SC 520

Digital Image Processing and Commmunication

Term Project

"Selection of Good Photographs for Printing"

Andrey Krokhin
Ikaro Silva

Introduction

The goal of this project was to develop an automatic image classification system. The images were assumed to be photographs in digital form taken by amateur photographers. The system's objective was to automatically recognize shots of very poor quality. In practice, such a system would skip printing them, thus avoiding charging the customer or having a human agent manually separate good and bad photographs. The determination of quality is somewhat subjective, of course. The ultimate judge of the photograph's quality will be the person viewing it, so there is no hard decision threshold no matter what parameters are used. The objective is to have an algorithm that will not reject any photographs judged by a human to be "good" and that will accept only a small number of the ones judged to be "bad" (it is assumed it is preferable to print some bad photographs rather than discarding some good ones). In this project, we concentrate on the two most common problems that cause poor shot quality: exposure and blur. For severely over- or under-exposed photographs, the image will be almost completely white or completely dark. Improper focusing or shutter speed can degrade the quality of the image, in the extreme cases causing the objects in it to be blurred to the point of being indistinguishable. Some experiments with restoration of the bad photographs were also performed.

Part 1

Over and Underexposed Images

Introduction

The first part of this project was focused on the detection and enhancement of images that were over or underexposed. Typically, when an image is exposed correctly the histogram of it's pixel intensity values will seem fairly symmetric (centered in the mid-range) and expanded of all intensity levels (Figure 1). Badly exposed images, on the other hand, will have a skewed histogram of the pixel intensity values. The histogram is skewed to one side and its mean can be centered in limits of the pixels ranges. In addition, the pixels in a poorly exposed image do not cover the complete range of intensities (or there is a big difference in the number of pixels covering the low, mid, and high ranges of intensities). Figure 1 show a typical histogram of normal, over, and undexposed images. Due to their histogram distribution, overexposed images will appear brighter than usual, while underexposed images will appear darker than normal (Jain 1989; Konrad class notes,2001).

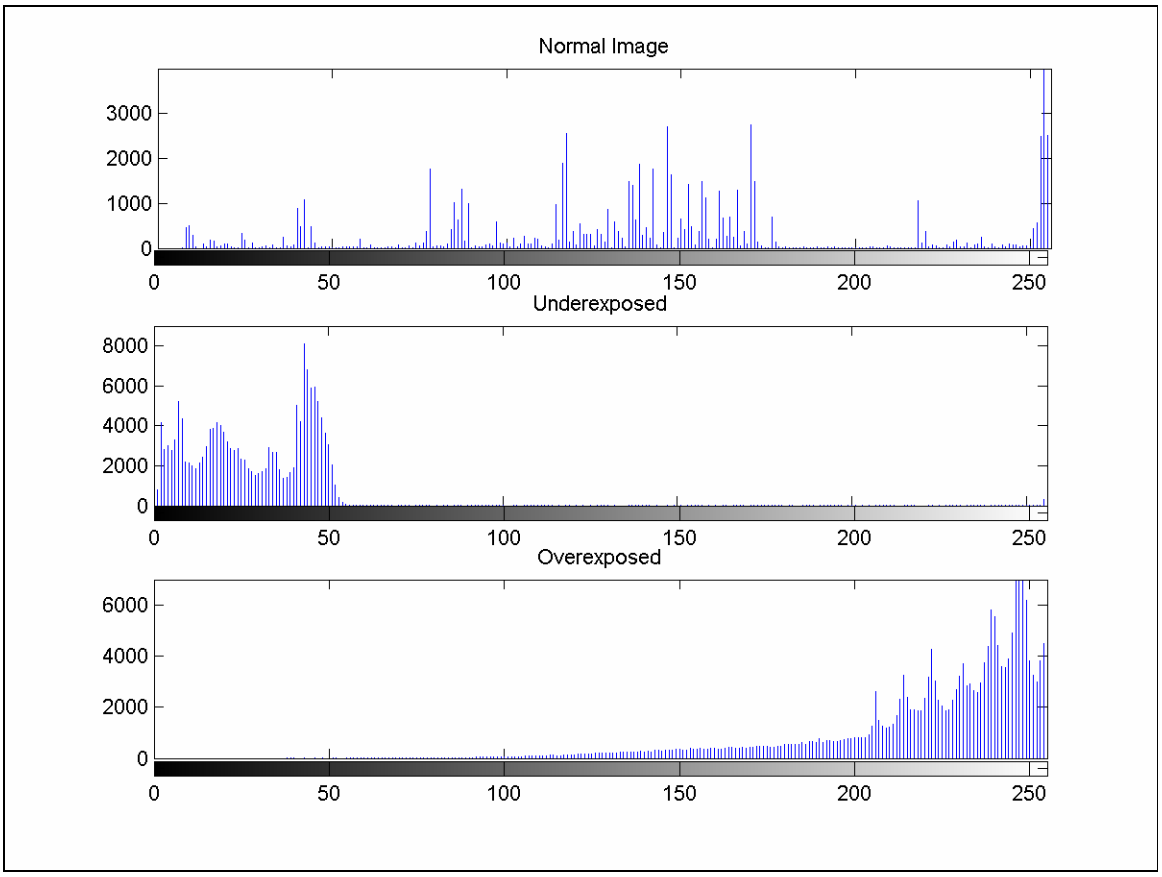


Figure 1-1. Typical histograms for normal, under, and overexposed images.

Goal

The goal of this part of the project was to detect and restore badly exposed images. The detection part consisted in utilizing features from the histogram of the image in the classification. In addition, our detection costs should be higher for misclassifying a good image than for misclassifying a bad image. Enhancement of the image was done in two approaches. Both of these approached tried to modify the histogram of the image in such a way as to maximize the dynamic range being used by the pixels. The training data consisted of 20 images of each type of exposure (total of 60 images). Images where in JPG format, mostly RGB (colored images) but 2 grayscale images were included as well.

Detection

The first step in the creation of the detection algorithm was to determine a threshold value for which to set the histogram values to zero (i.e., consider pixel values insignificant). The determination of this threshold value was obtained by iterating the following procedure through all of our 60 images (see Figure 2 for visualization):

Step 1) Convert the image into a hue-saturation-value (using HSV function in MATLAB). The value information (intensity information of the image) was used in the subsequent steps. This was necessary in order to accommodated colored images and map pixel value to an intensity space only (independent of color).

Step 1) Pick a value p that will be held constant through all the images (we tested with different p value from [0, 0.1, 0.02, 0.03, .0.5]. This is the parameter to be optimized.

Step 2) Set all the values in the histogram below: max value * p to zero (the red line in Figure 2).

Step 3) Find the width of the maximum continuous region above zero from Step 2 (green line in Figure 2).

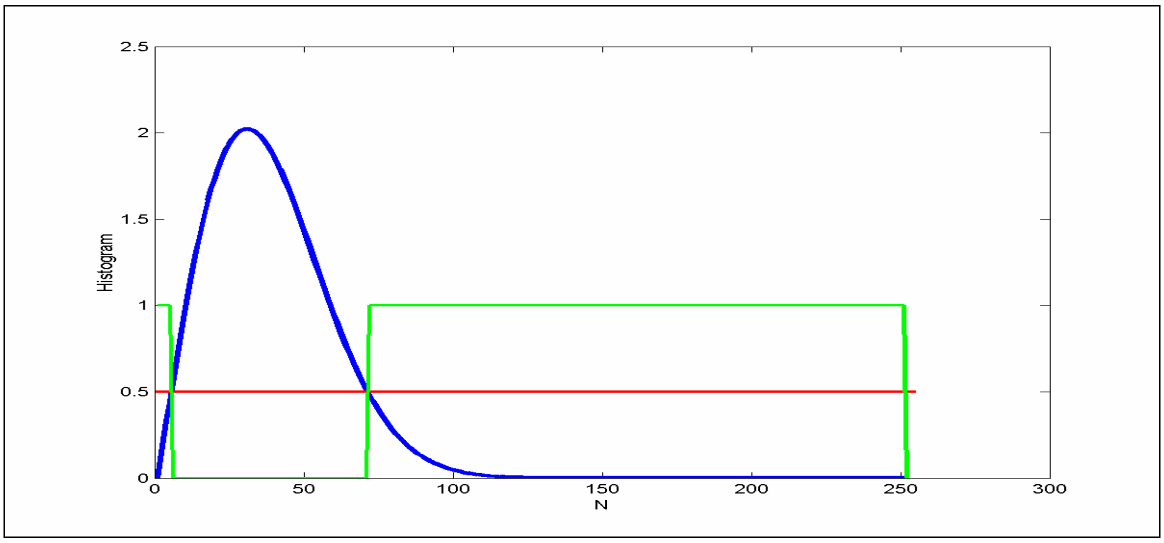


Figure 1-2. Simple sketch of the feature extraction procedure.

The procedure described above was done through values of p varying from 0.001 to 0.5 in steps of 0.005. Once this was done, a cost function was defined as:

$$J(x) = \left(mean\left(std(w_G), std(w_O), std(w_U)\right)\right) / \left|mean\left(mean(w_G - w_O), mean(w_G - w_U)\right)\right| \qquad (1)$$

The variables w_G w_U, and w_O represent the distribution of the width of the maximum continuous region below threshold. The cost function is thus a ratio of the average standard deviation for distribution of the maximum width for all sets and the absolute value of the average distance of the width distributions between the good and the poorly exposed images. A small value of J(x) could be generated for instance, by a set of images (good and bad) that had a fairly constant number of pixel values below certain threshold (small standard deviation of the width of the window), and small window widths for normal images, but large window widths for the bad images (yielding a large distance between the means of the window distribution). A plot of the cost function for several values of p was then obtained (Figure 3). The minimal value of p for this cost function was chosen to be 0.179. Thus our threshold for considering intensity values 'insignificant ' in a histogram is any value below 17.9% of the peak value.

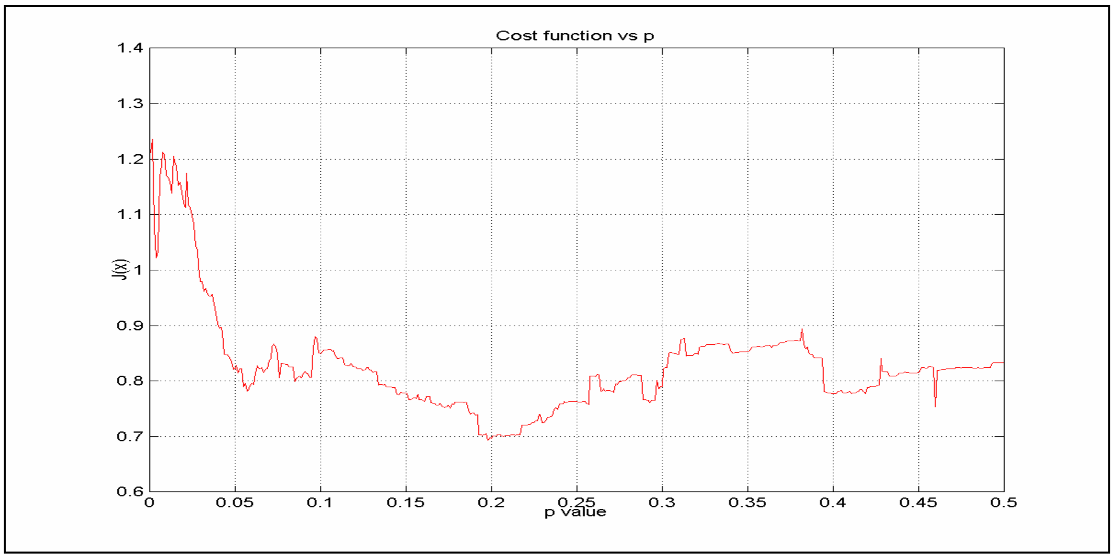
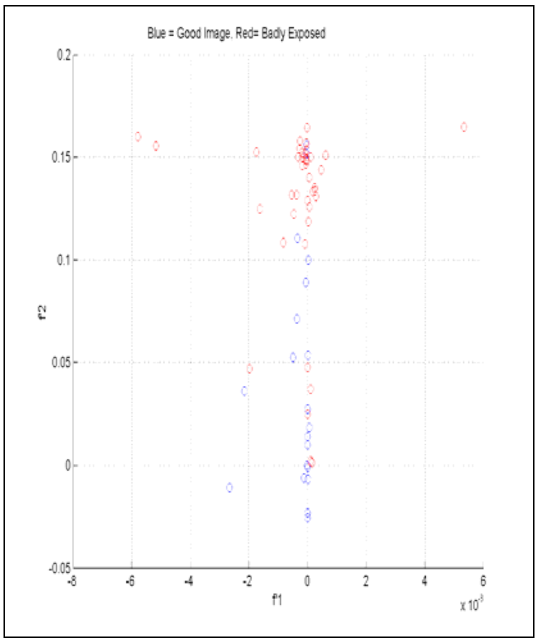


Figure 1-3. Plot of cost function vs p value.

Once the threshold value was determined from the procedure above, the remaining part of the detection algorithm consisted in obtaining a set of 11 features from the input HSV image. The 11 features used in the detection were: mean value of pixel intensities, standard deviation of pixel intensities, maximum window size below a threshold of 17.9% of the peak value of histogram, and 7 coefficients of a 6th order polynomial fitted to the histogram (using POLYFIT function). These 11 features were then used to perform a linear discriminant analysis in our training dataset. The discriminant analysis initially tried to maximize the distance between good and bad images and reduce the dimension of the feature space to two. A second subsequent linear analysis was done in the original 11 features to separate the over and underexposed groups (also reducing the feature dimensions to two). Figure 4 and 5 shows the scatter plots of the image set distribution after applying linear discriminant analysis.



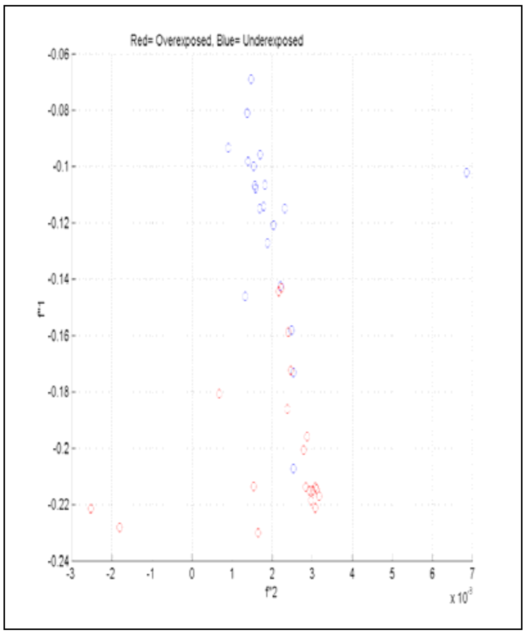


Figure 1-4, 1-5. Scatter plot of the image features after LDA.

The threshold values for classifying the images were obtained by visual inspection of the scattergrams above. Images were classified as good if their f’2 value were below 0.13 (bearing in mind the higher costs for misclassifying a good image than misclassifying a bad image). Images were classified as overexposed if their f'2 value was above 0.13 and their f''2 value below –0.16. In similar manner, images were classified as underexposed if their f'2 value was above 0.13 and their f''2 value were above –0.16.

Restoration

Restoration was attempted in two ways. The first restoration process was by contrast stretching according to Jain, 1989. For contrast stretching, after the detection of the image, the threshold was changed, with a p value of 1% (once the image was identified as badly exposed, this value yielded better results than 17.9%). The contrast stretching was done by first compressing the largest window region below threshold (green rectangle in Figure 2). This intensity range was either compressed to only one value if possible, or to a certain value such that the resulting new histogram would not have a peak at the compressed region that is bigger the threshold. The second option was necessary in order to reduce possible artifacts from the compression (i.e.: the new image would display very bright spots and unique colors that weren't present originally). After compression, the region above threshold was expanded until the newly compressed region. Note that this restoration process can work with the over and underexposed images, as well as images that do not utilize the mid intensities i.e., with peaks in the edges). The second restoration process was histogram equalization. This was done by using the MATLAB function HISTEQ on the HSV intensity values. The function was called with its default values, so the histogram of the resulting image was more uniformly distributed.

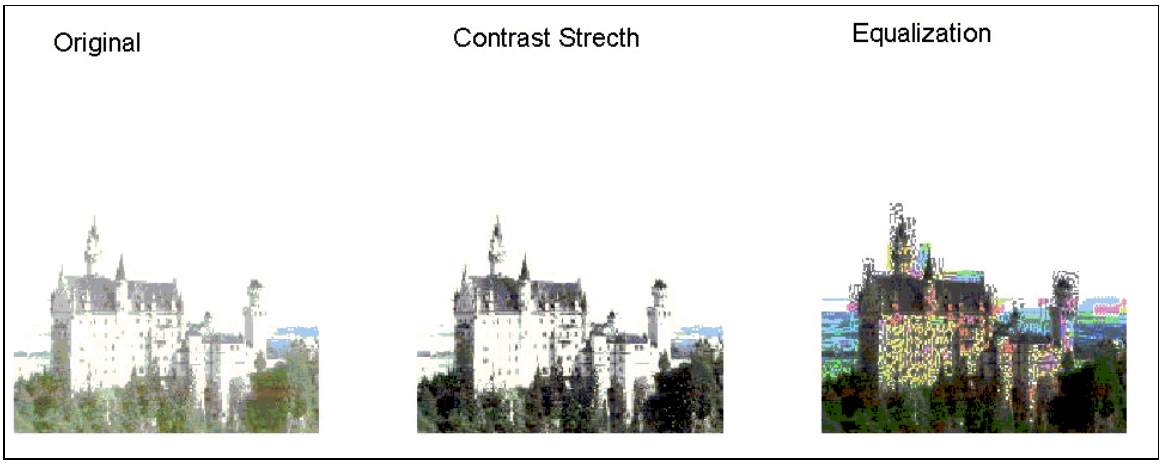
Results

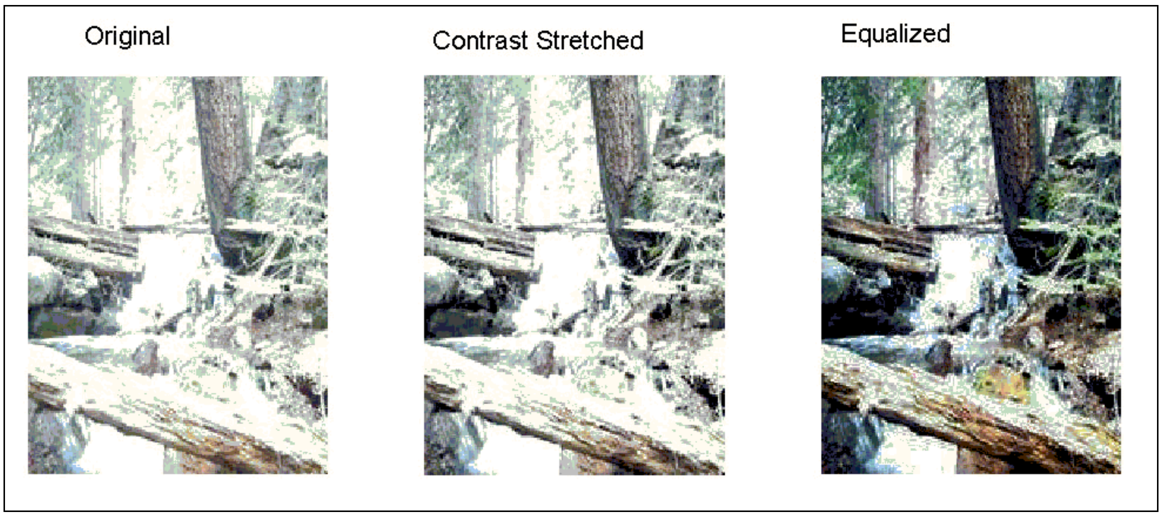
The detection procedure yielded the following result in the dataset (note that this was the same dataset used for the creation of the algorithm):

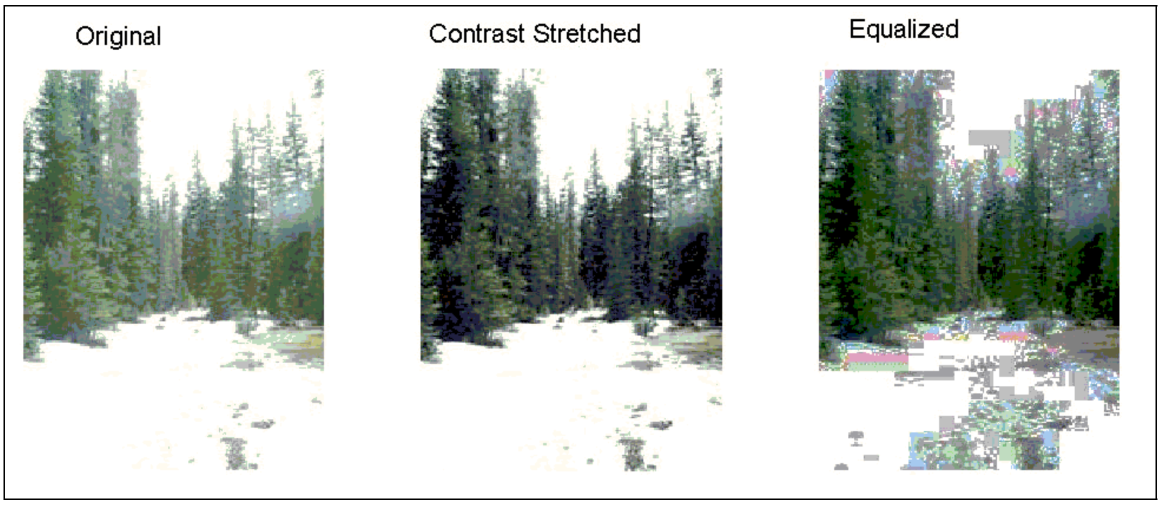
misclassification error of good images: 25%		
misclassification of overexposed images:	30%	(20% classified as good)
misclassification of underexposed images:	45%	(35% classified as good)

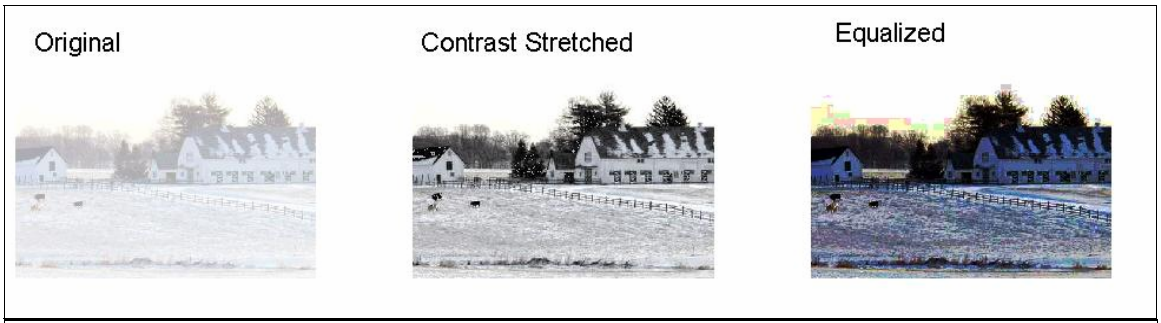
The high misclassification error in the badly exposed images were in part due to the lower costs for misclassifying them. As noted in the parentheses, it was much more unlikely for an overexposed image to be classified as underexposed than normal (and vice versa for the underexposed case). In addition, the error rate for the good images wasn't as low as desired. But, however, the criteria for selecting a "good" image for our original data set was somehow arbitrary and subjective, and rarely was the histogram of the image taken into account. This said, it is quite possible that a "good" image classified by the algorithm as bad, has in fact, pixel intensities regions that are not being used and can thus benefit from the restoration process. Since these error rates can be judged relative only to the intent of the user, our algorithm, when detecting a poor image, should return both the original and the two reconstructed images and let the user decide which is best. The pictures below give a sample demonstration of the restoration process done in some of the images in our dataset (note that performance of both restoration processes were not necessarily correlated).

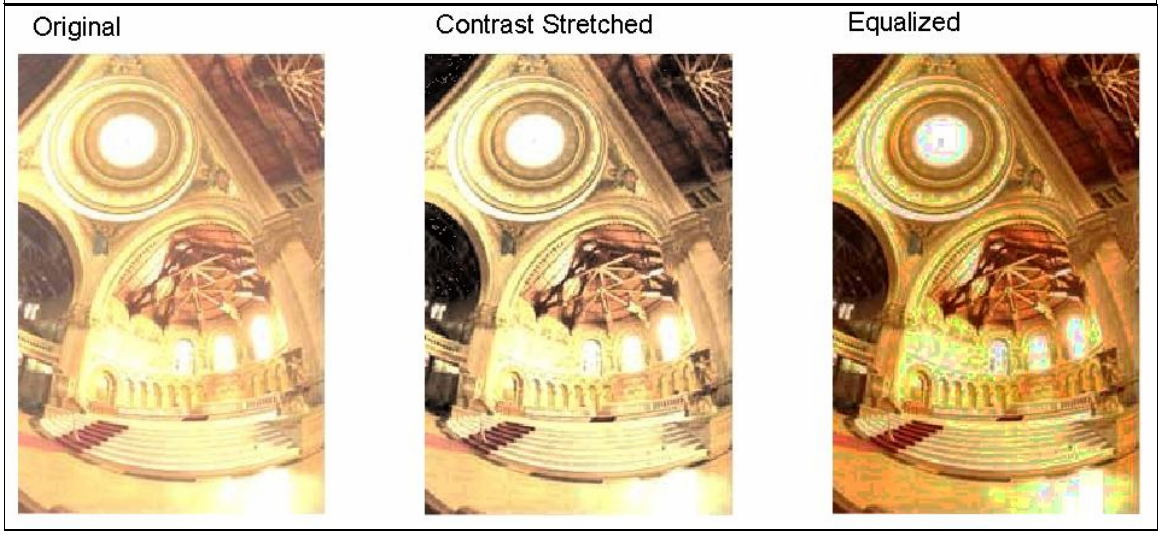
Overexposed Images



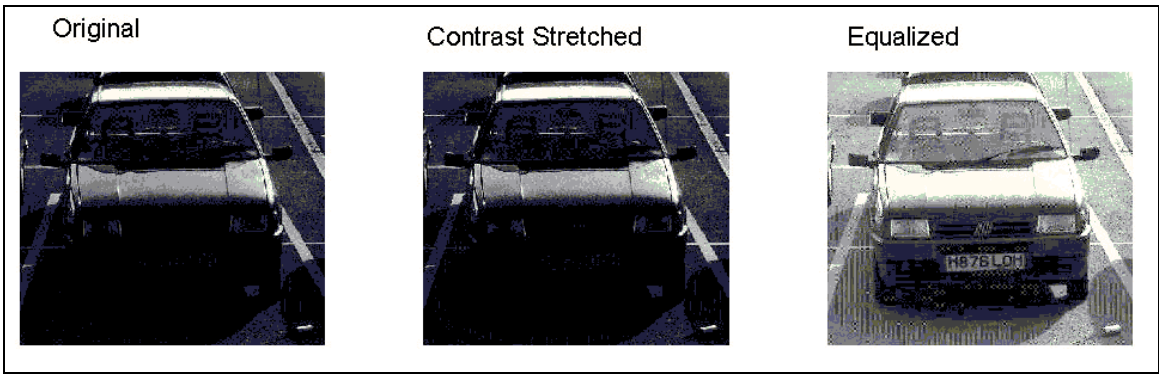


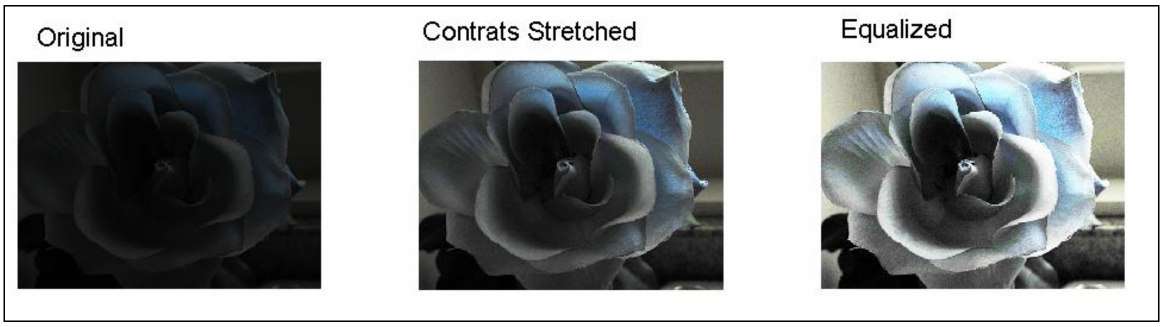


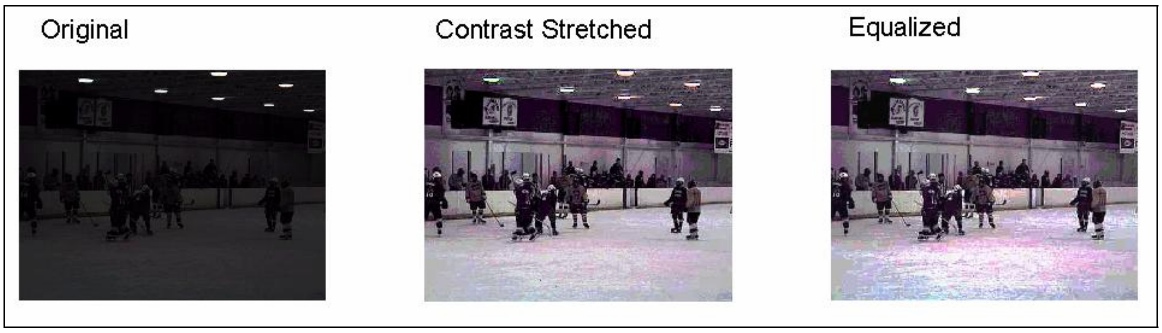


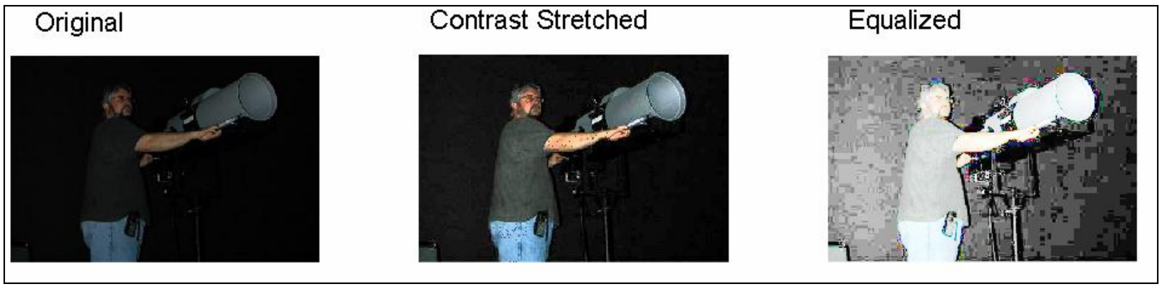


Underexposed Images






Original
Contrast Stretched
Equalized


Original
Contrast Stretched
Equalized

Part 2

Blurred and out-of-focus images

Blur is also a very common factor that degrades image quality. In amateur photography, blur is typically caused by incorrect focus setting or shutter speed (e.g. an object moves too fast for the selected duration of capture). As for over- and underexposed photographs, the determination of whether the photograph is blurred or not is somewhat subjective. It is possible for images to be somewhat blurred, yet be acceptable for printing. The goal was to reject the images that were severely blurred to the extent that no useful details were visible in most of the area.

Our first approach to blur detection was to look at the frequency-domain representation of an image. Since blur is a low-pass filtering operation, it seemed logical that normal photos would have more high-frequency content that blurred ones. We used the following technique: take the FFT of an image and calculate the energy of the coefficients that are not farther that *R* from the origin (the DC component). *R* was some value from 5 to 25. This energy was divided by the total energy of the image. The objective was to see if there was some threshhold *T* for normal images such that, for example, the first one percent of the Fourier coefficients would contain more than *T* percent of the total energy. If the first few coefficients contained less than that fraction of the energy, the image would be considered to be blurred.

After the calculations with a series of images were done, however, it became apparent that there was too much variation in spectral content between images for it to be considered a reliable criterion. The coefficients less than 10 units for the origin, for normal images, could contain anywhere from 70 to 95 percent of the total energy. This wide variation also held for blurred images. While many had energy content above 90 percent within *R*, some had as little as 60 percent, which was *less* than for any normal image. Apparently, this was caused by the fact that some blurred images contained stripes, ghost reflections, and high-frequency oscillations. Therefore, this method of classifying images was rejected.

Another approach was to detect blur by “edge content”, rather than frequency content. The idea was than normal images should contain many edges, while blurred ones contain few. MATLAB has a built-in edge detection routine, which was very useful, eliminating the need to write our own. Two edge detection methods were used: Canny and Laplacian of Gaussian (LoG). These are the most robust and well-known edge detection methods. MATLAB allows selecting the treshhold for edge detection, so that edges below the threshhold are ignored. (For the Canny detector, there are actually two thresholds, one for weak edges that are connected to strong edges). However, we found that changing the threshold has little effect on classification, since it chnages the number of edges uniformly for *all* images, but causes little *relative* change between normal and blurred images.

Formally, the classification criterion was defined as the number of pixels belonging to an edge divided by the total number of pixels in the image. Since the output of MATLAB edge detector is a binary image, the total number of “edge” pixels was simply the sum of all pixels in the image. Edges are represented as 1’s and therefore were included in the sum, while zeros were non-edges and did not contribute to the total. Thus the edge content criterion was computed as

$$f = \frac{E}{WH}$$

where *f* is the fraction of edge content, *E* is the total number of edge pixels, and *W* and *H* are the width and height of the image, respectively.

The variable *f* for blurred and normal images was plotted on the same graph.

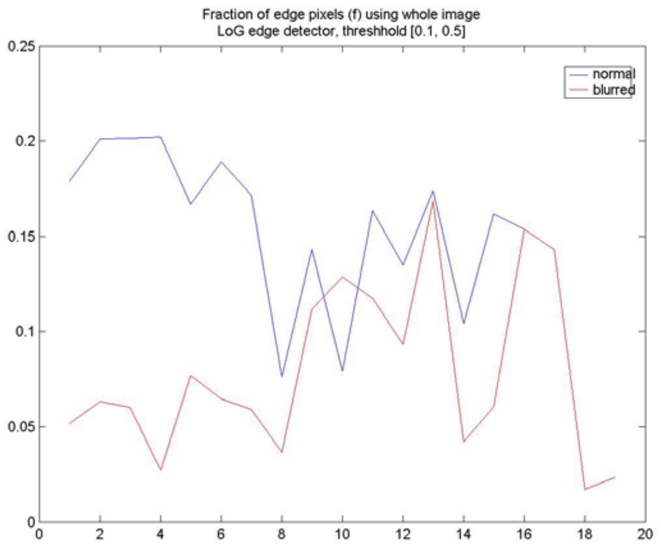


Fig. 2-1. Plot of *f* for Canny edge detector, normal and blurred images.

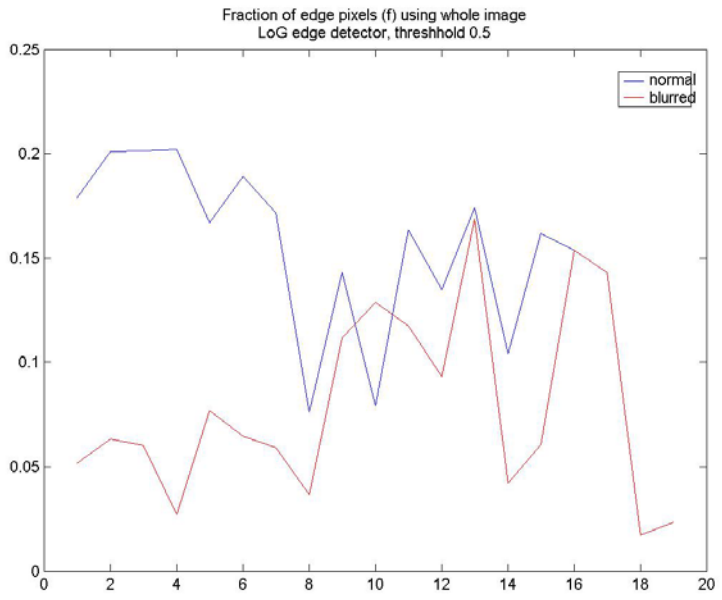


Fig. 2-2. Plot of *f* for LoG edge detector, normal and blurred images.

There was significant overlap between the two lines, but the result was definitely much better than the Fourier-domain classification. Part of the problem was that some good images contained uniform or black areas with few or no edges. To take this into account, the algorithm was modified to split the image into several blocks, compute the edge content in each one, and take the maximum among all blocks. This was based on the reasoning that in a normal image, there will be areas will contain many edges, while in blurred images, blur should be present more or less everywhere. The graph below shows the results for splitting the image into 2x2, 3x3, and 4x4 blocks, where the splits were made in horizontal and vertical directions. Since the images were assumed to be anisotropic, there was no reason to split it differently in the two directions.

The best class separation was achieved for a 3x3 split. As can be seen from the graph below, blurred and good images are better separated than when no block splitting was used.

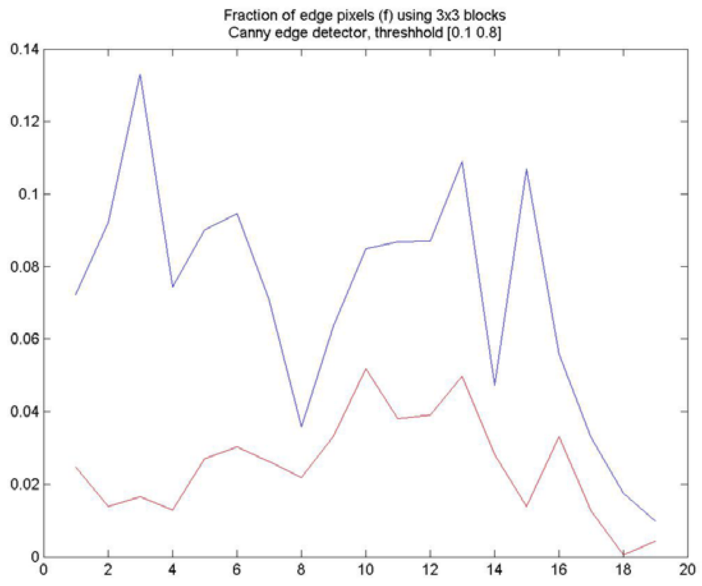


Fig. 2-3. Plot of *f* for Canny edge detector, splitting the image into 3x3 blocks and taking the max. *f*.

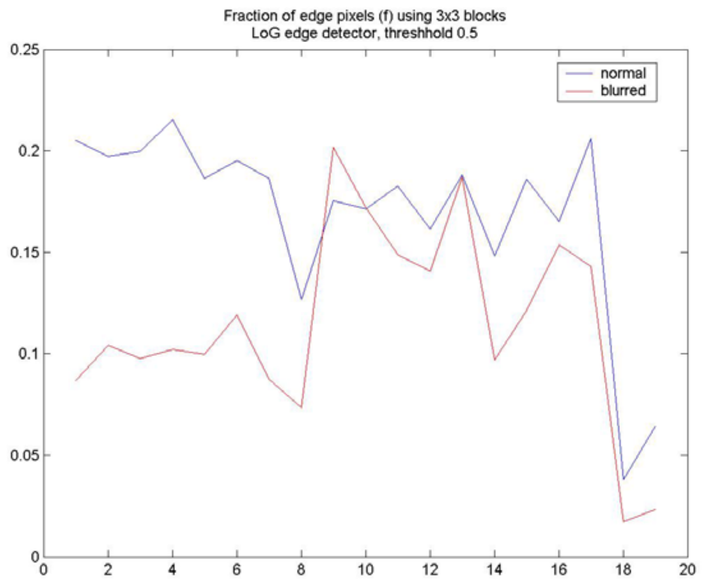


Fig. 2-4. Plot of *f* for LoG edge detector, splitting the image into 3x3 blocks and taking the max. *f*.

At first, simple threshholding was attempted to separate the blurred images from the normal ones according to edge content. When *f* exceeded a certain value using the Canny edge detector and another value using LoG, the image was considered to be normal. However, the classification error was still high: out of 19 normal images, 2 were misclassified as blurred, and out of 15 blurred images, 6 were misclassified as normal. Since we were using two classifiers (Canny and LoG), we thought that classification could be improved by exploiting the dimensionality of the problem. We used linear discriminant analysis (LDA) to project the two-dimensional classifier data onto a line that best separates the data in a least-squares sense. If we denote the direction of this line by ***w***, the projection can be written as

$$y = \boldsymbol{w}^t\boldsymbol{f}$$

where y is the new (one-dimensional) data. If we define the scatter for the projected samples belonging to class i as

$$\tilde{s}_i^2 = \sum_y (y - \tilde{m}_i)^2$$

the mean of *original* samples in class i as $\boldsymbol{m_i}$ (a vector), and the mean of projected samples in class i as m_i (a scalar), then best separation can be achieved by selecting $\boldsymbol{w}^t\boldsymbol{f}$ such that the criterion function

$$J(\boldsymbol{w}) = \frac{|\tilde{m}_1 - \tilde{m}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

is maximized.

This function can be rewritten in terms of scatter matrices:

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^t \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^t \boldsymbol{S}_W \boldsymbol{w}}$$

where

$$\boldsymbol{S}_i = \sum_{\boldsymbol{f}} (\boldsymbol{f} - \boldsymbol{m}_i)(\boldsymbol{f} - \boldsymbol{m}_i)^t$$

$$\boldsymbol{S}_W = \boldsymbol{S}_1 + \boldsymbol{S}_2 \text{ (within-class scatter matrix)}$$

$$\boldsymbol{S}_B = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^t \quad \text{(between-class scatter matrix)}$$

The solution $\boldsymbol{w}$ that optimizes $\boldsymbol{J}$ is

$$\boldsymbol{w} = \boldsymbol{S}_W^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2)$$

This is the Fisher’s linear discriminant, which maximizes the ration of between-class scatter to within-class scatter.

The results of applying LDA are shown on the graph. It is obvious that the samples are now much better separated into classes. The threshold separating good images from blurred is around zero. With this threshold, no blurred images are misclassified as good, and 3 good images out of 19 are misclassified as blurred. This represents a quite low error rate.

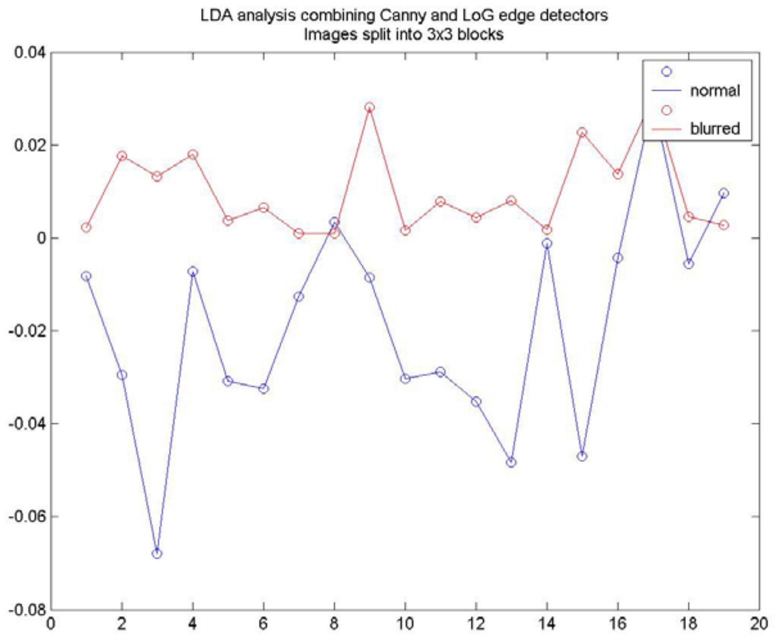


Fig. 2-5. Applying LDA to outputs from Canny and LoG algorithms.

In addition to detection, restoration of blurred images was also attempted using unsharp masking. Although generally, the PSF of the blur needs to be known for restoration, unsharp masking provides an easy and fast way for blind restoration, sometimes achieving improvements in the visual attractiveness of an image when the blur is more or less uniform and the SNR is not too low. An example of a photo restored in this way is shown below

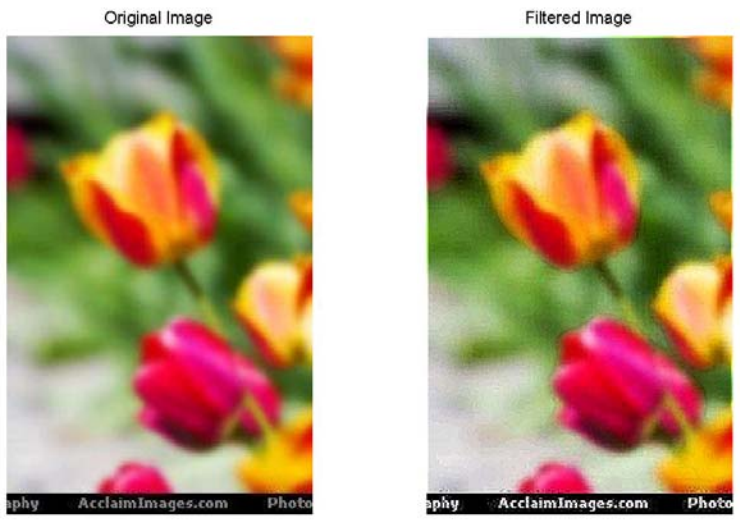


Fig. 2-6. Image deblurred using unsharp masking.

Conclusion

In our opinion, the performed simulations showed reasonably good results for a “beta version” of the system. Further development would research more sophisticated classifiers and pattern recognition techniques, since this is essentially a pattern recognition (classification) problems. One obstacle is the lack of formally defined criteria for what constitutes a “bad image”. If a standard could be established and accepted by the image processing/photography community, the system’s performance could be compared with others by a formal performance metric. On the other hand, the difficulty in constructing a good classifier would also be a difficulty for establishing a standard. If such a standard could be described mathematically, all photograph classification algorithms could replicate it. Perhaps a better solution would be to have universally accepted set of images labeled “good” or “bad”. If this set were very large, it could split into a training batch and a test batch. Some form of Bayesian estimator could probably be devised, attaching a specific weight to “false positives” and “false negatives”.

While this project did not teach us new image techniques, it did provide a good experience working on a “real-life” problem. It also provided good practice (implementing and coding) for some techniques learned in class, such as histogram stretching and edge detection.

Bibliography

"Carnegie Mellon Researcher Develops Intelligent Technology That Automatically Enhances Underexposed Photographs". Science Daily, 2003-12-10. <http://www.sciencedaily.com/releases/2003/12/031210072701.htm>

"Photograph Artifacts Detection and Correction". National University of Singapore, School of Computing. <http://www.comp.nus.edu.sg/~photo/>

"Life of a Digital Photo". Kentc's SOAPbox, Feb. 11, 2004. <http://blogs.msdn.com/kentc/archive/2004/02/11.aspx>

"Histogram Equalization". MATLAB Image Processing Toolbox User's Guide. Mathworks, 2004.

Rafael C. Gonzalez, Richard E. Woods. "Digital Image Processing". Prentice Hall, 2002.

Duda R., Peter H., Stork D. "Pattern Classification". Second Ed. John Wiley & Sons, Inc, 2001.

Jain A. "Fundamentals of Digital Image Processing. Information and System Sciences Series, Prentice Hall, 1989.

Ballester, C.; Bertalmio, M.; Caselles, V.; Sapiro, G.; Verdera, J.; "Filling-in by joint interpolation of vector fields and gray levels". IEEE Trans. on Image Processing, Aug. 2001.

Yamauchi, H.; Haber, J.; Seidel, H.-P.; "Image restoration using multiresolution texture synthesis and image inpainting". Proceedings of Computer Graphics International, July 2003.

Mathworks website, www.mathworks.com