Create datastores from NightOwls dataset

This document walks through the process of creating datastores (see: https://www.mathworks.com/help/matlab/datastore.html) from the annotations and images included in the NightOwls dataset - https://www.nightowls-dataset.org/ so that the data is in a format suitable to train object detection networks using the Deep Learning Toolbox and MATLAB. When the data is downloaded separately from this API, please point `annotationFile` to the folder where the annotation JSON files are, and `imageRoot` to the folder where the image data is.

```
% Specify locations where the annotations and images are.
clear
annotationFile = "C:\Users\akamath\Documents\data\nightowls\data-nightowls\nightowls_validation
imageRoot = "C:\Users\akamath\Documents\data\nightowls\data-nightowls\nightowls_validation\nigh
```

Here is how the annotation file can be read and parsed through - to access all the bounding boxes and any other information contained there. Specifically, there are image file names, and associated annotations tagged with 'IDs'. These are matched and only specific categories can be read - for example, the "pedestrian" category is something that we can select and include.

```
% Read JSON data from the annotation file into a structure.
text = fileread(annotationFile);
data = jsondecode(text);
```

Create a table out of the image file names.

```
dataTable = table;
dataTable.imageFileNames = string({data.images(:).file_name}');
```

Make placeholder for categories of boxes.

```
for idx = 1:numel(data.categories)
    % Selectively choose only pedestrian - change to generalize.
    if lower(string(data.categories(idx).name)) == "pedestrian"
        dataTable.(data.categories(idx).name) = repmat({[]}, height(dataTable), 1);
    end
end
```

Fill in the annotations from the annotations structure - these are connected using IDs - each annotation has an image_id associated to it, and each image has an ID that can be used to make this association.

```
for idx = 1:height(dataTable)
    if mod(idx, 1000) == 0
        fprintf("Processing %d out of %d annotations\n", idx, height(dataTable));
    end
    framesWithAnnotations = find([data.annotations(:).image_id] == data.images(idx).id);
    for jdx = 1:numel(framesWithAnnotations)
        annotIdx = framesWithAnnotations(jdx);
        dataTable.("pedestrian"){idx} = [dataTable.("pedestrian"){idx}; data.annotations(annotI
    end
end
```

```
Processing 1000 out of 51848 annotations
```

```
Processing 2000 out of 51848 annotations
Processing 3000 out of 51848 annotations
Processing 4000 out of 51848 annotations
Processing 5000 out of 51848 annotations
Processing 6000 out of 51848 annotations
Processing 7000 out of 51848 annotations
Processing 8000 out of 51848 annotations
Processing 9000 out of 51848 annotations
Processing 10000 out of 51848 annotations
Processing 11000 out of 51848 annotations
Processing 12000 out of 51848 annotations
Processing 13000 out of 51848 annotations
Processing 14000 out of 51848 annotations
Processing 15000 out of 51848 annotations
Processing 16000 out of 51848 annotations
Processing 17000 out of 51848 annotations
Processing 18000 out of 51848 annotations
Processing 19000 out of 51848 annotations
Processing 20000 out of 51848 annotations
Processing 21000 out of 51848 annotations
Processing 22000 out of 51848 annotations
Processing 23000 out of 51848 annotations
Processing 24000 out of 51848 annotations
Processing 25000 out of 51848 annotations
Processing 26000 out of 51848 annotations
Processing 27000 out of 51848 annotations
Processing 28000 out of 51848 annotations
Processing 29000 out of 51848 annotations
Processing 30000 out of 51848 annotations
Processing 31000 out of 51848 annotations
Processing 32000 out of 51848 annotations
Processing 33000 out of 51848 annotations
Processing 34000 out of 51848 annotations
Processing 35000 out of 51848 annotations
Processing 36000 out of 51848 annotations
Processing 37000 out of 51848 annotations
Processing 38000 out of 51848 annotations
Processing 39000 out of 51848 annotations
Processing 40000 out of 51848 annotations
Processing 41000 out of 51848 annotations
Processing 42000 out of 51848 annotations
Processing 43000 out of 51848 annotations
Processing 44000 out of 51848 annotations
Processing 45000 out of 51848 annotations
Processing 46000 out of 51848 annotations
Processing 47000 out of 51848 annotations
Processing 48000 out of 51848 annotations
Processing 49000 out of 51848 annotations
Processing 50000 out of 51848 annotations
Processing 51000 out of 51848 annotations
```

Here, we append the folder names to the image files listed in the annotation JSON file, so that MATLAB can find it and create a datastore out of. A datastore is

```
% Add the full path to the image data, and create a datastore using this.
dataTable.imageFileNames = fullfile(imageRoot, dataTable.imageFileNames);
imds = imageDatastore(dataTable.imageFileNames');
```

If we are using this data for training, we can do the following: Using the pedestrian data from the dataTable, we can also create something called a boxLabelDatastore - so that all the bounding boxes can be put together with the images and mini-batches could be constructed from the combination of these for training.

```
bxds = boxLabelDatastore(table(dataTable.pedestrian));

% Then combine the two, run any transforms necessary for augmentation,
imbx = combine(imds, bxds);
```

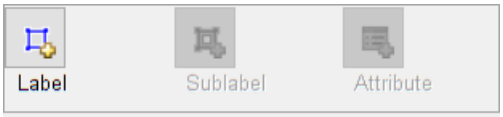
imbx can be used directly in functions like trainSSDObjectDetector or trainYOLOv2ObjectDetector if you are considering using these architectures, or, with trainNetwork or a custom training loop for other customized detector architectures.

If we are instead attempting to visualize or maybe even make minor corrections to annotations in the dataset (if they're found to be incorrect), here's how we can do so using interactive applications:

```
% Specify where the images are.
dataSource = groundTruthDataSource(dataTable.imageFileNames);

% Create 'label definitions' - to hold metadata about the labels itself.
ldc = labelDefinitionCreator();
addLabel(ldc, "pedestrian", labelType.Rectangle);
labelDefs = create(ldc);

% Add the annotations itself - choose all but the imageFileName column.
labelData = dataTable(:, 2:end);

gTruth = groundTruth(dataSource, labelDefs, labelData);
```

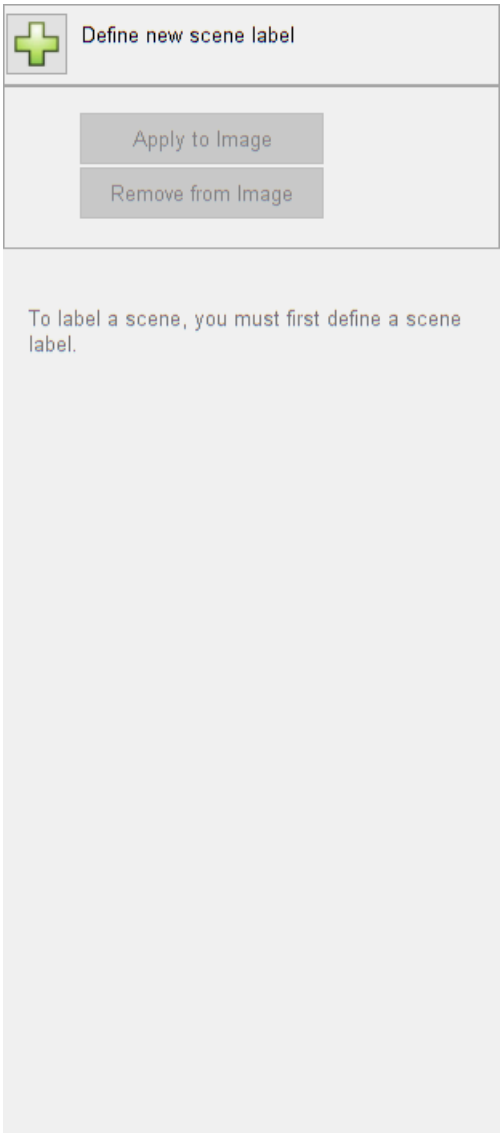
And then, one can open this in a labeler App using:

```
imageLabeler; % Load gTruth using 'load labels'
```

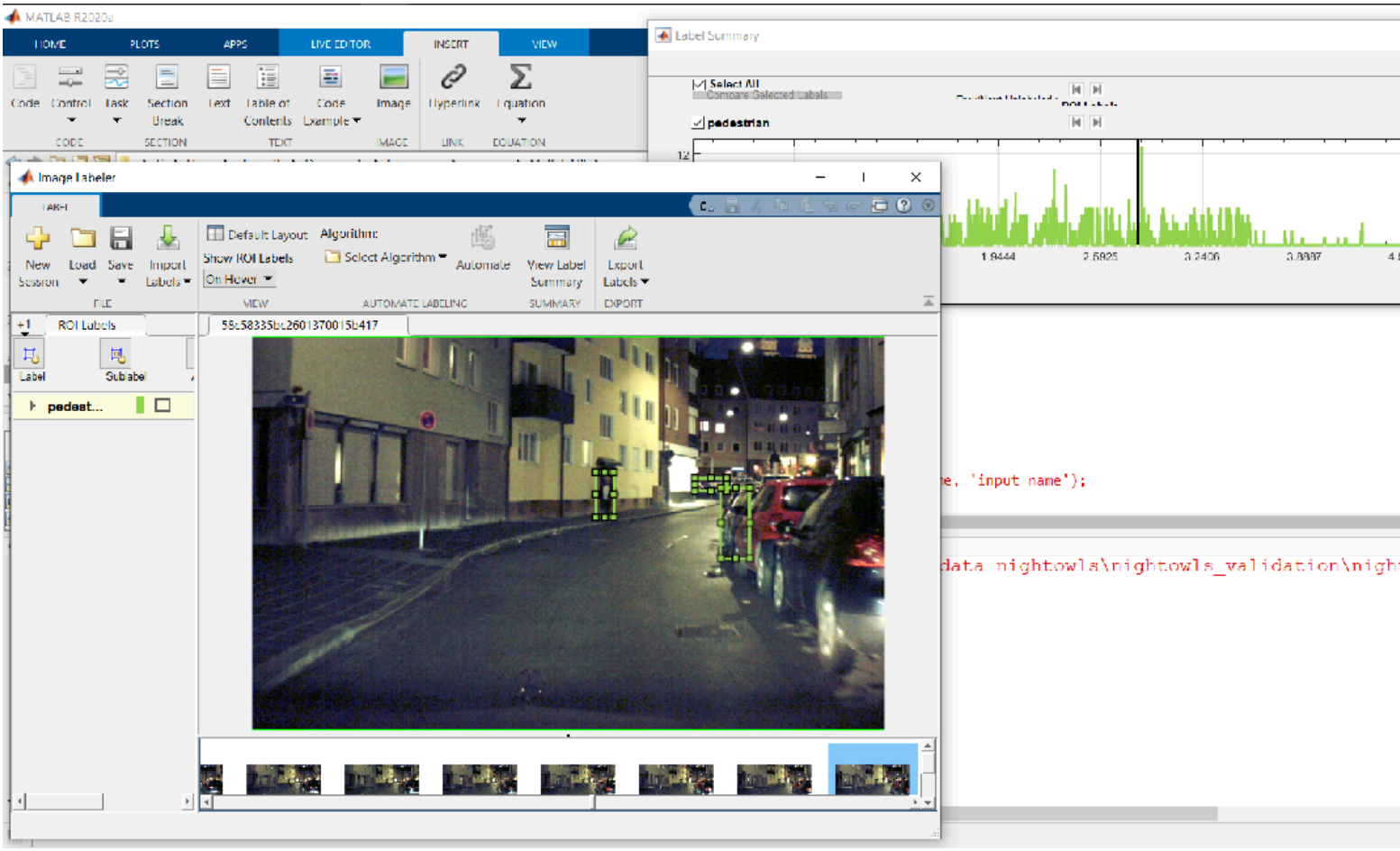


To label an ROI, you must first define one or more of the following label types:

- Rectangle label
- Line Label
- Pixel label



For example, here's a screenshot of detecting a potential labeling error (see the right-most label between the two cars) - there's no pedestrian labeled there in the previous or next frames - and so it looks slightly suspicious.


MATLAB R2020a
HOME
PLOTS
APPS
LIVE EDITOR
INSERT
VIEW
Code
Control
Task
Section Break
Text
Table of Contents
Code Example
Image
Hyperlink
Equation
Label Summary
Select All
Compare Selected Labels
pedestrian
Image Labeler
New Session
Load
Save
Import Labels
Default Layout
Show ROI Labels
On Hover
Algorithm
Select Algorithm
Automate
View Label Summary
Export Labels
ROI Labels
Label
Sublabel
pedest...
'input name');
data nightowls\nightowls_validation\nigh