SAINT VINCENT COLLEGE

DEPARTMENT OF PHYSICS

JOHN PIERRE MENEGHINI

COMPUTATIONAL MODELING OF X-RAY ATTENUATION AND SIMULATION OF MEDICAL IMAGING

BACHELOR OF SCIENCE

Saint Vincent College

Draft: *April 18, 2024*

Saint Vincent College

DEPARTMENT OF
PHYSICS

COMPUTATIONAL MODELING OF X-RAY ATTENUATION AND SIMULATION OF MEDICAL IMAGING

JOHN PIERRE MENEGHINI

Thesis submitted in partial fulfillment of the requirements
for the degree of *Bachelor of Science*

Adviser: Fr. Michael Antonacci O.S.B
Associate Professor

BACHELOR OF SCIENCE

Saint Vincent College
Draft: *April 18, 2024*

Abstract

Regardless of the language in which the dissertation is written, usually there are at least two abstracts: one abstract in the same language as the main text, and another abstract in some other language.

The abstracts' order varies with the school. If your school has specific regulations concerning the abstracts' order, the NOVAthesis LaTeX (novathesis) (LaTeX) template will respect them. Otherwise, the default rule in the novathesis template is to have in first place the abstract in *the same language as main text*, and then the abstract in *the other language*. For example, if the dissertation is written in Portuguese, the abstracts' order will be first Portuguese and then English, followed by the main text in Portuguese. If the dissertation is written in English, the abstracts' order will be first English and then Portuguese, followed by the main text in English. However, this order can be customized by adding one of the following to the file `5_packages.tex`.

```
\ntsetup{abstractorder={<LANG_1>,...,<LANG_N>}}
\ntsetup{abstractorder={<MAIN_LANG>={<LANG_1>,...,<LANG_N>}}}
```

For example, for a main document written in German with abstracts written in German, English and Italian (by this order) use:

```
\ntsetup{abstractorder={de={de,en,it}}}
```

Concerning its contents, the abstracts should not exceed one page and may answer the following questions (it is essential to adapt to the usual practices of your scientific area):

1. What is the problem?
2. Why is this problem interesting/challenging?
3. What is the proposed approach/solution/contribution?
4. What results (implications/consequences) from the solution?

Keywords: One keyword, Another keyword, Yet another keyword, One keyword more, The last keyword

Contents

List of Figures

List of Tables

List of Listings

Glossary

Acronyms

aaa	acronym aaa *(p. 18)*
aab	acronym aab *(p. 18)*
aba	acronym aba *(p. 18)*
abbrev	abbreviation of a longer text *(p. 18)*
bbb	acronym bbb *(p. 18)*
DI	Department of Computer Science *(p. 2)*
FCT	NOVA School of Science and Technology *(pp. 2, 10)*
NOVA	NOVA University Lisbon *(p. 2)*
novathesis	NOVAthesis LaTeX *(pp. i, iv, v, 1–4, 6–11, 13)*
novathesis.cls	novathesis class *(p. 14)*
xpto	and extension of a xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto *(p. 18)*

Symbols

1

X-ray Tracing

Ray tracing is a computer graphics technique that involves tracing the path of rays as they pass through a virtual scene. Due to its ability to create highly-realistic images, it has been used extensively in animations, video games, scientific computing, and by designers who need a physically accurate design of a product. While not the only rendering technique or the fastest in computer graphics, its ability to consistently produce physically realistic renders has made it an indispensable tool in many industries. By accurately simulating the behavior of light, ray tracing can capture intricate details of how light interacts with objects, resulting in realistic shadows, reflections, refractions, and global illumination effects. This level of visual fidelity allows for the creation of visually stunning and immersive experiences that were previously challenging to achieve. As hardware capabilities continue to advance and real-time ray tracing becomes more accessible, its applications are expanding, and its impact on various fields is growing, pushing the boundaries of what is possible in computer graphics and visual simulation [5].

1.1 The Brute Force Algorithm

The rays in a scene are described mathematically as the parametric representation of a line. In three dimensions, the point at the end of the line $\vec{r}$ at parameter t is given by the following equation:

$$\vec{r}(t) = \vec{r}_0 + \hat{d}t, \tag{1.1}$$

where $\vec{r}_0$ points from the origin to the start of the line and $\hat{d}$ is a unit vector parallel to the direction of the line. The described ray is shown in Figure 1.1.

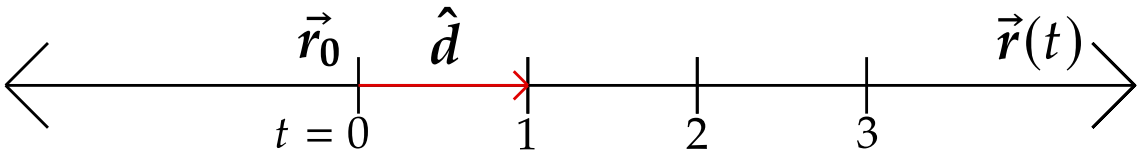


Figure 1.1: A diagram of a parametric ray.

Unlike in real life where light rays end at a camera, we instead trace light rays from the camera to locations in the scene. While one could trace light rays from a light source (forward ray tracing), many of these rays would end up missing the camera entirely, not affecting the image. More efficiently, the light rays are instead traced from the camera to the light source (backward ray tracing), drastically cutting back on the computational load while producing an identical image for most practical cases [5]. So, in the case of backward ray tracing, the method used in this research, rays are emitted from the camera; therefore, $\vec{r_0}$ represents the location of the camera in the scene.

To define the area of the scene visible from the point of view of the camera, one can create a rectangular surface in which all rays pass through, called a viewport. Increasing the size of the viewport increases the amount of the scene that is visible in the final image, and vice versa. This viewport is broken up into a grid, the resolution of which determines the resolution of the final image. Rays are then sent through each point in this grid and out into the scene.

So far, we have a camera, light rays, and a way to define what parts of the scene are visible; all three of which are visualized in Figure 1.2. It is important to note that, by convention, the z-axis in the scene points opposite the viewport.

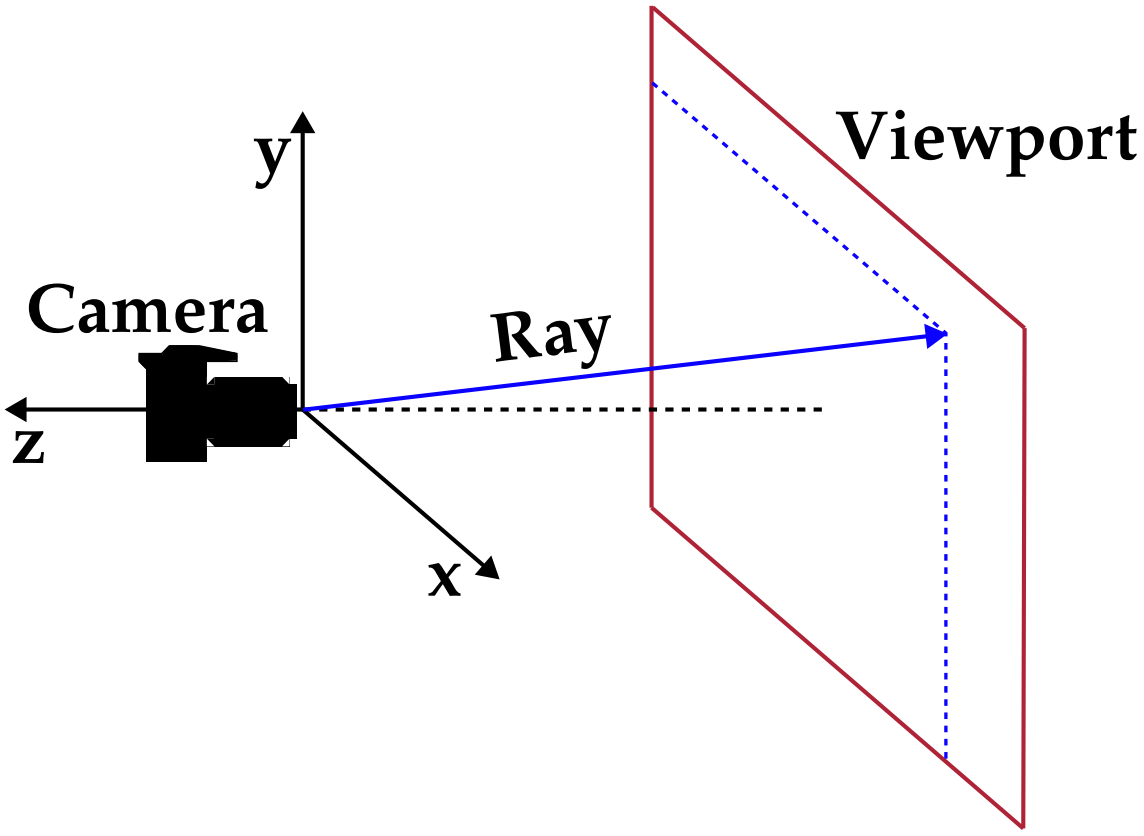


Figure 1.2: A basic ray-tracing setup with a camera, ray, and viewport.

To set the location of the viewport in the scene, we must first define three new vectors: $\overrightarrow{HO}$, $\overrightarrow{VE}$, and $\overrightarrow{FL}$, which are vectors pointing from the left most to right most point (-x to

x direction), the bottom most to top most point (-y to y direction), and from the center of the viewport to $\vec{r_0}$, respectively. A vector pointing from $\vec{r_0}$, to the bottom-left corner $\overrightarrow{LHC}$ can be calculated using the following equation:

$$\overrightarrow{LHC} = \vec{r_0} - \frac{1}{2}\overrightarrow{HO} - \frac{1}{2}\overrightarrow{VE} - \overrightarrow{FL}, \tag{1.2}$$

which was obtained geometrically from Figure 1.3 [11].

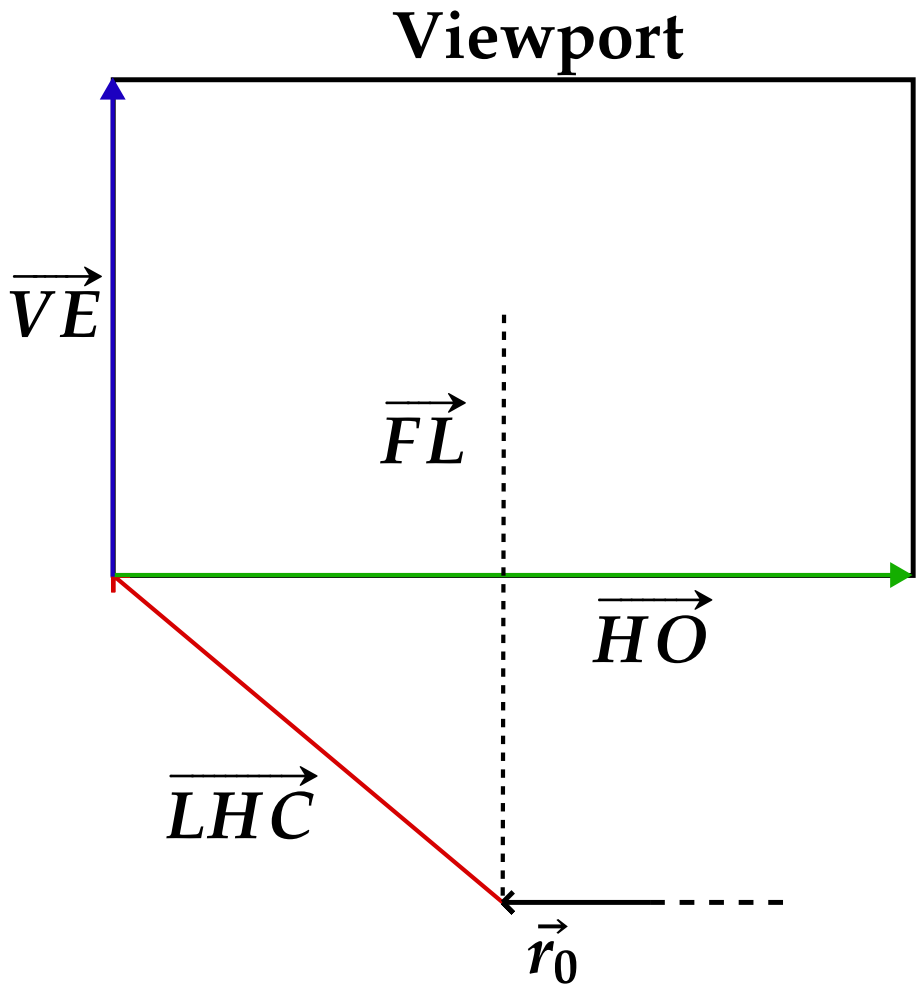


Figure 1.3: A diagram showing the geometric representations of $\overrightarrow{HO}$, $\overrightarrow{VE}$, $\overrightarrow{FL}$, and $\overrightarrow{LHC}$ with respect to the viewport.

With $\overrightarrow{LHC}$ pointing to the left-hand-corner of the viewport in the scene, $\overrightarrow{HO}$ and $\overrightarrow{VE}$ can be scaled to define a ray pointing from the camera to any point on the viewport. In particular, the viewport's horizontal and vertical coordinates, u_i and v_j, are defined as such,

$$u_i = i/(\text{image width} - 1); \tag{1.3}$$

$$v_j = j/(\text{image height} - 1), \tag{1.4}$$

where the image width and height are the resolution of the rendered image, i is an integer ranging from [0, image width - 1], and j is an integer ranging from [0, image height - 1]. Note that both u_i and v_j range from [0, 1].

Therefore, using these newly defined viewport coordinates, a ray taking the form of Equation 1.1, pointing from the camera to the pixel coordinates (i, j) is given by the following equation:

$$\vec{r}_{i,j}(t) = \vec{r}_0 + t\left(\overrightarrow{LHC} + u_i\overrightarrow{HO} + v_j\overrightarrow{VE}\right), \tag{1.5}$$

which is represented geometrically in Figure 1.4 [11].

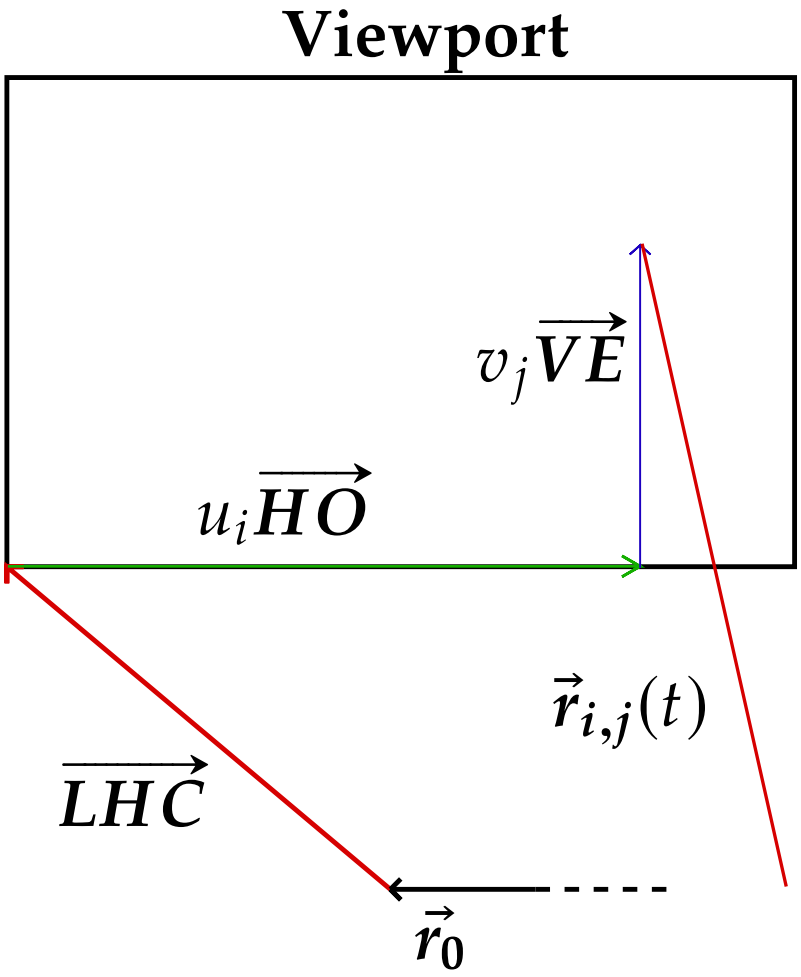


Figure 1.4: A diagram showing the geometric representation of Equation 1.5.

1.2 Ray-Triangle Intersections

With the end goal to be able ray-trace x-rays through an arbitrary user-defined mesh, we first need to choose which shape to compose said mesh with. Due to its efficient ray-intersection algorithm and its ease of parallelization on Graphics Processing Units (GPUs), these constituents, referred to as primitives, are typically chosen to be triangles.

This following section will develop the ray-triangle intersection algorithm used by the x-ray tracer. In particular, the algorithm used was invented by Tomas Möller and Ben Trumbor, and is subsequentially refered to as the Möller-Trumbore ray-triangle intersection algorithm [4].

1.2.1 Barycentric Coordinates

To motivate the unique coordinate system used by the algorithm, let us first review the concept of center of mass for 3 massive point particles. If 3 objects each have their own location in space $\vec{r_i}$, each with mass m_i, the center of mass $\vec{R}$ is given by:

$$\vec{R} = \frac{m_1\vec{r_1} + m_2\vec{r_2} + m_3\vec{r_3}}{m_1 + m_2 + m_3}. \tag{1.6}$$

Notice a few properties of $\vec{R}$ that are intuitively clear:

- $\vec{R}$ will always lie in the plane containing the point particles.
- $\vec{R}$ will always lie in the triangle T with vertices $\{\vec{r_i}\}$.

This means that by changing $\{m_i\}$, $\vec{R}$ will span all set of points in T. This is the motivation behind barycentric coordinates, the coordinate system employed by Möller and Trumbore. In particular, to represent a vector $\vec{v}$ pointing to the surface of a triangle $T \in \mathbb{R}^3$ with vertices $\vec{v_0}$, $\vec{v_1}$, and $\vec{v_2}$, barycentric coordinates can be utilized. These coordinates $\{\alpha, \beta, \gamma\} \in \mathbb{R}$ act as the masses of the vertices of T. Thus, any point $\vec{v}$ along the plane P of T can be represented by

$$\vec{v} = \alpha\vec{v_0} + \beta\vec{v_1} + \gamma\vec{v_2}. \tag{1.7}$$

For $\vec{v}$ to be located on or within T, the following requirements on the barycentric coordinates must be satisified:

- $\alpha + \beta + \gamma = 1$ (1.8)
- $\alpha, \beta, \gamma \geq 0$ (1.9)

With regard to the center of mass interpretation, Equation 1.8 removes the computation of the denominator in Equation 1.6, while Equation 1.9 eliminates the possibility of negative mass. For a rigorous proof of the requirements, see (find something).

1.2.2 Möller-Trumbore Intersection Algorithm

The algorithm consists of 2 main steps:

1. Check if ray intersects with P;
2. If so, check if intersection lies outside of T.

1. To check if the ray described by Equation 1.1 intersects with P, one can define two vectors that lie along 2 edges of T:

$$\vec{e_1} = \vec{v_1} - \vec{v_0}; \tag{1.10}$$

$$\vec{e_2} = \vec{v_2} - \vec{v_0}. \tag{1.11}$$

If $\vec{p} = \hat{d} \times \vec{e_2}$, then let $C = \vec{e_1} \cdot \vec{p}$. The line $\vec{r}$ does not intersect the plane P iff $C = 0$, which happens iff $\hat{d} \parallel P$. Thus, if $C = 0$, then $\vec{r}$ does not intersect P; otherwise, $\vec{r}$ will intersect P.

2. Solving Equation 1.8 for α and substituting into Equation 1.7 results in

$$\vec{v} = (1 - \beta - \gamma)\vec{v_0} + \beta\vec{v_1} + \gamma\vec{v_2}. \tag{1.12}$$

After distributing and rewriting Equation 1.12 in terms of $\vec{e_1}$ and $\vec{e_2}$,

$$\vec{v} = \vec{v_0} + \beta\vec{e_1} + \gamma\vec{e_2}. \tag{1.13}$$

Setting $\vec{r} = \vec{v}$ to find the barycentric coordinates for the ray-triangle intersection and solving for $\vec{r} - \vec{v_0}$ yields

$$\vec{r} - \vec{v_0} = -\hat{d} + \beta\vec{e_1} + \gamma\vec{e_2}. \tag{1.14}$$

One can then show that taking $(\vec{r} - \vec{v_0}) \cdot \vec{p}$ gives the following equation for β:

$$\beta = \frac{(\vec{r} - \vec{v_0}) \cdot \vec{p}}{C}, \tag{1.15}$$

and similarly, taking $\hat{d} \cdot [(\vec{r} - \vec{v_0}) \times \vec{e_1}]$ results in

$$\gamma = \frac{\hat{d} \cdot [(\vec{r} - \vec{v_0}) \times \vec{e_1}]}{C}. \tag{1.16}$$

To then determine t at which the ray-triangle intersection occurs, one can take $\vec{e_2} \cdot [(\vec{r} - \vec{v_0}) \times \vec{e_1}]$, yielding

$$t = \frac{\vec{e_2} \cdot [(\vec{r} - \vec{v_0}) \times \vec{e_1}]}{C}. \tag{1.17}$$

Since Equation 1.8 is assumed upon the construction of Equation 1.12, one only needs to verify that Equation 1.9 is satisfied to determine if there is a ray-triangle intersection. In addition, since certain scenarios might require a finite length ray described by $t \in [t_{\text{min}}, t_{\text{max}}]$, one would also check if the t of the intersection falls within this range. The implementation of this algorithm used in the C++ x-ray tracing code is shown in Listing 1.

```cpp
bool triangle::hit(const ray &r, float t_min, float t_max, hit_record &rec) const {
  float kEpsilon = 1e-9;

  vec3 e1 = v1 - v0;
  vec3 e2 = v2 - v0;
  vec3 pvec = cross(r.direction(), e2);
  float C = dot(e1, pvec);

  // ray is parallel to triangle
  if (std::abs(C) < kEpsilon) return false;

  float inv_C = 1.0 / C;

  vec3 tvec = r.origin() - v0;
  float beta = dot(tvec, pvec) * inv_C;

  // hit point is outside of triangle
  // if beta > 1.0 -> alpha and/or beta < 0
  if (beta < 0.0 || beta > 1.0) return false;

  vec3 qvec = cross(tvec, e1);
  float gamma = dot(r.direction(), qvec) * inv_C;

  // hit point is outside of triangle
  // if beta + gamma > 1.0 -> alpha < 0
  if (gamma < 0.0 || beta + gamma > 1.0) return false;

  float t = dot(e2, qvec) * inv_C;

  // hit point is outside of ray's range
  if (t < t_min || t > t_max) return false;

  // record intersection point
  rec.t.push_back(t);
  rec.p.push_back(r.at(t));

  return true;
}
```

Listing 1: C++ implementation of the Möller-Trumbore ray-triangle intersection algorithm.

1.3 Meshes and Their Intersections

From these triangle primitives, one can easily construct a mesh. In particular, one can create an STL file, which is a list of triangles with each element containing the vertices and unit normal vector of a particular triangle. These files, which must describe closed meshes, can then be specified in the x-ray tracing code, along with their position in the scene and their composing material M. An example STL mesh is shown in Figure 1.5.

(a) Top (b) Bottom

Figure 1.5: The rendered (a) top and (b) bottom of an STL mesh representing an ancient Chinese coin. This particular mesh is composed of 33,558 triangles.

The key quantity of interest for an x-ray traversing through one of these meshes S is the total length l of the ray that exists within the mesh. To calculate l, the x-ray tracing code

1. Loops through each triangle T_i in S and checks for an intersection using the Möller-Trumbore algorithm described in Section 1.2.2. If the ray does not intersect with any T_i, then the ray is said to not intersect S and $l = 0$.

2. If the ray intersects N times with S, then the parameters t_i of the intersected T_i's are sorted in ascending order.

3. Since a ray can only enter then exit a mesh,

$$l = \sum_{i=0}^{\frac{N-1}{2}} (t_{2i+1} - t_{2i}). \tag{1.18}$$

1.4 Intensity Drop

With l obtained from Equation 1.18, one can then calculate the effect on the ray's intensity I after traversing S using the exponential attenuation law developed in the typical

undergraduate modern physics course [10]:

$$I = I_0 e^{-\mu_m(M,E)\rho l}, \tag{1.19}$$

where I_0 is initial intensity of the ray before entering S, $\mu_m(M, E)$ is the mass-attenuation coefficient of the material M composing S at energy E, and ρ is the mass density of M.

Furthermore, if one considers a single photon travelling a distance l through S, the ratio $\frac{I}{I_0}$ would represent the probability of completing the traversal. Thus, one can use the multiplication rule of probabilities to generalize Equation 1.19 to find the relative intensity $\frac{I_k}{I_0}$ of a ray after traversing a k number of different S's. In particular,

$$\frac{I_k}{I_0} = \prod_i^k \frac{I_i}{I_0} = \exp\left[-\sum_i^k \mu_m(M_i, E)\rho_i l_i\right], \tag{1.20}$$

where M_i, ρ_i, and l_i is the material, mass density, and path length, respectively, of the i-th traversed S.

Recall that these rays are being sent out from a source (camera) to a grid of points of a specified resolution (viewport). In the case of x-ray imaging, the camera would represent the x-ray tube, while the viewport would be the x-ray detector. So, these meshes would be placed between the tube and detector, and rays would lose intensity in their path from the tube to the detector, revealing details of the meshes' geometries and compositions. In order to visualize these variations in intensities, the x-ray tracing code directly converts the relative intensity values into pixels of an 8-bit grayscale image. These relative intensities obtained from Equation 1.20 are converted to integers in the range of 0 to 255, where $\frac{I_k}{I_0}$'s of 1 would become 255 (white pixel), and values of 0 would stay 0 (black pixel). It's important to note the versatility in image interpretation: the inversion of image color, a practice variable across different X-ray machines, does not compromise the information contained. This manipulation, sometimes employed by radiologists for enhanced detail visibility, alters none of the underlying data. Rather, it serves to accentuate certain details, making them more discernible visually. Moreover, advanced imaging software often allows radiologists to adjust the contrast and brightness of x-ray images post-processing, further aiding in the identification and analysis of specific features or anomalies within the image.

For a flowchart summarizing the x-ray tracing algorithm introduced in this chapter, see Figure 1.6.

1.5 Mass-Attenuation Coefficients and Material Definition Sources

In practice, one needs a source of the μ_m's used in Equation 1.20. The source of these values used by the x-ray tracing code is NIST's tables of x-ray mass-attenuation coefficients obtained by Hubbell [3]. These tables are given for each element Z, from which we can

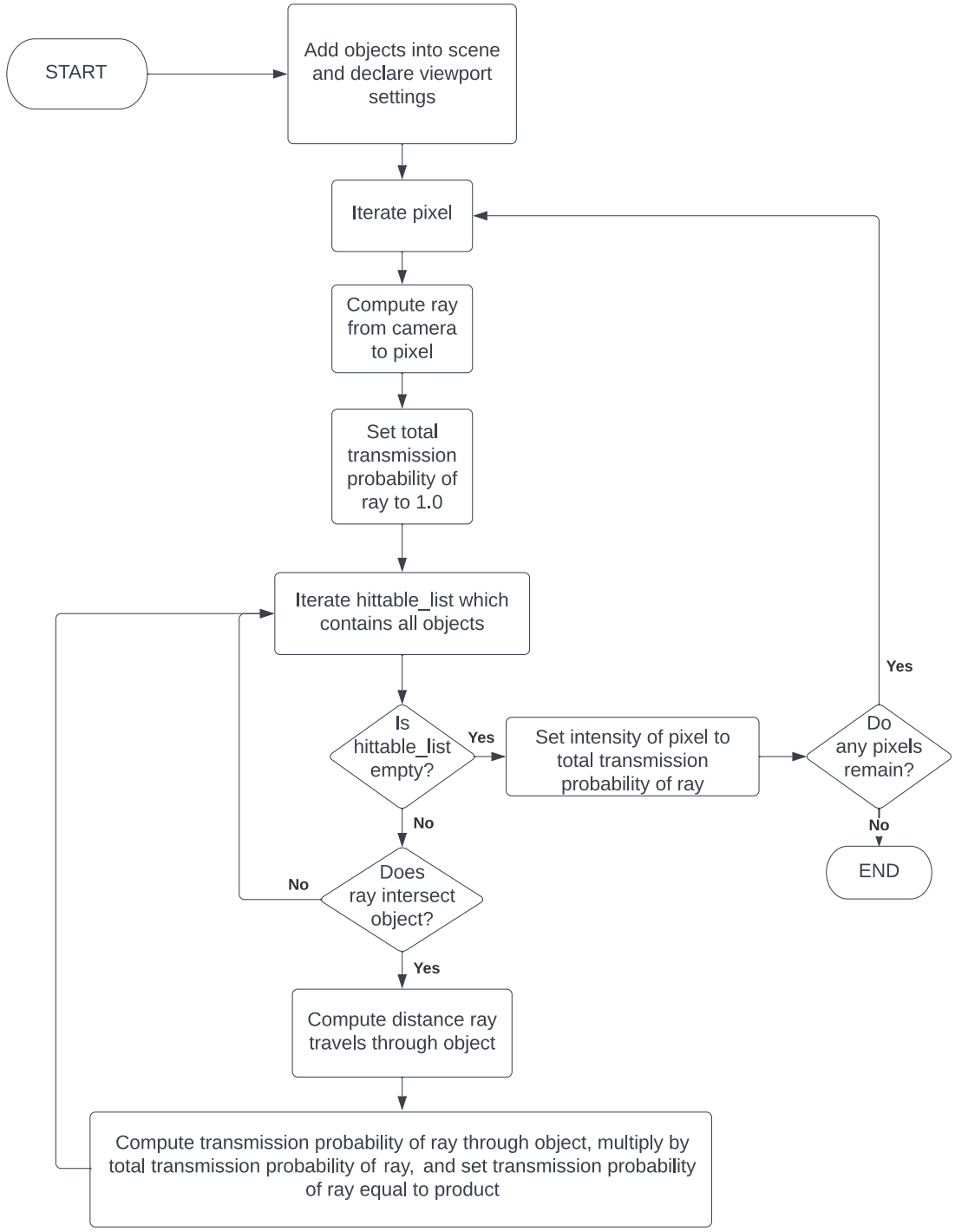


Figure 1.6: A flowchart summarizing the x-ray tracing algorithm introduced in Chapter 1.

approximate μ_m of a material M by taking the weighted average of the mass fractions of its constituent elements and their respective μ_m's. In particular, the mass-attenuation coefficient of a particular M at E, composed of elements Z_i with mass fractions w_i, is given by

Figure 1.7: A rendered x-ray image (grayscale inverted) of the ancient Chinese coin shown in Figure 1.5, with M being cortical bone and $E = 30\,\text{keV}$.

$$\mu_m(M, E) = \sum_i w_i \mu_m(Z_i, E), \tag{1.21}$$

with Z_i and w_i of common materials provided by NIST [3].

2

Viability of an X-ray Spectrum's Effective Energy for Use in Simulation

Currently, this ray-tracing model assumes a single energy x-ray (monoenergetic) is traveling through the objects in the scene; however, with actual x-ray machines, a wide range of energies (polyenergetic) is observed. This observed spectrum has the following explanation: inside an x-ray tube, electrons build up on a filament and are accelerated to an anode as a result of a high voltage being applied across the filament (cathode) and the x-ray target (anode). As the electrons hit the target, often made of tungsten, the electrons have a low probability of being slowed down and deflected by the positive charge of the protons in the target nucleus. The kinetic energy of the electrons lost due to this attraction is converted to an electromagnetic wave of equal energy, and increases with proximity to the nucleus. This phenomenon is called *bremsstrahlung radiation* and is the only source of these x-rays at kVp's below 69.5 keV. In the extremely rare case that an electron loses all of its kinetic energy due to this nucleus interaction, the maximum energy x-ray is produced and is equal to the accelerating voltage of the tube, which is referred to as the spectrum's kVp (kilovolt peak). Once a tube's kVp rises above 69.5 keV, the binding energy of tungsten's K shell, a vacancy will be left in the K shell and electrons from the L shell, which has a binding energy of 11.5 keV, will fill this vacancy, producing an x-ray of energy $69.5 - 11.5 = 57.0$ keV. As a result, as the kVp of the tube increases, intensity spikes resulting from electron transitions will appear in the x-ray energy spectrum [9].

Therefore, to better model reality, it would be best to redesign the ray-tracing model to produce photons according to an energy distribution; however, this would involve thousands of photons per pixel, which would drastically increase the render time of the simulation. An alternative option would be to find the singular x-ray energy that would result in the same attenuation as a particular energy spectrum produced by a tube, which is called the effective energy [9]. With this energy, one would be able to obtain approximately the same results as the spectrum of energies produced by the x-ray machine, eliminating the need to complicate the model and increase computation time. While this effective energy technique would solve many of the simulation's problems, the factors that affect

the effective energy must first be determined.

2.1 Methods

In particular, the following factors were tested: an object's composition, geometry (thickness), and position in the x-ray detector. To test any of these factors, one must first determine how to obtain the effective energy, E_{eff}, from an x-ray image. Solving Equation 1.19 for μ_m yields the following relationship:

$$\mu_m = \frac{\ln(I_0/I)}{\rho l}. \tag{2.1}$$

With μ_m being a function of effective energy, one can refer to experimental data to relate a value of μ_m to its associated effective energy. Therefore, with NIST's data [3] and Equation 2.1, one can find the I_0, I, and l for a particular x-ray image and determine the effective energy of the spectrum at the set kVp.

To obtain these values, a small surgical x-ray machine called the GE MiniView 6800 Mini C-Arm was used in the anatomy lab at Saint Vincent College. In order to test the composition dependence of the effective energy, images were taken of aluminum (labeled N), iron, magnesium, water, and a plastic scintillator. The thickness and density of each attenuator can be found in Table 2.1. Several images were taken for each object, with kVps ranging from 40 keV to 60 keV. For each image, the object was placed in the center of the imaging area of the C-Arm, and the kVp was set with the kVp controller located on the machine. Once ready, a remote was used to activate the machine, which then sent x-rays from the tube toward the imaging area. A diagram of this setup is presented in Figure 2.1.

Attenuator	Thickness (cm)	Density (g/cm^3)
Aluminum (N)	0.23	2.7
Aluminum (I)	0.081	2.7
Iron	0.01	7.87
Magnesium	0.1	1.46
Plastic Scintillator	1.98	1.03
Water	5.80	1.0

Table 2.1: The thicknesses and densities of materials used in obtaining effective energy values.

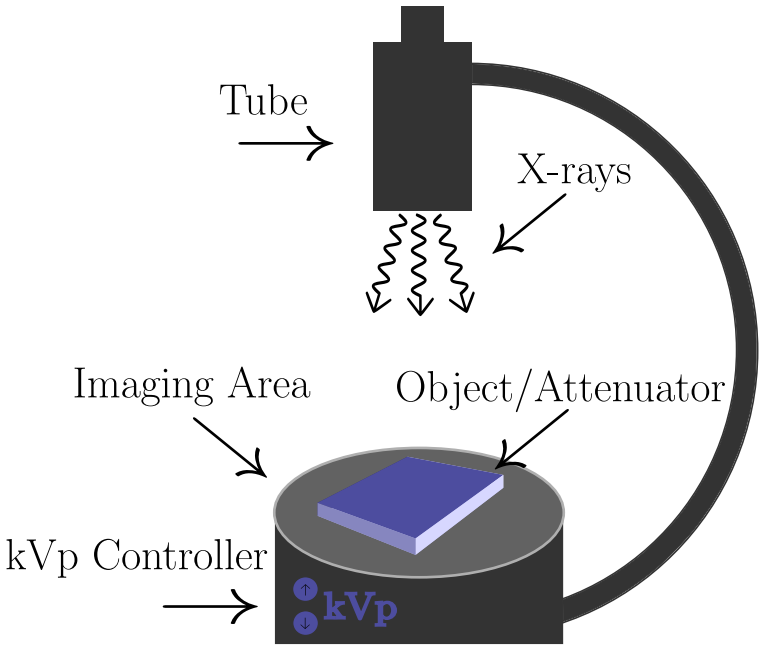


Figure 2.1: Diagram of GE MiniView 6800 Mini C-Arm used in effective energy measurements.

To test the geometrical dependence of effective energy, an image was taken of a slightly thinner sheet of aluminum (labeled I) and compared to an image of the thicker aluminum (labeled N) used for the composition experiment. Additionally, for the position dependence, an image was taken of aluminum shifted towards the edge of the imaging area. For both of these experiments, the images taken were compared to images taken at the same kVp to observe only the change in effective energy due to the factors adjusted, rather than a change due to a shift in the spectrum.

With images taken, I/I_0 had to be extracted from each image to determine μ_m using Equation 2.1. For example, consider the x-ray image in Figure 2.2: along an x-ray's journey from the tube to the detector, it encounters two different attenuators, the air and the object placed in the imaging area (with the machine setting a completely non-attenuated beam equal to 1 in the image, where 1 is white and 0 is black, the only values we can measure from pixel inspection are the relative intensities of the x-ray. Because of this, I will now simply refer to the pixel intensity (0-1) and will replace I/I_0). In the image, I becomes I_{bkg} and I_{ctr} for the pixel intensity in the background and center of the image, respectively. I_{bkg} refers to attenuation caused by the air, while I_{ctr} refers to attenuation caused by both the air and the object. Using Equation 1.20, one can show that $I_{\text{ctr}} = I_{\text{bkg}} I_{\text{obj}}$, where I_{obj} is the attenuation caused by the object. Therefore, the intensity drop caused solely by the object is given by

$$I_{\text{obj}} = \frac{I_{\text{ctr}}}{I_{\text{bkg}}}. \tag{2.2}$$

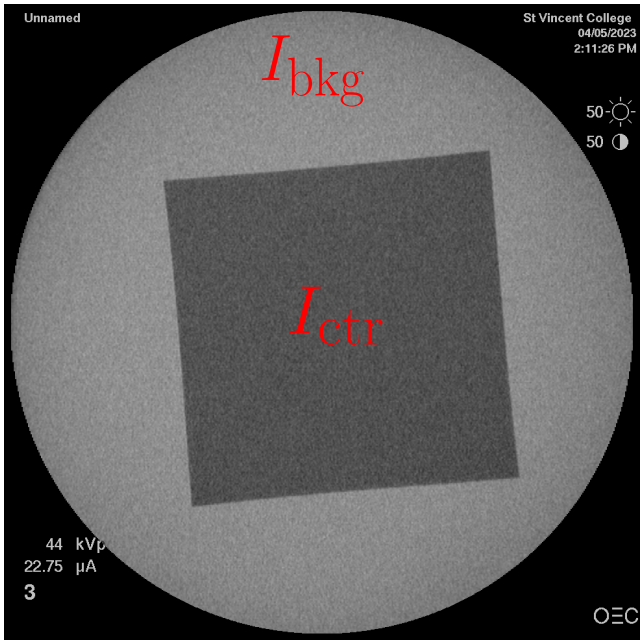


Figure 2.2: X-ray image of Al at 44 kVp demonstrating the locations of the pixel values used in the calculation of I_{obj}.

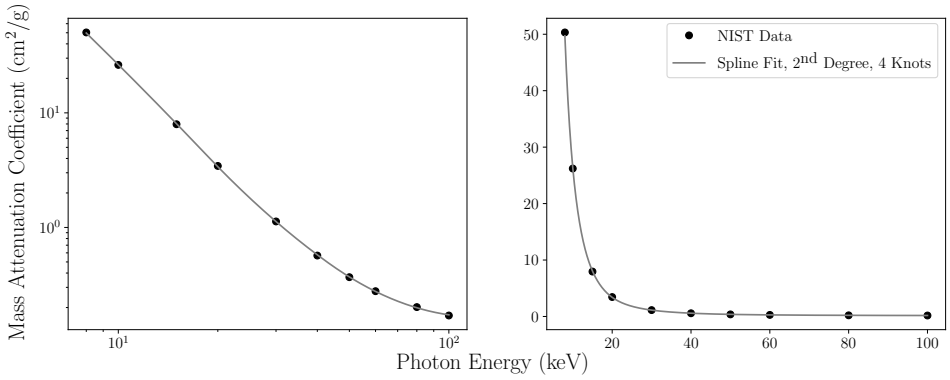
To obtain the relative intensity values in the background and center of an image, the average pixel value was taken over a square region in the desired sections of the image. With this relative intensity, extracted with the Python code in Listing 2, and the thickness of the object, the mass attenuation coefficient and effective energy of the interaction were determined using Equation 2.1 and NIST's database.

```python
        def get_relative_intensity(path, background_intensity=0):
        """
        Calculates the relative intensity of a square in the center of an image.
        :param path: Path to image
        :param background_intensity: Intensity of background
        :return: Relative intensity
        """
            # load image
            img = cv2.imread(path, 0)

            # convert to float and normalize
            img = img.astype(np.float32)/255

            # get mean pixel value and std dev
            # of square in center of image
            center = (int(img.shape[0]/2), int(img.shape[1]/2))
            length = 100
            square = img[int(center[0]-length/2):int(center[0]+length/2),
            int(center[1]-length/2):int(center[1]+length/2)]

            mean, std = cv2.meanStdDev(square)

            # if intensity is 1 or 0, return False,
            # so they are not used as data points
            if mean[0][0] >= 0.999 or mean[0][0] <= 0.001:
                return False

            # obtain relative intensity
            # by normalizing by background intensity
            # (I_square = I_background * I_object -> I_object = I_square /
            ↪  I_background)
            mean[0][0] = mean[0][0] / background_intensity


            # convert image to rgb
            img = cv2.cvtColor(img, cv2.COLOR_GRAY2RGB)

            # draw red square on image
            # and relative intensity and std on image
            cv2.rectangle(img, (int(center[0]-length/2),
            ↪  int(center[1]-length/2)),
            (int(center[0]+length/2), int(center[1]+length/2)), (0, 0, 255), 3)

            cv2.putText(img, f"Intensity: {mean[0][0]:.3f} +- {std[0][0]:.3f}",
            ↪  (int(center[0]-length/2),
                        int(center[1]-length/2)-10), cv2.FONT_HERSHEY_SIMPLEX, 1,
                        ↪  (0, 0, 255), 2)

            return ufloat(mean[0][0], std[0][0])
```

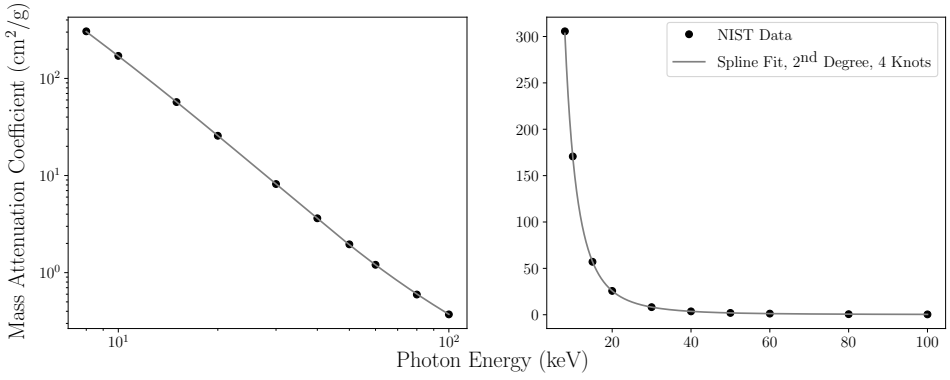
Listing 2: X-ray image relative intensity extraction code implemented in Python3.

2.2 Results

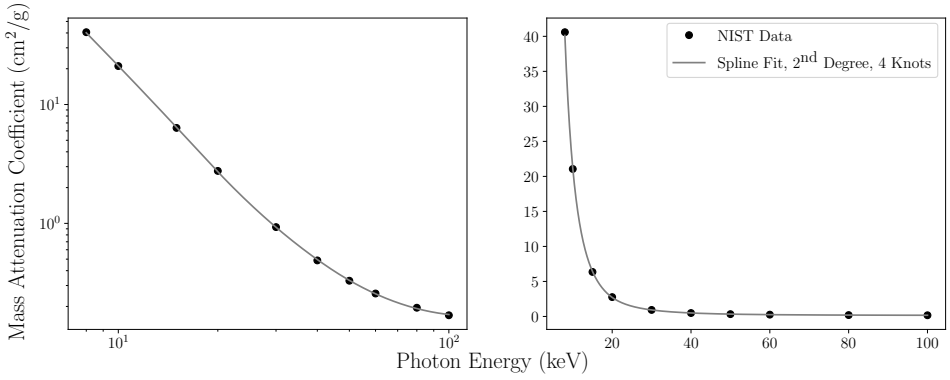
To relate the calculated attenuation coefficient of a particular object to the effective energy, a 2nd degree, 4 knot B-spline was used to fit NIST's mass attenuation coefficient data [3]. This fit, implemented with the Python package Scikit-Learn [6], provided $\mu_m = f(E_{\text{eff}})$; however, $E_{\text{eff}} = f(\mu_m)$ was needed. The described fits are shown in Figure 2.3.



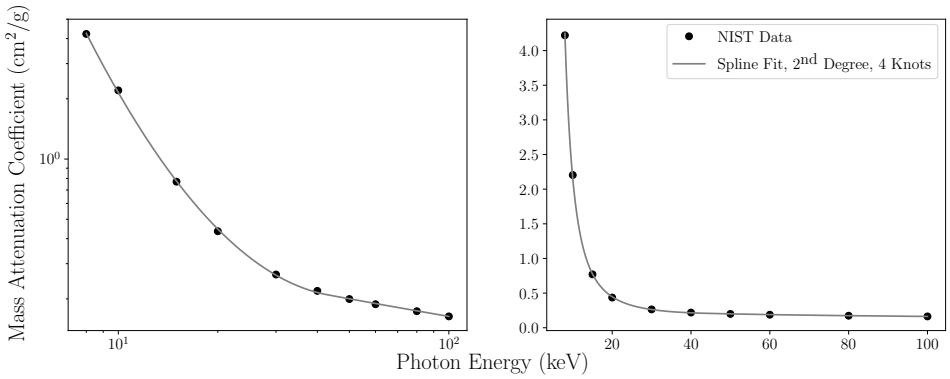
(a) Al



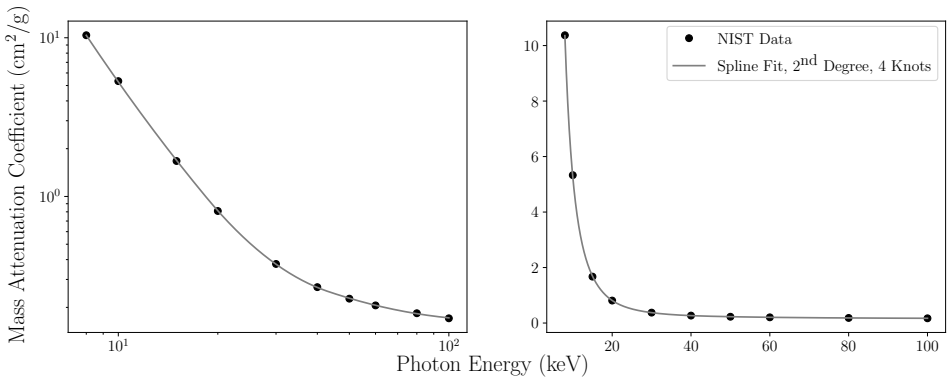
(b) Fe



(c) Mg



(d) Plastic Scintillator



(e) Water

Figure 2.3: The NIST mass attenuation coefficient data (in cm^2/g) vs. effective energy (in keV) for the Al, Fe, Mg, plastic scintillator, and water fitted using the described B-Spline. (Continued)

Because f represents an abstract model, an inverse was unable to be taken, so an evenly spaced E_{eff} array of 10,000 values ranging from 0.8 keV to 100 keV was substituted into our model, providing its associated μ_m array. Given a particular $\mu_m \pm \sigma$, where σ represents the sample standard deviation (Std.) of the mass attenuation coefficient, the E_{eff} was found by first determining the index of the element in the μ_m array closest to the particular μ_m. The associated E_{eff} was then the element in the E_{eff} array located at the determined index. The range of error for a particular E_{eff}, [E_{eff} Min, E_{eff} Max], was calculated by repeating the indexing process for $\mu_m + \sigma$ and $\mu_m - \sigma$. The results of this indexing technique for each X-ray image can be found in Table 2.2.

Material	kVp (kV)	I/I_0	I/I_0 Std.	μ/ρ (cm^2/g)	μ/ρ Std. (cm^2/g)	E_{eff} (keV)	E_{eff} Max (keV)	E_{eff} Min (keV)
Aluminum	40.0	0.310	0.025	1.895	0.131	24.552	25.215	23.954
	42.0	0.425	0.026	1.385	0.099	27.635	28.444	26.917
	44.0	0.567	0.025	0.920	0.071	32.576	33.698	31.591
	46.0	0.668	0.023	0.655	0.056	37.747	39.302	36.403
	48.0	0.727	0.023	0.516	0.052	42.145	44.380	40.277
	50.0	0.888	0.020	0.192	0.037	83.926	100.000	69.711
Iron	40.0	0.353	0.030	13.239	1.078	25.316	26.089	24.626
	42.0	0.432	0.031	10.668	0.898	27.331	28.196	26.558
	44.0	0.595	0.030	6.595	0.642	32.392	33.588	31.352
	46.0	0.678	0.027	4.937	0.514	35.870	37.287	34.646
	48.0	0.761	0.028	3.473	0.461	40.599	42.687	38.860
	50.0	0.911	0.024	1.186	0.340	60.335	69.195	54.704
Magnesium	40.0	0.632	0.030	2.638	0.274	20.200	20.982	19.529
	42.0	0.832	0.029	1.054	0.200	28.444	31.058	26.531
	44.0	0.874	0.027	0.776	0.176	32.382	36.412	29.668
	46.0	0.909	0.027	0.551	0.172	37.967	46.193	33.431
	48.0	0.940	0.024	0.353	0.145	48.162	72.627	39.936
Plastic Scintillator	40.0	0.578	0.03	0.268	0.025	29.392	32.861	27.000
	42.0	0.645	0.03	0.215	0.023	40.341	57.906	33.845
Water	44.0	0.059	0.025	0.489	0.075	25.500	28.076	23.669
	46.0	0.220	0.030	0.261	0.024	41.096	46.984	37.369
	48.0	0.325	0.029	0.194	0.015	68.652	86.603	58.403

Table 2.2: The results for each non-saturated (intensities not equal to 0 or 1) X-ray image for the composition experiment.

The graphs of the E_{eff} for each kVp and material combination can be found in Figure 2.4. To validate the experimental results, simulated E_{eff} values were obtained with the Python module SpekPy [7], using the X-ray tube assembly specifications as provided by the manufacturer [1].

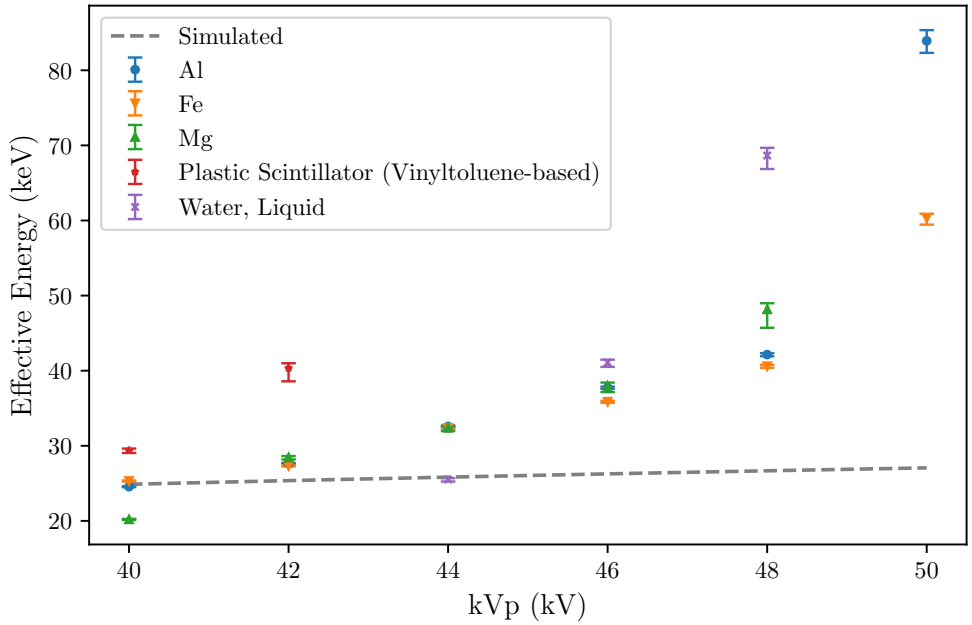


Figure 2.4: Graph of effective energy (in keV) vs. kVp (in kV) for the X-ray image data contained in Table (2.2).

Figure 2.4 shows relative agreement and an approximately linear increase for the Al, Fe, and Mg, up until 50 kVp, where a sudden spike in E_{eff} is observed. For the plastic scintillator and water, vastly different values are observed. Upon further inspection by increasing the range of the fit shown in Figure 2.3, the two values increase at an exponential rate. Furthermore, the rate of increase of E_{eff} with respect to kVp is greater for the monoatomic attenuators in the experiment compared to their simulated values.

For the offset experiment, the intensity values for the image with the centered Al (labeled N) were obtained with the extraction code shown in Listing 2. On the other hand, the intensity values for the image with the offset Al were obtained manually using the software ImageJ. [8]. The results of this experiment can be found in Table 2.3.

Position	I/I_0	I/I_0 Std.	μ/ρ (cm^2/g)	μ/ρ Std. (cm^2/g)	E_{eff} (keV)	E_{eff} Max (keV)	E_{eff} Min (keV)
Center	0.310	0.025	1.90	0.13	24.55	25.21	24.00
Offset	0.14	0.05	3.2	0.5	20.5	21.9	19.4

Table 2.3: The results of X-ray images of a centered and offset Al (labeled N) at 40 kVp for use in the offset experiment.

The results of this experiment show unexpected discrepancies between the centered and offset Al. A slight difference in E_{eff} is expected due to the point-like nature of the x-ray source; however, it can not explain the statistical inequivalency observed due to the large distance (44.95 cm)[1] between the x-ray source and detector; therefore, demonstrating a dependence on the object's position in the imaging area for the E_{eff}.

For the thickness-varying experiment, x-ray images of Al were taken at 40 kVp. The thickness of the Al was varied by using two different attenuators, one of thickness 2.286 mm (labeled N) and the other of thickness 0.813 mm (labeled I), as found in Table 2.1. The intensity values for each image were obtained with the code in Listing 2. The results of this experiment can be found in Table 2.4.

Thickness (mm)	I/I_0	I/I_0 Std.	μ/ρ (cm^2/g)	μ/ρ Std. (cm^2/g)	E_{eff} (keV)	E_{eff} Max (keV)	E_{eff} Min (keV)
2.286	0.310	0.025	1.90	0.13	24.55	25.21	24.00
0.813	0.575	0.029	2.52	0.23	22.15	22.92	21.39

Table 2.4: The results of X-ray images of Al (labeled N) and Al (labeled I) at 40 kVp for use in the thickness-varying experiment.

The results of this experiment, once again, show unexpected discrepancies in E_{eff} between the two thicknesses. While the E_{eff}'s are closer than what was measured for the offset experiment, the two values are statistically inequivalent; therefore, demonstrating a thickness dependence of the E_{eff}.

2.3 Discussion

The discrepancies between E_{eff} for the monoatomic materials and the compounds have been explored extensively with little resolution. Several different theories have been explored, including the thickness of the sample, possible code errors, and the normalization of the image.

Upon looking for differences between the compounds and monoatomic materials, it was observed that both the compounds had a significantly larger thickness than the monoatomics, as seen in Table 2.1. It was then considered whether or not the increased height of the sample could lead to an increased probability of x-rays scattering off the object and onto the detector. This effect would result in a seemingly larger background and could lead to unexpected E_{eff} values. Upon manual examination of the images, the background was affected; however, the difference in intensity was minor and could not reasonably result in the exponential increase that was measured, especially since the discrepancies are still present in images where the background is non-existent.

Furthermore, since weighted averages of monoatomic attenuation coefficients were taken in the data processing code to obtain the attenuation coefficients for the compounds, it was considered whether or not a possible error was present in the code that could lead to this exponential behavior. Upon careful examination of the code, no errors were found, and for extra reassurance, a few images of the compounds were manually inspected, and verified the processing code's results.

Additionally, it was found that *Heine & Thomas* (2008) [2] performed a similar E_{eff} calibration using a mammography system. While their overall technique was similar to the methodology of this experiment, one key difference was the normalization factor used

in the processing of the intensity data. In particular, they normalized the pixel value by mAs, which is the tube current multiplied by the exposure time. This effectively would be the incident intensity of the x-rays and is equivalent to I_0. However, this normalization was not able to be done for this experiment, since the C-arm does not provide any data on the exposure time for images. Still, this normalization would be the same for each kVp; therefore, this cannot explain the drastic differences between the E_{eff}'s of the monoatomic and compound materials that were measured at higher kVps. On the other hand, since the current of the tube was provided for each image and increased at a rate proportional to the kVp, this could explain why the E_{eff}'s of the monoatomic materials increased at a rate greater than what was predicted by the simulation.

Finally, combining these results with the unexpected differences in E_{eff} that were measured in the offset and varied thickness experiments, it is possible the x-ray machine may be doing some sort of post-processing of the images that are not described in the manual [1]. Without knowing the specifics of this processing, the original, unfiltered images can not be calculated. Overall, the measurements obtained from this experiment reveal that the E_{eff} approach for ray-tracing cannot be used for the GE MiniView 6800 Mini C-Arm.

Bibliography

[1] "Ge Mini View 6800 Mini C-Arm". en. In: *Soma Technology Inc.* () (cit. on pp. 19, 20, 22).

[2] J. J. Heine and J. A. Thomas. "Effective x-ray attenuation coefficient measurements from two full field digital mammography systems for data calibration applications". In: *BioMedical Engineering OnLine* 7.1 (2008-03), p. 13. ISSN: 1475-925X. DOI: 10.1186/1475-925X-7-13. (Visited on 2023-04-28) (cit. on p. 21).

[3] J. H. Hubbell. "Photon mass attenuation and energy-absorption coefficients". In: *The International Journal of Applied Radiation and Isotopes* 33.11 (1982), pp. 1269–1290 (cit. on pp. 9, 11, 13, 17).

[4] T. Möller and B. Trumbore. "Fast, minimum storage ray/triangle intersection". In: *ACM SIGGRAPH 2005 Courses*. 2005, 7–es (cit. on p. 5).

[5] J. Peddie. *Ray Tracing: A Tool for All*. en. Cham: Springer International Publishing, 2019. ISBN: 978-3-030-17489-7 978-3-030-17490-3. DOI: 10.1007/978-3-030-17490-3. URL: http://link.springer.com/10.1007/978-3-030-17490-3 (visited on 2023-05-12) (cit. on pp. 1, 2).

[6] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on p. 17).

[7] G. Poludniowski et al. "Technical Note: SpekPy v2.0—a software toolkit for modeling x-ray tube spectra". en. In: *Medical Physics* 48.7 (2021-07), pp. 3630–3637. ISSN: 0094-2405, 2473-4209. DOI: 10.1002/mp.14945. URL: https://onlinelibrary.wiley.com/doi/10.1002/mp.14945 (visited on 2023-04-28) (cit. on p. 19).

[8] C. A. Schneider, W. S. Rasband, and K. W. Eliceiri. "NIH Image to ImageJ: 25 years of image analysis". en. In: *Nature Methods* 9.7 (2012-07), pp. 671–675. ISSN: 1548-7091, 1548-7105. DOI: 10.1038/nmeth.2089. URL: http://www.nature.com/articles/nmeth.2089 (visited on 2023-04-28) (cit. on p. 20).

[9] J. A. Seibert. "X-Ray Imaging Physics for Nuclear Medicine Technologists. Part 1: Basic Principles of X-Ray Production". en. In: *JOURNAL OF NUCLEAR MEDICINE TECHNOLOGY* 32.3 (2004) (cit. on p. 12).

[10] R. A. Serway, C. J. Moses, and C. A. Moyer. *Modern physics*. 3rd ed., student's ed. Thomson Brooks/Cole, 2005, p. 592 (cit. on p. 9).

[11] P. Shirley. *Ray Tracing in One Weekend*. 2020-12. URL: https://raytracing.github.io/books/RayTracingInOneWeekend.html (cit. on pp. 3, 4).