

CARESSING SOUND AND IMAGE



By: Saehee Sarah Min
Student #: 47148986
Supervisor: Dr. S.Fels
Project Code Number: SSF3

#207-5642 Dalhousie Rd.
Vancouver, BC V6T1W4

April 5, 2002

Dr. Fels
Department of Electrical and Computer Engineering
University of British Columbia
2356 Main Mall
Vancouver, BC
V6T 1Z4

Dear Dr. Fels:

SUBJECT: Submission of EECE 496 Final Report

For the EECE 496 final report assignment, I have prepared the enclosed report titled, "Caressing Sound and Image."

The focus of this report is to investigate the creation of an interface to study the semantics of touch and caress. The comparison of various interpolation methods used to recover the image signal will be highlighted as well as some background needed to develop the interface.

I am certain that this report will meet the requirements of the final report assignment. If further information is required, please contact me by email at smin@ece.ubc.ca.

Sincerely,

Sarah Min

Encl: Final Report, "Caressing Sound and Image"

#207 5642 Dalhousie Rd.
Vancouver, BC V5R 3V8

April 5, 2002

Jane Pavelich
Department of Electrical and Computer Engineering
University of British Columbia
2356 Main Mall
Vancouver, BC
V6T 1Z4

Dear Ms. Pavelich:

SUBJECT: Submission of EECE 496 Final Report

For the EECE 496 final report assignment, I have prepared the enclosed report titled, "Caressing Sound and Image."

The focus of this report is to investigate the creation of an interface to study the semantics of touch and caress. The comparison of various interpolation methods used to recover the image signal will be highlighted as well as some background needed to develop the interface.

I am certain that this report will meet the requirements of the final report assignment. If further information is required, please contact me by email at smin@ece.ubc.ca.

Sincerely,

Sarah Min

Encl: Final Report, "Caressing Sound and Image"

Caressing Image and Sound

By: Sarah Min, 47148986

**Submitted to:
Dr. S. Fels & Ms. J. Pavelich**

**April 5, 2002
Electrical and Computer Engineering 496
University of British Columbia**

ABSTRACT

“Caressing Image and Sound”

By: Sarah Min

The MTC Express Multi-Touch Controller is a newly developed device that measures multipoint pressures. This device allows a user to interact with various computer-simulated virtual surfaces. A well-applied sampling theorem and image reconstruction method such as interpolation can recover the actual shape of an object on the controller. Moreover, this technology can be expanded to allow for the exploration of touch and caressing of various objects. One interpolation method that can be used to recover an image signal is a sinc interpolation. This interpolation is widely used for image processing because it can represent an object smoothly due to a curvilinear sinc signal. There are two ways to construct the sinc interpolator: using Pythagoras Theorem and using pre-interpolated data points. Both methods can recover an image smoothly, but the sinc interpolation using pre-interpolated data points creates a smoother and more detailed image than the interpolation using the Pythagoras Theorem.

TABLE OF CONTENTS

Abstract	ii
List of Figures	iv
Glossary	v
List of Abbreviations	vi
1.0 Introduction.....	1
2.0 Real Interface Hardware and Signals.....	2
2.1 MTC Express Multi-Touch Controller	2
2.2 Procedure of Transmitting and Receiving Data	3
2.3 Data Mode.....	5
2.3.1 Uncompressed Data Protocol.....	5
2.3.2 Compressed Data Protocol.....	5
3.0 Reconstruction of a Discrete Digital Signal.....	8
3.1 Sampling Theorem.....	8
3.2 Interpolation.....	9
3.2.1 Linear Interpolation.....	9
3.2.2 Planar Interpolation.....	10
3.3.3 Sinc Interpolation.....	11
4.0 Virtual Interface Software Development	12
4.1 Virtual Interfaces with Various Interpolators	12
4.1.1 Sinc Interpolator Using Taxel Data with Pythagoras Theorem.....	13
4.1.2 Sinc Interpolator Using Two Pre-Interpolated Data	14
4.2 Graphical Analysis on Interpolation Methods	15
5.0 Conclusion	22
Appendix A – C Programming Code for Sinc Interpolation with Pythagoras Theorem	
Appendix B – C Programming Code for Sinc Interpolation with Pre-Interpolated Data	
Appendix C – Graphical Representation in Sphere Mode	
Appendix D – Graphical Representation in Cube Mode	
Appendix E – C Programming Code with OpenGL for Sphere Graphic Mode	
Bibliography	

LIST OF FIGURES

Figure 1 MTC Express Multi-touch Controller	3
Figure 2 MTC Express Functional Procedure	4
Figure 3 Compressed Data Protocol Packet Structure	6
Figure 4 Pressure Representations in Bubbles with Uncompressed Data Protocol.....	6
Figure 5 Pressure Representations in Bubbles with Compressed Data Protocol.....	7
Figure 6 Proper Sampling Rate	8
Figure 7 Rebuilt Image with Proper Sampling Rate	8
Figure 8 Sampling Rate with MTC Express Controller.....	9
Figure 9 Rebuilt Image with Under-sampling Rate	9
Figure 10 Reconstructed Signal Using Linear Interpolation	10
Figure 11 Planar Interpolations.....	10
Figure 12 Sinc Signal.....	11
Figure 13 Reconstructed Signal Using Sinc Interpolations	11
Figure 14 Layout of Interpolated Data.....	12
Figure 15 Sinc Interpolation Using Pythagoras Theorem.....	13
Figure 16 Sinc Interpolation Using pre-interpolated Data.....	14
Figure 17 Front View of Pressures Applied on the Pad with Linear Interpolations	16
Figure 18 Front View of Pressures Applied on the Pad with Planar Interpolations	16
Figure 19 Side View of Pressures Applied on the Pad with Linear Interpolations	17

GLOSSARY

- OpenGL : Computer programming environment for 2 and 3 dimensional graphics application.
- Linux : Computer working environment.
- Byte : A unit of data composed of 8 binary bits.
- Packet : A group of bytes carrying a part of information.
- Undersampling : A sampling rate that is not high enough to recover an original signal.
- Interpolation : A method used in DSP to fill out the missing data.
- Pythagoras theorem : A mathematical theory: $a^2 + b^2 = c^2$, where c is the hypotenuse, and a and b are the lengths of the two shorter sides of the triangle.

LIST OF ABBREVIATIONS

- DSP : Digital Signal Processing
- API : Application Program Interface
- PC : Personal Computer

1.0 INTRODUCTION

This report investigates the creation of an interface to study the semantics of touch and caress. Creating an interface between a flat touch pad and computer simulated virtual surface and matter visually helps to understand the movement of object. The interface to be created will involve understanding discrete digital sampling theories, the mathematics involved to model the behavior of the real and virtual interface and programming involving the C and OpenGL* language. This is a significant project because the technology developed can eventually be expanded to allow for the exploration of touch and caressing of various objects.

The need for this project stemmed from an interest in virtual reality and interfacing real movements with virtual surfaces and matter. Throughout the project, the work involved will include all of the above-mentioned topics that need to be understood to accomplish the various tasks. However, the majority of the work will involve programming the software for the virtual interface in C on a Linux* machine.

The scope of this report will cover the development of the interface including the software development but will not cover direct references to programming code for the compression data protocol due to the matter of confidentiality. Necessary theory that is not common knowledge to an electrical engineer will also be covered. This report will be divided into the following sections, Real Interface Hardware and Signals, Reconstruction of a Discrete Digital Signal, Virtual Interface Software Development and a Conclusion.

2.0 REAL INTERFACE HARDWARE AND SIGNALS

2.1 MTC EXPRESS MULTI-TOUCH CONTROLLER

The MTC Express Multi-Touch Controller from Tactex Controls is a mouse-pad sized device, which provides a user interface between a real world and a computer simulated virtual world. It has an array of seventy-two sensing elements called taxels, which measure multipoint pressures. Once the pressure is collected from each taxel, the MTC Express Multi-touch Controller transmits the data to a personal computer (PC). Different inputs from the hardware interface can be used to create and develop various virtual surfaces.

The first way to determine the location of a taxel is to set the first column on the left to correspond to $x = 1$ and the top row to correspond to $y = 1$. Therefore, the row-column coordinates start at (1, 1) on the top left corner and end at (7, 12) on the bottom right corner. The following figure (Figure 1) illustrates the row-column coordinates frame of the pad as well as its configuration.

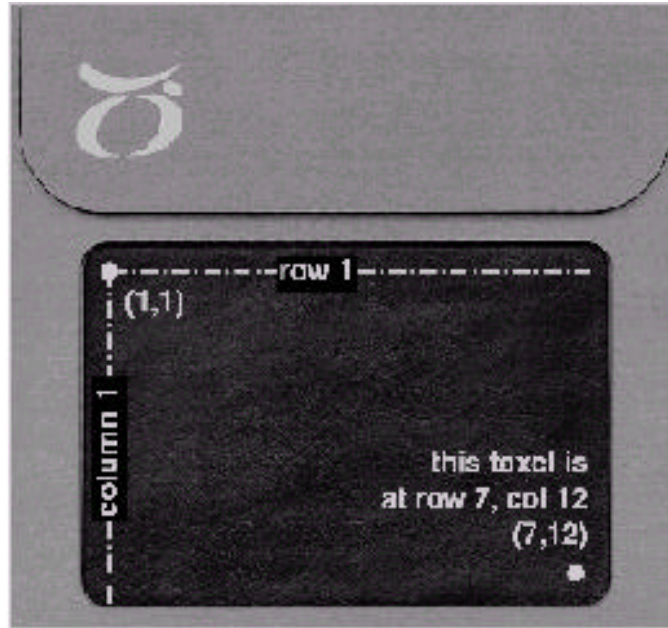


Figure 1 MTC Express Multi-touch Controllerⁱⁱ

2.2 PROCEDURE OF TRANSMITTING AND RECEIVING DATA

In order to transmit and receive data from the MTC Express Multi-Touch Controller, several procedures have to be performed. First, a connection between the PC and the controller is established through the RS232 serial port. Then the Application Program Interface (API) from Tactex provides a means to calibrate, or normalize the pad. This process is executed to prevent reading mistaken data due to the nature of the MTC Express Smart Fabric surface. Without calibration, there may be touch response errors, such as different response of different touches under the same pressure. After normalizing, the controller gathers the data, transmits it to the computer and keeps iterating this procedure until it receives a stop signal from the PC. As soon as the controller receives the stop signal, it terminates the normalization and disconnects with the computer. The functional procedures are also illustrated in the following flow chart.

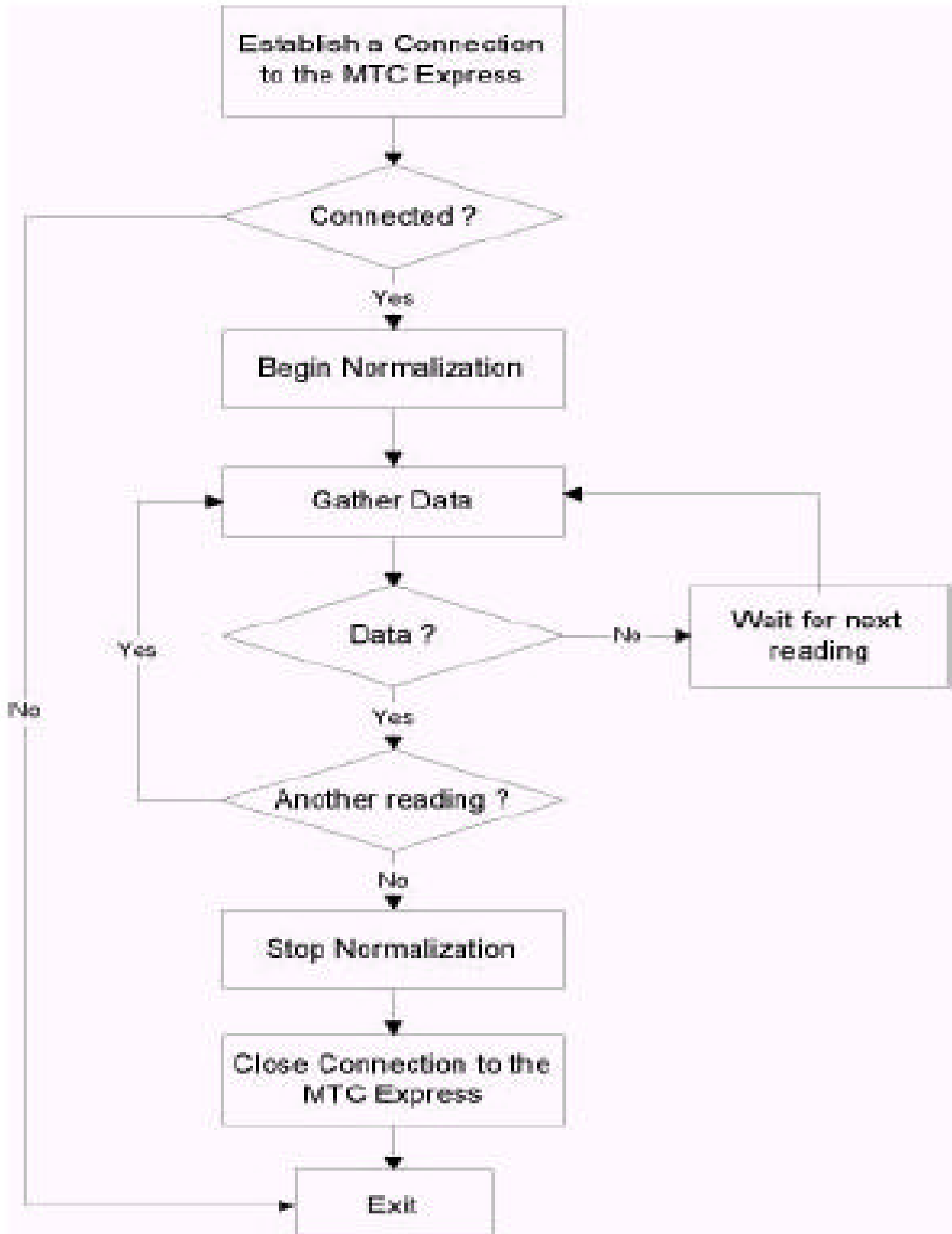


Figure 2 MTC Express Functional Procedureⁱⁱⁱ

2.3 DATA MODE

The MTC Express Multi-Touch Controller can retrieve pressure data from taxels in two different modes: uncompressed data protocol and compressed data protocol. The type of data mode to use can be chosen by a user command to the controller.

2.3.1 UNCOMPRESSED DATA PROTOCOL

Uncompressed data protocol uses a fixed-length packet* every sample interval. Each packet is comprised of a set number of bytes*, and each byte represents the pressure for each taxel. Comparison with the compressed data protocol will be addressed in the following section.

2.3.2 COMPRESSED DATA PROTOCOL

Compressed data protocol uses a variable-length packet every sample interval. Since one byte in the packet can represent more than one taxel, the packet is guaranteed to be smaller than when the byte represents one taxel, and therefore this protocol is faster than the uncompressed data protocol that was one byte per taxel.

Figure 3 illustrates the compressed data protocol structure. A typical packet is composed of a fixed number of data sub-packets, and each sub-packet has a variable number of bytes depending on the number of active taxels. A mapping byte is present in the beginning of a sub-packet to indicate the number of active taxels and their positions in the following bytes within the sub-packet.

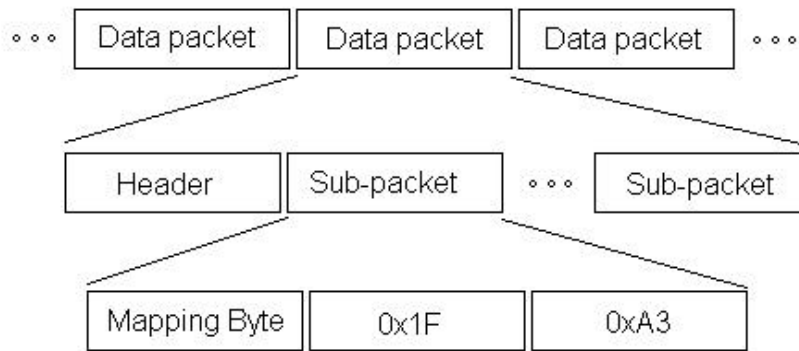


Figure 3 Compressed Data Protocol Packet Structure

The structure of the compressed data protocol indicates that the maximum size of a packet can be four times smaller than that of the uncompressed data protocol packet. Therefore, the sending rate for the compressed mode can also be four times faster than the uncompressed rate. However, as Figure 4 and 5 illustrate, the uncompressed mode has a higher resolution which means the range of pressure that it can represent is wider than the pressure range of the compressed data protocol.

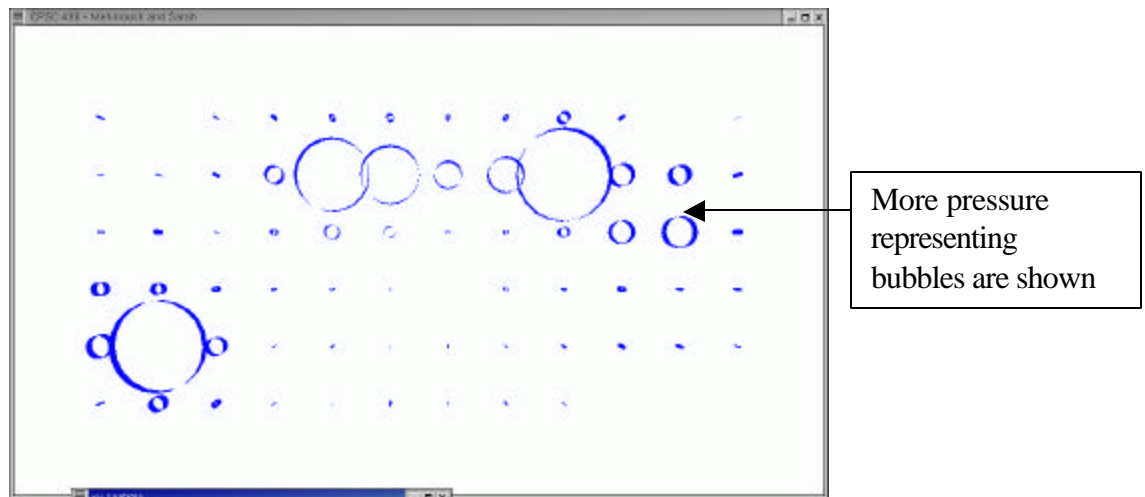


Figure 4 Pressure Representations in Bubbles with Uncompressed Data Protocol

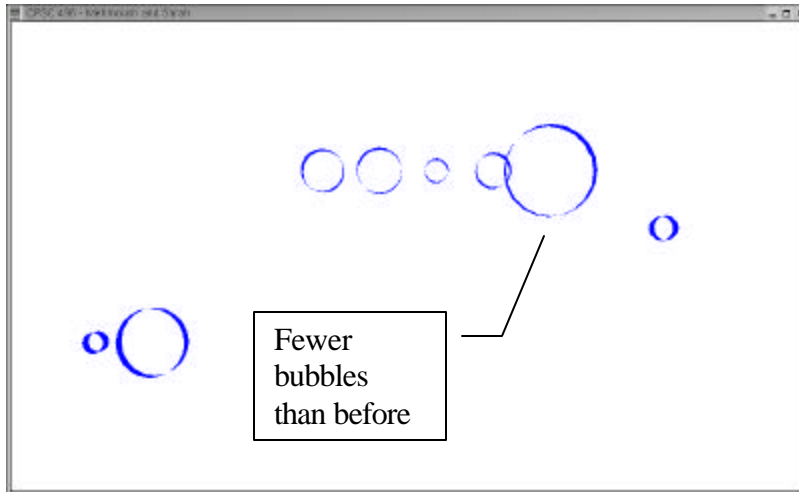


Figure 5 Pressure Representations in Bubbles with Compressed Data Protocol

3.0 RECONSTRUCTION OF A DISCRETE DIGITAL SIGNAL

The data signal from the hardware interface is only useful when the actual image can be reconstructed from it. In this section, necessary sampling theorem and image reconstruction methods behind the implementation of the computer-simulated virtual interface are covered.

3.1 SAMPLING THEOREM

The creation of a virtual reality, including the shape of a hand on the pad, requires enough data to reconstruct the exact signal or image as it was before sampling. The Sampling Theorem, Shannon or Nyquist theorem, states that a real signal can be completely recovered from its samples if, and only if, the sampling rate is at least twice the bandwidth of the signal. Therefore, in order to recover the shape of hand on the pad, it is required to collect data from at least two or more points for every finger, as seen in Figure 6. The recovered image of fingertips with the correct sampling rate is shown in Figure 7

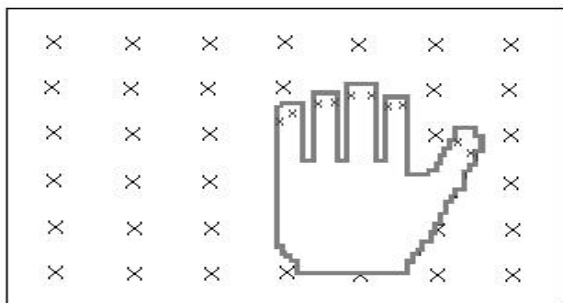


Figure 6 Proper Sampling Rate

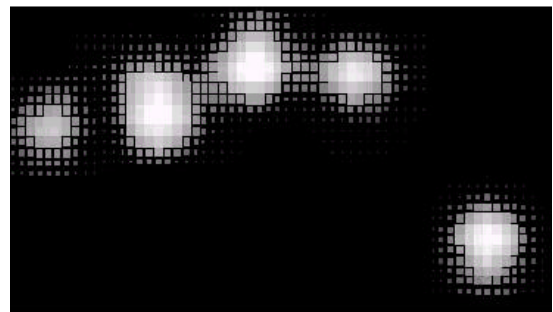


Figure 7 Rebuilt Image with Proper Sampling Rate

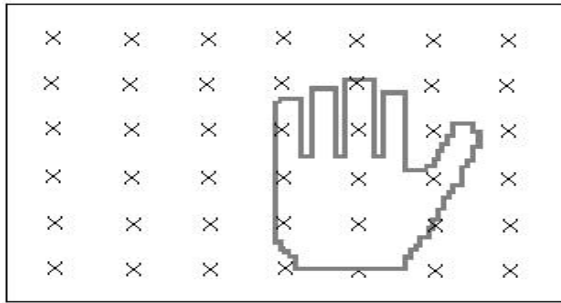


Figure 8 Sampling Rate with MTC Express Controller

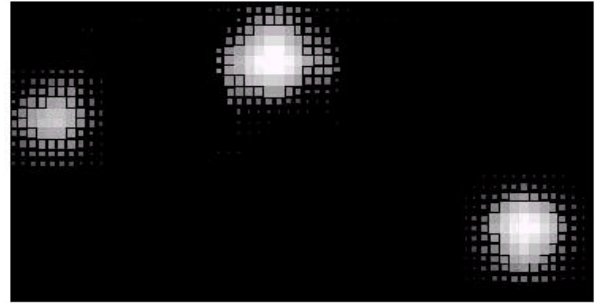


Figure 9 Rebuilt Image with Under-sampling Rate

However, because the taxels on the MTC Express Multi-Touch Controller are positioned every 1.5cm (Figure 8), it undersamples* narrow objects such as fingers. Therefore, if a user places his/her finger in between taxels, the reconstruction of the figure at the position will be ignored (Figure 8). In addition, it is hard to distinguish between one object covering two taxels and two small objects touching each other and also covering two taxels. The reconstructed image can be either broader than the actual size or can be represented as one blob.

3.2 INTERPOLATION

Interpolation* is one of the methods that recover the image by computing the missing points from neighboring taxel data. Even though this method cannot reconstruct the image exactly as the actual image, it helps to create an image similar to the original. The interpolation methods commonly used are linear interpolation, planar interpolation, and sinc interpolation.

3.2.1 LINEAR INTERPOLATION

Linear interpolation is the most basic interpolation method to reconstruct an image or signal. While other interpolations interpolate calculated numbers between two sampled

data, this interpolator only connects two points with a straight line. As seen in the following illustration (Figure 10), the line in between the points is not curvy and can cause a faulty recovered signal because most object have a curvilinear shape and is not flat like the line represents.

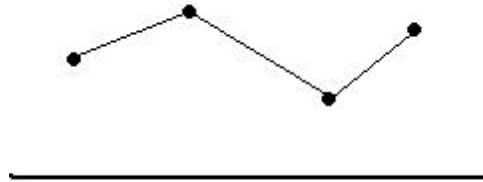


Figure 10 Reconstructed Signal Using Linear Interpolation

3.2.2 PLANAR INTERPOLATION

The planar interpolation is done by connecting one taxel point to three different neighboring taxel points (Figure 11), while linear interpolation only connects two adjacent taxel points. This method can be applied only when the signal is two or three-dimensional. The reconstructed image by planar interpolation is more accurate than the image by linear interpolations because it is recovered in relation with three adjacent taxel data points.

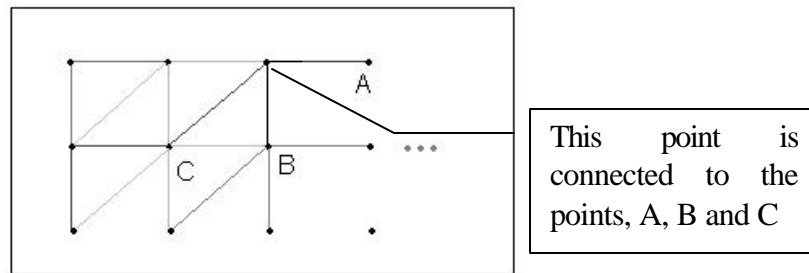


Figure 11 Planar Interpolations

3.3.3 SINC INTERPOLATION

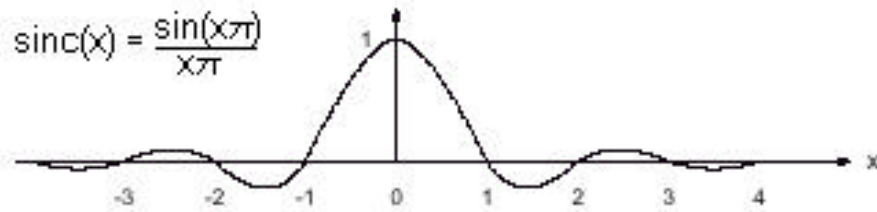


Figure 12 Sinc Signal

A sinc interpolator can be implemented by summing the calculated values from sinc equations multiplied by neighboring sampled data. Figure 13 illustrates how to reconstruct a signal using the sinc interpolator. The square point in between the sampled data A and C can be calculated from the following equation (Equation 1):

$$B = A \times \frac{\sin c(\mathbf{p}\frac{1}{2})}{\mathbf{p}\frac{1}{2}} + C \times \frac{\sin c(-\mathbf{p}\frac{1}{2})}{-\mathbf{p}\frac{1}{2}}$$

Equation 1 Sinc Interpolator Calculation

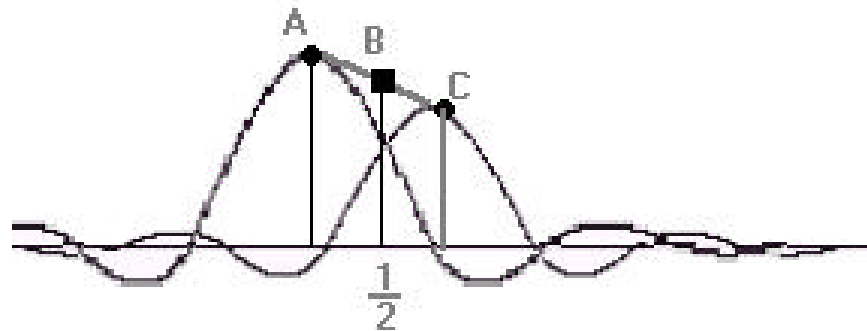


Figure 13 Reconstructed Signal Using Sinc Interpolations

4.0 VIRTUAL INTERFACE SOFTWARE DEVELOPMENT

The virtual interfaces with various interpolators, including linear interpolator, planar interpolator and sinc interpolator, are implemented to compare its efficiency. In this section, the results of these various interpolators are evaluated and discussed in detail.

4.1 VIRTUAL INTERFACES WITH VARIOUS INTERPOLATORS

In order to create a computer-simulated interface, sinc interpolators are used to reconstruct the image signal as mentioned in the section 3. Although the best results are achieved when the number of interpolated points is infinite, only two interpolated points between two taxel data points are used to achieve simplicity. Figure 15 illustrates the configuration of the taxel points and the pre-set interpolated points.

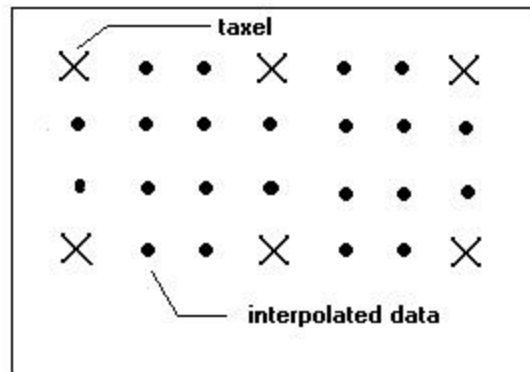


Figure 14 Layout of Interpolated Data

Note that the sinc interpolator implemented is based on theory covered in the previous section, but the signal reconstructed should be three dimensional; so the sinc interpolators are created in two different ways: using taxel data with Pythagoras Theorem* and using two pre-interpolated data points.

4.1.1 SINC INTERPOLATOR USING TAXEL DATA WITH PYTHAGORAS

THEOREM

The first method to implement a sinc interpolator is using only two adjacent taxel data points. Illustrated in Figure 16, the points in parallel with two taxel data points can be easily calculated, using the ratio of distance from the each taxel point (equation 1). However, in order to determine the relationships between the taxel points and the four interpolated points in the middle, Pythagoras Theorem is used so that those points can be equally related in distance to the other points.

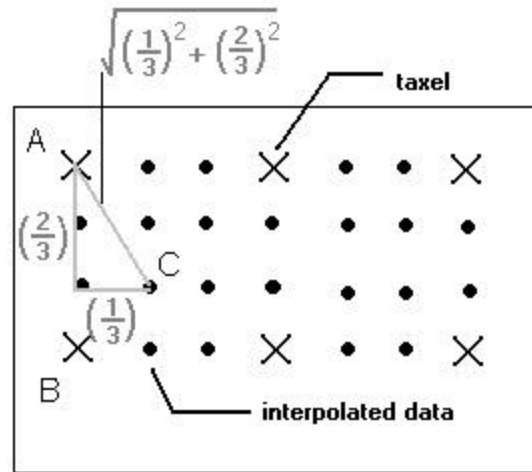


Figure 15 Sinc Interpolation Using Pythagoras Theorem

The C code to implement this interpolator can be found in the Appendix A. The equation for the points in the middle is (Equation 2):

$$C = A \times \frac{\sin c\left(\mathbf{p} \times \sqrt{\left(\frac{1}{3}\right)^2 + \left(\frac{2}{3}\right)^2}\right)}{\mathbf{p} \times \sqrt{\left(\frac{1}{3}\right)^2 + \left(\frac{2}{3}\right)^2}} + B \times \frac{\sin c\left(\mathbf{p} \times \sqrt{\left(\frac{1}{3}\right)^2 + \left(\frac{1}{3}\right)^2}\right)}{\mathbf{p} \times \sqrt{\left(\frac{1}{3}\right)^2 + \left(\frac{1}{3}\right)^2}}$$

Equation 2 Sinc Interpolator Calculation Using Pythagoras Theorem

4.1.2 SINC INTERPOLATOR USING TWO PRE-INTERPOLATED DATA

In the sinc interpolation using two pre-interpolated data points, the points in parallel with two taxel data points are determined by the technique mentioned in the previous section. The method of obtaining the points in the middle is different as illustrated in Figure 17. Instead of calculating numbers directly out of the taxel data, two points on the side, which are in parallel with the actual point, are calculated first. Then those results are used to determine the final points. The equation for this method can be found in Equation 3, and the C code for this interpolation method can be found in the Appendix B.

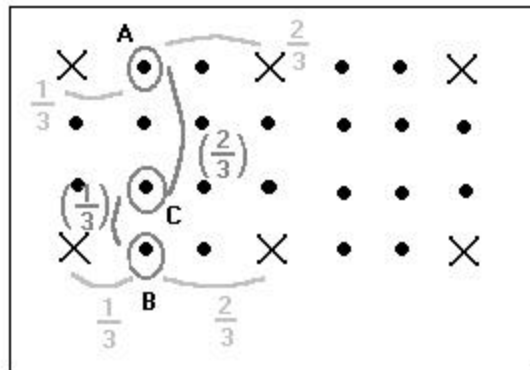


Figure 16 Sinc Interpolation Using pre-interpolated Data

$$(1) A = X1 \times \frac{\sin c(\mathbf{p}\frac{1}{3})}{\mathbf{p}\frac{1}{3}} + X2 \times \frac{\sin c(\mathbf{p}\frac{2}{3})}{\mathbf{p}\frac{2}{3}}$$

$$(2) B = X3 \times \frac{\sin c(\mathbf{p}\frac{1}{3})}{\mathbf{p}\frac{1}{3}} + X4 \times \frac{\sin c(\mathbf{p}\frac{2}{3})}{\mathbf{p}\frac{2}{3}}$$

$$(3) C = B \times \frac{\sin c(\mathbf{p}\frac{1}{3})}{\mathbf{p}\frac{1}{3}} + A \times \frac{\sin c(\mathbf{p}\frac{2}{3})}{\mathbf{p}\frac{2}{3}}$$

Equation 3 Sinc Interpolator Calculation Using Pre-interpolated Data

4.2 GRAPHICAL ANALYSIS ON INTERPOLATION METHODS

The computer-simulated interface is created in C language with OpenGL. OpenGL is an environment for developing portable, interactive 2D and 3D graphics applications. In order to observe the effectiveness of sinc interpolators, various modes of representation such as lines, cubes, and spheres were created (Appendix C and D). Moreover, image reconstruction with linear interpolation as well as planar interpolation is implemented. The C code for the graphical interfaces for different modes are in Appendix E. The following illustrations (Figure 18 – 22) represent reconstructed images in the line mode with various interpolation methods. The images represent the pressure of fingertips on the MTC Express Multi-Touch Controller.

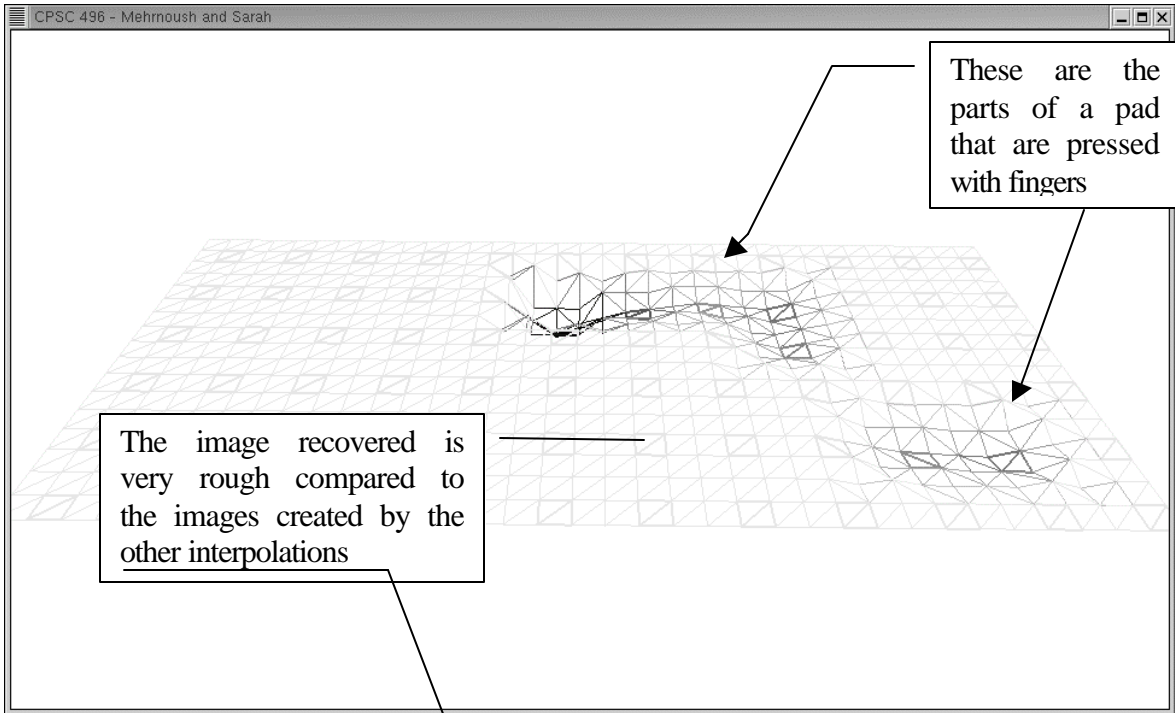


Figure 17 Front View of Pressures Applied on the Pad with Linear Interpolations

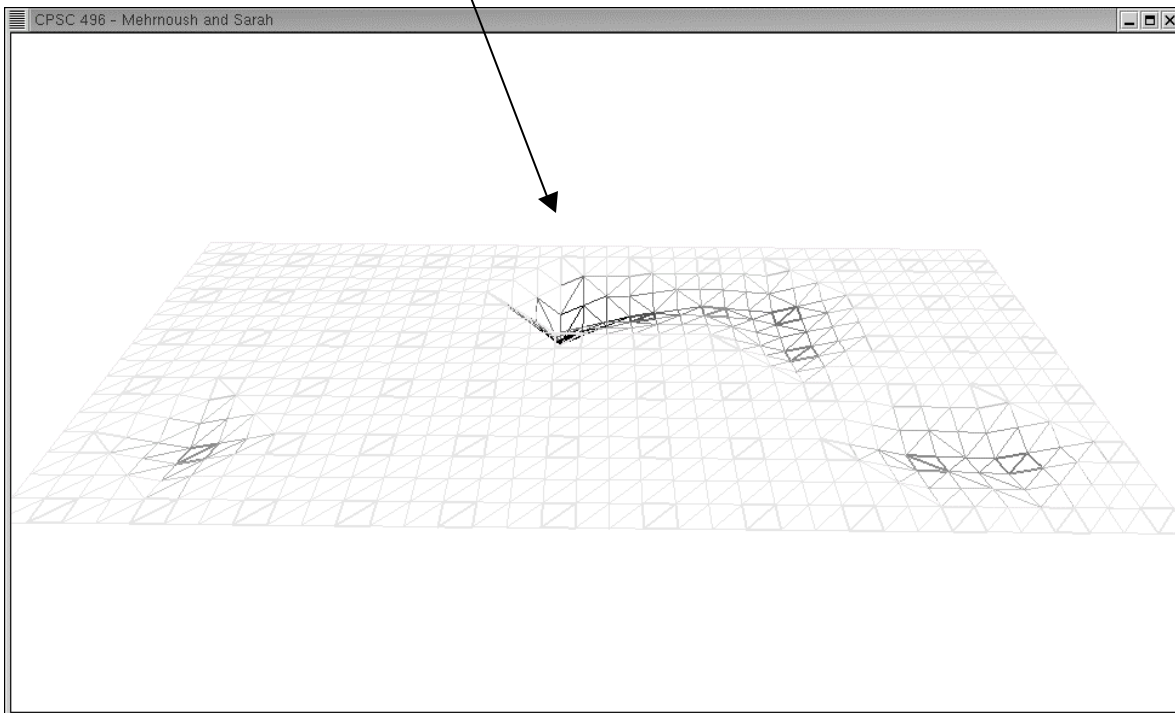


Figure 18 Front View of Pressures Applied on the Pad with Planar Interpolations

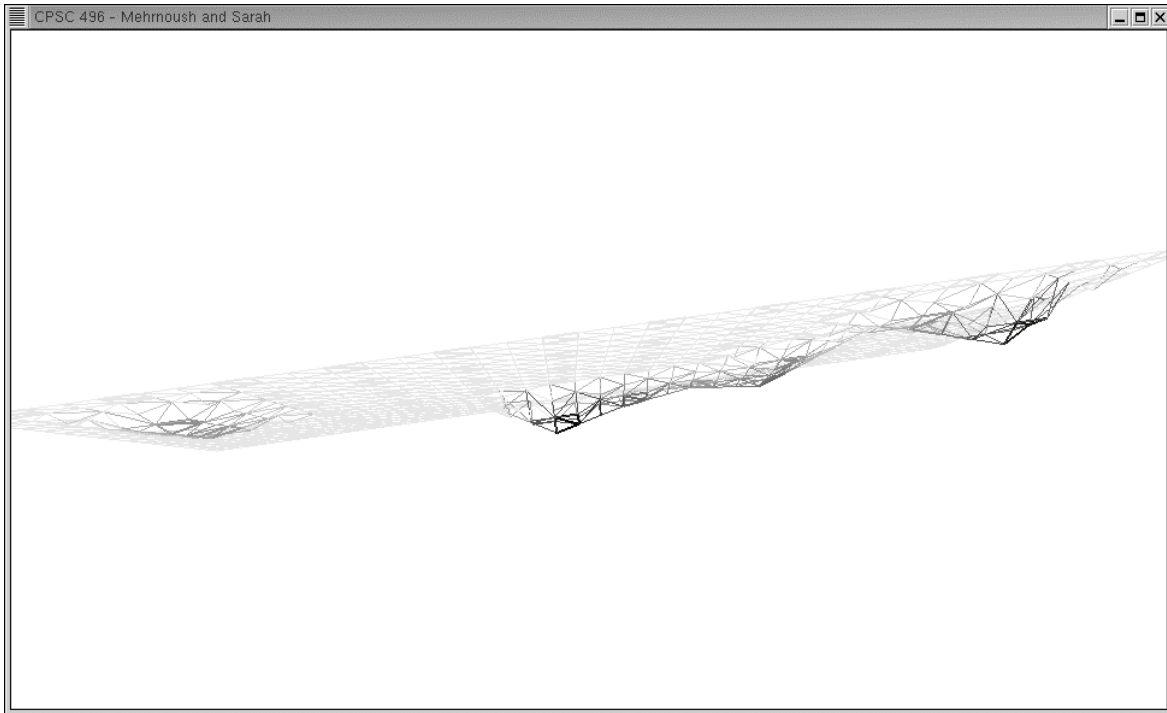


Figure 19 Side View of Pressures Applied on the Pad with Linear Interpolations

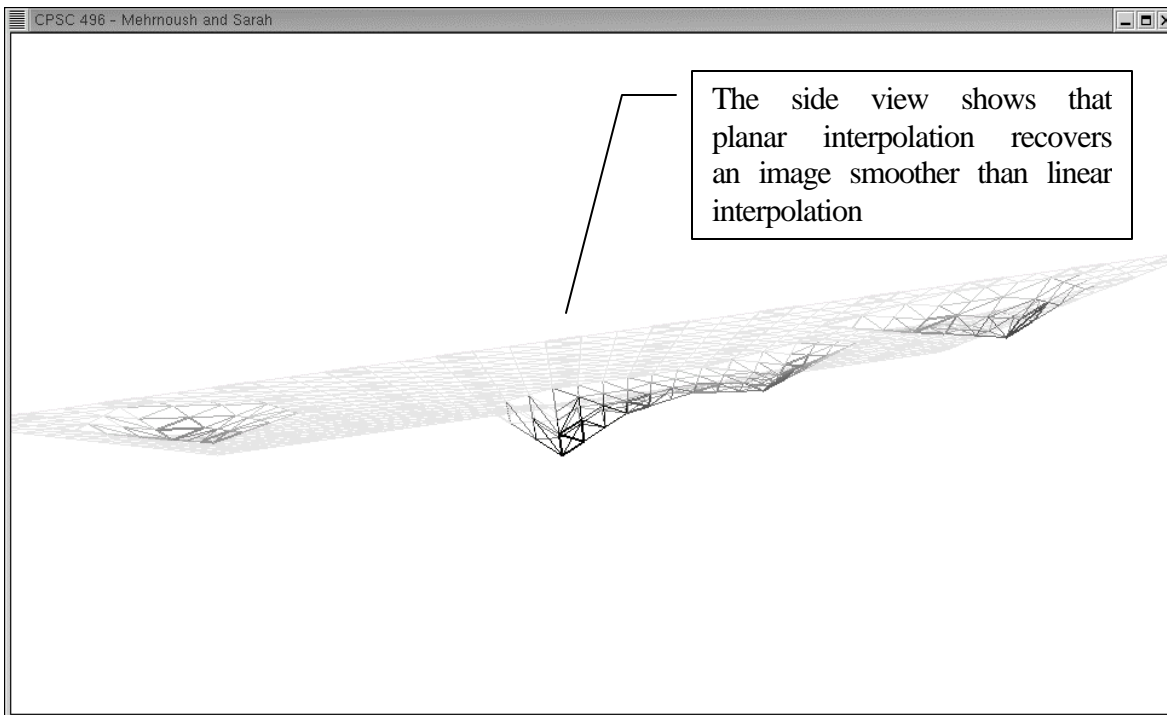


Figure 20 Side View of Pressures Applied on the Pad with Planar Interpolations

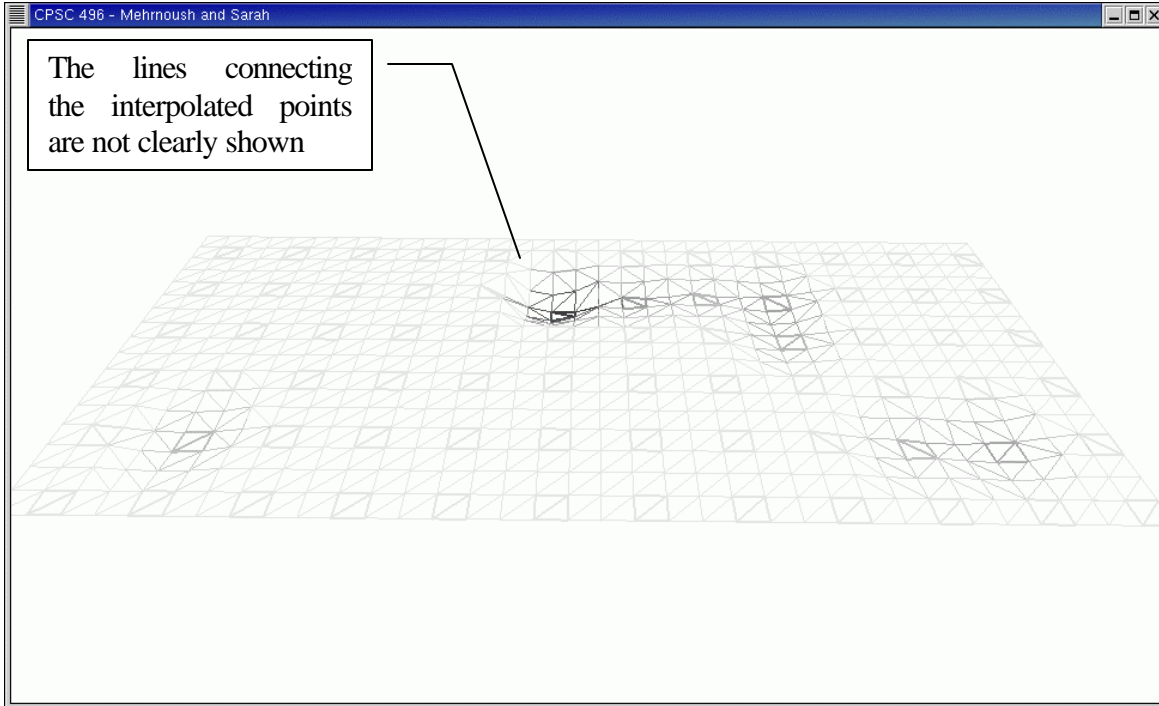


Figure 21 Front View of Pressures Applied on the Pad with Two-Taxel-Data Sinc Interpolations Using Pythagoras Theorem

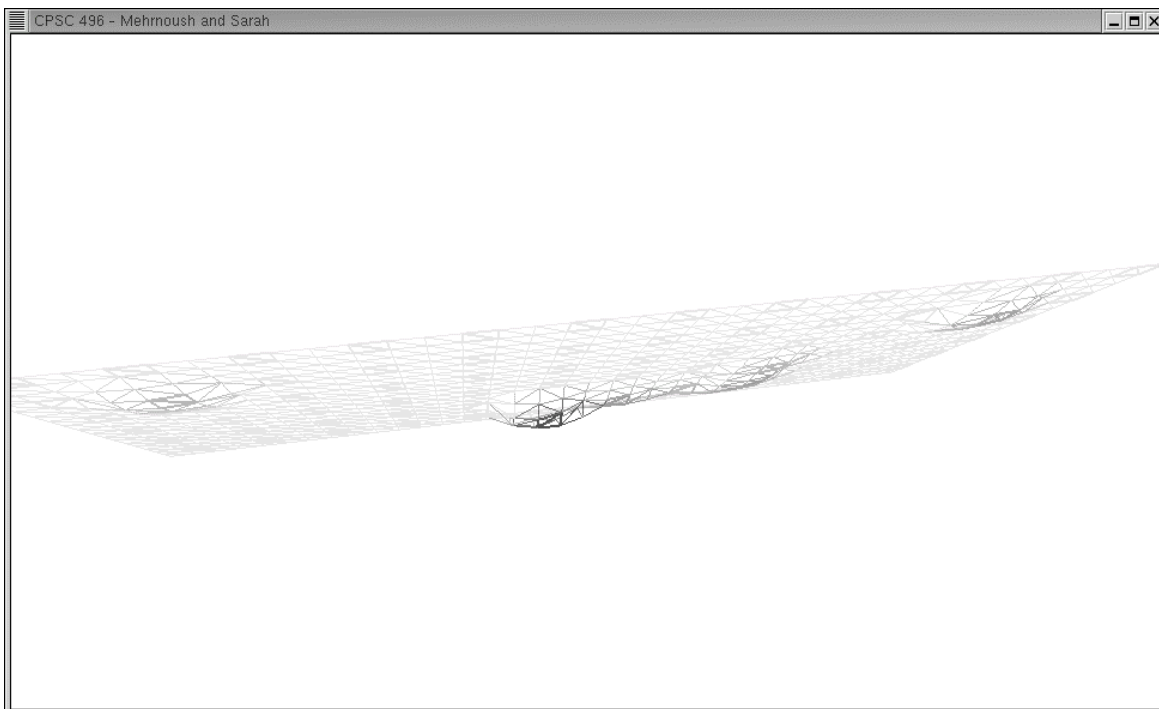


Figure 22 Side View of Pressures Applied on the Pad with Two-Taxel-Data Sinc Interpolations Using Pythagoras Theorem

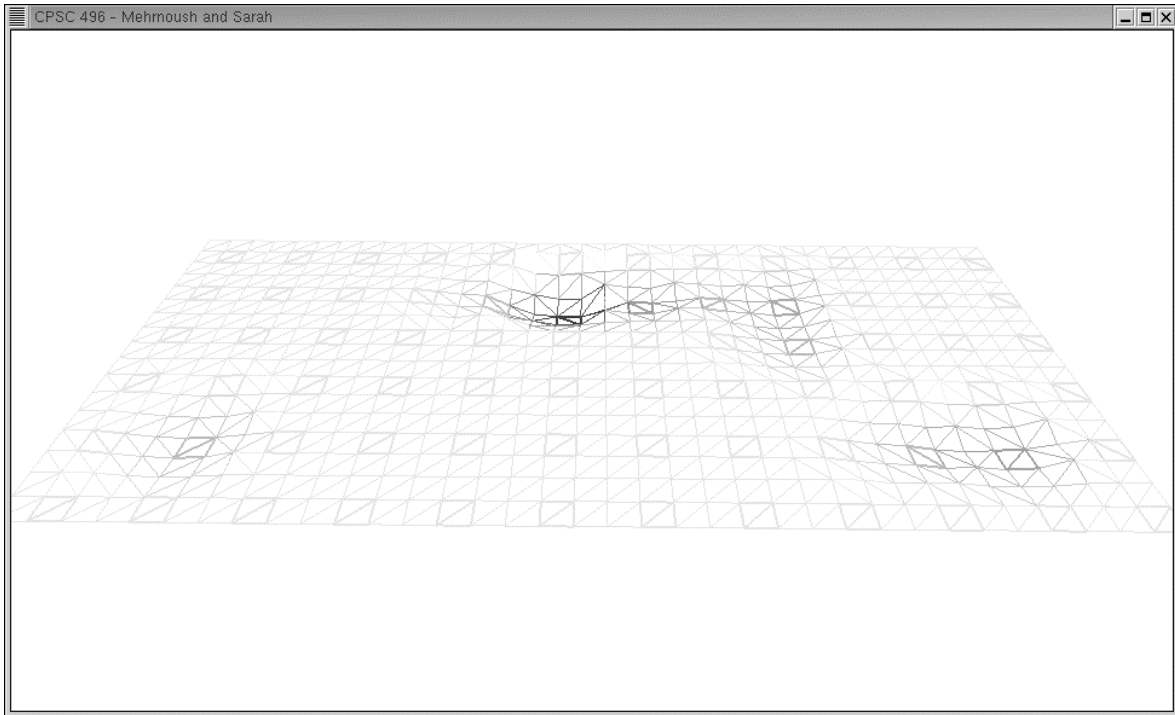


Figure 23 Front View of Pressures Applied on the Pad with Four-Taxel-Data Sinc Interpolations Using Pythagoras Theorem

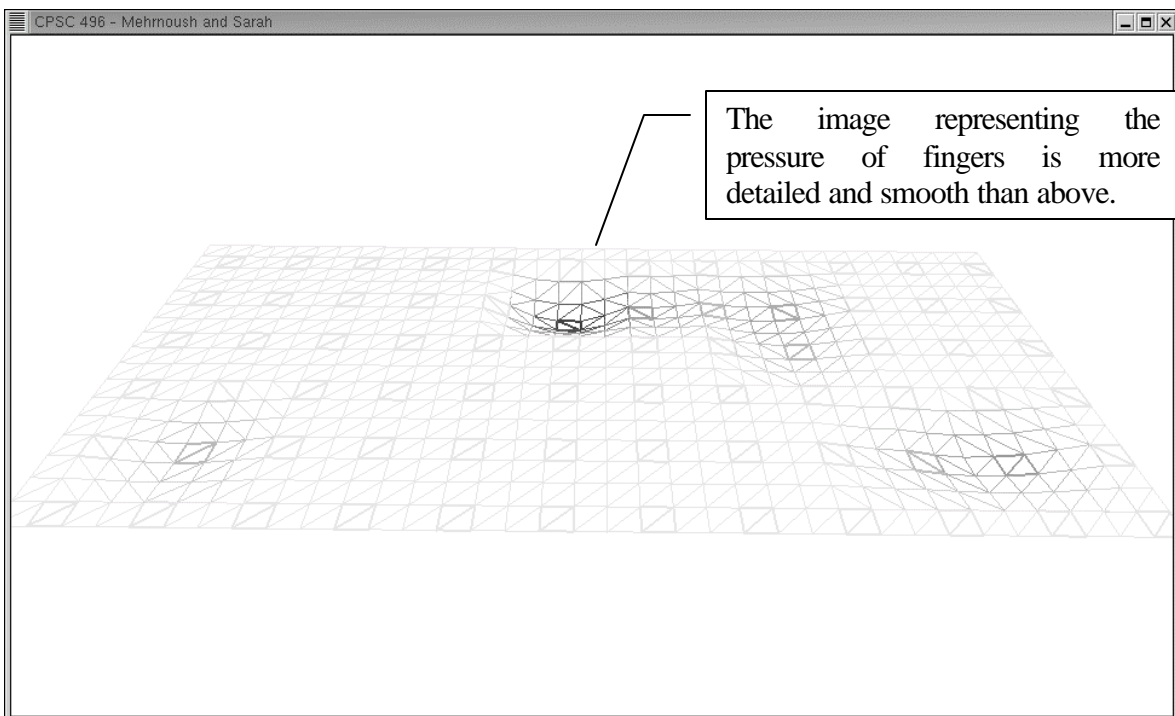


Figure 24 Front View of Pressures Applied on the Pad with Sinc Interpolations Using Pre-interpolated Data

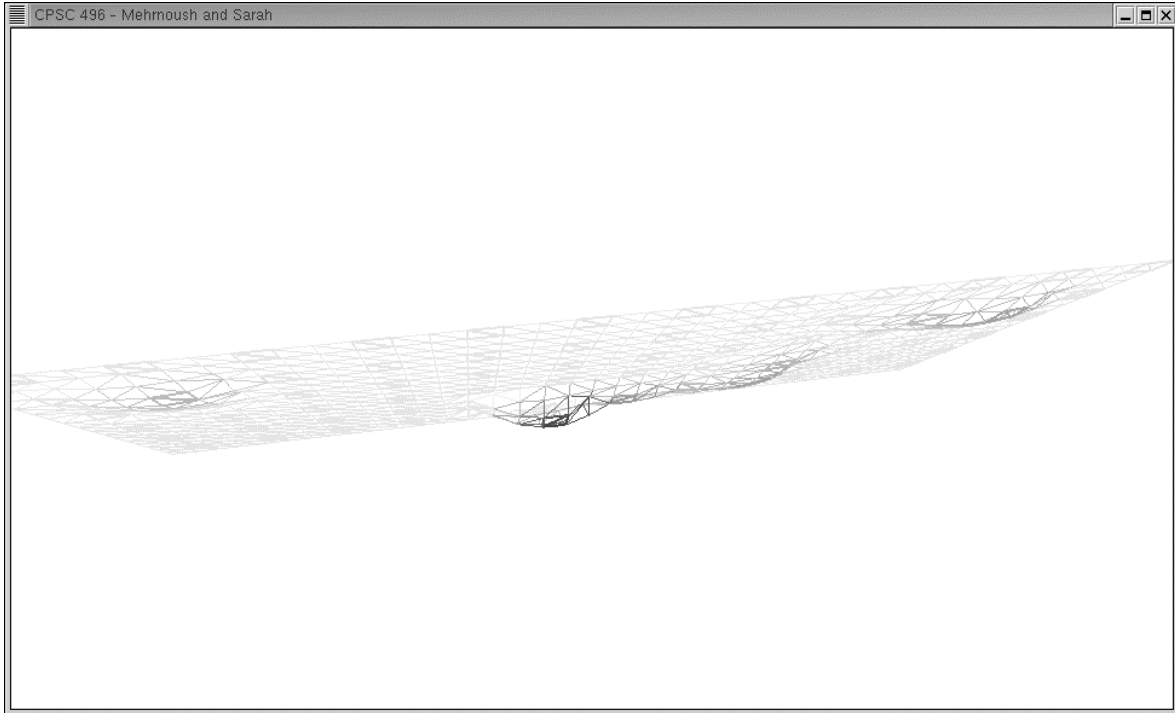


Figure 25 Side View of Pressures Applied on the Pad with Four-Taxel-Data Sinc Interpolations Using Pythagoras Theorem

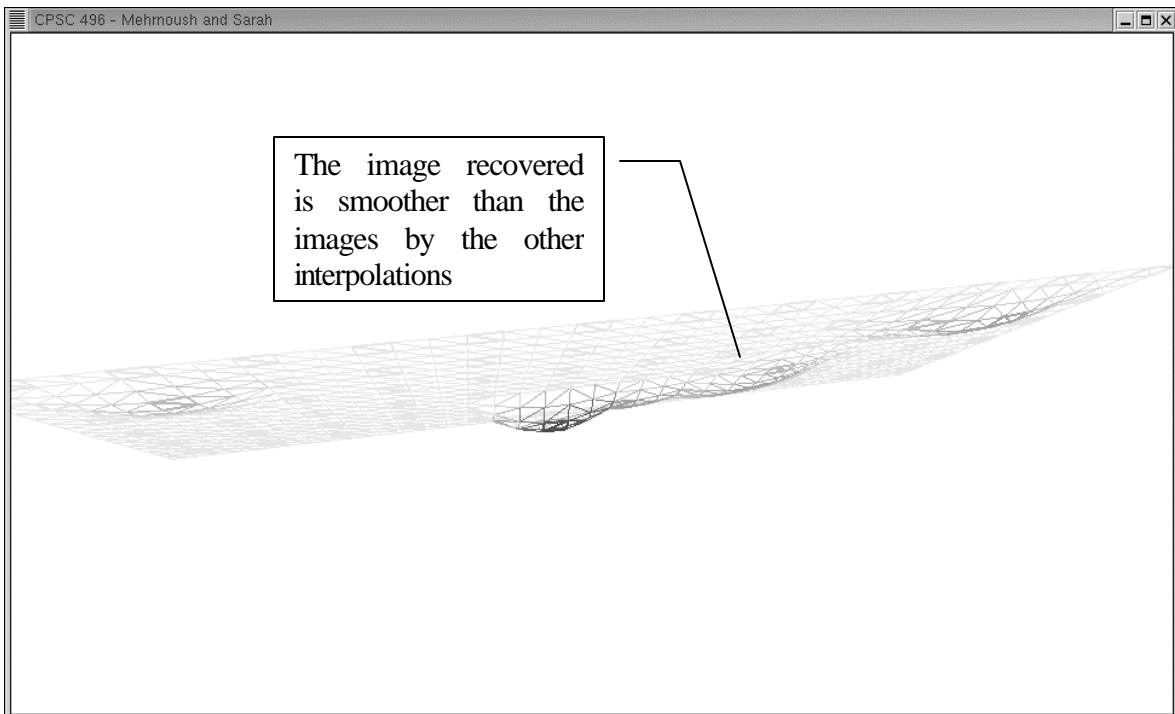


Figure 26 Side View of Pressures Applied on the Pad with Sinc Interpolations Using Pre-interpolated Data

As seen in the previous figures, the image reconstruction by the linear and planar interpolations results in broader and rougher images than the actual image. On the other hand, the sinc interpolation creates a smoother and narrower image than those by the linear and planar interpolations. Among the images by sinc interpolation, it is obvious that the image is clearer and more detailed when more taxel data points are used (Figure 21 and 22). The sinc function with pre-interpolated data also has smoother representations than the one created with Pythagoras Theorem.

5.0 CONCLUSION

The MTC Express Multi-Touch Controller is a touch pad that can measure multipoint pressure. With this device, when data signals for the image are correctly sampled and reconstructed, the movement of a finger, or even an entire hand, on the pad can be shown on the computer-simulated environment. This development is significant for future technology because it can eventually be expanded to allow for the exploration of the touch and caressing of other virtual simulated matters, including sand and water.

In order to reconstruct the image accurately, the Nyquist Theorem has to be applied for the data being sampled. Even though the sampling rate is lower than what is defined in the Nyquist Theorem, the creation of the image of hand movement can be attempted with the use of proper interpolators.

Typically a sinc interpolator is widely used for image processing due to its curvilinear characteristic. The sinc interpolator can be implemented in two different ways: using Pythagoras Theorem and using pre-interpolated data points. Both methods reconstruct more accurate images than using linear interpolation or planar interpolation. The sinc interpolation method with pre-interpolated data creates the best images.

APPENDIX A – C PROGRAMMING CODE FOR SINC INTERPOLATION WITH PYTHAGORAS THEOREM

```
void interpolate ()
{
    int i, j, p;
    p = 0;
    // insert taxel values
    for (i = I_LAYERS; i < TAXELROWI; i += CHUNK)
        for (j = I_LAYERS; j < TAXELCOLI; j += CHUNK)
            taxelValsl [i][j] = taxelValsLocal [p++] / 300.0;

    for (i = I_LAYERS; i < TAXELROWI; i += CHUNK) {
        for (j = I_LAYERS; j < TAXELCOLI; j += CHUNK) {

            //sinc interpolation with Pythagoras theorem (used only two taxel data points)
            if ((i != 16) && (j != 34)){
                taxelValsl [i][j+1] = taxelValsl [i][j]*sincvalue[6] +taxelValsl [i][j+3]*sincvalue[1];
                taxelValsl [i][j+2] = taxelValsl [i][j]*sincvalue[7] +taxelValsl [i][j+3]*sincvalue[0];
                taxelValsl [i+1][j] = taxelValsl [i][j]*sincvalue[6] +taxelValsl [i+3][j]*sincvalue[1];
                taxelValsl [i+2][j] = taxelValsl [i][j]*sincvalue[7] +taxelValsl [i+3][j]*sincvalue[0];
                taxelValsl [i+1][j+1] = taxelValsl [i][j]*sincvalue[3] +taxelValsl [i+3][j]*sincvalue[5];
                taxelValsl [i+2][j+1] = taxelValsl [i][j]*sincvalue[4] +taxelValsl [i+3][j]*sincvalue[2];
                taxelValsl [i+1][j+2] = taxelValsl [i][j+3]*sincvalue[3] +taxelValsl [i+3][j+3]*sincvalue[5];
                taxelValsl [i+2][j+2] = taxelValsl [i][j+3]*sincvalue[4] +taxelValsl [i+3][j+3]*sincvalue[2];
            }

            if ((i == 16) && (j != 34) ){
                taxelValsl [i][j+1] = taxelValsl [i][j]*sincvalue[6] +taxelValsl [i][j+3]*sincvalue[1];
                taxelValsl [i][j+2] = taxelValsl [i][j]*sincvalue[7] +taxelValsl [i][j+3]*sincvalue[0];
            }

            if ((i != 16) && (j == 34) ){
                taxelValsl [i+1][j] = taxelValsl [i][j]*sincvalue[6] +taxelValsl [i+3][j]*sincvalue[1];
                taxelValsl [i+2][j] = taxelValsl [i][j]*sincvalue[7] +taxelValsl [i+3][j]*sincvalue[0];
            }
        }
    }
}
}
```

APPENDIX B – C PROGRAMMING CODE FOR SINC INTERPOLATION WITH PRE-INTERPOLATED DATA

```

void interpolate_using_preinterpolated_data()
{
    int i, j, p;
    p = 0;
    // insert taxel values
    for (i = I_LAYERS; i < TAXELROWI; i += CHUNK)
        for (j = I_LAYERS; j < TAXELCOLI; j += CHUNK)
            taxelValsl [i][j] = taxelValsLocal [p++] / 300.0;

    for (i = I_LAYERS; i < TAXELROWI; i += CHUNK) {
        for (j = I_LAYERS; j < TAXELCOLI; j += CHUNK)

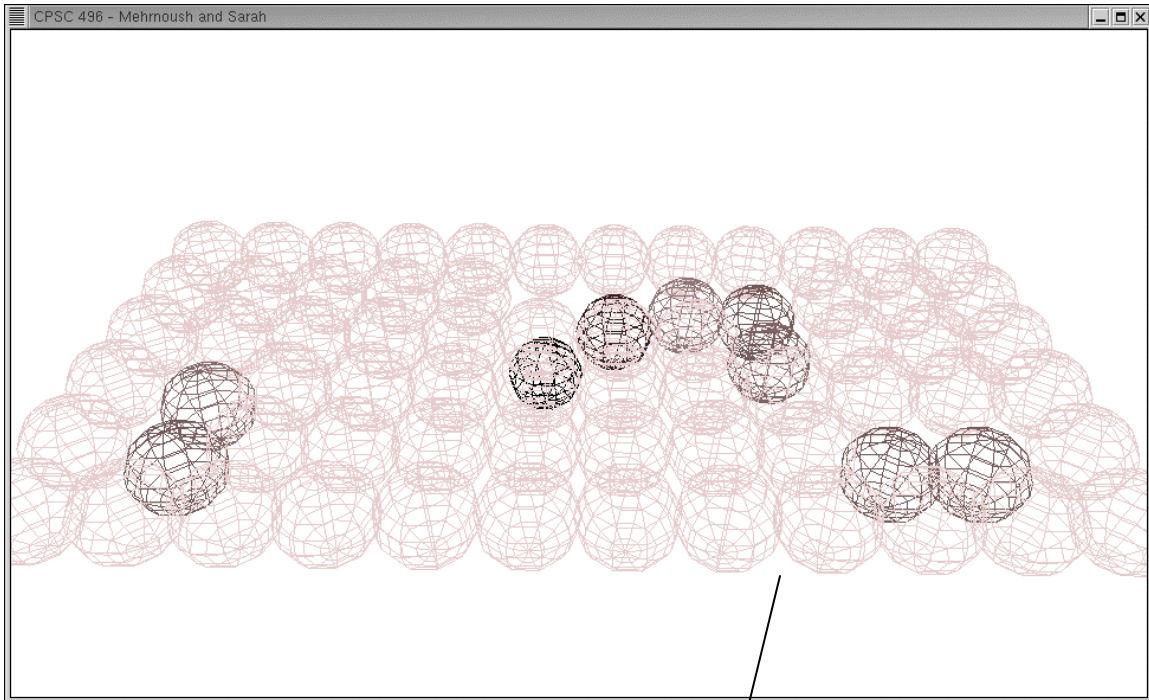
            { // sinc interpolation: using pre-interpolated values
                if ((i != 16) && (j != 34)){ //calculating the side points first
                    taxelValsl [i][j+1] = taxelValsl [i][j]*sincvalue[6] +taxelValsl [i][j+3]*sincvalue[1];
                    taxelValsl [i][j+2] = taxelValsl [i][j]*sincvalue[1] +taxelValsl [i][j+3]*sincvalue[6];
                    taxelValsl [i+1][j] = taxelValsl [i][j]*sincvalue[6] +taxelValsl [i+3][j]*sincvalue[1];
                    taxelValsl [i+2][j] = taxelValsl [i][j]*sincvalue[1] +taxelValsl [i+3][j]*sincvalue[6];
                }

                //calculating the points in the middle
                if ((i == 16) && (j != 34) ){
                    taxelValsl [i][j+1] = taxelValsl [i][j]*sincvalue[6] +taxelValsl [i][j+3]*sincvalue[1];
                    taxelValsl [i][j+2] = taxelValsl [i][j]*sincvalue[1] +taxelValsl [i][j+3]*sincvalue[6];
                }
                if ((i != 16) && (j == 34) ){
                    taxelValsl [i+1][j] = taxelValsl [i][j]*sincvalue[6] +taxelValsl [i+3][j]*sincvalue[1];
                    taxelValsl [i+2][j] = taxelValsl [i][j]*sincvalue[1] +taxelValsl [i+3][j]*sincvalue[6];
                }
            }
        }
    }
}

//for(j)
//for(i)
for (i = I_LAYERS; i < TAXELROWI; i += CHUNK) {
    for (j = I_LAYERS; j < TAXELCOLI; j += CHUNK)
        { // sinc interpolation: using pre-interpolated values
            if ((i != 16) && (j != 34)){
                taxelValsl [i+1][j+1] = taxelValsl [i+1][j]*sincvalue[6] +taxelValsl [i+1][j+3]*sincvalue[1] ;
                taxelValsl [i+1][j+2] = taxelValsl [i+1][j]*sincvalue[1] +taxelValsl [i+1][j+3]*sincvalue[6] ;
                taxelValsl [i+2][j+1] = taxelValsl [i+2][j]*sincvalue[6] +taxelValsl [i+2][j+3]*sincvalue[1];
                taxelValsl [i+2][j+2] = taxelValsl [i+2][j]*sincvalue[1] +taxelValsl [i+2][j+3]*sincvalue[6];
            }
        }
    }
}
//for(j)
//for(i)
}
//main

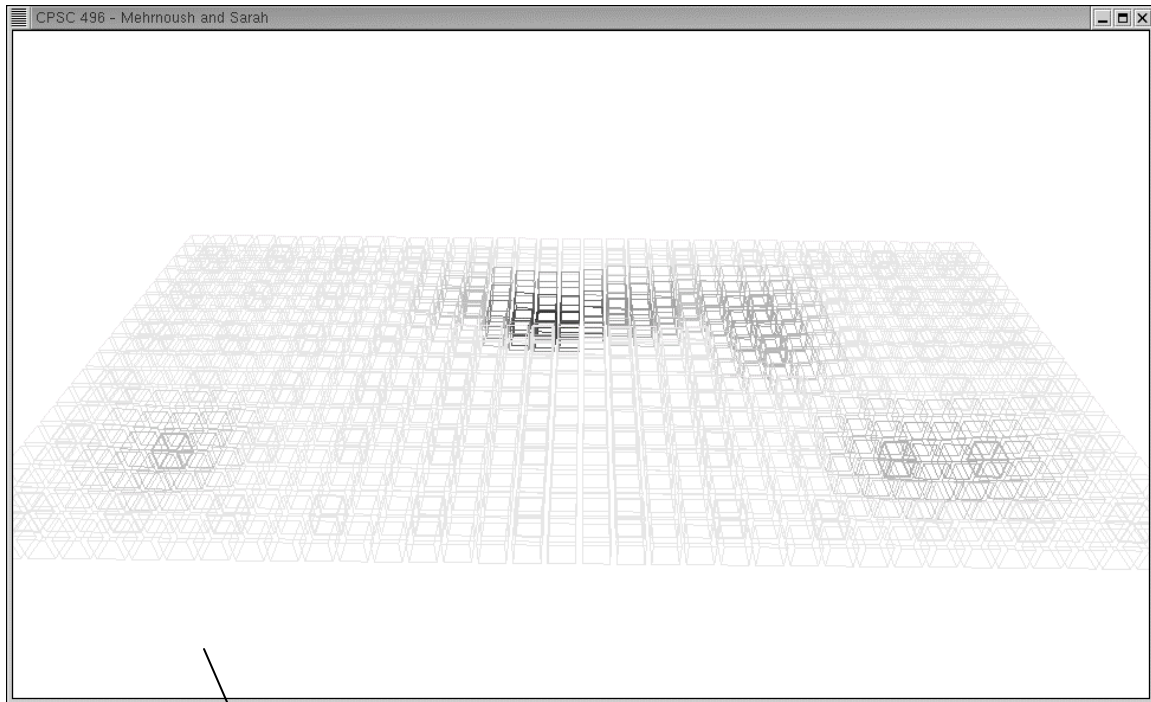
```

APPENDIX C – GRAPHICAL REPRESENTATION IN SPHERE MODE



This image represents the pad, where the spheres represent taxels. The black spheres are the taxel points that are pressed by fingers

APPENDIX D – GRAPHICAL REPRESENTATION IN CUBE MODE



This image represents the pad filled with a layer of cubes. The cubes represent taxels and interpolation points.

APPENDIX E – C PROGRAMMING CODE WITH OPENGL

FOR SPHERE GRAPHIC MODE

```
{
    glTranslatef (-TAXELCOL * d * 0.5 - 0.5 * d,
                 TAXELROW * d * 0.5 - 0.5 * d,
                 0.0); //setting the position of the center of the image

    for (i = 0; i < NUMTAXEL; i++) {

        glTranslatef (d, 0.0, 0.0);

        if (!(i % TAXELCOL) && (i)) {
            glTranslatef (0.0, -d, 0.0);
            glTranslatef (-TAXELCOL * d, 0.0, 0.0);
        }
        //move the position of the spheres pressed
        glColor3f (0.9 - taxelValsLocal [i] / 128.0,
                 0.8 - taxelValsLocal [i] / 128.0,
                 0.8 - taxelValsLocal [i] / 128.0);
        glTranslatef (0.0, 0.0, -taxelValsLocal [i] / 300.0);
        glutWireSphere (0.27, 10, 10);
        glTranslatef (0.0, 0.0, taxelValsLocal [i] / 300.0);

        //move back to its original position
    }
}
```

BIBLIOGRAPHY

ⁱ “MTC Express Guide,”P.1, Figure 1,
http://www.tactex.com/Media/MTC_Express_Guide.pdf

ⁱⁱ “MTC Express Guide,”P.18, Figure 14

ⁱⁱⁱ MTC Express Guide,”P.26, Figure 16