CS24110 Assignment: Beautify! - Automatic enhancment of digital images via histogram equalisation

Lukasz Wrzolek
Department of Computer Science, Aberystwyth University, Aberystwyth, SY23 3DB, UK

This paper presents a novel approach to image enhancement based on the computing histogram and cumulative histogram of three different colour channels and transform that histograms by using function to shift pixel values to obtain uniform histogram. This shifting algorithm is histogram equalisation. This algorithm is simple but not easy to develop. The algorithm is tested on five test images, three grayscale images and one image of my friend[1]. The results show that the alghoritm is able to improve the quality of an image, but also can destroy it. These aspects are discussed in this paper.

I. Introduction

IN this day and age almost everyone have it's own computer. Most people also have smartphones with digital cameras and use them to take the photos and publicate them after applying different filters to make pictures nice and pretty. Every basic filter operation is based on image processing aspects. The most popular filter which can be used in applications like Instagram™ or Facebook™ is histogram equalisation.

We can find a lot of applications which can support users with basic or more advanced image manipulation tools. The famous page YouTube™ contains tutorial videos where people are teaching others how to use each program function and what it does. The most popular programs to modify the image are Adobe™ Photoshop®, Gimp[2], Picasa[3]. Picasa and Gimp are free software so we can download it and use full version from the beginning. Photoshop has free trial for 30 days when you download it from non-official pages. At the official page we have to buy a "Picture Package" which includes photoshop for $8,5£/permonth$[4].

This software has methods to enchance the image without any user input. Even person without knowledge can use simple options to improve image. Program is doing everything for them. We can select specific areas to blur the image using gaussian blur[5] or different one, doing changes only at foreground or background. Methods of doing operations like that are really complicated and hard to develop. These programs give huge possibilities to change the image, from basics operations like changing brightness or contrast to enchance the edges or create different blur effects. Some of that effects with source code can be found at "JH Labs" web site[5].

This paper is concerned with a new method to enhance a given image without any user input. The idea behind the approach is to create a histogram and cumulative histogram of the values of input image. We are using histogram methods to store the data and prepare it to enhance the image using histogram equalisation. In theory, this should create a balanced image; So the histograms of images after equalisation process should present balanced contrast and cumulative histograms should look like symetric Pythagorean Triangle[6], and it's called uniform histogram.

The remainder of the paper is organised as follows: in Section II the methodology of the approach is given at both a high and low level. The results are given in Section III and then discussions and conclusions are drawn in Section IV. This paper contain 7 pages.

II. Methodology

In this section the methodology behind the histogram equalisation method to image enhancement is described. As well as a high-level overview of how the method works, a mathematical outline and pseudocode are also provided.

A. High-level Overview

The basic idea behind the proposed automatic enhancement approach is histogram equalisation method of a given image. After that process image of Cumulative Histogram will be similar to the Pythagorean Triangle. For an 8-bit gryscale image, this method transformes the images

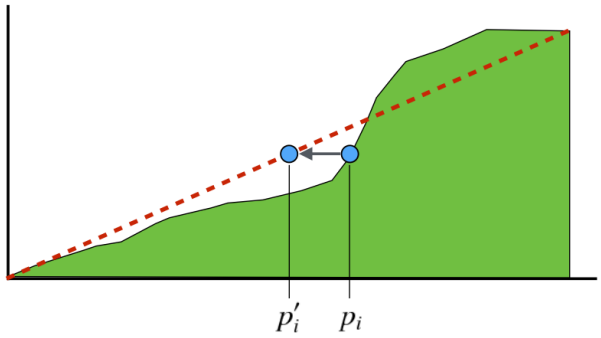


Fig. 1. An image of the proposed enhancement method. At the picture we can see cumulative histogram of an image and red line starts at the beggining of the histogram $(0,0)$ and ends at the highest value of coordinate system $(255,1)$. $X-axis$ describes intensity values, range from 0 to 255; $Y-axis$ is cumulative frequency. Range of it is from 0 to 1. Pixels $p(i)$ and $p(i)'$ show how the pixels are shifted in histogram equalisation method.[7]

Algorithm 1 Histogram Equalisation Algorithm

function HISTOGRAM EQ(I) ▷ where $|I| = m \times n$
  $ArrayList < int[] > histogram = histogramTraingle(inputImage)$ ▷ creating an object of ArrayList to have access to the values after equalisation
  $I' = \text{zeros}(u, v, 3)$

  for each p **in** I **do**
    $getColor()[8]$ for each channel

    //do equalise for each channel
    $K = 256$
    $p'_i = p_i \times ((K-1)/(u \times v))$
    // tresholding
    if $p'_i \geqslant 255$ **then**
      $p'_i = 255$
    end if

    //set pixels
    $pixels = colorToRGB(red, green, blue)$
  end for

  return I'
end function

Algorithm 2 Computing Histogram Algorithm

function COMPUTEHISTOGRAM (I) ▷ where $|I| = m \times n$

  $int[]histogram = hisotgram[256]$ ▷ Different histogram for each cahnnel separatelly
  $histogram = \text{zeros}(histogram.length)$ ▷ Filling up the histogram

  for each p **in** I **do**
    $getColor()[8]$ for each channel

    $histogram[color]++$ ▷ color definied for R, G, B separatelly
  end for
  $ArrayList < int[] > hist = newArrayList < int[] > ()[9]$
  $hist.add(histogram)$ ▷ Storing each histogram in the array list
  return histogram
end function

to version with balanced brightness and contrast. It looks similar to method where the brightness or contrast are added to the each pixel. For a 24-bit true color image the method creates three different histograms 2, one for each color and storing the values at ArrayList of type $< int[] >$. It is a preparation before transforming the original cumulative histogram to uniform cumulative histogram. We need to find pixel intensity value for each color channel and multiply that value by the linear equation:

$$K = 256; linear = \frac{(K-1)}{(m \times n)} \tag{1}$$

$$p_i \times linear = p'_i \tag{2}$$

Figure 1 details a sketch of the proposed method. The current pixel value of each colour channel (p(i)) is found and then the intensity shift is found by calculating the actual pixel value multiplied with the "linear equation"; it gives us new ($p(i)'$) value. It's important to remember about Thresholding 1

B. Detailed Description

The proposed histogram equalisation algorithm is good for images with poor intensity distribution. "Histogram equalisation is the process by which we make all values in an image equally probable. The resulting image will have a new, more uniformly distributed histogram. In reality, of course, the grey level values are discrete. In the continuous domain there is an infinite number of values in any interval. In digital images, we have only a finite number of pixels in each range. As the range is stretched, and the number of pixels in it is preserved, there is only a finite number of pixels with which the stretched range is populated. The histogram that results is spread over the whole range of grey values, but it is far from flat. Histogram equalisation will not "flatten" a histogram. It redistributes intensity distributions. If the histogram of an image has many peaks and valleys it will still have paeaks and valleys after equalization."[10]

That process will produce an enhanced image $I'(u, v)$ using a point operation function f. That is:

$$I'(u, v) = f_{\text{he}}(p) = \lfloor H(p) \times \frac{(K-1)}{(u \times v)} \rfloor \tag{3}$$

where $f_{\text{he}}(p)$ is a homogeneous point operation and the geometry of I' is exactly the same as I.

When we are considering a grayscale image x with n_i as a number of occurences of gray level i. The probability of an occurence of a pixel of level i in the image is:

$$p_x(i) = p(x = i) = n_i/n; 0 \leqslant i < L \tag{4}$$

Where L is total number of gray levels in the image; n total pixels in the image; $p_x(i)$ - normalized pixel value to range $[0, 1]$.

"To obtain a normalised representation of a histogram we need to convert it to a probability distribution (probability density function).

$$\sum_{i=0}^{K-1} h(i) = m \times n \tag{5}$$

This can be done by dividing each entry in the histogram by the sum of all pixels:

$$p(i) = \frac{h(i)}{m \times n} \tag{6}$$

Now, $p(i)$ gives us the *probability* of pixel value i occuring in the image. The probability of i being any possible value is 1:

$$\sum_{i=0}^{K-1} p(i) = 1 \tag{7}$$

The "probability theory version" of the cumulative histogram is the **cumulative distribution function (CDF)**, P(i).

$$P(i) = \frac{H(i)}{H(K-1)} = \frac{H}{m \times n} = \sum_{j=0}^{i} \frac{h(j)}{m \times n} \tag{8}$$

$$P(i) = \sum_{j=0}^{i} p(j), for 0 \leqslant i < K \tag{9}$$

The above function is monotonically increasing with extereme values at" [7]:

$$P(0) = p(0) \tag{10}$$

$$P(K-1) = \sum_{i=0}^{K-1} p(i) = 1 \tag{11}$$

C. Implementation

The description in the previous section is focused on a single colour channel; however, since the algorithm needs to work on RGB images, the above method will need to be repeated for each colour channel. To achieve this, the implementation will need to keep track of the actual pixel values for the three colour channels separately and then ensure that each pixel value for three channels is multiplied by linear function and the value after that is in range [0..255]. As well as this, a thresholfding operation will need to be employed so that the modified intensities do not fall out of the displayable range of intensities.

The pseudocode algorithm is given in Algorithm 1. The function takes as input an RGB $m \times n$ image I and then starts by producing a blank output image of the same dimensions (line 2). Lines 5-18 is a for loop through the input image summing each pixel channel and then taking the value of each colour; Red, Green Blue separatelly; and then multiplying that value with linear equation to shift the pixel to the final position p'_i. Once the value of the pixel is calculated we need to check if new value is not bigger that 255. To do that we need to create if statement where actual pixel value is compared with 255, and if p_i is bigger than 255, value is automatically set to 255. Finally, the output image is filled by adding the shift amount to each pixel value found in the input image. The function returns the modified image (line 20).

III. Results

The proposed method was tested on the provided five images and three extra greyscale images and one image of my friend from her Facebook profile. The results of performing the histogram equalisation algorithm on each of these images are shown in Figure 2.

In all of the images the algorithm causes a change in the modified image, with the least significant change being seen in Image 6 (Figure 2 (k, l)). Although seven of nine images have natural looking results. These Images are $1, 2, 5, 6, 7, 8 and 9$. Image 4 looks quite natural, but we can observe that colours are over saturated and do not comply with something we would perceive in real world.

The modified version of Image 3 appear in slightly different colours because in this image the pixels have lower intensities, so we have more pixels in range [0..99]; also we can spot low balanced contrast. When we are look at the Figure 3 and find histograms of Image 3 ((e),(f)) we can spot that original histograms were low balanced and there was a lot of pixels to shift at the mid of the cumulative histogram to occur uniform histogram. Table I and II show the number of pixels at range [200..255] at each colour channel. To show why the results for each five colour images are so different we are comparing tables where we have amount of pixels before and after equalisation. We can see that the amount of the pixels of each colour channel after equalisation is similar on every Image. Last column of that table III calculates the difference between amount of the pixels before equalisation and after equalisation, in other words it shows how many pixels were shifted from that area. The result will be always positive because Δ is in $absoutevalue - \|$ marks. This can be explained when the histograms are examined. Figure 3 shows the histogram before (e) and after (f) processing for Image 3. We can see that a lot of pixels have small values and the image has unbalanced contrast; when we compare this two histograms (e) and (f), we can see how many pixels were shifted to occur bigger values. When we look at the table I and table II amount of the blue pixels is smaller at this range; we have more red and green pixels shifted from different value. Thats why in the Image 3 our sky has been deformed, pixels have been shifted and contrast cahnged.

The similar effect appears in Image 4; the sky is also deformed, but when we look at the histogram 3 ((g),(e)), we will see that this histogram shows better balanced contrast; Contrast was changed by the algorithm, but that change wast taht big as at Image 3. Here we have a lot of pixels in the midde range called "Midtones". To help us understand what happend with the sky we can look at the table I. There is a huge amount of green and blue high valued pixels which will be moved at the process of equalisation. At this image $8,14804 \times 10^5$ pixels where shifted in all channels. Table III presents amount of pixels added from each channel and the percentage which they represent in the picture.

IV. Discussion & Conclusions

The results show that the proposed method works best when the image has poor intensity distribution and balanced contrast. If the image has most of the pixels set to low intensity value and non-balanced contrast,

TABLE I
THE TABLE SHOWS AMOUNT OF PIXELS OF EACH COLLOUR CHANNEL. THE NUMBER OF PIXELS IS DEVIDED BY 10^5

Image	Red_{pixels}	$Green_{\text{pixels}}$	$Blue_{\text{pixels}}$
Image 1	0,16499	0,31685	0,40966
Image 2	0,00328	0,02198	0,15061
Image 3	0,02211	0,05680	1,83034
Image 4	0,61143	3,22839	4,68551
Image 5	0,80754	2,05799	2,21122

TABLE II
THE TABLE SHOWS AMOUNT OF PIXELS IN RANGE [200;255] OF EACH COLLOUR CHANNEL AFTER HISTOGRAM EQUALISAION. AFTER OBSERVATION WE CAN SPOT THAT AMOUNTS OF PIXELS OF EACH COLOR ARE SIMILAR IN THE IMAGES.

Image	Red_{px}	$Green_{\text{px}}$	$Blue_{\text{px}}$
Image 1	0,14149	0,14	0,13951
Image 2	0,14225	0,14225	0,14172
Image 3	0,14225	0,14219	0,12692
Image 4	0,14173	0,13311	0,10245
Image 5	0,13890	0,12831	0,12289

the algorithm will stretch the historgram and image will appear deformed. Stretching histrogram during histogram equalisation method changes intensity and contrast. Also Image can appear with different colour distribution and change the hue of that image. Perfect example can be seen in Image 4 ((g),(h))2.

Therefore, a few key properties of the algorithm can be noted:

1) The algorithm works best when the histogram has good intensity distribution and amount of each colour pixels will have similar "Highlights" part of that histogram.
2) If the image has low contrast the alghoritm will stretch the intensity values and produce deformed Image.
3) The algorithm will work best when histogram of an image is correctly exposed.

Future improvement in Image Processing development:

1) Histogram equalisation works great when it is dealing with grayscale images. Sometimes the image can appear with pepper and salt noise after equalisation. To get rid of that effect I will use small kernel filter (3×3) or apply Gaussian filter.
2) Histogram equalisation has problem with low contrast images. It will work better when the input image will be transformed from RGB to HSV colour space and then apply equalisation on $'V'$ channel. That process will not change color values; it will change only intensity values.

I	$sum \times 10^5$	$Total\%$	$\lvert\Delta \sum_{i=200}^{K-1} p(i), \sum_{i=200}^{K-1} p'(i)\rvert$
I 1	0,89150	3.77	0,4705
I 2	0,17587	0.56	0,25583
I 3	1,90925	8.09	1,49789
I 4	8,52533	36.1	8,14804
I 5	5,07675	21.5	4,68665

TABLE III
THIS TABLE SHOWS THE AMOUNT OF PIXELS OF EACH COLOUR CHANNEL AND PERCENT VALUE OF ALL PIXELS. LAST COLUMN PRESENT THE DIFFERENCE BETWEEN NUMBER OF PIXELS BEFORE EQUALISATION AND AFTER. THE NUMBER OF PIXELS IS DIVIDED BY 10^5

3) The tables I, II, III showing how many pixels the alghoritm shifted during histogram equalisation process. Even if our eyes cannot detect big difference in the image composition (best example Image 5 ((i),(j)) Figure2), the amount of the pixels obtains the biggest intensity values are driasticly changed.

In conclusion, this paper has presented a method for automatic image enhancement based on histogram equalisation method for each colour channel separately. This histogram equalisation algorithm is not easily implemented in three different channels at the beginning. Firstly I tested algorithm for only one grayscale channel. When it was working and I figured out the method how to deal with one channel it was easy to implement it for three separated channels. The most important thing in doing a histogram equalisation is thresholding. Without that we will loose pixels and information from image. In some cases histogram equalisation is able to improve the image; in other it can completly destroy the image and produce unnatural looking image. However that method is my first of many steps in Image Processing and it can be extended in the future by adding different colour spaces and doing equalisation on different channels.

REFERENCES

[1] P. Toczko, "friend," https://www.facebook.com/pauli.toczi?fref=ts, she agreed to use her photo in this paper.
[2] "Gimp," http://www.gimp.org, accessed: November 2015.
[3] "Picasa," https://picasa.google.com/, accessed: November 2015.
[4] "Adobe photoshop," http://www.adobe.com/uk/products/photoshop.html, accessed: November 2015.
[5] "Jh labs," http://www.jhlabs.com/ip/blurring.html, accessed: November 2015.
[6] "Pythagorean triangle," https://www.mathsisfun.com/pythagoras.html, accessed: November 2015.
[7] D. H. Strange, "Histogram equalisation," 2015, lecture 4 / Lecture 5- Histogram Equalisation.
[8] "getColor(), Oracle Java SE7," http://docs.oracle.com/javase/7/docs/api/java/awt/Color.html, accessed: November 2015.
[9] "Histogram equalisation," http://developer.bostjan-cigan.com/java-image-histogram-equalization/, accessed: October 2015.
[10] C. P. Maria Petrou, *Image Processing: The Fundamentals*. John Wiley & Sons Ltd, 2010, page 370 at attached pdf page 391.

(a) Image 1 Original

(b) Image 1 Modified

(c) Image 2 Original

(d) Image 2 Modified

(e) Image 3 Original

(f) Image 3 Modified

(g) Image 4 Original

(h) Image 4 Modified

(i) Image 5 Original

(j) Image 5 Modified

(k) Image 6 Original

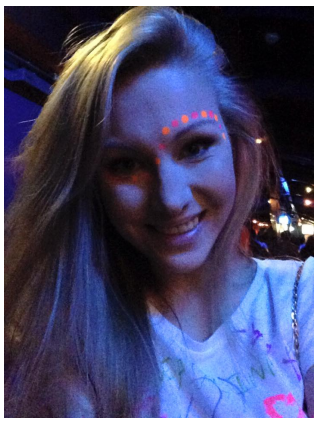
(l) Image 6 Modified

(m) Image 7 Original

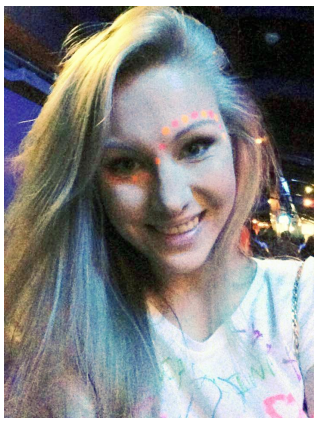
(n) Image 7 Modified

(o) Image 8 Original

(p) Image 8 Modifiedl

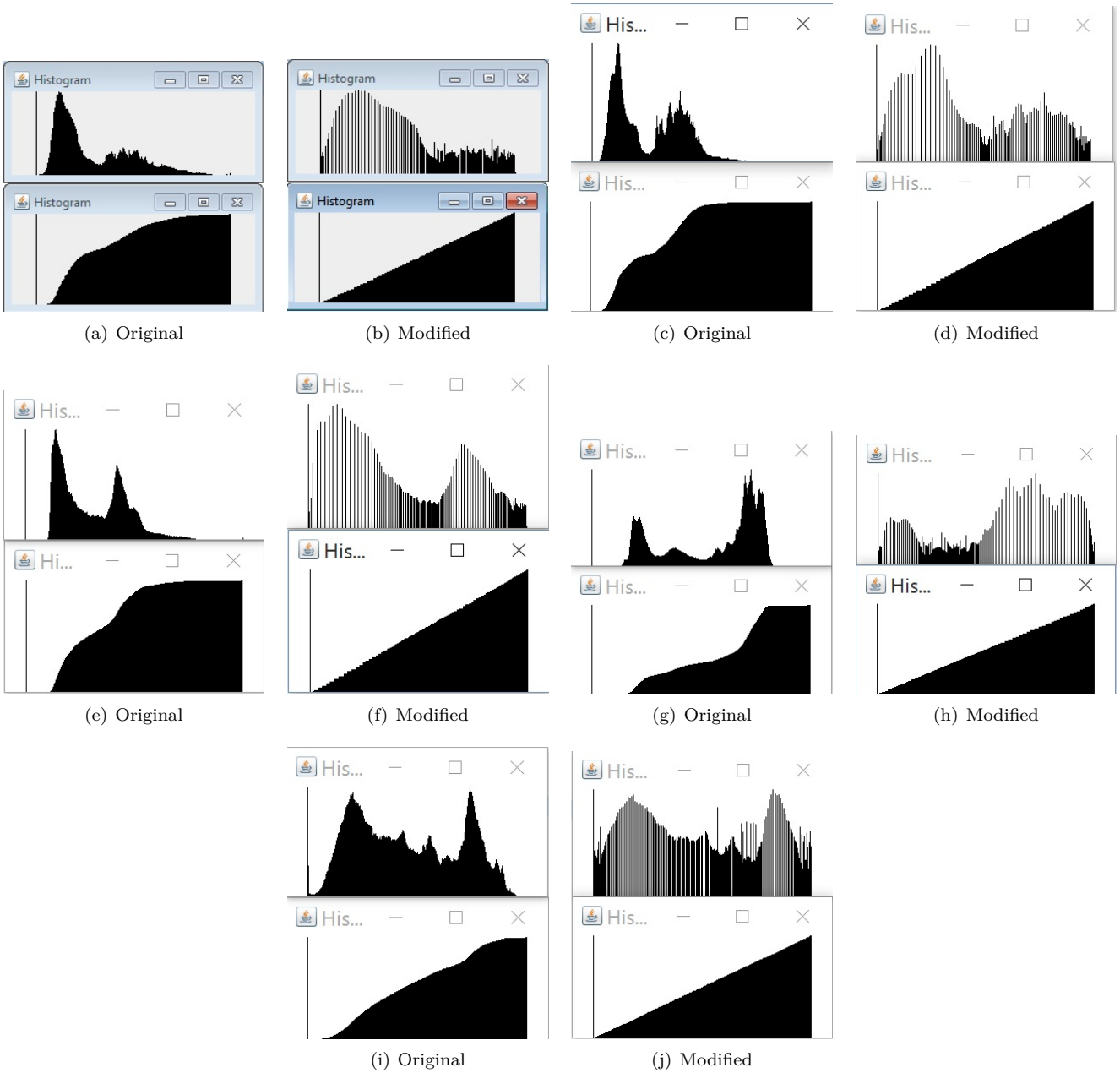


(q) Image 9 Original



(r) Image 9 Modified

Fig. 2. Results of performing the histogram equalisation algorithm on each of the five test images and three grayscale images. For the first two images (a, c), the algorithm produces really good looking results, brightness and conrast are balanced and details at the foreground are better presented, also the images have better colour balance (b, d). Performing the algorithm on Image 3 (e) causes a drastic change in colour and makes the image abstract (f). It could be argued that this manipulation of an image was something expected but it is very clear that colors that it created are not natural and cannot be compared to things we percieve during natural sunset, whereas the results for Image 4 (g) are far more natural and the image looks like picture taken during sunset (h). Performing the algorithm on Image 5 (i) we can see better contrast adjustment, edges are better exposed and colours are clearer (j). When we perform the algorithm on 3 grayscale images results are similar. Edges and shadows are corrected, details in the pictures look better. We can easily find the difference by comparing background in Image 6 (k) with equalised Image (l). In the background of an Image we can spot the difference in the mountains.



(a) Original (b) Modified (c) Original (d) Modified

(e) Original (f) Modified (g) Original (h) Modified

(i) Original (j) Modified

Fig. 3. Average RGB histograms and cumulative histograms from images 1-5 (Figure 2 (a - j)). The effect seen in Images at Figure 2 occurs because all of the pixels were mapped to create cumulative histogram (a) and shifted by the alghoritm to create new cumulative histogram (b) similar to the triangle. We can spot that the modified histogram (b/d/f/h/j) has beed stretched so in some places we havemissing values; it is the result of histogram equalisation. We can observe that modified cumulative histograms are not always linear.

(a) Image 1 Original (b) Image 1 Harrys (c) Image 1 Mine

(d) Image 2 Original (e) Image 2 Harrys (f) Image 2 Mine

(g) Image 3 Original (h) Image 3 Harrys (i) Image 3 Mine

(j) Image 4 Original (k) Image 4 Harrys (l) Image 4 Mine

(m) Image 5 Original (n) Image 5 Harrys (o) Image 5 Mine

Fig. 4. This Figure shows us the comparision between algorithm developed by Harry Strange and mine. As we can see four of five images look better. Only image 3 (f) compared to Harrys (e) look worst; maybe if I would change the sky after histogram equalisation the image will look better, but for now Image is not natural. Algorithm proposed by Harry in most of the Images show washed out results ((b),(e),(h)). His Image 5 (n) doesn't change most. The contrast is better balanced only. Probably if I would use his algorithm before the equalisation the results will be a lot better. Although the 3rd Image (i) is destroyed I think that my solution is better at all because Harrys algorithm present washed out effect in most of the images, where mine solution has problem with low balanced contrast images. All of the images with poor intensity destribution looks better a lot ((a),(d)).