Efficient Tracking Algorithm Evaluations through Multi-Level Reduced Simulations

Uraz Odyurt[1,2,*], *Ana-Lucia* Varbanescu[3,**], and *Sascha* Caron[2,4,***]

[1]Faculty of Engineering Technology, University of Twente, Enschede, The Netherlands
[2]National Institute for Subatomic Physics (Nikhef), Amsterdam, The Netherlands
[3]Computer Architecture for Embedded Systems, University of Twente, Enschede, The Netherlands
[4]High-Energy Physics, Radboud University, Nijmegen, The Netherlands

Abstract. Subatomic particle track reconstruction (tracking) is a vital task in High-Energy Physics experiments. Tracking, in its current form, is exceptionally computationally challenging. Fielded solutions, relying on traditional algorithms, do not scale linearly and pose a major limitation for the HL-LHC era. Machine Learning (ML) assisted solutions are a promising answer. Current ML model design practice is predominantly ad hoc. We aim for a methodology for automated search of ML model designs, consisting of complexity reduced descriptions of the main problem, forming a complexity spectrum. As the main pillar of such a method, we provide the REDuced VIrtual Detector (REDVID) as a complexity-aware detector model and particle collision event simulator. Through a multitude of configurable dimensions, REDVID is capable of simulations throughout the complexity spectrum. REDVID can also act as a simulation-in-the-loop, to both generate synthetic data efficiently and to simplify the challenge of ML model design evaluation. Starting from the simplistic end of the spectrum, lesser designs can be eliminated in a systematic fashion, early on. REDVID is not bound by real detector geometries and can simulate arbitrary detector designs. As a simulation and a generative tool for ML-assisted solution design, REDVID is open-source and reference data sets are publicly available. It has enabled rapid development of novel ML models.

1 Introduction

The adoption of data-centric solutions involving Machine Learning (ML) for computational aspects of natural sciences in general and High-Energy Physics in particular, is apparent. The expected challenges of the High-Luminosity upgrade to the Large Hadron Collider (HL-LHC) for instance, is widely believed to be addressable using such solutions.

When it comes to ML-assisted solution design, or more precisely, ML model design and training, the main set of challenges can be stated as: *the process of ML model design and training is an explorative, data-demanding, and costly endeavour*. Finding a suitable ML model, satisfying the problem at hand, predominantly has been (and still is) a manual ad-hoc effort, with the reliance on experience and expertise of the designer. Systematic approaches

*e-mail: uodyurt@nikhef.nl
**e-mail: a.l.varbanescu@utwente.nl
***e-mail: scaron@nikhef.nl

for automated Design-Space Exploration (DSE) are needed. However, during this DSE, any form of per design training of ML models is extremely costly.

To achieve highest efficiency, we propose an iterative approach and decompose the problem into consecutive complexity levels. Our focus is the critical task of subatomic particle track reconstruction, a.k.a., *tracking*. We designed and implemented the *REDuced VIrtual Detector (REDVID)*, as a highly configurable reduced simulation framework. Reduced simulations are used to generate collision event data, representing different complexity levels for the tracking problem. Both REDVID [1] and data sets generated with REDVID [2], utilised in ML model design for tracking [3], are publicly available.

2 Background and motivation

2.1 HEP experiments

When talking about *HEP experiments*, we refer to high-energy particle collision events. Two types of collision experiments are performed at LHC: proton-proton and ion-ion collisions. Protons are extracted from hydrogen atoms, while ions are actually heavy lead ions. Beams of particles are sent down the beam pipe in opposing directions and made to collide at four specific spots. These four spots are the residing points of the four major detectors installed at LHC, namely, ALICE [4], ATLAS [5], CMS [6] and LHCb [7].

Take the ATLAS detector for instance. The role played by ATLAS in the study of fundamental particles and their interactions, rely on two main tasks, *tracking* and *calorimetry*. Through tracking, i.e., particle track reconstruction, the momentum, p, of a particle can be calculated, while the energy, E, is calculated through calorimetry. Having the momentum and the energy for a given particle, its mass, m, can be determined, following the *energy-momentum relation* expressed as,

$$E^2 = (mc^2)^2 + (pc)^2.$$

In the above equation, c represents the speed of light and is a constant. The mass measurement allows the study of the properties for known particles, as well as potentially discovering new unknown ones. As such, it is fair to state that *particle track reconstruction is one of the major tasks in high-energy physics*.

2.2 Role of simulation in HEP

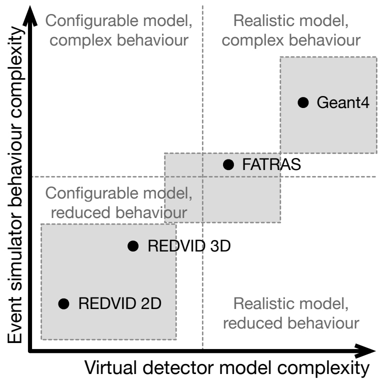
Simulation allows for, amongst others, the validation and training of particle track reconstruction algorithms. Two distinguished stages are considered for HEP event simulations, i.e., *physics event generation* and *detector response simulation* [8]. Event generation as the first stage, involves the simulation of particle collision events, encompassing the processes involved in the initial proton-proton or ion-ion interactions. Event generation is governed by intricate sets of physical rules and is performed by software packages such as Herwig [9] and Pythia [10], i.e., physics-accurate simulations.

Detector response simulation, the second stage, integrates the movement of the particles generated by the first stage through a detector geometry, simulating the decay of unstable particles, the interactions between particles and matter, electromagnetic effects, and further physical processes such as hadronisation. Common event simulators providing such functionality include Geant4 [11], FLUKA [12] and MCNP [13]. In accelerator physics applications, event simulators are used to simulate the interactions between particles and sensitive surfaces in an experiment, as well as with so-called passive material, such as support beams. Interactions with sensitive surfaces may undergo an additional *digitisation* step, simulating the digital signals that can be read out of the experiment. Considering the example of ATLAS, three data generating simulators are notable, namely, Geant4, FATRAS [8] and ATLFAST [14].

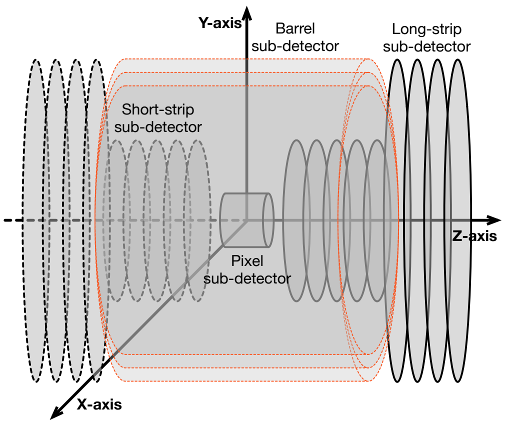
Following the Monte Carlo simulation approach, FATRAS has been designed to be a fast simulator. It is capable of trajectory building based on a simplified reconstruction geometry, does provide support for material effects, as well as particle decay, and generates hit data.

ATLFAST follows a different trajectory simulation approach and doesn't generate hit data, making it unsuitable for tracking studies. ATLFAST relies on hard-coded smearing functions based on statistics from full simulations. These functions are dependent on particle types, momentum ranges and vertex radii. Such details are specific to the design elements of the virtual detector geometry. A change in the design will require finding new functions.

REDVID fills the gap for a reconfigurable framework that is suitable for first-phase solution exploration and design. This is due to the deliberate reduction in complexity, for both the generated data and the problem description, while keeping the high-level causal relations in place. REDVID is end-to-end parametric, i.e., all the generated data is built upon the simplified detector geometry (see Figure 1b for an example) and randomised particle trajectories, both reconfigurable. REDVID has been developed in Python, making its integration with Python-based ML design workflows seamless. Figure 1a plots REDVID's positioning versus other well-known tools, as we consider it.



(a) Simulation complexity spectrum



(b) Simplified and configurable detector geometry

Figure 1: (a) Simulation complexity spectrum is shown from the most simplistic to the most realistic, with high complexity rates for both model and simulator. Depending on the enabled features, different simulators are capable of providing different levels of complexity, depicted as grey areas. (b) Example detector geometry from REDVID, enabling inclusion/exclusion of different sub-detector types, with control over sub-layer counts, sizes and placements.

3 Problem transformation

To be able to eliminate designer bias and at the same time, enable traversal of most possible ML model designs, systematic and automated approaches are to be adopted. The major obstacle for such an approach is the high cost of candidate model training, which in combination with a large design-space, leads to unrealistic overall search time. In this context, by *cost* we refer to computational burden and time as a closely related metric. Note that the considered *search* is the effort towards ML model design, i.e., deep learning models.

3.1 Reduction for a complexity spectrum

Assuming that a ML-assisted solution and thus a ML model design is being sought, solving approximate and simplified versions of a complex problem is more efficient in terms of inflicted model training cost. The more drastic of a simplification adopted, the higher the achieved cost reduction will be. We propose gradually increasing problem complexity by moving through a series of simplified versions of the original problem.

Focusing on approximate versions, none will lead to the intended solution. Such a stagnation will not benefit the cost-effectiveness of design-space search as well. For any given approximate version, or more generally, any arbitrary problem, there is more than one viable design, performing above (better) the acceptable limits. Moving closer to the original problem is necessary to differentiate between equally performant solutions. Early validation/elimination of lesser designs is the main advantage of a step-wise complexity increase.

To achieve a step-wise increase in the problem complexity, from simplified to on par with the original, forming a complexity spectrum, it is paramount to dissect the original problem and define its complexity invoking dimensions. A generic example for such a dimension is *scale*. For instance, in the context of tracking, the number of tracks, the number of hits, the frequency of events and so forth are familiar elements of experiment scale. A more specific example of a complexity dimension in this context is the track function representation. One can think of linear extrusions, quadratic extrusions, helical extrusions and so forth.

3.2 Notation for complexity levels

As such, one can define different consecutive levels of complexity for a given problem with known complexity dimensions. A complexity level C_p is defined by its complexity dimensions, $\{d_i, d_j, d_k, \dots\}$. Each dimension, here denoted as $i, j, k, \dots$, depending on the definition, can be represented in different steps. Different levels may contain different subsets of available dimensions, e.g.,

$$\begin{aligned} C_1 &= \{d_i, d_j, d_k\}, \\ C_2 &= \{d_i, d_j, d_k, d_l\}, \\ C_3 &= \{d_i, d_j, d_k, d_l, d_m\}. \end{aligned}$$

This can be formally expressed as: for an arbitrary complexity level C_p, the set of complexity dimensions D for the problem, and key-value pairs (d_q, v_q) as dimension and step value,

$$C_p = \{(d_q, v_q) \mid d_q \in D,\ v_q \in \{0, \dots, s_q\},\ q \in \{1, \dots, |D|\}\}. \tag{1}$$

Here, s_q is the step value associated with the dimension d_q, which could also be 0, denoting the exclusion of that particular dimension from that level.

However, it defeats the purpose to redesign, or even retrain models from scratch at every level. The move from one complexity level to the next must ensure the validity of previous learning, even if not in full. With the increase of complexity and for instance, scale, it is probable that certain redesigns could render learning partially void. To ensure such a constant increase in complexity means that every complexity level must include new additional elements compared to its preceding level. In other words, C_{p+1} is always a *proper superset* of C_p, ensuring the growth in complexity from one level to the next, which can be denoted as

$$C_p \subset C_{p+1}, \quad \forall p \geq 1. \tag{2}$$

Our proposed transition between different complexity levels is visually depicted in Figure 2.

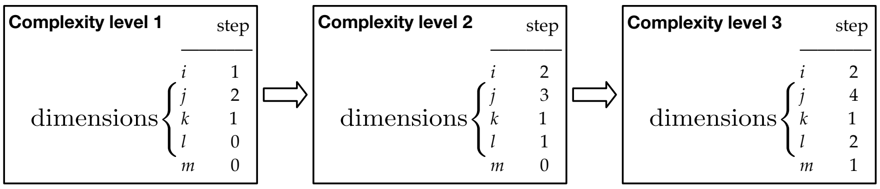


Figure 2: Visual representation of Equations (1) and (2), where the increase in complexity from one level to the next is enforced by addition of key-value pairs, i.e., increments in steps.

3.3 Reduced simulations

REDVID is intended to enable cost-effective ML model design search and evaluation. While this approach will be most advantageous with an automated ML model design search, e.g., Neural Architecture Search (NAS) techniques, it has been demonstrated through manual utilisation in [3]. Using REDVID, we achieve reduced simulations by customising two main categories of configurations, virtual detector model and particle collision event behaviour.

Virtual detector model elements

The virtual detector model is generated following the provided configuration file and can be of arbitrary design. This model in essence is composed of geometric shapes in the form of cylinder and/or disk. While it would be sensible to follow an arrangement resembling real-world detectors, there are no constraints for it. For our dataset generation, we have considered a virtual detector arrangement, inspired by the ATLAS detector at LHC. One way to move along the complexity spectrum is to increase/decrease detector resolution, e.g., layer shapes.

Particle collision event simulator

Using the simulator, randomised propagations of tracks within a collision event can be generated. The track propagation will depend on the considered detector behaviour, which contains another subset of complexity dimensions. A simple example would be the type of track extrusion, i.e., linear, helical uniform, or helical expanding.

4 Implementation

Written in Python, REDVID [1][1] supports simulation execution for both two-dimensional (2D) and three-dimensional (3D) detector geometries. While the 2D geometry is useful for sanity checks and educational assignments, this paper's focus is the 3D geometry.

4.1 Fundamentals

The simulator software is composed of three main modules. A *detector generator*, to spawn a detector based on the provided geometric specifics and configuration, an *event simulator*, to execute experiments involving many events, following the experiment configuration, and *reporting*, to collect the generated data set, as well as automated report generation.

For the case of the 3D space, we have opted for the cylindrical coordinate system to represent all elements, i.e., sub-detectors, tracks and hits. An example of a simplified detector

[1]Also available at: https://VirtualDetector.com/redvid/repository

geometry is depicted in Figure 1b. The cylindrical coordinate system is a convenient choice, as we are considering the Z-axis as the beam pipe in LHC experiments and all geometric shapes defined within a detector, whether disks or cylinders, are actually of the type cylinder.

In this coordinate system, hit points can be precisely defined given the tuple $(r_{\text{hit}}, \theta_{\text{hit}}, z_{\text{hit}})$. Hit point calculation depends on the virtual detector layers and generated track functions, which follows track randomisation protocols. Alongside other randomisations such as coordinate smearing, track randomisation is the main source of non-determinism for simulations. Different track randomisation protocols can be considered, but within the context of this paper, we follow *Protocol 1 - Last layer hit guarantee*. Hits are guaranteed to occur on the farthest layer of every sub-detector type, which means the farthest layer of every sub-detector type is the randomisation domain for the landing points of tracks. A hit guarantee on the last layer will also guarantee hits on the previous layers for that sub-detector type. This protocol is designed to maximise the number of hits per sub-detector type within the data set.

4.2 Configuration interface

The available configuration list, provided by users through configuration files, is extensive. Briefly, three main categories are present. Future versions of REDVID shall include each category in a separate file to improve modularity.

- Execution configuration: Here we cover settings such as anchor path, output format, parallel execution, performance monitoring, plot generation and so forth.
- Event behaviour configuration: Here we cover settings for experiment behaviour, e.g., event count, track count, track function, track randomisation protocol, and hit smearing.
- Virtual detector profile(s): Here the full set of geometric specifics, i.e., layer count, layer size, included layer types, and layer positioning, for either a 2D or a 3D detector is covered.

5 Tracking data sets

While REDVID simulations are highly flexible and from detector geometry to event generation behaviour, can be configured in different fashions, data sets are expected to conform to our methodology. Following the principle of having an increasing trend for problem complexity, i.e., fulfilling Equation (2), the following data sets can be noted:

1. `redvid_3d_noisy-100k-events-10-to-50-tracks`: 100 000 events, randomised ([10, 50]) linear tracks per event, noisy hit coordinates

2. `redvid_3d_noisy-100k-events-10-to-50-helical-tracks`: 100 000 events, randomised ([10, 50]) helical tracks per event, noisy hit coordinates

3. `redvid_3d_noisy-100k-events-50-to-100-helical-tracks`: 100 000 events, randomised ([50, 100]) helical tracks per event, noisy hit coordinates

This collection of data sets (available on Zenodo [2]), representing 3 consecutive complexity levels, is just one example. Users can come up with data set variations suitable for the use-case at hand. Here, while the event count is fixed at 100 000, randomised track count increases from [10, 50] to [50, 100] (transition from 2 to 3). At the same time, event simulation behaviour moves from linear to helical track functions (transition from 1 to 2).

6 Related work

Although the overall available data is abundant, corner case data is rather scarce. Such real-world data, or data synthesised with accurate (in our case physics-accurate) simulations is complex in terms of data dimensionality and granularity. This complexity is directly resulting from the complexity of the real system, or the accurate (physics-accurate) modelling of the system in case of simulations. We touched upon the complexity of simulators such as Geant4 in Section 2, as well as the dependence on these simulators by tools like ATLFAST.

The first challenge, lack of annotated data for one or more specific scenarios, has been recognised in the literature [15]. The second challenge though, the issue of complexity, is not as well known. A closely related acknowledgement has been made regarding the complexity level of models for simulations [16].

The two main shortcomings of previous efforts towards the use of ML in physics problems have been use-case specificity [17], and the lack of user-friendly tools [18]. As noted by Willard et al. [17], the efforts surrounding the use of ML for physics-specific problems are focused on sub-topics, or even use-cases. Our ambition is a detector-agnostic method.

The point from [17] regarding the computational efficiency of Reduced-Order Models (ROMs) matches our motivation. Where our work differs is in the placement of our ROM within our methodology. Our reduced model of a detector is considered as the model for simulations resulting in synthetic data generation, which is different than ML-based surrogate models as ROMs [19, 20], or ML-based surrogate models built from ROMs [21].

7 Conclusion

We have introduced our approach for reducing the cost of a systematic and automated ML model design search. We explained the principle of problem complexity reduction at different levels, alongside the formal notation for involved complexity dimensions. This method is by no means a way to find generic solvers, as low-complexity variants of the problem might not need a ML-assisted solution at all, breaking the incremental progression. Since such an approach will always be problem specific, we have applied it to the non-trivial problem of tracking. The REDVID simulation framework is our developed tool for this purpose, acting as a data generator for collision events at different complexity levels. A manual application of our approach has been effectively demonstrated with the TrackFormers project [3].

References

[1] U. Odyurt, Reduced virtual detector (redvid) (2025), 10.5281/zenodo.14947467

[2] U. Odyurt, N. Dobreva, TrackFormers - Collision Event Data Sets (2024), 10.5281/zenodo.14386134

[3] S. Caron, N. Dobreva, A.F. Sánchez, J.D. Martín-Guerrero, U. Odyurt, R.R. de Austri Bazan, Z. Wolffs, Y. Zhao, TrackFormers: In Search of Transformer-Based Particle Tracking for the High-Luminosity LHC Era (2024), 10.48550/arXiv.2407.07179

[4] T.A. Collaboration, K. Aamodt, A.A. Quintana, R. Achenbach, S. Acounis, D. Adamová, C. Adler, M. Aggarwal, F. Agnese, G.A. Rinella et al., The ALICE experiment at the CERN LHC (2008). 10.1088/1748-0221/3/08/S08002

[5] T.A. Collaboration, G. Aad, E. Abat, J. Abdallah, A.A. Abdelalim, A. Abdesselam, O. Abdinov, B.A. Abi, M. Abolins, H. Abramowicz et al., The ATLAS Experiment at the CERN Large Hadron Collider (2008). 10.1088/1748-0221/3/08/S08003

[6] T.C. Collaboration, S. Chatrchyan, G. Hmayakyan, V. Khachatryan, A.M. Sirunyan, W. Adam, T. Bauer, T. Bergauer, H. Bergauer, M. Dragicevic et al., The CMS experiment at the CERN LHC (2008). 10.1088/1748-0221/3/08/S08004

[7] T.L. Collaboration, A.A.A. Jr, L.M.A. Filho, A.F. Barbosa, I. Bediaga, G. Cernicchiaro, G. Guerrer, H.P.L. Jr, A.A. Machado, J. Magnin et al., The LHCb Detector at the LHC (2008). 10.1088/1748-0221/3/08/S08005

[8] K. Edmonds, S. Fleischmann, T. Lenz, C. Magass, J. Mechnich, A. Salzburger, Tech. rep. (2008)

[9] G. Corcella, I.G. Knowles, G. Marchesini, S. Moretti, K. Odagiri, P. Richardson, M.H. Seymour, B.R. Webber, HERWIG 6: an event generator for hadron emission reactions with interfering gluons (including supersymmetric processes) (2001). 10.1088/1126-6708/2001/01/010

[10] T. Sjöstrand, S. Mrenna, P. Skands, PYTHIA 6.4 physics and manual (2006). 10.1088/1126-6708/2006/05/026

[11] S. Agostinelli, J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce, M. Asai, D. Axen, S. Banerjee, G. Barrand et al., Geant4—a simulation toolkit, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment (2003). 10.1016/S0168-9002(03)01368-8

[12] T. Böhlen, F. Cerutti, M. Chin, A. Fassò, A. Ferrari, P. Ortega, A. Mairani, P. Sala, G. Smirnov, V. Vlachoudis, The FLUKA Code: Developments and Challenges for High Energy and Medical Applications (2014). 10.1016/j.nds.2014.07.049

[13] R. Forster, L.J. Cox, R.F. Barrett, T.E. Booth, J.F. Briesmeister, F.B. Brown, J.S. Bull, G.C. Geisler, J.T. Goorley, R.D. Mosteller et al., MCNP™ Version 5 (2004). 10.1016/S0168-583X(03)01538-6

[14] E. Richter-Was, D. Froidevaux, L. Poggioli, Tech. rep. (1998)

[15] C.M. de Melo, A. Torralba, L. Guibas, J. DiCarlo, R. Chellappa, J. Hodgins, Next-generation deep learning based on simulators and synthetic data (2022). 10.1016/j.tics.2021.11.008

[16] L. Chwif, M. Barretto, R. Paul, On simulation model complexity (2000)

[17] J. Willard, X. Jia, S. Xu, M. Steinbach, V. Kumar, Integrating scientific knowledge with machine learning for engineering and environmental systems (2022). 10.1145/3514228

[18] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, L. Zdeborová, Machine learning and the physical sciences (2019). 10.1103/RevModPhys.91.045002

[19] G. Chen, Y. Zuo, J. Sun, Y. Li, Support-vector-machine-based reduced-order model for limit cycle oscillation prediction of nonlinear aeroelastic system (2012). 10.1155/2012/152123

[20] M.F. Kasim, D. Watson-Parris, L. Deaconu, S. Oliver, P. Hatfield, D.H. Froula, G. Gregori, M. Jarvis, S. Khatiwala, J. Korenaga et al., Building high accuracy emulators for scientific simulations with deep neural architecture search (2021). 10.1088/2632-2153/ac3ffa

[21] D. Xiao, C. Heaney, L. Mottet, F. Fang, W. Lin, I. Navon, Y. Guo, O. Matar, A. Robins, C. Pain, A reduced order model for turbulent flows in the urban environment using machine learning (2019). 10.1016/j.buildenv.2018.10.035