Wood Knots Detection Project

Use this script to complete the wood knots detection project. Follow along with each step and return to Coursera to answer quiz questions when prompted.

Loading the Ground Truth Data

The ground truth for the training images in the wood knots dataset is stored in a ".mat" file in the course files. To access the ground truth data, you'll need to load it and then update the image file locations to match your system. To do this, run the following section of code.

```
load WoodKnotsGroundTruth
```

Overwrite the image file locations stored in the ground truth variable to match your system.

```
testPath = overwriteGTruthLocations(gTruthTrain);
```

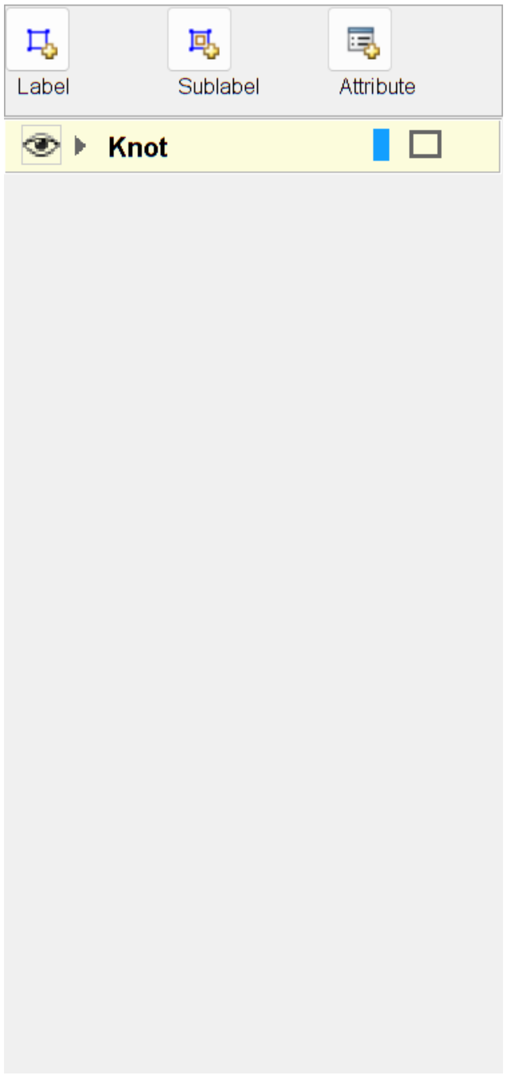
```
All the file paths are resolved in this groundTruth object.
```

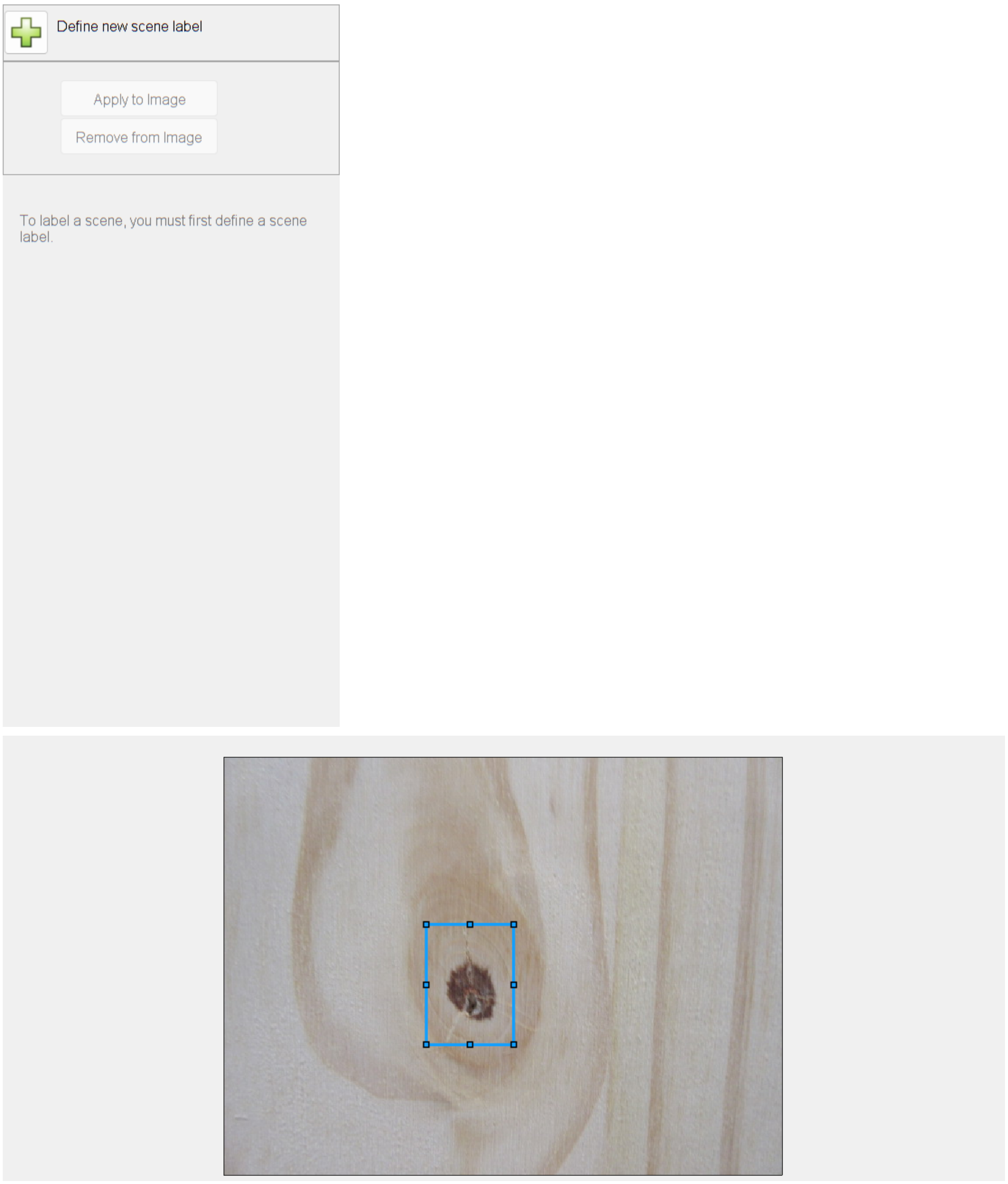
Step 1: Prepare the Dataset

First, familiarize yourself with the wood knots dataset.

- Run the line below to load the training images and ground truth labels into the Image Labeler app.
- Then review the ground truth bounding boxes to guide you for the next task. Note how bounding boxes are drawn relatively tight around the knots, with minimal extra area included.

```
imageLabeler(gTruthTrain)
```


Label
Sublabel
Attribute
Knot

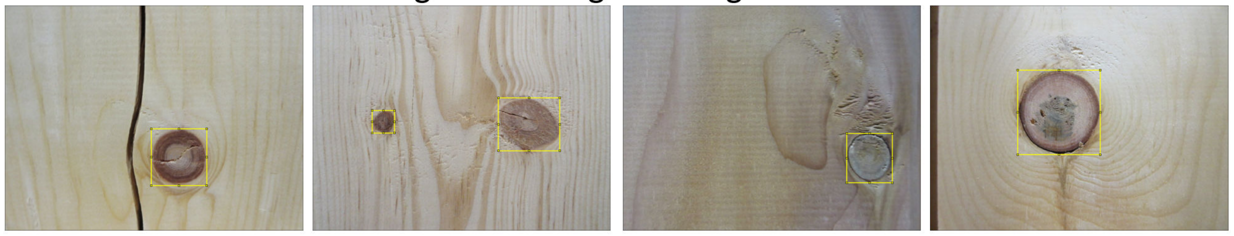


To label a scene, you must first define a scene label.

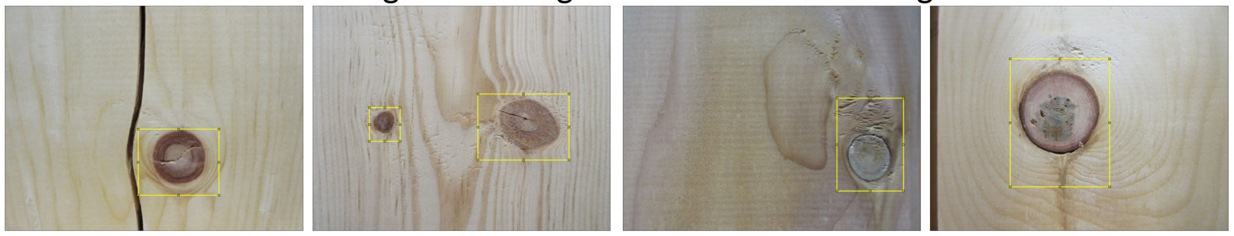
The course files also include 10 test images that have not been labeled. You'll need to create the ground truth for these images so you can evaluate your trained model later.

- Run the line below to load the unlabeled test images into the Image Labeler app.
- Use the Image Labeler app to create the ground truth labels for the knots. Make sure to match the training images for how bounding boxes are drawn around knots. Some examples of correct and incorrect labeling are shown here:

Correct labeling – bounding boxes tight around the knots

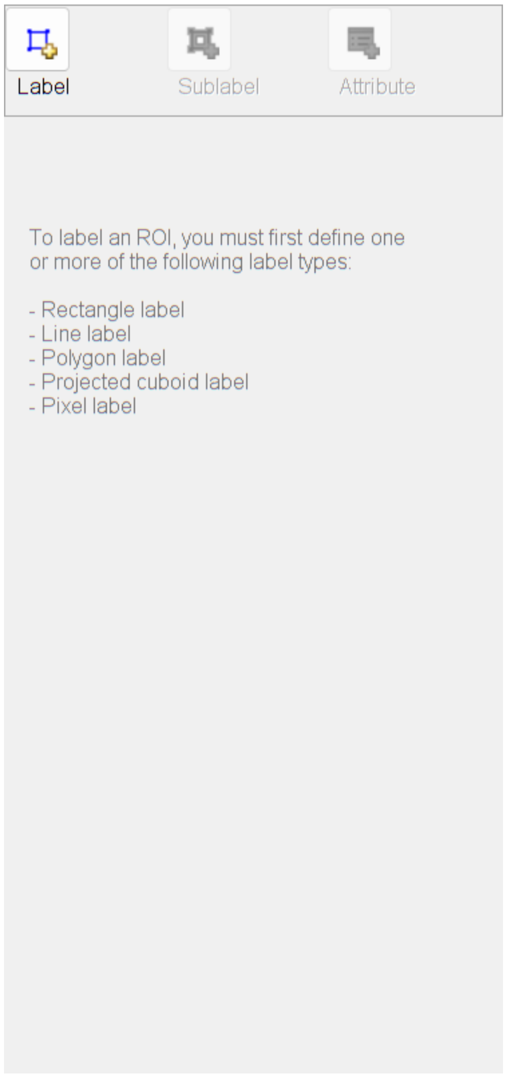


Incorrect labeling – bounding boxes loose or including extra area



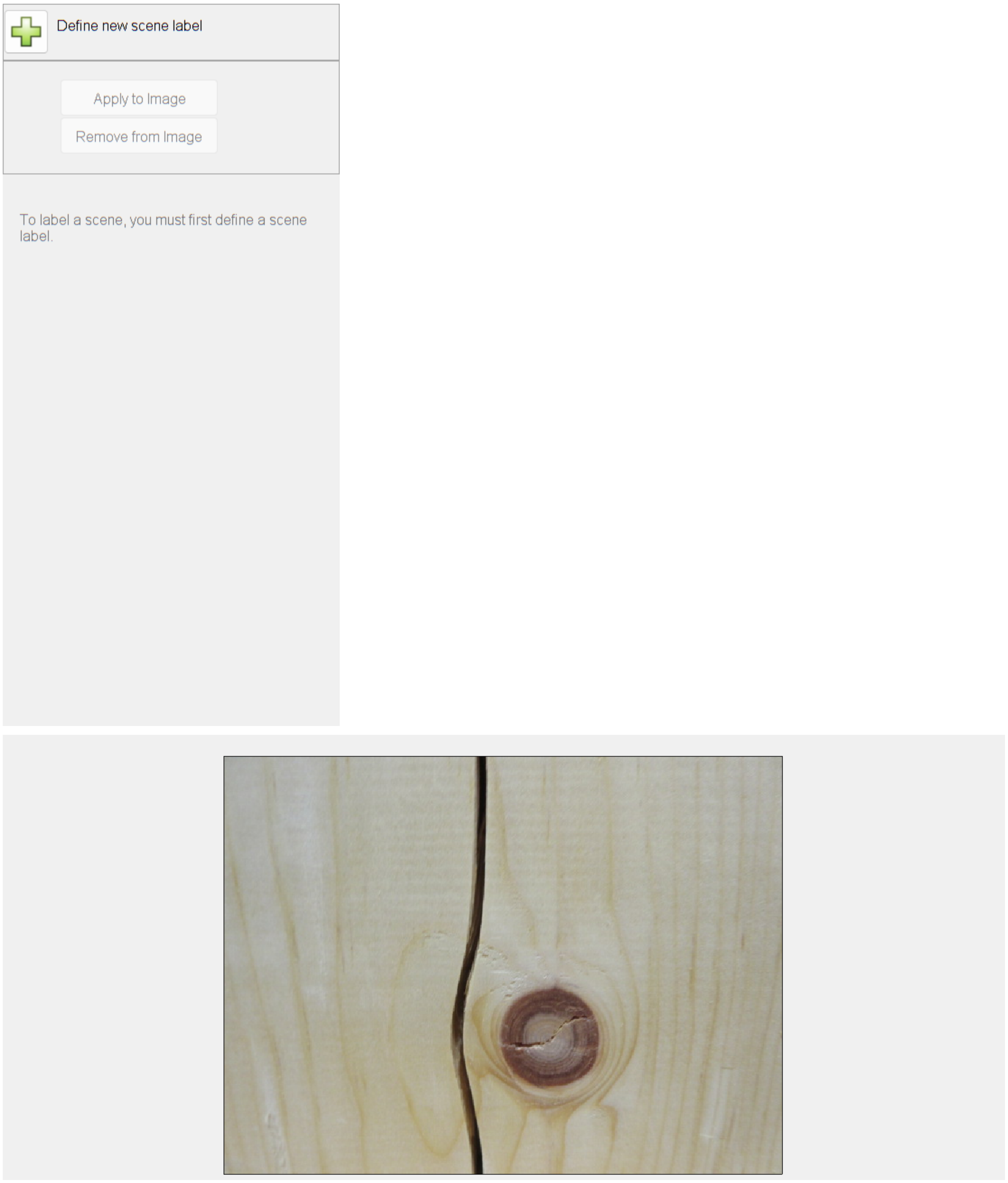
- Export the ground truth as a mat file named `WoodKnotsGroundTruthTest.mat`.

```
imageLabeler(testPath)
```

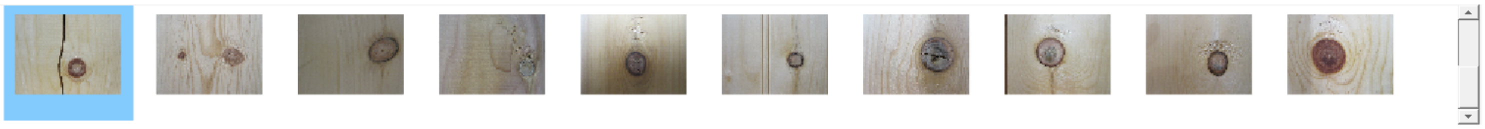


To label an ROI, you must first define one or more of the following label types:

- Rectangle label
- Line label
- Polygon label
- Projected cuboid label
- Pixel label



To label a scene, you must first define a scene label.



To evaluate your detector's performance later, you'll need to compare its results to this test set ground truth.

- Run the code below to load the ground truth you just saved into your MATLAB workspace.
- If you get an error here, make sure the ".mat" file is named correctly and is located in the MATLAB search path.

```
load WoodKnotsGroundTruthTest.mat
```

You'll also need to load the test images as an image datastore before running them through your trained detector.

```
imdsTest = imageDatastore(testPath);
```

Well done! Before proceeding to the next step:

- Take the quiz "Project: Wood Knots Detection Step 1".

Question 1

There are 90 images in the **training** set. How many of these images in the **training** set have more than one wood knot?

Ans : 12

To proceed with the project, you must first confirm the knots you labeled in the test set. What is the **height** in pixels of the bounding box for image "IMG_3781.JPG" in the **test** dataset? (Note: this question accepts a range of values.)

The coordinates for each bounding box are found in your ground truth variable for the test images. These are stored as an array of 4 numbers: [xmin, ymin, width, height].

Ans : 106

Step 2: Train and Evaluate a Detector

Next you'll use the training ground truth we've provided, `gTruthTrain`, to train an ACF object detector.

- Fill in and run the code to train the detector here.
- Use the default settings for the ACF detector.
- If you need help, you can reference the script `ObjectDetectionWithML.mlx`.

```
% Fill in your code here
```

```
objectTrainingData = objectDetectorTrainingData(gTruthTrain);
acfDectector = trainACFObjectDetector(objectTrainingData)
```

```
ACF Object Detector Training
The training will take 4 stages. The model size is 30x28.
Sample positive examples(~100% Completed)
Compute approximation coefficients...Completed.
Compute aggregated channel features...Completed.
--------------------------------------------
Stage 1:
Sample negative examples(~100% Completed)
Compute aggregated channel features...Completed.
Train classifier with 103 positive examples and 515 negative examples...Completed.
The trained classifier has 30 weak learners.
--------------------------------------------
Stage 2:
Sample negative examples(~100% Completed)
Found 66 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 103 positive examples and 515 negative examples...Completed.
The trained classifier has 80 weak learners.
--------------------------------------------
Stage 3:
Sample negative examples(~100% Completed)
Found 21 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 103 positive examples and 515 negative examples...Completed.
The trained classifier has 107 weak learners.
--------------------------------------------
Stage 4:
Sample negative examples(~100% Completed)
Found 10 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 103 positive examples and 515 negative examples...Completed.
The trained classifier has 126 weak learners.
--------------------------------------------
ACF object detector training is completed. Elapsed time is 12.7042 seconds.
acfDectector = 
  acfObjectDetector with properties:

             ModelName: 'Knot'
    ObjectTrainingSize: [30 28]
       NumWeakLearners: 126
```

Now that your detector is trained, use it to detect the wood knots on the test set.

- Fill in and run the code to apply the detector using the default settings of the `detect` function.
- Remember, the `detect` function can accept the test datastore variable, `imdsTest`, to run on all the images in the datastore.

```
% Fill in your code here

bboxes = detect(acfDectector,imdsTest)
```

bboxes = 10×2 table

	Boxes	Scores
1	[310,254,135,145]	149.6352

	Boxes	Scores
2	[128,225,47,51]	105.3154
3	[427,113,187,200]	108.4443
4	[480,265,102,109]	99.1845
5	[272,216,135,145]	126.5182
6	[393,222,102,109]	143.5581
7	[320,136,224,240;352,72,...	[130.4411;42....
8	[160,87,187,200;187,140,...	[31.0064;157....
9	[381,227,121,129]	122.0053
10	[135,111,238,255]	133.1785

Finally, visualize your detection results and calculate the detection metrics miss rate.

- Fill in and run the code to do this here.
- Remember, if you need help, you can reference the script `ObjectDetectionWithML.mlx`

```
% Fill in your code here

evaluateDetectionMissRate(bboxes,gTruth.LabelData)
```

```
ans = 0.0909
```

```
evaluateDetectionPrecision(bboxes,gTruth.LabelData)
```

```
ans = 0.9091
```

You should see results that could use some improvement. Don't worry; you'll fix those soon.

- Take the quiz "Project: Wood Knots Detection Step 2".

Question 1

Look at the detection scores for your trained ACF detector on the test data. They generally fall between what range of values?

Ans : Between 0 and 200

Question 2

What is the value of the **miss rate** metric for your trained detector on the test dataset?

Ans: 0.0909

Question 3

How could you modify the detect function to correctly detect this knot using the ACF detector you've already trained?

Ans: Decrease the threshold value at which a detection is marked as true.

Step 3: Improving the Detections

Now that you have a trained detector, your next step is to improve it.

- Based on the last question in the quiz "Project: Wood Knots Detection Step 2", modify the `detect` function to capture the missing detection. Specifically, lower the `"threshold"` value below the default of -1.
- Fill in and run the code here to again: visualize your detection results and calculate the detection miss rate.

```
% Fill in your code here

objectTrainingData1 = objectDetectorTrainingData(gTruthTrain);
acfDectector1 = trainACFObjectDetector(objectTrainingData1)
```

```
ACF Object Detector Training
The training will take 4 stages. The model size is 30x28.
Sample positive examples(~100% Completed)
Compute approximation coefficients...Completed.
Compute aggregated channel features...Completed.
--------------------------------------------
Stage 1:
Sample negative examples(~100% Completed)
Compute aggregated channel features...Completed.
Train classifier with 103 positive examples and 515 negative examples...Completed.
The trained classifier has 30 weak learners.
--------------------------------------------
Stage 2:
Sample negative examples(~100% Completed)
Found 66 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 103 positive examples and 515 negative examples...Completed.
The trained classifier has 80 weak learners.
--------------------------------------------
Stage 3:
Sample negative examples(~100% Completed)
Found 21 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 103 positive examples and 515 negative examples...Completed.
The trained classifier has 107 weak learners.
--------------------------------------------
Stage 4:
Sample negative examples(~100% Completed)
Found 10 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 103 positive examples and 515 negative examples...Completed.
The trained classifier has 126 weak learners.
--------------------------------------------
ACF object detector training is completed. Elapsed time is 10.9097 seconds.
acfDectector1 =
  acfObjectDetector with properties:

             ModelName: 'Knot'
    ObjectTrainingSize: [30 28]
       NumWeakLearners: 126
```

```
bboxes1 = detect(acfDectector1,imdsTest,"Threshold",-3);
```

- Take the quiz "Project: Wood Knots Detection Step 3".

```
evaluateDetectionMissRate(bboxes1,gTruth.LabelData)
```

```
ans = 0
```

```
evaluateDetectionPrecision(bboxes1,gTruth.LabelData)
```

```
ans = 0.9860
```

Extra Credit: Removing Redundant Detections

Use this last section to modify your detection results. See if you can merge the overlapping, redundant detections. Hint: use the `selectStrongestBbox` function. You can find the documentation page for this function here.

```
% Fill in your code here
```

Helper Function

This function overwrites the image file locations stored in the ground truth variable to match your system.

```
function testPath = overwriteGTruthLocations(gTruthTrain)
% Find the location of the image files
```

```
trainPath = fileparts(which("IMG_3177.JPG"));
if isempty(trainPath)
    error("The course files are not on the MATLAB search path." + ...
        "Follow the instructions when downloading the course " + ...
        "files to add them to your path.")
end

% Overwrite locations for images
currentPath = string(fileparts(gTruthTrain.DataSource.Source{1}));
changeFilePaths(gTruthTrain,[currentPath, trainPath]);

% Save the test path
testPath = erase(trainPath,'Train') + "Test";
end
```