PixLabelCV - Labeling images for semantic segmentation fast, pixel-precise and offline

Dominik Schraml
Ilmenau University of Technology
SQB Ilmenau
Werner-von-Siemens Str. 9,
98693 Ilmenau, Germany
dominik.schraml@tu-ilmenau.de

Konstantin Trambickii
SQB Ilmenau
Werner-von-Siemens Str. 9,
98693 Ilmenau, Germany
konstantin.trambitckii@sqb-ilmenau.de

Gunther Notni
Ilmenau University of Technology
Ehrenbergstrasse 29,
98693 Ilmenau, Germany
gunther.notni@tu-ilmenau.de

ABSTRACT

Image annotation, also called labeling is a necessary task for any supervised learning approach to obtain ground truth data for model training. This article offers a comprehensive survey of contemporary image annotation tools, grouping freely accessible ones based on their service range, speed, and data privacy assurances.
In our exploration for tools capable of executing pixel-precise semantic labeling, we identified a shortage of swift, free image annotation tools that don't require users to upload their data to third-party servers. Therefore, we introduce "PixLabelCV" - a lightweight, fast, offline, and standalone annotation tool primarily developed to aid human annotators in achieving pixel-perfect labels promptly. Uniquely crafted to be freely available (open source) and non-server-based, it ensures enhanced privacy and efficiency. Hence, it is aimed to serve as an ideal tool to facilitate labeling data for smaller labs and businesses.
At its core, PixLabelCV fuses conventional labeling techniques such as delineating objects with rectangles or polygons with multiple computer vision algorithms. Spanning basic thresholding in RGB or HSV color space to more intricate procedures like flood fill or watershed the tool instantaneously computes and exhibits the resulting segmentations. Annotators can swiftly add these segments to a class label or refine them by adjusting parameters or markers before a quick repetition. To further augment the user experience, additional functionalities like morphological closing are incorporated, facilitating an intuitive labeling process. Another standout feature is its ability to uniquely assign pixels to singular classes, eliminating any potential overlap-induced ambiguities.

Keywords
Semantic Segmentation, Image Annotation Tools, Pixel-Precise Labeling, Computer Vision Algorithms, Annotation Software

1 INTRODUCTION

With the increasing use and application of deep neural networks for various tasks, labeled[1] training data for supervised learning is also increasingly needed. In this context, for many tasks such as autonomous driving or medical image analysis, it is not sufficient to identify only the objects present in the image, but it is also necessary to determine the exact position and size of the objects in terms of pixels. This process of assigning every pixel of an image to an object class is called semantic segmentation. In practice, this usually means that the resulting label is an equal-sized image in which each pixel has the value of the resulting class. However the cost of annotating every pixel to generate training data for a semantic segmentation network is immense [9]. Furthermore, studies by Zlateski et al. [17] on the quality of labels for semantic segmentation of urban scenes have shown that a larger coarsely annotated dataset can yield the same performance as a smaller finely, meaning pixel-precise, annotated dataset. Conversely, this means that a smaller number of images with precise semantic class labels is sufficient to achieve the same performance as is possible with a larger number of coarsely annotated images.

[1] The terms "labeling" and "annotation" are used interchangeably in the context of this work. In general, the term "label" is most often used to describe any output from a machine learning model, while "annotation" refers to the type of label, such as an image-sized pixel mask.

A particular burden for assigning a semantic class label to each pixel is the fact that object boundaries can be complex and therefore difficult to accurately annotate [14]. More precisely Zlateski et al. [17] deduce from the time annotators took during their experiment that it is very hard for humans to draw pixel-precise labels, as it depends on their vision, dexterity, etc., and will vary greatly among different individuals.

Our goal is to reduce this effort by providing an image annotation tool for semantic segmentation that is pixel-perfect yet fast and easy to use, after initial acclimatization.

2 IMAGE ANNOTATION PROGRAMS

2.1 Evaluation Criteria

We base our evaluation of image annotation tools on several critical criteria, essential for optimizing both the annotation process and the quality of the resulting data.

The first criterion is the **range of functions** offered by the tool. This encompasses the variety of shapes that can be annotated and extends to the inclusion of sophisticated tools, such as integrated AI models and advanced computer vision algorithms.

The most important evaluation factors are **speed** and **offline usability**. The latter includes setup and installation and goes hand in hand with security, as there is no need to upload your confidential data to third-party servers, which enables faster processing times and contributes to data security.

Label precision forms the cornerstone of effective semantic segmentation. High precision, particularly pixel-perfect mask creation over simpler coordinate-based shapes or bounding boxes, directly impacts the quality of machine learning models trained with annotated data. Precise pixel masks circumvent issues such as depth ordering in overlapping objects, ensuring each pixel is accurately classified without ambiguity.

While **ease of use** significantly contributes to a tool's appeal, its subjective nature makes it challenging to quantify. In this context, it is also important to note the tools excluded from our evaluation. Commercially licensed platforms and machine learning annotation tools, such as Hasty.ai, Labelbox, Prodigy, RectLabel and Supervisely were not considered due to their exclusive commercial licensing. Additionally, tools that were not accessible, faced technical issues, or failed to meet minimum usability requirements, such as GTCreator [4], RhobanTagger, DeepLabel, semantic-image-label-tool Scalabel, labelImg, LabelStudio, Labelflow and MedTagger were also excluded. Furthermore, tools demanding extensive user data rights, such as V7labs and Diffgram were omitted due to our data protection criteria.

2.2 Comparative Analysis of Image Annotation Tools

In our systematic evaluation of image annotation tools, as detailed in Table 1, we sought tools that balance functionality, user experience, and data privacy. The analysis showed a preference for server-based, browser-operated tools, predominantly outputting in JSON, XML, and CSV formats. These formats are less suited for semantic segmentation where precision is paramount, as they typically save only the coordinates of drawn shapes, potentially leading to inaccuracies and overlapping shapes.

Among the tools listed, only CVAT, label-studio, PixelAnnotationTool, Semantic Segmentation Editor and S3A offer the capability to save labeling results as pixel masks and are open source. While specialized tools like PixelAnnotationTool (utilizing the watershed algorithm for region creation) and S3A (semi-automatic labeling of printed circuit boards) target specific annotation tasks, they are limited in broader usability due to their restricted toolsets. Label Studio provides basic brush tools for pixel-precise labeling beyond simple shape annotations, while Semantic Segmentation Editor introduces a more advanced "magic tool" for efficient segmentation. However, CVAT stands out as the most feature-rich tool, equipped with a comprehensive suite of automation tools including advanced computer vision techniques like smart scissors and automatic annotation capabilities via the TensorFlow Object Detection API. It supports a wide range of export formats, making it a robust tool for complex labeling projects.

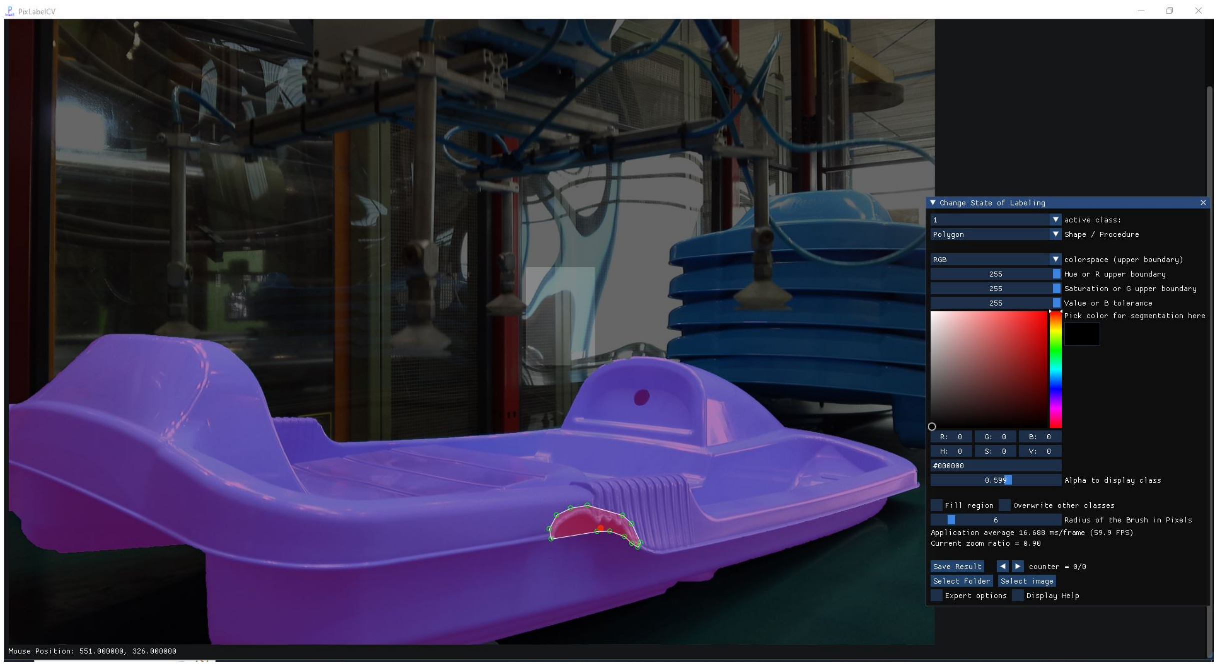
However, CVAT's extensive setup requirements for projects, annotators and "jobs", as well as the need to set up the server for local use, are not neglectable hurdles. Moreover, CVAT, like its counterparts, does not solve the depth order conflict inherent in overlapping object annotations.

3 PIXLABELCV

3.1 Structure of the Program

Recognizing a gap in the availability of lightweight, swift, and offline image annotation tools for semantic segmentation, that allows half-automated pixel-precise annotation, we developed our own solution.

The basic principle of using the program differs from that of the other software tested. By allowing the application of computer vision algorithms a segmentation in the chosen region of interest (ROI) is performed in the time of milliseconds, leading to a preliminary class mask, which is presented to the annotator. Provided the mask is precise enough to meet the annotator's needs, it can be swiftly added to the overall image mask. This iterative process continues until the whole image is labeled.


PixLabelCV

▼ Change State of Labeling
1 — active class:
Polygon — Shape / Procedure
RGB — colorspace (upper boundary)
255 — Hue or R upper boundary
255 — Saturation or G upper boundary
255 — Value or B tolerance
Pick color for segmentation here
R: 0 G: 0 B: 0
H: 0 S: 0 V: 0
#000000
0.599 — Alpha to display class
Fill region — Overwrite other classes
6 — Radius of the Brush in Pixels
Application average 16.688 ms/frame (59.9 FPS)
Current zoom ratio = 0.90
Save Result ◀ ▶ counter = 0/0
Select Folder Select image
Expert options Display Help

Mouse Position: 551.000000, 326.000000

Figure 1: PixLableCV interface, annotations of the sled (purple) using GrabCut or Watershed algorithm and the defect (red) using polygon annotation.

The PixLabelCV software[2] facilitates the annotation process by enabling human annotators to demarcate regions within images. These regions can be assigned to various classes using standard annotation shapes, including rectangles, circles, and polygons. Additionally, individual pixels in the image can be marked to belong to a specific class. After delineating a ROI, users have the option to apply specific computer vision algorithms. For instance, they can employ thresholding in either HSV or RGB color space, setting both upper and lower limits for each color channel. Alternatively, the floodfill algorithm can be initiated within the ROI, originating from the current cursor location. Furthermore, the watershed algorithm can be utilized to segment the image based on the marked points. Details on the technical implementation can be found in Appendix A.

3.2 Usage of the Program

After starting the program, the user can either load an individual image or select a directory containing multiple images. When a directory is chosen, the program automatically recognizes images in BMP, JPG, PNG, or TIFF formats and starts with the first image.

Class Selection: Choose the label class using the combobox or by pressing the respective class number key.

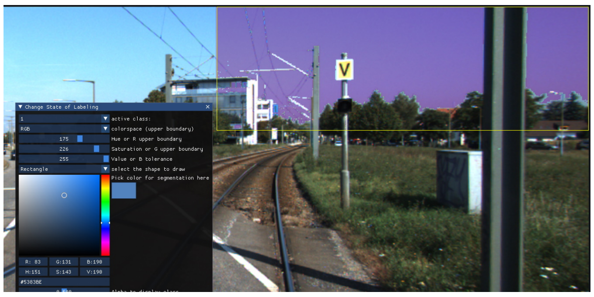
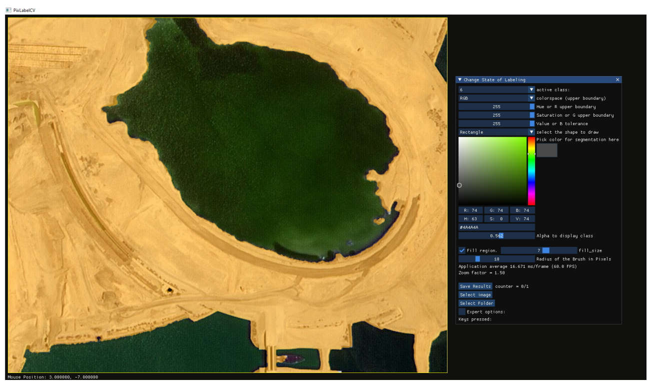

▼ Change State of Labeling
1 — active class:
RGB — colorspace (upper boundary)
175 — Hue or R upper boundary
226 — Saturation or G upper boundary
255 — Value or B tolerance
Rectangle — select the shape to draw
Pick color for segmentation here
R: 83 G:131 B:190
H:151 S:143 V:190
#5383BE

Figure 2: Applying a Threshold to segment the region "sky".

For classes greater than 10, hold the *Alt* key while pressing the class number key. The selected class can be changed anytime even after the segmentation to add the resulting region to another class or remove it.

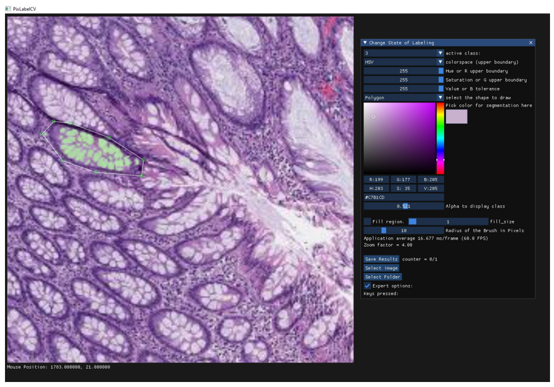
Drawing Shapes and Markers: For basic annotations, enclose the object within a rectangle. More intricate shapes can be outlined using polygons, which can be further refined as shown in Figure 1. Round structures are best represented using the circle tool: initiate by positioning it on an edge contour, adjust segmentation parameters or apply flood fill, then fine-tune by dragging the contour and modifying the radius. A pixel brush, whose radius is adjustable, is ideal for adding or subtracting pixels from a specific class region. Additionally, marker points can be set for each class, which serve as seed for the application of the watershed algorithm in segmentation tasks. The same applies to the GrabCut implementation.

[2] PixLabelCV software will be available at `https://sqb-ilmenau.de/pixlabelcv`. The source code can be found on GitHub: `https://github.com/dominiks-dev/PixLabelCV`



```
PixLabelCV
▼ Change State of Labeling
6            active class:
RGB          colorspace (upper boundary)
255          Hue or R upper boundary
255          Saturation or G upper boundary
255          Value or B tolerance
Rectangle    select the shape to draw
             Pick color for segmentation here
R: 74   G: 74   B: 74
H: 63   S:  0   V: 74
#4A4A4A
0.562        Alpha to display class
✓ Fill region.   7   fill_size
10           Radius of the Brush in Pixels
Application average 16.671 ms/frame (60.0 FPS)
Zoom factor = 1.50
Save Results   counter = 0/1
Select image
Select Folder
Expert options:
Keys pressed:
Mouse Position: 3.000000, -7.000000
```

(a) Segmenting large ground area (highlighted yellow) accurately from the water area (landcover dataset).



```
PixLabelCV
▼ Change State of Labeling
3            active class:
HSV          colorspace (upper boundary)
255          Hue or R upper boundary
255          Saturation or G upper boundary
255          Value or B tolerance
Polygon      select the shape to draw
             Pick color for segmentation here
R:199   G:177   B:205
H:203   S: 35   V:205
#C7B1CD
0.521        Alpha to display class
Fill region.   1   fill_size
10           Radius of the Brush in Pixels
Application average 16.677 ms/frame (60.0 FPS)
Zoom factor = 4.00
Save Results   counter = 0/1
Select image
Select Folder
✓ Expert options:
Keys pressed:
Mouse Position: 1703.000000, 21.000000
```

(b) Pixel-precise cell segmentation using thresholding in a polygon (Histopathology dataset).

Figure 3: Examples of efficient PixLabelCV usage.

Segmentation Process: During the segmentation process, algorithms like thresholding, flood fill, watershed [12] or GrabCut [13] are applied using previously set markers or shapes. For thresholding, users can activate a color picker, which auto-selects the object's pixel color for segmentation. This selection can be refined by adjusting the upper and lower boundary HSV values, as illustrated in Figure 2. Segmentation is initiated with mouse actions or through an optimized keyboard setup.[3] Once segmentation is satisfactory, it can be added to the chosen class. If adjustments are needed, parameters can be tweaked and segmentation can be performed again within a few milliseconds. Flood fill segmentation starts at the cursor's pixel and users can choose if the current segmented pixel mask should replace pixels that have already been assigned to other classes. The same applies to the GrabCut algorithm for separating the current class from the background (see Figure 4).

Modifications: For any modifications, unwanted regions or individual pixels can be removed by either allocating them to class 0 or background. This class is designed specifically for erasing unnecessary elements, with the brush tool also available for this task. It is also possible to overwrite all pixels that already belong to another class by explicitly setting this option. The pixel brush tool is versatile, enabling both the addition and erasure of pixels from a given class region. Lastly, the software facilitates precise labeling with its zoom functionality, swiftly magnifying up to 4x when needed.

Efficient Annotation Strategy: To optimize the annotation workflow, a highly effective and swift technique involves beginning with the placement of watershed marker points across the entire image for every class. This initial segmentation can be computed in a few hundred milliseconds, even for 4k images, producing a relatively accurate pixel mask. If necessary, adding a few more markers can refine this result before further enhancing the mask's accuracy using other functions such as floodfill or the pixel-brush. This approach not only speeds up the process but also leverages the software's capabilities for achieving precise pixel masks with minimal manual intervention.

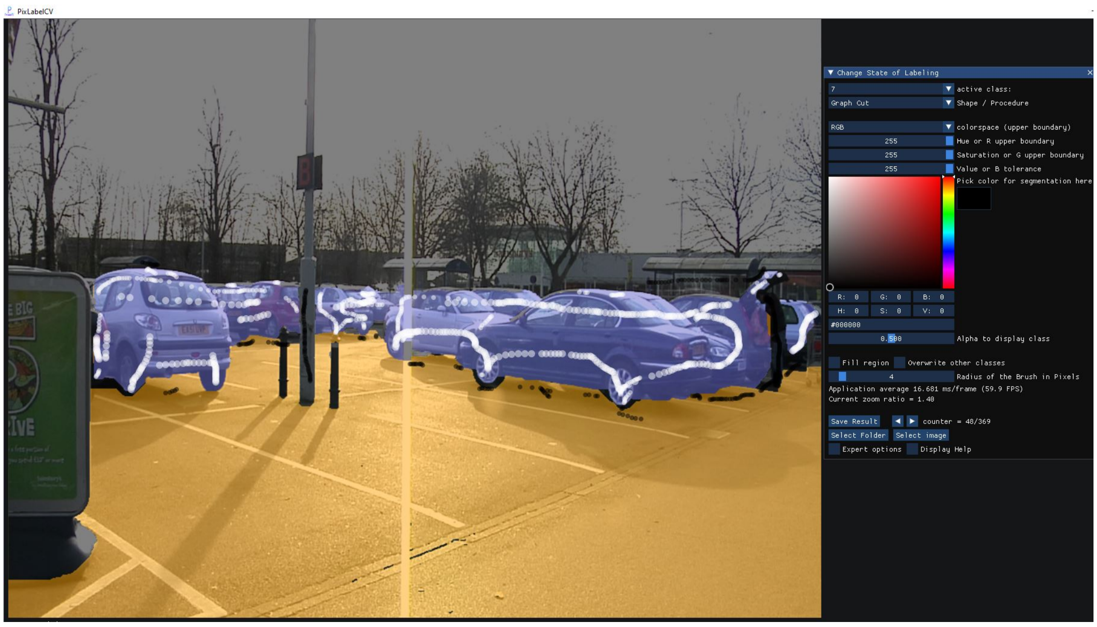
Moreover, it's advisable to initially focus on segmenting objects or classes that are enclosed within others. For shared boundaries between distinct classes, start the segmentation with simpler-to-demarcate regions, employing the flood fill algorithm for efficiency. Afterward, encompass the entire region within a bounding box and designate it to the secondary class. Due to the program's design, which by default prevents pixels already assigned to one class from being overridden by another, any pixels not segmented in the initial step will automatically be attributed to the secondary class. This strategy streamlines the annotation process by eliminating the need to meticulously draw an additional polygon, with these computations being rapidly performed in just milliseconds.

Saving and Advancing: Upon completing the annotation, the output is a pixel mask where each pixel corresponds to its class value. This mask can be stored as an 8-bit depth PNG file or if specifically chosen each class mask can be saved in a separate file so that multiple labels per image are possible. If a directory path was initially provided containing multiple images, a single button press both saves the current annotation and loads the subsequent image for labeling.

3.3 Comparison to Segment Anything Model (SAM)

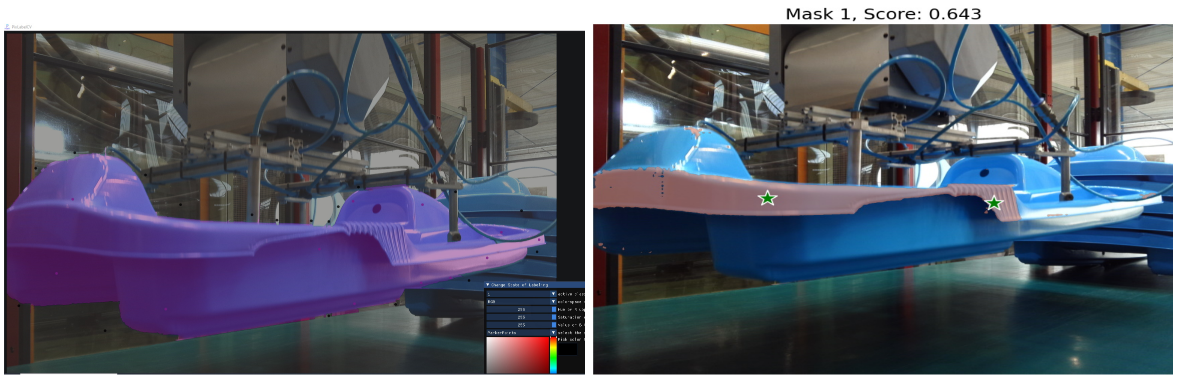
The Segment-Anything Model (SAM) [10] is notable for its ability to segment unknown objects within images. Despite its extensive capabilities, the authors

[3] All operations mentioned are assigned to keys conveniently located around the 'ASDF' keys, optimizing for dual-hand usage thereby enhancing efficiency in labeling.



```
▼ Change State of Labeling
7            active class:
Graph Cut    Shape / Procedure
RGB          colorspace (upper boundary)
255          Hue or R upper boundary
255          Saturation or G upper boundary
255          Value or B tolerance
             Pick color for segmentation here
R: 0   G: 0   B: 0
H: 0   S: 0   V: 0
#000000
0.500        Alpha to display class
Fill region   Overwrite other classes
4            Radius of the Brush in Pixels
Application average 16.681 ms/frame (59.9 FPS)
Current zoom ratio = 1.40
Save Result  ◀ ▶  counter = 48/369
Select Folder  Select image
Expert options  Display Help
```

Figure 4: Graph Cuts - white (class) and black (background) - to segment cars.



(a) Results of our tool (PixLabelCV) setting marker points and applying watershed algorithm.

(b) SAM segmentation result on sled using only few point inputs and no bounding rectangle.

Figure 5: Comparing SAM to watershed algorithm on custom data.

of SAM have acknowledged specific shortcomings, including occasionally missing finer structures in images, creating minor hallucinated disconnected components, and not consistently delineating sharp boundaries. This last limitation is particularly evident when juxtaposed against techniques that use a 'zoom-in' approach. Moreover, while SAM's design emphasizes versatility for a broad range of applications, this generality can sometimes result in precision trade-offs.

While SAM demonstrates proficiency when provided with a clear bounding rectangle around an object, as exemplified by its performance on public datasets like KITTI, it struggles when domain-specific expertise is crucial for pixel-accurate segmentation. We encountered this limitation with medical images and our proprietary dataset, especially for objects lacking a clear bounding rectangle, such as the sled in our injection-molded dataset (see Figure 5). Additionally, while the decoder part of SAM operates quickly, encoding an image can take many seconds to multiple minutes, depending on the image size and is restricted to a maximum input image size of 1024x1024 pixels. In terms of operational efficiency, SAM proved to be slower with larger images and consumed substantial amounts of memory compared to our tool. When provided with ample input points, SAM's performance becomes comparable to our tool.

In conclusion, while SAM offers a versatile approach to image segmentation, our labeling program adeptly mitigates some of its shortcomings, particularly in refining masks that are either too expansive or too constricted. Although SAM's one-shot characteristic presents clear advantages for well-known datasets such as urban scenes where the model has been extensively trained, it struggles with large images and user-specific datasets from sectors like industrial manufacturing.

4 USER STUDY

4.1 Setup of the Study

To evaluate our labeling program for semantic segmentation, we conducted a user study with five participants. The small number was chosen for feasibility, focusing more on gaining qualitative insights into the usability and effectiveness of our program rather than conducting a quantitative hypothesis test. The study was carried out as follows. First, the students were given a brief introduction to each tool to ensure they had a basic understanding of how to use them. Then the task was to annotate one to three images from four different categories and datasets (KITTI, landcover, MHIST and our custom dataset of an injection molded part[4]). After performing the task each participant filled out a questionnaire in open-ended design which contained the following questions about the two tools:

1. Which tool did you find faster for annotation images? Why? Options: CVAT, PixLabelCV, no difference
2. Which tools did you use the most often?
3. How much faster is it to use the computer vision tools to annotate instead of only annotating with shapes?
4. Which tool do you believe allowed you to annotate images more accurately? Options: CVAT, PixLabelCV, no difference
5. Rate your overall satisfaction with each tool: Scale from 1 (very unsatisfied) to 5 (very satisfied)

4.2 User Study Evaluation and Findings

Analysis of the user study results indicated that PixLabelCV, with an average satisfaction score of 3.4 (SD 0.49), is slightly behind CVAT's 3.6 (SD 0.49) out of 5 in terms of usability and overall user satisfaction. Given that all ratings were either 3 or 4 out of 5, we consider PixLabelCV to be roughly on par with one of the leading state-of-the-art labeling tools.

Regarding the speed of annotation, half of the participants found PixLabelCV to be faster, while the other half favored CVAT, with one participant reporting no significant difference between the two tools. It should be noted that this assessment was based on the time taken to label a series of images from different domains (with only one per domain). For both tools, the combination of advanced computer vision techniques (*watershed* for PixLabelCV and *"smart scissors"* for CVAT) with shape tools, particularly the polygon shape, was credited for the perceived increase in labeling speed.

The most notable distinction between the tools emerged in their accuracy. A vast majority - 80% of participants - identified PixLabelCV as enabling more precise labeling compared to CVAT. This underlines PixLabelCV's capability to support detailed and accurate annotations, essential for high-quality semantic segmentation.

Participants identified key features and advantages of each program. PixLabelCV's incorporation of the *watershed algorithm* was particularly valued for its effectiveness in images with distinct color differences and blurriness, such as medical images. This feature positions PixLabelCV as particularly advantageous for projects requiring complex segmentation tasks. Conversely, the *"smart scissors"* feature in CVAT was acknowledged for its proficiency in swiftly generating accurate polygon labels, illustrating CVAT's efficiency in more conventional annotation scenarios.

4.3 Analysis

While the relatively small number of participants may be perceived as a limitation of our user study, it is important to note that our focus was on evaluating the practical utility and relevance of PixLabelCV rather than conducting a quantitative hypothesis test. The study reveals that PixLabelCV is on par with CVAT in terms of usability and excels in enabling more precise annotations, affirming its capability as a powerful tool for complex segmentation tasks and as a viable alternative for unique and custom data applications.

Integration and Efficiency: PixLabelCV distinguishes itself through the seamless integration of diverse tools to achieve enhanced boundary precision. The process typically begins with an initial, coarse delineation of boundaries using the watershed method. This is followed by further refinements via the pixel-brush and other tools for precise, custom-tailored results. This integrated toolset proves especially advantageous for repetitive tasks, such as quality control in industrial settings, where speed and accuracy are paramount. Notably, PixLabelCV demonstrates an inherent advantage in speed over AI-model-dependent approaches like SAM, particularly for sets of similar scenes.

Offline Usability and Performance: PixLabelCV is engineered for flexibility, leveraging GPU capabilities for accelerated algorithm performance when available, with fallback to CPU processing. This architecture supports mobile use on laptops, eliminating the need for

[4] The dataset contains images of injection molded sleds for quality assessment. There are images of good products and ones of sleds that show one of the various defects that occasionally occur during injection molding, such as short shot, flash and color streak.

server connectivity and data transfer, thereby streamlining the labeling process.

Handling Large Images: The tool's capability to efficiently process and segment large images, such as those found in the Landcover.ai dataset, without significant delays, highlights its suitability for extensive geographic or environmental datasets.

Deployment and Precision: PixLabelCV's minimalistic design and straightforward deployment process facilitate quick setup and immediate use. Its precision, particularly evident in medical imaging tasks, leverages subtle color and brightness variations for meticulous segmentation, further enhanced by features like flood-fill for better edge distinction.

Considerations: While PixLabelCV excels in many areas, it encounters limitations in generic scenes where pre-trained AI models are prevalent. In such scenarios, tools capable of pre-labeling with task-specific AI may offer more efficiency. Additionally, the tool's rich feature set and reliance on image processing principles may pose a learning challenge for novices, contrasting with the more intuitive nature of platforms like CVAT or the straightforwardness of SAM for basic tasks.

Domain Applicability: PixLabelCV's effectiveness varies by application domain. While it offers considerable advantages for medical imaging and specialized custom datasets, its utility may not extend as effectively to areas well-served by existing AI models, such as street scenes.

In conclusion, PixLabelCV represents a significant advancement in semantic image annotation, particularly for users seeking high-speed, precision-driven tools for specialized applications. However, potential users must balance these benefits against the need for a foundational understanding of image processing and the tool's specific capabilities.

5 CONCLUSION

Image annotation remains a pivotal component in the realm of supervised machine learning and deep learning paradigms. Our comprehensive examination of current image annotation tools revealed a significant gap in the market: the absence of swift, freely available annotation tools that also prioritize data privacy. In response, we introduced "PixLabelCV," a streamlined yet powerful offline tool crafted to enable annotators to quickly generate pixel-perfect labels. By harnessing core computer vision techniques such as the watershed and flood fill algorithms and integrating these with basic shape tools, PixLabelCV significantly enhances precision, facilitating the rapid development of high-quality pixel masks.

The user study conducted to evaluate PixLabelCV against established tools like CVAT highlighted its strengths, particularly in precision and specialized task performance. PixLabelCV demonstrated a competitive edge, standing on par with CVAT in terms of usability while outperforming it in accuracy as acknowledged by the majority of participants. This underscores PixLabelCV's potential as a potent tool for complex segmentation tasks, offering a robust alternative for unique and custom data and use cases.

Looking forward, we aim to amplify PixLabelCV's capabilities by integrating the Segment-Anything Model (SAM) through the ONNX Runtime. The fusion of SAM with PixLabelCV's existing computer vision methodologies promises to enhance initial segmentation efforts, paving the way for a more simple usability for novices.

In summation, PixLabelCV aims to catalyze both the precision and speed of the annotation process, thereby optimizing the generation of premium-quality labeled datasets.

6 REFERENCES

[1] Alper Aksac, Douglas J. Demetrick, Tansel Ozyer, and Reda Alhajj. Brecahad: a dataset for breast cancer histopathological annotation and diagnosis. *BMC research notes*, 12(1):82, 2019. 8

[2] B Albertina, M Watson, C Holback, R Jarosz, S Kirk, Y Lee, and J Lemmerman. Radiology data from the cancer genome atlas lung adenocarcinoma [tcga-luad] collection. *The Cancer Imaging Archive*, 10:K9, 2016. 8

[3] Hassan Alhaija, Siva Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *International Journal of Computer Vision (IJCV)*, 2018. 8

[4] Jorge Bernal, Aymeric Histace, Marc Masana, Quentin Angermann, Cristina Sánchez-Montes, Cristina Rodríguez de Miguel, Maroua Hammami, Ana García-Rodríguez, Henry Córdova, Olivier Romain, Gloria Fernández-Esparrach, Xavier Dray, and F. Javier Sánchez. Gtcreator: a flexible annotation tool for image-based datasets. *International Journal of Computer Assisted Radiology and Surgery*, 14(2):191–201, 2019. 2

[5] Adrian Boguszewski, Dominik Batorski, Natalia Ziemba-Jankowska, Tomasz Dziedzic, and Anna Zambrzycka. Landcover.ai: Dataset for automatic mapping of buildings, woodlands, water and roads from aerial imagery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1102–1110, June 2021. 8

[6] Gabriel J Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation

and recognition using structure from motion point clouds. In *Computer Vision–ECCV 2008: 10th European Conference on Computer Vision, 2008, Proceedings, Part I 10*, pages 44–57. Springer, 2008. 8

[7] Kenneth Clark, Bruce Vendt, Kirk Smith, John Freymann, Justin Kirby, Paul Koppel, Stephen Moore, Stanley Phillips, David Maffitt, Michael Pringle, et al. The cancer imaging archive (tcia): maintaining and operating a public information repository. *Journal of digital imaging*, 26:1045–1057, 2013. 8

[8] Omar Cornut. Dear imgui. `https://github.com/ocornut/imgui`, Accessed on: January 2023. 8

[9] Tejaswi Kasarla, Gattigorla Nagendar, Guruprasad M Hegde, Vineeth Balasubramanian, and CV Jawahar. Region-based active learning for efficient labeling in semantic segmentation. In *2019 IEEE winter conference on applications of computer vision (WACV)*, pages 1109–1117. IEEE, 2019. 1

[10] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023. 4

[11] Thomas Köhler, Attila Budai, Martin F Kraus, Jan Odstrčilik, Georg Michelson, and Joachim Hornegger. Automatic no-reference quality assessment for retinal fundus images using vessel segmentation. In *Proceedings of the 26th IEEE international symposium on computer-based medical systems*, pages 95–100. IEEE, 2013. 8

[12] Fernand Meyer. Color image segmentation. In *1992 international conference on image processing and its applications*, pages 303–306. IET, 1992. 4

[13] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut" interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 23(3):309–314, 2004. 4

[14] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 77:157–173, 2008. 2

[15] Amber L Simpson, Michela Antonelli, Spyridon Bakas, Michel Bilello, Keyvan Farahani, Bram Van Ginneken, Annette Kopp-Schneider, Bennett A Landman, Geert Litjens, Bjoern Menze, et al. A large annotated medical image dataset for the development and evaluation of segmentation algorithms. *arXiv:1902.09063*, 2019. 8

[16] Jerry Wei, Arief Suriawinata, Bing Ren, Xiaoying Liu, Mikhail Lisovsky, Louis Vaickus, Charles Brown, Michael Baker, Naofumi Tomita, Lorenzo Torresani, et al. A petri dish for histopathology image analysis. In *19th International Conference on Artificial Intelligence in Medicine, Proceedings*, pages 11–24. Springer, 2021. 8

[17] Aleksandar Zlateski, Ronnachai Jaroensri, Prafull Sharma, and Frédo Durand. On the importance of label quality for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1479–1487, 2018. 1, 2

A PIXLABELCV - TECHNICAL IMPLEMENTATION

Our program, named "PixLabelCV", operates as a Direct3D11 graphics application. It integrates DearImGui [8] to manage graphics, user input, and the display of GUI elements. Image processing is primarily conducted using the Open Source Computer Vision Library (OpenCV). Once processed, the resultant image is transferred to the Direct3D graphics buffer for rendering. All other operations, including filesystem access and basic computations, are executed using plain C++ in compliance with the ISO C++17 standard. The program is compiled into an executable file, which needs only to be copied together with a few dynamically linked libraries and can then be started directly without installation required.

B DATASETS USED

In addition to other and customer-specific data sets with injection molded objects, the following data sets were used to test the segmentation programs. Our thanks go to the providers at this point.

- BrainTumour from Medical Segmentation Decathlon [15]
- CamVid (Cambridge-Driving Labeled Video Database) dataset [6] obtained from kaggle
- CT Medical images from cancer imaging archive [2] [7] obtained from kaggle
- HRF (High-Resolution Fundus) [11]
- KITTI dataset [3]
- Landcover.ai V1 [5]
- MHIST: A Minimalist Histopathology Image Analysis Dataset [16]
- BreCaHAD [1]

C TABLE OF LABELING PROGRAMS

Name	Type of tool	Shapes for annotation	Export Formats	Pixel Mask	License	Source code
Coco-annotator	server, docker image	rectangle, circle, polygon, pixel brush, magic wand,	JSON	no	MIT Licence	yes
Colabler	offline installer	only brush for pixel wise labeling	JSON, XML, PNG	yes, separate 32bit Png per class	free for personal use	no
CVAT	server, can be run locally, docker images	rectangle, polygon, ellipse, pixel-brush, AI assistant, Scissors using CV	JSON, XML, PNG and more	yes, 16 bit png and 24 color png	MIT license	yes
DarkLabel	offline installer	rectangle	XML, TXT, CSV	no	free for non-commerical use	no
imglab	server, free online tool available	rectangle, circle, polygon, point	JSON, XML	no	MIT license	yes
labelme	offline, python package	rectangle, polygon, circle, line, point	JSON	JSON convert to PNG	GPL-3.0	yes
label-studio	server, can be run locally via python script	rectangle, polygon, pixel-brush (alternatively)	JSON, CSV, COCO, PNG	yes, separate .png file (8bit) per class	Apache-2.0	yes
Make-Sense	server, can be run locally, docker image	rectangle, polygon, line, point	JSON, CSV, TXT, XML	no	GPL-3.0	yes
MyVision	online, browser based	rectangle, polygon	JSON, CSV, TXT, XML	no	GPL-3.0 license	yes
PixelAnnotationTool	offline, executable .exe	pixel-brush (for watershed)	Binary PNG	yes, png with color values	LGPL-3.0	yes
Ratsnake	offline, executable .jar	polygon, gridbased-"brush"	JPG	yes, separate .jpg file (8bit) per class	sourcecode	no
Remo	server running on local machine, access via browser	rectangle, polygon	JSON, CSV	no	CC BY-ND 4.0	yes
Roboflow	machine learning platform with annotation tool, browser-based	rectangle, polygon, "smart polygon", ai assistant	JSON	no	commercial, free for academic and personal use	no
Semantic Segmentation Editor	server, can be run locally, docker image	rectangle, polygon, magic tool	PNG, JSON	yes, 32 bit color png	MIT license	yes
Semi-Supervised Semantic Annotator (S3A)	python package (py qt)	automated, rectangle, polygon, ellipse, point	CSV, PNG, cusom zip file	yes, 16 bit png	free to use (except Qt licence)	yes
Superannotate	desktop installer and online, browser based	rectangle, ellipse, polygon, polyline, cuboid	JSON, PNG	yes, 32 bit color png	free for startups and academic use	no
Universal Data Tool	server, can be installed locally and online tool	rectangle, polygon, point	JSON, CSV	no, coverter does not work	MIT Licence	yes
VGG Image Annotator (VIA)	online, browser based	rectangle, ellipse, polygon, line	JSON, CSV	no	BSD 2-Clause	yes
VoTTÂ	server, can be hosted locally	rectangle, polygons	JSON, CSV, Azure	no	MIT license	yes

Table 1: Overview of labeling tools