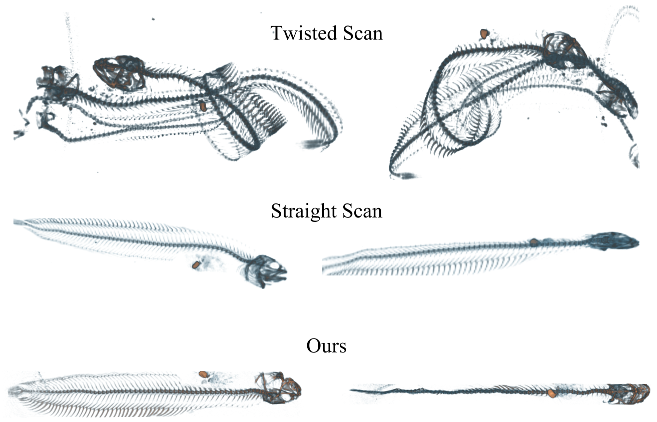

Figure 11. Here we scan a fish in both a twisted (top) and straightened (middle) pose. We compare the straightened scan with the scan produced by our method.

the ground truth volume) between the reconstructed volume and the ground truth.

Baseline Comparison.
The motivation for our project is the avoidance of the practical hurdle of scanning each fish individually in a CT scan, enabling to scan tens of fishes in the same session. Since our algorithm deforms the raw data acquired by the scanner, we want to evaluate the possible artefacts and errors introduced in the results. To quantify these effects, we performed the following experiment: we scanned a fish individually in a straight pose, and scanned the same fish in a bent pose when acquired together with other fishes. We then compared the raw straight scan, with the untwisted fish created with our algorithm. The two results are similar (Figure 11), with our result being more straight, since it is extremely challenging to physically straighten (and keep in a stable position) the fish for the entire duration of the scan. This result indicates that our software solution is an ideal replacement over individually scanning straight fishes, since it provides superior quality and massively reduces the scanning time.

User Experiment 1: Single CT dataset processed by multiple users.
To evaluate our system, we selected a representative CT scan, and processed it in 3 different ways: (1) An expert user used a combination of existing tools, using TopoAngler to segment and Blender to deform using FFD, (2) the same expert user processed the dataset with our algorithm, and (3) a group of 10 novice users (marine biologists that never used our system before) processed it using our system. The 10 users were asked to approximately match the result produced by the expert user using FFD.

We attach a video of (1) and (2) in the supplementary material, and we show the 10 straightened fishes by novice users in Figure 9. The expert user has ample experience with 3D modeling software and practiced the task for 3 times both for FFD and with our software. He spent 7 minutes and 36 seconds for FFD, and only 1 minute and 33 seconds with our system. The novice users were trained in a 10 minutes overview of the software, and then spent between 6 and 19 minutes, with an average of 12.5 minutes, to complete their task. This indicates that our system provides a massive speedup over baselines for expert users, while also allowing novice users to be productive in this task with minimal training. Note that for these experiments we only included the time required to define the deformation function, since it is the only fair timing that we can use to compare different methods. Loading times, export times, and volumetric surface extraction were not included since they were done with different software stacks.

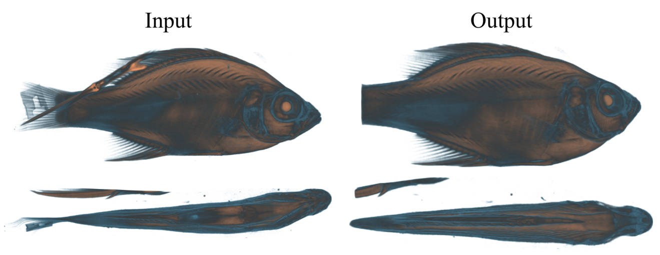

Figure 12. Our system can handle a wide variety of fish geometries.

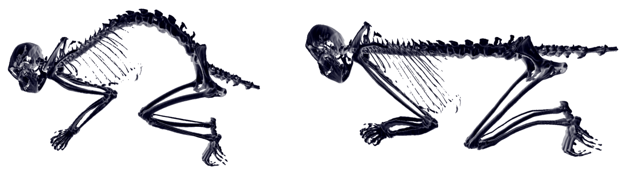
Figure 13. Example where the spine of a primate was straightened.

User Experiment 2: Untwisting a tank of fishes
Our system is interactive: The large majority of the time is spent in the segmentation phase, and in the generation of the tetrahedral mesh and solution of the linear system to compute the skeleton. To quantify the timings of the different phases we asked one marine biologists to process a large CT volume containing 18 fishes. A breakdown of the timings is provided in Table 1. The overall (user + processing) average processing time per fish is around 6 minutes. These timings include the entire processing, and to put them in context with the acquisition pipeline it takes an experienced biologist 30 minutes to prepare, label, photograph, and pack the tube for scanning, and another 4-12 hours for the actual scan to complete. Our system thus adds a minor overhead to the overall acquisition pipeline.

Showcase of Results.
Figures 10 and 12 demonstrate the capabilities of our system under a variety of challenging situations.

Application of Unwind on species beyond fishes.
Note that the initial target application of Unwind was for straightening fishes. More than 40 people in the CT scanning community have so far been trained on using Unwind at the FHL CT scanning facility. Currently, five of them are using it routinely and the data are being uploaded to MorphoSource.org as open access data sets. However, since we made our tool public, it has gained tremendous response from the community and has already been used to straighten and

deform other species of animals for scientific analysis. Specifically, it has been used to straighten and measure spines of small primates, snakes, eels, and the "flaps" of stingrays. Figure 13 shows an example where the spine of a primate (Chlorocebus Aethiops) was straightened.

Expert Feedback

The overall feedback we received from our collaborators was positive with them finding the software useful, intuitive and easy to use compared to existing off-the-shelves softwares. In fact, it took one of our collaborators only "*5 to 8 minutes*" to explain the working of the tool to a student, who was then able to use it independently. This has now become a staple application for them and has been integrated as part of their daily workflow.

One of the primary advantages they found with straightening fish was when doing morphometrics analysis, especially when trying to look at the fish in a specific anatomical plane. As one of our collaborators mentions: "*Before, we had to have versions of the scan rotated at different angles to measure things in a somewhat straight line. Now we can just straighten a single scan which saves a lot of time, storage space, and confusion with different versions of files*". Our collaborator went on add that in addition to the analysis, our tool also significantly helps save time when preparing results for communication as the following quote testifies: "*it makes creating figures a lot easier. We don't have to spend hours looking for the perfect scan, we can just fix any one we have*".

However there were some difficulties that were discovered as the software was being used. Cases were discovered where there was a significant gap between parts of the fish, due to which the dilation performed was not sufficient to create a single connected component. Such cases are possible if the skeleton of a fish is not necessarily connected (i.e., they might be connected via other tissues that have lower density than the bone), of the scanned fish is broken. We then had to then tweak both our interface, as well as the way the central axis of the fish is computed to handle such scenarios (see Section 5.1). There were also some interface requests, in particular, regarding the supported file formats. Currently, our software supports importing data from a stack of .bmp files, which is the output from the CT scanning software, and exports the straightened fish as a raw binary image. Given that users can also have processed data in other formats (tiff stack, DICOM stack, or nrrd files, etc.), having the having the ability to handle these formats will make it easier and faster to work with the tool.

Based on one of the suggestions for next steps, we will be looking into also doing the reverse—bend a fish to specific angles. This can be greatly useful for a variety of studies as well. For example, marine biologists also work with scans of fish with muscle fibers stained. In such cases, they would like to take a straight fish, and then see what the difference in muscle fiber length would look like in a bent fish.

We attach the unmodified transcript of our interview in the additional material.

LIMITATIONS AND CONCLUDING REMARKS

We introduced a novel system for supporting the creation of a large encyclopedia of CT scans of fishes, providing a user-assisted procedure to undo the unwanted deformation introduced in the scanning process. We demonstrated the utility of our system, evaluating quantitatively the error introduced in baseline comparisons, and qualitatively over real-world scans. The system is available as open source, and it is in active use in our collaborator's labs.

Limitations.

Our system has a high GPU memory requirement, which requires a high-end graphics workstation to run. Reducing this requirement is important to foster the applicability of this system and enable it to be used by the community at large. Another limitation of our system is that for exemplars with high distortion, the initial deformation estimation can be inaccurate, requiring more user interaction than for other exemplars (Figure 5). Much of the distortion is a result of the assumption that there is a straight line between the fish's snout and its tail. Fish skulls are highly complex and extremely diverse across species. One way to fix these large deformations would be to define the skull of the fish prior to the de-warping process. If we constrain slices between the front and back of the skull, it is likely to prevent most of the distortion and improve the efficiency of the workflow.

Future Work.

We believe that, after several months of usage, we will have access to sufficient data to replace the original deformation estimation with a data-driven model, trained on the data manually processed by the users. We are also porting our application to WebAssembly, to run directly in a web browser and making it easier to deploy in biological labs.

An additional, and surprising, application for this system is strategically bending fish that were scanned straight. Many scientists research fish muscle morphology and how local shape change during locomotion contributes to swimming kinematics. To do this, they stain specimens with iodine before scanning so that the muscle fibers are radio-opaque just like the skeleton. A modified version of Unwind could be developed to warp scanned fish to a shape they might attain during swimming, and look at the change in length of the muscle fibers.

ACKNOWLEDGMENTS

The authors would like to acknowledge Thomas O'Mahoney for the contribution of the primate scan in Figure 13. This work was partially supported by the Moore-Sloan Data Science Environment at NYU; NASA; the NSF CAREER award 1652515; NSF awards OAC-1835712, CHF-1908767, NSF IIS-1901091, CNS-1229185, CCF-1533564, CNS-1544753, CNS-1730396, CNS-1828576; CNS-1626098, a gift from Adobe Research; a gift from nTopology; a gift from AMD, and the NVIDIA NVAIL at NYU. APS was supported by the Seaver Institute and NSF DBI-1759637. C. T. Silva is partially supported by the DARPA D3M program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA. We would also like to acknowledge our trial users Kelly Diamond, Jonathan Huie, Karly Cohen, Todd Clardy, and Graham Short.

REFERENCES

[1] Alan H. Barr. 1984. Global and Local Deformations of Solid Primitives. *SIGGRAPH Computer Graphics* 18, 3 (Jan. 1984), 21–30. DOI: http://dx.doi.org/10.1145/964965.808573

[2] A. Bock, H. Doraiswamy, A. Summers, and C. Silva. 2018. TopoAngler: Interactive Topology-Based Extraction of Fishes. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2018), 812–821. DOI: http://dx.doi.org/10.1109/TVCG.2017.2743980

[3] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Levy. 2010. *Polygon Mesh Processing*. AK Peters.

[4] Marcel Campen, Cláudio T. Silva, and Denis Zorin. 2016. Bijective Maps from Simplicial Foliations. *ACM Transactions on Graphics* 35, 4, Article 74 (2016), 15 pages. DOI: http://dx.doi.org/10.1145/2897824.2925890

[5] H. Carr, J. Snoeyink, and U. Axen. 2003. Computing Contour Trees in All Dimensions. *Comput. Geom. Theory Appl.* 24, 2 (2003), 75–94.

[6] M. Chen, D. Silver, A. S. Winter, V. Singh, and N. Cornea. 2003. Spatial Transfer Functions: A Unified Approach to Specifying Deformation in Volume Modeling and Animation. In *Proceedings of the Workshop on Volume Graphics*. 35–44. DOI: http://dx.doi.org/10.1145/827051.827056

[7] Zhen Chen, Daniele Panozzo, and Jérémie Dumas. 2018. Half-Space Power Diagrams and Discrete Surface Offsets. *CoRR* abs/1804.08968 (2018). http://arxiv.org/abs/1804.08968

[8] Minsuk Choi, Cheonbok Park, Soyoung Yang, Yonggyu Kim, Jaegul Choo, and Sungsoo Ray Hong. 2019. AILA: Attentive Interactive Labeling Assistant for Document Classification Through Attention-Based Deep Neural Networks. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, New York, NY, USA, Article 230, 12 pages. DOI: http://dx.doi.org/10.1145/3290605.3300460

[9] S. Claici, M. Bessmeltsev, S. Schaefer, and J. Solomon. 2017. IsometryâĂŘAware Preconditioning for Mesh Parameterization. *Computer Graphics Forum* 36, 5 (2017), 37–47. DOI: http://dx.doi.org/10.1111/cgf.13243

[10] Sabine Coquillart. 1990. Extended Free-form Deformation: A Sculpturing Tool for 3D Geometric Modeling. *SIGGRAPH Computer Graphics* 24, 4 (1990), 187–196. DOI: http://dx.doi.org/10.1145/97880.97900

[11] C. Correa, D. Silver, and M. Chen. 2007. Illustrative Deformation for Data Exploration. *IEEE Transactions on Visualization and Computer Graphics* 13, 6 (2007), 1320–1327. DOI: http://dx.doi.org/10.1109/TVCG.2007.70565

[12] Carlos D Correa, Deborah Silver, and Min Chen. 2007. Volume Deformation via Scattered Data Interpolation.. In *Volume Graphics*. 91–98.

[13] Ryan Cross. New 3D scanning campaign will reveal 20,000 animals in stunning detail. (????).

[14] Mason Dean, Dan Huber, Brian Goo, Nicole Danos, Kenshu Shimada, and Adam Summers. 2012. On the jaws of lamniform sharks. *Integrative and Comparative Biology* 52 (2012).

[15] P. Degener, J. Meseth, and R. Klein. 2003. An Adaptable Surface Parameterization Method. In *Proceedings of the 12th International Meshing Roundtable*. 201–213.

[16] Xiao-Ming Fu and Yang Liu. 2016. Computing Inversion-free Mappings by Simplex Assembly. *ACM Transactions on Graphics* 35, 6, Article 216 (2016), 12 pages. DOI: http://dx.doi.org/10.1145/2980179.2980231

[17] Xiao-Ming Fu, Yang Liu, and Baining Guo. 2015. Computing Locally Injective Mappings by Advanced MIPS. *ACM Transactions on Graphics* 34, 4, Article 71 (2015), 12 pages.

[18] Xifeng Gao, Tobias Martin, Sai Deng, Elaine Cohen, Zhigang Deng, and Guoning Chen. 2016. Structured volume decomposition via generalized sweeping. *IEEE Transactions on Visualization and Computer Graphics* 22, 7 (2016), 1899–1911.

[19] K. C. Hall, P J. Hundt, J. D. Swenson, A. P. Summers, and K. D. Crow. 2018. The evolution of underwater flight: The redistribution of pectoral fin rays, in manta rays and their relatives (Myliobatidae). *Journal of Morphology* 270, 8 (2018), 1155–1170. DOI: http://dx.doi.org/https://doi.org/10.1002/jmor.20837

[20] Sungsoo (Ray) Hong, Yea-Seul Kim, Jong-Chul Yoon, and Cecilia R. Aragon. 2014. Traffigram: Distortion for Clarification via Isochronal Cartography. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 907–916. DOI: http://dx.doi.org/10.1145/2556288.2557224

[21] K. Hormann and G. Greiner. 2000. MIPS: An Efficient Global Parametrization Method. In *Curve and Surface Design*. 153–162.

[22] Alec Jacobson, Zhigang Deng, Ladislav Kavan, and JP Lewis. 2014. Skinning: Real-time Shape Deformation. In *ACM SIGGRAPH 2014 Courses*.

[23] Christopher Johnson and Charles Hansen. 2004. *Visualization Handbook*.

[24] M. A. Kolmann, J. M. Huie, K. Evans, and A. P. Summers. 2018. Specialized specialists and the narrow niche fallacy: a tale of scale-feeding fishes. *Royal Society Open Science* (2018). DOI: http://dx.doi.org/10.1098/rsos.171581

[25] Shahar Z. Kovalsky, Noam Aigerman, Ronen Basri, and Yaron Lipman. 2015. Large-scale Bounded Distortion Mappings. *ACM Trans. Graph.* 34, 6, Article 191 (Oct. 2015), 10 pages. DOI: http://dx.doi.org/10.1145/2816795.2818098

[26] Shahar Z. Kovalsky, Meirav Galun, and Yaron Lipman. 2016. Accelerated Quadratic Proxy for Geometric Optimization. *ACM Transactions on Graphics* 35, 4, Article 134 (2016), 11 pages. DOI: http://dx.doi.org/10.1145/2897824.2925920

[27] Soonhyeon Kwon, Younguk Kim, Kihyuk Kim, and Sungkil Lee. 2018. Heterogeneous volume deformation and animation authoring with density-aware moving least squares. *Computer Animation and Virtual Worlds* 29, 1 (2018). https://onlinelibrary.wiley.com/doi/abs/10.1002/cav.1784

[28] François Labelle and Jonathan Richard Shewchuk. 2007. Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. In *ACM Transactions on Graphics*, Vol. 26. 57.

[29] Wilmot Li, Lincoln Ritter, Maneesh Agrawala, Brian Curless, and David Salesin. 2007b. Interactive Cutaway Illustrations of Complex 3D Models. *ACM Transactions on Graphics* 26, 3, Article 31 (July 2007). DOI: http://dx.doi.org/10.1145/1276377.1276416

[30] Xin Li, Xiaohu Guo, Hongyu Wang, Ying He, Xianfeng Gu, and Hong Qin. 2007a. Harmonic Volumetric Mapping for Solid Modeling Applications. In *Proceedings of the ACM Symposium on Solid and Physical Modeling*. 109–120. DOI: http://dx.doi.org/10.1145/1236246.1236263

[31] Yaron Lipman. 2012. Bounded Distortion Mapping Spaces for Triangular Meshes. *ACM Transactions on Graphics* 31, 4 (2012), 108:1–108:13.

[32] Shixia Liu, Weiwei Cui, Yingcai Wu, and Mengchen Liu. 2014. A survey on information visualization: recent advances and challenges. *The Visual Computer* 30, 12 (2014), 1373–1393.

[33] Marco Livesu, Marco Attene, Giuseppe Patané, and Michela Spagnuolo. 2017. Explicit cylindrical maps for general tubular shapes. *Computer-Aided Design* 90 (2017), 27–36. DOI:http://dx.doi.org/https://doi.org/10.1016/j.cad.2017.05.002

[34] U. Meier, O. López, C. Monserrat, M. C. Juan, and M. Alcañiz. 2005. Real-time Deformable Models for Surgery Simulation: A Survey. *Computer Methods and Programs in Biomedicine* 77, 3 (2005), 183–197. DOI: http://dx.doi.org/10.1016/j.cmpb.2004.11.002

[35] Jörg Mensmann, Timo Ropinski, and Klaus Hinrichs. 2008. Interactive Cutting Operations for Generating Anatomical Illustrations from Volumetric Data Sets. *Journal of WSCG* 16, 2 (2008), 89–96.

[36] M. Nakao, K. W. C. Hung, S. Yano, K. Yoshimura, and K. Minato. 2010. Adaptive proxy geometry for direct volume manipulation. In *2010 IEEE Pacific Visualization Symposium*. 161–168. DOI: http://dx.doi.org/10.1109/PACIFICVIS.2010.5429597

[37] Megumi Nakao, Yuya Oda, Kojiro Taura, and Kotaro Minato. 2014. Direct volume manipulation for visualizing intraoperative liver resection process. *Computer Methods and Programs in Biomedicine* 113, 3 (2014), 725–735. DOI:http://dx.doi.org/https://doi.org/10.1016/j.cmpb.2013.12.004

[38] Andrew Nealen, Matthias Mueller, Richard Keiser, Eddy Boxerman, and Mark Carlson. 2006. Physically Based Deformable Models in Computer Graphics. *Computer Graphics Forum* (2006). DOI: http://dx.doi.org/10.1111/j.1467-8659.2006.01000.x

[39] Jorge Piazentin Ono, Arvi Gjoka, Justin Salamon, Carlos Dietrich, and Claudio T. Silva. 2019. HistoryTracker: Minimizing Human Interactions in Baseball Game Annotation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, New York, NY, USA, Article 63, 12 pages. DOI: http://dx.doi.org/10.1145/3290605.3300293

[40] Roi Poranne, Marco Tarini, Sandro Huber, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Autocuts: Simultaneous Distortion and Cut Optimization for UV Mapping. *ACM Transactions on Graphics* 36, 6, Article 215 (2017), 11 pages. DOI: http://dx.doi.org/10.1145/3130800.3130845

[41] Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable Locally Injective Mappings. *ACM Transactions on Graphics* 36, 2, Article 16 (2017), 16 pages. DOI: http://dx.doi.org/10.1145/2983621

[42] Teseo Schneider and Kai Hormann. 2015. Smooth bijective maps between arbitrary planar polygons. *Computer Aided Geometric Design* 35–36, C (2015), 243–354. Proceedings of GMP.

[43] Christian Schüller, Ladislav Kavan, Daniele Panozzo, and Olga Sorkine-Hornung. 2013. Locally Injective Mappings. In *Proceedings of the Symposium on Geometry Processing*. 125–135. DOI: http://dx.doi.org/10.1111/cgf.12179

[44] Thomas W. Sederberg and Scott R. Parry. 1986. Free-form Deformation of Solid Geometric Models. *SIGGRAPH Computer Graphics* 20, 4 (1986), 151–160. DOI:http://dx.doi.org/10.1145/15886.15903

[45] Anna Shtengel, Roi Poranne, Olga Sorkine-Hornung, Shahar Z. Kovalsky, and Yaron Lipman. 2017. Geometric Optimization via Composite Majorization. *ACM Transactions on Graphics* 36, 4, Article 38 (2017), 11 pages. DOI: http://dx.doi.org/10.1145/3072959.3073618

[46] Eftychios Sifakis and Jernej Barbic. 2012. FEM Simulation of 3D Deformable Solids: A Practitioner's Guide to Theory, Discretization and Model Reduction. In *ACM SIGGRAPH 2012 Courses*. Article 20, 50 pages. http://doi.acm.org/10.1145/2343483.2343501

[47] Jason Smith and Scott Schaefer. 2015. Bijective Parameterization with Free Boundaries. *ACM Transactions on Graphics* 34, 4, Article 70 (2015), 9 pages.

[48] Olga Sorkine and Marc Alexa. 2007. As-Rigid-As-Possible Surface Modeling. In *Proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*. 109–116.

[49] M. R. Stocker, S. J. Nesbitt, B. T. Kligman, D. J. Paluh, A. D. Marsh, D. C. Blackburn, and Parker W. G. 2019. The earliest equatorial record of frogs from the Late Triassic of Arizona. *Biology Letters* (2019).

[50] Guo-Dao Sun, Ying-Cai Wu, Rong-Hua Liang, and Shi-Xia Liu. 2013. A survey of visual analytics techniques and applications: State-of-the-art research and future challenges. *Journal of Computer Science and Technology* 28, 5 (2013), 852–867.

[51] Andrea Tagliasacchi, Thomas Delame, Michela Spagnuolo, Nina Amenta, and Alexandru Telea. 2016. 3D Skeletons: A State-of-the-Art Report. *Computer Graphics Forum* 35, 2 (2016), 573–597. DOI: http://dx.doi.org/10.1111/cgf.12865

[52] Simon J. Walton and Mark W. Jones. 2006. Volume Wires: A Framework for Empirical Non-linear Deformation of Volumetric Datasets. *Journal of WSCG* 14, 1–3 (2006), 81–88.

[53] Yalin Wang, Xianfeng Gu, and Shing-Tung Yau. 2003. Volumetric harmonic map. *Communications in Information & Systems* 3, 3 (2003), 191–202.

[54] Gregory Watkins-Colwell, Kevin Love, Zachary Randall, Doug Boyer, Julie Winchester, Edward Stanley, and David Blackburn. 2018. The Walking Dead: Status Report, Data Workflow and Best Practices of the oVert Thematic Collections Network. *Biodiversity Information Science and Standards* 2 (2018). DOI: http://dx.doi.org/10.3897/biss.2.26078

[55] Andrew Witkin. 1997. Physically Based Modeling: Principles and Practice Particle System Dynamics. *ACM Siggraph Course* (1997).