The International Journal Of Engineering And Science (IJES)
||Volume||2 ||Issue|| 3 ||Pages|| 15-24 ||2013||
ISSN(e): 2319 – 1813 ISSN(p): 2319 – 1805

2d To 3d in Real Time

[1,]Prof. H.N.Patil, [2,] Rohit Devare, [3,] Kshitija Bathe, [4,] Kirti Bhosale, [5,] Shraddha Devdikar

[1,]*Faculty, E&Tc, Pune University/Pvpit College, City Pune, India,*
[2,3,,]*E&Tc, Pune University/Pvpit College, City Pune, India,*
[4,]*E&Tc, Pune University/Pvpit College, City Pune, India,*
[5,]*E&Tc, Pune University/Pvpit College, City Pune, India,*

----------Abstract----------

In this paper, we present a simple 3D paint system that does not require any critical programming or technical bias. Presently many 2D to 3D conversion system needs a prior knowledge of the software and its use, so it's difficult to be used by everyone. We have made an attempt to use the simplest of all possibly usable technologies in 2D-3D conversion to aid the conversion process in the paint system. This paper proposes a simple system that has a plausible implementation through high frequency sound and the known propagation delay of acoustic waves to trilaterate distances. The system developed consists of sensing real time data (drawing), instant conversion and displaying the 3D drawing on the MATLAB. This project achieves the goals by first sensing the real-time drawing, calculating the time delay and hence the distances from it and then displaying its 3D drawing simultaneously from all angles.

Index Terms*— Artistic interface, MATLAB, Real- time, Trilateration*

Date Of Submission: 13.March, 2013 ⟺ Date Of Publication: 25 March 2013

I. INTRODUCTION

Although three-dimensional (3D) displays enhance visual quality more than two-dimensional (2D) displays do, the depth information required for 3D displays is unavailable in the conventional 2D content. Therefore, converting 2D videos into 3D ones has become an important issue in emerging 3D applications.The 3D video signal processing has received considerable attention in visual processing. Given advances in 3D display technology, humans aspire to experience more realistic and unique 3D effects. In addition to generating better visual experiences than conventional 2D displays, emerging 3D displays have many applications, including broadcasting, movies, gaming, photography, camcorders, and education.This system achieves the goals by first sensing the real-time drawing, calculating the time delay and hence the distances from it and then displaying its 3D image simultaneously from all angles. The project describes the short-term as well as long-term impacts of the use of this system in 3D application areas. We have tried to develop a technology that ensures integrated 2D-3D data conversion which is going to prove very user-friendly.Finally, the project hypothesizes some future developments of real-time data conversion. Current technology primarily uses highly advanced techniques to perform the job of conversion. But these techniques pose certain shortcomings. The primary benefit of real-time conversion with the help of ultrasonic sensors is that it can produce the desired output faster and with same amount of integrity.We use the simple technique of drawing using trilaterated coordinates from ultrasonic delays.

II. RELATED WORK

This work builds on several areas of related research. Here, we contrast our approach with techniques in Real-Time Painting with an Expressive Virtual Chinese Brush, Novel 2D-3D conversion system using edge information, drawing on air: Input techniques for controlled 3D line illustration.

2.1 Real-Time Painting with an Expressive Virtual Chinese Brush [1]:

To drive the virtual brush, a 6-DOF input device is built by combining affordable sensor components. Ultrasonic device and miniature gyroscopes is used to detect the brush position and orientation, respectively.These sensors are attached to a real brush or a brush like object, which the user manipulates to drive

the virtual brush in real time. Users can calibrate the input device to map a real supporting surface to the virtual one.

2.2 A novel 2D-3D conversion system using edge information [2]:

The proposed algorithm utilizes edge information to group the image into coherent regions. A simple depth hypothesis is adopted to assign the depth for each region and a cross bilateral filter is subsequently applied to remove the blocky artifacts. The proposed algorithm is quality-scalable depending on the block size. Smaller block size will result in better depth detail and large block size will have lower computational complexity.

2.3 Drawing on air: Input techniques for controlled 3D line illustration [3]:

Drawing on Air integrates two complimentary approaches to drawing 3D curves, one-handed drag drawing and two handed tape drawing.One-handed is generally easier to learn to control than two handed, whereas two-handed feels more controllable to expert users. One-handed is also more appropriate for circular shapes that would require one's arms to cross if drawn with the two-handed approach.In the one-handed drawing mode, both of these operations are performed with one hand. The artist drags around the brush. The drawing is constrained to move along the "tow rope," which describes the tangent of the curve.

III. PROPOSED 3D PAINT SYSTEM

This is a 3D paint system that has a plausible implementation through high frequency sound and the known propagation delay of acoustic waves to trilaterate distances.

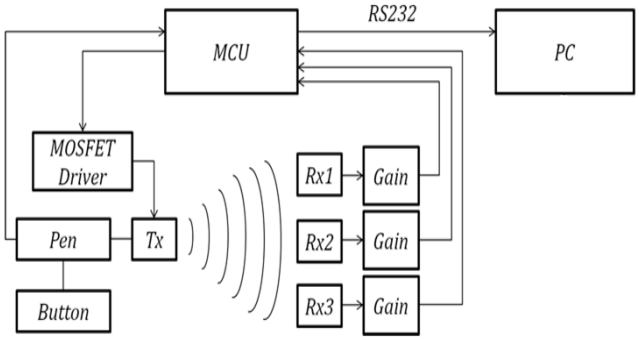

Fig.1 Block Diagram

We implement the pen with a single critical button to indicate drawing/not drawing. The artist can hold it and press it only when they want to make a stroke. To limit the physical footprint of the device, to avoid relaying button states up to MATLAB as the microcontroller did not need to know the button states and to avoid wasting cycles and time to capture and transmit them (as UART communication is very slow), implementing rest of the UI in software made the program much more extensible: new features could simply be added in code without any new hardware development. By interfacing with MATLAB we can display the drawing in high quality plots that are perfect for exporting. Further, MATLAB supports full GUI design which provides an interface for the artist to select various brush sizes, colors, and styles. The program operates in a 'paint' mode to draw or in a 'camera' mode in which the artist can use the pen as a virtual camera to look around their drawing.

In the sections below, we describe the details and implementation of the system.

2.4 Distance Trilateration

The foundation for the device relies on the fact that the speed of sound is constant in a given medium. Under everyday conditions, the speed of sound in air is 340.29 m/s. This means that we have a bijective mapping between time and distance for sound propagation. By emitting a sound pulse and recording the time delay between emission and detection, we calculated a displacement via the following equation:

$$\text{displacement} = \text{velocity X time}$$

One delay measurement will give you a position in one dimension along the line-of-sight of the transmitter and receiver, yet in three dimensions all a single delay value will tell you is that the emitter was somewhere on the surface of a sphere centered at the receiver and with a radius equal to the time delay multiplied by the speed of sound. Positioning another receiver uniquely gives two spheres of possible locations centered around each receiver, and the two sphere's intersection (generally a two-dimensional circle) gives all the possible locations

that satisfy both delay measurements. Adding yet another unique receiver further stipulate the position of the pen exactly. The resulting coordinate system and placement of the three receivers can be seen below.

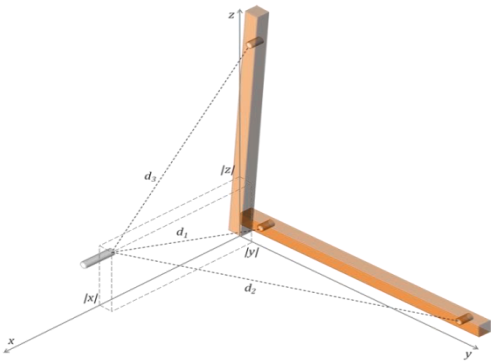

Fig.2 Mathematical coordinate system for trilateration

We now have three equations (spheres located around each receiver) and three unknowns (where *l* is the displacement of each receiver along a given axis), thus a single unique position for the pen can be determined.

These sphere equations are:

$$d_1^2 = x^2 + y^2 + z^2 \qquad (1)$$

$$d_2^2 = x^2 + (y-l)^2 + z^2 \qquad (2)$$

$$d_3^2 = x^2 + y^2 + (z-l)^2 \qquad (3)$$

The z and y coordinates can be calculated directly from the delay measurements, and then using these values x can be solved for. We end up with the following:

$$z = (d_1^2 - d_3^2 + l^2) / 2l \qquad (4)$$

$$y = (d_1^2 - d_2^2 + l^2) / 2l \qquad (5)$$

$$z = Real(d_1^2 - y^2 + z^2)^{-1/2} \qquad (6)$$

Note that we are taking the real part of the square root in the x calculation as the delay measurements are mathematically imperfect and taking the real part is a very good approximation .These equations allow us to perform what is called a trilateration.

2.5 Logical Structure

There are three primary components involved with the system: the microcontroller, the hardware, and the PC running MATLAB. The microcontroller's primary function is to facilitate the rapid acquisition of time delays between the transmitter and three receivers as it has inherent ability to interface with analog hardware and communicate with higher-level machines like a PC. The microcontroller coordinates the emission of a sound pulse according to many timing specifications. For instance, it cannot emit too fast or the receivers would become confused as to which received pulse corresponds to which emitted pulse. Additionally, some receivers might not have received a pulse if the pen was directed away at too great an angle so it needs to be prepared to handle measurement timeouts. The microcontroller is responsible for keeping all pulses and receptions locked in step to ensure quality data. Task of signal processing is assigned to it to improve responsiveness. The microcontroller is also to follow a protocol to transfer data up to MATLAB. When MATLAB requests data, the microcontroller is to respond via UART serial communication with a packet containing information relating to the pen's status (button pushed or not) and the three delay values. MATLAB is where the bulk of the artist interaction is centralized. It is to produce a fully functional GUI so that the artist can change various parameters about the paintbrush (i.e. color, etc.) and receive visual feedback as to what they are drawing.

2.6 User Interface-MATLAB

We created a GUI in MATLAB using the stock GUIDE GUI development environment. It allows us to graphically draw out where we wanted each component and then write back-end code to support it. The GUI draws the xyz-coordinate data in 6 separate ways according to the following table:

I. xyz-coordinate data in GUI

Name	**Description**
3D Space	A 3D projection of the data. The camera is used to move around this plot.
Stereo – Spectroscopic	Two plots (one for left eye and one for right eye). The camera is used in this plot.
Stereo – Anaglyph	A single 2D plot with red and blue data overlaid for use with R-B glasses.
XY, YZ, and XZ Projections	2D plots of each orthogonal projection.

It also provides the artist the ability to change the following parameters:

II. Various parameters available

Button Name	**Type**	**Options**
Brush Color	Toggle	Various reds, greens, blues, oranges, etc.
Brush Size	Dropdown	[1-10]
Brush Type	Dropdown	[Solid Line, Dashed Line, Dotted Line, Dash-Dot Line]
Pen Mode	Toggle	[Pen, Camera]
Background Color	Toggle	[Black, White]
Display Mode	Toggle	[3D Space, Orthogonal Planes, Stereo]

MATLAB also has to provide buttons for exiting the GUI and clearing the window of any drawings properly. Finally, the MATLAB program has to also allow the artist to export their drawing as JPEGS and display the connection status of the 32. If the connection is opened correctly, it displays “Open” in green letters in the connection status window and “Closed” in red letters otherwise.

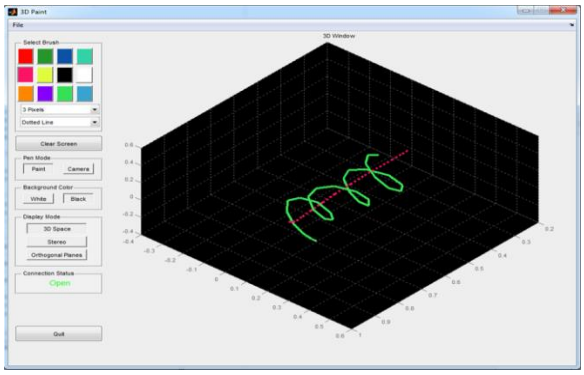

Fig.3 3D Paint MATLAB GUI

MATLAB initializes by setting up the serial object to talk with the 32, creating various global variables that are passed around the GUI (like brush color), and configuring various particulars about the GUI (like plot color and axis size). It also creates a very important timer object. The timer object works almost exactly like a 32 timer would. We have set up how often we want it to execute (every 50 msec in our case) among other parameters and then programmed a callback function for it. The vast majority of our code resides within this timer callback function. The first thing the callback does is send a ‘y’ to the 32 letting it know that it wants data and then wait to receive the data. When we get the packet, we unpack it by assigning the three delays and button state to variables used within MATLAB. Comma delimiting the variables and using %d to format all incoming data works well.

Fig.4 Data packet structure

Additionally, surrounding the code by try/catch statements catches any erratic packets that we do not want to process. Cycle delays are scaled by a parameter called SLEW_OFFSET. We noticed that the instrumentation amplifiers has a slew that is quite quick even when applying massive gain, yet the slew still causes a noticeable offset in the cycle delay times because the interrupt isn't triggered precisely when the pulse is received. But the delay is generally constant so we can correct for it. The best way to calibrate this value is to observe the data streams of x, y, and z data points vs. time and adjust the value until all streams are perfectly isolated from one another when the pen is moved in only one dimension. For instance, if the value is not properly set, and pen is moved up and down, some coupling between the z value and the x and y values is seen. The other parameter that is fine-tuned for this is the distance between each pair of receivers. Next, if the packet is good, MATLAB processes the data from cycle delays into distance in meters. If the program is in paint mode it buffers the distance into a buffer specifically for paint mode. Because MATLAB pulls values from the 32 at a rate not equal to the rate that they were produced on the 32, breaks in the data were reintroduced. To mitigate this, a very small sliding window mean filter exactly like the one on the 32 is introduced. In the case of paint mode the window is of length 3 to not detract from responsiveness too much. Each delay stream has its own mean filter. Next the filter delays are trilaterated into xyz-coordinates via the above equations and a linear interpolation between a point and its predecessor is produced.

One critical feature of the MATLAB program is to display a cursor of the current pen position on all the plots for the artist to have an idea where the pen is relative to the rest of the drawing before making a new stroke.If instead the program is in camera mode, then the incoming distance measurements are placed in a separate mean filtering buffer. The reason for separate buffers is that the camera mode can afford a bit more delay but has to be much more stable compared to the draw mode. In camera mode slight variations introduced by the artist's hand jiggling have to be smoothed out so that looking around the drawing is very fluid. Further, in draw mode feedback delay is limited, but in camera mode this requirement is not as tight, so we can mean more values (we're using 10 currently for camera mode) and accept a slightly longer delay for the smoother camera positioning. In camera mode the xyz-coordinates are simply used to specify the location of the camera (the camera always points at the central point in the design space). Further, for the stereo plots we positioned the camera for the left and right eye according to a formula. Each of the toggle buttons and drop down menus on the GUI have a corresponding call back function. In each callback function, the appropriate global variable is set by whatever the artist selects for use elsewhere in the program. This is accepted only by the Display Mode toggle button, which turns on and off the visibility of the respective axes objects on top of updating a global variable. The exit button functions by simply deleting and deallocating any necessary objects for proper termination of the GUI. The clear button functions by clearing all axes objects. On the whole the MATLAB code is relatively straightforward to implement. The GUI overhead provides some interesting insight into MATLAB programming, and use of the online documentation was critical to producing working code. There were many cumbersome and strange oddities of the MATLAB GUI that has to be worked out. The biggest upside to using MATLAB is the extensive infrastructure it already houses for producing things like 3D plots, exporting images, rotating plots, etc.

2.7 Hardware Implementation

The hardware for this project provides very simple functionality. Complexity of hardware is minimized to interfere as little as possible with the artist's workflow. We have designed and implemented compact, efficient analog circuitry and supporting stands to optimize portability, footprint, and quality.

2.8 The Stand and Pen

The stand was implemented out of thin rectangular wood totaling 1 inch on a side. It featured a center joint made of a screw and nut about which the stand could fold so that the device could be easily moved about. All three receivers are mounted in one plane facing the design space so that the pen can always be pointed normal to that plane. This along with the fairly wide directionality of the Rx/Tx pairs allows a single Transmitter to maintain constant connectivity with all three Rx's simultaneously. Each receiver is moved 47.00 cm from the origin.

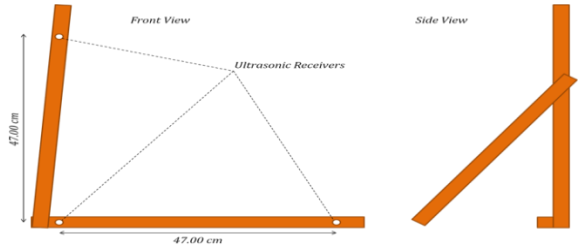

Fig.5 2D schematic of the final stand

The pen consists of a single pull single throw push button and the ultrasonic transmitter. The button's circuitry is very simple as elucidated by the following schematic.

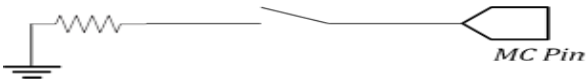

Fig.6 Pen button schematic

2.9 Transmitter Drivers

Ultrasonic Rx/Tx pairs are sharply resonant at 40 kHz, which is appropriate for use with the ATmega32 that can easily generate the necessary signal to drive them. They have a -6 dB point for a beam angle of 60° which is sufficiently spread for the project. Further, they have a fairly constant response over a wide range of operation temperatures (30°C-80°C). Also, they were designed to handle 30 V RMS driving voltage, so we could drive a very strong acoustic pressure wave from them.

Fig.7 An ultrasonic receiver mounted in the supporting stand

We need to drive the ultrasonic transmitter with a very powerful signal to produce the strongest response possible on the receiver side. MOSFET drivers provided a very simple and effective design and thus chose a Microchip TC4428.As we were gaining a square wave, and as the microcontroller 40 kHz output (0-5 V) is perfectly suited for logic lows and highs a single driver chip with both inverting and non-inverting internal drivers are used. The non-inverting driver outputs Vcc if it sees a voltage greater than .8 V on the input and ground otherwise. The inverting driver swapped this mapping.

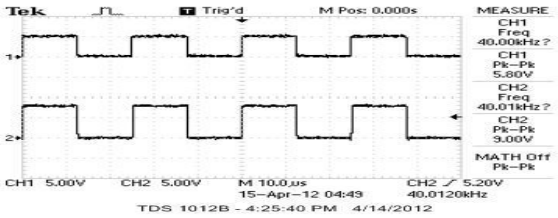

Fig.8 Channel 1: Microcontroller output; Channel 2: MOSFET output

Setting Vcc to 9 V (so that the device could be operated using a 9 V battery) and running 40 kHz output from the 16 to both the inverting and non-inverting inputs, an 18 V swing from a single power source is generated, because the inverting and non inverting outputs each produce a 9 V swing perfectly out of phase. The following oscilloscope capture shows the inverting (channel 1) and non-inverting (channel 2) output.

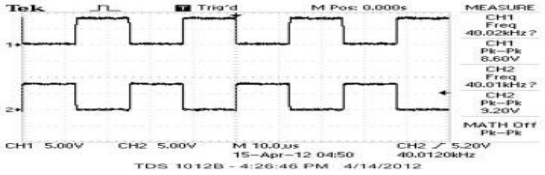

Fig.9 Channel 1: Inverting output; Channel 2: Non-inverting output

Hooking up these outputs to either pin on the ultrasonic transmitter produces a sufficiently strong acoustic signal according to the following schematic.

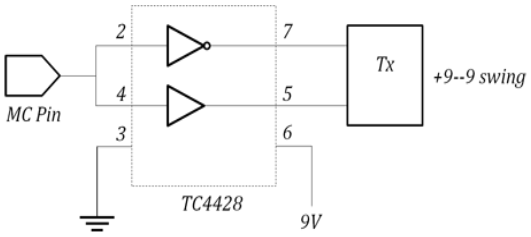

Fig.10 Transmitter driver circuit of TC4428

2.10Receiver Circuitry

Receiver circuits are generally implemented with a cascaded series of op-amps that perform gain and band limiting filtering. Signal is gained using a single instrumentation amplifier followed by a high-speed op-amp. The AD620 is an instrumentation amplifier based on a modification of the classic three op amp approach. This instrumentation amplifier is ideal due to its larger operating voltage (+/- 18 V) and fast slew rate 1.2 V/µs. A high slew rate is essential so that the amplifier could properly gain the quickly oscillating 40 kHz signal and minimize the time delay between received pulse and triggering of the microcontroller.The AD620 is a low cost, high accuracy instrumentation amplifier that requires only one external resistor to set gains of 1 to 1000. The amplifier accepts a differential input (perfect for the ultrasonic receiver) and only requires a single resistor to set the gain. The internal gain resistors, R1 and R2, are trimmed to an absolute value of 24.7 k□ , allowing the gain to be programmed accurately with a single external resistor using the following equation:

$$G = 1 + 49.4k\Omega / R_g \qquad (7)$$

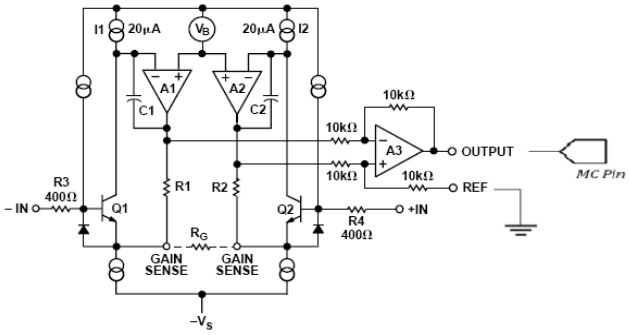

Fig.11 Simplified Schematic of AD620

III. EXPERIMENTAL RESULTS

In this section, we report on these results.

3.1 Delay Accuracy & Stability

The device operates very well within a reasonable design space. We limited the axes length (and thus implied operational range) to roughly 1 meter. Within this region, our raw data is fairly accurate, so post filtering we see very good results. The following plot shows the pen held at a constant distance from a single receiver. The raw data points are plotted along with a filtered curve.

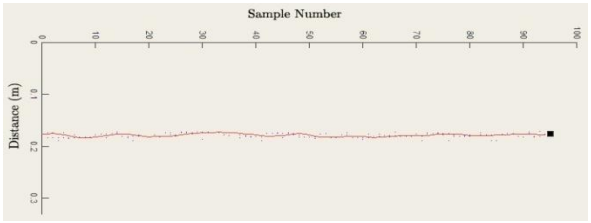

Fig.12 Single receiver accuracy

For this test we held the pen at 17.42 m from the transmitter, and MATLAB reported a mean filtered distance of 17.45 m, which was an acceptable error for our purposes. As witnessed from the plot, the data deviated very little from this average. When we tested all receivers at once, we ensured that their delay streams accurately

represented how the pen moved and that they all remained stable while the pen was stationary. The following diagram shows actual data while the pen was stationary.

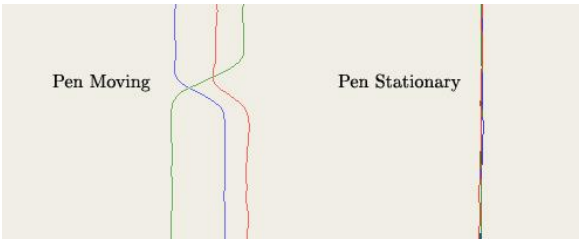

Fig.13 Multi receiver accuracy

3. 2 Isolation

The hallmark test for proper functionality was watching how our trilateration equations interpreted pen movements in real time. Initially we faced a large amount of coupling between the x, y, and z coordinates, with each coordinate unexpectedly following when another was varied. We corrected for this by manually calibrating two variables in the MATLAB code for optimal results. The first, as previously mentioned, was the variable SLEW_OFFSET, which allowed us to effectively reduce coupling between the different coordinates. The second was the length variable, which set the variable l used in our trilateration equations.The following images portray the quality of isolation we managed to achieve. For each panel, the pen was displaced in a different orthogonal direction. Hence, in each graph, two of the three lines should have ideally remained completely flat, while the third, representing the coordinates in the direction of motion, should have varied in a sinusoidal pattern. In these graphs, blue represents the x coordinates, green represents the y coordinates, and red represents the z coordinates.

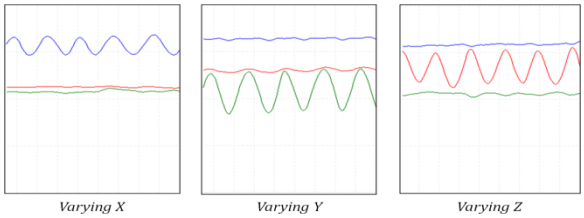

Fig.14 Coordinate isolation

3.3 D Accuracy

Ultimately, 3D Paint was able to fairly accurately represent the motion of the artist's pen. We tested the accuracy of the device by attempting to draw a perfect right angle in the yz-plane. We held a rectangular box and used the pen to trace out one of its corners, trying our best to maintain a fixed distance between the pen and the frame. This resulted in the images below, with a nearly perfect right angle seen in the yz-plane, and flat lines seen in the xy and xz-planes.

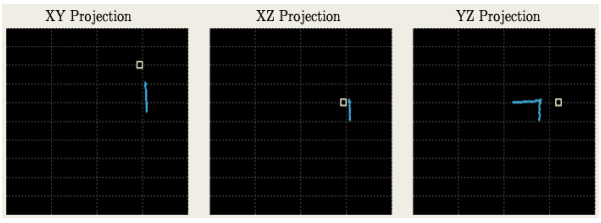

Fig.15 Right angle accuracy

3.4 Different Display Modes

The following GUI exports represent the functionality of the device as a whole. We will use the same drawing, a helix with a line through the middle, throughout this discussion so that the reader can see how each plot represents the same data differently.First is the 3D projection of the helix. In this mode, as previously iterated, the artist can navigate around the helix by switching into camera mode and moving the drawing pen.

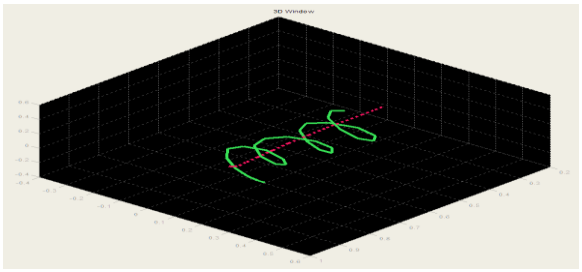

Fig.16 '3D Space' helix

Next are the three orthogonal projections of the data onto the xy, yz, and xz planes. We found this to be the best mode to draw in because lining up the pen and thinking in 2D was much simpler.

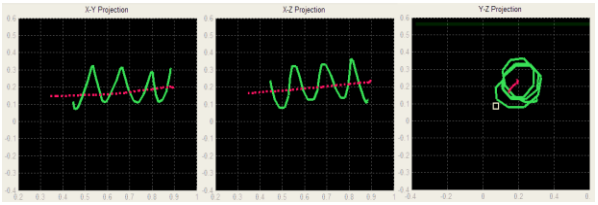

Fig.17 'Orthogonal Planes' helix

The following two images depict the stereo representations of the data. First is the stereoscopic representation, which is simply the same 3D data, plotted from slightly separated angles to simulate viewing a real object with two eyes. By getting very close and crossing your eyes much like a Magic Eye (or by using special stereoscopic goggles) the image can be seen to pop out of the screen. This stereo plot will also rotate with camera position.

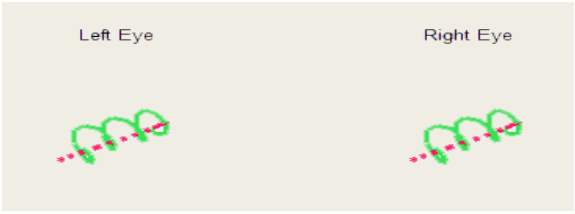

Fig.18 'Stereoscopic' helix

Finally we have the anaglyphic representation. This plot does not rotate with camera angle, and displays the image in a 2D plot with red/blue representations (slight depth offsets) overlaid.

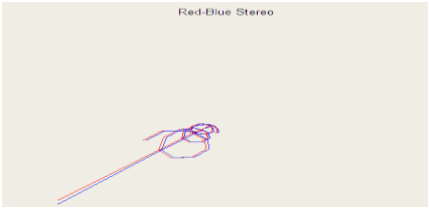

Fig.19 'Anaglyphic' helix

3.5 Speed of Execution

3D Paint is currently able to execute with a 0.5 second time lag between the artist moving the pen and the stroke being portrayed in the MATLAB GUI. The image below portrays this time lag. Here, the artist moves the pen at 4.0 seconds (gridline). However, the curve does not change direction until around 4.5 seconds.

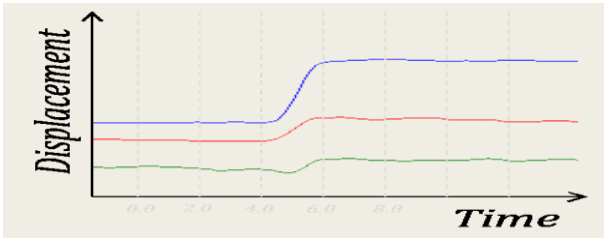

Fig.20 Response time

MATLAB is not well suited to operating in real time. Shifting some of the signal processing onto the ATmega32 helped improve the speed incrementally.Our device avoids the pesky dangers related to radio frequency or electromagnetic transmission. Hence, we focused on the usual issues of electrical insulation and grounding. While constructing the hardware, we ensured minimal strain on wires and pins to prevent unanticipated electrical shorts. If hooked up incorrectly some of the ICs (particularly the MOSFET driver) may generate high amounts of heat.Due to the limited range and directionality of our ultrasonic transmitters and receivers, undesired interference due to other devices (especially those deploying ultrasonic waves) is unlikely if conflicting devices are well isolated.There are no hazardous components, and short of ripping the circuit apart, few possibilities for injury to the artist. Operation of the device literally only requires the artist to press a button on the pen and select a few options from a MATLAB GUI.

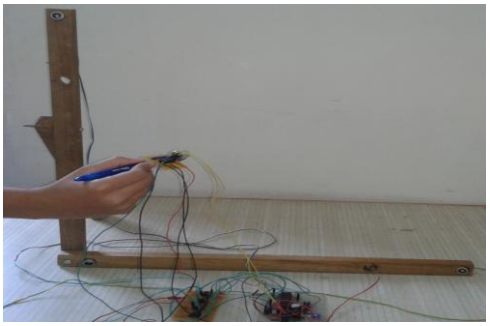

Fig.21 The artist's point of view

IV. CONCLUSION

This work has presented a novel 2D-to-3D conversion algorithm. Through our project we are trying to design a three-dimensional paint program to make three-dimensional drawing process simpler. This system consists of hardware-wooden frame and pen, a microcontroller, and a PC running MATLAB. Use of the AVR ATmega32 microcontroller makes high speed execution and efficient C code compilation. Wooden frame allows accurate positioning of sensors which helps find the position of the user's pen. Ultrasonic sensors locate the position of the ultrasonic transmitter pen and MATLAB is used as GUI.Our project will make the three-dimensional drawing process user friendly and less complex. Capable of generating a comfortable 3D effect, the proposed algorithm is highly promising for 2D-to-3D conversion in 3D applications.

ACKNOWLEDGEMENT

We are thankful to our esteemed principal Dr.S.D.Shirbahadurkar, Principal, Padmabhooshan Vasantdada Patil Institute of Technology, Prof.S.D.Joshi, H.O.D, Electronics and Telecommunication Dept., project coordinator Prof. S.S.Kendre, our project guide Prof.H.N.Patil and all department staff with the help and constant advice of whom this project is success. Lastly, we express our thanks to the internet-various websites and e-books which provided unending guidance.

REFERENCES

[1] Nelson S.H. Chu and Chiew-Lan Tai Hong Kong University of Science and Technology "Real time painting with an expressive virtual Chinese brush" in IEEE Computer Graphics and Applications, September-October 2004

[2] Chao-Chung Cheng,Chung-Te Li, and Liang- Gee Chen "A novel 2D- 3D conversion system using edge information" in IEEE Transactions on Consumer 1744 Electronics, Vol. 56, No. 3, August 2010

[3] Daniel F. Keefe,robert C. Zeleznik,David H. Laidlaw "Drawing on air: input techniques for controlled 3D line illustration" in IEEE Transactions on visualization and computer graphics, vol. 13, no. 5, September-October 2007