ARL-TN-1064 ● June 2021

Enhanced Annotation for Semantic Segmentation on Unstructured Video Sequences for Robotic Navigation

by Christine Kwon and Maggie Wigness

ARL-TN-1064 ● JUNE 2021

Enhanced Annotation for Semantic Segmentation on Unstructured Video Sequences for Robotic Navigation

Christine Kwon
College Qualified Leaders Program, University of Notre Dame

Maggie Wigness
Computational and Information Sciences Directorate,
DEVCOM Army Research Laboratory

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* June 2021	**2. REPORT TYPE** Technical Note	**3. DATES COVERED (From - To)** June–August 2020
4. TITLE AND SUBTITLE Enhanced Annotation for Semantic Segmentation in Unstructured Video Sequences for Robotic Navigation		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Christine Kwon and Maggie Wigness		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) DEVCOM Army Research Laboratory ATTN: FCDD- RLC-IA Adelphi, MD 20783-1138		**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TN-1064
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 unlimited.

13. SUPPLEMENTARY NOTES
ORCID ID: Maggie Wigness, 0000-0003-1707-8106

14. ABSTRACT
Methods of visual perception provide identification of different landmarks and terrain that can help improve the intelligence of a robot, allowing it to efficiently optimize its paths to avoid obstacles and rough terrain. Semantic segmentation is a type of perception task that applies labels to every pixel within an image after being rigorously trained on large datasets that are accurately annotated. Due to human error there are bound to be annotated images with mislabeled or unlabeled pixels that can distort learning, affecting the visual perception of a robot. To address and correct these errors we propose automated relabeling algorithms. We exploit minute changes in object location between consecutive frames in a video sequence by referencing and comparing related pixels in adjacent frames; if the label values of those pixels in the neighboring images match, we can infer the label of the unlabeled pixel at the corresponding location in the image of interest. As a way to collect more evidence, we extend this approach to use peripheral pixels within a radius threshold in the neighboring images. These pixel-wise labeling solutions and analyses of their resulting annotated images will enable faster annotation and error correction by eliminating human labeling effort. We provide initial results of our automatic annotation inference and discuss the implications this will have on machine learning models used to provide perception information to autonomous robots.

15. SUBJECT TERMS
visual perception, semantic segmentation, automated annotation, robot perception, visual-aware navigation

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Maggie Wigness
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	UU	21	**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

Robust visual perception for autonomous robots in various environments is becoming increasingly imperative for navigation tasks. Robots must be able to successfully and effectively identify landmarks within an environment to ameliorate their path planning and decision making. There are a number of different visual perception tasks (e.g., object classification, object detection, and semantic segmentation) that can provide robots with the environmental context, and each task requires a different type of annotated data to learn its respective perception model.[1]

We specifically focus on the task of semantic segmentation to provide visual perception for a robot, where each pixel in an image is identified as a semantic class (e.g., vehicle, road, or tree). Semantic segmentation is generally a supervised machine learning (ML) task, meaning that the algorithm being used requires raw image data and ground truth annotations to guide the learning process. The ground truth annotations are of the same format that we want the algorithm to learn to output. In the case of semantic segmentation, the ground truth for the training images is the class label for each pixel. Figure 1 provides an example of a semantic segmentation annotation. Ground truth annotations are generally collected by a human, and for complex scenes like urban environments the annotation process can be very time consuming.

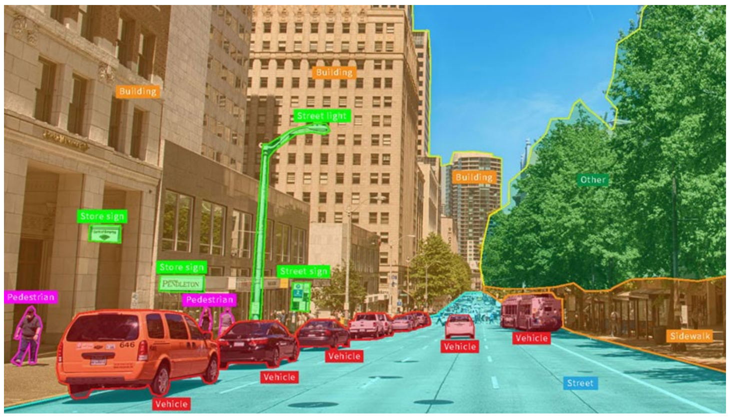


Fig. 1 Example pixel-wise annotation of an urban environment. A human annotator must draw the boundaries between every object in the scene and then assign a semantic label. Each color in this image represents a different semantic class.

For our project, we wanted to investigate the ground truth annotations that come from humans. Recently the US Army Combat Capabilities Development Command

Army Research Laboratory collected video sequences recorded by robots in an outdoor environment. The initial annotation process for this data consisted of outsourcing the labeling to human annotators who were supposed to label every pixel within each frame in a video sequence. Figure 2 shows an example of a raw image (left) from the outdoor environment and the ground truth annotation (right) that was collected from a human. The data consists of a number of objects (e.g., tree, bush, fence, and barrier) and terrains (e.g., grass, asphalt, and mud). However, there is an evident negative consequence when human annotators are labeling each frame: human error. Human error can be incorrectly labeling pixels (e.g., saying grass terrain is a tree) or forgetting to label a pixel altogether. These types of errors can be damaging to the training of an ML algorithm, causing the ability to detect certain landmarks to deteriorate immensely and resulting in degraded visual perception for robots.

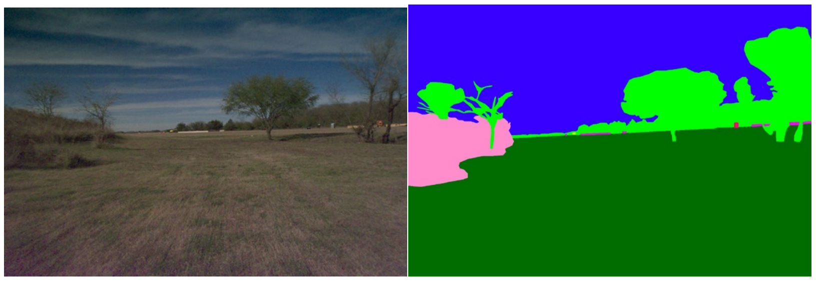


Fig. 2 Example raw image and its corresponding annotation representation. Both pieces of information are needed to train a supervised semantic segmentation algorithm.

This research is focused on removing or correcting any possible annotation errors before the data is used to train a visual perception model. By correcting human errors, we believe that more-robust ML models can be generated to provide the best quality visual perception for robots operating in unstructured environments. Since annotation is already time consuming, we do not want to introduce more human effort into the label correct process. Thus, we discuss methods of automated correctional labeling for annotated frames in video sequences. This automation identifies potential errors (i.e., mislabeled or unlabeled) in the labels, infers the correct label, and relabels pixels without any human intervention.

2. Image Annotation Background

Image annotation is an essential task to collect the necessary information needed for supervised ML. Different types of visual perception tasks can be applied to an image and each require specific image annotation types. Each type of annotation requires varying degrees of human labeling effort. We provide a brief background on the different types of image annotation and associated visual perception tasks.

Image annotation, in general, is the process of assigning an image with one or more class labels (e.g., vehicle, road, and building). This type of annotation is a human-powered task where human annotators use a predetermined label set, often referred to as a dataset ontology, to attach semantics to an image. Image annotations are conventionally categorized into different types with varying degrees of difficulty. Some of the most common include label assignment to the whole image, bounding box annotation, polygon annotation, annotation of lines and splines, and semantic segmentation.

Figure 3 shows examples of the first four types of annotations as provided by Ambalina.[1] Assigning a single label to an entire image is the easiest type of annotation since it does not require the human annotator to localize a specific object. These annotations are useful for classification tasks. Bounding box, polygon, and line and spline annotation requires the human user to localize objects (i.e., define their boundaries in the images) in addition to providing a semantic label. This results in a greater degree of effort but also provides more information. These types of annotations are used for detection tasks, where localization of landmarks is the ultimate goal.

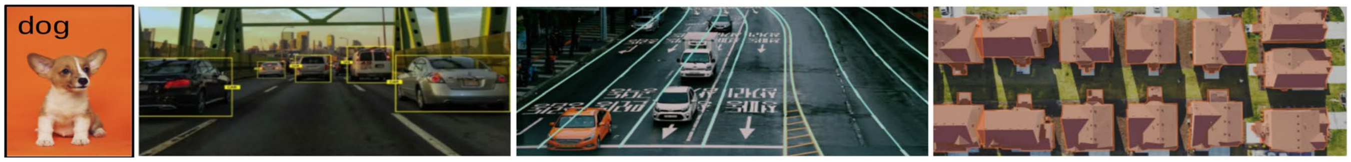


Fig. 3 Examples of different types of image annotations: (left to right) whole image label assignment, bounding box annotation, lines and splines annotation, and polygon annotation[1]

Semantic segmentation is one of the most time consuming methods of image annotation, but also provides the most-dense information about the contents of an image. This type of image annotation considers every pixel within an image and associates each pixel with a semantic class from the predetermined ontology. Figures 1 and 2 provide examples of semantic segmentation in urban and off-road unstructured environments, respectively. Semantic segmentation is becoming the most popular form of visual perception to run on autonomous vehicles since it provides the most-precise information about landmarks and terrain in an environment.

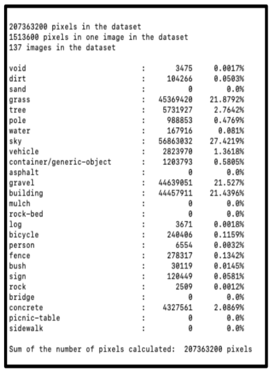
Because semantic segmentation is known as the most-dense method of image annotation, it is one of the most difficult and time consuming methods. Annotating every single pixel is a very tedious task, and therefore humans are prone to produce errors that involve mislabeled or unlabeled pixels within a video sequence. The rest of this technical note outlines how we address these errors.

3. Methodology

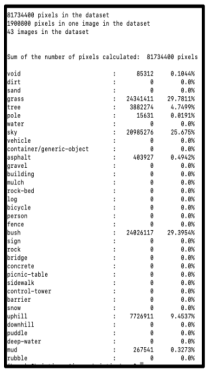
This technical note focuses on the effects of methods that perform automatic label inference to relabel images with unlabeled pixel annotations. The presence of unlabeled pixels within video sequences exist mainly due to human error during the annotation process. The presence of unlabeled pixels results in missing environment context and could potentially degrade visual perception learning. This research, specifically, is concerned with the process that is performed prior to ML training (i.e., evaluating video sequences and correcting annotations in each frame within those video sequences to ensure each specific landmark is annotated accurately).

The research focused on three video sequences. Through the programming language of Python, we were able to produce dataset statistics that analyze the frequency of semantic classes in the annotated frames of those three video sequences. The frequency is represented as a percentage of total pixels labeled for each object class. Furthermore, a graphical representation of each video sequence was produced to visualize where class annotations occur throughout the duration of the video sequence. Figure 4 shows the output of our initial analysis of the annotations.

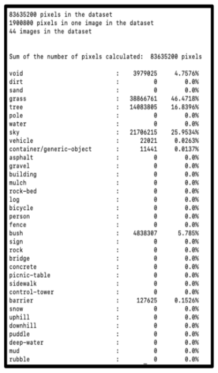
Moreover, these statistics quickly identify when images with unlabeled pixels occur within a video sequence. We use this information to deploy innovative automatic inference methods to label these unlabeled pixels. Our experiments and results in the rest of this technical note focus on one specific video sequence that contains images with alarmingly high concentrations of unlabeled pixels. Figure 5 shows the statics of this particular video sequence and illustrates the frames of the video sequence that we test our automatic label inference on.



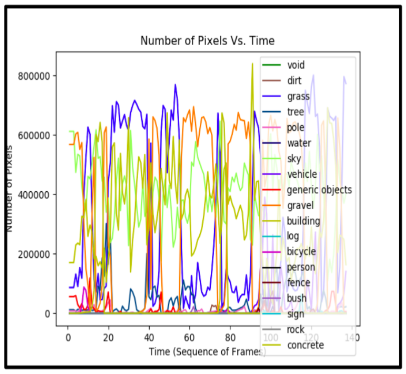
```
207363200 pixels in the dataset
1513600 pixels in one image in the dataset
137 images in the dataset

void                            :       3475     0.0017%
dirt                            :     104266     0.0503%
sand                            :          0        0.0%
grass                           :   45369420   21.8792%
tree                            :    5731927    2.7642%
pole                            :     988853    0.4769%
water                           :     167916     0.081%
sky                             :   56863032   27.4219%
vehicle                         :    2823970    1.3618%
container/generic-object        :    1203793    0.5805%
asphalt                         :          0        0.0%
gravel                          :   44639051    21.527%
building                        :   44457911   21.4396%
mulch                           :          0        0.0%
rock-bed                        :          0        0.0%
log                             :       3671    0.0018%
bicycle                         :     240406    0.1159%
person                          :       6554    0.0032%
fence                           :     278317    0.1342%
bush                            :      30119    0.0145%
sign                            :     120449    0.0581%
rock                            :       2509    0.0012%
bridge                          :          0        0.0%
concrete                        :    4327561    2.0869%
picnic-table                    :          0        0.0%
sidewalk                        :          0        0.0%

Sum of the number of pixels calculated:  207363200 pixels
```

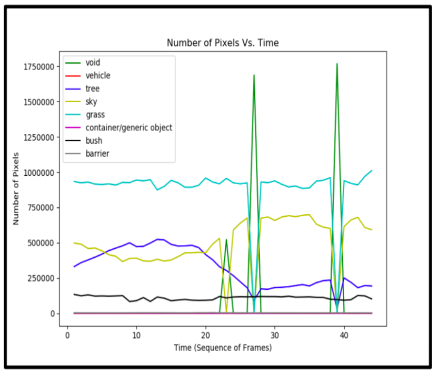


```
81734400 pixels in the dataset
1900800 pixels in one image in the dataset
43 images in the dataset


Sum of the number of pixels calculated:  81734400 pixels

void                            :      85312    0.1044%
dirt                            :          0       0.0%
sand                            :          0       0.0%
grass                           :   24341411   29.7811%
tree                            :    3882274    4.7499%
pole                            :      15631    0.0191%
water                           :          0       0.0%
sky                             :   20985276    25.675%
vehicle                         :          0       0.0%
container/generic-object        :          0       0.0%
asphalt                         :     403927    0.4942%
gravel                          :          0       0.0%
building                        :          0       0.0%
mulch                           :          0       0.0%
rock-bed                        :          0       0.0%
log                             :          0       0.0%
bicycle                         :          0       0.0%
person                          :          0       0.0%
fence                           :          0       0.0%
bush                            :   24026117   29.3954%
sign                            :          0       0.0%
rock                            :          0       0.0%
bridge                          :          0       0.0%
concrete                        :          0       0.0%
picnic-table                    :          0       0.0%
sidewalk                        :          0       0.0%
control-tower                   :          0       0.0%
barrier                         :          0       0.0%
snow                            :          0       0.0%
uphill                          :    7726911    9.4537%
downhill                        :          0       0.0%
puddle                          :          0       0.0%
deep-water                      :          0       0.0%
mud                             :     267541    0.3273%
rubble                          :          0       0.0%
```



```
83635200 pixels in the dataset
1900800 pixels in one image in the dataset
44 images in the dataset


Sum of the number of pixels calculated:  83635200 pixels

void                            :    3979025    4.7576%
dirt                            :          0       0.0%
sand                            :          0       0.0%
grass                           :   38866761   46.4718%
tree                            :   14083805   16.8396%
pole                            :          0       0.0%
water                           :          0       0.0%
sky                             :   21706215   25.9534%
vehicle                         :      22021    0.0263%
container/generic-object        :      11441    0.0137%
asphalt                         :          0       0.0%
gravel                          :          0       0.0%
building                        :          0       0.0%
mulch                           :          0       0.0%
rock-bed                        :          0       0.0%
log                             :          0       0.0%
bicycle                         :          0       0.0%
person                          :          0       0.0%
fence                           :          0       0.0%
bush                            :    4838307     5.785%
sign                            :          0       0.0%
rock                            :          0       0.0%
bridge                          :          0       0.0%
concrete                        :          0       0.0%
picnic-table                    :          0       0.0%
sidewalk                        :          0       0.0%
control-tower                   :          0       0.0%
barrier                         :     127625    0.1526%
snow                            :          0       0.0%
uphill                          :          0       0.0%
downhill                        :          0       0.0%
puddle                          :          0       0.0%
deep-water                      :          0       0.0%
mud                             :          0       0.0%
rubble                          :          0       0.0%
```





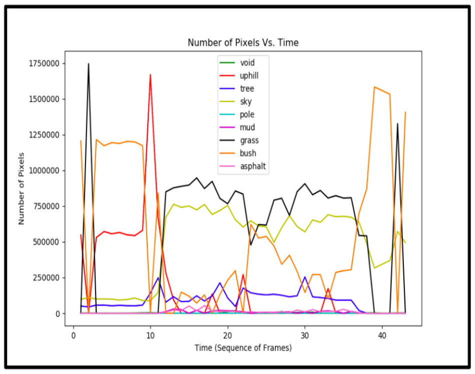
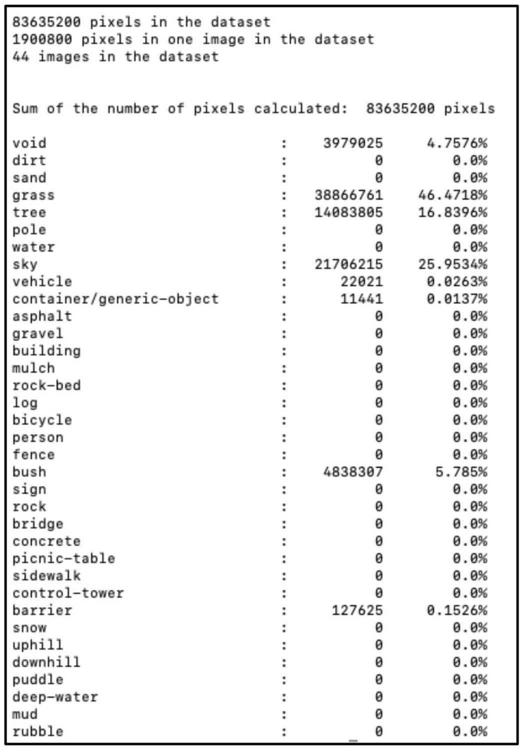
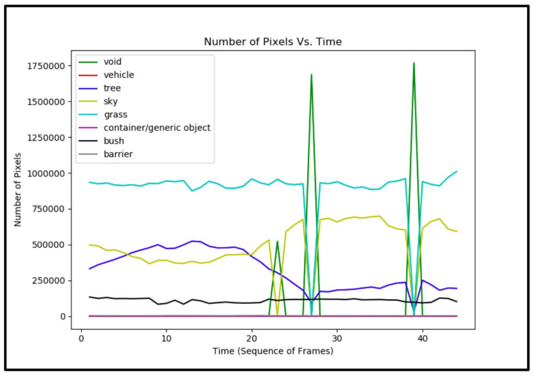


Fig. 4 Three video sequences were analyzed to determine the frequency of semantic classes. Top row: Pixel frequency of each class shown as a raw pixel count and percentage. Bottom row: Line plots that visualize the frequency of semantic classes as a function of time. This visual representation makes it easy to find where an object class most frequently occurs within a video sequence.



```
83635200 pixels in the dataset
1900800 pixels in one image in the dataset
44 images in the dataset


Sum of the number of pixels calculated:  83635200 pixels

void                           :     3979025      4.7576%
dirt                           :           0        0.0%
sand                           :           0        0.0%
grass                          :    38866761     46.4718%
tree                           :    14083805     16.8396%
pole                           :           0        0.0%
water                          :           0        0.0%
sky                            :    21706215     25.9534%
vehicle                        :       22021      0.0263%
container/generic-object       :       11441      0.0137%
asphalt                        :           0        0.0%
gravel                         :           0        0.0%
building                       :           0        0.0%
mulch                          :           0        0.0%
rock-bed                       :           0        0.0%
log                            :           0        0.0%
bicycle                        :           0        0.0%
person                         :           0        0.0%
fence                          :           0        0.0%
bush                           :     4838307       5.785%
sign                           :           0        0.0%
rock                           :           0        0.0%
bridge                         :           0        0.0%
concrete                       :           0        0.0%
picnic-table                   :           0        0.0%
sidewalk                       :           0        0.0%
control-tower                  :           0        0.0%
barrier                        :      127625      0.1526%
snow                           :           0        0.0%
uphill                         :           0        0.0%
downhill                       :           0        0.0%
puddle                         :           0        0.0%
deep-water                     :           0        0.0%
mud                            :           0        0.0%
rubble                         :           0        0.0%
```



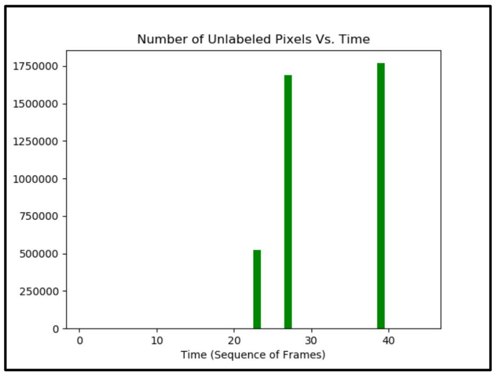


Fig. 5 The video sequence used to test our methods for automatic inference of pixels that are unlabeled. This is a short video sequence, but it contains several images with high concentrations of unlabeled pixels as seen in the bottom right image.

3.1 Adjacent Frame Evidence

Temporal ordering of images is a unique characteristic of video sequences. In essence, there are minute changes in object location from frame to frame (with a high frame rate), so there is a high probability that we will have equivalent class pixel values throughout multiple consecutive frames. Therefore, when we come across a frame with unlabeled pixels, we can use the fully annotated neighboring frames to estimate the labels of those unlabeled pixels.

We were able to create a Python function that creates a sliding window in a video sequence that represents three consecutive frames. Specifically, we look at windows with the center frame that contains unlabeled pixels. The other frames in this window represent the previous and next images in the video sequence with respect to the center frame with unlabeled pixels. The function analyzes the center image, pixel by pixel, to calculate the coordinate of each unlabeled pixel. For each unlabeled pixel coordinate, the function evaluates the label associated with the same pixel coordinate in the adjacent previous and next images. At their respective pixel coordinate, if the pixel label from the previous and next images are equivalent, the function will relabel the unlabeled pixel with that matching pixel label. This first approach is very conservative in the sense that an unlabeled pixel value is only inferred if neighboring frames provide unanimous agreement.

In the left window of Fig. 6 it is fairly obvious that the unlabeled pixels (shown in black) from the center frame should be labeled as "blue", which represents the sky class. However, the middle and right window examples have a much larger portion of unlabeled pixels, and it is much more difficult to visually determine the labels of the unlabeled pixels. Accordingly, we use the adjacent frames from the windows, which represent fully annotated images, and our previously described inference method to infer an annotation label and relabel unlabeled pixels of the image of interest. The bottom of Fig. 6 shows the results of this automatic inference.

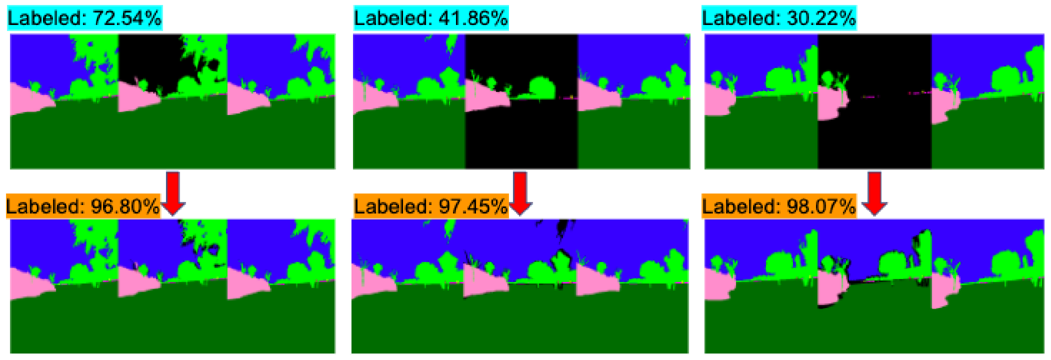


Fig. 6 Method that makes use of the information encoded in temporal ordering of images. The top row shows three slide windows, where the middle frame contains unlabeled pixels, and the bottom row shows the inferred label results after using label evidence from neighboring frames.

With this simple method we are able to infer most of the unlabeled images with less than 5% of the pixels remaining unlabeled after running our inference. However, in some cases the adjacent frames used as evidence support to infer an unlabeled pixel might disagree. For example, the previous frame may have a green pixel representing grass and the next frame may have a blue pixel representing sky. This is most often seen at the region boundaries of classes in the annotations and is expected to happen since the robot collecting the data is in motion and landmarks will shift from frame to frame. In this case, our algorithm leaves the unlabeled pixel as is since there is no agreeing evidence, and this leaves regions of unlabeled pixels within the image. In the bottom row of Fig. 6, the newly labeled images still contain some regions of unlabeled pixels (shown in black). Another shortcoming of this approach arises when dealing with consecutive frames containing unlabeled pixels. In this case, there is no agreeing supporting evidence, and running our inference will result in very little improvement in the total pixels labeled.

3.2 Adjacent Frame Evidence with Neighborhood Radius

To address the shortcoming previously mentioned in Section 3.1, we extend the neighborhood of pixel evidence in adjacent frames to run our automatic labeling inference. We use the same three consecutive frames defined previously to make up the sliding window. Rather than simply comparing the single pixel labels from neighboring frames at the same respective pixel coordinate as the unlabeled pixel, our code uses a neighborhood radius that collects and gathers fully annotated pixels and calculates the most commonly occurring pixel label from this neighborhood.

This approach is able to gather more evidence for the voting algorithm that we created to calculate the most frequently occurring pixel label from the collection of surrounding annotated pixels. Using these pixel label evidence statistics we can automatically assign a label to the unlabeled pixels. As seen in Fig. 7, this method assigns a label to every unlabeled pixel, noted by the 100% measure of pixels labeled, since the most frequent label from the evidence is always chosen as the inferred label.

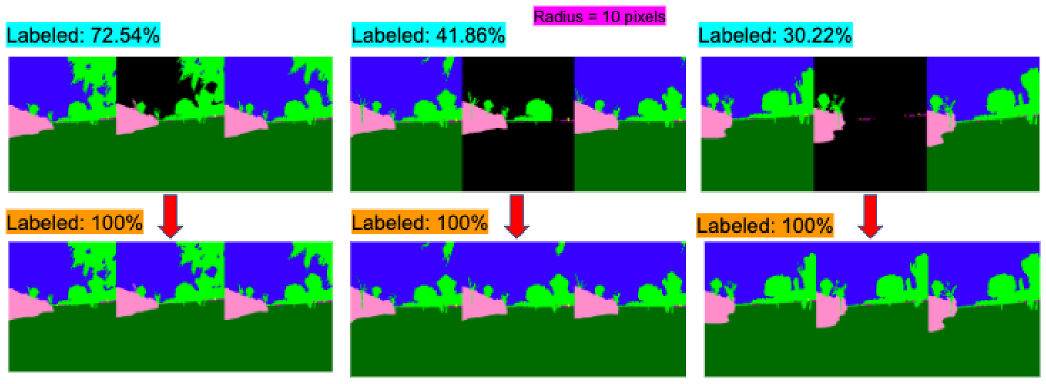


Fig. 7 Results of our approach that uses a neighborhood radius around the unlabeled pixels to gather label evidence from adjacent frames in the sliding window. In these three images, the radius is fixed to 10 pixels.

However, there are present disadvantages to this method. When calculating the most commonly found pixel value within the neighborhood radius, the function calculates a maximum number of occurrences from the voting algorithm. In consequence, we are confronted with instances where we have an equivalent number of occurrences of two or more class labels in which the voting algorithm will arbitrarily choose one of these to relabel an unlabeled pixel. Additionally, the maximum vote from the evidence does not guarantee that the label selected will be accurate. Consider the case when there are 20 pixels providing label evidence, and 18 of these pixels are themselves unlabeled and 2 of these pixels represent grass. Our algorithm would select grass at the label to be inferred even though only 10%

of the queried pixels match that label. The radius parameter selection for this approach is also an important factor to consider. In the instance of using a smaller radius, there is little evidence provided to make an inference decision. However, by using an extremely large radius, there is a high probability of calculating an incorrect label value due to examining many pixels far from the unlabeled pixel of interest.

3.3 Voting Thresholding

Our next approach uses voting thresholding to avoid some of the disadvantages mentioned in Section 3.2. In this setting the threshold defines how much agreeing evidence is needed to infer a pixel value. This helps resolve the issue of having two or more labels with maximum number of occurrences (i.e., ties for the max vote) or little evidence support due to unlabeled evidence, overall eliminating possible inference labeling errors.

This method uses the same algorithm described in Section 3.2 that uses a neighborhood radius to collect label evidence. However, rather than using a maximum voting algorithm to infer the pixel label, this method uses a function that will determine if the respective maximum number of occurrences is good enough to make an inference based on some threshold. The function saves and updates the number of occurrences for each pixel label detected within the neighborhood radius so as to calculate the pixel label with the greatest number of occurrences. Based on the total number of pixels used to collect evidence, the calculated maximum number of occurrences of the pixel value is converted to a percentage value. If the max label percentage is greater than or equal to the defined threshold, the function will automatically relabel the unlabeled pixel with the label with the maximum number of occurrences. In the case that the maximum does not surpass the threshold, the unlabeled pixel will remain void.

Results of this approach can be seen in Fig. 8 using the image on the far left with unlabeled pixels. The inference and thresholding function is applied to the image with a threshold of 50%. As this function is applied to the image with increasing radii values, fewer and fewer of the originally unlabeled pixels receive an inferred label. In addition, the landmarks within the unstructured environment become less clearly distinguished and detected as the set radius increases, indicating an escalation in the number of classes with label agreement in the neighborhood of pixels, which could lead to incorrect inference. In this particular unlabeled image, a smaller radius with a threshold of 50% results in a more accurate label inference of the image.

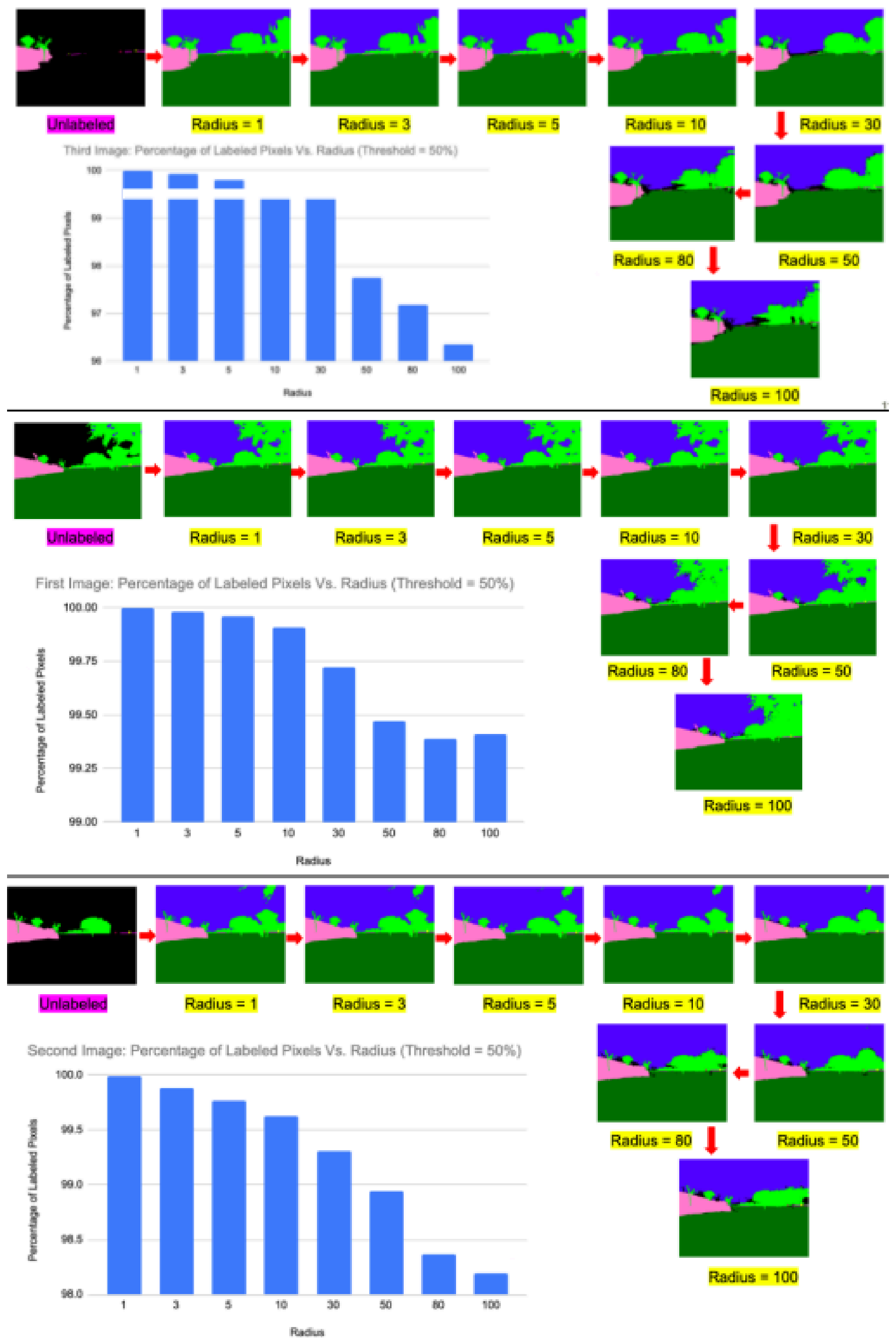


Fig. 8 Method of thresholding applied to three images with unlabeled pixels with multiple tested fixed radii values. The threshold is fixed as 50%. The bar plot compares the concentration of labeled pixels vs. various fixed radii values.

Although this method helps address shortcomings from the methods discussed in previous sections, we still encounter the problem of existing regions of unlabeled pixels. Although smaller radii with a threshold of 50% applied to this image results in almost a 100% label inference (i.e., nearly all pixels are assigned labels), there still remains unlabeled pixels within the image. Despite the fact that this issue might seem insignificant due to the unlabeled pixels not being visible by the human eye, other video sequences might consider those unlabeled pixels essential to the visual perception of a robot. For future work, we believe it is essential to consider methods that make use of a large amount of agreeing evidence to infer labels within an incomplete annotated image.

4. Conclusions and Future Work

The methods discussed in this technical note were designed to address the need for automatic label correction. As such, the three methods of label inference require no human effort to relabel annotated frames within the video sequences. The presented methods progressively provided more-sophisticated solutions to addressing relabeling an annotated image. The foundation of the work relies on the temporal correlation that can be extracted from sequential frames in the video sequence. Using neighboring frames, we can evaluate the pixel label at the same coordinate in the image and use voting mechanisms to infer the label. We presented a technique that required strict agreement between neighboring frames; however, this method often leaves a large number of pixels unlabeled due to disagreement. Next, we presented a method that gathers label evidence from a radius of neighbors to label every unlabeled pixel with the most common value within the neighborhood.

Nonetheless, there are drawbacks to this method. We are confronted with the problem of having multiple maximum numbers of occurrences of two or more labels plus the issue of the most frequent label still not providing significant evidence for inference, with implications in incorrect inferencing. Again, to solve these drawbacks, we concluded with an approach that uses a threshold value to determine if enough evidence can be found in the neighboring region. Although this method does not require that every unlabeled pixel be labeled, it produces results in which the labeled concentration of relabeled images is extremely close to 100%.

These methods demonstrate effective automatic inference on these annotated images that furthers development to automate robotic navigation. These automatic labeling functions will provide additional information in the training stage of robots for future ground navigational work. Using the knowledge acquired from building these automatic inference methods, some further research topics to invest in may involve further advanced complex methods of semantic annotation techniques to label both unlabeled and mislabeled pixels. The methods of automatic inference discussed strongly correlate to the research of correcting mislabeled pixels to maintain label consistency. Figure 9 provides an example of label inconsistency. The presence of mislabeled pixels is also due to human error when annotating images within a video sequence. Therefore, these methods of label inference can directly be applied to pixels that are mislabeled by observing ordered annotated frames to identify inconsistent labels from frame to frame. We hope to address this application in future work.

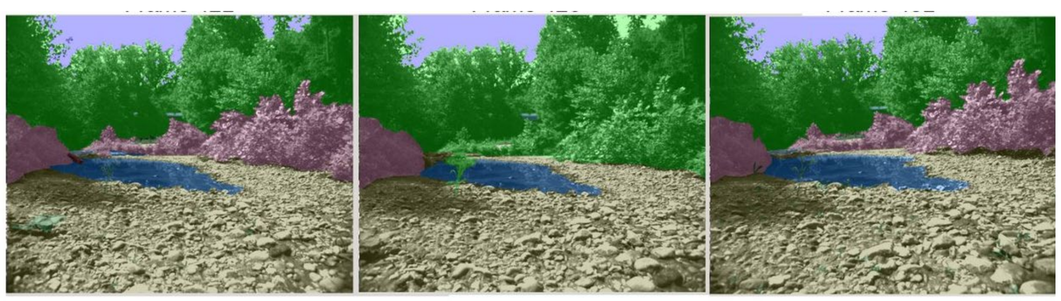


Fig. 9 Middle image is an example of inconsistent (mislabeled) labels in an image since the vegetation on the right hand side is labeled as tree (seen as green) instead of bush (seen as pink). Consecutive frames shown in this figure emphasize the inconsistency in labels.

5. References

1. Ambalina L. 5 types of image annotation and their use cases. 2019 June [accessed 2021 Mar 17]. https://lionbridge.ai/articles/an-introduction-to-5-types-of-image-annotation/.

List of Symbols, Abbreviations, and Acronyms

ML	machine learning

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLD DCI TECH LIB
1 (PDF)	DEVCOM ARL FCDD RLC IA M WIGNESS