Initial meeting

12 November 2021 10:20

Gadgetron & BART - need to download and run - need to pick one (probably Gadgetron).
Before Christmas. Can download GEANT-4? Maybe do both.
Need CT don't care about MRI.
First step is to play around with it.
GEANT-4 simulator. Use to determine deviation. Need to do tutorial.
Need own simulator.
Check out past write-ups. https://www.hep.phy.cam.ac.uk/~lester/teaching/PartIIIProjects/index.html
Best reference:
F. Keizer et al. “A compact, high resolution tracker for cosmic ray muon scattering tomography using semiconductor sensors”. In: Journal of Instrumentation 13.10 (Oct. 2018), P10028–P10028. doi: 10.1088/1748-0221/13/10/p10028. url: https://doi. org/10.1088%2F1748-0221%2F13%2F10%2Fp10028.

Can SSH onto computers in the department. Need equivalent of "screen". Julia A. Can use Chris' desktop. Apply for account straight away.

Plan - quite broad, what am I doing when? Region of validity/physics also.

Feasibility

08 December 2021 15:38

Read an interesting paper, mostly about MRI, but still very applicable
https://cdn0.scrvt.com/39b415fb07de4d9656c7b516d8e2d907/1800000003913232/f165deac5e03/magnetom-flash-68_gadgetron-open-source-image-reconstruction_hansen_1800000003913232.pdf

Some conclusions:

- It uses a client-server architecture - so I can just install it on a cavendish desktop (or on the cloud) and send data via TCP/IP from my laptop
- I likely need to write a new "reader module", which can be written in C++, python or MatLab. Suggested to initially write in python, then later convert (parts) to C++ for perf.
- There are a number of different reconstruction routes, and a number of "Gadgets" - I should look into what type of reconstruction I need to do. Probably best to read some papers, and best case scenario all the gadgets already exist. But hopefully at least *some* steps exists. I can always implement two instances of gadgetron, separated by my own gadget, for example.
- May need to use docker - and this *might* fix my integration tests on my machine
- But overall, looks ***very feasible*** which is good news!

Initial pass at Keizer

09 December 2021 11:10

- The paper contains lots of information about set up and errors - but I think the best approach will be to write the algorithm first, then add on this at the end - so come back to later on
- Paper contains useful information about scattering theory
- Type of data produced discussed in section 2.4
- Ideally, want to write **generic** code which doesn't depend on detector set-up. So look at other papers to see what the data has in common.
- Some more algorithm information can be found in references 16/19/20

https://iopscience.iop.org/article/10.1088/1748-0221/13/10/P10028/pdf

https://arxiv.org/ftp/arxiv/papers/1805/1805.11002.pdf - another example, to read at a later date.

Gadgetron Installation Tips

09 December 2021 11:47

- Use the previous commit for instructions - Ninja doesn't work
- There is a better description of how to run the Integration tests
- May need to also install the python interface for *ismrmrd* so the module can be found
- May need to *chown* for test repo - but try running download step without *sudo* first
- Docker installation may or may not be required. If the tests don't work try a reboot first. Then do the DNS messing.

To fix hdf5 problem - all LINKLIBRARIES in build.ninja add a /hdf5/serial/include for every hdf5/serial. Then for all cmake_install.cmake files. Do the same to oldrpath. And newrpath. **Or just copy the header file** from serial/include to serial.
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib

Week 1 - Gadgetron Hello World

31 January 2022 16:05

- Hit various issues with working on HEP computers
- Pivoted to use Azure
- Tried installing without ninja, but issues with finding hdf5.h, which weren't fixed by updating PATH
- After trying for a while, delete everything, and attempt installation again, but this time with ninja

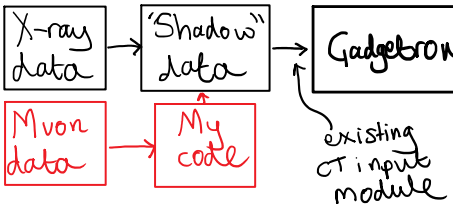
Installing in Windows

- For ANT, need to SAVE path variables

Viewing The Image
sudo apt install default-jre // Install Java Runtime Environment
sudo apt install ant // Install ant
sudo apt install hdfview // Install hdfview

Meeting 2

16 March 2022 18:05



$$\theta_0 = \frac{z \cdot 13.6\,\text{MeV}}{p\beta c}\sqrt{\frac{x}{X_0}}\left[1 + 0.038 \cdot \ln\left(\frac{x}{X_0}\right)\right]$$

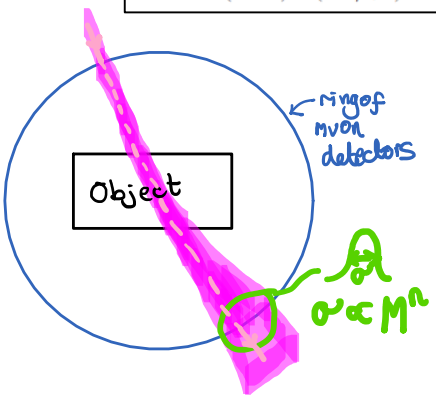
$$X_0 = \left[\frac{A \cdot 716.4\,\text{g cm}^{-2}}{Z(Z+1)\ln(287/\sqrt{Z})}\right]\left[\frac{1}{\rho}\right]$$

We talked about shoehorning the muon data into "shadow" data, then passing directly through the CT gadgetron.

- As muons pass through an object, they are deflected.
- Collecting data for ~20 muons entering at the same angle and position, should get a gaussian spread.
- The width of the gaussian is in some way related to the amount of mass passed through.
- Work out what shadow data would be generated by this amount of mass, and repeat for all incoming muon angles/positions.
- You then have fake CT data to feed to gadgetron.

Initial pass would be to create eg. a python script to generate fake muon data for a known set of massive objects - assuming the idealised gaussian relationship. Then see if can recreate the set of shapes from this fake output data. Some extensions/thoughts:

- The relationship may not actually be gaussian - does this affect the results?
- This requires taking ~20 times more data points than eventually generated for the CT data - an extension of the project would be writing a complete input layer to take muon data straight to gadgetron.



Writing program to generate fake data

17 March 2022 11:03

1. Find model for how spread of Gaussian is related to average material crossed.
2. Pick shapes to scan.
3. Write program to work out average material crossed for muon "beam".
4. Generate input data.
5. Generate output data with gaussian of appropriate width.

(1)

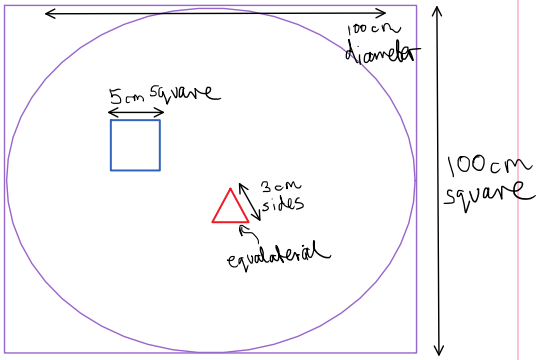
$$\theta_0 = \frac{z \cdot 13.6\,\text{MeV}}{p\beta c}\sqrt{\frac{x}{X_0}}\left[1 + 0.038 \cdot \ln\left(\frac{x}{X_0}\right)\right]$$

$$X_0 = \left[\frac{A \cdot 716.4\,\text{g cm}^{-2}}{Z(Z+1)\ln(287/\sqrt{Z})}\right]\left[\frac{1}{\rho}\right]$$

z=1

Depends on atomic number, mass number & density.
Also depends on β for muons - this is fixed for cosmic rays - average energy 4 GeV. So $\beta \approx 1$. More precisely $\beta = 0.9867$. Since I'm only using an average energy, which is given to 1 s.f. I'll use β to the same accuracy, so set $\beta = 1$. I can easily change this later, as it will just be a single input number to the code.

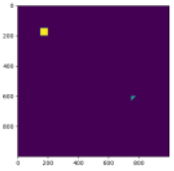
(2)
I'll use a Uranium Square and a Lead Triangle. The Keizer paper mentions using muon tomography to scan warships for nuclear material, and lead is quite dense, and the sort of thing that might be used in buildings. Both are elements, which makes it easier to apply the formula



(3)
First step is to represent the situation in python. A first attempt could involve an array of vectors. A square array to represent the box (we start by imagining in 2 dimensions). Then at each point, we need the density, atomic number and mass number of any object present. If no object, either zeros, or a bool? I'll experiment with different array sizes, to get the best possible resolution.

Radon transform / inverse radon transforms
https://en.wikipedia.org/wiki/Radon_transform

(4)
I can now create an array representing the physical situation. I changed to a right-angled triangle, as this is easier to represent in a square array. I have a "material" object containing all the required physical properties. I can easily change the size of the region, and the sizes and positions of the square/triangle. I did a quick check of generating a density map:



I think for my first attempt at data, some bigger objects would be helpful (these are the same dimensions as in my initial sketch, part (2).

(5) Next, need to write a function to determine which array elements a ray passes through. This will need to know the position, angle (in degrees) and array element length. It will assume a square array. Need to work out whether CT data assumes a square or a circle for cone beam, or if this is configurable. A square is preferable. Shouldn't really affect this generation phase, so going with a square.

(6) Add a X_0 value to the Material class, to simplify the next part. Then write the function which adds up all the scattering. I don't think I can just add up each element, because the scattering is not proportional to distance travelled through the material. I can work out how far is travelled through a single object, but I'm not sure how to accurately add the angles up for multiple objects.

Essentially need some sort of derivative - the formula has probably been integrated from something.

Can average material properties and calculate. Or try adding in quadrature (requires uncorrelated).

> Eq. (27.10) describes scattering from a single material, while the usual problem involves the multiple scattering of a particle traversing many different layers and mixtures. Since it is from a fit to a Molière distribution, it is incorrect to add the individual θ_0 contributions in quadrature; the result is systematically too small. It is much more accurate to apply Eq. (27.10) once, after finding x and X_0 for the combined scatterer.

Investigation of required CT data type and Gadgetron Set-up

17 March 2022 11:03

1. Research into what gadgets and input modules are typically used for CT.
2. What data format does the input module expect?
3. Find exact relationship between amount/type of material, and amount of shadow.

Conversion design

17 March 2022 11:04

1. Work out which steps need to be done.
2. High Level Design - pitch to Chris.
3. Low Level Design - pitch to Chris.

Which steps need to be done, conceptually:

- I'll start with the conceptual data, so a set with same incidence angle and position (This is not how the data comes in, but I can add an initial processing layer on the top if time)
- Work out the amount of mass along the line
  - Function to work out gaussian spread (may just be one line of built in function)
  - Function to work out amount of mass
- For a given incidence position, work out the amount of mass at each angle
- Convert into cone-beam x-ray data
  - This is the complicated step
  - NeXuS data format is used for CT and Muons which is promising (and is supported in CIL)
- Feed this data into CIL

FANBEAM.

Might be easier to just create a custom acquisition geometry as no concrete CT examples in NeXus docs.

NeXus thoughts:
Likely just need entry and Data (sample & Instrument not required at earliest point). The NXdata contains axis labels and titles as well as the data. The example on the page is instructive, and seems to take simple python lists, which is what I already have the data in, so yay! Look for definition for cone beam data - this is likely also built into CIL, so makes life easier (eg. don't have to specify axes). We actually want fanbeam.

Acquisition geometry:

- Fan beam (= cone beam 2D)
- The "angles" correspond to rotating the object, which is equivalent to rotating the source and detector around the object (in a clockwise direction, with 0 degrees corresponding to "to the left")
- Can then specify the number of detector pixels and the pixel height, which is equivalent so specifying the angle the ray leaves the detector at. First pixel corresponds to bottom pixel.

X-ray photon energies are 100eV - 100keV (10^-4 - 10^-1). Example attenuation coefficients at 10-3 & 10-1:
H - 7/0.2
Lead - 4000/10
Uranium - 6000/8
Platinum - 3000/10

So we need to specify an X-ray energy, but then we can just use some typical value, to first approximation.

These values are in cm^2/g. So need gcm^-2 (ie density x distance as we have already).

For the initial pass, its fine to just pick a random number in the appropriate range - as this should still give us a relative density map.

Which CT package to use?

04 April 2022 10:40

A paper: https://pubs.rsna.org/doi/pdf/10.1148/radiol.2015132766

- I have a mild concern that the IR method corrections may assume the data is actually CT data, with the associated noise being down to being an x-ray source, which might be counterproductive. But may not matter in the grand scheme of things. Still could be worth seeing if a radon method can be found.
- IR requires "more computing power". Slight worry about if my VM will handle, but it seems its more computing power compared to the 1970s so should be fine! But still bear in mind.

Two options:

http://www.ccpi.ac.uk/Flagship - this is "multichannel" - implies more info than just the mass required? Refers to different energy channels. But is, crucially, open source. The page http://www.ccpi.ac.uk/CIL is probably more useful. Contains the "core imaging libraries" and installation instructions. Part I likely to be more useful (no reference to multichannel). This is an IR algorithm. It is written in python. Examples of how to use found in jupyter notebooks. May well be able to copy-paste one of these. They also include making figures which will save me even more time so sounds promising. Only issue is 2D vs 3D - need to check. Check shows that some of the images produced are 2D slices, so should be fine. The second example seems closest to CT. They also have neutron tomography which could be interesting. There's a paper to read at https://royalsocietypublishing.org/doi/10.1098/rsta.2020.0192. This looks generally promising, should just have to input geometry of detector.

https://github.com/dchansen/protonCT. This uses gadgetron, but the GPU elements (which I think are problematic with my VM). https://en.wikipedia.org/wiki/Proton_computed_tomography . This uses protons rather than x-rays to scan. So does have a charge factor, so more similar to muons in that sense. Obviously opposite charge and a composite particle unlike muons. Gut feeling is that this is going to be less useful and more complicated than a comparison to x-rays themselves. I may be following a similar process though so keep the links.

Google results:
https://www.slicer.org/ seems to do CT slices on initial inspection.
https://www.opensourceimaging.org/project/mirt/ does x-ray CT. Uses Matlab.
https://github.com/CERN/TIGRE CBCT. Says it is "off the shelf". Also by CERN. Is 3D so will have to look into.

There are many more, but hopefully one of the 4 highlighted will work. Doesn't need to be the optimal algorithm at this stage, so ease of use is the main focus.

Meeting 05/04

05 April 2022 11:22

- I want to re-check the exact data expected. I think its position and angle for in and out.
    - Actual data is multiple points
- We have consistently referred to a "ring" of dectectors. Does this represent reality (ie. Expect a 2D image). Is this ring expected to be manoeuvrable so expect 3D. Could always start with 2D either way.
    - Doesn't really matter at this point. Can start in 2D.
- Already know what type of CT (eg. CBCT) or is that for me to find out?
- Discussion of packages and information so far.

CIL "Hello World"

14 April 2022 14:15

conda create --name cil -c conda-forge -c intel -c astra-toolbox/label/dev -c ccpi cil cil-astra ccpi-regulariser tigre tomophantom=1.4.10 ipywidgets

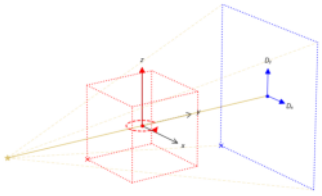
From <https://discord.com/channels/929016277266219038/930461090024931378>

There are a number of examples in the "CIL Demos" repo. They are helpfully split into "weeks". They are Jupyter notebooks. I need an environment to run them in, so using "Binder". First step is to get that running.

Week 1, part 0 - CIL Geometry

Show_geometry(), which should print an image of the detector geometry (very useful!) has an "AttributeError". Known bug with matplotlib 3.5 - there is a fix, but CIL is not my code so tried downgrading to 3.4.3. This worked!

It looks like I will need to use a "cone beam geometry". The different angles of entry will correspond to the different "fanning" angles. Cone beam geometry shown below:



Cone beam actually appears to be only one point source so not what I want. Want multiple source positions.

There's a bunch of stuff about offsets for imperfect CT set-ups. Not going to be needed for initial set up, but might come back to later. Just skim-reading for now.

Image geometry more confusing. But can likely just use defaults, which are based on acquisition geometry.

Week 1, Part 1 - An Introductory Demo

This is a full demo. Got demo working, and will come back to later.

Had to stop because problem with binder hub

SSH things

jupyter notebook --no-browser --port 1307

03 May 2022 11:15

To ssh onto department computer:
ssh -o "ProxyJump jcs213@gw.hep.phy.cam.ac.uk" jcs213@PCLU.hep.phy.cam.ac.uk
To set up ssh tunnel
ssh username@xx.xx.xx.xx -NL 1234:localhost:1234

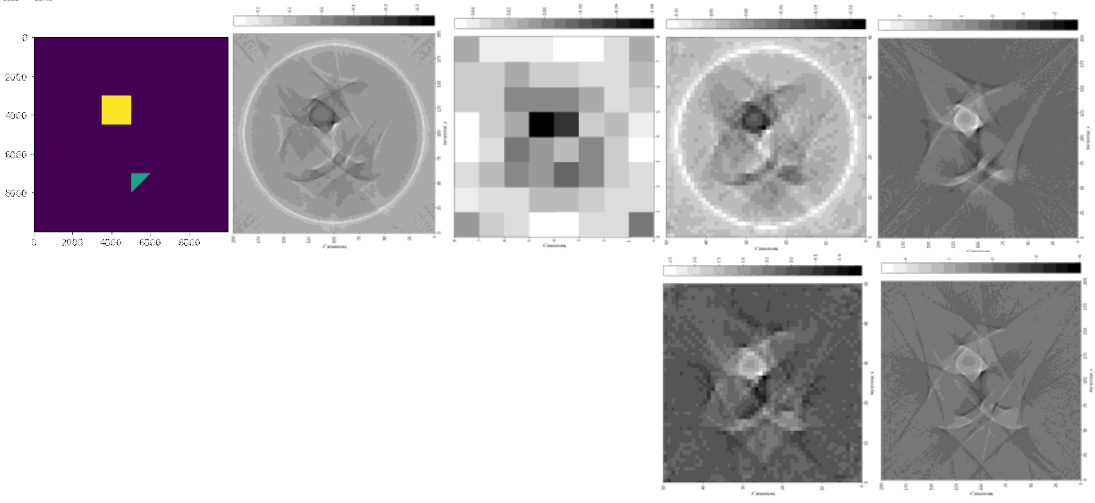
So can try
ssh jcs213@PCLU.hep.phy.cam.ac.uk -NL 1234:localhost:1234
But probs need
ssh -o "ProxyJump jcs213@gw.hep.phy.cam.ac.uk" jcs213@PCLU.hep.phy.cam.ac.uk -NL 1307:localhost:1307

This works!!!!! Maybe a little slow but is generalyy working.

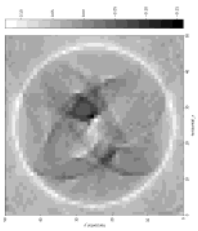
scp -r -o "ProxyJump jcs213@gw.hep.phy.cam.ac.uk" .\process_data\ jcs213@PCLU.hep.phy.cam.ac.uk:/work/jcs213

Initial pass

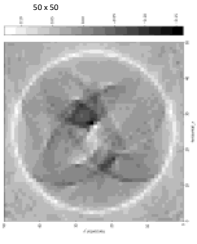
20 May 2022 15:43



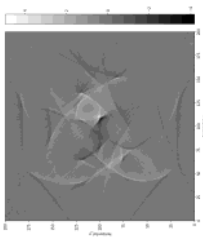
200 x 200



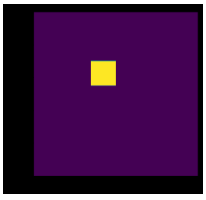
8 x 8



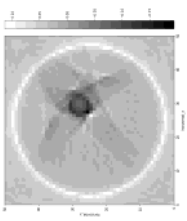
50 x 50

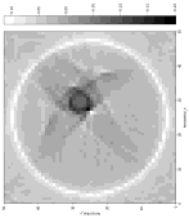


Generate_data_2

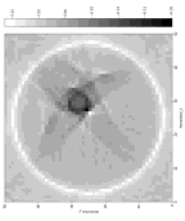


50 x 50

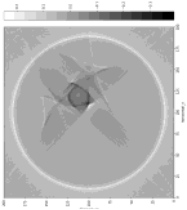




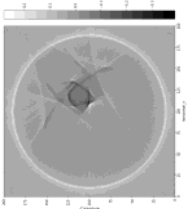
50 x 50



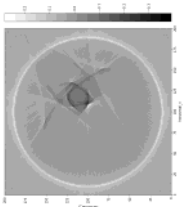
With averaging

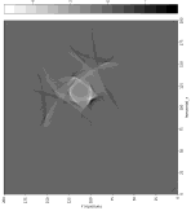


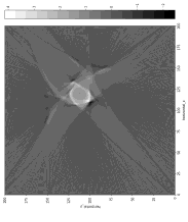
With exact calculation

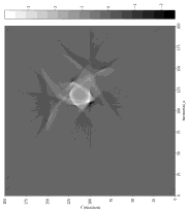


Increased exponential

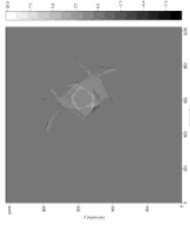


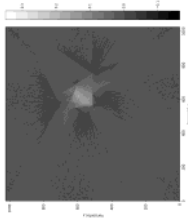






Smaller attenuation coefficient

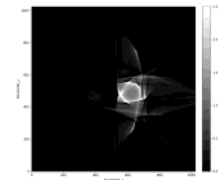


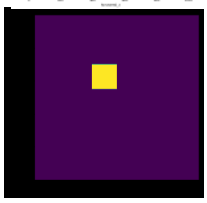


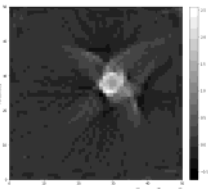
5000

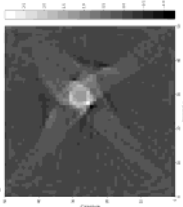
0.001

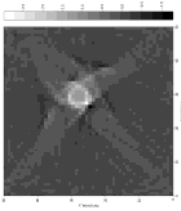
1

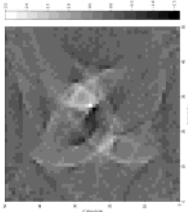


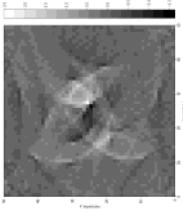






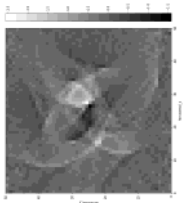


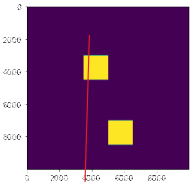


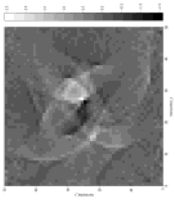


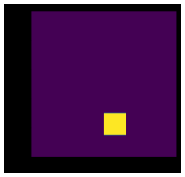
New generation 50x50

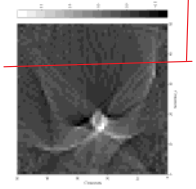
Old generation 50x50

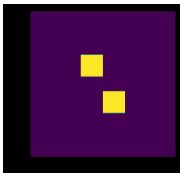


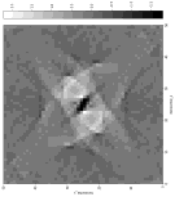


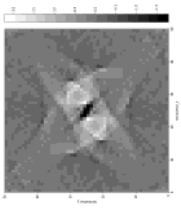




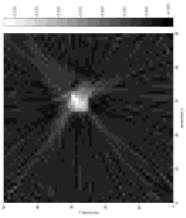




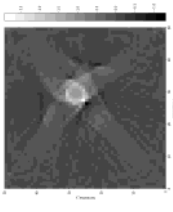




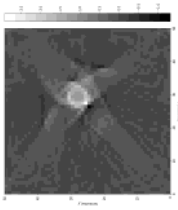
Coef = 5000



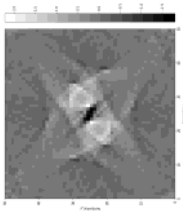
Coef = 0.001



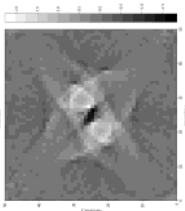
Coef = 1



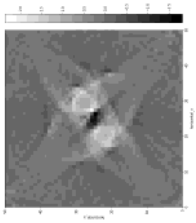
Coef =10



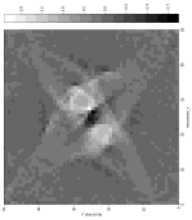
Coef = 250



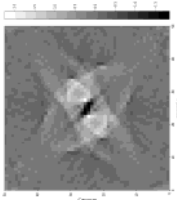
Coef = 50



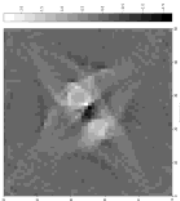
Coef = 20



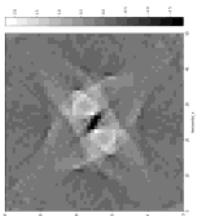
Coef = 13

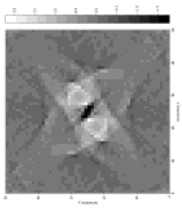


Coef = 5000

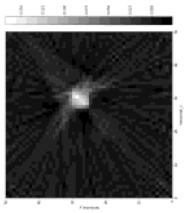


Coef = 11

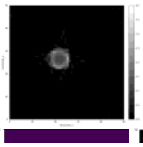




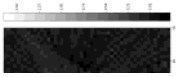
5000

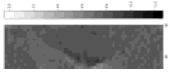


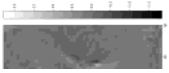
0.001

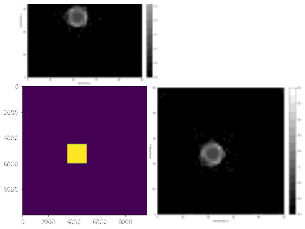


10









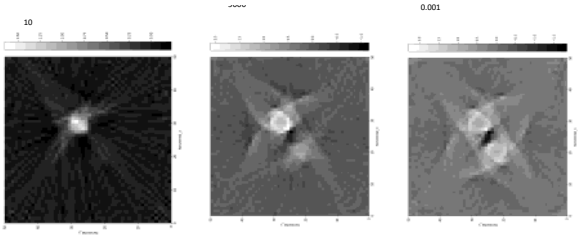
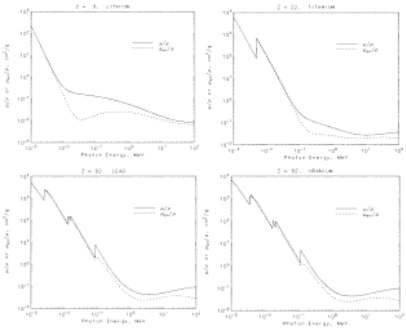
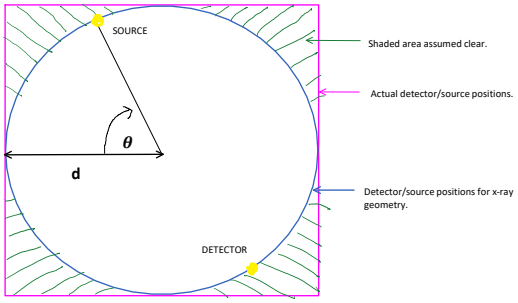
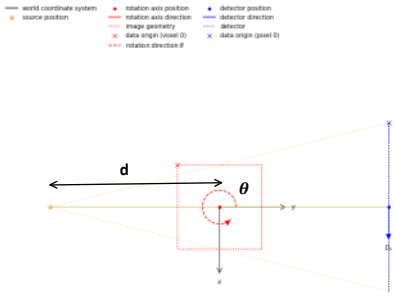

10
0.001
0.1
0.01
1

Figure making

21 May 2022 17:25







A	B	3
C	D	3

1 5

1	2
0	3

23 May 2022 09:57

Discussion of value of C.
Discussion of results.
Is c=1?

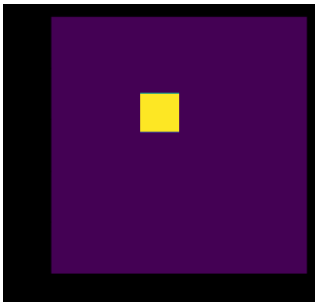
The effect of changing pixel number

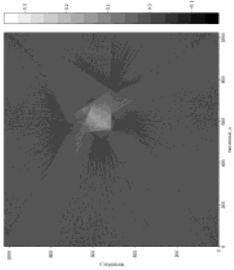
24 May 2022 11:53

50x50

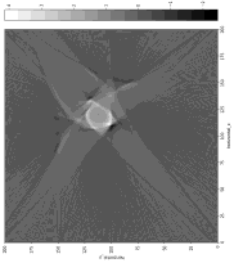
200x200

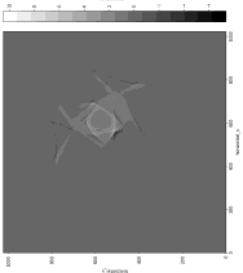
1024x1024



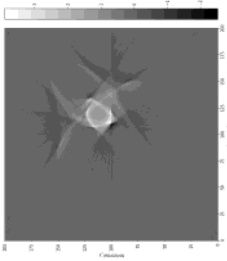


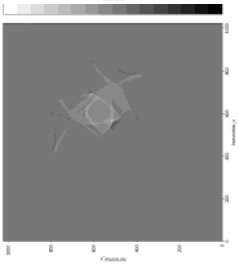
Mu = 0.001





Mu = 1





Mu = 5000

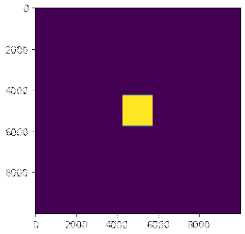
Central square

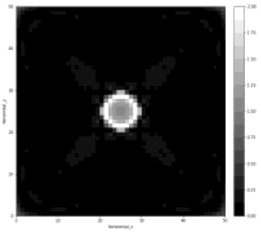
24 May 2022 12:09

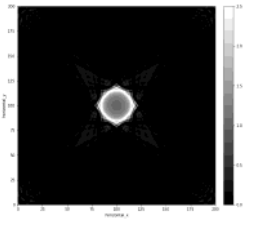
50x50

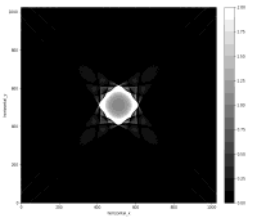
200 x 200

1024x1024

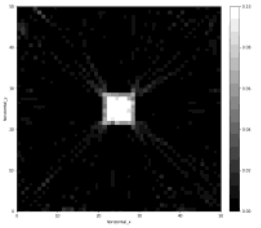


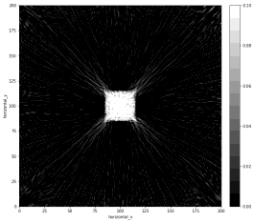


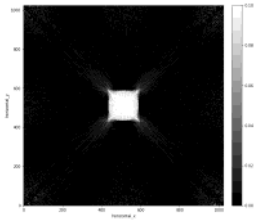




Coefficient = 1

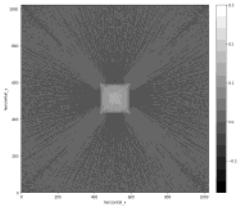


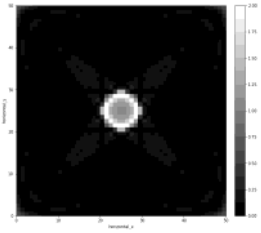


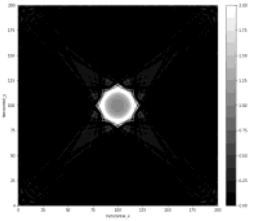


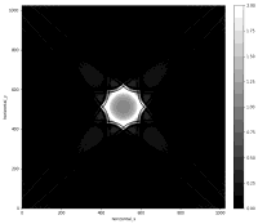
Coefficient = 0.001

Artefacts:

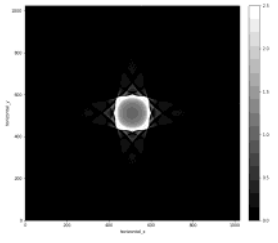








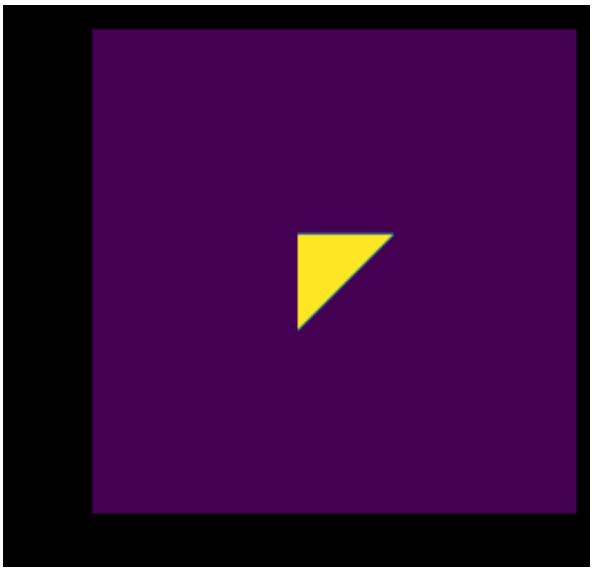
Coefficient = 1000

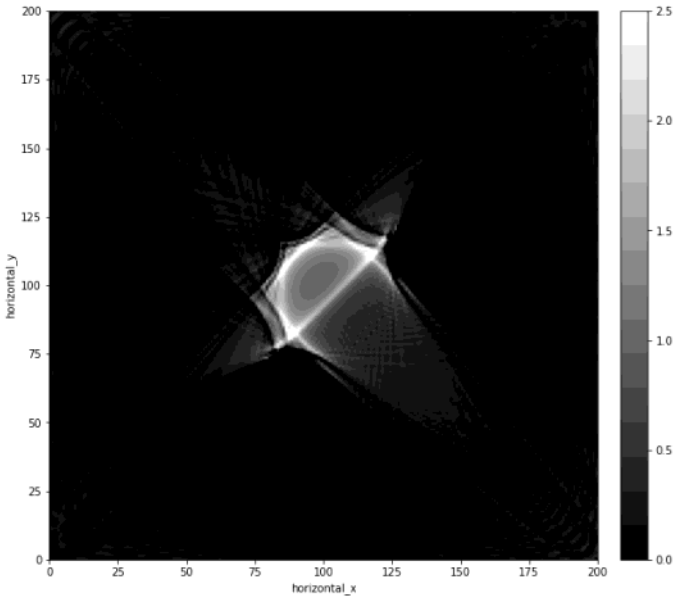


Central Triangle

25 May 2022 10:50

200x200



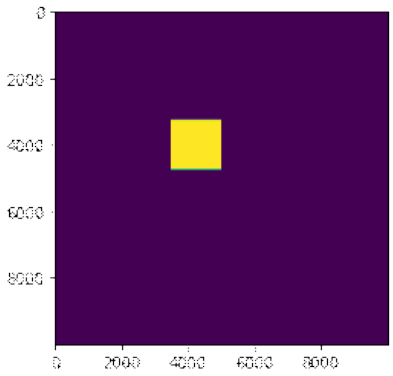


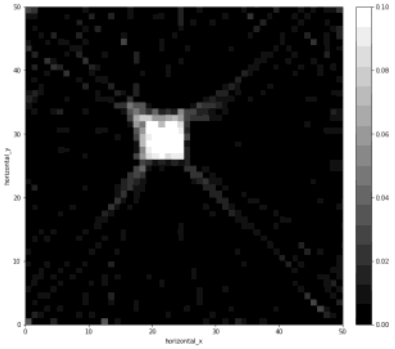
Offset Square

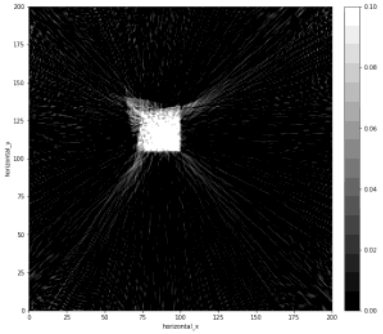
25 May 2022 15:07

50x50

200x200







attenCo = 0.001

Presentation things

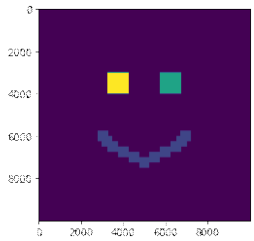
25 May 2022 16:02

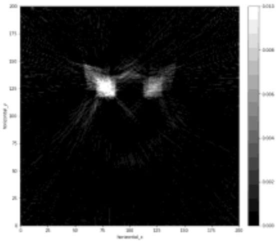
$$\theta_0 \propto \sqrt{F(A,Z)\rho x}(1 + 0.038\ln[F(A,Z)\rho x]\ )$$

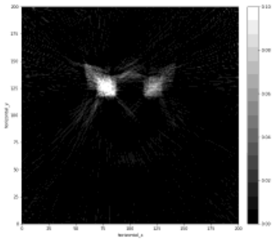
$$F = F(A,Z)$$

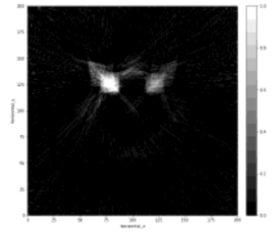
Smiley faces

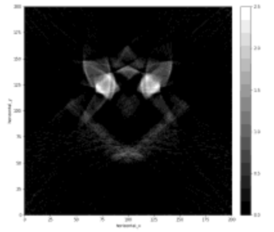
25 May 2022 18:59

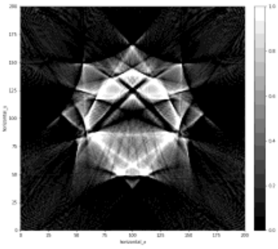


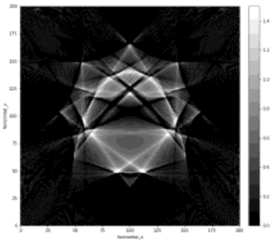


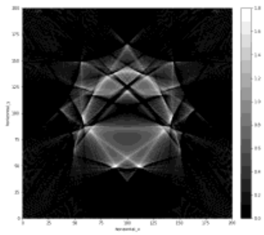


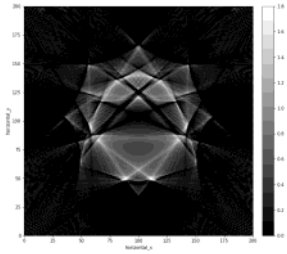


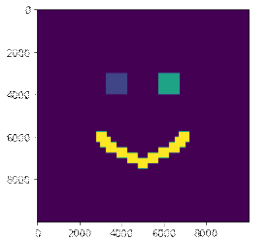


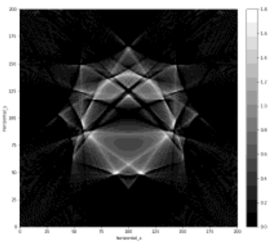


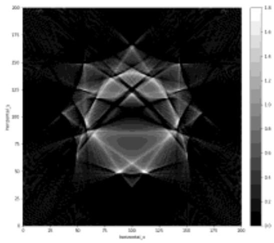


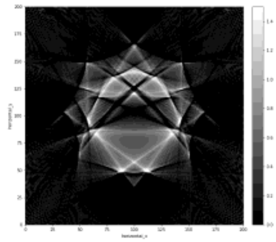


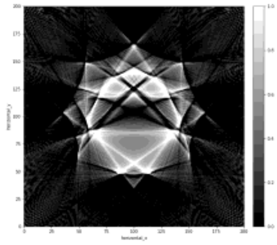


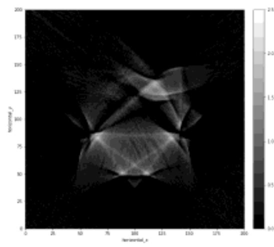


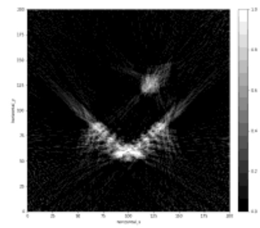


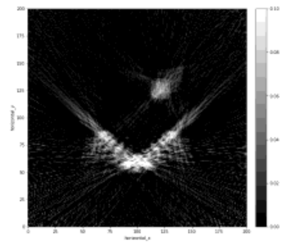


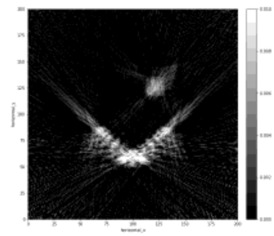


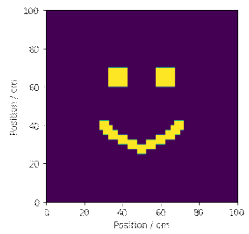


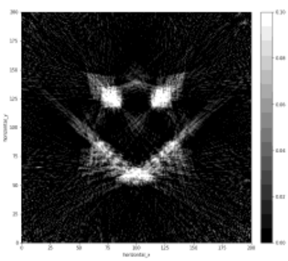


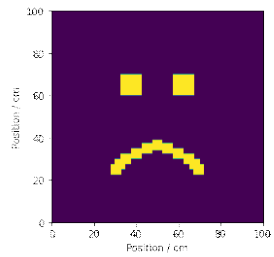












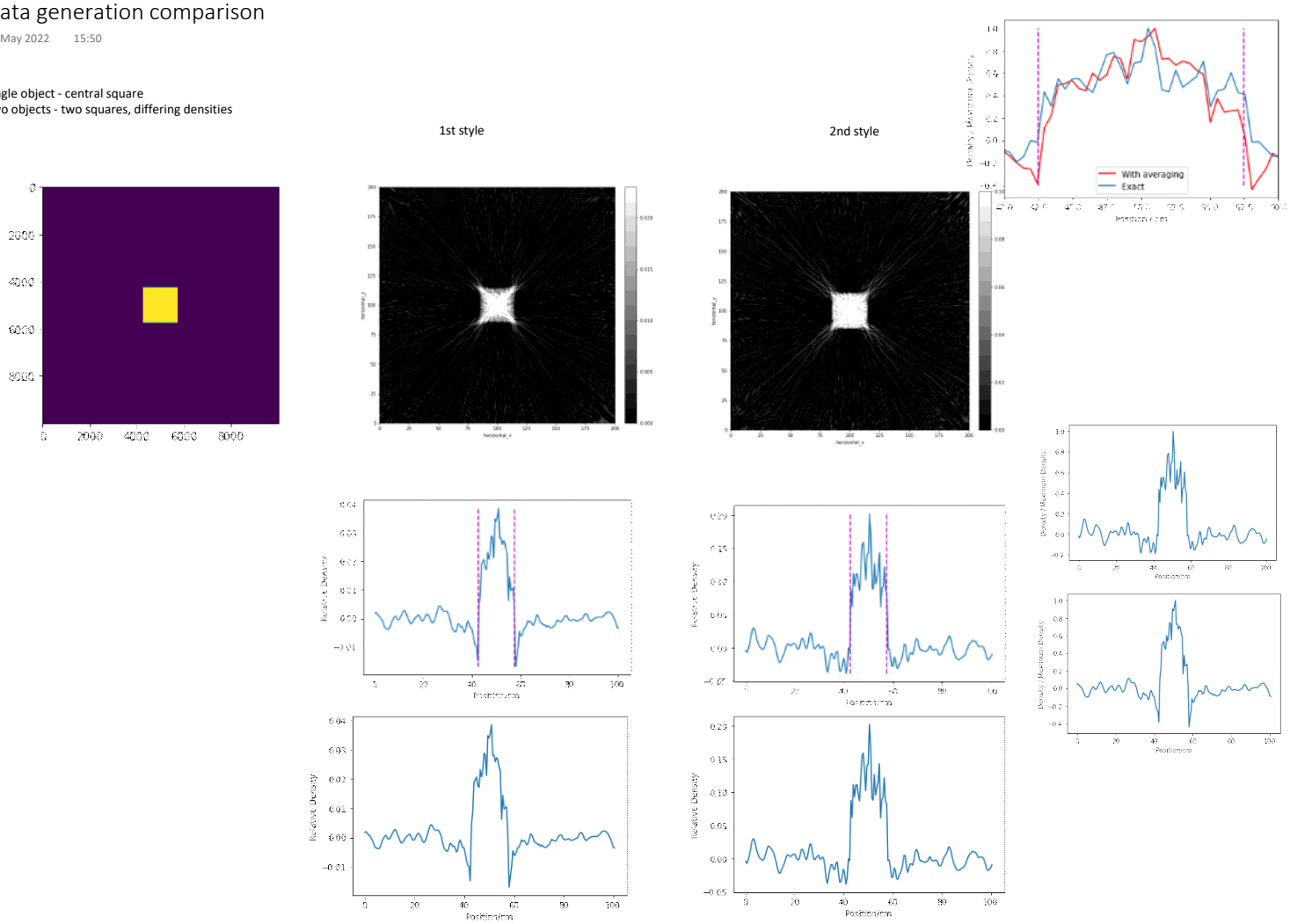
Data generation comparison

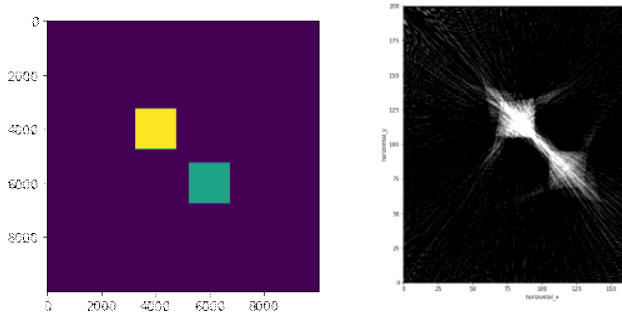
28 May 2022 15:50

Single object - central square
Two objects - two squares, differing densities

1st style

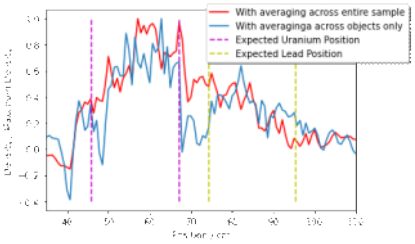
2nd style





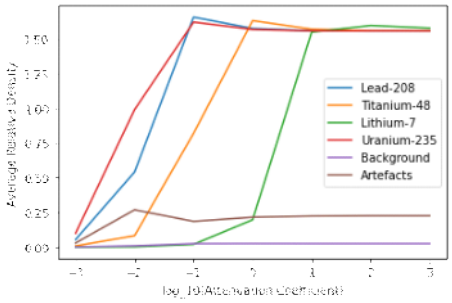
0.001

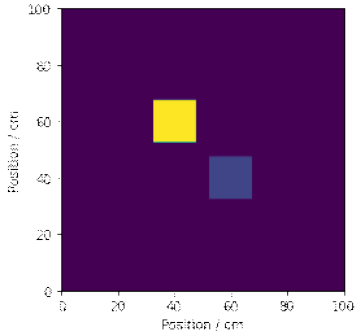
1

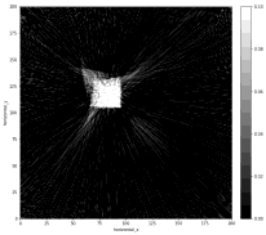

With averaging across entire sample
With averaginga across objects only
Expected Uranium Position
Expected Lead Position

Attenuation Coefficient Comparison

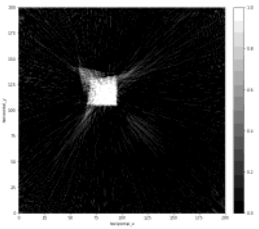
29 May 2022 16:14



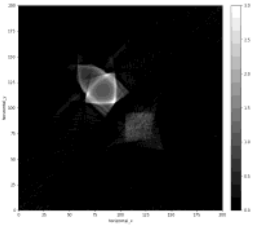




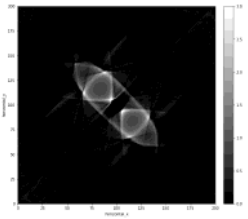
0.001



0.01



0.1



1