**CHROMA AND LUMA MODIFICATION**

**BASED IMAGE COMPRESSION**

**PROJECT REPORT**

**Submitted by**

***In partial fulfilment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

***in***

**ELECTRONICS AND COMMUNICATION ENGINEERING**



**UNIVERSITY COLLEGE OF ENGINEERING, TINDIVANAM**

**ANNA UNIVERSITY : CHENNAI-600025**

**JUNE - 2022**



**BONAFIDE CERTIFICATE**

This is to certify that this project report “**chroma and luma modification Based image compression**” is the bonafide work of \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

Dr. K.Venkatalakshmi,M.E.,Ph.D., Mr.B.Pradeep Khanth

**HEAD OF THE DEPARTMENT SUPERVISOR**

Assistant Professor, Lecturer,

Department of ECE, Department of ECE,

University College of Engineering, University college of Engineering, Tindivanam, Tindivanam,

Melppakkam-604001. Melpakkam-604001.

Submitted for the University project Viva-Voce held on \_\_\_\_\_\_\_\_\_\_

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

At the outset we like to express our gratitude to the God almighty. We express our thanks to our Dean(i/c) Dr.P.THAMIZHAZHAGAN, M.E., Ph.D., for his constant inspiration and encouragement throughout the project.

For mostly , we pay our grateful acknowledgement and extend our sincere attitude to our Head of the Department Dr.K.Venkatalakshmi, M.E.,Ph.D. we extend our sincere thanks to Mr.B.Pradeep Khanth, M.E., Project internal guide for better guidance and constant encouragement in completing this project work successfully.

We thank all our teaching and non –teaching faculty members of the department and our fellow friends for helping us in providing valuable suggestions and timely ideas for the successful completion of the project.

And also, we extend our immense thanks to our parents and siblings who have been a great source of inspiration and rendered us great support during the course of this project work . We sincerely thank all of them.

**Arunachalam J**

**Berjin Singh J**

**Chandrasekaran T**

**Gunapathi M**

**TABLE OF CONTENTS**

**CHAPTER TITLE PAGE NO**

I ABSTRACT

1. INTRODUCTION

Image Compression Fundamentals

Colour Specification

Spatial Sampling of Colour Component

The Flow of Image Compression Coding

Need of Compression

Principle of Compression

2. COLOR CONVERSION

3. DOWN SAMPLING########

4. DISCRETE COSINE TRANSFORM

5. ZIGZAG SCANNING

6. AVERAGE METHOD#########

7. INTERPOLATION

BILINEAR INTERPOLATION

BICUBIC INTERPOLATION

8. EQUALIZATION######

9. PERFORMANCE ANALYSIS

10. IMPLEMENTATION AND RESULTS#######

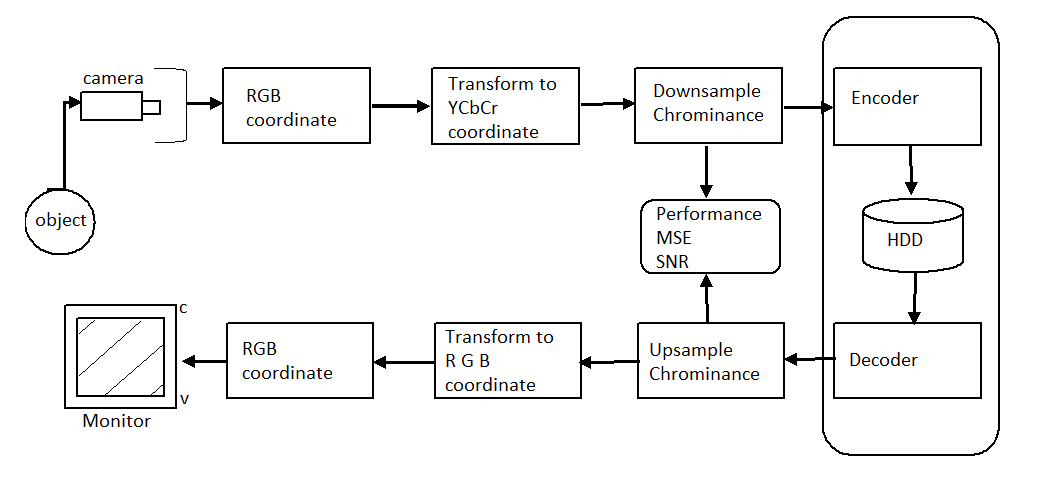
**ABSTRACT**

In color image compression, the chroma components are often sub-sampled before compression and up-sampled after compression. Although sub-sampling the chroma components saves the bit-cost for compression, it often induces extra colour distortions in the compressed images. In this letter, we propose two approaches to tackle this problem. First, we propose a sub-sampling method in the transform domain and apply it to both chroma components. Then, based on this sub-sampling, we propose a method to modify the luma component. In our proposed luma modification algorithm, the distortions that occurred in the two chroma components can be coupled together and utilized to modify the luma component. With our proposed chroma sub-sampling and luma modification algorithms, we can achieve a low RGB distortion in practical image coding. The experimental results demonstrate that our proposed methods offer more significant coding gains compared with the state-of-the-art methods for the compression of Color images.

**1.Introduction**

**1.1 Image Compression Fundamentals**

Image compression is an application of data compression that encodes the original image with few bits. The objective of image compression is to reduce the redundancy of the image and to store or transmit data in an efficient form. Fig 1.1 shows the block diagram of the general image storage system. The main goal of such system is to reduce the storage quantity as much as possible, and the decoded image displayed in the monitor can be similar to the original image as much as can be. The essence of each block will be introduced in the following sections.



**1.2 Colour Specification**

The Y, Cb, and Cr components of one color image are defined in YUV color coordinate, where Y is commonly called the luminance and Cb, Cr are commonly called the chrominance. The meaning of luminance and chrominance is described as follows:

**Luminance:** received brightness of the light, which is proportional to the total energy in the visible band.

**Chrominance:** describe the perceived color tone of a light, which depends on the wavelength composition of light chrominance is in turn characterized by two attributes – hue and saturation.

1.**hue:** Specify the color tone, which depends on the peak wavelength of the light

**2.saturation**: Describe how pure the color is, which depends on the spread or bandwidth of the light spectrum

**1.3 Spatial Sampling of Colour Component**

Because the eyes of human are more sensitive to the luminance than the chrominance, the sampling rate of chrominance components is half that of the luminance component. This will result in good performance in image compression with almost no loss of characteristics in visual perception of the new upsampled image.

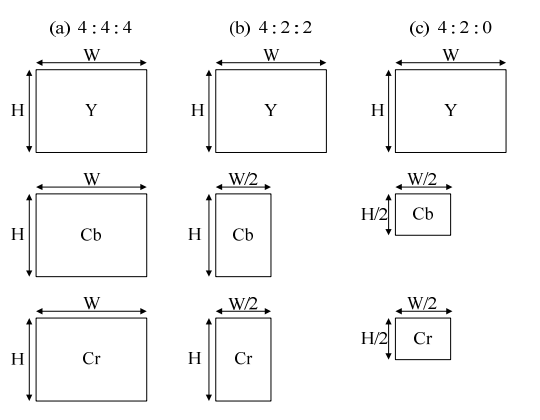


Fig. 1.2 The three different chrominance downsampling format

There are three color formats in the baseline system:

**4:4:4 format:** The sampling rate of the luminance component is the same as those of the chrominance.

**4:2:2 format:** There are 2 Cb samples and 2 Cr samples for every 4 Y samples. This leads to half number of pixels in each line, but the same number of lines per frame.

**4:2:0 format:** Sample the Cb and Cr components by half in both the horizontal and vertical directions. In this format, there are also 1 Cb sample and 1 Cr sample for every 4 Y samples.

At the decoder, the down-sampled chrominance components of 4:2:2 and 4:2:0 formats should be up-sampled back to 4:4:4 format.

**1.4 The Flow of Image Compression Coding**

What is the so-called image compression coding? Image compression coding is to store the image into bit-stream as compact as possible and to display the decoded image in the monitor as exact as possible. Now consider an encoder and a decoder as shown in Fig. 1.3. When the encoder receives the original image file, the image file will be converted into a series of binary data, which is called the bit-stream. The decoder then receives the encoded bit-stream and decodes it to form the decoded image. If the total data quantity of the bit-stream is less than the total data quantity of the original image, then this is called image compression. The full compression flow is as shown in Fig. 1.3.

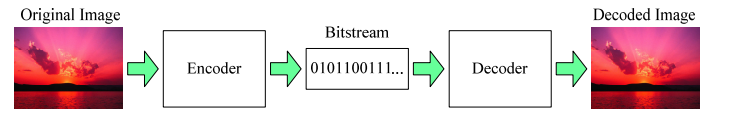
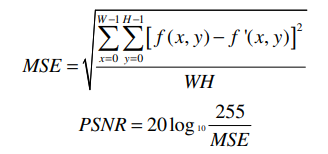


Fig. 1.3 The basic flow of image compression coding

In order to evaluate the performance of the image compression coding, it is necessary to define a measurement that can estimate the difference between the original image and the decoded image. Two common used measurements are the Mean Square Error (MSE) and the Peak Signal to Noise Ratio (PSNR), which are defined in below respectively. f(x,y) is the pixel value of the original image, and f’(x,y)is the pixel value of the decoded image. Most image compression systems are designed to minimize the MSE and maximize the PSNR.



**1.5** **Need of Compression**

Uncompressed images can occupy a large amount of memory in RAM and in storage media, and they can take more time to transfer from one device to another. Table 1 below shows the comparative size from normal text to high compressed image. Examples given in Table1clearly shows need for sufficient storage space and more bandwidth because long transmission time is required for uncompressed image. So there is only one solution is to compress the image.

**1.6** **Principle of Compression**

Digital image is basically array of various pixel values. In the digital image Pixels of neighborhood are correlated and so that this pixels contain redundant bits. By using the compression algorithms redundant bits are removed from the image so that size image size is reduced and the image is compressed. Image compression Have two main Components: redundancy reduction and irrelevant data reduction redundancy reduction is achieved by removing extra bits or repeated bits. While in irrelevant reduction the smallest or less important information is omitted, which will not received by receiver. There are three types of redundancies. Coding redundancy is present when less number of code words required instead of larger symbol. Inter pixel redundancy results from correlated pixels of an image. In psycho visual redundancy data is ignored by the normal visual system. Image compression is applied to reduce the number of bits which represent the image.

**2. Color Conversion**

The primary colors are the building blocks of [color theory](https://www.colorsexplained.com/color-theory/" \t "_self). Primary Colors of Light are **Red, Green, Blue**. The RGB primary commonly used for color display mixes the luminance and chrominance attributes of a light. In many applications, it is desirable to describe a color in terms of its luminance and chrominance content separately, to enable more efficient processing and transmission of color signals.

One important task in image processing applications is the color space conversion. Real-time images and videos are stored in RGB color space, because it is based on the sensitivity of color detection cells in the human visual system. In digital image processing the YCbCr color space is often used in order to take advantage of the lower resolution capability of the human visual system for color with respect to luminosity. Thus, RGB to YCbCr conversion is widely used in image and video processing.

Given a digital pixel represented in RGB format, 8 bits per sample, where 0 and 255 represents the black and white color, respectively, the YCbCr components can be obtained according to equations

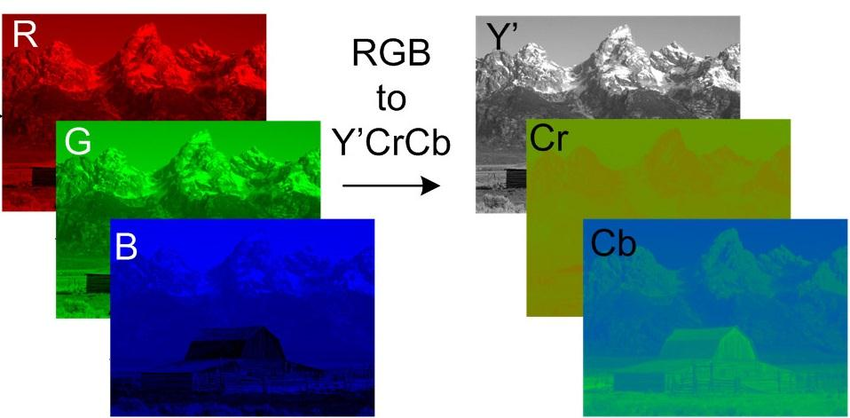
Yi 0.257 0.504 0.098 Ri 16

Cbi = −0.148 −0.291 0.439 Gi + 128

Cri 0.439 −0.368 −0.071 Bi 128

Image and Video consumes a lot of data. One of the reasons is because they are represented in the RGB format. However, is not worth to store or transmit information in this color space representation, once it has a large bandwidth. Thus all the pixels should be converted to YCbCr to accomplish that.

To understand the effect of converting RGB to YCbCr,

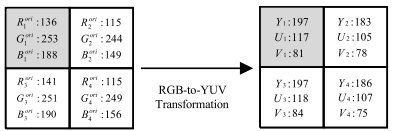


Similarly, if we would like to transform the YUV coordinate back to RGB coordinate, the inverse matrix can be calculated from (1.1), and the inverse transform is taken to obtain the corresponding RGB components.

Ri 1.164 0 1.596 Yi − 16

Gi = 1.164 −0.391 −0.813 Ui − 128

Bi 1.164 2.018 0 Vi − 128





**3. Down sampling**

We have to downscale the image for various reasons like:

* It makes the data of a more manageable size
* Reduces the dimensionality of the data thus enabling in faster processing of the data (image)
* Reducing the storage size of the data

There are also some other uses of this technique depending on the usage.

It is sometimes confused with image compression which is a different thing and serves a different use altogether. Here we are concerned with just the shrinking of the image. Well, what does that mean? That essentially means throwing away some of the (non-essential) information.

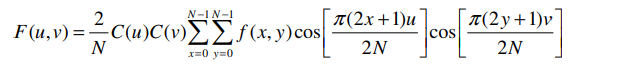
From this, we can draw a hint that we need to discard some of the rows and/or columns from the image. We need to give away some of the information.

There are many algorithms used in various techniques for down sampling

**4. Discrete Cosine Transform**

The next step after color coordinate conversion is to divide the three color components of the image into many 8×8 blocks. The mathematical definition of the Forward DCT and the Inverse DCT are as follows:

**Forward DCT**

****

****

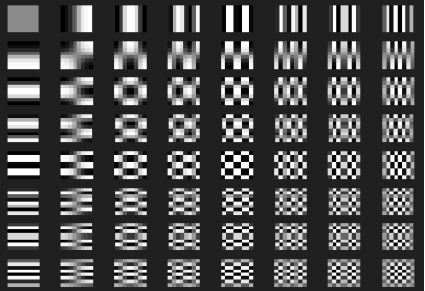


The f(x,y) is the value of each pixel in the selected 8×8 block, and the F(u,v) is the DCT coefficient after transformation. The transformation of the 8×8 block is also a 8×8 block composed of F(u,v).

The DCT is closely related to the DFT. Both of them taking a set of points from the spatial domain and transform them into an equivalent representation in the frequency domain. However, why DCT is more appropriate for image compression than DFT? The two main reasons are:

1. The DCT can concentrate the energy of the transformed signal in low frequency, whereas the DFT can not. According to Parseval’s theorem, the energy is the same in the spatial domain and in the frequency domain. Because the human eyes are less sensitive to the low frequency component, we can focus on the low frequency component and reduce the contribution of the high frequency component after taking DCT.

2. For image compression, the DCT can reduce the blocking effect than the DFT. After transformation, the element in the upper most left corresponding to zero frequency in both directions is the “DC coefficient” and the rest are called “AC coefficients.”



**Inverse DCT**

The inverse discrete cosine transform (IDCT) decodes an image into the spatial domain from a representation of the data better suited to compaction. IDCT-based decoding forms the basis for current image and video decompression standards. In H.263, the input to the IDCT comes after the dequantization step and zig-zag positioning. An 8x8 block of input values range from -2048 to 2047 and output values in the range -256 to 255. This information is used to reconstruct the image. An original image has no prediction applied and is labeled as an I-picture (INTRA) in the standard. Its pixel values range from 0 to 255. A difference image (INTER) has prediction applied and is labeled either as a P-picture or B-picture (when bi-directional prediction occurs). Its pixel values range from - 255 to 255. The actual formula used depends on whether QP is even or odd, which is specified by the standard to prevent the accumulation of IDCT mismatch errors.

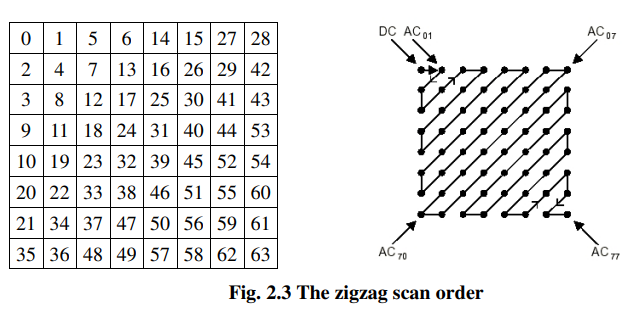
The 2D inverse discrete cosine transform is given by the following formula





**5. Zigzag scanning**

After quantization, the DC coefficient is treated separately from the 63 AC coefficients. The DC coefficient is a measure of the average value of the original 64 image samples. Because there is usually strong correlation between the DC coefficients of adjacent 8×8 blocks, the quantized DC coefficient is encoded as the difference from the DC term of the previous block. This special treatment is worthwhile, as DC coefficients frequently contain a significant fraction of the total image energy. The other 63 entries are the AC components. They are treated separately from the DC coefficients in the entropy coding process.



**6.Average Method**

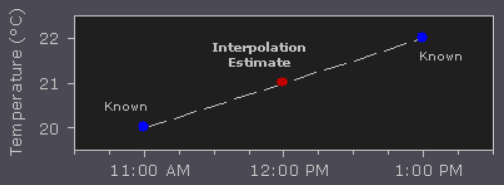
**7.Interpolation**

Image interpolation occurs in all digital photos at some stage - whether this be in bayer demosaicing or in photo enlargement. It happens anytime you resize or remap (distort) your image from one pixel grid to another. Image resizing is necessary when you need to increase or decrease the total number of pixels, whereas remapping can occur under a wider variety of scenarios: correcting for lens distortion, changing perspective, and rotating an image.

Even if the same Image resize or remap is performed, the results can vary significantly depending on the interpolation algorithm. Itis only an approximation, therefore an image will always lose some quality each time Interpolation is performed. This tutorial aims to provide a better understanding of how the results may vary-helping you to minimize any Interpolation-Induced losses in image quality.

**CONCEPT**

Interpolation works by using known data to estimate values at unknown points. For example: If you wanted to know the temperature at noon, but only measured it at 11AM and 1PM, you could estimate its value by performing a linear interpolation:



If you had an additional measurement at 11:30AM, you could see that the bulk of the temperature rise occurred before noon, and could use this additional data point to perform a quadratic interpolation:

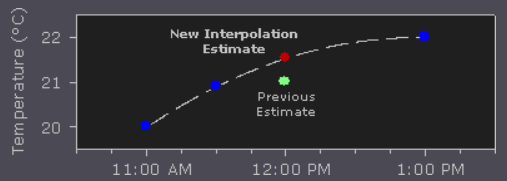
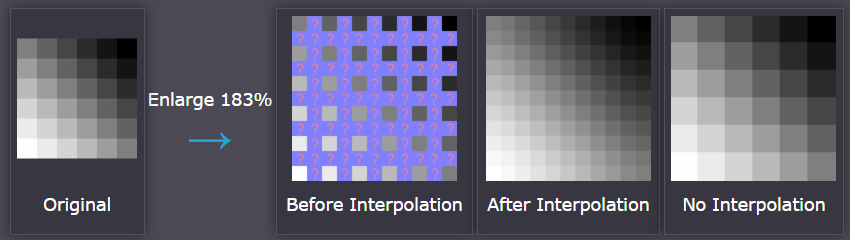


Image interpolation works in two directions, and tries to achieve a best approximation of a pixel's color and intensity based on the values at surrounding pixels. The following example illustrates how resizing/ enlargement works:

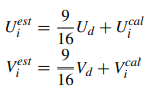


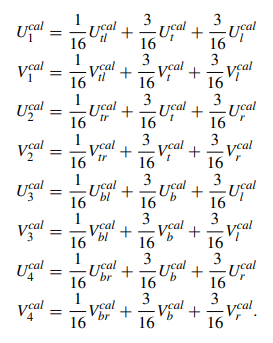
Unlike air temperature fluctuations and the ideal gradient above, pixel values can change far more abruptly from one location to the next. As with the temperature example, the more you know about the surrounding pixels, the better the interpolation will become. Therefore results quickly deteriorate the more you stretch an image, and interpolation can never add detail to your image which is not already present.

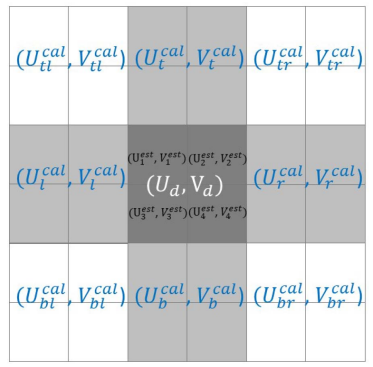
**7.1 Bilinear Interpolation**

Bilinear interpolation considers the closest 2x2 neighborhood of known pixel values surrounding the unknown pixel. It then takes a weighted avarege of these 4 pixels to arrive at its final interpolated value. This results in much smoother looking images than nearest neighbor.

To better estimate the four (U, V)-pairs of Brec,UV in terms of Ud and Vd at the server side, based on the bilinear interpolation technique, we refer to the eight neighboring downsampled (U, V)-pairs of Brec,UV. The former four downsampled (U, V)-pairs, (Ucaltl , Vcaltl ),(Ucal t , Vcal t ),(Ucaltr , Vcaltr ), and (Ucall , Vcall ), have been calculated by performing the proposed chroma downsampling method on the four reference 2×2 YUV blocks in a row-major order. The latter four downsampled (U, V)-pairs, (Ucalr , Vcalr ), (Ucalbl , Vcalbl ), (Ucalb , Vcalb ), and (Ucalbr , Vcalbr ), could be calculated by performing one existing chroma downsampling method, e.g., 4:2:0(A), on the four corresponding reference 2 × 2 chroma blocks. Based on the above eight calculated (U, V)-pairs and the unknown parameter-pair (Ud, Vd), at the server side, as derived in the Appendix, the four upsampled (U, V)-pairs in Brec,UV, i.e., (Uest i , Vest i ) for 1 ≤ i ≤ 4,(see figure) can be estimated by

with 





**7.2 Bicubic Interpolation**

Bicubic goes one step beyond bilinear by considering the closest 4x4 neighborhood of known pixels - for a total of 16 pixels. Since these are at various distances from the unknown pixel, closer pixels are given a higher weighting in the calculation. Bicubic produces noticeably sharper images than the previous two methods, and is perhaps the ideal combination of processing time and output quality.

**Bicubic interpolation** is a 2D system of using cubic splines or other polynomial technique for sharpening and enlarging digital images. It is commonly used in computer image editing software, by re touchers and editors when upscaling or resampling an image. When we interpolate an image, we are actually distorting the pixels from one grid to another.

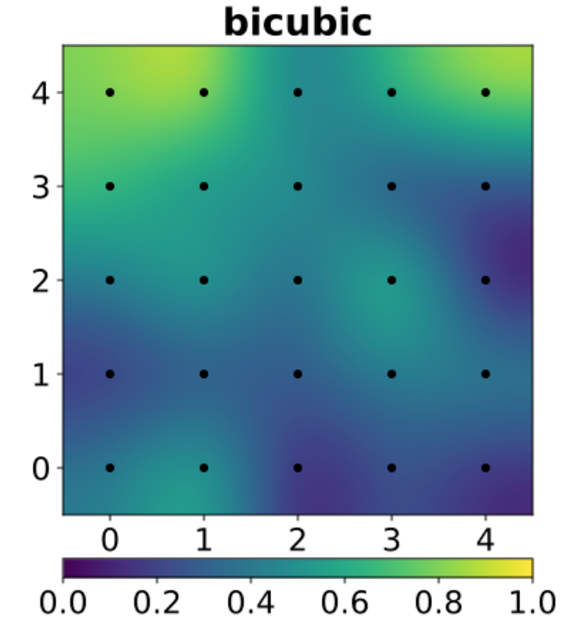
Since we are adding, not subtracting pixels, details could be lost. In order to preserve the sharpness and details, each pixel has to be approximated with its surrounding pixels in order to get the closest value. It is like duplicating a pixel to fill up the space created in the image by upscaling. Therefore, the values must be approximately similar or the same as its nearest pixel neighbor. You need to add 8 million new pixels to the image to fill up the space when upscaling. They just have to be accurate enough to recreate the details and preserve the sharpness for image clarity.

There are 2 common interpolation algorithms, adaptiveandnon-adaptive. The adaptive methods depends on what they are interpolating, whereas non-adaptive methods treat the pixels equally. Adaptive algorithms are used in many proprietary techniques in specialized professional image editing software . Non-adaptive algorithms include the following: nearest neighbor, bilinear, bicubic, spline (to name a few). In general bicubic interpolation can be accomplished using [Lagrange polynomials](https://en.wikipedia.org/wiki/Lagrange_polynomial), [cubic splines](https://en.wikipedia.org/wiki/Cubic_spline) or [cubic convolution](https://en.wikipedia.org/wiki/Bicubic_interpolation#Bicubic_convolution_algorithm) algorithms.

When we interpolate, we are estimating unknown data from known data.. When we want to upscale the image, we are approximating the new values based on that of the surrounding pixels. For example if you want to increase the image size to 24 MP at 6000 x 4000 pixels, you are adding more pixels that were not present before. That is an additional 8 MP.

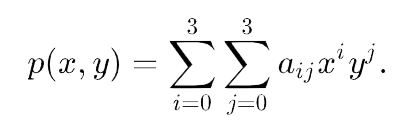
Since we are adding, not subtracting pixels, details could be lost. In order to preserve the sharpness and details, each pixel has to be approximated with its surrounding pixels in order to get the closest value. It is like duplicating a pixel to fill up the space created in the image by upscaling. Therefore, the values must be approximately similar or the same as its nearest pixel neighbor. You need to add 8 million new pixels to the image to fill up the space when upscaling. They just have to be accurate enough to recreate the details and preserve the sharpness for image clarity.

Consider the following grid.



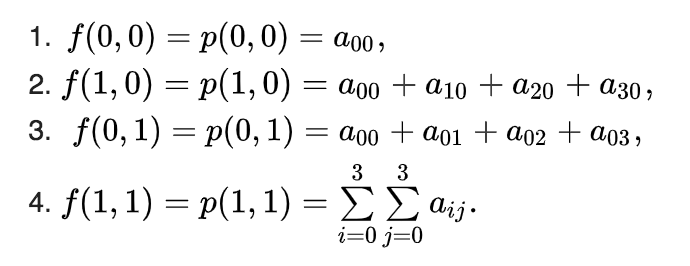
This is a (4,0) x (0,4) square with each square representing a pixel. It has a total of 25 pixels (5 x 5). The black dots represent the data being interpolated, which totals 25 dots. The colors indicate function values, so in this example we see they are not radially symmetric. This allows for smoother resampling with little image artifacts. Bicubic interpolation is often chosen over bilinear or nearest-neighbor interpolation because of this, but it takes more time to process images. If the quality is of concern, then bicubic would be the best choice as well. While bilinear interpolation processes 2x2 (4 pixels) squares, bicubic interpolation processes 4x4 (16 pixels) squares.

Let’s say you were given this function:



(0,0) (1,0) (0,1) (1,1)

These are 4 corners of the unit square. This can be represented as the following:



This requires determining the values of 16 coefficients of p(x,y). There are more steps in the process, but this is the basic formula. This is what creates the interpolated surface of the 2D image. It is all about taking the values of a point at p(x,y) on a grid and interpolating them to approximate the value of its surrounding point.

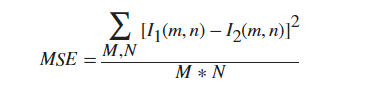
**8.Equalization**

**9. Performance analysis**

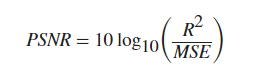
The PSNR block computes the peak signal-to-noise ratio, in decibels, between two images. This ratio is used as a quality measurement between the original and a compressed image. The higher the PSNR, the better the quality of the compressed, or reconstructed image.

The mean-square error (MSE) and the peak signal-to-noise ratio (PSNR) are used to compare image compression quality. The MSE represents the cumulative squared error between the compressed and the original image, whereas PSNR represents a measure of the peak error. The lower the value of MSE, the lower the error.

To compute the PSNR, the block first calculates the mean-squared error using the following equation:



In the previous equation, M and N are the number of rows and columns in the input images. Then the block computes the PSNR using the following equation:



In the previous equation, R is the maximum fluctuation in the input image data type. For example, if the input image has a double-precision floating-point data type, then R is 1. If it has an 8-bit unsigned integer data type, R is 255, etc.

### Computing PSNR for Color Images

Different approaches exist for computing the PSNR of a color image. Because the human eye is most sensitive to luma information, you can compute the PSNR for color images by converting the image to a color space that separates the intensity (luma) channel, such as YCbCr. The Y (luma), in YCbCr represents a weighted average of R, G, and B. G is given the most weight, again because the human eye perceives it most easily. Compute the PSNR only on the luma channel.

**10.Implementation and results**