

Title

Title

Master thesis  
submitted by

Nils Braun

DATUM

Institut of Experimental Nuclear Physics (IEKP)

Advisor: Prof. Dr. Michael Feindt  
Coadvisor: Prof. Dr. Ulrich Husemann

Editing time: November 2014 – November 2015



Title

Title

Masterarbeit  
eingereicht von

Nils Braun

DATUM

Institut für experimentelle Kernphysik (IEKP)

Referent: Prof. Dr. Michael Feindt  
Korreferent: Prof. Dr. Ulrich Husemann

Bearbeitungszeit: November 2014 – November 2015



---

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

**Karlsruhe, DATUM**

.....  
(Nils Braun)

Masterarbeit angenommen.

**Karlsruhe**

.....  
(Prof. Dr. Michael Feindt)

# Contents

<b>1. Introduction</b>	<b>1</b>
1.1. Belle II and the High Luminosity Frontier . . . . .	1
1.2. Track Finder for Particle Physics Experiments . . . . .	1
<b>2. Experimental Setup</b>	<b>3</b>
2.1. SuperKEKB and the Belle II Detector . . . . .	3
2.2. The Tracking Detectors . . . . .	3
<b>3. Track Finder Theory and Multivariate Classification</b>	<b>7</b>
3.1. The Belle II Analysis Framework ( <b>basf2</b> ) . . . . .	7
3.2. Working Principle of the implemented CDC Track Finder in <b>basf2</b> . . . . .	8
3.3. The used Figures Of Merit . . . . .	10
3.4. Multivariate Classification . . . . .	11
<b>4. Track Finding in basf2</b>	<b>13</b>
4.1. The <b>TrackFindingCDC</b> Package . . . . .	13
4.2. The Background Hit Finder . . . . .	13
4.3. Improvements on the Legendre Track Finder . . . . .	14
4.4. Improvements on the Stereo Hit Finder . . . . .	15
4.5. The <b>SegmentTrackCombinerModule</b> . . . . .	15
4.6. Further Approaches . . . . .	15
4.7. Additional software changes . . . . .	15
<b>5. Analysis of the implemented Track Finder</b>	<b>17</b>
5.1. Comparison of the two track finder . . . . .	17
5.2. Tracking efficiency with different input parameters . . . . .	17
<b>6. Momentum estimation of slow particles with the ADC in the VXD</b>	<b>19</b>
6.1. Prestudies . . . . .	19
6.2. Incooperation in the helix fit . . . . .	19
<b>7. Summary</b>	<b>21</b>
<b>8. Acknowledgement</b>	<b>23</b>
<b>A. Listings</b>	<b>25</b>





# 1. Introduction

## 1.1. Belle II and the High Luminosity Frontier

Standard Model, Belle discovery, CKM, how Belle II will help, new Physics

## 1.2. Track Finder for Particle Physics Experiments

The purpose of track finding and why it is that important to be as good as possible Possible inefficiencies Background

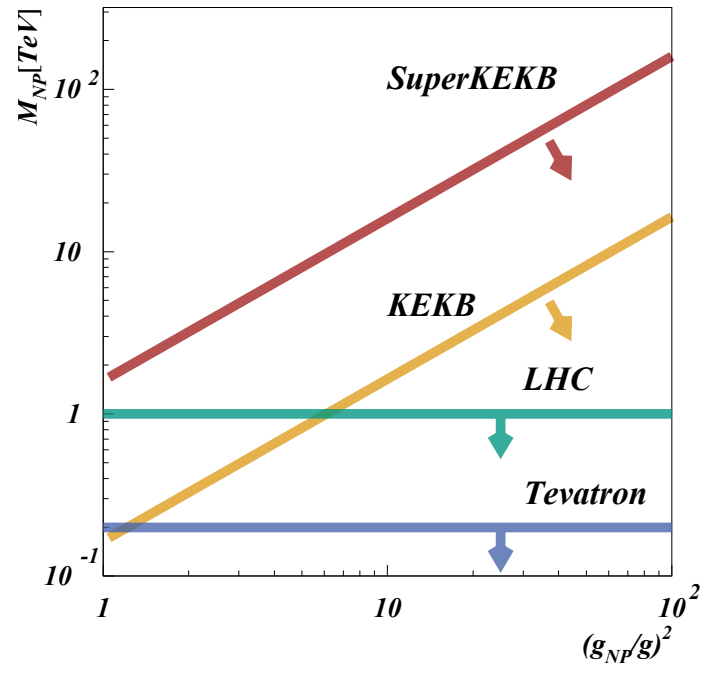


Figure 1.1.: [1]

## 2. Experimental Setup

In the following sections the experimental setup for this thesis - the Belle II detector and the SuperKEKB collider - are described as far as it is necessary for the tracking software. Detailed information can be found elsewhere [1].

### 2.1. SuperKEKB and the Belle II Detector

The future Belle II is currently being build at the KEK high energy research facility in Tsukuba, Japan. It is a general purpose  $4\pi$  detector for high energy particle experiments. A scheme of the whole detector with the planned measuring devices can be found in figure 2.1.

The Belle II detector is used for measuring the electron positron collisions induced by the SuperKEKB collider. As its predecessor KEKB its center of mass energy is tuned in such a way that it lays near the  $\Upsilon(4S)$  resonance. This beam energy is chosen that way because a  $\Upsilon(4S)$  particle decays in nearly all cases into a pair of B-mesons. The daughter particles of these B mesons open a window to a very wide window of physical studies like CP violation or B-meson spectroscopy. The energy of the colliding electrons and positrons is asymmetric which leads to a slightly asymmetric collision as well. This fact is exploited in measuring the decay length of the two formed B-Mesons - one of the main ingredients for quantifying the CP violation in the B-Meson system. The usage of electrons and their antiparticles for collisions leads to very clean events compared to other collision partners like protons or even gold atoms. Clean means in this context a reduced number of created decay products (only 11 charged tracks on average) and more or less no pile up (only one collision occurring at the same time) are expected. These circumstances can be used for high precision measurements on the formed particles.

### 2.2. The Tracking Detectors

As every general purpose high energy particle physics detector the Belle II detector is composed of several single measuring devices - each one with a different task. For collecting as many information on the physical process in the collision as many information on the created decay products are needed. These include

- (1) Identification of the created particles (for example by their mass),
- (2) Measurement of the kinematic properties (like momentum and energy) of the charged particles,
- (3) Resolution of their origin position - also called their vertex.

The tracking detectors described in this thesis are mainly for recording the momenta of the particles as well as their vertex position. Further more special applications are identification of the particle type by their energy loss.

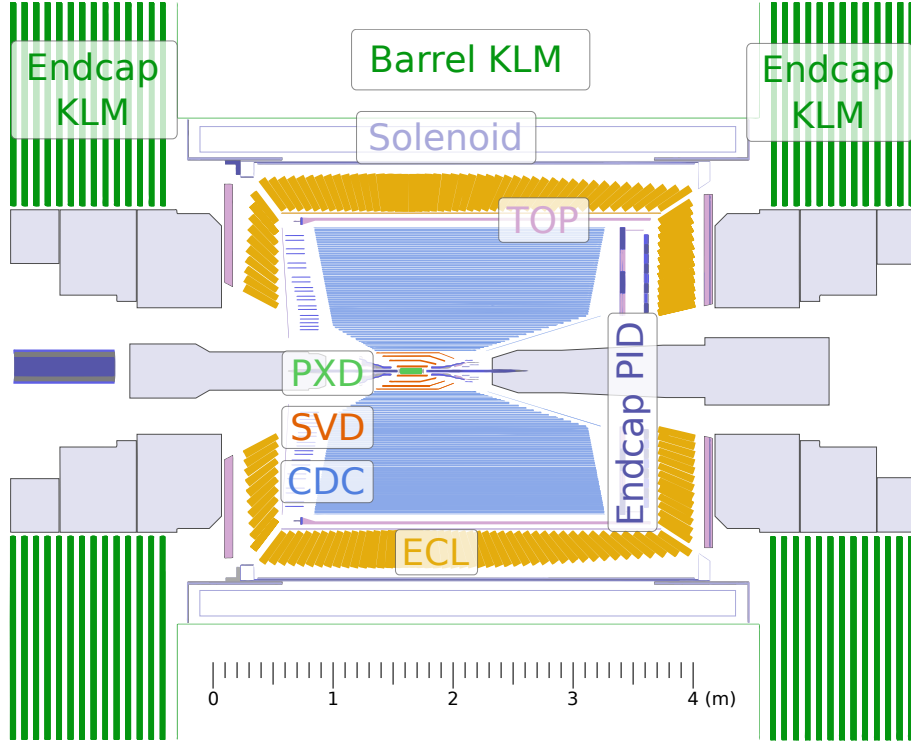


Figure 2.1.: Scheme showing the planned Belle II detector in top view. The single measuring devices are shown with their names. Some more information on the tracking devices can be found in the text. Taken from [2].

The measuring principle of the momentum and also the vertex is to let the charge particles fly through a sensor region with space-resolved measurement and obtain their full path through the detector - a so called track - by combining the different location information - the so called hits. By applying a magnetic field - in Belle II it has a nominal field strength of  $B = 1.5 \text{ T}$  parallel to the beam axis - the  $z$  axis in the coordinate system of the detector - the charged particles are curled. By measuring the curvature of the curled tracks an estimation of the momentum in the plane perpendicular to the beam axis - the  $r$ - $\phi$ -plane - can be made. The remaining  $z$ -direction parallel to the beam axis can be computed with the angle of the track to the beam axis.

There exist many different ways to measure the position of a charge particle. Each of them exploits the charge of the particles and their interaction with materials which makes these techniques unusable for neutral particles. In the following two of the used ways in Belle II are described in more detail.

### 2.2.1. CDC

CDC is an acronym for Central Drift Chamber and is the largest tracking detector in Belle 2. The CDC is a typical gas chamber detector with 14336 sense wires arranged in 56 layer and 9 superlayers. These wires are strained mostly parallel to the beam axis in a cylindrical tank flooded with gas. The gas is a mixture of 50 % helium and 50 % ethane. For more information on gas chamber detectors see for example [3]. Each charged particle produced in the collision of the electron and positron interacts with the gas and can ionize the helium atoms. The energy deposit of a charged particle is given by the Bethe formula [4]

$$-\left\langle \frac{dE}{dx} \right\rangle = \frac{4\pi n z^2}{m_e c^2 \beta^2} \cdot \left( \frac{e^2}{4\pi\epsilon_0} \right)^2 \left[ \ln \left( \frac{2m_e c^2 \beta^2}{I(1 - \beta^2)} \right) - \beta^2 \right] \quad (2.1)$$

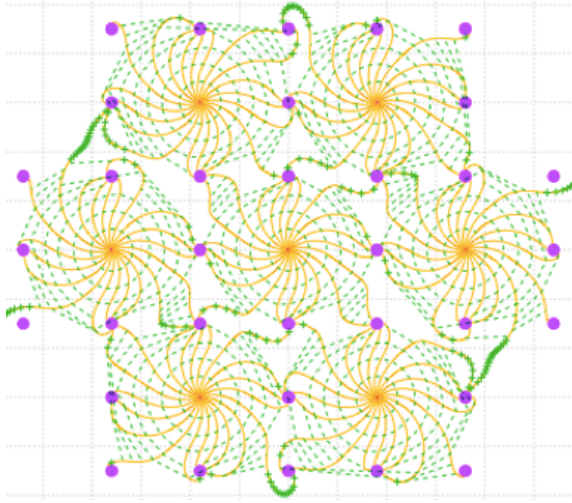


Figure 2.2: Simulation of the electron field in an extract of the wires in the central drift chamber. The violet circles depict the field wires whereas the red small points are the sense wires. The yellow paths are examples for the drift paths of ionized electrons. This simulation includes the distortion by the magnetic field. Taken from [5].

which describes the average energy per travel length. This energy is transferred to the shell electrons of the helium gas atoms. As the ionization energy of helium is very low (about 20 eV) compared to the typical particle energies (up to several 100 eV) the helium atoms release their electrons very easily. These free electrons get accelerated in the electric field induced between the sense wires and field wires with a high negative voltage attached. An instructive simulation of the typical electronic field distribution between 7 sense wires and 34 field wires can be found in figure 2.2. As the electric field near the sense wires increases with  $\propto 1/r$  the free electrons gain more and more energy so that they can ionize helium atoms by themselves as well. This whole flood of free charge can be detected by the electronics attaches to the sense wires and produce a sharp peak there. The more immobile helium atomic cores get absorbed later by the field wires. Because the free electrons have to drift through the gas the measured peak has a delay up to 500 ns for the largest drift cells to the initial charged particle passage. With analysis the time and location information of each measured peak the path of the charged particle through the CDC can be reconstructed. A typical event display of the simulated passage of one charge pion with 1 GeV can be seen in figure ???. Each wire and each drift cell is almost rotationally symmetric. Therefore there is no way in obtaining information on the direction of the electron flood hitting the sense wire. Therefore each hit produced a so called drift circle rather than a single hit point. These drift circled can also be seen in the figure.

If all CDC wires were strained parallel to the beam axis there were no way in measuring the angle between the charged particles and the beam axis. That is the reason why some of the wires are installed with a small tilting angle. These wires are called stereo wires in contrast to the axial wires without tilting angle. A measured hit on the stereo wires does not give information neither on the position along the beam axis nor in the layer parallel to it. Only the information collected among all axial and stereo hits lead to a correct estimation of the momentum in all directions. A sketch of some axial and stereo wires sharing the same distance from the beam axis - also called a layer of wires - are shown in 2.4.

### 2.2.2. VXD

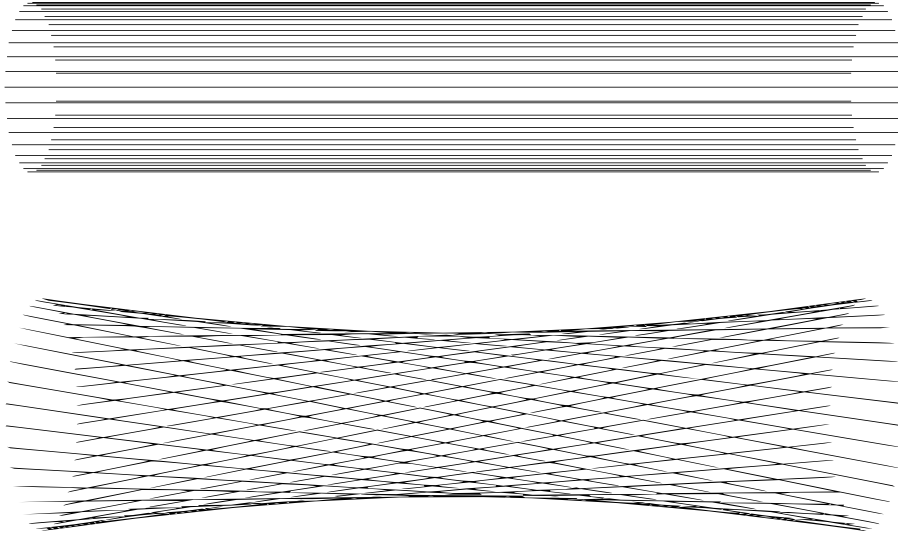


Figure 2.3.: Drawing sketching the axial (top) and stereo wires in the CDC. The skewing against the beamline of the stereo (bottom) wires is exaggerated. Taken from [6].

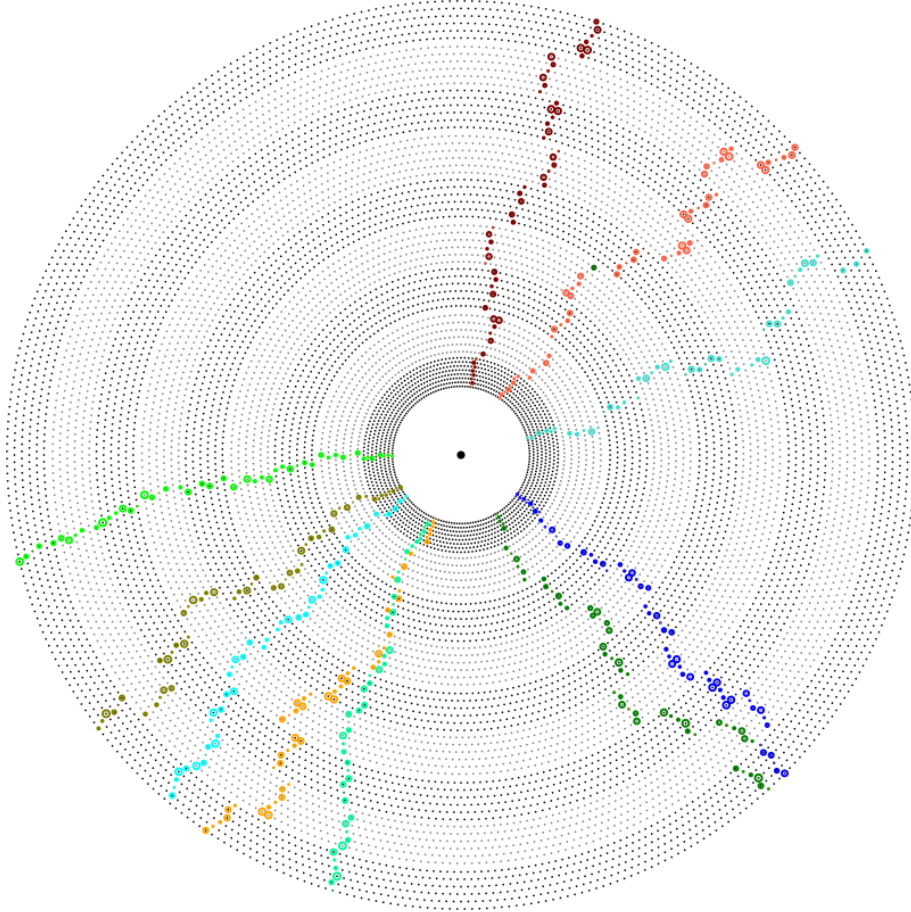


Figure 2.4.: Simulated event of the passage of  $10 \pi^\pm$  with approximately 10 GeV through the CDC shown in a plane perpendicular to the beam axis. The interaction point is drawn as a black circle in the center. Each particle is colored differently to distinguish between them. Only the CDC detector is shown for better visibility. The wires are shown as gray (stereo) or black (axial) points. The hidden wires are depicted with their drift circles. The appearing disruption in the circular paths of the particle arise due to the stereo and axial wires.

## 3. Track Finder Theory and Multivariate Classification

Before explaining the implemented changes to the track finder for the Belle II experiment in more detail, the principles of the Belle II software framework are explained briefly. More information can be found elsewhere [1]. Afterwards common figures of merit for all track finders are explained and discussed and the working principles of the already implemented track finders are illustrated.

### 3.1. The Belle II Analysis Framework (`basf2`)

For simulation, data acquisition, data processing and analysis of the Belle II experiment the Belle Analysis Software Framework 2 (`basf2`) is used. Although - guided by its name - it seems to be build on top of the old software framework used for the Belle experiment it is a complete rewrite of the software using modern programming principles in the coding languages C++ [7] and Python 2.7 [8]. Together with external programming libraries like ROOT [9] or EvtGen [10] that are already on the market this framework builds the base for every software written for the experiment.

The software is divided into several packages - each serving a single purpose or summarizing code for a single detector. Examples for the packages are CDC, SVD or the tracking packaged which is described in more detail in later chapters.

Each usage of the Belle Analysis Software Framework - if it is either a simulation, a reconstruction or an analysis does not matter - consists of processing one or more so called *paths* build with *modules*. These modules perform a dedicated small task like simulating the hard scattering event (the so called `EvtGen` module), writing out data to a root file (the module is called `RootOutput`) or performing a track reconstruction (for example with the module `TrackFinderCDCAutomaton`). The presence, the order and the parameters of the modules are determined in *steering* files written with python. The modules itself can be written in C++ or python.

In these steering files a path is created, filled and passed to the framework which handles loading the corresponding C++ libraries and calling the modules for every event that should be processed. An example of a small steering file for track finding can be found in listing 3.1. Caused by this extremely modular structure not only parallel processing but also debugging of intermediate steps can be performed much easier.

Because many modules need the data produces by other modules before there is a need for intermodular communication. This communication is performed within the framework with the help of the data store. This class as a wrapper around a collection of named `TClonesArrays` from the ROOT library [11] which can store lists of instances of nearly arbitrary C++ classes. It is used widely in the framework to store all sorts of things like

missing

Figure 3.1.: The visualization of the intermodular communication while processing the path implemented with the steering file described in listing 3.1.

More text

the hit information produced by the particles in the simulation or the found tracks after the track finding modules. The modules have read and write access to every so called store array in the data store. A visualization of the data flow between the modules created with the steering file in 3.1 can be found in figure 3.1. The data store can be written to or read from disk using ROOTs own serialization mechanism together with data member dictionaries for the C++ classes created by the C++ interpreter of ROOT called CINT [12].

missing

Listing 3.1: Python steering file to create a typical basf2 path. After loading the needed python libraries the path is created and filled with the modules. In the end this path is processed and for each event the modules are executed in the given order and with their given parameters. For more information on the used modules see their documentations.

## 3.2. Working Principle of the implemented CDC Track Finder in basf2

One part of this work was the improvement and further development of the track finder modules for the CDC tracking detector. Therefore the working principles of the two track finder for this detector are described here briefly. For more information on the first track finder - the legendre track finder - see [13]. More information on the second described track finder - the automaton track finder - can be found in [6].

The general purpose of a track finder algorithm is to partition all measured wire hits into exclusive sets of hits that may come from the same charged particle passing through the detector. It does so by using several assumptions on the charged particle producing the wire hits like the form of their trajectory and therefore the possible patterns of the hits. After fitting a mathematical model of a trajectory to these hits one can gain information on the momentum or the vertex position of this particle. There are several different approaches to find the correct sets of hits which are in the following called tracks.

The reason to have two track finders for the CDC is their different ansatz. The legendre track finder is a so called global track finder whereas the automaton track finder is a local one. A global algorithm uses the information of all wire hits simultaneously. The legendre track finder does this by applying a mathematical projection to the wire positions which should in principle project all hits belonging to the same track onto the same coordinates. A local algorithm however tries to use neighboring wire hits to construct clusters of hits. These clusters are then enlarged by using neighborhood relations until a full track can be found. In the following these two principles are described in more detail.

### 3.2.1. The Legendre Track Finder

The principle of using the legendre transformation for tracking algorithms in high energy particle experiments was first described by Alexopoulos [14]. It uses an extended version of the hough transformation introduced by Paul V.V. Hough in 1962. The algorithm uses



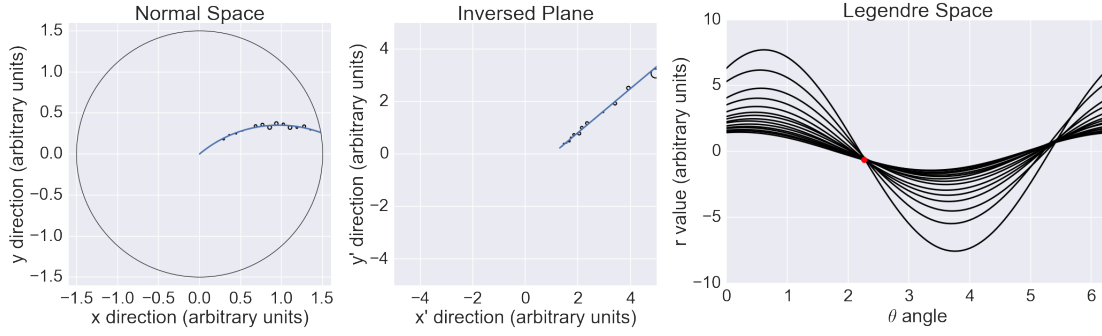


Figure 3.2.: Transformation of some wire hits belonging to the same charge particle (left side) to the inverted plane (middle) and to the legendre space with the sinusoidal functions (right). As described in the text, the circular trajectory is first transformed into lines and then into intersecting sinograms. The intersection corresponds to the parameters of the trajectory and are marked with a red circle.

the fact that each trajectory in the  $r$ - $\phi$ -plane of the detector can be described by a circle - assuming no energy loss - because of the applied magnetic field. In a first approximation one can also assume that each particles comes from the interaction point which is valid for the bigger part of the decay products. Therefore the trajectory in the  $r$ - $\phi$  direction can be described by two parameters: the radius  $R$  of the circle and the angle  $\phi$  between an arbitrary but fixed axis and the tangent to the circle at the interaction point.

The idea is now to transform the  $x$  and  $y$  coordinate pair of every axial wire hit together with the drift length  $R$  with the function

$$x' = \frac{2x}{x^2 + y^2 - R^2} \quad y' = \frac{2y}{x^2 + y^2 - R^2} \quad R' = \frac{2R}{x^2 + y^2 - R^2}$$

$$r = x' \cos(\theta) + y' \sin(\theta) \pm R'$$

into the legendre space as it can be seen in figure 3.2. In the first step, the wire hits are transformed to the inversed plane. With this transformation each circular trajectory through the interaction point is mapped onto a line going through the origin. After that each drift circle is transformed into a pair of sinusoidal functions. This function is constructed in that way to use the fact that each trajectory of a charged particle responsible for a wire hit must touch the drift circle tangentially. Each point on the constructed sinusoidal functions correspond to one possible trajectory of a particle which could have created such a hit. The coordinates in the legendre space are related to the two parameters describing the trajectory as mentioned before. There are two sinusoidal functions because the osculation point can be on the far or the near side of the drift circle - the trajectory circle can contain the drift circle or not.

With using the information of a single hit one ends up with an infinite number of trajectory hypothesis. But as a charged particle passed many drift cells until it leaves the CDC detector - in some cases up to 100 hits - several wire hits are created with the same trajectory parameters. As these same parameters correspond to the same point in the legendre space, the sinusoidal functions of the wire hits intersect in this point as it can also be seen in figure 3.2. The task of the legendre algorithm is now to do the transformation of the hit coordinates and find those intersections.

Imperfections due to energy loss and material effects make the sinusoidal function not interact in one single point but rather in smeared area. To copy with this problem but still find the intersections with a good performance a peak search in the legendre space is

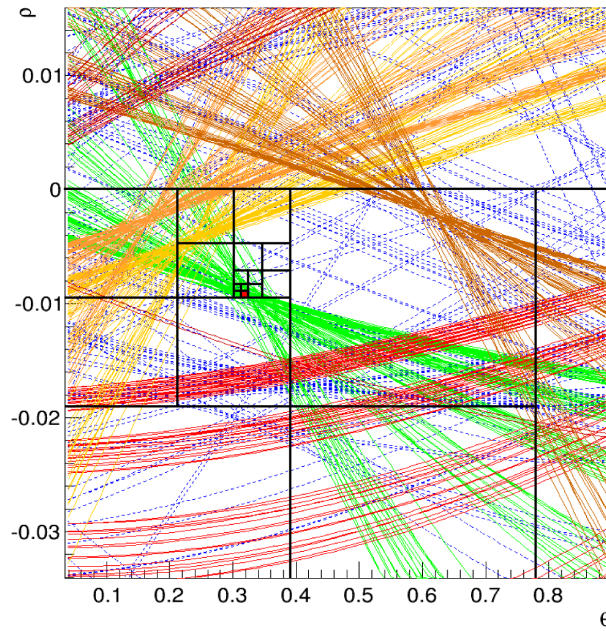


Figure 3.3.: Depiction of a quad tree search to find the bin with the most interceptions of sinograms shown in red here. Only the first round of the search is shown for better visibility. Each color shows sinograms that belong to the same charged particle. Taken from [?].

applied. The legendre space is divided into small bins. For each bin the number of sinusoidal functions passing this area is counted. The bin with the highest weight is assumed to be the bin with the highest number of sinusoidal intersections. From the wire hits contributing to this bin a new track is created and the search is repeated until a threshold in the bin entry is undercut. As the legendre space is mostly empty this procedure can be further improved in performance by refining the bin division from very coarse bin sizes to finer ones only for those bins which have a certain amount of sinusoidal functions in them. Because these bins are divided into 4 subbins the concept is also called a quad tree search. The whole search is depicted in figure 3.3.

#### 3.2.1.1. Stereo Hit Finding

#### 3.2.2. The Automaton Track Finder

Clusterizer, Automaton-Principle

### 3.3. The used Figures Of Merit

For testing and developing and also for later usage in the experiment setup we need to compile numbers from the implemented track finder algorithms to show how well they work. There are three different classes of figures of merit to describe a track finder. All three classes listed here are described in more detail below. The three classes are:

- the efficiency (like the hit efficiency or the finding efficiency, also split up for different particle types, momentum regions or areas in the detector)
- the error rate (like the clone or fake rate)
- the computational performance (like timing and memory consumption)

The last class should be quite clear and is measured by the basf2-own measuring algorithms. As the tracking is part of the online reconstruction and may be once adopted for the high level trigger, the timing performance is very important.

The other two classes can best be described with the algorithm how they get computed. It starts with a full Monte Carlo simulation of generic BB events with the full detector simulation afterwards. The created hits can then be parted into distinct sets - each describing a simulated tracks, called MCTrackCand (they are called track candidates to fit the naming convention of the track finder). Only those tracks are kept as a MCTrackCand, that have at least 3 hits in the CDC - otherwise they can not be fitted and even if there were a chance to find them via track finding, they could not be used for physics. After that, the simulated hits with the stripped MC information are used for track finding with the method to be evaluated. This can be done easily as the simulated hits are transformed in a format that is similar to the format that will be used later for the data coming from the experiment. After the track finder has produced a list of track candidates also, we can use the saved MC information to match tracks from the track finder to tracks from the mc algorithm (also called MC track finder) by counting the number of hits they share. The different cases are depicted and described in table 3.1.

The finding efficiency now describes the rate of MCTrackCands which are labeled matched by their total amount. Building this ratio can also be done for bins in various variables, like perpendicular momentum ( $p_T$ ), angle in the curling plane ( $\phi$ ), number of tracks per event (multiplicity) and many more. A perfect track finder would have a finding efficiency of 100 %. In most of the cases, the finding efficiency drops for tracks in a certain region of these variables - like for low momentum tracks. The hit efficiency is the mean of the ratios between the number of hits in the MCTrackCands matched to a non-fake track candidate to the number of hits in total to this track. Here also a perfect track finder would have a hit efficiency of 100 %. As in most of the cases the track finder loses some outlying hits the hit efficiency drops. The fake and clone rates are just the number of track candidates labeled as fake or clone by the matching algorithm divided by the number of found tracks in total. A perfect track finder would have both number set to 0 %. A high fake rate is caused by a track finder with too loose cuts when putting together single pieces of tracks to a big track or by one which picks up background hits often. A track finder with a high clone rate on the other hand has too harsh cuts and splits up tracks into more than one piece.

### 3.4. Multivariate Classification

Classification, BDTs

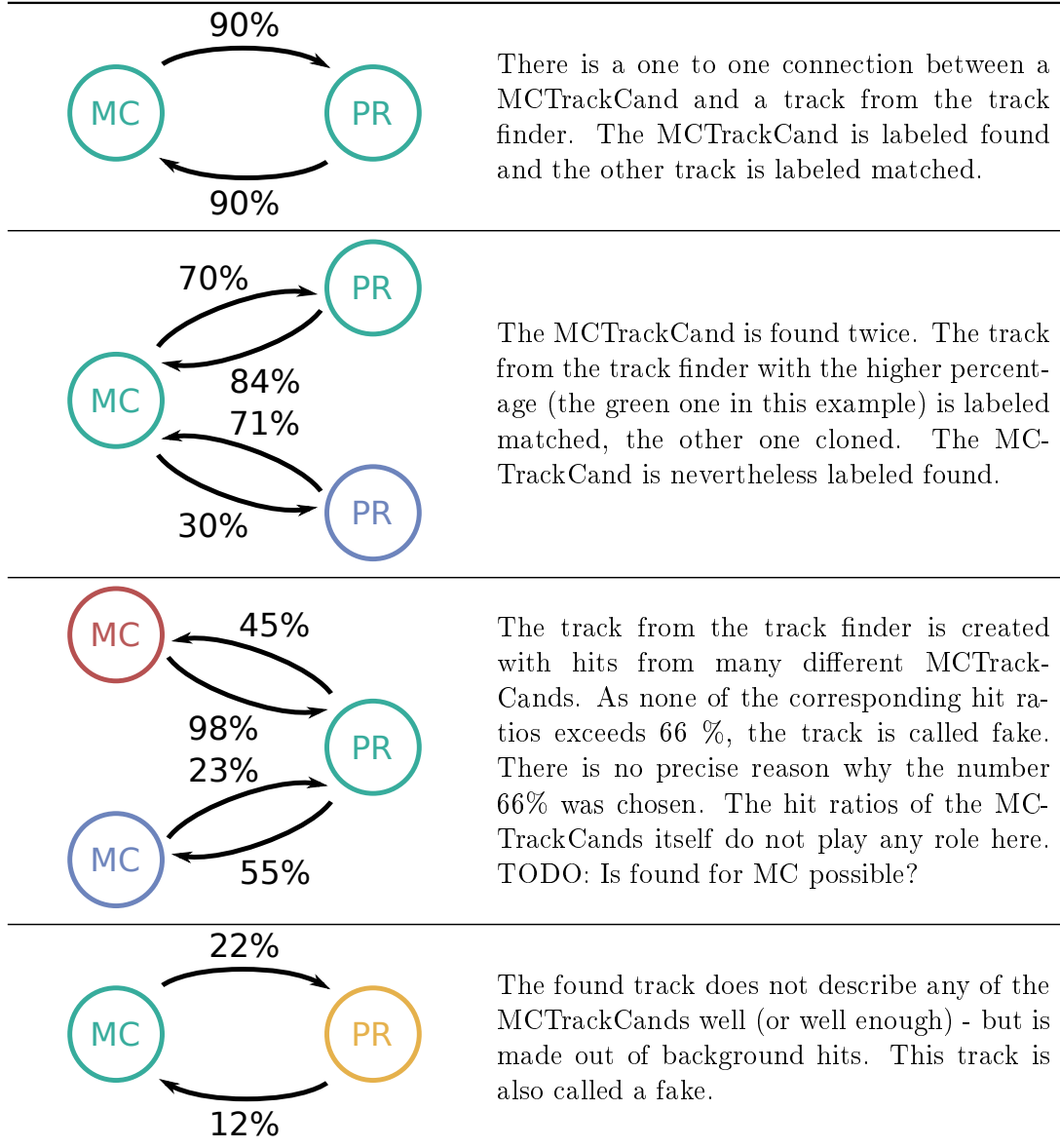


Table 3.1.: This tabular shows the four different cases for the matching between tracks found by the track finder (on the left side of the pictures) and MCTrackCands (shown on the right side). The different colors differentiate between different tracks. The connection between tracks shows that these two tracks share hits. The two percentages on the arrows are the percentages of hits they share in respect to the total number of hits in the MCTrackCand/track candidate from the track finder.

## 4. Track Finding in basf2

After having described the principle of the track finders implemented in **basf2** in the previous chapters we can now go on to show the actual implementation. The two track finders were already written and one of the tasks of this thesis was to put them together and improve them. Because we want to make use of the benefits of both track finders we use the workflow shown in figure 4.1. The shown steps are all described in more detail further down.

### 4.1. The TrackFindingCDC Package

Common code basis

### 4.2. The Background Hit Finder

The beforementioned track finders are tuned to not pick up background hits into a found track candidate or even form a candidate from background hits only. Nevertheless even a single background hit per track can lead to a reduced momentum resolution or can even cause the fit to fail completely. Together with the reduced combinatorics and therefore reduced computing time when throwing away unusable hits there is the need to distinguish between signal and background hits even before the tracking starts. This is the purpose of the BackgroundHitFinder module described in this section.

For deciding if a hit is to be used in the track finder, the module uses implemented filters. As these filters are used widely in the whole CDC tracking framework, they will be described here in more detail. To filter out bad hits, not only the information of one hit but the information of so called clusters of hits is used. These clusters are formed by one of the presteps in the local track finder - but can also be used standalone. They are created by clusterizing all CDCHits in such a way as one cluster includes the maximum number of connected hits. Two hits are called connected if there is no non-fired geometrically wire between them. Figure ?? shows an event display in the  $r$ - $\phi$ -plane with the clusters drawn in different colors. Figure ?? shows the hit purity of ANZahl of the clusters from typical events. The hit purity is computed as the ratio between the number of hits in a cluster which belong to a signal track divided by the number of all hits (including the ones coming from background only). As can be seen clearly in this histogram a cluster

Figure 4.1.: The proposed workflow and combination of the two track CDC finders in basf2. The green boxes refer more or less to one module. The arrows describe parts of the data flow between the modules. For clarity not all necessary parts are shown here.

is either full of background or full of signal hits - for that throwing away a background cluster does not involve deleting hits than are needed for signal tracks.

In every event, each formed cluster is passed to the chosen filter with the chosen filter parameters. This filter can return every number as a result including the C++ `std::nan`, which is chosen as the value for indicating that the cluster should not be used further for track finding as it is marked as background by the filter. Actually, the non NaN numbers returned by the filter are not used further in the following algorithms and are just described here for later reference.

This filter algorithm is a very general one and leaves the possibility open to compare many different filters. For this as well as for any other used filter in the CDC tracking, more or less 5 filters are implemented by default which can be switched by using module parameters. These filters are:

**BaseFilter** is a filter that neglects every item - it is just for basing every other filter on a common ground class and not for usage.

**AllFilter** is the counterpart of the BaseFilter and accepts all items. It can be used for testing purposes to make a filter fully transparent.

**SimpleFilter** is a filter based on self-implemented cuts. The variables to cut depend on the planned usage of the filter and need to be implemented by the programmer.

**TMVAFilter** is based on a trained boosted decision tree. It passed the variables defined in a so called VarSet to the BDT and returns the result to result between 0 and 1. If the result is lower than a certain cut definable on runtime, the TMVAFilter returns `std::nan`. The variables and their calculation can be freely defined by the programmer. In most of the cases the performance of a well-trained TMVAFilter exceeds the one of a simple filter.

**RecordingFilter** returns a constant definable on runtime and can therefore not directly be called a filter. Instead its purpose is to write the same variables as the TMVAFilter uses together with a truth information of every incoming item into a ROOT TNTuple. The output file with this TNTuple can later be used to train the BDT to categorize according to the truth information. This filter can of course only be used on simulated data.

**MCFilter** filters the items according to the truth information compiled with the MC information that also the recording filter uses.

As an example for the mentioned VarSet, the variables for distinguishing the background clusters from the signal clusters is described in table ?? . The truth information used for the BDT training is also described there. The trained BDT has PARAMETERS... EVENTS were used for training.

After the training procedure the corresponding TMVAFilter can be used. The results are summarized in ?? .

Results for Clusters/Hits, Results for Legendre Finder

### 4.3. Improvements on the Legendre Track Finder

#### 4.3.1. The class QuadTreeProcessorTemplate

#### 4.3.2. Postprocessing after the track finding

#### 4.3.3. Results

Timing

## 4.4. Improvements on the Stereo Hit Finder

### 4.4.1. Principle of the StereoQuadTree

### 4.4.2. Results

Timing

## 4.5. The SegmentTrackCombinerModule

### 4.5.1. Principle of the Segment Track Combiner

+ Task

### 4.5.2. Used Filters

### 4.5.3. Results

## 4.6. Further Approaches

### 4.6.1. A quad tree for segments

### 4.6.2. Filter for tracks

### 4.6.3. VXD-Merger??

## 4.7. Additional software changes

### 4.7.1. The class RecoTrack

### 4.7.2. The ipython\_handler





## 5. Analysis of the implemented Track Finder

### 5.1. Comparison of the two track finder

### 5.2. Tracking efficiency with different input parameters

#### 5.2.1. Tracking efficiency with different particle and background types

#### 5.2.2. The TMVA filters



## 6. Momentum estimation of slow particles with the ADC in the VXD

### 6.1. Prestudies

### 6.2. Incooperation in the helix fit



## 7. Summary



## 8. Acknowledgement





## A. Listings



## B. List of Figures

1.1.	[1] . . . . .	2
2.1.	Scheme showing the planned Belle II detector in top view. The single measuring devices are shown with their names. Some more information on the tracking devices can be found in the text. Taken from [2]. . . . .	4
2.2.	Simulation of the electron field in an extract of the wires in the central drift chamber. The violet circles depict the field wires whereas the red small points are the sense wires. The yellow paths are examples for the drift paths of ionized electrons. This simulation includes the distortion by the magnetic field. Taken from [5]. . . . .	5
2.3.	Drawing sketching the axial (top) and stereo wires in the CDC. The skewing against the beamline of the stereo (bottom) wires is exaggerated. Taken from [6]. . . . .	6
2.4.	Simulated event of the passage of $10 \pi^\pm$ with approximately 10 GeV through the CDC shown in a plane perpendicular to the beam axis. The interaction point is drawn as a black circle in the center. Each particle is colored differently to distinguish between them. Only the CDC detector is shown for better visibility. The wires are shown as gray (stereo) or black (axial) points. The hidden wires are depicted with their drift circles. The appearing disruption in the circular paths of the particle arise due to the stereo and axial wires. . . . .	6
3.1.	The visualization of the intermodular communication while processing the path implemented with the steering file described in listing 3.1. . . . .	8
3.2.	Transformation of some wire hits belonging to the same charge particle (left side) to the inverted plane (middle) and to the legendre space with the sinusoidal functions (right). As described in the text, the circular trajectory is first transformed into lines and then into intersecting sinograms. The intersection corresponds to the parameters of the trajectory and are marked with a red circle. . . . .	9
3.3.	Depiction of a quad the search to find the bin with the most interceptions of sinograms shown in red here. Only the first round of the search is shown for better visibility. Each color shows sinograms that belong to the same charged particle. Taken from [?]. . . . .	10
4.1.	Proposed workflow in the CDC tracking . . . . .	13



## C. List of Tables

3.1. Matching routine for compiling the FOM. . . . .	12
--	----



## D. Bibliography





# Bibliography

- [1] **Belle II Collaboration**, T. Abe *et al.*, “Belle II Technical Design Report,” tech. rep., 2010. [arXiv:1011.0352 \[physics.ins-det\]](#).
- [2] C. Pulvermacher, “dE/dx particle identification and pixel detector data reduction for the Belle II experiment,” Master’s thesis, KIT, 2012.  
<https://ekp-invenio.physik.uni-karlsruhe.de/record/48263>.
- [3] C. Grupen, A. Böhrer, and L. Smolík, *Particle Detectors*. Cambridge monographs on particle physics, nuclear physics, and cosmology. Cambridge University Press, 1996.
- [4] H. Bethe and J. Ashkin, “Passage of radiation through matter,” *Experimental Nuclear Physics* **1** no. II, (1953) .
- [5] **KEK High energy accelerator research organisation**, “New electronics tested for Belle II central drift chamber,”. <http://www2.kek.jp/proffice/archives/feature/2010/pdf/BelleIICDCDesign.pdf>.
- [6] O. Frost, “A Local Tracking Algorithm for the Central Drift Chamber of Belle II,” Master’s thesis, KIT, 2013.  
<http://ekp-invenio.physik.uni-karlsruhe.de/record/48172>.
- [7] **International Organization for Standardization (ISO)**, “ISO International Standard ISO/IEC 14882:2014(E) – Programming Language C++,” 2014.  
<https://isocpp.org/std/the-standard>.
- [8] **Python Software Foundation**, “Python 2.7 programming language,” 1991.  
<https://www.python.org/>.
- [9] R. Brun and F. Rademakers, “ROOT – an object oriented data analysis framework,” *Nucl. Instrum. Meth.* **A389** no. 1, (1997) 81–86.
- [10] D. Lange, “The EvtGen particle decay simulation package,” *Nucl. Instrum. Meth.* **A462** (2001) 152–155.
- [11] “ROOT documentation for TClonesArray.”  
<https://root.cern.ch/root/html/TClonesArray.html>.
- [12] “ROOT documentation for CINT.” <https://root.cern.ch/drupal/content/cint>.
- [13] B. Kronenbitter, *Measurement of the branching fraction of  $B^+ \rightarrow \tau^+ \nu_\tau$  decays at the Belle experiment*. PhD thesis, KIT, 2014.  
<https://ekp-invenio.physik.uni-karlsruhe.de/record/48604>.
- [14] T. Alexopoulos, M. Bachtis, E. Gazis, and G. Tsipolitis, “Implementation of the Legendre Transform for track segment reconstruction in drift tube chambers,” *Nucl. Instrum. Meth.* **A592** (2008) 456–462.
- [15] V. Trusov, “Applying Legendre transformation method for Belle II tracking,” *DPG spring meeting* (2014) .