X-Ray Simulator Testing

Betty Zhao

Last Updated: March 5, 2018

Program Description

This program is a naïve x-ray simulator for 3D objects. The program takes in 3D objects (in ASCII STL format), and based on its geometry, renders a 2D image (in bitmap .bmp format) simulating the result of that object after undergoing an x-ray test. This program does not take into account the various ways that light can scatter, and it assumes uniform material properties for the entire object.

The x-ray simulator comprises two individual components, wrapped inside a bash script (`xray_simulator.sh`):

- `stltovoxels.py`: A third-party library that converts ASCII STL files to voxels, represented by a series of PNG files.
    - This program simulates the light interacting with the object.
- `sum_slices.py`: Sums up the white and black values of each PNG file (slice) and equally weights each slice to generate an x-ray simulation.
    - This program simulates the detector output.

Protocol Purpose

The purpose of this testing protocol is to ensure that each component and the entire system works correctly. Black-box (functional) testing will be done for these components, as the functionality is the most important aspect of this system. Each component will be treated as a separate black box with unknown internals while testing.

As mentioned in the README, there are several extensions that can be implemented to make this program more robust or accurate. These changes will occur within each separate program, and it is expected that the function of each black box does not change. These protocols can still be used while performing regression testing, to ensure that the changes to the programs have not compromised the program's functionality.

Test Cases

Test Case 1: Testing `stltovoxels.py`

Description/Purpose	This program will be used to determine whether the STL to voxel conversion is functioning properly. This simulates the photons being absorbed by each layer of the object.
Component(s) Tested	STL file, `stltovoxels.py`

Pre-conditions	• There is an STL file in the inputs folder
	• The required dependencies have been installed
	• Slices folder has been cleared to remove images from previous simulations
	• **Mandoline** has been downloaded
	  o https://github.com/revarbat/Mandoline
	  o This is a 3D slicer program for 3D printing that can output SVG (vector) files representing each layer
	  o Alternative: use another software that can slice 3D objects into images

Test Steps	1. Enter xray_simulation folder in a terminal 2. Enter on one line (replace FILENAME with file of interest): `> python3 src/stl-to-voxel/stltovoxel.py inputs/FILENAME.stl slices/slice.png` 3. Run Mandoline with the same input file 4. Compare outputs of slices generated by `stltovoxel` and Mandoline

Expected Result	• `stltovoxel.py` has terminated without errors
	  o If there are errors in parsing the STL file, it will be indicated, and therefore the STL file format is not compatible with the program
	• 20 - 150 PNG images (slices) are in the slices folder
	  o Slices are named slice0xxx.png
	  o PNG images resemble layers of the 3D object
	• Upon visual inspection:
	  o Mandoline slices should resemble PNG slices
	  o PNG slices should resemble the 3D object

Test Case 2: Testing **`sum_slices.py`**

Description/Purpose	This program will be tested to determine whether the images have been averaged appropriately to mimic the x-ray detector output. The output is generated through two different methods, with the results of each compared. The output is also tested to ensure the correct name and resolution.
Component(s) Tested	`sum_slices.py`
Pre-conditions	• There are slices (all belonging to the same 3D object and named slices0xxx.png) in the slices folder • The required dependencies have been installed • Image editor software is installed

Test Steps	1. Enter xray_simulation folder in a terminal
	2. Alternative: before running program, uncomment the line indicated for inverted image
	  a. Default is black background
	  b. Alternative option is white background
	3. Enter on one line (replace FILENAME with file of interest):
	`> python3 src/sum_slices.py FILENAME`
	  a. Convert output image into 2D array
	4. Using the image editor, overlay each slice on top of each other at 10% opacity for each image
	  a. Export image as bitmap file
	  b. Convert bitmap into 2D array
	  c. Normalize 2D array
	  d. Alternative: if image is inverted, then invert each slice before adding to the output

Expected Result	• `sum_slices.py` has terminated without errors
	  o 'Low contrast' or 'float to integer conversion' warnings may occur, but warnings can be ignored
	• Image editor result resembles bitmap output
	  o Compare difference between normalized 2D array result and 2D array of output bitmap
	  o Determine if values at each row and column differ by a certain maximum threshold (5%, for example)
	• Bitmap file is in outputs folder
	  o Name of output file is FILENAME.bmp
	  o Output has black or white background (depending on option selected)
	  o Output resolution is 102x102 pixels

Test Case 3: Testing **`xray_simulator.sh`**

Description/Purpose	The entire program will be tested to ensure that it properly simulates an x-ray and has met the required functionality.
Component(s) Tested	All components

Pre-conditions	• There is an ASCII STL file in the inputs folder called FILENAME.stl
	• The required dependencies have been installed
	• STL file has been 3D printed
	• X-ray machine is available
	• XRaySim has been downloaded
	  o http://xraysim.sourceforge.net/index.htm
	  o This is an open-source X-ray imaging simulator
	  o Alternative: use commercially available x-ray simulator software

Test Steps	1. Enter xray_simulation folder in a terminal
	2. Enter on one line (replace FILENAME with file of interest):
	`> bash xray_simulation.sh FILENAME`
	3. Find output image (FILENAME.bmp) in the outputs folder and note the rotation of the object
	4. Test the 3D printed object with an x-ray machine
	  a. Match orientation of the output file
	5. Run the input STL file through XRaySim
	6. Alternative:
	  a. Scan/crop x-ray output as 102x102 pixel .png, convert into 2D array, and normalize array
	  b. Crop XRaySim output into 102x102 pixel image, convert cropped XRaySim output into 2D array, and normalize array
	  c. Convert program bitmap into 2D array, and normalize array

Expected Result	• Through visual inspection:
	  o XRaySim output, x-ray output, and program output resemble each other
	  o Because assumptions are made, visual inspection is recommended for comparison
	  o The details of each test will be different
	• Alternative: Compare numeric values of each input to ensure they do not differ by a certain threshold
	  ▪ This may be inaccurate; cropping/scanning and estimations may cause differences