EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

ODYSSEUS – a Silicon Telescope for Test Beam Experiments

System Description and User Manual for the Data Analysis

29. May 2000

M. Deile, J. Dubbert, N.P. Hessey, A. Staude
University of Munich

Abstract

This note is mainly designed as a manual for the offline analysis of data taken in muon test beams (M2, X5 and H8) with the Silicon telescope of the University of Munich. The routines presented here are components of the offline analysis programs "MUTANT" (for M2 and X5) or "MUT8" (for H8) written by N. Hessey [1].

We explain the installation and usage of the routines and summarize the underlying algorithm. To give the user some background information about the detector system, the manual is preceded by a brief description of the hardware configuration and the readout.

ODYSSEUS

Omnipurpose
Detector
emplo**Y**ing
Silicon
Strips
for **E**xperiments
with m**U**on
Systems

1 System Description

1.1 Hardware Configuration

For the development and test of ATLAS muon detectors in test beam experiments it turned out to be essential to have a high precision reference system with a resolution far below 50 μm. For this purpose we have built a beam telescope consisting of 6 Silicon microstrip detectors aligned along the beam axis (Figure 1).

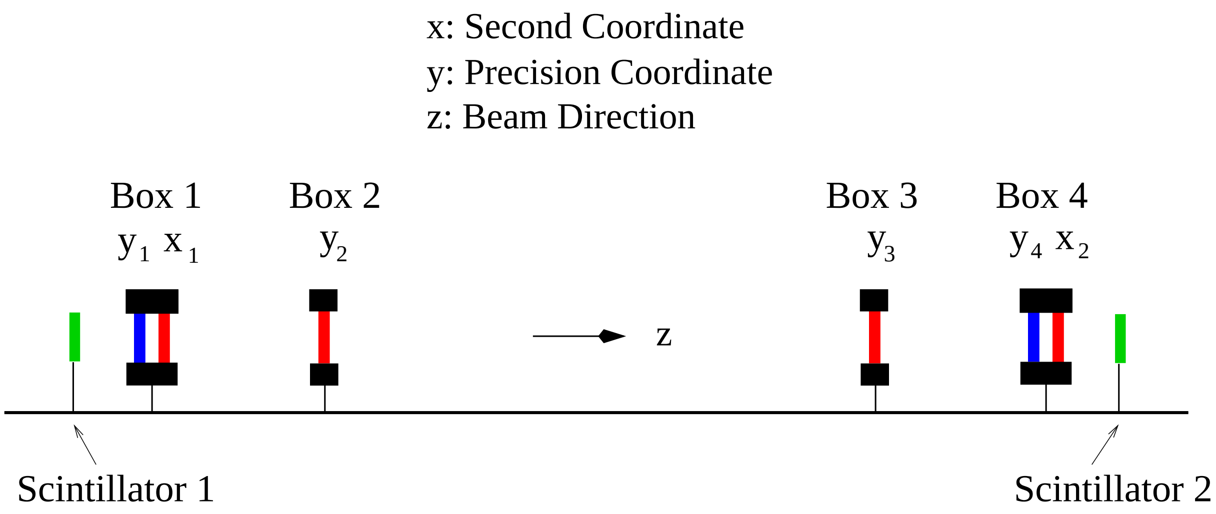

Figure 1: The Set-up of ODYSSEUS.

Each detector has 996 strips with a pitch of 50 μm and covers an area of 51 $\times$ 51 mm^2. Figure 2 shows a schematic detector cross-section, Figure 3 a view from the top.

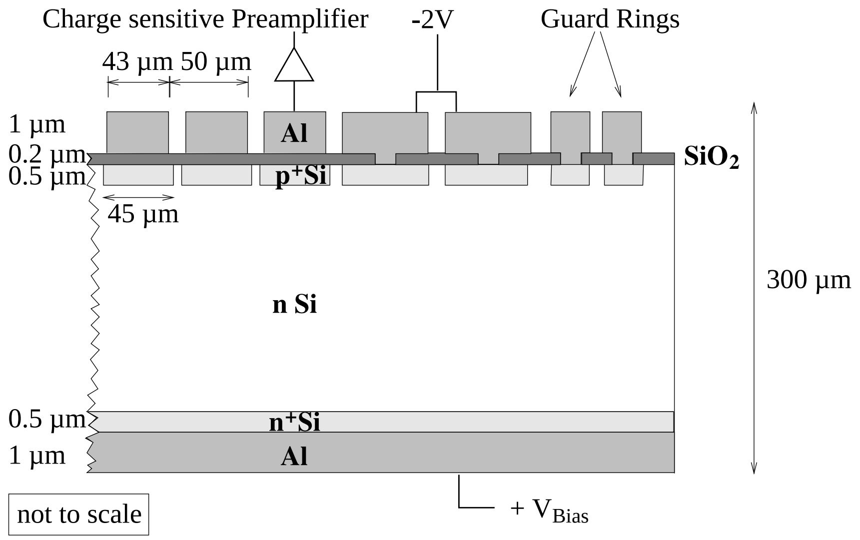

Figure 2: Schematic cut through a Silicon microstrip detector. All strips are read out.

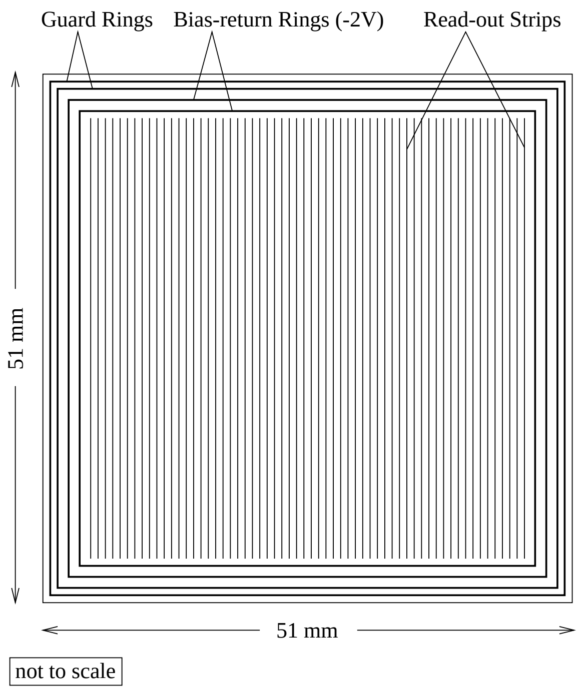

Figure 3: Schematic view of a Silicon microstrip detector from the top. In reality there are 23 guard rings, not only two.

Four detectors measure the precision coordinate of particle tracks (called "y") with a single detector resolution of about 7 μm. The two remaining detectors measure the second coordinate (called "x"). The 6 detectors are arranged in 4 boxes which are mounted on a support plate with a precision of 10 μm. The fine alignment is done by software (Section 3). The boxes 1 and 4 contain 1 x- and 1 y-plane each, whereas in the boxes 2 and 3 there is only a y-plane. The physical orientation of the (x-y)-frame (x horizontal, y vertical or vice versa) can be changed by turning the detector boxes around the z-axis (beam direction).

Each box consists of an Aluminium frame covered by Aluminium lids. The frame (Figure 4) carries one or two detector(s) with front-end hybrid board(s) and a repeater card. The lids were designed to be light-tight in order to keep the noise of the light-sensitive detectors low. At the position of the detector the lids have a window covered only with 200 μm Aluminium foil keeping multiple scattering small without compromizing the light-tightness (Figure 5). To fight against thermal leakage currents, the frame is cooled by a 30 W Peltier element.

The front-end electronics are located on hybrid boards (Figure 6) consisting of a ceramic base carrying eight VA2-chips [6] and a printed circuit board which distributes the bias voltages from the repeater card to the chips and sends the output signals from the chips to the repeater card. The detector and the ceramic board overlap by about 3 mm, where they are glued together. In this overlap zone the ceramic has a gold line on top, carrying the positive detector bias voltage. This gold line is connected to the back plane of the detector with conductive glue. Each detector read-out strip is bonded to one of the 128 channels of a VA2-chip. Each channel has a preamplifier, a shaper and a so-called sample & hold circuit which will be explained in the context of the read-out (Section 1.2).

Furthermore, the chip contains a multiplexer to read out all 128 channels through one analog differential output line.

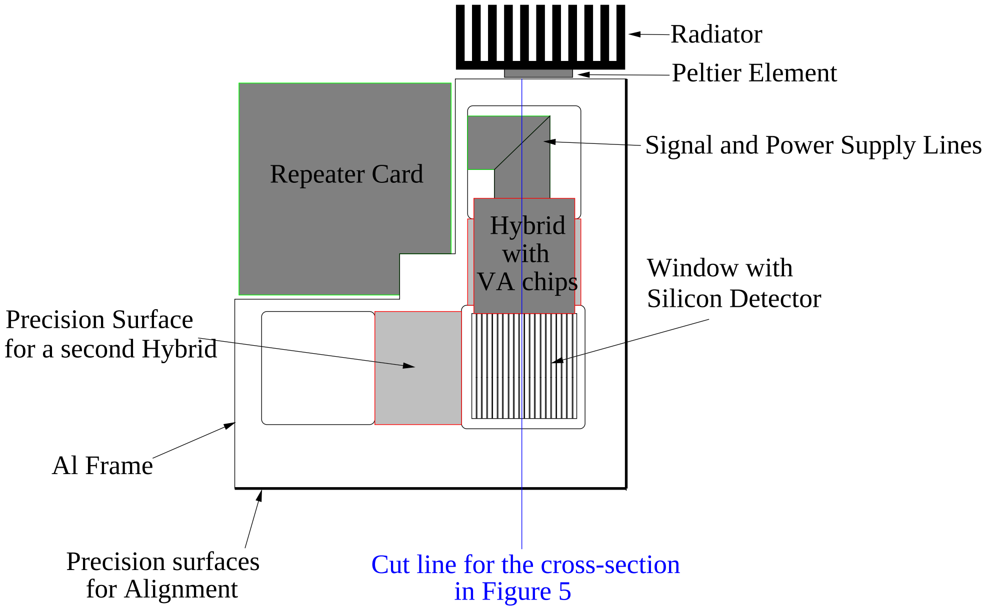

Figure 4: Frame with Silicon microstrip detector and the front-end electronics.

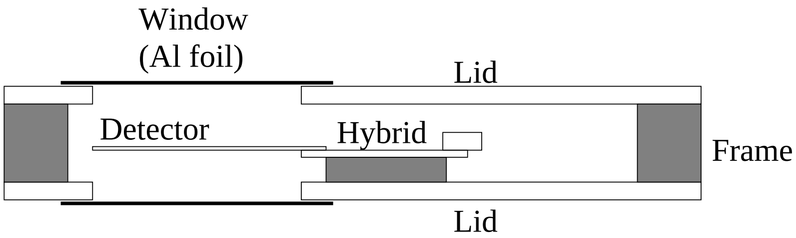

Figure 5: Cut through the detector box along the line indicated in Figure 4.

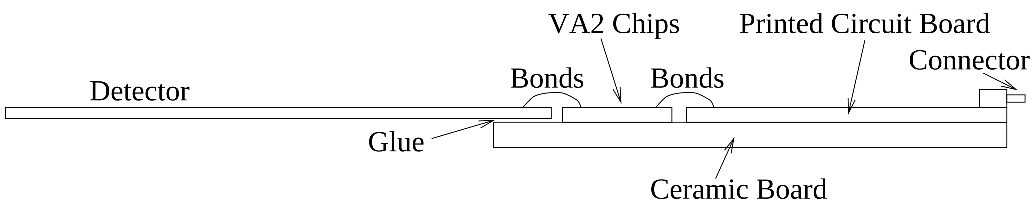

Figure 6: Side view of a detector with its hybrid board carrying the front-end electronics.

The repeater cards in all four detector boxes are interfaces between the hybrids, the power supplies, the sequencers (timing units) and the CRAMS[1)] (Flash ADCs):

[1)] Caen Readout of Analog Multiplexed Signals [5]

– The repeater cards take ±6 V from external power supplies and provide and stabilize all bias voltages and currents needed by the hybrids. The bias currents for preamplifiers, shapers and output buffers on the chips can be adjusted with potentiometers on the repeater cards. The feedback resistances of the preamplifier and the shaper are also controlled by potentiometers. The latter setting defines the shaping time, which in our set-up is about 2 μs.
– The detector bias voltages (+50 V to the detector back plane, -2 V for the guard rings) come from external power supplies. On the repeater cards they are filtered against noise and then transmitted to the hybrids.
– The timing signals from the sequencer are adapted to the logical standard of the hybrids.
– The raw differential output signal from the hybrids is amplified and adapted to the differential TTL standard for the transmission to the Flash ADCs.

1.2 Read-out

The full readout chain is drawn in Figures 7 and 8 while Figure 9 displays the read-out sequence.

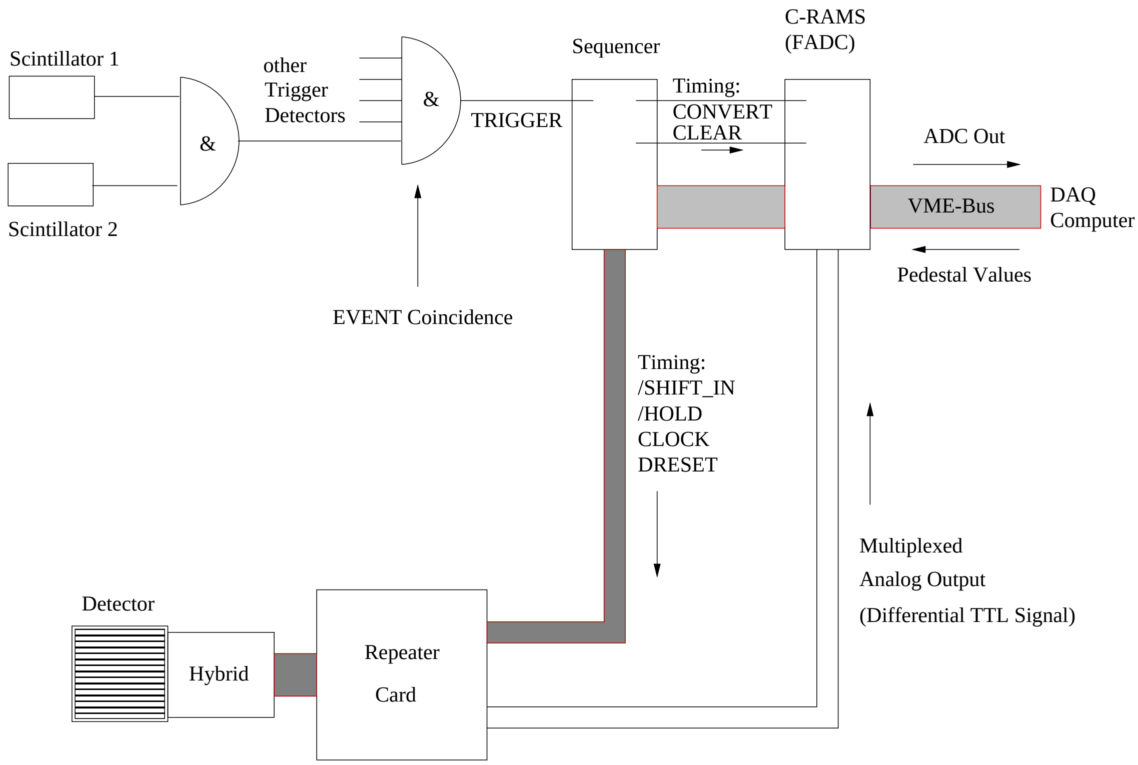

Figure 7: The Silicon telescope read-out chain.

When the sequencer receives a trigger, it immediately generates a `BUSY` signal which remains active during the whole read-out time and which can be used to avoid new triggers before the old event has been fully read. After a delay t_1 with respect to the trigger the

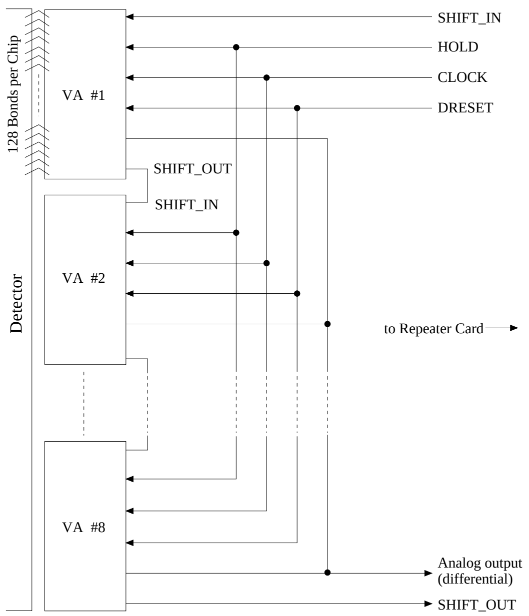

Figure 8: Schematic of the hybrid board showing the most important signal lines between the VA chips and the repeater card.

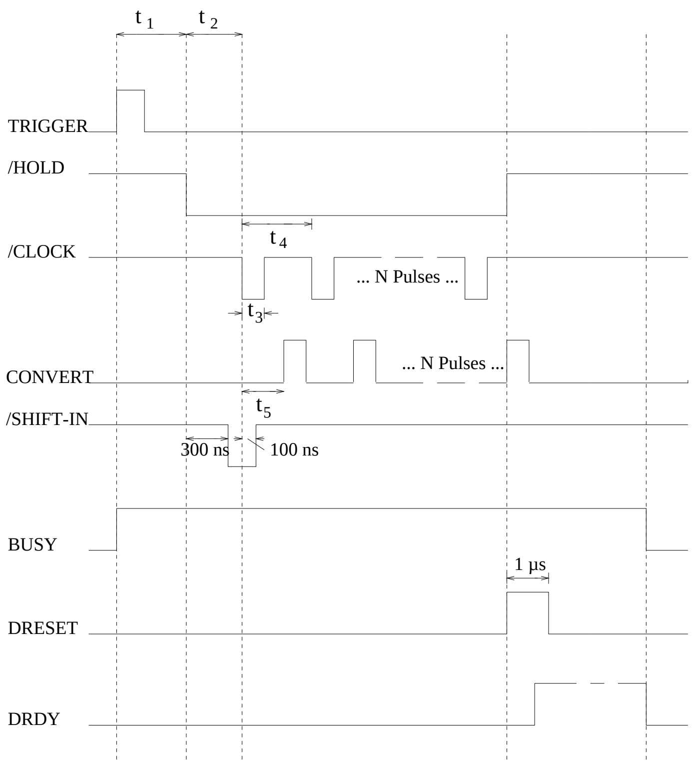

Figure 9: The Silicon telescope read-out sequence. The times constants t_1 to t_5 are set by software over the VME bus.

sequencer sends the HOLD signal to the VA chips via the repeater cards. The falling edge of this signal tells the sample & hold circuits in the VA chips to sample the the signal values at the shaper outputs of all channels simultaneously and to keep them constant during the whole read-out sequence. t_1 should be chosen such that the HOLD token arrives right at the peaking time of the detector signal pulse, i.e. about 2 μs after the physics event. 300 ns after HOLD the sequencer sends the SHIFT_IN signal to the first VA chip in each detector box. It tells the chips to start the sequential read-out of the individual channels as soon as the first CLOCK pulse arrives. The delay t_1 of this first falling edge of CLOCK with respect to HOLD is adjustable, but has to be chosen such that the SHIFT_IN signal is present when CLOCK sets in. Each of the 2048 CLOCK pulses sent by the sequencer switches the multiplexer in the VA chip to the next channel whose content is then given to the output. When the last (the 128$^{\mathrm{th}}$) channel of the first VA chip has been read, that chip generates a SHIFT_OUT signal which is given to the second chip as SHIFT_IN. Then the 128 channels of this second chip are read out and so on. When the last chip of a hybrid has been read, the SHIFT_OUT signal of this last chip can serve as SHIFT_IN for an eventual second hybrid in the detector box. In our system this is relevant for the boxes 1 and 4 which carry two crossed detectors.

The multiplexed signal values of all strips in a box (1 or 2 detectors) are fed into a CRAMS module (FADC). The digitization in the CRAMS is controlled by CONVERT pulses – clock pulses delayed by t_5 with respect to the original CLOCK – from the sequencer. Each CRAMS module has two channels; each channel can consecutively read 2016 pulse heights – one per VA channel – which is sufficient for a box with two detectors. Thus our set-up needs two CRAMS modules.

If the signal value of a CRAMS channel exceeds a certain threshold, the pedestal of that channel is subtracted from the signal. Finally the pedestal-subtracted signal value is given to the VME bus and recorded by the DAQ. After the last CONVERT pulse the sequencer sends a DRESET signal to all VA chips to reset them. As soon as the CRAMS has digitized all 2016 signal values it generates the DRDY ("data ready") signal which can be used in the DAQ programme to trigger the read-out of the CRAMS. When the computer has finished the read-out, it sends a CLEAR signal to the sequencer which deactivates the BUSY and passes the CLEAR on to the CRAMS.

Before each normal physics run the pedestals and their rms have to be measured. They vary with external conditions like temperature, humidity etc. For each channel a threshold is calculated according to

$$\text{threshold} = \text{pedestal} + 5 \times \text{rms} \tag{1}$$

After the pedestal run these thresholds and pedestals are written into the CRAMS memory.

1.3 System Performance

The performance of a reference tracker can be characterized by three quantities: the spatial resolution, the signal/noise ratio and the efficiency.

The passage of an ionizing particle through a silicon microstrip detector creates signals in a cluster of 1 to 5 adjacent strips. Our detectors have a mean cluster size of about 3 strips. The most pessimistic number for the spatial resolution is obtained by just using the centre of the strip with the highest deposited charge as the track position. This worst-case resolution is given by

$$\sigma_{\text{worst}} = \frac{\text{pitch}}{\sqrt{12}} = \frac{50\,\mu m}{\sqrt{12}} = 14.4\,\mu m \tag{2}$$

By determining the particle hit position as the centre of mass of the charge deposited on the strips near the track, a single detector resolution of 7 μm is obtained.

Without multiple scattering a linear track fit through the 4 y-planes would yield a tracking precision of

$$\sqrt{\frac{\sigma^2}{4}} = \frac{7\mu m}{2} = 3.5\mu m. \tag{3}$$

in the centre of the detector set-up (between the boxes 2 and 3). With multiple scattering however it is limited to 5 μm for muons with energies of the order 100 GeV.

The main contribution to the noise of a semiconductor detector is caused by the leakage current which is a strong function of the temperature:

$$I_{\text{leak}} \sim T^{3/2} e^{-\frac{E_{\text{gap}}}{2kT}} \tag{4}$$

High leakage currents have to be avoided not only to limit the noise, but also to prevent detector damage. In order to compensate the heat produced by the readout chips and components on the repeater cards, we added a Peltier cooling system which stabilizes the detector temperature at about 20°C. Thus the leakage current is kept below 100 nA per detector. The achieved signal-to-noise ratio, defined as

$$S/N \equiv \frac{\text{maximum of pulse-height spectrum}}{\text{rms spread of pedestal spectrum}} \quad , \tag{5}$$

is 62 $\pm$ 1 averaged over all strips in all detectors.

The efficiency of the total reference system has several components:

- The intrinsic efficiency of semiconductor detectors is very close to 100 %.
- The geometrical acceptance is given by the ratio

$$\frac{\text{detector surface}}{\text{trigger scintillator surface}} = \frac{51\text{mm} \times 51\text{mm}}{51\text{mm} \times 53\text{mm}} = 96\,\% \quad ,$$

provided that the scintillators and the silicon detectors are perfectly aligned with respect to each other and parallel to the beam.

- An important contribution to inefficiencies comes from dead or noisy channels, mostly caused by broken or damaged bonds or preamplifiers. Very noisy channels which produce hits in almost every event must be discarded from the analysis and therefore have to be considered as dead.

Figure 10 shows a typical efficiency plot for "ODYSSEUS".

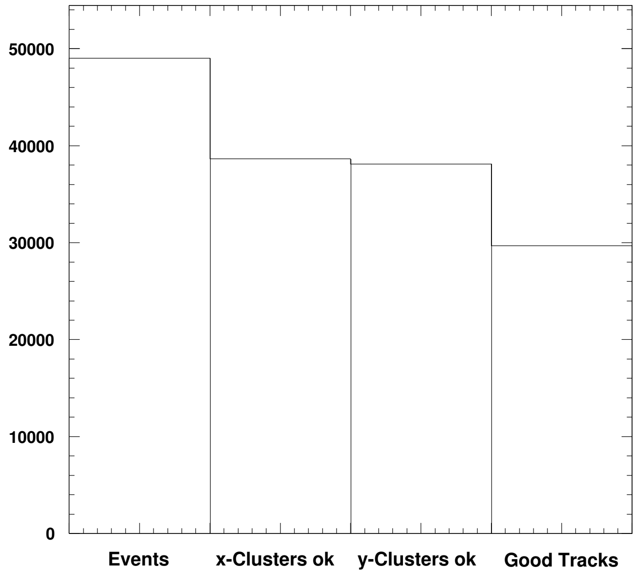

Figure 10: Efficiency plot for "ODYSSEUS" (see text).

The first bin contains the total number of events, i.e. scintillator triggers.

In the second bin only those events enter which have exactly one good hit cluster in both planes measuring the second coordinate. This selection cut is necessary because through two planes a unique track can only be determined if each plane has one measured track position. Even after elimination of generally noisy channels, 20 % of all events have more than one valid cluster in an x-plane, which explains the strong drop from the first to the second bin in the figure.

The third bin requires in addition at least one good cluster in at least three of the four planes measuring the precision coordinate. These three points are necessary to fit a track by least-squares minimization. Only very few events fail this selection test.

To be considered a "good track" (fourth bin in Figure 10), a track has to pass a cut on χ^2. This cut is chosen such that the confidence level of the measured χ^2 (upper tail probability of the χ^2-distribution for 2 degrees of freedom) is greater than 0.05. By applying this cut we discard mainly events whose real hit positions coincide with bad strips and remain undetected, while the track fit uses noise hits, resulting in a bad χ^2. Another contribution to losses at this cut is made by acceptance inefficiencies, i.e. tracks passing through the scintillators but not through the slightly smaller detectors.

Finally the silicon telescope has a typical yield of 60 % good tracks relative to the total number of events.

2 Installation of the Software

Before installing the Silicon Telescope software you should get acquainted with the main program MUTANT/MUT8 by reading the manual [1].

Don't mix up routines for MUTANT with routines for MUT8! The different DAQ systems (M2, X5: Birdie [2], H8: RD13 software [3]) resulted in different data formats. In the newest version (28.04.99) of silicon telescope software the analysis codes for MUTANT and MUT8 have been unified. The data-format specific parts are steered by a compiler flag (see below).

To run the Silicon Telescope analysis you need a MUTANT(MUT8) directory (from now on called **$MUT**) with the subdirectories:

- **db/stel** (data base)
- **hst** (histogram definitions)
- **src** (your analysis code)

The basic parts of the MUTANT/MUT8 code can be found on afs computers in:
/afs/cern.ch/user/h/hessey/public/mutant(mut8)/new
If you work on an HP System, you just have to link the library files **libmutant.a** (or **libmut8.a**) and **libIn.a** into your executable. For other platforms these libraries have first to be compiled.

For the silicon telescope analysis additional files are needed. Copy them from
/afs/cern.ch/user/d/deile/public/stel/pro
into your $MUT directory:

- Database files:

  $MUT/db/stel/stellong.geom or stelshort.geom: Geometry files. The files contain comments (marked by "!") explaining the meaning of the parameters. Note that there are two different baseplates for the telescope. For choosing the correct geometry file you have to find out which plate was used for your runs. At the last software revision the format of the geometry files was modified. The new geometry files have the label "V3" in the first line. Since the code is backwards compatible, you can still use old geometry files without any problem. However, the new version offers more possibilities. You hopefully needn't touch the source code anymore.

  $MUT/db/stel/detector_{$n$}.data (n = 1, ..., 6):** File containing bad strips which will be rejected in the analysis. **detector_{n}.data as well as **stel.geom** are initial default files which have to be modified in your alignment procedure. They can be different for each run.

  $MUT/db/stel/none.smooth: Dummy smoothing coefficient file for runs with disabled smoothing (see Section 3).

  $MUT/hst/stel.hst: Histogram definition file.

– Code:

$MUT/src/stel.h: Header file needed for all Silicon Telescope subroutines. Here the STEL specific constants and data structures are defined (see Section 5).

$MUT/src/act_stelgeom.c: This routine reads the data files **stel.geom** and **detector_{*n*}.data**. It is activated when you issue the KUIP command `STEL/GEOM` {*name of geometry file*} at the "MUTANT>" ("MUT8>") prompt or more usually in the **start.kumac**. The geometry information is stored in the structure "StelGeom".

$MUT/src/act_stelsmooth.c: It reads the smoothing coefficients at the KUIP command `STEL/SMOOTH` {*file name*} {*plane number*} and stores them in the structure "StelGeom".

$MUT/src/stel.cdf: In this KUIP command definition file the specific KUIP commands `STEL/GEOM` and `STEL/SMOOTH` are defined. Here you can also add your own KUIP commands. The makefile translates **stel.cdf** into the c-program **stelkcom.c** which is then compiled.

$MUT/src/analstel.c: The main silicon telescope analysis routine. It is called by **usevnt.c** and has 3 arguments: A pointer to the raw data bank (input), a pointer to a structure of type StelTrack (output) and a pointer to the geometry structure of type StelGeom (input); see Section 5 for the definitions. For the newest version the raw data decoding has been detached and put into separate routines **decstelMUTANT.c** (**decstelMUT8.c**). Therefore **analstel.c** is now common to MUTANT and MUT8.

$MUT/src/decstelMUTANT(MUT8).c: These routines are specific to the MUTANT/MUT8 data formats. They get the raw data bank from **analstel.c** and return an array of hits. At this stage no quality cut is applied to the hits.

$MUT/src/findstelclusters.c: This routine gets the list of hits from **analstel.c** and returns an array of hit clusters (hits in neighbouring strips).

$MUT/src/do_smoothing.c: Here a correction is applied to the centers of gravity of the hit clusters. The underlying algorithm ("smoothing") is explained in Section 4.3.

$MUT/src/makefile: It compiles the silicon telescope routines and collects the object files in the library **libstelMUTANT.a** or **libstelMUT8.a**. The precompiled libraries which you find in the **hp700** and **Linux** directory contain object files for HP and Linux systems. To compile the code correctly for MUTANT or MUT8, you have to define a preprocessor constant MUTANT or MUT8 with the "-D" compiler option.

Some example files showing how to include the silicon telescope routines into your MUTANT/MUT8 analysis are available in the **examples** subdirectory:

useventMUTANT(MUT8).stel.c (to be renamed to **usevnt.c** before compiling): This is the user's analysis routine which is called by the main program at each event. It calls analstel.c as well as your own analysis subroutines for the other detectors (eg. RPCs, CSCs, MDTs). Here you merge the information from all the subsystems.

makefileMUTANT(MUT8): It compiles the user code and links the general MUTANT/MUT8 routines from **libmutant.a** (**libmut8.a**) and the silicon software from **libstelMUTANT(MUT8).a** into the executable. You have to adapt the makefile to your local system.

startstel.kumac: The start file for MUTANT/MUT8 containing KUIP commands that have to be executed before processing a run (see the MUTANT manual [1]).

smoothing.kumac: This is a PAW kumac which calculates the smoothing coefficients for the next run of MUTANT/MUT8 using histograms filled during the previous run. The smoothing procedure will be explained in Sections 3.11b and 4.3.

– Pool for raw data and histogram files:
  You can create your temporary data storage area on atlas:
  /shift/atlasd01/data01/zp/{*your name*}
  Warning for newcomers: The data in this scratch zone are deleted from time to time (sometimes very quickly).

3 The Analysis Procedure

1. Copy the raw data to your pool:
   (a) X5-data:
   The data are written to Exabyte tapes which are stored in the CN vault. Find the tape numbers in the X5 logbook, then read the data using the `stagein` command. A useful shell script based on this command with all the necessary arguments is
   /afs/cern.ch/user/d/deile/public/mutant/stageexa (for Exabyte tapes).
   For getting X5-data taken in autumn 1997 there is a more comfortable way: In **/afs/cern.ch/user/h/hessey/public/mutant/db/tapes** there is a script **getrun** and two data base files **lj0085** and **lj0096**. You just have to type
   `getrun` {*run number*}
   and the file appears in your atlas pool (after several hours).
   (b) H8-data:
   Look at the file **~muontb/cdr/logfiles/list_of_files_copied_1** on hpamu01 which gives you the place of the data file in the atlas shift zone. Then copy the data with the command
   `rfcp` {*file in shift zone*} .
   The logfile also gives the tape number in the CN vault to which the run has been written. Read the data using the `stagein` command (e.g. with the script **/afs/cern.ch/user/d/deile/public/mut8/stage**).
   For a short time after data taking the files are also available in the **/rd13_data** directory on hpamu01.
2. In your **start.kumac** specify
   `STEL/SMOOTH none.smooth` {i} for each plane i. This disables the smoothing procedure for the first runs. Smoothing is only appropriate when the raw alignment is done.
3. In the first MUTANT/MUT8 runs concentrate on the Silicon telescope and don't analyse the other banks: In **usevnt.c** call only **analstel.c** .
4. For the raw alignment runs it is sufficient to process some 1000 or 2000 events (KUIP command `EVENT` {n}). It saves time.
5. Check your data base files and their paths given in **start.kumac**. Also check the paths of the **detector_{*i*}.data** files given in **stel.geom**.
   If a plane was broken while you were taking data, set the ADC threshold for this plane to at least 1000 (6th column in the first data block of the geometry file. The code will then automatically exclude that plane from the tracking.
   You also have to choose the correct spatial orientation of the planes by setting the two flags in the second data block of the geometry file.
6. Make sure that in **stel.hst** you book only the histograms you really want to look at. Otherwise the saving at the end of the run might take very long.
7. Compile MUTANT/MUT8 with `make`.
8. Type `mutant` (`mut8`). The KUIP prompt "MUTANT>" ("MUT8>") will appear.
9. Initialize MUTANT/MUT8 with your **start.kumac** and start the analysis with `go`.

10. When the run is finished, save the histograms with
    `save $pool/{`*run name*`}.hist`
    where **$pool** is your data storage area.
11. Histogram analysis in PAW: In PAW load the STEL histograms by typing
    `hi/file 1 $pool/{`*run name*`}.hist`
    `cd stel`

    (a) Alignment and update of the data base files:
    Important histograms:
    - Hit spectra (histograms 31 - 36):
      They show for each strip the number of hits. In general you will see some very high spikes. These are extremely noisy strips which fire permanently. For a proper continuation of the analysis hits in these strips have to be rejected by writing their numbers into the first column of the **detector_{*i*}.data** files. The second column must contain an integer code specifying the problem of the concerned strip:
      1 dead strip
      2 low gain
      3 noisy strip
    - ADC versus strip number (histograms 41 - 46):
      If there is a dominant number of entries at very low ADC values, the hardware threshold in the FADCs (CRAMS) hasn't been high enough to reject all the noise. In this case the software threshold in the geometry file (column 6) has to be increased.
      Some blocks of 128 strips have considerably lower ADC values than the other strips. This is due to a hardware fault in the preamplifier chips. **analstel** attempts to correct for this effect by scaling the concerned ADC values with factors specified in the last block of the geometry file.
    - Histogram 5 shows for each event selection the number of remaining events:
      1 Total number of events
      2 Events with no more than MAX_CLUSTER clusters per plane (MAX_CLUSTER is defined in **stel.geom**).
      3 Events with exactly 1 good cluster per working x-plane
      4 Events with at least 1 good cluster in at least 3 y-planes
      5 Events with a reconstructed track
      6 Events whose reconstructed track survives the track fit probability cut (defined in the geometry file).

      If after the first run there is no reconstructed track at all, the alignment is probably very bad. In this case the χ^2-criterion for a track has to be made less tight until the raw alignment is done. For this purpose the maximum acceptable χ^2 – defined in the geometry file – should be set to 10000 instead of 100.

- y-track residuals = $y_{hit} - y_{track}$ (histograms 501 - 504): These histograms are used for the correction of y-shifts of the detector planes with respect to each other.
  Direction of the adjustment: positive residual → reduce the y-offset in the geometry file (second column).
  To reduce the number of alignment iterations define the two outer y-planes as being correct and move only the inner ones.
- y-residuals versus x-coordinate (histograms 601 - 604): The y-residuals should not depend on the x-coordinate. If they do, the concerned y-plane is tilted around the z-axis. This is corrected in the fourth column of **stel.geom**.
- Track fit probability distribution (histogram 1001): The single plane resolutions in column 5 of **stel.geom** have to be chosen such that the track fit probability distribution is flat. An initial pessimistic estimate is the rms of a single strip square distribution: 50μm $/\sqrt{12} \approx 15\mu$m.

Repeat the above alignment corrections until the precision is at the micron level.

(b) Smoothing: When your alignment precision is at the micron level, do the PAW part of the smoothing procedure:
`exec smoothing`
The kumac uses the histograms 10041 - 10046, which have been filled during the previous run of MUTANT/MUT8, to calculate correction coefficients for the track position within a hit strip. These coefficients are written to the files **run{*run number*}p{*plane number*}.smooth** in the **$MUT/db/stel** directory. The theoretical background of the method will be explained in Section 4.3.

12. If step 11b has been executed, the MUTANT/MUT8 part of the smoothing procedure has to be enabled by specifying the names of the coefficient files (created in step 10b) in the **start.kumac**:
`STEL/SMOOTH run`{*run number*}`p` {*plane number*}`.smooth` {*plane number*}
13. The fine alignment can be pushed to the 0.01 μm level, if after each MUTANT/MUT8 run **smoothing.kumac** is executed in PAW.

4 The Algorithm

In this section we describe some particularly important parts of the code. It is not meant to be exhaustive.

4.1 Decoding of the Raw Data

4.1.1 Numbering Scheme of Detector Strips and CRAMS Channels

Table 1 shows the assignment between detector strips and CRAMS channels.

Detector	Measured Coordinate	CRAMS	CRAMS Channels	Channels Connected to Strips	Number of Connected Strips
1	y	1	0 - 1023	14 - 1009	996
2	x	1	1024 - 2015	1038 - 2015	978
3	y	2	0 - 1023	14 - 1009	996
4	y	3	0 - 1023	14 - 1009	996
5	y	4	0 - 1023	14 - 1009	996
6	x	4	1024 - 2015	1038 - 2015	978

Table 1: Assignment of Detector Strips to CRAMS Channels

4.1.2 Decoding Procedure in decstelMUTANT

Figure 11 shows the dump of the STEL bank in a raw event (it is produced if the command `DUMP ON` is issued before processing an event).
The 4 words of the bank header represent (see also Section 5):
- Bank length in short words (4 bytes each)
- Bank name (2 short words; the bytes within a word are swapped)
- Bank version (currently = 1)

```
Bank header:
  2a 5354 454c    1
Bank TSLE has 38 data words:
   0    5 400C 9025 400C      A2A1 400C B0CC 400C C013
4016    E    0    0    0         3 401A 20D9 401A 3028
C02E  39B    0    7 401C      B01B 401C C1B8 401C D10C
401C E01E 4061 A017 4061      B113 4061 C018
```

Figure 11: The STEL bank part of a hexadecimal raw event dump (MUTANT)

The raw data are arranged in short words. In the analysis always 2 consecutive words are rearranged into one longword which constitutes one piece of information.

There are 4 data blocks in the STEL bank, each corresponding to one CRAMS. The first long word of each block gives the number of data long words in that block. The data long words contain the CRAMS channel (strip number), the measured ADC value and 2 status bits. They are encoded as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
O	V	unused							Channel number											Channel data											

O: Overrange bit: 1 = FADC overrange

V: Validity bit: 1 = FADC value over pedestal

If the overflow bit is 1, we set the ADC value to 0, so that the hit will be ignored in the later analysis.

In the next selection hits are declared valid if their ADC value exceeds the threshold given in the geometry file.

4.1.3 Decoding Procedure in decstelMUT8

Figure 12 shows the dump of the STEL bank in a raw event (it is produced if the command `DUMP ON` is issued before processing an event).

```
Bank STEL  Size [decimal] = 24; ID [decimal] = 2800; ID-name = STEL

    1001 dead551b        4 40126025 40127118 40128023 4012d016        0
       5 40128048 4012916f 4012a0f6 4012b121 4012c03c        5 40125056
401260d7 405aa07f 405ac072 405ad017 cafe551b
```

Figure 12: The STEL bank part of a hexadecimal raw event dump (MUT8)

Unlike in MUTANT the raw data are already arranged in longwords. The first word is the bank version (currently 1001 in hex.). The second (dead551b) and last (cafe551b) words are labels for the beginning and the end of the STEL data. There are 4 data blocks in the STEL bank, each corresponding to one CRAMS. The first word of each block gives the number of data words in that block. The data words are encoded as in MUTANT (see Section 4.1.2).

4.2 The Hit Cluster Selection

The passage of an ionizing particle through a Silicon microstrip detector creates signals in a cluster of typically 1 to 5 adjacent strips. The mean cluster size in our detectors is about 3 strips.

A cluster is subject to the following quality criteria:

- It must not contain a dead strip (code 1 in the strip data files).
- It must not contain a noisy strip (code 3 in the strip data files) with an ADC-value < 100. Above this threshold the hit is likely to be good.

– The total cluster charge must be greater than a lower limit which is set in the geometry file. Empirically we have found that a reasonable value is 100 integrated ADC channels.

4.3 Determination of the Hit Position and the Smoothing Procedure

An ionizing particle which traverses a Silicon detector creates signals in typically 1 to 3 strips. The number of responding strips depends on the position of the track: If it goes through the center of a strip, we will get hits in 1 or 3 strips depending on the width of the charge cloud. If on the other hand the particle passes between two strips, these strips will have a signal. From the point of view of resolution, hit clusters with 3 strips are more advantageous than 1-strip clusters, because the lateral strips contain the information about the track position within the central strip. 1-strip clusters don't provide this information, and their resolution is only $\frac{d}{\sqrt{12}} \approx 14\mu$m.

Events with 4 or more responding strips are a minority for our detector system (≈ 10 %). Since furthermore low lateral hits are very susceptible to noise, we use only the 3 strips with the highest amount of deposited charge for the reconstruction of the track position, which is given by the center of mass of the deposited charge:

$$y_{hit} = \frac{q_{+1} - q_{-1}}{q_{-1} + q_0 + q_{+1}} \cdot d + n_0 \cdot d, \tag{6}$$

where n_0 and q_0 are the strip number and the deposited charge of the highest hit, q_{-1} and q_{+1} the charge of the left and right neighbour; d is the pitch.

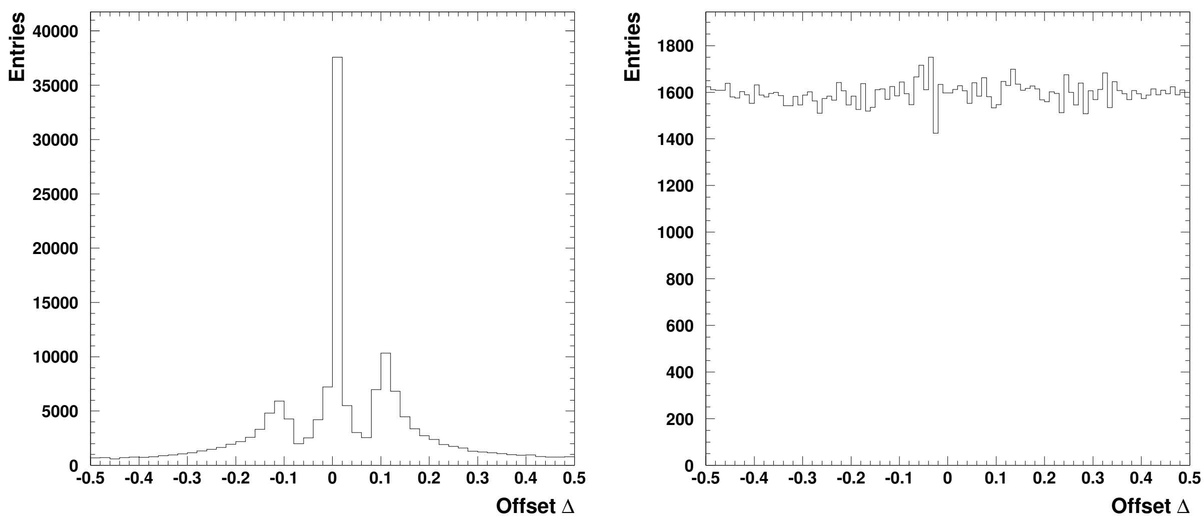

Figure 13: Offset distributions before and after smoothing

The term

$$\Delta := \frac{q_{+1} - q_{-1}}{q_{-1} + q_0 + q_{+1}} \tag{7}$$

(in our software we call it "offset") gives the distance of the track from the center of the strip with maximum charge. Δ can take values between -0.5 and +0.5. Assuming a uniform beam profile in the range of a strip pitch one would expect that the distribution of Δ is flat over [-0.5; +0.5]. As shown in Figure 13a this is not true: The distribution is strongly peaked at $\Delta = 0$.

This effect has the following reason: The tails of the charge distribution disappear in the noise and are cut away either by the threshold applied in the readout (CRAMS) or by us taking into account only the 3 highest hits of a cluster.

The non-constant distribution of Δ introduces a bias into the track reconstruction and has to be corrected. This correction is called "smoothing" and is based on the following theorem:

Be x a probability variable ranging in an interval $[-a, a] \subset \mathbf{R}$ with $a > 0$.
Be furthermore $f : [-a, a] \to \mathbf{R}_0^+$ the distribution function of x and

$$F(x) = \int_{-a}^{x} f(t)\, dt \quad (x \in [-a, a]) \tag{8}$$

the cumulative distribution function with the properties:

i. F is strictly monotonic on $[-a, a]$. This requires that there be only a finite number of points x_i with $f(x_i) = 0$.
ii. F fulfills the normalization condition $F(a) = 1$.
iii. f is finite on $[-a, a]$.

We are looking for a transformation

$$\Xi : x \in [-a, a] \to \xi \in [-a, a]$$

that meets the following conditions:

a. The distribution $g(\xi)$ with $\xi = \Xi(x)$ is normalized and flat over $[-a, a]$.
b. $\Xi(-a) = -a$
c. $\Xi(a) = a$
d. Ξ is differentiable.
e. Ξ is continuous.
f. Ξ is inversible.

The theorem's claim is that there is one unique transformation fulfilling the conditions (a) – (f). It is given by

$$\xi = \Xi(x) = 2\, a\, F(x) - a = 2\, a \int_{-a}^{x} f(t)\, dt - a \quad . \tag{9}$$

Proof:

I. $\underline{\Xi(x) = 2\,a\,F(x) - a \text{ is a solution:}}$

We have to show that the conditions (a) – (f) are fulfilled.

a. Conservation of probability implies

$$f(x)\,dx = g(\xi)\,d\xi \quad \text{with } \xi = \Xi(x),\ d\xi = \frac{d\Xi}{dx}\,dx \tag{10}$$

(The existence of $\frac{d\Xi}{dx}$ will be shown in (d)).

$$\begin{aligned} f(x) &= g[\Xi(x)]\,\frac{d\Xi}{dx} \\ &= g[\Xi(x)]\,2\,a\,f(x) \\ g(\xi) &= g[\Xi(x)] = \frac{1}{2\,a} \qquad \forall \xi \in [-a, a] \end{aligned}$$

Therefore $g(\xi)$ is flat. Integration over $[-a, a]$ confirms that $g(\xi)$ is normalized to 1.

b. $\Xi(-a) = 2\,a\,F(-a) - a = -a$ (by definition (8) of F).

c. $\Xi(a) = 2\,a\,F(a) - a = a$ (by normalisation ii).

d. Ξ is differentiable because F is differentiable by definition.

$$\frac{d\Xi}{dx} = 2\,a\,\frac{dF}{dx} = 2\,a\,f(x) \quad .$$

e. Since f is finite (iii) and defined over $[-a, a]$, its integral F is continuous over that integral. Continuity of F implies continuity of Ξ (9).

f. F is strictly monotonic (i); so Ξ is also strictly monotonic and hence inversible.

II. $\underline{\Xi(x) = 2\,a\,F(x) - a \text{ is the only solution:}}$

Condition (a) implies

$$g(\xi) = \frac{1}{2\,a} \qquad \forall \xi \in [-a, a]$$

Again we use the conservation of probability:

$$\begin{aligned} f(x) &= g[\Xi(x)]\,\frac{d\Xi}{dx} \\ &= \frac{1}{2\,a}\,\frac{d\Xi}{dx} \\ \frac{d\Xi}{dx} &= 2\,a\,f(x) \end{aligned}$$

Integration yields

$$\Xi(x) = 2\,a\,F(x) + c$$

with an integration constant c which is fixed by condition 1:

$$\Xi(-a) = -a \ \Rightarrow\ c = -a$$

Thus one obtains unambiguously the transformation

$$\Xi(x) = 2\,a\,F(x) - a \quad .$$

Let's now apply this theorem to our problem:

The probability variable x represents the offset Δ, ranging from $-a = -\frac{1}{2}$ to $a = \frac{1}{2}$. $f(x)$ is the distribution we want to smooth and $g(\xi) = \frac{1}{2\,a} = 1$ is the smoothed distribution obtained by the transformation

$$\Delta \rightarrow \tilde{\Delta} = \Xi(\Delta) = \int_{-\frac{1}{2}}^{\Delta} f(t)\,dt - \frac{1}{2} \quad .$$

In practice the situation is slightly different because our initial offset distribution is discrete (a histogram). To obtain the transformation from a discrete distribution f_i, the integral is replaced by a sum:

$$\Xi(\Delta_k) = \sum_{i=1}^{k} f_i - \frac{1}{2} \quad ,$$

where $f_i = \frac{n_i}{\sum_j n_j}$; n_i is the number of entries in the i-th bin.

Figure 14 shows an example for a measured transformation function. To transform then a particular measured Δ, we interpolate between the discrete values $\Xi(\Delta_k)$.

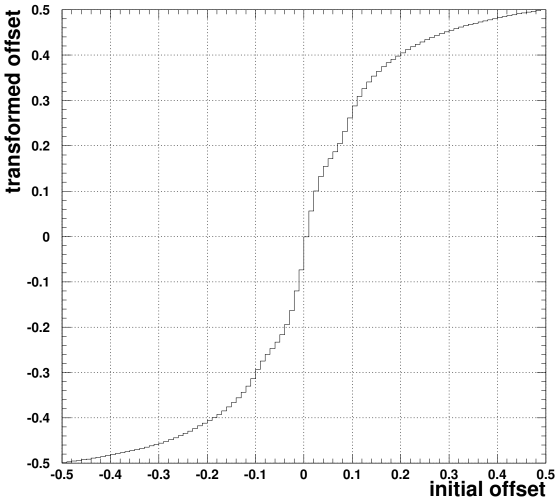

Figure 14: Offset transformation function Ξ

Important properties of the transformation Ξ are its continuity and monotony. They assure that neighbouring bins will be neither swapped nor separated by a gap. The inversibility prevents information getting lost in the transformation.

For 1-strip-clusters however the algorithm explained above doesn't work:

Due to the lack of position information these events all get $\Delta = 0$ producing a singularity at this value (i.e. a high spike in our discrete histograms of f_i, see Figure 13a), contradicting requirement iii. Thus the transformation $\Xi(\Delta$ will have a step (discontinuity) at $\Delta = 0$. It is not well defined in this point. Defining $\Xi(0)$ manually, e.g. as the middle of the step, doesn't solve the problem, because Ξ will attribute to all the entries in $\Delta = 0$ the same value $\tilde{\Delta} = \Xi(0)$. The spike would survive the transformation.

One still can get rid of this systematic error by accepting some additional statistical error:

Before the normal smoothing procedure the offsets of 1-strip-clusters are randomized using a Gaussian distribution. The width σ_Δ of the latter is chosen such that after the subsequent smoothing transformation the rms offset of 1-strip-clusters is $\frac{\text{stripwidth}}{\sqrt{12}} = 13\mu\text{m}$. A typical value for σ_Δ is 0.1 ($\hat{=}$ 5 μm). It is fixed in the declaration part of analstel.c.

The result of the full procedure is shown in Figure 13b.

4.4 The Track Fit

The determination of a unique track through our configuration of 2 x-planes and 4 y-planes imposes some conditions on the number of hit clusters per plane:

1. Since we have only 2 detectors for the x-coordinate, a unique track can only be found if in each x-plane there is exactly 1 cluster. (However, if the user doesn't need any second-coordinate information, he can disable this selection criterion in the geometry file. Then no x-track is calculated.)
2. For the y-coordinates of the track we require at least 1 cluster in at least 3 planes. The maximum cluster number as given in the geometry file however must not be exceeded in any of the planes.

For events satisfying 1. and 2. the program chooses the appropriate fitting routine according to the number of y-planes with valid clusters (3 or 4). In order to find the correct track, all hit combinations are fitted. However, track candidates are only accepted if their χ^2 doesn't exceed the limit specified in the geometry file. Finally the track with the least χ^2 is selected.

At last a track quality cut is applied: For a good track (i.e. `steltrk.quality` = 1) the track fit probability is required to be greater than a value defined in the geometry file. A reasonable value is 0.05. If the best track of the event fails this selection, `steltrk.quality` is set to -1, but the track is nevertheless passed to the calling routine.

If no track candidate passes the χ^2-preselection, `steltrk.quality` is set to 0, and the other members of the `steltrk` structure are meaningless.

5 Constants and Data Structures

These definitions are given in **stel.h** (for both MUTANT and MUT8):

```
#define N_STEL_PLANES 6
#define N_STEL_X_PLANES 2
#define N_STEL_Y_PLANES 4
#define N_CHAN_PER_PLANE 1024
#define N_STEL_CHANS (N_STEL_PLANES * N_CHAN_PER_PLANE)
#define N_CHAN_1DETBOX 1024
#define N_CHAN_2DETBOX 2016
#define N_STRIPS_PER_DETECTOR 996
#define N_CHIPS_PER_HYBRID 8
#define STEL_PITCH 0.05
#define N_SMOOTH 100
#define N_CHAN_PER_CHIP 128
#define N_STEL_BOXES 4
#define MAX_CLUSTERS 100
#define MAX_CLUSTERSIZE 50
#define MAX_STEL_RAWDATA 10000

typedef struct{
  int    version,
         adcThreshold[N_STEL_PLANES],
         x_readoutOrder,       /* readout order w.r.t. x-axis */
         y_readoutOrder,       /* readout order w.r.t. y-axis */
         maxcluster,           /* maximum number of clusters per plane */
         stripqual[N_STEL_PLANES][N_CHAN_PER_PLANE], /* strip quality */
         minclcharge,           /* min. cluster charge (ADC channels) */
         do_xtrack,                 /* second-coordinate tracking? */
         alive[N_STEL_PLANES];      /* plane alive ? */
  double xOffset[N_STEL_PLANES],    /* x-position of plane centre */
         yOffset[N_STEL_PLANES],    /* y-position of plane centre */
         z_x[N_STEL_X_PLANES],      /* z-position of x-plane centre */
         z_y[N_STEL_Y_PLANES],      /* z-position of y-plane centre */
         slope_x[N_STEL_X_PLANES],  /* tilt dy/dx of x-planes */
         slope_y[N_STEL_Y_PLANES],  /* tilt dy/dx of y-planes */
         resolution_x[N_STEL_X_PLANES], /* single plane resolution (x) */
         resolution_y[N_STEL_Y_PLANES], /* single plane resolution (y) */
         smooth[N_STEL_PLANES][N_SMOOTH + 1],   /* smoothing coeff. */
         chipscaling[N_STEL_PLANES][N_CHIPS_PER_HYBRID], /* ADC scaling */
         maxchi2raw,     /* maximum acceptable chisqr for raw track */
         minfitprob;     /* minimum acceptable track fit probability */
} StelGeom;
```

```
typedef struct {
  int    plane,                 /* plane number (0 to 5) of hit */
         strip,                 /* strip number */
         adc;                   /* ADC value */
} StelHit;

typedef struct {
  int    plane,                 /* plane number */
         charge,                /* sum of ADC values */
         quality,               /* 1 = good, 0 = bad */
         size,                  /* number of strips in the cluster */
         biggesthit;            /* index of cluster hit with maximum ADC */
  double offset;                /* position of track w.r.t. highest hit */
  StelHit *hitp[MAX_CLUSTERSIZE];    /* array of pointers to member hits */
} StelCluster;

typedef struct {
  int    quality;       /* 0 = no track, 1 = good track, -1 = bad track */
  Track  x,             /* second coordinate track (2 planes) */
         y;             /* precision coord.  track (4 planes) */
  double sumz,          /* from linfit.c: sum (z_i)      */
         sumz2,         /*                sum (z_i)**2   */
         sumw,          /*                sum (weight_i) */
         det,           /*                sumw * sumz2 - sumz * sumz */
         chi2;          /*                chisqr         */
} StelTrack;
```

The structure "Track" is defined in **~hessey/public/mutant/new/mutant.h** (MUTANT) or **stel.h** (MUT8):

```
typedef struct {
  double m,             /* slope */
         c;             /* intercept */
} Track;
```

In MUT8 **stel.h** has in addition the definition

```
#define BANK_HEADER_LENGTH 3
```

whereas for MUTANT the corresponding definition

```
#define BANK_HEADER_LENGTH 4
```

is given in **~hessey/public/mutant/new/constants.h**.

In MUTANT the pointer to the STEL-bank “stel” is an element of the structure “BankPtrs”, whereas in MUT8 it belongs to the “Event” structure (see **mutant.h** in N. Hessey’s **public/mut8/new** directory). The STEL-bank itself has the following composition:

– MUTANT:

```
typedef struct {
        unsigned short length;
        char  name[4];
        unsigned short version;
        short data[MAX_EV_LEN - 3];
} Bank;
```

with the maximum event length defined as

```
#define MAX_EV_LEN (1 << 15)
```

– MUT8:

```
typedef struct {
           int  length,
                id;
           char name[4];
           int  data[1];
} Bank;
```

Questions and Feedback

If you
- have suggestions for the improvement of the algorithm,
- find bugs in the code,
- find mistakes in the manual,
- don’t understand the manual,

contact: Mario.Deile@cern.ch

Acknowledgements

We would like to express our thanks to numerous people for their support in the construction and maintenance of our beam telescope:

First of all special thanks to Gerhard Lutz and Rainer Richter from the semiconductor laboratory of MPI Munich for contributing and testing the Silicon microstrip detectors. They also gave us valuable help in trouble shooting.

Ogmundur Runolfsson (CERN) and his colleagues bonded the detectors to the VA chips which was a difficult task due to the missing bonding pads. Later a complete hybrid had to be quickly rebonded. When we carefully asked whether they could do it, their answer was "There is always time for important things".

Christian Regenfus (Univ. of Munich, now in Zürich) adapted the repeater card electronics to the specific requirements of our VA2 hybrids. His explanations about the readout electronics were highly appreciated.

We are very happy to know Alan Rudge (CERN), the designer of our repeater cards, and Robert Boulter (CERN), who never hesitated to lend us immediate assistance when we had hardware troubles in busy periods of data taking. Upon Alan's diagnosis of broken bonds, Bob rescued precious beam time by immediately doing the urgent repair. He also contributed to the initial construction by glueing the hybrids to the detectors.

And of course we don't forget the support from our technicians Horst Herbert and Rolf Öhm (Univ. of Munich) who made the detector frames and the base plate.

References

[1] N.P. Hessey, MUTANT User Guide, January 1996, available on the WWW-page: http://www.etp.physik.uni-muenchen.de/atlas/testbeam.html

[2] N.P. Hessey, The ATLAS Muon Beam Test Data Acquisition System, 2nd Version, August 1995, available on the WWW-page: http://www.etp.physik.uni-muenchen.de/atlas/testbeam.html

[3] G. Polesello, DAQ shift guide for H8 Muon testbeam, modified rd13 note 169, July 1996

[4] J. Dubbert, Bestimmung der Ortsauflösung von Hochdruckdriftrohren für das Myon-Spektrometer des ATLAS-Experimentes, diploma thesis written in german, LMU München 1996, chapter 8 and appendix D

[5] CAEN 2-Channel C-RAMS, Mod. V 550, Technical Information Manual, September 1994

[6] The VA circuit, Integrated Detector & Electronics AS, Veritasveien 9, N-1322 Høvik, http://www.ideas.no

[7] A. Rudge, Some Notes on the System of Readout of the Viking Chip, CERN, September 1994