Designing Configurable IR Tomography Systems

Jonathan Whitaker, WHTJON002

October 22, 2018

UCT Department of Electrical Engineering

Declaration 1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretending that it is one's own. 2. I have used the <IEEE/Harvard> convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced. 3. This report is my own work. 4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it of as their own work or part thereof. Name: Jonathan Whitaker Student Number: WHTJON002 Date: October 2018

Acknowledgements

My God - the rock who is higher than I

My fiancée Eleanor, for the work parties, the many cups of tea, the helpful editing, the support and so much more

My friend and mentor Clare Griffiths, who showed me what a compassionate scientist looks like

My housemates , who put up with the noise of my machines (Alex, Chama), played my game (Norbert) and supported me (all)

My parents

My supervisor Prof. Andrew Wilkinson

Abstract

This project investigates the use of IR light for tomography, with a focus on sparse arrangements of emitters and detectors. Code is developed to test arbitrary sensor geometries in simulation. A rotating scanner is constructed to test these different geometries experimentally. Based on these early tests, two fixed arrangements are constructed and tested. Machine learning is investigated as a tool for making inferences based on the raw scan data, and a neural network based approach is demonstrated that can identify objects and infer their position with high accuracy.

Contents

List of Figures

List of Acronyms and Abbreviations

CT - Computed Tomography
IR - Infra-Red
ML - Machine Learning
fMRI - functional Magnetic Resonance Imaging
FBP - Filtered Backprojection
PET - Positron-Emission Tomography
ANN - Artificial Neural Network
MLP - Multilayer Perceptron
LED - Light Emitting Diode
PT - Phototransistor
GPIO - General Purpose Input/Output
CAD - Computer Aided Design
PLA - Poly-lactic Acid (a thermoplastic)
ADC - Analog to Digital Converter
CSV - Comma-Separated Variable
RMSE - Root Mean Squared Error
MAE - Mean Absolute Error
PCB - Printed Circuit Board

1 Introduction

Computed tomography (CT) is the process of building up an image of a cross-section through an object, usually done by measuring how much radiation of one kind or another passes through the object from different directions and then processing these measurements to reconstruct the image. The most common type of radiation used is X-ray radiation. However, nothing is stopping us from using other types, and for my project, I wanted to investigate using light - specifically near infra-red (IR) light.

1.1 Objectives

The goal of this project was to investigate the use of IR light for tomography. I wanted to see what was possible, to gain a better understanding of how tomography systems work, to see what I could achieve with small, fixed arrays of LEDs and phototransistors (PTs) and to potentially put my machine learning expertise to use in this field. When conceptualizing the project, I outlined the following steps I planned to take:

- Building a simulator. This would let me test different arrangements of sensors and emitters virtually. With this, I could begin building my intuition about what would work and what wouldn't. I could also start work on reconstruction algorithms and analysis without needing to build anything.
- Developing reconstruction algorithms. These will be needed throughout the project, both for validating the simulator and generating images from real-world data.
- Testing different arrangements in the real world. Some sort of motion platform would be needed to position an emitter and a detector in different locations around an object being scanned. This would let me simulate different arrangements experimentally, and give some valuable real-world information. It would also help me learn what these systems are capable of.
- Building some fixed arrangements. Choosing some fixed arrangements based on simulation and experimentation, and using them for different tasks. Can I reconstruct an image with so little information? Can I use machine learning to infer the position of an object within the ring of sensors? These will be used as case studies to show how useful these systems can be.
- Several 'stretch goals', most of which were beyond the scope of this project. Improving ML model performance with transfer learning, improving the simulator by modelling the behaviour of light (diffusion, diffraction etc), speeding up the simulation, model training and image reconstruction by utilising the GPU, etc.

These steps became my main objectives, and were followed fairly faithfully, with several different tasks being worked on concomitantly at any given time.

1.2 Motivation

I have been thinking about tomography for several years, beginning with some work I did on fMRI data in my first year and an abortive attempt to make a scanning electrical impedance tomography system soon afterwards. Although I had some understanding of how such systems work, I found that I didn't have any intuition for them. How many projections did I need to get a reasonable image? Would it be better to have more detectors, or fewer detectors scanned from more angles? Hence the emphasis throughout this project on exploration and testing of different configurations - all of these experiments have given me a much better understanding of how these systems behave.

As for my motivation for choosing this particular project, I must confess that it was at least partly informed by what hardware I had on hand. I knew I wanted to do a tomography project, and I was eager to get started. I came up with the idea of doing IR tomography at the end of 2017. I already had a few phototransistors available, and I had lots of stepper motors and drivers around if I needed a system that could scan. I was also considering the purchase of a 3D printer and could thus plan on having access to both a reliable motion platform and an easy way to rapidly prototype the mechanical parts. This meant that, if I really wanted to, I could have a good portion of my

project done by the time the second semester started and parts became available. For practical reasons, I only started work a week or so early, but the time spent planning and researching meant that I hit the ground running and began the semester with some good progress.

1.3 Background

When we look at how a standard CT scanner works, we see something like the arrangement shown in Figure 1.

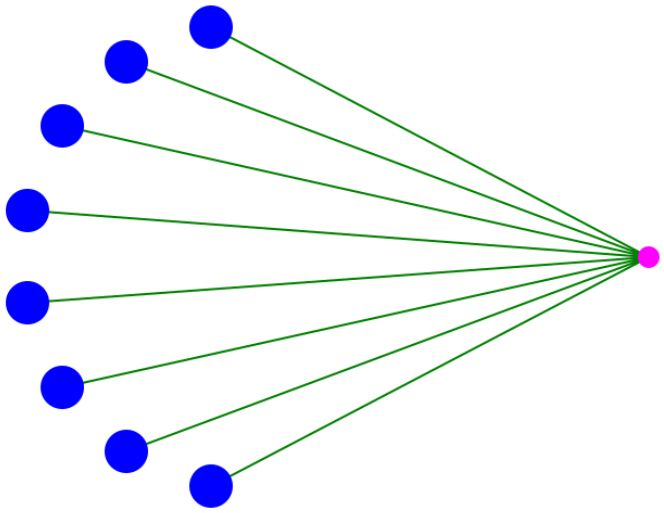


Figure 1: Typical sensor geometry in a CT scanner

One emitter (pink) and an array of detectors (blue). We measure how many X-rays make it to each detector and that gives us some information about what we're scanning along that path. By rotating the whole arrangement around the object being scanned, we get multiple 'views'. The more views, and the more detectors, the better the final image.

Say we want to do something similar with IR light as opposed to X-rays. We could simply mirror this arrangement, and scan around to get an image based on how much IR light gets through along each path. But IR emitters are cheap, and moving parts take time. What if we added more emitters, and tried to capture all our paths simultaneously?

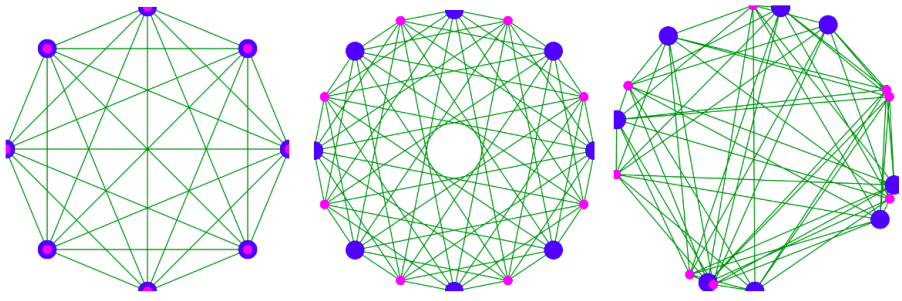


Figure 2: Potential arrangements for IR Tomography

We could potentially get a low-resolution image with no moving parts. Enough to locate a large object somewhere in the field of view, for example. And by scanning, we could increase the resolution where needed. But how do we know which arrangement is best? And for less trivial arrangements like the one shown on the right in Figure 2, how do we go from the measurements back to an image?

1.4 Scope and limitations

Although one goal of this project is to deliver some functioning IR tomography systems, its main purpose is to provide a foundation upon which future work can build. I had hoped to investigate the use of such systems for medical imaging but will have to leave that for another time.

The main limiting factor in a project such as this is time. The design of the rotating table was largely predicated on what parts were available immediately, and so progress could start straight away. However, the IR LEDs and phototransistors (PTs), as well as the Teensy boards used to control them in the fixed arrangements, were only received by mid-September. Given the remaining time, larger and more complex designs were not investigated. GPU programming or advanced ray-tracing for the simulation would also have taken a long time to implement without necessarily adding much to the end results.

1.5 Plan of Development

This report begins with a literature review, examining some existing optical tomography systems and providing background on the algorithms and components used throughout the project. After this, I introduce the method by which I simulate different scanning topologies, and then the way these are tested experimentally using a rotating platform developed for the purpose. I present two fixed arrangements developed based on results from simulation and compare their performance with the expected results. Finally, I present some potential applications, suggestions for improvement, conclusions and ideas for future work.

2 Literature Review

2.1 Optical Tomography

Optical tomography, alternately called visible light tomography (or near infrared tomography if the wavelengths used are outside the visible spectrum), uses light instead of X-rays or other radiation [1]. This has several advantages over other imaging methods, including lower cost, less danger (in the case of X-ray tomography) and lower complexity compared with methods such as MRI. It is also possible to take advantage of the various properties of light to image different things. For example, optical coherence tomography (OCT) uses interferometry to create a cross-section of tissue [2]. Optical tomography systems can also make use of multiple wavelengths of light to differentiate between similar materials [3].

Of course, optical tomography has some disadvantages as well. Light is easily reflected, scattered or diffracted, making reconstruction difficult. Additionally, many materials and tissues in the real world are completely opaque to radiation in this segment of the electromagnetic spectrum, making it impossible to image inside objects that block incoming light. There are some partial solutions, but these obstacles have historically meant that optical CT is rarely used outside of a few niche applications.

2.1.1 Example Applications

Optical tomography, especially a technique called diffuse optical tomography, has been used in various medical imaging applications [4]. It is especially effective at imaging breasts for breast cancer detection [5] and for non-invasive imaging of the brain and brain activity [6, 7]. Tissue tends to scatter light, but a lot of work has been done modelling the passage of light and going from readings back to images. Arridge [8] gives an overview of the problems and some of the algorithms and approximations used to solve them.

One niche application where optical CT has been extensively used is gel dosimetry. To measure how much radiation is being delivered to a target area in radiotherapy treatments, a clear polymer gel is irradiated as a patient would be. As radiation hits the gel, it turns more and more opaque, with the amount of radiation (the dose) corresponding to the final opacity of the gel. Imaging the irradiated gel sample reveals the shape of the irradiated area and the intensity of the radiation within that area, ensuring that the correct dose is delivered. Gore et al (1996) [9] describe a system for imaging cross-sections of a gel dosimetry sample, shown in Figure 3.

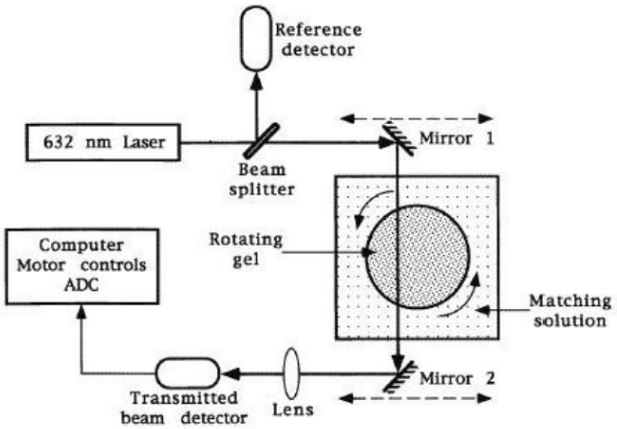


Figure 3: Schematic of an optical tomography system for measuring light transmitted through a gel sample (from [9]).

This arrangement allows multiple projections to be made through the sample from different angles. It avoids some of the common issues with optical systems by surrounding the sample with a solution whose refractive index is carefully matched to the gel, ensuring that the light travels in a straight line. This means that the rig can effectively measure attenuation without having to

account for refraction. Even a relatively small difference in refractive index between the gel and the surrounding solution can result in geometric distortions [10], so this step is important. Even with the constraints associated with this method, it is still considered superior in some cases and commercial machines are available and in use today [10].

Aside from these medical applications, optical-CT has also been investigated for use in industrial applications such as characterising foam structures or foam density [11, 12] or analysing multi-phase flows. While these problems are traditionally solved with techniques such as electrical resistance tomography [13], electrical capacitance tomography [14] or even X-ray tomography [15], optical-CT has the potential to provide lower-cost solutions in some cases.

2.2 Image Reconstruction

For tomography to work, we need some way to go from a set of readings back to an image of the cross-section we are scanning. This section will discuss some of the ways this is generally done.

2.2.1 The Radon Transform

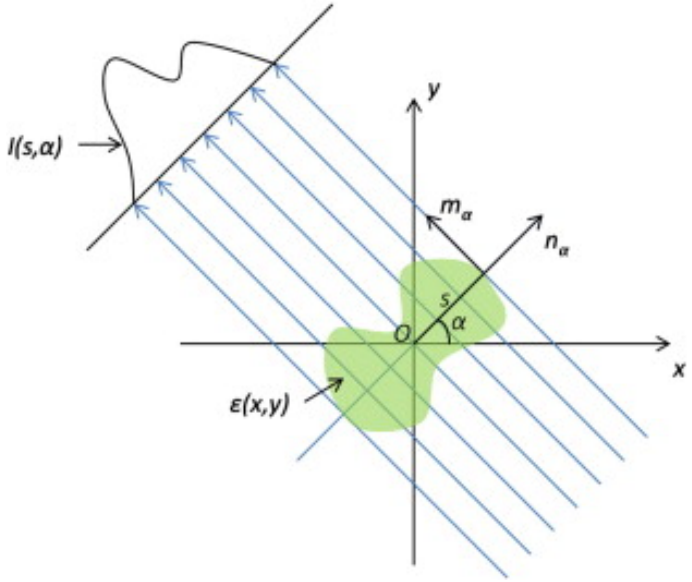


Figure 4: The Radon Transform (From [16])

When scanning, we measure how much radiation passes through the scanned object. Figure 4 illustrates this - for a given angle α, we measure the transmitted light along different paths, each characterised by the distance from the origin, s. Scans typically involve multiple projections from different α's. The output of this process is a sinogram, with transmitted intensity mapped to color as a function of s and alpha. This is called the radon transform. The value for a given s and α is given by the line integral of the attenuation coefficient (described by the function $f(x, y)$) along the path $xcos\alpha + ysin\alpha = s$ [17]. Thus the radon transform can be defined as

$$R(s, \alpha) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)\delta(xcos(\alpha) + ysin(\alpha) - s)dxdy$$

Because we're limited to finite numbers of detectors and projections (angles to scan from), we must consider the discrete case. Beylkin[18] describes it in detail, but to summarize:

$$R(s, \alpha) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)\delta(xcos(\alpha) + ysin(\alpha) - s)$$

To estimate the total attenuation along a given line, we sum the attenuation coefficients for points along that line. This is how it is implemented in the simulation described in Section 3 - an

image whose greyscale values encode the attenuation coefficient of an object is sampled at regular points along a line and the sum of these values is used to calculate transmitted intensity.

2.2.2 Filtered Back Projection

Given the radon transform of an image, how do we go back to the original image? There are different techniques available, but the most common is filtered back projection [19]. This is essentially the inverse of the radon transform. For the discrete case, with N projections:

$$f(x,y) \approx \sum_{n=0}^{N-1} R(xcos(n2\pi/N) + ysin(n2\pi/N), n2\pi/N)$$

As mentioned, other algorithms are also available. For example, algorithmic reconstruction techniques use an iterative approach to approximate a solution [20]. These can be computationally intensive but, in some cases, provide a better reconstruction.

2.2.3 Different Geometries

So far in this section we have been considering what is known as the parallel-beam geometry (Figure 5, left). However, most modern scanners use an arrangement like that shown on the right of Figure 5, known as the fan-beam geometry.

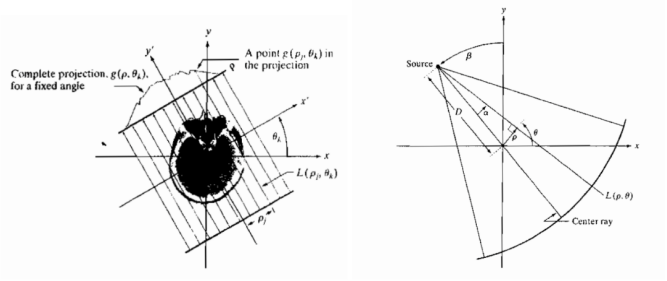


Figure 5: Parallel beam (left) and Fan-beam (right) geometries (from [17])

Gonzalex and Woods [17] explain the derivation of a suitable reconstruction algorithm for fan-beam geometries in Chapter 5.11 of their book, beginning with the reconstruction algorithm for the parallel case, converting to polar coordinates (r, γ), then performing a transform of coordinates to allow integration with respect to α and β (see Figure 5), eventually arriving at the following formula:

$$f(r,\gamma) = 1/2 \int_0^{2\pi} \int_{-\alpha_m}^{\alpha_m} g(Dsin\alpha, \alpha+\beta) s[rcos(\beta+\alpha-\gamma) - Dsin(\alpha)] Dcos(\alpha) d\alpha d\beta$$

Where D is the distance from the source to the origin, α_m is the maximum α considered and s[...] represents an Inverse Fourier transform. See [17] for a full explanation.

2.2.4 The ASTRA toolbox

Although implementing some of the reconstruction algorithms described above was instructive, this project will rely on an established tool specifically designed for tomographical applications - the ASTRA toolbox [21]. This is an open source tool that 'provides a highly efficient and highly flexible open source set of tools for tomographic projection and reconstruction.' [22].

It can be used with Matlab or Python, and provides a wide range of reconstruction algorithms, tools and useful functions. Importantly, it supports a fan-flat geometry similar to the one in Figure 5. This will be used (with some small modifications) to reconstruct images from scans taken with the rotating scanner described in Section 4.

2.3 Machine Learning

2.3.1 ML in tomography

Machine learning in various forms has been used in and around the field of tomography for some time. However, in most cases, it is applied to an auxiliary problem, with the task of image reconstruction being left to algorithms such as those described in the previous section. For example, ML may be used to classify the reconstructed images to accomplish tasks such as detecting glaucomatous abnormality [23] or identifying cancer in breast screening programs [24]. It has also been used to aid sorting and retrieval of medical images [25] or to perform automatic image alignment [26].

In some cases, the end goal of tomography is not the image produced but some information that can be derived from that image. For example, one might take continuous scans of the brain to measure the activity of a specific region over time. In this case, the overall image of brain activity is unimportant compared to the activation data for a specific site. Another example would be measuring drug delivery and re-uptake at certain locations in the brain using dynamic PET. One such study [27] used artificial neural networks to identify patterns in the PET data and construct images showing specific and non-specific binding of a tracer compound, concluding that "artificial neural network analysis is useful for identification of pharmacokinetic patterns in an unbiased way."

2.3.2 Neural Networks

Throughout this project, we will be investigating the use of ML to make certain inferences based on our input data. There are many different supervised learning algorithms that could be applied to most of these problems, but we will be focusing on neural networks. This choice was made after much consideration, and the rationale is explained in Appendix 1. To summarize the reasons: neural networks provided a class of models that could perform well in all tasks considered, able to deal with complexity and nonlinearity and allowing us to focus on relative performance (i.e. how much specific changes affected our ability to make inferences) rather than requiring a long and complicated model selection and tuning process for each new task encountered.

Neural networks, as hinted at by the name, are inspired by the working of our own brains. In fact, the correct term is 'artificial neural network' to distinguish from biological neural networks (brains). We are still far away from being able to model the complexity of a human brain, and so artificial neural networks don't attempt to accurately replicate how a brain works, relying instead on much simpler systems.

ANNs are able to 'learn' over time by being presented with various inputs and the expected outputs. The value of this lies in their ability to capture patterns and associations that might not be well understood, simply by looking at examples. This makes them particularly useful when trying to deal with many variables whose relationships are vague or not well characterised. With enough training data, they can be made to perform extremely well at certain tasks, with the advantage (or disadvantage, perhaps) that we need not understand the inner workings to take advantage of the result. For example, Veng-Pederson and Modi (1992) [28] show how neural networks can be used to model a complex system, mentioning that neural networks "are recognized mainly in terms of their adaptive learning and self-organization features and their nonlinear processing capability and are considered most suitable to deal with complex multivariate systems that are poorly understood and difficult to model by classical inductive, logically structured modeling techniques." This makes them a good fit for tasks such as the object recognition task in Section 5.2, where effects such as reflection and diffraction of light are not easy to model explicitly but could still provide useful information to a neural network.

The neural network architecture used throughout this project is a simple feed-forward architecture called the Multilayer Perceptron (MLP) that is included by default in Scikit-learn [31]. It consists of a set of input neurons (one for each input feature), a number of hidden layers and one or more output neurons. Each neuron takes a weighted sum of its inputs and passes this through an activation function g() (usually tanh(x) or 1 / (1 + exp(-x)), where x is the weighted sum of the inputs) to produce an output.

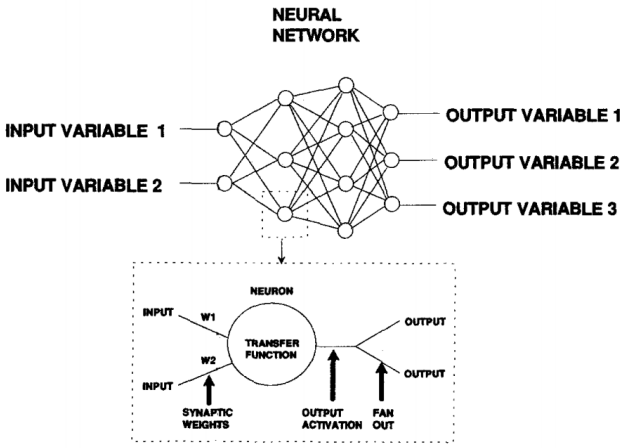


Figure 6: Diagram of a Feed-forward Neural Network, from [28]

In the case of classification, the final (output) layer has a neuron for each class. When functioning correctly, the output neuron corresponding to the class of the input should be close to one, with all other outputs close to 0. When used for regression, there is a single output neuron for each desired output.

The neural networks used in this project (unless otherwise stated) have three hidden layers, each with 20 neurons. The number of input neurons depends on the problem.

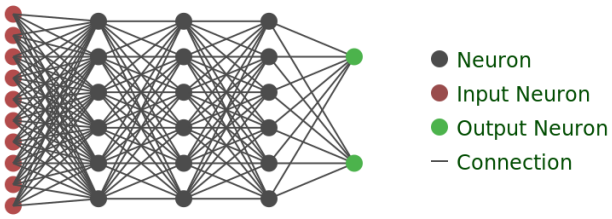


Figure 7: Neural Network Architecture used (simplified)

The network is trained with a set of labelled training data, consisting of an input matrix X and a corresponding matrix y containing the correct classes (in the case of classification) or values (in the case of regression). The network trains using Stochastic Gradient Descent [29] to minimise a loss function. The loss function is computed, the gradient of the loss function with respect to the weights is determined and the error is propagated backwards (backpropagation, [30]) and used to update the weights to reduce the error - i.e. minimise the loss function. This step is repeated many times until a threshold is reached or the improvement from each repetition is negligible. The loss function is composed of some measure of the error (which we want to be minimised), optionally combined with a regularization term to try and prevent the model assigning too much weight to any single input [32].

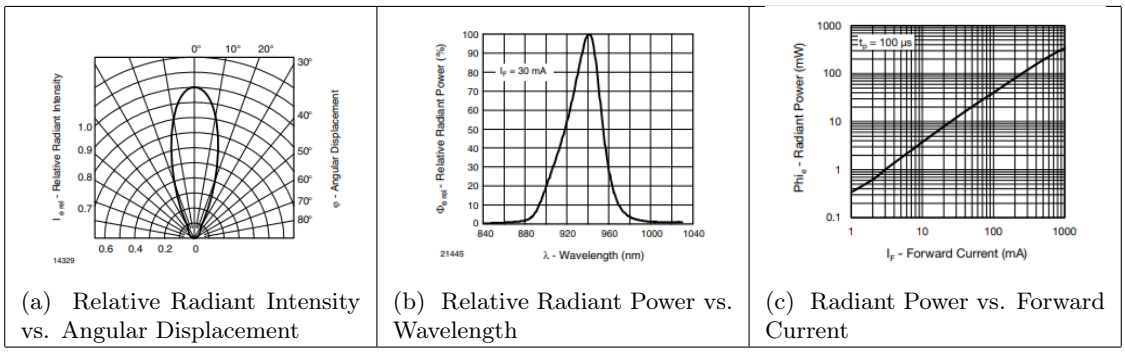
To assess the performance of a model, it must be tested on a data that was not used for training. This helps avoid a problem known as overfitting, in which the model becomes too specific and relies on the noise in the training data to learn the exact details of said training data while failing to model the true underlying trends in the data. This results in a model that performs well on

the training data but poorly on any new data - an unfavourable outcome. Throughout this report, models are assessed with either a dedicated test set or by using a technique called cross-validation, in which a set of labelled data is repeatedly split into training and test sets, a model being trained on each set of training data and then scored on the (unseen) test data and the average score over many runs being used as a metric for how well the model will perform on new test data.

2.4 Components

2.4.1 LEDs

The LEDs used in this project were all IR LEDs emitting radiation with a wavelength of 940nm. A few miscellaneous IR LEDs were used during testing but for most of the project the part used was the TSAL6200 IR LED. Figures 8(a)-(c) are included here from the datasheet [33], and highlight some key characteristics of this device (and IR LEDs in general). Figure 8(a) shows the beam spread - most of the intensity is focused forward. This has large implications when using this device in imaging, as the angle must be taken into account. Scuffing the surface of the LED resulted in a lower intensity beam with wider spread, which was used in some scans with the rotating scanner (Section 4).



(a) Relative Radiant Intensity vs. Angular Displacement

(b) Relative Radiant Power vs. Wavelength

(c) Radiant Power vs. Forward Current

Figure 8: IR LED characteristics (from [33])

Also of note is the radiant power curve in Figure 8(c). When driving these LEDs directly from a microcontroller pin, the maximum current available is 25mA (See Section 2.4.3) and 10mA might be considered nominal - this limits the maximum radiant power we can produce. These LEDs have a maximum forward current of 100mA [33], meaning that to achieve maximum power output they would need to be driven with a transistor or connected directly to a power supply rather than powered via a GPIO pin.

2.4.2 Phototransistors

Phototransistors are light-sensitive transistors that allow current to flow when light shines on them. The amount of current depends on the intensity of the light arriving at the phototransistor, making them a useful tool for measuring light intensity. Figure 9 shows this relationship for the TEFT4300[34] NPN phototransistors used in this project.

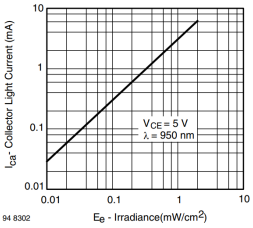


Figure 9: Collector Current vs Irradiance (From [34])

2.4.3 Microcontrollers

This project made use of three different microcontroller boards, namely the Teensy LC, Teensy 3.2 and Teensy 3.6 development boards made by PJRC [35]. Based around Kinetis' Cortex M family of chips [36], they are easy to use and had all the peripherals required for this project. Additionally, the ADCs on these chips are very good, offering an effective resolution of 12-13 bits ([35] and personal experimentation) compared with the 8-9 effective bits of resolution offered by, for example, an STM32f4 board [37].

The table below outlines some relevant specifications

	Teensy LC	Teensy 3.2	Teensy 3.6
Clock Speed (MHz)	48	72	180
Analog Resolution (bits)	12	16	16
IO pins	27	34	58, with 25 analog pins
Max IO current/pin	25mA	25mA	25mA

All three boards also give access to a 12 bit DAC and a host of other features that aren't used in this project.

2.4.4 Motors

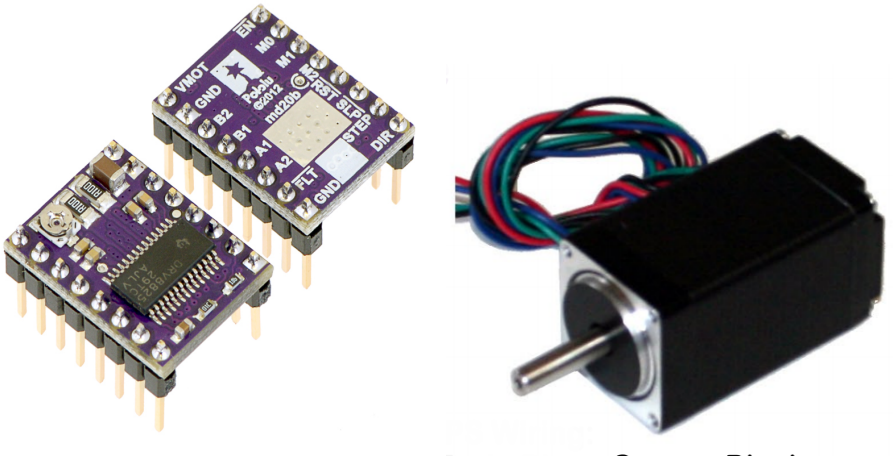


Figure 10: Motor driver boards and the stepper motors used [38, 40]

The rotating platform described in Section 4 must be capable of precisely positioning the two independently rotating components. Although this problem could be solved with simple DC motors and closed loop feedback, the most obvious approach is to use stepper motors. The rotor of a stepper motor can be rotated in small increments by energizing the coils in sequence [39], allowing fine control of the motor's movement. Stepper motor driver chips such as the DRV8825 used in this project are readily available and easy to use, and the control signals can be easily generated with a micro-controller. For this reason, stepper motors were selected for this project.

The motors used are NEMA 11 motors, model number 28BYGH501. They have a step angle of 1.8° / step and a holding torque of 1200 g.cm [40]. The DRV8825 driver can easily supply the 0.67 A/phase [38] required by the motors, and the torque and accuracy are more than sufficient given the low loads and high gear ratio involved in this design.

2.4.5 3D Printer

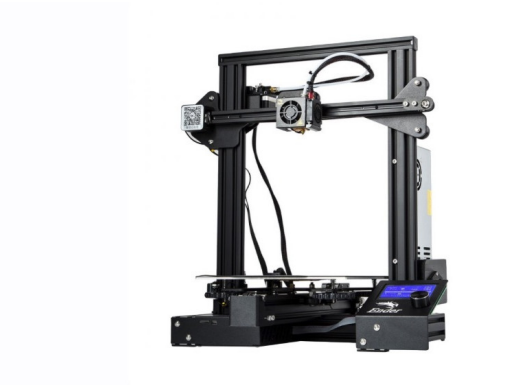


Figure 11: Creality Ender 3 3D Printer [41]

3D printing technology has improved dramatically in the past few years, with prices coming down and performance getting better, especially at the low end. In this project, a 3D printer is used to create physical components and mechanisms, to print objects of known shape (phantoms) to scan and to position objects being scanned in different locations.

The printer used was already in my possession at the start of the semester but was purchased in the knowledge that it would be useful for this project. It is a Creality Ender 3, purchased for R3,600. This is the first Chinese 3D Printer to be certified as open source [42], with all schematics, code and hardware design freely available on Github [43]. Despite the low price, it is an incredibly capable machine, with a rigid frame and high-quality components. It uses Fused Deposition Modeling [44] to create parts out of thermoplastics such as PLA or ABS. The print precision is $<$0.1mm over a 220mm x 220mm area, which is more than enough for this project.

2.5 Software Tools

The following languages, libraries and applications were used over the course of this project:

- The Python programming language[45]. Python is an interpreted language with an emphasis on simplicity, and is one of the most popular languages for scientific computing and data science.
- Jupyter[46]. Jupyter notebooks are interactive documents which can contain code, text and results within a single document. Jupyter supports many different languages, but was used exclusively with Python 3 for this project. The GitHub repository submitted along with this report contains summary notebooks with the code used for the experiments, interleaved with explanation and results.
- The Pandas library [47]. This is a Python library designed to handle tabular data efficiently. It is used for storing results, performing analyses and handling file IO.
- Scikit-learn [31]. Scikit-Learn is used for machine learning in Python. It provides a wide variety of machine learning algorithms and tools, especially suitable for initial testing and model selection.
- The ASTRA toolbox. As mentioned in Section 2.2.4, ASTRA provides a set of tools for tomography.
- CTSim [48] simulates the collection of X-ray data, and was used to check the correct functioning of the simulation before ASTRA was substituted in its place.

3 Simulation

3.1 Defining geometries

A way to simulate different arrangements of emitters and detectors was needed. The goal was to move away from a few well-defined geometries and allow the simulation of arbitrary arrangements. To make this possible, some code was developed that allows the emitter and detector locations (as well as optional direction and spread characteristics) to be specified before beginning the simulation. The positions are encoded as x and y values between -1 and 1, or as polar coordinates. These can then be re-mapped to a different coordinate space (for example mapping to coordinates in mm to match an experiment) or converted to pixel coordinates for the path tracing as described in the next section. There are also utility functions to rotate all emitters and detectors through a specified angle, generate test images and so on.

As an example, consider an attempt at simulating a CT scan with a fan-beam geometry. We have a single emitter (representing the X-ray source) and a curved ring of detectors. We can specify the arrangement as follows:

e = {'r':1, 'a':0}
d[i] = {'r':0.5, 'a':$\pi/2 + \pi * (i + 0.5)/N$ }, where i runs from 0 to N-1

This results in the arrangement shown in Figure 12 (with N=8).

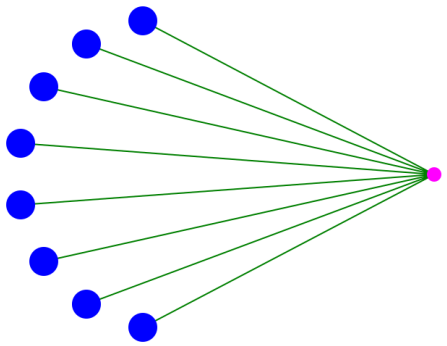


Figure 12: A fan-beam arrangement

The simulation code sums the input image along the paths from the emitter to the detector using the get_paths() function then rotates the whole arrangement slightly (rotate_des()), repeating the process to obtain the full set of projections.

3.2 Simulation Method

The simulation is simplistic in that it ignores optical effects such as diffraction. To estimate a given reading, we consider a single emitter-detector pair at a time. A greyscale input image encodes the transmissivity of the object being scanned. The code traces a path through this image, summing the pixel values for a set number of points along the line. This is equivalent to the line integral described in the Section 2.2.1 as part of the radon transform.

When comparing the output of the simulator to real-world data collected experimentally, it is necessary to transform the simulation output to match the observed readings. The total attenuation (p) is related to the received intensity (I) by the formula $p = ln(I/I_0)$, and is also equal to the line integral of the attenuation coefficient along the path travelled from emitter to detector. The simulated readings are transformed to account for this, and then multiplied by a scaling matrix that maps the transformed simulated readings to the observed readings (which accounts for additional effects such as beam spread and PT angular sensitivity). This scaling matrix is found by comparing a base set of simulated readings with the measured equivalents. With these steps completed, we

can use the simulation to predict the observed readings for a given configuration. This is useful when attempting to generate additional training data - see Section 5.3 for details.

3.3 Reconstruction

If a standard geometry is simulated, the resultant sinogram can be used to reconstruct the input image using standard tools such as the ASTRA toolbox. In other cases, some transformation or interpolation is required to get the data in a format that these tools can work with. For example, see the Section 4.5, where the output of the rotating scanner is adjusted to resemble a 'fan-flat' geometry so that ASTRA can reconstruct the images.

3.4 Comparison with existing tools

The purpose of the simulator was to allow the investigation of arbitrary arrangements that are not supported by other tools. However, it can also simulate the standard arrangements that are normally used, and this provides a convenient way to make sure it is working as expected. The outputs of two different, existing tools (ASTRA and CTSim) were compared to the output of the simulator. In all cases, the output of the simulator was similar to the output from the other tools. Figure 13 shows an input image and two sinograms, one generated by the simulation code developed for this project and one created using the ASTRA toolbox, both simulating the same fan-beam geometry.

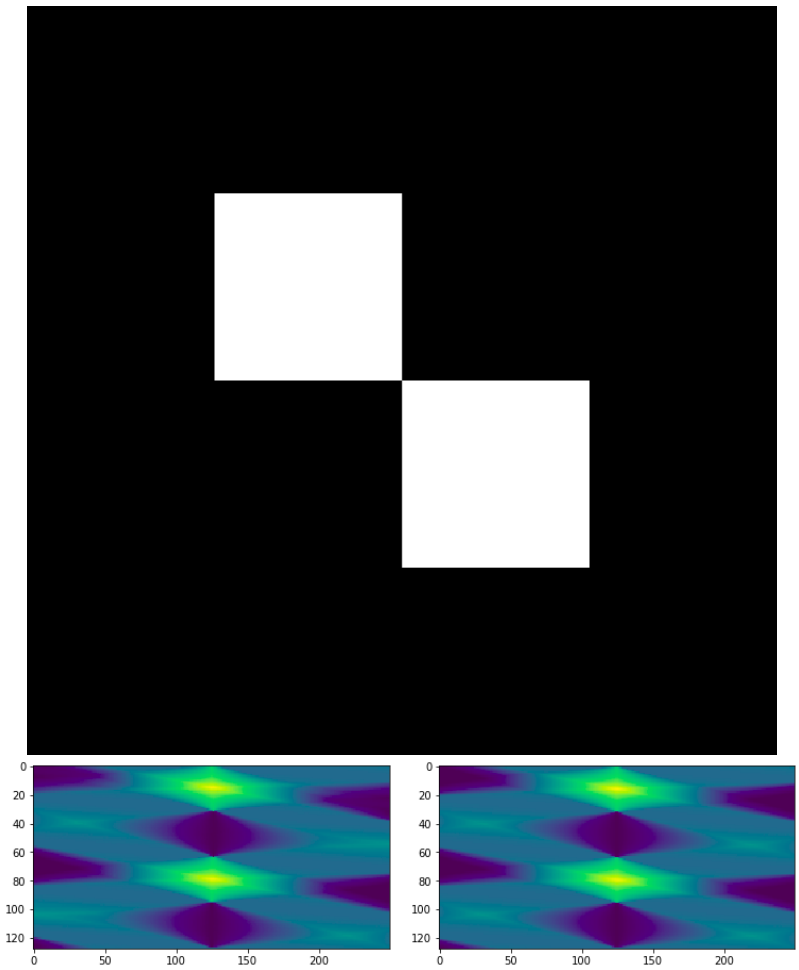


Figure 13: Comparing the output of the simulator (left) with the output from ASTRA (right) for an input image (top).

4 Rotating Scanner

4.1 Purpose

To allow rapid experimentation with different scanning arrangements, a rotating platform was developed that allows precise positioning of a single emitter-detector pair relative to a scanned object. By taking readings for a set of emitter and detector positions, complex arrangements can be simulated.

The platform is predominantly composed of 3D printed parts, moved by two stepper motors. One stepper motor controls the rotation of the object being scanned, while the other controls the rotation of the outer ring where one or more phototransistors (PTs) are mounted. An IR LED, laser or another light source can be placed in a fixed position at the desired distance from the rotating platform.

4.2 Physical Design

Onshape [ref] was used for the CAD. The central platform is driven by spur gears with an 80:25 reduction. The outer ring has a 160:1 reduction. The stepper motors used are 200 steps per revolution, meaning that the central platform can be placed in one of 640 positions and the outer ring can be positioned in one of 1280 positions. The stepper motor drivers used support micro-stepping up to 32x, potentially allowing even greater positional accuracy.

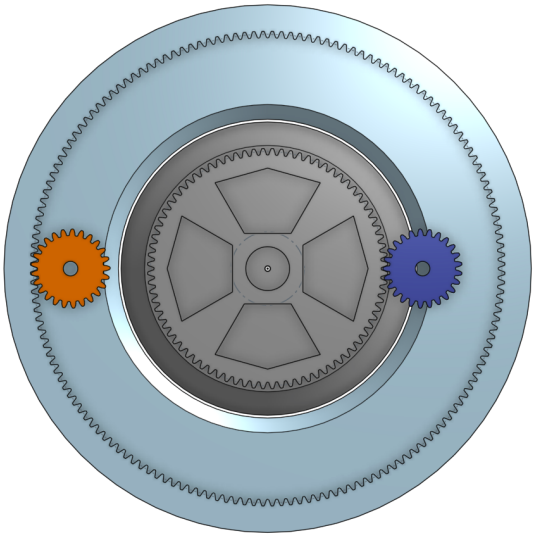


Figure 14: View of the platform from below

All components are mounted to a single base plate, with the stepper motors mounted underneath.

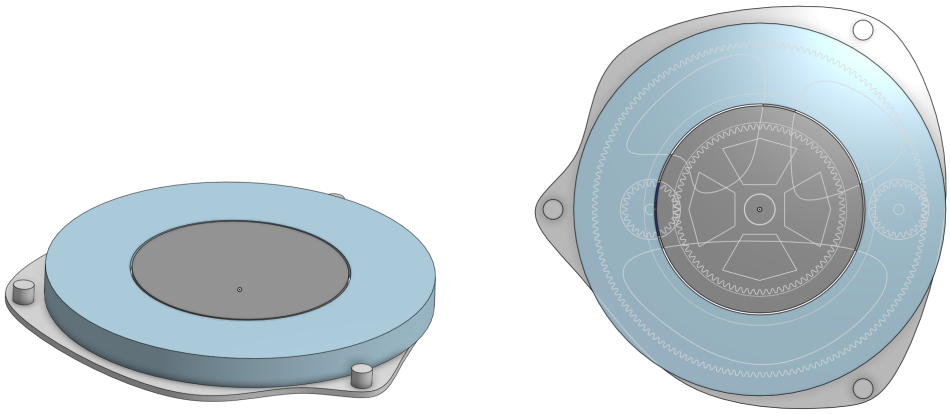


Figure 15: CAD Models

All parts were printed on a Creality Ender 3 3D printer using PLA plastic. Appendix 2 lists the CAD models of each component along with details such as print time and part weight. A layer height of 0.2mm was used throughout. The assembly fits together largely without fasteners, but some Loctite adhesive was used to secure the stepper motor mounts in place and hold the small gears on the stepper motor shafts.

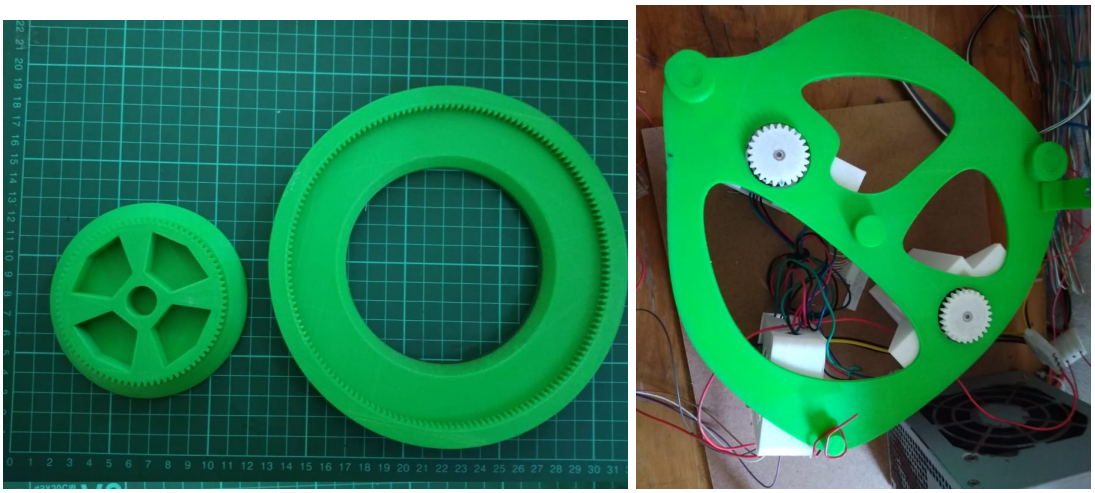


Figure 16: 3D printed parts (left) and base plate with motors mounted (right)

The top surfaces were spray-painted black as the green PLA used reflects IR light. An IR LED can be mounted to the base plate with a 3D printed clip or set further away on a stand. The PT is held in a bracket fixed to somewhere on the outer ring.

Figure 17: Completed scanner scanning a 3D printed shape

4.3 Stepper Motors

Initial prototyping was done with stepper motors salvaged from old laser printers. However, for the final design, the decision was made to go with known parts to aid replication and remove the uncertainty around the motor specifications. The 28bygh501 motor was used.

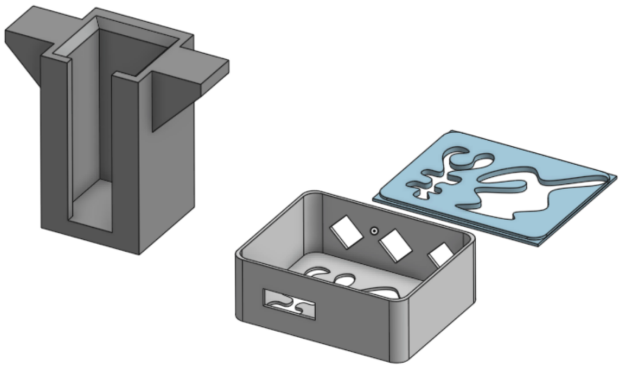


Figure 18: Stepper motor mount and electronics box

The motors are driven using a pair DRV8825 stepper motor drivers, mounted in a dedicated enclosure. These are capable of delivering up to 1.5 amps per phase - more than enough for the selected stepper motors. The current limit was set to 0.6A using the onboard potentiometer. A salvaged ATX power supply was used to provide power. Because the drivers were set to deliver only a small fraction of their maximum power, a high-pitched whine was observed in certain situations. This was solved by soldering a wire from ground to pin 19 of the chip to set it to 'slow decay' mode.

4.4 Control

A teensy uc [name] is used to control the stepper motors and take readings. It accepts serial commands from a PC to move the motors, take a reading or toggle an output. To take a scan, the outer ring is used to position the PT, take a reading, re-position, take another reading and so

on. Then the central plate is rotated a small amount (analogous to taking a scan from a different angle - a new 'view' and the process of moving the outer ring while taking readings is repeated.

Control is done from python code within a Jupyter notebook. The data is stored in a Pandas DataFrame for analysis and also saved as a CSV file for later use. The number of detector positions and the number of views can be controlled from within the code. Most scans were taken with 640 detector positions and 160 views - scans with fewer views or detectors could then be simulated by simply taking a subset of the data corresponding to fewer detector positions or views or interpolating between the two closest detector positions if required.

Both the Python code used for data acquisition on the host PC and the Arduino (C++) code that runs on the microcontroller are included in the GitHub repository accompanying this report.

4.5 Image Reconstruction

As mentioned previously, the ASTRA toolbox can account for fan beam arrangements, saving us from having to implement the FBP algorithm described in Section 2.2.3 from scratch. However, ASTRA assumes a geometry like that shown on the left in Figure 19 when using the 'fan-flat' geometry. The rotating scanner has a geometry like that shown on the right in Figure 19. Instead of detectors spaced evenly along a line, we have detectors spaced evenly along the circumference of a circle. To account for this, we first project the readings back onto a line, and then we use interpolation to estimate what readings an evenly spaced array of detectors on the same line would show. This interpolation step sacrifices some accuracy but allows us to use the built in reconstruction algorithms made available by the ASTRA toolbox. The 'Image Reconstruction' notebook in the GitHub repository contains a code listing showing the steps to go from a set of readings taken with the scanner to a reconstructed image.

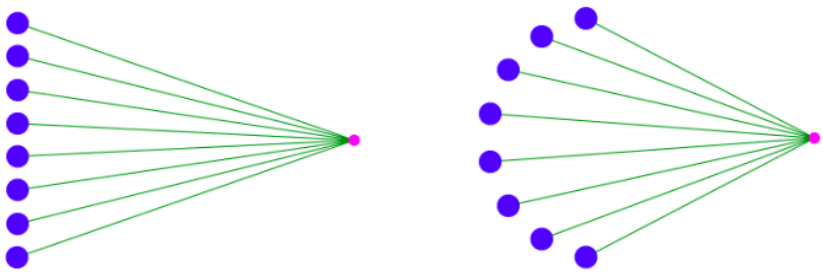


Figure 19: ASTRA's 'fan-flat' geometry (left) and the rotating scanner geometry (right)

An important step when performing the image reconstruction is ensuring the geometry is defined correctly. This includes specifying the relative distances from the center of the rotating platform to the emitter and the detector. In some cases, the detector is not positioned directly opposite the emitter at the start of a scan, which can introduce geometric distortions. To compensate, there is also an option in the code to shift the sinogram slightly during the interpolation step.

4.6 Results

The first successful scan was of a pen and a screwdriver placed vertically on the scanner, taken with 160 detector positions (ndetectors = 160) from 32 different angles (nviews=32). Figure 20 shows a photo of the scan in operation, the sinogram and the reconstructed image which was obtained by following the steps outlined in the previous section using the 'SIRT' algorithm and blurring the result with a 3x3 pixel gaussian filter. The two objects can clearly be seen.

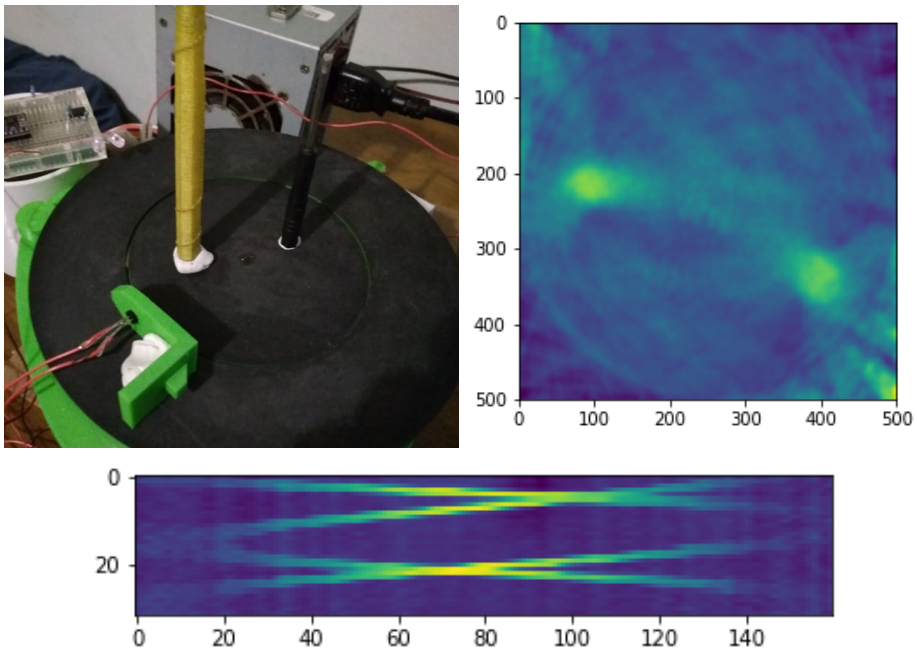


Figure 20: First scan in progress (top left), readings (bottom) and reconstruction (top right)

To simulate a scan with 16 views and 8 detectors, a subset of the measured readings is used to create an equivalent sinogram. Reconstructing an image from this data we get the results shown in figure 21, which also shows the difference in output between the filtered backprojection algorithm (FBP) and the iterative approach (SIRT).

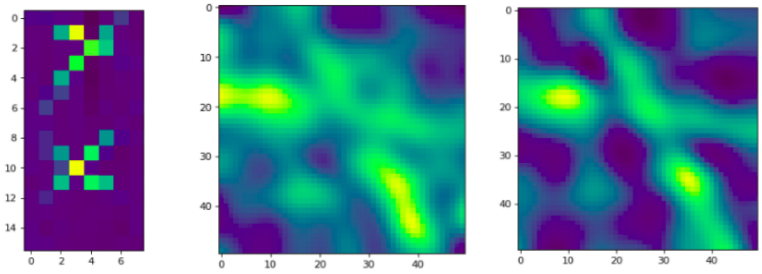
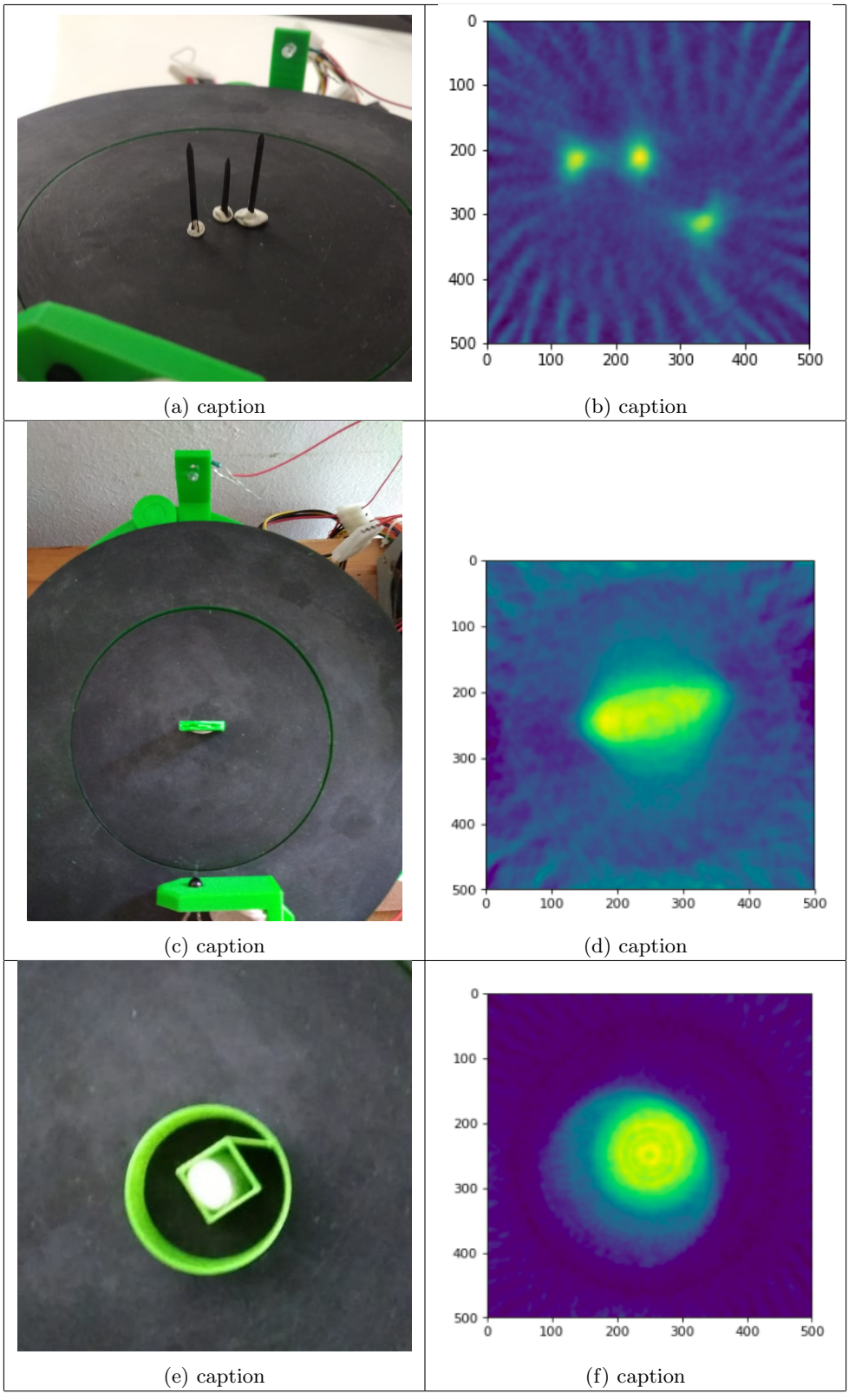


Figure 21: Reconstructing from a subset of the readings. nviews=16, ndetectors=8

Many scans were taken over the course of the project. All the data was stored for later analysis, and may prove useful for further investigation of reconstruction techniques. A small subset of these scans are included here in Figure 22, including those which show the capabilities and limitations of the system. The nails imaged in (a) are 1.6mm in diameter, with the two closest nails 7mm apart. This shows the impressive resolution achievable with this scanner. (e) and (f) illustrate the ability of IR light to penetrate thin walls of plastic - the internal structure can be seen quite clearly.



(a) caption (b) caption

(c) caption (d) caption

(e) caption (f) caption

Figure 22: Some example scans

5 A fixed arrangement with 8 LEDs and 8 PTs

The rotating platform gave some useful information, but the next step was to begin working on the question that had initially sparked my interest - what could be done given a limited number of sensors and emitters in a fixed arrangement. This section will describe the first set of experiments with an arrangement of 8 LEDs and 8 PTs in a ring as shown in Figure 23, examining the use of machine learning to interpret the readings and demonstrating the ability to locate an object within the ring and to distinguish between different objects.

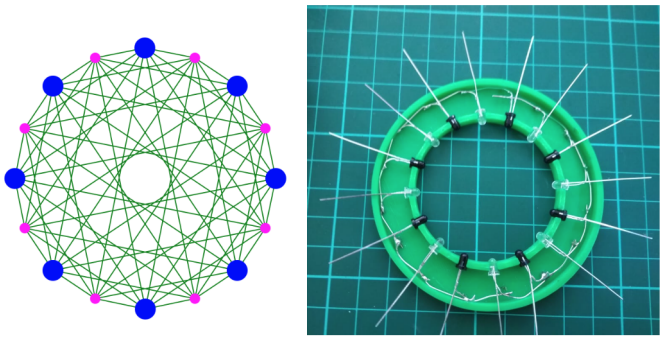


Figure 23: Sensor and Emitter placement in simulation (left) and the physical device (right)

5.1 Simulation

A simple classification task was created by dividing up the scanned area into four quadrants and attempting to predict which of these four areas contained an object. This can be re-created in simulation. A shape is drawn randomly in one of the four quadrants, and the simulator determines what readings would be expected for each emitter-detector pair (64 readings in total). The readings are used as model training inputs, with the quadrant number as the desired output.

Initial performance was not as good as expected, with only 90% accuracy once some simulated noise was added in the simulator code. Investigating, it was found that most errors were due to samples right on the boundary between quadrants, with a lot of overlap. Repeating the simulation but restricting the location of the simulated objects to areas more obviously within one quadrant or another resulted in near perfect accuracy. Figure 24 shows the difference between training samples placed randomly (high overlap) and those placed mostly within a given quadrant (low overlap).

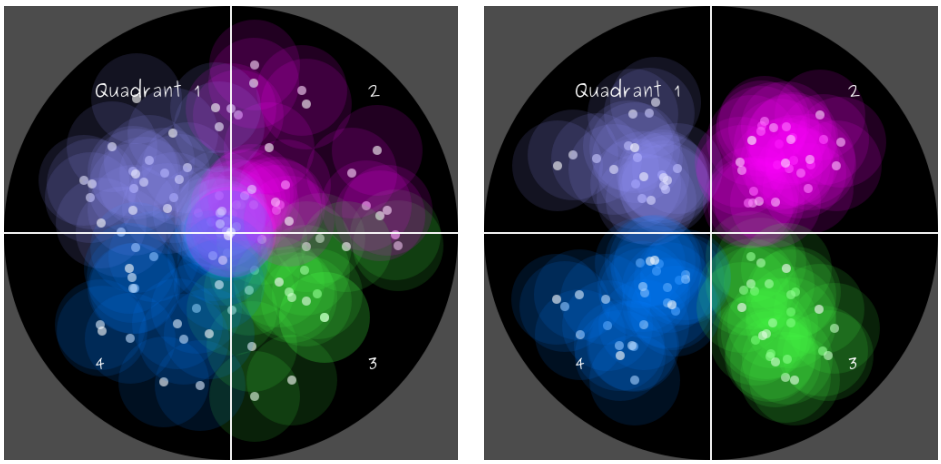


Figure 24: 100 random locations showing random placement (left) vs low overlap (right)

When overlap was reduced, the system was more accurate and required less training samples to reach its maximum accuracy. The graph in Figure 25 shows accuracy vs the number of training samples used for different amounts of overlap:

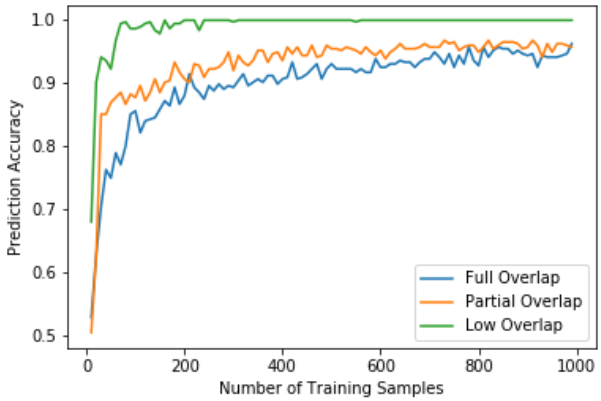


Figure 25: Model score vs number of samples used for training, for varying levels of overlap

As explained in the next section, we expect the experimental results to be similar to the partial overlap case. When manually placing the test objects in the ring, boundary zones were naturally avoided to ensure that the manual classification was correct.

5.2 Experimental Setup

5.2.1 Physical Construction

The sensor and emitter elements are held in place by a 3D printed template. They are equally spaced around a ring 50mm in diameter. A Teensy LC microcontroller board is used to drive the LEDs and take readings. Figure 26 shows the device under construction before the wiring was completed and the cover installed.

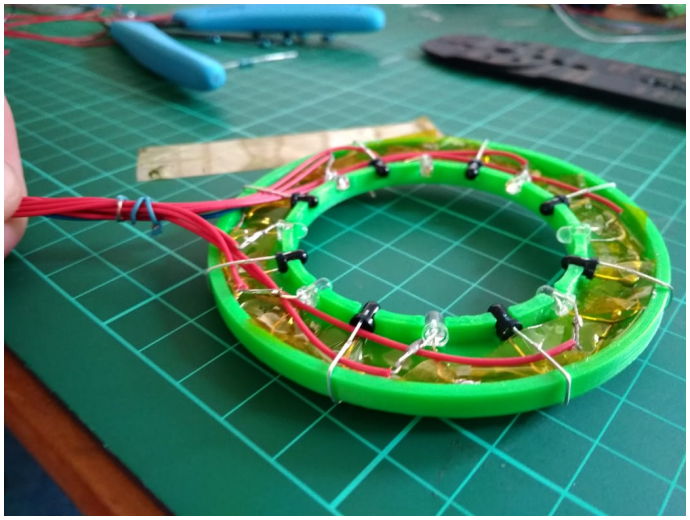


Figure 26: Device under construction.

Once the wiring was complete, everything was tested to ensure functionality before being secured in place with hot glue. A 3D-printed cap completed the enclosure.

5.2.2 Electronics

Figure X shows the method by which the amount of light transmitted is measured. All the LEDs and PTs share a common ground. Each LED is directly driven by an IO pin of the Teensy LC via a 330-ohm resistor (R1 in 27). The phototransistors are connected via a 220k resistor (R2 in 27) to +V, and the voltage across them is measured by the teensy's built-in ADC. Serial commands from the host PC can be used to turn the LEDs on and off and read the various analogue pins.

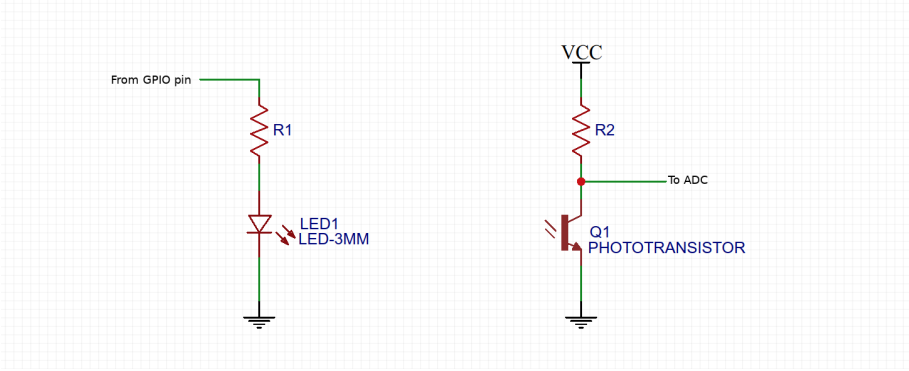


Figure 27: Circuit Diagram - Reading transmitted light

5.2.3 Data Collection

A dataset was created by manually placing one of four objects randomly within the scanned area and recording the sensor readings as each LED was sequentially illuminated. There are 100 sets of readings per quadrant per object, for a total of 2000 sets of readings (1600 with objects, 400 with the ring empty). Light levels were varied during the data collection and there was no post-processing of the collected data. There are 64 columns in the dataset with the raw readings, 8 with the base readings (no LEDs lit), one showing which object is present and one listing which quadrant the device was in, for a total of 74 columns. This data was saved as a CSV file.

During the data collection process, one of the four objects used (a black pen, silver pen, craft knife or finger) was inserted into the ring, and the object, as well as the location (i.e. which quadrant it was in), was recorded along with a set of readings. By moving the object around within a specified quadrant while readings were taken, the dataset could be built up fairly quickly.

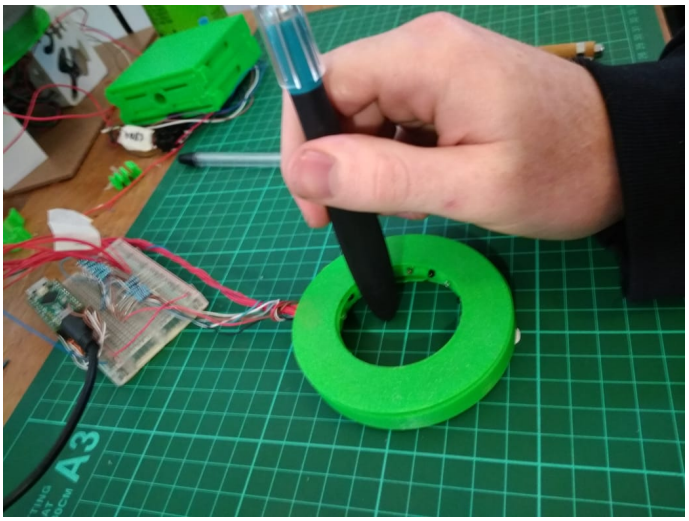


Figure 28: Taking Readings. Object 2, Quadrant 1

5.2.4 Machine Learning

The data was split into training and test sets, with the latter being used to assess the various models' performance. The training dataset was used to train a neural network built around the simple Multilayer Perceptron (MLP) Classifier architecture. The neural network was initiated with three hidden layers, each with 20 neurons (as was the case in simulation) and trained over 1000 iterations or until convergence. Using 75% of the data for training and the remaining 25% to test, the model was able to correctly predict the quadrant where the object was placed over 98% of the time. Figure 29 below shows how the model score improves as more training samples were used, compared to the results from simulation. We can see that, as expected, the accuracy based on experimental data is similar to that achieved from simulation with some overlap.

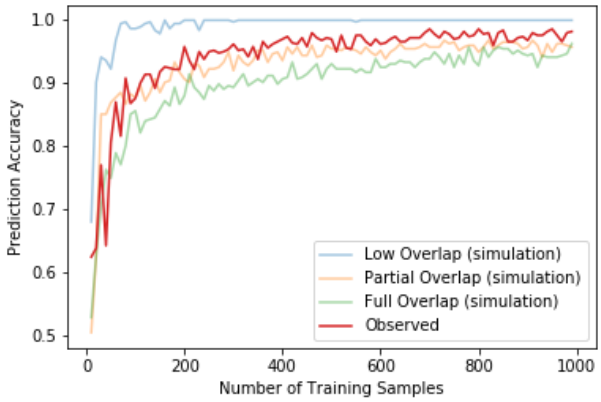


Figure 29: Predicting location area: performance with more training samples

More interesting is the second test - differentiating between the objects. Here, more subtle effects come into play that are not included in the simulation. Reflection, transmission and shape should, in theory, mean that each object interacts with the light slightly differently, making it possible to distinguish between them. A new model was created and trained to predict the object class. With 75% of the data used for training, the model makes correct predictions 85% of the time! Simplifying the problem down to just distinguishing between a finger or the black pen (a binary classification problem), the performance is even higher, at 97% accuracy. Again, more training data improves the predictions. This is shown in Figure 30.

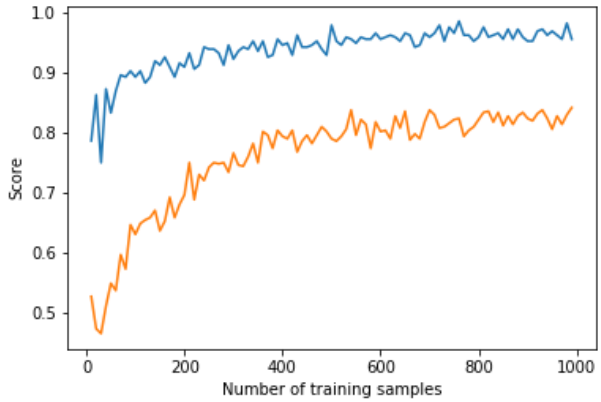


Figure 30: Object classification: performance with more training samples for multi-class classification (orange) and binary classification (blue)

5.3 Position inference and transfer learning

The previous sections show how ML can be used to solve simple problems based on the sensor readings. The next step is to try a slightly harder problem - predicting the exact location of an object within the ring, as x and y coordinates. Generating training data is time-consuming, so this section will also explore the possibility of using a small number of training samples in conjunction with simulation to train the models.

Beginning with pure simulation, we generate test images with a circle in a random location. The simulated readings are stored as the inputs and the position (x and y) as the desired outputs. With 500+ training samples, the neural network is able to correctly predict the location with an RMSE of $<$0.1r, where r is the radius of the ring. Figure 31 shows the distribution of prediction errors. 85% of the errors are within 0.1r, and 98% are within 0.2r. In the next section, we will show how this compares to a ring with 16 LEDs and 16 PTs.

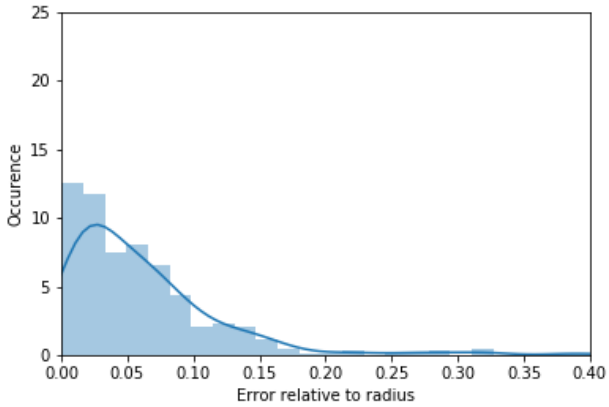


Figure 31: Distribution of Errors - Ring of 8 position inference (simulation)

Simulation reveals that many training samples are required for optimum performance, with more than 1000 samples required to reach optimum accuracy.

To gather a large number of training samples experimentally, a 3D printer was used to precisely position objects within the ring. For each (known) position, the readings and position could be recorded. 500 random positions were generated and converted to GCODE [49] to send to the printer. A 3D printed adapter holds a pen or other test object to the end effector of the printer. The readings taken at the different positions were stored in a CSV file for later analysis.

Keeping aside 100 sets of readings as a test set, we are left with 400 training samples. A model trained with these 400 training samples was able to predict the locations in the test set with a root mean squared error (RMSE) of 0.7cm. Adding 1000 simulated sets of readings to the 400 training samples and training a new model with the extra data resulted in a lower RMSE - 0.6cm. Repeating with varying amounts of training data from the experiment, we see that the model augmented with simulated data consistently outperforms one trained with only experimental data (Figure 32).

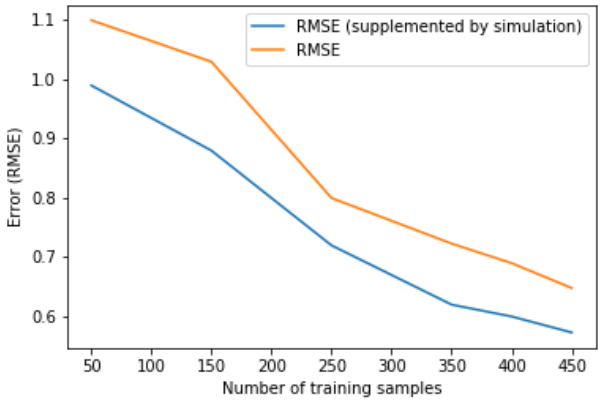


Figure 32: Decreasing Error (RMSE) with more training data, with and without augmentation from simulation.

This demonstrates the possibility of using simulation to drastically reduce the number of training samples required to achieve a given accuracy in a problem like this. Improving the simulator to more closely match the real world would likely improve performance even further.

6 A More complex fixed arrangement with 16LEDs and 16 PTs

6.1 Simulation

As with the smaller ring, simulation will be used to get an idea of how well the system is expected to perform. This section will also investigate the effect of considering only a subset of the total available paths (as illustrated in Figure 33) as well as image reconstruction.

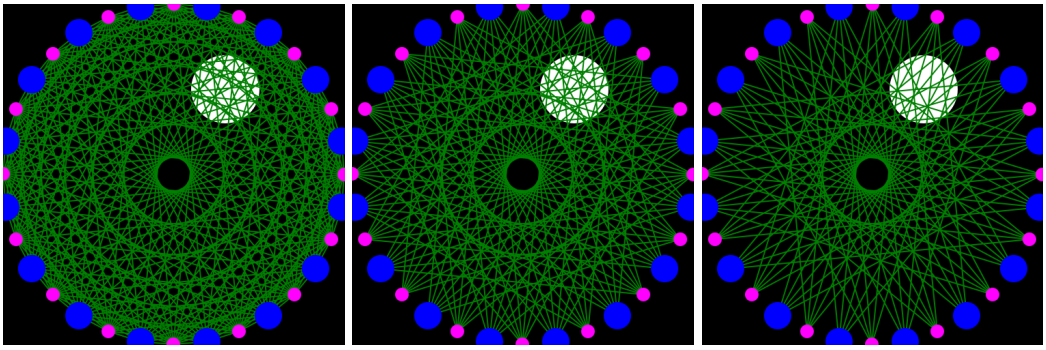


Figure 33: Considering all possible paths (left) vs sparse paths (middle, right)

As expected, doubling the number of emitters and detectors compared to the previous arrangement resulted in better performance on all tests considered. Additionally, considering only a subset of paths resulted in improved performance in some cases. This can be explained by overfitting of the neural network when the number of inputs is large in relation to the number of training samples - for these tests, 16*16=256 inputs, with only 1000 training samples. Focusing on only the 80 most important readings helps eliminate the effect of noise and prevent overfitting.

For the classification task described in Section 5.1, we observe the following scores:
Ring with 8 sensors and 8 emitters: 96.2%
Ring with 16 sensors and 16 emitters: 97.3%
Ring with 16 sensors and 16 emitters, considering only sparse paths: 99.0%

For position inference, as described in Section 5.3, the mean absolute error for the ring of 8 in simulation was 0.06r, where r is the radius of the ring. Doubling the number of sensors and emitters resulted in a reduced error of 0.05r - a small but significant improvement. With the sparse inputs, the error was still 0.05r - the other inputs were not particularly useful for position inference and could safely be discarded.

6.2 Construction

Construction followed a very similar pattern to the previous arrangement. A 3D printed frame holds the LEDs and PTs in place. A teensy 3.6 is used as it has easy access to enough digital pins and analog pins. Figures 34 shows the finished ring:

Figure 34: The finished ring of 16

The ground pins of the LEDs and PTs are soldered in a ring. The current limiting resistors are soldered to the LEDs and held in place with polyamide tape, and the remainder of the circuit is constructed on some proto-board. The Teensy 3.6 is inserted into some female headers and is used to read control the LEDs and take the readings.

6.3 Performance and Problems

The results from simulation indicate that this arrangement should perform far better than the previous one, and could even produce some reasonable images as output. However, the device did not perform as well as expected. This is likely due to a combination of several factors:

- Under-powered LEDs. The LEDs in this design are driven directly by the IO pins of the microcontroller, with an inline resistor limiting the current to <10mA. These LEDs can take 100mA, which would result in a roughly tenfold increase in radiated power. A better solution would have been to drive the LEDs with transistors so that the proper amount of current could be delivered.

- Larger radius. This ring is far larger than that used for the arrangement in Section 5. Because the light emitted by the LEDs spreads out in a beam, the irradiance drops off with distance according to the inverse-squared law. This is one reason why the smaller ring (which was electrically similar) worked so much better.

- When it was discovered that the signal was small, larger resistors (2M) were substituted in an attempt to increase the gain. Although this did result in a slightly larger observed signal, it also caused some other issues.

The readings are noisy and show almost no change when the LEDs are toggled on and off. Even with the larger resistors inserted, the readings appeared meaningless at first glance. However, it was still possible to make some inferences based on the readings, especially with heavy averaging. In a simple classification task equivalent to the quadrant based one described in Section 5.2, it was possible to correctly predict where an object had been inserted just under 90% of the time (depending on the amount of training data used). This is far less than the expected 99%, but shows that even with a badly designed system it is possible to extract some useful information.

Improving this device is listed as one avenue for future work (Section 10). Experiments with the rotating platform in Section 4 suggest that image reconstruction for this arrangement should be possible. It takes 3.3ms to acquire a full set of 256 readings with 32x hardware averaging, meaning that this device would be capable of imaging the space within the ring at 300 frames a second, opening up some interesting potential applications.

7 Use Case Examples

7.1 Gel dosimetry

As mentioned in Section 2.1.1, one application where optical CT is used today is gel dosimetry. A polymer gel turns opaque when irradiated, and imaging this allows approximate dosage in different locations to be inferred. The scanner used for this project is not calibrated and would need significant modifications before it could be used as a practical tool in this application. Nonetheless, it seems worth exploring.

Figure 35: Cube of jelly with milk used to create an opaque region

Unfortunately, irradiated gel samples could not be sourced in time. Jelly (flavoured gelatin) was used in place of the polymer gel. Blackcurrant flavour (like many foods with artificial colouring) is almost completely transparent to IR light but does have a refractive index different to that of air, which could potentially distort any images we attempt to take. To simulate the opaque regions which would be found in irradiated gel samples, milk was placed into hand-shaped incisions in the jelly and left to diffuse slightly. These milk-filled voids are what we will attempt to image.

For the first test, a cube of jelly was imaged, first with no opaque regions and then with milk dropped into a slot made with a key (Figure 35). Reconstructing both images, it is hard to see much detail. Some artefacts are present in the reconstruction, likely due to irregularities on the outside of the jelly reflecting light in unanticipated ways (Figure 36, middle row). However, we can get a much clearer picture by using the first scan as a baseline and reconstructing a new image based on the difference between the baseline and the second scan. This results in the bottom row of Figure 36, where the modified region can clearly be seen.

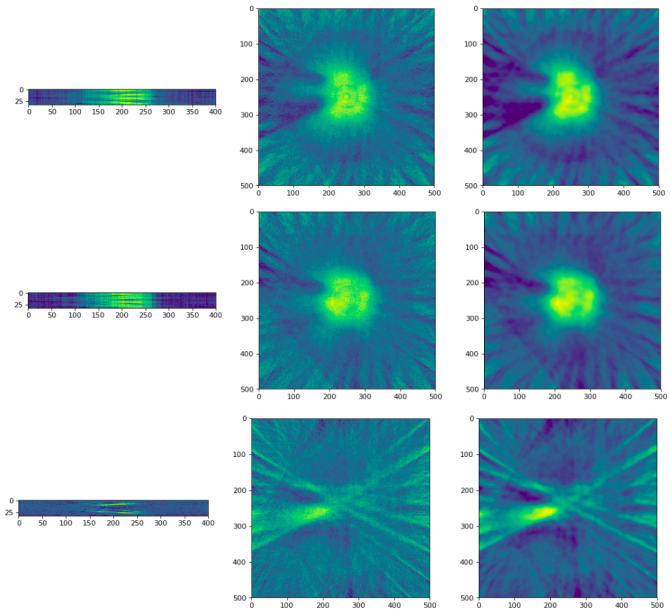


Figure 36: Sinogram, reconstruction and smoothed reconstruction for scan 1 with no feature (top), added milk feature (middle) and enhanced difference (bottom)

This approach seems promising. A second scan with a square cutout shows even better results (Figure 37). It's possible that with further refinement, a better illumination source and more detailed scans (all of these used only 32 views) this approach could perform very well, although it seems unlikely to prove useful beyond serving as an educational tool.

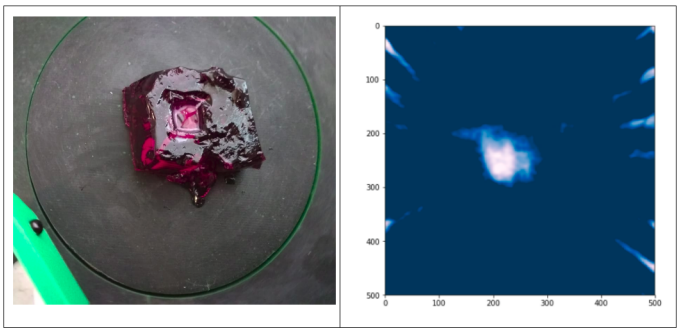


Figure 37: Scan 2 - Cube within a cube. Object (left) and reconstruction of the opaque region (right)

7.2 Input devices

Two simple examples are presented here to show the possibility of using some of the fixed arrangements described earlier as input devices for computing.

For the first example, the ring of sensors from Section 5 was used as a game controller. A finger placed within the ring is used to control an on-screen player, who must dodge enemies. Figures 38 and 39 show a screen-shot of the game being played and a student deeply focused on beating the developer's high-score.

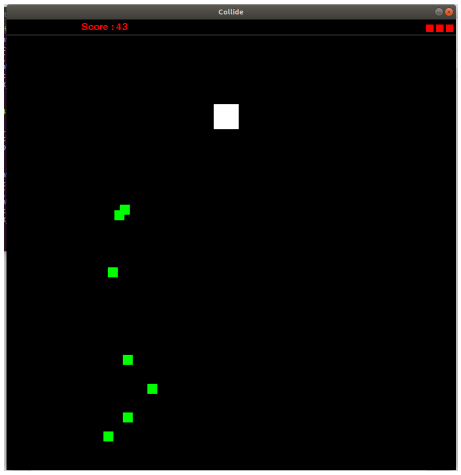


Figure 38: Screen-shot of the proof-of-concept game being played

Figure 39: Using the device as a game controller: early testing by a keen player

The total time needed to acquire and transmit a full set of readings and then to make a prediction based on those readings was 55 ms, slowing down the game a small amount. It is worth noting that with the Teensy 3.6 and some more efficient code it was possible to take a set of 256 readings (with 32x hardware averaging) in 3.3 ms, so this would not be an issue.

A second type of input device was constructed accidentally when examining other potential use-cases. A smaller version of the ring of 8 from Section 5 had been constructed to test the possibility of imaging or assessing the inside of plastic pipes. A 3D printed test piece was inserted into the ring (as shown in Figure 40) and to test how well the IR light was penetrating the plastic shell, a simple classification task was devised, attempting to train a model to infer the rotation of the 3D printed part. This was successful, and by switching to a regression model it was possible

to roughly predict the angle at which the insert was positioned. This results in a low-resolution encoder, and the rotation of the 3D printed insert can be mapped to volume control or used to simulate a mouse wheel scrolling with a few lines of code. This is not the most practical input device, but it does demonstrate the concept.

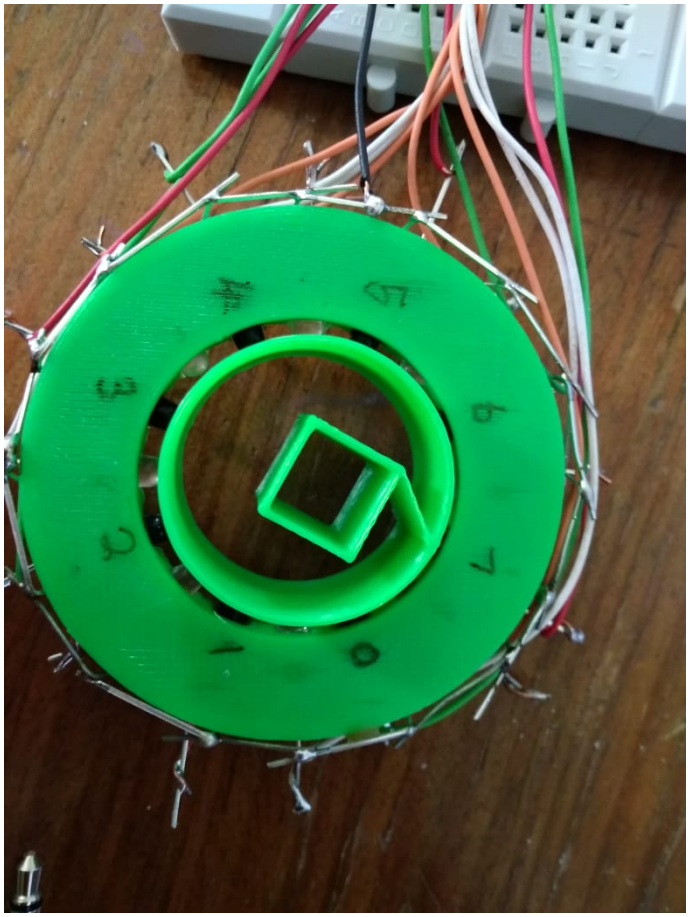


Figure 40: Creating a rotary encoder

8 Discussion

This project explored several different questions about the use of IR light for tomography and the possibility of creating useful systems with relatively few sensor elements. Some key lessons:

- Having a way to simulate different configurations was incredibly helpful for planning experiments, testing out ideas and predicting how well a given configuration will perform. It also meant that progress could be made rapidly. While the fixed arrangements were under construction, reconstruction algorithms and machine learning pipelines could be developed based on results from simulation, meaning that the experiments could begin as soon as the hardware was ready.
- Using light for tomography introduces errors due to optical effects such as refraction, but these are generally small enough that approaches that work for X-Ray CT will still produce adequate results. Techniques such as performing a baseline scan first to remove the signals that are unrelated to the feature of interest can improve the results further, as demonstrated in the gel dosimetry example.
- The rotating scanner was surprisingly capable, and the database of scans taken at high resolution was useful for quickly testing reconstruction algorithms and smaller configurations.
- It was possible to infer some useful information from the readings generated by the small fixed arrangements. Even with 8 LEDs and 8 phototransistors, the position of an object

within the ring could be established with a mean absolute error of less than 10% of the radius (0.23cm). These sparse sensor arrays could have some useful applications.

- It was possible to improve model performance using simulated data to augment a small training set. This is an important outcome - with refinements it could be possible to create devices that require very little calibration and data collection to function correctly.

9 Conclusions

The goal of this project was to investigate IR Tomography, with a focus on small arrays of sensors and emitters. Simulation and experimentation with the scanner provided a framework for planning and testing different configurations. The fixed arrangements performed well, showing the value of machine learning as a tool to interpret the readings obtained and demonstrating the value of testing ideas in simulation first. All of the initial objectives were met, and I consider this project a success. Additionally, the position inference experiments in Section 5.3 demonstrated how simulation could be used to improve model performance. This is an exciting outcome, and one I hope to investigate further in the future.

10 Recommendations for Future Work

The exploratory nature of this project has left many avenues for further investigation. Here, I list some future improvements or new research directions that I believe could prove fruitful.

- Improving the ring of 16 from Section 6 by rebuilding it or adding some LED driver circuitry to the existing board.
- Improving the measurement of transmitted light. By transmitting light modulated at high frequency and then measuring the ac component of the signal at the receiving end, the influence of background light could be reduced and a more accurate reading obtained.
- Experimenting with non-uniform arrangements. This was covered briefly in simulation, but there is rich potential to take this further. For example, consider a case where certain angles are blocked. An evolutionary algorithm could be used in conjunction with the simulator to find a high-performing arrangement of sensors and emitters given the number of PTs and LEDs available.
- Improving the simulator to better capture the behavior of light. One approach I considered was using ray-tracing software to simulate the passage of light through different mediums.
- Improving the machine learning models used. The ones chosen for use in this project were arrived at by exploration and some quick experiments. I know from past experience that some significant gains might be achieved with more work optimising inputs, feature engineering and hyper-parameter tuning.
- Speeding up the rotating scanner. With some experimentation it may be possible to increase the scanning speed dramatically, resulting in faster data capture.
- Designing stand-alone PCBs that hold all the components required to scan, as well as positioning LEDs and PTs in the correct position. With surface-mount LEDs this could become a very compact device with many potential applications.

References

[1] Jordan, K.J., Hilts, M. and Jirasek, A., 2017, May. Optical and X-ray computed tomography scanning of 3D dosimeters. In Journal of Physics: Conference Series (Vol. 847, No. 1, p. 012019). IOP Publishing.

[2] Huang, D., Swanson, E.A., Lin, C.P., Schuman, J.S., Stinson, W.G., Chang, W., Hee, M.R., Flotte, T., Gregory, K. and Puliafito, C.A., 1991. Optical coherence tomography. Science, 254(5035), pp.1178-1181.

[3] Corlu, A., Durduran, T., Choe, R., Schweiger, M., Hillman, E.M., Arridge, S.R. and Yodh, A.G., 2003. Uniqueness and wavelength optimization in continuous-wave multispectral diffuse optical tomography. Optics letters, 28(23), pp.2339-2341.

[4] Boas, D.A., Brooks, D.H., Miller, E.L., DiMarzio, C.A., Kilmer, M., Gaudette, R.J. and Zhang, Q., 2001. Imaging the body with diffuse optical tomography. IEEE signal processing magazine, 18(6), pp.57-75.

[5] Ntziachristos, V., Yodh, A.G., Schnall, M. and Chance, B., 2000. Concurrent MRI and diffuse optical tomography of breast after indocyanine green enhancement. Proceedings of the National Academy of Sciences, 97(6), pp.2767-2772.

[6] Culver, J.P., Siegel, A.M., Stott, J.J. and Boas, D.A., 2003. Volumetric diffuse optical tomography of brain activity. Optics letters, 28(21), pp.2061-2063.

[7] White, B.R., Snyder, A.Z., Cohen, A.L., Petersen, S.E., Raichle, M.E., Schlaggar, B.L. and Culver, J.P., 2009. Resting-state functional connectivity in the human brain revealed with diffuse optical tomography. Neuroimage, 47(1), pp.148-156.

[8] Arridge, S.R., 1999. Optical tomography in medical imaging. Inverse problems, 15(2), p.R41.

[9] Gore, J.C., Ranade, M., Maryanski, M.J. and Schulz, R.J., 1996. Radiation dose distributions in three dimensions from tomographic optical density scanning of polymer gels: I. Development of an optical scanner. Physics in Medicine & Biology, 41(12), p.2695.

[10] Oldham, M., 2004. Optical-CT scanning of polymer gels. In Journal of Physics: Conference Series (Vol. 3, No. 1, p. 122). IOP Publishing.

[11] Thomas, P.D., Darton, R.C. and Whalley, P.B., 1995. Liquid foam structure analysis by visible light tomography. The Chemical Engineering Journal and The Biochemical Engineering Journal, 56(3), pp.187-192.

[12] Thomas, P.D., Darton, R.C. and Whalley, P.B., 1998. Resolving the structure of cellular foams by the use of optical tomography. Industrial & engineering chemistry research, 37(3), pp.710-717.

[13] Wang, M. and Cilliers, J.J., 1999. Detecting non-uniform foam density using electrical resistance tomography. Chemical Engineering Science, 54(5), pp.707-712.

[14] Kjærsgaard-Rasmussen, J. and Meyer, K.E., 2011. Inside-out electrical capacitance tomography. Flow Measurement and Instrumentation, 22(2), pp.104-109.

[15] Hu, B., Stewart, C., Hale, C.P., Lawrence, C.J., Hall, A.R., Zwiens, H. and Hewitt, G.F., 2005. Development of an X-ray computed tomography (CT) system with sparse sources: application to three-phase pipe flow visualization. Experiments in fluids, 39(4), pp.667-678.

[16] Gornushkin, I.B., Merk, S., Demidov, A., Panne, U., Shabanov, S.V., Smith, B.W. and Omenetto, N., 2012. Tomography of single and double pulse laser-induced plasma using Radon transform technique. Spectrochimica Acta Part B: Atomic Spectroscopy, 76, pp.203-213.

[17] Gonzalez, R.C. and Woods, R.E., 2002. Digital image processing.

[18] Beylkin, G., 1987. Discrete radon transform. IEEE transactions on acoustics, speech, and signal processing, 35(2), pp.162-172.

[19] Pan, X., Sidky, E.Y. and Vannier, M., 2009. Why do commercial CT scanners still employ traditional, filtered back-projection for image reconstruction?. Inverse problems, 25(12), p.123009.

[20] Gordon, R., Bender, R. and Herman, G.T., 1970. Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography. Journal of theoretical Biology, 29(3), pp.471-481.

[21] van Aarle, W., Palenstijn, W.J., De Beenhouwer, J., Altantzis, T., Bals, S., Batenburg, K.J. and Sijbers, J., 2015. The ASTRA Toolbox: A platform for advanced algorithm development in electron tomography. Ultramicroscopy, 157, pp.35-47.

[22] van Aarle, W., Palenstijn, W.J., Cant, J., Janssens, E., Bleichrodt, F., Dabravolski, A., De Beenhouwer, J., Batenburg, K.J. and Sijbers, J., 2016. Fast and flexible X-ray tomography using the ASTRA toolbox. Optics express, 24(22), pp.25129-25147.

[23] Burgansky-Eliash, Z., Wollstein, G., Chu, T., Ramsey, J.D., Glymour, C., Noecker, R.J., Ishikawa, H. and Schuman, J.S., 2005. Optical coherence tomography machine learning classifiers for glaucoma detection: a preliminary study. Investigative ophthalmology & visual science, 46(11), pp.4147-4152.

[24] Menéndez, L.Á., de Cos Juez, F.J., Lasheras, F.S. and Riesgo, J.Á., 2010. Artificial neural networks applied to cancer detection in a breast screening programme. Mathematical and Computer Modelling, 52(7-8), pp.983-991.

[25] Rahman, M.M., Bhattacharya, P. and Desai, B.C., 2007. A framework for medical image retrieval using machine learning and statistical similarity matching techniques with relevance feedback. IEEE transactions on Information Technology in Biomedicine, 11(1), pp.58-69.

[26] Amat, F., Moussavi, F., Comolli, L.R., Elidan, G., Downing, K.H. and Horowitz, M., 2008. Markov random field based automatic image alignment for electron tomography. Journal of structural biology, 161(3), pp.260-275.

[27] Szabo, Z., Kao, P.F., Mathews, W.B., Ravert, H.T., Musachio, J.L., Scheffel, U. and Dannals, R.F., 1995. Positron emission tomography of 5-HT reuptake sites in the human brain with C-11 McN5652 extraction of characteristic images by artificial neural network analysis. Behavioural brain research, 73(1-2), pp.221-224.

[28] Veng-Pedersen, P. and Modi, N.B., 1992. Neural networks in pharmacodynamic modeling. Is current modeling practice of complex kinetic systems at a dead end?. Journal of pharmacokinetics and biopharmaceutics, 20(4), pp.397-412.

[29] Bottou, L., 2010. Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010 (pp. 177-186). Physica-Verlag HD.

[30] Rumelhart, D.E., Hinton, G.E. and Williams, R.J., 1986. Learning representations by back-propagating errors. nature, 323(6088), p.533.

[31] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J., 2011. Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), pp.2825-2830.

[32] Michael A. Nielsen, "Neural Networks and Deep Learning", Determination Press, 2015

[33] TSAL6200 Datasheet, Vishay Semiconductors - High Power Infrared Emitting Diode, 940 nm, GaAlAs, MQW. Document Number: 81010, Rev. 2.4, 13-Mar-14

[34] TEFT4300 Silicon NPN Phototransistor Datasheet, Vishay Semiconductors, Document Number: 81549, Rev. 1.7, 04-Aug-14

www.vishay.com/doc?91000

[35] Stoffregen, P - Teensy Specsheet, https://www.pjrc.com/teensy/techspecs.html, accessed October 2018

[36] K20 Sub-Family Datasheet - Freescale Semiconductor Document Number: K20P64M72SF1 Rev. 3, 11/2012

[37] How to get the best ADC accuracy in STM32 microcontrollers - Application Note AN2834, February 2017 (st.com)

[38] DRV8825 Stepper Motor Driver Carrier, High Current. https://www.pololu.com/product/2133, accessed October 2018

[39] Athani, V.V., 1997. Stepper motors: fundamentals, applications and design. New Age International.

[40] 'Nema 11 Hybrid Stepper Motor Datasheet' and specifications page from https://bilby3d.com.au, accessed October 2018

[41] 2018 Creality Ender 3 Review - https://all3dp.com/1/creality-ender-3-3d-printer-review/, accessed October 2018

[42] OSHWA Certification Directory http://certificate.oshwa.org/certification-directory/ accessed October 2018

[43] Creality documentation and design files, https://github.com/Creality3DPrinting/Ender-3 accessed October 2018

[44] Prajapati, D., Nandwana, S. and Aggarwal, V., Fused Deposition Modelling.

[45] Van Rossum, G. and Drake Jr, F.L., 1995. Python tutorial (p. 130). Amsterdam, The Netherlands: Centrum voor Wiskunde en Informatica.

[46] Pérez, F. and Granger, B.E., 2007. IPython: a system for interactive scientific computing. Computing in Science & Engineering, 9(3).

[47] McKinney, W., 2010, June. Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51-56).

[48] Rosenberg, K.M., 2002. CTSim 3. 5 User Manual. Heart.

[49] Joy, M. and Axford, T., 1991. GCODE: a revised standard for a graph representation for functional programs. ACM SIGPLAN Notices, 26(1), pp.133-139.

11 Appendix 1 - Model Selection

This appendix will describe the reasoning behind the use of a neural network based approach to machine learning in this project. Accompanying code and further explanation can be found in the 'Model Selection' folder of the Github repository submitted along with this report.

One goal of this project is to answer a fairly broad question: can we use machine learning to make inferences from the readings we'll be taking.

To answer this, we need to do some kind of machine learning. There will be many sub-tasks, both classification and regression. For each one, we could go through a rigorous model selection approach, but it's nice to have one or two good general purpose models that we can use. We don't care so much about the question 'what is the absolute best performance we could achieve here' - all we want to know is whether ML can be useful in solving a given problem.

So the goal is to find one or two models that can generalize to the different tasks we'll be trying. The chosen models will need to have the following characteristics:

- General purpose. Some algorithms might work for one specific problem, but we want a model that can be easily adapted to the different situations
- Robust. We will be using fairly small training datasets, so the model will need some way of combating overfitting. There is a danger that an overly complicated model will simply fit the noise in the training data and not generalize well.
- Capable of fitting complex functions. Simple linear models might work for some of the problems considered, but a good model would be able to handle the non-linearities inherent in our experimental setup. Light scatters, diffracts, reflects and attenuates. A more complex model would probably be better suited to this kind of problem.

Initial model selection was based on some quick experimentation, coupled with past experience in this field. However, to provide justification for the choices made, this section will re-create the model selection and tuning process in a more structured way.

The first task we will consider is the object identification task described in Section 5. A classification problem, with many different potential solutions.

With the problem defined, we can select some models and see how well they perform. Promising models are then investigated further, tuning key parameters and measuring how these changes affect performance. The following model classes were investigated:

- Logistic Regression
- Support Vector Machines (SVMs)
- Decision Trees
- Random Forest Classification (an ensemble learning approach)
- Neural Networks (Multi-layer perceptron type - a fully-connected feed-forward architecture as described in the literature review)

The SVM-based classifier performed badly at first, but adjusting the regularization parameter resulted in very good performance.

The Random Forest model performed well, and did even better with more estimators and some other tweaks.

The neural network classifier performed the best out of all the models tested.

The chosen network architecture has 3 hidden layers, each with 20 neurons. As Figures 43 illustrates, adding more hidden layers or using fewer hidden layers reduces performance. Increasing the size of the hidden layers increases complexity without improving performance for N>20. For this particular task, fewer than 20 may have been appropriate. For more complex tasks such as image inference, slightly more was found to be necessary. A size of 20 resulted in a good general purpose model, useful in all cases.

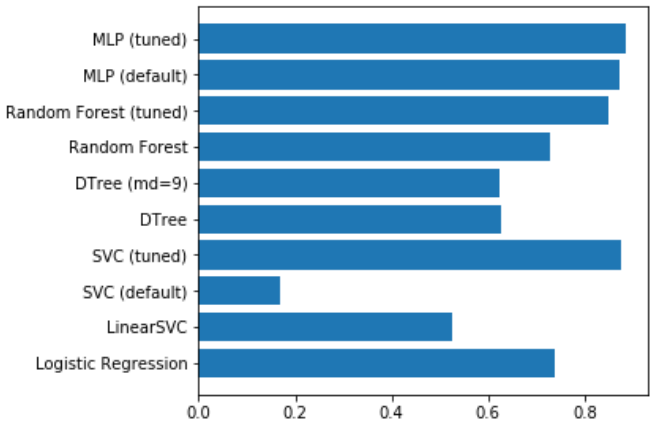


Figure 41: Comparing model performance

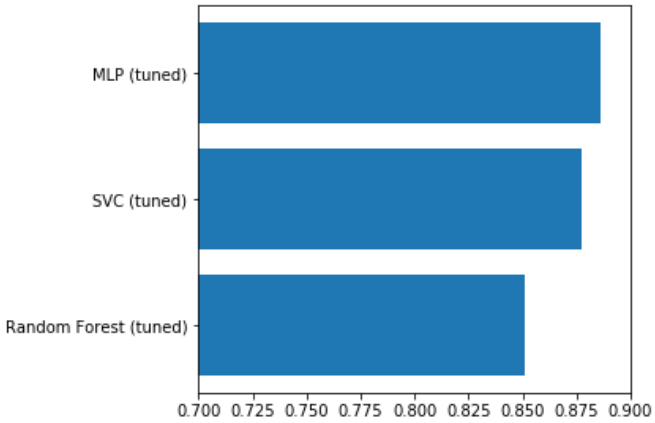


Figure 42: The best performing models

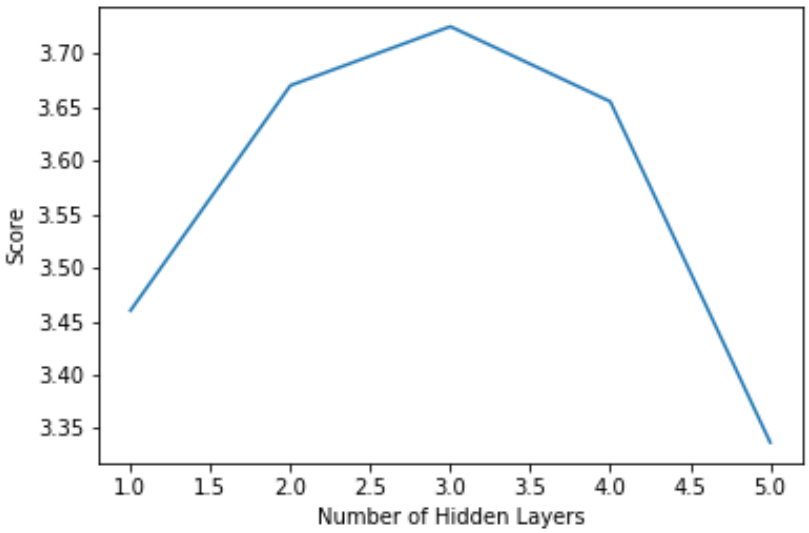


Figure 43: NN performance with more hidden layers

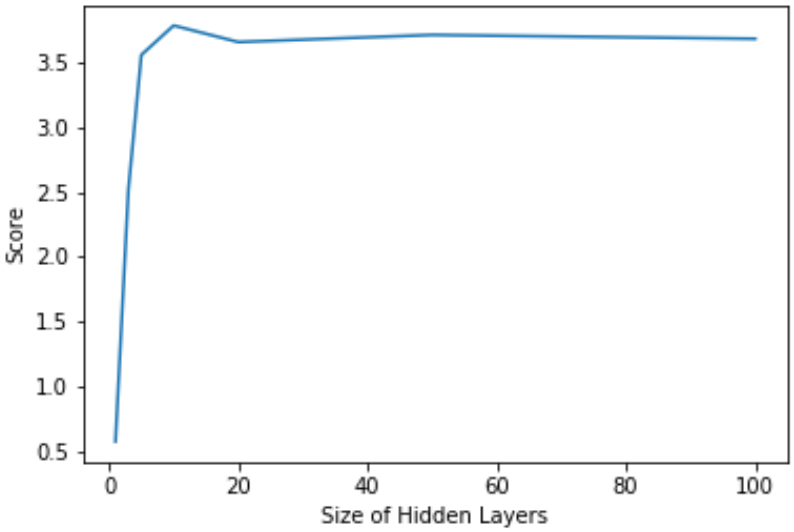


Figure 44: Performance vs Size of Hidden Layers

12 Appendix 2 - CAD Files

Many parts for this project were 3D printed. The STL files are available in the github repository accompanying this project, and editable versions can be found online at https://cad.onshape.com/documents/62c0b2d636

The parts for the rotating scanner are shown in the following table, along with approximate weight and printing time for each part. All parts were printed in PLA plastic at 50mm/s, with 40% infill.

Part	Print Time	Weight (g)	Total Weight (g)
Gear x2	6 minutes	2	4
Inner plate	4h28	47	47
Outer Ring	10h51	121	121
Electronics box	2h15	17	17
Stepper Mount x2	4h20	45	90
Base Plate	5h19	46	46

Total Weight: 325g Total Print time: 31 hours 29 minutes

Other parts used in this project:
Small ring: 55 minutes (7g)
Large ring (16 LEDS, 16 PTs): 2h53 (22g)

13 Appendix 3 - GitHub Repository

Code, data and supplementary material for this report is available on GitHub: https://github.com/johnowhitaker/CIRI

The link is included here, as the VULA assignment had some issues. The readme contains a brief description of the contents of the repository, duplicated here:

CAD Files: Contains .stl files for the 3D printed parts used to create the rotating platform and the fixed rings of sensors.

Misc Notebooks: In the interest of transparency, all notebooks created as part of this project are included here. There is a lot of duplication, and some have been modified and edited in such a way as to obscure their original purpose. Most should execute correctly but I cannot guarentee that all of them will work as expected. THESE SHOULD BE IGNORED - please see the selected

notebooks in the 'Summary Notebooks' folder.

Summary Notebooks: These notebooks show the majority of the code developed for this project with some explanations and example data. They should give an overview of the techniques used, show the results of the different experiments and provide a starting point for anyone wanting to try this for themselves.

Microcontroller Code: This folder contains the code for the Teensy boards used in this project. The files can be opened with the Arduino IDE. Each one was used for a different aspect of the project: - adc_speed_test was used to benchmark the ADC on the Teensy 3.6 - serial_c is the code used to control the rotating platform. It used various charachters as commands to move the two axes and read the ADC. Clunky, but it worked well enough that it was never improved. - r8_fast and r8_small are very similar. They take a set of readings and send it in response to a command from the host PC. Used for the ring of 8 and its smaller brother. See the related notebooks in 'Summary Notebooks' for examples of Python code to read the data over serial. - ring_of_8 was used before I switched to the approach in r8_fast. It gave control over each LED and ADC reading via serial. Again, clunky but functional - 36_read_adcs Takes a set of readings for the ring of 16 and sends over serial. Very similar to r8_fast. Can also be modified to only send the sparse readings (see section 6 of the report).

Model Selection: Contains a notebook and some example data demonstrating the model selection process and showing the rationale behind the neural network architecture used in this project.