Assigner: Alexander Hill
Contact: a.d.hill@liverpool.ac.uk
Deadline: TBC January 2023

Introduction to C++: Monte Carlo Project
Assignment 1

---

1 Introduction

Digital Tomosynthesis (DT) is a technique that has the potential to challenge the hegemony of Computed Tomography (CT) scanners in the field of X-ray imaging. Adaptix, a company with which the University of Liverpool collaborates closely, is interested in building an array of flat-panel sources (FPS) of X-rays to enable bedside 3-dimensional X-ray imaging. An FPS is comprised of an array of **emitters**, each providing a source of X-rays.

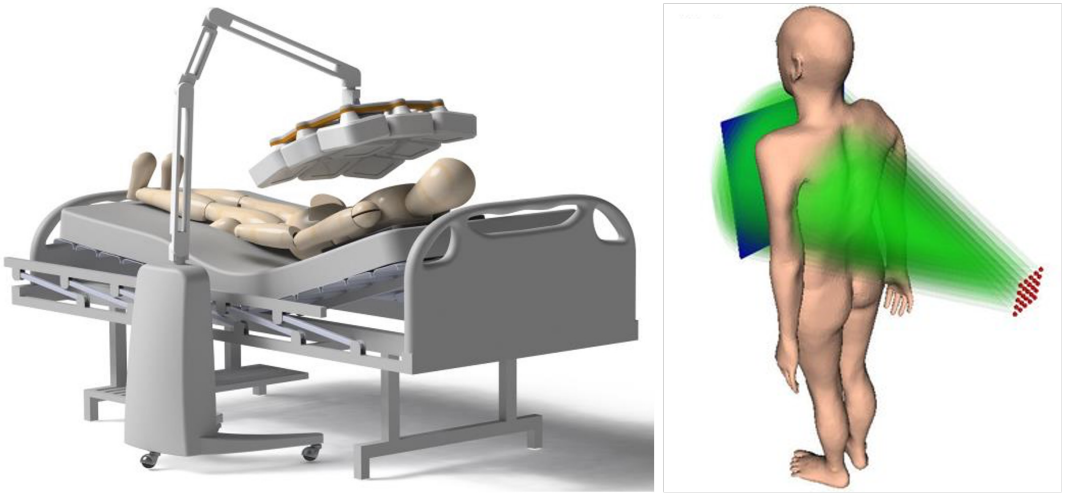

Figure 1: Left: a depiction of the proposed bedside DT scanner in operation. Right: a Geant4 simulation of one FPS in operation. The FPS is comprised of an array of 5x5 emitters.

In the construction of DTs, the flux of X-rays incident on the patient is important for two reasons. Firstly, we need to ensure that the radiation dosage is below certain acceptable levels, and secondly, we need to ensure that the radiation flux per unit area is sufficient to enable resolution enough to, for example, observe small tumours and breaks. Radiation 'leakage' is also to be avoided, i.e. we need to ensure that X-rays are not illuminating areas too far beyond the detector.

Previous members of LIV.DAT have constructed simulations to investigate the practicalities of the setup proposed by Adaptix (e.g. Primidis et al., 2021). In its most basic form, this simulation assumes that the beam of emission from each emitter takes the form of a cone, with its intensity described with a 2-dimensional Gaussian with a sharp truncation at a certain point. The beams from each emitter combine to produce the overall X-ray emission.

This raises the question whether or not this truncation is a fair thing to do. While computationally necessary to include in the Geant4 simulations, it is possible that the truncation leads us to underestimate the radiation leakage. Your task will be to construct a Monte Carlo simulation in C++ to test this.

The below tasks are the outline of a reasonably involved project. Do not worry if you do not make it as far as you would like through the bullet points, the important thing is that you have a chance to work with C++ on a project and gain an appreciation for the uses of Monte Carlo. That said, the more you put into it the more you get out, so please try to get as far as you can.

2 Outline

1. **Step One: One dimension, one emitter.**

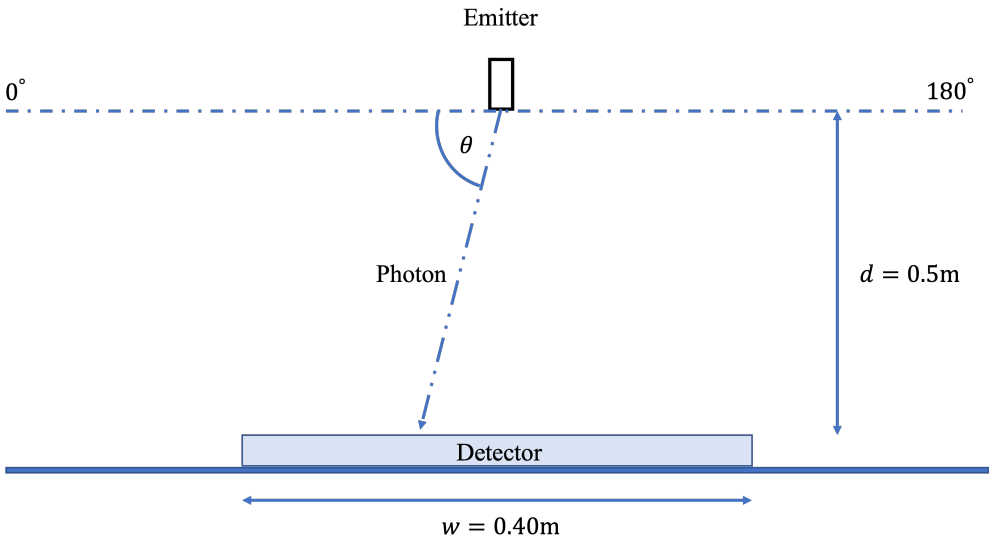

Figure 2: An illustration of the setup described in the first step.

(a) Aim: construct a simple Monte Carlo simulation that shows the spread of expected hits on a detector from a single emitter in one dimension. See Fig. 2 for a diagram of this setup.

(b) Build a random number generator to sample emission angles, θ, according a Gaussian distribution. The distribution should be bounded by 0° and 180°, have a mean of $\mu = 90°$, and a standard deviation $\sigma = 5°$. For this you should refer to `https://cplusplus.com/reference/random/normal_distribution/`.

(c) Use a curve-fitting algorithm to ascertain that the spread of samples drawn closely approximates the Gaussian probability distribution function you used to sample θ[1]. How many draws, n, did you need to recover μ and σ to within 1%?

(d) Given a distance $d = 0.5$m between the emitter and the detector, construct a histogram of predicted detection locations for n photons sampled from the above distribution.

(e) Assuming that the centre of the emitter lines up with the centre of the detector, and the width of the detector is $w = 0.4$m, what fraction of photons fall outside the detector?

(f) Create a new probability distribution function based on the above Gaussian, with the addition that any draws with θ outside ±10° of 90° is retaken. This models the sharp cone mentioned in the introduction. Compare the resultant detection histogram with the case of a regular Gaussian. Do the same for a uniform distribution between 80 and 100°.

[1]Try to do this in C++, which will require some research. If needed, use this Python package - `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html`

2. Part Two: One dimension, multiple emitters.

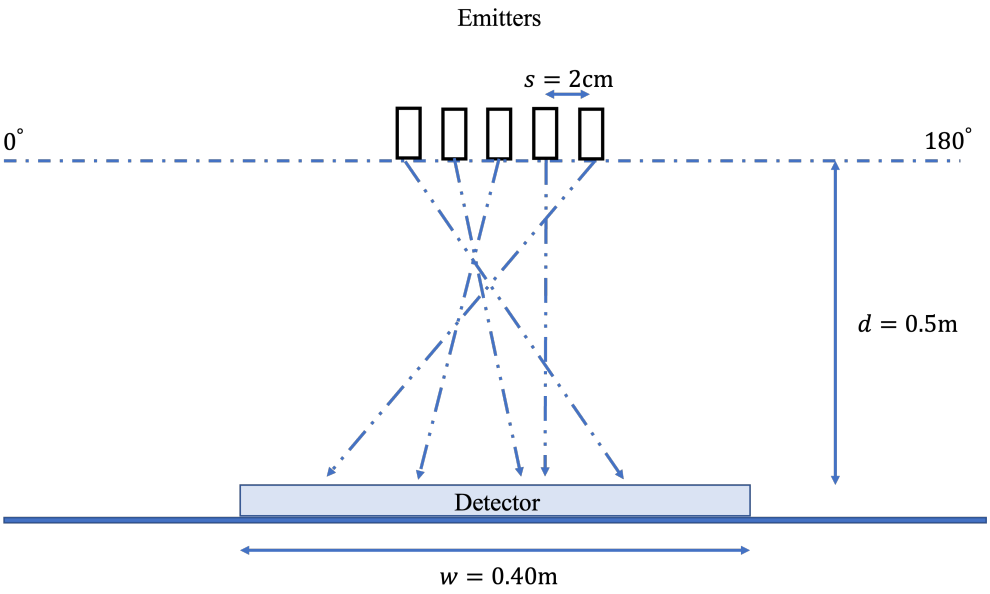

Figure 3: An illustration of the setup described in task two.

(a) Aim: construct a Monte Carlo simulation in one dimension for the case with multiple emitters. A depiction of the scenario is identified in Fig. 3.[2]

(b) Construct an odd number of emitters (three or five), each separated by a distance of 2cm. The central emitter should align with the centre of the detector. If n is the number of draws you found in the previous step to result in $< 1\%$ error, fire n photons from each of the emitters at the detector assuming (a) the regular Gaussian and (b) the Gaussian truncated at $\pm 10°$ of $90°$. In the determination of detection location, you will need to separately calculate the geometry for each emitter, so try to keep such calculations generalisable and dependent on only s.

(c) Compare the detection histograms for the two PDFs. Is the detection count under/overestimated in some areas rather than others?

[2]Consider constructing an 'Emitter' class. Attributes could include things like number of photons, probability distribution sampled from, distance to the detector, and offset from the centre of the detector.

3. **Part Three: Two dimensions/open ended.**

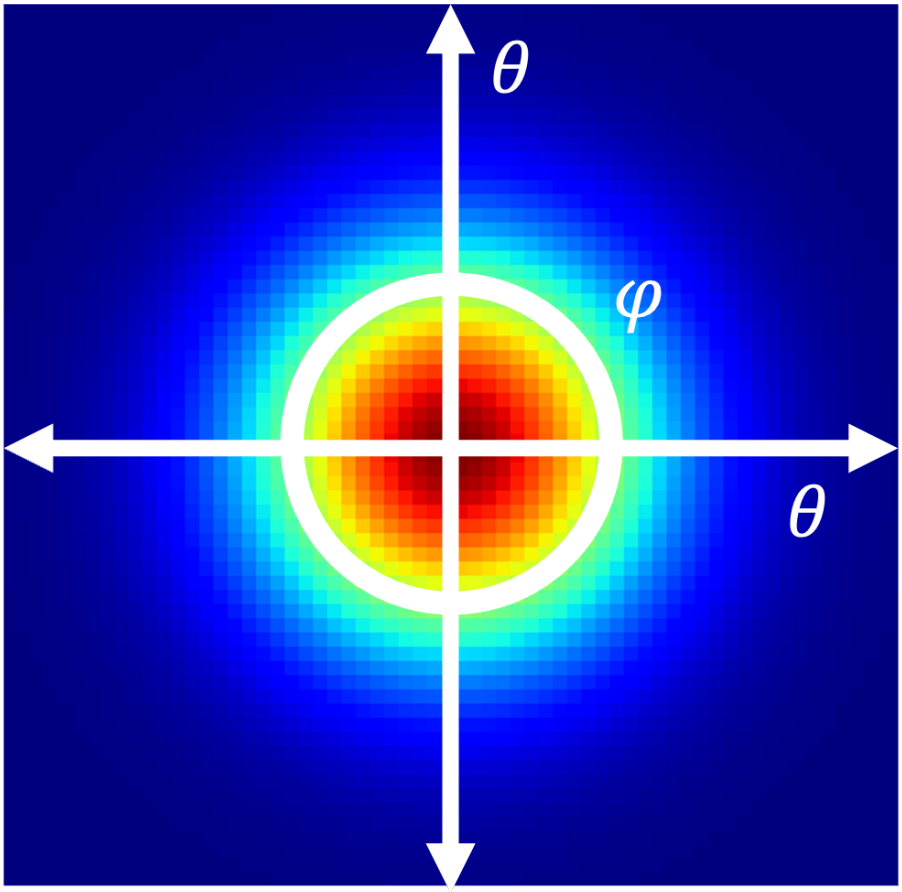

Figure 4: A 2-dimensional Gaussian probaility distribution function, described by angles θ and ϕ.

(a) Aim: extend the above study into two dimensions. This is open ended; while here I discuss only one emitter, please extend this to a 5×1 line or a 5×5 grid of emitters if you are able.

(b) In our new framework, assume that the source-to-detector distance is still 0.5m, and that the detector is 0.4m^2 in area. Consider a single emitter emitting in two dimensions. Photons will be emitting at a radial angle θ from the normal, and at a polar angle ϕ with respect to an arbitrarily defined 'north' or 'zenith'. First, we will need to sample from a 2-dimensional Gaussian distribution. One way of doing this would be to draw θ and ϕ separately for each photon. While θ is ostensibly sampled from a normal distribution bounded by 0 and 180°, ϕ is sampled from a uniform distribution between 0 and 360°. An illustration of a 2-dimensional Gaussian distribution is shown in Fig. 4.

(c) Explore the effects of Gaussian truncation. Where is intensity underestimated/overestimated in the case of real, continuous Gaussian rather than those truncated?

(d) If you've made it this far, consider changing hyper-parameters like s and d. Find any interesting results you can.

References

Primidis T., Wells S., Soloviev V., Welsch C., 2021, http://dx.doi.org/10.1088/2057-1976/ac3880 Biomedical Physics & Engineering Express, 8