COMPUTER GENERATED HOLOGRAPHY

Sponsored Professor: William Ames

Canh Ngoc Cao

I. INTRODUCTION

Three-dimensional (3D) displays have been used and applied widely in computer

II. INVESTIGATE APPROACH

A common approach is a ray-tracing method in which the contribution from each object point source is computed at each point in the hologram plane. This method requires one calculation per image point per hologram sample, so it is very slow. One of my goals in this project is to optimize the algorithm for the improvement of speed. I have discussed with Professor William Ames about my thesis, and we decide to focus on the computation of off-axis transmission holograms.

III. INVESTIGATE EQUIPMENTS ON CAMPUS

The hologram must be produced at very high resolution because it is not a recording of a focused image as in photography, but the recording of the interference of laser light waves that are bouncing off the object with another coherent laser beam. This is also true for the computer generated holographic images. A computer-generated hologram will be stored in a tiff format file. This tiff file then will be printed on film at high resolution. The two places on campus that can provide us with such equipment are the Interactive Multi-media Lab (IML) and Audio Video (AV). We created a series of test images, that are uncompressed tiff

Also, we need to assemble some equipment (laser, lenses, etc) for displaying the holograms that we produced. The equipment can be bought or the physics department can let us have access to their laboratory.

IV. GENERATE HOLOGRAM

Finally, my main goal for this thesis is to generate the actual holograms on computer. I will implement the computation of off-axis method and the ray-tracing approach. The program, that I implemented, is capable of generating holograms of points, lines and polygons. The speed of my program must be taken into consideration. W-13 (g(t) 3 (3 ( ) -r))(s)(.) e end
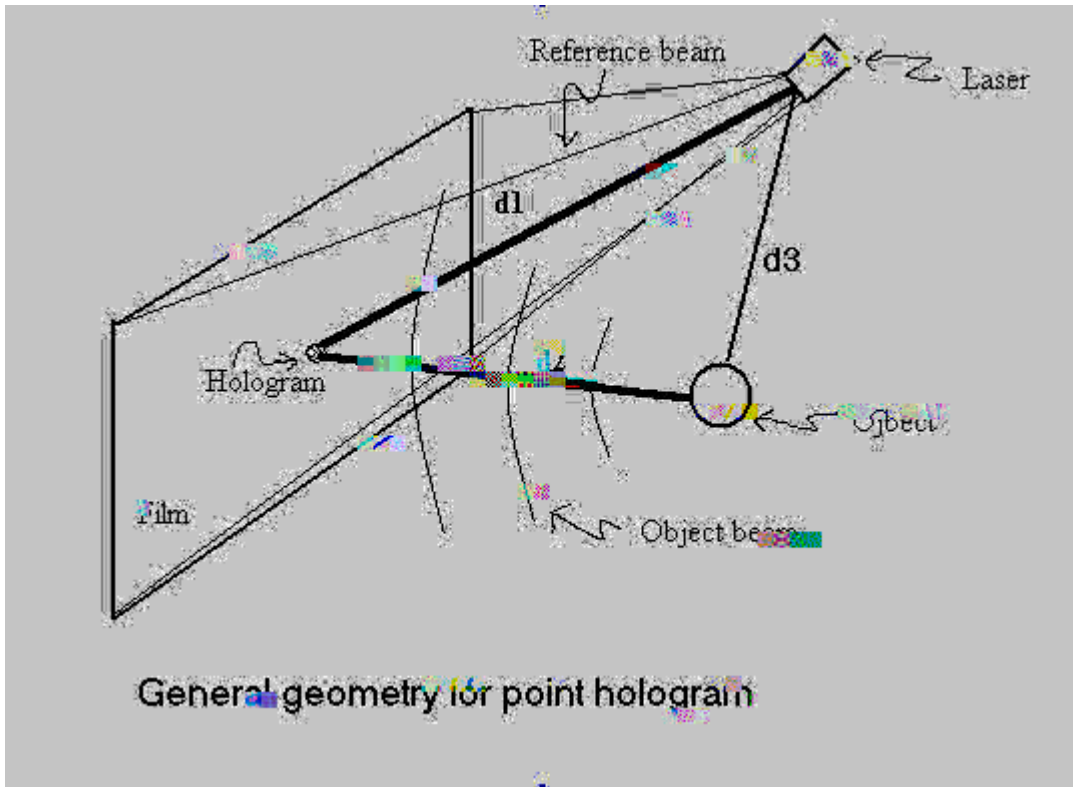
Figure 1

From the general geometry for making the point hologram, we can derive a mathematical

The distance from object **d1**, and reference beam **d2** and the distance from the reference source to the object **d3** can be computed with a simple C function. We can represent the position of the reference beam and the position of the object in three-dimensional coordinate (X,Y,Z). The coordinates of the image plane are also known therefore the calculation of **d1**, **d2,** and **d3** can be done with the following formula:

Since the magnitude of the wavelength is also known, I wrote a C++ function that computes the phase:

```
float point::phase(point3d imagepixel, point3d Refbeam, float WaveLength)
{
// Point is the coordinate of the object
```

The phase function will be called at each pixel of the image plane. The following point-hologram is produced by the above function:
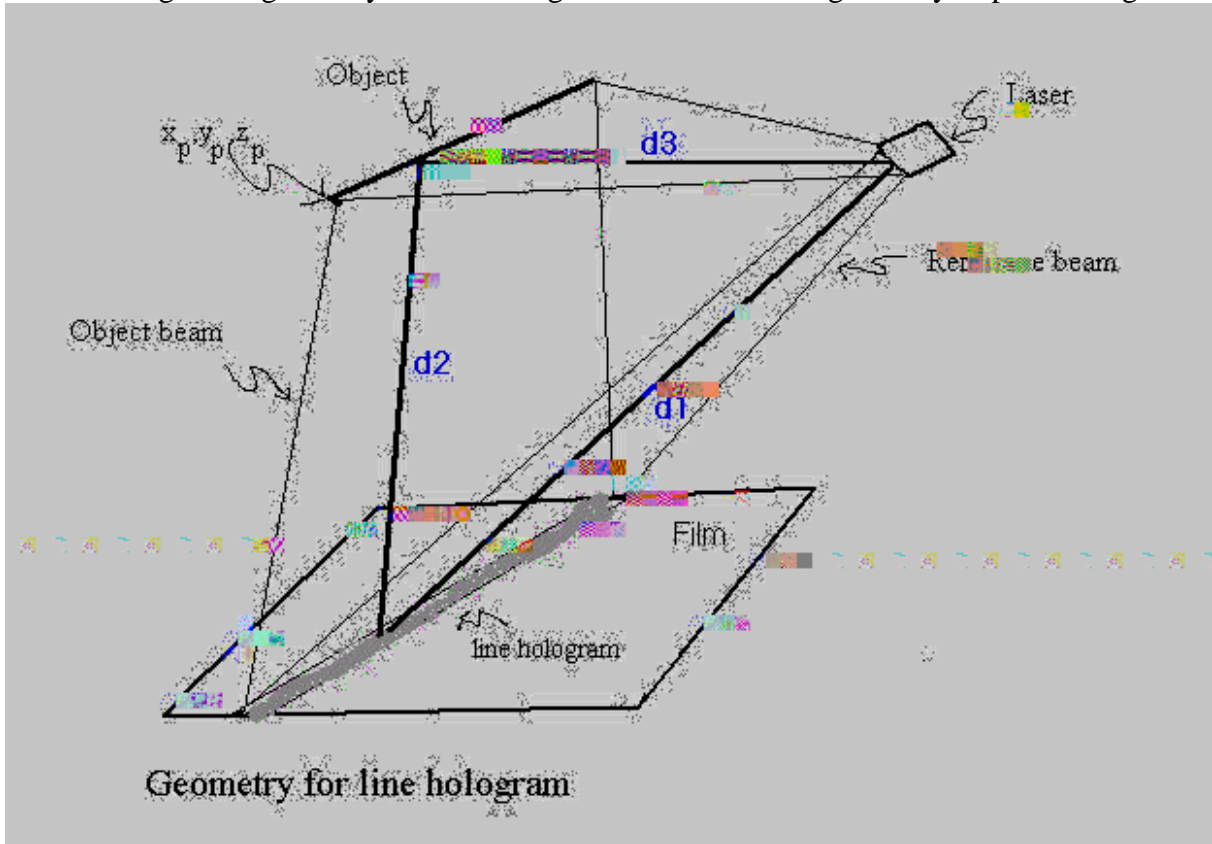
Notes: The below picture is cropped from the actual hologram because the actual dimension of the point hologram is 4096x2730 which is quite large.
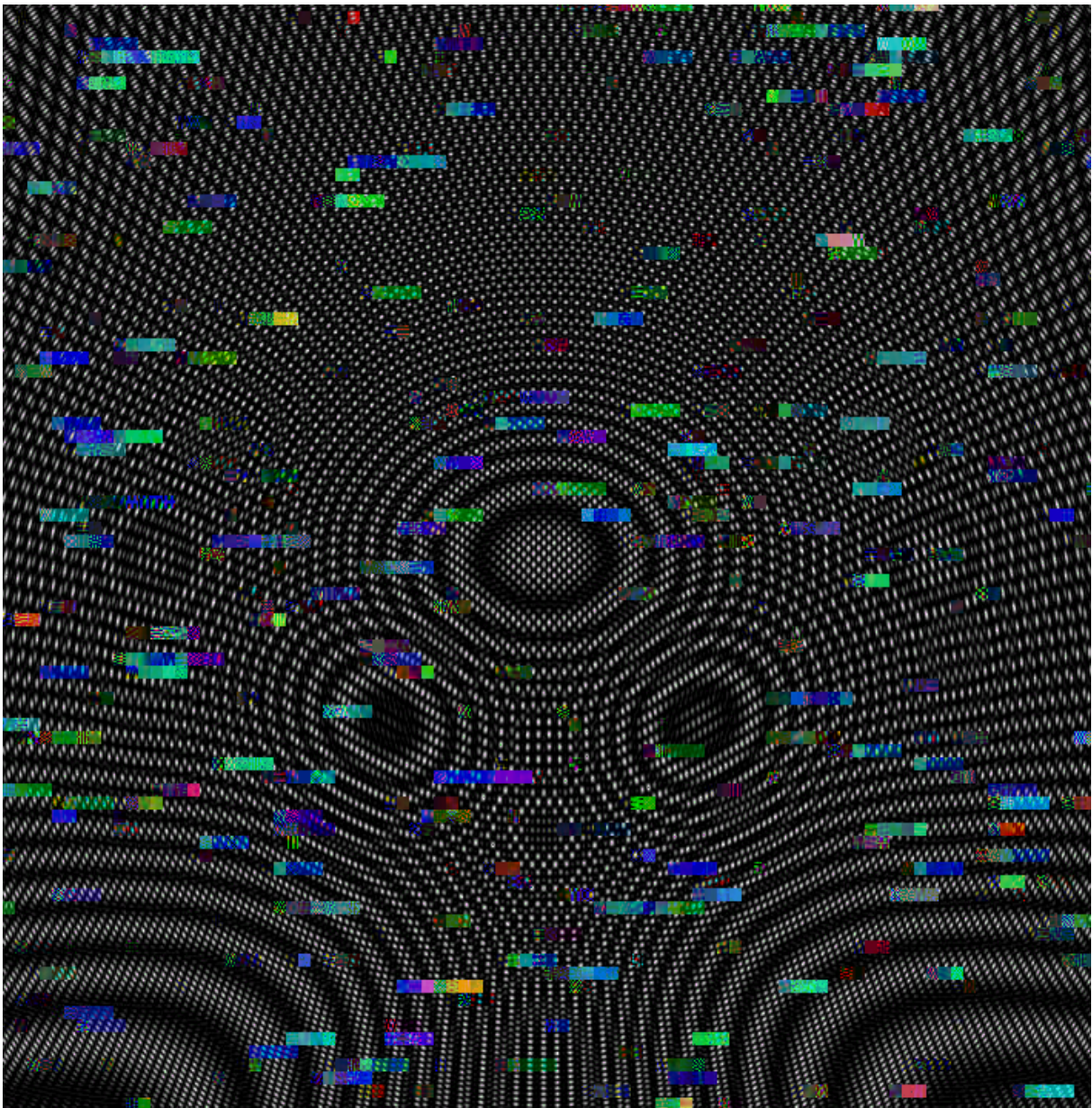
A point hologram

2. Line Hologram

The general geometry for line hologram is similar to the geometry of point hologram:

Figure 2

We now can examine the numerical computation of a line holographic pattern, beginning with the simple physics of point-source light propagation. We consider the position of the hologram at the $z = 0$ plane, and the hologram plane contains horizontal x-axis and vertical y-axis. Every point of the object, i.e. the line, emits light from position (x_p, y_p

Where t = 1/Numpoints, and Numpoints is the number of points between the two given end points that virtually form a line.

The distance from the points on the object to the hologram **d2**, and the distance of the reference beam **d1** and the distance from the object to the reference source **d3** can be computed similar to the point hologram. The computational algorithm of the phase for the line hologram differs from the point hologram's. At each pixel on the hologram, we also need to compute all the points (X,Y,Z) that make up the actual line. Since the wavelength is a constant, we can form an equation that represents the phase of light:

Phase of light = Cos((d2 + d3 – d1)/Wavelength)

```
// This loop will calculate all the points that made up the line
// the number of points must be given, we choose 100 points
 for (t = 0.0f; t <= 1.0f; t+=tincrement) {

  // compute the all the points that made up the line
   float x = point1.x + t*(pdx);
   float y = point1.y + t*(pdy);
   float z = point1.z + t*(pdz);

  // the distance from the line to the image plane
   dx=x-imagepixel.x;
   dy=y-imagepixel.y;
   dz=z-imagepixel.z;
  distance = sqrtf(dx*dx + dy*dy + dz*dz);

  // distance from the line to the reference source
   dx=x-Refbeam.x;
   dy=y-Refbeam.y;
   dz=z-Refbeam.z;
   float distance2 = sqrtf(dx*dx + dy*dy + dz*dz);

  // the phase of light of the line is the average sum of all the phases of points

   sum += cosf((distance+distance2-refdist)/WaveLength);
  }

  return sum;
}
```
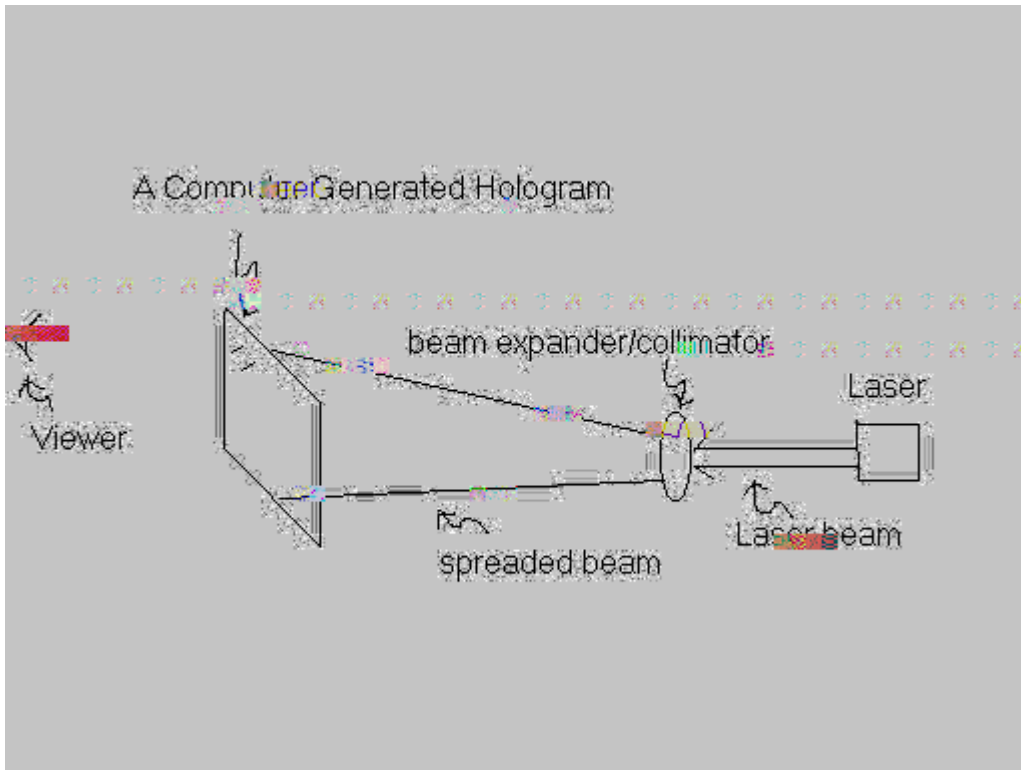
Figure 3: Viewing Computer Generated Hologram

4. Holograms are stored in TIFF format

At the beginning of the thesis, I produced a Computer Generated Hologram by emitting the hologram's pixels into a PGM (Portable Graymap) file format. Since I wrote my program on the Computer Science machine of Boston College, I was able to examine the outcome of the hologram with XV. However, the process used to produce the results, that can be viewed in three-dimensions, was to photograph the masks using a 35mm camera and slide film. The slides are illuminated with a laser to produce the projected hologram. The equipment, that can accomplish such task at Boston College's Interactive Media Lab, requires the file format to be in uncompressed TIFF file format. I discussed with Professor Ames, and we decided to write our own TIFF library based on the TIFF 6.0 Specification. The TIFF

library, that we wrote, will store uncompressed images. Due to nature of our TIFF library, the size of the Computer Generated Hologram is quite large. A line hologram with resolution of 4096x2730 occupies about 10 Megabytes of hard drive spaces.

5. Optimization

The running time of each Computer Generated Hologram is different depending on the resolution of the image. In fact, the running time is proportional to the image width and height. For example: the running time for a hologram of 4096x2730 will be proportional to the product of 4096 times 2730. We observed the running time of both the point hologram and line hologram programs, and we noticed that the line hologram program had the most running time. We examined the codes and found out that since the phase function of the line

$$D = \frac{D + \left(\dfrac{x^2 + y^2 + z^2}{D} \right)}{2}$$ Using this algorithm, we are able to reduce the running time of the line hologram program to 30%. The following line hologram is produced with our new square-root algorithm:
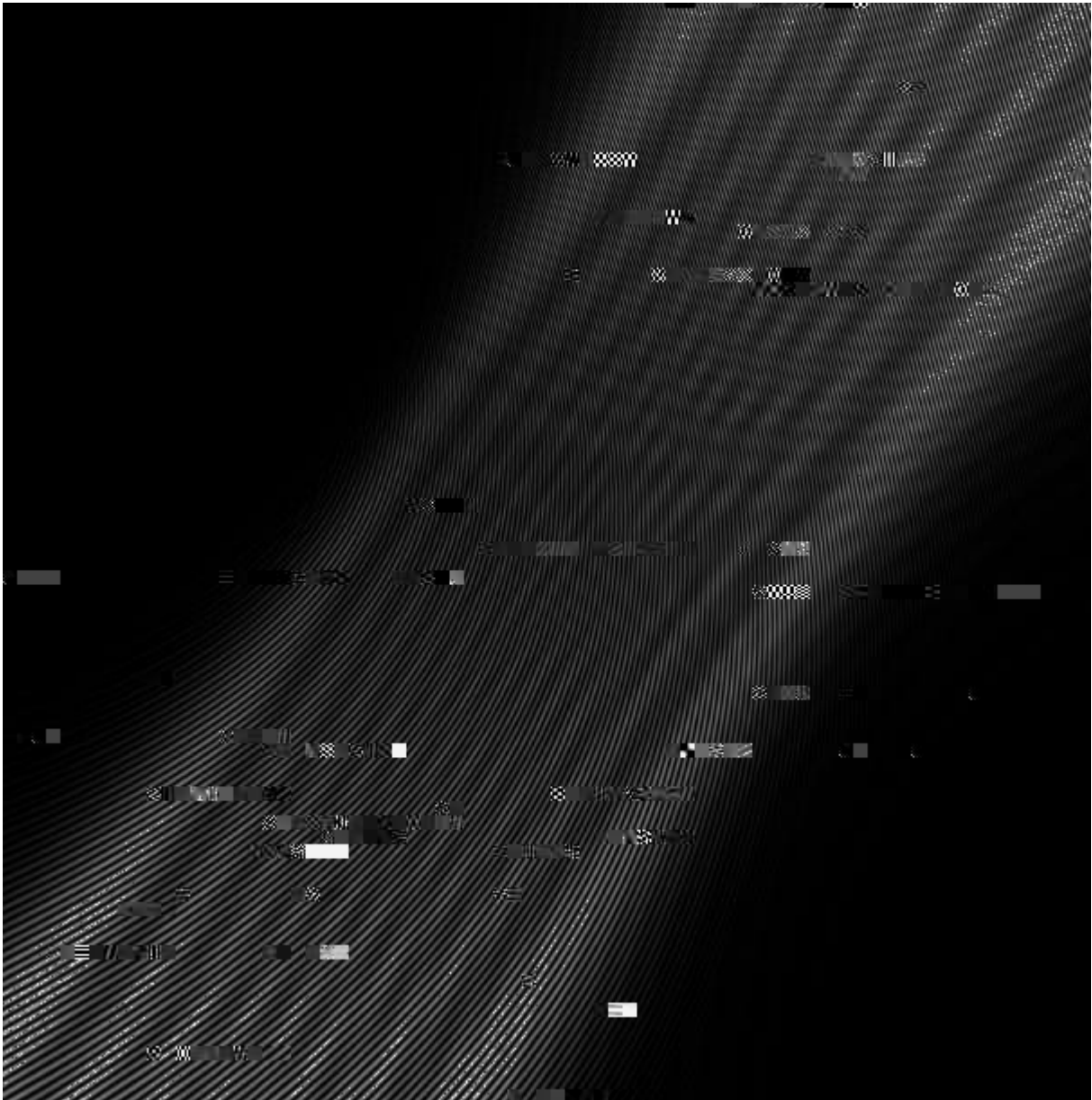
We also investigated in writing our own cosine function that uses the technique for iterative approximation. We got the negative result because Profession Howard Straubing pointed out that eventually the approximating cosine values would diverge throughout the loop. After failing the first attempt to modify the cosine function, I tried to apply the Mclaurin series to compute the cosine function. The formula is:

$$Cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \frac{x^{10}}{10!} + ...$$

This method also failed to improve the speed of the program if I used more than 10 polynomial terms. When I used fewer polynomial terms the approximating cosine value is not accurate enough.

IV. CONCLUSTION

Computer Generated Holograms are still at the beginning stage, and so much more can be done. My thesis can be improved in future if I can create a Computer Generated Hologram using an OpenGL library. The advantage of OpenGL is that it is possible to read the Z-buffer of an image. Using the Z-buffer values, it is also possible to compute the dimensions and depth cues of the image, thus more realistic holograms can be produced.

The applications of Computer Generated Holograms are many. Once the technology for producing interactive holograms can be done in real time, we can watch television in three-dimensional view. Teachers can enhance their lectures by showing an interactive hologram of any object and model. For this reason I felt that Computer Generated Hologram can be viewed as an effective teaching tool.

V. ACKNOWLEDGEMENT

My thesis has involved much time spent on researching the topic of Computer Generated Hologram and implementing computer program. I greatly appreciate the time and effort that my sponsored professor, William Ames, spent to make the final result possible. I have learned a great deal from my thesis. I also would like to thank Professor Howard Straubing for pointing out the problem of the approximating cosine function. Finally, I thank

Bibliography:

1. TIFF 6.0 Specification Coverage by Sam Leffler / sam@engr.sgi.com

2. Optimization of Hologram Computation for Real-Time Display
   By Mark Lucente, MIT Media Laboratory published in SPIE, Bellingham, WA, Feb. 1992

3. Optical Information Processing and Holography
   By W. Thomas Cathey, University of Colorado, Denver, Colorado

4. CRC Standard Mathematical Tables 25th Edition
   By William H. Beyer, Ph.D