**SegmentR: Deep Learning for Automated Segmentation with an R interface**

James D. Boyko12   
1 Department of Ecology and Evolutionary Biology, University of Michigan, Ann Arbor, Michigan 48109, USA  
2 Michigan Institute of Data Science, University of Michigan, Ann Arbor, Michigan 48109, USA

**Abstract**

The increasing digitization of biological data has generated biodiversity data at an unprecedented scale. However, extracting phenotypic information from these images at a large scale poses unique challenges for biologists. Manual image segmentation is time-consuming and can be subjective, while existing automated solutions often requires extensive coding experience or utilize coding languages not typically used by practicing ecologists and evolutionary biologists. Here, I present SegmentR, a user-friendly software package that leverages two state-of-the-art deep learning models – GroundinDINO and SegmentAnything. The SegmentR package provides an R-based interface, making it more accessible to biologists without coding experience. The SegmentR pipeline allows users to load images, automatically segment them based on text prompts, and extract regions of interest for downstream analysis. The package includes basic visualization and data processing functions to facilitate interpretation of the results and integration with existing analytical workflows. I demonstrate the utility of the SegmentR package by using it to automatically isolate the body of a fish from its fins and extract color information from a small dataset of flower images.

**Key Words**

citizen science, image segmentation, deep learning, iNatauralist, object detection

**1. INTRODUCTION**

The proliferation of digital imaging in biology has generated vast quantities of visual data across scales and systems. Citizen science initiatives like iNaturalist, massive digitization efforts of research museums, and international collaborative networks like Global Biodiversity Information Facility (GBIF) are centralizing vast quantities of biodiversity data and providing opportunity to study phenotypic variation across the tree of life and at large spatial scales. Yet, the sheer scale of the data generated and the fact that it is often not standardized, make it difficult to automate data collection and extraction. As such, most attempts to analyze biological images rely on manual segmentation and annotation. Though this approach can produce high-quality datasets, it becomes prohibitively time-consuming at scale.

One promising solution to the scale challenge is the utilization of computer vision and deep learning techniques to automate large parts of the image processing pipeline. Many successful deep learning applications in recent years have focused on computer vision tasks (LeCun et al. 2015; Goodfellow et al. 2016) and two key areas particularly relevant for biological image analysis are object detection and instance segmentation. Object detection algorithms aim to locate and classify distinct objects within an image (Redmon et al. 2016; Ren et al. 2017). Specifically, the goal of object detection is to draw bounding boxes around particular objects of interest and associate the bounding box with a particular class for a given image. For biological images, this could mean identifying and drawing boxes around all instances of a leaf in herbarium specimens (Weaver and Smith 2023), individual cells in fluorescence microscopy (Waithe et al. 2020), or individual bones from bird specimens (Weeks et al. 2023). However, most object detection algorithms are limited to a pre-determined set of classes because they are trained with a particular task in mind – i.e., one would not use the leaf object detector to identify instances of a bird femur. This can be problematic for biological datasets as existing pre-trained models are unlikely to generalize well to taxonomic groups outside of their training data. One way that object detectors have successfully generalized is by becoming multimodal (Liu et al. 2024). This has primarily been pursued by introducing a language module into the vision models. For example, Grounding DINO has combined visual and textual modalities and has succeeded in generalizing object detection to a greater extent than previous attempts (Liu et al. 2024). Grounding DINO is a transformed-based architecture that fuses language and vision modalities by linking the closed-set detector, DINO (Zhang et al. 2022), with grounded language-based pre-training (e.g., GLIP; Li et al. 2022). More practically, the effect of this design is that Grounding DINO is able to detect objects based on text prompts.

Though object detection is useful, it often is only the first step of biological image data analysis. Identifying instances of a particular thing and locating them within the image is often done so that we can then isolate those instances from the rest of the image. This is where segmentation becomes a crucial step. Segmentation models delineate pixel-level boundaries of each instance of an object (He et al. 2017; Kirillov et al. 2020). One such model is SegmentAnything (SAM), an image segmentation model trained on the largest segmentation dataset with over 1 billion masks and 11 million images (Kirillov et al. 2022). The large training dataset means that it achieves consistently high performance on zero-shot segmentation tasks (i.e., high performance on tasks it was not directly trained on) even when compared to fully supervised models (Kirillov et al. 2022). Importantly, segmentation models can take bounding boxes as input and segment objects at a particular location. This has led to the combining of Grounding DINO and SAM into GroundedSAM (Ren. et al 2024). This pipeline uses the bounding box output of Grounding DINO as the input of SAM for high quality instance segmentation. This approach can be further refined by using recently developed efficient versions of SAM such as SlimSAM, which achieve high accuracy while using far less training data (Chen et al. 2024). SlimSAM result in a model a fraction of the size of the original SAM (1.4% of the original parameters) and is ideally suited for biological research as the workflow can be run on moderately powerful personal computers. Thus, by combining the object detection and segmentation tasks, it becomes possible to focus analyses only on the specific regions of interest, rather than the entire scene.

In this work, I utilize two such pre-trained models: GroundingDINO (Liu et al. 2024) for object detection and a data efficient version of the Segment Anything Model (SAM) (Kirillov et al. 2022; Chen et al. 2024) for instance segmentation. To provide an accessible entry point for ecologists and evolutionary biologists, I have implemented this deep learning-powered pipeline as an R package called SegmentR. The R framework gives users who are already familiar with the R programming language an easier integration with their existing workflows and analytical tools. The base SegmentR package performs object detection and instance segmentation allowing for individual instances of focal taxa to be extracted from an image and utilized in downstream analyses.

**2. METHODS AND RESULTS**

**2.1 SegmentR features and examples**

*2.1.1 SegmentR installation*

SegmentR is based on a python implementation of pre-trained deep learning models. As such, users must have python3 installed on their computers. Furthermore, to install SegmentR one must have the conda package and environment manager. Thus, once python is installed, the user should install either Anaconda3 or Miniconda. Once these prerequisites are installed, SegmentR can be installed from github with the use of devtools (Wickham et al. 2022) and the command install\_github(“jboyko/SegmentR”). Once the package has been installed, the function setup\_conda\_environment() can be used to install the necessary python dependencies. setup\_conda\_environment will install all the necessary libraries, though setting up the environment may take minutes as library versions must be solved to ensure compatibility. During the conda environment setup, a warning message may appear if the user does not have the rust programming language installed on their computer. This is related to the transformers library, but can be safely ignored as the functions being used from transformers for SegmentR do not require rust.

*2.1.2 Description of the SegmentR pipeline*

Using SegmentR requires a user to input an image (specified by the path to the image) and a set of labels. The primary function of SegmentR, run\_grounded\_segmentation (Table 1), then preforms grounded segmentation based on the input image and labels. This function will also accept a path to a particular directory as input. When provided with a directory path, SegmentR automatically processes all PNG and JPEG images within that location as a batch. Users can specify additional image formats using the “pattern” argument (see Table 1). The package implements segmentation through a custom command-line interface (CLI) that communicates with a Python backend, offering significant performance advantages over traditional R-Python integrations. This choice is made because of the processing limitations of packages like reticulate. Using a command-line interface enables parallel execution of object detection and segmentation tasks for improved efficiency. The Python back-end then uses the transformers library, which is itself an API to download and train pre-trained models, to call the segmenter and detector models (Wolf et al. 2020). Note that once the desired models are downloaded and necessary libraries installed, the SegmentR software can be used entirely offline. Results are saved as a JSON file which can then be read into R using the function load\_segmentation\_results. Once the grounded segmentation is complete and the results are loaded into R, the segmentation can be plotted using plot\_seg\_results and subsequent analysis can be conducted on the resulting masks. Finally, the segmentation results can be exported as transparent PNG files using export\_transparent\_png, which provides options for mask processing including overlap removal, cropping to segment boundaries, and customizable file naming conventions (Table 1).

Table 1: Primary functions.

|  |  |  |
| --- | --- | --- |
| **Function** | **Arguments** | **Definition** |
| run\_grounded\_segmentation | path | Character string. Path to an input image or directory containing images. E.g., “/home/user\_name/project\_directory/images/” |
| labels | Character vector. Labels to detect in the image.  E.g., c(“a flower”, “a tree”). Labels are automatically appended with a “.” because it substantially improves the performance of groundingDINO. |
| threshold | Numeric. The minimum detection threshold for which detected objects will be segmented (default: 0.3). |
| detector\_id | Character string. ID of the detector model.  default: “IDEA-Research/grounding-dino-tiny” |
| segmenter\_id | Character string. ID of the detector model.  default: “Zigeng/SlimSAM-uniform-77” |
| output\_plot | Character string. Path or directory to save the output plot. |
| output\_json | Character string. Path or directory to save the output JSON. |
| show\_plot | Boolean. Whether a plot should automatically be shown after a segmentation. |
| create\_dir | Boolean. If the output directory doesn't exist, one will be created. |
| pattern | Character string. File pattern to match when path is a directory (default: "\\.(jpg|jpeg|png)$"). |
| recursive | Boolean. Whether to search for images recursively in subdirectories (default: FALSE). |
| conda\_env | Character string. Name of the conda environment to use.  default: “SegmentR-env” created by setup\_conda\_environment |
| load\_segmentation\_results | image\_path | Character string. Path to the original image file. |
| json\_path | Character string. Path to the JSON file containing segmentation results. The JSON file is created by grounded\_segmentation\_cli |
| export\_transparent\_png | input | Either a seg\_results list or image data (cimg object, array, or file path) |
| masks | Optional list of masks or single mask array (if not provided in seg\_results) |
| labels | Optional vector of labels for each mask |
| scores | Optional vector of confidence scores |
| output\_path | Character. Path where files should be saved. |
| score\_threshold | Numeric. Threshold for including results (0-1) |
| remove\_overlap | Boolean. Whether to remove any overlapping regions from masks with different labels. |
| return\_binary | Boolean. Whether only a binary mask should be returned. |
| crop | Boolean. Whether to crop the image to the edges of the segment. |
| prefix | Character. Prefix for output filenames (default: NULL uses "segment") |
| include\_score | Logical. Whether to include confidence score in filename (default: TRUE) |
| id\_padding | Integer. Number of digits to pad mask IDs with (default: 3) |
| plot\_seg\_results | seg\_results | A list containing segmentation results (image, label, score, box, mask). |
| mask\_colors | A named vector of colors for each label, or a color palette name from RColorBrewer. |
| background | One of "original", "grayscale", "transparent", or a specific color. |
| show\_label | Boolean Whether to display labels |
| show\_score | Boolean Whether to display scores |
| show\_bbox | Boolean Whether to display bounding boxes. |
| ... | Additional arguments to be passed to plot. |

*Example 1 – Specifying particular parts of an organism*

I demonstrate the ability of groundingDINO to specify and isolate a particular part of an organism. The subject of this analysis is a horned bream and the challenge lies in excluding the fins from the rest of the body. By leveraging the broad textual understanding provided by groundingDINO, we can use text prompts to detect and exclude various fin types from the analysis (Figure 1). The segmentation pipeline begins with a simple call to run\_grounded\_segmentation(), using two text prompts: “a fish.” and “a fin.” Although these prompts are generic, the model is successfully able distinguish the anatomical feature. After loading the segmentation results, I use the remove\_mask() function to exclude a poorly detected feature - in this case, an incorrectly identified fin (Figure 1e). The resulting segmentation can be exported in two formats: with overlap removal for clean segment separation, or without overlap removal to preserve the original detection boundaries. Both export options utilize cropping to minimize empty space in the output images. This example highlights both the strengths and limitations of the current implementation. While the algorithm successfully identified and excluded the more prominent caudal and pelvic fins, it struggled with the detection of dorsal, anal, and pectoral fins, suggesting that more specific prompts or fine-tuning might be necessary for complete fin detection.

A close-up of a fish

AI-generated content may be incorrect.

Figure - Example 1: isolating part of an organism. a) original image. b) The result of run\_grounded\_segmentation(). Plot produced by plot\_seg\_results(). The bounding boxes are generated by groundingDINO and segmentation (as indicated by the coloration of the fish and fins) is generated by SAM. c) All exported masks isolated from the background as produced by export\_transparent\_png(). d)The isolated fins and body the fish - also produced by export\_transparent\_png() with the remove\_overlap argument set to TRUE. e) the R code used to generate the segmentation.

*Example 2 – Batch segmentation of a small dataset*

In this example, I demonstrate SegmentR's batch processing capabilities and its integration into a broader image analysis pipeline. Using a dataset of four flower images sourced from iNaturalist, I showcase how SegmentR can serve as an efficient preprocessing step for color analysis using the recolorize package (Weller et al. 2024). The workflow begins with run\_grounded\_segmentation(), which processes all images in the specified directory using a single prompt: “an individual flower.” The get\_segmentation\_paths() function then retrieves the paths to all processed images, enabling batch loading of segmentation results. The subsequent automated processing, from image import to the export of transparent PNGs, completed in 39 seconds for all four images on a MacStudio with 96GB of memory and an Apple M2 Max chip (12 CPU cores). Each segment is exported as a cropped, transparent PNG with overlap removal to ensure clean isolation of individual flowers. The pipeline includes a manual selection step to identify the most suitable masks for color analysis. Although this is a manual step, it is likely still more efficient than manual flower delineation on large batches of images. Importantly, SegmentR facilitates reproducibility by automatically generating a comprehensive metadata table for each segmentation. This table includes image names, mask IDs, detection labels, confidence scores, bounding box coordinates, and mask dimensions. The structured format of this metadata allows users to document their mask selection process and maintain a record of the segmentation parameters, ensuring transparency and reproducibility in subsequent analyses. For instance, users can filter masks based on confidence scores or use bounding box coordinates to verify spatial relationships between detected objects.  
  
A collage of different flowers

AI-generated content may be incorrect.

Figure - Example 2: batch processing of four images. a) Each column is a different image sourced from iNaturalist under the CC-BY-NC license (from left to right the IDs of each image are 249435672, 243695619, 243146103, and 242477360). The top row is the original image. The second row shows bounding boxes around what was detected as “an individual flower.” The third row is the flower isolated and cropped. In cases where more than one flower was detected, the cropped images were manually selected for the highest quality. The final row is the output of recluster() from the recolorize package, with a legend showing the color pallete detected from the cropped and isolated flower image. b) the R code used to produce the segmentations and export individual parts.

**3. DISCUSSION**

The proliferation of digital imaging has led to an unprecedented accumulation of biodiversity data. However, the heterogeneous nature of these datasets presents significant challenges for standardization and analysis. Recent advancements in computer vision technologies have emerged as a promising solution to address these complexities. Thus, a powerful, accessible, and environmentally responsible way forward for academics may be the utilization of large pre-trained models. These models are trained on extensive datasets which few academic collaborations are able to match (e.g., the billion masks used to train SAM; Krillov et al., 2023).

The zero-shot object detection capabilities of groundingDINO (Feng et al., 2023) offer significant advantages when processing non-standardized citizen science data. The heterogeneous nature of images on platforms like iNaturalist, where organisms may be positioned anywhere within the frame, camouflaged against diverse backgrounds, or present alongside other species, highlights the value of general text-prompt-based object detection. However, it is crucial to understand the inherent limitations of these detection algorithms in biological applications. The examples presented in this paper primarily utilize common organisms (flowers and fish) that were likely well-represented in the training datasets of both groundingDINO and SAM (Segment Anything Model). While this suggests reliable performance for frequently photographed organisms, such as those commonly documented on citizen science platforms, the algorithms may exhibit reduced accuracy when applied to specialized scientific imagery. For instance, paleontological specimens, microscopy data, or museum collections may present unique challenges due to their absence from the models’ training data.

Nevertheless, even in scenarios where detection accuracy is suboptimal, SegmentR's automated approach can still offer significant time savings compared to manual segmentation. The algorithm's two-stage process - object detection followed by segmentation within the detected bounding box - means that even when text-prompt detection is imperfect, the subsequent segmentation step can still provide useful results. The time required to manually adjust a bounding box is substantially less than that needed for precise manual delineation of organism boundaries. However, the bounding box-based approach introduces new limitations. A notable constraint is the algorithm's inability to handle internal voids. For example, when processing images of coiled snakes or organisms with natural apertures, the bounding box methodology will include all pixels within the box's boundaries, potentially incorporating undesired background regions. This limitation could potentially be addressed through integration with specialized background removal algorithms or post-processing techniques.

A final consideration concerns the scalability of the R implementation. While SegmentR efficiently handles batch processing through separate export operations, many of its utility functions are optimized for scenarios where all images can be loaded into memory simultaneously. This approach may become problematic when processing large image datasets, as R's memory management may become a bottleneck. Although future package refinements will address some of these R-specific constraints, users working with extensive image collections may benefit from utilizing Python-based implementations that offer better memory management and processing controls.

**4. CONCLUSION**

The implementation of deep learning tools in R is an important step towards making these techniques widely available to biologists. R is one of the most widely used programming language for academic ecologists and evolutionary biologists, but most deep-learning developments take place in Python. Furthermore, the use of light-weight models, such as slimSAM (Chen 2024), allows for advanced deep-learning models to be run even on moderately powerful personal computers. By enhancing the accessibility of these tools, biologists can gain access to an increasing number of data sources. Nonetheless the automation provided by computer vision techniques will need to be balanced with careful verification of the results. Performance is not guaranteed to be the same across all images and increasing the accuracy of these models on diverse taxonomic groups will likely require some amount of additional data collection and fine-tuning (Gu et al. 2024). Furthermore, while SegmentR allows for a more automated collection of citizen science data, it cannot address all inherent limitations of this data source. Researchers must remain cognizant of sampling biases when interpreting results derived from these datasets. Researchers must reconcile the fact that though these tools are able to automatically extract vast quantities of data, high quality datasets will only be created if they are rigorously evaluated. As deep-learning techniques continue to integrate with ecology and evolutionary biology, it is important that they are used to complement traditional methods, rather than replace them.

**Acknowledgments**

The author thanks 2 anonymous reviewers and the AE for their detailed feedback that greatly focused and improved both the software and the manuscript. The author also thanks Thais Vasconcelos for her thoughtful feedback and testing of earlier versions of the software. This project is supported by the Eric and Wendy Schmidt AI in Science Postdoctoral Fellowship, a Schmidt Sciences Futures program.

**Data Availability Statement**

The development version of the package is available on GitHub: https://github.com/jboyko/SegmentR. All images and code used to generate the examples are taken from the vignette associated with the package. Data and image credits can be loaded using SegmentR::load\_segmentr\_example\_data.

**Conflict of Interest Statement**

The author declares no competing interests.

**References**

Chen, Z., Fang, G., Ma, X. & Wang, X. (2024). SlimSAM: 0.1% Data Makes Segment Anything Slim.

Chollet, F. (2021). *Deep learning with Python*. Second edition. Manning Publications, Shelter Island.

Cox, C. & Davis Rabosky, A. (2023). The Integrative Biology of Snake Coloration. pp. 179–217.

Dickinson, J.L., Zuckerberg, B. & Bonter, D.N. (2010). Citizen Science as an Ecological Research Tool: Challenges and Benefits. *Annual Review of Ecology, Evolution, and Systematics*, 41, 149–172.

Endler, J.A. (1993). The Color of Light in Forests and Its Implications. *Ecological Monographs*, 63, 1–27.

Finkbeiner, S.D., Briscoe, A.D. & Reed, R.D. (2014). Warning signals are seductive: Relative contributions of color and pattern to predator avoidance and mate attraction in *Heliconius* butterflies: BUTTERFLY VISUAL SIGNALS. *Evolution*, 68, 3410–3420.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning*. MIT Press.

Gu H., Dong H., Yang J., Mazurowski M.A. 2024. How to build the best medical image segmentation algorithm using foundation models: a comprehensive empirical study with Segment Anything Model.

He, K., Gkioxari, G., Dollár, P. & Girshick, R. (2018). Mask R-CNN.

Huyghe, K., Vanhooydonck, B., Herrel, A., Tadic, Z. & Van Damme, R. (2007). Morphology, performance, behavior and ecology of three color morphs in males of the lizard Podarcis melisellensis. *Integrative and Comparative Biology*, 47, 211–220.

Internationale Beleuchtungskommission (Ed.). (2004). *Colorimetry*. Publication / CIE. 3rd ed. Comm. Internat. de l’éclairage, Wien.

Kirillov, A., Wu, Y., He, K. & Girshick, R. (2020). PointRend: Image Segmentation as Rendering.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., *et al.* (2023). Segment Anything.

LeCun, Y., Bengio, Y. & Hinton, G. (2015). Deep learning. *Nature*, 521, 436–444.

Li, L.H., Zhang, P., Zhang, H., Yang, J., Li, C., Zhong, Y., *et al.* (2022). Grounded Language-Image Pre-training.

Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., *et al.* (2024). Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. University of California Press, pp. 281–298.

Maia, R., Gruson, H., Endler, J.A. & White, T.E. (2019). pavo 2: New tools for the spectral and spatial analysis of colour in r. *Methods in Ecology and Evolution*, 10, 1097–1107.

Mitchem, L.D. (2017). The ecology of color: how color affects visual behaviors in aquatic habitats in a sexually dimorphic species (Lucania goodei) and their major predator (Micropterus salmoides).

Murali, S. & Govindan, V.K. (2013). Shadow Detection and Removal from a Single Image Using LAB Color Space. *Cybernetics and Information Technologies*, 13, 95–103.

Narbona, E., Arista, M., Whittall, J.B., Camargo, M.G.G. & Shrestha, M. (2021). Editorial: The Role of Flower Color in Angiosperm Evolution. *Front. Plant Sci.*, 12.

Pohland, G. & Mullen, P. (2006). Preservation agents influence UV-coloration of plumage in museum bird skins. *J Ornithol*, 147, 464–467.

Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection.

Ren, S., He, K., Girshick, R. & Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.

Ren, T., Liu, S., Zeng, A., Lin, J., Li, K., Cao, H., *et al.* (2024). Grounded SAM: Assembling Open-World Models for Diverse Visual Tasks.

Szeliski, R. (2022). *Computer Vision: Algorithms and Applications*. Springer Nature.

Tsuchida, T., Koga, R., Horikawa, M., Tsunoda, T., Maoka, T., Matsumoto, S., *et al.* (2010). Symbiotic Bacterium Modifies Aphid Body Color. *Science*, 330, 1102–1104.

Van Belleghem, S.M., Papa, R., Ortiz-Zuazaga, H., Hendrickx, F., Jiggins, C.D., Owen McMillan, W., *et al.* (2018). patternize: An R package for quantifying colour pattern variation. *Methods in Ecology and Evolution*, 9, 390–398.

Vasconcelos, T. (2023). A trait-based approach to determining principles of plant biogeography. *American Journal of Botany*, 110, e16127.

van den Berg, C.P., Troscianko, J., Endler, J.A., Marshall, N.J. & Cheney, K.L. (2020). Quantitative Colour Pattern Analysis (QCPA): A comprehensive framework for the analysis of colour patterns in nature. *Methods in Ecology and Evolution*, 11, 316–332.

Ward, D.F. (2014). Understanding sampling and taxonomic biases recorded by citizen scientists. *J Insect Conserv*, 18, 753–756.

Weaver, W.N. & Smith, S.A. (2023). From leaves to labels: Building modular machine learning networks for rapid herbarium specimen analysis with LeafMachine2. *Applications in Plant Sciences*, 11, e11548.

Weller, H.I., Hiller, A.E., Lord, N.P. & Van Belleghem, S.M. (2024). recolorize: An R package for flexible colour segmentation of biological images. *Ecology Letters*, 27, e14378.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., *et al.* (2020). Transformers: State-of-the-Art Natural Language Processing. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, pp. 38–45.

Zhang, H., Li, F., Liu, S., Zhang, L., Su, H., Zhu, J., *et al.* (2022). DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection.