Including time information in track reconstruction for future colliders

Wei-Po Wang, Leonhard Reichenbach, Swathi Sasikumar

CERN Summer Student Program

Abstract

Precise track reconstruction is a critical process in achieving optimal measurements for collider experiments like the proposed Compact Linear Collider (CLIC), which operates at high energy with electron-positron collisions. In this context, conformal tracking emerges as a tracking algorithm utilizing cellular automaton within a conformal mapped plane. In this report, by considering the timing information of track hits in the conformal tracking algorithm, we have shown a 62 percent reduction in reconstruction run time.

Geneva, Switzerland
June 26th - August 18th, 2023

Contents

1 Introduction and motivation

The Compact Linear Collider (CLIC) [1] plans to utilize tightly packed electron-positron (e^-e^+) bunches. These bunches will have a small transverse size of 45 nanometers by 1 nanometer and a length of 44 micrometers during the 3 TeV operational phase [2][3]. The collision of these tightly packed bunches generates intense electromagnetic fields, leading to the emission of a significant number of photons. These emitted photons are a primary source of background, as they create incoherent electron-positron pairs, resulting in events like $\gamma\gamma \rightarrow$ hadron interactions. This background phenomenon, we named it "overlay" [1], leads to a substantial hit occupancy in the detector, increasing the complexity of the process of identifying particle tracks [3][4].

Motivation

To address these challenges, a four-dimensional tracking approach can be employed. By considering the timing of the track hits, we can mitigate overlay hits, possibly improving the association between tracks and vertices, and enhancing the overall performance of track reconstruction. This strategy becomes particularly important in CLIC's high-luminosity environment, where the presence of background events poses significant hurdles to accurate event reconstruction and analysis [5].

1.1 Compact Linear Collider (CLIC)

The study of particle interactions at the terascale is regarded as holding the solutions to numerous fundamental questions in particle physics. These questions encompass a wide range of topics, including understanding the mechanism behind electroweak symmetry breaking, potentially confirming new symmetries, comprehending Dark Matter, and conducting searches for various phenomena that extend beyond the Standard Model of particle physics [1]. The Large Hadron Collider (LHC) has made it possible to experimentally explore physics at the terascale. The global particle physics community, supported by organizations like the International Committee for Future Accelerators (ICFA), has reached a consensus that the outcomes obtained from the LHC will require complementing with experiments conducted at a lepton collider operating in the tera-electron-volt (TeV) energy range. CLIC is a linear collider at the multi-TeV scale that can offer a compelling and comprehensive physics program of discoveries and precision measurements, perfectly complementing the LHC [1]. The CLIC two-beam concept

[1] We name it overlay because we simulate the background events separately and overlay them onto the physics event.

proposed in 1986 provides a unique approach to reach multi-TeV energies with an $e^- e^+$ collider, and the concept has been studied and developed til today.

1.2 Conformal tracking

The reconstruction algorithm used is called conformal tracking. It is essentially a cellular automaton-based track finding in conformal space [4]. Conformal map-

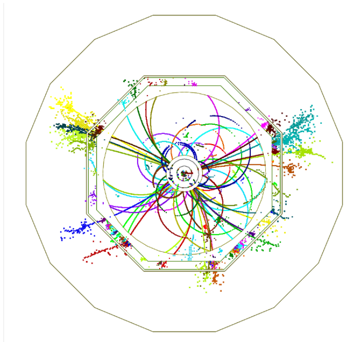

Figure 1: An event display of Japan Linear Collider showing the circular tracks of particles in a detector .

ping involves transforming Cartesian hit coordinates into a different space known as the conformal space. The transformation is given as

$$u = \frac{x}{x^2 + y^2} \; , \; v = \frac{-y}{x^2 + y^2} \tag{1}$$

The key concept of this mapping is that circles that pass through the origin of the Cartesian coordinate system can be transformed into straight lines in the conformal space. This transformation allows for simplifying the process of identifying circular tracks created by charged particles moving within a uniform magnetic field.

Once the hits are transformed to the conformal space, a cellular automaton is implemented. The cellular automaton contains two sections, the building and the extrapolation of cellular tracks. To construct cellular tracks, first, each hit in the input collection is considered as a seed hit. A nearest neighbor search is performed around this seed hit to locate neighboring hits within a specified polar angle region. The seed hit forms a "seed cell" with each neighbor hit respectively. Seed cells are then extended to include virtual hits situated along the extrapolation of the seed cell in the radial direction. From these virtual hits, another nearest neighbor search is conducted. Then the process is repeated to create multiple candidate tracks. Eventually, the best track is selected through selection methods such as linear fitting.

2 Timing cut in cell building process

The first place we took the timing information into consideration was the cell building process. This part of code is located under `ConformalTracking.cc` in the `buildNewTracks()` function. In this process, the neighbor hits of a seed hit is looped over to form a cell with the seed hit. Here, we've set two types of time cuts, the "time order cut" and the "max timing difference cut".

The time order cut is where we eliminate the neighbor hits who arrive before the seed hit which are the hits with a timing greater than the seed hit ($T_{neighbor} > T_{seed}$). For the max timing difference cut, we set a certain value (max timing difference) as the upper limit of timing difference between the seed hit and the neighbor hit, and we eliminate those with a timing difference greater the value of max timing difference. The underlying idea is that while the particles travel at a speed near light speed, the seed hit and its consecutive hit should have a timing difference smaller than the timing difference between a seed hit and an overlay hit.

2.1 Value of max timing difference

Now, it is important to find a proper value for the max timing difference (T_{max}). If the value is too big, too many hits will be eliminated and causing a drop in reconstruction efficiency. On the other hand, if the value is too snall, not many hits will be cut and thus not much improvement is made.

In order to obtain an appropriate value for T_{max}, we recorded the timing difference between the seed hits and their neighbor hits. In addition, since we are running the reconstruction for simulation files (.slcio), we are able to know which hits are left by overlay particles using the `isOverlay()` function under the class `SimTrackerHit`. Also, the `getMCParticle()` function helps us read which simulation particle caused a hit. With these functions, we are able to draw the histograms as shown below from reconstruction of one ttbar simulation file. The first histogram is of time difference between overlay hits and seed hits, and the second histogram stores the time difference between the seed hit and its actual consecutive hit. [2]

[2] These results are stored from the reconstruction of one ttbar simulation file which contains 25 events.

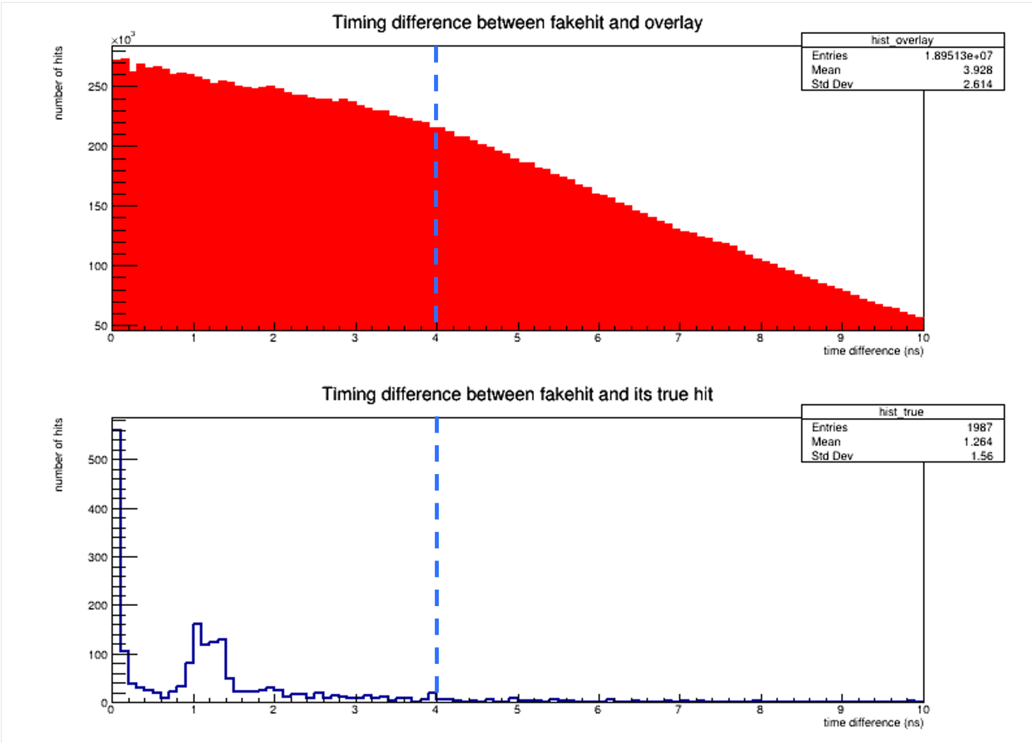

Figure 2: The upper histogram is of time difference between overlay hits and seed hits, and the lower histogram stores the time difference between the seed hit and its actual consecutive hit. The blue dash line represents the timing cut.

From the lower histogram in Fig. 2, we can see that most of the true consecutive hits have a time difference below 4 ns with their seed hits. Therefore, we set the value of T_{max} as 4 ns meaning that all the hits on the right of the blue dash line will be eliminated during the cell building process. This value of timing cut is set in the steering file [3] in the variable called `MaxTimeDifference`.

[3]The steering file used `https://github.com/RCLanger/CLICPerformance/blob/hit_information/clicConfig/clicReconstruction.xml`

2.2 Performance results

After implementing the "time order cut" and the "max timing difference cut", we compared the reconstruction efficiency and the reconstruction run time of the original code and the new code with timing cuts. And the reconstruction efficiency is defined as

$$\text{reconstruction efficiency} = \frac{\text{number of correct reconstructed tracks}}{\text{total tracks}} \tag{2}$$

And the run time is recorded using the `time` command in linux terminal.

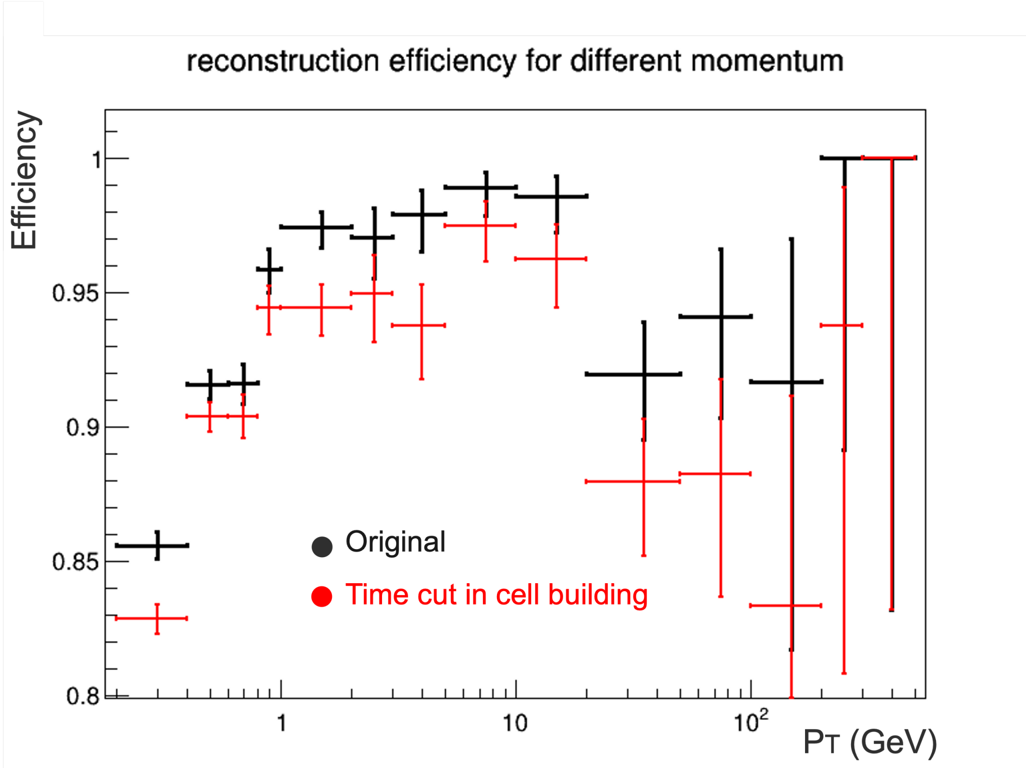

Figure 3: The efficiency plot of original code and code with timing cuts in cell building process.

The run time for the reconstruction of the same ttbar simulation file has improved from 191 minutes to 75 minutes. We can see from Fig. 5, since our time cut have eliminated some true hits a bit of the efficiency is sacrificed. On the other hand, with all the overlays being skipped during the cell building loop, the reconstruction run time is reduced by 61 percent.

3 Timing cut in cell extrapolation process

Next we move to the cell extrapolation process which corresponds to the `extrapolateCell()` and `extendSeedCells()` function under `ConformalTracking.cc`. Before entering this process, a seed hit has already formed seed cells with its neighbor hits. In the `extrapolateCell()` function, the conformal tracking algorithm linearly extrapolates a new hit from the two hits in the seed cell. This new hit is named "fakehit". It is essentially our estimation of where the next hit should be. Next, the algorithm loops over the neighbor hits of the fakehit to find the hit that corresponds to the fakehit in the `extendSeedCells()` function.

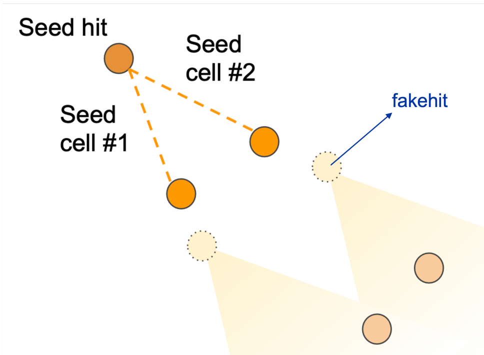

Figure 4: A drawing of the cell extrapolation process in conformal tracking [6].

3.1 Fakehit

Originally, the algorithm only calculated the position of fakehits, and did not consider the timing information when looking for the corresponding hit of the fakehit. In order to utilize the timing information, we first need to calculate the timing of the fakehit.

To obtain the timing of the fake hit, we first convert the hits from conformal plane back to Cartesian. Our conformal mapping $f(z)$ is basically a inversion to a complex number z, which can be written as

$$f(z) = \frac{1}{z} \tag{3}$$

where $z = x + iy$ while x and y are real. Therefore, by doing the transformation again, we can transform the hits back to the Cartesian plane.

In the Cartesian plane, the consecutive hits should align to form a circular track.

To utilize this feature, we first find the circle that passes through all three hits (two hits in the cell and the fakehit). The center of the circumcircle of point A,B,C can be written as

$$U_x = \frac{1}{D}\left[(A_x^2 + A_y^2)(B_y - C_y) + (B_x^2 + B_y^2)(C_y - A_y) + (C_x^2 + C_y^2)(A_y - B_y)\right] \quad (4)$$

$$U_y = \frac{1}{D}\left[(A_x^2 + A_y^2)(C_x - B_x) + (B_x^2 + B_y^2)(A_x - C_x) + (C_x^2 + C_y^2)(B_x - A_x)\right] \quad (5)$$

where

$$D = 2\left[A_x(B_y - C_y) + B_x(C_y - A_y) + C_x(A_y - B_y)\right] \quad (6)$$

This is obtained by simply plugging the position of three points into a equation of a circle. With the center of the circumcircle solved, we obtain the angle traveled between the three hits. Since angular speed of a particle is constant, the time spent in each interval should be proportional to the angle traveled. Finally, because the timing of first two hits is given, we obtain the value of fakehit timing.

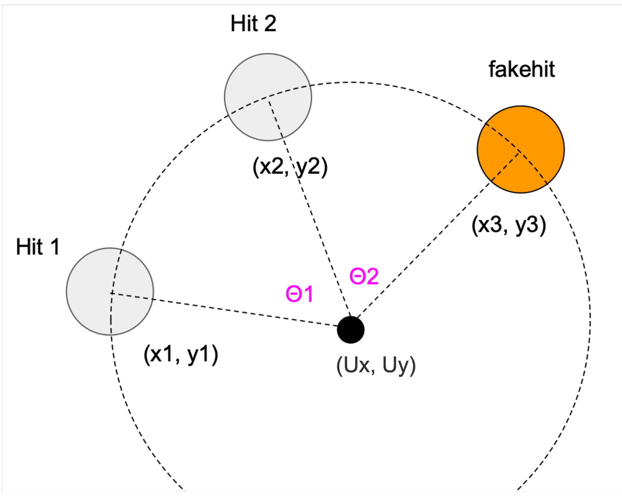

Figure 5: A drawing of the fakehit timing calculation.

3.2 Value of fakehit timing cut

With the timings of fakehits given, we can set a timing cut to eliminate neighbor hits that has a great timing difference with the fakehit. Now, similar to section 2.1, we need to find a proper value T_{max} for the timing cut.

Again, we record the timing difference between the fakehit and its neighbor hits. Then separate true track hits that corresponds to the fakehit and overlay hits as shown in Fig. 6.

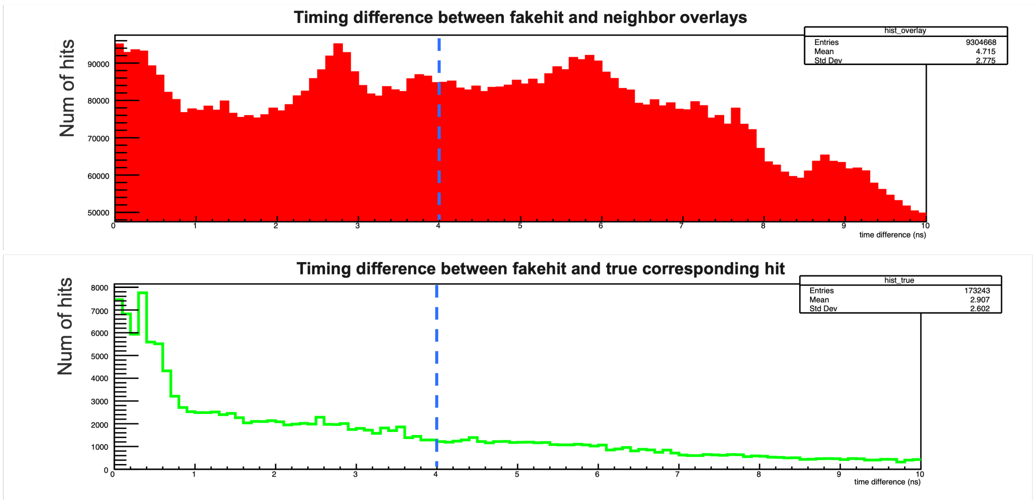

Figure 6: The upper histogram is of time difference between overlay hits and fakehits, and the lower histogram stores the time difference between the fakehits and its actual corresponding hit. The blue dash line represents the timing cut.

As a result, We set the timing cut value at 4 ns to try to cutoff the most overlays without losing too many real hits.

3.3 Performance results

After the cut in cell extrapolation process is implemented, we again compare the reconstruction efficiency and run time with the previous versions. As we can see, the efficiency as shown in Fig. 7 hasn't changed too much and the run time is improved from 75 minutes to 73 minutes, corresponding to a 3 percent reduction.

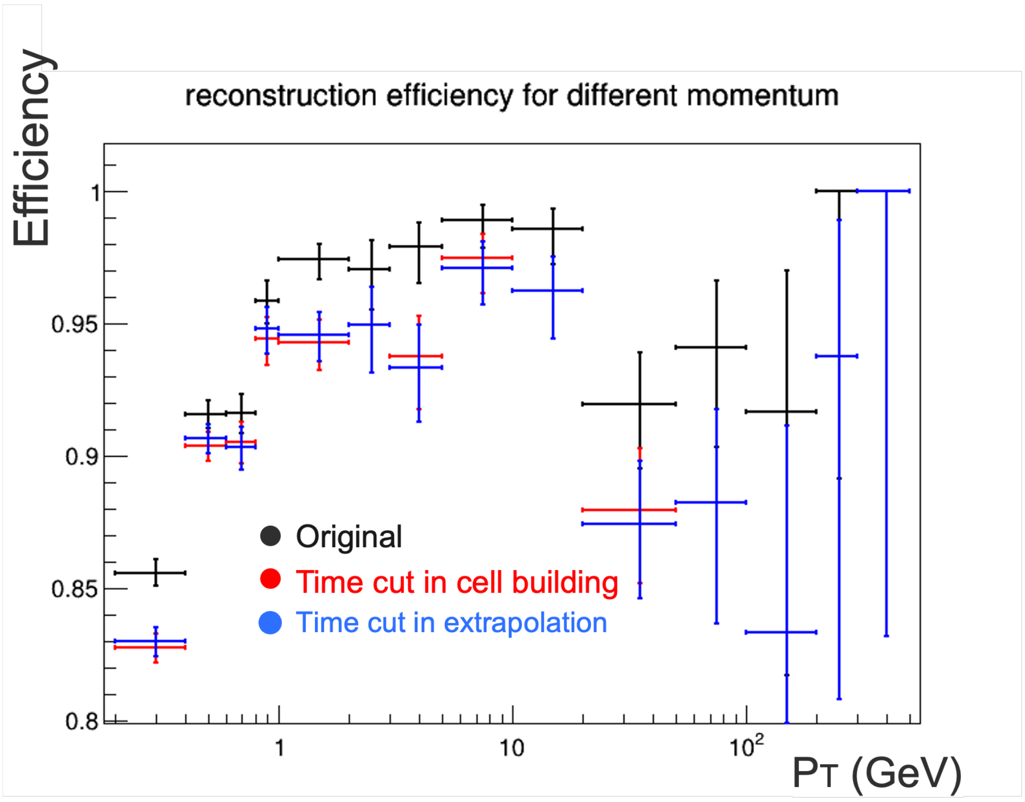

Figure 7: The efficiency plot of original code (black) , code with timing cuts in cell building process (red) and code with cut in cell building + extrapolation (blue)

4 Conclusion and outlook

By including the timing information in cell building and extrapolation process, we have reduced the reconstruction run time by around 62 percent while sacrificing only a bit of efficiency [4].

In the future, there are still a lot to be done for this topic. First of all, we can see from Fig. 5 and 7 that the error bars are great especially for particles with high transverse momentum. This is mainly because the results shown in this report are reconstructions of one simulation slcio file. The algorithm needs to be tested with more simulations to reduce the error. In this way, we will also be able to evaluate the efficiency loss more thoroughly and quantitatively. Also, it would

[4]The conformal tracking code editted during this project can be found at `https://github.com/webithug/ConformalTracking/blob/hit_information/src/ConformalTracking.cc#L1749`

be valuable to look into the fake reconstruction rate. As we eliminated the overlays, we might reduced the number of wrongly reconstructed tracks.

Besides, a simpler and more accurate way to calculate fakehit timing should be developed. As shown in Fig. 6, there is a tail that goes til the end of the histogram, meaning that some of the current estimation of fakehit timing is off by values greater than 10 ns. Another method we can try to calculate the fakehit timing is to go to the longitudinal plane and utilize the feature of particles' constant velocity in z direction.

In addition, the fakehit position estimation also needs small improvements. In the current conformal tracking code, under the `extendSeedCells()` function, the length of extrapolation is given by the `MaxDistance` variable set in the steering file. However, currently, there seems to be no proper value for the variable. This might also be a reason for the inaccuracy of the fakehit timing estimation.

Acknowledgments

I would like to thank Leonhard Reichenbach and Swathi Sasikumar who guided me through the whole project and were always there when I encounter any obstacles. My thanks are also due CERN and especially SFT for providing me this great environment to learn and work on this project during this amazing summer.

References

[1] CERN, A MULTI-TEV LINEAR COLLIDER BASED ON CLIC TECHNOLOGY, 2012. Available: https://project-clic-cdr.web.cern.ch

[2] A. Sailer, Simulation of beam-beam background at CLIC, 2010. arXiv: 1006. 3492.

[3] E. Brondolin, New developments in Conformal Tracking for the CLIC detector, 2019. [Online]. Available: https://twiki.cern.ch/twiki/pub/CLIC/CTD19/201904_CTD2019.pdf.

[4] E. Leogrande, Conformal tracking for the CLIC detector, 2018. Available: https://indico.cern.ch/event/658267/papers/2813728/files/8362-Leogrande.pdf

[5] Raffaela Langer, Including time information in conformal tracking for future colliders, 2021. Available: https://cds.cern.ch/record/2779428

[6] Erica Brondolin, Andre Sailer, Conformal Tracking for all-silicon trackers at future electron-positron colliders, ECFA Higgs Factories: 1st Topical Meeting on Reconstruction - 4th May 2022.