Digital Discovery

ROYAL SOCIETY OF CHEMISTRY

PAPER

View Article Online
View Journal | View Issue

Cite this: *Digital Discovery*, 2026, **5**, 835

LivePyxel: accelerating image annotations with a Python-integrated webcam live streaming

Uriel Garcilazo-Cruz,*[ab] Joseph O. Okeme [ab] and Rodrigo A. Vargas-Hernández *[abc]

The lack of flexible annotation tools has hindered the deployment of AI models in some scientific areas. Most existing image annotation software requires users to upload a precollected dataset, which limits support for on-demand pipelines and introduces unnecessary steps to acquire images. This constraint is particularly problematic in laboratory environments, where on-site data acquisition from instruments such as microscopes is increasingly common. In this work, we introduce LivePixel, a Python-based graphical user interface that integrates with imaging systems, such as webcams, microscopes, and others, to enable on-site image annotation. LivePyxel is designed to be easy to use through a simple interface that allows users to precisely delimit areas for annotation using tools commonly found in commercial graphics editing software. Of particular interest is the availability of Bézier splines and binary masks, and the software's capacity to work with non-destructive layers that enable high-performance editing. LivePyxel also integrates a wide compatibility across video devices, and it's optimized for object detection operations *via* the use of OpenCV in combination with high-performance libraries designed to handle matrix and linear algebra operations *via* Numpy effectively. LivePyxel facilitates seamless data collection and labeling, accelerating the development of AI models in experimental workflows. LivePyxel is freely available at https://github.com/UGarCil/LivePyxel.

Received 18th September 2025
Accepted 11th January 2026

DOI: 10.1039/d5dd00421g

rsc.li/digitaldiscovery

1 Introduction

The capabilities of vision models (ViM) depend critically on the quality and availability of images.[1–7] However, collecting high-quality annotated images is time consuming and this annotation bottleneck[8] negatively impacts the integration of ViM in highly specialized scientific domains, such as cellular imaging,[9,10] civil infrastructure,[11] crop profile characterization,[12] environmental microscopy[13] and marine conservation[14] to name a few. To address these data limitations, researchers have developed alternative approaches, including data augmentation schemes[15] and more powerful models capable of segmenting and classifying diverse object types with minimal training (*e.g.*, the segment anything model (SAM)).[16] However, these techniques still rely on the precision and accuracy of the initial annotations of domain experts. Ensuring high-quality specialized datasets requires the input of these experts,[17] which can be both a limiting factor and financially costly.[18] The increasing availability of affordable, high-resolution imaging hardware highlights the importance of easy-to-use open-source software. Such tools can lower technical barriers and facilitate broader participation from individuals who possess domain knowledge but lack the expertise to collect and preprocess large datasets systematically.

In the fields of object detection and image segmentation, accessible annotation tools must be compatible with a wide range of imaging devices. We argue that many scientific workflows, particularly those involving microscopes or specimen curation, would benefit greatly from the ability to capture and annotate images in real time. For instance, navigating a microscope slide or processing large biological collections often involves domain experts who serve simultaneously as annotators. Separating the capture and annotation steps can interrupt this workflow and reduce efficiency. In these contexts, the ability to annotate during image acquisition improves both efficiency and accuracy.

An image annotation tool focused on usability should also prioritize simple graphical user interface (GUI) components with a shallow learning curve, minimizing the need for technical support or costly training for specialized personnel. Although existing tools like LabelMe,[19] VGG image annotator (VIA),[20] and COCO Annotator[21] offer pixel-level annotations for segmentation tasks, they lack live camera integration, a critical gap for workflows requiring immediate feedback or iterative labeling. These tools, though flexible in formatting (*e.g.*, COCO JSON, Pascal VOC), often require users to navigate feature-heavy

[a]*Department of Chemistry and Chemical Biology, McMaster University, Hamilton, ON, Canada. E-mail: garcilau@mcmaster.ca*

[b]*School of Computational Science and Engineering, McMaster University, Hamilton, ON, Canada. E-mail: vargashr@mcmaster.ca*

[c]*Brockhouse Institute for Materials Research, McMaster University, Hamilton, ON, Canada*

interfaces or offline workflows (*e.g.*, biologists, field technicians). Moreover, most annotation software supports pixel-level annotation only through polygons or rectangular boxes. While effective for rigid geometries, these primitives are poorly suited for organic or curved structures. Approximating a smooth contour with polygons requires a high number of vertices, introducing annotation inefficiency and potential geometric bias that can propagate into downstream vision models. By contrast, graphic design software routinely uses Bézier splines to capture curves with minimal control points and high precision. This approach offers a more natural representation for biological or irregular shapes, making splines a compelling alternative to conventional polygon-based labeling. Commercial platforms such as Labelbox and Supervisely emphasize collaboration features but omit both live annotation and spline support. RectLabel (macOS-only) supports Bézier curves but does not allow on-site input. Web-based tools like CVAT[22] and Label Studio[23] provide scalability yet remain restricted to pre-recorded media. Together, these limitations highlight the absence of an integrated solution that combines live annotation, spline-based precision, and lightweight usability.

We present LivePyxel, developed here, an open-source Python GUI that integrates on-site video device input with Bézier spline-based segmentation, enabling precise pixel-level annotation of curved structures. LivePyxel was initially developed for on-demand annotation of microscopy images, but it also supports any video device accessible *via* OpenCV and Python. LivePyxel combines a lightweight, accessible interface with flexible annotation tools, including Bézier splines, polygons, and threshold-based masks, making it suitable for segmentation workflows across diverse research domains. The software is freely available at https://github.com/UGarCil/LivePyxel and can be installed through PyPI, with installation details found in the latest version of the repository along with tutorials, examples, and additional code.

The paper is structured as follows: Section 2 describes the specification and deployment details of LivePyxel, developed in this work. Section 3 demonstrates its application through an image segmentation task, where annotated images are used to train a ViM[24] in two scenarios: (1) segmenting eight different microorganisms and (2) performing data engineering with binary masks. LivePyxel is designed to streamline data collection and labeling, accelerating the development of AI models in experimental workflows that require on-site data manipulation or large-scale batch processing.

2 LivePyxel

LivePyxel is composed of an ecosystem of Python scripts that work cohesively to provide users with a graphical interface. This architecture provides a tool with a higher level of abstraction compared to most data science workflows, typically executed *via* Jupyter Notebooks. The abstraction level is higher because the code that makes the software does not directly encode the annotation process, but rather builds interoperability across objects that make up the graphical user interface (GUI). This architecture was chosen because the domain of the task addressed by the program naturally belongs to a graphics editor, and because graphical interfaces make the tool available to a wider audience of users with very little experience in programming and code.

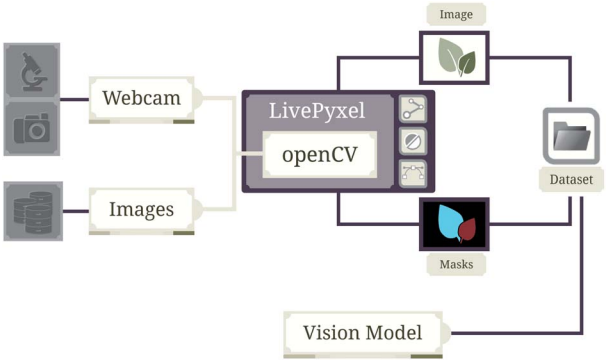

Fig. 1 Overview of the LivePyxel architecture, illustrating the integration of webcam and image inputs with OpenCV for on-site image annotation. The system supports input from live microscopy or camera feeds as well as pre-existing datasets, processes them through the LivePyxel interface, and outputs both annotated images and segmentation masks for storage or further analysis by a vision model.

The broad compatibility of LivePyxel with imaging devices is achieved through the use of the OpenCV[25] library, which enables on-site video input from virtually any camera and streams it directly to the annotation canvas. We chose OpenCV for its efficient handling of images as numerical matrices, leveraging a compiled back-end for performance while maintaining flexibility through seamless integration with NumPy[26] for fast manipulation in Python. The GUI itself is built using the Qt framework, which allows NumPy arrays to be rendered directly as images, resulting in a responsive and user-friendly interface. Fig. 1 is an overview of the LivePyxel architecture.

The LivePyxel GUI integrates several interactive components to streamline the annotation workflow. Mask display properties, such as opacity and binary threshold, can be adjusted using the sliders in the control section (Fig. 2A). Annotation categories are managed in the labels panel (Fig. 2B), where users can add, edit, or delete classes. The annotation panel (Fig. 2C) provides high-level controls for switching input sources, capturing frames, and toggling annotation mode. Drawing and editing actions are performed using the toolbar (Fig. 2D), which offers tools for creating, modifying, or erasing masks. The canvas (Fig. 2E) displays either live microscope or uploaded images, over which masks are layered non-destructively in real time. Navigation buttons (Fig. 2F) allow users to cycle through frames or dataset entries. Once annotations are complete, LivePyxel saves both the annotated images and corresponding masks into automatically generated subfolders within a user-specified directory. In addition to on-site webcam annotation, LivePyxel also supports traditional workflows by allowing users to upload pre-existing image datasets that follow the required folder structure.

2.1 Bézier splines for non-tradictional curvatures

A central feature of LivePyxel is its support for Bézier splines;[27] a mathematical representation of curves that originated in

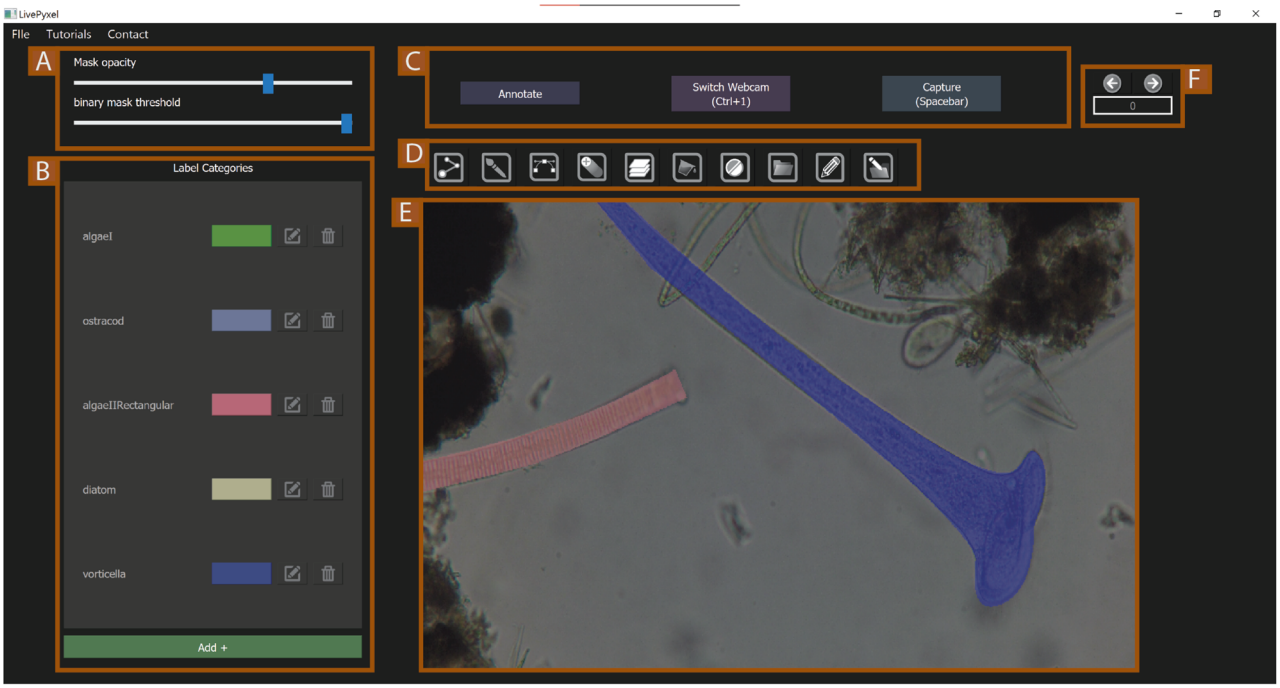

Fig. 2 Preview of the LivePyxel graphical user interface (GUI) with its main components labeled: (A) sliders for adjusting mask opacity and binary mask threshold, (B) labels panel for managing annotation categories, (C) annotation panel with main control buttons, (D) toolbar for selecting drawing and editing tools, (E) central canvas where objects are annotated on microscopy images in real time, and (F) navigation controls for browsing through image frames or dataset entries.

computer graphics and has become an industry standard for precision editing. Unlike traditional polygon tools, which approximate shapes using straight-line segments and are better suited for rigid structures like crystals, buildings, or mechanical parts, Bézier splines allow for smooth, continuous contours; see Fig. 3. This makes them ideal for segmenting organic, curved shapes typically encountered in biological datasets, such as cells, tissues, or protozoa.

Each Bézier unit is defined by three control points: two end points and a central handle that determines curvature; see Fig. 3B. By connecting multiple units, users can construct complex outlines with high fidelity to natural forms. This modular structure offers fine control over both curvature and sharp transitions. The advantage of splines for delimiting contours is illustrated by SplineDist,[28] which extends over the popular StarDist framework by modeling objects as planar parametric spline curves, allowing more flexible and smooth segmentation boundaries and solving issues with non-convex geometries. Moreover, the use of splines in the preparation of annotated data for segmentation tasks has been documented in clinical applications.[29,30] In biomedical imaging, the use of splines enables users to trace smooth cell membranes with highly organic contours and tightly coiled or angular biological features, something that would require excessive effort and precision with polygon tools. The ability to produce accurate, pixel-level masks with fewer interactions not only improves annotation speed but also reduces user fatigue and annotation bias. As a result, Bézier splines in LivePyxel improve both the efficiency and the quality of segmentation in tasks where precision is critical, such as microscopy, radiography, and digital morphology.

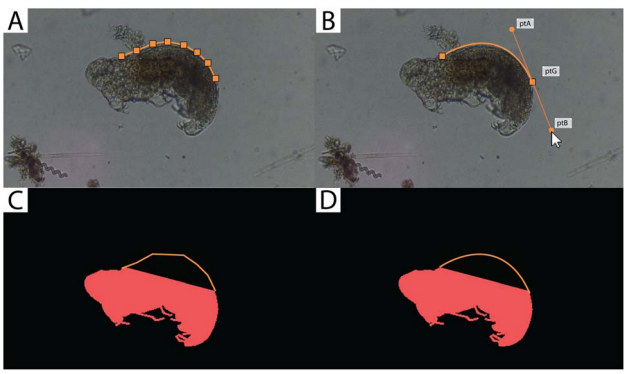

Fig. 3 Comparison between polygon (panels A and C) and Bézier-splines (panels B and D) in the delimitation of boundaries of a tardigrade's body. One Bézier unit is defined by three points. ptA and ptB constitute the start and end of the unit, whereas ptG serves as a point of gravity to pull the line away from the user's cursor, forcing it to become a curve.

2.2 Binary masks and multi-layered structure

Another important feature provided by LivePyxel is the generation and manipulation of binary masks, which enable semi-automated annotation in highly controlled imaging environments. LivePyxel generates binary masks by applying pixel-level operations using OpenCV throughout the frame, then assigning

a value of 0 or 1 to every pixel before creating an annotation mask. Using this technique, the user can isolate regions of high contrast against a uniform background, such as a white or black field. This proves to be especially useful in biomedical applications involving solid and organic objects,[31,32] and potentially significant in many other fields of biological science, where annotation targets are organic and contour-heavy. We showcase this feature in Section 3.2.

Binary masks significantly reduce annotation time by providing an initial segmentation that can be manually refined, in contrast to beginning an annotation from scratch. Particularly useful in situations where there are many structures to annotate in a controlled environment with a static background (see Section 3.2 targeting an example using snail shells), LivePyxel provides a way to critically boost the gathering of image/mask pair data. The generated dataset can further be augmented or 'engineered' to generate a much larger dataset (Fig. 7).

Finally, binary masks can also be used in combination with vector-based tools such as Bézier splines. Once a region is roughly defined using thresholding, the user can trace or refine its boundaries using spline functions to achieve pixel-level precision. This hybrid workflow enhances segmentation quality while minimizing manual effort, offering a powerful solution for datasets where both speed and anatomical accuracy are crucial.

LivePyxel employs a stack-based compositing system, in which each annotation exists on an independent layer logically stacked from bottom to top. This layered architecture facilitates non-destructive editing: individual segments or anatomical regions can be added, modified, or removed without affecting neighboring annotations; see Fig. 4. Each layer retains its own shape, color, and mask information, granting users fine-grained control over the annotation workflow. When the *Annotate* button is pressed, these layers are rasterized and composited in order, from bottom to top, into a single merged mask, encoding the labels in the form of RGB colors and overwriting every pixel with the top color (ignoring black). This final image can be saved or exported in formats compatible with most deep learning frameworks.

Importantly, this integrated pipeline also facilitates instance segmentation tasks, where models must not only classify each pixel but also associate it with a specific object instance. By storing each layer separately, LivePyxel can export instance-specific masks, with each layer representing a distinct object. This capability enables seamless integration with training workflows for models such as Mask R-CNN and SAM.

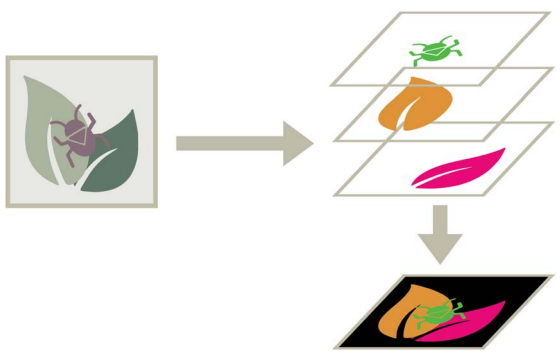

Fig. 4 Illustration of the multi-layer annotation process. Each annotation is stored as a separate layer within a stack, which is subsequently merged into a single raster image from bottom to top upon saving.

2.3 Webcam integration

LivePyxel, through the OpenCV library, integrates the webcam feed into the GUI. The program begins by searching for up to four camera devices connected to the computer, displaying the first device found. This multicamera feature is useful in situations where the user has multiple video-device stations connected to the application and may require flexibility in accessing different cameras in controlled environments. Once loaded, webcam images are rendered onto the canvas, allowing users to perform annotations either in on-site during live streaming or on captured frames. The user concludes an annotation by clicking on the Annotate button. The annotation will be stored in the folder specified by the user, in the form of an RGB image in the native webcam's resolution, and a matching mask with the pixel annotations encoded by the colors chosen by the user. LivePyxel is capable of reading most formats supported by OpenCV, including .jpg, .png, .tiff, and .bmp, and the annotated images are saved in .png format. The identity and color of each labeled class are stored in a .json file for its readability and its common use in a wide range of programming and app development applications.

2.4 Tablet integration

The use of drawing tablets in combination with graphics editors is a desirable practice as it provides notable advantages over the use of a computer mouse. Tablet integration improves annotations by reducing the time required to delimit the contour of objects without compromising precision. The libraries that integrate the core functionality of the program, OpenCV and pyQt, contain highly optimized compatibility with computer mouse-like devices, resulting in a smooth "plug-and-play" integration with LivePyxel. The "Free Hand" is a specific tool designed to improve the user experience with drawing tablets. It allows the user to automatically generate new vertices in a polygon by calculating the distance between the last vertex and the position of the tablet's stylus, adding a new vertex when the distance is greater than a reference threshold. This behavior allows the user to focus entirely on the hand movement, tracking the contour. This functionality was essential for creating the dataset used to train a vision model to segment eight different microorganisms in Section 3.1, using an Artist Display 15.6 Pro tablet.

3 Examples

Here, we present two distinct examples demonstrating the use of LivePyxel for labeling training data in pixel-level vision tasks. In Section 3.1, we showcase its application in a segmentation task involving eight different microorganisms, highlighting manual data annotation using Bézier splines and other built-in tools. In Section 3.2, we describe the development of a large

dataset through semi-automated labeling with binary masks, followed by data augmentation techniques to evaluate LivePyxel's scalability in dataset creation for training vision models. A video introduction to the use of binary masks for the semi-automation of dataset annotations is presented at: tutorial link. For both examples, the training pipelines and models were implemented using PyTorch.[33] Finally, in Section 3.3, we compared the labeling accuracy of LivePixel with four other annotation tools.

3.1 Water tank: image segmentation task

The presence of certain microorganisms in the environment often serves as an indirect indicator of ecosystem health.[34] These organisms, commonly referred to as indicator species (IS), are highly sensitive to chemical and physical changes in their environment, and their use has become widespread in environmental assessment studies.[35] Some datasets and data management digital infrastructures have been published in recent years,[36–38] highlighting the growing importance of automation in the identification and monitoring of these species communities. However, the ephemeral nature of these communities poses significant challenges in collecting and processing annotations for vision-model tasks at a rate comparable to the speed at which such communities change. To demonstrate the capabilities of LivePyxel, we curated a high-quality dataset capturing the biodiversity of water samples collected from an urban park in Toronto, Canada, in April 2025. We acquired 1250 image–mask pairs from a water tank, belonging to eight categories of microorganisms (see SI for more information).

We trained a U-Net model[24] for image segmentation across eight categories, initializing it with VGG-19 pretrained weights.[39] Each image in the dataset had an original resolution of 720 × 480 pixels. Due to the dataset's limited size, certain categories were underrepresented (see Fig. S2 in the SI). To mitigate this imbalance, we applied categorical weighting to the cross-entropy loss function and employed data augmentation techniques. Additional architectural and training details are provided in the SI. Training was carried out over 131 epochs using two NVIDIA V100 16 GB GPUs, with a total training time of approximately 8 hours. For evaluation, we reserved an independent test set of 200 images that were excluded from both training and validation.

Fig. 5 presents the final F1 scores of the trained U-Net model, which performed well in identifying the most abundant classes despite the limited dataset size. These scores reflect consistent accuracy for dominant taxa, but also highlight the challenges of detecting rare or visually ambiguous classes such as tardigrades, diatoms, and rotifers, an issue aligned with the class imbalance of the dataset, where these taxa collectively account for only about 10% of samples (see Fig. S2 in the SI). As shown in Fig. 6, the model also struggled to detect specimens that exhibit transparent bodies, highly variable morphologies, and poor representation in training data, such as *Vorticella* sp., resulting in incorrect contour recognition.

The environmental microorganisms dataset served as an example of the capacity of LivePyxel to rapidly produce quality

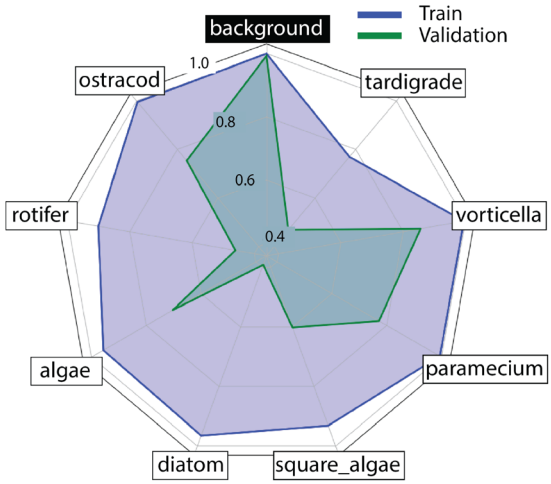

Fig. 5 The F1 scores achieved by a U-Net model, highlighting the performance across the eight different microorganisms and the background. The U-Net was initialized with the VGG-19 weights. For more details regarding the training, see the main text.

annotations of a rapidly changing community of species. The U-Net attained reliable F1 scores for the most abundant classes, but performance degraded for rare, transparent, or morphologically variable organisms (*e.g.*, *Vorticella* sp., diatoms, rotifers), reflecting the dataset's class imbalance. These outcomes underscore two complementary needs for ecological vision

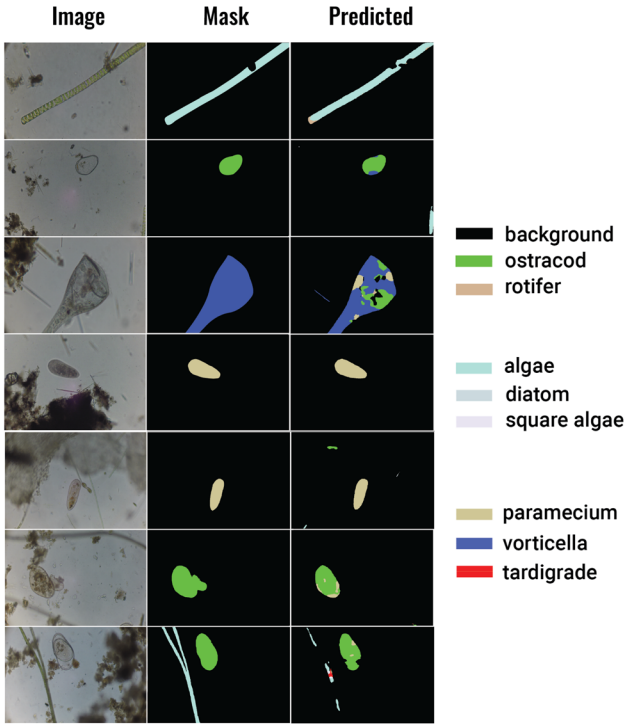

Fig. 6 Example predictions from the trained U-Net model on the water tank (EM) dataset. Each row shows the original microscopy image (left), the corresponding ground truth mask (middle), and the model's predicted segmentation (right). The legend on the right indicates the color coding for each class.

models deployed in dynamic microbial communities: (i) the importance of capturing morphological variability; and (ii) the use of training strategies that are robust to scarcity and ambiguous boundaries, including categorical weighting in the cross-entropy loss to account for rare classes, data augmentation and the potential in using data engineering techniques to boost the capacity of the network to recognize rare classes.

3.2 Snail shells: data engineering with binary masks

Creating training datasets for semantic segmentation is notoriously labor-intensive. Precise pixel-level annotations are required, and this process becomes especially difficult for images containing many small and round objects, where annotators must painstakingly outline each instance. For example, producing 1000 segmentation masks for a benchmark like MS COCO took about 22 (ref. 40) hours, highlighting the high cost of manual labeling. The performance of most vision models improves significantly with the size of the training dataset. This creates a strong incentive to explore automated and semi-automated annotation methods and advanced data augmentation to expand datasets without proportional human effort.[40]

In certain scenarios, a dataset can be constructed under highly controlled lighting and background conditions to semi-automate the annotation process. For example, using a uniform background (*e.g.*, a solid white or green backdrop) and exploiting simple threshold-based image operations to obtain binary masks for the foreground objects. In a controlled setup where the background is a known uniform color (such as white), images can be converted to grayscale (or other color space) and apply a threshold that classifies each pixel as background (0) or object (1) based on its intensity or hue.[41] This effectively binarizes the image, separating the objects from the backdrop. For example, researchers have captured objects on a plain white background under consistent lighting, which reduces the complexity of background removal and allows easy discrimination of the object by simple intensity thresholding when dealing with organic shapes.[42] In fields like medical imaging, placing surgical instruments in front of a green screen has been used to automatically extract tool masks – the monochromatic green background makes it easy to isolate the instrument by hue thresholding.[41] In such controlled conditions, the threshold can be tuned so that any pixel brighter (or darker) than a set value is labeled as background, while the rest are labeled as foreground (object). This semi-automated mask generation dramatically speeds up dataset creation, since the bulk of the annotation is handled by an algorithm (with minimal human correction for any errors).

We used LivePyxel to test its capabilities in the automation and generation of image/mask pairs in a highly controlled environment using an assortment of snail shells purchased at a dollar store in Toronto, Canada. The dataset is composed of 4 different classes, believed to correspond to different species of mollusks (see Fig. S3 in the SI).

The original training data set was collected by placing a large number of shells from the same category at the same time, using the binary mask tool to discriminate the background from the shells, and assigning a color to the final annotation (Fig. 7A). This resulted in 1400 individual annotations, each containing only a single class per image. To prepare these data, we used each binary mask to isolate the pixels of the object class within the original image. This yielded a trimmed version of the object with transparency, where all background pixels were removed and only the object pixels remained (Fig. 7B). These trimmed images were then stored along with their masks.

We implemented a randomized compositing strategy. For each of the 10 000 engineered samples, a base background image was selected, and the class folders were shuffled to introduce variation. A random number of classes were sampled, and for each selected class, a transparent image was randomly chosen and placed on the background (Fig. 7C). The same transformation, such as flipping or rotation, was applied to both the image cutout and its corresponding mask. This ensured that pixel-level alignment was preserved within an image/mask pair. Multiple instances from different classes were overlaid in succession, resulting in composite scenes that mimic realistic configurations with precise pixel-accurate masks for each object (Fig. 7D).

This pipeline produced a total of 10 000 synthetic image/mask pairs, significantly enriching the dataset and introducing diverse combinations of object instances, orientations, and overlaps. These engineered samples were subsequently used to train the same U-Net segmentation model used for Section 3.1, keeping the image size at 512. The model was trained for 24 hours on 4 NVIDIA A100 GPUs. In contrast with the water tank dataset (Section 3.1), the F1 scores for the snail shells dataset demonstrated consistently high performance across all classes, with minimal variation between training and validation sets and a very high F1 score across all categories (Fig. 8). The model also exhibited a rapid decline in the loss function, accompanied by a steep rise in F1 scores within the

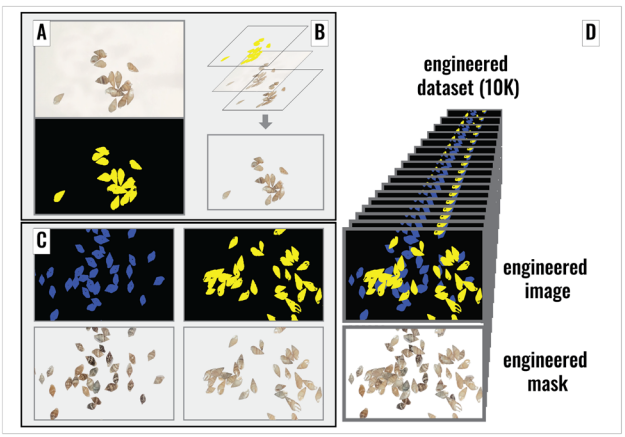

Fig. 7 Pipeline for engineered semantic segmentation dataset creation. (Panel A) The binary mask tool from LivePyxel captures an image/mask annotation of shells belonging to the same class. (Panel B) Masks are overlapped on top of images to retrieve transparencies. (Panel C) Transparencies are randomly stacked together, forming a new composite image on top of a background. (Panel D) We quickly engineered a dataset consisting of 10 000 image/mask pairs.

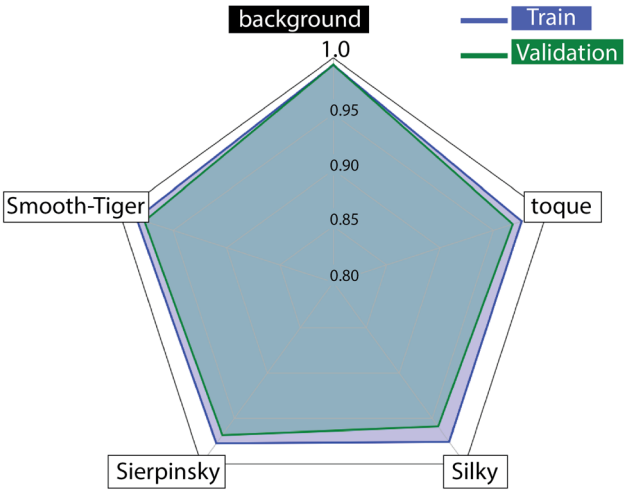

Fig. 8 F1 scores achieved by a U-Net model with VGG-19 backbone during training and validation. The plot highlights performance across 5 different classes.

first three epochs (Fig. S5 and S6 in SI). Fig. 9 showcases the predicted masks by the trained U-Net for some of the images in the dataset.

The snail-shells study demonstrates LivePyxel's capacity to automate the creation of masks under highly controlled setups, yielding fast, stable optimization and uniformly high F1 across classes. Compared to the environmental microscopy setting, the engineered dataset reduces annotation costs in time and user effort. The U-Net architecture, initialized with VGG-19 and trained on 10 000 composite image/mask pairs, converged within a few epochs and exhibited a minimal train–validation gap. These results underscore that with accurate binary masks and controlled backgrounds, segmenting small, round objects becomes straightforward under laboratory conditions and with the use LivePyxel.

3.3 Comparison between polygon and Bézier-splines

We evaluated the performance of LivePyxel against several widely used annotation tools. To ensure a fair comparison,

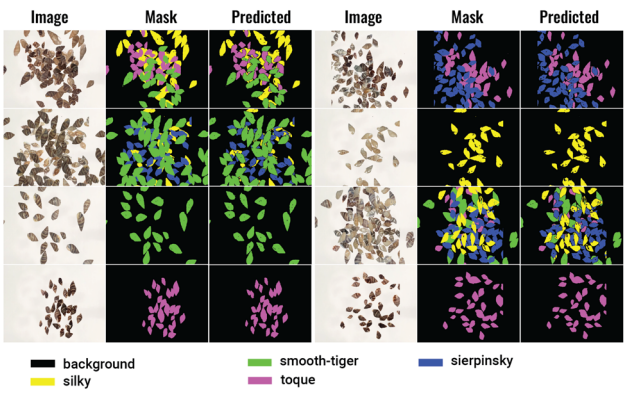

Fig. 9 Example predictions from the trained U-Net model on the snail shells dataset. Each set of three columns shows the original image (left), the corresponding ground truth mask (middle), and the model's predicted segmentation (right). The legend below indicates the color coding for each class.

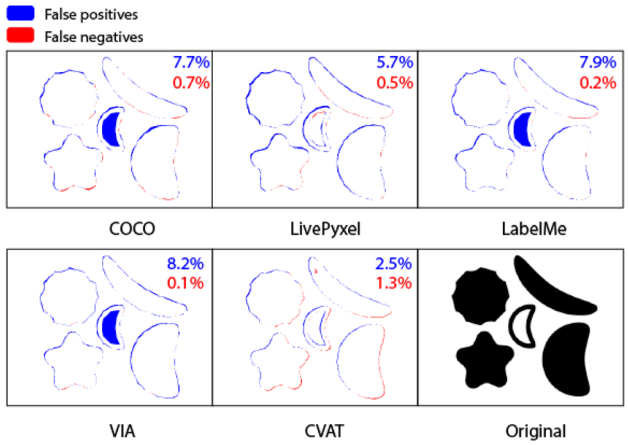

Fig. 10 Comparison of annotation accuracy across different annotation tools using a reference image (Original) with various levels of curvature complexity. Blue regions represent false positives (areas incorrectly included), while red regions denote false negatives (areas missed). Percentages in each panel indicate the total proportion of false positives (blue) and false negatives (red) for each tool relative to the total number of pixels in the image.

a reference image composed of well-defined polygons with varying degrees of curvature complexity was generated (panel labeled "Original" in Fig. 10).

Our results show that the accuracy achieved with LivePyxel is comparable to, or slightly higher than, other annotation software packages, while maintaining similar annotation times; see Section 5 in the SM. As highlighted in Section 3.2, a key advantage of LivePyxel is its capability to perform Boolean operations directly within the mask. This feature, uncommon among existing tools, enables rapid and flexible labeling of complex regions, including those with internal holes like the central object in Fig. 10.

In terms of performance, LivePyxel exhibited a balanced trade-off between false positives (5.7%) and false negatives (0.5%), comparable to other tools (Fig. 10). CVAT achieved the lowest overall error (2.5% false positives, 1.3% false negatives) through AI-assisted segmentation, but at the cost of a more complex setup, internet dependency, and non-local data handling. VIA offered the simplest installation (a standalone HTML file), whereas LabelMe required manual dependency management on some systems. COCO Annotator had the most challenging setup, involving Docker and SQL-based database configuration. Among these, only LivePyxel integrates Bézier-spline support, providing smoother boundary representation than polygon-based tools such as VIA, LabelMe, and COCO Annotator.

4 Summary

We present LivePyxel, an open-source pixel-level annotation library designed for both on-site image labeling and pre-saved image datasets. Built on OpenCV, LivePyxel efficiently handles image processing tasks and introduces Bézier splines as a key feature, enabling smoother and more precise mask boundaries.

To demonstrate LivePyxel's utility, we provide examples where the labeled images, generated with LivePyxel, are used for training a vision model, completing the annotation-to-inference loop. The datasets for both presented examples were captured using a GoPro camera mounted on a microscope, highlighting LivePyxel's compatibility with diverse imaging setups. Finally, LivePyxel supports the use of tablet-type devices, making the annotation process more ergonomic. In the near future, we aim to implement two additional features, (i) zoom integration, allowing users to work on finer details within LivePyxel's central canvas (Fig. 2E), and (ii) inspired by CVAT's AI integration tools, adding a tool capable of loading AI vision models like SAM to help accelerate the annotation process. Even in its present form, LivePyxel represents a significant innovation for labeling images, addressing an unmet need for smoother editing tools that could accelerate data generation.

Author contributions

U. G.-C. wrote the code for LivePyxel, annotated the datasets, and performed all experiments. J. O. O. and R. A. V.-H. guided the project. All authors wrote and approved the final version of the manuscript.

Conflicts of interest

The authors have no conflicts to disclose.

Data availability

LivePyxel and all trained vision models and data presented in this paper are freely available at https://github.com/UGarCil/LivePyxel and can be installed through PyPI. All datasets used in this study are available at https://doi.org/10.5281/zenodo.17858610.

Supplementary information (SI) is available. See DOI: https://doi.org/10.1039/d5dd00421g.

Acknowledgements

J. O. O. acknowledges the support from NSERC Discovery grant no. RGPIN-2024-06670. R. A. V.-H. acknowledges the support from the Digital Research Alliance of Canada and NSERC Discovery grant no. RGPIN-2024-06594.

References

1 M. Bilal, R. Podishetti, L. Koval, M. A. Gaafar, D. Grossmann and M. Bregulla, *Sensors*, 2024, **24**, 4777.
2 W. Chi, L. Ma, J. Wu, M. Chen, W. Lu and X. Gu, *Phys. Med. Biol.*, 2020, **65**, 235001.
3 D. Karimi, H. Dou, S. K. Warfield and A. Gholipour, *Med. Image Anal.*, 2020, **65**, 101759.
4 P. Singh, R. Chukkapalli, S. Chaudhari, *et al.*, *Sci. Rep.*, 2024, **14**, 10820.
5 A. Yu, Y. Quan, R. Yu, W. Guo, X. Wang, D. Hong, H. Zhang, J. Chen, Q. Hu and P. He, *Remote Sens.*, 2023, **15**, 4987.
6 V. Taran, Y. Gordienko, A. Rokovyi, O. Alienin and S. Stirenko, *Advances in Computer Science for Engineering and Education II*, 2020, pp. 183–193.
7 A. Đuraš, B. J. Wolf, A. Ilioudi, I. Palunko and B. De Schutter, *Sci. Data*, 2024, **11**, 921.
8 G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken and C. I. Sánchez, *Med. Image Anal.*, 2017, **42**, 60–88.
9 N. Tajbakhsh, L. Jeyaseelan, Q. Li, J. N. Chiang, Z. Wu and X. Ding, *Med. Image Anal.*, 2020, **63**, 101693.
10 E. Moen, D. Bannon, T. Kudo, W. Graf, M. Covert and D. Van Valen, *Nat. Methods*, 2019, **16**, 1233–1246.
11 Q. Yuan, Y. Shi and M. Li, *Remote Sens.*, 2024, **16**, 2910.
12 J. G. A. Barbedo, *Agronomy*, 2025, **15**, 1157.
13 J. Y. Luo, J.-O. Irisson, B. Graham, C. Guigand, A. Sarafraz, C. Mader and R. K. Cowen, *Limnol Oceanogr. Methods*, 2018, **16**, 814–827.
14 J. Sauder, V. Domazetoski, G. Banc-Prandi, G. Perna, A. Meibom and D. Tuia, *arXiv*, preprint, arXiv:2503.20000, 2025, DOI: 10.48550/arXiv.2503.20000.
15 S. Wang, C. Li, R. Wang, Z. Liu, M. Wang, H. Tan, Y. Wu, X. Liu, H. Sun, R. Yang, X. Liu, J. Chen, H. Zhou, I. B. Ayed and H. Zheng, *Nat. Commun.*, 2021, **12**, 5915.
16 A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár and R. Girshick, *arXiv*, preprint, arXiv:2304.02643, 2023, 4015–4026, DOI: 10.48550/arXiv.2304.02643.
17 C. F. Nodine, H. L. Kundel, C. Mello-Thoms, S. P. Weinstein, S. G. Orel, D. C. Sullivan and E. F. Conant, *Acad. Radiol.*, 1999, **6**, 575–585.
18 A. Hosny, C. Parmar, J. Quackenbush, L. H. Schwartz and H. J. Aerts, *Nat. Rev. Cancer*, 2018, **18**, 500–510.
19 B. C. Russell, A. Torralba, K. P. Murphy and W. T. Freeman, *Int. J. Comput. Vis.*, 2008, **77**, 157–173.
20 A. Dutta and A. Zisserman, *Proceedings of the 27th ACM International Conference on Multimedia*, New York, NY, USA, 2019.
21 J. Brooks, *COCO Annotator*, https://github.com/jsbroks/coco-annotator/, 2019.
22 C. Corporation, *Computer Vision Annotation Tool (CVAT)*, 2023, DOI: 10.5281/zenodo.8070041.
23 M. Tkachenko, M. Malyuk, A. Holmanyuk and N. Liubimov, *Label Studio: Data Labeling Software*, 2020, https://github.com/HumanSignal/label-studio, Open source software, maintained by HumanSignal. Accessed: 2025-11-06.
24 O. Ronneberger, P. Fischer and T. Brox, *Medical Image Computing And Computer-Assisted Intervention–Miccai 2015: 18th International Conference, Munich, Germany, October 5–9, 2015, Proceedings, Part III 18*, 2015, pp. 234–241.
25 G. Bradski, *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 2000, vol. **25**, pp. , pp. 120–123.
26 C. R. Harris, K. J. Millman, S. J. Van Der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, et al., *Nature*, 2020, **585**, 357–362.

27 D. F. Rogers, *An Introduction to NURBS: with Historical Perspective*, Elsevier, 2000.
28 S. Mandal and V. Uhlmann, *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, 2021, pp. 1082–1086.
29 M. Huellebrand, M. Ivantsits, L. Tautz, S. Kelle and A. Hennemuth, *Front. Cardiovasc. Med.*, 2022, **9**, 829512.
30 A. Materka and J. Jurek, *Sensors*, 2024, **24**, 846.
31 H. Chen, Y. Deng, B. Li, Z. Li, H. Chen, B. Jing and C. Li, *Life*, 2023, **13**, 743.
32 A. Gálvez, A. Iglesias, I. Fister, I. Fister, C. Otero and J. A. Díaz, *J. Comput. Sci.*, 2021, **56**, 101481.
33 A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai and S. Chintala, *Adv. Neural Inf. Process. Syst.*, 2019, **32**, 8024–8035.
34 J. Cairns, P. V. McCormick and B. Niederlehner, *Hydrobiologia*, 1993, **263**, 1–44.
35 A. A. Siddig, A. M. Ellison, A. Ochs, C. Villar-Leeman and M. K. Lau, *Ecol. Indic.*, 2016, **60**, 223–230.
36 Z. Li, C. Li, Y. Yao, J. Zhang, M. M. Rahaman, H. Xu, F. Kulwa, B. Lu, X. Zhu and T. Jiang, *PLoS One*, 2021, **16**, e0250631.
37 Q. Li, X. Sun, J. Dong, S. Song, T. Zhang, D. Liu, H. Zhang and S. Han, *ICES J. Mar. Sci.*, 2020, **77**, 1427–1439.
38 I. G. Goldberg, C. Allan, J.-M. Burel, D. Creager, A. Falconi, H. Hochheiser, J. Johnston, J. Mellen, P. K. Sorger and J. R. Swedlow, *Genome Biol.*, 2005, **6**, 1–13.
39 K. Simonyan and A. Zisserman, *arXiv*, preprint, arXiv:1409.1556, 2014, DOI: 10.48550/arXiv.1409.1556.
40 G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. D. Cubuk, Q. V. Le and B. Zoph, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2918–2928.
41 L. C. Garcia-Peraza-Herrera, L. Fidon, C. D'Ettorre, D. Stoyanov, T. Vercauteren and S. Ourselin, *IEEE Trans. Med. Imag.*, 2021, **40**, 1450–1460.
42 S. Mohana and C. Prabhakar, *Int. J. Image Graph. Signal Process.*, 2015, **7**, 11.