3D AEROSPACE

Work Package XX : Work Package Title

CADM Number XXXX

Annotation of Images for Machine Learning Applications

Project

Acronym	**eHermes Prototype α**
Title	Annotation of Images for Machine Learning Applications
CADM Number	XXX
Issue Number	001
Authors	Sampreet Sarkar
Contributor	Suresh Meghana
Reviewers	Benjamin KAWAK
Date	23rd February, 2021

Executive Summary

In this document, we will describe the guidelines for proper image annotation using the free and open-source tool labelImg. We will understand the general image annotation process, what we features would want in an ideal image annotation tool, and take a look at some Deep Learning frameworks and what format of annotated images they expect.

The document is broken up as follows:

- **Section 1: Introduction –** This section provides information on how the Machine Learning application fits into the eHermes system
    - **1.1: Technical Definitions –** In this subsection, we will walk through the relevant technical definitions that apply to this document.
    - **1.2: Overview of Software/Tools –** In this subsection, we highlight the software and tools used for this application.
- **Section 2:** An ideal Image Annotation Tool – In this section, we take a look at what our expectations are from an image annotation tool, and what options are available to us.
    - **2.1: Microsoft Visual Object Tracking Tool (VoTT) –** In this subsection, we take a look at the Visual Object Tracking Tool provided by Microsoft.
    - **2.2: SuperAnnotate Desktop –** This subsection provides information on the SuperAnnotate Desktop tool
    - **2.3: Supervise.ly –** This subsection will provide us with information about the Supervise.ly tool for image annotation
    - **2.4: labelImg –** This subsection provides information about the labelImg tool for image annotation.
- **Section 3: Labelling a dataset using the labelImg tool –** In this section, we will proceed to label a custom dataset with the labelImg tool.
    - **3.1: Building the dataset –** This section will provide help on how to structure the dataset, and provide guidelines on how to go forward with the image nomenclature.
    - **3.2: Accessing the dataset in labelImg –** This subsection will guide us on how to be able to access the whole dataset using the tool.
    - **3.3: Creating Bounding Boxes around ROI –** This subsection will help us understand how to annotate an image in labelImg.
    - **3.4: Saving each annotation in YOLO format –** This subsection will help us understand how to properly save each annotation for future access.
- **Appendix:** Here we provide some useful keyboard shortcuts to help accelerate the workflow.

Thus, this document will provide an explanation of why image annotation is such an important process, how we can choose the corresponding software based on the complexity, and shows the process of annotating a dataset with the help of an annotation tool.

Document history

Version	Date	Comments
1.0	31.01.2020	First release for the PDR KO

Applicable documents

AD	CADM	Title
AD1		
AD2		

Reference documents

RD	CADM	Title
RD1		
RD2		

Contents

Executive Summary....3

1 Introduction....7

1.1 Technical Definitions....7

1.2 Overview of Software/Tools....7

2 An ideal Image Annotation Tool....7

2.1 Microsoft Visual Object Tagging Tool(VoTT)....8

2.2 SuperAnnotate Desktop....9
2.3 Supervise.ly....9
2.4 labelImg....10
3 Labelling a dataset using the labelImg tool....11
3.1 Building the dataset....11
3.2 Accessing the dataset in labelImg....12
3.3 Creating Bounding Boxes around the ROIs....12
3.4 Saving each annotation in YOLO format....13
4 Appendix....14

Figures

User Interface of Microsoft VoTT. [source]....8
The user-interface for SuperAnnotate Desktop[source]....9
The user interface for Supervise.ly[source]....10
The User Interface for the labelImg tool[source]....11
Importing a dataset in labelImg....12
Options relevant to image annotation....13
Saving annotations in labelImg....14

Tables

Table 1: Useful keyboard bindings for working in labelImg....14

Acronyms & Abbreviations

Term	Description
PASCAL	Pattern Analysis, Statistical Modelling and Computational Learning
ROI	Region of Interest
VOC	Virtual Object Class
VoTT	Visual Object Tracking Tools
YOLO	You Only Look Once

1 Introduction

For the purpose of detecting trees using the eHermes receiver, we need to be able to identify three distinct classes of objects – the trees, the metallic posts, and the wooden posts. The trees are the main objects of focus here, the posts would be used to provide information about the distance between two trees. We use the YOLOv3 framework to detect the objects with speed and accuracy. This led us to procure and annotate the images for the dataset, the process for which we will highlight in this document.

The outline of the document can be briefly summarised as follows – In section 2, we will look at what we want out of an ideal image annotating tool, we will briefly take a look at various existing image annotation tools and then we will focus on the "labelImg" tool in particular. Section 3 would focus on a detailed explanation of the annotation process for a dataset, explaining the various settings we can customize for a simpler workflow.

1.1 Technical Definitions

- **YOLO:** YOLO, an acronym for "You Only Look Once", is an object detector which can detect images in almost real time with GPU. It was initially developed by the researcher Joseph Redmond, and has since been carried on with the latest version of the framework being YOLOv5.

1.2 Overview of Software/Tools

- **labelImg:** labelImg is an open-source and free image annotation tool which helps in labelling images and exporting the annotations in multiple formats, according to the Deep Learning framework of our choice.

2 An ideal Image Annotation Tool

In the context of Deep Learning and Machine Learning applications, we need to be able to provide a large, often huge, labelled dataset. We often resort to the various existing image annotation tools and software available to aid us in this activity. All of them offer good basic functionalities, while some offer added functionalities, and premium features.

Before we start to look at image annotation tools however, we need to understand our requirements from the tool. This is an important step, since a good knowledge of the requirements at this stage will help us guide our choice of software later on, and can save us a lot of unnecessary work.

There are many acceptable image annotation formats, some of which include – bounding boxes, polygonal segmentation, semantic segmentation, and so on. If we want the simplest annotation, i.e., to be able to draw a box around the object of interest, we would want to use the bounding box annotation format. If however, we have multiple objects very densely packed, it would not be wise to use a bounding box annotation for this purpose, as this would create overlapping boxes with

different classes, which can confuse the algorithm during the training procedure. In such a case, we would like to go with a somewhat complicated annotation format like polygonal segmentation. Semantic segmentation is used when we want to distinctly identify classes down at the pixel level, and this is one of the more complicated annotations to create.

There are a lot of good image annotation tools available on the internet, which support a wide range of functionalities. We will take a brief look at three of them, pertaining to three different use cases.

2.1 Microsoft Visual Object Tagging Tool(VoTT)

One of the most feature-rich image annotation tools on the internet is the Visual Object Tagging Tool by Microsoft. It is an extremely scalable and versatile tool, which can be directly connected to an Azure database, as well as with support for local files. VoTT can perform bounding box annotation, as well as polygonal segmentation for video annotation. The user interface is simple yet rich, as is demonstrated in Figure 1. However, if we decide to use the tool as a web application, we are limited to using an Azure cloud database, or from Bing image search. This is easily solved by downloading the desktop application, which lets the user specify the incoming connection as a local directory containing images.

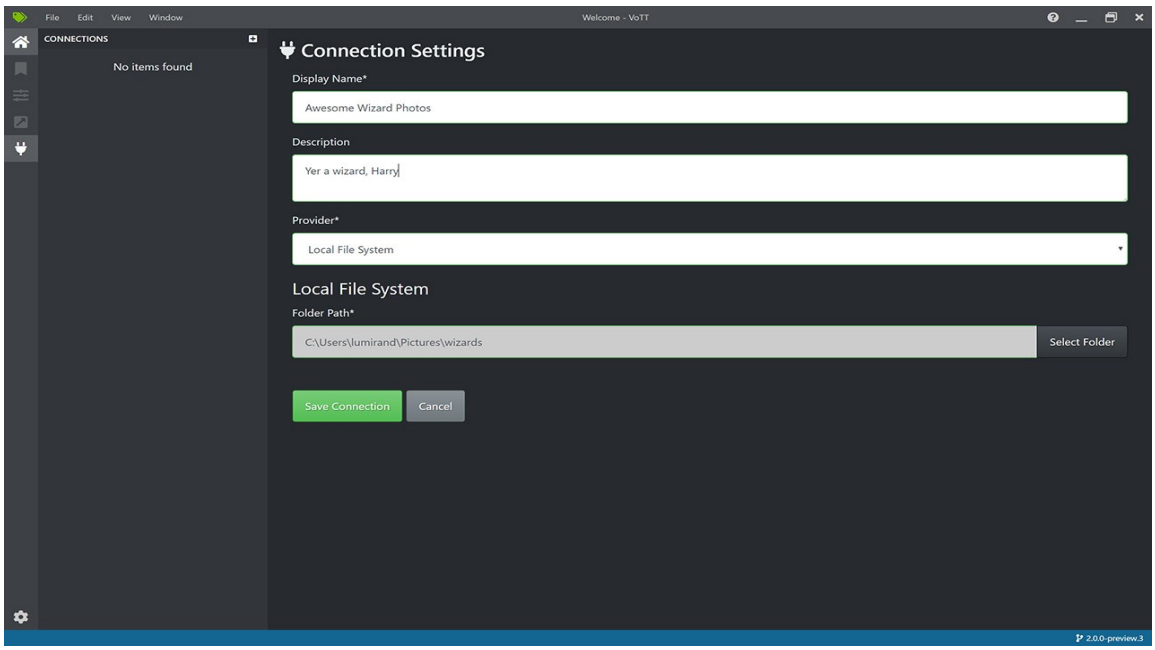

Figure 1: User Interface of Microsoft VoTT. [source]

The tool however, has a few key disadvantages which makes it unsuitable for our application. The main drawback is that using VoTT, we can only create bounding boxes, not label them. Thus, this would not be suitable for our classification task, since we need to be able to distinguish between various different classes of objects. Another key drawback of this tool is that while using the web application, we have to host our data on an Azure database, which can be expensive under some circumstances.

2.2 SuperAnnotate Desktop

With their recent partnership with OpenCV, SuperAnnotate has become one of the leading products for image annotation and labelling. With their use of AI accelerated annotation techniques, it becomes really easy to label huge batches of images with this software. They also provide image and video annotation, and supports multiple formats like keypoints, bounding boxes, polygons, 3D cuboid, etc. The annotation process is quick and simple, as can be seen in Figure 2.

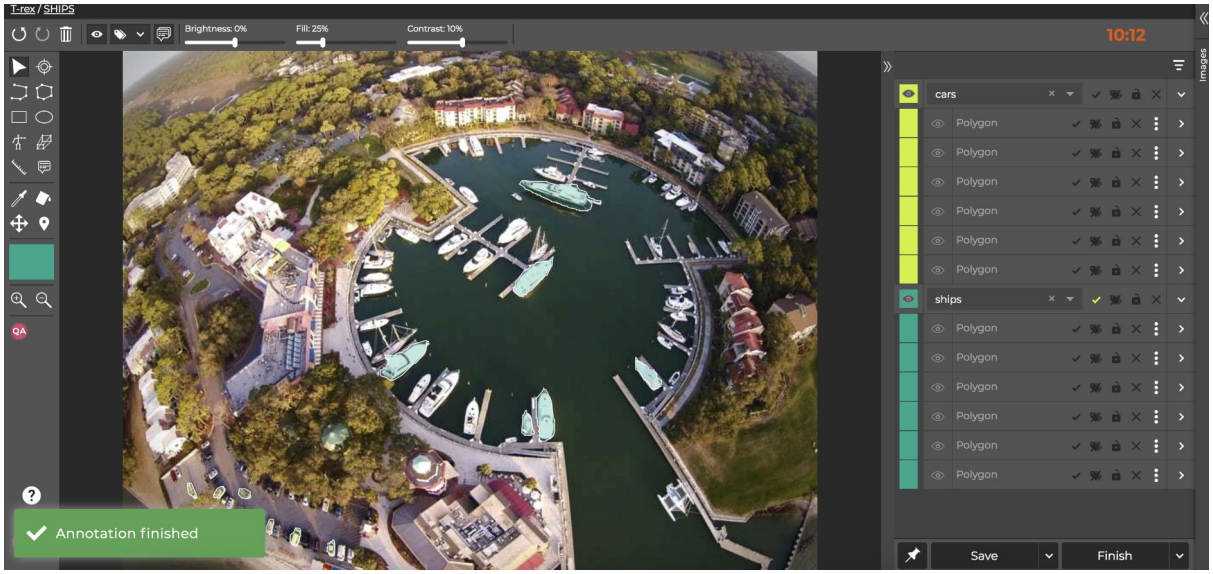

Figure 2: The user-interface for SuperAnnotate Desktop[source]

The only drawback of this software is that it is not free to use. They do offer a free trial for 14 days, during which we can annotate up to 100 images. However, that limit is too small for viable use in our application.

2.3 Supervise.ly

One of the leading web-based image annotation tools available on the internet for use, Supervise.ly is a feature-rich image annotation platform, which helps annotators accelerate their workflow with AI powered assistance. The user interface is very simple, as is demonstrated in Figure 3.

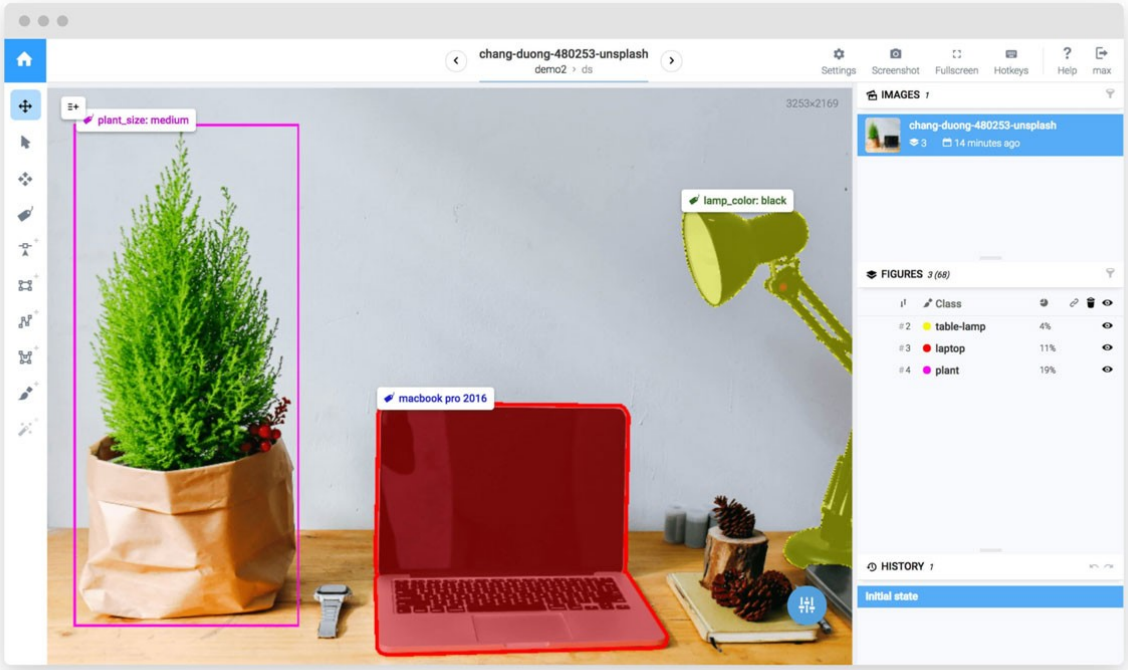

Figure 3: The user interface for Supervise.ly[source]

The fact that this platform can be used for both image annotation and experimentation with Neural Networks makes it a very strong contender in the battle of the perfect image annotation tool. This fact is also strengthened by the point that the tool can accommodate pixel-based annotation, as well as vector-based annotation. The main drawback at the moment is that the tool is not free, and the exact pricing information for enterprise level access is not known at the moment.

2.4 labelImg

The labelImg tool is an open-source and free piece of software that provides the minimal functionality for image annotation on dataset. It is written in Python, and is exceedingly customisable. The tool itself is very lightweight, but requires a few dependencies like the lxml and pyqt. The user interface is not very complicated, however it provides a lot of customisation for the user, as can be seen in Figure 4. Another feature of this tool which is very thoughtful is the support of annotations for multiple Deep Learning frameworks such as PASCAL VOC, YOLO, and CreateML – as such, it caters to a wide audience.

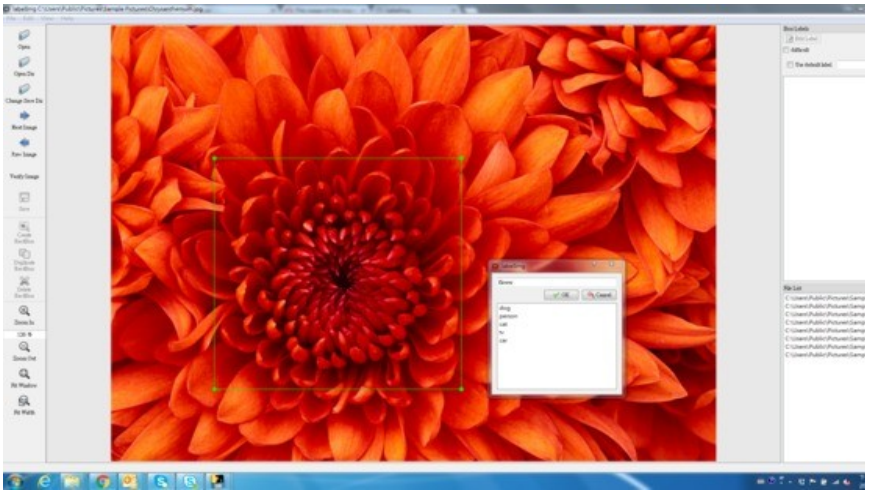

Figure 4: The User Interface for the labelImg tool[source]

The advantage of using labelImg is that it performs very well across platforms, so your workflow is not hindered if you are on a Windows machine or a Linux machine. We can also use the tool to verify the annotations to check if we have missed anything or not. The drawbacks of this tool are that the UI looks very old, and it cannot be used in a distributed setting, such that a team of annotators could access a local repository and distribute the workload amongst them.

3 Labelling a dataset using the labelImg tool

In this section, we will walk through the entire process of annotating our custom dataset. This will involve the following steps:

- **Building the dataset –** This will involve procurement of the images, and naming the images in a particular order to maintain chronology, and subsequently sanity.
- **Accessing the dataset in labelImg –** We have the option to choose from opening image by image, or opening the directory and using keyboard bindings to navigate across images.
- **Creating bounding boxes around ROIs –** Once we have our image opened, we will have to create bounding boxes around each object of interest.
- **Saving each annotation in YOLO format –** Finally, we need to save the locations of these bounding boxes in the YOLO specified format.

The following sections will describe each process in further detail.

3.1 Building the dataset

In principle, building a dataset is not a complicated task. We simply have to collect a significant amount of images in order to help our model learn the features. However, there are a few steps and guidelines which help us facilitate the labelling process. This would primarily involve imposing some kind of order of exploration of the images, of which the most common is as ascending numerical order, as “img-1.jpg”, “img-2.jpg”, etc. This may count as a purely cosmetic process, but the benefit

that we will reap from doing so is exponential in terms of fault diagnosis, which is more clearly observable during handling of large to huge datasets*(~1000+ images)*. Another guideline that we might want to follow is to store all the images within a specific directory, this will make our life easier by allowing us to import the directory directly in the labelImg tool.

3.2 Accessing the dataset in labelImg

We can choose one of two available options for accessing the dataset in labelImg – we might want to load an image, annotate it, and then move on to a new image. Or, we could instead import the entire directory containing all the images at once, and proceed to annotate each image relatively quickly. To import a directory in labelImg, we navigate to the menu bar on the left side of the application window, and select “Open Dir”, as shown in Figure 5.

Figure 5: Importing a dataset in labelImg

This would result in the locations of the images on disk being loaded on to the program as is visible on the right-hand side of the application window. From here on, we can proceed to label the images, the process for which is highlighted in the next subsection.

3.3 Creating Bounding Boxes around the ROIs

Now that we have been able to successfully import the dataset on to the application, we would like to be able to annotate the images for the Regions of Interest(ROIs). This process is fairly simple, as the application provides us with the option to create a rectangular bounding box, called “Create RectBox”. When we select this option, we will see a crosshair appear in the image area, which we can point at the start of our ROI. Then we can drag the mouse, making sure that we cover as much of the object of interest as is possible with a rectangular box, while making sure that we omit as much of the background and other unnecessary objects in the image.

Figure 6: Options relevant to image annotation

The options of interest have been highlighted in Figure 6, and a very important thing that we have to make sure during annotation of our dataset, is that we select the proper annotation export format. In our case, it is YOLO. This is a trivial step, however, overlooking this setting might cause us to export the annotations in an entirely different format, and we might only realise the consequences once we have finished labelling all the images, by which time it would be irreversible.

Once we have started to annotate the images, if at some point we are dissatisfied with the label or the bounding box, we can delete the particular bounding box by using the “Delete RectBox” option. Once we have completed the annotations for one image, we can click “Save” to save the annotation before proceeding to label the next image.

3.4 Saving each annotation in YOLO format

In this section, we want to understand what happens when we are labelling an image and also take a look at how YOLO expects a standard annotation to be. This will help us in debugging, in case we run into any problems. When we are working with a dataset, we will have a set of classes of objects that we will want to label – this is stored in the directory of the dataset as a text file, often named “classes.txt”. For each annotated image in the dataset, YOLO expects a text file containing information about the class of object, and its location in the image. This information is structured, and presented as follows:

```
<class> <bbox-x> <bbox-y> <bbox-width> <bbox-height>
```

The <class> field is an integer corresponding to the position of the particular labelled class in the classes.txt file, and it is set to 0 by default if we have only one class of objects to annotate. The fields <bbox-x> and <bbox-y> correspond to the x and y coordinates of the bounding box, which we create using the option “Create RectBox” as described in section 3.3 above. The <bbox-width> and <bbox-fields> are populated when we let go of the mouse and finalise the perimeter of the bounding box. In order to save the annotation, we can either save each annotation by hand, by using the “Save” option, or we can enable the “Auto Save mode” available under the “View” menu. This will save us from the hassle of saving each annotation separately, and help

speed up our workflow. The options relevant to saving the annotation are highlighted in Figure below.

Figure 7: Saving annotations in labelImg

Before saving the annotations, please make sure that you have selected the proper annotation export format. With this we have covered all the required steps for annotating and labelling the images. We can repeat this process until all the images in the dataset have been successfully annotated.

4 Appendix

In this section we will highlight a few of the important keyboard hotkeys which will help accelerate the annotation process while working with large datasets:

Table 1: Useful keyboard bindings for working in labelImg

Keyboard Binding	Function
Ctrl + U	Load all images from a directory
Ctrl + R	Change the default annotation of a target directory
Ctrl + S	Save the annotation
Space	Flag the current image as verified
W	Create a new RectBox
A	Move to previous image
D	Move to the next image

4.1 Annotation of blurry and partially blurry images

In the process of extraction of the images from the video streams recorded at the vineyards, we have observed that some images turn out to be a little blurry. In order to gain maximum accuracy from our model, we need to make sure we proper a well-labelled dataset, where each object is clearly defined, and the bounding boxes are drawn as close as possible. As a result, we need to weed out such blurry images. We will present a few examples of images and describe the process of eliminating the selected images from the dataset.

Figure 8: A partially blurry image

In Figure 8, we can see that the image is partially blurry, however we can still clearly identify the distinct objects at the foreground of the image. So, in such a case we will call the image partially blurry, and we will annotate only the objects that are clearly visible *(in this case, it was the two trees in the front)*.

Another possibility that might occur is that the image might be extremely blurred, to such an extent where if it were to be passed as a training input to the model, it would affect the accuracy negatively. We would be actively discarding such images. Figure 9 below shows an example from such an image.

Figure 9: A blurry image

As we can see, if we were to label and pass these images on to the Deep Learning model, we can expect the overall accuracy to go down. The main idea while deselecting images for annotation is to

be biased against partially and completely blurry images – the general idea should be to weed out even partially blurry images. However, if the size of the dataset is small, a few partially blurry images may be allowed.