TopoAngler: Interactive Topology-based Extraction of Fishes

Alexander Bock *Member, IEEE*, Harish Doraiswamy *Member, IEEE*,
Adam Summers, and Cláudio Silva *Fellow, IEEE*

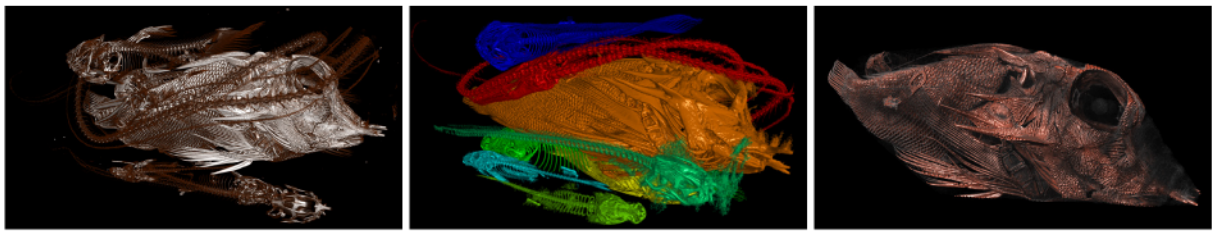


Fig. 1. Utilizing a Micro-CT scan containing nine distinct fishes (left) in a topological analysis results in a number of candidate features that correspond to entire or partial fishes (center). By enabling the user to select candidate features at different simplification levels, individual fishes can be interactively segmented (right).

Abstract—We present TopoAngler, a visualization framework that enables an interactive user-guided segmentation of fishes contained in a micro-CT scan. The inherent noise in the CT scan coupled with the often disconnected (and sometimes broken) skeletal structure of fishes makes an automatic segmentation of the volume impractical. To overcome this, our framework combines techniques from computational topology with an interactive visual interface, enabling the human-in-the-loop to effectively extract fishes from the volume. In the first step, the join tree of the input is used to create a hierarchical segmentation of the volume. Through the use of linked views, the visual interface then allows users to interactively explore this hierarchy, and gather parts of individual fishes into a coherent sub-volume, thus reconstructing entire fishes. Our framework was primarily developed for its application to CT scans of fishes, generated as part of the ScanAllFish project, through close collaboration with their lead scientist. However, we expect it to also be applicable in other biological applications where a single dataset contains multiple specimen; a common routine that is now widely followed in laboratories to increase throughput of expensive CT scanners.

Index Terms—Computational topology, join trees, branch decomposition, hierarchical segmentation, interaction, visualization system

1 Introduction

The advances in non-invasive scanning techniques, such as Computed Tomography (CT), in the last decade have changed the trajectory and magnitude of scientific discovery immensely. The most obvious impacts have been achieved in the medical domain where large financial expenditures make the technology readily usable on a daily basis to diagnose and treat humans and improve their well-being. At the same time, other fields have also benefited from this technology, with marine biology, being one such example, as the focus of this work.

CT scanners are now exceedingly being used to determine the skeletal structure of a variety of fishes. The 3D data allows scientists a better understanding of relationships among skeletal elements, the degree of skeletal mineralization, and enable population wide studies that were previously impossible. Both are useful for inferring function – the former feeds directly into inverse dynamics models and the latter reflects usage patterns and material strength. Rendered as surfaces the skeleton can be 3D printed to make physical models of function or also be used as input into a mathematical model, either for finite element modeling or computational fluid dynamics. 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.

The *ScanAllFish* project is being undertaken with the goal of creating a freely available database of scans of all the world's 30,000 fish species. However, there were many challenges in setting up a workflow towards this goal as Micro-CT devices are expensive, and thus are not widely available. Moreover, performing even a single scan at a useful resolution can take as long as 12 hours. To overcome these shortcomings, scientists decided to scan as many fishes as possible simultaneously. Each fish is first tagged with a radiopaque letter (Figure 2(a)), and then combined in a cheese cloth to form a "burrito" of fishes (Figure 2(b)). The cheese cloth is chosen as it protects the fishes from damages during the scanning and keeps them moisturized. This collection is then scanned (Figure 2(c)) to obtain a volume composed of up to a dozen or more fishes (Figure 2(d)).

While this workflow makes efficient use of the scanners in terms of time and cost, it results in new challenges extracting individual specimens from the obtained CT data. In particular, the volume contains many specimens with arbitrary orientation, close proximity of skeletal elements, and similar radiopaquenesses. Furthermore, fins from one specimen may lie very close to, and even between elements of another specimen. Also, unlike the human anatomy, fish specimens come in diverse shapes and sizes, thus having a very little or no a priori information. All of this makes the use of automatic segmentation algorithms difficult for such data [43, 45]. Thus, the current method employed by the scientists is to manually slice through the data to specify and extract the boundaries of sub-volumes that contain all of the skeletons of interest. Since these sub-volumes inevitably contain pieces of other fishes as well, the specimen of interest is extracted by using existing software that allows manual segmentation of the volume. Using this process, it takes scientists roughly 20 to 40 hours to fully isolate all fishes from a single scan.

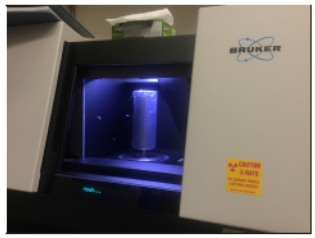
Skeletal regions, being almost opaque to the scanner, are high density regions of the volume. However, due to noise and missing connectivity of skeletons of individual fishes, using a single threshold does not provide a meaningful segmentation. We approach the challenge of

- *Alexander Bock, Harish Doraiswamy, and Cláudio Silva are with New York University. E-mail: {alexander.bock, harishd, csilva }@nyu.edu.*
- *Adam Summers is with the University of Washington. E-mail: fishguy@uw.edu.*

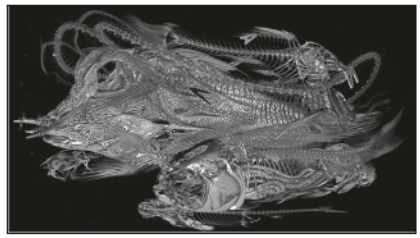
Manuscript received 31 Mar. 2017; accepted 1 Aug. 2017.
Date of publication 28 Aug. 2017; date of current version 1 Oct. 2017.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2017.2743980

(a) A fish prior to scanning with a labelling letter.

(b) A fish "burrito" containing up to a dozen fishes.



(c) Scanning of the fishes in a Micro-CT scanner



(d) The result is a single volumetric dataset containing all fishes.

Fig. 2. An overview of the stages for each fish until it is represented in a volume. Each fish is labeled with a radiopaque letter (a) which is used to later identify the fish in the scan. The entire dataset is wrapped in cheese cloth to keep the fish moist (b), before it is scanned in a Micro-CT scanner (c). Figure (d) shows a volume rendering of the resulting dataset where it is near impossible to separate the individual fishes.

capturing the segmentation with varying thresholds by using ideas from computational topology through the use of a join tree [4], which is a topological data structure that captures the merging of features across resolutions. The hierarchical segmentation provided by the join tree forms the backbone of our solution, but it is not sufficient as the join tree, while capable of detecting features of interest, does not reliably work in an unsupervised setting, thus necessitating user interaction.

Contributions. We present our application *TopoAngler* that combines techniques from computational topology with a visual interface to enable a user-driven, interactive segmentation of complex volumetric datasets. In particular, the volume is first segmented into a collection of *candidate features* using the topology of the volume. Using interactive tools, the user combines these candidate features into *meta features* that correspond to entire fishes, analogous to how constructive solid geometry approaches the construction of solid objects. Utilizing a human-in-the-loop approach is vital as the pattern recognition capabilities of the human coupled with the automatically generated suggestions of candidate features provides an optimal combination to effectively enable the extraction of fishes by selecting appropriate candidate features and removing inappropriate features from various levels of detail. Through an iterative design process with the *ScanAllFish* project, *TopoAngler* was designed to support their goal of providing a vast source of free datasets to the public and scientific community.
To summarize, our contributions are as follows:

- An interactive, human-in-the-loop approach for segmenting meta features of interest designed in close collaboration with the *ScanAllFish* project;
- Utilizing topological analysis to create candidate features;
- An intuitive interface to enable the selection of candidate features and creation of meta-features corresponding to entire fishes;
- The ability to export all selected meta-features for further analysis, and to make available to the wider research community;
- We present case studies that support the usefulness of our new approach. The domain scientists are able to segment the fishes in 15-20 minutes versus the 20-40 hours it took them. This leads to a two orders of magnitude improvement in their productivity.

Even though *TopoAngler* is tailored towards marine biology and the *ScanAllFish* project, we expect it to be useful other domains as well that require scanning and analyzing a large population of specimens but having scanning time as a bottleneck, for example laboratory studies where a large number of test animals have to be analyzed.

2 Related Work

Interactive Segmentation. Most of the interactive segmentation work has been performed in the field of medical visualization with the target of segmenting individual organs from human scans while assuming a priori information about the structure of the human body and segment clearly defined features. In the case of entire fish species, the internal structure changes drastically inside a single scan, and to the best of our knowledge, it is not possible to use existing techniques. Zhao and Xie provide an overview of interactive image segmentation in the medical field [47], while Olabarriaga and Smeulders provided the framework for interactive segmentations [32]. Early work on combining humans and algorithms for the use in interactive segmentations was performed by Höhne and Hanson [19] that enabled the user to segment volumes in a 3D view using morphological functions and providing immediate feedback. Freeborough et al. presented MIDAS, into which a variety of interactive segmentation algorithms can be implemented. They showed the usefulness of their system though a user study, which averaged around 10 minutes for a brain identification [15]. Nyström et al. presented BoneSplit, an interactive bone segmentation tool that utilizes a random walk algorithm that segments bones, which are initially picked by the user through the use of texture painting [31]. Outside the field of medical visualization, Protiere and Sapiro presented a system for the interactive segmentation of images through a sketch-based interface which they used successfully to extract features from images [37]. Nguyen et al. presented an interactive segmentation system that is based on convex active contours and also uses a sketch-based interface in order for the user to show segmentation intent [30]. Zhang and Ji utilized a Baysian Network model to interpret the user interaction and augment and improve the results of an initial automatic segmentation of images [46]. Bergo et al. presented an automatic segmentation based on analyzing the connectivity of image pixels [2]. Prassni et al. proposed an uncertainty-aware segmentation of medical data by assuming that the volume is composed of homogeneous regions [36]. Zhou et al. proposed a parallel mean shift algorithm based on path transmission, also for medical volume data segmentation [48]. Jeong et al. used an active ribbon formulation for neural processes to segment EM datasets [21]. Iassonov et al. survey the list of segmentation techniques used on CT scans of porous materials [20]. These techniques assume that the objects being segmented have uniform shapes (e.g., glass beads).

Topology. Topological data structures are naturally suited to capture interesting regions from scalar functions defined on volumetric meshes. It is therefore unsurprising that they have been used in several applications covering different scientific domains. In particular, two data structures are popular due to their efficiency – Morse-Smale complex [13], and contour trees [4], which is the basis for this work. Morse decompositions and Morse-Smale complexes have been used to identify features in several applications involving volumetric domains [3,23,38]. Merge trees and contour trees, being the more efficient one of the two structures, have been extensively used in several applications, including for designing feature-based transfer functions for volume rendering [9, 41, 44], topological simplification [5, 6], track burning cells during turbulent combustion [35], and identifying cloud systems [10]. The generality of this data structure has also enabled its use to identify interesting features in domains such as urban computing [8, 27].

Applications. A large amount of research has been undertaken in the field of interactive applications, a summary of which is outside the scope of this work. Sun et al. presented a thorough overview of techniques and applications in the field of Visual Analytics [40]. Liu et al. provided a similar survey on Information Visualization techniques [25]. Furthermore, there is a large body of research when designing applications with a human-in-the-loop paradigm. Munzner presents a four

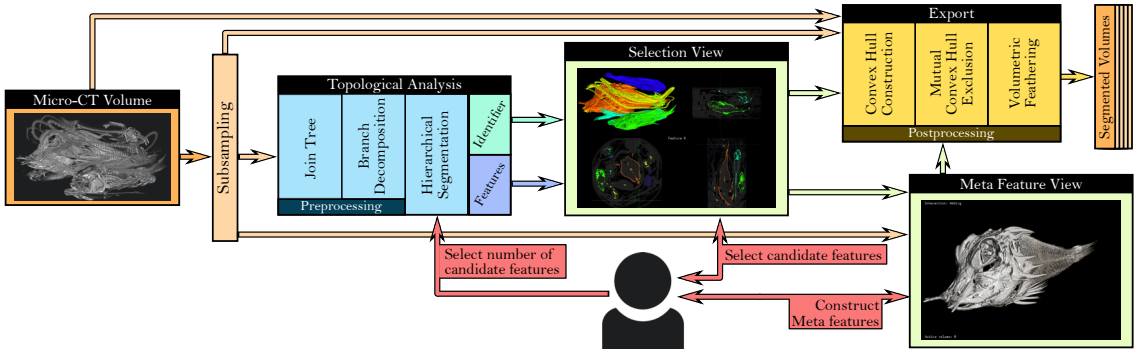


Fig. 3. A schematic overview of the system and its individual parts. After loading and subsampling the original Micro-CT volume, a topological analysis results in a set of voxel identifiers and feature identifiers. These are used to highlight individual candidate features. The user selects features of different simplification levels to construct meta features that are exported as a postprocessing step.

layer nested model containing iteratize loops [29]. Tory and Möller describe useful approaches for application system construction that we followed in our design [42]. Kirby and Meyer provides a good description on intradisciplinary work [22]. Graham and Kennedy provided a system to evaluate data quality when analyzing species taxonomy data [16], which might prove useful to the analysis of fish species. Correa et al. presented a method for the interactive deformation of volumetric data for exploration purposes [7].

3 System Overview

In this section, we first describe the currently employed workflow of the *ScanAllFish* project as it pertains to the scanning and segmentation of fish populations. Then, we present our improved workflow supported by *TopoAngler*, a multi-view application enabling users to interactively segment individual fishes from a combined volume. Figure 3 provides an overview of the components and processing steps in our system.

3.1 Current Workflow

The goal of the *ScanAllFish* project is to acquire a publicly available database of high-resolution CT scans of at least one specimen for every fish species on Earth in order to stimulate research into comparative anatomical studies and to foster discoveries about fish anatomy. So far, only 600 out of approximately 30,000 species have been scanned due to bottlenecks caused by limitations in the acquisition method.

The currently employed workflow utilizes a Micro-CT scanner for image acquisition in order to determine radiodense skeletal tissues as well as soft tissue anatomy through the use of contrast techniques. However, the availability of the Micro-CT scanner is the limiting bottleneck since not only is the CT scanning equipment expensive, but obtaining scans at the required resolution is time consuming as well. Therefore, it is necessary to perform scans of a large number of specimens simultaneously and then digitally dissect the individual fishes, which is also a time consuming task when performed manually. The currently used application requires the user to manually place arbitrarily oriented bounding boxes in orthogonal slice views around individual fishes, the contents of which are then extracted into separate volumetric datasets. The ability to only export entire bounding boxes is a major drawback especially in cases where the specimen are wrapped around other fishes. For example, a bounding box surrounding the fish in Figure 4 would potentially contain the entire dataset and the segmentation would thus be useless. Second, only providing orthogonal slices does not convey enough spatial information about the dataset to a novice user and thus increases the segmentation time even further.

3.2 Proposed Workflow

Inspired by the scientists' current workflow, we propose *TopoAngler*, a visualization system that was designed and developed in close collaboration with the *ScanAllFish* project in order to improve the segmentation throughput. Instead of forcing the user to detect fishes in the dataset and manually construct bounding boxes, our system makes use of a hierarchical segmentation to find features of interest in the dataset. A feature is a connected sub-volume resulting from the topological analysis of the dataset, and can vary in level-of-detail from representing individual bones, rib cages, or even entire fishes. The user can then interactively select a level-of-detail of interest and assemble *meta features*, e.g., entire fishes, from candidate features of different levels-of-detail. At the end, the user chooses whether they want to export only the selected features, or export the entire convex hull of the selected features. In either case, only a tight fitting bounding box around the features or convex hull, excluding other selected fishes, is exported.

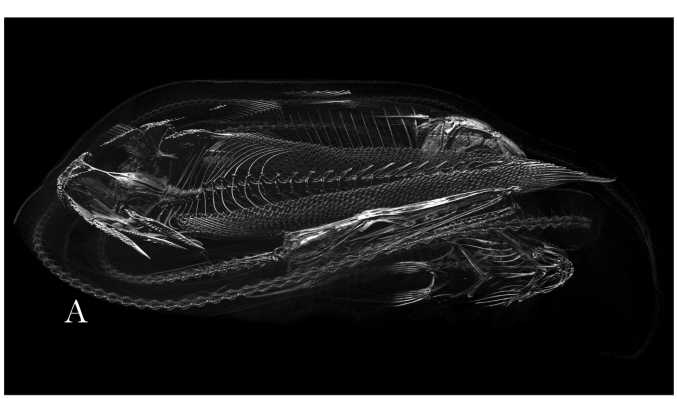


Fig. 4. An example (A) where a bounding box selection fails as other specimen are contained in the convex hull. These cases require a mutual convex exclusion, where the convex hulls of other objects are removed.

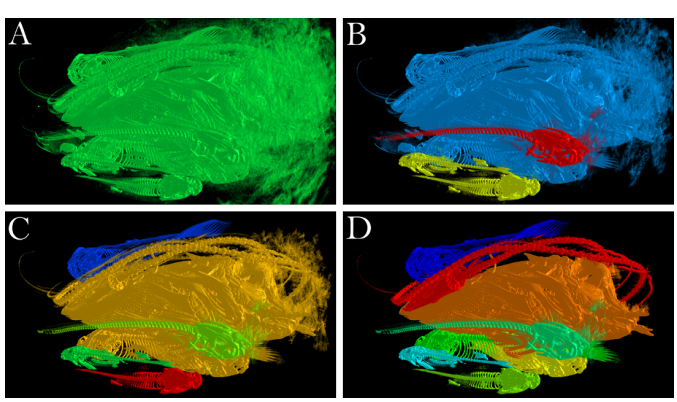
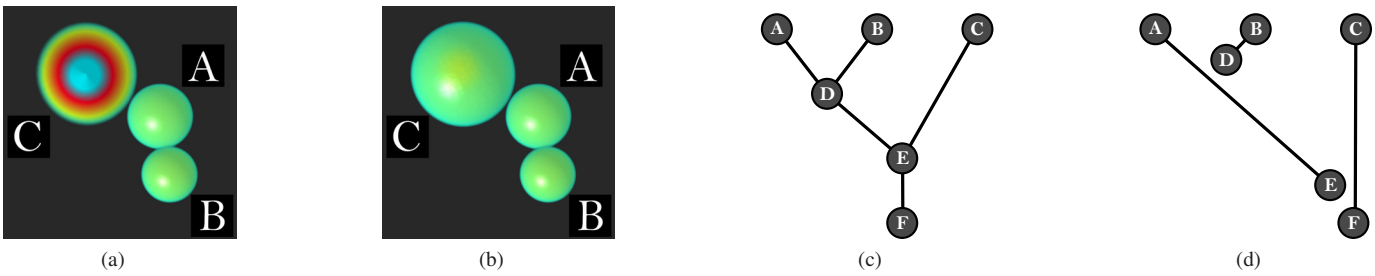


Fig. 5. The impact of the user changing the number of desired candidate features. Starting from a single feature (A), to three features (B), five features (C), before selecting one feature for each fish in the dataset (D).



(a) (b) (c) (d)

Fig. 6. **(a)** An example scalar function defined on a 3D volume with three maxima labeled A, B, and C. The function uniformly decreases with distance from the maxima. **(b)** A level set at a given real value corresponds to the shown spherical surfaces. A super-level set corresponds to the region inside the three spheres. **(c)** The join tree of the scalar function. **(d)** A branch decomposition of the join tree.

The workflow as presented in Figure 3 functions as follows. After loading the volumetric dataset containing multiple fishes, the join tree of the volume is computed in a pre-processing stage. The join tree encodes information about how the high density regions varies in this volume over multiple scales, thus providing an evolution hierarchy, which is utilized by the user interaction stage. In this stage, the user interactively selects the desired hierarchy, which is specified as the number of candidate features to extract based on the join tree hierarchy. Here, a low number of features indicates a high level of the hierarchy and thus coarser features. Figure 5 shows candidate features from a real world use case. All candidate features are then displayed as a collection of coordinated views, using two volumetric rendering views and three axis-aligned slice views. Here, the user selects individual features and combines them into *meta-features* that correspond to entire fishes. Adding the ability to add (remove) candidate features spanning different levels in the hierarchy to (from) these meta-features enables the user to start with a coarse selection of a specimen from a higher hierarchy level and then add or remove smaller features later. All constructed meta-features can be exported at the end of the user session once the user is satisfied with the segmentation. In order words the user interactively selects a set of *candidate features* $\mathscr{F}_i$ for each *meta-feature* $\mathscr{M}_j$. Given a function V that returns the voxels covered by a candidate feature or meta feature respectively, it holds that $V\left(\mathscr{M}_j\right) = \bigcup_i V\left(\mathscr{F}_i\right)$.

The following two sections describe the individual steps and our design decisions. First, Section 4 describes the topological concepts and the algorithm to compute the join tree and how it is utilized to generate a hierarchy of candidate features. Then, Section 5 explains our proposed system setup, the visualization and interaction methods involved in the steps, and how these components were designed to support the interactive segmentation of the volumetric datasets.

4 Hierarchical Segmentation

In order to achieve the goal of extracting entire fishes from the volume, we need to obtain a segmentation of the input volume such that each sub-volume corresponds to at most a single fish. This allows the domain scientist to select and group segments corresponding to the same fish. At the same time, we also want to minimize the amount of work done by the scientist by obtaining a fish with a small number of interactions.

In this section, we describe our approach achieving this through the use of concepts from computational topology. For completeness, we first introduce the mathematical concepts and briefly describe the techniques used before describing the application of these concepts in *TopoAngler*. We refer the reader to the following textbooks [12, 18, 26] for comprehensive discussions on these concepts.

4.1 Data Representation

The Micro-CT data provides density values at points over a uniform 3-dimensional grid. Formally, this data is represented as a *scalar function*, $f : \mathbb{R}^3 \to \mathbb{R}$, that maps points in the spatial domain, $\mathbb{R}^3$, to the set of real values $\mathbb{R}$. Figure 6(a) shows the volume rendering of an example scalar function defined on $\mathbb{R}^3$ using a transfer function.

For computational purposes, $\mathbb{R}^3$ is often discretized as a collection of tetrahedra. Given the CT data as a 3D structured grid, it is first converted into a tetrahedral mesh, K, using the Freudenthal's triangulation [28]. The scalar function is then represented as a piecewise linear (PL) function $f : K \to \mathbb{R}$, where the function is specified on the vertices of K (which correspond to the vertices of the 3D grid), and the function value at a point within a tetrahedron is computed using linear interpolation. Other works have shown alternative representations when this piece-wise linear assumption does not apply [24].

4.2 Level Set Topology

Points in the data corresponding to the skeleton of fishes, being opaque to the CT scanner, have a high function value. The collection of such locations having a high function value is mathematically captured as a *super-level set*. Formally, the preimage $f^{-1}(a)$ of a real value a of a scalar function f is called a *level set*, which is the set of all points on the domain having a function value equal to a. A *super-level set* of a real value a is defined as the preimage of the interval $[a, \inf)$. It is the collection of level sets $\{f^{-1}(x), \forall x \geq a\}$ — the set of all points having function value greater than or equal to a. The surface of the three spheres in Figure 6(b) denotes the level set for a given value, while the region enclosed by the spheres denotes the super-level set.

Join trees. Consider a sweep of the input function f in decreasing order of function value. The nature of topological change to the super-level sets of f when the sweep encounters a vertex determines its type. We are specifically interested in the following characterization of a vertex based on the change in the number of connected components:

- *regular*: The number of super-level set components do not change.
- *maximum*: A new super-level set component is created. The function shown in Figure 6(a) has three maxima A, B, and C.
- *join saddle*: Two super-level set components merge. Point D in Figure 6(c) represents one such saddle where the super-level set components created at maxima A and B merge into one.

The *join tree* tracks such changes in the connectivity of super-level sets of the input scalar function [4]. Nodes of the join tree correspond to the set of maxima and join saddles of f. Figure 6(c) shows the join tree corresponding to the input function shown in Figure 6(a). A vertex that is not regular, such as a maximum or a saddle, is called *critical*.

Branch decomposition. A *branch* of a tree is a *maximal* monotone path in the tree (defined by the function values of the nodes of the tree). A branch decomposition is the partitioning of the tree into a collection of branches having the following properties [34]:

- Every edge of the tree is in exactly one branch.
- There is exactly 1 branch connecting 2 leaves.
- All other branches connect 1 leaf with an internal node (saddle)

This generates a parent-child relationship between branches and, in turn, a hierarchy that is created by removing a childless branch one at a time. One possible branch decomposition of the join tree is shown in Figure 6(d). Next, we show how we use this hierarchy to obtain a collection of segmentations that are then used for selection by the user.

4.3 Topology-based Segmentation

The input volume's hierarchical segmentation is achieved in two steps.

1. Computing the augmented join tree. The *augmented join tree* of a scalar function f is the join tree of f augmented with regular vertices. We now briefly describe the algorithm to compute the augmented join tree of a PL function defined on a tetrahedral mesh K [4].

The algorithm uses the *upper link* of the vertices to determine the connectivity between edges of the join tree. Here, the *link* of a vertex v is defined as the sub-graph induced by vertices adjacent to v (see Figure 7). The *upper link* is defined as the sub-graph induced by adjacent vertices having a function value greater than v. A consistent comparison between vertices is ensured by a simulated perturbation of the function [11], thus no two vertices have the same function value.

The algorithm first sorts the vertices of K by decreasing function value. Next, each vertex v in this sorted list is processed as follows;

1. If the upper link of v is empty, then create a new component containing v and set v as its head,
2. If the upper link of v is not empty, find the components that contain the vertices in the upper link of v. Add an edge between v and the head of each of the components. Next, merge these components and set v as the head of the merged component.

Regular vertices form degree-2 nodes in the augmented join tree. The join tree is then computed by merging (removing) the regular nodes from the augmented join tree. Given an input mesh with n vertices, the above algorithm computes the join tree in $O(n\log n + n\alpha(n))$ time, where α is the inverse Ackermann function.

Consider an edge (c_1, c_2) of the join tree between two critical points c_1 and c_2. Regular vertices of the augmented join tree that connect c_1 and c_2 form a cylinder [9] that is a connected sub-volume where the topology of the super-level sets does not change. The cylinders corresponding to all leaf edges of the join tree, that are incident on a maximum, form the base set of input segments. The three segments corresponding to the join tree in Figure 6 are shown in Figure 8.

2. Computing the hierarchical segmentation. The second step computes the branch decomposition of the join tree. Given an importance measure defined on an edge (c_1, c_2), the branch decomposition is computed as follows [34]. All leaf edges, that are incident on a maximum, are first added to a priority queue, prioritized based on the importance measure. Then, at each step, the smallest leaf edge is removed, which forms a branch. When such a removal leads to a degree-2 node, then the two edges incident on that node are merged to form a single edge and its priority is appropriately updated. This procedure is illustrated in Figure 8. Given a tree with t nodes, this process takes time $O(t\log t)$.

As mentioned earlier, a hierarchy is defined by removing childless branches one at a time. When a branch is removed, its cylinder is merged with that of its parent. This process of removing branches represents the *simplification* of the join tree. The segments corresponding to a given hierarchy level are then defined as the collection of cylinders corresponding to all leaf edges of the join tree remaining after the above removal. That is, the sub-volume corresponding to an edge in this simplified join tree is the union of cylinders corresponding to all the edges of the original join tree that make up this leaf edge. For example, the join tree in Figure 8 with no simplification results in three segments.

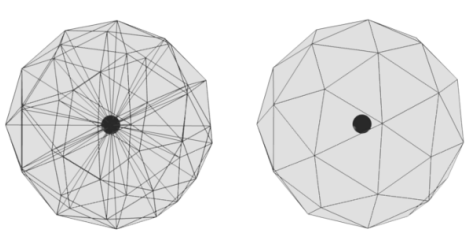


Fig. 7. Consider a vertex together with the tetrahedra incident on it (left). The link of the vertex is the mesh induced by vertices adjacent to it (right).

Removing the edge (B, D) results in two segments corresponding to edges (A, E) and (B, E). In this case, the segment for (A, E) is made up of cylinders of the join tree edges (A, D), (B, D), and (D, E).

Two common importance measures used in the above process are the function difference (i.e., $|f(c_1) - f(c_2)|$) and the hyper-volume [5], which is defined as the integral of the scalar function over the enclosed volume. The former results in a branch decomposition where the measure of the resulting branches is equal to the topological persistence [14, 33]. For this system, we tried both the measures, and found hyper-volume to provide a better segmentation for the CT data. This is because when two small parts of a fish's skeleton are disconnected, due to their high function value, they form two high persistent branches, and hence do not merge quickly in the hierarchy. On the other hand, since its volume is small, their hyper-volume is small, thus enabling an early merge of the two components. We therefore use hyper-volume as measure to obtain the segmentation.

5 TopoAngler Application

Our system for the segmentation of fishes consists of a multiple view setup in which the entire dataset is shown combined with a representation of the currently selected number of candidate features. Presenting these two datasets combined enables the user to spatially correlate interesting candidate features during the selection when constructing entire fishes from these individual features.

In order to improve the workflow utilized in segmenting individual volumes, we derived, together with the domain scientists from the *ScanAllFish* project, a list of requirements that the application needs to fulfill in order to be a substantial improvement over the previous workflow as described in Section 3.2:

- **R1** An intuitive interface for users of novice and intermediate level taking human factors for usability into account
- **R2** The ability to intuitively segment multiple specimen from a single volume
- **R3** The functionality to interactively correct and fine-tune selections
- **R4** Real-time feedback on the segmentation process
- **R5** The ability to export fishes into separate volumes for analysis

To address these requirements, our proposed system utilizes multiple linked views. Figure 9 shows the application in the middle of the segmentation of *Saurenchelys cognita*. In order to fulfill requirement **R1**, we designed the proposed system as an improvement of the previous tool, designing the system around the axis-aligned volume slices, rather than as a complete replacement. Furthermore, we added a feature selection method using a 3D volumetric rendering, as these visualizations tend to be easier to understand than mentally recombining orthogonal slices. By selecting individual candidate features in either the volumetric rendering or the slice rendering, the user can add or remove a

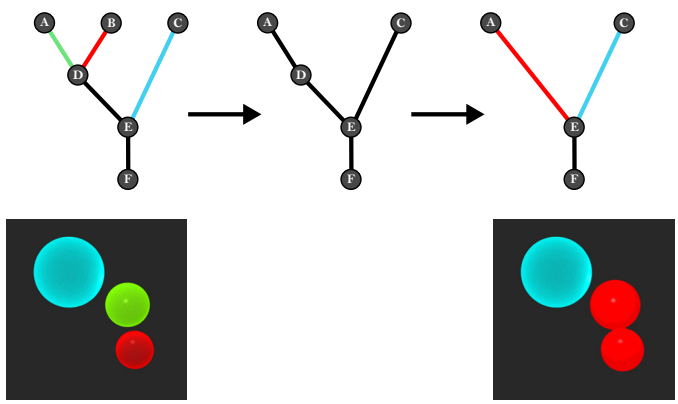


Fig. 8. A hierarchical segmentation of the volume is performed using the join tree. By increasing the simplification level (top), different branches of the tree are joined to form a simplified representation. This corresponds to the number of features that are shown in the rendering (bottom).

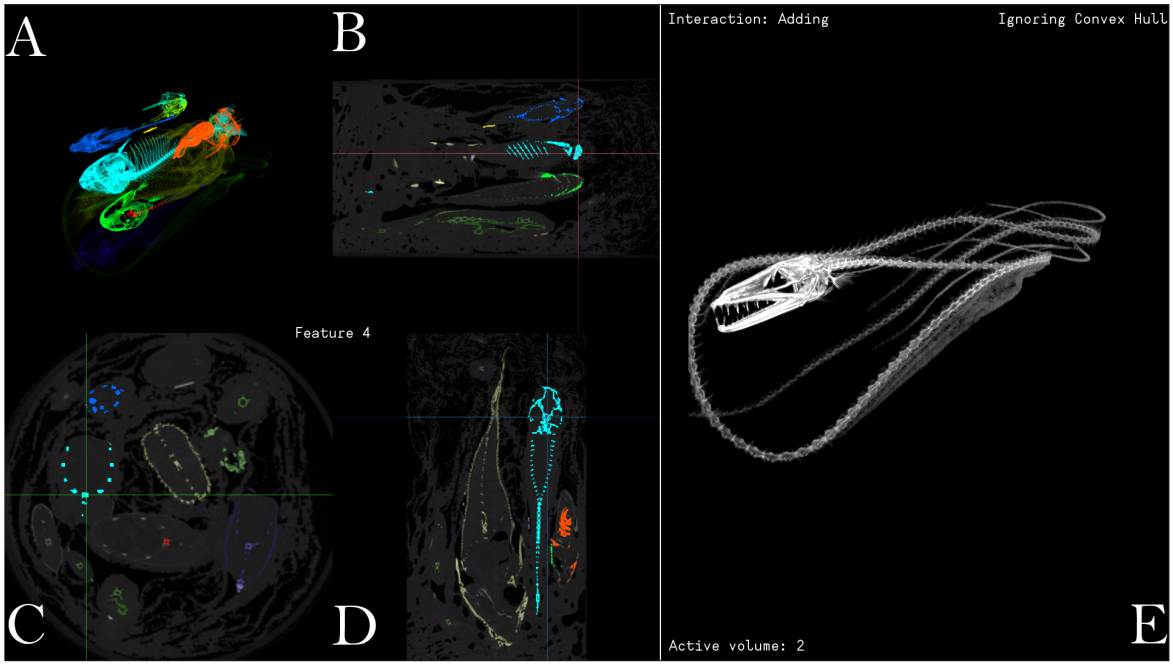


Fig. 9. An overview of the *TopoAngler* application while it is being used to segment *Saurenchelys cognita*. The *Selection View* (A – D) enables the user to select candidate features from the dataset. The *Meta Feature View* (E) shows the state of the current meta feature and enables the user to see which parts of the fish should be added next.

candidate feature, thus fulfilling **R2** and **R3**. For the purpose of implementing **R4**, the system provides feedback about the current state of the active meta feature segmentation in a separate view as a volumetric rendering. This view and the selection view are linked such that the user's spatial registration is not broken. Finally, for **R5**, the user has the ability to export a segmented fish and optionally everything inside the fish's convex hull using a tight fitting bounding box.

For efficiency, we perform operations on a scaled down version of the original volume for almost all operations. The scale factor can be chosen by the user, but for all discussions in this work, unless otherwise noted, we utilized a default scale factor of about $\frac{1}{8}$ in each dimension of the volume. Through our discussions, we found that users preferred the reduced loading and interaction time and that the accuracy of feature detection did not suffer greatly as all candidate features of interest consist of large blocks of voxels even in the smaller volumes. Only in the exporting step we use the original volume to extract the segmented meta features at their original resolution. Currently, this limits our application to volumes that fit the system's main memory and a reduced volume that fits the GPU's memory. This was not found to be a limiting factor for the domain experts, however, many out-of-core rendering techniques exist to address this issue shall the need arise.

The following sections address the individual components of our system. Before the acquired data can be used in the interactive interface, it is first pre-processed to compute the join tree and the corresponding cylinders as described in Section 4. Section 5.1 presents the results of using this data structure and how these results can be accessed in our system. Section 5.2 describes the multi-view visualization setup in the system, how they are used to convey the information about available candidate features and already constructed meta-features and also details all available interaction methods for selecting interesting candidate features interest and ultimately combining these into useful meta features. Finally, Section 5.3 describes all processing that occurs after the user has finished selecting all fishes in the dataset.

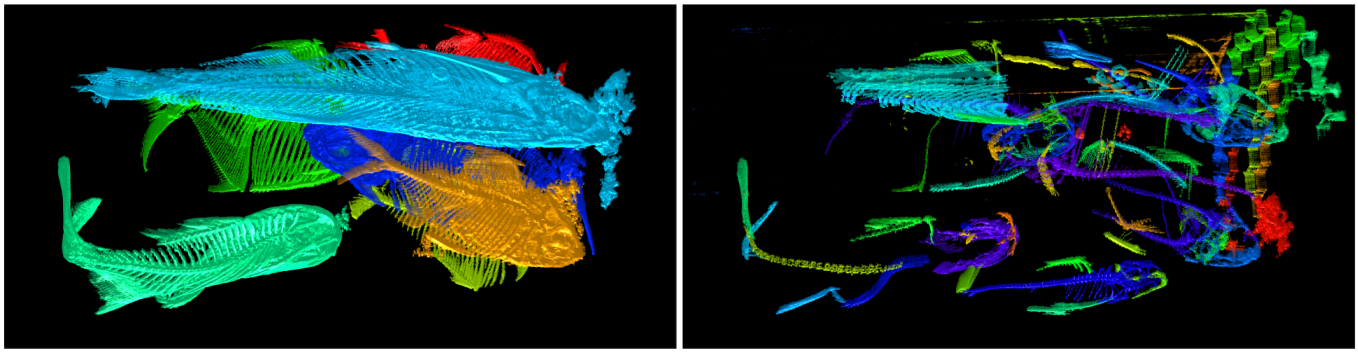
5.1 Hierarchical Segmentation

After computing the join tree (see Section 4), the user can interactively select a number n of candidate features. Only the n most prominent candidate features in the dataset will be shown and all remaining features in the data will automatically be removed in order to reduce the amount of visual clutter. By changing the granularity, and thus the number of returned candidate features, the user can select between exploring a small number of larger features, that most likely correspond to entire fishes or large parts of fishes, or a large number of smaller features, as it might be necessary to add or remove smaller sub-features to a fish. Figure 10 shows one of the use case datasets from Section 6 with two separate number of candidate features. Whereas the Figure 10(a) contains entire structures and makes it possible to select large parts of fishes easily, Figure 10(b) highlights smaller features inside fishes, thus enabling the detailed selection of individual parts. The user combines individual features of different scale into *meta features* in order to construct the fishes from the combined volume iteratively. It is these meta features that are the desired result that are the resulting product. The user starts with a small number of features that roughly correspond to entire fishes whose segmentation might have additional or missing structures. Then, for each fish, they modify the number of features interactively until a candidate feature is separated and it can be added or removed individually from the current fish. In practice we found that it is useful to start with coarse granularity with a lower feature number equal to the expected number of fishes to try obtaining the entire fish structure. This can then be tweaked for cases where the continuity assumption is broken, for example, when fishes have bones which are not connected to the rest of the skeleton or for tightly packed fishes.

Implementation Details. The preprocessing stage output is the join tree and an index volume. Each voxel of this volume contains a unique identifier indicating its respective join tree edge in the augmented join tree. For a specific granularity level, the join tree is simplified and returns a list of original edges (or indices) that corresponds to each edge of the simplified tree making up the individual features for this granularity level. The index volume and the indices are used as a mask to test each voxel's visibility and color coding.

5.2 Rendering

Figure 9 shows the five distinct subviews of the application's rendering component while it is in use to segment *Saurenchelys cognita*. The subviews are separated into two groups: the *Selection View* (A, B, C, D; Section 5.2.1) and the *Meta Feature View* (E; Section 5.2.2). The following subsections describe these two categories together with the respective interaction methods that are available to the user.



(a) Selecting a higher simplification level leads to larger features

(b) Selecting a lower simplification level leads to more detailed features

Fig. 10. Selecting a different number of candidate features leads to a change in granularity of detected features. Changing between different simplifications is required by the workflow in order to construct entire fishes with the necessary detail.

5.2.1 Selection View

The selection view component of our system consists of four subviews and is used to select individual candidate features at a selected simplification level to construct meta features. Using the feature information from the hierarchical segmentation, each view highlights voxels that belong to distinct candidate features. A color is assigned to each feature based on a user-changeable color map, which is used as an overlay in the rendering. For a low number of features, a linear sampling of the HSV color space proves sufficient, but for larger number of features a color map with higher perceptional distance is beneficial [17]. Features that are not used in any meta feature selection are shown in a fully saturated color in order to maximize the perceptual distance between the feature color and the gray-scale background of the underlying data. Features that have been used in any meta feature selection use a lower saturation in order to highlight that these features have been used.

It was a user decision to continue displaying these features so that they are able to remove features from the rendering view directly and, at the same time, maintain a spatial relationship between already selected and as-of-yet unselected candidate features. We now describe the rendering aspects of the *Selection View*, followed by describing the interaction methods that are available to the user in this view.

Rendering Components. The selection view consists of two types of rendering components, the *volume rendering* and an axis-aligned *slice view*. Both views can be used separately and play a mutually beneficial role to enable the user to select desired candidate features.

Volume Rendering. The volumetric rendering (A) of the system uses a standard transfer function-based volume raycasting approach. However, only the transparency channel of the transfer function is used in combination with the color mapping derived from the feature identifier as described above. This combination enables the user to modify the transparency of the volume ray casting by manipulating the transfer function, but still maintain the ability to discriminate between different candidate features through their color. The volume rendering component is useful in the early stages of the segmentation as it enables the user to easily segment large portions of the volume due to the increased spatial awareness compared to the slice view. In addition to the features, the currently segmented fish (see Section 5.2.2) is overlaid in order to provide the user with the necessary context to select new candidate features that are related to the meta feature that they are currently segmenting. This is useful especially when the user wants to add small features and when cropping the volume rendering, as this would otherwise lead to the spatial orientation becoming more difficult.

Slice View. In addition to the volumetric rendering, the application also provides three orthogonal slice views (B, C, and D) presenting the user with a familiar view of the data. The combination of volume rendering for an easy selection of large areas together with the slice views are useful for the user to be able to select small features which might be occluded in the volume rendering and might require precise interaction. Thus, it is possible for the user to scroll through the different slice views individually in order to find small features that they might want to add and selecting them at a higher accuracy.

Interaction. The interaction methods in the *Selection View*'s rendering components react to the component that was most recently used by the user. In both cases, the application presents the unique identifier of the feature that is selected with the mouse continuously. This feedback provides vital feedback to the user to determine, especially for small features, whether or not the mouse cursor is targeting a feature. Furthermore, the identifier is utilized by the user to distinguish between neighboring features that might have a similar but different color. This is useful in the case when using a low simplification with a large number of features at which the color separation between features becomes difficult. In our experiments this was not a problem as large parts of the volume were already selected by this step of the workflow and the remaining features were spatially well separated. Nevertheless, using the mouse and comparing the numerical feature identifier, the user has the ability to differentiate such features.

For the slice view, this value is retrieved from the location of the mouse cursor itself, which is further highlighted by a crosshair, even after the mouse cursor has been removed. This crosshair does not occlude the central part as the user will use this selection method in the cases where high precision is required to select small features, and an obtrusive crosshair would prevent this high precision interaction.

For the volumetric rendering, a probe ray is cast using the same transfer function as used for the rendering, and the identifier of the first feature that does not have full transparency and intersects this probe ray is used as the selected feature. We chose to use this method over the alternatives (such as toggling between the different features of a ray), as it was found to be the most intuitive to non-technical users that might not be familiar with volume raycasting techniques, and who would otherwise get confused. In the conducted user sessions, we found that being able to only select the first feature was never an issue as the users preferred to use the slice views for selections that required more precision. Furthermore, in order to provide the user with access to the occluded parts of the volume, the user has the ability to use clip planes to remove parts of the volume where desired.

After highlighting a feature with cursor, the user presses a button to add or remove the current feature to the meta feature. The currently edited meta feature is changed using the numeric keys, thus changing the volume that is visible in the *Meta Feature View* accordingly.

5.2.2 Meta Feature View

The *Meta Feature View* (E) presents a volume rendering of the currently selected meta feature such that the user receives immediate feedback. In order to maintain the user's spatial awareness, the same camera information is used in the two volume renderings (A and E) so that their views are always linked. However, the user can change their transfer functions individually if they want to highlight individual parts of the meta feature, for example to check if a specific element of the feature

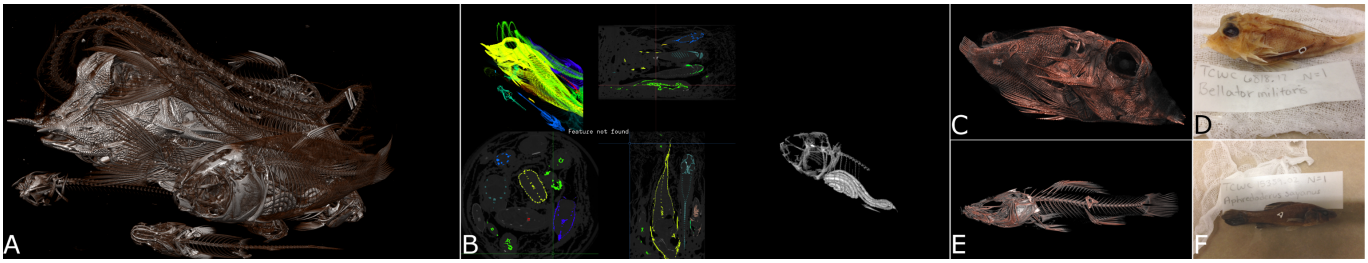


Fig. 11. This image shows results from the Group2 Scan use case. A shows a volume rendering of the entire scan, containing all nine fishes. The segmentation using the *TopoAngler* system (B) enables the user to segment the individual fishes, for example *Bellator militaris* (C,D) and *Aphredoderus Sayanus* (E,F).

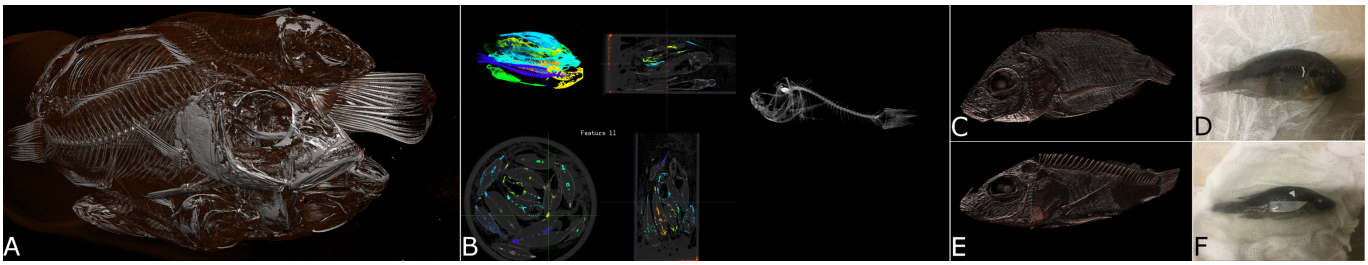


Fig. 12. This image shows the results of a segmentation with *TopoAngler* on the Cameroon Scan use case. A shows all ten fishes from the Cameroon crater lakes, B shows the system being using the segment the fishes, exemplified with *Pungu Maclareni* (C, D) and *Stomatepia Pindu* (C, D).

has been successfully added or whether a part of the volume is missing and must be iteratively edited. Furthermore, if a user chooses to employ a clip plane in the *Selection View* to cut parts of the volume, the same clip plane is applied to this view to maintain their linking.

5.3 Meta Feature Export

After the user selects all meta features of interest from the dataset, these meta features are exported. Adding to the notation introduced in Section 3.2, let C denote the convex hull of a candidate feature $\mathscr{F}_i$ or meta feature $\mathscr{M}_i$ respectively. For each meta feature $\mathscr{M}_i$, the user chooses whether the convex hull $\mathscr{C}_i = C(\mathscr{M}_i)$ or only the specific features which were selected ($\mathscr{M}_i$) are considered for exporting, the default being the entire convex hull. Exporting the convex hull of a fish makes it possible for the user to only select the surrounding bone structures and still capture all of the soft tissues of the fish, thus reducing the amount of detail the user needs to capture manually.

In our system, we make use of the popular quick hull algorithm [1] in order to construct the convex hull. For the case where the user selects the convex hull, we want to minimize the influence of other meta features $\mathscr{M}_j, j \neq i$, such that instead of saving $V(\mathscr{C}_i)$, we export $\mathscr{C}_i \setminus \bigcup_{j,j\neq i}(\mathscr{M}_j)$, thus removing all other meta features that lie within the selected meta feature. Then, all voxels inside this convex hull are selected. The user has the ability to apply a feathering operation with a changeable radius r to the selected voxels leading to every voxel with a distance of less than r to a selected voxel to also be selected. This is a widely used approach in image editing and was requested by the domain scientist, since most of the selections focus on the highly visible skeleton of the fishes and thus the convex hull would leave the thin layer of skin unselected. We found that a default feathering factor of $\frac{1}{20}$th of the volume resolution provided good results. In both cases, all voxels not in $V(\mathscr{C}_i)$ are set to a zero intensity value and the size of the volume is reduced to the bounding box of the exported voxels, thus effectively cropping the volume to be tight fitting. Finally, as a last step, the reduced volume is used as a mask on the original sized volume such that the correct voxels are exported at the original resolution instead.

6 Results

In this section, we illustrate the usefulness of *TopoAngler* by applying it to three scans of the *ScanAllFish* project. The use cases were selected and conducted by two of the scientists of the *ScanAllFish* project directly using the system in an informal setting using a think-aloud protocol. The first user, who is also a coauthor on this paper, was involved in the system's iterative design, hence there was a previous familiarity with the underlying concepts of the system. He conducted the first two use cases (Sections 6.1 and 6.2). The second user was unfamiliar with the system at the beginning and was chosen to test a novice user's performance; he conducted the third use case (Section 6.3). Both users have experience in marine biology of 25 and 7 years respectively. In the following sections, we describe the individual use cases and how the experts were using the system to segment the individual fishes. After each use case, each expert visually inspected the segmentation and approved them for further use in the *ScanAllFish* project.

All three use cases were scanned at the Karel F. Liem Biovisualization Center at the University of Washington's Friday Harbor Laboratory using a Bruker SkyScan 1173 Micro CT scanner. The users used *TopoAngler* to perform the use cases on an Intel Core i7-7700K with 64 GB RAM and a GeForce 1080 GPU. On the same machine the preprocessing times for all three cases, which includes constructing volume data from individual image files, subsampling the volume, and generating the join tree, takes between 5 and 7 minutes.

6.1 Group2 Case

This scan of nine fishes was performed by the expert user [39] and contains a variety of species. Most of the volume surrounding the scanning cylinder was removed and resulted in a resolution of $2000\times2008\times3863$ voxels with a voxel size of 8 µm. This volume was then subsampled to a resolution of $256\times257\times471$ and thus a voxel size of ≈ 32 µm. This subsampling was found sufficient by the domain expert as the reduction in processing time was preferable to the marginal accuracy increase that the original volume would have provided.

The user then utilized our visualization system and proceeded to segment the fishes contained in the dataset. Figure 11 (a) shows the volume rendering of the entire scan displaying all the various fishes that are contained in the scan. Figure 11 (b) shows the application while it is in use. Here, some of the fishes have already been segmented, and are thus being shown in lower saturation, whereas the user is currently segmenting *Bellator militaris*. Figures 11 (c,d,e,f) show the finalized segmentation of two examples (*Bellator militaris* and *Aphredoderus Sayanus*), comparing the volume rendering of the segmented fishes with pictures taken prior to the scan. The entire interactive segmentation sessions with the lead scientist was finished after about 15 minutes after which the extracted full resolution volumes were inspected by the user.

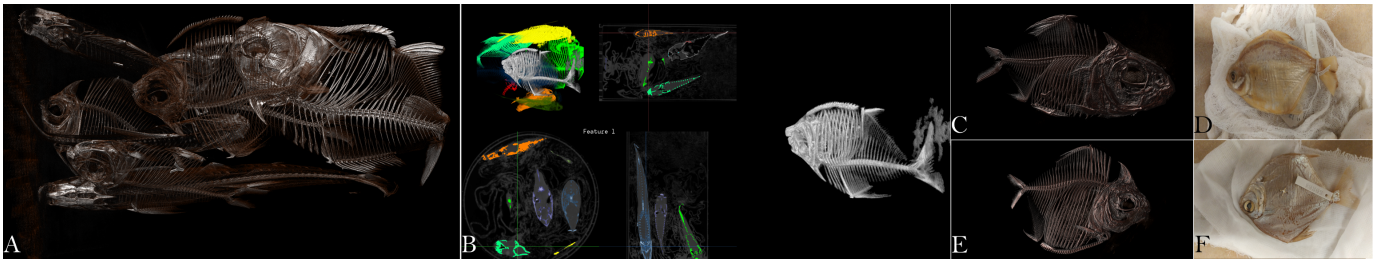


Fig. 13. This use case contains nine South American fishes, mostly Piranhas, that were segmented by the scientist that scanned the dataset. The entire dataset (A) was segmented using *TopoAngler* (B) in less than 10 minutes. Two examples of this segmentation are *Metynnis Hypsauchen* and *Pristobrycon Calmoni*.

6.2 Cameroon Creater Lake Case

This scan was provided by Professor Chris Martin at the University of North Carolina at Chapel Hill and contains ten rare fishes from the Cameroon creater lakes. The volume's size is $2224\times2224\times4580$ with a voxel size of 35.7 µm, and was subsampled to a resolution of $256\times256\times527$ with an effective voxel size of ≈ 321 µm.

The user interactively segmented this dataset in approximately 17 minutes. Similar to the previous use case, Figure 12 shows the two of the fishes in the dataset (*Pungu Maclareni* and *Stomatepia Pindu*), our interactive visualization system in use, and the resulting segmented volumes at the original resolution.

6.3 Piranha Case

This scan was performed by the same scientist that performed this use case, and contains nine South American piranhas. The complete volume size is $2784\times2784\times4653$ with a voxel size of 35.7 µm, and contains the entire scan capsule. It was resampled to a resolution of $256\times256\times428$ thus increasing the voxel size of 388 µm.

Since this use case was performed by a novice user that had no prior experience with the application, the first 20 minutes were spend on describing the system's underlying principles and providing the information contained in Sections 3, 4, and 5, including a hands-on explanation of the usage of the system. The user then familiarized himself with the system for 5 minutes and performed a segmentation of the nine fishes in about 7 minutes. Similar to the previous use cases, Figure 12 shows the two of the fishes in the dataset (*Metynnis Hypsauchen* and *Pristobrycon Calmoni*), *TopoAngler* in use, and the resulting segmented volumes at the original resolution.

6.4 User Feedback

The interactive segmentation sessions with the two users (A, B) were performed with a qualitative think-aloud method. This limits the comparability between the two subjects as each of them spend different amount of times on commenting on their thoughts and actions. While we fully acknowledge that two users are not a statistically significant sample size, it is important to recognize that user A was heavily involved in the development of the system and is the leading expert in the field of comparative marine biology as applied to fish populations. As a future work, we would like to perform a thorough user study by collecting the input and the results from more experts statistically.

Both expert users for the three use cases were very positive about the sessions. A feeling that "this is going to be so many hours of our lives saved", B feeling that "[it] is really easy" to use. The inclusion of the fishes' convex hull was very positively received, A stating that "everything I have seen today is an improvement over what we had before". On the other hand, the expert remarked on current limitations as "it would be nice to include all of the conversions into this system since it gets deployed to biologists". B followed up on that, asking for a possibility to export the segmented volumes into a specialized format for further scientific analysis.

Other thoughts confirmed the design decisions that occurred through the development cycle and we found that the expert used the linked view between the *Selection View* and the *Meta feature view* repeatedly to identify a fish that had not been selected before. B stating that "I want scroll through the features in order to see which of the features falls out first". Furthermore, both A and B heavily utilized volume rendering to inspect whether the selection of a fish was complete and which features to add, A stating that "I can actually see to which fish this piece belongs". In the initial stages, he browsed through the number of features to separate a specific fish he was looking for. Both users were very happy with the broad, initial selection using the volume picking, stating that "it is super handy to pick in the volume rendering. I can hover over it and compare the feature numbers".

The end results of the use cases were very successful as the generated volumes captured the fishes correctly, A stating that "It's a triumph", while B noted that "this does everything I would have asked for". The correctness of the segmented volumes was evaluated by the same user that performed the segmentation.

Limitations. One of the shortcomings of our current system is the preprocessing step, which subsamples the original volume, and then computes the join tree. Right now this is performed by a set of command line scripts. Before wide availability, we plan to make the software better integrated, such that users can easily perform all the required preprocessing through the user interface itself. *TopoAngler* currently extracts the subvolumes corresponding to fishes in the same orientation as the original volume. However, the fishes are not axis-aligned, thus causing the extracted volume to be larger than necessary, thus increasing file sizes. We plan to provide a better alignment algorithm that uses the dominant axis of the fish. Along similar lines, as we saw in some of the cases, it is possible for fishes to be highly curved. Biologists are interested in "unrolling" the fishes since it helps ease further analysis. This is an interesting direction which we plan to pursue in the future, which would require a priori knowledge about the skeletal structure of the fishes. We have not performed a formal user study to test the usability of the system. As we deploy *TopoAngler* to the large number of biologists of the *ScanAllFish* project, usability becomes an important aspect of the software and we will address in the future.

7 Conclusion

We present *TopoAngler*, a visualization and segmentation system that allows users to interactively extract entire fishes from Micro-CT scans containing several fishes. TopoAngler combines techniques from computational topology with a linked views-based visual interface that allows users to interactively gather and extract fishes from an input volume. We demonstrate use cases, and report feedback from the scientists involved in the project indicating the utility of our application. Our current prototype brings the overall time to segment the fish burritos from the project by two orders of magnitude bringing the time down from a week to a few minutes.

Acknowledgements

This work was supported primarily by the Moore-Sloan Data Science Environment at New York University and The Seaver Institute. Further support provided by NASA, DOE, and NSF awards CNS-1229185, CCF-1533564, CNS-1544753, CNS-1730396, and TCN-1701665. We would further like to thank the anonymous reviewers that helped improving the quality of this paper. The described concepts have been realized using the Inviwo visualization framework (www.inviwo.org).

REFERENCES

[1] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483, 1996.

[2] F. P. Bergo, A. X. Falcão, P. A. Miranda, and L. M. Rocha. Automatic image segmentation by tree pruning. *Journal of Mathematical Imaging and Vision*, 29(2):141–162, 2007.

[3] P.-T. Bremer, G. Weber, V. Pascucci, M. Day, and J. Bell. Analyzing and Tracking Burning Structures in Lean Premixed Hydrogen Flames. *IEEE TVCG*, 16(2):248–260, Mar. 2010.

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

[5] H. Carr, J. Snoeyink, and M. van de Panne. Simplifying Flexible Isosurfaces Using Local Geometric Measures. In *IEEE Visualization*, pages 497–504, 2004.

[6] Y.-J. Chiang and X. Lu. Progressive simplification of tetrahedral meshes preserving all isosurface topologies. *Computer Graphics Forum*, 22(3):493–504, 2003.

[7] C. Correa, D. Silver, and M. Chen. Illustrative deformation for data exploration. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1320–1327, 2007.

[8] H. Doraiswamy, N. Ferreira, T. Damoulas, J. Freire, and C. Silva. Using topological analysis to support event-guided exploration in urban data. *TVCG*, 20(12):2634–2643, 2014.

[9] H. Doraiswamy and V. Natarajan. Output-sensitive construction of Reeb graphs. *IEEE TVCG*, 18(1):146–159, 2012.

[10] H. Doraiswamy, V. Natarajan, and R. S. Nanjundiah. An Exploration Framework to Identify and Track Movement of Cloud Systems. *TVCG*, 19(12):2896–2905, 2013.

[11] H. Edelsbrunner. *Geometry and Topology for Mesh Generation*. Cambridge Univ. Press, England, 2001.

[12] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. Amer. Math. Soc., 2009.

[13] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Morse-Smale Complexes for Piecewise Linear 3-Manifolds. In *Symp. Comput. Geom.*, pages 361–370, 2003.

[14] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse-Smale Complexes for Piecewise Linear 2-Manifolds. *Disc. Comput. Geom.*, 30(1):87–107, 2003.

[15] P. A. Freeborough, N. C. Fox, and R. I. Kitney. Interactive algorithms for the segmentation and quantitation of 3-d mri brain scans. *Computer methods and programs in biomedicine*, 53(1):15–25, 1997.

[16] M. Graham and J. Kennedy. Vesper: Visualising species archives. *Ecological Informatics*, 24:132–147, 2014.

[17] P. Green-Armytage. A colour alphabet and the limits of colour coding. *JAIC-Journal of the International Colour Association*, 5, 2010.

[18] A. Hatcher. *Algebraic Topology*. Cambridge U. Press, New York, 2002.

[19] K. H. Höhne and W. A. Hanson. Interactive 3d segmentation of mri and ct volumes using morphological operations. *Journal of computer assisted tomography*, 16(2):285–294, 1992.

[20] P. Iassonov, T. Gebrenegus, and M. Tuller. Segmentation of x-ray computed tomography images of porous materials: A crucial step for characterization and quantitative analysis of pore structures. *Water Resources Research*, 45(9):n/a–n/a, 2009. W09415.

[21] W. K. Jeong, J. Beyer, M. Hadwiger, A. Vazquez, H. Pfister, and R. T. Whitaker. Scalable and interactive segmentation and visualization of neural processes in em datasets. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1505–1514, Nov 2009.

[22] R. M. Kirby and M. Meyer. Visualization collaborations: What works and why. *IEEE computer graphics and applications*, 33(6):82–88, 2013.

[23] D. Laney, P. T. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci. Understanding the Structure of the Turbulent Mixing Layer in Hydrodynamic Instabilities. *IEEE TVCG*, 12(5):1053–1060, Sept. 2006.

[24] S. Lindholm, D. Jönsson, C. Hansen, and A. Ynnerman. Boundary aware reconstruction of scalar fields. *IEEE transactions on visualization and computer graphics*, 20(12):2447–2455, 2014.

[25] S. Liu, W. Cui, Y. Wu, and M. Liu. A survey on information visualization: recent advances and challenges. *The Visual Computer*, 30(12):1373–1393, 2014.

[26] J. Milnor. *Morse Theory*. Princeton Univ. Press, 1963.

[27] F. Miranda, H. Doraiswamy, M. Lage, K. Zhao, B. Gonalves, L. Wilson, M. Hsieh, and C. Silva. Urban pulse: Capturing the rhythm of cities. *IEEE Transactions on Visualization and Computer Graphics*, 2016.

[28] D. W. Moore. Simplical mesh generation with applications. Technical report, Cornell University, 1992.

[29] T. Munzner. A nested model for visualization design and validation. *IEEE transactions on visualization and computer graphics*, 15(6), 2009.

[30] T. N. A. Nguyen, J. Cai, J. Zhang, and J. Zheng. Robust interactive image segmentation using convex active contours. *IEEE Transactions on Image Processing*, 21(8):3734–3743, 2012.

[31] I. Nyström, J. Nysjö, A. Thor, and F. Malmberg. Bonesplit-a 3d painting tool for interactive bone segmentation in ct images. In *International Conference on Pattern Recognition and Information Processing*, pages 3–13. Springer, 2016.

[32] S. D. Olabarriaga and A. W. Smeulders. Setting the mind for intelligent interactive segmentation: Overview, requirements, and framework. In *Biennial International Conference on Information Processing in Medical Imaging*, pages 417–422. Springer, 1997.

[33] V. Pascucci and K. Cole-McLaughlin. Parallel Computation of the Topology of Level Sets. *Algorithmica*, 38(1):249–268, 2003.

[34] V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli. The TOPORRERY: computation and presentation of multi-resolution topology. In *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, Mathematics and Visualization, pages 19–40. Springer, 2009.

[35] V. Pascucci, G. Weber, J. Tierny, P.-T. Bremer, M. Day, and J. Bell. Interactive Exploration and Analysis of Large-Scale Simulations Using Topology-Based Data Segmentation. *IEEE TVCG*, 17(9):1307–1324, 2011.

[36] J. S. Prassni, T. Ropinski, and K. Hinrichs. Uncertainty-aware guided volume segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1358–1365, Nov 2010.

[37] A. Protiere and G. Sapiro. Interactive image segmentation via adaptive weighted distances. *IEEE Transactions on image Processing*, 16(4):1046–1057, 2007.

[38] N. Shivashankar, P. Pranav, V. Natarajan, R. v. d. Weygaert, E. G. P. Bos, and S. Rieder. Felix: A topology based framework for visual exploration of cosmic filaments. *IEEE Transactions on Visualization and Computer Graphics*, 22(6):1745–1759, June 2016.

[39] A. P. Summers. Adamgroup2, May 2016.

[40] G.-D. Sun, Y.-C. Wu, R.-H. Liang, and S.-X. Liu. A survey of visual analytics techniques and applications: State-of-the-art research and future challenges. *Journal of Computer Science and Technology*, 28(5):852–867, 2013.

[41] S. Takahashi, Y. Takeshima, and I. Fujishiro. Topological Volume Skeletonization and Its Application to Transfer Function Design. *Graphical Models*, 66(1):24–49, 2004.

[42] M. Tory and T. Moller. Human factors in visualization research. *IEEE transactions on visualization and computer graphics*, 10(1):72–84, 2004.

[43] B. Wang, K. W. Liu, K. M. Prastawa, A. Irima, P. M. Vespa, J. D. van Horn, P. T. Fletcher, and G. Gerig. 4d active cut: An interactive tool for pathological anatomy modeling. In *2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI)*, pages 529–532, April 2014.

[44] G. H. Weber, S. E. Dillard, H. Carr, V. Pascucci, and B. Hamann. Topology-controlled volume rendering. *TVCG*, 13(2):330–341, 2007.

[45] P. A. Yushkevich, J. Piven, H. Cody Hazlett, R. Gimpel Smith, S. Ho, J. C. Gee, and G. Gerig. User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability. *Neuroimage*, 31(3):1116–1128, 2006.

[46] L. Zhang and Q. Ji. A bayesian network model for automatic and interactive image segmentation. *IEEE Transactions on Image Processing*, 20(9):2582–2593, 2011.

[47] F. Zhao and X. Xie. An overview of interactive medical image segmentation. *Annals of the BMVA*, 2013(7):1–22, 2013.

[48] F. Zhou, Y. Zhao, and K.-L. Ma. Parallel mean shift for interactive volume segmentation. In *Proceedings of the First International Conference on Machine Learning in Medical Imaging*, MLMI'10, pages 67–75, Berlin, Heidelberg, 2010. Springer-Verlag.