Data Conversion Documentation

Benjamin Nelson, Xingzi Xu, Logan Caraway

March 2021

Introduction

This document will outline the general workflow for creating labels for video files, converting those videos to a set of images and image labels and then splitting the data into testing and training sets.

Labeling Videos Using MATLAB

For video labeling, we found that the MATLAB video labeler was a relatively easy way to generate labels for video files. To use the video labeler, you will need to download the Computer Vision Toolbox. This can be done by finding the Add-Ons tab under the Home section in the toolbar. See figure one for the location of the Add-Ons tab. Click the drop-down box and select "Get Add-Ons". This will open the Add-on explorer. From here, simply search for "Computer Vision Toolbox" and install the MathWorks authored add-on with the same name. Once the requisite Add-Ons have been

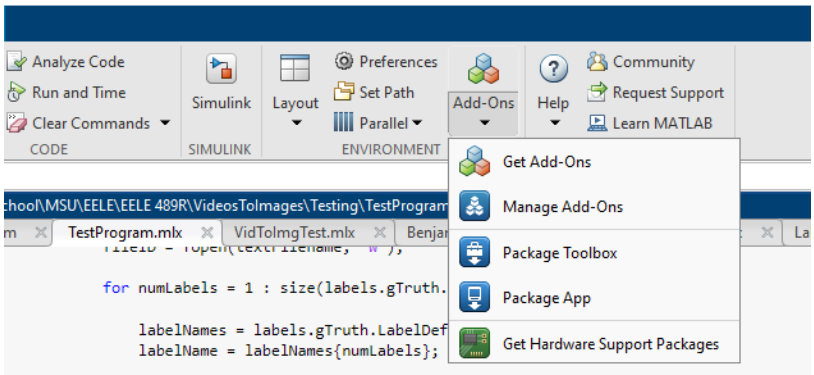

Figure 1: Location of the Add-Ons tab in MATLAB

installed type `videoLabeler()` into the MATLAB command window. This will open the Video Labeler GUI. A tutorial on this application can be found at https://www.mathworks.com/help/vision/ug/get-started-with-the-video-labeler.html. A tutorial video will also be included in the documentation.

```
0 Red_Buoy
1 Yellow_Buoy
2 Green_Buoy
3 Bin
4 Gate
5 Torpedo
6 Blue_Buoy
```

scale[h] .

Figure 2: Format of classes.cfg file

Converting Video Data to Image Data

After labeling all of the video data we then need to convert the labels from the proprietary `gTruth` MATLAB data structure, into a generic CSV file format. To do this, you will first need to create a text file called "classes.cfg". This file will contain the name of each object class and their corresponding label IDs. See figure 2 for an example of how this file should look. Each number is the label ID for the following object class. Note that the label IDs start at 0. Add as many classes as is necessary for your project. Next we need to set-up the file directory for the conversion program to work correctly. First we need to create a directory to store all of the data. If you want, you can create sub-directories to store the data for a specific class (i.e. a directory called BuoyData to store buoy information), however with the current version of the the YOLOv4 network, it is easiest to have all data in a single data folder (IMPORTANT - ALL IMAGE DATA MUST BE STORED IN A DIRECTORY NAMED "DATA"). Figure 3 shows an example of a folder set-up. Here we can see

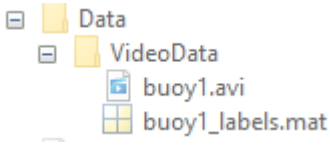

Figure 3: An example of the folder structure that should be used when running the data conversion program.

the video (buoy1.avi) and the file containing the labels (buoy1_labels.mat). Next, set up the rest of the directory according to the example in figure 4.

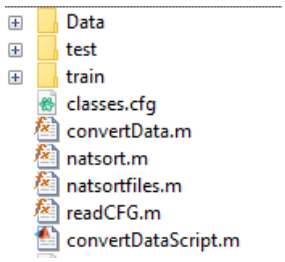

Figure 4: The directory that contains the "Data" folder should also contain every other file shown here.

Next, you just need to change the current PATH in MATLAB to the folder containing Convert-Data.m, and once that has been done just run the program in MALTAB. You should see that a new folder has appeared in the data folder. This folder contains the label information for each frame of each video in the VideoData folder. Figure 5 shows the output data after running the script. Not shown in this image are the text files associated with each jpeg image.

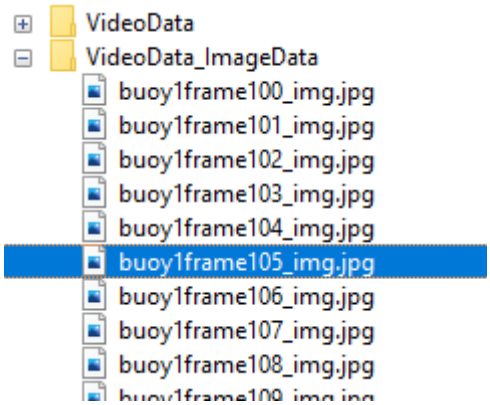

Figure 5: The results after running the conversion program with a single video+label file.

Splitting the data into a Testing and Training Set

The last step is to split the data into training and testing sets. To do this you simply need to run the "generateTest.py" script, while supplying it the correct parameters. First we need to tell the script where we want the training and testing sets to be stored. Figure 6 shows the code snippet that sets these parameters.

```
# Define the base file path where you want to testing and training data set to be stored
base_filepath = "E:\School\MSU\EELE\EELE 489R\Documentation\Testing\\"

# Define the directories for the training data set (obj) and testing set
obj_path = base_filepath + "train"
test_path = base_filepath + "test"
```

Figure 6: Code that tells python where to save the training and testing data.

The base_filepath variable tells the script the which directory we want to create the new data sets in, and the obj_path and test_path variables dictate what the names of the new folders will be. For example, changing "train" to "training" will result in this script storing the training data set at the file path "base\filepath\training\". Next we need to tell the script the file where all the image data is located. In my case it is located at the path "E:\School\MSU\EELE\EELE 489R\VideosToImages\Testing\Data\VideoData_ImageData\". Figure 7 shows the code that stores the path to the image data.

```
# Define where all the images and labels are stored on the local machine and create a list of all files contained
# in that directory
cur_data_path = base_filepath + "Data\\VideoData_ImageData"
```

Figure 7: Code that tells python were the image and label data is stored on the local machine.

Simply change the value stored in the cur_data_path variable to where ever the image data is stored on the local machine. Once the changes above have been made, simply run the script. You should now see that there are two new files at the location specified by the base_filepath variable each containing either the testing set or the training set.