Introduction

Tomographic imaging is a method of reconstructing images from their projections. While tomography is most frequently used in diagnostic medicine, it can also be used to capture images. While a conventional digital camera uses an array of sensor elements to measure the brightness of each pixel, a tomographic digital imager (or tomoscope) measures the brightnesses of multiple sections, or slices, of the target and then reconstructs an image using that data.

Purpose

The aim of the project, Digital Telescope Utilizing Tomography (DTUT), was to develop a tomoscope as described above and test its ability to image simple shapes and figures from a distance.

Background and Rationale

The mathematics behind tomography originated from a paper by Radon in 1917 on how to reconstruct a function from its projections. Today, tomography has many applications, such as medical diagnosis, the mapping of underground resources, imaging in the field of non-destructive materials testing, and generating three dimensional models using electron microscopy (Kak and Skaney, 1988). By far, the most common application of tomography is in medicine, where it can be used to create models of internal organs without performing a dissection. It is used in CAT (computed axial tomography) scans, MRI (magnetic resonance imaging), PET (positron emission tomography), and ultrasonography (Wiki, 2004).

While all of these use different forms of data acquisition, they use the same tomographic reconstruction techniques. The two most commonly used techniques are filtered back projection (FBP) and iterative reconstruction (IR) (Wiki, 2004). FBP uses the data to re-project each projection back onto the image, whereas IR repeats a function that makes the image look better.

Both methods are a compromise between image quality and computation time. Neither is perfect, but both can produce photographic quality images.

While tomography may be impractical for personal photographic uses, it may be feasible for applications in fields such as astronomy.

Science Literature Used

In general, tomography refers to the collection of techniques used to reconstruct an image from a collection of projections (see Figure 1 in the Appendix) (Kak and Skaney, 1988). While projections refer to the integral of a function at a given angle, the term is often used to describe a broad range of information derived from emitted energy interacting with a target (Kak and Skaney, 1988). Kak and Skaney's Principles of Computerized Tomographic Imaging was used by project members as a basis of the engineering and mathematics, providing insight in design and algebra; although project work was similar, the book mostly provided background. Many different types of physical phenomena have been utilized in tomography, such as visible light, microwaves, x-rays, nuclear magnetic resonance, and ultrasound (Wiki, 2004).

Fundamentally, one source of inspiration for DTUT was the process of “Computed Axial Tomography” (CAT). Each scan was created through a series of two-dimensional back projections taken around the object at multiple angles. CAT uses the Fourier slice theorem, also known as the central project theorem. Basically, the theorem proves that the Fourier transform, which was the breakdown of a function into sinusoids, was also identical to the normal central slice plane to the object (Hoffman). The image could then be calculated from the projections. Although the project looked at Fourier's method and FBP as a guide, the techniques used were different.

Another method that the team analyzed for the project was X-ray crystallography. In X-ray crystallography, an image was constructed from “slices” of the actual object. X-rays are

directed at crystals (because crystals amplify diffraction) and the diffraction patterns are recorded and used to calculate the image (Rupp). The Fourier transform is again utilized to create the image from the intensities of the diffractions. However, as mentioned, the team decided to forgo the conversion of the method to code and used instead the alternative method detailed.

Methodology

The tomoscope consisted of a mechanical component and a software component. The mechanical component collected data while the software component reconstructed an image from the data. Through the course of two months, many data samples and pictures, four of which used the final methodology and are displayed in the Appendix, were collected. They were all performed at night, approximately one to two hours after sunset outside. The images produced were thirty two by thirty two pixels and had 256 colors in grayscale.

The Mechanical Components of the Tomoscope

The function of the mechanical component of the tomoscope was to collect data that indicated the brightnesses of different slices of the target. It consisted of three parts: a target which was going to be imaged, a spool unit which held the light sensor, and a rotating disk which limited the view to a single slice.

The targets consisted of large pieces of cardboard with various shapes and figures cut out of them. The holes were covered with sketching paper to diffuse the light. The target being sampled was then mounted between two hurdles a few feet in front of a car. The car headlights illuminated the shapes and figures from behind, creating a target to be imaged.

The spool unit functioned to allow the sensor to be positioned correctly. The spool was a ten inch radius cable spool with holes drilled at the extreme of the radius to mount the sensor (see Figure 2). The spool rested on a mount which was suspended by platform which could be

moved up and down. The spool was arranged so that the center was at the same height as the target and the sensor was pointed towards it.

The sensor consisted of a PVC pipe painted black with a electronic sensing element attached to one end. This made the sensor's field of view limited so that some of the background light could not be detected. The sensor element was connected to a circuit board which recorded its output onto a memory card.

The spool was rotated to put the the sensor at twenty one different positions. The sensor was kept at each position for approximately thirty seconds so that many samples of each slice could be averaged to reduce noise. The rotating disk limited the sensors view to a single slice at a time.

The rotating disk consisted of two semicircular pieces of foam board with a gap in between, spun by a stepper motor (see Figure 3). It had two hundred steps per revolution and rotated at a rate of two revolutions per second. The stepper motor sent a signal to the circuit board identifying every time it took a step so that the angle of the slice for each sample could be later calculated. The disk was placed in between the spool unit and the target, approximately ten feet from the spool unit.

The result of these combined parts was that the sensor element would record many samples of the brightnesses of each of the 3804 different slices at each position (see Figure 4).

The Software Components of DTUT

The components are listed in the order that the data feeds through them and was processed to create the final image. Once the mechanical part of data collection was completed, the raw data was processed. The “raw data processor” controlled the data recording on the circuit board through a PIC processor. “Data analyzer” was then implemented to package the information outputted from the “raw data processor” into data structure easily accessed by

subsequent programs. The analyzer outputted the averaged brightness of each slice in each position. The “center of brightness locator” identified the position of the target. Next, “slice reconstruction” was employed, which processed the position of the target and the brightness of each slice. The output of “slice reconstruction” was the portion of each pixel covered by each slice, then followed by the brightness of that slice. “MatrixSolve” and “Iterative Gradient Method Image Generator” processed the output of “slice reconstruction” and produced an image, using an algebraic reconstruction method which treated the data as a system of linear equations. Both methods of image reconstruction utilized and processed the system of linear equations in different ways. Ultimately, the raw data, provided by the mechanical data collection, was processed into an algebraic system of linear equations and solved for the best solution of the system.

Raw Data Processor

A program on the PIC processor in the circuitry in the mechanical component recorded the raw data file. For each data sample, the file contained sixteen bits. The sixteen bits contained the outputs of the light sensor, the motor driver, and the "good data" switch. The output from the light sensor was simply the amount of light hitting the sensor element. The motor driver data indicated when the stepper motor took a step. The output of the "good data" switch simply indicated whether the switch was pressed at the time of that sample, which identified the sample as valid or invalid. Invalid data was data collected when the sensor was being moved from one position to the next. Valid data was data that was going be used in the calculations.

Data Analyzer

The function of the data analyzer was to process the raw data so that it could easily accessed by other programs. It read in a raw data file and outputted a processed data file.

The analyzer averaged the brightness data for each angle within each valid position. It used the output data from the motor to calculate which angle each brightness data corresponded to. It used the output data from the switch to distinguish different positions and identify valid data.

Center of Brightness Locater

The function of the center program was to allow the user to visually identify the position of the target in each position and insert this into the data file. It was necessary to know the angle of the target in each position in order to reconstruct an image. The position of the target was approximated by having the user identify the center of the brightness curve for each position. The center program read in a processed data file and outputted another processed data file.

The program displayed the averaged brightness data for each position on the screen. The user then moved a cursor to select where the center of the bump in the data corresponding to the target was. At this stage, the user could also choose to delete data which looked inconsistent.

Slice Reconstruction

In order to translate the data acquired to the equations needed to perform the aforementioned matrix operations, it was necessary to have an intermediate program to create the equations. This intermediate program involved the determination of the fraction of each pixel, the weight, covered by any slice. The weights in a given slice, multiplied by their brightnesses add up to the brightness of the slice. For example, an image cut at 45 degrees would have largely zeros for areas cut on the fringes, with only the pixels on the center diagonal having large fractions, but all set equal to a relatively large brightness:

$0*p_1 + 0*p_2 ... 23/81*p_{509}+11/81p_{510}...0*p_{1024} = 89$

where p_n was the brightness of the nth pixel.

The matrix program took in all the equations generated and used them in its calculations to create the image.

The program read in the brightness of the slice at each angle in that the slit that sweeps over increments of 360/1902. It used this data with the given position of the image to determine the area covered by the slit of each individual pixel. The method used was an approximation- each pixel was represented by eighty one dots. The program checked to see how many dots are contained within the two lines of the slit, recording the number out of eighty one as the area covered.

To visualize this method, consider Figure 4 in the appendix.

The resulting data, the amount of each pixel covered and the brightness of the corresponding slice were outputted into a file. Extraneous information such as slices where no pixel was enclosed, was removed from the file.

Matrix Solve

The MatrixSolve program was written to accomplish the last phase of the data manipulation—using the output of the Slicer program to solve for the optimal value of brightness for each of the 1024 pixels.

The core of this final program involved the least-squares solution method of linear algebra whereby a matrix equation A*x=b that possessed no solution was transformed into the system A'*x=b' that was guaranteed to possess solutions that are optimal to the original system. $A'=A^T*A$ and $b'= A^T*b$. This method was very similar to creating an error function and then setting the gradient vector components of the function to 0 to find the solution that minimizes the error function.

From then on the program first transformed a matrix containing the equations into reduced-row echelon form and then checked to determine whether the array was singular,

indeterminate, or determinate. The array never was singular, while if it was indeterminate, the program prompted the user to input seed(s) that will give all the variables a definite value. After the seeds were set, the program reran through the reduced-row echelon process during which, to reduce computational errors and prevent solution values from exploding in magnitude, the program "purified" unconditionally all insignificant values to zero at the end of each while loop the value was less than a predefined error limit.

The variable error was set optimally between 10^{-1} and 10^{-20} , and could changed to be dynamically decreased/increased during each program cycle. Then, the program normalized the solutions from zero to a specified number.

Finally the image was reevaluated through two parts: changing the contrast of the image, and setting a filter to eliminate most background noises. When the program terminated, a file of the image values was created. Although this program calculated the numerical values of the results for verification, graphic results were obtained from the next program.

Iterative Gradient Method Image Generator

The function of the gradient program was to generate an image from the complete data using an iterative method. It was similar to the method developed by Kaczmarz for finding the best solution to a system of equations and similar to Newton's method for finding the roots of an equation. It read in a complete data file and outputted an image file.

The gradient method generator used vector calculus to calculate how each pixel's brightness should be changed to create an image that fits the data better. The error was defined to be the sum of the squares of the differences of the brightness of each slice computed using the current image and recorded in the data file. The error could expressed as a function of the brightnesses of each of the pixels in terms of the measured brightnesses of each slice and the weights of each pixel in each slice. If the error was E, the brightness of the ith pixel was p_i, the

weight of the ith pixel in the jth slice was w_{ij}, the measured brightness of the jth slice was b_j, and n was the number of pixels then the error function can written as:

$$E(p_0, p_1, p_2 \ldots p_n) = \Sigma_j(b_j - \Sigma_i(w_{ij} * p_j))^2$$

By treating the error as a function of the brightness of each pixel, an n+1 dimensional surface could created. Each point on the surface corresponds to a unique image. It can easily be shown that this surface was concave upwards at every point in its domain, which implies that the error has an absolute minimum.

Since the gradient vector, *f*, of the curve at a given point points in the direction of greatest increase of the value of the function, the vector pointing opposite the gradient vector points in the direction of greatest decrease. It could be shown that the component of the gradient vector parallel to p_c, f_{pc}, was given by:

$$f_{pc} = \frac{dE}{dp_c} = -2 * \Sigma_j(w_{cj} * (b_j - \Sigma_i(w_{ij} * p_j)))$$

Therefore, by subtracting a sufficiently short vector that points in the same direction of the gradient, the point would move towards a minimum error. This corresponded to an image that would fit the data better after each iteration, and theoretically resemble the target more.

Results

The numerical results produced from the Gradient program revealed that the program was working satisfactorily as the errors appeared to be following steadily for the first few thousand cycles until the rate of decease asymptotically approached zero and the actual error approached the unknown minimum error. The data produced by the MatrixSolve program were extremely large for unclear reasons. What was curious was the phenomenon of increasing error with decreasing answer range. When the Matrix Solve program was fed more data than the minimal needed to gain accuracy, the errors steadily increased while the range of the answers lowered to

[–20, +30] from an original [–6000000, +7000000]. What was more interesting was the fact that the maximal and minimal values were always approximately equal in magnitude and that the Gradient program actually produced similar results initially when ran long enough, but restrictions were added to the program forcing values to remain non-negative.

The visual results were overall somewhat unsatisfactory (compare the digital photographs of the target with the actual results in the appendix). The images lacked clarity and crispness. Refer to the Analysis Section for a detailed explanation and Appendix for the pictures.

Analysis

Upon observation of the reconstruction of the triangle, it was evident that the reconstruction was significantly similar to the true picture of the triangle (see Figures 5 and 6). The true representation of the triangle had nearly uniform brightness covering the center of the object, while two of the three corners of the triangle were dim. A streak of enhanced brightness also marked the right side of the triangle (within the already bright portion). The reconstruction reflected this distinction. The bottom right corner, as well as the top corner of the reconstruction, was significantly darker than its surroundings. Despite such close resemblance, the reconstruction did not produce straight edges, and the reconstructed triangle possessed some aberrations, such as the strings of brightness seemingly spewing from its left and bottom side.

The true picture of the nested box had uniform brightness for the bottom three-quarters of the image (see Figures 7 and 8). The top quarter was significantly dimmer than its counterpart. The reconstruction seemed to be subject to a significant amount of background noise, as the picture was adulterated with random streaks of brightness. The reconstruction seemed to resemble the brighter portion loosely, yet the upper, dimmer portion was seemingly omitted from the reconstruction. The reconstruction did not appear to have uniform brightness; differing levels of brightness were evident. Furthermore, edges were apparently not straight in the

reconstruction, a distinction necessary for the distinguishing characteristic of a box. The two significant bright (lower) portions were divided (in the reconstruction) by a sort of partition, which resembled the actual image that was reconstructed. The data collection device could have lacked the sensitivity necessary to generate edges; the actual light emitted could have been too dim as well.

The reconstruction of the array of boxes bore slight resemblance to the true picture of the boxes (see Figures 9 and 10). The reconstructed image faintly resembled the shape of boxes. The second row of boxes seemed to be present in the reconstruction. However, the true picture had uniform brightness within all the boxes; this essence was not evident in the reconstructed image. The leftmost column of boxes appeared the brightest in the reconstruction. The first row did not appear to be any sort of collection of boxes. The spewing string phenomena were evident in the reconstruction (as in the triangle, previously noted). Perhaps the data collection mechanism did not have such sensitivity as to distinguish between different distinct "boxes" of brightness; the reconstruction appeared as one large strand of brightness.

The "smiley" image represented two circular regions above a curved region (see Figures 11 and 12). The curved region was considerably brighter than the two circular regions that rested above it. The curved region was uniformly bright, except for the rightmost corner, which appeared slightly fainter. Such a distinction was displayed in the reconstructed image, as the rightmost curved portion appeared darker than its counterpart. The curved shape in particular of the reconstruction was remarkably similar to that of the true image; when plotted against each other, the shapes followed a tight correlation**.** The reconstructed image never generated the two dimmer, circular regions above the curved region. The data collection device may not have been sensitive enough to collect such a faint light source. Furthermore, the circular regions could have

been too insignificant in size for detection by the data collection mechanism. The emission of light from the circular region could have been relatively too dim for detection.

All images were subject to background noise, discussed in mechanics (data collection process). Aberrations may be generally attributed to background noise, along with the inevitable error encountered with the manual selection of the “brightest slice”, designated in the data processing program, as well as in all software.

Error Analysis in Software

The center of brightness identified by the user may not have been the actual center of brightness of a given position. Also, the centers of brightness for each position were not guaranteed to represent the actual center of brightness on the image. This was because the slices are not quite parallel. This would have caused the calculations that involved the position of the image to be off and caused the equation set to not have reflected the original image.

One possible area of error lied within the slice area calculation program. The program utilized a simple approximation of the area covered of each pixel by a slice by using dots. The accuracy of the calculation would increase exponentially by each expansion of the dot square. However, the base number of dots used in the calculations was eighty one for the sake of speed of calculations. Therefore, the error could be minimized by the increase of the number of dots used in the calculation, resulting in a better approximation.

Errors could have also arisen from the MatrixSolve program, stemming from the huge number of computation involved. Although the program used long eighty bit precision, several billion flops (computations) was enough to skew the answers up to a considerable degree. The main problems lied in the fact that the program was done in essentially one large chunk, and was not recursive to any degree—thus, the errors tended to accumulate all through program execution until the very end, while in a recursive algorithm, the error was made specifically to degree with

each program loop. Part of the error resulted from the fact that tiny abscess values grew from the accumulation of error during primarily the numerous division operations done. These tiny values eventually causes all other values to explode in magnitude once they are the divisor, thus a purifying subroutine was added to check for and correct these values. However, the lost significant digits generally added up enough to produce an error on the order of 10000-20000. (NOTE: because of the large error produced, this program was not used to produce any actual pictures and still remains in the formulation/revision stage. The results were often too random to produce a coherent or decent image.)

In the iteration grapher, after any number of iterations, the error will not be have reached the minimum value. The error approached it as the function was iterated, but it would never reach it. The quality of the picture was limited by the amount of processing power and time available. Futhermore, there was no guarantee that a lower error corresponds to an image that resembles the target more. Smaller numbers of iterations tended to produce blurry pictures, but many iterations produced artifacts (anomalies in the image created by small discrepancies in the data).

Error Analysis of the Mechanical Components

The inaccuracies in the data due to mechanical issues were a major contributor to error in the results. The most significant source of error was the background noise present in the brightness data. The presence of inconsistent extraneous light at the site of image capturing made the data not correspond entirely to the target. The sensor element also had noise in its measurements which were inherent to the type of sensor, which would similarly make the brightness data not correspond to the target.

Other error could have come from misalignment. If the sensor tube was not aimed directly at the target or was not at the correct position, the sensor's field of view may not have included the entire target. This would make the data not correspond to the target.

Further error came from people uninvolved in the experiment inadvertently passing in between the sensor and the target. This disrupted the data collection process and skewed the data by preventing the sensor from collecting light.

Conclusion

This projects has produced images which resembled the targets, but suffered from several significant sources of error. The images assembled are in the right direction toward the path to perfecting this tomoscopic method, but fell short of perfection in their lack of crispness, solidity, and resemblance to the ideal images. The mechanical and software errors had accumulated to thwart this project from achieving more accurate results.

Several questions remained to be answered from this extended project. When the Matrix Solve program was originally applied to artificially created data of a big square, it produced a solution with a smaller error than the algorithm process, but nevertheless, the final image did not look like a square at all--in actuality, the image resembled a diamond with the four corners of the image dim and center bright—an anomaly repeated for real data as well. Another question lied in that when MatrixSolve used more data from the data file, the error actually increased while the range of data decreased to a more satisfactory level. Another puzzle arose from the fact that the MatrixSolve output had a smaller error but still does not resemble the actual image unlike the Gradient program. This puzzle may have been attributed to the seemingly chaotic fluctuations of the solution matrix as the final programs push it closer to the theoretical ideal solution.

Future work that could done to extend the results of the project including improving the mechanical aspects. One improvement that could be made is to improve the light sensor’s using a

more sophisticated sensor or to cool the light detector using dry ice or liquid nitrogen so that the thermal radiation from the environment does not strain the data. Also, blue, green, and red filters could be placed on the light collectors so that the three separate data in hues of blue, green, and red could collected and ultimately integrated to create a full-color picture.

Finally, to make this method actually feasible for distant, faint stellar objects, it would be necessary to position the rotating slit very far from the light collector. To accomplish, optimally a geostationary satellite would be used. The result of this addition would ideally create a high resolution high contrast picture of an astronomical body like Jupiter.

The software changes that should be implemented were revisions of the algorithm and MatrixSolve programs to allow for less errors (perhaps through artificial data types with extra precision) and for more efficiency as the time needed for computation was still proportional to n cubed where n was the total number of pixels--thus the 1000 by 1000 picture would take over a billion times longer than the 32 by 32 picture.

More work was being done to investigate the addition of Filtered Back Projection through Fast Fourier Transforms for better digital signal processing to eliminate the background noise, which received up to 90% of the strength of the actual object in the raw data. More research could be committed on digital processing applications like image compression.

As the mechanical component of the tomoscope functioned properly and created distinctively recognizable objects, the project demonstrated that it was possible in practice to construct a working tomoscopic mechanism with the detailed methods.

Appendix

Note: All source code for the software can be found at http://moonraker.0catch.com

Below are two scans from Principles of Computerized Tomographic Imaging and two sketches of the mechanical devices used.

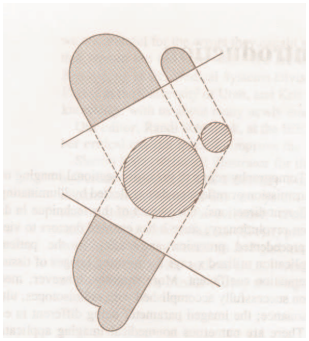


Figure 1: Image of Projections

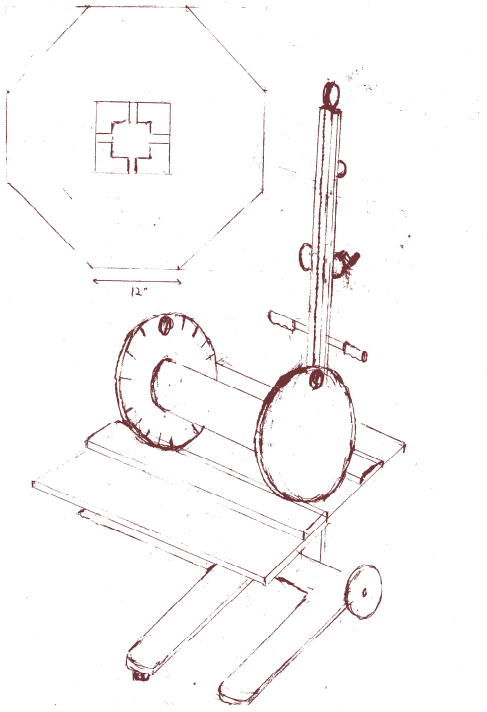


Figure 2: Tomoscope platform and image filter

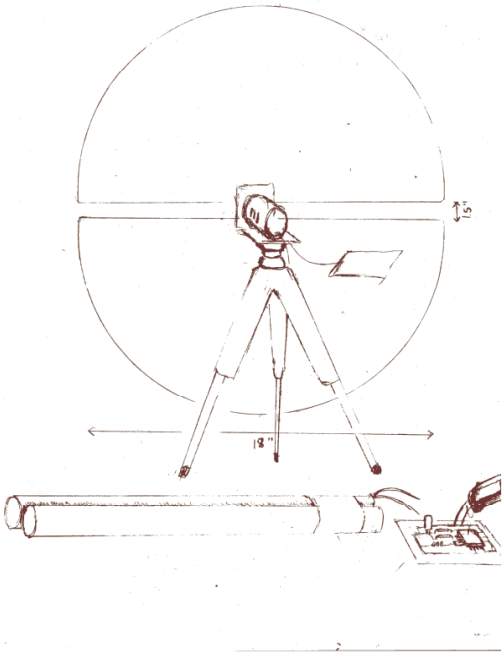


Figure 3: Setup of motorized slit and tomoscope

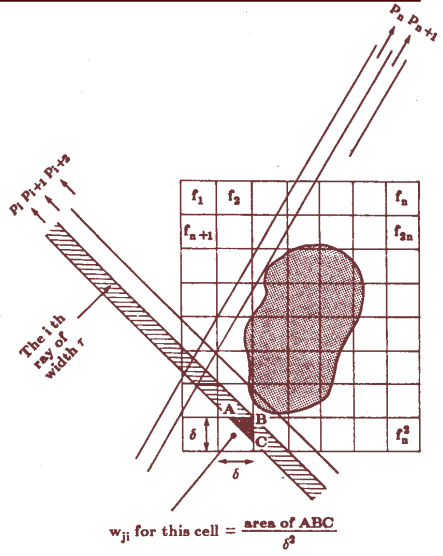
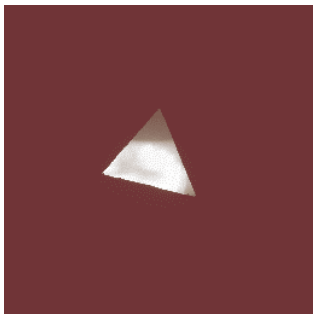
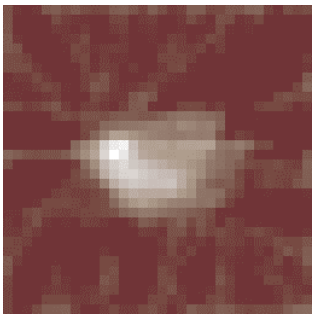


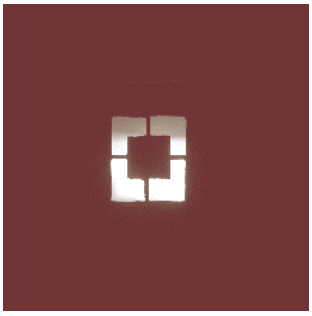
Figure 3: Representation of slices over an image composed of pixels.

Pictures on the left are photographs of the actual projections, while the generated image is on the right. Note: The generated images have been rotated during the image capturing process.

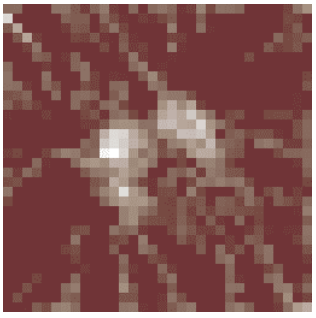


Figures 5 and 6
Triangle Image

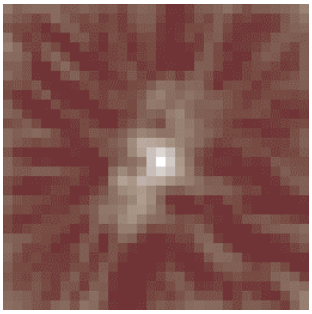




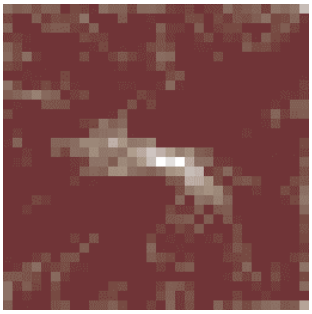
Figures 7 and 8
Nested Boxes



Figures 9 and 10
Array of Boxes



Figures 11 and 12
Smiley



Bibliography

Wiki Staff (2004) “Computed Axial Tomography”
Retrieved June 4, 2004 from
http://en.wikipedia.org/wiki/Computed_axial_tomography

Hoffman, Forrest (Undated) “An Introduction to Fourier Theory”
Retrieved September 8, 2004
http://aurora.phys.utk.edu/~forrest/papers/fourier/

Rupp, Bernhard (Undated) “Crystallography 101”
Retrieved September 8, 2004
http://www-structure.llnl.gov/Xray/101index.html

Kak, A., & Slaney, M. (1988).
Principles of Computerized Tomographic Imaging.
New York: IEEE Press

Wiki Staff (2004) “Tomogram”
Retrieved September 20, 2004 from
http://en.wikipedia.org/wiki/Tomogram

Lay, David C. (2001)
Linear Algebra, 6th Edition.
Addison-Wesley.