XRaySim user manual

Version 0.7

Koushik Viswanathan

Contents

Chapter 1

Introduction

Thank you for trying out XRaySim - the open-source X-ray radiography/ Computed Tomography simulation package. This document is meant to serve as a reference guide for the various sub-systems inside XRaySim, and also to help you get the most out of the simulation package. For the latest releases, please check the official website at http://xraysim.sourceforge.net. Documentation on the source and the various internals is also available on the website here: http://xraysim.sourceforge.net/docs.htm. Several links to other interesting information on the net are available in the appendix at the end of this document and also on the web at http://xraysim.sourceforge.net/links.htm. Bug fixing is currently done on a case-by-case basis. For additional features and support, feel free to send the author an email - contact information is again available on the sourceforge website.

1.1 Brief History

XRaySim can attribute its origins to an initial (not very impressive) undergraduate research effort. The Center for Non-Destructive Evaluation at the Indian Institute of Technology, Madras was looking to develop its own digital radiography system for industrial CT testing (which, incidentally is currently in its final stages of development). As part of that, two undergraduate students started off by writing some simulation code to help them evaluate their reconstruction algorithms. Initially, tomographic reconstruction warranted the need for a 2D simulation tool, but as time grew a generalized 3D tool became necessary. Hence, a program called 3DSim came into existence.

3DSim had very limited capabilities. It was written in BASIC using a (very primitive) DirectX wrapper, working with DirectX 8.1. There were several problems with this:

- It worked only with meshes of low to moderate complexity. This restricted its use to trivial objects like spheres/ cylinders
- It had a very limited user-interface and was largely command-line driven (despite being in BASIC)
- As a result of the BASIC codebase, most of the algorithms were highly inefficient - most of them were just random hacks put together for specific cases, the code was not general enough.

- It was written with DirectX, which limited its use to windows systems alone

3DSim served the purpose well enough. When computation time became an issue, it was decided that the entire code would be rewritten from scratch, using C++. No pre-defined libraries were going to be used for simulation computations and an emphasis would be laid on the actual algorithms and their extensibility. This soon became a central issue to its development. The new version was tweaked to support the OBJ format (the Autodesk DXF and the Alias FBX formats were also considered) instead of the DirectX .x file specification used earlier. Also, the name 3DSim presupposed that a '2DSim'existed, which was not the case. Thus, XRaySim was born. It featured a whole new user interface and was written with open-source libraries.

XRaySim was open-sourced just over a year ago with a pre-alpha release. Soon, an alpha release followed which demanded some severe bug-testing. Currently, XRaySim is classified as beta software and is at version 0.7 at the time of this document. It is written with only open-source libraries. XRaySim is released under the terms of the GNU General Public License which allows everyone to view and use the code free of cost. The full terms of the license can be found on the XRaySim website.

1.2 Pre-compiled binaries

XRaySim is open-source software and periodic releases are made on the website. The source-code repository (SVN) is not currently online, though that should hopefully change soon. The releases contain pre-compiled binaries for Windows, along with the (cross-platform) source. The latest version can be downloaded from the project website (http://xraysim.sourceforge.net/downloads.htm). Older releases are archived by the Sourceforge file release system, and can be accessed from the project page (http://sourceforge.net/projects/xraysim)

Currently, pre-compiled binaries are provided for Microsoft Windows only. They have been tested on Vista (32-bit) and XP (32-bit and 64-bit). Windows 7 support is not guaranteed, though there is no reason why these binaries shouldn't work. In case of any trouble, please recompile the code from scratch using the instructions in the sections below.

The Windows versions were compiled using the Microsoft VC++ 2008 Express compiler and hence need the Microsoft Visual C++ 2008 redistributables to work[3]. Please download and install the package applicable to your OS and CPU architecture. The redistributable files should come preinstalled on Vista, and on later Service Packs for Windows XP. Please ensure that your video card drivers are upto date. XRaySim needs OpenGL 1.5 support to work correctly (for the use of Vertex arrays and Vertex Buffer Objects). In the unlikely case that you have a much older card, please follow the configuration instructions below to disable VBOs/ vertex arrays.

1.3 Compiling from source

XRaySim depends on the following open-source libraries:

- **FLTK** *(http://www.fltk.org)* - FLTK is a platform-independent windowing system for easily writing user-interfaces, released under the GNU GPL. XRaySim is written against FLTK 2.0, which does not have a stable release yet. In order to get compile, download a weekly snapshot from the website, or grab the latest source from the FLTK repository.

- **Boost.Spirit** *(http://boost-spirit.com/home)* - Boost.Spirit is a small and extremely versatile generic string parser framework. The scripting system controlling XRaySim's material system uses Boost.Spirit extensively. Luckily, the package needs no compilation, just get the sources from the website and point your IDE to the location of the headers before compiling.

- **CImg** *(http://cimg.sourceforge.net)* - CImg is a generic, cross-platform and easy to use image loading library. It contains one header file (which is already included with the XRaySim source), so you do not need to download anything extra. XRaySim is pre-configured to use the CImg header included with the source distribution. In case you need to use a newer version of the library (for implementing some advanced features), download and overwrite the file in the XRaySim source distribution.

- **GLEW** *(http://glew.sourceforge.net)* - GLEW is a very useful wrapper for using OpenGL extensions across platforms without headaches. Pre-compiled GLEW libraries are available from the website which need to be linked with the XRaySim source.

Depending on which IDE/ compiler combination you are using, instructions will vary:

- **Microsoft Windows** : XRaySim has been succesfully compiled on Windows using the Visual C++ 2008 Express and the Visual C++ 2005 Express compilers. It should also compile with the MinGW compiler without any hassles. The use of an IDE is optional. The above libraries should be downloaded first. Point your search directories to find the .c** .h and .lib files in them, and compile the entire source distribution.
- **Linux/ MacOSX** : Unfortunately, no precompiled binaries are available for these platforms as of yet. Using the gcc compiler, XRaySim should (ideally) compile on both of these platforms. A linux distribution is currently in the works and a MacOSX version should be in the works soon, depending on how early the author can get his hands on a system :-) . If you are interested in helping out with the Linux or Mac releases, please feel free to send the author an email (contact information is available on the XRaySim website)

1.4 Bug-fixing and contributing to XRaySim

Currently, XRaySim is written and maintained by only one individual, and will probably stay that way until the entire package reaches version 1.0. This is partially because several modules currently exist in a nascent state and all of the pre-planned features should find their way into the package by version 1.0. Please note that this applies to the inclusion of new features only and not bug-reports. As is well known, bug-fixing is

a continuous (and possibly eternal) process which needs inputs from as many sources as possible. Hence, please report any bugs of any sort that might crop up during your use of XRaySim. This would help other people using it and would also help make this package that little bit more polished. Bug reporting can be done by setting up a ticket on the sourceforge process or by just sending the author an email (contact details are available on the project website).

Chapter 2

Getting Started

This chapter deals with getting to grips with XRaySim and getting it to work for your particular simulations. A brief introduction is supplied here with more detailed notes in the subsequent chapter. If you are a first time user, you might want to follow along to get an idea of all of the basic internals of XRaySim and hence gain an overview of the package's functioning.

2.1 Your first simulation

XRaySim is meant to be a complete imaging suite. The current release consists only of the forward simulation module. This is supposed to mimic the setup you see in an industrial radiography/ medical CT system. The idea is that, given a source/ detector combination with a number of preset parameters, the package can determine the radiographic simulation of any CAD object (represented as a triangular mesh) at any orientation and position. It can also generate sinograms (2D image, with a number of collimated projections) for the object at an arbitrary cutting plane.

The following sections will walk you through getting your first scan simulations with XRaySim. For first time users, it is recommended that you read along in the same order. If you need specific information, feel free to jump to the corresponding section for more detailed instructions. So with that, lets get started with our first simulation!

2.2 Importing data

XRaySim is meant to work seamlessly with commercial CAD packages. It does this through the use of commonly established file formats for data exchange.

CAD models for use with XRaySim must be exported as polygonal meshes from the corresponding modeling package. More specifically, XRaySim works with triangulated meshes through the Stereolithography (.STL) and Wavefront OBJ (.OBJ) file formats. STL files come in two sub-formats - ASCII and binary. The former was first introduced for quick testing and for use with simple meshes (of very low polygon count). Please

note that XRaySim works only with the latter (binary STL files) because of their widespread use and ability to describe extremely large and complex meshes.

The OBJ format, on the other hand, has been relatively unchanged over the years. The specification, however, allows for some variations with regards to what data can actually be in the file. XRaySim needs vertex texture coordinate data inside the file (data prefixed with *vt*) to be able to function properly. Currently, it does'nt use the texture coordinates, but it might in the future. Please note that exporting your data without using the vertex texture coordinates could cause XRaySim to crash. Also, XRaySim does'nt use the material file (.MTL) file that comes along with the OBJ file, it has its own material system (see the material system specifics below).

The current version of XRaySim uses its own dimensioning system. This is to generalize data exchange between various packages ensuring cross compatibility. Dimensions inside the package are measured in World Units (WU). These can take up any dimensions you'd want to work with. Just remember to convert the object's dimensions accordingly. When loading your own models, XRaySim allows you to specify the scale in WUs. This corresponds to the longest side of the axis-aligned bounding box of the model. This calculation has to be performed apriori in order to dermine the required scale for the particular object. The other sides are scaled accordingly maintinaing the model's relative size intact.

2.3 Custom materials

XRaySim's material parsing system comes with a couple of materials pre-scripted. The data for these materials is taken from the NIST database for elements and compounds [4]. All of the material definitions are stored in the *materials.dat* file in the same folder as the executable (atleast by default, XRaySim is compiled to look for the file in this location. This could, however, be very easily changed by making changes in the source)

A sample material file will look something like this:

```
#setup materialcount 1
Material
{
        Name : XYZ1;
        MatID : 1;
        AtomicNo : 26;
        Z/A : 0.46556;
        I : 286.0;
        Density : 7.874E0;
        Colour 0.3 0.4 0.3 1.0;
        PointCount : 6;
        V 1.00000E-03  9.085E+03;
        V 1.50000E-03  3.399E+03;
        V 2.00000E-03  1.626E+03;
        V 3.00000E-03  5.576E+02;
        V 4.00000E-03  2.567E+02;
```

```
        V 5.00000E-03   1.398E+02;
}
```

The first line in the above file is the header. All header lines begin with a # in front of them, much like in C or C++. Currently, only 1 header command is enabled in the scripting system, though this can be easily extended (check the source code documentation for more information on this). The first line indicates how many material definitions are present in the current script file. Please note that if this number is lesser than the actual number, some material definitions will not be read; if the number is greater, the application will throw an error.

Material definitions are pretty straight-forward. Each definition begins with the keyword **Material** (case sensitive), following by a set of parantheses. Within these parantheses, each of the parameter definitions are provided, again following a syntax similar to C/ C++. The number of parameter keywords is limited - these are (all keywords are case-sensitive):

- **Name** - A name (preferably unique) that XRaySim will use to display the material on the 'available materials' list, can be upto 255 characters long, and *must* contain only alpha-numeric characters (no special symbols allowed). Spaces, if present, will be ignored and the words concatenated.
- **MatID** - A unique ID number to refer to this material internally. This need not be in any order as long as it remains unique.
- **AtmoicNo** - Atomic number of the element. For elements, this value is used to internally validate the attenuation co-efficient, but is not used in the actual simulations. For compounds without an atomic number, the calculation is usually performed using a linear combination of the constituent elements. This effectively defines the mean atomic number.
- **Z/A** - Ratio of atomic mass to atomic number. Again, this value is used only internally, and for compounds, use a weighted average for each of the constituent elements (this is already available for a large collection of compounds from the NIST compounds database [5])
- **I** - Mean excitation energy for the element. For compounds, again a weighted mean is computed depending on the relative proportions of the constituent elements. (Check [5] for already published data)
- **Density** - Density of the element/ weighted density of the compound in g/cc
- **Colour** - Colour of the material. This is used by XRaySim to update the 3D display showing the object. The colour has nothing to do with the calculations, it is only used as a visual aid. The colour is specified as 4 floating point numbers - R, G, B, A. The syntax is

  $$Colour < red >< green >< blue >< alpha >;$$

  Please note that there is no colon symbol after the keyword **Colour**
- **PointCount** This specifies the number of sampling points for the attenuation coefficient (which follow this command). Any energy outside this range will return an undefined attenuation coefficient (of -1.0). Using a count higher than the number of samples will result in the application throwing an error

- **V** $\langle$ *energy in MeV*$\rangle$ $\langle$ *attenuation in* cm^2/g $\rangle$ - The lines using this syntax that follow the **PointCount** definition define the attenuation coefficient as a function of the beam energy. The energy is specified in MeV and the attenuation constants are in cm^2/g. Values can also be expressed in Exponential format. For instance, 0.001 can also be written as $1.0E-03$.

2.4 Image formats

XRaySim works with images using the CImg library. It natively works with 24-bit bitmaps. The .BMP format was chosen over other popular formats for its simplicity of use and the fact that it has no compression losses. This ensures that all of the data stored into 24-bit bitmaps (8-bits per channel) can be recovered without any changes at all. The trouble with compression-less data is that file sizes can become rather large. A typical high resolution scan in XRaySim (2048x2048) will occupy over 12 MB.

To offset this, XRaySim also has an internal file format (.FP) in the works. This is a simple compressed format, based on the fact that floating point data typically needs 4 bytes per pixel (bitmaps need 3 bytes,but is quantized in 8 bits) to retain full precision. This should hopefully find its way into the next public release.

Enabling other file formats is very easy with the CImg library, though that would need a recompile of the entire source code. Further, if XRaySim is compiled with the ImageMagick library, support for additional formats will be available (including PNG files, and with libTIFF, TIFF files).

Chapter 3

Scan run-through

This chapter will provide more detailed look at the user-interface elements inside XRaySim, explaining each of them in detail. This should help you get your first scan simulations done in no time. In case you need information on getting your own data into XRaySim, please review the previous chapter on *Getting Started*. This chapter assumes that you have successfully brought the model file into XRaySim from your CAD package.

3.1 Basic interface

When you first load up XRaySim, the main window is displayed with multiple UI widgets. The screen should look like the picture shown in Fig 3.1

The screenshot shows the basic user interface. The main window consists of a number of panels, labelled as shown in the figure. The various panels are described in sequence below:

- **3D Viewport** - The viewport is your view into the scene. As seen in Fig 3.1, the view shows the source (at a point on the y or green axis), and the detector plane. Also seen is the slicing plane for tomographic data generation. The source is rotated in this plane and various projections are stacked to generate a sinogram. More detailed instructions follow.
- **Realtime preview** - The realtime preview window shows a preview of the actual radiographic image of the object at the current orientation and position. This is updated automatically as you manipulate the object in space or change scan parameters.
- **Scene Controls** - The scene controls panel helps you alter the object's position and orientation in space. All manipulations are object centric and have to be performed on a specific object. The drop-down box lists all the objects currently in the scene. To change a particular object's position/ orientation, select it from the drop-down menu first. The three buttons in the middle control translation while the bottom three buttons set the corresponding Euler angles. Please note that rotation is order dependent, so the order in which the object is rotated along each of the axes determines the final orientation. The *Reset Transform*resets all transformations for the currently selected object.

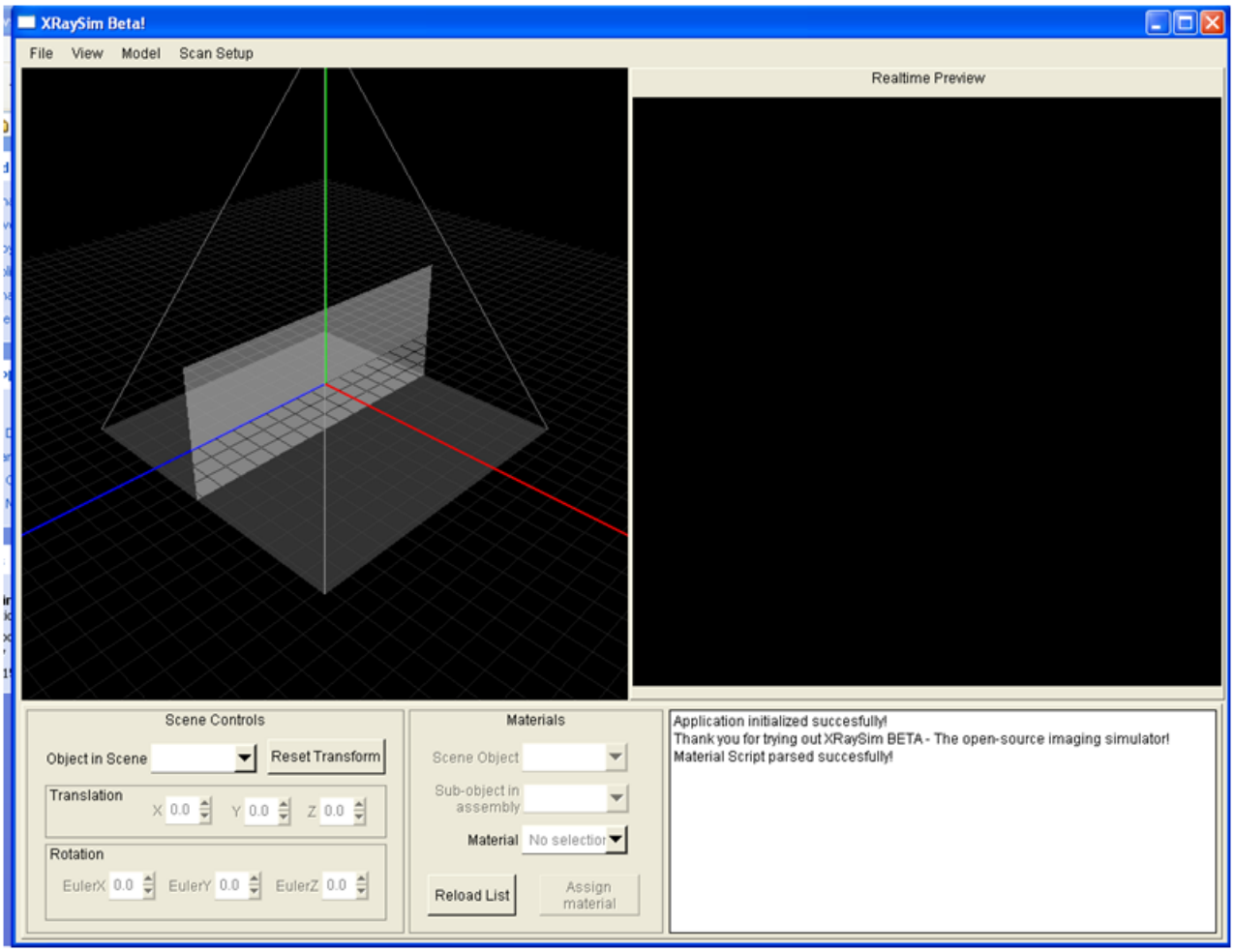

Figure 3.1: XRaySim user interface

- **Materials** - The materials panel controls the material mapping between objects and materials defined in the external script. To change a mapping, select the model in the scene from the first drop-down menu, and the required material from the third drop-down list, and click on the *Assign Material* button. Clicking on *Reload List* reparses the material script again and refreshes the list
- **Logging window** - This window shows message logged by XRaySim's logging system

3.2 Quick scan

This section will run you through a simple scan using XRaySim. It will show you how to perform the basic steps necessary to go from a model to a radiographic image. It is advisable to follow the next few steps in order to get a grasp of how the entire system works.

- **Load a model** - The model is loaded into the scene selecting **Model -⟩ Import CAD model** from the menubar. This brings up the model loading sub-window with a number of options. Select the model file by clicking on the **Browse** button and locating your file in the browse window that comes up.
  - The scale system used in XRaySim is World units (WU). The system is explained in more detail in a previous chapter. The scale control adjusts this rescale between 0.1 and 3.0 WU
  - Selecting the **Append to Scene** checkbox will load the model into the scene without removing the existing contents
  - Checking the **Recenter Coordinates** will recompute the object's mesh coordinates and center it around the origin. It is recommended that you do this to ensure that the object's data is cleaned, ensuring easy manipulations
  - Selecting the **Recompute Mesh Normals** will force XRaySim to recalculate the object's normal coordinates, ignoring any normal data already present in the file
- **Assign a material** - Once the object is successfully loaded into scene, it should appear in the drop-down menu on the materials panel. In order to assign a material to the object, select it from this list, choose a material from the **Material** drop down list and click on **Assign Material**
- **Set the scan geometry** - Various geometric parameters can then be set for the simulation. These can be set by selecting **Scan Setup -⟩ Source Settings** from the menubar. The source-detector distance, and detector offsets can also be set this way. As you set the various parameters, the viewport is updated to show the results. Again, all of the distances are in WU.
- **Specify detector settings** - Selecting the **Scan Setup -⟩ Detector Settings** brings up the Detector settings subwindow
  - The pixel array size determines the size of the radiographic image or sinogram. This physically corresponds to the number of pixels in one detector row. For 2D radiographs, this translates into edge length for a square detector (both dimensions being equal to this number). Click and drag the pointer to change the value

  - The energy band that the detector picks up can be set using the **Min Energy** and **Max Energy** dials. These are circular dials and should be set by selecting a point at the required angular position. The actual values are displayed in the box to the right of these controls. These controls act like a sort of inherent filter removing all of the extra X-rays outside this band

- **Realtime scan** - The Realtime Scan preview window in the main window should now show a radiographic image computed on the fly. By changing the orientation of the model, you should see the image update automatically, allowing for realtime visualization

3.3 High resolution scans

XRaySim is also capable of producing high resolution scans (of upto 2048x2048 pixels) and can save them to file. In order to generate a high resolution scan, setup your scene first, as described in the previous section. Now, go to **Scan Setup -⟩ Radiography Simulation -⟩ Raster Scan** to generate a high resolution scan. This should bring up the detector view window displaying the sample image in the center. The size of the image being large, use the scrollbars to navigate to a different region in the image. In case your image is cut off in the center, change the detector size in the **Detector Settings** window. Scan computation times are dependent on the size of the detector as well as the complexity of the model.

From this window, you can save your scan to the hard disk, by selecting the **Save BMP**button to generate a 24-bit bitmap. Also, simple histogram equalization can be performed by going to **Image -⟩ Histogram Equalization -⟩ Global Equalization**. Equalization stretches the image to the entire 8-bit range, thus increasing image contrast.

3.4 Sinogram generation

XRaySim can also generate sinograms for tomographic reconstruction. A sinogram is a collection of 1D projections along a plane, taken at a number of different angles. These can then be used to reconstruct the data in the plane using the Fourier Slice theorem and some reconstruction algorithms. For a more comprehensive treatment of the subject, please refer to [6].

In order to generate sinogram data, setup the scene as required first, by assigning materials and setting the object orientations. Then select **Scan Setup -⟩ Tomography Scan -⟩ Generate Sinogram**. Sinogram data is generated along the plane shown in the 3D viewport. The plane properties, along with the number of projections, and collimation type can be altered by bringing up the Tomography settings subwindow. This is done by going to **Scan Setup -⟩ Tomography Scan -⟩ Settings**

Once sinogram generation is done, a browse window will appear. Again, the image can be saved to disk at any location. The sinogram image is a standard 2048 x 2048 bitmap, with the sinogram data occupying the top-left of the image, of size $p * q$ pixels, where p is the detector array size and q is the number of projections. The sinogram generation algorithm has not been optimized very well, so computation times could be on the higher side (as opposed to the Radiography simulation).

Chapter 4

Additional information

Sources for additional information and support for XRaySim are listed here. The main source of information as always is the XRaySim website - http://xraysim.sourceforge.net. This is the place to go to for the latest updates and release information.

Additional pointers to information on Non-Destructive Evaluation in general can be found on the web. [7] contains publications on various sub-fields concerning NDE (including X-ray imaging and ultrasonic testing). Excellent introductory material is also available on the net from other sources - [8] and [9] are good places to start.

The Center for Non-Destructive Evaluation at the Indian Institute of Technology, Madras is a research lab involved in the development of NDT technologies for various inspection systems. More information on the activities of the research group is available on its website [10]. For support information, the author of XRaySim can be reached directly via email [11].

We hope XRaySim helps you in your research activities (for that is what it was designed for in the first place). Please feel free to give us feedback of any sort on any of its features/ shortcomings, we're always excited to hear from you!

Bibliography

[1] C. Bellon, G.R. Jaenisch, 'aRTist Analytical RT Inspection Simulation Tool', International Symposium on Digital industrial Radiology and Computed Tomography, June 25-27, 2007

[2] Federal Institute for Materials Research and Testing, berlin - http://www.bam.de

[3] Microsoft Visual Studio 2008 redistributable from Microsoft's download center: http://www.microsoft.com/downloads/

[4] NIST X-ray attenuation database online - http://physics.nist.gov/phyrefdata/xraymasscoef/cover.html

[5] NIST X-ray data for compounds - http://physics.nist.gov/phyrefdata/xraymasscoef/tab2.html

[6] A.Kak and M.Slaney, Principles of Computerized Tomographic Imaging, online book retrieved from http://www.slaney.org/pct

[7] Database and Journal of Non-destructive testing, accessed on the web at : http://www.ndt.net

[8] Introduction to Non-Destructive Evaluation by Clint Logan, Lawrence Livermore National Laboratory, http://www.llnl.gov/str/Logan.html

[9] Non-destructive evaluation and Advanced Actuators technologies from the Jet Propulsion Lab, NASA, http://ndeaa.jpl.nasa.gov

[10] Center for Non-Destructive Evaluation, IIT Madras - http://www.cnde-iitm.net

[11] Koushik Viswanathan, email ID : koushik.viswanathan@gmail.com