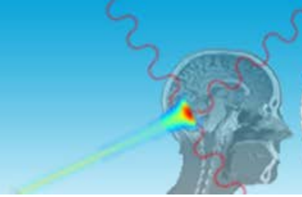


Phase II pCT Geant4 code: short update

George Dedes, Jannis Dickmann, Philipp Wesp, Guillaume Landry

Department of Medical Physics, LMU Munich

Loma Linda pCT workshop 2018

History

- Started about 3 years from the code shared by Valentina Giacometti and Pierluigi Piersimoni

- That code already implemented the pCT phase II scanner geometry in detail

- The output of the code (ascii or binary) contained already the information needed by the reconstruction (reconstruction data output):
  - T, V, U coordinates in mm and calibrated WEPL in mm

- Was already versioned in github, contained in the master branch

Reconstruction data output development

- Development in that code (reconstruction data output):
  - Cleaning obsolete/commented parts
  - Added functionalities that made it easier running in computer clusters (passing random number seeds, output paths and file names, etc)
  - Added the back then new wedge calibration phantom
  - etc

Reconstruction data output pros and cons

- Pros:
  - Easy to eventually obtain a reconstructed image from this simulation
  - No need to care about calibration and other „real world“ details
  - Calibration was performed once in a stand-alone code and hardcoded in the simulation
  - Passing of some arguments from the command line

- Cons:
  - Some simple things needed to be done in the source code and then recompile. For example choosing a phantom
  - If for whatever reason the user needed to redo the calibration, the tools were not readily available
  - Calibration wouldn‘t use the same tools already developed for the experimental scans
  - By skipping the whole calibration process, the code was very useful for producing an image quickly, but didn‘t allow for a deeper understanding of all the subtleties of the scanner

Simulation of raw data

- The decision was taken to have a code that simulates the real scanner response and the output data as realistically as possible

- What was already there:
  - The main part of the code that created the digitized output of the trackers (#FPGA, #chip, #strip)
  - A simple digitization of the energy detector output (#ADC)

- What needed to be still done:
  - Make the code more realistic wherever possible
  - Interface the code with the existing preprocessing software (wasn‘t really done)
  - Add new phantoms
  - Test it, clean it, debug it, etc

- What could be added:
  - Many things, among others an easy way to run the code via macro input files

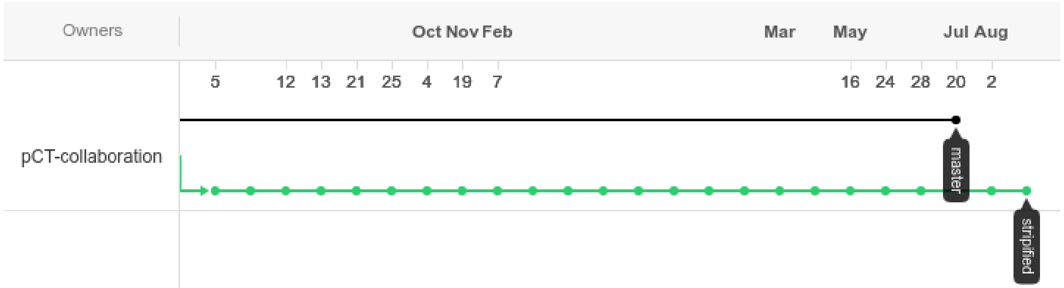
Current status of code

- The reconstruction data output version is not lost:
  - ➢ The code is in the master branch of the github repository
  - ➢ To make finding this code simple in the future, it is tagged as version v1.1
  - ➢ Due to limited resources, I don‘t do any further development in that code. Only bugfixes
  - ➢ Eventually the more realistic version of the code will be merged into the master branch as v2.X. The reconstruction data output will be always reachable as v1.X. It might be moved to a separate branch

- The raw data output version:
  - ➢ It is currently contained in the branch called stripified and frequently updated
  - ➢ It receives constant development and maintenance from the LMU team and occasionally from other users
  - ➢ It is in development status: not clean, bugs are not to be ruled out, changes often

Raw data output code

- As mentioned, currently in the stripified branch

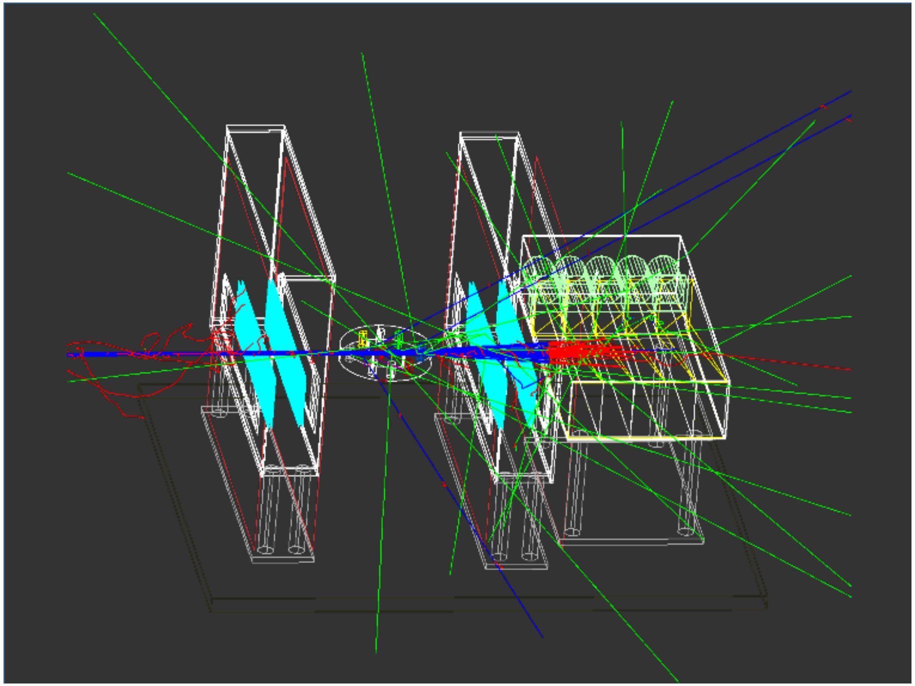


- Output data:
  - ➢ For every proton: #FPGA, #chip, #strip, #ADC

- Realistic data:
  - ➢ Our scanner doesn‘t have a homogeneous T-V response. This is corrected in the preprocessing. The Geant4 code simulates a realistic T-V response
  - ➢ The #ADC for every proton contains detector resolution and a pedestal
  - ➢ Changed polysterene RSP to match the one of the real phantom
  - ➢ A few days ago it was found that small differences between simulation and experiment originate from the light production saturation. Although we don‘t simulate optical photons, Birk‘s effect is now parameterized in the simulation

Raw data output code

- The geometry of the scanner hasn’t changed (detailed):



Raw data output code

- Phantoms:
  - All the existing ones: CTP404, Edge, Catphan, Water, …
  - Refined Water phantom to add walls
  - Wedge phantom and bricks for the calibration
  - To be added by September: read in of the pediatric head phantom in DICOM form

- Beams:
  - Simulation of several simple beams as well as reading of phase space data

- Messenger (input file functionality):
  - The code was modified in order to be customizable via input macro files
  - For most of the things a user might need to do, no writing of code/recompilation is needed
  - A new phantom will have to implemented in C++, but this can be done upon request

Raw data output code

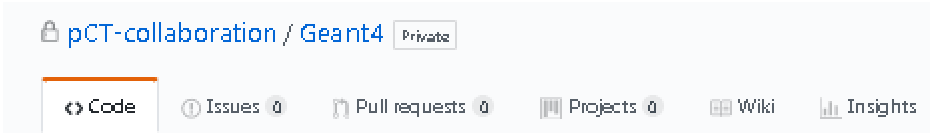
- Simulation workflow (same as for experimental data):
  - Simulate calibration runs: „empty“ run, wedge + 0, 1, 2, 3, 4 bricks (binary data)
  - Simulate N projections (angles) of whatever phantom (imaging scan - binary data)
  - Process with our preprocessing software the calibration runs

    -> get Wcalib.txt and TVcorr.txt
  - As the simulated setup is „static“, there is no need to resimulate and reprocess the calibration for every imaging scan simulation (might add to the repository the latest Wcalib and Tvcorr files)
  - Preprocess the imaging scan using the Wcalib.txt and Tvcorr.txt

    -> get your data for reconstruction (binary data with T,V,U and WEPL in mm)
  - Reconstruct image

Raw data output code

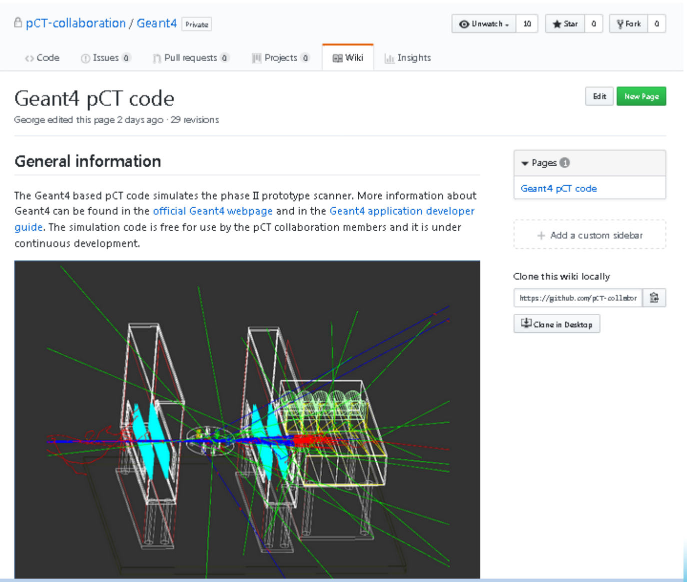
- Where can I find step-by-step instructions?
  - Last week we started a wiki page in github
  - Incomplete right now
    -> will be more useful in the coming weeks
  - Intention to start a forum webpage



Raw data output code

- Where can I find step-by-step instructions?
  - ➢ Last week we started a wiki page in github
  - ➢ Incomplete right now

    -> will be more useful in the coming weeks
  - ➢ Intention to start a forum webpage



pCT-collaboration / Geant4 Private

Unwatch 10 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights

Geant4 pCT code

George edited this page 2 days ago · 29 revisions

Edit New Page

General information

The Geant4 based pCT code simulates the phase II prototype scanner. More information about Geant4 can be found in the official Geant4 webpage and in the Geant4 application developer guide. The simulation code is free for use by the pCT collaboration members and it is under continuous development.

Pages

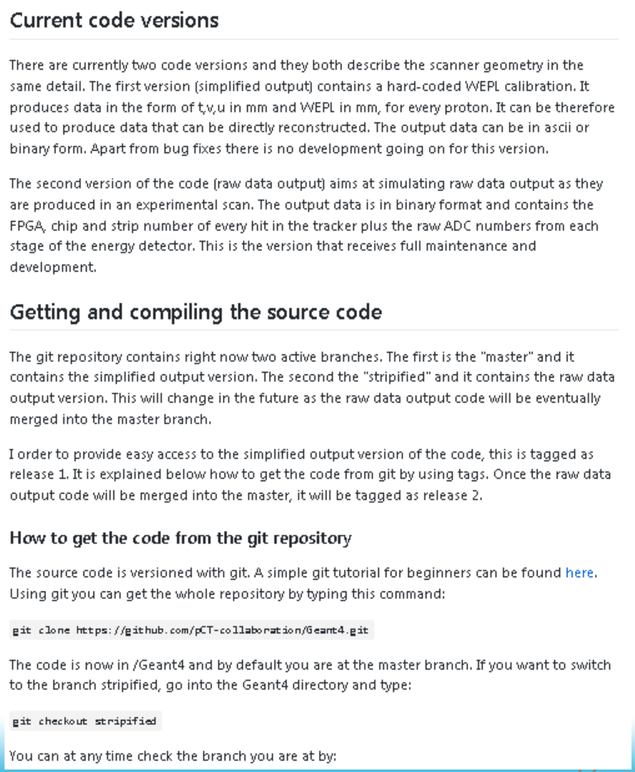
Geant4 pCT code

Add a custom sidebar

Clone this wiki locally

https://github.com/pCT-collabor

Clone in Desktop



Current code versions

There are currently two code versions and they both describe the scanner geometry in the same detail. The first version (simplified output) contains a hard-coded WEPL calibration. It produces data in the form of t,v,u in mm and WEPL in mm, for every proton. It can be therefore used to produce data that can be directly reconstructed. The output data can be in ascii or binary form. Apart from bug fixes there is no development going on for this version.

The second version of the code (raw data output) aims at simulating raw data output as they are produced in an experimental scan. The output data is in binary format and contains the FPGA, chip and strip number of every hit in the tracker plus the raw ADC numbers from each stage of the energy detector. This is the version that receives full maintenance and development.

Getting and compiling the source code

The git repository contains right now two active branches. The first is the "master" and it contains the simplified output version. The second the "stripified" and it contains the raw data output version. This will change in the future as the raw data output code will be eventually merged into the master branch.

I order to provide easy access to the simplified output version of the code, this is tagged as release 1. It is explained below how to get the code from git by using tags. Once the raw data output code will be merged into the master, it will be tagged as release 2.

How to get the code from the git repository

The source code is versioned with git. A simple git tutorial for beginners can be found here. Using git you can get the whole repository by typing this command:

```
git clone https://github.com/pCT-collaboration/Geant4.git
```

The code is now in /Geant4 and by default you are at the master branch. If you want to switch to the branch stripified, go into the Geant4 directory and type:

```
git checkout stripified
```

You can at any time check the branch you are at by:

Raw data output code

- How to run the code:
  - To run the code simply type: ./pCT_phaseII run.mac
  - pCT_phaseII is the executable created when compiling the code
  - run.mac (it can have any name) is the input macro file
  - An example run.mac exists in github

Raw data output code

- How to run the code:
  - To run the code simply type: ./pCT_phaseII run.mac
  - pCT_phaseII is the executable created when compiling the code
  - run.mac (it can have any name) is the input macro file
  - An example run.mac exists in github

- A couple of example imput macro commands:
  - Choose phantom

```
#Set phantom : AcrylicPhantom,WaterPhantom,WedgePhantom,HermanTarget,Catphan,CTP404,EdgePhantom or NothingPhantom
/pctPhase2/det/setPhantom WaterPhantom
/pctPhase2/det/update
```

Raw data output code

- How to run the code:
  - To run the code simply type: ./pCT_phaseII run.mac
  - pCT_phaseII is the executable created when compiling the code
  - run.mac (it can have any name) is the input macro file
  - An example run.mac exists in github

- A couple of example imput macro commands:
  - Set rotation angle

```
#Set phantom rotation angle (degrees - no need for additional unit definition). Not for calibration phantom
/pctPhase2/det/setPhantomRot 0.0
```

Raw data output code

- How to run the code:
  - To run the code simply type: ./pCT_phaseII run.mac
  - pCT_phaseII is the executable created when compiling the code
  - run.mac (it can have any name) is the input macro file
  - An example run.mac exists in github

- A couple of example imput macro commands:
  - Set calibration brick thickness

```
#Set brick thickness for calibration phantom and add length unit
#/pctPhase2/det/calBrickThickness 50.8 mm
```

Raw data output code

- How to run the code:
  - To run the code simply type: ./pCT_phaseII run.mac
  - pCT_phaseII is the executable created when compiling the code
  - run.mac (it can have any name) is the input macro file
  - An example run.mac exists in github

- A couple of example imput macro commands:
  - Choose where to write your results

```
# Set results output path/file type
/pctPhase2/results/setOutputPath ../output/Test
```

Raw data output code

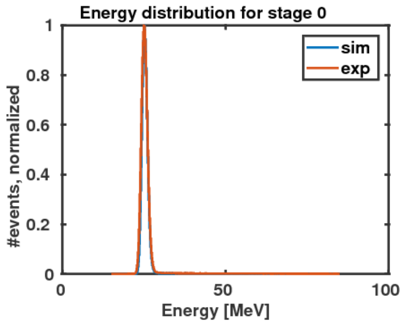
- How to run the code:
  - To run the code simply type: ./pCT_phaseII run.mac
  - pCT_phaseII is the executable created when compiling the code
  - run.mac (it can have any name) is the input macro file
  - An example run.mac exists in github

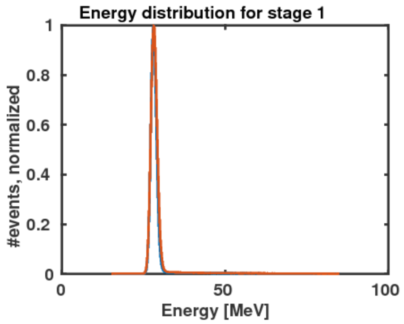
- A couple of example imput macro commands:
  - Choose beam type, location and energy

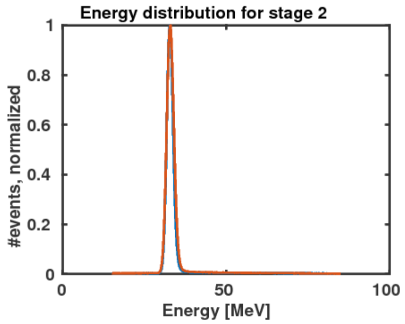
```
#Rectangle beam configuration
#/pctPhase2/gun/beamtype  rectangle
#/pctPhase2/gun/setBeamXmin  -200 mm
#/pctPhase2/gun/setBeamXmax   200 mm
#/pctPhase2/gun/setBeamYmin  -50 mm
#/pctPhase2/gun/setBeamYmax   50 mm
#
/gun/particle proton
/gun/energy 200 MeV
```

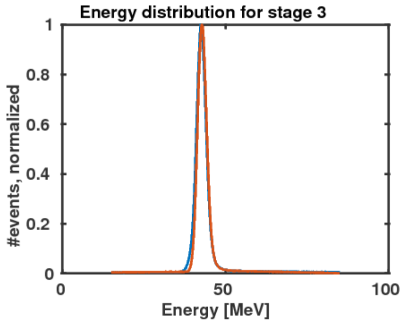
Raw data output code

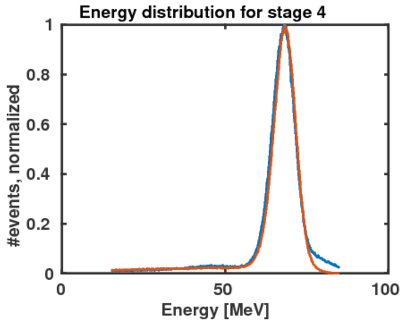
- A few validation results:

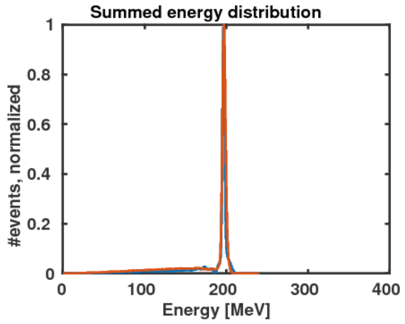








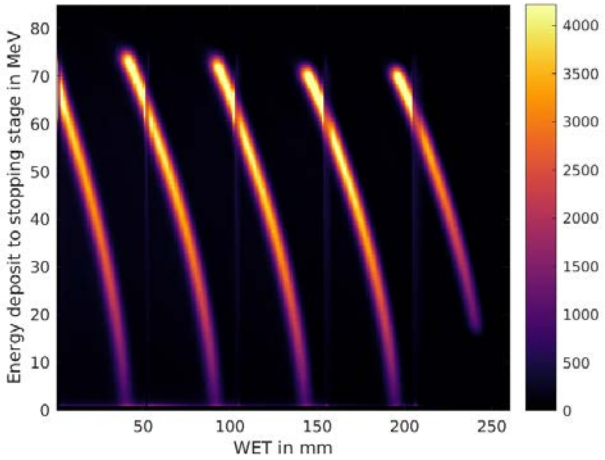




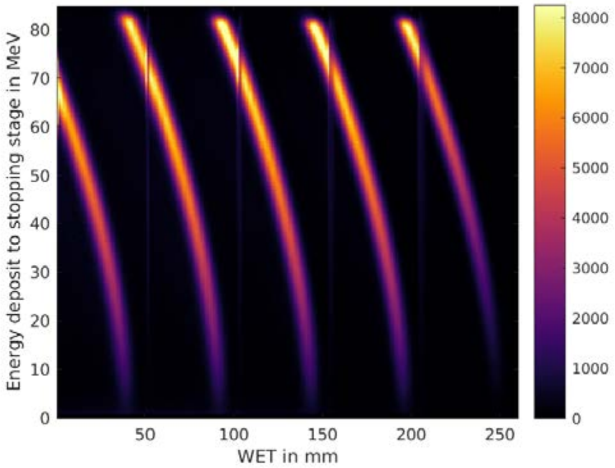
Raw data output code

- A few validation results:

Experiment



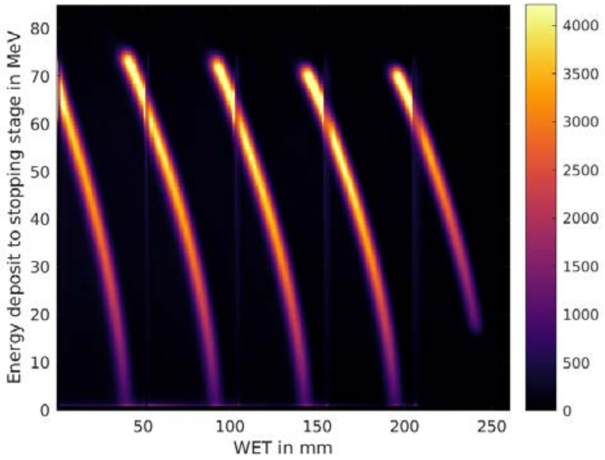
Simulation wo Birk‘s effect



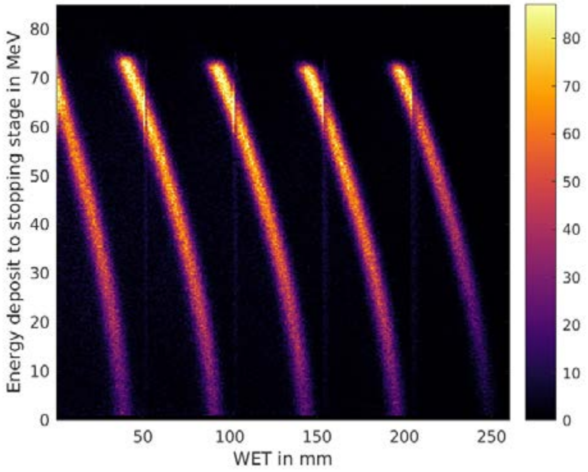
Raw data output code

- A few validation results:

Experiment

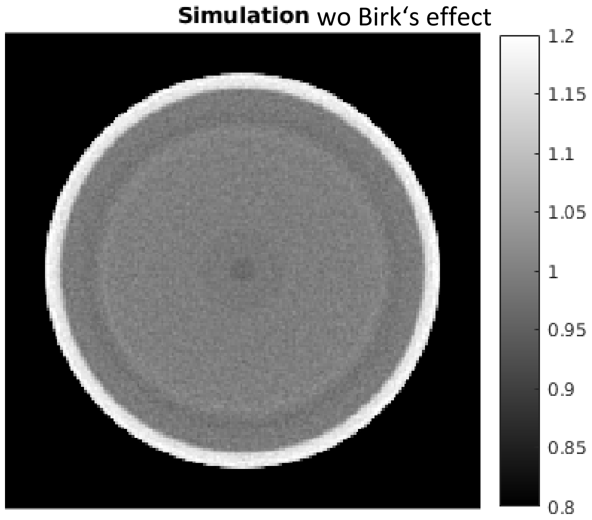


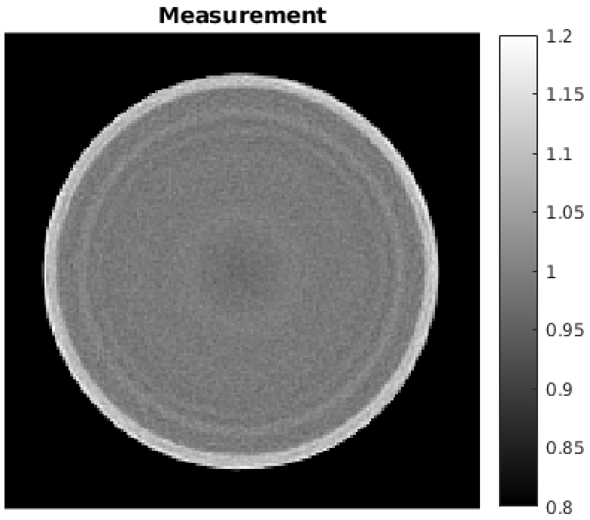
Simulation with Birk‘s effect



Raw data output code

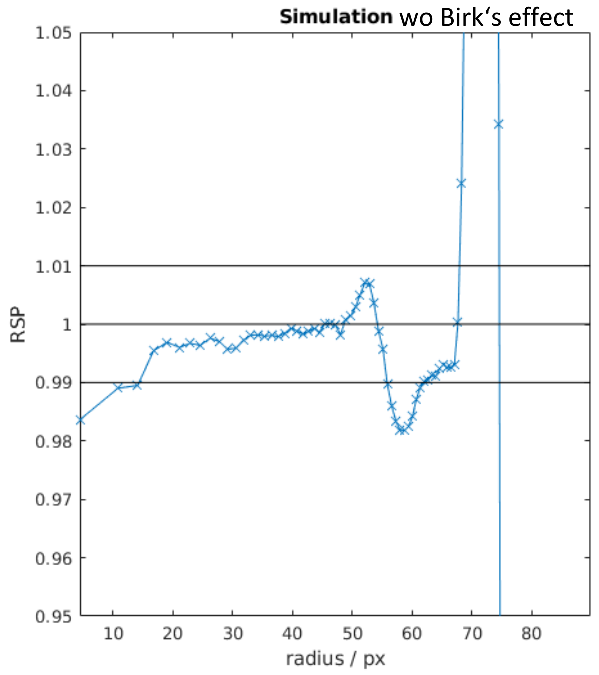
- A few validation results:

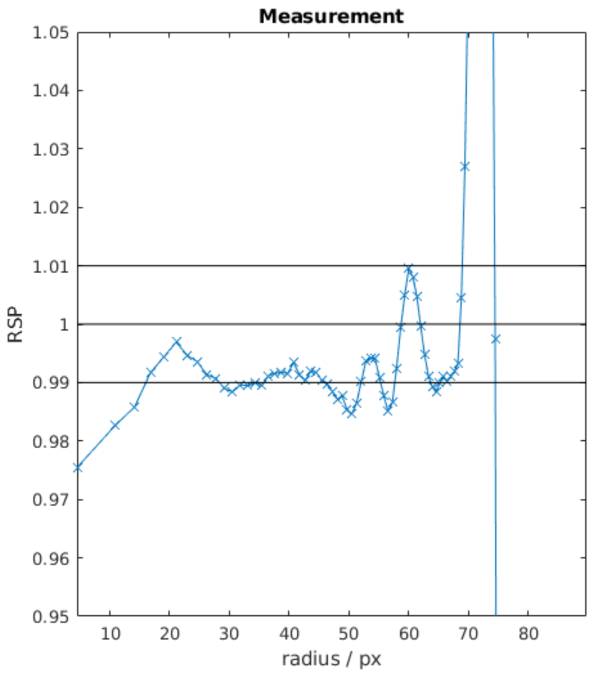


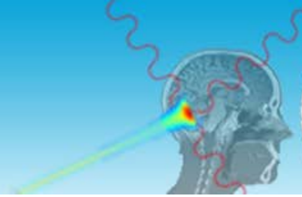


Raw data output code

- A few validation results:







Many thanks to:

- **Vladimir Bashkirov and Reinhard Schulte** for

hosting me multiple times in Loma Linda and discussing the subtle detector effects and parameterizations

- **Robert Johnson** for

investing in Santa Cruz last year and since then time to interface the Geant4 code with the preprocessing software

- **Valentina Giacometti and Pierluigi Piersimoni** for

sharing all their existing Geant4 code

- **Stuart Rowland** for

triggering the documentation of the code with his questions