



LIETUVOS MOKSLO TARYBOS FINANSUOTO IMPRESS-U PROGRAMOS PROJEKTO

**„IRIS-HEP: MOKSLO PROGRAMINĖS ĮRANGOS PLĖTIMAS DALELIŲ FIZIKOS REIKMĖMS
UKRAINAI, LENKIJAI IR LIETUVA“**

(Nr. S-IMPRESSU-24-4, reg. data 2024-06-26 d.)

2024 M. STUDENTŲ STAŽUOČIŲ PATVIRTINTI PROJEKTAI

Projekto vykdytojas: Vilniaus Universitetas

Projekto vadovas: dr. Andrius Juodagalvis

2026 Vilnius

Improving CICADA trigger algorithm

Applicant: Liudas Ablašinskas

Mentor: Lino Gerlach

Project Duration: 3 weeks

Proposed start date: 07 August 2024

Project Context and Description

In high-energy physics experiments, such as those conducted at the Large Hadron Collider (LHC), it is often impossible to record every particle collision due to the high rate of data. For this reason, dedicated low-latency algorithms (*triggers*) are deployed that decide in real-time if an event should be stored for further processing, reducing the initial event rate of 40 MHz down to roughly 1 kHz.

These triggers typically look for signatures in the detector that physicists deem interesting, such as highly energetic particles, or specific decay topologies. However, hand-crafted algorithms like these bear the risk of overlooking unknown types of signatures like they might occur in Beyond the Standard Model (BSM) scenarios.

The Calorimeter Image Convolutional Anomaly Detection Algorithm (CICADA) project [4] aims to provide a model-independent trigger based on anomaly detection. It uses low level CMS experiment's trigger calorimeter information as the input to convolutional auto-encoder to find anomalies produced during LHC proton-proton collisions. Quantization Aware Training and Knowledge Distillation are used to compress the model for sub-500 ns inference on Field-Programmable Gate Arrays.

Within the cylindrical calorimeters [1], trigger towers are specific regions where the energy deposits are summed and monitored. Each tower corresponds to a small section of the calorimeter, typically covering a range in pseudorapidity (η) and azimuthal angle (ϕ) [2]. With its initial convolutional layer, CICADA treats the (η, ϕ) -indexed energy deposits as ordered data, exploiting the proximity of neighboring pixels like in images from a digital camera. However, as the data actually stems from a cylindrical detector, there are relations between neighboring pixels that CICADA does not yet exploit.

Solution

In this project I will be implementing circular padding on convolutional neural networks and investigating its potential effects on the performance of CICADA. The models will be trained on simplified simulated dataset to gauge the potential impact of such padding techniques for the CICADA use case.

Project Timeline

Time frame Objectives

Time frame	Objectives
Week 1	• Study auto-encoders and knowledge distillation, the basic principles of the Calorimeter Image Convolutional Anomaly Detection Algorithm (CICADA) • Select or generate a simplified dataset with the same cyclic nature as the calorimetric inputs for CICADA
Week 2	• Train a simple auto-encoder to replicate the input data and / or train a supervised student model on soft targets [3]. • Find the minimal model size that performs well • Implement circular padding for the model(s) above
Week 3	• Compare the models' performance with and without circular padding • Analyse and present the findings

References

1. Energy of Electrons and Photons (ECAL) | CMS Experiment, [<https://cms.cern/detector/measuring-energy/energy-electrons-and-photons-ecal>].
2. The CMS electromagnetic calorimeter project - CERN Document Server, [<https://cds.cern.ch/record/349375>].
3. Knowledge Distillation for Anomaly Detection, Adrian Pol et. Al, [<https://arxiv.org/pdf/2310.06047>].
4. The CICADA project (web), <https://cicada.web.cern.ch/>

Develop of Clad Tutorials for CMS/HEP

Applicant: Austėja Jurgaitytė

Mentor: David Lange

Project Duration: 5 weeks

Project Context and Description

Particle tracking is an important part of the processing and analysis of data received from particle detectors, such as the Compact Muon Solenoid (CMS). Tracking is the step that determines the momentum of charged particles escaping from the collision point. It identifies individual particles by reconstructing their trajectories from points where charged particle “hits” were measured by the detector and interpreting them. [1] Due to the Lorentz force, charged particles move in a helical motion when affected by the magnetic field (neglecting other effects due to material interactions, etc). This means we can figure out a specific particle trajectory through the detector by fitting a helix function to data points in such a way that the distance from the data points and the helix would be minimized. In mathematical terms, we need to find optimal helix parameters by minimizing a loss function composed of the least squared distances, thus giving the best estimation of these parameters.

For this purpose we can use Clad - an automatic differentiation tool for C++, [2] to efficiently minimize the loss function. Automatic differentiation is a set of techniques used to evaluate derivatives of functions programmatically. It calculates function derivatives exactly (up to numerical precision limits) and is not confined to closed-form expressions. [3] Automatic differentiation makes use of the chain rule and intermediate variables, calculated using primitive arithmetic operations and elementary functions, present in every computer calculation. [4] There are two main modes of automatic differentiation: forward and reverse accumulations. Forward accumulation (Clad’s Forward mode) traverses the chain rule from inside to outside, evaluating the intermediate functions and their derivatives with respect to one independent variable and returns a single numeric output, or in Clad’s case, a function [4] [2]. Reverse accumulation (Reverse mode) computes the gradient of a function during the reverse pass, after function evaluation, using the stored dependencies of the expression. [5]

Goals

In this project, we will focus on creating a Clad based demonstration of finding the best fit helix parameters given a set of data points. Given the short timescale of the project, the complexity of the resulting tutorial will depend on complexities encountered during the research. At the same time, during the project, demos and reproducers of various Clad functionalities will be built and tested. Another goal of this project will be to contribute Clad code fixing the missing functionalities that we find along the way, such as adding support for the atan2 function differentiation.

Project timeline

- Study the code of Clad and understand how to use it and its usage limitations. (week 1-2)
- Create a simplified helix class that Clad can differentiate successfully. (week 2)
- Create an objective function that takes a set of data points and the code that finds the helix that best fits the minimum distances to these points with Clad. (week 3-4)
- Find, and fix or report, missing functionalities and bugs. (week 4-5)

References

- [1] CERN CMS Experiment Website, Tracking
[<https://cmsexperiment.web.cern.ch/detector/identifying-tracks>]
- [2] Clad Github [<https://github.com/vgvassilev/clad>]
- [3] Compiler Research website, Automatic differentiation
[https://compiler-research.org/automatic_differentiation]
- [4] Wiki Website on Automatic Differentiation
[https://en.wikipedia.org/wiki/Automatic_differentiation]
- [5] MathWorks website, Automatic Differentiation Background
[<https://ch.mathworks.com/help/optim/ug/autodiff-background.html>]

Adding RNTuple to the Analysis Grand Challenge

Applicant: Giedrius Juškevičius

Mentor: David Lange

Project Duration: 8 weeks

Proposed start date: 29 July 2024

Project Context and Description

The High-Luminosity Large Hadron Collider (HL-LHC) project aims to crank up the performance of the LHC in order to increase the potential for discoveries after 2029 [1]. However, analysis workflows commonly used at the LHC experiments do not scale to the requirements of the HL-LHC [2]. To address this challenge, the IRIS-HEP software institute started the “Analysis Grand Challenge” (AGC) project. It includes a pipeline including a binned analysis, reinterpretation and end-to-end optimization of a physics analysis use cases. It also includes the development of the required cyber-infrastructure to execute them in order to demonstrate technologies envisioned for HL-LHC [3].

The LHC experiments use detectors to analyze the myriads of particles produced by collisions in the accelerator. The biggest of these experiments, CMS (Compact Muon Solenoid) and ATLAS, use general-purpose detectors to investigate the largest range of physics possible [4]. Due to differences in the sensors, separate data analysis workflows must be used in the AGC software. For this project, we will focus specifically on optimizing the CMS workflow [5]. At the time of this proposal, the most compact data tier format available in Open CMS data is NanoAOD (Nano Analysis Object Data) [6, 7], which will be our main focus for optimization. AOD data is derived from the RECO information to provide data for physics analysis in a convenient, compact format and is usable directly by physics analyses [8].

A NanoAOD file contains a main TTree named events. Since its inception, the ROOT project supports the columnar storage of arbitrary C++ types and collections through TTree. However, the expected increase in the amount of experiments data that needs to be processed and the fact that TTree was not designed to make optimized use of modern hardware and storage systems, called for a new, modernized re-engineering of TTree [9].

RNTuple is the new, still under development, backward-incompatible ROOT columnar I/O subsystem targeting high performance, reliability, and easy-to-use robust interfaces [9]. In its current state, RNTuple can read and write data with an API similar to the TTree one, including bulk reading. RNTuple’s supported type system is more limited than TTree’s, yet already powerful enough to represent, for instance, CMS NanoAODs, and ATLAS PHYS, PHYSLITE and full AOD files. The ATLAS and CMS frameworks have included experimental RNTuple models in their integration builds [10]. In this project, we will use these models to convert data from TTree format to RNTuple format, aiming to optimize data processing capabilities.

Solution

In this project, our focus will be on analyzing existing Python, ROOT, and C++ code within the AGC/CMS repositories. Our main goal is to transform TTree event data structures into RNTuple formats to enhance pipeline efficiency. By the project's conclusion, optimized pipeline components will be delivered to the AGC repository, complete with thorough documentation.

Goals

1. Study AGC code: understand it and learn how to run it;
2. Convert some NanoAOD input data to RNTuple;
3. Understand and implement what pieces of the AGC need to be changed;
4. Test and evaluate performance to check how faster the pipeline is with RNTuple;
5. Integrate code into AGC repository and write/update documentation.

Project Timeline

Time frame	Objectives
Week 1	• Study AGC code: understand it and learn how to run it.
Weeks 2-4	• Convert some NanoAOD input data to RNTuple. • Understand and implement what pieces of the AGC need to be changed.
Weeks 5-6	• Test and evaluate performance to check how faster the pipeline is with RNTuple.
Weeks 7-8	• Integrate code into AGC repository. • Write/update documentation.

References

- [1] CERN Website, “High-Luminosity LHC” [<https://www.home.cern/science/accelerators/high-luminosity-lhc>].
- [2] A. Held, O. Shadura, “The IRIS-HEP Analysis Grand Challenge” (2023), [<https://pos.sissa.it/414/235>].
- [3] IRIS-HEP Website, “The Analysis Grand Challenge”, [<https://iris-hep.org/projects/agc.html>].
- [4] CERN Website, “Experiments”, [<https://home.cern/science/experiments>].
- [5] IRIS-HEP Github Repository, “analysis-grand-challenge/.../ttbar_analysis_pipeline.ipynb”, [https://github.com/iris-hep/analysis-grand-challenge/blob/main/analyses/cms-open-data-ttbar/ttbar_analysis_pipeline.ipynb].
- [6] CERN TWiki Website, “The CMS NanoAOD data tier”, [<https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBookNanoAOD>].
- [7] M. Peruzzi, G. Petrucciani, A. Rizzi, “The NanoAOD event data format in CMS” (2020), [<https://iopscience.iop.org/article/10.1088/1742-6596/1525/1/012038>].
- [8] CERN TWiki Website, “Analysis Overview: an Introduction-> AOD” [<https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBookAnalysisOverviewIntroduction>].
- [9] J. Lopez-Gomez, J. Blomer, “RNTuple performance: Status and Outlook” (2023), [<https://iopscience.iop.org/article/10.1088/1742-6596/2438/1/012118/pdf>].
- [10] J. Blomer, P. Canal, F. de Geus, J. Hahnfeld, A. Naumann, J. Lopez-Gomez, G. Lazzari Miotto, V. E. Padulano, “ROOT’s RNTuple I/O Subsystem: The Path to Production” (2024), [https://www.epj-conferences.org/articles/epjconf/pdf/2024/05/epjconf_chep2024_06020.pdf].

Improving latency and scalability of the user runtime job log collecting and exposure in the REANA reproducible analysis platform

IRIS-HEP project proposal
Jelizaveta Lemeševa

REANA is “a reproducible analysis system allowing scientists to run containerized data analysis pipelines on remote computer clouds”¹. It allows researchers to express their computational analyses using declarative workflow languages (CWL, Snakemake, Yadage) that are then executed on a target compute backend. The researchers can easily execute and rerun the analysis with different input data, parameters or code.

Scientists interact with REANA via a command-line client or via the web interface. The client allows to send requests to connected REANA cluster for their execution. The REANA cluster consists of several micro-services that accept client requests and schedule analysis workflows and jobs for the execution on the supported compute backends (Kubernetes, HTCondor, Slurm). The REANA cluster schedules user workflows and jobs for execution by sending requests to compute backend’s API. The compute backend executes the job and notifies the REANA cluster about its completion. The REANA cluster then retrieves the logs of the jobs from the compute backend and stores them in a PostgreSQL database for persistency. The client retrieves these logs from the REANA cluster and displays them to the user.

In the current setup, users can access workflow and job logs only after the job execution has finished. When a workflow consists of many jobs, the researcher can therefore consult logs of already finished steps, but not of the step that is currently being executed.

The goal of this IRIS-HEP project proposal is to enhance the REANA job logging system with a possibility to capture logs of executing processes “live”, so that the user would have access to the running job logs even during job execution. Certain jobs can run for several hours so it would often be beneficial to peek into the execution process without much latency delays.

The technological solution to this problem could consist of introducing a sidecar container that would be deployed alongside the main job, reading the logs in real time and pushing them to the database. An alternative approach could be to deploy daemonsets that would listen to all user workload on a given cluster node and dispatch the results to a new log collector service, living outside of the database, for example in an OpenSearch. This may make the log collection more

¹ <https://docs.reana.io/>

scalable at the price of introducing a new infrastructure component and making necessary changes to the REANA application to expose thusly collected logs back to the clients in the traditional way. Further evaluation is needed to decide which approach to take.

In either case, it is desired that the logs are exposed to the user over the command-line client and the web interface with very low latency.

Objectives:

- Design a solution to efficiently capture live workflow logs and job logs whilst they are being executed.
- Analyse the pros and cons of the various technical approaches to capturing Kubernetes pod logs with low latency.
- Implement the chosen solution by amending the necessary REANA micro-service components.
- Update REANA client and web interface to display live log streams to the user.
- Run benchmarks to assess the performance of the developed solution.

The project will use an agile and iterative development model with the following tentative time plan:

Task	Weeks
Introduction to REANA infrastructure	1
Rapid prototyping of technical approaches to capturing main job logs from the Kubernetes pods	2
Design of the selected live logging system	3-4
Backend implementation	5-8
Client implementation	9-10
Performance benchmarks	11-12

Development of flexible interface for rapid geometry modelling and exchange between CAD tools and simulation platforms dedicated for HEP experiments

Applicant: Rimantas Naina

Mentors: Tomasz Szumlak, [Jakub Hajduga](#)

Simulation of complex HEP detection setups is critical to the understanding of the expected low-level response of respective sub-detectors, and reconstruction performance for crucial quantities such as impact parameters or lifetime of unstable composite particles or radiation damage. In each case, a detailed geometry of the detector is crucial for configuring the simulation chain. Building complicated geometry using custom HEP tools is typically very time-consuming and complex from the coding point of view. On the other hand, many enterprise tools offer high-quality interfaces for prototyping, templating and integrating complicated geometries. In this project, we propose to study the possibility of creating a custom interface between professional tools for complicated geometrical building and HEP platforms such as Geant4 or Fluka. If successful, such a tool would bring a much more reliable way of preparing and testing complicated detector models for the HEP community.

This project stands out for its focus on improving the utilisation of CAD-type geometries within Geant4 simulations. It is unique in its emphasis on developing a multifunctional toolkit for managing multiple CAD geometry elements and a rapid conversion mechanism for STL files to Geant4-compatible formats. These innovations, along with the design of a user-friendly interface for streamlined geometry configuration, are the main objectives of this project. We need to prepare a comprehensive development environment and establish a foundational understanding of the relevant libraries and tools, such as Geant4, STL, CAD, and GDML formats. Early efforts are directed towards creating a basic application framework in Geant4 that integrates initial STL/CAD geometry loading capabilities. This task is not trivial, and often, it is a significant entry threshold for people starting their activities in this field. One of our objectives is to streamline this process and define the software stack configuration easily and transparently.

As the project progresses, advanced features for dynamic geometry management are implemented. These allow for the handling of multiple independent geometry elements that can be modified without recompilation through external configuration files, such as TOML. This setup facilitates the addition, removal, and modification of geometries in a flexible manner.

A significant portion of the project is dedicated to the user interface development, which involves prototyping and implementing a tool (either command-line based or graphical user interface) that allows users to configure and manage geometries easily. This interface supports the precise positioning and configuration of multiple geometry elements as dictated by their original CAD specifications.

Parallel to interface development, the project focuses on creating a high-efficiency wrapper for the conversion of STL files into Geant4-compatible formats. This wrapper integrates seamlessly with the developed tools, ensuring that the conversion process is both quick and accurate,

catering to a variety of geometry complexities and configurations. In the final stages, comprehensive testing of the application is conducted to ensure that all components, including geometry loading, simulation processing, and user interface functionality, operate cohesively and effectively. The project concludes with the finalisation of detailed documentation to aid both users and future developers and a report summarising the progress and outcomes.

In summary, we aim to significantly enhance the capability of Geant4 simulations to integrate and manage complex CAD geometries more effectively and intuitively. By developing a suite of tools that simplify the processes of geometry manipulation and conversion, this initiative promises to advance the field of simulation by making sophisticated geometrical simulations more accessible and manageable for researchers and engineers alike.

Project Timeline Overview

Weeks 1-12:

- Throughout the project duration, focus on understanding the libraries and tools that support CAD-type geometry in simulations. Learn to effectively utilize STL geometry within Geant4. This ongoing familiarization process will provide the necessary foundation for tool development and integration. The final goals are to develop tools for handling multiple CAD geometry elements simultaneously, create a wrapper for rapid conversion of STL files to Geant4-compatible formats, and design a user interface for easy geometry configuration and management. .

Week-by-Week Breakdown

Weeks 1-2: Initial Setup and Planning

- Set up the development environment.
- Create a project repository and establish version control practices.
- Review Geant4, STL, CAD, and GDML formats and tools.
- Develop a barebones Geant4 application framework.
- Begin initial documentation covering project setup and basic usage.

Weeks 3-5: STL/CAD Integration

- Research and document existing tools and libraries for loading STL/CAD geometry into Geant4.
- Develop and test a basic module for loading STL/CAD geometries into Geant4.
- Incorporate the STL/CAD loading feature into the barebones Geant4 application.

Weeks 6-8: Geometry Management and Simulation

- Implement functionality to handle multiple independent geometry elements from CAD without recompiling the application.
- Develop mechanisms to add, remove, and modify geometry elements through external configuration files (e.g., TOML).
- Integrate sensitive detector definitions into the geometry elements.
- Test and validate the integration with simple geometry setups.

Weeks 9-10: Conversion and Optimization

- Design and prototype a user interface for configuring geometries,(e.g. command-line based TOML configuration or a simple GUI overlay).
- Implement the chosen user interface for basic geometry management.
- Refine the user interface for ease of use and enhanced functionality.
- Ensure the interface allows for configuring multiple geometry elements and setting their positions as defined in the CAD models.
- **Develop and integrate a wrapper for rapid conversion of STL files to Geant4-compatible formats, incorporating all features developed during the project.**
- Optimize the conversion process for efficiency and accuracy.
- Conduct thorough testing and validation with various geometry configurations and complexity levels.

Weeks 11-12: Final Integration and Testing

- Conduct comprehensive testing of the entire application, including geometry loading, simulation, and user interface.
- Debug and resolve any issues or inconsistencies identified during testing.
- Finalize documentation, including user guides and developer notes.
- Plan for future enhancements or maintenance tasks post-deployment.
- Preparing a report on the progress made.