Image to Sound and Sound to Image Transform

Haralampos C. Spyridis* - Aggelos K. Moustakas**

* Professor of Music & Informatics, Head Manager of the Sound Engineering Laboratory of
Music Department, University of Athens.
hspyridis@music.uoa.gr
** Physics Scientist, MSc Candidate of Informatics – University of Piraeus, Greece.
agpetmous@ath.forthnet.gr

ABSTRACT

Objective of our paper is the presentation of a software we wrote, which creates the sound analog of an image, as well as the image analog of a sound.

To the best of our knowledge, the proposed process is original.

I. INTRODUCTION

We present software, in which via an algorithm it produces melodies converting the RGB information of each pixel on a digital image to sound frequencies. These frequencies are quantized on the tempered European major/minor scale.

More specifically, after the reading of the RGB information of every pixel on the image saved in the computer we:

- Convert the chromatic information into a "monochromatic" frequency of the optical range (spectrum) and a new image is generated.
- Transform the frequencies of the optical range into frequencies of the human acoustic range.
- By means of a mathematical formula we group the sound frequencies and match them to notes of the tempered European scale with a specific pitch and duration.

Therefore, we are able to listen to the image via its resulted monophonic music analog.

Moreover, reversing the above procedures, we create the optic analog of a monophonic melody. We design images by using the resulting RGB information, transforming the frequencies of the tempered European scale, in which the monophonic melody is composed. The same also applies in polyphonic compositions.

II. IMAGE TO SOUND CONVERSION

A. CIE and the RGB Model

The main obstacle we have to overcome is color representation. Computers usually use RGB color space and therefore we have to find a match between RGB values and color wavelengths, which presents a serious problem.

International Commission On Illumination (CIE) is using a mixture of three CIE primaries, **X**,**Y**,**Z** to represent any real color [9]. However, these primaries correspond to no real color; they are just convenient mathematical constructs.

Therefore, a given color C of wavelength λ can be expressed by the following function:

$$C_\lambda = X\mathbf{X} + Y\mathbf{Y} + Z\mathbf{Z} \tag{1}$$

The values XYZ, called *tristimulus values*, are being calculated using the CIE color matching functions denoted as $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, $\bar{z}(\lambda)$ (Fig. 1):

$$\begin{aligned} X &= k\int_\lambda \bar{x}(\lambda)\Phi(\lambda)d\lambda \\ Y &= k\int_\lambda \bar{y}(\lambda)\Phi(\lambda)d\lambda \\ Z &= k\int_\lambda \bar{z}(\lambda)\Phi(\lambda)d\lambda \end{aligned} \tag{2}$$

where Φ(λ) is the spectral distribution of light stimulus and *k* is a normalizing constant.

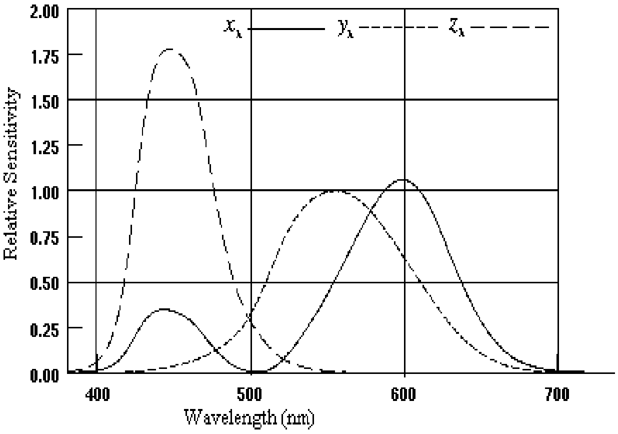

Fig.1 The CIE 1931 color matching functions

The normalization of X,Y,Z values results in the production of the x,y,z values:

$$x = \frac{X}{X+Y+Z} \qquad y = \frac{Y}{X+Y+Z} \qquad (3)$$

Since x+y+z=1, z can be omitted. Therefore, we use *x,y* values, known as *chromaticity coordinates*, to plot the *CIE (x, y) chromaticity diagram* (Fig.2).

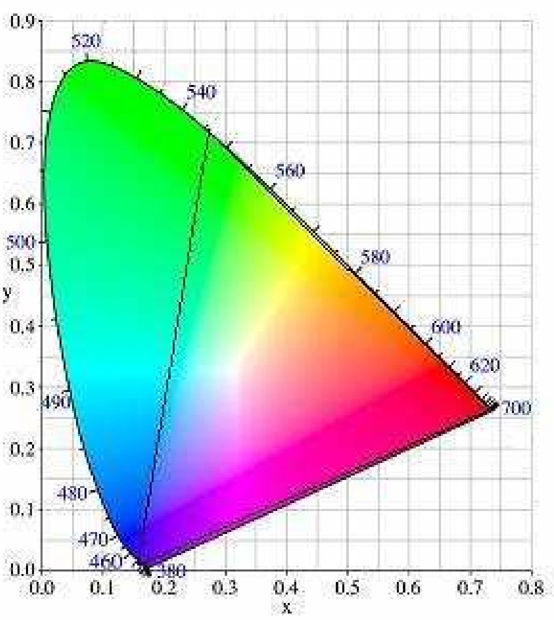

Fig.2 CIE 1931 Chromaticity diagram, with RGB gamut.

In computers, the chromatic model that is basically used is RGB [5], according to which every color occurs as a mix of Red (R), Green (G) and Blue (B) (700 nm, 546.1 nm, and 435.8 nm respectively). For every pixel in a true color screen correspond 3 bytes of information. Each of them, valued from 0 to 255 (2^8=256), defines the weight of participation of each of the three primary colors in the composition of the desired color. The RGB curves are shown in Figure 3.

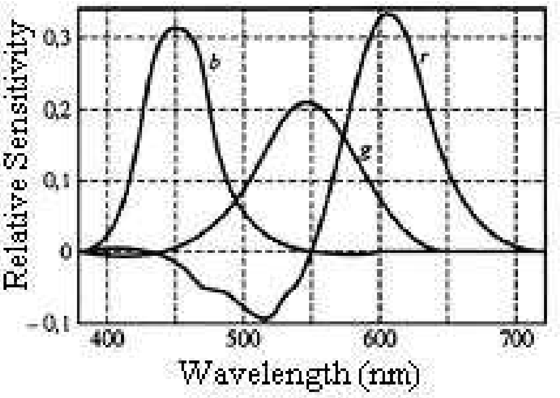

Fig 3: RGB curves construct the rest of the colors.

A thorough examination of the curves representing the primary colors in the RGB model shows a weakness in the representation of some frequencies in the optical frequency range. This weakness springs from the Red curve, which takes negative values in some places of the wavelengths of the optical frequency range. This means that red has to be added to the monochromatic stimulus to make the match.

The above weakness makes RGB model incapable of representing all real colors. The triangle in Figure 2 shows the purely additive area of RGB, which constitutes the RGB gamut. Colors outside of this gamut can not be represented properly. Note that each display device has its own RGB gamut. The underlying reason for RGB model's failure to represent all the possible colors is the fact that the color calculations are translated from the wavelength domain into the display domain [1,5].

Specifically, the RGB model defines a three-dimension area which contains all the colors which could be possibly displayed on a PC.

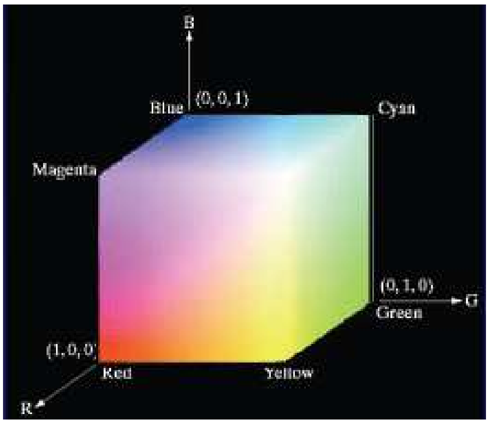

Fig 4: The 3-dimension area defined by the RGB, which contains all the colors a PC may display.

Each axis of the 3 dimension area gets (2^8=256) values defined as [0,255] or [0,1] in normalized values. Point (0,0,0) represents black color, and (1,1,1) white color. Every other color is calculated with the help of the following linear formula:

$$F = r\,\mathbf{R} + g\,\mathbf{G} + b\,\mathbf{B} \qquad (4)$$

So, the RGB color space is produced (Fig. 4).

Due to the nature of the color information in the computers, there is not an effective one-to-one mapping between the frequencies and the RGB values.

We were therefore forced to use a heuristic method of transformation. Bruton algorithm [2] was chosen because of its very good results when transforming frequencies to RGB values. Figure 5 shows the RGB values for the optical wavelengths as they resulted from the Bruton algorithm.

The above algorithm converts the frequencies of the optical frequency range into RGB values. In order to have frequency values of the optical frequency range for the RGB values of each pixel on the image that would be converted into music, the following procedure was applied:

Each colour value (r,g,b) in each pixel on the image that will be converted into music, should match to a colour value (r’,g’,b’) in the optical frequency range, which has resulted from the Bruton algorithm.

The match is successful by minimizing the function,

$$f\left(r',g',b'\right)=\sqrt{(r-r')^2+(g-g')^2+(b-b')^2} \quad (5)$$

which resembles the distance between them in the three-dimension area. With this way we calculate the closest (r’,g’,b’) value on the Euclidean space, with which we replace the original (r,g,b) value for the pixel on the image.

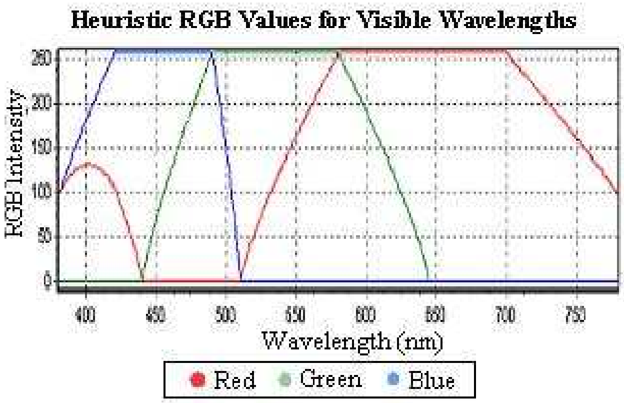

Fig 5: RGB values for the optical wavelengths from the application of the Bruton algorithm.

It is a fact that an image is a static piece of art, while a music composition is a piece that progresses in time. We scan the image line-by-line and every line pixel-by-pixel from left to right. In essence, this procedure of gathering the optical information via scanning the pixels of the image, as progression in time since it regards a time-series, is what we convert into a monophonic music composition; that is in a procedure in time of attribution of the sound information. Scanning the image every two or three etc. lines at the same time, according to the above procedure, results in two or three etc. time-series, which transformed into sound, construct a music composition with two or three melodic lines.

For practical reasons of memory and duration of the produced music composition, the aforementioned procedure does not apply to all the pixels of the image, but only to those that are selected at a step defined by the user on the horizontal and vertical axis.

Utilizing the above energies, see Fig. 6, we convert the RGB values for the color of each pixel of an image into frequencies of the optical frequency range.

Furthermore, having the RGB values for the color of each pixel (that correspond to frequencies of the optical frequency range) we re-design the image using these values.

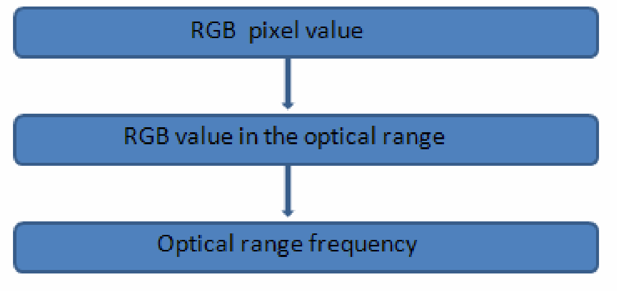

Fig 6: How we convert RGB values to optical-frequencies.

B. Optical to Acoustical Frequency Conversion

Between the set of the optical frequencies and the acoustic one there can not be an one-to-one mapping because these two sets contain a different number of elements.

We know that the ratio of two frequencies defines an interval, the magnitude (M) of which is proportional to the logarithm of this ratio [8]

$$\left[M = k\cdot\log_2\left(\frac{f_1}{f_2}\right),\ f_1 > f_2,\ k = constant\right] \quad (6)$$

The ratio of an interval (optical or acoustic) to the interval of the bound (limit) values (optical or acoustic) defines the relative interval (optical or acoustic) and the ratio of the corresponding magnitudes of these intervals defines the relative magnitude of that interval.

We can now define an one-to-one mapping between the relative magnitudes of the optical and acoustic “intervals”, by which we will successfully and objectively convert them from one form to the other. Thus, we go from the optical frequencies to the acoustic ones and vice versa. The above are depicted with the formula:

$$\frac{\log\left(\frac{f_{optical}}{f_{optical}^{\min}}\right)}{\log\left(\frac{f_{optical}^{\max}}{f_{optical}^{\min}}\right)} = \frac{\log\left(\frac{f_{acoustic}}{f_{acoustic}^{\min}}\right)}{\log\left(\frac{f_{acoustic}^{\max}}{f_{acoustic}^{\min}}\right)} \quad (7)$$

It must be noted that formula (7) is a result of Weber-Fechner’s psychophysics law, according to which the response is proportional to the logarithm of the cause. In other words, what we see and listen is proportional to the logarithm of the wave frequency causes (optical and acoustic respectively).

The defined acoustic range in music terms is between the frequency range of 8 octaves and specifically from 16.352 Hz to 7902.1 Hz (C_0 – B_8). The defined optical range is arguably their wavelengths from 380 nm to 780 nm.

Table I shows some typical colors, along with their RGB values and the sound frequency values in which they are transformed, when we apply our proposed procedure.

TABLE I.
TYPICAL COLORS AND THE CORRESPONDING SOUND FREQUENCIES

Color	R	G	B	Sound Frequency (Hz)
pink	255	192	203	828.740
orchid	218	112	214	29.294
violet	238	130	238	31.200
magenta	255	0	255	32.530
purple	128	0	128	20.311
blue	0	0	255	59.458
green	0	255	0	209.802
ivory	255	255	240	629.305
yellow	255	255	0	629.305
olive	128	128	0	341.833
brown	165	42	42	5525.350
red	255	0	0	1559.010
orange	255	128	0	1067.173
gold	255	215	0	749.580
cyan	0	255	255	149.082

C. Grouping of the Frequencies with the use of the Correlation Coefficient

Every point of the resulted time-series is an ordered pair of time – acoustic frequency. We shall call this time series pitch curve.

Aiming to conclude in a music melody, it is necessary to group in a way the resulting frequencies of each pitch curve in order to create notes of specific pitch and duration.

After that, we represent the pitch curve as a function in time; that is f(t).

We consider that the f(t) function is periodic; that its period (T) is equal to the duration of the time series from the moment of the scanned image.

The analysis of the function in Fourier series is of the form

$$f(t) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty}\left(a_n \cos n\omega_0 t + b_n \sin n\omega_0 t\right) \quad (8)$$

where $\omega_0 = \frac{2\pi}{T}$, we approached it with

$$S_k(t) = \frac{1}{2}a_0 + \sum_{n=1}^{k}\left(a_n \cos n\omega_0 t + b_n \sin n\omega_0 t\right) \quad (9)$$

We found out that for k=48 the error in the above approach $\varepsilon_k(t) = f(t) - S_k(t)$ is satisfactorily small.

After that we transformed (9) in a sequence of step functions by applying the following algorithm:

Step 1: (initial conditions). Find the first derivative of (9)

Step 2: Calculate the number of points of the first derivative curve, when approximated by a least-square straight line, gives a correlation coefficient r grater than a value set by the user. This value of the correlation coefficient r influences the duration of the notes, which will be resulted in the final melody as it will be explained later on. The correlation coefficient is given from the formula:

$$r = \frac{n\sum xy - \sum x \sum y}{\sqrt{[n\sum x^2 - \left(\sum x\right)^2][n\sum y^2 - \left(\sum y\right)^2]}} \quad (10)$$

where n is the number of points under examination for their linearity, x's are their abcissi; that is time, and y's are their ordinates; that is $S'_k(t)$ values.

The greater is r's value, the points fall very close to a straight line. The step goes to step 3 every time a point, let us suppose x_0, when calculated, is the reason to decrease the value of the correlation coefficient r. This means that the point x_0 does not lie on the same straight line with the rest n points, but from it starts a new straight line of the pitch curve.

Step 3: The ordinates of the n points that were calculated in Step 2 are replaced on the pitch curve by their mean values. The procedure continues by returning to Step 2 with a new initial point x_0. When the whole domain of the $S_k(t)$ runs out, the pitch curve (9), that is $S_k(t)$ has been replaced by a sequence of step functions.

One can understand now the role the value of the correlation coefficient plays; on one hand for the pitch and on the other for the duration of the notes in the resulted melody.

Since the correlation coefficient of a number of n adjacent frequency values, falls into the limits, set by the user, then these frequencies, grouped, give a pitch note equal to the mean value of those frequencies and of duration in sixteenths equal to the number of the grouped values.

D. Frequency Transformation into Tempered European Pitches and Polyphonic Music Generation

Next step is to transform the frequencies of the notes that resulted from each time-series, into frequencies of the tempered European scales (major/minor). This can be succeeded by replacing the frequency of each note with the closest tempered-frequency of the European scale. In this way a melody in European musical notation is produced.

The monophonic or polyphonic melody produced, is saved in a midi file and is possible to listen or edit it with any score processing software.

Our software has the following capabilities:

- Choice of different musical instruments for the application of each voice (melody)

- Choice of the ampitus of each musical instrument
- Choice of the musical measures
- Regulation of tempo
- Performing of the polyphonic melody and,
- Display of the notes from all the voices in a matrix.

E. Reversing the Process and Redesigning of the Image

From the resulting monophonic or polyphonic melody, we have a value of tempered frequency of the human acoustic frequency range for each pixel of the initial image. Applying the conversion formula (7), for every given acoustic frequency it gives its corresponding frequency of the optical frequency range.

The limits are for the optical frequencies those of the optical frequency range and for the acoustic ones those that were used for the conversion of the optical frequencies into acoustic ones. Applying the Bruton algorithm the corresponding RGB values are received and the initial image is redesigned, having the melody as the 'source'. The consequent image is more or less distorted, because not only the acoustic frequencies have been quantized according to the tempered European scale but also altered due to their grouping with the use of the correlation coefficient.

F. Resume

To summarize, the process our software executes for the conversion of an image to a music piece are outlined in Figure 7.

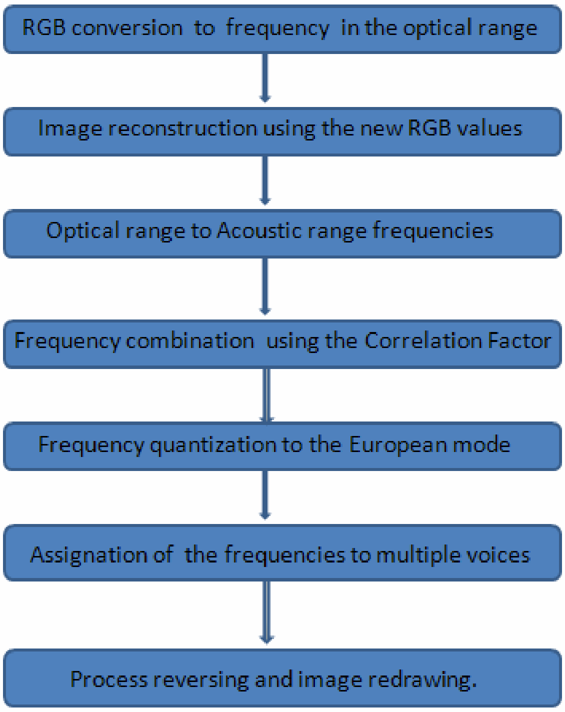
RGB conversion to frequency in the optical range
Image reconstruction using the new RGB values
Optical range to Acoustic range frequencies
Frequency combination using the Correlation Factor
Frequency quantization to the European mode
Assignation of the frequencies to multiple voices
Process reversing and image redrawing.

Fig 7: The flow-chart of our software.

III. SOUND TO IMAGE TRANSFORM

A. Melody introduction

The input of the musical information (pitch, duration) of each note of any new monophonic melody is carried out through a virtual 'clavier', identical to the piano one.

The user by clicking on one of the keys chooses the note that will be introduced. Before that, he should have selected its duration via a user friendly menu. The pitch and duration are added on a matrix, which can be edited via many options (add, copy, remove and edit). This matrix is saved in a file and from this file the time-series of a voice will be created. If we would like to compose a polyphonic piece, then we create such a file for every voice, out of which the time-series of the corresponding voice will be created.

B. Software Options

The software provides the capability to choose a plethora of voices, which we will use and introduce the relevant files we have created. We also select the time signature of the piece (e.g. 2/4,3/4 etc.) as well as whether the piece starts with an incomplete measure. Finally, we give the number of measures the duration of which sets the length of each line on the image to be designed.

C. Image Designing Process

The note replacement with their corresponding frequency and build of the time-series of each voice is achieved as follows:

Every note reproduces its frequency on the time series as many times as is its duration in sixteenths. For every point of the time-series of each voice we apply the conversion formula (7). In this manner, an optical frequency for every acoustic frequency occurs. With the Bruton algorithm and the aforementioned methodology, we receive RGB values, out of which every point of each time-series of the monophonic or polyphonic music is converted and we design the image line-by-line.

The height of each line is an input parameter of our program. The length of each line on the image is proportional to a definite number of measures of the melody, which is given as a parameter in the software, as it is mentioned. The sequence of the lines is definite and fixed. At the first line of the image the first voice is designed, at the second line the second voice and so forth until the number of voices ends. The next line of the image starts again with the first voice and so forth.

It must be noted that the pauses (rests) are designed with the white color. If an incomplete measure exists, it is ignored and is not designed at the beginning. The last measure is filled with the initial incomplete measure.

Thus, the final visual representation of a melody is produced.

Below we present two transformation examples of music composition to image. Figure 8 shows the 3-voice XVI Fuge from the piece DIE KUNST DER FUGE by J. S. Bach. From the smooth change in colors we can determine that the music piece uses sequences of small musical intervals with a few exceptions only.

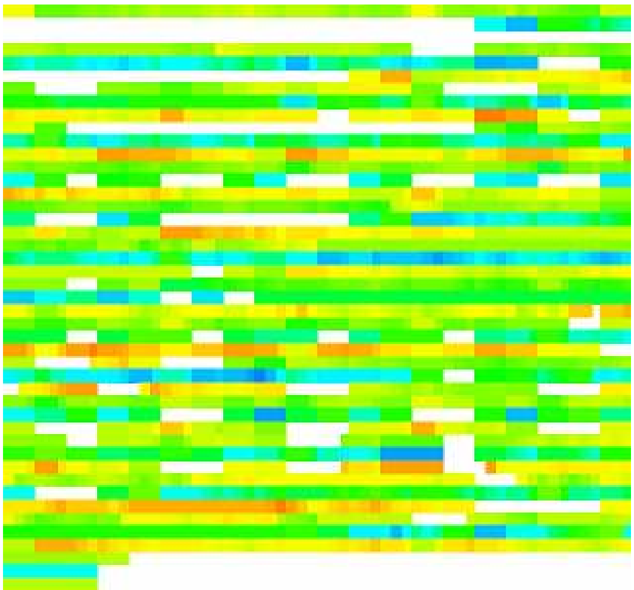

Fig 8: : Example 1 (Visualization of the three-voice fuge XVI from the piece DIE KUNST DER FUGE by J. S. Bach)

Examining the image, one could determine some fundamental morphological conclusions like the repetition of melodical and rhythmical sets. This study becomes more supervisory when on the image we point out the intervals (long lines) and the points where the note duration changes (short lines), as Fig. 9 shows.

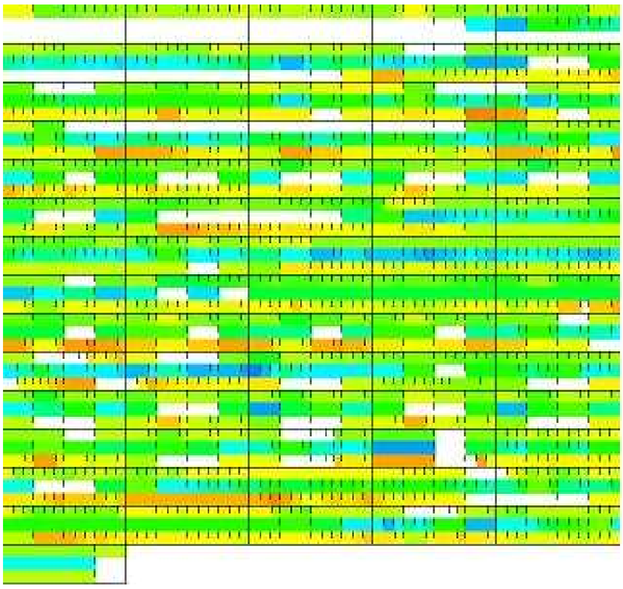

Fig 9: Example 2 (Visualization of the three-voice fuge XVI from the piece DIE KUNST DER FUGE by J. S. Bach with noticeable separation of the intervals and note duration)

In general, the longer the musical intervals are the more vibrant the color changes become, while the step-by-step (according to small musical intervals) movement of the voices is depicted by smooth color changes.

Fig.10 depicts the atonal composition «φεπ» by H. C. Spyridis, built on the same rhythmical sets with the above fuge by J. S. Bach. The melodic lines of this piece are based on long musical intervals, fact that is portrayed in the multitude and contrast of colors.

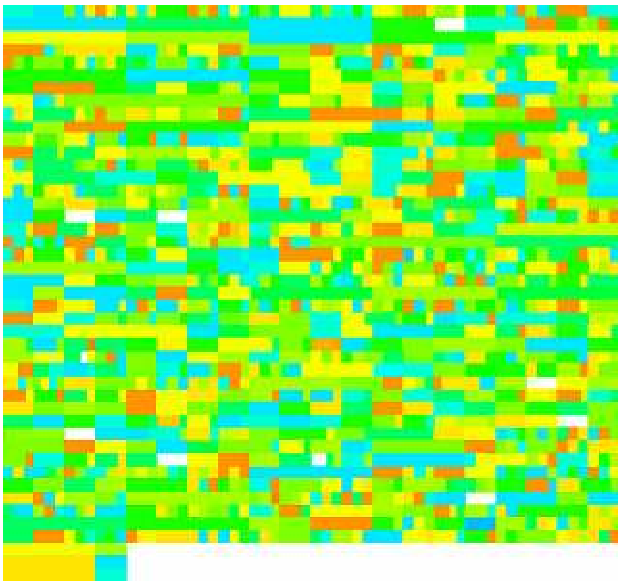

Fig 10: Example 3 (Visualization of the atonal composition «φεπ» by H. C. Spyridis)

IV. CONCLUSION

The presented software utilizing algorithms of converting acoustic frequencies into RGB color values, has the capability to produce melodies from images, as well as design images transforming melodies.

We are still concerned with the quest of an aesthetically better method for the mapping of the “monochromatic” frequencies of the optical frequency range into RGB values, as well as the improvement of its inverse process, with new algorithms [6].

To conclude, the restrictions of the RGB model lead to the research of better methods for the representation of the chromatic information [4, 7] in order to succeed in an improved precision for the conversion of paintings or photographs into music.

ACKNOWLEDGMENT

We would like to thank Mr. Petros Moustakas, Musicologist, for his aid, cooperation and consultancy to this software.

REFERENCES

[1] Garrett M. Johnson & Mark D. Fairchild, *Computer Synthesis of Spectroradiometric Images for Color Imaging Systems Analysis*, The Sixth Color Imaging Conference: Color Science, Systems, and Applications.

[2] Dan Bruton, *Color Science Web Page*, http://www.physics.sfasu.edu/astro/color.htm

[3] H. Spyridis, E. Roumeliotis, And H. Papadimitraki - Chlichlia, *A computer approach to the construction and analysis of a pitch-curve in music*, ACUSTICA, vol. 51, No 3, pp 180-182, (1982).

[4] G.M. Johnson and M.D. Fairchild, *Full-Spectral Color Calculations in Realistic Image Synthesis*, IEEE Computer Graphics & Applications, 19:4 47-53 (1999).

[5] R. Hall, *Illumination and Color in Computer-Generated Imagery*, Springer, Berlin, 1989.

[6] Smits B., *An RGB to Spectrum Conversion for Reflectances*, Journal of Graphics Tools, Vol. 4, No. 4, pp. 11-22, 1999.

[7] Greg Ward , Elena Eydelberg-Vileshin, *Picture perfect RGB rendering using spectral prefiltering and sharp color primaries*, Proceedings of the 13th Eurographics workshop on Rendering, June 26-28, 2002, Pisa, Italy.

[8] H. C. Spyridis, *Physical and Musical Acoustics*, Grapholine Publications, Thessaloniki 2005, pp. 412-415 (in Greek).

[9] Yoshi Ohno, *CIE Fundamentals for Color Measurements*, IS&T NIP16 Conference, Vancouver, Canada, Oct. 16-20, 2000.