Introducing SPROUT (Semi-automated Parcellation of Region Outputs Using Thresholding): an adaptable computer vision tool to generate 3D segmentations

Yichen He[1,2]*, Marco Camaiti[2]*, Lucy E. Roberts[2], James M. Mulqueeney[2,3], Marius Didziokas[4], Anjali Goswami[2]

1. School of Biosciences, University of Sheffield, Sheffield, UK
2. Department of Life Sciences, Natural History Museum, London, UK
3. Department of Ocean & Earth Science, National Oceanography Centre Southampton, University of Southampton, UK
4. Department of Mechanical Engineering, University College London, London, UK

*co-first authors. Email: yichen.he@nhm.ac.uk, marco.camaiti1@nhm.ac.uk

Abstract

The increased availability of 3D image data requires improving the efficiency of digital segmentation, currently relying on manual labelling, especially when separating structures into multiple components. Automated and semi-automated methods to streamline segmentation have been developed, such as deep learning and smart interpolation, but require pre-labelled data, and specialized hardware and software. Deep learning models in particular often require the manual creation of extensive training data, particularly for complex multi-class segmentations. Here, we introduce SPROUT, a novel, semi-automated computer vision method providing a time-efficient and user-friendly pipeline for segmenting and parcellating image data. SPROUT generates seeds (representing parts of an object) based on specified density thresholds and erosion of connected components, to achieve element separation. Seeds are grown to obtain fully-parcellated segmentations. We compare SPROUT's performance to that of smart interpolation and apply it to diverse datasets to demonstrate the utility and versatility of this open-source 3D segmentation method.

Introduction

Rapid analysis of image data is critical for numerous applications, from scientific research to medicine, engineering, and security. High-resolution 2D and 3D image data are being generated at rates unthinkable just a few years ago (*1*, *2*), placing increasing strain on users to process these data efficiently. One example of vast increases in image data comes from the field of natural history. The digitisation of natural history specimens plays a crucial role in preserving and making collections of biological, geological, and archaeological data accessible, enabling researchers to study specimens non-invasively and in greater detail (*3*–*5*). By converting specimens into digital formats, scientists can

apply advanced computational techniques to extract and analyse phenotypes in detail (*6–8*). In particular, 3D images produced using computed tomography (CT), magnetic resonance (MRI), electron tomography and phase-contrast scanning have become commonplace in natural history, alongside the rise of large 3D image libraries for open data sharing (*9–12*).

The increased availability of 3D data brings countless benefits, but also new challenges for data processing (*2*, *13*). One of the major bottlenecks in the processing of 3D images is the segmentation of scans. While tools using smart interpolation or deep learning have greatly sped up the process of segmenting whole specimens (*14*, *15*), there remain significant barriers to quickly separating 3D images into the individual components of an object (parcellation). For example, individually separating the bones of an entire skeleton, particularly when there are varying numbers of elements across a set of images, can present a lengthy and challenging process. As the capability to efficiently generate 3D image data expands at a rapid pace, efficiently extracting detailed and specific information from these images has become a key priority to conduct large-scale analyses.

For these imaging modalities, image segmentation offers a solution for extracting detailed regions from a scan (*16*). While these terms are used variably in different applications, here we use the term image segmentation to refer to the isolation of specific elements of an image, which has also been described as defining regions of interest from a scan (*17*). Many segmentation tasks remain manual at present, particularly when the required data are complicated or domain-specific, meaning that the extraction of data is limited by the expertise and speed of a given user, and is prone to the interference of subjective biases (*18*). This poses a significant issue to many fields such as the biological sciences, medical science (e.g., segmenting tumours from medical CT scans, (*19*, *20*)), engineering and material sciences (e.g., separating the different parts of an engine to study their respective interactions (*21*, *22*)), and geology (e.g., separating specific aggregated microstructures into their finer components (*23*, *24*)).

Recent methodological advances have attempted to improve the efficiency of 3D image segmentation in different ways. Most prominently, deep learning has shown much promise in automatically segmenting biological images. Deep learning models such as U-net (*14*) and V-net (*25*) have proven effective at segmenting CT datasets based on highly specific training data. Several open-source tools, such as MONAI (*26*), nnU-Net (*27*), Biomedisa (23, 24), and ilastik (25), as well as segmentation software such as Avizo (Thermo Fisher Scientific, USA), VGStudio (Volume Graphics, Germany), and Dragonfly (Object Research Systems, Canada), implement deep and machine learning techniques for segmentation and provide the ability to train custom models.

However, as the complexity and variability of input data increases, or when extensive domain knowledge is required, the amount of training and post-processing required to

obtain useful data can also be expected to increase accordingly (*8*, *28*, *29*). Improving the performance of these models can be time-consuming and difficult due to their nature as 'black boxes' (*30*, *31*), requiring significant efforts to tune models and model architectures to handle new datasets. Additionally, the creation of training data for these models requires high domain knowledge from the user, limiting their performance and applicability, in addition to being time-consuming (*32*). In general, the main issues with these AI-based methods are the time scalability and specificity of the tasks, leading to diminishing return scenarios when these factors are not properly accounted for.

Another promising avenue to handle these tasks has been through the use of classic computer vision algorithms (*33*). Some examples of these methods are thresholding (*34*), interpolation (*35*), morphological transformations (*36*), watershed (*37*), and smart-grid tools (*38*), as well as seed-based methods (*39*). Sophisticated extensions of these approaches exist, such as smart interpolation as implemented in Biomedisa (*35*), which uses weighted random walk models incorporating the underlying volumetric data to predict segmentation. Compared to deep learning, these have the advantage of not requiring any training data or expensive graphics processing units (GPU). These methods typically rely on heuristic inputs (e.g., preset thresholds or seed locations) to operate based on the inherent properties of an image, such as the grayscale values of stacked CT images that form a 3D volume. As they do not require any other information besides what is already contained in the images themselves, non-deep learning, computer vision-based workflows are more adaptable to variable inputs (*40*). While this does still require some domain knowledge, variability in results can be mitigated by the ability to quickly generate multiple outputs based on different input parameters (i.e., threshold selection), limiting the user's contribution to a series of relatively simple steps and choices.

Although tools using deep learning and smart interpolation have proven quite successful at handling varied biological datasets, they come with some drawbacks. Insufficient resolution of images can significantly impact the ability of these methods to separate elements, due to the presence of voxels with common grayscale values between labels causing the overgrowth of one label onto another. This means that results often require significant manual corrections to yield biologically-relevant segmentations.

More sophisticated combinations of these classic computer vision methods have recently emerged, such as the newly released BounTI (*41*), an add-on for the image processing software Avizo. BounTI uses an initial threshold to separate regions on a 3D input image. Once a seed with the desired separations is selected, it can be grown to a target threshold. The ideal result of these operations is a multi-class ("parcellated") segmentation. Importantly, BounTI (*41*) performs well with lower-resolution scans, which may even be preferred for some datasets because small image sizes speed up processing time, and can actually achieve better discrimination of grayscale voxels. However, seed creation and growth in BounTI can take significant amounts of time, as

choosing an initial threshold is more skill-intensive and sensitive to user error. BounTI also does not support parallel processing, which exacerbates its inefficiency when segmenting large datasets. BounTI also works best with proprietary software (i.e., Avizo) whose cost and required operational knowledge can make it inaccessible to standard users.

To face the highlighted challenges in image segmentation, we present an open-source Python toolkit called SPROUT (Semi-automated Parcellation of Region Outputs Using Thresholding). This tool integrates computer vision methods (e.g., thresholding and morphological transformations) to generate fine-grained multi-class segmentations on 3D cross-sectional scans with customized user inputs. Similar to BounTI (*41*), SPROUT is based on seed generation and growth and performs well with lower-resolution scans. However, SPROUT incorporates additional methods (e.g., morphological erosion) for seed creation to achieve potential better separations between elements. It also includes the ability to generate merged seeds (combining the elements generated in multiple seeds), reducing the need for manual corrections. Because SPROUT is not based on deep learning, it does not require the time-consuming process of generating training datasets. Additionally, SPROUT supports batch processing and full parallelization, with optional add-ons for data loading and visualisation in Avizo, maximising its efficiency for analysing large datasets.

To demonstrate the capabilities of SPROUT, we compare the accuracy and processing time of its output to both raw and manually corrected results from smart interpolation using Biomedisa (35). We also show that SPROUT—a user-friendly and freely-accessible tool—can produce time-efficient, reliable, and meaningful segmentations of 3D image data from a wide range of biological, medical, and engineering fields.

Methods

Software requirements

The SPROUT implementation requires Python v 3.10 and is built using a range of well-supported libraries. The following libraries are integral to the workflow:

- **NumPy** (v 1.26.4)
- **Pandas** (v 2.21)
- **Scikit-image** (v 0.22.0)
- **Tifffile** (v 2024.2.12)
- **PyYAML** (v 6.0.1)
- **Trimesh** (v 4.3.1)
- **Open3D** (v 0.18.0)
- **Matplotlib** (v 3.8.3)

The software has been tested and verified to run on Windows 10/11 and Ubuntu 24.04 LTS platforms. It was not tested but is expected to run on MacOS. Avizo add-ons have been tested with Avizo 2020, Avizo 2022 Pro, and Avizo 2023 Pro. Additionally, SPROUT supports multi-core processing, making it well-suited for deployment on high-performance computing (HPC) clusters and cloud computing platforms such as AWS EC2.

The source code, including scripts for Avizo add-ons, a detailed tutorial, vignettes, and test scan data, can be found at https://github.com/PhAInomics/SPROUT. Example input files, such as .yaml configuration files for running SPROUT functions and the suture mapping file, are also available in the GitHub repository.

Overview

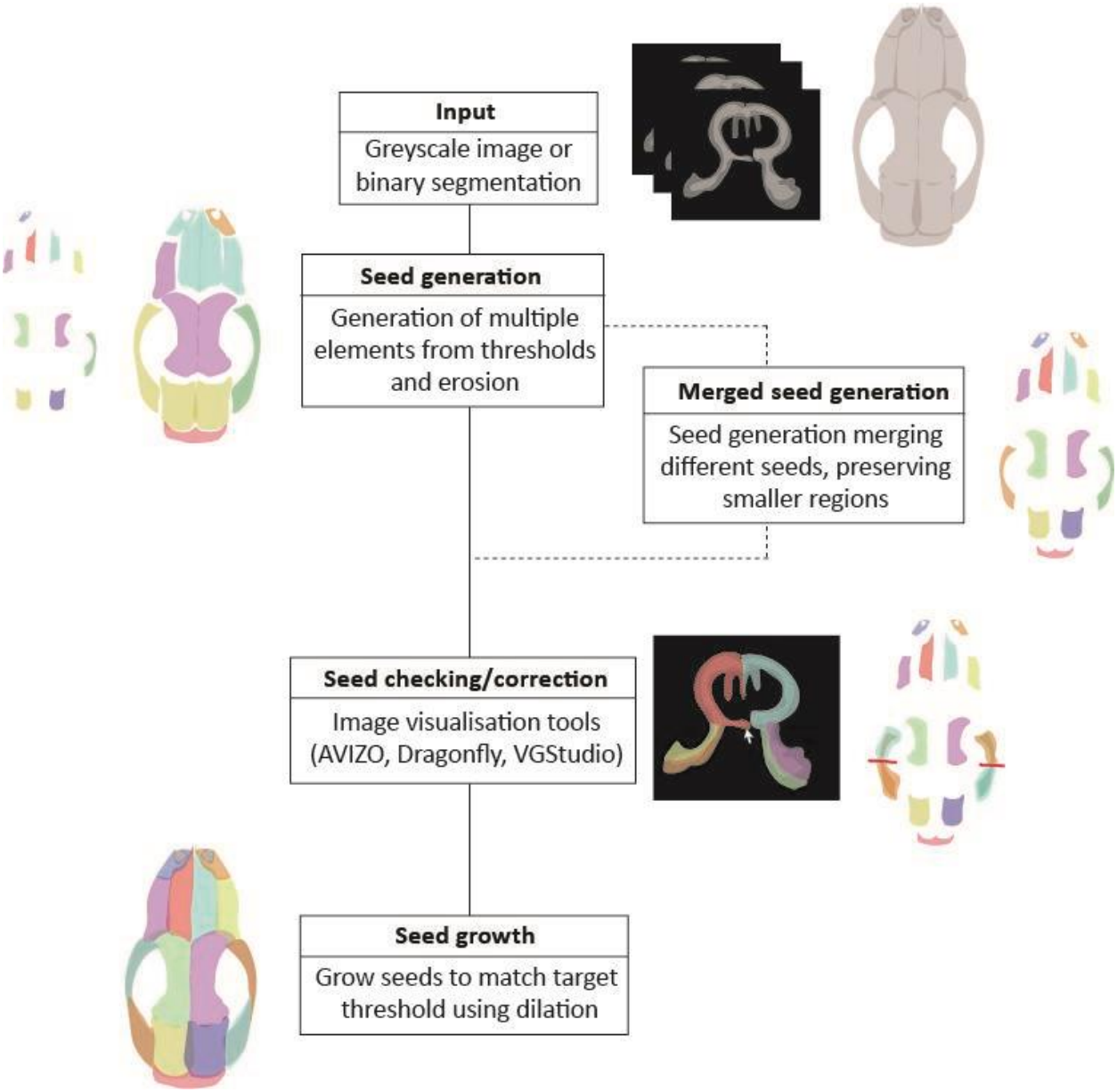

Figure 1. A schematic flowchart describing the sequential steps of the SPROUT process, using a mammalian skull as an example. Seeds are created from an initial 3D input on a given threshold value for maximum separation, then grown to a target threshold, obtaining a segmented object as the output.

SPROUT was originally developed to solve the time-consuming problem of segmenting individual bones of the mammalian skull from micro-CT scans. It has since been modified and tested on various datasets, evolving into a semi-automated workflow for segmenting fine-grain elements in cross-sectional images, especially of biological specimens. SPROUT has demonstrated potential as an efficient and effective method for producing accurate and biologically meaningful segmentations.

Due to the nature of organisms or the quality of scanning systems, clear separations among target elements are not always evident, particularly when elements are connected or semi-connected within a larger structure, as is often the case for cranial bones (*42*). This workflow uses a combination of computer vision methods to identify separations between these elements. For many scans, reducing the size of the segmented regions can potentially increase separations among individual elements of interest. This is especially relevant in biological specimens because the connections between components such as bones and organs often appear as thin areas or weak signals (i.e., lower intensities). The workflow extracts separated regions that correspond to these individual elements in a step known as seed generation, where multiple seeds can be generated based on the input parameters.

Following the seed generation step, a candidate seed needs to be selected. The separated regions of the candidate seed may not accurately represent the target elements, as seeds are typically smaller. A seed growth step that is similar to region growing (*43*) is applied, using the separated regions as seeds (i.e., starting positions) and growing them back to match the target elements (Fig. 1). This combination of seed generation and controlled growth enables precise, multi-class segmentation of biological structures in cross-sectional scans.

Seed Generation

The seed generation step applied a threshold to generate a binary segmentation, isolating the foreground (e.g., the target object) from the background. Higher threshold values typically produce smaller foreground regions. Morphological erosion was then applied to the binary segmentation, reducing the foreground to smaller regions. By applying thresholds, erosion, or combining both techniques, the foreground size can be reduced while increasing the separation between regions. A disconnected components algorithm is subsequently used to identify these separated regions, retaining the largest regions

based on the target number of segments. SPROUT enables users to easily combine and fine-tune these techniques, making it possible to segment multiple distinct regions effectively.

Additionally, for datasets with pre-existing binary segmentations, users can directly input these segmentations into the seed generation step by setting the threshold to one. High-quality binary segmentations can be efficiently generated through various approaches, such as thresholding (*34*, *42*) or deep learning (*15*, *28*), as isolating the foreground is often simpler compared to multi-class segmentation tasks.

The make_seeds.py script is used to create seeds for segmentation, using the following key inputs: (1) input and output information, including an input image and an output folder; (2) a threshold value; (3) the number of erosion steps and (4) the target number of segments. Additional, optional, parameters include the direction of the erosion (the default is a sphere with a radius of one) and a boundary mask, which identifies non-target regions to help create more separations—especially useful for complex images.

Seed Selection

With different input settings, multiple seeds can be generated; however, only one seed is selected as the candidate seed for the growth step. To choose the best seed, the method offers seed statistics, including the volume of the binary segmentation generated by various thresholds and the volume of each segmented element. This helps the user to understand how the method performs on their data and potentially apply initial filtering using these statistics.

It is important to check the seeds. Users can visualise and check seeds by volume rendering them using cross-sectional image manipulation tools such as 3D Slicer (*44*), Dragonfly (Object Research Systems, Canada) and Avizo (Thermo Fisher Scientific, USA). We also provide a make_mesh.py function to create coloured meshes of each segmented region, which users can then check using mesh manipulation tools such as MeshLab.

Common issues include under-segmentation, over-segmentation, and disappearing regions. These problems arise due to a trade-off between more regions with fewer separations (under-segmentation) or regions with more separations (over-segmentation and disappearing regions, Fig. 1). Seeds can be regenerated by adjusting the input parameters to address these issues. Another solution is manually separating or merging inaccurate regions, which is especially effective when dealing with nearly correct seeds.

Seed Growth

Once a candidate seed is selected, the next step is to grow the seed to better represent the original morphology of the target elements, as regions derived from seeds are typically

smaller than the actual target elements. We apply morphological dilation to grow each element in the seed. To prevent the dilation from including unwanted regions (e.g., backgrounds), we use a binary segmentation as a boundary to regulate the growth process. A common use case involves applying a range of thresholds, starting from a value higher than the target threshold and gradually decreasing to the target threshold. The target threshold is defined as the threshold value that produces the most accurate binary segmentation. By progressively lowering the threshold, the boundary (i.e., the binary segmentation) grows incrementally, making the growth step steady.

The seed growth step is run using the function make_grow.py. The key inputs are (1) input and output information, including an input image, the candidate seed and an output folder; (2) a threshold or a list of thresholds and (3) the number of dilations. Other inputs include the direction of the dilation (the default is a sphere with a radius of one), specified segmentation classes to grow, the output folder, and how many intermediate growth results to save.

As in the seed selection step, users can view the growth statistics or visualise the growth result. If inaccurate segments are observed, adjustments can be made by modifying growth parameters or performing manual corrections, ensuring that the grown regions accurately represent the biological structures of interest.

Additional tools

The steps introduced above are the fundamental functions of the SPROUT workflow. Additional functions have been integrated to enhance SPROUT's efficiency and versatility in processing complex biological data.

Merged Seed Generation

As mentioned earlier, the trade-off between remaining regions and separations can pose a challenge for generating a good seed. To address this, we developed the make_seeds_merged.py function. This function creates a merged seed from multiple seeds, detects separations and retains areas that are completely eroded. Thus, it creates a seed with maximum separations while preserving as much area as possible.

The inputs for make_seeds_merged.py are similar to those used for make_seeds.py, including basic input and output information and the number of segments. However, the merged seed generation provides two distinct methods for merging seeds based on the inputs:

1. Threshold List with Fixed Erosion: This approach requires a list of thresholds alongside a fixed erosion number. The function generates seeds at each threshold level with the same erosion and then merges them. This approach allows seeds

generated at higher thresholds (with more separations) to be combined with those at lower thresholds (with greater region preservation).

2. Single Threshold with Varying Erosions: This approach requires a single threshold and a fixed erosion number. The function generates seeds by applying erosions, starting from none (zero) up to the given erosion count with the input threshold. The seeds are then merged to include minimally eroded areas and more separated regions.

Through these two types of merged seed generation, make_seeds_merged.py offers flexibility for different segmentation needs, meaning that SPROUT can generate high-quality seeds with little or no manual intervention. Furthermore, if certain patterns, particularly biological ones, can be leveraged, it is possible to automatically set inputs to batch-generate merged seeds and growth results for multiple scans. The primary dataset used for the evaluation of SPROUT, dog limbs, were segmented using this automated version of SPROUT, as discussed in Results.

Batch Processing and Parallelization

All functions of SPROUT support batch processing. Users can batch-generate seeds, merged seeds, and growth results with global or individual scan input. Individual scan parameters might be better suited when differently formated specimens or images are batch-processed together. For example, some scans require fewer erosions to achieve good seeds, while others may require more. This flexibility helps maximise segmentation accuracy across a diverse range of scans. Logs are saved, including segmentation statistics, runtime performance, and error handling, providing users with comprehensive data for troubleshooting and optimizing batch runs.

The SPROUT workflow also supports various levels of parallel processing to fully utilise available CPU cores. In batch processing, it supports the simultaneous parallel processing of multiple images. Parallelization has also been implemented for generating seeds, merged seeds, meshes and growth results, using each segmentation class concurrently. In addition, a parallelized version of BounTI has been implemented as well. Memory usage has been optimized to ensure minimal memory consumption when processing scans. It is important to note that parallelization requires multi-core computing devices and sufficient memory to read and process multiple scans. We report on the performance of test runs using parallelization in the Results section.

Avizo Add-ons

Manual checking, data cleaning and processing are crucial tasks in image analysis, especially in recent evolutionary morphology studies that now commonly use micro-CT scans. With datasets growing increasingly (*45*, *46*), these manual processes can be quite demanding and time-consuming (*47*). As the dataset sizes in these studies are typically not large enough from a statistical standpoint to ignore some errors, ensuring data (e.g.,

fine-grained segmentation) accuracy is essential. Minimising errors is, therefore, of great importance. To address this, we leverage the existing micro-CT software Avizo (Thermo Fisher Scientific, USA), where we have developed Avizo add-ons with graphic user interfaces (GUIs) for users to apply the SPROUT workflow and manipulate generated results easily.

Avizo is a powerful tool with GUIs for manipulating and visualising images such as micro-CT scans. However, using it within our workflow would require manually opening and applying multiple functions and filters. We have built add-ons that minimise unnecessary operations. These add-ons support pre-processing, batch loading of results, result manipulation (e.g., merging and splitting), and GUI-supported generation of seeds, growth results, and sutures.

Data & Results

Evaluation and comparison: dog limbs

To test the effectiveness and accuracy of SPROUT segmentation we performed a comparative analysis on nine sets of CT images of the domestic dog *Canis familiaris* (University of Liverpool Small Animal Teaching Hospital). We cropped each CT stack to focus on the forelimb and set SPROUT thresholds to focus on the extraction of the bones. These scans encompass different breeds, voxel sizes and limb orientations. However, as the grayscale values of different regions for these scans are similar, we used a batch-processed pipeline including merged seed generation and growth to generate the segmentation automatically. We produced the merged seed by incrementally increasing the threshold from 100 to 200 by intervals of 5 (Fig. 2).

For comparison to the SPROUT segmentations, we also generated results using the smart interpolation tool in Biomedisa, pre-segmenting every 10-50 slices (*48*, *35*). We then manually corrected the Biomedisa results, using this as our ground truth. Smart Interpolation requires pre-segmented slices, which we created for the scapula, humerus, radius and ulna. To compare this directly with SPROUT we removed all of the segments from the SPROUT output except for these four, and re-ordered the labels according to the order used for Smart Interpolation. Pairwise DICE scores (ranges from 0 to 1, is a measure of overlap between two segmentations) were calculated for segmentations from three sets of results: (1) Ground truth segmentation (manually corrected Biomedisa smart interpolation result); (2) Biomedisa smart interpolation result; (3) SPROUT result.

Outputs from the three sets of segmentations were remarkably similar. The mean DICE score comparing the ground truth with the uncorrected Biomedia smart interpolation outputs and with the SPROUT outputs are all between 0.98 and 0.99, representing 98-99% similarity between the ground truth and the outputs of both methods (Fig. 3). Differences between segmentations generated by SPROUT and the ground truth model

are observed to be increased separation of elements in the SPROUT output, such as, for example at the proximal portion of the humerus and ulna (Fig. 2). Despite the strong similarity in the final result of the SPROUT and Biomedisa outputs, these results represent marked differences in time input. Using SPROUT we batch processed the nine dog specimens using a single script, which ran for 23 hours. We then visualised the outputs and manually corrected the labels in Avizo. The total manual/user time input for manual correction was less than 10 minutes per specimen. Biomedisa requires a pre-segmentation step of manually labelling slices of each element, which took approximately 30 minutes per specimen for this dataset, whereas SPROUT does not require pre-segmentation. In addition to the difference in user input, SPROUT outputs every element within the scan that is captured by thresholding; when segmenting the limb bones for example SPROUT also segments out the mandible, skull, ribs and bones of the foot (Fig. 2).

Table 1. Information on the dog specimens used for the assessment of SPROUT's performance compared with other segmentation methods.

Specimen number	**Breed**	**Side**	**Voxel size x, y (mm)**	**Voxel size z (mm)**
200000	Greyhound	Left	0.7578125	1
200000	Greyhound	Right	0.7578125	1
242343	Border Collie	Left	0.401	0.3
242343	Border Collie	Right	0.401	0.3
255544	French Bulldog	Right	0.273	0.3
256334	French Bulldog	Left	0.698	0.3
256334	French Bulldog	Right	0.698	0.3
257098	Dachshund	Left	0.415	0.3
257098	Dachshund	Right	0.415	0.3

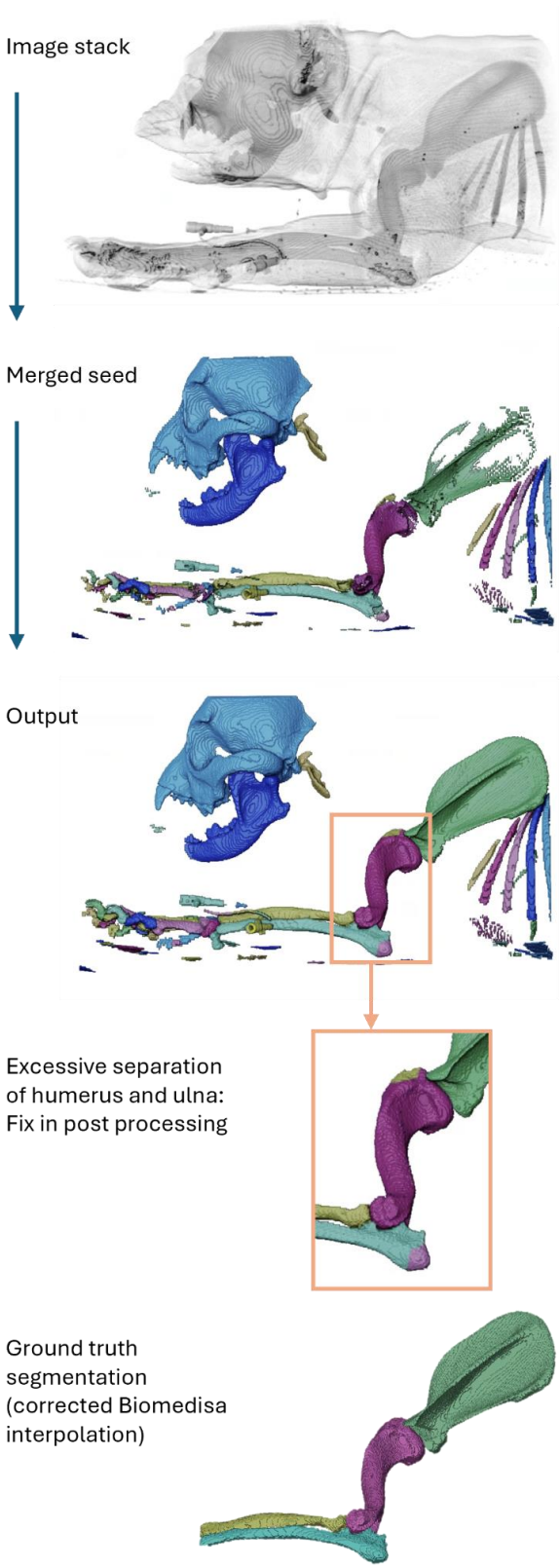

Figure 2. Application of the SPROUT workflow to segment the forelimb of a French bulldog specimen 256334 (left side) from a medical CT scan.

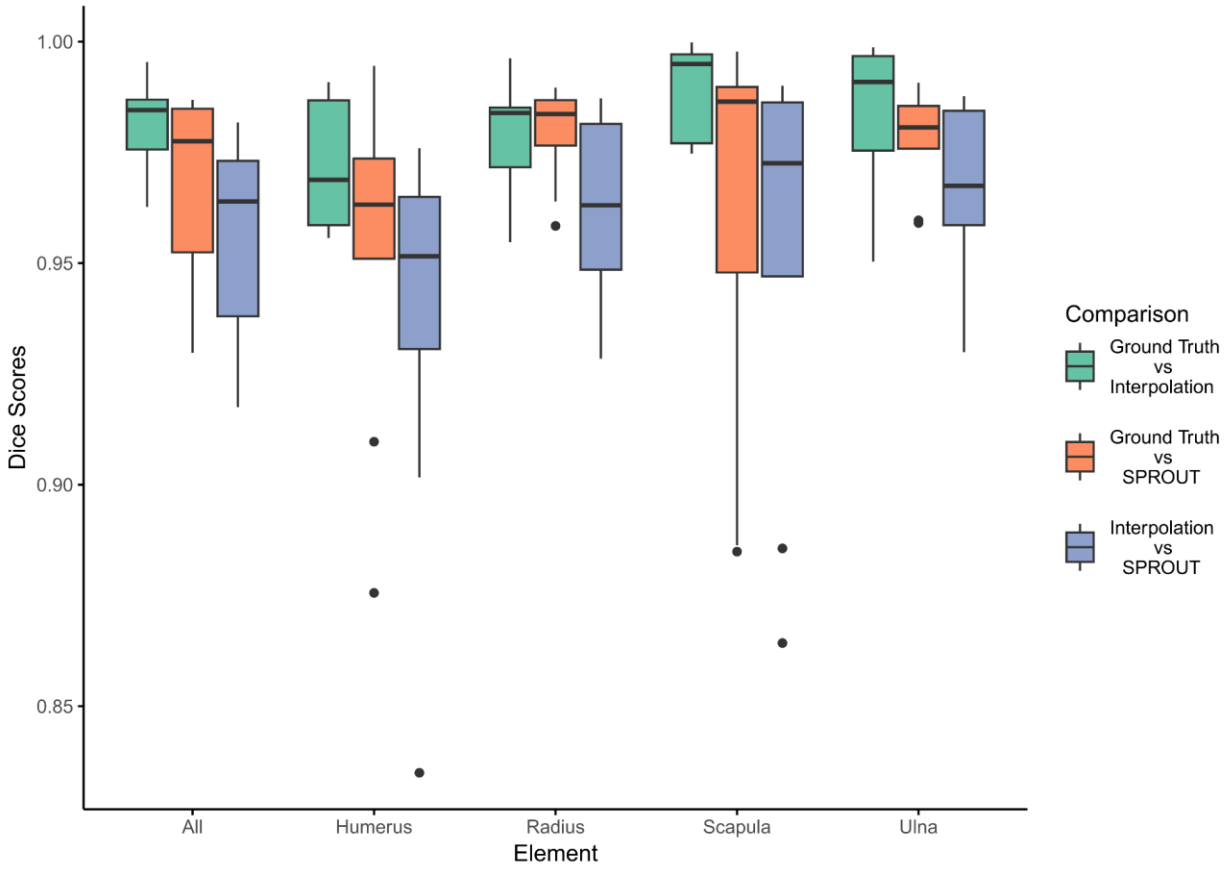

Figure 3. Boxplot of the DICE scores comparing the ground truth (manual correction of the Biomedisa-generated smart interpolation) segmentations with raw outputs of the Biomedisa smart interpolation, and those of the SPROUT segmentation.

Demonstrations of SPROUT on additional datasets: from human hearts to concrete blocks

We also performed SPROUT segmentation on five additional volumetric datasets to test the versatility of the method on different scan modalities and subjects (Table 2, Fig. 4). This includes MRI volumetric data of a human heart (Pace *et al.* 2015), micro-CT data of a whole biological specimen (*Tiliqua scincoides*, California Academy of Sciences CAS:HERP:254658), (*Orycteropus afer,* AMNH 51909), binary segmentations of foraminifera generated from initial segmentation of micro-CT data (µ-VIS X-ray Imaging Centre, University of Southampton, UK), and micro-CT data of non-biological material (steel-reinforced concrete, Hampe 2024).

Table 2. Sample, source, scan details, and seed threshold values for the example datasets used to demonstrate SPROUT's applicability to diverse image data and data modalities.

Scan type	**Object**	**Specimen number**	**Voxel size x, y (mm)**	**Voxel size z (mm)**	**Seed thresholds**	**Source**
Micro-CT	Whole specimen: *Tiliqua scincoides*	CAS:HERP:254658	0.120954	0.120954	100 - 220	California Academy of Sciences & oVert TCN (downloaded from Morphosource) ark:/87602/m4/M74717
Micro-CT	Partial specimen: *Orycteropus afer*	AMNH 51909	0.202	0.202	173, 181	The University of Texas High-Resolution X-ray CT Facility & American Museum of Natural History (dowloaded from Digimorph)
Binary segmentation	Planktonic Foraminifera (*Menardella limbata)*	st014_bl4_fo1	0.00175	0.00175	1	Ocean Drilling Program (ODP) Site 925, µ-VIS X-ray Imaging Centre, University of Southampton, UK
MRI	Human heart	pat0	0.9	0.9	40 - 120	Pace, D. F. et al. (2015) 10.1007/978-3-319-24574-4_10
Micro-CT	Non-biological: Steel reinforced concrete	N/A	0.0020998	0.0020998	70 - 140	Hampe, M. (2024) https://doi.org/10.5281/zenodo.10785685

These datasets demonstrate that the pipeline described above is applicable to a broad range of systems. In applying SPROUT to the micro-CT scan of a whole skeleton of a skink, the MRI scan of a human heart, and the micro-CT scan of a slab of steel-reinforced concrete, the same process as detailed above for the dog limb example was used, with the only modification being in the range of thresholds and the target number of segments used to generate seeds. We expect that this protocol will therefore be sufficient for the majority of users. However, we demonstrate two further examples that deviate slightly from this protocol due to either the modality or complexity of the data.

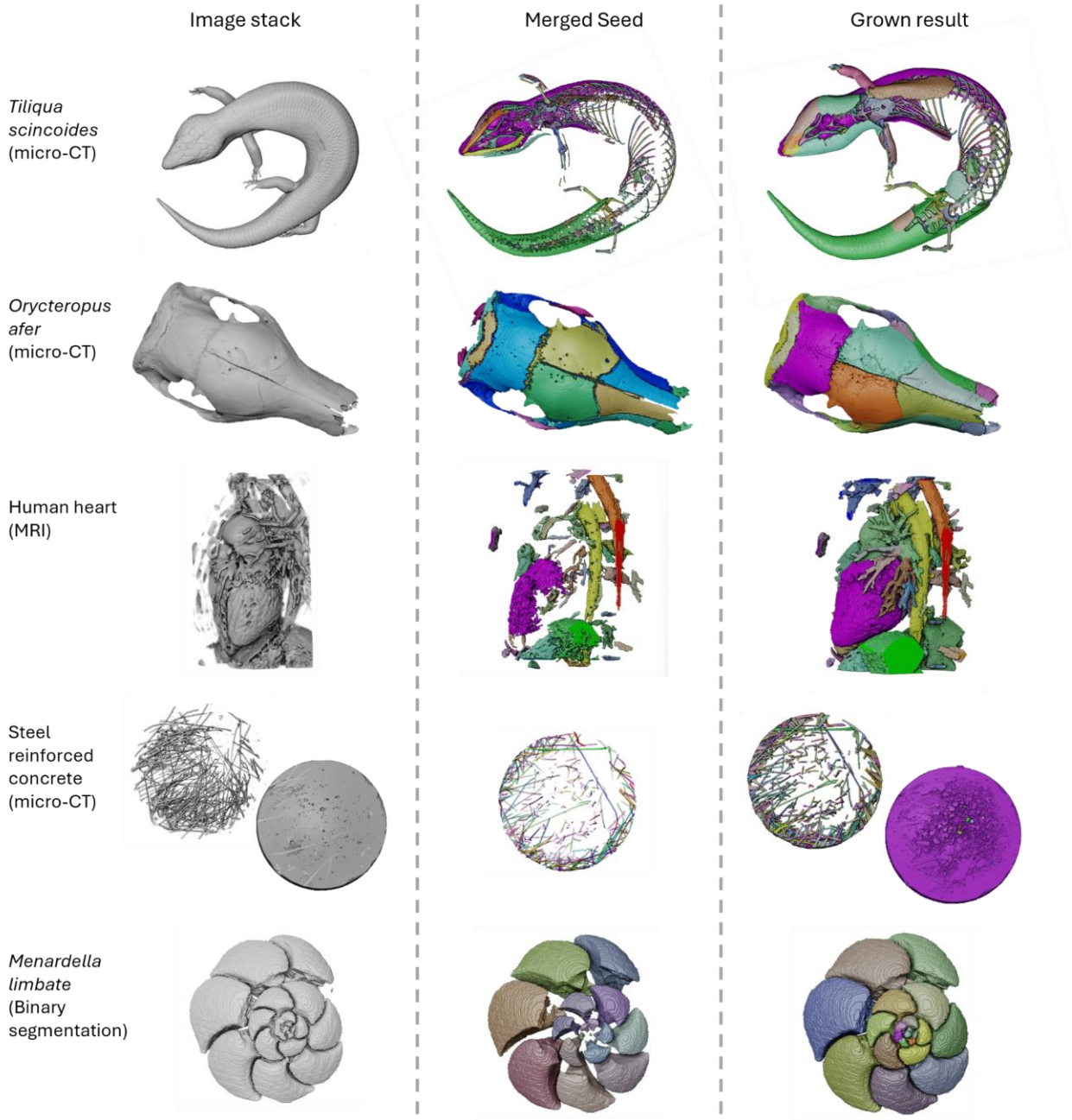

Figure 4. Volumetric representations of the image stacks, merged seeds, and grown results of SPROUT segmentations, using micro-CT, MRI, and binary segmentation inputs. Specimen details are in Table 2.

Structures with incomplete separations between segments: mammal skulls

Separating the individual bones of a mammalian skull can represent a significant challenge for segmentation, as there tends to be a large variability in the contact between bones and the grayscale values of the space in between bones, which can be further complicated by biological factors such as suture fusion. To solve these problems, the

SPROUT workflow requires specific alterations both in the seed generation and in the seed growth processes. In the aardvark skull example (*Orycteropus afer*: Fig. 4), the creation of a boundary layer mask was required (e.g., here generated using the 'tophat' function in Avizo) to help maintain a clearer separation between skull bones during seed generation. This boundary layer, which essentially enhances the separation between elements by creating continuity between the darker voxels of an image (note that bone is represented by the lighter voxels) is input in a modified version of the pipeline that keeps this mask in place when generating seeds. After a good seed is selected, the boundary mask is also input into the seed growth process to better represent the target elements. We apply morphological dilation to grow each element in the seed. A list of binary segmentations, segmented by a range of thresholds from high to low, is used to guide the dilation boundary. This approach regulates the growth of different elements, as binary segmentations from a high threshold have smaller areas to grow. As all seeds dilate at the same rate of expansion, this boundary layer prevents the overgrowth of one seed on top of another, acting as a stopper to growth and allowing the elements to maintain the required separation. This results in a skull with completely separated parts as the final product (Fig. 4).

Binary segmentations: deep learning generated images of planktonic foraminifera

As introduced in the seed generation section, SPROUT can also be applied to binary segmentations, which are simpler for deep learning and computer vision tasks as they are often used to distinguish background from the object of interest. In cases where specimens have a variable number of features, such as planktonic foraminifera with differing chamber counts, traditional deep learning techniques may struggle to segment datasets into individual regions accurately. Some chambers are connected together, so a direct disconnected component would not separate the chambers (Fig. 4). SPROUT addresses this by parcellating whole regions within binary segmentations into distinct regions, enabling the extraction of individual features such as chambers even when the number of classes varies across the dataset.

As described above, SPROUT typically begins by generating a binary segmentation using a thresholding approach. Erosions and disconnected component analysis are then applied to create seeds with separated regions. In this process, binary segmentations can be treated as grayscale images, where setting the threshold to 1 allows for the use of erosions to separate attached components. For the foraminifera binary segmentation dataset, we set the threshold to 1 and found that ten erosions effectively separated all connected chambers. Although high erosion may remove small chambers due to size variation, SPROUT's merged seed function effectively preserves them. This approach ensures consistent and fully automated input configurations for all foraminifera binary segmentations. Once regions are separated, dilation is applied to grow each region back

to its original size. For the foraminifera dataset, each region that is maintained in the final erosion step is dilated ten times to match the number of erosions, ensuring no overall alteration in the size of each component.

Discussion

In this paper, we have presented SPROUT, a new tool for the efficient semi-automated segmentation of 3D-image data. We demonstrated SPROUT's performance by applying it to diverse datasets from the biological, medical, and engineering sciences. Additionally, we compared the accuracy and time efficiency of SPROUT's results with those obtained using another widely used semi-automated method—smart interpolation.

Deep learning methods are considered to be the state-of-the-art for many image segmentation tasks due to their ability to learn complex patterns and generalize across diverse datasets (*14*, *49*). Compared to deep learning-based methods, SPROUT does not require large scan datasets or labelled training images, making it more accessible for domain-specific applications. Many deep learning-based image segmentation methods rely on extensive training data (*14*, *25*). While pre-trained deep learning networks may need little to no task-specific training data (*50*), their performance can still be limited when applied to highly specialised tasks.

In recent years, semi- and self-supervised methods have emerged, allowing unlabeled target data to be utilized for generating high-quality segmentations (*51–54*). However, these approaches may face significant challenges when dealing with domain-specific images with complex segmentation tasks, particularly for 3D images. Due to the limited number of specimens or the expense of scanning them, achieving sufficiently large dataset sizes to support effective use of these tools can be challenging. For example, in natural history collections, a dataset of even a few hundred 3D scans is often considered to be large (*6*, *47*), but is insufficient for semi- or self-supervised deep learning. In addition, tasks requiring fine-grained, multi-class segmentation pose additional challenges. Creating high-quality training datasets is time-intensive and requires people with domain knowledge, and building and tuning deep learning models for such purposes can be technically demanding (*55*). Finally, in terms of computational costs, SPROUT requires minimal processing power and can be run on CPU RAM without requiring Graphics Processing Unit (GPU) usage, making it more accessible to users with average hardware capabilities.

As a non-AI method, SPROUT integrates classic computer vision techniques such as thresholding, morphological transformations, connected components, and region growing. These well-established methods result in straightforward, interpretable input configurations that are easy for users to set, thereby offering a clear advantage of

foreseeing the result during method tuning in contrast to the more black-box design of deep learning-based predictions (*56*). With the useful features of merged seed generation and batch processing, SPROUT takes advantage of domain-specific datasets with patterns, enabling the pipeline to run automatically (as demonstrated with the dog limbs and planktonic foraminifera datasets detailed in Results). The implementation of parallelization further enhances the pipeline's efficiency, making it suitable for processing large-scale datasets. SPROUT's flexibility is also evident in its ability to handle diverse input data modalities (e.g., micro-CT, MRI, SEM), across a wide range of resolutions. Furthermore, it is explicitly designed to process datasets with any specified number of label classes, ensuring compatibility with a wide range of applications.

Additionally, the SPROUT pipeline is highly modular, being compatible in its implementation with imaging programs such as Avizo (Thermo Fisher Scientific, USA), Dragonfly (Object Research Systems, Canada), Fiji/ImageJ (*57*) and Meshlab (*58*). Each of these tools can be used for threshold selection, seed and growth checking and correction. As most of our workflow was implemented using Avizo for these steps, we include Avizo add-ons to make the pre-processing, loading, visualisation and segmentation editing in Avizo more efficient.

As demonstrated by our comparison using the dog limb dataset, SPROUT produces similar levels of accuracy to smart interpolation tools, such as that available in Biomedisa (*35*). However, interpolation requires the user to pre-segment slices from given image data, and the output will only include the elements that have been pre-segmented. By contrast, SPROUT requires minimal to no input data manipulation and dataset creation, making it unnecessary to use segmentation or visualisation software (e.g., Avizo, Dragonfly) before computation or at any point of the pipeline.

In addition, SPROUT is not only an alternative segmentation pipeline to deep learning methods, but can also serve as a tool for generating training data for deep learning models. This is particularly useful in scenarios where manual labeling of data is difficult or time-consuming. By leveraging SPROUT's capability to produce accurate, fine-grained segmentations without requiring labelled data or expensive hardware, researchers can efficiently create high-quality labelled datasets, which can then be used to train deep learning models for complex tasks, such as multi-class segmentation.

In terms of SPROUT's limitations, the generated outputs require assigning the correct segmentation component ID to each segmented element, as the raw SPROUT output IDs are currently ordered by component size. Therefore, class IDs have to be assigned manually. Several methods could help automate this process. Component IDs could be assigned using anatomical knowledge. For example, Brombacher et al. (59) assign IDs to foraminifera shell chambers based on the angles and sizes of each chamber.

Additionally, 3D image classification AI models could be trained to classify components into the correct IDs (*60*, *61*) based on appropriately labelled SPROUT outputs.

Memory limitations can also be a constraint when running SPROUT, as it requires loading the entire input image into memory for computation. This means that SPROUT cannot process scans larger than the available system memory. When using parallelization to process multiple scans simultaneously, the maximum size of the input image will decrease depending on the number of scans being processed at the same time. In contrast, the sliding window technique (*62*, *63*), is commonly employed in AI-based 3D image segmentation, allowing users to adjust window sizes to fit within the available GPU memory. However, RAM is much more affordable than GPU (*64*). For example, in the field of evolutionary morphology, most consumer-grade computers now are equipped with sufficient memory to handle the largest micro-CT scans of biological specimens. Consequently, memory size generally is not a significant limitation.

Ultimately, our paper demonstrates that SPROUT is a powerful tool for the efficient segmentation of 3D images. Through comparisons with other existing methods and demonstrations of its applications to various real data, we highlight the key advantages of this tool, as follows: (1) the ability to generate multi-class segmentations of 3D scans without requiring extensive datasets or training data; (2) provision of an efficient pipeline with easy-to-understand configurations, featuring tools such as merged seed generation, batch processing, and parallelization; (3) the flexibility to process a wide range of data types, resolutions, and variable label class numbers without significant alterations to the workflow or its speed. Furthermore, SPROUT is open-source, relies on common Python libraries, requires only moderate hardware capabilities, and features easily interpretable input configurations that demand limited domain knowledge, making it a significant advancement in the efficiency of 3D segmentation workflows. These advantages extend across multiple disciplines and data platforms, including biological, medical, geological, and engineering sciences, as well as potential applications in other, yet unforeseen fields. SPROUT can also serve as a valuable tool for generating training datasets for deep learning models, especially in scenarios where manually labelling data is time-consuming or impractical. By leveraging SPROUT's ability to produce accurate, fine-grained segmentations without the need for expensive computational devices, researchers can create high-quality labelled datasets efficiently. These datasets can then be used to train deep learning models for more complex tasks, such as multi-class segmentation or domain-specific applications.

Acknowledgements

We'd like to thank A. Sharp and E. Grisan for suggestions and technical insights which helped the development of SPROUT. We thank T. Macrini and G. Pandelis for contributing the aardvark and the skink scans to Digimorph and Morhosource, respectively. Foraminiferal specimens were scanned at the µ-VIS X-ray Imaging Centre at the University of Southampton. Dog scans courtesy of Liverpool Veterinary School Clinical Collaboration Platform, with thanks to T. Maddox. This work was funded by Leverhulme Trust grant RPG-2021-424, NERC grants NE/S007210/1 and NE/P019269/1, Royal Society APEX grant AA21\100124, and BBSRC grant BB/X014819/1. Ethics approval for dog scans granted by University of Liverpool Veterinary Research Ethics Committee (VREC), approval codes VREC1311 (non-greyhound breeds) and VREC1339 (greyhounds).

References

1. V. Cnudde, M. N. Boone, High-resolution X-ray computed tomography in geosciences: A review of the current technology and applications. *Earth-Science Reviews* **123**, 1–17 (2013).
2. Y. He, J. M. Mulqueeney, E. C. Watt, A. Salili-James, N. S. Barber, M. Camaiti, E. S. E. Hunt, O. Kippax-Chui, A. Knapp, A. Lanzetti, G. Rangel-de Lázaro, J. K. McMinn, J. Minus, A. V. Mohan, L. E. Roberts, D. Adhami, E. Grisan, Q. Gu, V. Herridge, S. T. S. Poon, T. West, A. Goswami, Opportunities and Challenges in Applying AI to Evolutionary Morphology. *Integrative Organismal Biology* **6**, obae036 (2024).
3. B. P. Hedrick, J. M. Heberling, E. K. Meineke, K. G. Turner, C. J. Grassa, D. S. Park, J. Kennedy, J. A. Clarke, J. A. Cook, D. C. Blackburn, S. V. Edwards, C. C. Davis, Digitization and the Future of Natural History Collections. *BioScience* **70**, 243–251 (2020).
4. R. S. Beaman, N. Cellinese, Mass digitization of scientific collections: New opportunities to transform the use of biological specimens and underwrite biodiversity science. *ZooKeys*, 7 (2012).
5. V. Blagoderov, I. Kitching, L. Livermore, T. Simonsen, V. Smith, No specimen left behind: industrial scale digitization of natural history collections. *ZK* **209**, 133–146 (2012).
6. C. R. Cooney, J. A. Bright, E. J. R. Capp, A. M. Chira, E. C. Hughes, C. J. A. Moody, L. O. Nouri, Z. K. Varley, G. H. Thomas, Erratum: Corrigendum: Mega-evolutionary dynamics of the adaptive radiation of birds. *Nature* **552**, 430–430 (2017).
7. J. J. Medina, J. M. Maley, S. Sannapareddy, N. N. Medina, C. M. Gilman, J. E. McCormack, A rapid and cost-effective pipeline for digitization of museum specimens with 3D photogrammetry. *PLoS ONE* **15**, e0236417 (2020).
8. J. M. Mulqueeney, A. Searle-Barnes, A. Brombacher, M. Sweeney, A. Goswami, T. H. G. Ezard, How many specimens make a sufficient training set for automated three-dimensional feature extraction? *R. Soc. Open Sci.* **11**, rsos.240113 (2024).
9. D. M. Boyer, G. F. Gunnell, S. Kaufman, T. M. McGeary, MORPHOSOURCE: ARCHIVING AND SHARING 3-D DIGITAL SPECIMEN DATA. *Paleontol. Soc. Pap.* **22**, 157–181 (2016).
10. R. S. Beaman, N. Cellinese, Mass digitization of scientific collections: New opportunities to transform the use of biological specimens and underwrite biodiversity science. *ZooKeys*, 7 (2012).
11. T. G. Davies, I. A. Rahman, S. Lautenschlager, J. A. Cunningham, R. J. Asher, P. M. Barrett, K. T. Bates, S. Bengtson, R. B. J. Benson, D. M. Boyer, J. Braga, J. A. Bright, L. P. A. M.

Claessens, P. G. Cox, X.-P. Dong, A. R. Evans, P. L. Falkingham, M. Friedman, R. J. Garwood, A. Goswami, J. R. Hutchinson, N. S. Jeffery, Z. Johanson, R. Lebrun, C. Martínez-Pérez, J. Marugán-Lobón, P. M. O'Higgins, B. Metscher, M. Orliac, T. B. Rowe, M. Rücklin, M. R. Sánchez-Villagra, N. H. Shubin, S. Y. Smith, J. M. Starck, C. Stringer, A. P. Summers, M. D. Sutton, S. A. Walsh, V. Weisbecker, L. M. Witmer, S. Wroe, Z. Yin, E. J. Rayfield, P. C. J. Donoghue, Open data and digital morphology. *Proceedings of the Royal Society B: Biological Sciences* **284**, 20170194 (2017).

12. A. Y. Hsiang, A. Brombacher, M. C. Rillo, M. J. Mleneck-Vautravers, S. Conn, S. Lordsmith, A. Jentzen, M. J. Henehan, B. Metcalfe, I. S. Fenton, B. S. Wade, L. Fox, J. Meilland, C. V. Davis, U. Baranowski, J. Groeneveld, K. M. Edgar, A. Movellan, T. Aze, H. J. Dowsett, C. G. Miller, N. Rios, P. M. Hull, Endless Forams: >34,000 Modern Planktonic Foraminiferal Images for Taxonomic Training and Automated Species Recognition Using Convolutional Neural Networks. *Paleoceanog and Paleoclimatol* **34**, 1157–1177 (2019).
13. M. D. Lürig, S. Donoughe, E. I. Svensson, A. Porto, M. Tsuboi, Computer Vision, Machine Learning, and the Promise of Phenomics in Ecology and Evolutionary Biology. *Front. Ecol. Evol.* **9**, 642774 (2021).
14. O. Ronneberger, P. Fischer, T. Brox, "U-net: Convolutional networks for biomedical image segmentation" in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science* (Springer, Munich, Germany, 2015)vol. 9351, pp. 234–241.
15. S. T. Schwartz, M. E. Alfaro, *Sashimi* : A toolkit for facilitating high-throughput organismal image segmentation using deep learning. *Methods Ecol Evol* **12**, 2341–2354 (2021).
16. G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. W. M. Van Der Laak, B. Van Ginneken, C. I. Sánchez, A survey on deep learning in medical image analysis. *Medical Image Analysis* **42**, 60–88 (2017).
17. H. D. Cheng, X. H. Jiang, Y. Sun, J. Wang, Color image segmentation: advances and prospects. *Pattern Recognition* **34**, 2259–2281 (2001).
18. L. Joskowicz, D. Cohen, N. Caplan, J. Sosna, Inter-observer variability of manual contour delineation of structures in CT. *Eur Radiol* **29**, 1391–1399 (2019).
19. M. Ali, S. O. Gilani, A. Waris, K. Zafar, M. Jamil, Brain Tumour Image Segmentation Using Deep Networks. *IEEE Access* **8**, 153589–153598 (2020).
20. K. K. D. Ramesh, G. K. Kumar, K. Swapna, D. Datta, S. S. Rajest, A Review of Medical Image Segmentation Algorithms. *EAI Endorsed Trans Perv Health Tech* **7**, e6 (2021).
21. X. Xie, T. Wan, B. Wang, L. Xu, X. Li, A Mechanical Parts Image Segmentation Method Against Illumination for Industry. In Review [Preprint] (2020). https://doi.org/10.21203/rs.3.rs-119471/v1.
22. C. Chen, C. Zhang, J. Wang, D. Li, Y. Li, J. Hong, Semantic segmentation of mechanical assembly using selective kernel convolution UNet with fully connected conditional random field. *Measurement* **209**, 112499 (2023).
23. P. Guntoro, Y. Ghorbani, P.-H. Koch, J. Rosenkranz, X-ray Microcomputed Tomography (µCT) for Mineral Characterization: A Review of Data Analysis Methods. *Minerals* **9**, 183 (2019).
24. H. Safari, B. J. Balcom, A. Afrough, Characterization of pore and grain size distributions in porous geological samples – An image processing workflow. *Computers & Geosciences* **156**, 104895 (2021).
25. F. Milletari, N. Navab, S.-A. Ahmadi, "V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation" in *2016 Fourth International Conference on 3D Vision (3DV)* (IEEE, Stanford, CA, USA, 2016; http://ieeexplore.ieee.org/document/7785132/), pp. 565–571.
26. M. J. Cardoso, W. Li, R. Brown, N. Ma, E. Kerfoot, Y. Wang, B. Murrey, A. Myronenko, C. Zhao, D. Yang, V. Nath, Y. He, Z. Xu, A. Hatamizadeh, W. Zhu, Y. Liu, M. Zheng, Y. Tang, I.

Yang, M. Zephyr, B. Hashemian, S. Alle, M. Z. Darestani, C. Budd, M. Modat, T. Vercauteren, G. Wang, Y. Li, Y. Hu, Y. Fu, B. Gorman, H. Johnson, B. Genereaux, B. S. Erdal, V. Gupta, A. Diaz-Pinto, A. Dourson, L. Maier-Hein, P. F. Jaeger, M. Baumgartner, J. Kalpathy-Cramer, M. Flores, J. Kirby, L. A. D. Cooper, H. R. Roth, D. Xu, D. Bericat, R. Floca, S. K. Zhou, H. Shuaib, K. Farahani, K. H. Maier-Hein, S. Aylward, P. Dogra, S. Ourselin, A. Feng, MONAI: An open-source framework for deep learning in healthcare. arXiv [Preprint] (2022). https://doi.org/10.48550/ARXIV.2211.02701.

27. F. Isensee, P. F. Jaeger, S. A. A. Kohl, J. Petersen, K. H. Maier-Hein, nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. *Nat Methods* **18**, 203–211 (2021).
28. M. Bardis, R. Houshyar, C. Chantaduly, A. Ushinsky, J. Glavis-Bloom, M. Shaver, D. Chow, E. Uchio, P. Chang, Deep Learning with Limited Data: Organ Segmentation Performance by U-Net. *Electronics* **9**, 1199 (2020).
29. P. A. Narayana, I. Coronado, S. J. Sujit, J. S. Wolinsky, F. D. Lublin, R. E. Gabr, Deep-Learning-Based Neural Tissue Segmentation of MRI in Multiple Sclerosis: Effect of Training Set Size. *Magnetic Resonance Imaging* **51**, 1487–1496 (2020).
30. D. Castelvecchi, Can we open the black box of AI? *Nature* **538**, 20–23 (2016).
31. R. Shwartz-Ziv, N. Tishby, Opening the Black Box of Deep Neural Networks via Information. arXiv [Preprint] (2017). https://doi.org/10.48550/ARXIV.1703.00810.
32. A. Ratner, C. De Sa, S. Wu, D. Selsam, C. Ré, Data Programming: Creating Large Training Sets, Quickly. doi: 10.48550/ARXIV.1605.07723 (2016).
33. E. Elyan, P. Vuttipittayamongkol, P. Johnston, K. Martin, K. McPherson, C. F. Moreno-García, C. Jayne, Md. Mostafa Kamal Sarker, Computer vision and machine learning for medical image analysis: recent advances, challenges, and way forward. *Art Int Surg*, doi: 10.20517/ais.2021.15 (2022).
34. S. S. Al-amri, N. V. Kalyankar, K. S. D., Image Segmentation by Using Threshold Techniques. doi: 10.48550/ARXIV.1005.4020 (2010).
35. P. D. Lösel, T. Van De Kamp, A. Jayme, A. Ershov, T. Faragó, O. Pichler, N. Tan Jerome, N. Aadepu, S. Bremer, S. A. Chilingaryan, M. Heethoff, A. Kopmann, J. Odar, S. Schmelzle, M. Zuber, J. Wittbrodt, T. Baumbach, V. Heuveline, Introducing Biomedisa as an open-source online platform for biomedical image segmentation. *Nat Commun* **11**, 5577 (2020).
36. O. Ramos-Soto, E. Rodríguez-Esparza, S. E. Balderas-Mata, D. Oliva, A. E. Hassanien, R. K. Meleppat, R. J. Zawadzki, An efficient retinal blood vessel segmentation in eye fundus images by using optimized top-hat and homomorphic filtering. *Computer Methods and Programs in Biomedicine* **201**, 105949 (2021).
37. V. Grau, A. U. J. Mewes, M. Alcaniz, R. Kikinis, S. K. Warfield, Improved Watershed Transform for Medical Image Segmentation Using Prior Information. *IEEE Trans. Med. Imaging* **23**, 447–458 (2004).
38. P. Parascandolo, L. Cesario, L. Vosilla, M. Pitikakis, G. Viano, “Smart Brush: A real time segmentation tool for 3D medical images” in *2013 8th International Symposium on Image and Signal Processing and Analysis (ISPA)* (IEEE, Trieste, Italy, 2013; http://ieeexplore.ieee.org/document/6703826/), pp. 689–694.
39. N. E. A. Khalid, S. Ibrahim, M. Manaf, U. K. Ngah, “Seed-based region growing study for brain abnormalities segmentation” in *2010 International Symposium on Information Technology* (IEEE, Kuala Lumpur, Malaysia, 2010; http://ieeexplore.ieee.org/document/5561560/), pp. 856–860.
40. N. O’Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, J. Walsh, “Deep Learning vs. Traditional Computer Vision” in *Advances in Computer Vision*, K. Arai, S. Kapoor, Eds. (Springer International Publishing, Cham, 2020; http://link.springer.com/10.1007/978-3-030-17795-9_10)vol. 943 of *Advances in Intelligent Systems and Computing*, pp. 128–144.

41. M. Didziokas, E. Pauws, L. Kölby, R. H. Khonsari, M. Moazen, BounTI (boundary-preserving threshold iteration): A user-friendly tool for automatic hard tissue segmentation. *Journal of Anatomy* **245**, 829–841 (2024).
42. L. Friedli, D. Kloukos, G. Kanavakis, D. Halazonetis, N. Gkantidis, The effect of threshold level on bone segmentation of cranial base structures from CT and CBCT images. *Sci Rep* **10**, 7361 (2020).
43. R. Adams, L. Bischof, Seeded region growing. *IEEE Trans. Pattern Anal. Machine Intell.* **16**, 641–647 (1994).
44. S. Pieper, M. Halle, R. Kikinis, "3D Slicer" in *2004 2nd IEEE International Symposium on Biomedical Imaging: Macro to Nano (IEEE Cat No. 04EX821)* (IEEE, Arlington, VA, USA, 2004; http://ieeexplore.ieee.org/document/1398617/)vol. 2, pp. 632–635.
45. E. Maire, P. J. Withers, Quantitative X-ray tomography. *International Materials Reviews* **59**, 1–43 (2014).
46. P. J. Withers, C. Bouman, S. Carmignato, V. Cnudde, D. Grimaldi, C. K. Hagen, E. Maire, M. Manley, A. Du Plessis, S. R. Stock, X-ray computed tomography. *Nat Rev Methods Primers* **1**, 18 (2021).
47. E. C. Hughes, D. P. Edwards, J. A. Bright, E. J. R. Capp, C. R. Cooney, Z. K. Varley, G. H. Thomas, Global biogeographic patterns of avian morphological diversity. *Ecology Letters* **25**, 598–610 (2022).
48. P. Lösel, V. Heuveline, "Enhancing a diffusion algorithm for 4D image segmentation using local information" in *Medical Imaging 2016: Image Processing* (SPIE, 2016; https://www.spiedigitallibrary.org/conference-proceedings-of-spie/9784/97842L/Enhancing-a-diffusion-algorithm-for-4D-image-segmentation-using-local/10.1117/12.2216202.full)vol. 9784, pp. 707–717.
49. A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, R. Girshick, Segment Anything. arXiv [Preprint] (2023). https://doi.org/10.48550/ARXIV.2304.02643.
50. K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition. arXiv [Preprint] (2015). https://doi.org/10.48550/ARXIV.1512.03385.
51. D. Lin, J. Dai, J. Jia, K. He, J. Sun, "ScribbleSup: Scribble-Supervised Convolutional Networks for Semantic Segmentation" in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Las Vegas, NV, USA, 2016; http://ieeexplore.ieee.org/document/7780713/), pp. 3159–3167.
52. T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A Simple Framework for Contrastive Learning of Visual Representations. arXiv [Preprint] (2020). https://doi.org/10.48550/ARXIV.2002.05709.
53. K. He, X. Chen, S. Xie, Y. Li, P. Dollár, R. Girshick, Masked Autoencoders Are Scalable Vision Learners. arXiv [Preprint] (2021). https://doi.org/10.48550/ARXIV.2111.06377.
54. Z. Li, Y. Zheng, X. Luo, D. Shan, Q. Hong, "ScribbleVC: Scribble-supervised Medical Image Segmentation with Vision-Class Embedding" in *Proceedings of the 31st ACM International Conference on Multimedia* (ACM, Ottawa ON Canada, 2023; https://dl.acm.org/doi/10.1145/3581783.3612056), pp. 3384–3393.
55. Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, J. Liang, UNet++: Redesigning Skip Connections to Exploit Multiscale Features in Image Segmentation. *IEEE Trans. Med. Imaging* **39**, 1856–1867 (2020).
56. Z. Salahuddin, H. C. Woodruff, A. Chatterjee, P. Lambin, Transparency of deep neural networks for medical image analysis: A review of interpretability methods. *Computers in Biology and Medicine* **140**, 105111 (2022).
57. T. J. Collins, ImageJ for Microscopy. *BioTechniques* **43**, 5 (2007).
58. P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia, MeshLab: an Open-Source Mesh Processing Tool. The Eurographics Association [Preprint] (2008).

https://doi.org/10.2312/LOCALCHAPTEREVENTS/ITALCHAP/ITALIANCHAPCONF2008/129-136.
59. A. Brombacher, A. Searle-Barnes, W. Zhang, T. H. G. Ezard, Analysing planktonic foraminiferal growth in three dimensions with foram3D: an R package for automated trait measurements from CT scans. *J. Micropalaeontol.* **41**, 149–164 (2022).
60. H. R. Roth, C. T. Lee, H.-C. Shin, A. Seff, L. Kim, J. Yao, L. Lu, R. M. Summers, "Anatomy-specific classification of medical images using deep convolutional nets" in *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)* (IEEE, Brooklyn, NY, USA, 2015; http://ieeexplore.ieee.org/document/7163826/), pp. 101–104.
61. P. Raffy, J.-F. Pambrun, A. Kumar, D. Dubois, J. W. Patti, R. A. Cairns, R. Young, Deep Learning Body Region Classification of MRI and CT Examinations. *J Digit Imaging* **36**, 1291–1301 (2023).
62. C. K. Koç, Analysis of sliding window techniques for exponentiation. *Computers & Mathematics with Applications* **30**, 17–24 (1995).
63. A. Serag, A. G. Wilkinson, E. J. Telford, R. Pataky, S. A. Sparrow, D. Anblagan, G. Macnaught, S. I. Semple, J. P. Boardman, SEGMA: An Automatic SEGMentation Approach for Human Brain MRI Using Sliding Window and Random Forests. *Front. Neuroinform.* **11** (2017).
64. J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, J. C. Phillips, GPU Computing. *Proc. IEEE* **96**, 879–899 (2008).