Semi-Supervised Semantic Annotator (S3A): Toward Efficient Semantic Labeling

Nathan Jessurun[‡*], Daniel E. Capecci[‡], Olivia P. Dizon-Paradis[‡], Damon L. Woodard[‡], Navid Asadizanjani[‡]

Abstract—Most semantic image annotation platforms suffer severe bottlenecks when handling large images, complex regions of interest, or numerous distinct foreground regions in a single image. We have developed the Semi-Supervised Semantic Annotator (S3A) to address each of these issues and facilitate rapid collection of ground truth pixel-level labeled data. Such a feat is accomplished through a robust and easy-to-extend integration of arbitrary python image processing functions into the semantic labeling process. Importantly, the framework devised for this application allows easy visualization and machine learning prediction of arbitrary formats and amounts of per-component metadata. To our knowledge, the ease and flexibility offered are unique to S3A among all open-source alternatives.

Index Terms—Semantic annotation, Image labeling, Semi-supervised, Region of interest

Introduction

Labeled image data is essential for training, tuning, and evaluating the performance of many machine learning applications. Such labels are typically defined with simple polygons, ellipses, and bounding boxes (i.e., "this rectangle contains a cat"). However, this approach can misrepresent more complex shapes with holes or multiple regions as shown later in Figure 9. When high accuracy is required, labels must be specified at or close to the pixel-level - a process known as semantic labeling or semantic segmentation. A detailed description of this process is given in [CZF+18]. Examples can readily be found in several popular datasets such as COCO, depicted in Figure 1.

Semantic segmentation is important in numerous domains including printed circuit board assembly (PCBA) inspection (discussed later in the case study) [PJTA20], [AML+19], quality control during manufacturing [FRLL18], [AVK+01], [AAV+02], manuscript restoration / digitization [GNP+04], [KBO16], [JB92], [TFJ89], [FNK92], and effective patient diagnosis [SKM+10], [RLO+17], [YPH+06], [IGSM14]. In all these cases, imprecise annotations severely limit the development of automated solutions and can decrease the accuracy of standard trained segmentation models.

Quality semantic segmentation is difficult due to a reliance on large, high-quality datasets, which are often created by manually labeling each image. Manual annotation is error-prone, costly,

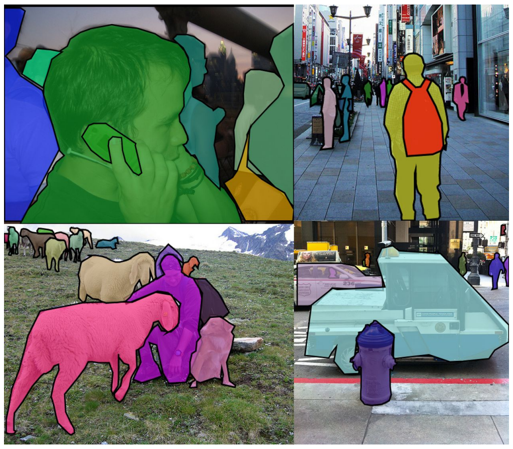


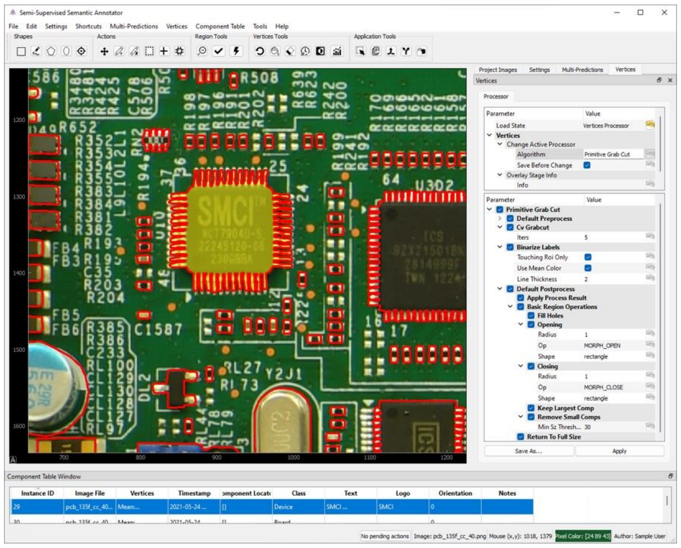
Fig. 1. Common use cases for semantic segmentation involve relatively few foreground objects, low-resolution data, and limited complexity per object. Images retrieved from https://cocodataset.org/#explore.

and greatly hinders scalability. As such, several tools have been proposed to alleviate the burden of collecting these ground-truth labels [itL18]. Unfortunately, existing tools are heavily biased toward lower-resolution images with few regions of interest (ROI), similar to Figure 1. While this may not be an issue for some datasets, such assumptions are *crippling* for high-fidelity images with hundreds of annotated ROIs [LSA+10], [WYZZ09].

With improving hardware capabilities and increasing need for high-resolution ground truth segmentation, there are a continually growing number of applications that *require* high-resolution imaging with the previously described characteristics [MKS18], [DS20]. In these cases, the existing annotation tooling greatly impacts productivity due to the previously referenced assumptions and lack of support [Spa20].

In response to these bottlenecks, *we present the Semi-Supervised Semantic Annotation (S3A) annotation and prototyping platform -- an application which eases the process of pixel-level labeling in large, complex scenes.*[1] Its graphical user interface is shown in Figure 2. The software includes live app-level property customization, real-time algorithm modification and feedback, region prediction assistance, constrained component table editing based on allowed data types, various data export formats, and a highly adaptable set of plugin interfaces for domain-specific extensions to S3A. Beyond software improvements, these features play significant roles in bridging the gap between human annotation efforts and scalable, automated segmentation methods [BWS+10].

* *Corresponding author: njessurun@ufl.edu*
‡ *University of Florida*


Semi-Supervised Semantic Annotator
File Edit Settings Shortcuts Multi-Predictions Vertices Component Table Tools Help
Shapes
Actions
Region Tools
Vertices Tools
Application Tools
Project Images Settings Multi-Predictions Vertices
Save As...
Apply

Vertices

Processor

Parameter	Value
Load State	Vertices Processor
Vertices	
Change Active Processor	
Algorithm	Primitive Grab Cut
Save Before Change	✓
Overlay Stage Info	
Info	

Parameter	Value
✓ Primitive Grab Cut	
✓ Default Preprocess	
✓ Cv Grabcut	
Iters	5
✓ Binarize Labels	
Touching Roi Only	✓
Use Mean Color	✓
Line Thickness	2
✓ Default Postprocess	
✓ Apply Process Result	
✓ Basic Region Operations	
✓ Fill Holes	
✓ Opening	
Radius	1
Op	MORPH_OPEN
Shape	rectangle
✓ Closing	
Radius	1
Op	MORPH_CLOSE
Shape	rectangle
✓ Keep Largest Comp	
✓ Remove Small Comps	
Min Sz Thresh...	30
✓ Return To Full Size	

Component Table Window

Instance ID	Image File	Vertices	Timestamp	omponent Locati	Class	Text	Logo	Orientation	Notes
29	pcb_135f_cc_40...	Mean:...	2021-05-24 ...	[]	Device	SMCI ...	SMCI	0	
30	pcb_135f_cc_40	Mean:	2021-05-24	[]	Board			0	

No pending actions Image: pcb_135f_cc_40.png Mouse (x,y): 1018, 1379 Pixel Color: [24 89 43] Author: Sample User

Fig. 2. S3A's interface. The main view consists of an image to annotate, a component table of prior annotations, and a toolbar which changes functionality depending on context.

Application Overview

Design decisions throughout S3A's architecture have been driven by the following objectives:

- Metadata should have significance rather than be treated as an afterthought,
- High-resolution images should have minimal impact on the annotation workflow,
- ROI density and complexity should not limit annotation workflow, and
- Prototyping should not be hindered by application complexity.

These motives were selected upon noticing the general lack of solutions for related problems in previous literature and tooling. Moreover, applications that *do* address multiple aspects of complex region annotation often require an enterprise service and cannot be accessed under open-source policies.

While the first three points are highlighted in the case study, the subsections below outline pieces of S3A's architecture that prove useful for iterative algorithm prototyping and dataset generation as depicted in Figure 3. Note that beyond the facets illustrated here, S3A possesses multiple additional characteristics as outlined in its documentation (https://gitlab.com/s3a/s3a/-/wikis/docs/User' s-Guide).

Processing Framework

At the root of S3A's functionality and configurability lies its adaptive processing framework. Functions exposed within S3A are thinly wrapped using a `Process` structure responsible for parsing signature information to provide documentation, parameter information, and more to the UI. Hence, all graphical depictions are abstracted beyond the concern of the user while remaining trivial

[1]. A preliminary version was introduced in an earlier publication [JPRA20], but significant changes to the framework and tool capabilities have been employed since then.

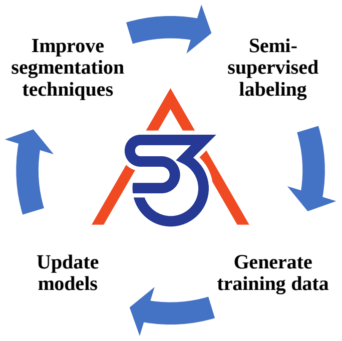


Fig. 3. S3A's can iteratively annotate, evaluate, and update its internals in real-time.

to specify (but can be modified or customized if desired). As a result, incorporating additional/customized application functionality can require as little as one line of code. Processes interface with PyQtGraph parameters to gain access to data-customized widget types and more (https://github.com/pyqtgraph/pyqtgraph).

These processes can also be arbitrarily nested and chained, which is critical for developing hierarchical image processing models, an example of which is shown in Figure 4. This framework is used for all image and region processing within S3A. Note that for image processes, each portion of the hierarchy yields intermediate outputs to determine which stage of the process flow is responsible for various changes. This, in turn, reduces the effort required to determine which parameters must be adjusted to achieve optimal performance.

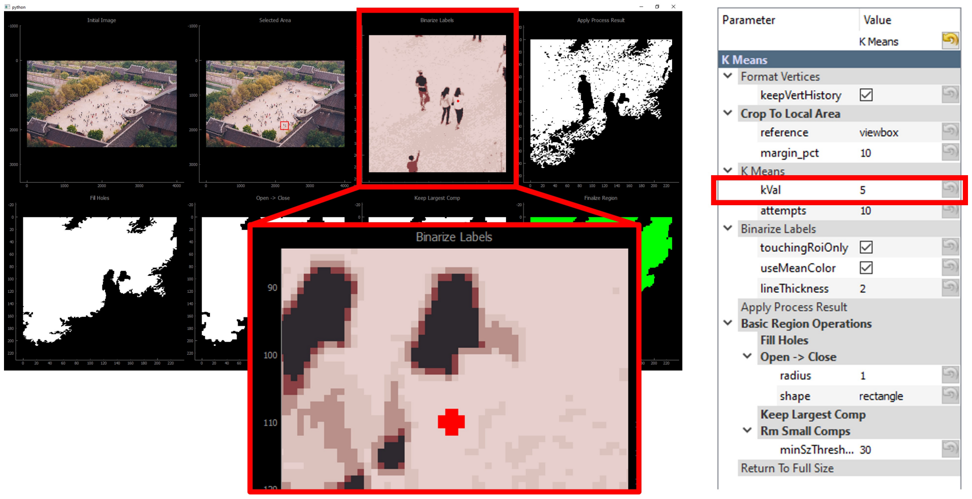
Plugins for User Extensions

The previous section briefly described how custom user functions are easily wrapped within a process, exposing its parameters within S3A in a GUI format. A rich plugin interface is built on top of this capability in which custom functions, table field predictors, default action hooks, and more can be directly integrated into S3A. In all cases, only a few lines of code are required to achieve most integrations between user code and plugin interface specifications. The core plugin infrastructure consists of a function/property registration mechanism and an interaction window that shows them in the UI. As such, arbitrary user functions can be "registered" in one line of code to a plugin, where it will be effectively exposed to the user within S3A. A trivial example is depicted in Figure 5, but more complex behavior such as OCR integration is possible with similar ease (see this snippet for an implementation leveraging `easyocr`).

Plugin features are heavily oriented toward easing the process of automation both for general annotation needs and niche datasets. In either case, incorporating existing library functions is converted into a trivial task directly resulting in lower annotation time and higher labeling accuracy.

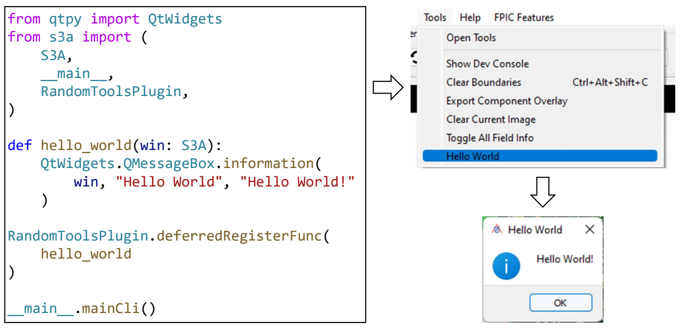
Adaptable I/O

An extendable I/O framework allows annotations to be used in a myriad of ways. Out-of-the-box, S3A easily supports instance-level segmentation outputs, facilitating deep learning model training. As an example, Figure 6 illustrates how each instance in the image becomes its own pair of image and mask data. When several instances overlap, each is uniquely distinguishable depending on the characteristic of their label field. Particularly helpful for



Parameter	Value
	K Means
K Means	
Format Vertices	
keepVertHistory	☑
Crop To Local Area	
reference	viewbox
margin_pct	10
K Means	
kVal	5
attempts	10
Binarize Labels	
touchingRoiOnly	☑
useMeanColor	☑
lineThickness	2
Apply Process Result	
Basic Region Operations	
Fill Holes	
Open -> Close	
radius	1
shape	rectangle
Keep Largest Comp	
Rm Small Comps	
minSzThresh...	30
Return To Full Size	

Fig. 4. Outputs of each processing stage can be quickly viewed in context after an iteration of annotating. Upon inspecting the results, it is clear the failure point is a low k value during K-means clustering and segmentation. The woman's shirt is not sufficiently distinguishable from the background palette to denote a separate entity. The red dot is an indicator of where the operator clicked during annotation.



```python
from qtpy import QtWidgets
from s3a import (
    S3A,
    __main__,
    RandomToolsPlugin,
)

def hello_world(win: S3A):
    QtWidgets.QMessageBox.information(
        win, "Hello World", "Hello World!"
    )

RandomToolsPlugin.deferredRegisterFunc(
    hello_world
)

__main__.mainCli()
```

Fig. 5. Simple standalone functions can be easily exposed to the user through the random tools plugin. Note that if tunable parameters were included in the function signature, pressing "Open Tools" (the top menu option) allows them to be altered.

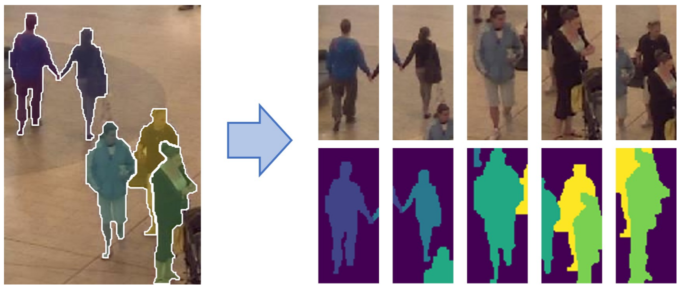


Fig. 6. Multiple export formats exist, among which is a utility that crops components out of the image, optionally padding with scene pixels and resizing to ensure all shapes are equal. Each sub-image and mask is saved accordingly, which is useful for training on multiple forms of machine learning models.

models with fixed input sizes, these exports can optionally be forced to have a uniform shape (e.g., 512x512 pixels) while maintaining their aspect ratio. This is accomplished by incorporating additional scene pixels around each object until the appropriate size is obtained. Models trained on these exports can be directly plugged back into S3A's processing framework, allowing them to generate new annotations or refine preliminary user efforts. The described I/O framework is also heavily modularized such that custom dataset specifications can easily be incorporated. In this manner, future versions of S3A will facilitate interoperability with popular formats such as COCO and Pascal VOC [LMB+14], [EGW+10].

Deep, Portable Customizability

Beyond the features previously outlined, S3A provides numerous avenues to configure shortcuts, color schemes, and algorithm workflows. Several examples of each can be seen in the user guide. Most customizable components prototyped within S3A can also be easily ported to external workflows after development. Hierarchical processes have states saved in YAML files describing all parameters, which can be reloaded to create user profiles. Alternatively, these same files can describe ideal parameter combinations for functions outside S3A in the event they are utilized in a different framework.

Case Study

Both the inspiration and developing efforts for S3A were initially driven by optical printed circuit board (PCB) assurance needs. In this domain, high-resolution images can contain thousands of complex objects in a scene, as seen in Figure 7. Moreover, numerous components are not representable by cardinal shapes such as rectangles, circles, etc. Hence, high-count polygonal regions dominated a significant portion of the annotated regions. The computational overhead from displaying large images and substantial numbers of complex regions either crashed most annotation platforms or prevented real-time interaction. In response, S3A was designed to fill the gap in open-source annotation platforms that addressed each issue while requiring minimal setup and allowing easy prototyping of arbitrary image processing tasks. The subsections below describe how the S3A labeling platform was utilized to collect a large database of PCB annotations along with their associated metadata[2].

Large Images with Many Annotations

In optical PCB assurance, one method of identifying component defects is to localize and characterize all objects in the image. Each

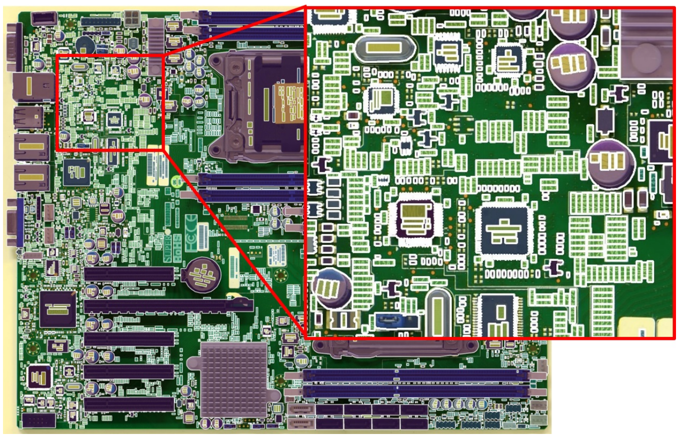


Fig. 7. Example PCB segmentation. In contrast to typical semgentation tasks, the scene contains over 4,000 objects with numerous complex shapes.

component can then be cross-referenced against genuine properties such as length/width, associated text, allowed orientations, etc. However, PCB surfaces can contain hundreds to thousands of components at several magnitudes of size, necessitating high-resolution images for in-line scanning. To handle this problem more generally, S3A separates the editing and viewing experiences. In other words, annotation time is orders of magnitude faster since only edits in one region at a time and on a small subset of the full image are considered during assisted segmentation. All other annotations are read-only until selected for alteration. For instance, Figure 8 depicts user inputs on a small ROI out of a much larger image. The resulting component shape is proposed within seconds and can either be accepted or modified further by the user. While PCB annotations initially inspired this approach, it is worth noting that the architectural approach applies to arbitrary domains of image segmentation.

Another key performance improvement comes from resizing the processed region to a user-defined maximum size. For instance, if an ROI is specified across a large portion of the image but the maximum processing size is 500x500 pixels, the processed area will be downsampled to a maximum dimension length of 500 before intensive algorithms are run. The final output will be upsampled back to the initial region size. In this manner, optionally sacrificing a small amount of output accuracy can drastically accelerate runtime performance for larger annotated objects.

Complex Vertices/Semantic Segmentation

Multiple types of PCB components possess complex shapes which might contain holes or noncontiguous regions. Hence, it is beneficial for software like S3A to represent these features inherently with a `ComplexXYVertices` object: that is, a collection of polygons which either describe foreground regions or holes. This is enabled by thinly wrapping `opencv`'s contour and hierarchy logic. Example components difficult to accomodate with single-polygon annotation formats are illustrated in Figure 9.

At the same time, S3A also supports high-count polygons with no performance losses. Since region edits are performed by image processing algorithms, there is no need for each vertex to be manually placed or altered by human input. Thus, such non-interactive shapes can simply be rendered as a filled path without a large number of event listeners present. This is the

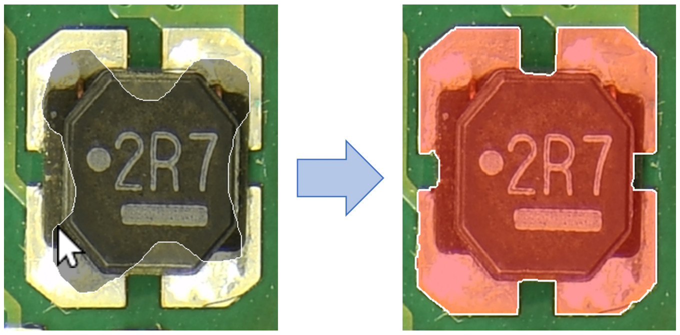


Fig. 8. Regardless of total image size and number of annotations, Python processing is be limited to the ROI or viewbox size for just the selected object based on user preferences. The depiction shows Grab Cut operating on a user-defined initial region within a much larger (8000x6000) image. The resulting region was available in 1.94 seconds on low-grade hardware.

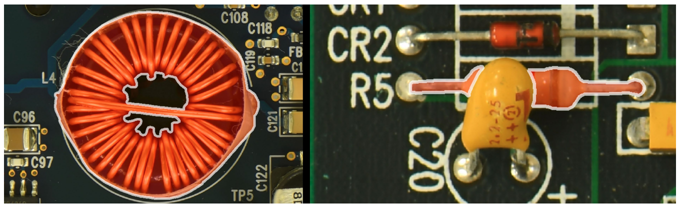


Fig. 9. Annotated objects in S3A can incorporate both holes and distinct regions through a multi-polygon container. Holes are represented as polygons drawn on top of existing foreground, and can be arbitrarily nested (i.e. island foreground is also possible).

key performance improvement when thousands of regions (each with thousands of points) are in the same field of view. When low polygon counts *are* required, S3A also supports RDP polygon simplification down to a user-specified epsilon parameter [Ram].

Complex Metadata

Most annotation software support robust implementation of image region, class, and various text tags ("metadata"). However, this paradigm makes collecting type-checked or input-sanitized metadata more difficult. This includes label categories such as object rotation, multiclass specifications, dropdown selections, and more. In contrast, S3A treats each metadata field the same way as object vertices, where they can be algorithm-assisted, directly input by the user, or part of a machine learning prediction framework. Note that simple properties such as text strings or numbers can be directly input in the table cells with minimal need for annotation assistance[3]. In conrast, custom fields can provide plugin specifications which allow more advanced user interaction. Finally, auto-populated fields like annotation timestamp or author can easily be constructed by providing a factory function instead of default value in the parameter specification.

This capability is particularly relevant in the field of optical PCB assurance. White markings on the PCB surface, known as silkscreen, indicate important aspects of nearby components. Thus, understanding the silkscreen's orientation, alphanumeric characters, associated component, logos present, and more provide several methods by which to characterize / identify features of their respective devices. Both default and customized input validators were applied to each field using parameter specifications, custom plugins, or simple factories as described above. A summary of the metadata collected for one component is shown in Figure 10.

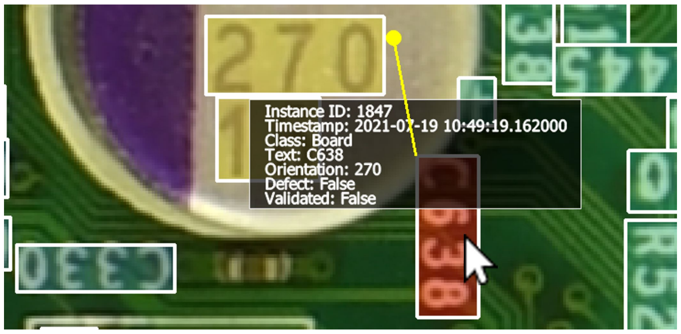


Fig. 10. Metadata can be collected, validated, and customized with ease. A mix of default properties (strings, numbers, booleans), factories (timestamp, author), and custom plugins (yellow circle representing associated device) are present.

Conclusion and Future Work

The Semi-Supervised Semantic Annotator (S3A) is proposed to address the difficult task of pixel-level annotations of image data. For high-resolution images with numerous complex regions of interest, existing labeling software faces performance bottlenecks attempting to extract ground-truth information. Moreover, there is a lack of capabilities to convert such a labeling workflow into an automated procedure with feedback at every step. Each of these challenges is overcome by various features within S3A specifically designed for such tasks. As a result, S3A provides not only tremendous time savings during ground truth annotation, but also allows an annotation pipeline to be directly converted into a prediction scheme. Furthermore, the rapid feedback accessible at every stage of annotation expedites prototyping of novel solutions to imaging domains in which few examples of prior work exist. Nonetheless, multiple avenues exist for improving S3A's capabilities in each of these areas. Several prominent future goals are highlighted in the following sections.

Dynamic Algorithm Builder

Presently, processing workflows can be specified in a sequential YAML file which describes each algorithm and their respective parameters. However, this is not easy to adapt within S3A, especially by inexperienced annotators. Future iterations of S3A will incoroprate graphical flowcharts which make this process drastically more intuitive and provide faster feedback. Frameworks like Orange [DCE+] perform this task well, and S3A would strongly benefit from adding the relevant capabilities.

Image Navigation Assistance

Several aspects of image navigation can be incorporated to simplify the handling of large images. For instance, a "minimap" tool would allow users to maintain a global image perspective while making local edits. Furthermore, this sense of scale aids intuition of how many regions of similar component density, color, etc. exist within the entire image.

Second, multiple strategies for annotating large images leverage a windowing approach, where they will divide the total image into several smaller pieces in a gridlike fashion. While this has its disadvantages, it is fast, easy to automate, and produces reasonable results depending on the initial image complexity [VGSG+19]. Hence, these methods would be significantly easier to incorporate into S3A if a generalized windowing framework was incorporated which allows users to specify all necessary parameters such as window overlap, size, sampling frequency, etc. A preliminary version of this is implemented for categorical-based model prediction, but a more robust feature set for interactive segmentation is strongly preferable.

Aggregation of Human Annotation Habits

Several times, it has been noted that manual segmentation of image data is not a feasible or scalable approach for remotely large datasets. However, there are multiple cases in which human intuition can greatly outperform even complex neural networks, depending on the specific segmentation challenge [RLFF15]. For this reason, it would be ideal to capture data points possessing information about the human decision-making process and apply them to images at scale. This may include taking into account human labeling time per class, hesitation between clicks, relationship between shape boundary complexity and instance quantity, and more. By aggregating such statistics, a pattern may arise which can be leveraged as an additional automated annotation technique.

2. For those curious, the dataset and associated paper are accessible at https://www.trust-hub.org/#/data/pcb-images.

3. For a list of input validators and supported primitive types, refer to PyQtGraph's Parameter documentation.

References

[AAV+02] C Anagnostopoulos, I Anagnostopoulos, D Vergados, G Kouzas, E Kayafas, V Loumos, and G Stassinopoulos. High performance computing algorithms for textile quality control. *Mathematics and Computers in Simulation*, 60(3):389–400, September 2002. doi:10.1016/S0378-4754(02)00031-9.

[AML+19] Mukhil Azhagan, Dhwani Mehta, Hangwei Lu, Sudarshan Agrawal, Mark Tehranipoor, Damon L Woodard, Navid Asadizanjani, and Praveen Chawla. A review on automatic bill of material generation and visual inspection on PCBs. In *ISTFA 2019: Proceedings of the 45th International Symposium for Testing and Failure Analysis*, page 256. ASM International, 2019.

[AVK+01] C. Anagnostopoulos, D. Vergados, E. Kayafas, V. Loumos, and G. Stassinopoulos. A computer vision approach for textile quality control. *The Journal of Visualization and Computer Animation*, 12(1):31–44, 2001. doi:10.1002/vis.245.

[BWS+10] Steve Branson, Catherine Wah, Florian Schroff, Boris Babenko, Peter Welinder, Pietro Perona, and Serge Belongie. Visual recognition with humans in the loop. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 438–451, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[CZF+18] Qimin Cheng, Qian Zhang, Peng Fu, Conghuan Tu, and Sen Li. A survey and analysis on automatic image annotation. *Pattern Recognition*, 79:242–259, 2018. doi:10.1016/j.patcog.2018.02.017.

[DCE+] Janez Demšar, Tomaž Curk, Aleš Erjavec, Črt Gorup, Tomaž Hočevar, Mitar Milutinovič, Martin Možina, Matija Polajnar, Marko Toplak, and Anže Starič. Orange: Data mining toolbox in Python. 14(1):2349–2353.

[DS20] Polina Demochkina and Andrey V. Savchenko. Improving the accuracy of one-shot detectors for small objects in x-ray images. In *2020 International Russian Automation Conference (RusAutoCon)*, page 610–614. IEEE, September 2020. URL: https://ieeexplore.ieee.org/document/9208097/, doi:10.1109/RusAutoCon49822.2020.9208097.

[EGW+10] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, jun 2010. URL: https://doi.org/10.1007/s11263-009-0275-4, doi:10.1007/s11263-009-0275-4.

[FNK92] H. Fujisawa, Y. Nakano, and K. Kurino. Segmentation methods for character recognition: From segmentation to document structure analysis. *Proceedings of the IEEE*, 80(7):1079–1092, July 1992. doi:10.1109/5.156471.

[FRLL18] Max K. Ferguson, Ak Ronay, Yung-Tsun Tina Lee, and Kincho. H. Law. Detection and segmentation of manufacturing defects with convolutional neural networks and transfer learning. *Smart and sustainable manufacturing systems*, 2, 2018. doi:10.1520/SSMS20180033.

[GNP+04] Basilios Gatos, Kostas Ntzios, Ioannis Pratikakis, Sergios Petridis, T. Konidaris, and Stavros J. Perantonis. A segmentation-free recognition technique to assist old greek handwritten manuscript OCR. In Simone Marinai and Andreas R. Dengel, editors, *Document Analysis Systems VI*, Lecture Notes in Computer Science, pages 63–74, Berlin, Heidelberg, 2004. Springer. doi:10.1007/978-3-540-28640-0_7.

[IGSM14] D. K. Iakovidis, T. Goudas, C. Smailis, and I. Maglogiannis. Ratsnake: A versatile image annotation tool with application to computer-aided diagnosis, 2014. doi:10.1155/2014/286856.

[itL18] Humans in the Loop. The best image annotation platforms for computer vision (+ an honest review of each), October 2018. URL: https://hackernoon.com/the-best-image-annotation-platforms-for-computer-vision-an-honest-review-of-each-dac7f565fea.

[JB92] Anil K. Jain and Sushil Bhattacharjee. Text segmentation using gabor filters for automatic document processing. *Machine Vision and Applications*, 5(3):169–184, June 1992. doi:10.1007/BF02626996.

[JPRA20] Nathan Jessurun, Olivia Paradis, Alexandra Roberts, and Navid Asadizanjani. Component Detection and Evaluation Framework (CDEF): A Semantic Annotation Tool. *Microscopy and Microanalysis*, 26(S2):1470–1474, August 2020. doi:10.1017/S1431927620018243.

[KBO16] Made Windu Antara Kesiman, Jean-Christophe Burie, and Jean-Marc Ogier. A new scheme for text line and character segmentation from gray scale images of palm leaf manuscript. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 325–330, October 2016. doi:10.1109/ICFHR.2016.0068.

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[LSA+10] L'ubor Ladický, Paul Sturgess, Karteek Alahari, Chris Russell, and Philip H. S. Torr. What, where and how many? combining object detectors and crfs. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 424–437, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[MKS18] S. Mohajerani, T. A. Krammer, and P. Saeedi. A cloud detection algorithm for remote sensing images using fully convolutional neural networks. In *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*, page 1–5, August 2018. doi:10.1109/MMSP.2018.8547095.

[PJTA20] Olivia P Paradis, Nathan T Jessurun, Mark Tehranipoor, and Navid Asadizanjani. Color normalization for robust automatic bill of materials generation and visual inspection of pcbs. In *ISTFA 2020: Papers Accepted for the Planned 46th International Symposium for Testing and Failure Analysis*, International Symposium for Testing and Failure Analysis, pages 172–179, 2020. URL: https://doi.org/10.31399/asm.cp.istfa2020p0172https://dl.asminternational.org/istfa/proceedings-pdf/ISTFA2020/83348/172/425605/istfa2020p0172.pdf, doi:10.31399/asm.cp.istfa2020p0172.

[Ram] Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. 1(3):244–256. URL: https://www.sciencedirect.com/science/article/pii/S0146664X72800170, doi:10.1016/S0146-664X(72)80017-0.

[RLFF15] Olga Russakovsky, Li-Jia Li, and Li Fei-Fei. Best of both worlds: Human-machine collaboration for object annotation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 2121–2131. IEEE, June 2015. URL: http://ieeexplore.ieee.org/document/7298824/, doi:10.1109/CVPR.2015.7298824.

[RLO+17] Martin Rajchl, Matthew C. H. Lee, Ozan Oktay, Konstantinos Kamnitsas, Jonathan Passerat-Palmbach, Wenjia Bai, Mellisa Damodaram, Mary A. Rutherford, Joseph V. Hajnal, Bernhard Kainz, and Daniel Rueckert. DeepCut: Object segmentation from bounding box annotations using convolutional neural networks. *IEEE Transactions on Medical Imaging*, 36(2):674–683, February 2017. doi:10.1109/TMI.2016.2621185.

[SKM+10] Sascha Seifert, Michael Kelm, Manuel Moeller, Saikat Mukherjee, Alexander Cavallaro, Martin Huber, and Dorin Comaniciu. Semantic annotation of medical images. In Brent J. Liu and William W. Boonn, editors, *Medical Imaging 2010: Advanced PACS-based Imaging Informatics and Therapeutic Applications*, volume 7628, pages 43 – 50. International Society for Optics and Photonics, SPIE, 2010. URL: https://doi.org/10.1117/12.844207, doi:10.1117/12.844207.

[Spa20] SpaceNet. Multi-Temporal Urban Development Challenge. https://spacenet.ai/sn7-challenge/, June 2020.

[TFJ89] T. Taxt, P.J. Flynn, and A.K. Jain. Segmentation of document images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1322–1329, December 1989. doi:10.1109/34.41371.

[VGSG+19] Juan P. Vigueras-Guillén, Busra Sari, Stanley F. Goes, Hans G. Lemij, Jeroen van Rooij, Koenraad A. Vermeer, and Lucas J. van Vliet. Fully convolutional architecture vs sliding-window cnn for corneal endothelium cell segmentation. *BMC Biomedical Engineering*, 1(1):4, January 2019. doi:10.1186/s42490-019-0003-2.

[WYZZ09] C. Wang, Shuicheng Yan, Lei Zhang, and H. Zhang. Multi-label sparse coding for automatic image annotation. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, page 1643–1650, June 2009. doi:10.1109/CVPR.2009.5206866.

[YPH+06] Paul A. Yushkevich, Joseph Piven, Heather Cody Hazlett, Rachel Gimpel Smith, Sean Ho, James C. Gee, and Guido Gerig. User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability. *NeuroImage*, 31(3):1116–1128, July 2006. doi:10.1016/j.neuroimage.2006.01.015.