IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Tupper's Self Referential Formula For Image Compression

Tushar Pardhe

Engineer
Information Technology
Netaji Subhas University of Technology, New Delhi, India

Abstract: Tupper's self-referential formula is a mathematical equation that plots itself in a (x, y) plane. Jeff Tupper's formula first appeared in his "SIGGRAPH" paper in which he discussed about different graph plotting algorithms and other graph equations. However, the specialty of this formula is that, it can be applied to any image of size 17x106. This research paper uses his formula to transfer image data from sender to receiver in a lossless manner.

Index Terms **- Image processing, Tupper's formula, self-referential formula, bits, images, optimization, image storage, file size, image size.**

Introduction

For the size of a computer file is a measure of how much data it contains or how much storage it takes up. Typically, file size is stated in byte-based units of measurement. Generally, images are stored as a sequence of bits. The sizes of the images can varyfrom kilobytes to megabytes. [3] When a file is written to a file system, which is the case in most modern devices, it may consume slightly more disk space than the file requires. This is because the file system rounds the size up to include any unused space left over in the last disk sector used by the file. (a sector is the smallest amount of space addressable by the file system. The size of a disk sector ranges from several hundred to several thousand bytes.) the unused space is called slack space or internal fragmentation. [4] although smaller sector sizes allow for denser use of disk space, they decrease the operational efficiency of the file system. Image optimization is a process of manipulating image data for faster load times, while retaining the overall quality of the image. There are many algorithms that optimize loading times; but most of them use various compression techniques which deteriorates overall quality of the image. This research paper proposes an algorithm that stores image data in a compressed form without loss of information. [5] Tupper's formula is special because it builds itself at a certain distance above the y-axis. This works because the equation tells you which (x, y) coordinates are coloured and which ones are not. So, between certain values of k and k + 17 on the y-axis and between 0 and 106 on the x-axis is the equation graphs itself. But the key is the value k. If we plot the equation and look at it up to between the height of k and k + 17 on the y-axis, we get a graph of the equation itself. Let us understand the equation:

$$\frac{1}{2} < \left\lfloor \operatorname{mod}\left(\left\lfloor \frac{y}{17} \right\rfloor 2^{-17\lfloor x \rfloor - \operatorname{mod}(\lfloor y \rfloor, 17)}, 2 \right) \right\rfloor$$

Fig 1.1: TUPPER'S SELF-REFERENTIAL EQUATION

In PLAIN TEXT:
1/2 < FLOOR(MOD(FLOOR(Y/17)*2^(-17*FLOOR(X)-MOD(FLOOR(Y),17)),2))
If,
K=4858450636189713423582095962494202044581400587983244549483093085061934704708809928450644769865524364849997247024915119110411605739177407856919754326571855442057210445735883681829823754139634338225199452191651284348332905131193199953502413758765239264874613394906870130562295813219481113685339535565290850023875092856892694555974281546386510730049106723058933586052544096664351265349363643957125565695936815184334857605266940161251266951421550539554519153785457525756590740540157929001765967965480064427829131488548259914721248506352686630476300.

THEN THE RESULT WILL BE AS FIG 1.2,

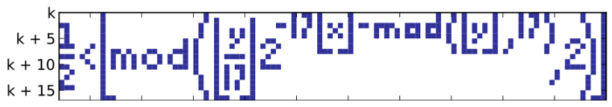

FIG 1.2: GRAPH PLOT FOR TUPPER'S FORMULA

However, the application of this equation is ubiquitous as it can represent any equation or image based on k value, and for each of them k value always remains the same. For instance, if

K=2352035939949658122140829649197960929306974813625028263292934781954073595495544614140648457342461564 887325223455620804204796011434955111022376601635853210476633318991990462192687999109308209472315419713652 238185967518731354596984676698288025582563654632501009155760415054499960

We get Euler's equation:

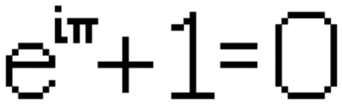

FIG 1.3: EULER'S EQUATION [6]

But why is this formula so famous? Because, we can plot anything that fits 17x106 2d matrix including images. Taking this into account now we share image data without storing the whole string of bits rather we can use their unique "k" value and recreate them without loss of data (lossless compression [7 - 8]).

I. Methodology And Algorithm

In this section we will discuss the proposed algorithm and the stepwise process of how Tupper's formula can be applied on images of any dimension.

Steps are as follows:

1. *Adding Padding of bits*

Consider A to be an image with dimensions 1024x916. These numbers are referred as resolution. The term "resolution" is often thought to be equivalent to the number of pixels in a digital image, but international digital camera standards indicate that it should instead be called "total number of pixels" and "number of recorded pixels" in reference to an image sensor. Since, the size of the image is big, Tupper's formula won't be able to plot this image on the plane. However, if we split the image into various sub images of 17x106 dimensions, we can get their "k" values.

We divided the image into various sub images. But what if after division some images are not 17x106? To solve this problem, we need to add padding. Let's consider the above image, the height and the width of the image wasn't exactly divisible by the divisors so we need to round them off to the highest possible value.
So, 1024 / 17 gives us 61 and 916 / 106 gives us 9. (rounded off to highest value). So, now we need to multiply 61 x 17 and 9 x 106 to get the final values and subtract the answer from the original dimensions to get the number of 0s that we need to add as padding.
17x61 = 1037
1037 – 1024 = 13
Which means we need to add 13 rows of padding. Similarly, we need to add padding for columns,
106x9 = 954
954 – 916 = 38
i.e. we need to add 38 columns. Padding of zeros won't affect the original image as they denote empty value.
Splitting into parts

So, now we have an image with dimensions that are divisible by 17 and 106 divide the number of rows i.e. After equating the values into division formula, we get 61 number of rows and 9 number of columns which means we can divide the image into 549 sub parts by iterating over the original image. After doing this step we can move to the next step of generating K values.

2. *Generating K values*

Now, we have the individual images that can be solved using Tupper's formula mentioned in Fig 1.1. We equate individual (x, y) values of the images to get the k value for every sub image.

Although, now we have individual k values, but as we need to arrange them in a particular order to decompress the image on the receiver end we require their positions as well.
We know that the original image was 1024x916. Dividing 1024 by 17 we will get **61** sub images; and 916 by 106 will give us **9** sub images. (Rounding off to the bigger number) So, in total we will have 61 multiplied by 9 i.e. 549 images.
So basically if we travel left to right we have to arrange 9 images in the first row than, we can proceed filling up the second row and so on until we reach the 61st row. This way we will be able to replicate the whole image without any loss of data.

Denote each k value of sub image as an alphabet: a, b, c, d …till n; where n is the number of columns. We can say that our final solution array will be:
F = [[a, b, c, d, e, f, g, h, i], [j, k, …], […till n], …]
OD (Original Dimensions) = 1024x916

In this the subarray defines individual row of the final image row 1 = subarray 1 and so on, till we reach the last row. We pass these two values to the receiver and move to the next step in which we decrypt it on the receiver end.

3. Decrypting the image

Till step C, we have stored the individual k values of the images but we also need to store the original image dimensions. As it will help us remove the padding that we added to the original image.

1. On the receiver end first we equate the k value into the Tupper's equation to get back the original sub images in their bit form (string of 0s and 1s). Repeating this step for every k value we will get all the original sub images that were divided in the Step A.
2. In this step we will stitch these sub images to get the original image. We have the original image dimensions i.e. 1024x916 so we again divide 1024 by 17 and 916 by 106 and get the rounded off values. (round of to the bigger value i.e. 7.1 to 8).
3. Taking the quotient after division we will know that there are 61 rows of sub images and 9 columns of sub images.
4. So now we pick our sub images one by one and place left to right until we reach the 9th image. After that we move to the next row and repeat the same step till we reach the final row. Since, we stored the k value in a left to right fashion and in individual row format doing this will take O(n) time and constant space complexity which is significantly better compared to other file compression techniques.

4. Removing excess padding (if any)

After completion of all the previous steps we will have the image exactly matching the original image. But, in some cases like the stated example we have to add extra bits to reach the desired dimension. So, now we need to omit these extra bits without losing the original data. We already know the original image size and our current image size which will be 1037x954 so we are off by 13 rows and 38 columns, in simpler terms we have 13 rows and 38 columns of extra zeros. We can simply, remove the extra rows and columns by slicing them like placing a cookie cutter on a cookie and remove the excess part, we will be able to harness our desired image.

II. Advantages Of Proposed Solution

A. Reduction of File Size: Reducing file size remains the most important benefit of image compression using Tupper's Formula. This means that the image will take up less hard disk space and retain the same physical size or dimensions. This file size reduction works great on the web, allowing webmasters to create image-rich sites without using a lot of bandwidth or storage space. Compression of data also allows users to upload and store data in a concise and storage efficient manner. [9-11]

B. Speed of Delivery: Smaller file sizes mean less disk and memory space, as well as faster transfers whether you're downloading images from your hard drive or displaying them on a web page. Compressed files render faster than uncompressed files. one. Compressed images are almost always used on the Internet where transmission speed is more important than quality, but they can make a significant difference even offline if someone needs to transfer images between devices using a wireless or wired connection. Smaller dimensions provide faster transmission.

C. Faster Loading Speeds for slow devices: Some electronic devices, such as computers and cameras, can be slow to load large, uncompressed images. For example, a CD drive can only read data at a certain speed and cannot display large images in real time. Also, for some web hosts that transfer data slowly, compressed images are still required for websites to function properly. Other forms of storage media, such as hard drives, also have difficulty loading uncompressed files quickly. Compressing images can speed up data loading on slow devices.

D. Lossless Compression: There are two basic types of compression a Lossy compression and a lossless compression; it is clear by their names that one incurs some amount of loss while compressing the overall size of the image whereas the other doesn't. In our case, the compression is lossless.

There are various types of file formats [12] that are used to store image information, some of them are enumerated below:

1. GIF: GIF stands for Graphics Interchange Format, a raster image format introduced by CompuServe in 1987. With support of up to 8 bits per pixel, images can have up to 256 different RGB colors. One of the great things about the GIF format is that you can create animated images that you can't create with the other formats described here. [13]
2. JPEG: Joint Photographic Experts Group (JPEG) is an image format that uses lossy compression to create smaller files. One of the great things about JPEG is that it allows designers to fine-tune the compression ratio. So, when used correctly, you get the best image quality and the smallest reasonable file size. Because JPEG uses lossy compression, images saved in this format are prone to "artifacts" where you can see pixilation and strange halos around certain areas of the image. This is most common in areas of the image where there is a sharp contrast between colors. In general, the higher the contrast of an image, the higher the quality should be in order for the final image to look beautiful. [14]
3. PNG: PNG (Portable Network Graphics) is another bitmap format that uses lossless data compression and was created to replace the GIF image format. PNG has long been largely unsupported by Internet Explorer, making it less common than GIF and JPEG formats, but is supported correctly by all major browsers. PNG files support gamut-based color (24-bit or 32-bit RGBA), grayscale, RGBA, and RGB color spaces. One of the great things about PNG is that it supports a variety of transparency options, including alpha transparency. [15]
4. PBM: The PBM format is a black and white file format with the lowest common denominator. Serves as a common language for a large suite of bitmap conversion filters. This format does not consider efficiency, so it is simple and versatile enough to easily develop programs that convert to almost any other graphic format and manipulate images. The name "PBM" stands for "Portable Bitmap" and is just an intermediate format because it is expensive and less expressive than the format often used to store and send

files. In its purest form, it exists only in the channel between two different programs. In our case we are using PBM file format since the data is stored as series of 0s and 1s.

The big advantage and advantage of lossless compression is that you can maintain image quality while reducing the file size. It not only improves website performance, but also retains image quality.

E. Easy Decoding: In some cases, the decoding algorithm is tedious and requires high computation power. However, Tupper formula is unique in its way that it solves the problem with basic algebra. Whereas, other algorithms like JPEG format require various transformations and other lossy algorithm to store image data. [16-18]

III. Conclusion And Future Applications

In science and manufacturing, image processing and compression is at the forefront of the technological revolution, along with the Internet, integrates real-time production with the display of virtual data to help make processes smoother and more flexible than ever before. Therefore, the future of image compression will build a framework for new technologies and business models. To stay relevant and competitive, assembly lines are automated faster than ever before. The industry now wants to minimize the waste of resources, so it is also very cautious about energy consumption, recycling and scrap management. In such an environment, it's no wonder that technologies such as machine vision and NLP help with automation and quality control. The image processing system helps, for example, measure, count, check, and inspect products much faster than humans. This helps manage the final stages of the manufacturing process. This process is quick and efficient. This means less overall production time and more efficient use of resources and time.

To summarize, our compression or image transformation and rebuilding method is unique and faster compared to other methods. Moreover, it saves a lot of space while delivering information without any loss of data. More techniques can also be applied to further reduce the size of the k values like hashing the values to reduce the string length or using a long string of k values joined by some differentiator such as ";" so that we don't have to iterate over whole array. Future holds a lot of potential for this solution, we can apply this formula on RGB images by transforming them to 3D "numpy" array using python and store the information as bits or can transform them to grayscale values for easy and fast transformation.

IV. References

[1] Tupper, Jeff. "Reliable Two-Dimensional Graphing Methods for Mathematical Formulae with Two Free Variables"
[2] "Pedagoguery Software: Grafeq" : Http://www.peda.com/grafeq/
[3] Jedec Solid State Technology Association (November 2019). "Terms, Definitions, And Letter Symbols For Microprocessors, And Memory Integrated Circuits". Jesd 100b.01. P. 8. Retrieved 2009-04-05.
[4] "What Is Slack Space?". It Pro. 2010-01-19. Retrieved 2018-02-17.
[5] Woods Richard E,gonzalez Rafael C, "digital Image Processing", Pearson Education Singapore Private Ltd,2008.
[6] Journal of Geometric Mechanics 5, 3 (2013), 319-344, Arxiv:1209.6576 [math.ap]
[7] Chandresh K. Parmar, Kruti Pancholi, "a Review On Image Compression Techniques", Journal Of Information,
[8] Athira B. Kaimal, S. Manimurugan, C.s.c. Devadass, "image Compression Techniques: A Survey",
[9] Dugad, Rakesh, And Narendra Ahuja. "A Fast Scheme for Image Size Change In The Compressed Domain." Ieee Transactions on Circuits And Systems For Video Technology 11.4 (2001): 461-474.
[10] Lund, Arnold M. "The Influence Of Video Image Size And Resolution On Viewing-distance Preferences." Smpte Journal 102.5 (1993): 406-415.
[11] Ito, Ryo, Hidenori Kuwakado, And Hatsukazu Tanaka. "Image Size Invariant Visual Cryptography." Ieice Transactions on Fundamentals Of Electronics, Communications And Computer Sciences 82.10 (1999): 2172-2177
[12] Miano, John. Compressed Image File Formats: Jpeg, Png, Gif, Xbm, Bmp. Addison-wesley Professional, 1999.
[13] Eppink, Jason. "A Brief History of The Gif (So Far)." Journal Of Visual Culture 13.3 (2014): 298-306.
[14] Wallace, Gregory K. "The Jpeg Still Picture Compression Standard." Ieee Transactions on Consumer Electronics 38.1 (1992): Xviii-xxxiv.
[15] Roelofs, Greg. Png: The Definitive Guide. O'reilly Media, 1999.
[16] O'rourke, Thomas P., And Robert L. Stevenson. "Improved Image Decompression for Reduced Transform Coding Artifacts." Ieee Transactions on Circuits And Systems For Video Technology 5.6 (1995): 490-499.
[17] Zhang, Xi, And Xiaolin Wu. "Ultra-High-Fidelity Deep Image Decompression With L_∞-constrained Compression." Ieee Transactions on Image Processing 30 (2020): 963-975.
[18] Subramanya, A. "Image Compression Technique." Ieee Potentials 20.1 (2001): 19-23.