Vision, Modeling, and Visualization (2022)
J. Bender, M. Botsch, and D. Keim (Eds.)

Semi-Automatic Particle Tracking for and Visualization of Particle Detector Data

R. Eschbach[1], K. Messerschmidt[1], R. Keidel[1], and A. Wiebel[1] on behalf of the *Bergen pCT Collaboration*

[1]Hochschule Worms University of Applied Sciences, Germany

Abstract

In high energy physics, tracking particles in point-based particle detector data is important to reconstruct the particles' trajectories. As the numbers of particles can be in the hundreds or over a thousand, automatic algorithmic tracking is usually preferred over manual tracking. However, in some cases, manual tracking is needed as a baseline to assess the quality of the algorithmic tracking. Tracking particle locations manually is time-consuming and challenging when dealing with thousands of particles in the same data frame.

In this paper, we describe Semi-Automatic Particle Tracking *(SAPT), a collection of methods that aid manual particle tracking and visualization. These methods aim to make related particle hits easier to recognize by stretching the data and hiding likely irrelevant hits based on an angle criterion. They also help with finding the most likely track among a set of intuitively selected detector hits. These methods, together with a prediction of the most probable continuation of a track that can simply be accepted by the human user, accelerate the manual tracking process tremendously. We demonstrate the usefulness and efficiency of our methods by applying them to simulation data of a detector for proton computed tomography (pCT).*

1. Introduction

In high energy physics, tracking particles and thus reconstructing trajectories of particles in space has been supported algorithmically for many years [Gro87, PRK88]. The conventional algorithms [PAB*17] and, more recently, the machine learning models [CFG*18, ABC*20] used for this purpose, usually have to be developed or adapted for the specific particle detectors in use. To assess the quality of an algorithmic track reconstruction method for a certain detector, the resulting tracks should optimally be compared against the correct (*real*) trajectories of the particles as a baseline or ground truth. These trajectories, however, are only available in the case of a simulated detector. They obviously cannot be available for the real detector because here they are what the algorithmic reconstruction is striving to obtain in the first place. This dilemma in the context of the real detector can be solved by experts performing the particle tracking on the real detector data manually and using these tracks as ground truth. Unfortunately, manual tracking is challenging and time-consuming when dealing with many particles in the same data frame.

In this paper, we therefore present a collection of visualization, interaction, and prediction methods that supports experts in manually tracking particles in a fast and accurate way. The result is a framework for *semi-automatic particle tracking* (SAPT). Our application case is proton tracking for proton computed tomography (pCT) [CK76, SBK*04] using a layered detector design (see figure 1) as suggested by the *Bergen pCT Collaboration* [AKSP*20].

Most of the presented methods are applicable for aiding manual particle tracking in any point cloud representing locations of moving particles at different time steps. Few of the methods depend on a *layered* structure of the detector, and some on the fact particles usually do not take arbitrarily sharp turns.

2. Background, Context and Data

The particle data used for demonstrating SAPT comes from a model of a particle detector designed for proton computed tomography (pCT).

2.1. Background and Context

Proton computed tomography is an imaging technique first proposed by Allan MacLeod Cormack [CK76, Cor63] in 1963. It describes the use of proton trajectories for imaging, similar to X-ray computed tomography (often simply called CT). This method was later picked up in the context of proton radiation therapy. Here, tumors are targeted with the so-called Bragg peak effect of fast-moving protons. Currently, the amount of energy and thus the speed a proton needs to be deposited in the tumorous cells can be calculated from X-ray tomography images. Using pCT for the imaging instead would lead to less possible side effects due to X-ray radiation exposure and a more precise calculation of the needed energy. Additionally, pCT could be performed with the same device in the same room as the radiation therapy. This avoids relocation of the patient and thus possible movement of the tumor between imaging (and thus therapy planning) and the actual therapy.

2.2. Data

To develop a particle detector and the techniques essential for creating the pCT system, the *Bergen pCT Collaboration* [AKSP*20] has been

DOI: 10.2312/vmv.20221210

established. The data used to demonstrate our methods stems from simulations performed during the development phase of the detector. The detector consists of a number of pixel-based sensor layers (see figure 1). Each layer captures the positions (*hits*) of the particles passing it and the energy deposited when a particle interacts with the sensor layer. The overall energy deposited by all hits of one particle, i. e. the hits belonging to one trajectory, and the direction in which the particle enters the detector are used to compute the pCT. When particles interact with the sensor layers or the material between these layers, the direction they are moving in can change slightly. As a result, the sequence of hits of a single particle does not form a straight line but a slightly curved path (see e. g.figures 2 and 10). This non-straight shape and the fact that a single data frame (detector read-out frame) usually contains the hits of numerous particles necessitates sophisticated tracking methods to correctly reconstruct the particle trajectories from the hits.

Figure 1 shows a data set with 10,000 particles. A data set with 100 simulated particles, resulting in 87 trajectories and 2014 hits, is used for all other figures and the video accompanying this paper. Due to primary particles splitting into secondary particles before they hit the detector, the amount of simulated particles and trajectories is not equal. Secondary particles, particles that split off from other particles, are removed from our data set as those are not relevant for the tracking. Depending on the effectivity and efficiency of the developed detector, the number of particles in one frame will lie in between the two above-mentioned numbers of particles [PAB*17].

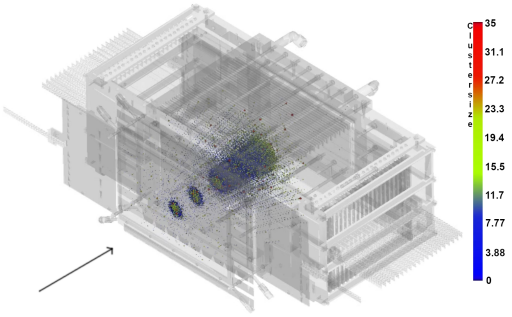

Figure 1: *Rough model of detector geometry with hits of 10,000 simulated protons entering the 43 (2 front-tracker and 41 detector) layers from the lower left. The arrow indicates the proton beam direction. Hits in the detector layers are color coded by the number of detector pixels firing at the same time (cluster size). The colorful color scale has been chosen for illustration purposes only.*

3. The Problem

Currently, the automatic tracking algorithms are fed with data from a Monte Carlo simulation [PAB*17]. These help to develop different algorithms and assist in training machine learning approaches. This simulated data, however, might differ from the real detector data, making the developed algorithm unstable. For this eventuality, ground truth data needs to be manually created from the real detector data. For several hundred or a thousand particle tracks, naive manual tracking is an arduous task. Manually tracking one hundred tracks by naively clicking on a single particle at a time can take a lot of time.

This is why, in this paper, we propose multiple support functions that aid in the manual tracking so that creating a ground truth from the detector data can be done as quickly as possible.

4. Related Work

The previous work related to our methods can be divided into three main categories: algorithmic tracking, semi-automatic tracking and selection methods. We discuss these methods in separate subsections in the following.

4.1. Algorithmic Tracking

A large variety of algorithms for tracking the movement of particles have been proposed in the past. Here, we will focus on more recent methods that have been developed especially for our *pCT* application and for the Large Hadron Collider (LHC) at CERN [CER21]. This allows us to understand what inspired some of SAPT's methods.

Usually, algorithmic tracking methods rely on two types of information. The first type are geometric properties of the particle locations. Examples are spatial proximity of the points, the angle formed between three consecutive points and the shape of curves the points should lie on. The second type are physical properties at the particles' location. An example for this being the energy deposited at the measured particle location. Furthermore, it is not uncommon to use a combination of the two types.

Recently, a number of methods for tracking particles in the LHC using machine learning methods (ML) have been developed during a competition called *TrackML* [ABC*20]. Many of these methods relied on starting at a triplet of points and extending this seed by extrapolating it into a helix shape. The point closest to this extrapolation would be added to the existing tracking to extend it. In this case, a roughly helix-like shape is expected due to the special magnetic field used in the LHC or the detector itself. The extrapolation step in some methods depends not only on the last three points, but also on the local magnetic field as a physical property.

In the context of detectors as the one shown in figure 1, algorithms based on the angle formed between successive detector hits have been developed. One of these algorithms (see [PAB*17] based on [SF10]) uses the relative speed and the composition of the detector to calculate an expected angle. An improved version adds a more accurate calculation for the scattering [PMO*20]. In an additional step, the energy deposited when particles hit detector layers can be used to filter out wrongly reconstructed tracks [PAG*21].

A different method [Sik17] tries to keep competing choices open as long as possible. Here, track candidates and their corresponding hits (the nodes) form a usually highly connected network (a bipartite graph), allowing for multiple hit to track assignments (edges). After that, a depth-limited search on sub-graphs is performed. It aims at maximizing the number of hits on tracks, and minimizing the sum of a track-fit measure.

Broadening the view, we also looked at an approach [ESF21] that focuses more on biological cells. This approach uses a three stage object tracking. Here the object is first detected, then classified and tracked [ESF21].

Some particle tracking algorithms are mainly based on calculating an approximated turn angle for the detector hits as it goes from one layer to the next. These calculations include the kinetic energy of the particle. The problem here lies in having only the deposited energy of a particle in a certain point, and thus needing to sum up from the end to get an approximate kinetic energy. That would mean the user always has to start at the end of the track. What we can adopt from the algorithms is the principle of a turn angle, as this can be calculated between the positions of the detector hits. Inspired by the mentioned works, the methods in SAPT use extrapolation, an angle criterion and a graph search to support the user. An algorithm [vdE02] using Kalman filters [Kal60] to estimate the course of tracks could be useful in future extensions of SAPT.

Many more automatic tracking techniques which work on data with characteristics different than those of our point cloud data have been devised in the past. One example are feature tracking methods that work on 2D or 3D time-dependent flow simulation data [PVH*03, YMS*21]. Other approaches work on image data, e. g., for tracking biological cells [ESF21]. As the characteristics of the data used by these techniques are so different, they are not applicable in our case.

4.2. Semi-Automatic Tracking

Many of the semi-automatic tracking techniques proposed in the past focus on time-dependent 2D image data [ESF21]. One example of semi-automatic tracking working on sets of 2D images is the software TrackMate [TPS*17]. It is a plugin of the Fiji [SACF*12] ImageJ distribution that allows automatic, semi-automatic and manual tracking of particles, which in their terminology means cells. TrackMate allows fully manual tracking of a cell or fully automatic tracking of a cell. It also allows changing the parameters for the automatic tracking and manual correction of errors. Due to the latter, it can be seen as a semi-automatic approach.

Our data, being only one full set of hits in different detector layers, could be interpreted as representing slices (layer) in time. But TrackMate's algorithms work on pixel data, while our data is a somewhat structured point cloud. Converting this point cloud into pixel data would lose accuracy and the possibility of adding additional detector data, like the energy deposition, to the tracking. Working with pixel data is also more time-consuming than working with a point cloud, especially when increasing the resolution of the image to overcome the loss of accuracy. To test this, we converted our point cloud data to pixel data and tried tracking the particle movement with track mate. While lower image resolutions manage the tracking in less than one minute, no trajectory is tracked correctly. We tested resolutions up to eight times the minimum resolution of our data and while the accuracy did go up to roughly five percent, the tracking did take more than one hour. Therefore, TrackMate's approach is not one we can follow. Nevertheless, the idea of taking fully automatic tracked data and giving the user the ability to manually correct the tracks could be a good extension for the future. This can be coupled with a visual representation of the probability of the tracks' correctness (e. g. color coding).

4.3. Selection Methods

Since we have a 3D collection of detector hits and need to find a track among them in a projection on a 2D screen, we also examined approaches for 2D selection of objects in 3D. These approaches use different techniques to deal with the issue of bridging the gap between 2D and 3D in a meaningful way.

The first approach deals with the selection of 3D lines in a direct volume rendering [WPVH13], i. e. in an inherently 3D rendering, of continuous 3D data. It builds a graph through a set of candidate points [WVFH12] and uses Dijkstra's algorithm [Dij59] to find the shortest path through it. The second approach addresses selection of structures in 3D point clouds. It builds a 3D triangle mesh out of a 2D lasso selection. The mesh surrounds the points belonging to the desired structure [YEII12]. This approach has been later extended to work more locally and thus be more precise [YEII16]. Additionally, a GPU-based acceleration has been proposed [KK18]. Although they deal with different data [WPVH13] or do not aim at lines [YEII12] (here tracks), the interaction [YEII12] and graph-based methodology [WPVH13] of these approaches inspired our SAPT method.

5. Methods

In order for the methods to be applicable, the data needs to adhere to a few basic assumptions. The particle detector data needs to contain the $(x,y,z) \in \mathbb{R}^n$ coordinates and the corresponding energy deposition E_{dep} of the hits. Also, the coordinate that is on the layer axis needs to be exactly equal to all other coordinates on the same layer. E. g., if the layer axis is the z axis, which we assume throughout this paper, we assume all hits of the same layer share exactly the same z coordinate. Since the detector is constructed of multiple layers, most of our methods assume, a track can only include a maximum of one hit per layer.

We can divide our methods into *visualization* and *selection helper* functions. The visualization functions give the user a better understanding of the relation between hits, while at the same time only impacting the visual appearance and not the selection. The selection helper functions on the other hand impact and filter the selection of the user. A method can be a mix of these two categories and aid the user visually while impacting the selection.

5.1. Stretching the Data

The first pure *visualization* function is a simple stretching. Stretching the data with respect to the x and/or y axes can make the space between hits in one layer more apparent (see figure 2), and thus allows the user to better see how hits are related in z direction. A fast and easy way, e. g., a keyboard shortcut, to scale the whole data in three axes individually is beneficial for the workflow.

5.2. Selection of Multiple Hits

Having to click and precisely target each hit is a tedious task, especially if there are numerous hits. It would involve a lot of zooming and rotating to select one single hit. Therefore, selecting multiple hits at once is a better option as it enables the user to perform a loose selection around the hits without need for a precise click. For multi-selection methods, we employed three different approaches: *brush*, *box*, and *lasso* (see figure 3). With the brush selection, users can draw over hits like they would with a paintbrush. It is the slowest of the methods, but it also allows being the most precise. The box selection lets the user drag a box around their selection. This is the fastest selection method but also the least precise as it forms a rectangle around the dragged area.

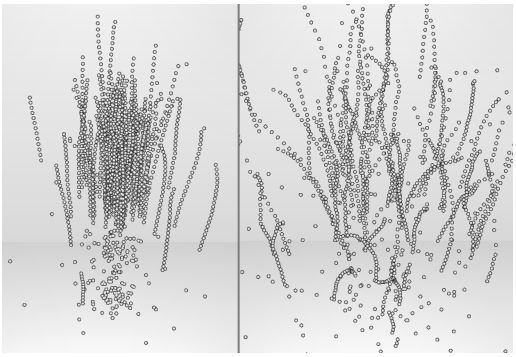

Figure 2: *Stretched visualization of data detector hits. Left: No scaling yields an extremely cluttered image. Right: x and y scaled by* 4 *makes different tracks better distinguishable.*

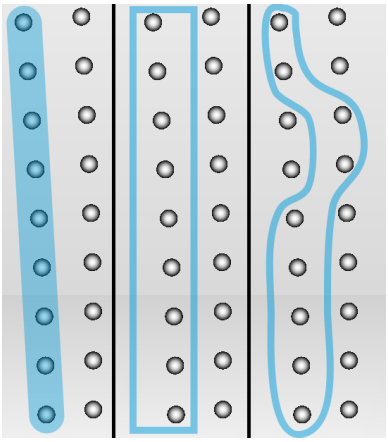

Figure 3: *The different selection types. From left to right, brush selection, box selection and lasso selection. The brush and lasso selections are free hand, while the corners of the box are dragged from point A to point B. Box and lasso select what is inside the selection, while brush selects what is covered by it.*

As a middle ground between these two methods, we have the lasso. Like the brush it can be freely drawn and like the box it selects the hits inside the boundary curve. Since brush and lasso are free hand selection methods, they are most effective with a graphics tablet and a digital pen.

5.3. Adaptive Visibility

This is another *visualization* function, but in contrast to the stretching, the adaptive visibility can also be used as a pre-filter for the selection. It especially helps when selecting multiple hits, as in section 5.2. There it can filter out unrelated hits before they get mistakenly added to the track. A hit can be made invisible or less visible ("hidden") if the angle deviation α it would need to connect to the currently selected hit is higher than the specified maximum (see figure 4). Furthermore, using the distance between the two hits and the thickness of the detector layer, a factor for the angle maximum can be calculated. This factor allows us to apply the maximum on a per-layer basis and not as an absolute angle. The angle is calculated between the line from the previous hit to

Figure 4: *Adaptive visibility with an angle of ten degrees. The orange hit is the currently selected hit. Hidden hits are displayed as transparent.*

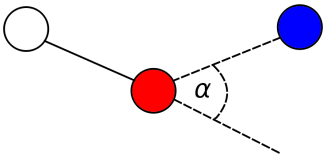

Figure 5: *Angle calculation of adaptive visibility. Red is the currently selected hit and blue is the hit we want to calculate the angle* α *to. The white hit is the previous connection.*

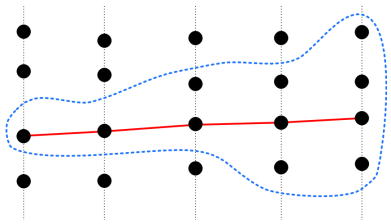

Figure 6: *When selecting multiple hits in one layer, our approach finds the one hit that results in the* straightest path. *The gray dotted lines are the layers and the blue dotted line is the selection (here a lasso selecting 11 hits). The red line is the result of the selection.*

the selected hit and the line that would be newly created (see figure 5). Based on the simulated data and the distribution of their angle changes, a good value for the maximum angle α_{max} can be calculated. α_{max} should be roughly between ten and twenty degrees [PMO*20].

5.4. Straightest Path

When using the multi-selection techniques (see section 5.2), there is a high probability of unrelated hits in the selection. This is due to the 2D nature of the selection, the 3D nature of the detector data and the dense clustering of the hits. Therefore, we calculate the path with the smallest angle deviation, which we call *straightest path*, among the hits of the selection. This allows the users to be less precise with their selection and lets the algorithm take over filtering unrelated hits out (see figure 6).

To find the *straightest path* we first create a graph containing every possible connection between the selected hits in consecutive layers with a weight. This weight is the angle of the resulting line between the two hits and a perpendicular line through the detector layers (see figure 7).

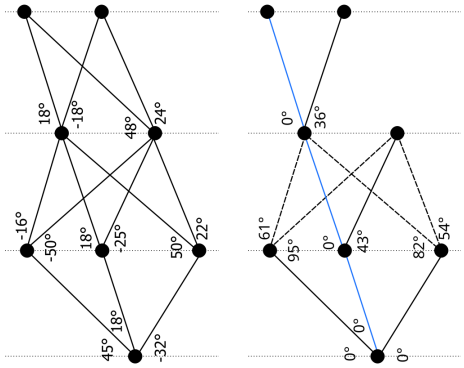

Figure 7: *The created Dijkstra graph is on the left while the result is on the right. Every edge has the absolute angle as weight. In the result the weight is the sum of degrees a particle has to turn to get to the edge. The layers are denoted by the gray dotted lines.*

This graph is the prerequisite for the Dijkstra [Dij59] algorithm that is employed hereafter.

In contrast to a normal Dijkstra algorithm, where the weight of an edge is absolute, our implementation uses a relative weight. This means the weight of an edge is relative to the weight of the preceding edge. So what the algorithm uses as weight is not the one that has been calculated in the previous step, but it is the difference between the absolute weight of the current edge and the absolute weight of the previous edge (see figure 7). All edges that originate in the first layer of selected hits have a weight of zero, as there is no preceding edge that could determine the probability of those edges. If there are multiple hits in the first selected layer, a graph is built for each one of them and the path with the lowest weight among them is chosen as the straightest path. The processing of the graphs can be easily parallelized for a better performance, as they do not interact with each other. This is also included in our implementation. Every multi-selection by a user and the prediction mechanism described in section 5.5 use the straightest path method.

5.5. Prediction

As we now know how to find the *straightest path* among a set of hits, we can also give the user a preview of how the track might continue. Employing the method described in the previous section on the surrounding layers of the already constructed track, we can calculate the most likely continuation of said track. Normally all the hits on the surrounding layers need to be taken into account for the calculation, but using the constraint of the adaptive visibility (see section 5.3) as a pre-filter for the prediction, the number of hits to check can be drastically reduced. After calculating the prediction, the candidate hits should be shown to the users (see figure 8) so they can decide whether the prediction is correct or not. An option to confirm and thus insert the prediction into the current track, e. g., using a keyboard shortcut, can make the workflow faster. In our implementation, we chose the number of prediction layers to be three. Here we try to find a balance between the length of the prediction and the performance. For more details on the choice of the number of layers and performance, see section 7.2.

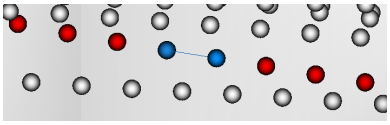

Figure 8: *After selecting hits, a prediction shows how the track could be continued. Blue: Selected track. Red: Prediction.*

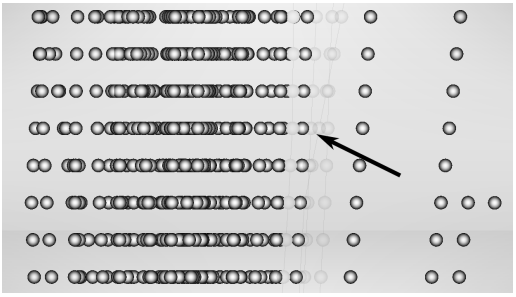

Figure 9: *Hiding tracks with an opacity of 0.1. Because of the surrounding tracks being hidden, other tracks can be isolated and thus rendered better visible. The black arrow points to a now isolated and thus better visible path of particles.*

6. Workflow

Using the above-mentioned methods usually results in the workflow described in the following. First, one tracks the outer paths, as they are normally isolated and can be easily selected. This tracking can be done quickly using multi-selection such as brush, box, or lasso. After a track is done, it can be hidden from view to not occlude other paths (see figure 9). When all the outer paths are done, normally a highly crowded middle section is left. This middle section can be untangled with the stretching method. After stretching the data, one can now rotate to visually isolate a path. For the now isolated path, again one can use multi-selection to track it. Since it is sometimes not possible to fully isolate a path, wrong hits might get into the selection. For this, one can use the option to deselect hits or the undo and redo options. Both are provided by our implementation. Especially in these crowded areas, the prediction can be a good help with highlighting the path and showing the related hits.

As can be seen in the video accompanying this paper, providing keyboard shortcuts for actions, such as scaling the data and inserting the prediction into the current track, saves the user a lot of time. In general, every action that leads to the user needing to click outside of the viewer is also available as a shortcut. Since the users will be using their mouse or digital pen while tracking, all shortcuts are accessible with one hand. Because the methods have their limitations, that will be discussed in section 7.3, there also is an option to enable and disable them individually. Also, parameters for the methods are helpful if the user can edit their values. Such parameters can be the maximum angle for the adaptive visibility or the opacity of the hidden hits. A good value for the opacity is between 0.0 and 0.1. One either does not want to see the hidden hits at all, or one wants to see them slightly to not miss a hit. The scaling factor of the stretching method can also be seen as a parameter, but it normally is not changed before the tracking. It is normally changed interactively while tracking with immediate visual feedback.

7. Results

The results of applying the methods described in this paper are better demonstrated interactively than explained through words or images. Thus, a video, showing the different methods in action, accompanies this paper. It starts with a comparison of the multi-selection types and the prediction. First only on *free tracks*, tracks that can be seen unobstructed, and then also on *occluded tracks*, tracks that are obstructed by other tracks. For a better comparison, it also shows the tracking with single clicks, i. e., without any support by our method. In the video, it is apparent that single clicks are by far the slowest and most tedious method. When tracking an occluded track, it is highly recommended using the adaptive visibility when tracking with single clicks. Else, it is almost impossible to see the right path. The brush selection is by far faster, but the user still has to brush almost directly over the hits. Brushing has the added benefit of being intuitive when using a graphics tablet. But this still requires more precise work than the other selection methods. Especially for occluded tracks, the stretching can help the user to not accidentally brush over the wrong hits. It helps to isolate previously occluded tracks, so the user can see them better. Box and lasso on the other hand do not differ much. If users find at least two hits of the same path, they can select them and after that roughly select the rest of the path. The straightest path algorithm should calculate the right track in most cases. The result of using all methods for a whole data set can be seen in figure 10.

The prediction in itself is not useful when tracking a free (non-occluded) track. Such a track is already visible, so multi-selection would be faster than letting the algorithm figure out a prediction. For the occluded tracks, it gives the user an easy way to build the track step by step.

All methods have been presented to the *Bergen pCT Collaboration* on multiple occasions and were met with great appreciation. Many of the methods have been refined through the feedback of our collaboration partners. Additionally the predicition mechanism was inspired by one of the domain scientists.

7.1. Accuracy of SAPT

In addition to the attached video, we also manually tracked 20 data sets of 100 simulated particles each. If a track in the manual data set is equal to a track in the simulated data, it is considered correct. Comparing every track then creates the accuracy of this manual data set. The average accuracy of all the 20 data sets in this experiment amounts to around 98.77% While we calculated the accuracy of the data sets, we also tracked the time it took to reconstruct the trajectories manually. Tracking 100 simulated particles takes about 17 minutes on average. Since the goal for SAPT is to help ground truth data out of the real detector data, an accuracy close to 100% is needed. Due to SAPT's accuracy, there is about one wrongly tracked trajectory per data set. This error rate is acceptable for creating ground truth data. From the evaluation of the current algorithm for automatic reconstruction by the Bergen pCT Collaboration, we know that the efficiency of tracking 100 simulated particles amounts to about 77% [PMO*20]. This results in about 20 wrongly reconstructed trajectories per data set. Therefore, SAPT's methods reach an accuracy of approximately 1.3 times the accuracy of the algorithmic solution.

As the read-out rate of the detector is not yet determined, we also tested the SAPT methods with a 1000 particles and a 10,000 particles read-out. Here, we tracked the innermost 100 particles, as this is sufficient to test whether manual tracking at this density is possible. While the data set with a 1000 particle read-out can be tracked, the time increases to 30–40 minutes and the accuracy decreases to about 96%. This is still an acceptable result for creating ground truth data. The data set containing the 10,000 particle read-out rate could not be reconstructed in our experiment. While one or two tracks could be isolated through stretching the data, most of the particles are too close to each other to distinguish individual trajectories. More details regarding the described evaluation can be found in [Esc22].

7.2. Performance of Prediction

As mentioned in section 5.5 we chose the prediction mechanism to work on three layers to reach acceptable performance. This choice is based on experiments we performed. We calculated the minimum, maximum, and average execution time of the prediction for one of our data sets (see figure 11). For the calculation, we took every hit in our data of 100 simulated particles and calculated the prediction for 1–8 prediction layers separately. We also evaluated the use of the adaptive visibility as a pre-filter. This was done automatically, as was the tracking of the execution time. Therefore, we base our statistics on the data for around 200 values per layer.

The minimum execution time is negligible, as it is about the same short time for any number of layers. However, it should be noted, using the prediction without the adaptive visibility results in a better best case time due to fewer comparisons and calculations. Due to the worst case and average time, we made the adaptive visibility mandatory for the use of the prediction and chose the number of prediction layers to be three by default. This ensures a calculation time less than a tenth of a second in every case, which does not hinder the user when tracking. Long prediction times would lead users to either not using the prediction or needing to wait a comparatively long time in which they could have used multi-selection for the same result.

The performance of the prediction is dependent on the characteristics of the data set. The more hits are the same layer, especially hits that are close to each other, the bigger the graph for Dijkstra's algorithm gets, and the more paths need to be calculated. If the hits spread over more layers, the prediction is faster. Although we suggest three layers for typical data sets, depending on the input data a different number of prediction layers might be of use.

7.3. Generalization and Limitations

Most presented methods, i. e. adaptive visibility, *straightest path* and prediction, rely on a layered structure of the data. This, however, does not constrains them solely to our pCT data. All of these methods generalize well to any type of layered point cloud. A specifically important example of such data is time-based data, where a time step or time slice would correspont to a layer.

Depending on the input data, limitations of the methods described in the paper become more or less noticeable. First, if particles typically take sharp turns and the corresponding hits thus have a high deviation in angle compared to the preceding layer, neither the prediction nor the *straightest path* nor the adaptive visibility methods will give satisfactory results. In our application case this rarely happens and in the occasion can be easily recognized by the user.

This is also a limitation in existing tracking methods. E. g., in one

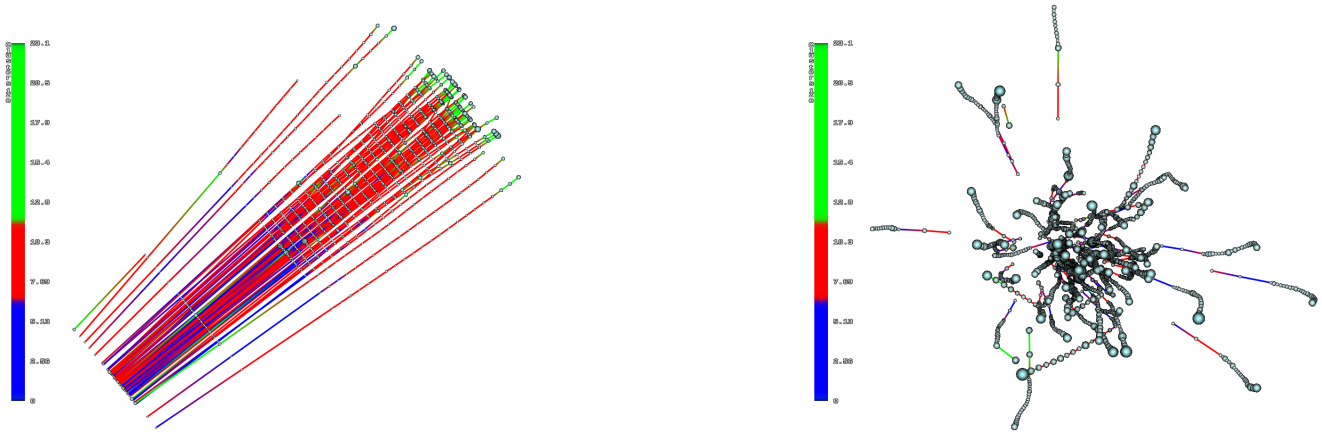

Figure 10: *Result of manually tracking 100 simulated particles creating 87 trajectories. Two views of the same tracking (left: side, right: front) are shown. The discontinuous color scale has been chosen for illustration purposes only.*

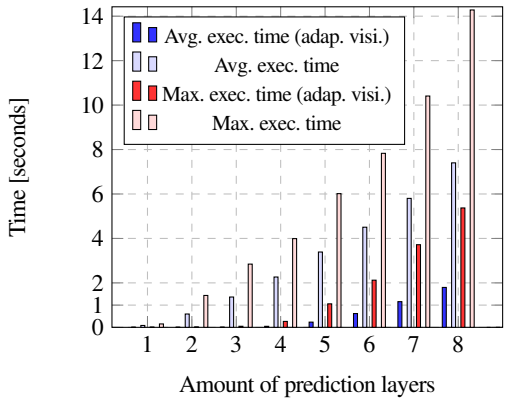

Figure 11: *Performance comparison of prediction with/without adaptive visibility filtering. Data was created using 100 particles using 43 (2 front-tracker and 41 detector) layers resulting in about 2000 primary hits.*

of the algorithms that is also using an angle based method works with a maximum scattering angle [PMO*20]. This angle defines the maximum allowed difference in angle from one hit to another. An optimal value for this angle has to be found, as too low values result in the track ending early and too high values result in wrong hits chosen for the track. That is the benefit of a semi-automatic approach. By having a user direct and supervise the algorithm, minor errors can be corrected on the spot or may be avoided from the start.

Since the straightest path and the prediction methods build a graph over the whole selection, their performance is highly impacted by the number of hits inside the selection. Especially impactful are multiple hits on the same layer. The more hits the selection has on one layer, the more branched the graphs will be. This limitation can be counteracted by using the adaptive visibility as a pre-filter for the shortest path and the prediction method. But the bigger the data set and the more crowded the data, the more this limitation becomes apparent.

8. Conclusions

We presented SAPT, a collection of methods for semi-automatic particle tracking. The benefits of SAPT when compared to straight-forward manual tracking are apparent. Especially in crowded areas where the tracks occlude each other tremendously, the adaptive visibility, prediction and most notably the *straightest path* method highly simplify and accelerate the user's workflow. With the presented methods, the user can be less precise and thus faster while selecting and still obtain the same precise track.

We plan to extend our methods to include the deposited energy, or other relevant physical quantities, of the particle. This could lead to a more precise calculation of the right track among a set of detector hits. Another possible way to enhance our methods would be to employ more automatic tracking algorithms, that could be used as the basis for multi-selection and prediction.

As of fall 2021, an implementation of all SAPT methods is available free and open source in the *OpenWalnut* [EHWS10] software at `http://source.openwalnut.org/`.

Acknowledgements

This work was partly supported by the German federal state of Rhineland-Palatinate (*Forschungskolleg SIVERT*). Many thanks go to the undergraduate students of team *SIVERT-Vis* who developed an early prototype during the *team-oriented project* course at Hochschule Worms. We thank the OpenWalnut community for providing their software as the basis for implementing the presented techniques. We thank the *Bergen pCT Collaboration* for providing data and consulting.

References

[ABC*20] AMROUCHE S., BASARA L., CALAFIURA P., ESTRADE V., FARRELL S., ET AL.: The tracking machine learning challenge: Accuracy phase. In *The NeurIPS '18 Competition* (Cham, 2020), Escalera S., Herbrich R., (Eds.), Springer International Publishing, pp. 231–264. doi:10.1007/978-3-030-29135-8_9. 1, 2

[AKSP*20] ALME J., KEIDEL R., SEIME PETTERSEN H. E., PIERSIMONI P., RÖHRICH D., ET AL.: A high-granularity digital tracking

calorimeter optimized for proton ct. *Frontiers in Physics 8* (2020), 460. doi:10.3389/fphy.2020.568243. 1

[CER21] CERN: The large hadron collider, 2021. URL: https://home.cern/topics/large-hadron-collider. 2

[CFG*18] CALAFIURA P., FARRELL S., GRAY H., VLIMANT J.-R., INNOCENTE V., ET AL.: TrackML: A high energy physics particle tracking challenge. In *2018 IEEE 14th International Conference on e-Science (e-Science)* (2018), IEEE, p. 344. doi:10.1109/eScience.2018.00088. 1

[CK76] CORMACK A. M., KOEHLER A. M.: Quantitative proton tomography: preliminary experiments. *Phys. in Med. & Biology 21*, 4 (July 1976), 560–569. doi:10.1088/0031-9155/21/4/007. 1

[Cor63] CORMACK A. M.: Representation of a function by its line integrals, with some radiological applications. *Journal of Applied Physics 34*, 9 (1963), 2722–2727. doi:10.1063/1.1729798. 1

[Dij59] DIJKSTRA E. W.: A note on two problems in connexion with graphs. *Numerische Mathematik 1*, 1 (1959), 269–271. doi:10.1007/BF01386390. 3, 5

[EHWS10] EICHELBAUM S., HLAWITSCHKA M., WIEBEL A., SCHEUERMANN G.: OpenWalnut - an open-source visualization system. In *Proc. of 6th High-End Vis. Workshop* (Dec. 2010), Benger W., Gerndt A., Su S., Schoor W., Koppitz M., Kapferer W., Bischof H.-P., Pierro M. D., (Eds.), pp. 76–78. 7

[Esc22] ESCHBACH R.: Immersive visualization and semi-automatic reconstruction of particle tracks in the context of pCT. Bachelor's Thesis, Hochschule Worms University of Applied Sciences, 2022. 6

[ESF21] EMAMI N., SEDAEI Z., FERDOUSI R.: Computerized cell tracking: Current methods, tools and challenges. *Visual Informatics 5*, 1 (Mar. 2021), 1–13. doi:10.1016/j.visinf.2020.11.003. 2, 3

[Gro87] GROTE H.: Pattern recognition in high-energy physics. *Reports on Progress in Physics 50*, 4 (Apr. 1987), 473–500. doi:10.1088/0034-4885/50/4/002. 1

[Kal60] KALMAN R. E.: A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering 82*, Series D (1960), 35–45. doi:10.1115/1.3662552. 3

[KK18] KÖSTER M., KRÜGER A.: Screen space particle selection. In *Proceedings of the Conference on Computer Graphics & Visual Computing* (Goslar, DEU, 2018), CGVC '18, Eurographics Association, pp. 61–69. doi:10.2312/cgvc.20181208. 3

[PAB*17] PETTERSEN H. E. S., ALME J., BRINK A., CHAAR M., FEHLKER D., ET AL.: Proton tracking in a high-granularity digital tracking calorimeter for proton ct purposes. *Nuclear Instruments and Methods in Physics Research Section A Accelerators Spectrometers Detectors and Associated Equipment 860* (07 2017), 51–61. doi:10.1016/j.nima.2017.02.007. 1, 2

[PAG*21] PETTERSEN H. E. S., AEHLE M., GARTH C., GAUGER N. R., KEIDEL R., ET. AL: Investigating particle track topology for range telescopes in particle radiography using convolutional neural networks. *Acta Oncologica 60*, 11 (2021), 1413–1418. PMID: 34259117. doi:10.1080/0284186X.2021.1949037. 2

[PMO*20] PETTERSEN H. E. S., MERIC I., ODLAND O., SHAFIEE H., SØLIE J., RÖHRICH D.: Proton tracking algorithm in a pixel-based range telescope for proton computed tomography. *arXiv preprint arXiv:2006.09751* (06 2020). URL: https://arxiv.org/abs/2006.09751v1. 2, 4, 6, 7

[PRK88] PÜHLHOFER F., RÖHRICH D., KEIDEL R.: Track recognition in digitized streamer chamber pictures. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 263*, 2 (1988), 360–367. doi:10.1016/0168-9002(88)90971-0. 1

[PVH*03] POST F. H., VROLIJK B., HAUSER H., LARAMEE R. S., DOLEISCH H.: The state of the art in flow visualisation: Feature extraction and tracking. *Computer Graphics Forum 22*, 4 (2003), 775–792. doi:10.1111/j.1467-8659.2003.00723.x. 3

[SACF*12] SCHINDELIN J., ARGANDA-CARRERAS I., FRISE E., KAYNIG V., LONGAIR M., ET AL.: Fiji: an open-source platform for biological-image analysis. *Nature Methods 9* (June 2012), 676–682. doi:10.1038/nmeth.2019. 3

[SBK*04] SADROZINSKI H.-W., BASHKIROV V., KEENEY B., JOHNSON L., PEGGS S., ET AL.: Toward proton computed tomography. *IEEE Transactions on Nuclear Science 51*, 1 (2004), 3–9. doi:10.1109/TNS.2003.823044. 1

[SF10] STRANDLIE A., FRÜHWIRTH R.: Track and vertex reconstruction: From classical to adaptive methods. *Reviews of Modern Physics 82*, 2 (May 2010), 1419–1458. doi:10.1103/revmodphys.82.1419. 2

[Sik17] SIKLÉR F.: Combination of various data analysis techniques for efficient track reconstruction in very high multiplicity events. *EPJ Web of Conferences 150* (2017), 00011. doi:10.1051/epjconf/201715000011. 2

[TPS*17] TINEVEZ J.-Y., PERRY N., SCHINDELIN J., HOOPES G. M., REYNOLDS G. D., ET AL.: Trackmate: An open and extensible platform for single-particle tracking. *Methods 115* (2017), 80–90. Image Processing for Biologists. doi:10.1016/j.ymeth.2016.09.016. 3

[vdE02] VAN DER EIJK R. M.: *Track reconstruction in the LHCb experiment*. PhD thesis, Amsterdam U., 2002. Report number: CERN-THESIS-2002-032. URL: https://hdl.handle.net/11245/1.198399. 3

[WPVH13] WIEBEL A., PREIS P., VOS F. M., HEGE H.-C.: 3D Strokes on Visible Structures in Direct Volume Rendering. In *EuroVis - Short Papers* (2013), Hlawitschka M., Weinkauf T., (Eds.), The Eurographics Assoc. doi:10.2312/PE.EuroVisShort.EuroVisShort2013.091-095. 3

[WVFH12] WIEBEL A., VOS F. M., FOERSTER D., HEGE H.-C.: WYSIWYP: What you see is what you pick. *IEEE TVCG 18*, 12 (Dec. 2012), 2236–2244. doi:10.1109/TVCG.2012.292. 3

[YEII12] YU L., EFSTATHIOU K., ISENBERG P., ISENBERG T.: Efficient structure-aware selection techniques for 3d point cloud visualizations with 2DOF input. *IEEE TVCG 18* (12 2012), 2245–2254. doi:10.1109/TVCG.2012.217. 3

[YEII16] YU L., EFSTATHIOU K., ISENBERG P., ISENBERG T.: CAST: Effective and efficient user interaction for context-aware selection in 3D particle clouds. *IEEE TVCG 22*, 1 (Jan. 2016), 886–895. doi:10.1109/TVCG.2015.2467202. 3

[YMS*21] YAN L., MASOOD T. B., SRIDHARAMURTHY R., RASHEED F., NATARAJAN V., HOTZ I., WANG B.: Scalar field comparison with topological descriptors: Properties and applications for scientific visualization. *Comp. Graph. Forum 40*, 3 (2021), 599–633. doi:10.1111/cgf.14331. 3

Appendix A: Members of the Bergen pCT Collaboration

Johan Alme a), Gergely Gábor Barnaföldi c), Rene Barthel d), Viatcheslav Borshchov l), Anthony van den Brink d), Mamdouh Chaar a), Viljar Eikeland a), Georgi Genov a), Ola Grøttvik a), Håvard Helstrup e), Ralf Keidel k), Chinorat Kobdaj m), Shruti Mehendale a), Ilker Meric e), Odd Harald Odland a), b), Gábor Papp g), Thomas Peitzmann d), Helge Egil Seime Pettersen b), Pierluigi Piersimoni a), Attiq Ur Rehman a), Matthias Richter h), Dieter Röhrich a), Andreas Tefre Samnøy a), Joao Seco i), j), Hesam Shafiee a), f), Arnon Songmoolnak a), m), Jarle Rambo Sølie a), f), Ganesh Tambave a), Ihor Tymchuk l), Kjetil Ullaland a), Monika Varga-Kofarago c), Lennart Volz i), j), Boris Wagner a), RenZheng Xiao a), n, Shiming Yang a), Christoph Garth o), Nicolas R. Gauger p), Steffen Wendzel k), Alexander Wiebel k), Tobias Kortus k), Alexander Schilling k), Raju Ningappa Mulawade k), Sebastian Zillien k), Max Aehle p), Viktor Leonhardt o)

a) Dep. of Physics and Technology, University of Bergen, 5020 Bergen, Norway
b) Dep. of Oncology and Medical Physics, Haukeland University Hospital, 5021 Bergen, Norway
c) Dep. for Th. Phys., Heavy-Ion RG, Wigner RCP of the Hung. Acad. of Sci., Budapest, Hungary
d) Institute for Subatomic Physics, Utrecht University/Nikhef, Utrecht, Netherlands
e) Dep. of Comp., Math. and Phys., West. Norway University of Applied Science, Bergen, Norway
f) Dep. of Electrical Engineering, West. Norway University of Applied Sciences, Bergen, Norway
g) Institute for Physics, Eötvös Loránd University, 1/A Pázmány P. Sétány, Budapest, Hungary
h) Dep. of Physics, University of Oslo, Oslo, Norway
i) Dep. of Biomed. Phys. in Rad. Oncology, German Cancer Research Center, Heidelberg, Germany
j) Dep. of Physics and Astronomy, Heidelberg University, Heidelberg, Germany
k) Center for Technology and Transfer, University of Applied Sciences Worms, Worms, Germany
l) LTU, Kharkiv, Ukraine
m) Institute of Science, Suranaree University of Technology, Nakhon Ratchasima, Thailand
n) College of Mechanical & Power Engineering, China Three Gorges University, Yichang, China
o) Scientific Visualization Lab, TU Kaiserslautern, Kaiserslautern, Germany
p) Scientific Computing, TU Kaiserslautern, Kaiserslautern, Germany