Implementation of *Motion Without Movement* on Real 3D Objects

Lyudmil Vladimirov Antonov

under the direction of
Ramesh Raskar, Ph D
Mitsubishi Electric Research Laboratories

Research Science Institute
July 30, 2002

Abstract

Researchers have developed a technique such that when the colors of a static image are toggled on a screen, the illusion of continuous movement in a certain direction is produced. This phenomenon is known as *motion without movement*. In our research we aim to extend the applications of this technique and apply it to real three-dimensional objects.

In order to achieve the projection of images onto three-dimensional shapes, we use projectors called *shader lamps*, which apply an algorithm to a two-dimensional image so that it appears undistorted when projected onto a three-dimensional object. The resulting effect is that a static object appears to be moving continuously in a desired direction.

In addition to applying the *motion without movement* technique to entire objects, we also examine its use on parts of an image as small as a pixel. Using a technique called *optical flow*, we determine the exact movement oparts of an image by taking a second image similar to the original, where image shapes have moved and determine in which direction. Finally, we extend the motion without movement technique beyond its previous applications to use on color rather than solely grayscale images, thus producing even more realistic results.

1 Introduction

Numerous techniques exist that address the display of motion both on projected and computer-based images. However, most of them have drawbacks that limit their use. For example, if objects in a scene are set in a movie loop, at the end of the loop they must return to their starting position, or else a jerky display is observable as the movie replays. Recently a technique called *motion without movement*(MWM) was developed which does not have this limitation[1]. Instead of attempting to move the picture,the MWM technique is applied to a static image while still giving the impression of continuous motion. With this technique, the image shapes remain in the same position while creating the illusion of motion merely through toggling the colors of the boundaries of a shape, instead of actually moving them.

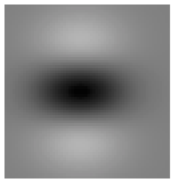
To implement the MWM technique on real, three-dimensional objects, we use devices called *shader lamps* together with a program that corrects the image distortion of the image as it is being projected. *Shader lamps* are described by Ramesh Raskar as "light projectors that render imagery directly in our real physical surroundings" [2]. In our research we extend the MWM technique by combining it with shader lamps, thus making it applicable to real, static 3D objects. In addition, we examine MWM use on parts of images being projected, by using *optical flow*, a technique by which we determine the exact direction of motion of image parts by taking a second image similar to the original, where the original image shapes have physically moved. We apply the so-determined motion to the original image shapes using MWM, thus creating the illusion that they are moving in the direction in which they actually moved in the second image. Finally, we extend the MWM technique beyond its previous applications to be used on color rather than solely grayscale images, thus producing even more realistic and astonishing results.

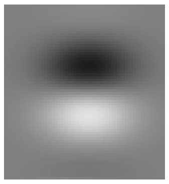
2 Motion Without Movement Technique

2.1 Outline

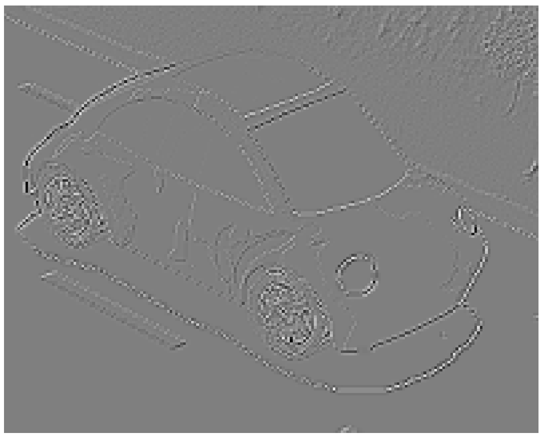
The process of applying the MWM technique consists of four stages:

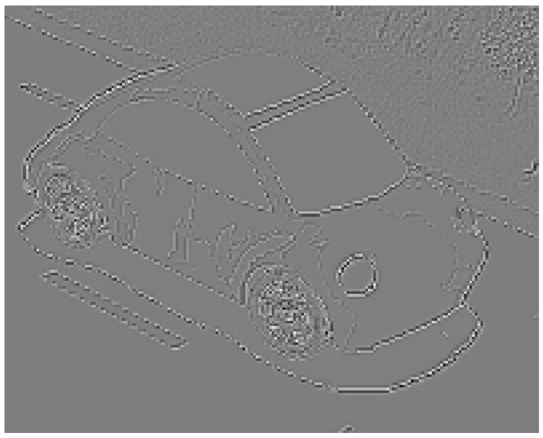
- Constructing specific image transformations, which we call *filters*(See Appendix A), such that when applied to an original image, it appears to move. We apply two transformations that we call *even*(a) and *odd*(b) filters(9 x 9 arrays):

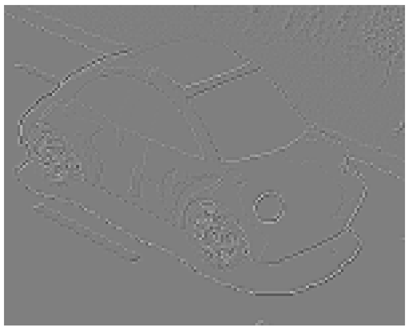

(a) Example of MWM even filter for upward motion

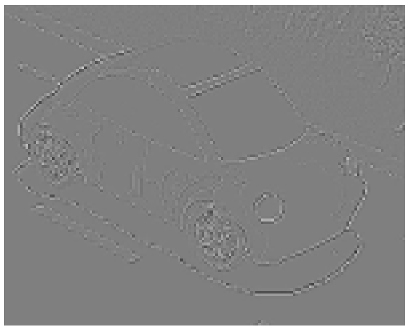

(b) Example of MWM odd filter for upward motion

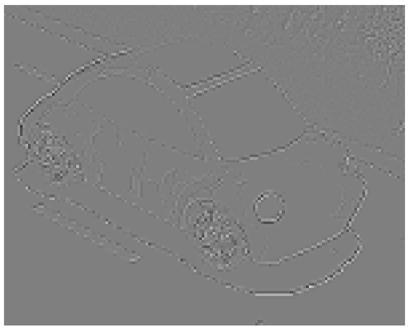
- Converting the original image into two new images of the same size, the "basic" for the construction of sequences of frames. These basic images are called *even*(c) and *odd*(d) images and are formed so that the edges of the shapes within them are marked black and white, respectively.


(c) MWM even image


(d) MWM odd image


(e)


(f)


(g)

(h)

(i)

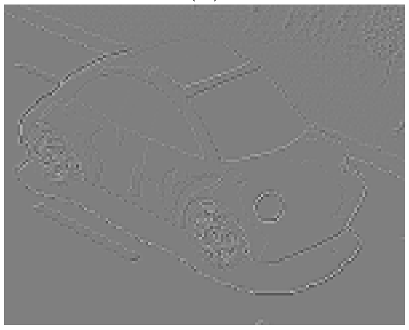
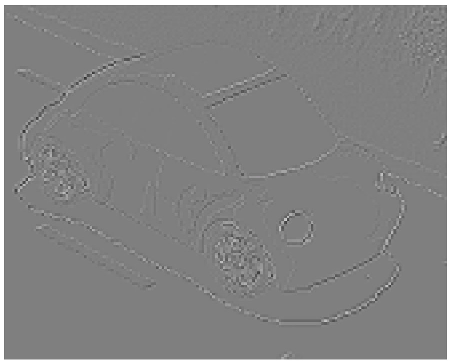
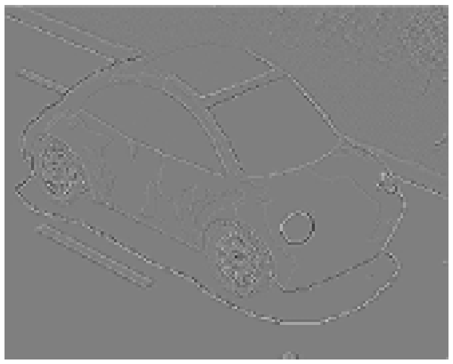

(j)

Table 1: Movie frames resulting from MWM technique

- Obtaining the sequence of frames representing a MWM illusion effect as a linear combination of the *even* and the *odd* images with non-linear coefficients (e-j).

- Restoring some of the lost details during the technique appliance. That technique is applied on every frame in the video frame sequence produced by MWM. An example of a frame and its detailed version is as follows:


(k) a normal video sequence frame


(l) a detailed video sequence frame

2.2 Motion Illusion

In order to achieve the desired effect of illusionary motion in a given direction, we make use of *filters* and an operation called a *convolution*(Appendix A), which allow us to detect the edges of image shapes and apply the color transformations over them.

We use the combination of two filters to achieve the perceived motion in fixed image shapes, as described by W. Freeman[1]. The first filter employed is the second derivative of the Gaussian - G_2 [1], and the second is the Hilbert transformation of G_2, H_2[?]. The two filters are considered identical, except that the second is phase-shifted 90 degrees as a result of the Hilbert transformation over G_2. Such pair of filters is defined as a *quadrature pair*, G_2 is said to be an even filter and H_2 an odd filter [1].

By convolving the original image, $I(x, y)$, with the filters separately, two images are produced called $E(x, y)$ and $O(x, y)$, by the even and odd filters respectively, used for their creation. We can therefore describe $E(x, y)$ and $O(x, y)$ as below:

$$E(x, y) = I(x, y) \otimes G_2$$

$$O(x, y) = I(x, y) \otimes H_2$$

The images created through this convolution (c and d) create the foundation for the application of our technique. The linear combination of the basic images E(x,y) and O(x,y) with non-linear coefficients to cycle through the images, produces series of frames(e-l). It is these images that, when set in a movie loop gradually transform one into other, producing the illusion of motion. They are created according to the following formula:

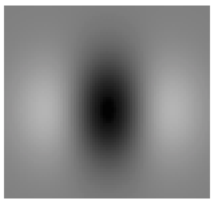
$$D_t(x, y) = cos(\omega_t)E(x, y) + sin(\omega_t)O(x, y)$$

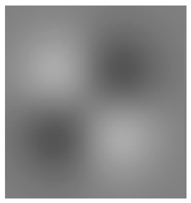
where $D_t(x, y)$ is a certain image in the sequential movie loop consisting of n images, t runs from 1 to n, and ω_t runs from 0 to 360 degrees with a desired step, which controls the number n of the images produced, thus controlling the smoothness of the technique. The number of frames, n, is connected with the step, q, through the following equation :

$$n = 360/q$$

However, the created this way image can only appear to move to the right. In order to extend this technique to movement in any thetial direction, we rotate our filters so that they are rotated by the desired angle θ, and then apply them over the image. For achieving this we use *steerable filters*[1]. The rotation is done through a linear combination of a fixed number of filters that are rotated in an exact direction, and which equations for creating are known. For the G_2 filter there are three, whereas for the H_2 filter they are four. Their equations are given as follows :

$$\begin{aligned} G_{2a} &= 0.9213(2x^2-1)e^{-(x^2+y^2)}(F_{g1}) \\ G_{2b} &= 1.834xye^{-(x^2+y^2)}(F_{g2}) \\ G_{2c} &= 0.9213(2y^2-1)e^{-(x^2+y^2)}(F_{g3}) \end{aligned} \tag{1}$$


G_a filter


G_b filter

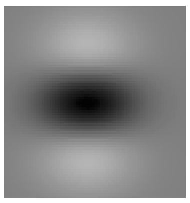
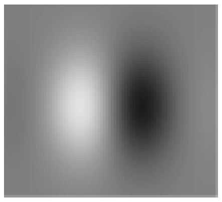
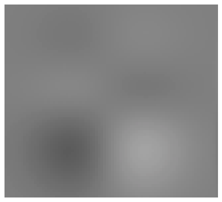
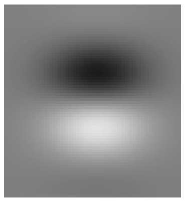

G_c filter

Table 2: Filters rotated by an exact angle. x and y are positions in the filter matrix

$$\begin{aligned} H_{2a} &= 0.9870(-2.254x + x^2)e^{-(x^2+y^2)}(F_{h1}) \\ H_{2b} &= 0.9870(-.7515 + x^2)e^{-(x^2+y^2)}(F_{h2}) \\ H_{2c} &= 0.9870(-.7515 + x^2)e^{-(x^2+y^2)}(F_{h3}) \\ H_{2d} &= 0.9870(-2.254y + y^2)e^{-(x^2+y^2)}(F_{h4}) \end{aligned} \tag{2}$$


H_a filter


H_b filter


H_c filter

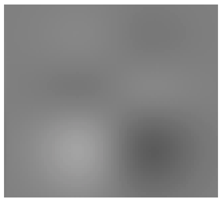

H_d filter

Table 3: Filters rotated by an exact angle. x and y are positions in the filter matrix.

The following dependence between the basic filters creates the rotation of the filter on an angle θ:

$$G_2^\theta = cos^2(\theta)G_{2a} - 2sin(\theta)cos(\theta)G_{2b} + sin^2(\theta)$$

$$H_2^\theta = cos^3(\theta) - 3cos^2(\theta)sin(\theta)H_{2c} + 3cos(\theta)sin^2(\theta)H2c - sin^3(\theta)H2d$$

When these filters are applied with specific angle θ on the two basic images, $E(x, y)$ and $O(x, y)$, and then the combinations between $E(x, y)$ and $O(x, y)$ are made, the result is a movie loop in which we have the original image $I(x, y)$ appearing to move in the desired direction specified by angle θ.

One resulting drawback is that details are lost during the application of the filter. Some of the details can be restored by adding $negativeI(x, y) \otimes G_2^{\theta(x,y)+\pi/2}$ to each frame, where $\theta(x, y)/2$ is the angle at which the filter is applied plus 90 degrees [1].

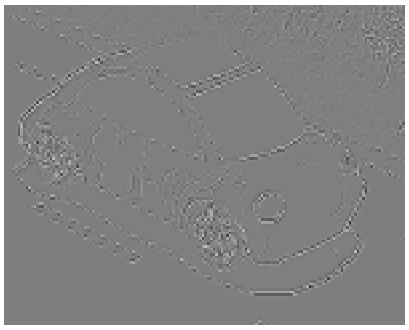
3 Extensions of motion

3.1 Applying multiple motion illusions on a single image

We extend MWM, applying it over different areas of the image so that individual shapes within the image can move in different directions. An example of this extension is an image of a car sitting on the road, which we want to appear to be traveling down the road. This cannot be done using the original technique, since the motion of the car should be forward, while that of the road and the background should be backward.

For achieving both of the motion illusions we apply individual steerable filters, specified by an angle θ, to every pixel of the image. This way we can control the direction on which the pixel appears to be moving. The following figure shows us the even and the odd image for that example.

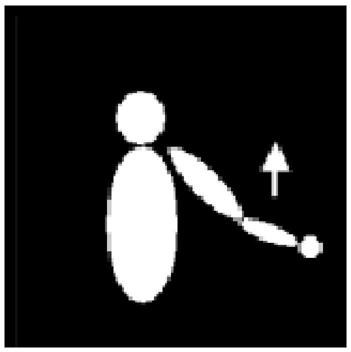
(c)MWM even image with two moving parts


(d)MWM odd image with two moving parts

The effect is reached by "moving" the rectangle in which the car fits, forward, while applying back used motion to the rest of the image. As it can be observed from the images, parts of the road and the background that fit into the car's rectangle appear to be moving in the same direction as the car. Because of this drawback we need to introduce a way to accurately determine the shapes we need to move in different directions.

3.2 Optical Flow

To accurately determine which shapes of an image should move, and in what direction, we use *optical flow*. This technique takes two input images and outputs a vector map showing how each pixel has moved from the first to the second image. The second image contains the same image shapes as the first one, but moved in a desired directions. The image can be considered as the next frame in part of a film loop where actual motion occurs. By having the vector map produced from optical flow between the two images, we calculate the angle, ω, on which every pixel has moved originating from the first image. An example of optical flow input and output is given by the following images:


Original image

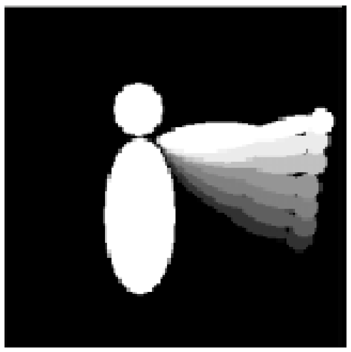
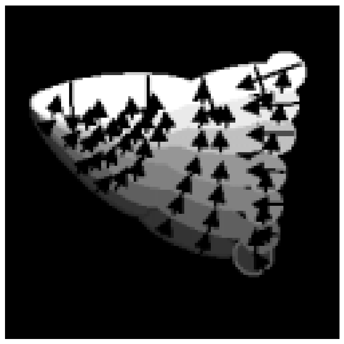


Image with moved shapes


resulting vector map

We can now apply the MWM technique with our partial movement extension and get a sequence of frames on which motion illusions in a different direction have been applied, but which this time have accurate motion directions.

In our research we use *shader lamps* for creating the static image and the one that has details changed.

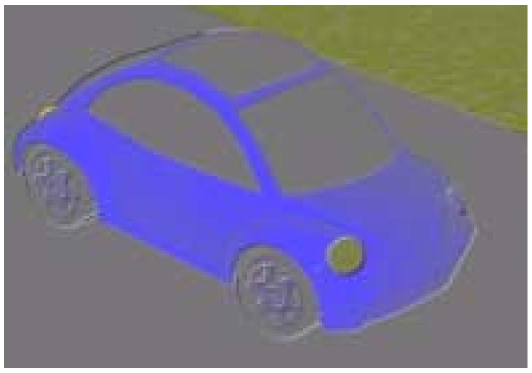
3.3 Color Appliance

The MWM technique was originally applied only to grayscale images. We implement this technique for color images as suggested by Freeman [1]. Our research utilizes two sets of image channels for representing color, RGB and YUV(See Appendix B).

We at first use YUV images since the technique is applied only on grayscale images, and the Y channel holds the needed information - the luminance of the image. For the actual visualization of the produced video frame sequence, we use the RGB channel color set since this is the visualization standard most devices are using, including shader lamps. Conversions between the channel sets are done before and after the MWM technique usage so that the correct channel set is used in each stage.

Color is implemented through several stages :

- Converting the original RGB image into a YUV image
- Creating the two basic images from the Y channel of the YUV image - $E(x, y)$ and $O(x, y)$
- Creating a sequence of frames that when played in a movie loop give us the illusion of motion
- Replacing the Y channel values of the original YUV image with the pixel values of the grayscaled movie frames created(Y holds the grayscale of the image), thus resulting in a color sequence of YUV images
- Converting each video frame to RGB channel set, so it can be displayed


(c)MWM even image (colored)

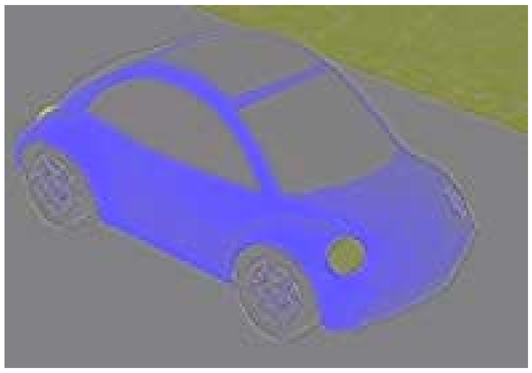
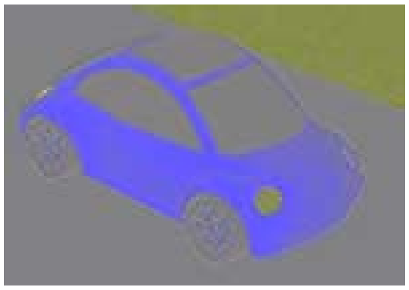
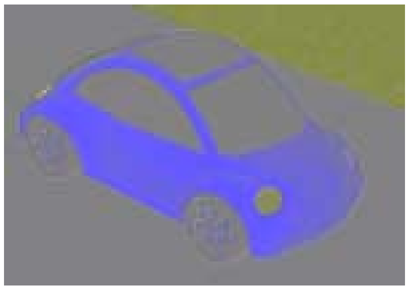
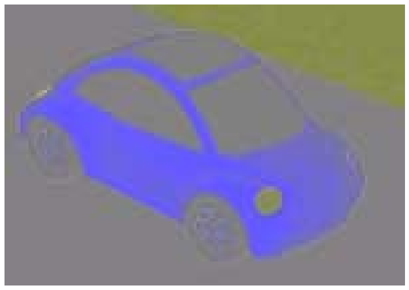
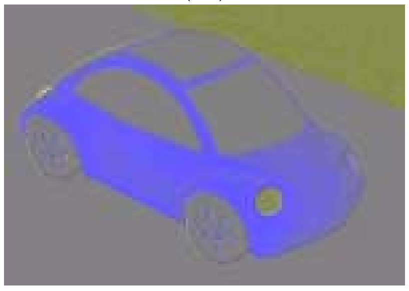

(d)MWM odd image (colored)

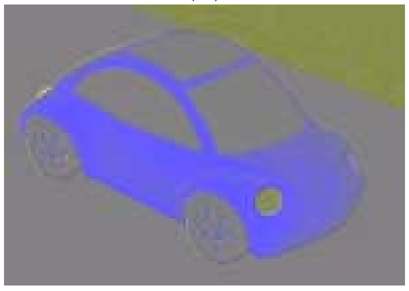
Table 4: although the basic images are grayscale, here they are colored to be able to see the effect of coloring better


(m)


(n)


(o)


(p)(


(q)

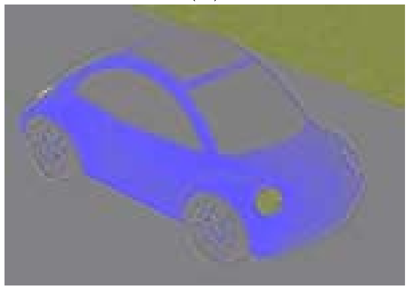

(r)

Table 5: Color movie frames resulting from MWM technique

3.4 Projecting on Real Objects

We make use of a computer application called *shader lamps*, which keep track of real tangible objects, and by utilizing a computer-run algorithm that corrects for distortion, projects image over them. This application is used for projecting a sequence of such images over an object, thus creating a movie loop.

In our research we used the *shader lamps* for creating the image that is to be projected. Then on this image we apply the *MWM* technique to produce the two basic grayscale images (even and odd). For extending the effect of the technique we take a second image from the shader lamps application, where the objects which we want to apply the motion, have moved. Then we use optical flow between the two images and apply the result of the combination between this technique and the MWM technique on the first image. After applying the result of this combination, we color the image as described in Section 3.3 .

The result is that we get a colored sequence of images on which we have the *motion without movement* technique applied. The images in this film sequence are actually just color intensity modifications of the first image captured from the *shader lamps*. When sequentially projected, these images give us the illusion of the scene objects moving in such way that they appear to flow into the second image. However, because these images are projected onto objects in our reality, the real objects give the illusion of motion but remain in a fixed position.

4 Results

In our research we have extended one interesting and useful technique for motion illusions, overcoming the limitations of existing computer-based graphics movement algorithms. We implemented a continuous motion display technique applied on real, three-dimensional objects and further improved it to produce a more realistic result. We managed to make parts of the image move in different directions by applying the technique to small shapes within the image. Based on this extension, we utilized optical flow for determining how each shape should in fact appear moving. Since our world is surrounded by color objects, the color is yet another important issue we examined. Though the color range available when using the technique is still limited, we were able to accurately portray objects of pastel colors. Coloring the object unifies it with our reality convincing us that the effect is not achieved artificially, but is a natural phenomena.

5 Conclusion

The motion without movement technique is a recently developed method where when applied to a static image, it gives in order to give the impression of continuous motion of the shapes within it. Previously, MWM was limited in its applications, because only one direction of movement could be applied to a single grayscale image. We developed modifications so that we can apply the technique to different parts of the image, thus producing different motion directions. We addressed the color limits, producing color images. Furthermore we project the image on real objects using computer application called shaderlamps, which results in applying the technique on objects instead of only on images. This technique has a wide variety of applications where motion is required, but no movement of actual pixels is allowed. For example, in the visualization of a focused car on the screen or as a 3D object, traveling down the road, the effect of "traveling" is achieved only by moving the background; the car stays in one place. Previously, MWM was limited in its applications because it could have been applied only on whole images, but not on parts of them.

An example for the application of MWM that has all of the mentioned technique extensions is a hand pointing to an "Eat at Joe's" sign. If we can apply the technique to the hand, which is a real color object, alone, it will appear that it is pointing to the sign and at the same time moving toward it, thus drawing people to eat at Joe's.

There are several future areas for inspection. One of them is to optimize the technique, such that one may apply it quickly to any desired object. We also hope to minimize or even prevent any color loss. Once these are achieved, we hope to make MWM the preferred technique in motion illusion.

6 Acknowledgments

I would like to thank my mentor, Ramesh Raskar, for his patient guidance and support throughout this project. The wonderful environment to which he introduced me helped me a lot for learning and experimenting. Though I had everything I needed for completing my project, he was always by my side, helping me achieve the results I was pursuing.

I would like to thank Kendrick and Paul, for reading and evaluating this paper, and for their support during the process of writing it. Their honest feedback greatly improved the content and grammar of the paper.

I especially want to thank Jenny Sendova for her overall support during the Research Science Institute program, as well as her help in the paper writing process.

I would also like to especially thank Julie Hanover for her guidance in editing the paper. Her valuable suggestions made the paper much more understandable and pleasurable to read.

Me gratitude is extended to Liz Wood, Jason Chiu, and Miro Jorisic for providing their kind help during the paper editing.

Thanks also to the Cyril and Methodius Foundation for funding me for this trip to the United States and to Jenny Sendova, Nikolai Dushkov, Neli Dimirova, Oleg Mushkarov, and all of the jury that chose me for attending the Research Science Institute.

Last but by no means least is my gratitude to the Center of Excellence in Education for providing me with the opportunity to participate in the RSI research program. Thanks especially to Dr. John Dell, Director of RSI, and Mr. Matt Cain, Assistant Director, for their overall support.

References

[1] FreeMan,William T.: *Motion Without Movement*, July 1991.

[2] Raskar, Ramesh: *Shader Lamps: Animated Real Objects With Image-Based Iillumination*, July 2000.

[3] http://www.webartz.com/fourcc/fccyvrgb.htm. http://www.webartz.com/fourcc/fccyvrgb.htm

[4] Inell Image Processing Library and Open CV

[5] http://www.dai.ed.ac.uk/HIPR2/convolve.htm.

A Convolution

Convolution is a mathematical operation, that as described in [5] provides a way for multiplying together two arrays that are of the same dimensions (i.e 1D, 2D, 3D), but can differ in size. The result of this operation is a third array of numbers that has the same dimensionality as the ones from which it was produced and the size of the bigger of them.

When convolution is used in the context of image processing, one of the input arrays is usually the pixels of a two dimensional grayscale image, whereas the other also is a two-dimensional array, the *kernel*, but which in this paper we will call *filter*. An example of a matrix filter is :

$$F = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

whereas we represent the pixels of the grayscales image as:

$$I = \begin{pmatrix} I_{1,1} & \dots & I_{M,1} \\ I_{2,1} & \dots & I_{M,2} \\ \vdots & \ddots & \vdots \\ I_{N,1} & \dots & I_{M,N} \end{pmatrix}$$

The actual operation is performed by sequentially placing the center of the filter (eg. position 2,2 the filter F above) over every pixel of the image. For each position where the filter overlaps I, an output value is calculated by multiplying each cell value with their underlying pixel, and summing the results. If the filter is applied over a pixel where some of the filter values are left without underlying pixels, the resulting sum is taken only from the values that have underlying pixels. The output is then stored as a pixel in the same position as the one on which filter was applied, but in a different image.

The result when applying the filter F over pixel in position $[p, q]$, $I_{p,q}$, is produced by the following equation:

$$
\begin{aligned}
O_{p,q} \quad = \quad & F_{1,1}I_{p-1,q-1} + F_{2,1}I_{p,q-1} + F_{3,1}I_{p+1,q-1} + \\
& F_{1,2}I_{p-1,q} + F_{2,2}I_{p,q} + F_{3,2}I_{p+1,q} + \\
& F_{1,3}I_{p-1,q+1} + F_{2,3}I_{p,q+1} + F_{3,3}I_{p+1,q+1}
\end{aligned} \tag{3}
$$

where O are the pixels for the output image.

The symbol used for convolution is $\otimes$ and generally the convolving image of size M x N with filter m x n can be represented with the following formula :

$$O(i,j) = \sum_{k=1}^{m} \sum_{i=1}^{n} I_1(i+k-1, j+l-1) \otimes F(k,l)$$

where i runs from 1 to M and j runs from 1 to N. The resulting image O with dimensions M by N is said to be convolved with the filter F :

$$O(M,N) = I(M,N) \otimes F$$

B Color Channel Sets - YUV and RGB

A computer-based image consists of pixels whose colors are determined by a combination of image color channels. Although the channels typically used are Red, Green and Blue (RGB), there are also other sets of channels that can be employed to produce pixel color.

A channel set called YUV is an example of a different set of channels that can be used for color representation. The Y channel of the set holds the luminance of the image (i.e, grayscale). The U and V channels store the image chrominance for red and blue, respectively. The transformation between this channel set and RGB can be performed according to the following table :

Conversion from RGB to YUV	Conversion from YUV to RGB
$Y = 0.29 * R + 0.58 * G + 0.11 * B$	$R = Y + (1.40 * (V - 128))$
$U = -0.16 * R - 0.33 * G + 0.50 * B + 128$	$G = Y - (0.34 * (U - 128) - (0.71 * (V - 128))$
$V = 0.50 * R - 0.41 * G - 0.08 * B + 128$	$B = Y + (1.77 * (U - 128)$

Table 6: Transformation between RGB and YUV set of channels [3]