Istituto Nazionale di Fisica Nucleare

GRAN SASSO SCIENCE INSTITUTE

digitizationpp: towards a faster digitization

S. Piacentini, G. Dho

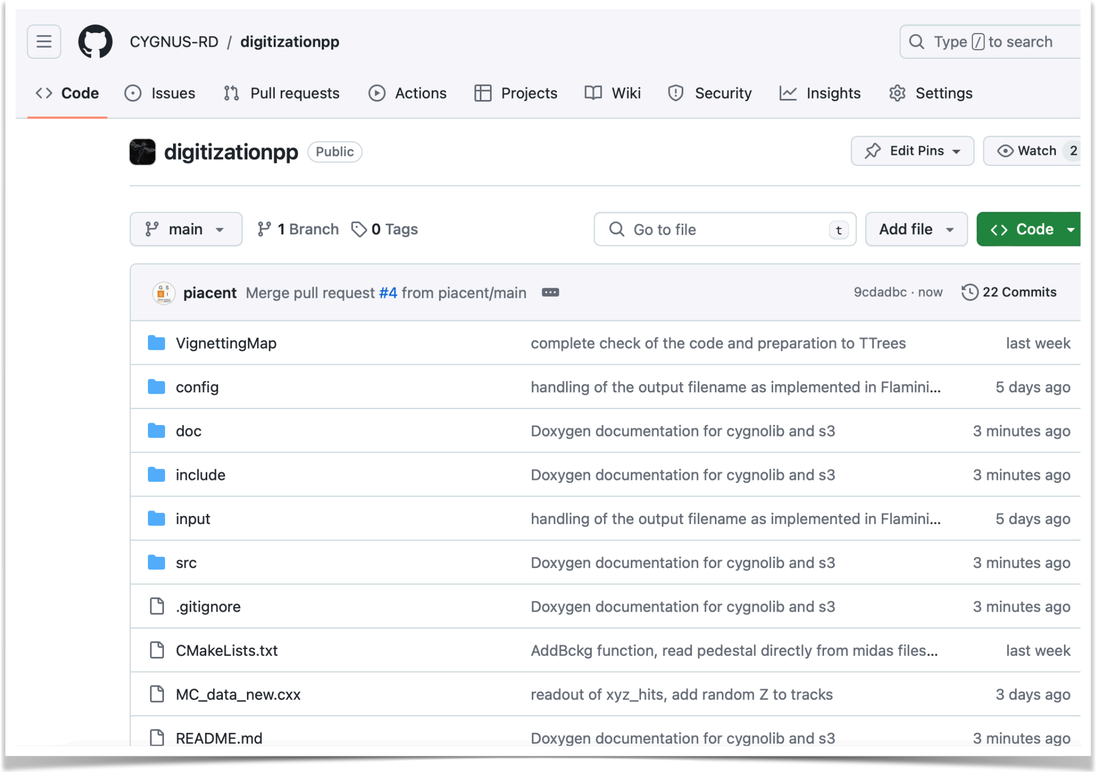
Simulation Meeting - 20/05/2024

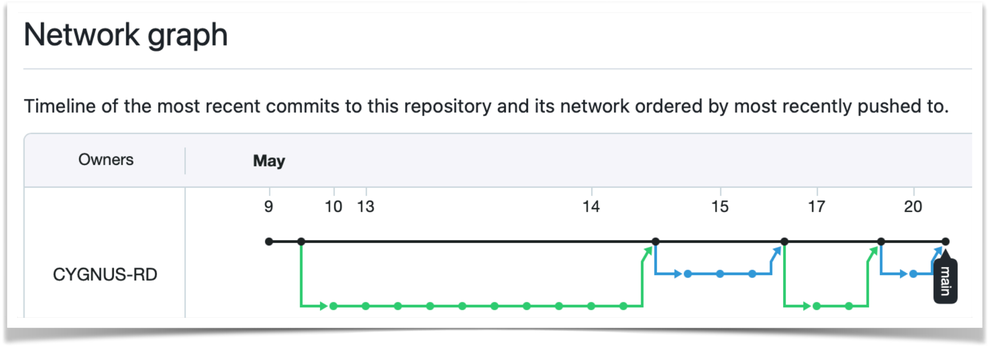
Motivations

- Digitization is the **slower step** of our MC simulation:
  - Relatively low MC statistics simulated so far
  - Showstopper to many groups working on the analysis or the simulation (e.g. data - MC comparison, training of ML data selection algorithms, etc.)
- Digitization is **coded in python** (fast and easy to code, not the best choice if the goal is the performance)
- Idea: **convert the algorithm to C++**. Why C++?
  - It's compiled → generally faster
  - We have more experience with it rather than other compiled codes (e.g. Fortran).

The new repository

- We created a new repository: CYGNUS-RD/digitizationpp
- It's not yet completed, we will release the first version as soon as we implement all the steps of the digitization.
- Work flow (if you want to join the effort):
  - Create your own fork
  - Test and commit changes to your fork
  - If something is tested and complete enough, submit a pull request





Status

- Dependencies:
  - ROOT [**C++ standard > 17**]
  - ROOTANA, for the handling of MIDAS files
  - OPENCV, for possible applications and operations with pictures

digitizationpp

** work in progress **

Digitization code in C++

Dependencies

- ROOT [compiled with the C++17 standard]
- ROOTANA [https://bitbucket.org/tmidas/rootana/src/master/]
- OPENCV [https://docs.opencv.org/4.x/d7/d9f/tutorial_linux_install.html]

Before compiling, set the variables ROOTANASYS and OPENCVSYS in your environment:

```
export ROOTANASYS="/path/to/rootana/installation/" export OPENCVSYS="/path/to/opencv/installation/"
```

- I created a CMakeLists to compile the code
  - it assumes ROOTSYS, ROOTANASYS, and OPENCVSYS environmental variables are set in your environment

Status

- Dependencies:
  - ROOT [**C++ standard > 17**]
  - ROOTANA, for the handling of MIDAS files
  - OPENCV, for possible applications and operations with pictures

digitizationpp

** work in progress **

Digitization code in C++

Dependencies

- ROOT [compiled with the C++17 standard]
- ROOTANA [https://bitbucket.org/tmidas/rootana/src/master/]
- OPENCV [https://docs.opencv.org/4.x/d7/d9f/tutorial_linux_install.html]

Before compiling, set the variables ROOTANASYS and OPENCVSYS in your environment:

```
export ROOTANASYS="/path/to/rootana/installation/" export OPENCVSYS="/path/to/opencv/installation/"
```

- I created a **CMakeLists** to compile the code with **cmake**
  - it assumes ROOTSYS, ROOTANASYS, and OPENCVSYS environmental variables are set in your environment

Status

- I created a **CMakeLists** to compile the code with **cmake**
  - it assumes ROOTSYS, ROOTANASYS, and OPENCVSYS environmental variables are set in your environment

Installation

```
git clone https://github.com/CYGNUS-RD/digitizationpp.git
```

```
cd digitizationpp
```

```
export CXX="/path/to/your/c++17/compiler"
```

```
mkdir build-dir && cd build-dir
```

```
cmake ..
```

```
cmake --build .
```

Suggested usage

Put all the MC `.root` files you want to digitize in the `input/` folder, then:

```
cd build-dir
```

```
./digitizationpp ../config/ConfigFile_new.txt
```

The output file will be saved in the `OutDir/` folder.

Status: code structure

- VignettingMap
- config
- doc
- include
- input
- src
- .gitignore
- CMakeLists.txt
- MC_data_new.cxx
- README.md
- Z_A_isotopes.csv

config files

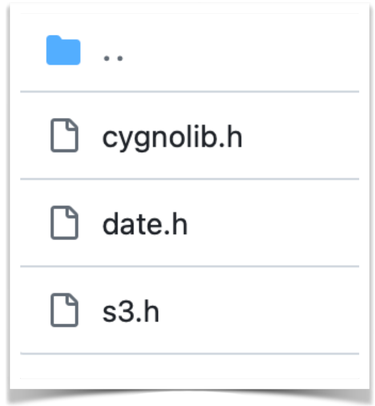
doxygen files

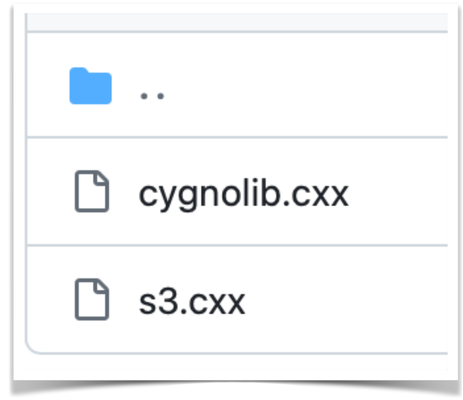
header library .h files

input Geant4/SRIM files

source library .cxx files

main()





Proto - doxygen documentation

- So far only available for the two libraries cygnolib and s3

Generate documentation inside the `doc/html` folder:

```
doxygen doc/doxygen.cfg
```

Documentation should be then available at `doc/html/index.html`.

```
/**
 * @brief Cygnolib classes, functions, and implementations
 *
 *
 */
namespace cygnolib {


    /**
     * @class Picture
     * @brief A class for providing tools to handle the pictures collected by the CY(
     * @author CYGNO Collaboration
     *
     * @details This class can be used to create an object that contains an image col
     * camera
     *
     */
    class Picture {
    public:

        /**
         * @brief Constructor.
         * @details This constructor passes the dimensions of the image in pixels.
         *
         * @param[in] heigth Height of the image in pixel. Default value is 2304.
         * @param[in] width Width of the image in pixel. Default value is 2304.
         *
         */
        Picture(unsigned int height = 2304, unsigned int width = 2304);
```

CYGNO digitizationpp 0.1.0

C++ library to digitize CYGNO MC events

Main Page | Namespaces | Classes | Files

cygnolib Namespace Reference

Cygnolib classes, functions, and implementations. More...

Classes

class	**DGHeader** A class for providing tools to handle the Digitizer header collected by the CYGNO DA(
class	**Picture** A class for providing tools to handle the pictures collected by the CYGNO cameras. M
class	**PMTData** A class for providing tools to handle the PMT data collected by the CYGNO DAQ. Mo

Functions

TMReaderInterface *	**OpenMidasFile** (std::string filename) This function opens the MIDAS file.
bool	**FindODBDumpBOR** (TMidasEvent &event, bool verbose=false) This function finds a MIDAS BOR (Begin Of Run) ODB dump.
MVOdb *	**GetODBDumpBOR** (TMidasEvent &event, MVOdbError *odberror=NU This function finds a MIDAS BOR (Begin Of Run) ODB dump.
bool	**FindBankByName** (TMidasEvent &event, std::string bname, bool vert This function finds a MIDAS bank by its name.
void	**InitializePMTReadout** (std::string filename, bool *DRS4correction, std &table_nsample)

https://github.com/flaminiadigiamba/digitization/tree/master

Status

- At least in its first release the code will be a **“clone” of the original Python Flaminia’s version** (same config files, same way of handling input/output files, etc.)

- Features added so far:
  - Parsing of config file
  - Complete ROOT interface
  - Parsing of the .root + .py SRIM input files
  - Creation of the .root output file
  - Addition of pedestal as a random picture from a MIDAS file:
    - Using the recopp library I already developed, the code checks if the required pedestal .mid file is in the chosen tmp folder, if not downloads it from the cloud, it opens it and it imports a random picture.
  - Loading vignetting map
  - Starting now to implement the “physics” of the code (change in coordinate between SRIM and Geant, etc.)

Open questions and issues for the simulation group

1. **I need input files** to test the code by way of example:
    1. a ER Geant4 file
    2. a NR Geant4 file
    3. a NR SRIM file

2. Inputs of the code:
    1. Input files folder → the code digitize every file contained in a user defined folder: **is this ok?**
    2. Output files folder → the code writes the output files in a user defined folder (creates it if it does not exist) + a subfolder named as the input filename: **is this ok?**

Example:

Input folder = input/

files:

input/file1.root

input/file2.root

ALL of them are digitized

Output folder = output/

files:

output/file1/histogram_00001.root

output/file2/histogram_00001.root

Conclusions

- A lot of work to do to implement the “physics”
- A huge space for optimization and for a cleaning of the code:
  - Now almost everything is in a very long main, dedicated library? Dedicated class?
  - Starting to add a doxygen-friendly description of the functions (so far done only of the cygnolib and the s3 libraries) → **automatic documentation!**
- We slowly but constantly proceed since the start of this activity last week. Any volunteer is welcome!