International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 06 | June 2019 www.irjet.net p-ISSN: 2395-0072

Compress Image without Losing Image Quality using nQuant Library

Rishikesh Mhatre[1], Prof. Shubhangi Mahadik[2]

[1]*Bharati Vidyapeeth's Institute of Management and Information Technology, Navi Mumbai, India*
[2]*Bharati Vidyapeeth's Institute of Management and Information Technology, Navi Mumbai, India*

Abstract – *This paper attempts to give best image compression technique which can be lossless image compression technique there are many library used for compressing the png image but many of them are lossy compression techniques which reduce image quality after the compression is done but nQuant library which can compress png image without losing its quality it perform 0 to 100 percent compression which can user select according percentage image size will compress with maintaining a proper aspect ratio and image quality.*

Key Words: **Programmatically setting the nQuant compression, PNG, PNG Algorithm for Compression**

1. INTRODUCTION

As the demand is increasing for image processing for improving performance of any website rendering and it will take many websites to load slowly if image size is big and in government form submission they required less size image e.g 50kb is allowed, if the image is of png format image then many websites fail to compress the png image size and if compression is happen the photo pixels get damaged and make image blurry it happen due to lossy compression techniques.

This is applicable social networking like Facebook, WhatsApp, Instagram where they used this kind of algorithm to compress the image size to make page reload process faster but there some drawbacks that image is compressed looses its detailed quality to avoid this drawback we introduced one nQuant library which also implemented programmatically with these websites too.

1.1 Microsoft Nuget nQuant Technique

nQuant is a .net color quantizer producing high quality 256 color 8 bit PNG images using an algorithm optimized for the highest quality possible.

nQuant was originally developed as part of a larger effort I have been developing called RequestReduce which is an http module that automatically minifies and merges CSS as well as sprites their background images on the fly. I wanted the sprited files to be optimized and I was not satisfied with the size of the 32 bit images that the .net library was producing nor was the quantization result of like PNGQuant and PNGNQ of acceptable quality. I set out to create this quantizer and the results are images 3x smaller than their 32 bit originals with practically no perceptible quality loss.

This algorithm often produces optimized images with less quality degradation than other quantizers due to its methodology of optimizing based on clusters of colors that are closely related to one another rather than simply finding the most used colors in an image. nQuant's algorithm takes Wu's three dimensional RGB based quantization strategy and adapts it to work with transparent PNGs.

How to use this library implement below code

1. Either download nQuant from this site or add it to your Visual Studio project seamlessly via Nuget. To use Nuget nQuant Libray, Install-Package nQuant from the Package Manager Console.
2. If you do not use Nuget, add nQuant to your project and add a reference to it.
3. If you are using C#, you would call nQuant as follows.

```
var quantizer = new WuQuantizer();
using(var bitmap = new Bitmap(sourcePath))
{
  using(var quantized =
quantizer.QuantizeImage(bitmap, alphaTransparency,
alphaFader))
  {
    quantized.Save(targetPath, ImageFormat.Png);
  }
}
```

1.2 Microsoft Nuget nQuant Versions

Version	Downloads	Last updated
1.0.3	233,511	1/25/2013
0.9.8	18,540	10/9/2011
0.9.6	639	9/11/2011
0.9.5	1,459	9/6/2011

2. Comparison between compression technique

Method	Advantages	Disadvantages
Wavelet	a)High compression ratio b)state-of-the art c)low encoding complexity d)it produces no blocking artifacts	a)Coefficient quantization b)bit allocation c)less efficient
JPEG/DCT	a)current standard	a)Coefficient

	b) good quality and less amount of compression c)comparatively fast with others methods	quantization b)bit allocation
VQ	a)Simple decoder b)no coefficient quantization	a)slow codebook generation b)small bpp
Fractal	a)good mathematical encoding frame b)resolution free decoding	a)slow encoding
Genetic algorithm	a)capable of handlinn complexity and irregular solution spaces b)robust technique	a)repeated fitness function evaluation for complex problem b)not more efficient
nQuant	a) RGB based quantization strategy b)work with transparent PNGs	a) it is applicable for 32 bit image only

2.1 Image Compression Lifecycle

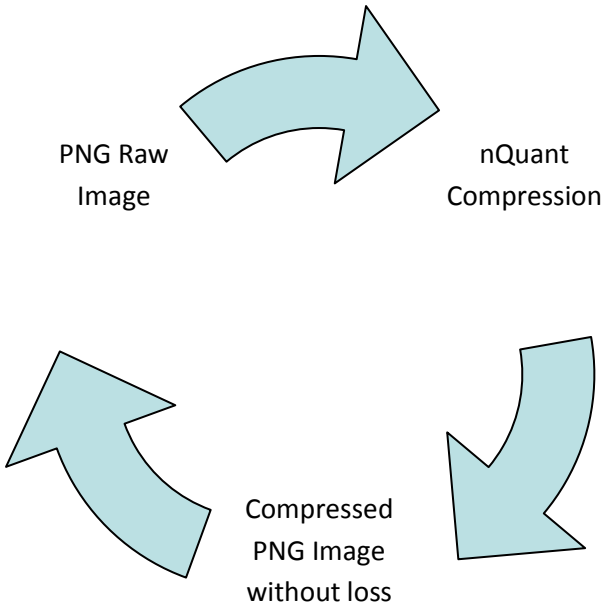

Image Compression without loosing quality

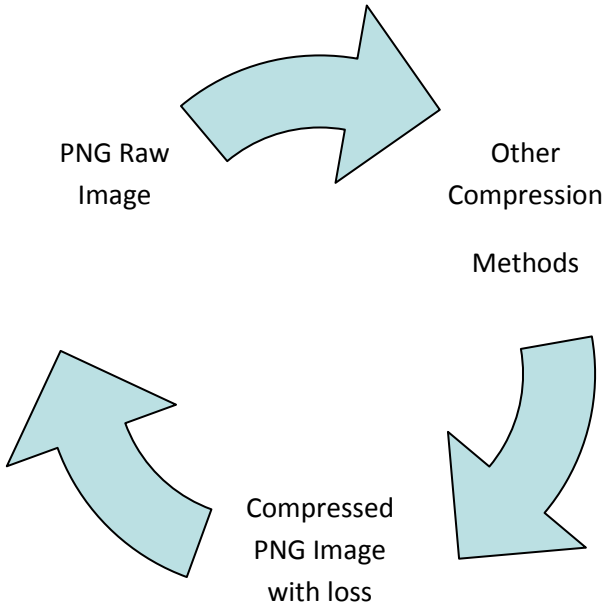

Image Compression with loosing quality

3. CONCLUSIONS

We have summarized the characteristics of four up- to-date image coding algorithms based on Wavelet, JPEG/DCT, VQ, Fractal, Genetics approaches, nQuant. Any of the five approaches is satisfactory when the 0.5 bits per pixel is requested. However, for a very low bit rate, for example 0.25 bpp or lower, the embedded zerotree wavelet (EZW) approach is more applicable and superior to other approaches. For practical applications, we conclude that

(1) Wavelet transform based compression algorithms are strongly recommended,

(2) JPEG/ DCT based approach might use an adaptive quantization table,

(3) VQ technique is not good for a low bit rate compression and it is slow processed,

(4) Fractal approach should utilize its resolution-free decoding property for a low bit compression and slow in processing,

(5) nQuant compression technique created for compressing 32 bit png images it is RGB based quantization strategy and work with transparent PNGs

REFERENCES

[1] https://compresspng.com/.
[2] https://www.iloveimg.com/compress-image/compress-png.

[3] Fundamental of Computer Graphics Book by Erik Reinhard, Kelvin Sung, Michael Ashikhmin, Michael Gleicher, Peter Shirley, Peter Willemsen, Stephen R. Marschner, and William B. Thompson.

[4] https://ezgif.com/optipng.

[5] https://www.websiteplanet.com/webtools/imagecompressor/

[6] https://archive.codeplex.com/?p=nquant