Lossy Compression in the Chroma Subsampling Process

PAVEL POKORNY
Department of Computer and Communication Systems
Faculty of Applied Informatics, Tomas Bata University in Zlín
Nad Stráněmi 4511, 760 05 Zlín
CZECH REPUBLIC
pokorny@fai.utb.cz http://fai.utb.cz

Abstract: - Nowadays, compression algorithms are frequently used in the computer software field. The reason is the large amounts of data that is processed by applications. These compression algorithms, in different forms, are deployed in many raster image processing cases. The reasons for this are the quite high compression ratios and also, the usually high compression/decompression speed (the success of these two major parameters of the compression algorithms depends on the image information content). In addition, the images can often allow one to use lossy compression algorithms which cause a substantially greater reduction of the image data size. The JPEG graphic format is the most widely used. Here, the loss rate is determined in two ways by the subsampling color components and the quantization of the coefficients that are obtained by the Discrete Cosine Transform calculation. In this paper, attention is focused on the subsampling process of the color components. The content compares the most frequently used sampling techniques - mainly to the quality and size of the processed image.

Key-Words: - Image Compression, Lossy Compression, Image Quality, Sampling, Colors

1 Introduction

Data compression is the art or science of representing information in a compact form. We create these compact representations by identifying and using structures that exist in the data. Data can be characters in a text file, numbers that are samples of speech or image waveforms, or sequences of numbers that are generated by other processes. The reason we need data compression is that more and more of the information that we generate and use is in digital form - in the form of numbers represented by bytes of data. And the number of bytes required to represent multimedia data can be huge. For example, in order to digitally represent 1 second of video without compression (using the CCIR 601 format), we need more than 20 megabytes, or 160 megabits [1].

Compression can be either lossy or lossless. Lossless compression reduces bits by identifying and eliminating statistical redundancy. No information is lost in lossless compression i.e. the original data can be recovered exactly from the compressed data. Lossless compression is possible because most real-world data have statistical redundancy. For example, an image may have areas of color that do not change over several pixels; instead of coding "red pixel, red pixel ..." the data may be encoded as "279 red pixels". This is a basic example of run-length encoding; there are many schemes to reduce file size by eliminating redundancy. Many files and graphic formats use this principle with different modifications.

Lossy compression techniques involve some loss of information, and data that has been compressed using lossy techniques generally cannot be recovered or reconstructed exactly. In return for accepting this distortion in the reconstruction, we can generally obtain much higher compression ratios than is possible with lossless compression [1]. Lossy data compression schemes are designed by research on how people perceive the data in questions. For example, the human eye is more sensitive to subtle variations in luminance than it is to the variations in color. JPEG image compression or Wavelet Transform work in part by rounding off nonessential bits of information [2].

Programs using simple or complex graphics are appearing in virtually every area of computing applications: e.g. games, education, desktop publishing or graphical design, just to mention a few. These programs all have one factor in common. The images they use consume prodigious amounts of memory or disk storage space. The raster representation method is very often used for images (a display mode for the vast majority of hardware). The amount of image data depends on the resolution and color depth in this case. For example, a single image with a resolution 800 x 600 pixel and 224 color depth consumes over 1.4 MB of memory. It isn't hard to imagine applications that would require literally hundreds of these images to be accessed [9].

Compression offers the solution to this problem. In images, its objective is to reduce the irrelevance and redundancy of the image data in order to be able to store or transmit data in an efficient form. Lossy methods are especially suitable for natural images such as photographs in applications where minor (sometimes imperceptible) loss of fidelity is acceptable in order to achieve a substantial reduction in bit rate.

A range of different lossy compression methods are used currently. One of them is to reduce color space to the most common colors in the image (i.e. the selected colors are specified in the color palette that is included in the image file). Another method is based on transform coding. This is the most commonly used method and forms the core of the jpeg images [3][11][8].

The third lossy compression method is chroma subsampling, which takes advantage of the fact that the human eye perceives spatial changes of brightness more sharply than those of color [4][5].

This paper is focused on this (third) compression method. The second chapter describes the theory of the subsampling process. The first section of the second chapter presents the color transformations that are necessary in order to separate brightness and color components. The second section lists the most commonly used subsampling methods at present. The third chapter shows the created application and describes both the user's and programmer's views. The fourth chapter provides information about the results that these created applications allowed us to obtain and a discussion about data-saving and decreasing image quality.

2 Chroma Subsampling

The Chroma subsampling process requires color transformations. The following text also begins by describing these, and then goes on to delineate chroma subsampling methods.

2.1 Color Transformation

As mentioned above, the subsampling algorithms concern only the color components of the picture. The images themselves are mostly represented by the RGB or CMYK color models, or by the internal color space of CCD cameras. For this reason, the first step is to separate the brightness (sometimes called "luma") and color components. This conversion process usually means a transformation to one of the models used in color analog television techniques e.g. YIQ (the PAL TV standard), YUV (the NTSC TV standard) or YCbCr (the SECAM TV standard).

This transformation between two color spaces is lossless, i.e. there is no possibility of losing any image information. So, if one converts an image into a second one with any color space used in TV techniques, it is possible to convert it back to the RGB or CMYK color model at any time and we always get a result identical to the original image. [10]

The transformation equation between RGB and YIQ is:

$$\begin{aligned} Y &= 0.299\times R + 0.587\times G + 0.114\times B \\ I &= 0.596\times R - 0.275\times G + 0.321\times B \\ Q &= 0.212\times R - 0.523\times G - 0.311\times B \end{aligned} \tag{1}$$

The relation between RGB and YIQ is given by the equation:

$$\begin{aligned} Y &= 0.299\times R + 0.587\times G + 0.114\times B \\ U &= -0.147\times R - 0.289\times G + 0.436\times B \\ V &= 0.615\times R - 0.515\times G - 0.100\times B \end{aligned} \tag{2}$$

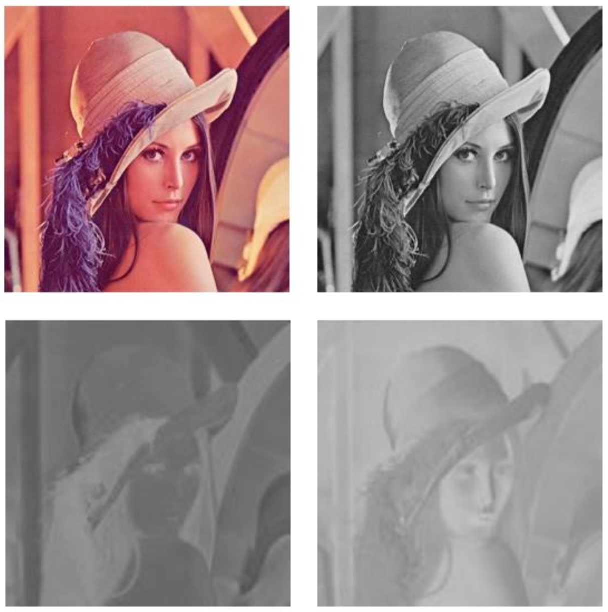


Fig. 1 Top left: Original image; Top right: The Y component; Bottom left: The Cb component; Bottom right: The Cr component [12]

The RGB and YCbCr transformation equation is:

$$\begin{aligned} Y &= 0.299 \times R + 0.587 \times G + 0.114 \times B \\ Cb &= -0.1687 \times R - 0.3313 \times G + 0.5 \times B + 128 \\ Cr &= 0.5 \times R - 0.4187 \times G - 0.0813 \times B + 128 \end{aligned} \quad (3)$$

The main concern of this article is the last-mentioned color model (YCbCr), because it is used very often, for example, in the JPEG graphic format. The reverse conversion is performed according to the following formula:

$$\begin{aligned} R &= Y + 1.402 \times (Cr - 128) \\ G &= Y - 0.34414 \times (Cb - 128) - 0.71414 \times (Cr - 128) \\ B &= Y + 1.772 \times (Cb - 128) \end{aligned} \quad (4)$$

Figure 1 shows the original image and the same image with the separated components Y, Cb and Cr. It is possible to clearly see that the most significant component is precisely Y in this picture. In contrast, the Cb and Cr components reported low intensity. For this reason, the subsampling algorithms refer only to these components.

2.2 Subsampling

There are several ways how to make the chroma subsampling process. It is possible to find them in all analog color television standards (SECAM, PAL, NTSC) in a certain form. Usually, two subsampling methods are performed - these methods are often used in the JPEG graphic format. The first method calculates the ratio of the color of the neighboring pixels that are placed on the screen in a row. The second algorithm calculates the average color of the four neighboring pixels (these pixels are placed in the two adjacent rows and columns on the screen). Both methods are lossy; when the algorithm calculates the final color component, in the first case, the size of the raster data is reduced by about 33%, in the second case by about 50%. [6][12]

In practice, subsampling is designated by a string of 3 (or sometimes 4) integers separated by colons, e.g. 4:2:2:4. The relationship among the integers denotes the degree of vertical and horizontal subsampling. At the outset of digital video, subsampling notation was logical; unfortunately, technology outgrew the notation. [7] Today's notation means:

• The first number represents the horizontal sampling reference (the width of the area in pixels, which are subsampled). Usually, this value is equal to 4.
• The second number is the horizontal factor of the Cb and Cr components (i.e. number of chrominance samples in the rows of the processed area).
• The third number describes the vertical factor of the Cb and Cr components (i.e. number of chrominance samples in the columns of the processed area).
• The fourth number is like the first number. The difference is, it represents the horizontal sampling reference for the alpha channel (i.e. transparency) in the case where the image supports it. This number is not used without the transparency.

The most commonly used leading digit of 4 is a historical reference to a sample rate roughly four times the NTSC or PAL color subcarrier frequency; the notation originated when subcarrier-locked sampling was under discussion for component video [7]. The most common chroma subsamplings are:

• 4:4:4 – without subsampling. Each of the three YCbCr components has the same sample rate, like the input resolution. This scheme is commonly used in cinematic post-production and a different compression method is usually used in the following steps.
• 4:2:2 - both chroma components are each subsampled by a factor of 2 horizontally; their effective positions are coincident (co-sited) with alternate brightness. Many digital video formats and interfaces use this scheme.
• 4:1:1 - in this scheme, the Cb and Cr components are each subsampled by a factor of 4 horizontally and are coincident with every fourth brightness sample. Currently, this subsampling is used for low-end and consumer applications.
• 4:2:0 – the Cb and Cr components are each subsampled by a factor of 2 horizontally and a factor of 2 vertically. This scheme is used in JPEG/JFIF still-frames in computing, in H.261 (for video-conferencing), in MPEG-1, in consumer 576i25 DV25, and in most variants of MPEG-2. Different technologies support different variants of 4:2:0 subsampling. In JPEG/JFIF, H.261, and MPEG-1, Cb and Cr are sited interstitially, horizontally halfway between alternate brightness samples. In MPEG-2, Cb and Cr are co-sited horizontally.
• 4:1:0 – This ratio uses half of the vertical and one-fourth of the horizontal color resolutions, with only one-eighth of the bandwidth of the maximum color resolutions used. An uncompressed video in this format with 8-bit quantization uses 10 bytes for

every macro-pixel (which is 4 x 2 pixels). Some codecs support this ratio - but it is not widely used.
• 3:1:1 – the Cb and Cr components are each subsampled by a factor of 3 horizontally. The chroma samples are then divided by every third brightness sample. 36 bytes of the R, G and B components are also reduced to 20, effecting approximately 2:1 compression. [7]

The above described differences among chroma subsamplings 4:4:4, 4:2:2, 4:1:1 and 4:2:0 are shown in Figure 2.

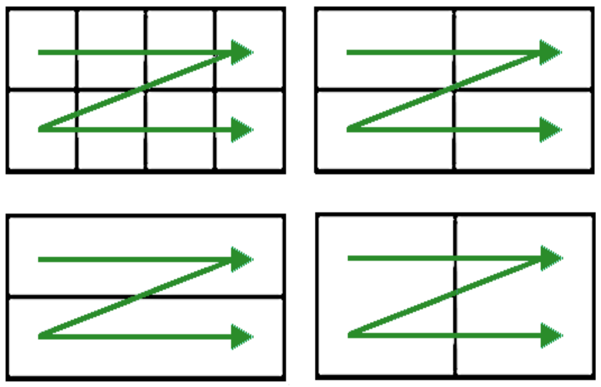


Fig. 2 Top left: Subsampling 4:4:4; Top right: Subsampling 4:2:2; Bottom left: Subsampling 4:1:1; Bottom right: Subsampling 4:2:0

3 A Testing Application
In order to evaluate the results of the subsampled color components, an application based on HTML5 and the Javascript programming language was created. [13] The advantage of these techniques is there simplicity and flexibility because this application can be run in any web browser.

3.1. User's View
The user interface of this application is shown in Figure 3. Its largest part is a pair of two windows that are defined by the HTML5 canvas element. In the left window, the original image is shown, while the right window represents the output image, which is generated according to the set parameters. The lower part of the user interface contains the whole control commands and parameters. The first button allows one to load any image from a file (bmp, gif, png and jpg are supported image formats). A pop-up menu (drop-down list) is to the right of this button, where the user can choose the subsampling method. The third button includes text "Calculate" and clicking the mouse on it starts a subroutine in the Javascript language, which ensures the subsampling calculation according to the selected method and then displays its result in the right window. The last button, called "Reset Output", erases subsampling results and clears the right window.

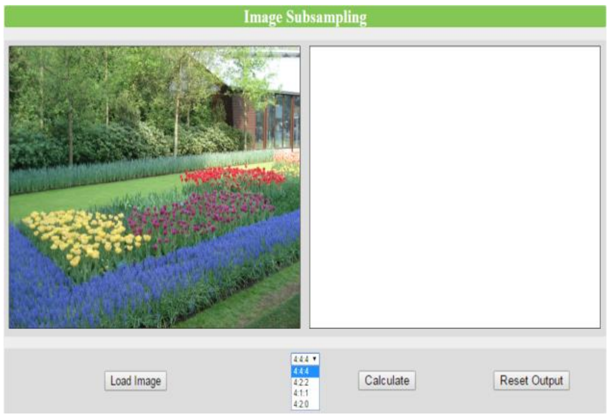


Fig. 3 User interface of application to test the image subsampling

3.2. Programmer's View
The design of the created application is organized into two files - index.html and script.js. The index.html file contains the whole HTML code and also all properties that are set by the help of Cascading Style Sheets (CSS). The HTML code is normally determined by tags. Each of them has its own identifier (the "id" attribute). The buttons (tag <input>) have additional "value" properties, which contain a text description of each button. The drop-down menu for selecting the type of color subsampling is designed by the <select> tag. This tag includes the "name" attribute that uniquely identifies it. The content of the <select> tag is formed by the <option> tags that represent all of the options of this selection menu. Each option has its own "value" attribute, with a unique name, in order to clearly identify which method has been chosen.

Both images (input and output) are displayed in the windows. Each window is defined by the <canvas> tag. These tags have their own attributes - dimension (width and height) and a unique identifier for working in the correct window. The HTML <canvas> element is used to draw graphics or pictures on the fly, via scripting (usually JavaScript).

The location of all user environment elements is given by Cascading Style Sheets (CSS). CSS is a stylesheet language that describes the presentation of an HTML of XML document. In our application, it mainly set the colors of each tag and the background color behind them. It also aligns the texts in the user environment (the "text-align" parameter) and all edges including borders ("margin", "border", "border-radius", and "padding" properties). For mutual placement of the windows, the left canvas has the property "float: left" set additionally.

3.2.1 Javascript

The whole program code is divided into individual functions. Global initializations represent only exceptions from this. These initializations include the canvasses specifications, access to user interface buttons and declarations of global variables.

The user can draw on the canvas using its primary rendering context, which can be obtained on the basis of an object handle. This handle can be derived by calling the "document.getElementById" function, which contains a single parameter (object identifier). In this case, this means the id attribute of relevant canvas. Access to the context is then given by calling the "getContext" method, see below:

```
var canvas1 = document.getElementById ("Can1"),
canvas1Context = canvas1.getContext ("2d");
```

Using this, it is possible to generate the contexts of both canvasses. These contexts provide one with a large number of methods that programmers can use to draw many graphic elements - like rectangles, complex paths or images, to set customizing drawing styles, or use various transformations.

In order to communicate with the user through the application interface, it is also necessary to obtain the handles of whole command elements. These elements include buttons and the drop-down menu, as mentioned above. To do this, we use the "document.getElementById" function again.

The application also contains only two global variables (objects) – "image" and "imageData". The "image" variable represents an image object. Its declaration is performed by the command:

```
image = new Image();
```

After this initialization, the programmer can set the image source and draw it on the selected canvas. The "imageData" object represents pure data of an image, i.e. typically R(ed), G(reen), B(lue) and the A(lpha) values of each pixel in the given image. This is necessary so as to have direct access to the whole range of color and transparency components and to calculate the subsampling process.

When the user clicks on the selected button (Load Image, Calculate or Reset Output), this action causes an event that can be detected by the help of the "onclick" method. This method can run any function that can do whatever a programmer wants.

Clicking the Load Image button runs the function that ensures loading the selected image from a file. The loaded image is stored in the image object and it is immediately drawn on the canvas with the help of the "drawImage" method. This method allows drawing in different forms – direct draw, image scaling before drawing or image clipping in the rectangle form or scaling and drawing on the canvas in the defined position.

The Reset Output button causes the calling of a simple function, where only one command deletes the output canvas window. This command is performed by the help of the "clearRect" method.

The function that is called by clicking on the Calculate Button is the most complex. It has to perform several operations in the correct order:

- "image" to "ImageData" conversion
- RGB to YCbCr transformation
- chroma subsampling based on selected method
- YCbCr to RGB transformation
- "ImageData" to "image" conversion

The "image" to "ImageData" conversion is performed by the "getImageData" method which transfers pixel values from the input canvas into the "ImageData" object. The reverse process at the end of the operation list is performed by the "putImageData" method. The color transformation processes and chroma subsampling calculation are programmed based on the algorithms described in the previous chapters of this paper.

4 Results and Discussion

To test the created application, several different images were used. The commonly used jpg graphic format was avoided because it already includes lossy compressed data and the results would be inaccurate. That is the reason why images were used and why were rendered in the Blender software [15]. The rendered images were exported to the PNG graphic format because this format uses the lossless compression algorithm. Figure 4 shows a zoomed selected part of one image - a roof with a chimney in 64 x 64 pixel resolution. The original image is in the top left corner, the other images represent outputs created using the 4: 2: 2, 4: 1: 1 and 4: 2: 0 subsampling algorithms. The differences are not very noticeable in the full resolution of these images. So, this zoomed part of the image was created in order to see more visible differences in the various types of the subsampling algorithms. The most significant changes are logically evident after using the 4: 1: 1 and 4: 2: 0 subsampling. [14]

The subsampling process causes merging (usually by using the averaging calculation) of the color components of the neighboring pixels. In essence, this leads to the blurring effect that is more pronounced with the increasing number of merged pixels. This implies that this method is particularly suitable for photographs or rendered images which don't contain sharp edges or color gradients. In these cases, the subsampling of the color components is the most visible.

Table 1 shows the data-saving for each type of subsampling. This is the total savings of the original data, i.e. the percentage numbers include all three components.

Tab. 1 Data-saving using different subsampling methods

Subsampling method	4:4:4	4:2:2	4:1:1	4:2:0
Save	0%	33%	50%	50%

A comparison of Figure 4 and Table 1 evokes the idea of seeking a compromise between image quality and the quantity of image data. If the highest image quality is preferred, avoiding the losing compression algorithms including the subsampling processes is the best solution. The second thing is, if the images are intended for common user purposes, some loss rate can usually be tolerated, because the image quality can be decreased very slightly. And, Table 1 shows that the user can obtain very interesting memory storage savings.

There is a problem to define the term “acceptable image quality”. How much image information can be lost so one ever could say that image still provides sufficiently high quality image information? Classical methods that identify an image quality (expressed errors) are not applicable because their solution offers strange results. For example, a comparison of the differences between the original image and the image with the subsampled color components can often give the result that none of the pixels have the same value. So, the typical method for error measurement (i.e. the sum of the squares of the differences of color pixel values) can return high values, even where the differences between these images are almost unnoticeable. [11]

5 Conclusion

Subsampling algorithms offer a very interesting option to reduce the amount of image data. Due to the fact that the loss concerning image information is the least noticeable to the human eye, this process is preceded by the conversion process, where separated brightness and color components are separated. Then, subsampling is only used for these color components.

Currently, several different types of subsampling algorithms are used in the image processing process and - depending on its type, this leads to the corresponding compression results. Practical experience shows that the subsampling process is suitable for images with high color depths and larger resolutions; and lost colors are more visible in images that contain sharp edges or color gradients.

In order to test the subsampling algorithms, the application in HTML5 in combination with the JavaScript programming language was created. This application allows one to subsample components by using 4:4:4, 4:2:2, 4:1:1 and 4:0:0 methods. Currently, these standards are often used in some graphics images and many video formats. This application that was created could be further extended in the future. This extension can contains further subsampling algorithms including transparency or the support of different color representations (e.g. YUV, YIQ). In addition, this application could support other user-friendly features such as zoom in/out and navigation of the images, or save the output into a file.

The increased compression ratios can be further reached by using it in combination with other compression algorithms - just as the JPEG standards do. Future work will also proceed in this direction.

References:

[1] K. Sayood, *Introduction to Data Compression*, 3rd edition, Elsevier, 2006, 680 p., ISBN 9780126208627.

[2] A. Khasman and K. Dimililer, Haar Image Compression Using a Neutral Network, *ACC'08 Proceedings of the WSEAS International Conference on Applied Computing Conference*, 2008, pp. 412-417, ISBN 9606766671.

[3] M. Petrou and C. Petrou, *Image Processing: The Fundamentals*, A John Wiley and Sons. Ltd., 2010, 794 p., ISBN 9780470745861.

[4] H. Chen, S. Mingzhe and Eckehard Steinbach, Compression of Bayer-pattern video sequences using adjusted chroma subsampling. *IEEE*

Transactions on Circuits and Systems for Video Technology, vol. 19, 2009, pp. 1891-1896.

[5] Ch. Glenn, Toward Better Chroma Subsampling Recipient of the 2007 SMPTE Student Paper Award, *SMPTE Motion Imaging Journal*, vol. 117, 2008, pp. 39-45.

[6] C. Hass (2008). JPEG Chroma Subsampling [Online]. Available: http://www.impulseadventure.com/photo/chroma-subsampling.html

[7] Ch. Poynton (2008). Chroma Subsampling notation [Online]. Available: http://w.poynton.com/PDFs/Chroma_subsampling_notation.pdf

[8] Prepressure contributors (2015). The JPEG file format [Online]. Available: http://www.prepressure.com/library/compression-algorithm/jpeg

[9] M. Nelson, Lossy Graphics Compression, *The Data Compression Book*, 2 edition, Willey, Cambridge, UK, 1995, pp 216-255.

[10] Y. Wiseman (2013). The still image lossy compression standard – JPEG [Online]. Available: http://u.cs.biu.ac.il/~wiseman/jpeg2.pdf

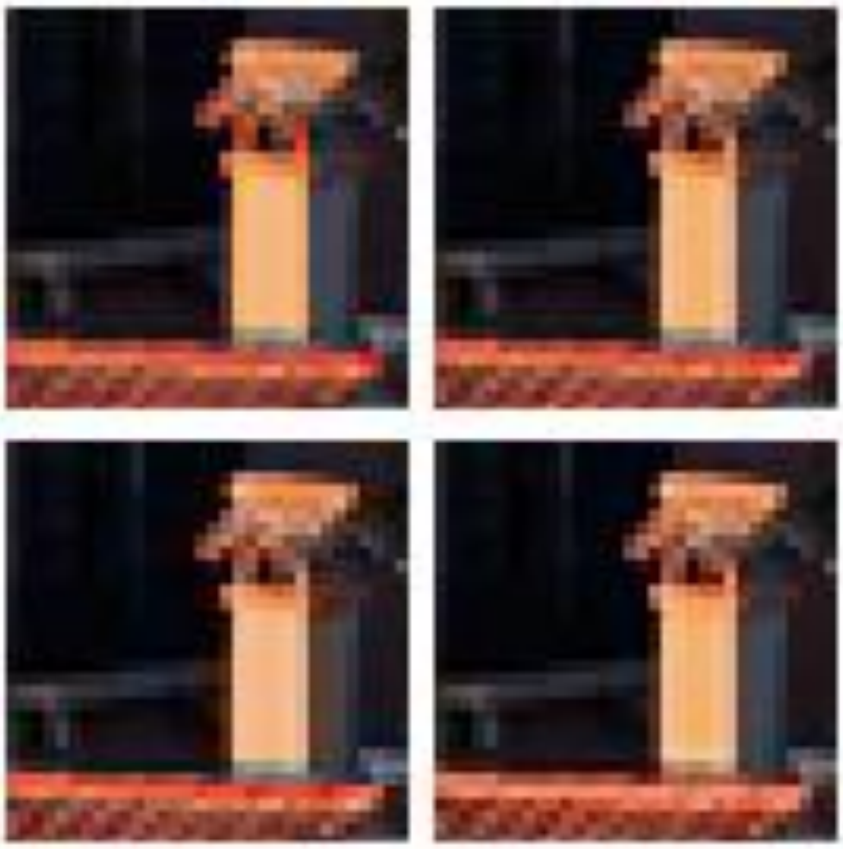
[11] P. Tisnovsky (2006). Lossy compression with JPEG [Online]. Available: http://www.root.cz/clanky/ztratova-komprese-obrazovych-dat-pomoci-jpeg/#ic=serial-box&icc=text-title

[12] P. Tisnovsky (2006). Programming JPEG – color transformation and subsampling [Online]. Available: http://www.root.cz/clanky/programujeme-jpeg-transformace-a-podvzorkovani-barev/#ic=serial-box&icc=text-title

[13] W3C contributors (2014). HTML5 - a vocabulary and associated APIs for HTML and XHTML [Online]. Available: http://www.w3.org/TR/html5/

[14] Red contributors (2015). Chroma subsampling techniques [Online]. Available: http://www.red.com/learn/red-101/video-chroma-subsampling

[15] Blender Foundation (2015). Blender – Free and Open 3D Creation Software [Online]. Available: http://www.blender.org/



Top left: Original image; Top right: The 4:2:2 image subsampling;
Bottom left: The 4:1:1 image subsampling; Bottom right: The 4:2:0 image subsampling