American Journal of Applied Mathematics and Statistics, 2014, Vol. 2, No. 4, 246-248
Available online at http://pubs.sciepub.com/ajams/2/4/14
© Science and Education Publishing
DOI:10.12691/ajams-2-4-14

Analyze JPEG File Size for Random Pictures Using Random Function in Mathematica

Tanvir Prince*

Mathematics, Hostos Community College, City University of New York, USA
*Corresponding author: tprince@hostos.cuny.edu

Received July 14, 2014; Revised August 09, 2014; Accepted August 13, 2014

Abstract Using the random function in Mathematica (a mathematical software), random numbers between 0 and 1 were generated which in turn used to create random pictures. The random values represent the gray scale of the black and white pictures. Thus hundreds of pictures were created randomly and compressed using JPEG algorithm. Then a directory list containing the file size is created and imported in the Microsoft excel worksheet to analyze. All the code is given in the paper that is used so that it can be carried out by any interested reader. As a byproduct, this can be used as a classroom activity or as a project for any college level mathematics courses.

Keywords: *random pictures, mathematica, JPEG compression*

Cite This Article: Tanvir Prince, "Analyze JPEG File Size for Random Pictures Using Random Function in Mathematica." *American Journal of Applied Mathematics and Statistics* vol. 2, no. 4 (2014): 246-248. doi: 10.12691/ajams-2-4-14.

1. Introduction

Nowadays images are everywhere- from newspapers to internet. Most of the images that we see in our everyday life go through some types of compression to save space and transmission time through internet. In actuality, an image nothing but consists of a lot of pixels. For a black and white pictures, each pixel contains a numerical value between 0 and 1 which represent its gray level – 0 being completely white and 1 being completely black. For a color picture, each pixel contains a vector of length three where the value represent the magnitude of red, green and blue. Here red, green and blue are the primary colors from which all the other colors can be created. We use mathematica software and its random function generators to create hundreds of random value between 0 and 1. Then these random numbers were used to create random pictures. For simplicity, we only created pictures of size 16 by 16. But this size can be changed easily by modifying the code in the appropriate places. Later we exported these random pictures using "JPEG" extension and created an excel file using their file size. At the end we explore this to analyze and make some conclusion about their behavior.

2. Image to Pixel

Let us look at the following pictures:

Figure 1. A color picture of a flower [1]

We can simply drag this picture in the mathematica [2] notepad to start explore its various features. One of the command which convert the pixels into its numerical value is the command:

ImageData[picture]

This will usually generate a huge array of data of which only a little part is shown below (because of the space).

```
…………..(0.01960784313725490.2274509803921568
60.00784313725490196)(0.03921568627450980.2470588
23529411780.03529411764705882)(0.066666666666666
670.25490196078431370.058823529411764705)(0.02745
09803921568620.235294117647058820.01568627450980
392)(0.0431372549019607840.250980392156862740.031
37254901960784)(0.070588235294117650.25882352941
1764730.054901960784313725)(0.03921568627450980.2
50980392156862740.023529411764705882)(0.05490196
07843137250.26666666666666660.0392156862745098)
(0.082352941176470590.274509803921568650.0588235
29411764705)(0.0549019607843137250.2666666666666
66660.023529411764705882)(0.066666666666666670.27
843137254901960.043137254901960784)(0.0901960784
31372550.28235294117647060.06274509803921569)(0.0
74509803921568630.29019607843137260.035294117647
05882)(0.082352941176470590.294117647058823540.05
0980392156862744)(0.101960784313725490.298039215
68627450.058823529411764705□)……….continue….
```

Here only about 15 pixel values are shown and each pixel value is enclosed in parenthesis. If the reader looks carefully then he can find that each parenthesis contains three values. These are the values of RGB (red, green and blue value of the pixel).

3. Going Backward

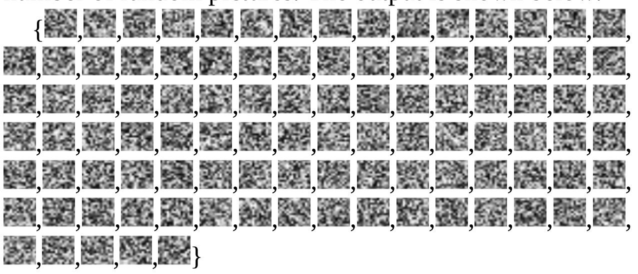
Now we will invert the process. In other words, we will start with numbers and will create images from it. To make it simple, we will only consider black and white picture of size 16 by 16. So each picture has 256 pixels. In real life, a typical picture usually have dimension in the range of hundreds to even thousands. For example, the picture in Figure 1 have the dimension 259 by 194. So the total number of pixel in that picture is 50,246. We avoid the big numbers here because it will take a long time to process the data and there is no additional benefit that we will gain from this. The interested readers can change the code in the appropriate places to change the size of the random pictures. We use the build in command "RandomReal" to generate the random numbers. To be more precise, we use:

RandomReal[{0,1},{16,16}]

The above command generate a random 16 by 16 matrix where each entry is between 0 and 1. This will converted to a random picture. The following command:

Table[Image[RandomReal[{0,1},{16,16}]],{*i*,1,100}]

Creates a list of 100 random images. The readers can change the last entry of the command to change the number of random pictures. The output is shown below:



4. Exporting Pictures

Now it is time to export all these pictures into a folder where we will extract the file size of each one of them. Here we have several choice on the format of the saved pictures. We choose "JPEG" format since it is one of the most common type of image compression used in general. To learn more about image compression see [3,4,5]. For more technical reading see [6] where some of the algorithm and the mathematics behind the image compression can be found.

We use the following command to do this:

Export["C:\\Users\\prince\\Documents\\papers\\random_picture\\random_picture_items\\image001.jpg",*T*,"VideoFrames"]

The command specify the full path of the folder where we want to save the pictures.

5. Exporting File Size to Excel

Now we use the "Directory Listing" to export the all the file size into an excel file where further analysis of the result can be done. Only a small part of the table is given below

2/17/2014	8:39	AM	771	image001.jpg
2/17/2014	8:39	AM	771	image002.jpg
2/17/2014	8:39	AM	761	image003.jpg
2/17/2014	8:39	AM	769	image004.jpg
2/17/2014	8:39	AM	767	image005.jpg
2/17/2014	8:39	AM	773	image006.jpg
2/17/2014	8:39	AM	770	image007.jpg
2/17/2014	8:39	AM	764	image008.jpg
2/17/2014	8:39	AM	770	image009.jpg

Note that by default the date and time of the export is given in two different column of the excel file. The fourth column is the file size. The unit of the file size is Byte.

6. Analyze Result

We created a histogram of the file size and get the following figure:

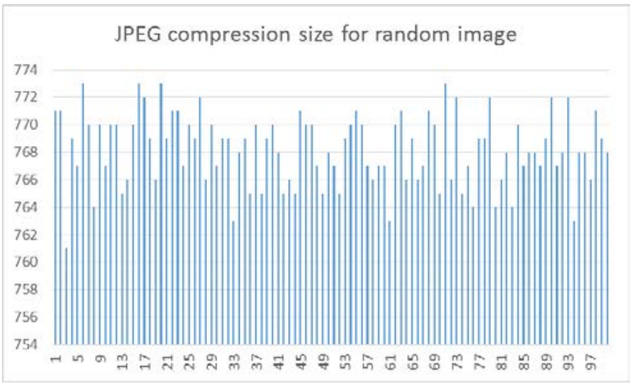


Figure 2. Histogram of the file size

The following 3D pie chart is more explanatory then the histogram:

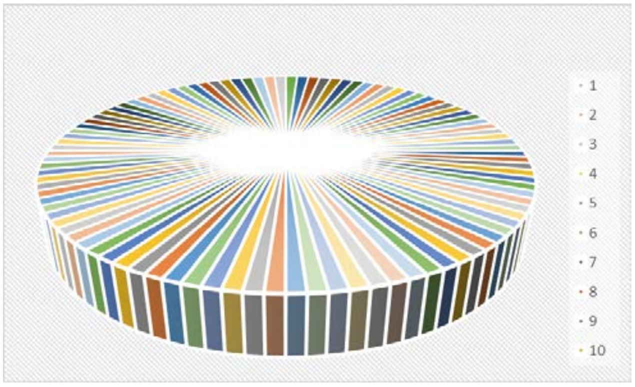


Figure 3. 3D pie chart of the file size

From Figure 2 and Figure 3 above, it is clear that the distribution is almost uniform throughout.

7. Conclusion

The uniform distribution of the file size testify the true random nature of the random function generator of the Mathematica software. Of course, no software can guarantee an ideal randomness but the mathematica software get very close to the actual situation. The experiment can be repeated using other software like “maple” or “matlab” to compare the result with each other. We invite interested readers to perform the similar experiment using different software platform. This can be assigned as a short project to a college level mathematics class. As a bi-product, students will be familiarized with the use of mathematica software. To learn more about mathematica see [7,8].

References

[1] In Courtesy of en.wikipedia.org.

[2] For more information about the software and to download the product the readers can visit www.wolfram.com.

[3] Image Compression. (2011). *Image Compression: How Math Led to the JPEG2000 Standard*. Retrieved from Image Compression: How Math Led to the JPEG2000 Standard. (2011). Retriwww.whydomath.org/node/wavlets/basicjpg.html.

[4] Group, S. (n.d.). *JPEG Compression: What it is - when to use it - and when not to*. Retrieved from Retrieved from the university of Oslo website: http://folk.uio.no/inf9540/SVD.pdf.

[5] Rahman Z., J. D. (n.d.). Image enhancement, image quality, and noise, Photonic Devices and Algorithms for Computing., (pp. VII, Proc. SPIE 5907).

[6] Prince, T., Franco, S., Salva, I., & Windolf, C. (2014). Mathematics Behind Image Compression. *Journal of Student Research*, *3*(1), 46-62. Retrieved from http://www.jofsr.com/index.php/path.

[7] Purdue University. (n.d.). *A Brief Introduction to Mathematica*. http://www.cs.purdue.edu/homes/ayg/CS590C/www/mathematica/math.html

[8] Wolfram Mathematica. (2013). Hands-on Start to Mathematica. Retrieved from Wolfram: http://www.wolfram.com/broadcast/screencasts/handsonstart/.