1) (a) $N_L$=450 lines per frame
$N_P$=520 pixels per line
$N_{FPS}$=25 Hz
P=(4*8+2*8+0*8)/4=12 bits/pixel
Bit rate = $N_L$ * $N_P$ * $N_{FPS}$ * P
= 450 *520*25*12
=70200000 bits/sec

(b) P=(4*8+2*6+0*6)/4 = 11 bits/pixel
Bit rate= $N_L$ * $N_P$ * $N_{FPS}$ * P
= 450*520*25*11
= 64350000bits/sec

So for 10 min bit rate=64350000*60*10
=3.861*10^10
=38610 Mbits

2) (a)
Given the interval [-4,4], we can calculate the step size as follows
Step Size=(Xmax-Xmin)/level; here level is 32
So, Step Size=(4-(-4))/32=0.25
Therefore quantization values are -3.75, -3.50, -3.25, -3.00, -2.75, -2.50, -2.25, -2.00, -1.75, -1.50, -1.25, -1.00, -0.75, -0.50, -0.25, 0, 0.25, 0.50, 0.75, 1.00, 1.25, 1.50, 1.75, 2.00, 2.25, 2.50, 2.75, 3.00, 3.25, 3.50, 3.75,

Quantization sequence:
1.75, 2.25, 2.25, 3.25, 3.25, 3.25, 2.50, 2.75, 2.75, 1.5, 1.0, 1.25, 1.25, 1.75, 2.25, 2.25, 2.25, 2.0, 2.25, 1.25, 0.25, -1.25, -1.25, -1.75, -1.0, -2.25, -1.50, -1.50, -0.75, 0, 1.0

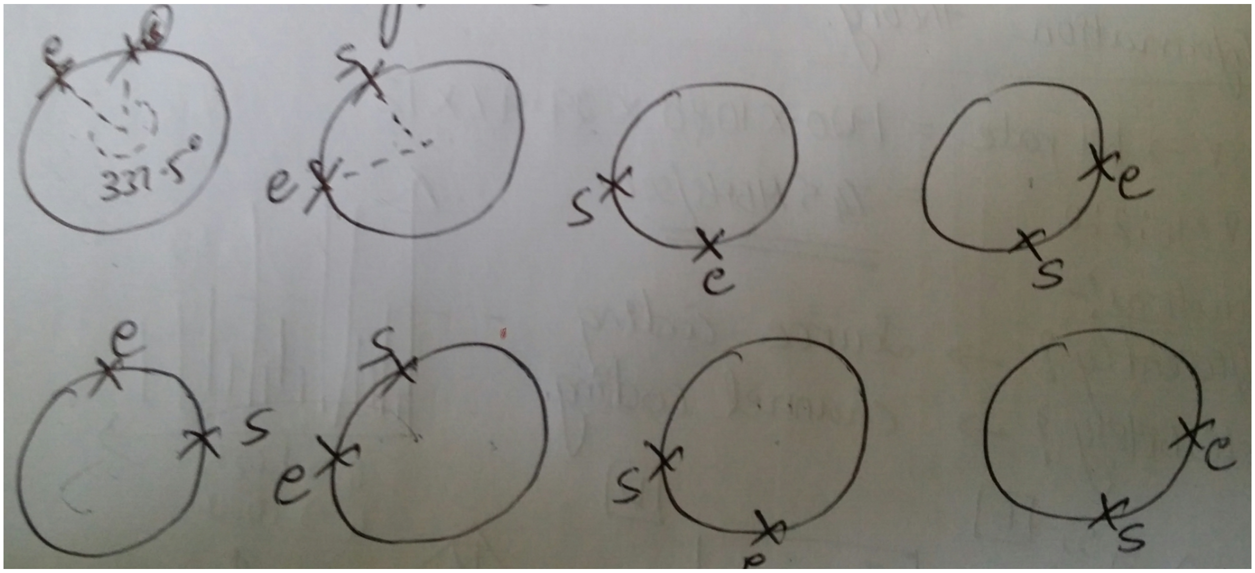
Level: 22, 24, 24, 28, 28, 28, 25, 26, 26, 21, 19, 20, 20, 22, 24, 24, 24, 23, 24, 20, 16,10, 10, 8, 11, 6, 9, 9 ,12, 15, 19.

(b) Level=32=2^5, so 5bits
Total number of bits =5*31 = 155 bits

3) (a) Speed = 36 Km/hr
Diameter= 0.4244 m
Circumference= $2\pi r$= 0.4244*3.14= 1.332616 m
Distance covered in a sec =1/100 Km
= 10m/sec
1 rotation = 1.332616 m
? = 10m
Therefore total rotation is 7.50 rot/sec

(b) As recorded by the Camcorder with frame rate of 8 fps, the wheel in the projected movie is rotating with the rate of 7.50 rot/sec. So the degree by which the wheel rotates for a frame when recorded by camcorder is (360*7.5)/8=337.5 degree.

In other words, the wheel is rotating in the opposite direction for an angle of 22.5 degree as perceived by the user (cause of this is aliasing).



(360-337.5)degree=22.5 degree
Therefore for 8 Frames, the angle by which it rotates is (22.5*8)= 180degree.

Number of rotation done by the wheel is 180/360 = 0.5 rot/sec , that is half rotation. The rotation is seen by the user in the opposite direction of the actual rotation of the wheels.

C) In order to prevent aliasing the frame rate should be half the Nyquist rate, frame rate can be at max 15fps.
1 rotation - 1.332616m.

Therefore Max speed of the car = 15*1.33=19.95m/s

In km/hr the speed is (19.95*60*60)/1000=71.82 Km/hr

Analysis 1 :
The primary method to calculate the error is using PSNR (Peak Signal to noise ratio), here the signal is the original and noise is the resultant data. $MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$,
PSNR=20*log$_{10}$ (MAX)-10*log$_{10}$(MSE).
(Site:https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio)

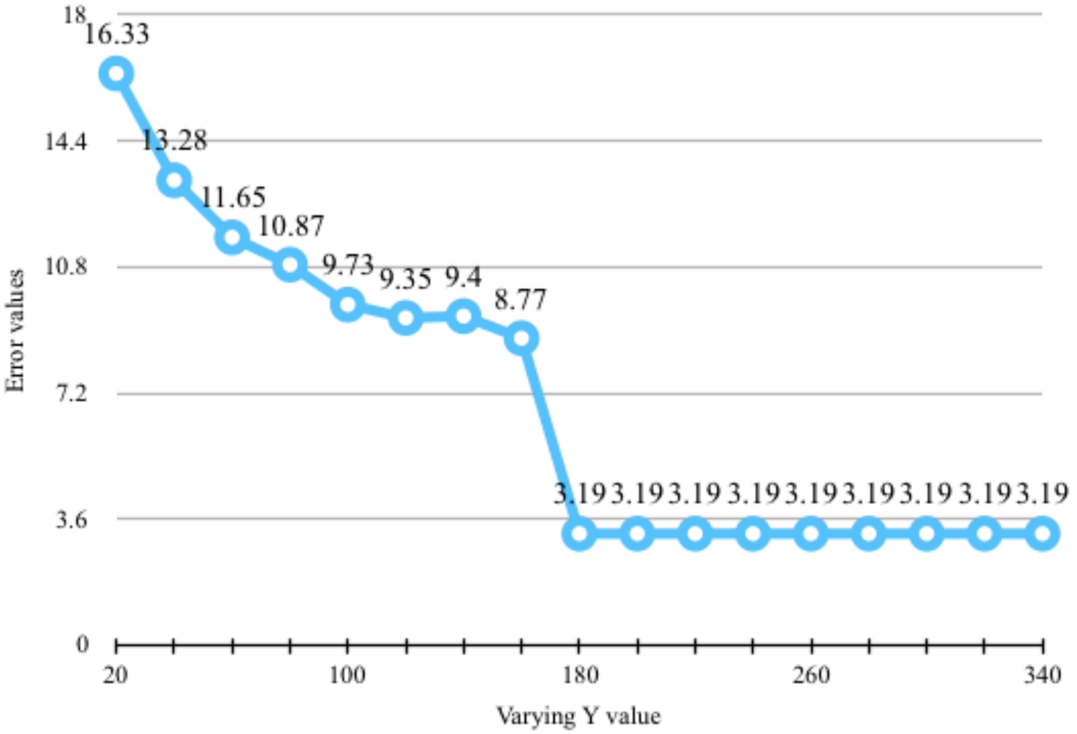


Fig 1: Graph obtained by varying Y along X axis

This is a graph obtained by varying Y keeping U and V constant. The curve is declining till some value of Y, after which it remains the same. As the value of Y increases the error metric (PSNR in dB) value decreases but not the error. This is due to loss of sample (luminance) in the original image. When the value of Y exceeds half the value of width, the error metric remains constant from then.

The error vales along Y-axis are in dB and interval along X-axis is 20 for Y values.

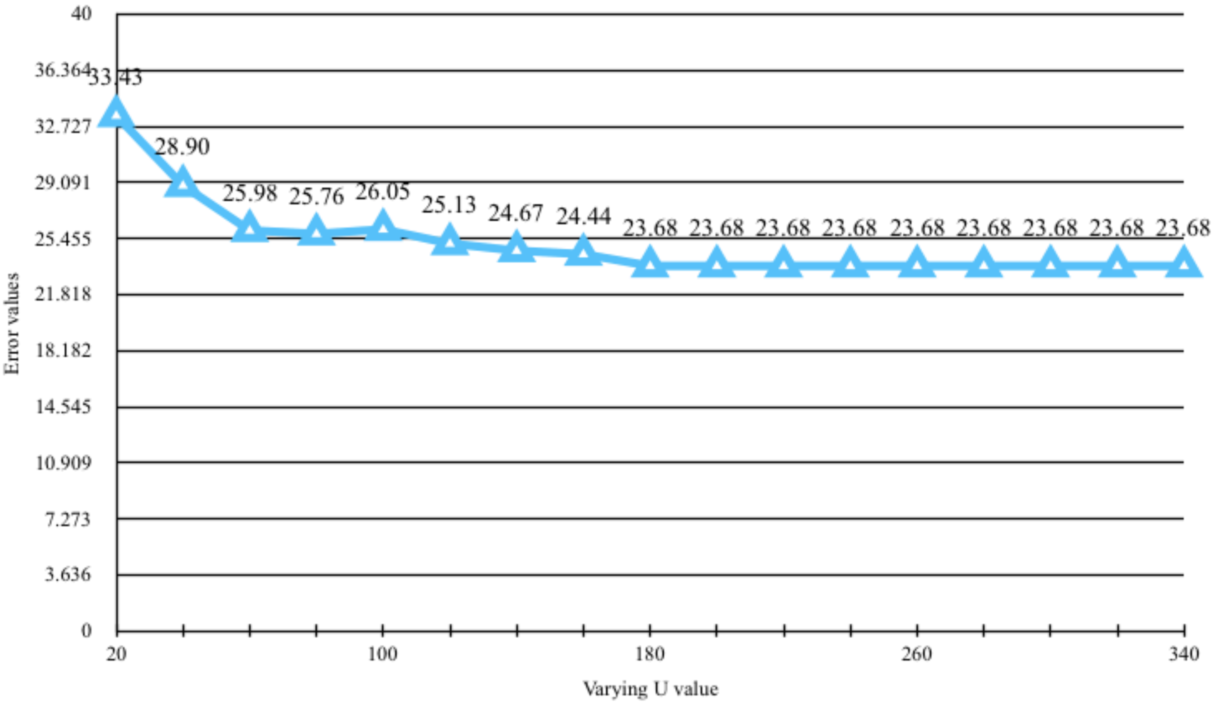


Fig 2: Graph obtained by varying U along X-axis

From the above graph we can noticed that the PSNR value (error metric) reduce to some extent and at one point in time the value increases which further simultaneous decreases in its later interval. After some point the value remains the constant. The values are increasing and decreasing and decreasing, which is not visible in the graph clearing because graph is plotted for an interval of 20 along X axis. But at few points it is visible the there is increase and decrease in the error metric values. Here Y axis represents error value in dB and X-axis the value of U for an interval of 20.

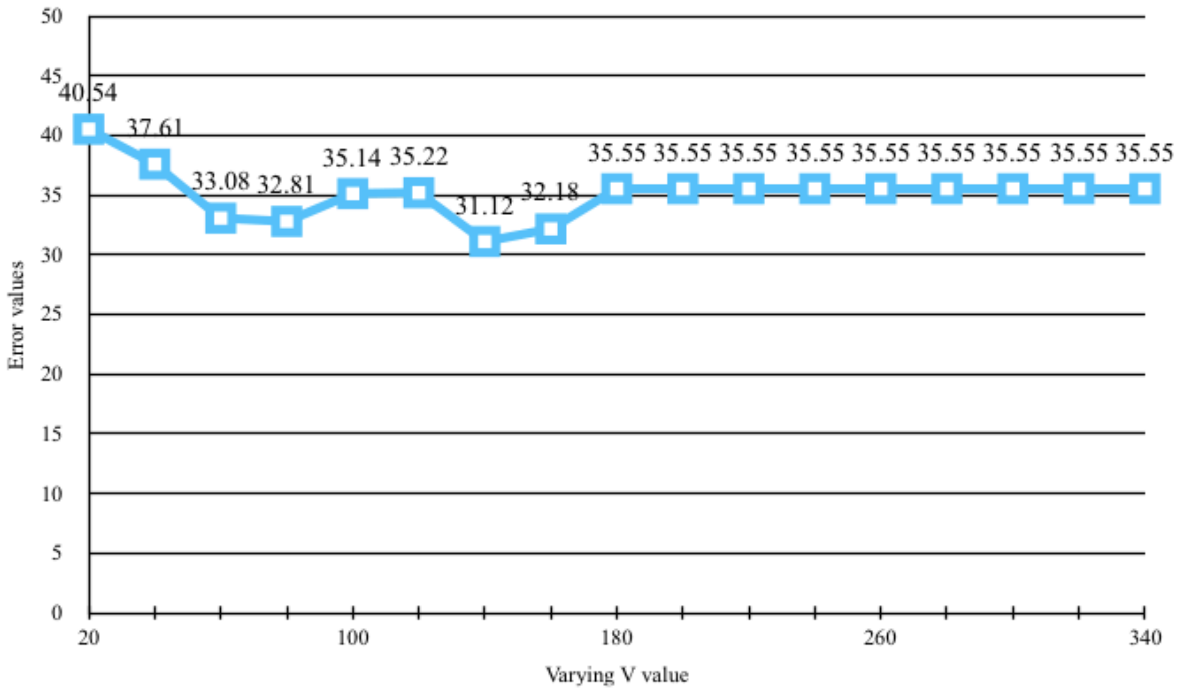


Fig 3: Graph representing varying V along X-axis with error along Y-axis

In the above graph, the value of the error metric for given value of V is increasing and decreasing without any pattern in it. At on point in time the error metric value remains the same. There is a drastic decrease at some value of V compared to it neighboring values.

In conclusion, the error value does not depend on the value of Y, V or U, it basically depends on change in the image introduced by sampling (for image compression).

Analysis 2:

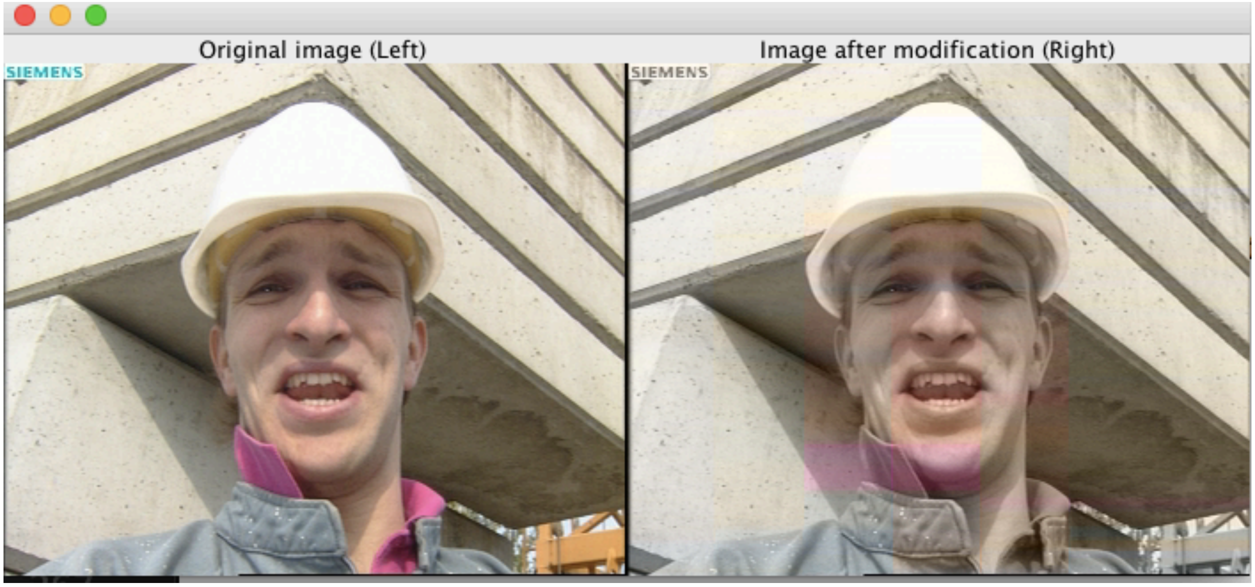
(A) I observe Color artifacts after sampling the images along U and V keeping Y constant at 1. The main cause of color artifacts (composite artifacts) is decrease in color saturation within in fine detail in the image. The decrease in color resolution is not visible in fine colored image but visible in less detailed image. The main cause of this artifacts in my program is sampling, where each lossy pixel values are represented using their adjacent pixel value (averaging between them) which causes them to have faded appearance in the output image. The artifacts and improvised image is shown below (C)

(B) One method that would reduce the artifacts(error) to some extent is to calculate distance between each pixels Y, U, V value with it know neighbors (the pixel value that will not be sampled). Then setting the pixel value for the lost pixels with its nearest neighbors value. The distance between a pixel and its adjacent is found using Euclidean Distance [$d(x,y) = SQRT( (R_x-R_y) + (G_x-G_y) + (B_x-B_y) )$ ]. This will reduce the artifacts to some extent compared to averaging method.

```
Algorithm: To improve output image after sampling
Step 1 : Start
Step 2 : Declare three 2 dimension array to store Y U V values
Step 3 : For height in image height, do
Step 4 : For width in image width, do
Step 5 : For span in width+1, do
Step 6: Use Euclidean Distance to calculate distance between neighboring pixel vales.
Step 7: if array of span (array[span]) minus array of width (array[width]) is less than array of width +(y or v or u) value i.e (array[width+y]-array[span], array[width+u]-array[span], array[width+v]-array[span]) minus array of span then
            7.1 : array of span = array of width (array[width]), else
            7.2 : array of span = array of width+(y or u or v), (array[width+y], array[width+u], array[width+v])
Step 8: Perform quantization.
Step 9:Output image
Step 10: end
```

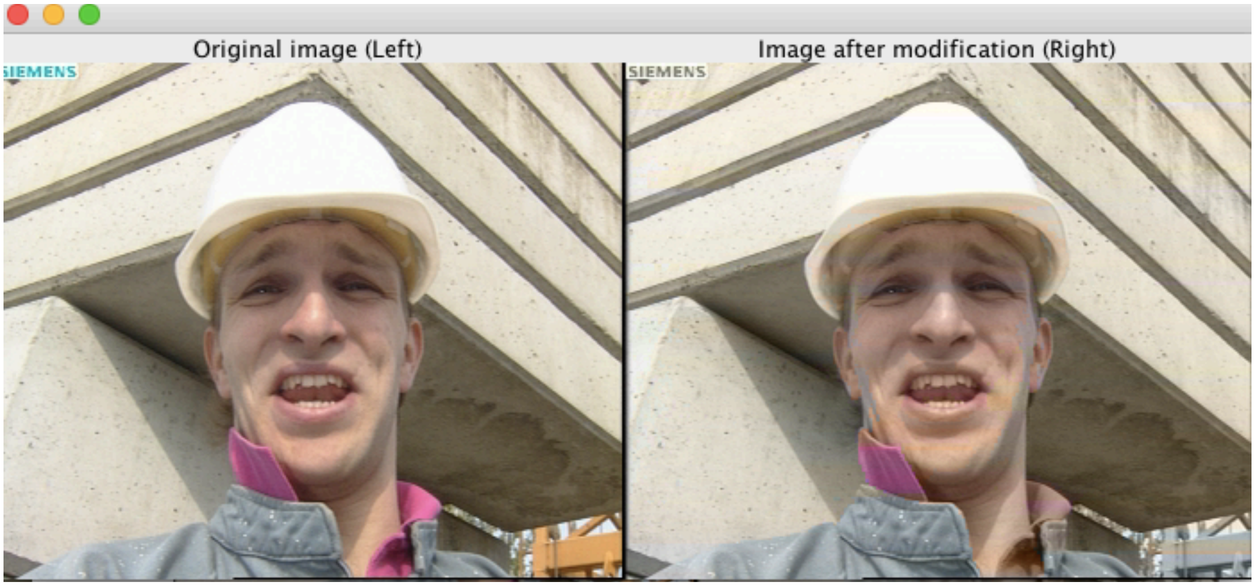
In step 5 for loop is traversing only the lost pixel and calculating the Euclidean difference between the neighbor pixels, if the right neighbor is less than the left neighbor then copy value of right neighbor to array[span]. Continue this process to rest of the values along width and height dimension.

C) Before modifying the algorithm
Sampling ratio is 1:50:50 256



This is the original image and modified image before applying the algorithm, where in you can see faded appearance of color near color and alongside the face. This is due to sampling values, where the intermediate YUV values are thrown away and replaced with average of its neighbor. Even the color quality is not that good because of averaging the values between pixel and setting all the lossy pixel value to the same averaged value.

After modifying the existing algorithm
Sampling Ratio is 1:50:50 256



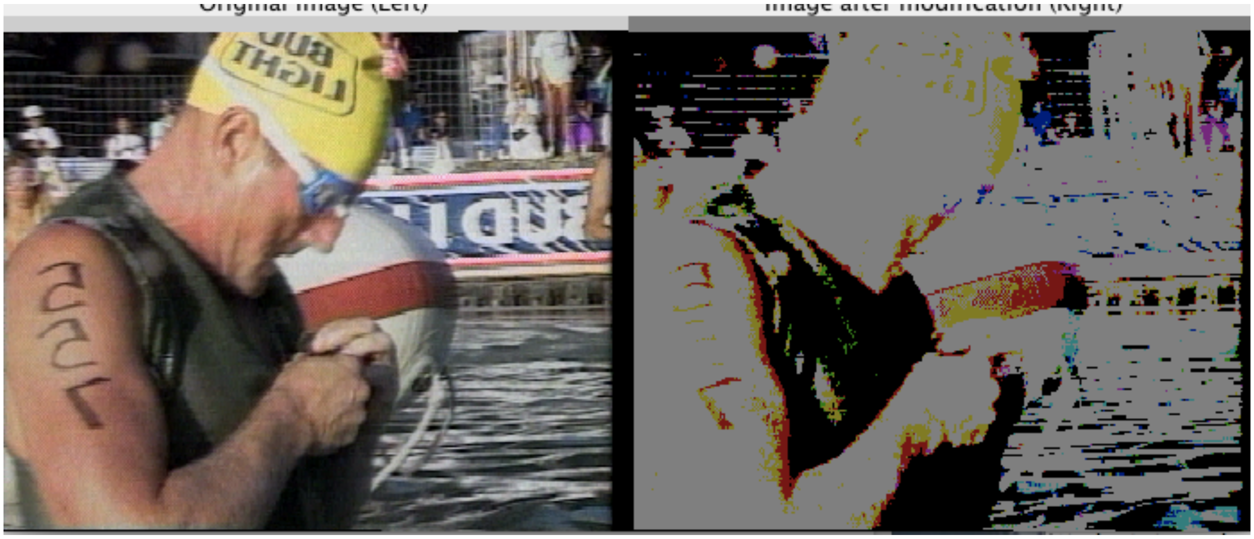
The artifacts is reduced in the second image compared to the first image after using different filtering approach.
The reducing in the artifacts is due to sampling the value of lossy pixel with the pixels who value closely matches with that pixel value. But as seen in the picture there is also some artifacts in the image near the color which show a Smooth faded appearance along the color (pink). The color quality is increased in this compression algorithm.

The modified algorithm is better than the original algorithm is because of the quality of the image it produces and the reduction in the number of artifacts in the image that is visible to human eye.

Analysis 3:
(A)
Algorithm: Quantization using histogram for various region in the image.
Step1 : Start

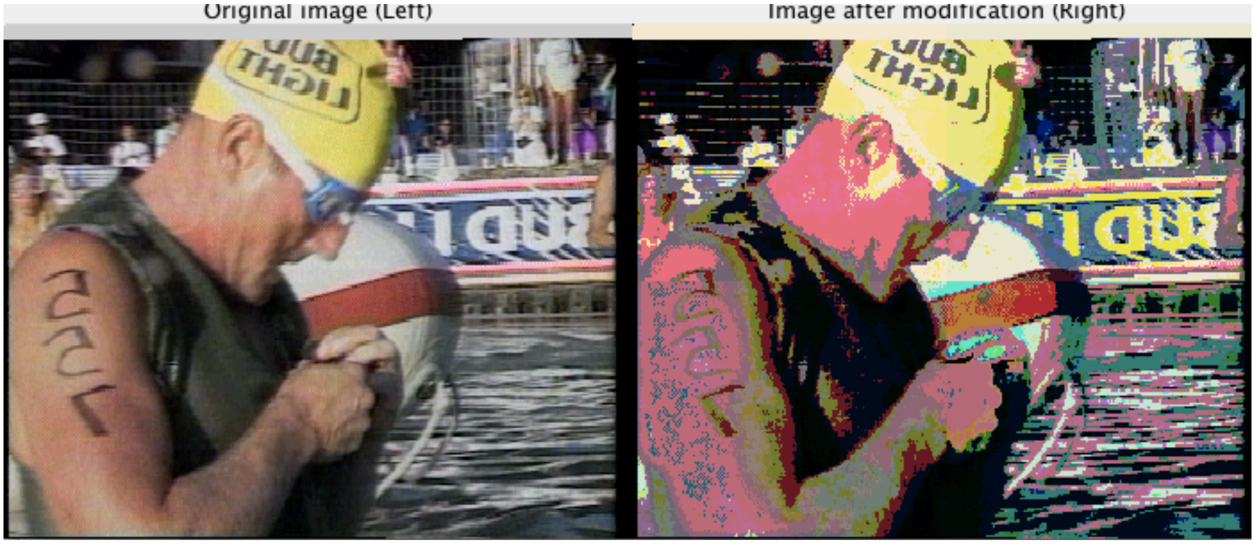
Step 2 : Divide the image into 4 parts, for each part of the image find the histogram for the image. Choose the most repeated color in the histogram as a base colors for quantization.
Step 3: Perform sampling if required by the user
Step 4: Apply quantization separately to all the four part of the image by using the most frequent color value.
Step 5: For each R, G, B values find to which level does the value quantized to (rounding to the nearest Quantized values)
Step 6 : Combine all four parts of image to get the output image.
Step 7: Output the image.
Step 8: end

(b) Before applying new algorithm for quantization.
For 1:1:1 2



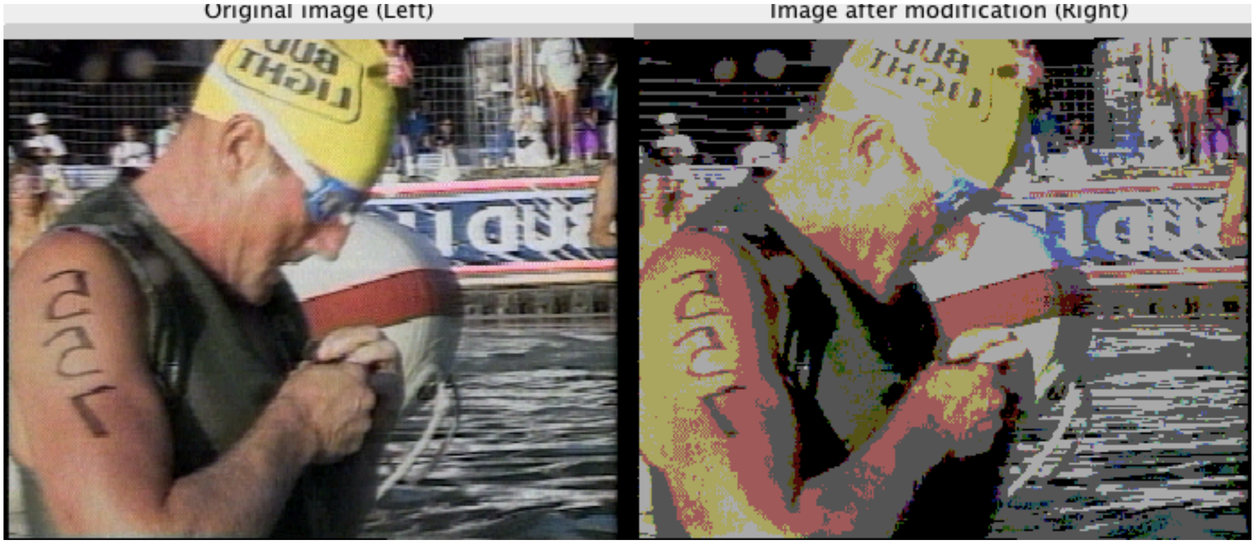
Since Q=2, there are only 8 color in the image 2 per channel as a result of which the image is not that good.

After applying algorithm to improve quantization of the image
For 1:1:1 2



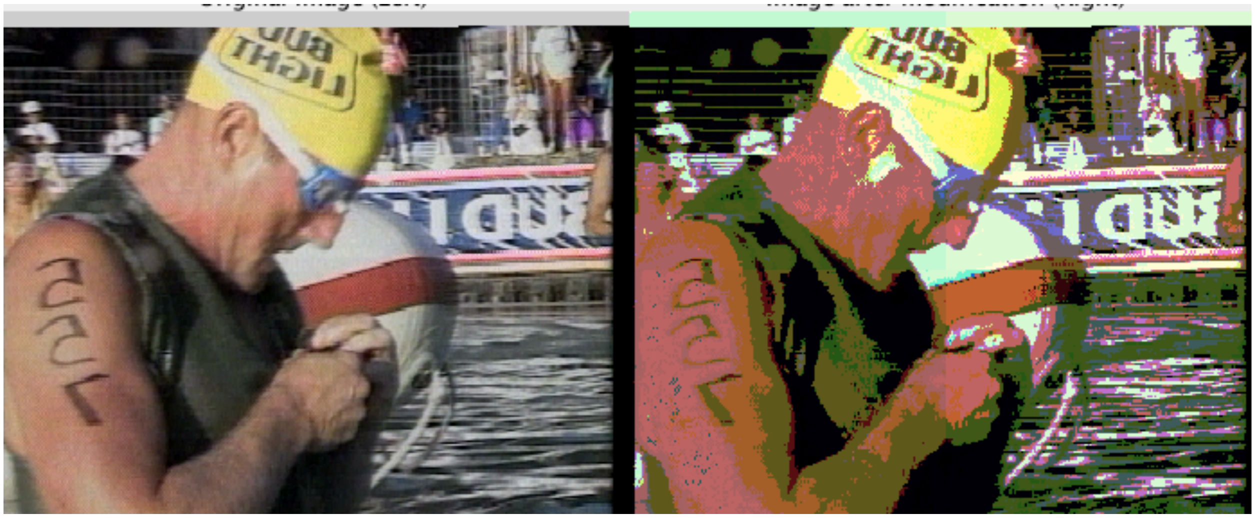
The use of histogram increased the quality of the image along all the four regions in the image. It is basically taking the max occurred color and quantizing all the other color to that value.

For 1:1:1 3
Before modification



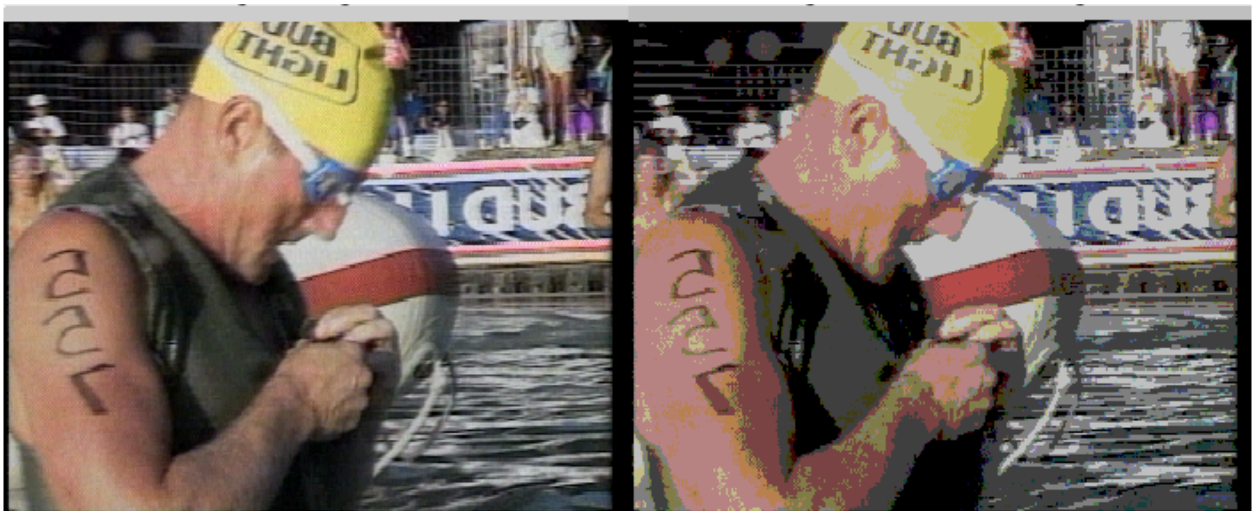
For Q=3 the the image looks a little better be the range of color that can be used has increased.

After modification using histogram.
1:1:1 3



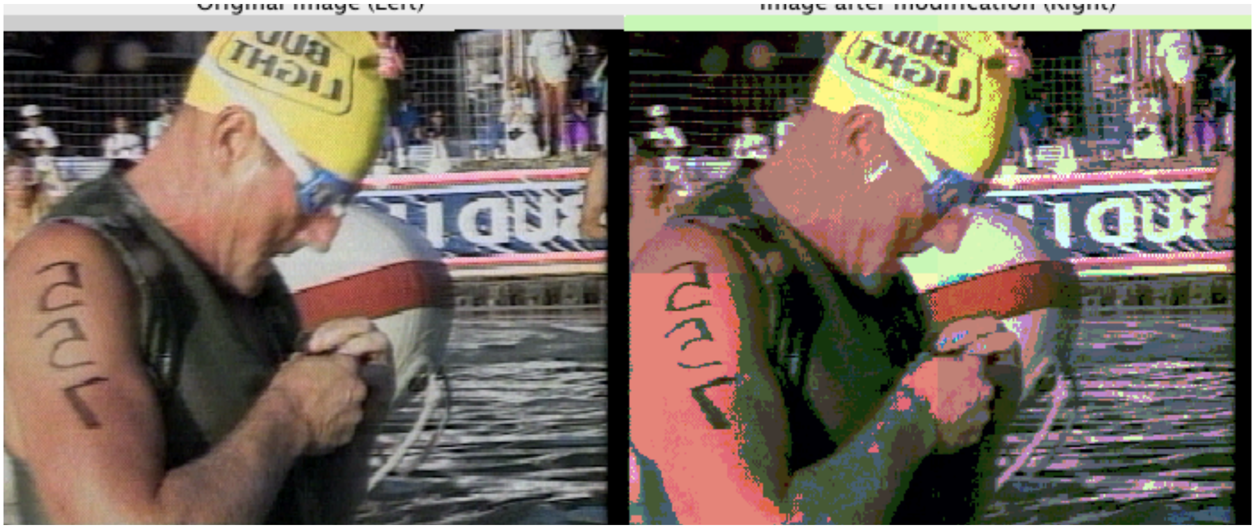
In this image there is an increase in the colors to represent an image, with few fine details as seen in this image compared to the previous image for the same quantization value.

Before Modification 1:1:1 4



This is for Q=4, when Q=4 there are more color to represent an image as a result of the image quality has increased compared to Q=3.

After Modification
1:1:1 4



Applying the algorithm I mentioned will increase the quality further by taking the max repeating value of colors in the image.

The idea of histogram to apply in my algorithm is drawn from the below link http://www.leptonica.com/color-quantization.html, where the image color is represented in histogram. But I have modified the concept based on my is idea by dividing the image into 4 part so that there is maximum color available along all the 4 regions rather than having only one range of values for entire image.

This algorithm increases the visual quality of the image by having the color that repeated the most across all the region and rounding the original image back after quantization back to it. In the previous method the algorithm was just rounding the values to it nearest values based on the error difference between them. In other words, rounding the value to the neighbor will less error. This will decrease the quality of the images the number of quantization level decreases. But in the modified algorithm the quantization level remains the same but takes the maximum repeated color value to increase the quality of the image.