**“Justification of the use of CCSDS 122.0-B-2 Compression Standard in Space Missions over other Compression Methods through the assessment of Image Quality”**

CS39440 Major Project

Author: Charlie Curtis ([chc73@aber.ac.uk](mailto:chc73@aber.ac.uk))

Supervisor:

3rd March 2022

Version 1.0 (Draft)

This report is submitted as partial fulfilment of a BSc degree in

Computer Science (G400)

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Wales, UK

Declaration of originality

I confirm that:

* This submission is my own work, except where clearly indicated.
* I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
* I have read the regulations on Unacceptable Academic Practice from the University’s Academic Registry (AR) and the relevant sections of the current Student Handbook of the Department of Computer Science.
* In submitting this work, I understand and agree to abide by the University’s regulations governing these issues.

Name …………………………………………

Date ……………………………………………

Consent to share this work

By including my name below, I hereby agree to this project's report and technical work being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name …………………………………………

Date ……………………………………………

Acknowledgements

I am grateful to…

I’d like to thank…

Abstract

Include an abstract for your project. This should be approximately 300 words.

The abstract is an overview of the work you have done. Highlight the purpose of the work and the key outcomes of the work.

Contents

Table of Contents

[2 Background, Analysis & Process 1](#_Toc101710894)

[2.1 Background 1](#_Toc101710895)

[2.2 Analysis 2](#_Toc101710896)

[2.3 Process 2](#_Toc101710897)

[3 Use of Algorithms 3](#_Toc101710898)

[3.1 Bit Plane Encoder (BPE) 3](#_Toc101710899)

[3.2 JPEG 2000 (J2K) 4](#_Toc101710900)

[3.3 Mean Squared Error (MSE) 4](#_Toc101710901)

[3.4 Peak Signal to Noise Ratio (PSNR) 5](#_Toc101710902)

[3.5 Structural Similarity Index (SSIM) 5](#_Toc101710903)

[4 Development of Script 5](#_Toc101710904)

[5 Method 8](#_Toc101710905)

[5.1 Sample Preparation 8](#_Toc101710906)

[5.2 Processing 9](#_Toc101710907)

[6 References 10](#_Toc101710908)

# **Background, Analysis & Process**

## **Background**

The European Space Agency (ESA) will soon launch the second part of their ExoMars mission. Rosalind Franklin is the rover that shall travel to the surface of Mars with the goal of attempting to establish if there is or ever has been life on Mars.

One of the rovers many instruments is PanCam, a camera system tasked with capturing how the planet looks and preparing these images to be sent back to Earth. Part of this preparation is the compression of the images to reduce their storage space and allow for a more streamlined and hopefully less error prone transfer back to Earth. The ESA has chosen to use the CCSDS 122.0-B-2 compression standard, the purpose of this report is to establish a justification for this, and reasons why other compression methods such as JPEG 2000 (J2K) are not used. This is done through the encoding of sample images taken from the Aberystwyth University PanCam Emulator (AUPE), through both compression algorithms, after which the results are compared against each other.

A sensible approach to begin seemed to be to conduct some preliminary research into image compression. This included not only image compression algorithms used in a variety of disciplines including medicine, graphic design, and military applications, but also, the theory behind image compression, i.e., lossy versus lossless compression, techniques used in image compression, including - but not limited to; quantization, discrete cosine transform (DCT), fractal compression – and performance comparisons of some compression algorithms.

The goal of this was to try and establish what sort of numerical operations these algorithms were performing, and through this, trying to build an understanding of how the quality assessment algorithms - that would be used as part of the overall analysis of the compression algorithms - might begin to calculate numerical quality in a general sense. This was very useful for understanding metrics such as Mean Square Error (MSE), Root Mean Square Error (RMSE), and Peak Signal to Noise Ratio (PSNR).

During this preliminary research, a grouping system of image quality assessment (IQA) methods was identified. Methods are grouped into Full Reference (FR), Reduced Reference (RR), and No Reference (NR) IQA methods. They refer to if a ‘perfect’ quality - original image is used as a reference as part of the algorithm to compare to the compressed version. For this project, FR seemed the most appropriate since there *are* original images to be accessed and so if the more comprehensive algorithms can be exercised as part of testing of PanCam and as part of this project, the greater assurance we can have as to our understanding of the level of compression and quality of the images being sent back by Rosalind Franklin.

One motivation for this project was having the opportunity to produce something that could potentially have real world implications towards the operational effectiveness of the European Space Agencies (ESA) Rosalind Franklin rover. Moreover, developing a tailored script (written in Python) that’s sole purpose is to facilitate the gathering, processing, and analysis of the numerical aspects of IQA algorithms.

## **Analysis**

The projects focus is on IQA algorithms. To that end, regardless of the direction of the project, these algorithms would have to play a significant role in any conclusions or findings. This led to the belief that a comparative study was the best approach, since the result that objective IQA algorithms provide are numerical values, for the quality of images. This leaves great scope for statistical analysis.

With this in mind, the logical progression was to compare the already selected compression algorithm with another - that was not selected to be a part of PanCam - to try and establish the reasons that the aforementioned algorithm was chosen over the other. This led to the construction of the overall research question for this project:

**Given the quality of compressed images, what reasons are there for the use of the CCSDS 122.0-B-2 compression standard on PanCam, and why are other image compression algorithms not used in the context of interplanetary communication?**

This in turn lead to more specific questions regarding the technicalities of the project and its process, such as; Which IQA algorithms would be best suited to a project of this nature? What sort of analysis and statistical metrics should be applied to any collected data? What would be the best method to approach this question?

The chosen research question also has the advantage of leaving the door open for further research or variations of this project, for instance, delving further into the question of which IQA algorithms are most suitable as metrics for assessing image quality in the context of space mission or conducting the same or similar research with other possibly more advanced image compression algorithms other than J2K.

## **Process**

This project is a comparison using objective measures. Mathematical processes conducted by the computer and then interpreted by the user. Due to this fact, the data collected was always going to Quantitative data. A positivism-based [1] research project, collecting primary data for processing and to develop conclusions from.

An inductive process, aiming to answer the overall research question first proposed at the beginning of the project. [2] Given that the compression algorithm has already been decided upon for the rover, and that trying to produce a conclusive set of results would be beyond the scope of this project, the project took an approach indicative of an exploratory project, trying to establish the reasons for the selection of compression algorithm.

These are very blanket terms that best describe a process that was dynamic and constantly changing based on complications encountered at various stages and the capricious availability of pre-existing libraries and/or implementations of IQA algorithms.

Additionally, regarding the production of a technical submission. The project was research focused and so, a technical submission was not going to be of the same nature or standard of complexity as the likes of a project involving the development of some bespoke software, because of this, following an Agile Development approach was challenging. Nonetheless, a SCRUM approach was loosely adopted and maintained over the lifecycle of the project, with a ‘sprint’ being a weeklong process, and the planning for said sprint being conducted at the end of the previous sprint.

Finally, in the forthcoming sections the report will address the following subjects to aid in building an understanding of the process, and algorithms involved.

In the next section, a discussion of the algorithms used for both the compression of the images, as well as those used for the assessment of image quality, to establish how they work, and therefore why these particular algorithms have been chosen for the process. Upon conclusion of this discussion a description of the method that was adopted, not only for the collection of the data the project relies upon, but also a description of the process for the analysis of this data that is necessary for the development of a based, well-founded conclusion. This will be followed by the presentation of the results and then a discussion of these results, what they mean, and what can be learnt from them. The report is to be finalised, with a critical analysis of the process and administration surrounding the project as a whole - from beginning - to end.

# **Use of Algorithms**

This section will explore the various algorithms that are used as part of this project including algorithms for both the compression of sample images; namely the CCSDS standard and JPEG 2000, as well as those used in the image quality assessment process – Mean Squared Error, Peak Signal to Noise Ratio, and the Structural Similarity Index. The compression algorithms are as follows:

## **Bit Plane Encoder (BPE)**

BPE is a command line implementation of the CCSDS 122.0-B-1 algorithm. The algorithm is based on wavelet transform and bit plane scanning. [3]

Bit plane encoding operations involve the conversion of image pixel values into their binary form, and then slicing. The process by which the binary values are separated by the most a least significant bits and then different bit planes constructed. Subsequently, the original image is reconstructed using the greatest value planes. This creates an image that has been lossy compressed using a significantly reduced amount of data as the refinement bits in the lower value planes are removed. [4] One of the main factors for this is use of binary representation of pixel values, because with current hardware standards this is particularly convenient. [5]

Wavelet transform is an adaptation of the Fourier Transform, that considers a wavelet of frequency, in this case, pixel intensity. The function operates in a localised region of the image known as a ‘window’ and based on the pixel intensity and window function (function used to select the area of the image for encoding) essentially quantizes the pixel values. [6][7]

On PanCam – this CCSDS algorithm is used by default at a ratio of 1, which achieves an x8 compression, and for quick look images at a ratio of 0.1; x80 compression.

## **JPEG 2000 (J2K)**

J2K is a compound compression algorithm curated from a series of image compression procedures and similarly, is based on wavelet transforms. [8] J2K is an accessible, scalable, and popular compression algorithm which is why it was selected as part of this project. Moreover, when trying to discern reasons for CCSDS 122.0-B-1, comparing it to a relatively more standard compression algorithm seemed like a good step, as it helps to develop a picture of the magnitude of difference between an inter-disciplinary ‘basic’ method, and a bespoke solution for the demanding environment that is space and missions in space.

For this project – OpenJPEG was employed. [9] This is again, a command line implementation, alike to BPE, with parameters to change the compression ratio up to x9000 compression.

Given the parameters for compression outlined by the team behind PanCam, for this project. OpenJPEG was used with compression rates of x8 and x80, this is to bring it in line with what is to be used on Rosalind Franklin, and so produce an analysis and discussion/conclusions that are valid.

## **Mean Squared Error (MSE)**

Mean Squared Error is the first of three Image Quality Assessment metrics that were used in this project. MSE measures the difference between a predicted set of data, and an actual, measured set. It is the second moment of error [10]. In this scenario, the predicted is the original ‘perfect’ quality image, and the actual is the compressed version, this makes MSE a full reference objective method. MSE is defined by:

Where is the original image, and

the distorted image.

MSE has a clear physical meaning and can be applied in multiple ways to linear algebra, this makes it one the most used metrics in image quality assessment [11].

Because it is a measure of error, values closer to 0 mean a greater quality, and thus, less information loss during compression.

## **Peak Signal to Noise Ratio (PSNR)**

Peak Signal to Noise Ratio is the ratio between the maximum possible signal value (pixel value) and the power of distorting noise where the signal is the perfect reference image, and the noise the error yielded by compression [10].

PSNR can be defined by:

Where is the maximum possible signal value; In the case of an 8-bit (unsigned) image this would be . The sample images however used in this project are all 16-bit, and as such .

## **Structural Similarity Index (SSIM)**

SSIM is a perception-based method [10]. The only of the three IQA algorithms to be based on the Human Visual System (HVS). Algorithms that replicate the HVS consider different aspects of the image rather than the image as a whole, aspects such as brightness, contrast, and texture [12].

SSIM specifically – considers luminance, contrast, and structure [10][11]. SSIM can be defined as [13]:

Where:

is the average of

is the average of

is the variance of

is the variance of

Where:

is pixel range (for 16-bit this is 65535)

= 0.01 and = 0.03 (these are default values)

This overall equation may be further broken down into component formulas for luminance, contrast, and structure. SSIM operates by assessing an area of each.

# **Development of Script**

The script that was worked on and developed as part of this project was originally started in the interest of ‘insurance.’ A fall back in case libraries or command line tools for various functions and requirements of the overall process weren’t available. Of course, most of these were available in the form of BPE and OpenJPEG as previously mentioned, but also additional tools that will be touched upon in later sections such as ImageMagick [14]. Initially the intention of the script was to run the objective IQA algorithms but has since been adapted to also generate absolute error maps, convert images between 8 and 16 bits, and perform the statistical analysis. This is achieved through the use of various libraries along with some basic arithmetic and data type manipulation.

The script can be found at the technical submission point for this project as well as on this projects GitHub repository.

The following section will discuss the use of the libraries in question, as well as some of the major functions in the script that were used extensively throughout the process.

To begin, a list of the libraries:

* Python Image Library – PILLOW (PIL) [15]
* Scikit-Learn – sklearn [16]
* SciKit-Image – skimage [17]
* Numpy [18]
* SciPy – specifically stats module [19]
* MatPlotLib – pyplot module [20]
* Pandas [21]

PIL was necessary to deal with the loading and saving of the images in a streamlined fashion. It also allowed for identification of the ‘mode’ of the image, (whether or not the image is 8 or 16bits.) Sklearn contained a prewritten MSE calculation and skimage an SSIM calculation, this meant that these two metrics didn’t need to be hardcoded.

Numpy was essential for collecting image pixel values, storing, and then handing these over to the relevant functions that take an array-like object. Moreover, arithmetic was made significantly easier given the nature of NumPy arrays to iterate through themselves, without the need for an additional loop, this benefits readability massively.

Finally, SciPy, MatPlotLib and Pandas were all required for the calculation of the resultant statistics that would be used to inform the analysis section of this report. Any generated figures were produced within MatPlotLib, and statistical calculations such as Pearson’s correlation coefficient and two-sample t-tests were all done from SciPy. Pandas was required to facilitate these calculations.

In relation to the development of functions. For the majority of the script’s lifecycle, it was in an experimental/developmental stage, meaning some functions were added and then later removed, some incorporated into other functions, etc. This inevitably means that some functions – particularly at the start of the process – were and have been used more than others. Moreover, some functions were simply written to aid in the debugging process and had no bearing on the processing of any data, so were removed from the final version of the script.

There are two main functions that played pivotal roles in the process (discounting those functions that compute the various IQA metrics) those are; *collect\_greyscale\_pixels()* and *load\_raw\_image()*. In the following passage, the necessity and process of developing these essential functions will be discussed, beginning with *collect\_greyscale\_pixels()*.

The purpose of this function is to collect the pixel values from a grey scale image. If the passed image is not grey scale, it will be transformed as such. This is because the images that get passed to the compression algorithms must be in grey form to be run through the algorithm since they can only process single channel images, as a result any RGBA/RGB images passed, are converted.

This conversion is achieved by collecting the images ‘mode’ (i.e., the colour nature and pixel density) of each image being included in the process. The modes that are mentioned as part of the implementation are ‘L’, ‘I’, and ‘I;16’.

Referring to the PIL documentation [22] it can be known that:

* ‘I’ is 32-bit signed integer pixels
* ‘L’ is 8-bit black and white images
* ‘I;16’ is 16-bit unsigned pixel integers

In the event the image is 8 bit and already black and white (mode ‘L’.) Then all that happens is the pixel values are read into a NumPy array, and the shape of the array adjusted to create a 2-dimensional array, that equates to the resolution of the image that was originally passed to the function.

When the image mode is returned to be ‘I’ this usually means one of the 16-bit grey scale sample images has been passed, and so the image is converted to ‘I;16’ before being read into the NumPy array and the arrays shape changed as aforementioned. This conversion means that the dynamic range of a given pixel is reduced and moreover, makes conversion to 8-bit that much easier, which when calculating absolute error maps for interesting cases, is welcome. Moreover, it aids in reproducibility.

The main reason for the importance of this function, is that the functions to calculate MSE, PSNR, and SSIM all require an array-like object to be passed as a parameter. This means that the image cannot simply be loaded in PIL and then passed into the function, the individual pixel values must be collected into an array. Only then can the various IQA metrics be computed. Moreover, these metrics require the images to be of the same pixel-depth, hence the need for conversion, pre-collection.

The final else statement within the function is a legacy statement from the debugging process, however, it was decided to leave it in, in the event that if the script is run with any other images, the basic pixel collection can be applied and stope any errors being thrown.

Finally, the *load\_raw\_image()* function. Because BPE requires to be provided with raw image data, this meant the original sample images had to be converted into a ‘.dat’ format. The decode delegate for this format in ImageMagick, is not included in the basic install, as a result, this function was required, in order to reproduce a PNG image from the compressed DAT files. Utilising the NumPy *fromfile()* [23] function to generate an array from the raw data file.

Initially this is a 1-dimensional array, however, reshaping the array to be the square root of the array size, creates the final 2-dimensional array that has the same dimensions as the original image.

Without the function, the compressed DAT files coming out of BPE, would not be able to be converted back into a PNG image, this function was applied to every image that was tested, and without it the final IQA metrics and statistical analysis could not be conducted as there would be no compressed images to utilise.

In all, the script went from being an answer to a potential worst-case scenario, to an integral part of the project and method, going through many different stages with varying levels of capability, before reaching its final version.

# **Method**

The method can be divided into three distinct sections, listed below in the order within which they were conducted:

1. Sample preparation
2. Processing
3. Analysis

Each section will be addressed in turn whilst also making clear how they fit into the overall method.

## **Sample Preparation**

The sample images were provided by the Aberystwyth University PanCam Emulator (AUPE). PanCam, and therefore AUPE, have three cameras that this project is concerned with; two Wide Angle Cameras (WAC) and one High Resolution Camera (HRC.) Because of the nature of the camera, the HRC takes colour images. These images can have various coloured filters applied that capture different things in the images taken. These images require a separate CCSDS compression algorithm in order to be processed, because of this, these multi-channel images were excluded from the project and only the WAC images considered. Most of the WAC images however were processed as they take the form of 16-bit grey scale images. This made them very easy to compute.

The sample images come as PNG images. BPE can only process images in a raw format, as such the images had to be converted into DAT format for the purposes of compression through BPE. With OpenJPEG however, this was not necessary.

This conversion was simply done through ImageMagick, this is a command line tool for processing, editing, and formatting images. This tool was used, and proved particularly useful, in this conversion process. See the command below:

magick convert P00\_T00\_L01\_00.png gray: P00\_T00\_L01\_00.dat

This example of a magick command is representative of those that were used in the project. In the scenario of the example an image entitled “P00\_T00\_L01\_00.png” is converted to a ‘.dat’ format.

This command was executed semi autonomously utilising a bash command, it was executed in each directory containing sample images, in order to convert all into binary format. Throughout the overall method, a naming convention was adopted that made interpreting the results of executing these bash scripts much more intuitive and fluid.

Images simply converted to a different format merely had the file extension changed – whether that be .png, .dat, or .j2k. For the images parsed to BPE, this was not enough however, as BPE simply compresses the file and returns it. To overcome this, when compressed, the images had ‘\_compressed\_’ appended, followed by the ratio of compression for instance - ‘1’, all of this was appended prior to the file extension. Once decoded, ‘\_compressed’ was replaced by ‘\_decompressed\_’ again with the compression ratio, and again, prior to the file extension. This made managing all the files significantly easier and developing a spreadsheet of results similarly as perceptive.

Once the original sample PNG images were converted into their respective binary versions. They could then be passed to the BPE for encoding.

With respect to the J2K compression, this conversion step was not required. The original images could simply be used as they were and OpenJPEG would handle any conversion, and subsequently compression of the images.

## **Processing**

Image processing was the longest stage in the process. This is largely due to each image having to be passed through each algorithm twice. Once, in order to achieve an x8 compression, and twice for an x80 compression rate. The reason for this is that these two values are the extremes according to the PanCam documentation. These compressions are used as a default, and for ‘quick look’ images, respectively.

When encoding with BPE, the trend of being conducted semi autonomously through bash commands was adhered to. Given the nature of the files that were being run through BPE (raw binary file) the tool requires that some parameters be specified when the compression command is run. In this instance this information was the width and height of the image, pixel depth, and the compression ratio (for this project limited to either 1 or 0.1.) In this scenario all these values are the same from image to image which made processing time significantly less of a hindrance.

In addition, for the encoding and decoding of images using BPE, the bash ‘time’ key word was used to effect, in order to report on the real time taken to compress and decompress images. This would prove valuable in the overall analysis of the process and results.

The following are the compression (for a ratio of 1 and 0.1) and decompression commands used within BPE respectively, to achieve the generation of compressed version (at both levels) of the original images:

time ./bpe -e P00\_T00\_L01\_00.dat -o P00\_T00\_L01\_00\_compressed\_1.dat -w 1024 -h 1024 -b 16 -r 1

time ./bpe -e P00\_T00\_L01\_00.dat -o P00\_T00\_L01\_00\_compressed\_1.dat -w 1024 -h 1024 -b 16 -r 0.1

time ./bpe -d P00\_T00\_L01\_00\_compressed\_1.dat -o p00\_T00\_L01\_00\_decompressed\_1.dat

Note that during decompression the extra parameters are not required as these are collected from the compressed data stream.

Once this process is complete, the filenames finishing with ‘decompressed\_X.dat’ where X is either compression ratio, are now ready to be read, saved as PNG images, and then data collected from them such as file size and compression/decompression time, as well as the scores from the various IQA metrics.

The process with J2K was significantly easier. Since OpenJPEG can handle PNG image files, this means that no manual conversion is required, the image can be passed to the tool as it is and processed from there. For compression rates of x8 and x80 respectively:

opj\_compress -i P00\_T00\_L01\_00.png -o P00\_T00\_L01\_00.j2k -r 8

opj\_compress -i P00\_T00\_L01\_00.png -o P00\_T00\_L01\_00.j2k -r 80

A much more streamlined process, generating compressed images in a significantly shorter time. This along with the semi autonomy provided by bash commands made this process by far the most accessible.

At this stage, data is ready to be collected on the J2K compression process, however, as previously discussed there is still a step remaining in order to turn the raw data files generated from the BPE, into useable PNG images from which data on the BPE process can be collected.

This final step involves utilising the developed script, that was done alongside the project, to collect the raw data and generate PNG images. The steps involved in this are explained in section 4, however, once this process was complete data, data from the BPE compression process too, was able to be collated.

The data collection was as may be expected from such a task; the times outputted from the compression tasks were recorded, along with the original and compressed file sizes.

The last step was to run each image through the MSE, PSNR, and SSIM functions and record these results. Obviously, for both compression levels, and for both algorithms. This lead to the construction of tables x and x, in section 5.3.

## **Analysis**

# **References**

1. Clark, A.M. (1998), The qualitative-quantitative debate: moving from positivism and confrontation to post-positivism and reconciliation. Journal of Advanced Nursing, 27: 1242-1249. <https://doi.org/10.1046/j.1365-2648.1998.00651.x>
2. Business Research Methodologies <https://research-methodology.net/research-methods/> - accessed 20/04/2022
3. Bit plane encoder

<http://hyperspectral.unl.edu/> - accessed 22/04/2022

1. Bit-Plane Slicing

<https://theailearner.com/2019/01/25/bit-plane-slicing/> - accessed 22/04/2022

1. F. Auli-Llinas and M. W. Marcellin, "Scanning Order Strategies for Bit plane Image Coding," in IEEE Transactions on Image Processing, vol. 21, no. 4, pp. 1920-1933, April 2012,doi:10.1109/TIP.2011.2176953.
2. Description of the process of wavelet transforms - <https://www.youtube.com/watch?v=kuuUaqAjeoA> – accessed 22/04/202
3. A Tutorial of the Wavelet Transform – Chun-Lin, Liu; 2010 <https://www.researchgate.net/profile/Vladimir-Kulchitsky/post/Is-there-any-difference-in-the-tiling-of-the-time-frequency-plane-by-the-orthogonal-and-biorthogonal-wavelet-basis-functions/attachment/59d629c179197b807798844a/AS%3A337039875690496%401457367976408/download/WaveletTutorial.pdf> - accessed 22/04/2022
4. JPEG2000 compression - <https://jpeg.org/jpeg2000/> - accessed 22/04/2022
5. OpenJPEG GitHub repository- [https://github.com/uclouvain/openjpeg/tree/v2.4.0](https://github.com/uclouvain/openjpeg/tree/v2.4.0%20-%20accessed%2022/04/2022) - accessed 22/04/2022
6. Umme Sara, Morium Akter, Mohammed Shorif Uddin, “Image Quality Assessment through FSIM, SSIM, MSE and PSNR—A Comparative Study” (2019) **DOI:** [10.4236/jcc.2019.73002](https://doi.org/10.4236/jcc.2019.73002)
7. Zhou Wang, Alan C. Bovik, “Modern Image Quality Assessment Synthesis Lectures on Image, Video, and Multimedia Processing” - (2006), Morgan & Claypool <https://doi.org/10.2200/S00010ED1V01Y200508IVM003>
8. Yusra A. Y. Al-Najjar, Dr. Der Chen Soong “Comparison of Image Quality Assessment: PSNR, HVS, SSIM, UIQI” International Journal of Scientific & Engineering Research, Volume 3, Issue 8, August-2012 1  
   ISSN 2229-5518 https://www.ijser.org/researchpaper/Comparison-of-Image-Quality-Assessment-PSNR-HVS-SSIM-UIQI.pdf
9. Wang, Z.; Simoncelli, E.P.; Bovik, A.C. (2003-11-01). Multiscale structural similarity for image quality assessment. Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers, 2004. Vol. 2. pp. 1398–1402 Vol.2. [doi](https://en.wikipedia.org/wiki/Doi_(identifier)):[10.1109/ACSSC.2003.1292216](https://doi.org/10.1109%2FACSSC.2003.1292216). [ISBN](https://en.wikipedia.org/wiki/ISBN_(identifier)) [978-0-7803-8104-9](https://en.wikipedia.org/wiki/Special:BookSources/978-0-7803-8104-9).
10. ImageMagick - image modification tool

<https://imagemagick.org/script/index.php> - accessed 23/04/2022

1. Python Image Library (PILLOW)

<https://python-pillow.org/> - accessed 23/04/2022

1. SciKit-Learn <https://datagy.io/python-scikit-learn-introduction/> -accessed 23/04/2022
2. Scikit-image <https://scikit-image.org/> - accessed 23/04/2022
3. Numpy <https://numpy.org/> - accessed 23/04/2022
4. SciPy <https://scipy.org/> - accessed 23/04/2022
5. MatPlotLib <https://matplotlib.org/> - accessed 23/04/2022
6. Pandas <https://pandas.pydata.org/> - accessed 23/04/2022
7. PILLOW documentation on image modes - <https://pillow.readthedocs.io/en/stable/handbook/concepts.html#concept-modes>
8. NumPy.fromfile() function - <https://numpy.org/doc/stable/reference/generated/numpy.fromfile.html> - accessed 23/04/2022