CDF/DOC/SEC_VTX/CDFR/6327
Version 1.0 , February 20, 2003

The Beam Position in Run II

Hartmut Stadie, Wolfgang Wagner, Thomas Müller
Institut für Experimentelle Kernphysik, Universität Karlsruhe

Raymond Culbertson, Hans Wenzel
Fermilab

Abstract

The position of the beam line is an important input for many physics analyses. This note describes the algorithms that are used to find the beam position in Run II. We present some studies of the properties of the beam. In the Appendix we give examples of how to obtain the beam position from the database and describe the programs and scripts used to fill the beam position database.

1 Introduction

The transverse profile of the luminous region inside B0 can be described by Gaussian functions in x and y. The means of these Gaussians define the beam position. The width of these Gaussians varies along the length of the interaction region due to the focusing of the beams. This can be expressed by the following equation [1, 2]:

$$\sigma(z) = \sqrt{\epsilon(\beta^* + \frac{(z - z_{min})^2}{\beta^*})} \tag{1}$$

, where ϵ is the transverse emittance, β^* the amplitude function at the interaction point, and z_{min} the z position of the minimum of the amplitude function. Using the Run IIa design parameters for β^* of 35cm and ϵ[1] of 1.40×10^{-9}m [3] the minimal

[1] using $\epsilon = \frac{\epsilon_{N,95\%}}{6\pi(\beta\gamma)_{rel}}$

transverse width is approximately 22μm. Thus the knowledge of the beam position in x and y gives a good estimate of the primary vertex position of an event. Unfortunately the beams are not centered in the detector and not parallel to the detector z-axis. The dependence of the x and y position of the beam on z can be described by a line:

$$\bar{x} = x_{slope} z + x_0; \bar{y} = y_{slope} z + y_0 \tag{2}$$

The longitudinal luminosity distribution can be approximated as [1, 4]:

$$\frac{dL}{dz} \propto \frac{exp(-(z-z_0)^2/2\sigma_z^2)}{1+\frac{(z-z_{min})^2}{\beta^{*2}}} \tag{3}$$

, where z_0 is the point at which the centers of the proton and anti-proton bunches collide, the cogging point, z_{min} is the z position of the minimum of the amplitude function, β^* is the value of the amplitude function at the minimum and σ_z is the longitudinal bunch length. Using the expected values for Run IIa this equation can be approximated by a Gaussian with a width of 30cm.

With the CDF-II detector the beam position can be measured using tracks found with the *Central Outer Tracker* (COT) as well as with tracks found with the silicon tracking detectors (L00, SVX II, ISL). These two measurements allow to check the global alignment of the two tracking systems and there are different tables in the database for the different measurements. The beam positions obtained with the COT are stored in the CotBeamPosition tables and the beam positions found using silicon tracks in the SvxBeamPosition tables respectively. The beam positions are calculated and stored on a run by run basis. To distinguish between the beam positions obtained for a run using different reconstruction passes, the database tables contain a so called *history code*. A list with all used codes and their meaning can be found on the web pages of the alignment group [5].

2 The Algorithms

We apply the same two algorithms as have been used in Run I [6]. The first algorithm is based on the reconstruction of the primary vertex for each event using tracks. This is the default algorithm for fitting the beam line. The second algorithm is based directly on tracks. Using the correlation of the d_0 and the φ_0 parameter of a track in the parameterization used at CDF [7]. The results of this algorithm are stored in the database as well to allow to cross check the default beam line. To distinguish between the beam line results from different algorithms, the database tables contain an *algorithm code*. This integer value is 20 for the vertex algorithm results and 10 for the results from the D-φ fits.

2.1 Using the Primary Vertex of the Event

The primary vertex is reconstructed using tracks with the *vxprim* module [8]. During production two clones of the *vxprim* module are running.

One module produces a vertex collection named *cot_vertices*. In the vertex fit the track parameters derived from measurements in the Central Outer Tracker (COT tracks) are used. The track requirements for the COT tracks are:

- $p_T > 0.5$ GeV
- $|d_0| < 3.0$ cm
- "standard" COT track
- found by SL tracking algorithm

The other module produces a vertex collection named *svx_vertices*. Only tracks that have been reconstructed using measurements in the silicon detectors are used in the vertex fit. The track requirements for these (silicon) tracks are:

- $p_T > 0.5$ GeV
- $|d_0| < 3.0$ cm
- outside-in tracks:
  - "standard" COT track
  - at least four phi silicon hits
- at least five phi and two stereo silicon hits for stand-alone tracks

From the distributions of the primary vertices one can directly derive the beam position and other beam parameters. Profile plots of the vertex positions in x respectively y over z are used to fit the beam line. Figure 1 shows the profile plots for run 155130.

During production a special module, named *FitBeamModule*, fills the separate profile plots with the primary vertices reconstructed using COT stand-alone tracks and silicon tracks. After production a script combines the plots for the different streams, makes line fits through the data points of the plots, and fills the data base with the results of these beam line fits.

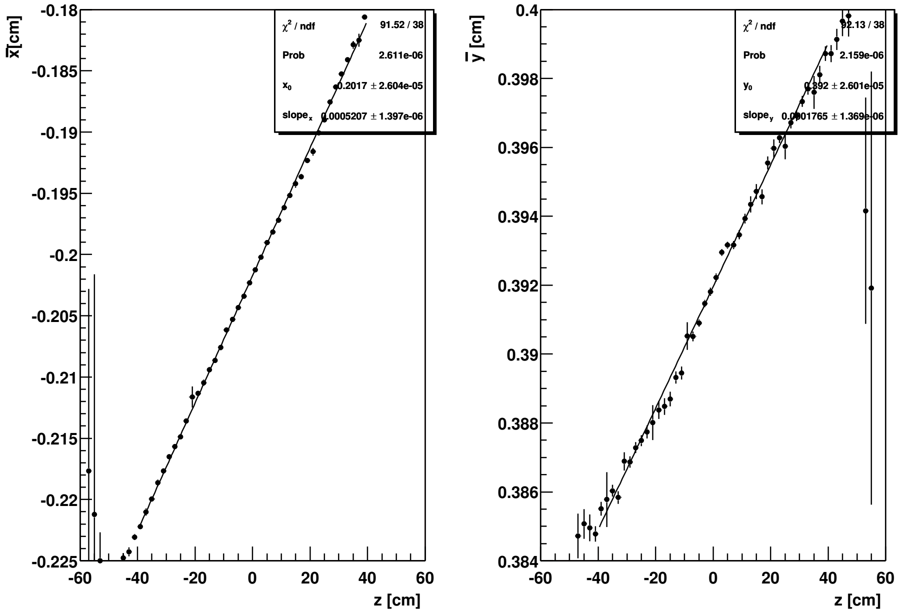

Figure 1: Profile plots of the vxprim vertices reconstructed with silicon tracks for run 155130.

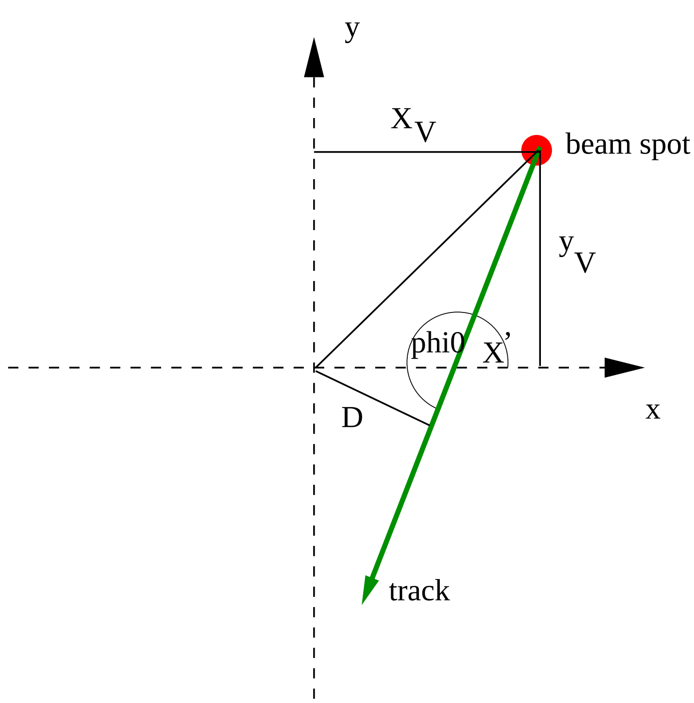

Figure 2: Figure to illustrate the correlation between the impact parameter D and the angle φ of a track.

2.2 Using the D-φ Correlation

As can be seen in Figure 2 the impact parameter D of a track coming from a primary vertex at (x_V, y_V) can be parameterized to first order using the track parameterization of CDF[7]:

$$D(x_V, y_V, \varphi_0) = -x_V \sin\varphi_0 + y_V \cos\varphi_0 \tag{4}$$

Using the z_0 parameter of the track and the beam line parameters $\vec{x} = (x_0, y_0, x_{slope}, y_{slope})$ x_V and y_V can be written as:

$$x_V = x_0 + z_0 x_{slope} \tag{5}$$

$$y_V = y_0 + z_0 y_{slope} \tag{6}$$

Inserting 5 and 6 in 4 gives:

$$D(z_0, \varphi_0) = -x_0 \sin\varphi_0 + y_0 \cos\varphi_0 - z_0 \cdot x_{slope} \sin\varphi_0 + z_0 \cdot y_{slope} \cos\varphi_0 \tag{7}$$

With $\vec{g} = (-\sin\varphi_0, \cos\varphi_0, -z_0 \sin\varphi_0, z_0 \cos\varphi_0)$ this can be written as:

$$D(z_0, \varphi_0) = \vec{x} \cdot \vec{g} \tag{8}$$

For n tracks the χ^2 of the difference between the measured and the calculated impact parameter using the beam position is:

$$\chi^2 = \sum_{i=1}^{n} \left(\frac{D_i - \vec{x} \cdot \vec{g}}{\sigma_i} \right)^2 \text{, with } \sigma_i^2 = \sigma_{D_i}^2 + 2\sigma_{Beam}^2 \tag{9}$$

Minimizing this χ^2 one can obtain the beam position as:

$$\vec{x} = V\vec{b} \text{, with } V^{-1} = \sum_{i=1}^{N} \frac{\vec{g}_i \vec{g}_i^T}{\sigma_i^2} \text{ and } \vec{b} = \sum_{i=1}^{N} \frac{D_i \vec{g}_i}{\sigma_i^2} \tag{10}$$

All COT tracks that pass the following cuts are used in a first fit:

- $p_T > 1.0$ GeV
- $|d_0| < 3.0$ cm
- "standard" COT track
- found by SL tracking algorithm

The track requirements for the Si tracks are:

- $p_T > 1.0$ GeV

- $|d_0| < 3.0$ cm
- outside-in tracks:
  - "standard" COT track
  - at least four phi silicon hits
- at least five phi and two stereo silicon hits for stand-alone tracks

After the fit any track that has a large impact parameter with respect to the calculated beam line result is cut away and the beam position is fitted again using the remaining tracks. This procedure is iterated. Between each iteration the impact parameter cut is tightened until less than 60% of the starting tracks survive. The values of $\vec{b}$ and V are stored in a histogram. As for the vertices the histograms from all the data files of a run are combined and the fit result and its covariance matrix are calculated using formula 10. The result is put into the database.

3 Stability of the Beam Position

We investigate all beam line entries for the silicon between the runs 138425 and 147724. The entries have been made with the 4.5.2 production pass. Figure 3 shows

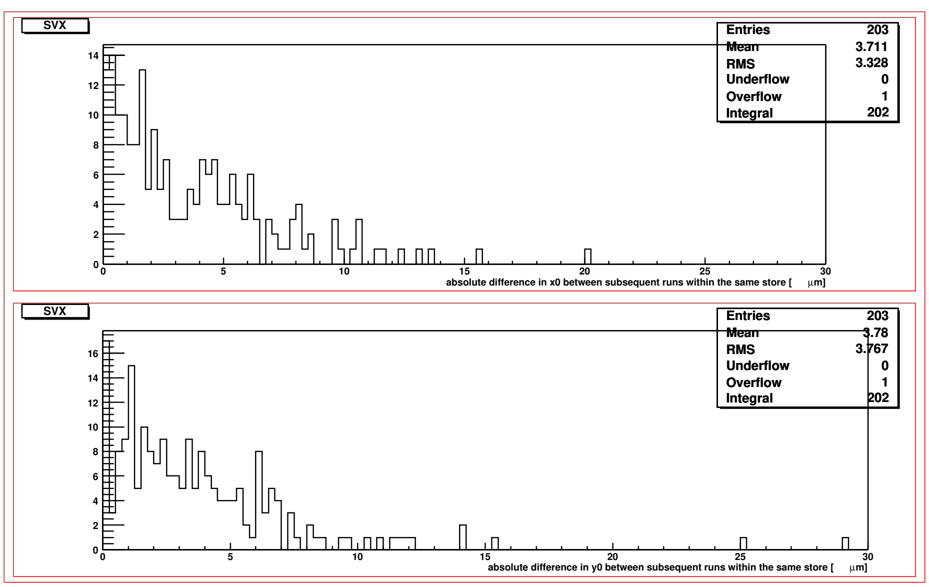

Figure 3: Absolute difference in the x_0 (top) and y_0 (bottom) of the silicon beam line for subsequent runs within the same store.

the absolute difference in the x_0 and y_0 of the beam lines for subsequent runs within the same store. The mean shift between runs is around 3.3 μm in x and 3.8 μm in y. In store 1291 there is one big shift between run 144456 and 144457 by 66.5 μm in x and 105.7 μm in y. There is no similar shift in the COT beam line for this runs.

Recently the *FitBeamModule* has been modified to create new versions of the histograms used in the beam line fits for every 500 000 events. This allows to monitor the beam position during a single run. For a test we look at Stream G data that has been produced with offline version 4.8.4. The silicon tracks have been refitted using the *TrackRefitModule* from offline version 4.9.1. We have investigated 51 long runs. The Figures 4 and 5 show the beam positions obtained for different divisions

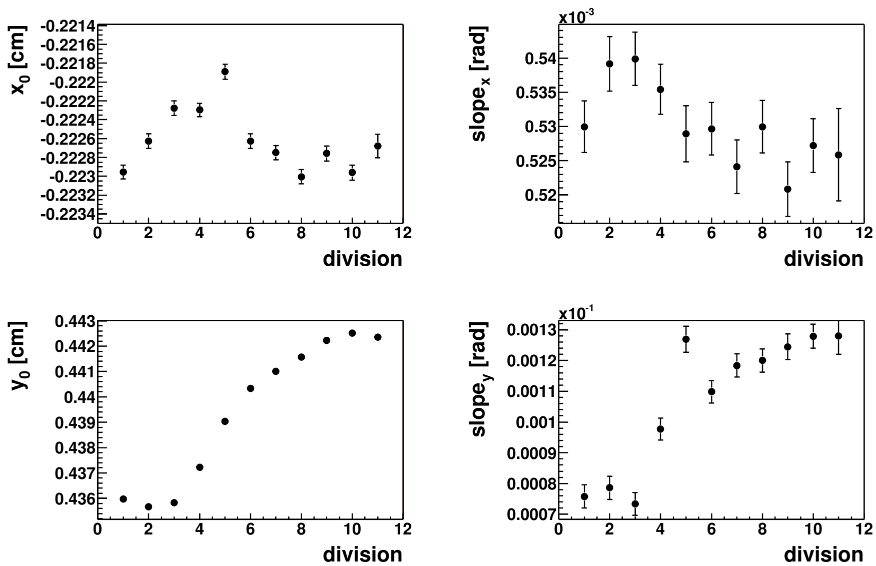

Figure 4: Beam position parameters for different divisions of run 148824.

of the two runs with the largest movements: run 148824 and run 150395. Figure 6 shows the maximum difference between the beam position parameters within one run for the 51 investigated runs. The movement of the beam is in the order of 5 μm for most of the analyzed runs. This result agrees with the first test as the analyzed runs are rather long.Nevertheless at least for two runs there are differences in the order of a few times the expected width of the beam. To take this into account, the beam position is fitted separately for each 500 000 events division and stored in the data base. This started with the history code 19 entries.

4 Size of the Beam Spot

The database table for the svx beam position contains fields for the beam spot size. To get the beam width the x and y coordinates of the primary vertices are plotted

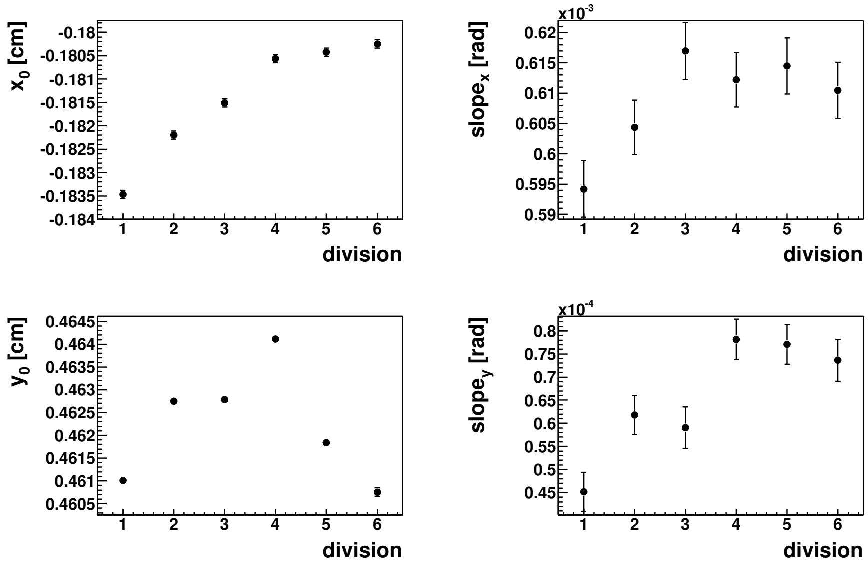

Figure 5: Beam position parameters for different divisions of run 150395.

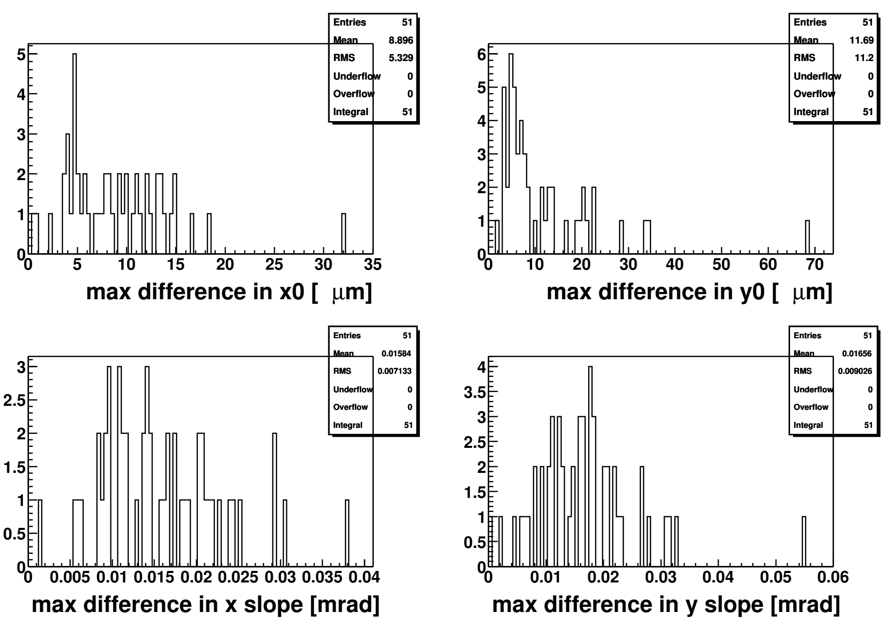

Figure 6: Maximum difference in the parameters for the beam position per division in the analyzed and rather long runs.

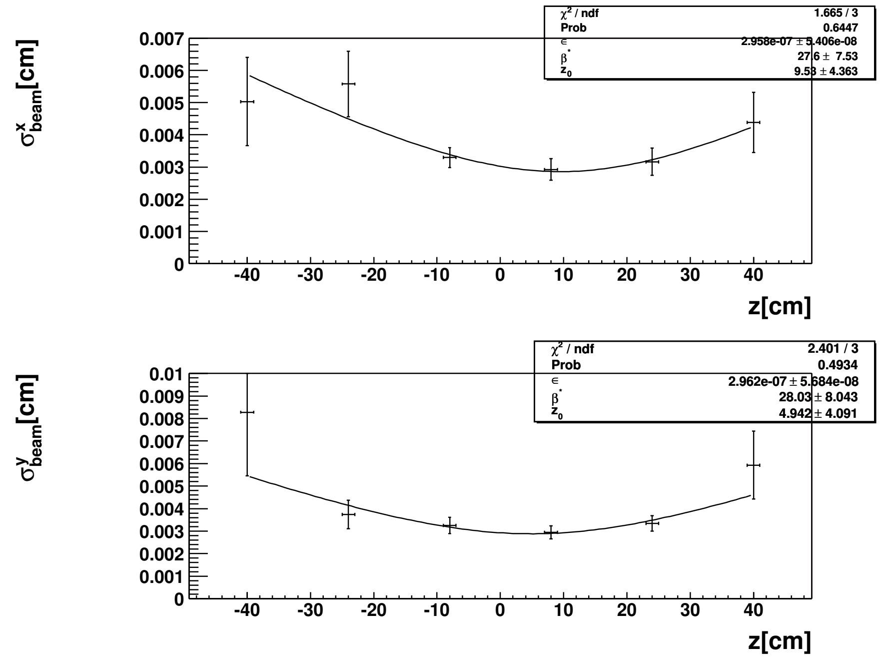

Figure 7: Beam width over z for run 144674.

for six certain z ranges. To limit the effect of the slope of the beam line the length of these range is 1 cm. Only vertices with an invariant mass above 50 GeV are used for the plots. Vxprim should reconstruct these vertices very well. In a first approach to obtain the mean beam width a Gaussian is fitted to the distributions. The weighted mean of the six values for the Gaussian widths is calculated. After subtracting the mean vertex fit resolution in quadrature the obtained value is filled into the database as the mean beam width. This procedure is comparable to what is done in [9] and used in the database entries with history code 17 and 19.

For this note we have tried to do better by correcting for the beam movements: The histogram of the x and y vertex positions for six different z ranges are created for every division of 500 000 events as it is done for the profile plots. For each division we correct for the fitted beam position and sum up the histograms for all divisions. Then a Gaussian is fitted to the resulting histograms. We correct these widths for the mean resolution of the vertex fits. After the fit we know the width of the beam for six different values of z. Figure 7 shows the result for run 144674. As the width of the beam should depend on z according to equation 1 we have tried to fit this function to the distribution. More data points are needed to really constrain the problem. For many runs the fit failed, the results for square root of ϵ times β^* for the successful fits is shown in Figure 8. This is the minimal width of the beam. The fitted transverse widths are larger than the value calculated in section 1.

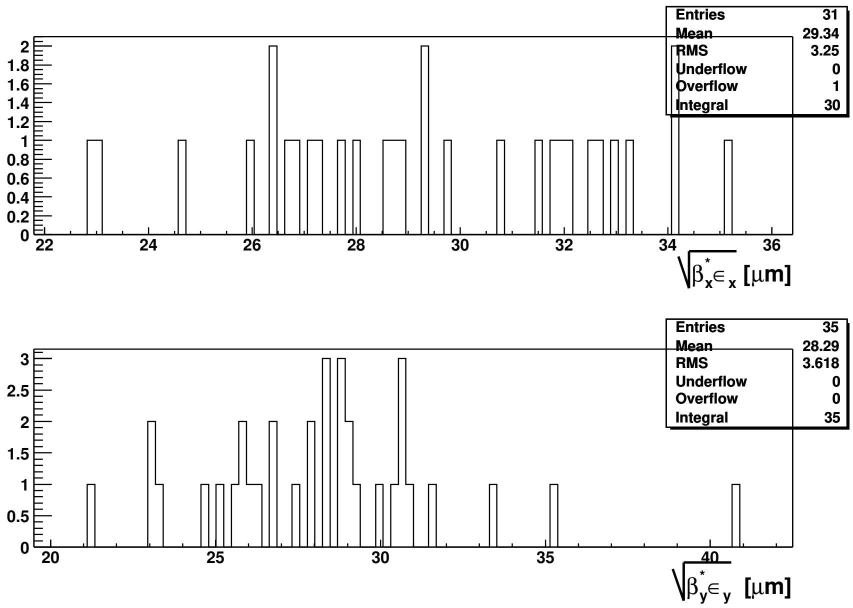

Figure 8: Minimal beam width of runs for that the fit for the beam width parameters converged.

References

[1] M. Martens and P. Bagley, "Luminosity Distribution During Collider Run II,". http://www-ap.fnal.gov/~martens/luminosity/beam_size.html.

[2] H. Wenzel, "The Primary Interaction Vertex,". CDF Note-4066.

[3] Fermilab Beams Division, "Run II Handbook,". http://www-bd.fnal.gov/runII/index.html.

[4] B. Flaugher, "Efficiency of the 60 cm Z-vertex cut,". CDF Note-5094.

[5] Run II Alignment group http://www-cdf.fnal.gov/internal/upgrades/align/alignment.html.

[6] H. Wenzel, "Fitting the beam position with the SVX,". CDF Note-1924.

[7] H. Wenzel, "Tracking in the SVX,". CDF Note-1790.

[8] H. Stadie, W. Wagner, H. Wenzel, and Th. Müller, "Vxprim in Run II,". CDF Note-6047.

[9] W. Wagner, "An estimate of the average beam width at CDF,". CDF Note-6104.

[10] Alignment source in the code browser
http://cdfcodebrowser.fnal.gov/CdfCode/source/Alignment/.

A Accessing the Beam Line Database

The alignment package contains helper classes to retrieve the beam position for a certain run. These classes are named *CotBeam* and *SvxBeam*. A typical used case in a AC++ module looks like this:

```
//header file
#include "Alignment/SvxBeam.hh"

class MyMod : public AppModule
{
public:
//...
private:
  SvxBeam* _svxbeam;
};

//source file
//initialize svxbeam to default
MyMod::MyMod(const char* const theName,
             const char* const theDescription)
  : AppModule(theName,theDescription),_svxbeam(NULL)
{
  //...
}
MyMod::~MyMod()
{
 delete _svxbeam;
 //...
}

//create SvxBeam object
AppResult MyMod::beginJob(AbsEvent* aJob)
{
  _svxbeam = new SvxBeam("default");
  // use time dependent beam positions
  _svxBeam->setUseTimeDep(true);
  //...
}

//load beam position for new run
AppResult MyMod::beginRun(AbsEvent* aRun)
{
```

```
  _svxbeam->loadRun();
  //...
}


AppResult VxPrim::event(AbsEvent* anEvent)
{
 //...
 //get the beam spot for a known z coordinate
 if(_svxbeam->isValid())
 {
    HepVector seedvertex = _svxbeam->position(seedz);
    HepSymMatrix seedcov = _svxbeam->cov3(seedz);
 }
```

Further information can be found in the web documentation of the alignment group [5] in the "Beamline Database" section.

A.1 Correcting for a Different Alignment

If you are using a test alignment and would like to use the beam line database, you can make a correction to the beam line entry. The SVX beam line should be retrieved into x, y, x_{slope}, and y_{slope} and do the following:

$$\begin{aligned} x+ &= Tx \\ y+ &= Ty \\ x_{slope}+ &= Ay (\text{the x - y interchange is correct}) \\ y_{slope}+ &= -Ax \end{aligned} \tag{11}$$

Tx, Ty, Ax and Ay are the difference between the space frame alignment entry from the test alignment you are using and the entry from the alignment that created the beam line entry. In the alignment, the SVX is aligned to the COT so the COT beam line does not need correcting.

There exists an useful wrapper around the *CotBeam* and *SvxBeam* classes, that automatically chooses the best beam position available and corrects it for global alignment changes. The class is called *BeamWrapper* and is part of the Alignment package [10].

B Executables and Scripts of the Alignment Package

B.1 FitBeam

FitBeam makes beam line database entries using a Root file containing the histograms created by the *FitBeamModule*.

Usage: FitBeam <root hist file> <runnumber> <history> <dbid> <aligndbid> <alignmentrun> <alignment version> <alignment status> <production version> <vertices/dphi/both> [nocot/nosvx]
Parameters:

- root hist file: the location of the Root file containing the histograms for the beam line fits
- runnumber: the number of the run to which the fitted beam line belongs.
- history: the history code for the beam line database entry.
- dbid: identifier of the database in which the entries should be written, e.g. "onotl_dev"
- aligndbid: identifier of the database from which the silicon alignment tables should be read.
- alignment run: run number of the silicon alignment used in the processing of the data.
- alignment version: version number of the silicon alignment used in the processing of the data.
- alignment status: status of the silicon alignment used in the processing of the data.
- production version: name of the production version used in the processing of the data.
- vertices/dphi/both: specify one of the three keywords:
  - vertices: only the algorithm based on the primary vertices will be used to make beam line entries.
  - dphi: only the algorithm based on the D-φ correlation of tracks will be used to make beam line entries.
  - both: both algorithm will be used to make the beam line entries.
- nocot/nosvx: This parameter is optional.

– nocot: only the SvxBeamPosition tables will be written to the database.
– nosvx: only the CotBeamPosition tables will be written to the database.

FitBeam uses the histograms inside the Root file to fit the beam line. This information is inserted into the beam position table. The history code inside the table is set to the value specified when calling *FitBeam*. For SvxBeamPosition entries the alignment used in the silicon tracking is stored with the entry: The global alignment of the silicon detectors is written to the *Spare* fields. The alignment tables for the silicon system are parents of the *RunKey* of a SvxBeamPosition entry. These alignment tables are specified by *aligndbid, alignment run, alignment version,* and *alignment status*. The values can be found on the Alignment web pages. The values for the 4.8.4 production are "ofotl_prd_read 100030 1 GOOD" for example. Additionally the algorithm field of the *RunKey* of a *SvxBeamPosition* or *CotBeamPosition* entry contains the *production version*, that is another parameter of *FitBeam*. In a last step the *RunKey* containing the beam position table is inserted into the database that has been specified by *dbid*.

B.2 FillMissingCotEntry and FillMissingSvxEntry

The executables try to make a beam line entry for a given run by interpolating the beam line position of the previous run and the following run. Only runs that belong to the same store will be used.

Usage: FillMissingSvxEntry <runnumber> <history> <dbid_write> < dbid_read > <algorithm>
Usage: FillMissingSvxEntry <runnumber> <history> <dbid_write> <dbid_read> <algorithm>
Parameters:

- runnumber: the number of the run for that the beam line entry should be made.
- history: the history code for the beam line database entry.
- dbid_write: identifier of the database in that the entry should be written, e.g. "onotl_dev"
- dbid_read: identifier of the database from that the entries should be read, e.g. "ofotl_prd_read"
- algorithm: the algorithm code of the beam line entries to use for the interpolation.

The algorithm value for the new entry is the algorithm value used to retrieve the beam line entries decremented by five.

B.3 UsedSetVersion

UsedSetVersion returns the version of the latest used set for a beam line entry for a given history code.

Usage: UsedSetVersion <dbid> <runnumber> <usedsetname> <history code> <pass index>
Parameters:

- dbid: identifier of the database from that the entries should be read, e.g. "ofotl_prd_read"
- runnumber: the number of the run for that the version of the used set should be retrieved.
- usedsetname: the name of the used set. Right now this is PAD_PHYSICS_SVXBEAM for SvxBeamPosition tables and PAD_PHYSICS_COTBEAM for CotBeamPosition tables.
- history code: the history code of the beam line entry for that the version of the used set should be retrieved.
- pass index: the index of the calibration pass to that the used set should belong.

The last argument is needed for the output. The output is in a format that can be used to fill a version map: <passindex> <runnumber> <runnumber> <used set version>. *UsedSetVersion* will list all used sets for beam line entries defined for a given run if one specifies "999" as the history code.

B.4 MakeUsedValidSet

MakeUsedValidSet makes used and valid sets for the latest beam line entry for a given history code.

Usage: MakeUsedValidSet <run number> <dbid> <history code> [cot or svx]
Parameters:

- run number: the number of the run for that the used and valid sets should be created.
- dbid: identifier of the database from that the entries should be read and to that the sets should be written to.
- history code: the history code of the beam line entry for that the used and valid set should be created.

- cot or svx: This an optional argument. If "cot" is specified the script tries to create a used and valid set only for the CotBeamPosition table. If "svx" is specified the script tries to create a used and valid set only for the SvxBeamPosition table. If nothing is specified the used and valid sets are created for the CotBeamPosition and the SvxBeamPosition table.

The name of the valid and used set for CotBeamPosition entries is PAD_PHYSICS_COTBEAM, and PAD_PHYSICS_SVXBEAM for SvxBeamPosition tables. Right now used and valid sets are only created for beam line entries having an algorithm code of 15 or 20. This means used and valid sets are created for beam line entries made from the vertices or interpolated from entries made with the vertices.

B.5 fitbeam

This script makes the beam line entries for a given run using a given histogram file and creates used and valid sets for these entries afterwards.

Usage: fitbeam <root hist file> <history> <dbid> <aligndbid> <alignmentrun> <alignment version> <alignment status> <production version>
Parameters: please see *FitBeam*.
The *fitbeam* script needs the same parameters as *FitBeam*. The only difference is that the run number is obtained from the histogram file name. For this the file name has to be of the format beamXXXXXX.root, where XXXXXX is the run number. After obtaining the run number *FitBeam* is called to make the database entries.

B.6 fitbeamjob

The script *fitbeamjob* creates beam line entries for all runs for that histogram files newer that certain file exist. At first it looks for new histogram files and then tries to merge all histogram files belonging to the same run for that a new file exists. This is done by a script called *mergefiles*. This script assumes a certain directory structure and format of the Root file names. Afterwards the script *fitbeam* is called for the merged histogram file. All the *fitbeam* parameters are defined within the script. The script is intended to be used with the histogram files that are created during production. Please see Appendix C for more information about running this script.

B.7 updateBeamlinePass.sh

The script *updateBeamlinePass.sh* fills the version map in the database by looking for used sets for beam line entries.

Usage: updateBeamlinePass.sh <pass name> <dbid> <history code> <low run> <high run> [nocot/nosvx] [nowrite]
Parameters:

- pass name: the name of the pass for that the versions maps should be updated.
- dbid: the identifier of the database to that the version map should be written.
- history code: the history code of the beam line entries that should be used for the versions map.
- low run and high run: specify the range of run numbers that should be used.
- nocot/nosvx: if "nocot" is specified only the PAD_PHYSICS_SVXBEAM versions map is updated, respectively only the PAD_PHYSICS_COTBEAM versions map for "nosvx".
- nowrite: this parameter is optional. If set, no entries are written into the database, but all the steps before are done.

Firstly the script gets a list of the numbers of all runs that are within the specified range and need an entry. Later the PAD_PHYSICS_COTBEAM and PAD_PHYSICS_SVXBEAM versions maps are merged with the PROD_PHYSICS_CDF map to create PAD_PHYSICS_CDF. To allow this all runs that appear in the PROD_PHYSICS_CDF versions map need an entry in the other two version maps. After getting the list of run numbers the script tries to create new used sets for all beam line entries with the right history code. The version numbers of these used sets are used to fill the versions map. Some runs in the list do not have a beam line entry because the run has not been processed or the run has been too short to fit a beam line. In this case the script tries to create a beam line by interpolating existing ones using the *FillMissingCotEntry* and *FillMissingSvxEntry* executables and then creates a used set for this new entry. There might be runs that still lack a used set; in this case a used set that points to a beam line entry that contains "0" in every field is created. When this is done the versions map is updated in the data base with the new entries. Please see Appendix D for more information about running this script.

B.8 copyBeamlinePass.sh

The script *copyBeamlinePass.sh* fills the version map in the database by copying the used set version numbers from an existing version map.

Usage: copyBeamlinePass.sh <output pass> <input pass> <low run> <high run>
Parameters:

- output pass: the index of the pass that should be populated.
- input pass: the index of the pass from that the entries should be copied.
- low run and high run: specify the range of run numbers that should be used.

If an old versions map needs to be updated for a few runs, it is useful to just copy the still valid entries from the old map into the new one. This is much faster than running *updateBeamlinePass.sh* again.

B.9 checkForMissingEntries

The script *checkForMissingEntries* checks for all runs that have an entry in the PROD_PHYSICS_CDF versions map for missing beam line entries.

Usage: checkForMissingEntries <dbid> <history code> <low run> <high run> [nocot/nosvx]
Parameters:

- dbid: identifier of the database from that the entries should be read, e.g. "ofotl_prd_read"
- history code: the history code of the beam line entries that should be used for the versions map.
- low run and high run: specify the range of run numbers that should be used.
- nocot/nosvx: if "nocot" is specified only silicon beam lines are checked, respectively only the cot beam lines used for "nosvx".

The script queries run summary database to get the number of events of a run, its luminosity, and the status of the SVX and the COT detector. If the run condition allows for a beam line entry the beam line database is checked for this run. An error message appears if this fails. Right now the script requires an integrated recorded luminosity of at least $20nb^{-1}$ for the run. The *cot status* should not be equal to 0 to expect a cot beam line. For the silicon beam line we require *cot status* and *svx status* different from 0.

C Making Beam line entries

The production histogram files for run 152177 stream B processed with 4.8.4a_08 is located at: *fcdflnx3: cdfprod0/4.8.4a_08/Stream_B/hist/sum_hist_151277_b.root*. The scripts that updates the database uses all histograms files that are newer than a specified data file named *fitbeamjob.date*. In *Alignment/scripts/fitbeamjob* the production version, history code, used alignment are specified:

```
HISTAREA=/cdf/scratch/cdfprod0
PRODUCTIONVERSION=4.8.4a_08
STREAMS="B C D E G H J"
DATEFILE=/cdf/home/stadie/beamfits/fitbeamjob.date
```

```
# script locations
MERGESCRIPT=./Alignment/scripts/mergefiles
FITBEAMSCRIPT=./Alignment/scripts/fitbeam

# history code
HISTORY=17
# global alignment
#4.3.x: -0.10106  0.13742   0.0007609  -0.0003540
#4.5.0: -0.10625  0.15003   0.0007568  -0.0003142
#4.3.x: 100016   3   TEST
#4.5.x: 100020   3   TEST
#4.6.x: 100020   3   TEST
#4.8.x: 100030   1   GOOD
ALIDBID=ofotl_prd_read
ALIRUN=100030
ALIVERSION=1
ALISTATUS=GOOD
# database identifier
DBID=onotl_prd
```

PRODUCTIONVERSION specifies the cdfsoft2 version used for production. Right now this is 4.8.4a_08. The histogram files for all data streams listed in *STREAMS* will be used. *DATEFILE* is the name of the file used to select histogram files. Only histogram files that are newer than this date file will be used. Right now this file resides in a private directory. So one needs to copy this file to an area with write-access (*cp -p /cdf/home/stadie/beamfits/fitbeamjob.date my_favorite_dir/fitbeamjob.date*) and change the variable inside the script. *HISTORY* is the history code for the new entries. *ALIDBID, ALIRUN, ALIVERSION, and ALISTATUS* specify the alignment used for the silicon system during this production.

C.1 How to update the Database

How to update the beam line database with history code 17 entries for the 4.8.4a_08 production:

- create a test release and add the Alignment package:

```
setup cdfsoft2 4.9.1
relnew beamrel
addpkg Alignment history_17
```

- rebuild the binary and test binary targets.

```
gmake
gmake tbin
```

- copy the date file into the test release with *cp -p* and update *Alignment/scripts/fitbeamjob* script to use the right date file.
- run the update:

```
nice Alignment/scripts/fitbeamjob |& tee fitbeamMMDDYY.log
```

It is very useful to keep the log files in a common place. Right now the log files are located at:
/cdf/home/stadie/beamfits/log

C.2 How to change the Script for a new Production Pass

- When a new production version is used one has to change *Alignment/scripts/fitbeamjob*:
  - increment history code for the *HISTORY* variable.
  - set *PRODUCTIONVERSION* to the name of the new production pass. This has to be the name of the directory that contains the histogram and log files of this production.
  - set the *ALI* variables to the version of the silicon alignment that is used in this pass.
- If histogram names have been changed or histograms have been added in *FitBeamModule*, the Root script to merge the histogram files needs to be modified. The script is located at *Alignment/script/MergeFiles.C*.
- if the scripts work commit the new version and tag the Alignment package:

```
cvs tag Alignment history_<history code>
```

D Making Pass Lists

If one wants to create new pass lists, one has to create the PAD_PHYSICS_COTBEAM and PAD_PHYSICS_SVXBEAM *calibgroups*, e.g. for pass 09:

```
~cdfsoft/dist/releases/development/CalibDB/bin/PassTool.pl -c /@cdfonprd
  -p PAD_PHYSICS_COTBEAM -n 09 -w
~cdfsoft/dist/releases/development/CalibDB/bin/PassTool.pl -c /@cdfonprd
  -p PAD_PHYSICS_SVXBEAM -n 09 -w
```

Then one can start filling the *group_versions_maps*:

- create a test release and add the Alignment package.

```
setup cdfsoft2 4.9.1
relnew beamrel
addpkg Alignment history_17
```

- rebuild the binary and test binary targets.

```
gmake
gmake tbin
```

- run the update.

```
nice Alignment/scripts/updateBeamlinePass.sh 09 onotl_prd 20 \
145001 152122 nosvx write |& tee updateBeamlinePass_09_cot_MMDDYY.log
nice Alignment/scripts/updateBeamlinePass.sh 09 onotl_prd 20 \
145001 152122 nocot write |& tee updateBeamlinePass_09_svx_MMDDYY.log
```

This will look for beam line entries with history code 20 within the run range 145001 to 152122. It will try to create missing beam line entries by interpolating existing ones for runs within the same store. If this fails a used set pointing to a beam line entry containing zeros will be put into the versions map to achieve completeness.