Computer Vision
Final Project Report

Muhammed Okumuş

15044017

The Data

The data set is called RealSR V1[kaggle]. It consists of high and low resolution images. Only high resolution images are used for training and testing the SVM models.

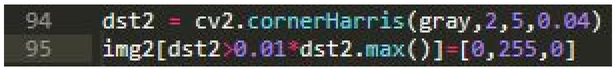
Pre-Processing

Processing images at high resolution is very costly so different classifiers are trained using different resolutions ranging from 100x100 pixels to 400x400 pixels. This way we can compare the effect of image resolution on the classifiers accuracy.

After resizing the training data Harris Corner Detection routine is performed on each image. And 2 matrices are generated for Sobel X axis and Y axis.

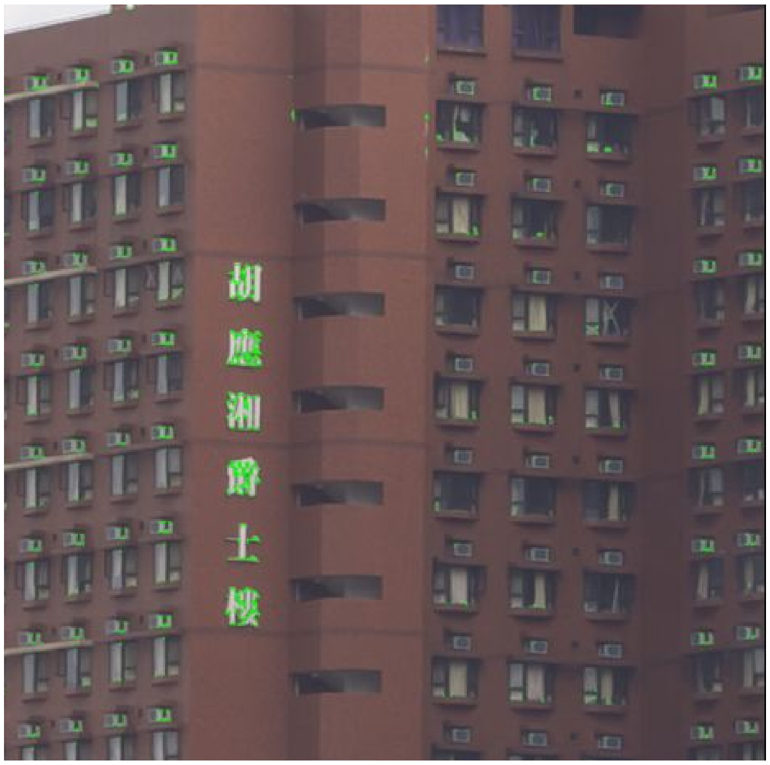
Modelling The Data Returned From Harris Detector

Harris detector returns a NxM sized matrix for NxM sized image. Each point in this matrix has a confidence value, higher the value is more likely to be a corner. So by typing:



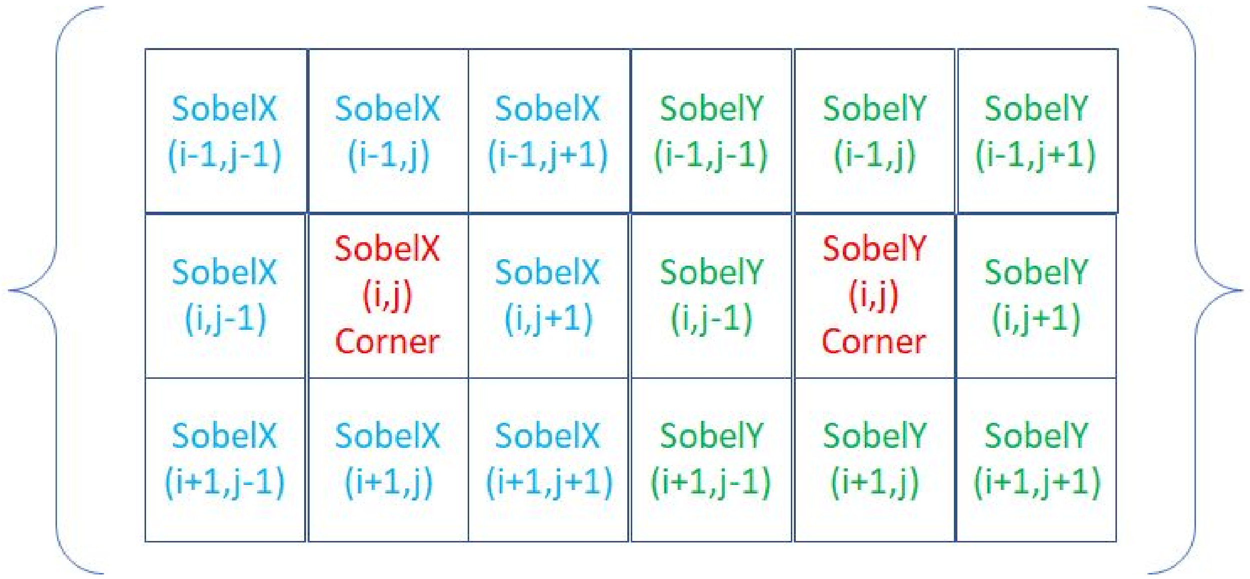
```
dst2 = cv2.cornerHarris(gray,2,5,0.04)
img2[dst2>0.01*dst2.max()]=[0,255,0]
```

We mark the pixel as corner if the confidence is higher than 1% of the highest confidence.



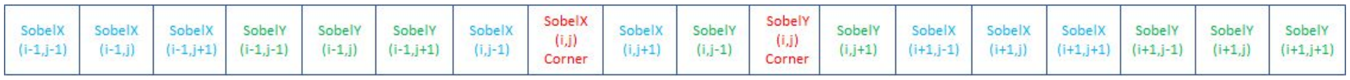
Using Sobel Gradient To Create Groups

After getting the result from Harris Detector, we loop through the matrix it returned and if the confidence value is above the threshold we decided, the pixel corresponding to that corner and 1 pixel around that corner are used for creating the model. It’s simply a 3x3 block in which the middle element is the detected corner. We apply both sobel for both the X and Y axis so the resulting model looks like this before flattening it to 1 dimension.



SobelX (i-1,j-1)	SobelX (i-1,j)	SobelX (i-1,j+1)	SobelY (i-1,j-1)	SobelY (i-1,j)	SobelY (i-1,j+1)
SobelX (i,j-1)	SobelX (i,j) Corner	SobelX (i,j+1)	SobelY (i,j-1)	SobelY (i,j) Corner	SobelY (i,j+1)
SobelX (i+1,j-1)	SobelX (i+1,j)	SobelX (i+1,j+1)	SobelY (i+1,j-1)	SobelY (i+1,j)	SobelY (i+1,j+1)

After flattening



SobelX (i-1,j-1)	SobelX (i-1,j)	SobelX (i-1,j+1)	SobelY (i-1,j-1)	SobelY (i-1,j)	SobelY (i-1,j+1)	SobelX (i,j-1)	SobelX (i,j) Corner	SobelX (i,j+1)	SobelY (i,j-1)	SobelY (i,j) Corner	SobelY (i,j+1)	SobelX (i+1,j-1)	SobelX (i+1,j)	SobelX (i+1,j+1)	SobelY (i+1,j-1)	SobelY (i+1,j)	SobelY (i+1,j+1)

We do this for each corner and non-corner pixels. Corner groups are labeled as class 1(positive meanwhile non-corner groups are labeled as class -1(negative).
Corner threshold is variable and tested for 1%, 3% and 5% confidence corners for different classifiers.

The Classifiers

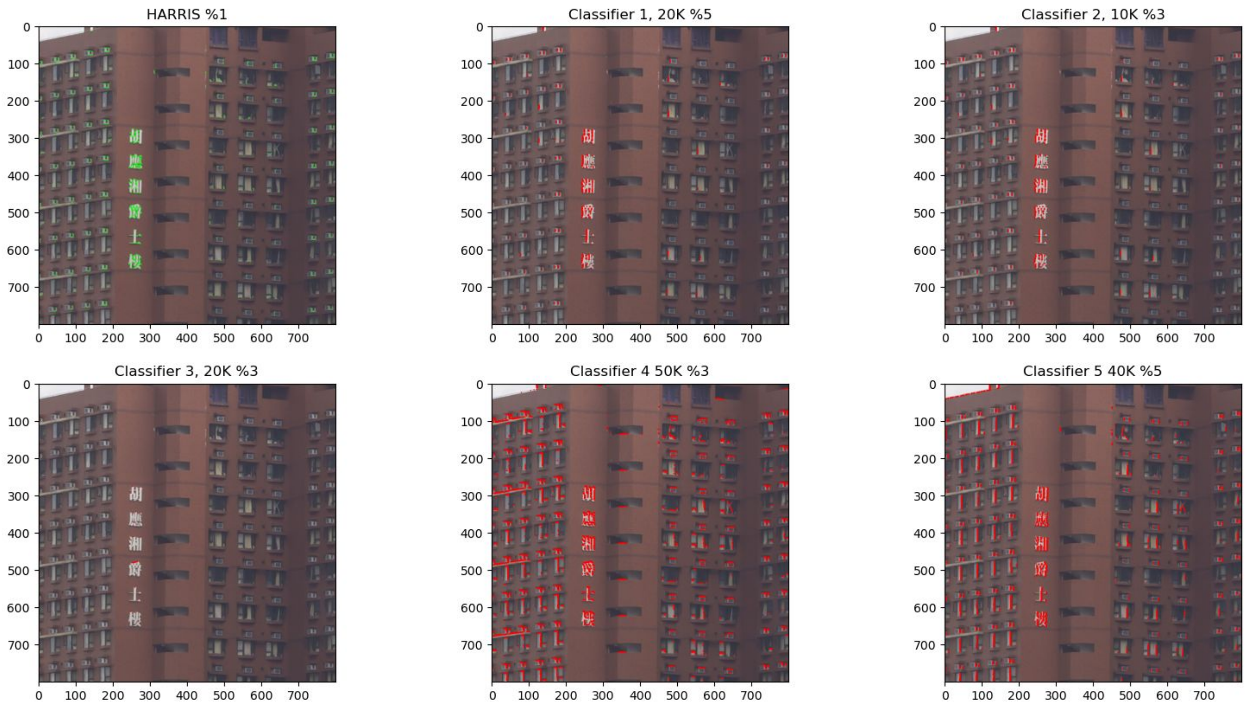
There are multiple classifiers trained from random corners found across 207 images in the training set. There are a total of 30.576.146 corners.

- Classifier 1, 20.000 corners, trained by 200x200px resolution images with %5 confidence
- Classifier 2, 10.000 corners, trained by 100x100px resolution images with %3 confidence
- Classifier 3, 20.000 corners, trained by 100x100px resolution images with %3 confidence
- Classifier 4, 50.000 corners, trained by 150x150px resolution images with %3 confidence
- Classifier 5, 40.000 corners, trained by 400x400px resolution images with %5 confidence

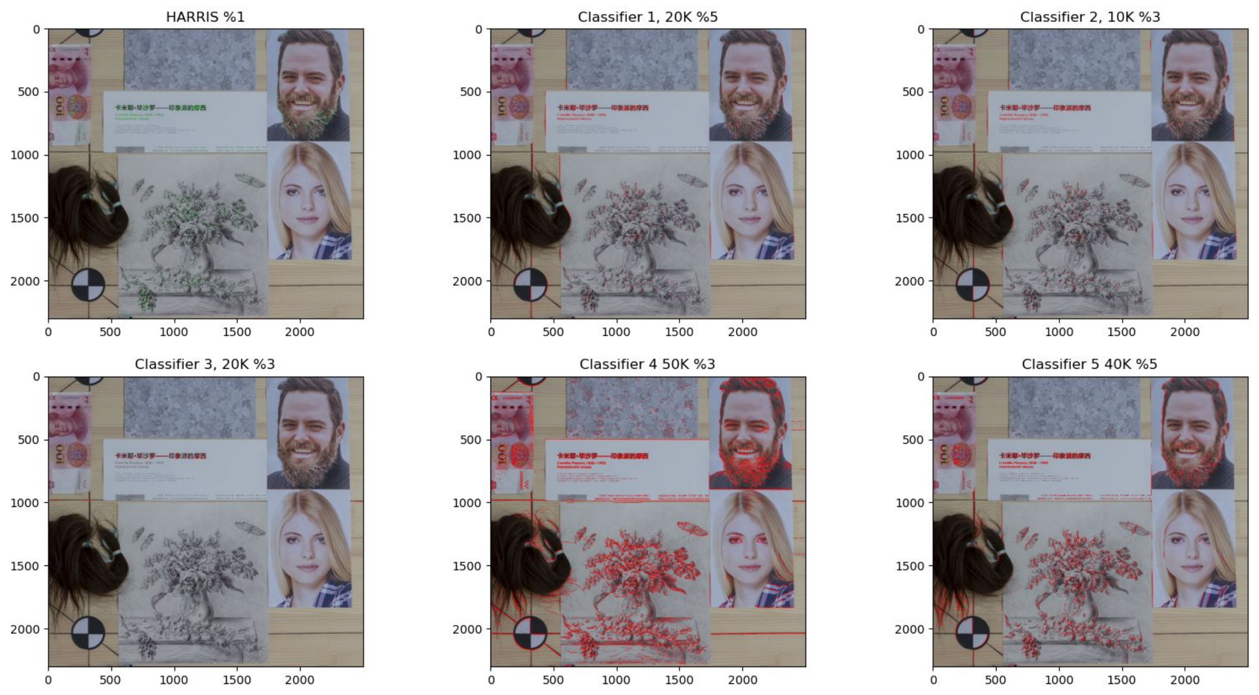
Testing Data

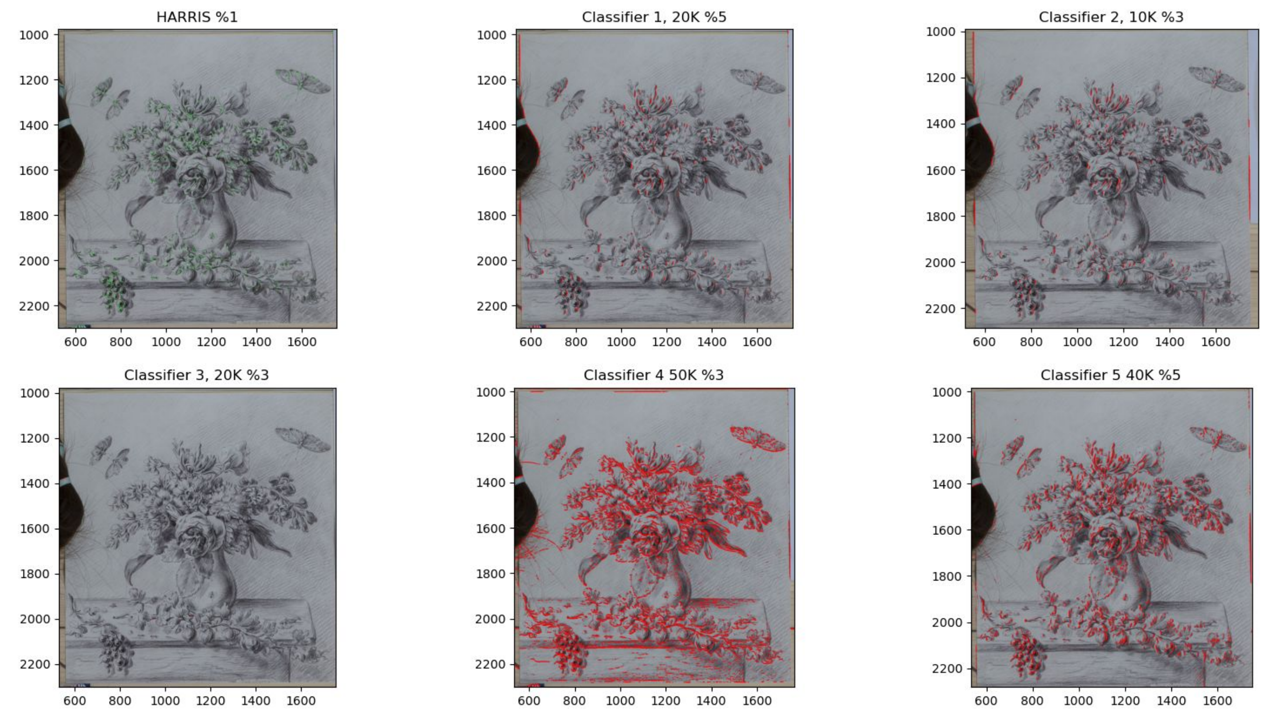
Testing data ranges from 800x800 px and 2300x2600px PNG images. They are not resized and comparison is done versus Harris Detector with %1 confidence corners. Max resolution images can be viewed under the project/results folder.

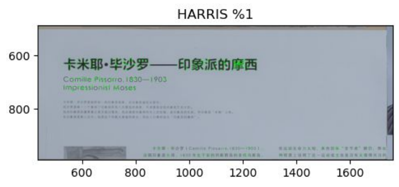
Test 1(Canon_001_HR, 800x800)

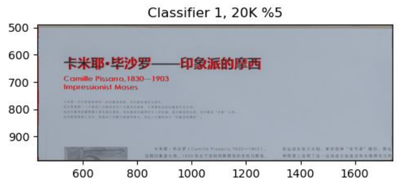


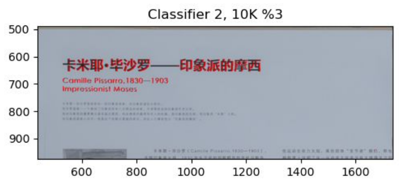
Test 2(Canon_011_HR, 2500x2300)

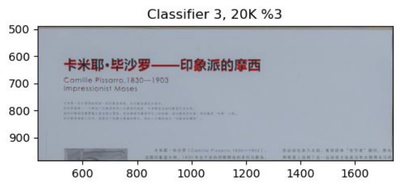


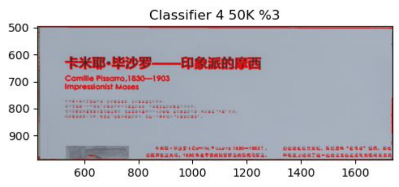


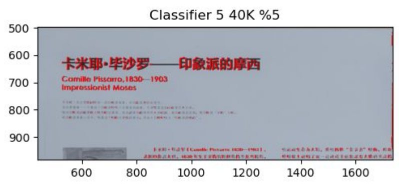




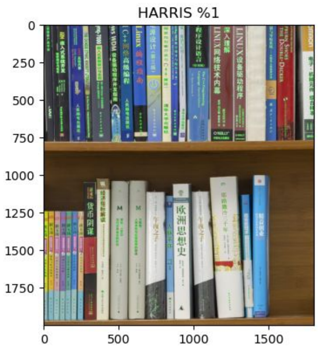


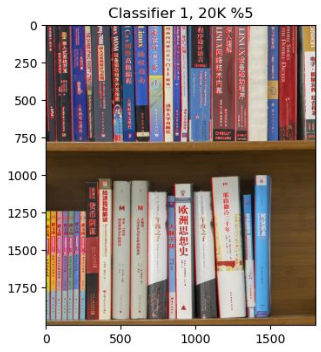


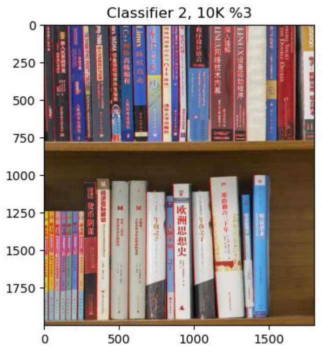


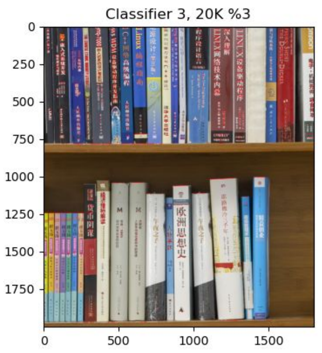


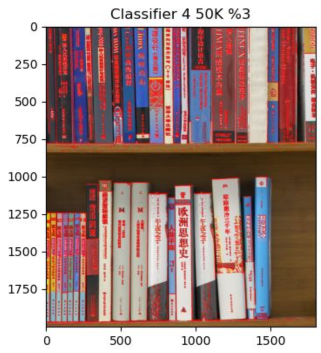
Test 3(Canon_013_HR, 1800x2000)

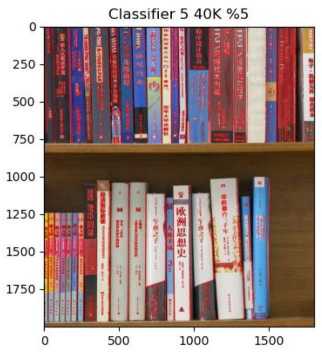


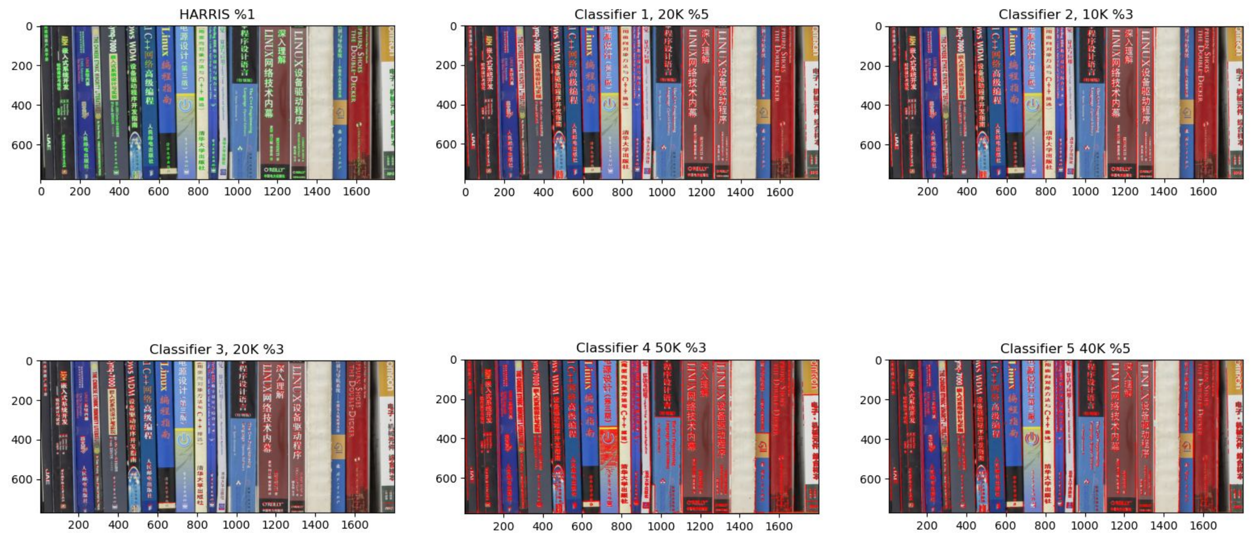




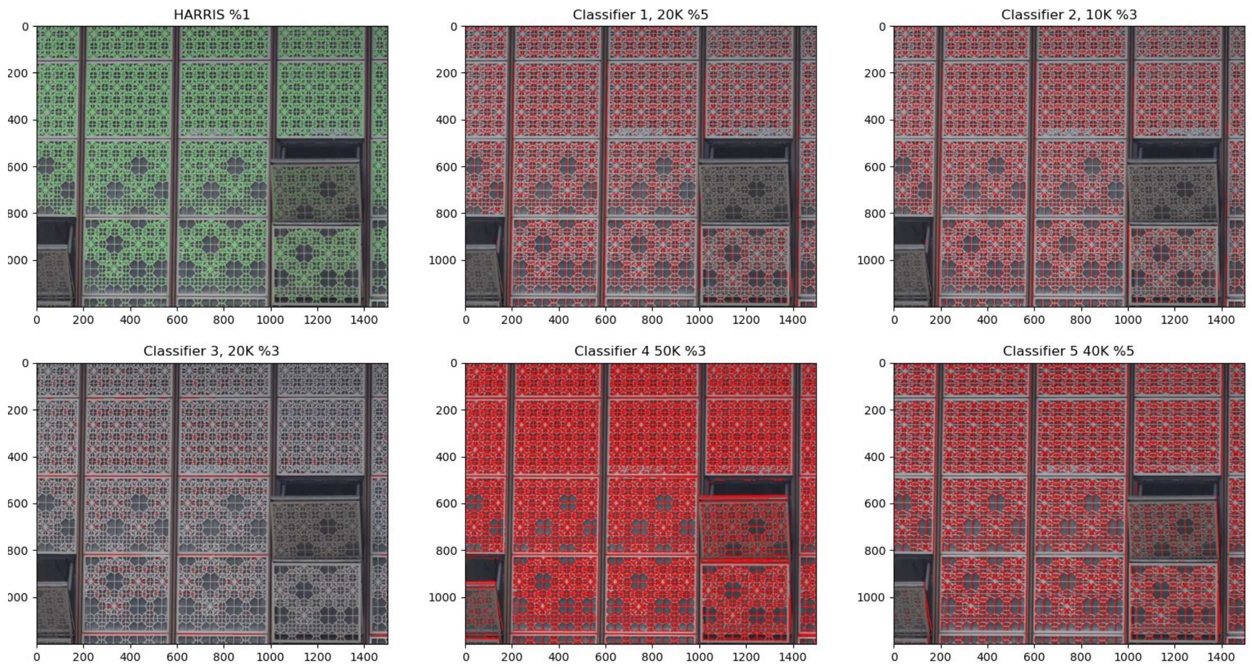


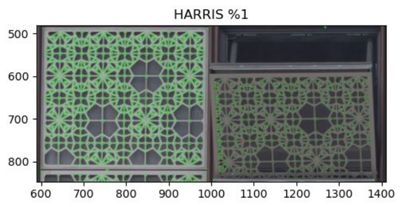


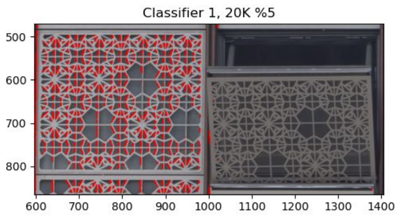


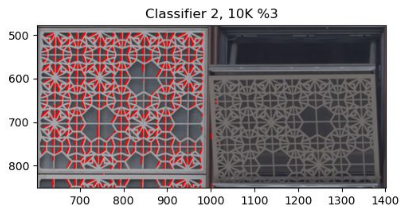


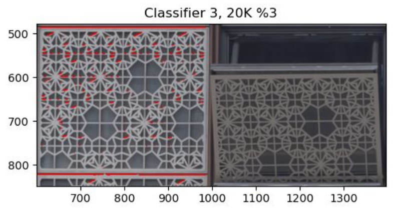
Test 4(Canon_006_HR, 1500x1200)

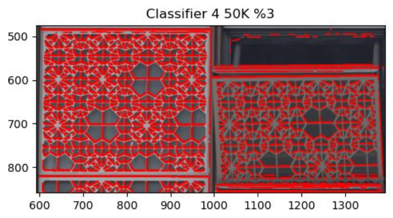


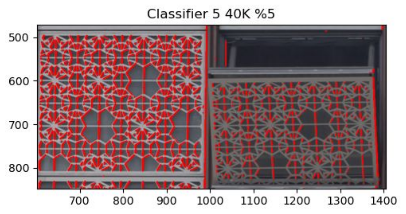












Rest of the test can be found in the results folder.

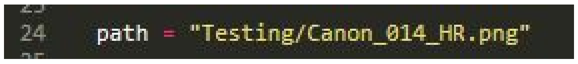
Discussing the Results

I noticed that using Sobel gradient as a base to create the groups has a side effect. The classifiers can detect corners but they also have lots of false positive results on edges. In order to get rid of edges, the data should be modelled to distinctify corners further. Using only sobel on x and y axises naturally result in edge detection.

Also I noticed using higher resolution data to train the classifier can result in better predictions in some cases, like in Figure_1(classifier 1 and classifier 5).

Recreating Tests and Training Models

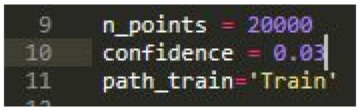
To recreate the test , change the path in the main.py file to the image you want to test.



```
path = "Testing/Canon_014_HR.png"
```

Run command: “python main.py”

Pre-trained models can be found in the Models folder. If you want to train a model with different parameters or images, go to train.py file and change confidence, number of data points to be trained from and training images path.



```
n_points = 20000
confidence = 0.03
path_train='Train'
```

Run command: “python train.py”