Restructuring AutoBub C++ code into Python (pyAutoBub)

Babar Ali

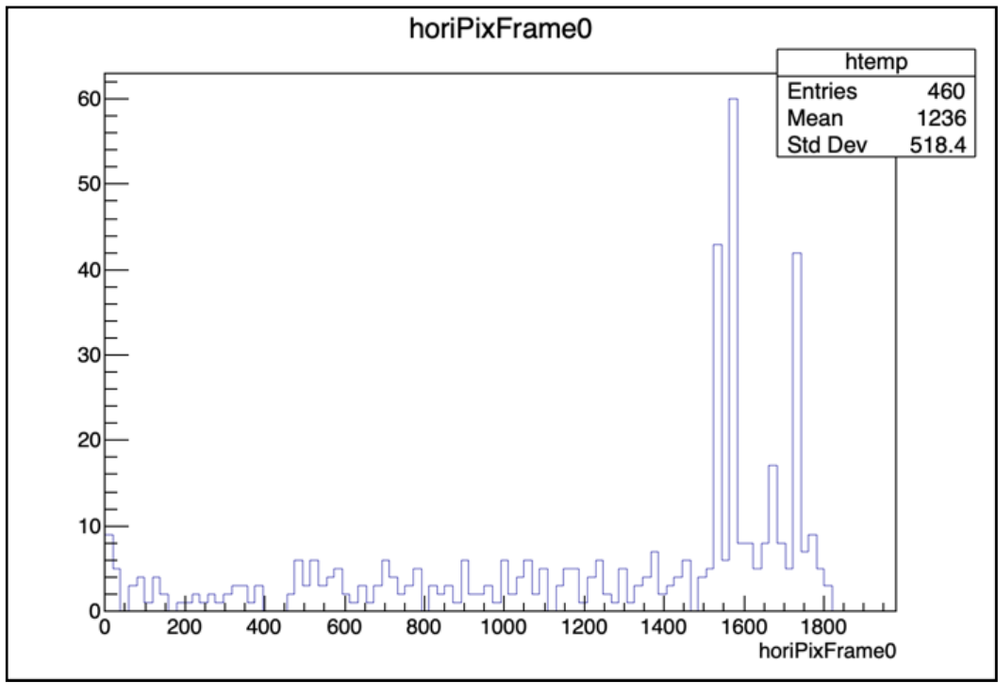
PICO Collaboration Meeting
Prague August 2024

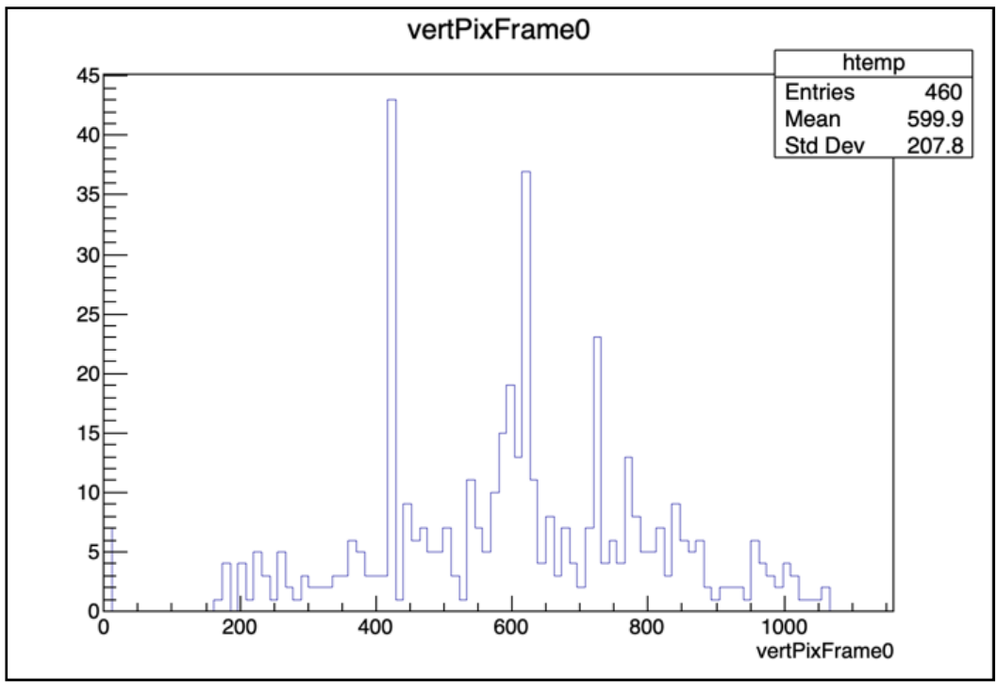
AutoBub

- AutoBub was originally developed by Pitam Mitra (University of Alberta).
- Many people have made contributions to the code and/or serve to maintain it.
- Over the years a lot of changes were made.
- A lot of exceptions were introduced to run the code for specific cameras, runs and chambers.

AutoBub running modes

Column based data file





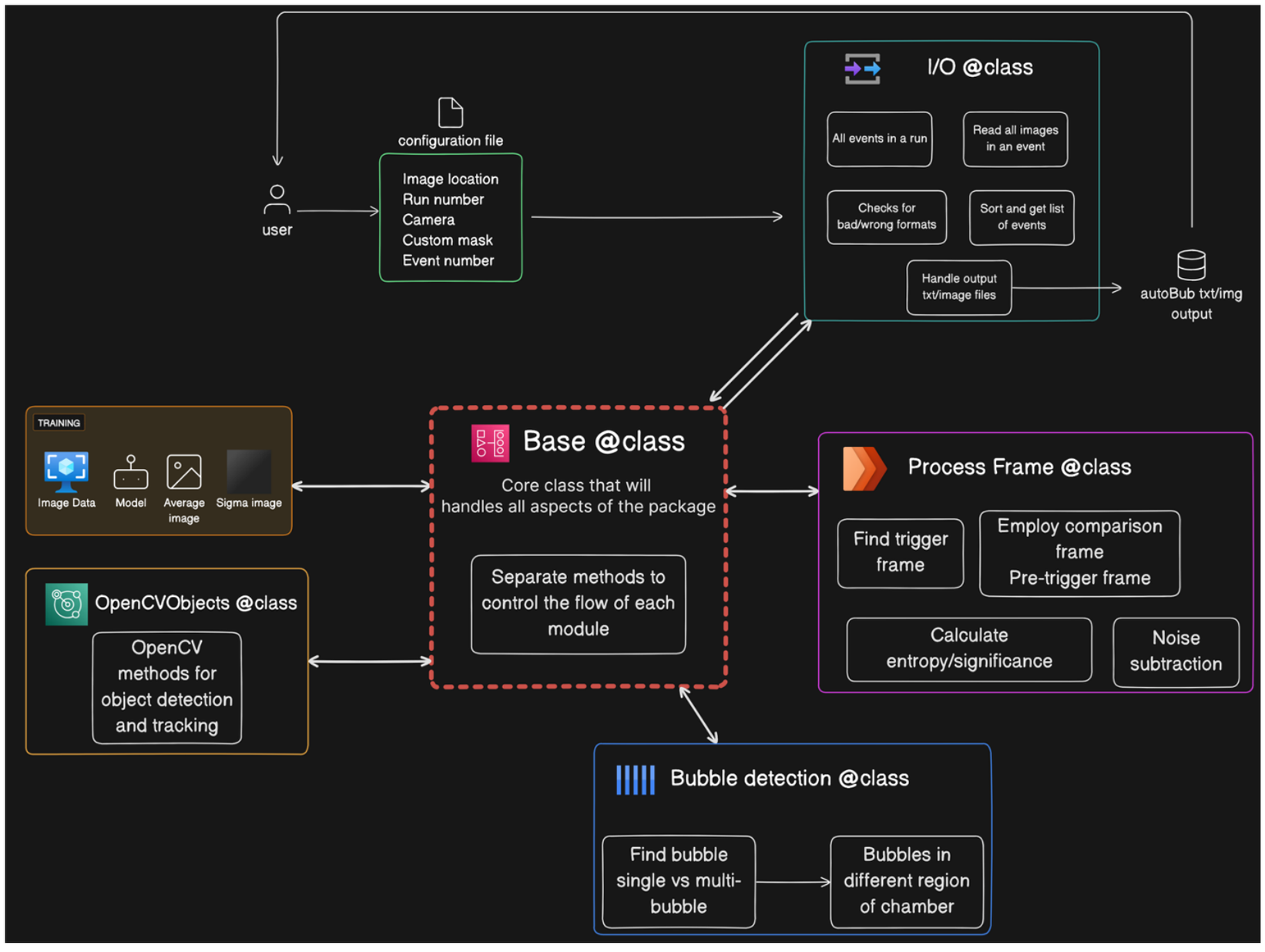
The location, in pixel space, where the bubble is found in the frame0 frame of the image

Goals

- Restructure C++ code into python.
- Code must be user friendly, robust and scalable
- It will be implemented using object oriented programming (OOP).
- Compare the output results.
- Integrate the code into the analysis processing chain.
- Sub-project of PICOCode.

pyAutoBub development

- Code will be divided into separate, interchangeable modules or components.
- Each module will handle a specific aspect of the functionality and can be developed, tested, and maintained independently.
- This approach ensures improved organization, reusability, ease of maintenance, and scalability.



pyAutoBub structure

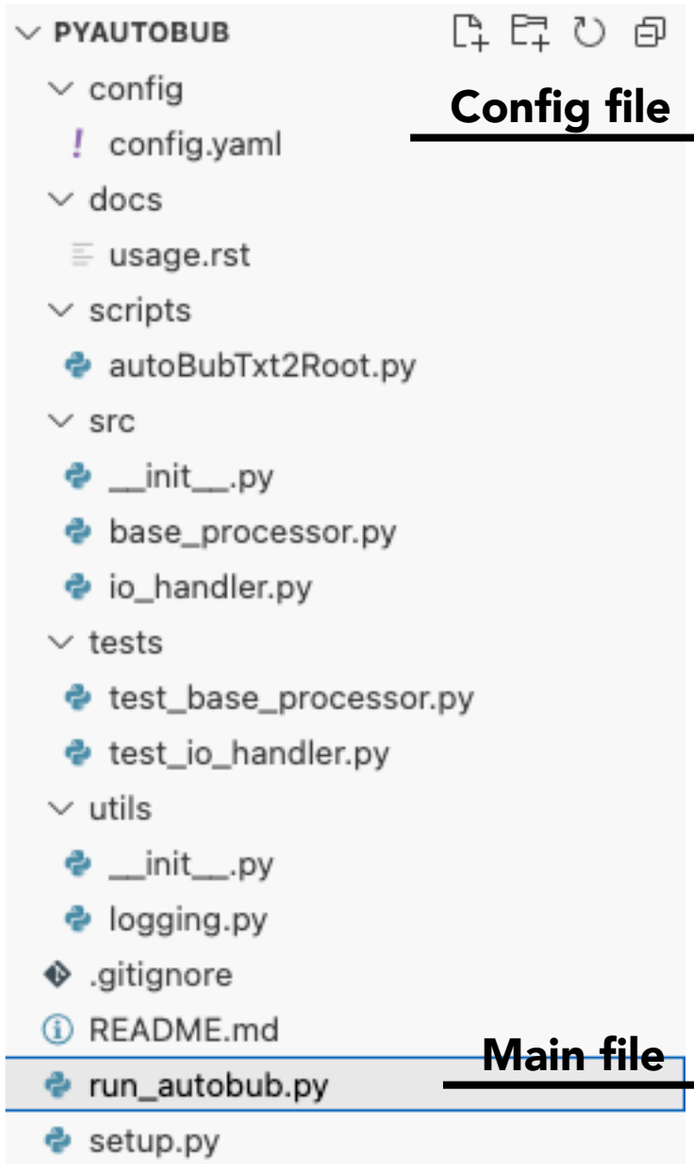
Documentation ←

Useful scripts ←

All source code {

Tests for source code {

Helper functions ←



Config file →

config > config.yaml

```yaml
runNumber: 20240131
imageFormat: png
inputFileFormat: zip
frameOffset: 30
eventNumber: 10
camMask_dir: /path/to/cam_masks
data_dir: /path/to/data
out_dir: /path/to/output
```

Main file →

run_autobub.py

```python
from src import BaseProcessor as Base
from utils import getLog

logger = getLog(__name__)

def execute():

    logger.info("Starting the pyAutoBub.....")

    processor = Base('config.yaml')
    processor.process()

if __name__ == "__main__":
    execute()
```

README.md

pyAutoBub package

This is pyAutoBub.

pyAutoBub builds upon a legacy C++ implementation. The goal was to translate the existing C++ code into Python, enhancing its performance and robustness in the process. pyAutoBub is written from scratch and aims to identify bubbles in images from a bubble chamber and return information about the bubble candidates, including position and movement information.

Acknowledgments

We acknowledge the work done in the original C++ implementation by Pitam Mitra (University of Alberta, PhD'18).

Some useful documentation from Pitam's work:

- Pitam's PhD Thesis: https://www.snolab.ca/pico-docdb/cgi/ShowDocument?docid=3252
- AutoBub Algorithm: https://www.snolab.ca/pico-docdb/cgi/ShowDocument?docid=2031
- AutoBub Image Analysis: https://www.snolab.ca/pico-docdb/cgi/ShowDocument?docid=1578

TO-DO list

- Set up the project structure
- Read the config file data
- Implement the training module
- Option to run on a single event, all events or a couple of events
- Code to read the camera masks
- Implement image processing, bubble detection, and other modules
- Integrate the OpenCV module
- Option to run the code for bulk events or wall events
- Unit tests for individual modules
- Implement error handling
- Optimize the performance of critical modules, heavily relying on NumPy and pandas
- Option to run 3D reconstruction
- Incorporate code for multi-bubble detection (Minya's code)
- Set up CI pipelines for automated testing
- Set up the Docker image
- Document the code with Sphinx (using the NumPy coding style guide)

Questions?