Aperture, ISO and shutter speed detection and classification

The goal of this project is to design a scheme that can detach the aperture, ISO and shutter speed of a given picture. Since this three component is the most important elements to take a good picture. For this project we are focus on two kind of the picture: the portrait photo and landscape photo. For the portrait photo, I want to use the http://vis-www.cs.umass.edu/lfw/ as my training dataset. For landscape photo, I want to use https://www.kaggle.com/puneet6060/intel-image-classification as my training dataset for landscape. I can find the information from EXIF metadata fields to label the aperture, ISO and shutter speed of each training picture.

I want to design this project into a classification problem. For each training pictures, I need to find parameter to represent the aperture, parameter to represent the ISO and parameter to represent the shutter speed. Then for each element, I will plot the parameter and use the best clustering technique to classify each picture. I will try to design a neural network to predict these elements. For testing, I will then take portrait photos and landscape photos with different element setting (set aperture, shutter speed to constant and try all different ISO in same condition etc.) to test my trained model.

To quantify the ISO of the pictures, I want to find the Signal-to-noise ratio (SNR) to represent the ISO of the picture. To quantify the aperture of the picture, I want to find the blurriness of the edge of the picture to represent the aperture of the picture. The shutter speed will be represent by the ISO and aperture together.

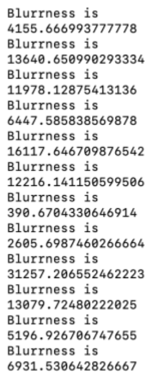
Dev_2 feature extraction:

We need to capture the features from the image to detach the ISO and aperture. The key feature to detach the ISO is the brightness of the picture and the feature to detach the aperture is the blurriness of the edge of the pictures.

Now the problem is I want to have a method that can use a single float number to represent the total blurriness of the graph. The existing method to find the blurriness is computing the Fast Fourier Transform (FFT) of the image. After FFT, we examine the distribution of low and high frequencies to see if there is a low amount of high frequencies. This method can only show weather the image is blur or not. We can choose the number of the high frequencies to represent the blurriness of the image. However, it is hard to determine what is a low number of high frequencies.

Another approach [1] uses the variation of the Laplacian transform. It takes the grayscale of the image and convolve it with a cv2.CV_64F kernel. Then take the standard deviation squared (variance) of the response. The variance can show the blurriness of the image. The Laplacian is often used for edge detection. If an image contains high variance then there is a wide spread of responses, which can represent a non-blur image. If the image contains low variance, indicating there are very little edges in the image. Therefore the image is blurred (less edges). Using the variance value of the Laplacian transform can represent the blurriness of the image and it is more accurate than the FFT method.

I using the second method to show the blurriness of the landscape image set from https://www.kaggle.com/puneet6060/intel-image-classification. I convert the image to grayscale and apply Laplacian transform on it to find the variance of the Laplacian as the blurriness of the image. I printed the blurriness of the image and show the command line screen shot below:

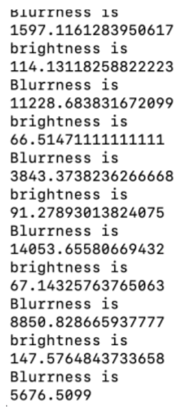


```
Blurrness is
4155.666993777778
Blurrness is
13640.650990293334
Blurrness is
11978.12875413136
Blurrness is
6447.585838569878
Blurrness is
16117.646709876542
Blurrness is
12216.141150599506
Blurrness is
390.6704330646914
Blurrness is
2605.6987460266664
Blurrness is
31257.206552462223
Blurrness is
13079.72480222025
Blurrness is
5196.926706747655
Blurrness is
6931.530642826667
```

Figure 1- Screen shot to show the blurriness of the images

The next steps of the aperture finding is to connect the blurriness of the image to the actual aperture (extract from EXIF information). Then build a CNN to train to detach the aperture.

I used the python Pillow library to calculate the brightness of the image. I used the ImageStat module to calculates the global statistics for an image. We can calculate the brightness using the data from statistics of an image. I can use the average pixel brightness to represent the total brightness of the image. I can also use the RMS pixel brightness to represent the total brightness of the image. For now I used the RMS brightness to represent the total brightness. The screen shot below shows the brightness of the image of given dataset.



```
Blurrness is
1597.1161283950617
brightness is
114.13118258822223
Blurrness is
11228.683831672099
brightness is
66.51471111111111
Blurrness is
3843.3738236266668
brightness is
91.27893013824075
Blurrness is
14053.65580669432
brightness is
67.14325763765063
Blurrness is
8850.828665937777
brightness is
147.5764843733658
Blurrness is
5676.5099
```

Figure 2- Screen shot to show the brightness of the images

To further detach the ISO I need to have three data of an image. The brightness, the exact ISO (from EXIF information) and the peak signal to noise ratio (PSNR use to detach the noise of an image under high ISO). I will try to build the CNN based on brightness and SNR of the picture to predict the ISO of a image.

Reference:
[1] J. L. Pech-Pacheco, G. Cristobal, J. Chamorro-Martinez and J. Fernandez-Valdivia, "Diatom autofocusing in brightfield microscopy: a comparative study," *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000,*