ARL-TR-8514 • Sep 2018

US Army Research Laboratory

SuperLabel: A Superpixel Labeling Interface for Semantic Image Annotation

by Maggie Wigness

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-8514 ● SEP 2018

US Army Research Laboratory

SuperLabel: A Superpixel Labeling Interface for Semantic Image Annotation

by Maggie Wigness
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
September 2018	Technical Report	January–August 2018

4. TITLE AND SUBTITLE
SuperLabel: A Superpixel Labeling Interface for Semantic Image Annotation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Maggie Wigness

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
US Army Research Laboratory
ATTN: RDRL-CII-A
Adelphi, MD 20783-1138

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-8514

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
primary author's email: <maggie.b.wigness.civ@mail.mil>.

14. ABSTRACT
Manual image annotation is a tedious, yet necessary, task to collect labels that can be used for supervised machine learning algorithms. Most existing interfaces used to assign labels to pixels in an image require the user to hand outline each distinct object or visual concept in the image. This task can be quite time consuming, especially for objects of non-rectangular shape (e.g., trees or people). This report introduces a new annotation interface called SuperLabel that makes use of superpixel segmentation to automatically define object regions so the user simply has to assign a label to these regions instead of also manually defining them. The report provides details on how to set up and launch SuperLabel, and describes the labeling functionality provided by the system to label sets of images.

15. SUBJECT TERMS
image annotation, image labeling graphical user interface, superpixel labeling, image ground truth collection

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Maggie B Wigness
Unclassified	Unclassified	Unclassified	UU	26	19b. TELEPHONE NUMBER (Include area code) 301-394-3927

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

1. Introduction

The computer vision community has made great advancements with the use of deep learning for tasks such as object recognition,[1,2] detection,[3,4] and semantic segmentation.[5,6] These state-of-the-art algorithms have far surpassed previous techniques for these tasks, but the cost of learning is reflected in the large sets of labeled training examples required to achieve this performance. For example, state-of-the-art object classifiers are usually trained using the ImageNet dataset,[7] which contains over one million images.

The total human hours spent collecting labels for the ImageNet dataset was roughly 19 years,[8] but this work of course was being done in parallel via crowdsourcing on Amazon Mechanical Turk. The labeling task for this particular dataset is relatively simple. The human is shown an image and provides a single object label that describes the image contents. Example images and labels from the ImageNet dataset can be seen in Fig. 1. Note that a human is not asked to provide specific details about the location of the object. The assumption of many image classification benchmark datasets is that the object takes up most of the image and is roughly centered.

Fig. 1 Example images from the ImageNet dataset and labels that would be assigned by a human annotator when hand labeling each of the images

As learning tasks become more complex (e.g., semantic segmentation), the workload to collect ground truth annotations for the training data significantly increases. For semantic segmentation, the learning algorithm assigns a label to each pixel in an image. Figure 2 shows an image with pixel-wise label annotations, where each pixel is visually shown as a color that relates to its specific label assignment. This means an algorithm is localizing and recognizing visual concepts, where visual concepts can refer to objects, terrains, and any other distinct area in an environment (e.g., sky). There are several labeling interfaces and techniques that have been used to help with this annotation task. We review some of this technology in the next

section, but in short, these existing frameworks provide an interface for manual annotation (i.e., full human oversight), where a human hand outlines each distinct visual concept region in the image to complete the labeling.

Fig. 2 Example image that has been assigned pixel-wise labels. Each pixel is overlaid with a color that maps to the semantic label associated with that pixel. For example, red indicates building and blue indicates sky.

Even with advances in domain adaptation, learners usually experience a performance degrade when tested on data from a different domain than its training data.[9] This ultimately means to achieve top performance for a domain, data from that domain should be collected, labeled, and used to help train the learner. The relevance of this becomes critical for applications in environments that are quite different than existing benchmark datasets used by the computer vision community. This is particularly true for Army-relevant environments, where the scene may be highly unstructured, undergoing rapid change, consist of objects not currently recognized by existing technology, and many other differing factors. Further, making use of public crowdsourcing can be difficult when data cannot be publicly released, meaning labeling is done in house.

This technical report addresses the need for an interface to collect pixel-wise label information for large visual datasets in a fully supervised manner, but provides ways to reduce the overall effort of a human during this process. Specifically, this technical report introduces *SuperLabel*: a labeling interface that does not require any hand outlining of regions in the image, but instead uses unsupervised segmentation algorithms to automatically define superpixel regions. The unsupervised segmentation is dynamic, giving the user oversight and control of the defined boundaries to collect accurate labels. This is particularly important as the US Army Research Labora-

tory (ARL) makes an effort to generate datasets that are more military-relevant than existing benchmark datasets currently used in the academic community.

2. Other Labeling Techniques

There are several other labeling techniques and interfaces available for a user to collect pixel-wise label information. These techniques are briefly described. Note, there are companies that provide label annotation services that have their own in-house, non-publicly available annotation software. Outsourcing label annotation is beyond the scope of this report, however, and will not be discussed.

2.1 Polygon Labeling

Polygon labeling is commonly used so the user can define a precise region of pixels that should be assigned the same label. In other words, the user must draw a polygon around each object or visual concept in the image and then assign a label to each of the drawn regions. This type of labeling gives the user full control over defined regions, but can require a lot of effort to mark these regions.

LabelMe[10] was one of the first annotation tools that allowed a user to outline and assign labels to polygons in an image. To select a polygon, the user makes line segments by clicking around the boundary of a visual concept in the image. For objects in the image that have a relatively simple polygon shape (e.g., a rectangular box), polygon construction requires only a few mouse clicks. However, many objects in real-world scenes have shapes that are much more difficult to generate polygons for. In many cases this is because the objects are non-rigid (e.g., people). Other times the objects are simply composed of shapes that do not have many straight line edges (e.g., vehicles and trees).

There are several other annotation interfaces that include polygon labeling. Matlab includes the Image Labeler app,[11] and Sloth[12] is a labeling framework that can be configured to individual researcher formats, but both include polygon labeling.

2.2 Bounding Box Labeling

Bounding box labeling allows a user to define a rectangular region of interest and assign a label to all pixels inside this bounding box. Assigning labels to bounding boxes is less work than polygon labeling since a user only needs to define two points to generate a box. However, this label assignment does not provide pixel-wise labeling precision as most objects are not perfectly rectangular. This results in background pixels being assigned the label as well. Often this type of labeling is more relevant to object detection tasks, where detectors output bounding boxes around objects of interest.

Matlab's Image Labeler and Sloth also include bounding box labeling functionality. Additionally, interfaces like the Simple Image Annotation tool[13] and Labelbox[14] are specifically designed for this type of fast label assignment.

2.3 Synthetic Data

On the opposite spectrum of label annotation is the research effort that is put into the generation and use of synthetic data for learning. For example, the Synthia[15] dataset is a synthetic environment developed using the Unity development platform. The generation of this dataset eliminates the need for human labeling effort since labels are automatically assigned to pixels when an environment is created.

Although synthetic data generation is advancing quickly, and the visual output looks remarkably real, there is still a domain gap that is being addressed in the synthetic to real domain adaptation research area.[16,17] Again, this issue can be heightened for Army-relevant domains, where synthetic data is often designed to resemble that of existing benchmark datasets from city environments that are highly structured.

3. Software and Labeling Setup

This section serves as a quick start guide to using the SuperLabel software. The software dependencies, location of data input/output, how to specify labels, and how to launch the software are all discussed. Throughout this section, the location of the SuperLabel software package is referenced as *~/SuperLabel*.

3.1 Software Dependencies

SuperLabel is written in Python and has been tested on a platform running Ubuntu 16.04 and Python 2.7. The following is a list of Python library dependencies used for graphical user interface (GUI) generation, image viewing, and label annotation overlay:

- cv2
- numpy
- PyQt5
- skimage

3.2 Data Input and Output

Each dataset to label needs a directory to store the image input, the label output, and a text file that defines the semantic labels a user can assign to the data. This directory needs to be created in *~/SuperLabel*. An example dataset is provided with the software and is located at *~/SuperLabel/example_data*. This is used as a running example throughout this report.

3.2.1 Input Images

Images to be labeled need to be in a directory called *images*, which can be found in the dataset directory. When the GUI is launched, image files from this directory are sorted and displayed in order by their string file name. For data streams, frames are easily viewed in order when image files are given sequential integer names. For example, the images in *~/SuperLabel/example_data/images* have the following names:

```
00001.png
00002.png
.
.
.
00100.png
```

3.2.2 Output Labels

Label files are generated or updated as a user interacts with the labeling interface. The label files are saved in the dataset *label_files* directory (e.g., *~/SuperLabel/example_data/label_files*). Each image has an associated label file with the same file name and a numpy extension. For example, image 00001.png will have the associated label file 00001.npy.

The label file is represented as 2-D numpy array with the same shape as the image. Each pixel in the 2-D array has an integer value, where -1 indicates the pixel has not been assigned a label, and labels $0, 1, \ldots, N-1$ indicate which of the N possible labels the user has assigned to the pixel.

3.3 Loading Labels

A list of labels specific to the dataset needs to be stored in the dataset *colormap.txt* file. This file defines the label and associated visual color of that label in the GUI. Each line of the file represents a label and should be formatted as "index, label, R, G, B".

The *index* should be a sequential integer beginning at 0 and ending at $N - 1$ for N labels in the file. The *label* is the natural language label the user can select during labeling, and *R*, *G*, *B* are the RGB color values associated with the label. The *~/SuperLabel/example_data/colormap.txt* file contains 10 labels:

```
0 building 255 0 0
1 road 255 128 0
2 cone 127 0 255
3 grass 0 102 0
4 tree 0 255 0
5 pole 0 255 255
6 water 0 128 255
7 sky 0 0 255
8 vehicle 255 255 0
9 object 255 0 127
```

3.4 Launching the GUI

The GUI must be launched from the *~/SuperLabel* directory. The following is the general form to launch the labeling interface:

```
python src/gui_main.py dataset [-skip n] [-start i]
```

The parameters to this interface launch are as follows:

- **dataset**: dataset directory that contains the images directory, label colormap file, and where the label output will be stored.
- **-skip**: defines that every n^{th} image will be displayed; default value is 1 so every image will be shown as the user hits the *Next* button.
- **-start**: defines the i^{th} image that will be shown first to the user; default value is 1 so it starts at the first image in the sorted order.

Using the example data, the GUI can be launched as follows:

```
python src/gui_main.py example_data -skip 5 -start 1
```

This will begin by displaying the first image in the data stream, and display every fifth image each time the user hits the *Next* button.

Once launched, the GUI can be resized to make the image display larger. Launching the GUI automatically puts the user in a *view only* mode. The next section describes the specific details of the labeling functionality and how a user can use this functionality to assign labels to the pixels in the dataset images.

4. GUI Basic Functionality

The labeling GUI can be seen in Fig. 3. Different areas on the GUI provide the user with various functionality. Most of this functionality is related to label assignment and is discussed in Section 5. The rest of the basic functionality is outlined in the figure and discussed in the rest of this section.

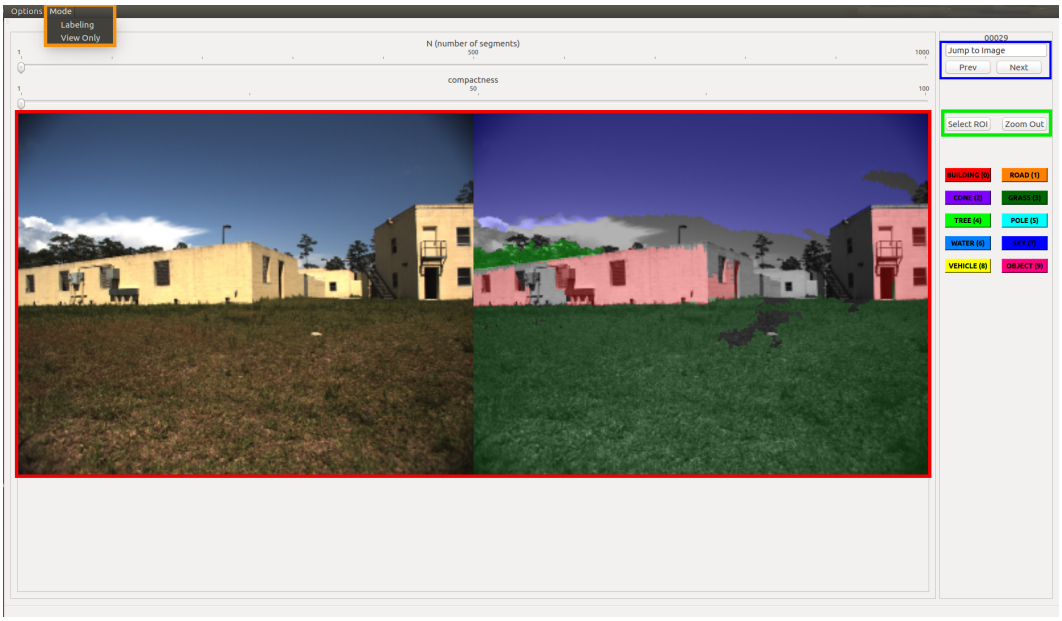

Fig. 3 Basic layout of the label annotation interface

4.1 Image Display

Two images are always displayed in the GUI. The area outlined in red in Fig. 3 shows the current image being viewed, and to the right is a visualization of its corresponding label annotation. In the label annotation image, all unlabeled pixels (i.e., label of -1) are displayed as their grayscale value, and other pixels are shown overlaid with their associated label color. The specific colors are determined by the content of colormap.txt discussed in Section 3.3, and the label buttons to the right of the image display also encode these colors.

4.2 Moving Between Images

The user can view different images in the dataset using the *Prev* and *Next* buttons or by typing an image file name in the top right corner of the GUI (outlined in blue in Fig. 3). The *Prev* and *Next* buttons move the user backward and forward, respectively, n images, where n is defined as the skip count parameter passed in when launching the GUI (see Section 3.4). The data stream is treated as a loop when performing this action near the beginning and end of the set of ordered images.

Entering an image file name directly into the line edit allows the user to make larger or irregular (i.e., not increments of n) moves throughout the dataset instead of via

button clicks. The name of the image file currently being displayed can always be found above these buttons and the line edit.

4.3 Image Zoom

A user can select a region of interest (ROI) within the image to crop and zoom. This functionality can be completed with the buttons outlined in green as seen in Fig. 3. The procedure to zoom in on an ROI is as follows:

1. Click the *Select ROI* button.
2. Click the top left corner of the ROI and drag to the bottom right corner before releasing.
3. Inspect the bounding box generated.
4. Click the *Zoom In* button to accept the bounding box and zoom in.
5. Click the *Reset* button to reject the bounding box and re-select.

When zoomed in on an ROI, the user may click the *Zoom Out* button to return to the original image view. Further, the user may only zoom in on an ROI from the original view of the image, not a zoomed-in version of it.

4.4 Modes of Operation

There are two modes of operation that can be toggled in the *Mode* menu at the top of the GUI, *labeling* and *view only*. This part of the GUI is outlined in orange in Fig. 3. The default mode after launching the GUI is view only. The functionality described thus far is available in both modes. Additional view-only mode functionality is described in the rest of this section, and Section 5 describes the labeling mode functionality.

The view-only mode is established to let the user visually inspect the current label annotation for a particular set of images without the possibility of overwriting the current labeled state. All label assignment functionality is disabled while in view-only mode.

While in view-only mode, the user has the ability to click a pixel in the image display area (either the raw image or the label overlay image) and see its current

label assignment. The label is displayed in the GUI below the *Prev* and *Next* buttons. Figure 4 shows the label output outlined in blue when the user clicks an area of the grass (indicated by the mouse pointer outlined in red). Being able to see the specific label name for a pixel can be beneficial as the number of labels being used on a dataset increases and color annotations become more difficult to discern.

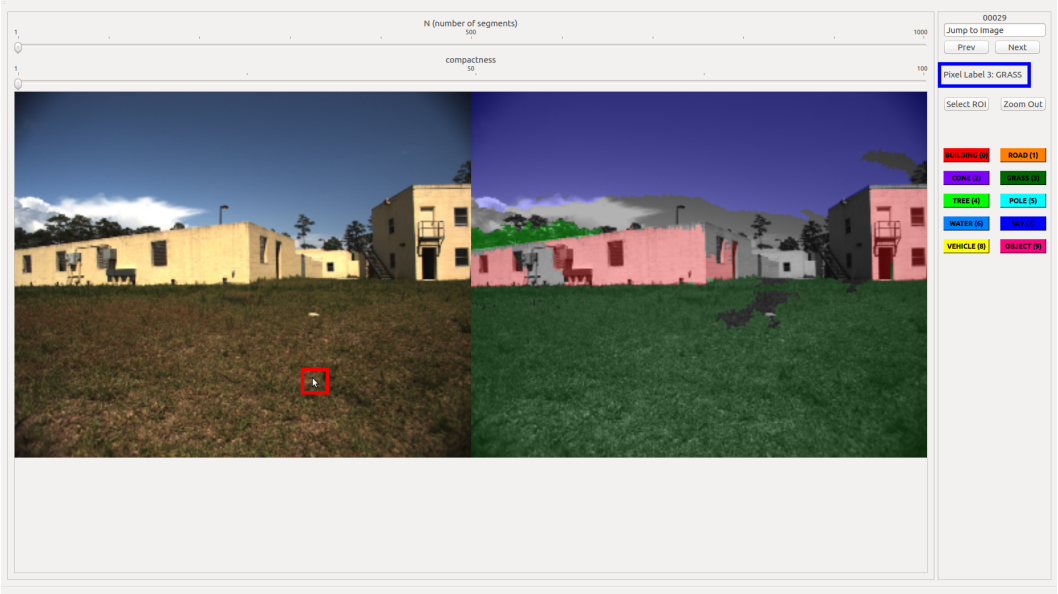

Fig. 4 Example label output displayed in GUI (outlined in blue) when a user clicks a pixel (indicated by mouse outlined in red) in view-only mode

5. Labeling Mode Functionality

The labeling mode allows the user to interactively assign labels to pixels in images from the dataset. One of the primary goals is to provide an interface that does not require a user to generate a polygon outline of each class instance in the image. Instead, the tool leverages unsupervised segmentation to automatically find the regions and boundaries of classes, and the user then simply has to specify a label for the automatically generated outline. Superpixel segmentation is used to facilitate this approach.

5.1 Superpixel Segmentation

A superpixel is composed of multiple connected pixels that share similar color or intensity levels in an image. Superpixel segmentation algorithms[18–21] are commonly used to reduce the number of primitives that must be processed. In the case of assigning labels to image pixels, the generation of superpixels allows a user to assign

a label to each superpixel instead of each pixel. Further, by breaking the image into a set of superpixels, region outlines or boundaries are automatically determined and do not have to be precisely annotated by the user. When in labeling mode, instead of displaying the raw image only, the superpixel segmentation output is shown in the image display area as seen in Fig. 5. Each region enclosed by the red boundary markers represents a superpixel.

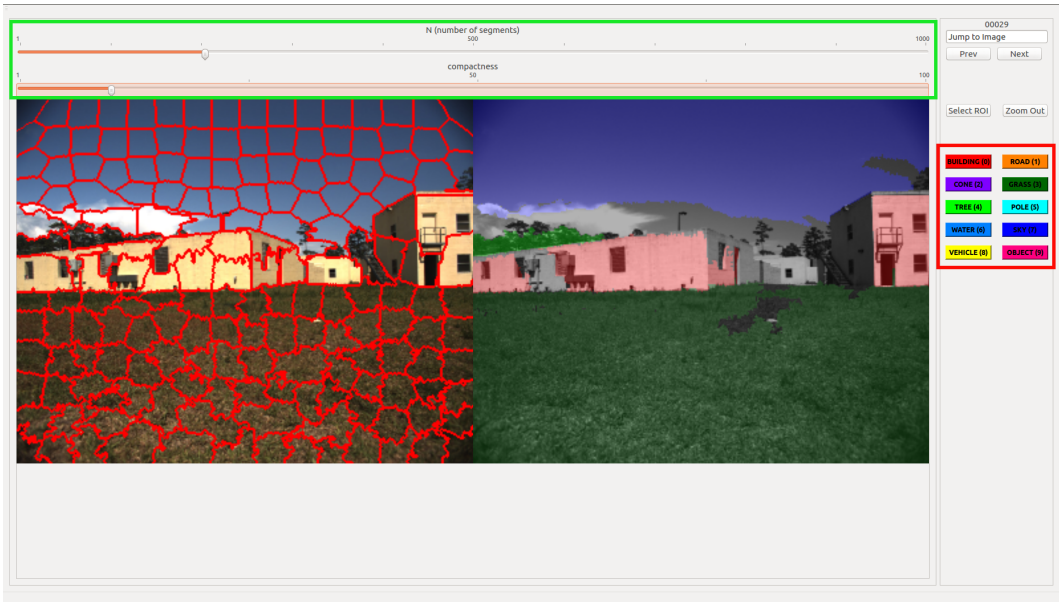

Fig. 5 Example GUI when a user is in labeling mode. The superpixel segmentation can be seen in the image display area. The green outlined area contains slider bars to dynamically change the parameters to the segmentation algorithm, and the red outlined area denotes the location of label buttons that can be used to assign a label to superpixels.

This annotation tool uses the Simple Linear Iterative Clustering (SLIC) segmentation algorithm[19] to have a user assign labels to superpixels. SLIC represents a pixel in an image with a 5-D feature space, $p = [l, a, b, x, y]$. The first three dimensions are values extracted using the CIELAB colorspace, where l is for lightness, a is the green-red color component, and b is the blue-yellow color component. The last two dimensions, x and y, represent the location of p in the image. Image pixels are then clustered using this feature representation.

5.2 Parameters

Two parameters can be defined using the interface and passed to SLIC to produce the superpixel segmentation:

- N — the number of the superpixels to output.
- compactness — a value to balance color and space proximity during superpixel construction.

These parameters can be entered by the user using the two sliding bars at the top of the GUI, which are outlined in green in Fig. 5.

The value N is the approximate number of superpixels that SLIC with output. The top row of Fig. 6 shows SLIC segmentation output on the same image when varying the value of N. A larger value of N produces superpixels for a user to assign labels to, but often produces superpixels that more accurately respect true region boundaries (i.e., each superpixel represents exactly one label). With smaller values (e.g., the segmentation when $N = 50$), there are some superpixels that consist of pixels from multiple objects/terrain/concepts in the scene. This can be seen in the $N = 50$ top right corner superpixel. Most pixels are sky but some represent the pole.

The bottom row of Fig. 6 shows SLIC segmentation output when varying the compactness parameter. Larger compactness values weight space proximity higher, yielding superpixels shapes that look more square as seen in the compactness $= 100$ output on the far right. A smaller compactness value weights color more heavily and can result in superpixels with more irregular shapes.

5.3 Dynamic Segmentation

The sliding parameter bars for N and compactness allow SLIC segmentation to be run dynamically when labeling an image. The range of values available on the slider bar are as follows:

- N = [1,1000]
- compactness = [1,100]

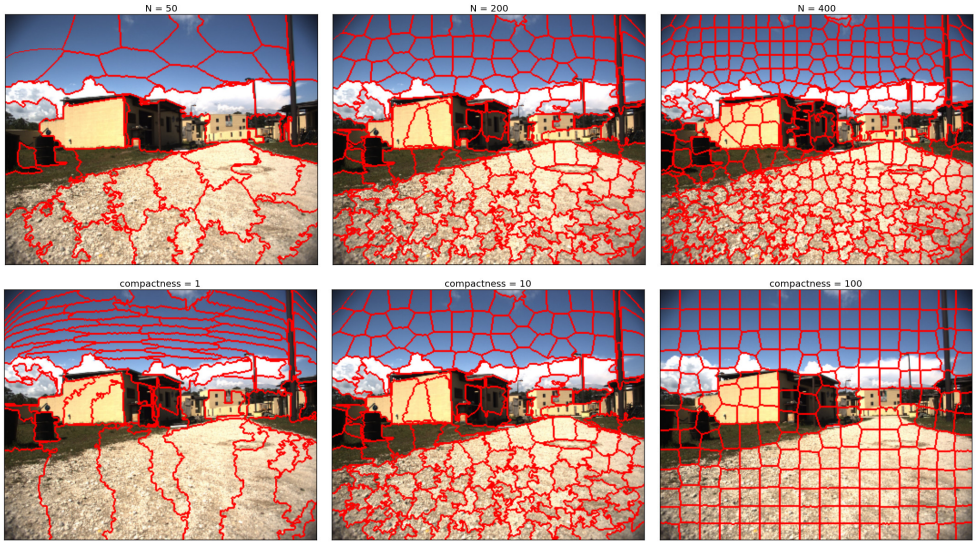

Fig. 6 SLIC superpixel output when the parameters N (number of segments) and compactness are varied

As the user changes these values, the output of the segmentation changes in real-time. This dynamic segmentation provides greater flexibility to produce more accurate labeling. The example segmentation in Fig. 6 shows that visual concepts in the image may be segmented better or worse depending on the parameter setting. Thus, the user can label as much of the image accurately as possible under one parameter setting, and update the parameters to get better segmentation for remaining areas of the image.

Ultimately, the functionality to dynamically segment an image provides the user with the opportunity to label large areas of the image with a single label assignment for visual concepts that are easier to accurately segment. Notice the large accurate superpixels in the $N = 50$ segmentation (top row of Fig. 6) for much of the sky and road. Adjusting the parameters, the user could then generate a segmentation with smaller superpixels to label more complex regions of the image.

5.4 Label Assignment

Each possible label (see Section 3.3 on how to define these) is represented by a button in the right side of GUI, which is denoted in Fig. 5 by the red outline. Notice on each button following the natural language semantic label is a unique integer ID for the label. This integer ID indicates the corresponding key press that can be used to also represent the label. **Thus, there are two ways to select the label to assign to part of the image:**

- Click the label button.
- Press the integer key associated with the label button.

Note that only the first 10 labels have associated integer key presses (i.e., 0–9).

To assign a label the user must first “mark" which superpixel(s) to label, and then select the corresponding label button or key press. **There are two ways to perform superpixel marking in this interface. The first allows a user to mark a single superpixel, while the second allows the user to mark multiple superpixels:**

- Click and release: the user clicks the mouse anywhere inside the desired superpixel boundary and then releases.
- Click and drag: the user clicks inside a superpixel boundary and then drags the mouse through any other superpixel boundaries that should also be marked with the same label.

During the marking stage a user can use the two marking techniques in any combination. Superpixels are added to a labeling queue as long as marking techniques are performed, and when a label button is selected every superpixel currently in the queue will be assigned the selected label.

When a label button is clicked, the image label display immediately updates so the user can see the current set of labels assigned to the image.

5.5 Toggling Between Modes

Any time the user toggles between view-only and labeling modes, the current state of the GUI is reset to the following properties:

- The zoom is defined as the full image size.
- All superpixels marked and stored in the labeling queue are removed.
- Both slider bars are set to a default value of 1.

6. Conclusion

It is fast and easy to collect large amounts of raw visual data. However, the process of manually annotating this data to provide ground truth information that machine learning algorithms can learn from can be very tedious and time consuming. Although there is effort in learning from unlabeled or weakly labeled data, the performance of supervised learners given large amounts of labeled training data still far exceeds these other lines of research. Tools that ease the burden of image labeling are important to ensure fast and accurate annotation results.

The use of the SuperLabel software enables researchers to collect accurate pixel-wise label annotations that can be used for machine learning tasks such as semantic segmentation, object detection, and scene understanding. By leveraging unsupervised superpixel segmentation, the manual annotation effort of identifying and outlining distinct regions in an image can be eliminated. Instead, through dynamic parameter selection, the user can generate different but accurate region boundaries for each visual concept in the data. The SuperLabel GUI enables fast superpixel marking techniques and a real-time visual update of the label assignments made by the user.

7. References

1. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In: Proceedings of the International Conference on Learning Representations; 2015 May; San Diego, CA.

2. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Proceedings of the Conference on Neural Information Processing Systems; 2012 Dec; Lake Tahoe, NV. p. 1097–1105.

3. Hariharan B, Arbeláez P, Girshick R, Malik J. Hypercolumns for object segmentation and fine-grained localization. In: Proceedings of the Conference on Computer Vision and Pattern Recognition; 2015 Jun; Boston, MA. p. 447–456.

4. Girshick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the Conference on Computer Vision and Pattern Recognition; 2014 Jun; Columbus, OH. p. 580–587.

5. Chen LC, Papandreou G, Kokkinos I, Murphy K, Yuille AL. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. Transactions on Pattern Analysis and Machine Intelligence. 2018;40(4):834–848.

6. Lin G, Milan A, Shen C, Reid ID. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In: Proceedings of the Conference on Computer Vision and Pattern Recognition; 2017 Jul; Honolulu, HI.

7. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L. Imagenet: A large-scale hierarchical image database. In: Proceedings of the Conference on Computer Vision and Pattern Recognition; 2009 Jun; Miami, FL. p. 248–255.

8. Fei-Fei L. Crowdsourcing, benchmarking and other cool things. 2010 [accessed 2018 Jul 27]. http://www.image-net.org/papers/ImageNet_2010.pdf.

9. Saenko K, Kulis B, Fritz M, Darrell T. Adapting visual category models to new domains. In: Proceedings of the European Conference on Computer Vision; 2010 Sep; Crete, Greece. p. 213–226.

10. Russell BC, Torralba A, Murphy KP, Freeman WT. Labelme: a database and web-based tool for image annotation. International Journal of Computer Vision. 2008;77(1-3):157–173.

11. Documentation: Image labeler. MathWorks [accessed 2018 Aug 27]. https://www.mathworks.com/help/vision/ref/imagelabeler-app.html.

12. Welcome to sloth's documentation! [accessed 2018 Aug 27]. https://sloth.readthedocs.io/en/latest/.

13. Simple image annotation (simanno) tool for machine learning projects. GitHub [revised 2018 Jul 03; accessed 2018 Aug 27]. https://github.com/faisalthaheem/simanno.

14. Labelbox. [accessed 2018 Aug 27]. https://labelbox.com/.

15. Ros G, Sellart L, Materzynska J, Vazquez D, Lopez AM. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In: Proceedings of the Conference on Computer Vision and Pattern Recognition; 2016 Jun; Las Vegas, NV. p. 3234–3243.

16. Zhang Y, David P, Gong B. Curriculum domain adaptation for semantic segmentation of urban scenes. In: Proceedings of the International Conference on Computer Vision; 2017 Oct; Venice, Italy.

17. Hoffman J, Wang D, Yu F, Darrell T. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. arXiv preprint arXiv:1612.02649; 2016.

18. Felzenszwalb PF, Huttenlocher DP. Efficient graph-based image segmentation. International Journal of Computer Vision. 2004;59(2):167–181.

19. Achanta R, Shaji A, Smith K, Lucchi A, Fua P, SÃijsstrunk S. Slic superpixels compared to state-of-the-art superpixel methods. Transactions on Pattern Analysis and Machine Intelligence. 2012;34(11):2274-2282.

20. Liu MY, Tuzel O, Ramalingam S, Chellappa R. Entropy rate superpixel segmentation. In: Proceedings of the Conference on Computer Vision and Pattern Recognition; 2011 Jun; Colorado Springs, CO. p. 2097–2104.

21. Shi J, Malik J. Normalized cuts and image segmentation. Transactions on Pattern Analysis and Machine Intelligence. 2000;22(8):888–905.

List of Symbols, Abbreviations, and Acronyms

2-D 2-dimensional

ARL US Army Research Laboratory

GUI graphical user interface

ROI region of interest

SLIC Super Linear Iterative Clustering

1 (PDF) DEFENSE TECHNICAL INFORMATION CTR DTIC OCA

2 (PDF) DIR ARL IMAL HRA RECORDS MGMT RDRL DCL TECH LIB

1 (PDF) GOVT PRINTG OFC A MALHOTRA

7 (PDF) ARL RDRL CII A M WIGNESS J ROGERS D BARAN E STUMP G WARNELL P DAVID D SUMMERS-STAY

2 (PDF) ARL RDRL CII D HAN J FOSSACECA