SegBuilder: A Semi-automatic Annotation Tool for Segmentation

Md Alimoor Reza, Eric Manley, Sean Chen, Sameer Chaudhary, Jacob Elafros
Drake University, Des Moines, Iowa, USA
{md.reza, eric.manley, sean.chen, sameer.chaudhary, jake.elafros@drake.edu}@drake.edu

Abstract

This paper addresses the problem of image annotation for segmentation tasks. Semantic segmentation involves labeling each pixel in an image with predefined categories, such as sky, cars, roads, and humans. Deep learning models require numerous annotated images for effective training, but manual annotation is slow and time-consuming. To mitigate this challenge, we leverage the Segment Anything Model (SAM) [23] – a vision foundation model. We introduce SegBuilder, a framework that incorporates SAM to automatically generate segments, which are then tagged by human annotators using a quick selection list. To demonstrate SegBuilder's effectiveness, we introduced a novel dataset for image segmentation in underwater environments featuring animals such as sea lions, beavers, and jellyfish. Experiments on this dataset showed that SegBuilder significantly speeds up the annotation process compared to the publicly available tool, Label Studio [31]. SegBuilder also includes a free-form drawing tool, allowing users to create correct segments missed by SAM. This feature is particularly useful for scenes with shadows, camouflaged objects, and part-based segmentation tasks where SAM falls short. Experimentally, we demonstrated SegBuilder's efficacy in these scenarios, showcasing its potential for generating pixel-wise annotations crucial for training robust deep learning models for semantic segmentation.

1. Introduction

Semantic segmentation models are gaining critical importance due to their broad range of applications in fields such as autonomous driving, medical imaging, and virtual/augmented reality. The goal of a semantic segmentation model is to automatically assign labels to each pixel in an image, facilitating dense pixel-level prediction. The availability of a sufficient amount of pixel-level image annotation data is pivotal for training deep neural network models intended for semantic segmentation tasks. Manual annotation for image segmentation, facilitated by tools such as LabelStudio [31] or LabelMe [33], typically involves annotators drawing polygons, which poses scalability challenges due to its time-consuming nature, labor intensity, and the need for user expertise. To address these issues, we propose SegBuilder, a semi-automatic tool designed to enhance the efficiency of manual annotation. SegBuilder streamlines the annotation process by utilizing a vision-foundation model such as the Segment Anything Model (SAM) [10] to create object masks. These masks represent meaningful regions but lack inherent class information. We enable human annotators to quickly assign labels from a selection list, achieving pixel-wise annotation for each selected mask in a semi-automatic manner. This approach accelerates annotation, reduces labor requirements, and enhances cost-effectiveness compared to current tools. Our contributions are threefold, as discussed next.

First, we introduced a novel semi-automatic web-based tool called SegBuilder, representing a contribution at the system level in terms of both the underlying approach and the implemented tool itself. It is a novel system that combines SAM-generated segmentation with a human-in-the-loop approach to semi-automate the annotation process. It can be applied across different domains and types of image data, making it versatile for a wide range of applications. We have publicly released our tool at https://github.com/alimoorreza/segbuilder-v1, facilitating access for the research community and encouraging potential collaborative improvements and future development.

Second, we demonstrated the effectiveness of SegBuilder by conducting comprehensive experimentation for a variety of segmentation annotation tasks in diverse environments such as outdoor and underwater settings.

i) Semantic segmentation annotation task. We began by fully annotating a dataset of 635 images using SegBuilder in a semi-automatic manner. To demonstrate the efficiency of SegBuilder, we compared the time savings achieved with those of manual annotation methods [31]. Our comparison with existing manual annotation tools [31] indicates that SegBuilder can accelerate the annotation process by a factor of more than three. To further illustrate the application of these annotations, we trained a semantic segmentation

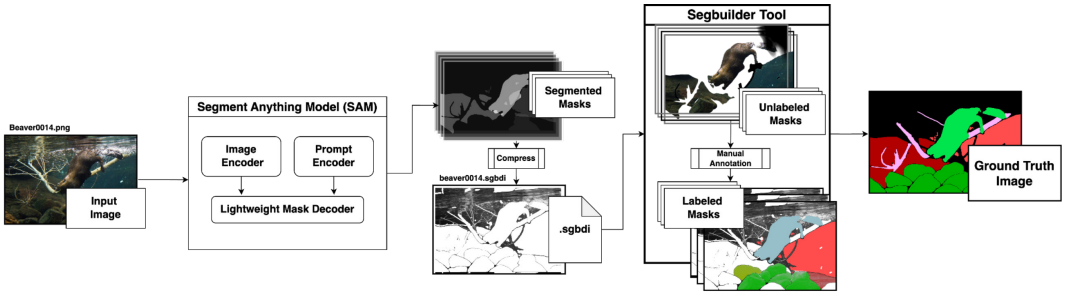


Figure 1. Semi-automatic pixel-level annotation process using SegBuilder.

model using the annotated images, thereby demonstrating the efficacy of our annotations.

ii) Scene annotation under challenging scenarios. Segmentation using SAM [23] and its derivatives, such as LabelMe-with-SAM [3], fails to automatically segment scenes in challenging scenarios, including i) camouflage, and ii) shadows [10]. We selected a subset of images featuring camouflage and shadows to demonstrate SAM's inconsistent segmentation performance in these contexts. Our tool, SegBuilder, enables the readjustment of annotations using additional features, and our experiments showed that it effectively facilitates the segmentation of images containing camouflage and shadows. Although the SAM Adapter [10] was introduced to enhance the detection of camouflaged objects and shadows, our approach does not involve proposing a new adaptation method. Instead, SegBuilder provides an alternative solution for addressing missing regions in camouflage and shadow detection through a human-in-the-loop strategy, rather than integrating a new trainable module like SAM Adapter.

iii) Part annotation task. The part annotation task involves annotating images with individual parts (e.g., refrigerator door, cabinet door, microwave handle, etc.) [26, 35]. Since SAM was trained on full object mask annotations by design, it will not perform well at predicting parts. However, part annotation is crucial for applications in robotics and virtual reality (VR) or augmented reality (AR). Therefore, applications aimed at predicting parts require further annotation, which is where our tool becomes useful. SegBuilder allows us to readjust the annotations using our additional tool. We demonstrated that semantic part annotation on the PartImageNet dataset [20] using SAM fails to cover important object parts. In contrast, SegBuilder effectively addresses these annotation failures, ensuring comprehensive coverage of all relevant parts.

Finally, we contribute a new dataset called **UWSv2** ("Underwater Segmentation version 2") of 635 images which complements the existing underwater datasets by adding more diversity to the animal-centric semantic segmentation dataset [22].

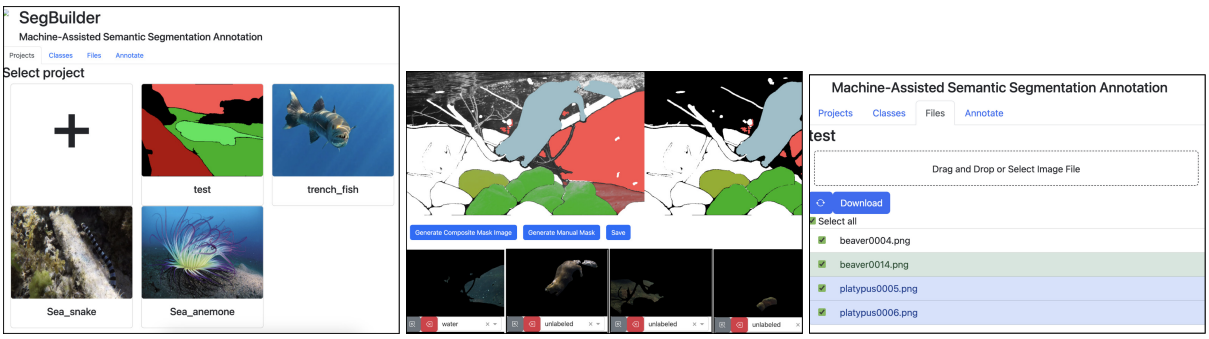
Why we need a new tool like SegBuilder. Our tool enables the integration of SAM while simultaneously providing functionality to correct and refine the segmentation annotations. There are two key reasons why our tool could prove useful. *First,* SAM struggles to segment images that are perceptually challenging even for humans, such as those involving camouflage or shadows [10]. It also faces difficulties with systems that require part annotation [36]. Therefore, simply integrating SAM is inadequate unless it is adapted through our refinement annotation tool to address issues with incomplete annotations. LabelMe [29], a well-known annotation tool in computer vision, does not natively integrate with models like SAM. Although some community members have created SAM integrations for LabelMe, these projects are often personal, with limited documentation and maintenance. For instance, LabelMe-with-SAM [3][1] faces challenges in effectively combining LabelMe with SAM due to limited documentation and lack of native integration with large models. *Second,* many of these systems offer AI capabilities like SAM integration, but only through an additional paid subscription [3], which could hinder academic research efforts due to the cost barrier.

2. Related Work

Annotation tools for segmentation. Pixel-level image annotation is significantly more time-consuming compared to bounding box annotation for object detection. Segments are typically annotated as polygonal regions defined by a series of points on the image. Various tools exist for this purpose, including LibLabel [18], the COCO-Stuff Annotation Tool [12], and LabelMe [29, 33]. A more recent addition to these tools is Label Studio [31].

Segmentation. Classical segmentation methods employ a variety of algorithms, such as minimum spanning tree-based segmentation [15], normalized cut [30], and clustering-based segmentation like SLIC [7]. These techniques have the advantage of not relying on training data; however, they are often prone to under-segmentation or

[1] https://github.com/originlake/labelme-with-segment-anything



(a) SegBuilder Interface: Project Tab (b) SegBuilder Interface: Annotation Tab (c) SegBuilder Interface: Files Tab

Figure 2. Sample views of the interface of our web-based tool SegBuilder.

over-segmentation and require parameter tuning. More recently, models like GPT-4V [27] and LLaVa [24] have been trained on diverse, internet-scale data using self-supervised learning. These models effectively align visual concepts with language descriptions and can be fine-tuned for numerous downstream applications. These foundation models [8, 38] have demonstrated impressive zero-shot generalization across a wide range of tasks. Following this trend, the Segment Anything Model (SAM) [23]—a vision foundation model—has demonstrated a remarkable ability to segment scenes in open-world settings. This capability has paved the way for its application to a wide range of problems, including segmentation for RGB-D scenes [9], medical image segmentation [25], image inpainting [39], image tagging [40], and object tracking [37]. Building on this foundation, we developed SegBuilder to leverage SAM for dense pixel-level image annotation task.

Object part segmentation. Motivated by the foundational observation of Fischler and Elschlager [17], pictorial structure framework was introduced which posits that objects can be represented as a collection of their parts in a deformable configuration, several energy-based models have been proposed for object detection [13, 14, 16]. Segmentation models for object parts were also proposed either in probabilistic graphical model [35] or deep neural network [26] frameworks. Several benchmarks have been proposed in the literature in both synthetic and real image formats such as PartImageNet [20], Pascal Part [11], PACO (Part and Attributes in Common Objects) [28], OPD [21], OPDMulti [32], OPD-Synth [21], and PartNet-Mobility [36].

3. SegBuilder Framework

The goal of SegBuilder is to enable fast, web-based segmentation annotations of a collection of images by utilizing SAM-generated image masks augmented with manual editing for cases when SAM fails. The dense pixel-level annotation process pipeline of SegBuilder is shown in Figure 1. Next, we briefly describe SegBuilder's annotation tool, project and image management features, and implementation and deployment considerations.

3.1. SegBuilder Annotation Tool

The primary SegBuilder function is the annotation tool, which displays an image along with a series of SAM-generated masks sorted from largest to smallest 2b. SegBuilder stores each mask as a compressed binary mask with the same dimensions as the image in order to preserve the tight segmentation quality provided by SAM. Each mask has several user interface components to facilitate speedy annotation:

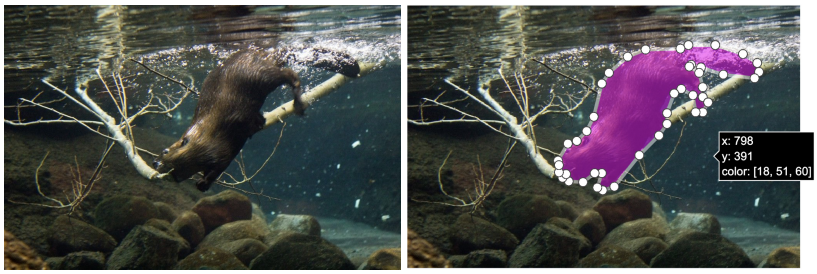
- **Label Drop-Down:** a drop-down menu in which the user may select a label to associate with the mask (defaulting to a project-specific value, usually *unlabeled*)
- **Move-to-Front Button:** a button to move the mask to the front of the mask ordering
- **Edit Button:** a button to edit the mask
- **Delete Button:** a button to delete the mask

The user may also generate custom masks by drawing on the displayed image using a free-form drawing tool (see Fig. 3b), which creates a polygon that can then be converted into a mask using the *Generate Manual Mask* button, which can then be labeled like the SAM masks. The drawing feature allows for zooming, panning, and moving of polygon points. When an existing mask is edited, it is converted into a polygon and may be edited in the same way.

When the user has finished labeling the masks, they click the *Generate Composite Mask Image*, which layers each of the masks in order, rendering each mask in the color assigned to its label. A variant composite image, in which the masks are shown semi-transparently on top of the original image, is also displayed.

3.2. Project and Image Management

SegBuilder allows the user to organize their images into projects (see Fig. 2a), in which each image shares the same



(a) Input image (b) Polygon annotation module.

Figure 3. Polygon-based annotation module in SegBuilder. Polygon based annotation to address missing segments. This tool can be used to annotate part-based segmentation.

set of possible class labels. For each project, there is a *Classes* tab where the user can enter class labels and assign a color to them. These are the values that will be used to populate the annotation tool's mask drow-down menus. The interface includes options for uploading and downloading the color scheme in JSON format so that it can be utilized by other tools in a computer vision pipeline.

The *Files* tab lists the projects image files and includes a file-upload drop zone for adding new images (see Fig. 2c). Files which do not yet have any masks available are shown in white (not ready for labeling), files with masks but no composite mask images are shown in blue (ready for labeling), and files which have had composite mask images generated are green (finished labeling). This tab also includes a bulk download feature in which image files, their mask collections *SegBuilder Image File (.sgbdi)*, and their composite mask images can be selected, compressed into a zip file, and downloaded.

3.3. Implementation and Deployment

SegBuilder uses Dash [1], a Python web framework developed by Plotly [4], that is built on React [5] for the front-end and Flask [2] for the back-end. The source code can be found at `https://github.com/alimoorreza/segbuilder-v1`. It is configured to run in a Docker container, which can be deployed to a compatible web server or run locally. By default, it utilizes Amazon Web Services resources for file storage (S3) and user/project databases (DynamoDB); though it can be configured to emulate these locally for development or single-user mode. There is a separate utility for creating new users, though users can change their own passwords through the user interface.

Because of the compute resources needed for generating SAM masks, we have provided a separate utility that users can use to generate these on their own machines, which creates a *SegBuilder Image File (.sgbdi)* which can be uploaded to a SegBuilder the same way normal image files are uploaded. SegBuilder can be deployed to automatically generate SAM masks using cloud compute resources and can also host a publicly available instance of SegBuilder. To demonstrate its functionalities, we collected a new dataset in an underwater environment and performed its dense pixel-wise annotation, which we discuss next.

4. Dataset

We introduce a novel dataset comprising 635 images with dense pixel-wise annotations, accomplished using our SegBuilder tool. To collect the images in our dataset, we used query terms such as *goldfish, platypus, hippo, beaver*, and *lobster*. This dataset enhances the existing underwater dataset by incorporating greater diversity into the current animal-centric dataset for semantic segmentation, known as the UWS dataset [22]. It includes 21 new underwater animal categories not present in the existing animal-centric dataset for semantic segmentation [22], thereby complementing the UWS dataset. The UWS dataset did not contain any fish categories. In our dataset, we incorporated various species of fish, such as *barracouta, billfish, coho, eel, goldfish, jellyfish, lionfish, puffer, rock beauty, sturgeon*, and *tench*. Additionally, we included new animal categories: *beaver, duck, dugong, hippo, lobster, platypus, nautilus, sea cucumber, sea lion*, and *sea snake*. We also added approximately 50 more images of two animals already present in the UWS dataset: *sea anemone* and *sea urchin*. Following the UWS dataset [22], eight background categories were considered during the annotation process: *coral, rock, water, sand, plant, human, iceberg*, and *reef*. We refer to this new dataset as **UWSv2** ("Underwater Segmentation version 2"). Figure 5 illustrates the distribution of images across various animal categories.

Tool	Average Time (sec)	Average Time (min:sec)
Label Studio	230.34	3:50
SegBuilder	69.02	1:09

Table 1. Comparison of average annotation times

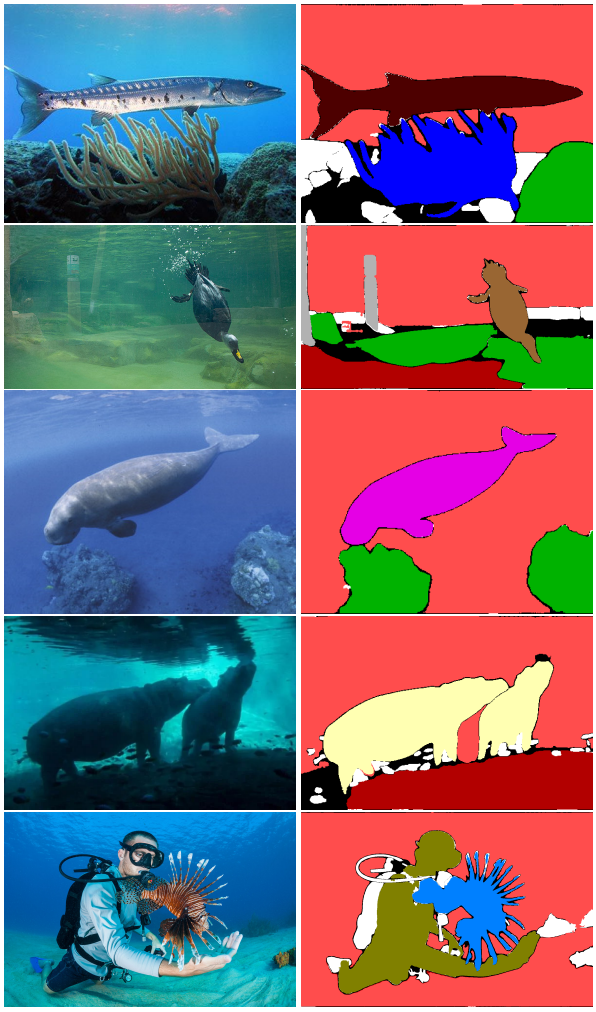


Figure 4. Sample annotated images using SegBuilder in the new dataset. Each row denotes a separate sample image in our dataset containing animals such as *barracouta, duck, dugong, hippo,* and *lion fish*. The first column denotes the RGB images and second column denotes the semantically annotated outputs from SegBuilder.

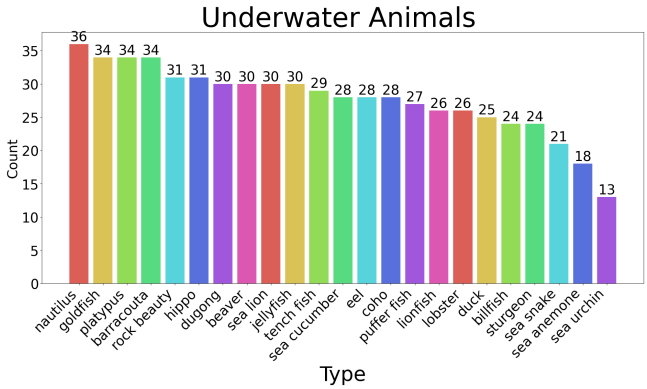


Figure 5. Distribution of animals in our new dataset introduced in this work.

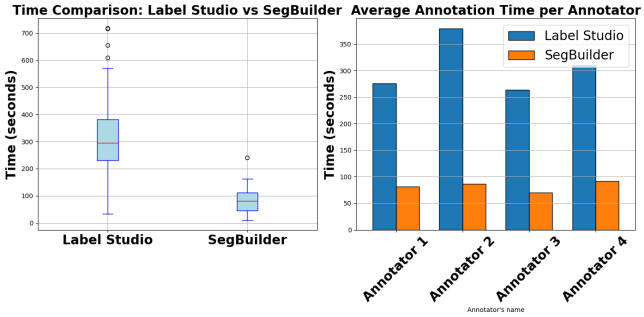


Figure 6. Comparison of annotation times between SegBuilder and the baseline tool Label Studio [31].

5. Experiments

5.1. SegBuilder for Semantic Segmentation

We demonstrate the application of SegBuilder for semantic segmentation tasks using our newly introduced **UWSv2** dataset, although it is applicable to any segmentation task. First, we present the annotation efficiency achieved by our semi-automatic method in comparison to manual annotation tools such as Label Studio [31]. Subsequently, we demonstrate the effectiveness of the annotated images in training a semantic segmentation model, evaluated both qualitatively and quantitatively.

SegBuilder vs. LabelStudio [31] Annotation Comparison. We conducted an efficiency comparison of annotation for dense pixel-wise tasks against the existing tool Label Studio [31]. A subset of 23 images representing all animal categories in our dataset was randomly selected and annotated by **4** in-house annotators. The same set of images was annotated first using our tool SegBuilder and then reannotated using Label Studio for comparison. Annotators recorded their annotation times for both tools. Figure 6 provides a comparison of the average annotation times for the four individual annotators. On average, annotating an image with Label Studio takes 3 minutes and 50 seconds, while annotation with SegBuilder is completed in 1 minute and 9 seconds. Thus, SegBuilder achieved **3.34x faster** annotation times compared to Label Studio. Detailed statistics are provided in Table 1. It is important to underscore that SegBuilder operates on the segments generated by SAM. With the aid of additional editing tools offered by SegBuilder, its output can only improve in quality, as evidenced in Figure 11 illustrates the part annotation experiment, which demonstrates consistent performance without deterioration. Therefore, we adopted SegBuilder's refined annotations as our final ground truth. In our next experiment, we demonstrated that the ground truth produced by SegBuilder can be used to train a semantic segmentation model.

Model	Class Names										
	Sea Anemone	**Sea Urchin**	**Coral**	**Rock**	**Water**	**Sand**	**Plant**	**Human**	**Reef**	**Other**	**Beaver**
HRNetV2 [34]	0.7456	0.2799	0.0000	0.3085	0.7933	0.4409	0.3277	0.4157	0.4502	0.0794	0.7796
	Duck	**Dugong**	**Hippo**	**Lobster**	**Platypus**	**Nautilus**	**Sea Cucumber**	**Sea Lion**	**Sea Snake**	**Barracouta**	**Billfish**
HRNetV2 [34]	0.6061	0.8479	0.7757	0.8587	0.7377	0.9222	0.5058	0.6634	0.7358	0.7945	0.7692
	Coho	**Eel**	**Goldfish**	**Jellyfish**	**Lionfish**	**Puffer**	**Rock Beauty**	**Sturgeon**	**Tench**		**mIoU**
HRNetV2 [34]	0.8233	0.5877	0.7133	0.8382	0.9234	0.6311	0.5132	0.7360	0.7847		**0.6254**

Table 2. Semantic segmentation results on the newly introduced and annotated dataset utilizing SegBuilder.

Semantic segmentation model. For underwater semantic segmentation with diverse animal categories, HRNetV2 [34] has been reported to achieve the best performance, as shown in the work of Imran et al. [22]. Accordingly, we trained HRNetV2 on our dataset. We randomly split the dataset into a training set of 500 images and a test set of 135 images. Standard data augmentation techniques, such as random flip, shift, and rotation, were applied during training. The method was evaluated using the standard mean IoU metric for semantic segmentation. We report the individual IoU for each class in Table 2. HRNetV2 [34] model was trained for 360 epochs and the best mIoU of **62.54%** was achieved in 22^{nd} epoch. Thus, we established baseline performance using HRNetv2 on the newly introduced dataset **UWSv2**. The model was trained on two Nvidia Titan Xp GPUs.

5.2. SegBuilder for Scene Segmentation under Camouflage and Shadows

While SAM and its derivatives have shown promising results in general image segmentation tasks, they face challenges in certain complex scenarios such as: camouflage and shadows due to altered visual features and ambiguous boundaries [10]. To address these limitations, SegBuilder allows for manual readjustment of annotations, significantly improving the accuracy and efficiency of the segmentation process. SAM's segmentation performance exhibits notable variability across different image conditions and object types. This inconsistency stems from limitations in training data, challenges in generalization, dependency on input prompts, and inherent model constraints. SAM's performance can be particularly unreliable in edge cases that fall outside its learned parameters, such as images with camouflaged objects or complex shadow patterns. These variations in effectiveness highlight the need for flexible annotation tools that can leverage SAM's strengths while providing intuitive interfaces for manual correction in challenging cases.

Experimental setup. We conducted two experiments to showcase the efficacy of our tool in handling images with camouflage and shadows. Due to the unavailability of data from the work of [10], we utilized the Adaptive Camouflaged Dataset (ACD1K) [19] for our camouflage experiments and the SBU Shadow dataset [6] for our shadow experiments. We randomly selected 24 images from these two datasets with the sample distribution shown in Figure 9.

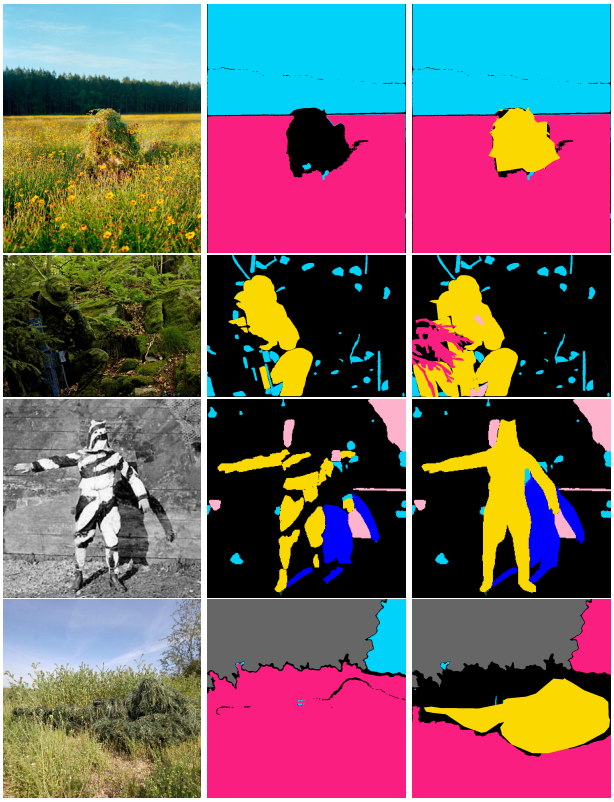


Figure 7. Annotation comparisons between SegBuilder and SAM on images featuring camouflaged objects. The figure displays several sample images arranged from top to bottom. Each row shows an image (left column), its corresponding SAM segmentation output (middle column), and the segmentation obtained using SegBuilder (right column). The segmentation results clearly demonstrate SAM's inability to accurately segment camouflaged objects.

Experimental observations. The capability to readjust annotations in our tool proved essential in handling the intricate boundaries created by shadows. This feature enabled annotators to promptly correct errors made by the automated system, resulting in both time savings and enhanced accuracy. Figure 8 showcases sample images with shadows

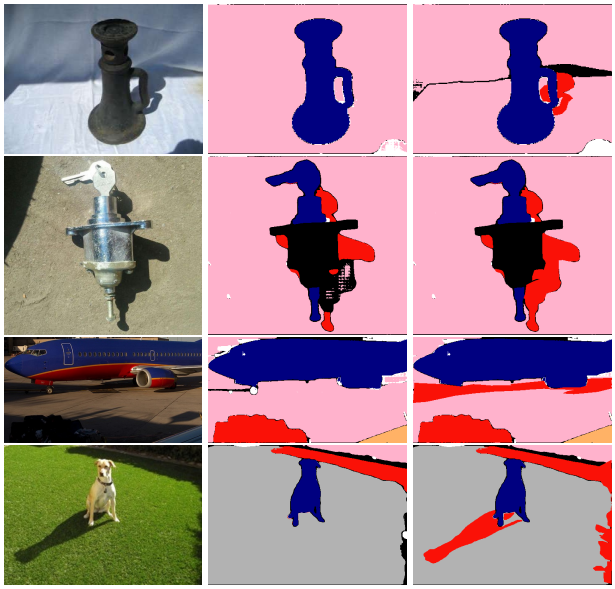


Figure 8. Annotation comparisons between SegBuilder and SAM on images featuring objects with shadows. The figure displays several sample images arranged from top to bottom. Each row shows an image (left column), its corresponding SAM segmentation output (middle column), and the segmentation obtained using SegBuilder (right column). The segmentation results clearly demonstrate SAM's inability to accurately segment shadows.

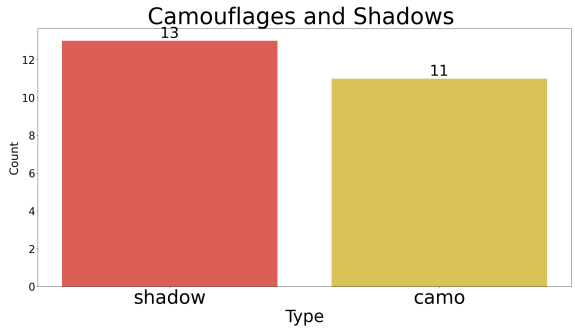


Figure 9. The distribution of images with camouflages and shadows used in our experiments.

where SAM fails to produce accurate segmentation, contrasting with SegBuilder's ability to correct annotations effectively. Similar observations were made in our camouflage experiments, depicted in Figure 7, where SAM struggled to achieve accurate segmentation. In contrast, SegBuilder allowed for adjustments to annotations, facilitating correct segmentation outcomes.

5.3. SegBuilder for Part-based Segmentation

Object part segmentation has useful applications in fine-grained object classification, pose estimation, and object re-identification.

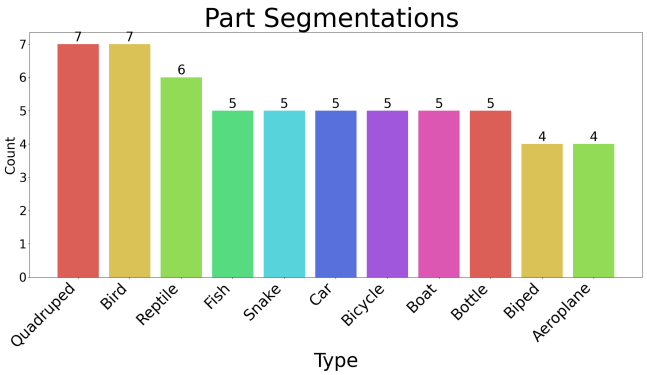


Figure 10. The distribution of images from 11 super-categories taken from the PartImageNet dataset [20] used in our experiments.

Although SAM [10] has demonstrated remarkable capabilities in general object segmentation, it exhibits significant limitations in part segmentation. SAM often struggles to accurately delineate individual parts within objects due to reasons such as lack of part-specific training, ambiguous boundaries, and high variability in part appearances. These limitations underscore the necessity for a specialized tool dedicated to part-based annotation.

To address SAM's shortcomings in part segmentation, our SegBuilder framework offers several advantages. With its polygon-based approach, SegBuilder provides users precise control over part boundaries while maintaining efficiency. To validate the effectiveness of SegBuilder compared to SAM, we conducted experiments using the PartImageNet [20] dataset. This dataset was selected for its diverse range of objects and well-defined part annotations, establishing a robust benchmark for part segmentation tasks.

Experimental setup. PartImageNet provides fine-grained annotations for 158 objects categories under 11 super-categories, i.e., *fish, quadruped, reptile, snake, bird, aeroplane, car, biped, bicycle, boat*, and *bottle*. We selected a subset of 58 images from PartImageNet [20], covering all the 11 super-categories. Figure 10 shows the distribution of images from different super-categories used in our experiment. This dataset specifies a variable number of part annotations based on the super-categories. For instance, the *aeroplane* category comprises five parts: *Head, Body, Wing, Engine,* and *Tail*. In contrast, the *boat* category includes two parts: *Body* and *Sail*. We compared SAM-only annotation, where SAM was applied to selected images without additional prompts or human intervention, with SegBuilder-assisted annotation, which utilized our SegBuilder tool for the same images, allowing human guidance in the process.

Experimental observations. Our experiments yielded the following key findings. SegBuilder consistently produced more accurate part segmentations compared to SAM alone. SAM often missed smaller or less distinct parts,

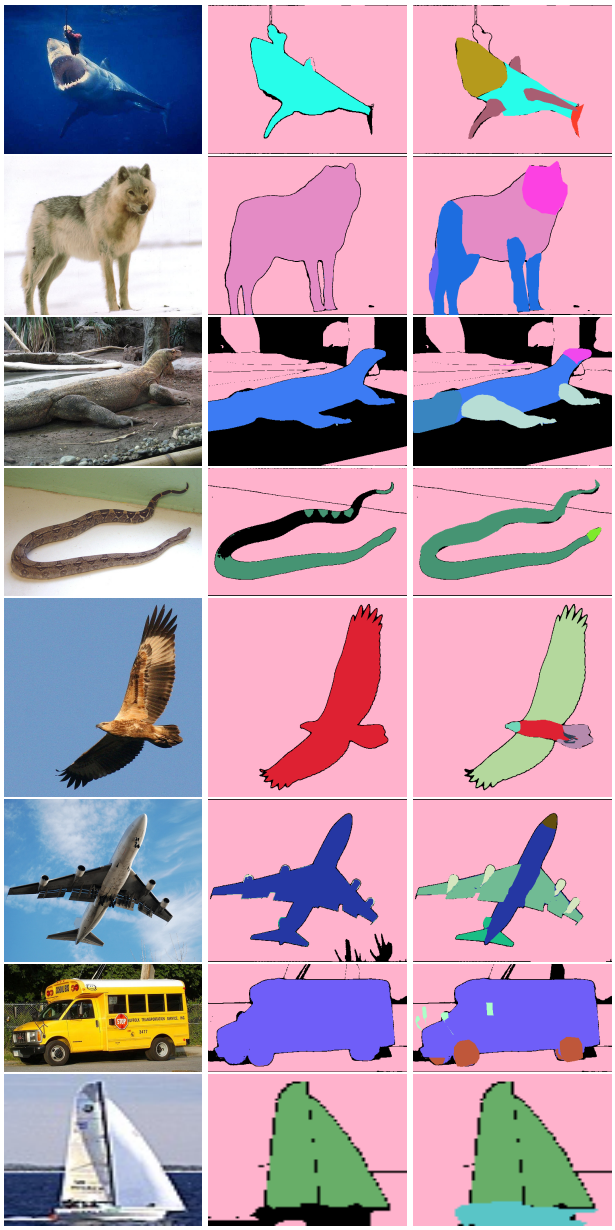


Figure 11. Part segmentation on sample images from the PartImageNet dataset. From top to bottom, we present sample images from each of the super-categories: *fish, quadruped, reptile, snake, bird, aeroplane, car,* and *boat*. Each row displays an image (left column), its corresponding SAM segmentation output (middle column), and the segmentation obtained using SegBuilder (right column). The segmentation results clearly show that SAM fails to accurately segment essential parts. The additional tool we introduced in SegBuilder allows for reannotation of regions corresponding to different parts.

while SegBuilder allowed annotators to capture all relevant parts, resulting in an increased part detection rate. Figure 11 presents qualitative results on representative images for the supercategories *fish, quadruped, reptile, snake, bird, aeroplane, car*, and *boat*. Samples for the remaining three supercategories are provided in the supplementary material. The visual comparison, particularly in the first three rows, demonstrates that SAM (center column) fails to capture object parts in the *fish, quadruped*, and *reptile* supercategories. The missing parts, such as *Head, Body, Fin, Tail* for *fish* and *Head, Body, Foot, Tail* for both *quadruped* and *reptile*, have been re-annotated using the SegBuilder (third column) with the assistance of the **Edit** tool. The remaining four rows in Figure 11 illustrate a similar recovery of part annotations using SegBuilder in instances where SAM fails. These results demonstrate that, while SAM is a powerful tool for general segmentation, it falls short in the specialized task of part segmentation. SegBuilder, with its polygon-based approach and human guidance, offers a more effective solution for creating high-quality part annotations.

6. Conclusion

In this paper, we introduce SegBuilder, a semi-automatic pixel-level annotation tool. By leveraging the remarkable zero-shot capabilities of the vision foundation model SAM, SegBuilder creates useful segments that annotators can further tag using a drop-down list. We demonstrated the effectiveness of SegBuilder for general semantic segmentation tasks on a novel dataset introduced in this paper. Additionally, we showed that SegBuilder provides complementary benefits in situations where SAM fails to obtain reasonable segments. SegBuilder includes a free-form drawing tool that allows users to create the correct segments missed by SAM. This capability is particularly useful in scenes with shadows or camouflaged objects and for part-based segmentation tasks. We have publicly released our tool for use by the research community.

References

[1] Dash python user guide. https://dash.plotly.com. 4

[2] Flask: web development one drop at a time. https://flask.palletsprojects.com. 4

[3] Labelme with segment anything. https://github.com/originlake/labelme-with-segment-anything. 2

[4] Plotly open source graphing library for python. https://plotly.com/python. 4

[5] React: The library for web and native user interfaces. https://react.dev/. 4

[6] Shadow dataset by stony brook university (SBU Shadow). 6

[7] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012. 2

[8] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein,

Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021. 3

[9] Jun Cen, Yizheng Wu, Kewei Wang, Xingyi Li, Jingkang Yang, Yixuan Pei, Lingdong Kong, Ziwei Liu, and Qifeng Chen. SAD: Segment Any RGBD, 2023. 3

[10] Tianrun Chen, Lanyun Zhu, Chaotao Deng, Runlong Cao, Yan Wang, Shangzhan Zhang, Zejian Li, Lingyun Sun, Ying Zang, and Papa Mao. Sam-adapter: Adapting segment anything in underperformed scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3367–3375, 2023. 1, 2, 6, 7

[11] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1971–1978, 2014. 3

[12] Microsoft COCO. Coco annotation tool. https://github.com/jsbroks/coco-annotator, 2015. 2

[13] David Crandall, Pedro Felzenszwalb, and Daniel Huttenlocher. Spatial priors for part-based recognition using statistical models. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 10–17. IEEE, 2005. 3

[14] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. Ieee, 2008. 3

[15] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59:167–181, 2004. 2

[16] Pedro F Felzenszwalb and Daniel P Huttenlocher. Pictorial structures for object recognition. *International journal of computer vision*, 61:55–79, 2005. 3

[17] Martin A Fischler and Robert A Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on computers*, 100(1):67–92, 1973. 3

[18] Andreas Geiger, Martin Lauer, Christian Wojek, Christoph Stiller, and Raquel Urtasun. 3d traffic scene understanding from movable platforms. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2014. 2

[19] Ali Haider. Adaptive camouflaged dataset (acd1k), 2023. 6

[20] Ju He, Shuo Yang, Shaokang Yang, Adam Kortylewski, Xiaoding Yuan, Jie-Neng Chen, Shuai Liu, Cheng Yang, Qihang Yu, and Alan Yuille. PartImageNet: A large, high-quality dataset of parts. In *European Conference on Computer Vision*, pages 128–145. Springer, 2022. 2, 3, 7

[21] Hanxiao Jiang, Yongsen Mao, Manolis Savva, and Angel X Chang. OPD: Single-view 3d openable part detection. In *European Conference on Computer Vision*, pages 410–426. Springer, 2022. 3

[22] Imran Kabir, Shubham Shaurya, Vijayalaxmi Maigur, Nikhil Thakurdesai, Mahesh Latnekar, Mayank Raunak, David Crandall, and Md Alimoor Reza. Few-shot segmentation and semantic segmentation for underwater imagery. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11451–11457. IEEE, 2023. 2, 4, 6

[23] 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. 1, 2, 3

[24] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023. 3

[25] Jun Ma, Yuting He, Feifei Li, Lin Han, Chenyu You, and Bo Wang. Segment anything in medical images. *Nature Communications*, 15:1–9, 2024. 3

[26] Shujon Naha, Qingyang Xiao, Prianka Banik, Md Alimoor Reza, and David J Crandall. Part segmentation of unseen objects using keypoint guidance. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1742–1750, 2021. 2, 3

[27] OpenAI. Gpt-4 technical report, 2023. 3

[28] Vignesh Ramanathan, Anmol Kalia, Vladan Petrovic, Yi Wen, Baixue Zheng, Baishan Guo, Rui Wang, Aaron Marquez, Rama Kovvuri, Abhishek Kadian, et al. Paco: Parts and attributes of common objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7141–7151, 2023. 3

[29] 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

[30] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000. 2

[31] Label Studio. Label studio. https://labelstud.io, 2021. 1, 2, 5

[32] Xiaohao Sun, Hanxiao Jiang, Manolis Savva, and Angel Xuan Chang. OPDMulti: Openable Part Detection for Multiple Objects. *arXiv preprint arXiv:2303.14087*, 2023. 3

[33] Antonio Torralba, Bryan C Russell, and Jenny Yuen. Labelme: Online image annotation and applications. *Proceedings of the IEEE*, 98(8):1467–1484, 2010. 1, 2

[34] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. *CoRR*, abs/1908.07919, 2019. 6

[35] Jianyu Wang and Alan L Yuille. Semantic part segmentation using compositional model combining shape and appearance. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1788–1797, 2015. 2, 3

[36] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11097–11107, 2020. 2, 3

[37] Jinyu Yang, Mingqi Gao, Zhe Li, Shang Gao, Fangjing Wang, and Feng Zheng. Track anything: Segment anything meets videos, 2023. 3

[38] Sherry Yang, Ofir Nachum, Yilun Du, Jason Wei, Pieter Abbeel, and Dale Schuurmans. Foundation models for decision making: Problems, methods, and opportunities. *arXiv preprint arXiv:2303.04129*, 2023. 3

[39] Tao Yu, Runseng Feng, Ruoyu Feng, Jinming Liu, Xin Jin, Wenjun Zeng, and Zhibo Chen. Inpaint anything: Segment anything meets image inpainting. *arXiv preprint arXiv:2304.06790*, 2023. 3

[40] Youcai Zhang, Xinyu Huang, Jinyu Ma, Zhaoyang Li, Zhaochuan Luo, Yanchun Xie, Yuzhuo Qin, Tong Luo, Yaqian Li, Shilong Liu, et al. Recognize anything: A strong image tagging model. *arXiv preprint arXiv:2306.03514*, 2023. 3