Geometry and Alignment Software for the CLAS12 Silicon Vertex Tracker

P. Davies[1], V. Ziegler[2], M. Ungaro[2], Y. Gotra[2], A. Kim[3], and G.P. Gilfoyle[4]

[1] *University of Surrey, Guilford, UK*
[2] *Jefferson Lab, Newport News, VA*
[3] *University of Connecticut, Storrs, CT*
[4] *University of Richmond, Richmond, VA*

August 9, 2017

Abstract

The CLAS12 detector is currently under construction in Hall B as part of the CEBAF 12 GeV Upgrade. The Silicon Vertex Tracker (SVT) is a position sensitive detector subsystem in CLAS12, and is the closest one to the target. This document is a guide to the geometry of the SVT, and the software used to model it for simulation and reconstruction. The sensors of the SVT consist of long, narrow strips of p-type silicon with aluminum electrodes on an n-type, bulk silicon substrate. There are 256 strips in a sensor, with a readout pitch at the upstream end of 156 μm, and a stereo angle of $0 - 3\,^{\circ}$. The location of the sensor strips must be known to a precision of a few tens of microns in order to accurately reconstruct particle tracks with the required position resolution of 60 μm specified in the CLAS12 design. The geometry of the SVT has been well defined according to the design specification after consultation with the design team, and software was been developed to align the sectors using real cosmic data. The simulation and reconstruction software now receive the same geometry from one source.

Abbreviations

API	Application Programming Interface
BMT	Barrel Micromegas Tracker
BST	Barrel Silicon Tracker
CAD	Computer Aided Design
CCDB	CLAS Constants DataBase
CEBAF	Continuous Electron Beam Accelerator Facility
CLARA	CEBAF Large Acceptance Spectrometer
CND	Central Neutron Detector
CTOF	Central Time Of Flight
CVT	Central Vertex Tracker
DC	Drift Chamber
DOCA	Distance Of Closest Approach
EC	Electromagnetic Calorimeter
FMT	Forward Micromegas Tracker
FTOF	Forward Time Of Flight
GDML	Geometry Description Markup Language
GEANT	Geometry and Tracking
GEMC	Geant4 Monte-Carlo
HFCB	Hybrid Flex Circuit Board
HTCC	High Threshold Cherenkov Counter
LTCC	Low Threshold Cherenkov Counter
PCAL	Pre-shower Calorimeter
SVT	Silicon Vertex Tracker
XML	Extensible Markup Language

Contents

1 Software Background

1.1 Overview of Reconstruction Software

Figure 1 shows an overview the Java library written specifically for the CLAS12 detector, called the CLAS Offline Analysis Tools (COATJAVA) [1]. The library consists of various packages for simple applications such as file I/O, plotting and geometry, as well as event simulation, viewing, reconstruction, and analysis. The source code is developed in a Github repository at `https://github.com/JeffersonLab/clas12rec`.

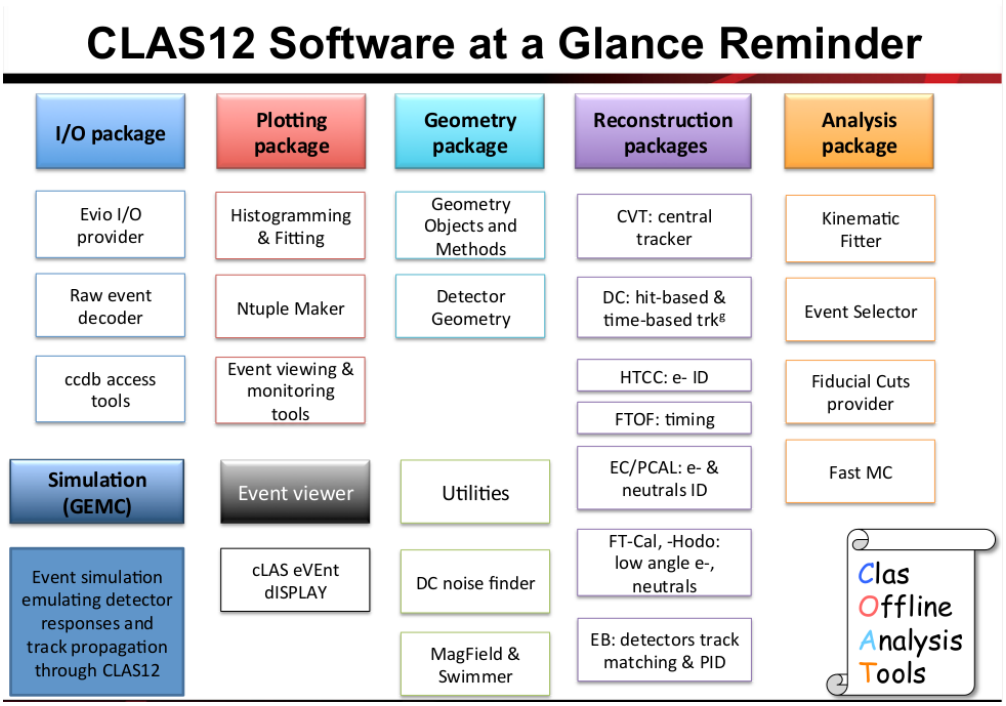

Figure 1: Overview of the packages in COATJAVA.

The CLAS12 Reconstruction and Analysis (CLARA) framework manages services that perform specific functions during event reconstruction, such as providing geometry for the sensors and calculating the properties of particle tracks. CLARA is able to take advantage of hyper-threading to improve performance.

A special file format called EVIO (Event Input Output) is used to store the raw signals captured by the detectors [2]. A different format Hipo (High Performance Output) is used for the reconstructed data in banks of information, such as voltage peaks in sensor strips and possible tracks of particles. [3]

Figure 2 shows a flowchart of the Reconstruction process. The first service is the Event Reader, which reads an EVIO file from disc. Each forward detector has standalone Hit Based Tracking (HBT) and Time Based Tracking (TBT) services. HBT identifies spatial information about where particles have triggered hits in the sensors. TBT uses an iterative process to refine the initial hits from the HBT and calculate accurate

values for the event times and path lengths of tracks by "swimming" particles through the magnetic field. If a good fit is found, then that reconstructed particle is added to the event, and the process begins again, this time using the knowledge that some hits now belong to a possible particle track. The CVT and CTOF provide additional HBT information for large angle tracks.

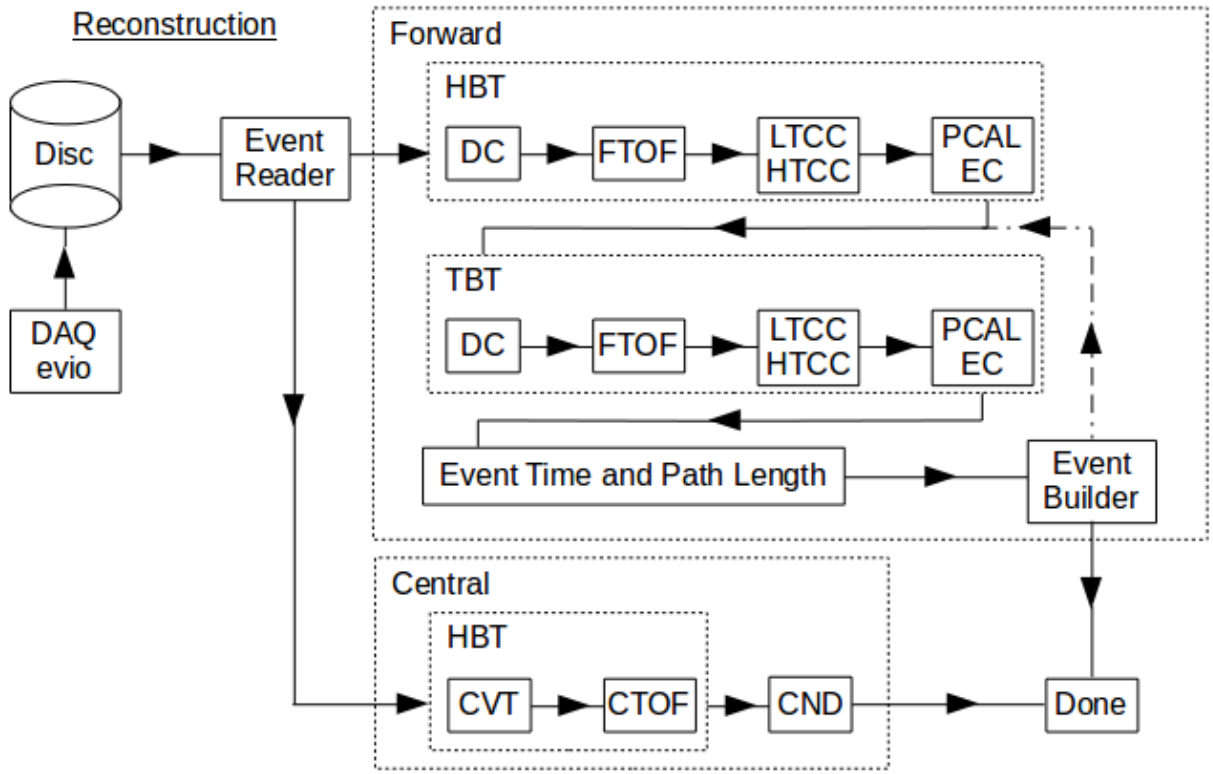

Figure 2: Flowchart of the reconstruction process. Each solid box represents a service in CLARA. The Forward and Central service blocks can run in parallel.

1.2 Simulation of the CLAS12 Detector: GEMC

The simulation software for the CLAS12 detector is called GEMC (GEANT4 Monte Carlo). [4] It is able to model the geometry of the target and detectors, the response of the detector to the passage of particles, calculating how they interact with the environment at each time step, including multiple scattering, and produce raw data in the same format as the DAQ in CLAS12 in order to calibrate the Reconstruction.

GEMC uses Perl scripts to build the structure and properties of the detector geometry. [5] For the SVT, this process starts with the Perl script `bst.pl` (The SVT was previously called the Barrel Silicon Tracker).

2 User Manual

This document is designed to be a guide to the geometry of the Silicon Vertex Tracker (SVT), one of the detector subsystems in the CLAS12 detector, and the software used to model it for simulation and reconstruction.

2.1 Ideal Geometry of the Silicon Vertex Tracker

The SVT consists of 132 identical silicon layers grouped in pairs and assembled in sectors to form regions. Figure 3 shows the final form of the detector. Additional details are available in the documents at `https://userweb.jlab.org/~pdavies/doc/`. Figure 4 shows a photograph of the SVT during assembly.

Figure 5 shows the right-handed Cartesian coordinate system of the lab frame and the numbering convention of the SVT sectors. The z-axis points downstream along the beamline into the page, the y-axis points upwards, and the x-axis points to the left when looking downstream. Spherical polar coordinates are used for rotations, with r the radius from the beamline, θ the angle of elevation from the z-axis, and ϕ the angle of azimuth from the x-axis. The centre of the target is ideally located at the origin.

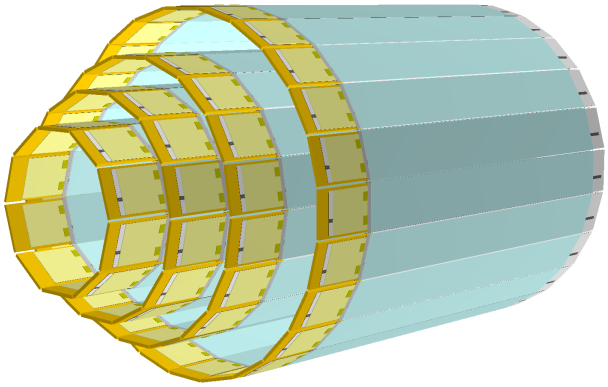

Figure 3: Visualization of the SVT in ROOT. (not showing wirebond between sensor cards)

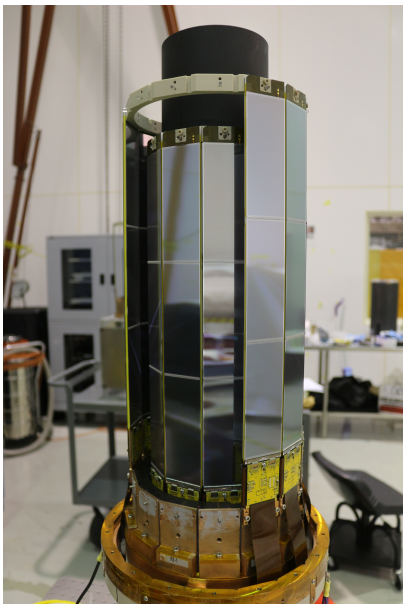

Figure 4: Photograph of the SVT during assembly.

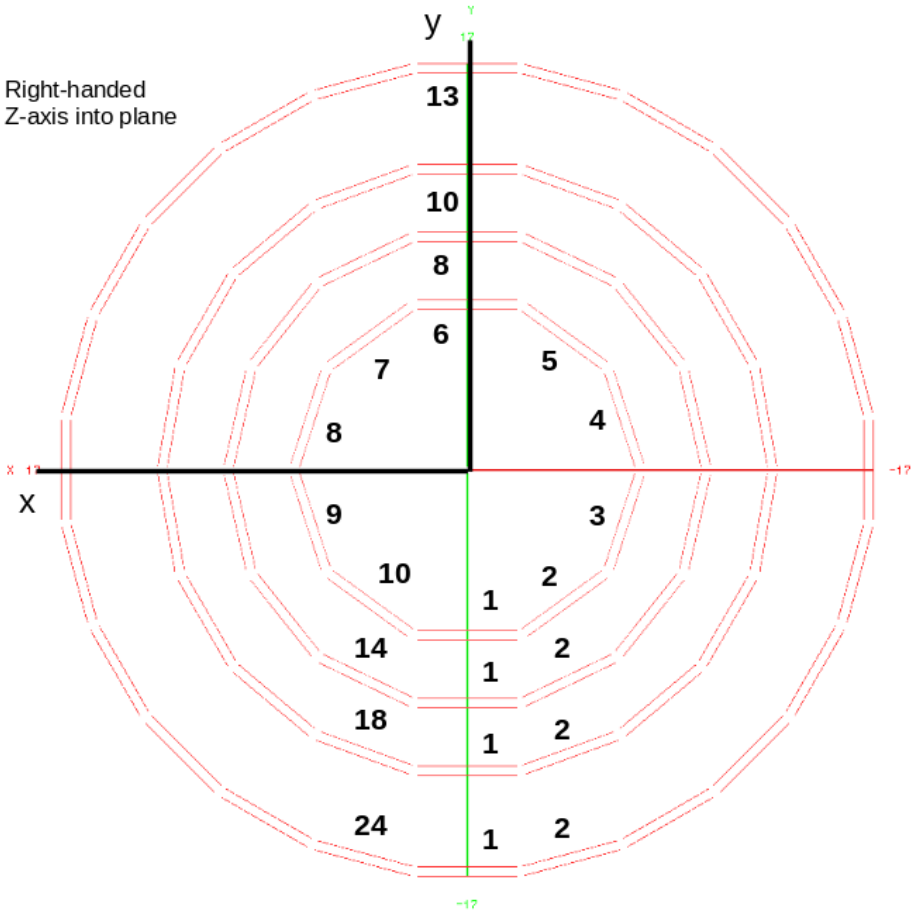

Figure 5: Diagram showing sector numbering convention. The sectors are arranged around the target in 4 concentric circles, called regions. Sector 1 for all regions is located at the bottom of the cylindrical structure ($x = 0, y < 0$). Up is towards the top of the page here (Sector 13 in Region 4).

2.1.1 Sensor Strips

There are 256 sensor strips in each sensor layer, arranged on a stereo angle up to 3 °, and extending over 3 rectangular sensor cards each of size 111.62 mm x 42 mm and thickness 320 µm down the beamline, named Hybrid, Intermediate, and Far, respectively. See Figure 6. The cards were cut from 6 in. wafers, two sensors per wafer. [6]

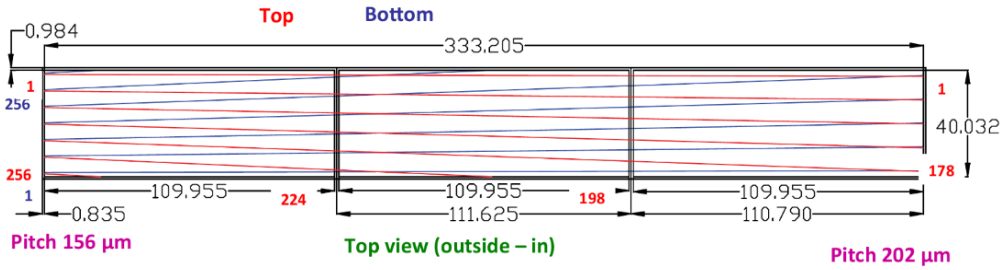

Figure 6: Plan view from outside the top of the SVT of a sensor module (region 1, sector 6, for example), showing the strips of module U in red, and V in blue. Units are mm.

The stereo angle increases linearly from 0 ° on the first strip to 3 ° on the final one. Figure 6 shows this structure for two sensor layers superimposed on top of one another.

Figure 7 illustrates how the strips are not completely continuous. Each sensor card contains a rectangular active zone where the strips are engraved, surrounded by dead zones, and there is a small air gap between the cards, called a micro-gap. A triplet of cards is called a sensor module, or just a module.

Pairs of sensor modules are grouped together, one flipped around the z-axis so that the strips overlap when viewed from above or below, to form a sector module, or just a sector. See Figure 6. The strips and modules are arranged this way so that a particle will pass through both layers and generate two signals. The end points of the strips that fired are used to generate two lines using the geometry. The crossing point of these lines defines the z-position.

The two modules in each sector are labeled U and V, for inner and outer respectively (in alphabetical order when looking from the inside out).

Figure 7: Diagram showing module structure along z axis.

Figure 8 shows a schematic needed to describe the method used to generate the strips. The start point is always along the x axis, but the end point may be on the far side (downstream) or on the side of the module along the z axis of the strip coordinate system. The strip is converted to the 'local' coordinate system by a translation along the x axis to put the z axis in the centre of the module. The U layer is generated by default. For the V layer, a strip from the U layer is simply rotated about the z axis by 180 ° in the local frame.

There are actually 512 strips etched into the silicon, but only half of them are used for reading data. As a result, they alternate between sensitive (readout) and intermediate (unused) when looking along the x axis.

The horizontal position w of the start of a strip from the upstream edge along the x axis is given in Equation (1), where d is the distance to the first sensitive strip (parameter `STRIPOFFSETWID`), p is the spacing between the sensitive strips (parameter `READOUTPITCH` with value 0.156 μm), and s is an index for the strip counter with values [0:255]. The width of the layer W is equal to $2d + Np$ (not the same as the width of the module because of dead zones).

$$w = d + s \times p \tag{1}$$

The angle α each strip makes with the horizontal is given by Equation (2), where α_{max} is the maximum stereo angle in radians (3°), and N is the number of sensitive strips in a layer.

$$\alpha = \alpha_{max}/N \times s \tag{2}$$

The x component of the distance from the projected end of the strip to the start point is call q and is given by Equation (3), where L is the length of the layer (parameter `STRIPLENMAX`). This is not the same as the length of the module (parameter `MODULELEN`) because of the dead zones at either end of the sensor cards).

$$q = L \tan \alpha \tag{3}$$

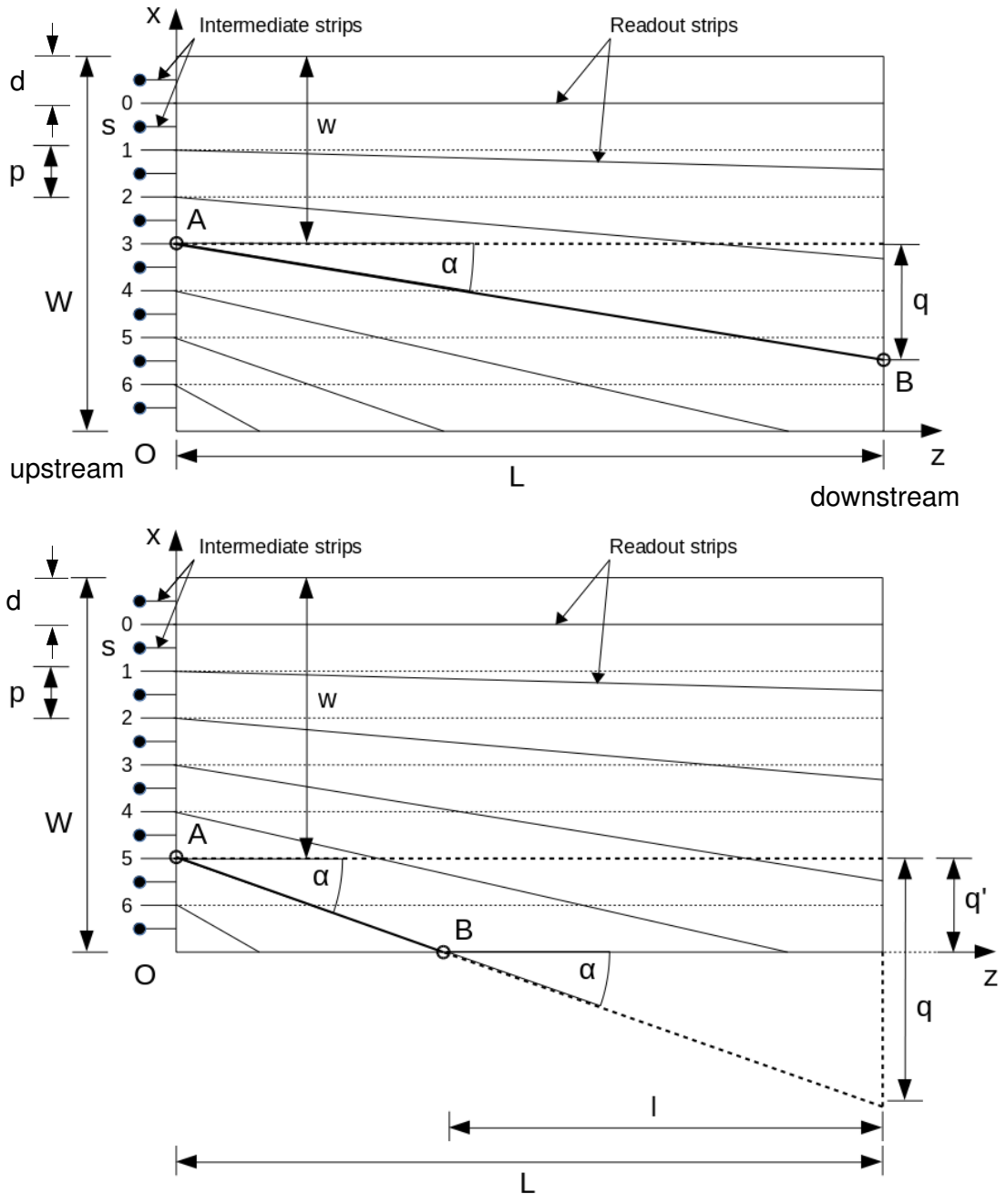

Figure 8: Layer U in the strip coordinate system. Beam comes from the left (upstream).

Top: The strip ends on the far, downstream edge, showing 7 strips for clarity. The horizontal position w of the start of a strip from the upstream edge along the x axis is given in Equation (1), where d is the distance to the first sensitive strip, p is the pitch or spacing between the sensitive strips), and s is an index for the strip counter. The quantity q is the change in x from the strip start point to the downstream end point.

Bottom: Second case where the strip ends on the side of the module (not the downstream side). The length q' is the portion of q that resides on the actual layer for the bottom case, and q for the top case. The value of $q' - q$ is used to identify each case, with $q' - q = 0$ for the first, and $q' - q < 0$ for the second.

For the case where the strip ends on the far side downstream, the start point $A(x, z)$ is given by Equation (4) where W is the width of the layer and w is the distance from the edge as shown in Figure 8.

$$x_A = W - w \tag{4a}$$

$$z_A = 0 \tag{4b}$$

The end point is given by $B_1(x, z)$ by Equation (5).

$$x_B = x_A - q \tag{5a}$$

$$z_B = L \tag{5b}$$

For the case where the strip ends on the long side and not on the downstream edge, the start point $A(x, z)$ is the same as the first case, given by Equation (4), but the end point $B_2(x, z)$ is given by Equation (6).

$$x_B = 0 \tag{6a}$$

$$z_B = x_A / \tan\alpha \tag{6b}$$

2.1.2 Backing Structure

Figure 9 shows a cross section of a sector. Each pair of sensor modules is supported by a backing structure, consisting of a carbon fiber base, a layer of bus cable to ground the carbon fiber, and a copper rail with 9 pads to supply the high voltage bias. The rail is embedded in the layer of epoxy glue that binds the silicon sensors to the pads and bus cable. Figure 11 shows the locations of the pads.

Figure 10 shows a photograph of a test module. At the upstream end, a pitch adapter sits between the module and a Hybrid Flex Circuit Board (HFCB) (labeled pcBoard in the figure), with wirebond connecting the 256 readout strips on the silicon to the pitch adapter, and then from the pitch adapter to the 2 electronic chips on the board, which each take input from 128 strips. A copper heatsink is mounted between the carbon fiber layers under the HFCBs up to the start of the modules, with the separation maintained by rohacell foam along the rest of the length. At the downstream end, a plastic mount with dowel holes is placed as an insert into the foam. Each region is mounted on a copper support ring by bolts on the copper heatsink, and a light plastic ring, made of peek material, is used to maintain the circular shape at the downstream end, which is free-standing. A convenient convention is to call the upstream end of the SVT the copper end, and the downstream end the peek end. [6] Dimensions for the volumes of these materials are located in Appendix

A.

Figure 9: Diagram showing the cross section of a sector module in the tracking volume. The silicon sensor modules would be fixed to the top and bottom with the epoxy glue.

Additional figures describing the ideal geometry of the SVT can be found in the previous Geometry Document. [7] Technical drawings for the components are available here. [8] Figure 11 features one of the drawings for the bus cable assembly located in the filesystem with `cd /group/hallb_inst_svt/_1\ SVT\ Latest\ Documents/_17\ Bus\ Cable/Bus\ Cable\ Panel\ V3.3/Mechanical`.

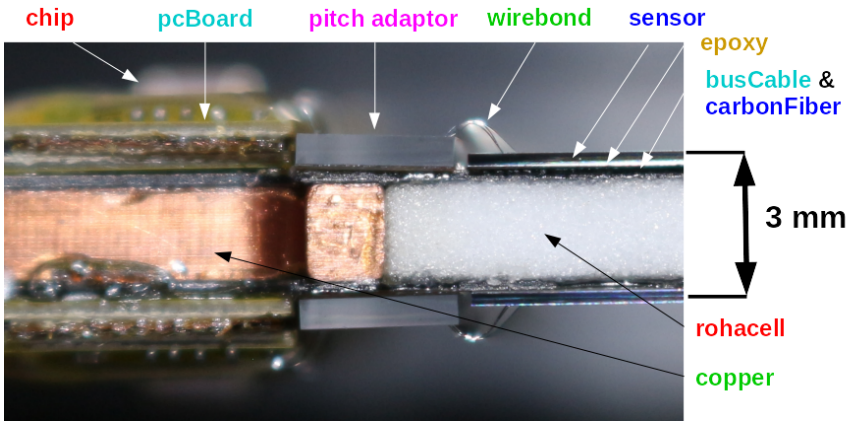

Figure 10: A close-up photograph of the side of a module. GEMC uses the material names for some components, such as copper for the heatsink and pcBoard for the HFCB. The kapton tape that wraps around the side to protect the rohacell is not shown.

There are 3 fiducial points on each sector, highlighted in Figure 12. Two are on the upstream copper support on either side of the x-axis (Cu1 and Cu2), and one on the downstream peek support (Pk3), offset

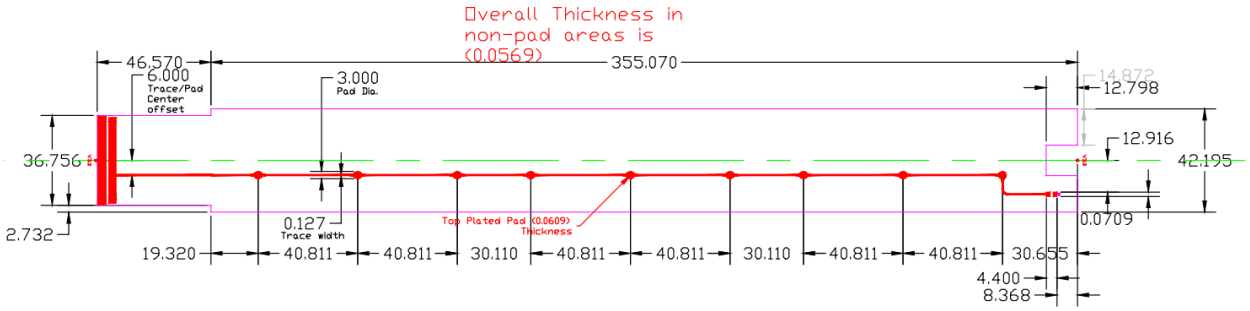

Figure 11: Drawing of the high voltage rail and pads.

to the right. They are used to accurately locate the position and orientation of the sector modules. Figure 13 shows a cross section of the SVT in the XY plane, with an overlay of one sector.

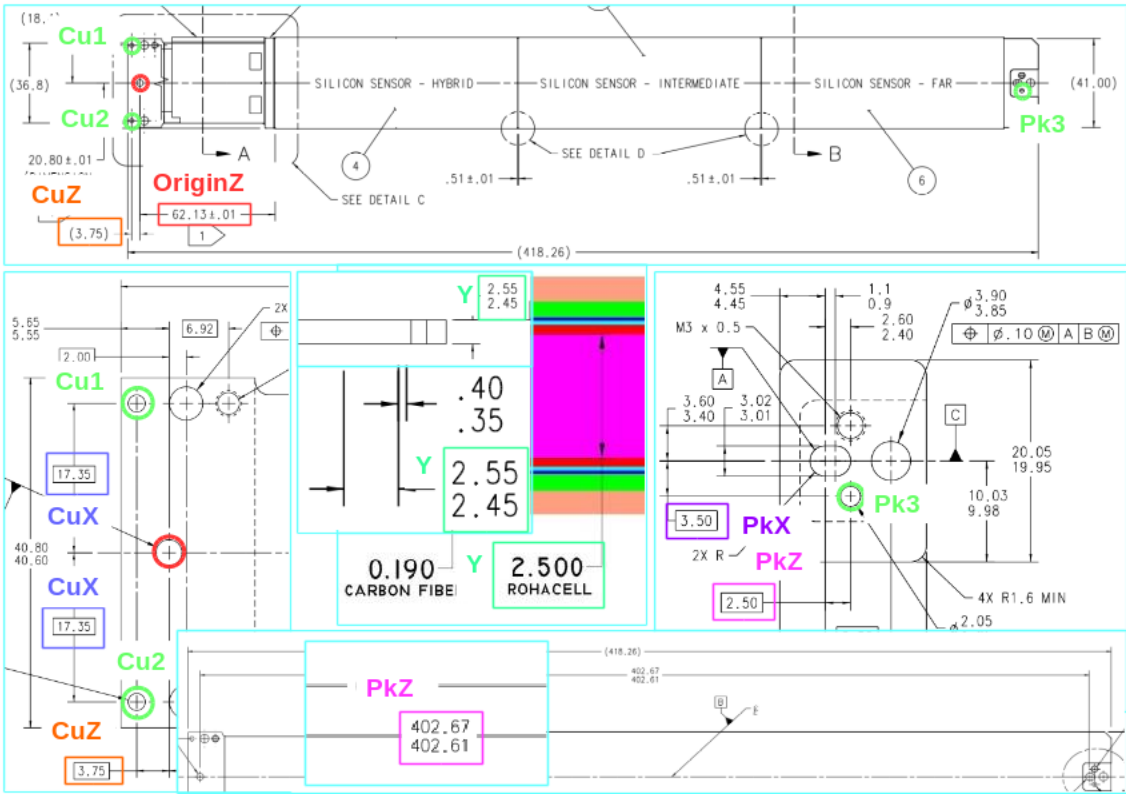

Figure 12: Technical drawings of a sector module with highlights for the fiducial points and values used to locate them. Note that Pk3 is off-center. [8]

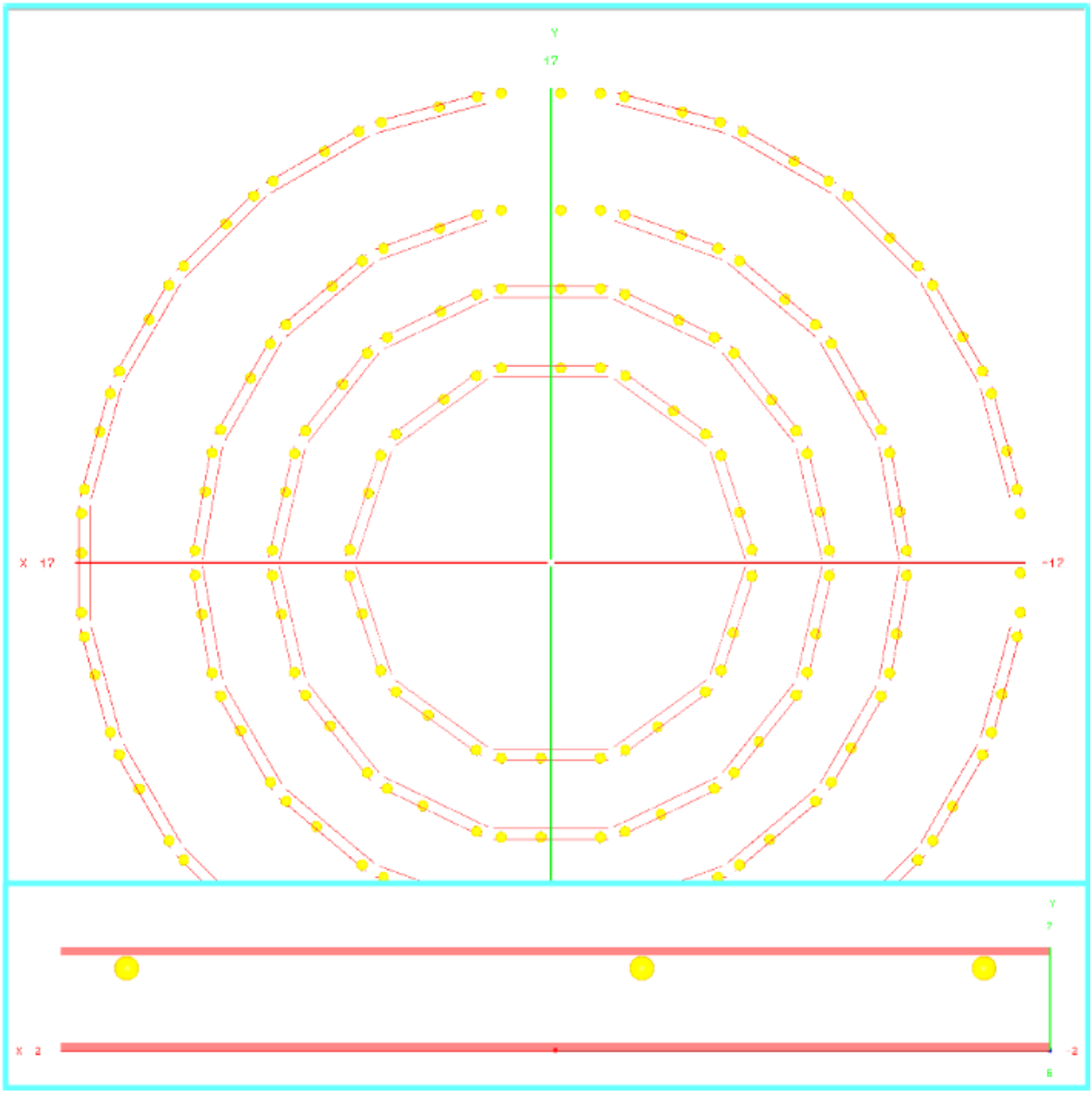

Figure 13: Cross section of the SVT in the XY plane (Z axis into page), with an overlay of one sector. Modules shown in red, and fiducial points in yellow.

2.2 CCDB :: Core Parameters

A minimal set of core parameters (also known as constants) are used to generate the entire geometry. They are recorded in the CLAS Constants Database (CCDB) in the directory `/geometry/cvt/svt/`. The following list gives the name of each table, and includes a brief description of the constants contained within. See Appendix A for a detailed example of how to use the interactive interface and values for a set of alignment shifts calculated from the fiducial survey and volume dimensions in tables `alignment`, `box`, and `tube`.

- `svt` Fundamental parameters shown in Table 1, such as the number of regions (NREGIONS), and constants used to orient the structure in space (PHI0, SECTOR0).
- `region` Arrays of region specific parameters shown in Tables 2 and 3, such as the number of sectors (NSECTORS), the z-position of the start of the layers (Z0ACTIVE), and two radial constants (REFRADIUS and SUPPORTRADIUS).
- `fiducial` Parameters for the location of the fiducial points on a sector.
- `alignment` A set of alignment shifts calculated from analysis of the fiducial survey results.
- `material/box` Volume dimensions of passive materials in the backing structure.
- `material/tube` Volume dimensions of passive materials in the backing structure.

In Table 3, REFRADIUS was inherited from an outdated version of the SVT tables, and pointed to a reference point on the inner side of the U layer's silicon volume (inner as in towards the target). SUPPORTRADIUS points to the inner side of the heatsink, where it connects to the copper support ring. The former was derived from previous calculations that were lost, so the latter was read from current technical drawings instead. The values of REFRADIUS have been updated using the new values of SUPPORTRADIUS, to maintain backwards compatibility.

Table 1: Fundamental Constants. Length unit: mm. Angle unit: degrees.

Parameter	*Value*	Description
NREGIONS	4	Number of regions in the SVT
NMODULES	2	Number of modules in a sector
NLAYERS	8	Number of layers along the radius of the SVT
NSENSORS	3	Number of sensors in a module
NSTRIPS	256	Number of strips in a module
NFIDUCIALS	3	Number of fiducial points in a sector
NPADS	8	Number of pads along the HV rails in a sector
NTOTALSECTORS	66	Total number of sectors in the SVT
NTOTALFIDUCIALS	198	Total number of fiducial points in the SVT
PHI0	270.000	Starting rotation around the z-axis of sector modules
SECTOR0	90.000	Starting orientation around the z-axis of sector modules
STEREOANGLE	3.000	Angular spread of the sensor strips in a layer
READOUTPITCH	0.156	Distance between start of strips along front of layer
STRIPOFFSETWID	0.048	Offset of first intermediate sensor strip from edge of active zone
STRIPLENMAX	333.429	Maximum length of a strip
LAYERPOSFAC	0.500	Fractional location of sensor layer along height within a module
PHYSSENLEN	111.625	Length of physical sensor
SILICONTHK	0.320	Thickness of module (made of silicon)
PHYSSENWID	42.000	Width of physical sensor
ACTIVESENLEN	109.955	Length of active zone in sensor
ACTIVESENWID	40.032	Width of active zone in sensor
DEADZNLEN	0.835	Length of dead zone in sensor
DEADZNWID	0.984	Width of dead zone in sensor
MICROGAPLEN	0.112	Width between sensors
MODULEWID	42.000	Width of sensor module
MODULELEN	335.099	Length of sensor module
LAYERGAPTHK	3.162	Internal thickness between modules
PASSIVETHK	0.331	Total thickness of passive materials between a module and rohacell
SECTORLEN	418.255	Length of a sector
FIDCUX	17.350	X position of copper fiducials (symmetric about z-axis)
FIDPKX	3.500	X position of peek fiducial
FIDORIGINZ	62.130	Z position of reference point for the fiducials
FIDCUZ	3.750	Z position of copper fiducials
FIDPKZ0	402.624	Z position of peek fiducials
FIDPKZ1	2.500	Z position of peek fiducials

Table 2: Region dependent parameters. NSECTORS: Number of sectors in a region. STATUS: Whether the region is to be used depending on the current configuration [0:1] (ignored by geometry service). Z0ACTIVE: Z-position for edge of first active zone in a module. Length unit: mm

REGION	NSECTORS	STATUS	Z0ACTIVE
1	10	1	-219.826
2	14	1	-180.380
3	18	1	-141.206
4	24	0	-83.405

Table 3: Region dependent parameters. REFRADIUS: Radial distance from origin to outer side of U (inner) module. SUPPORTRADIUS: Radial distance from origin to inner side of copper heatsink at it's thickest part. LAYERRADIUS: Radial distance from origin to middle of each sensor module. Length unit: mm.

REGION	REFRADIUS	SUPPORTRADIUS	LAYERRADIUS U	LAYERRADIUS V
1	65.447	65.403	65.287	68.769
2	93.047	93.005	92.887	96.369
3	120.482	120.435	120.322	123.804
4	161.362	161.315	161.202	164.684

Table 4: Fiducial Constants. Length unit: mm.

Parameter	*Value*	Description
CuX	17.350	Distance to copper fiducial point from centre line of sector.
PkX	3.500	Distance to peek fiducial point from centre line of sector.
OriginZ	62.130	Distance to reference point on sector from edge of module.
PkZ0	402.624	Distance to secondary reference point on sector for peek fiducial point from OriginZ.
PkZ1	2.500	Distance to peek fiducial point from secondary reference point.

2.3 SVT Geometry API

The SVT package consists of the following classes for use by the end user. Here we give a summary of the package, followed by more details and examples below. The flowchart in Figure 14 shows how they relate to each other through function calls.

- `SVTConstants`: Reads core parameters and volume dimensions from CCDB and calculates derived parameters and dimensions. Also reads alignment shifts from selected source (CCDB or file), if enabled.
- `SVTAlignmentFactory`: Generates ideal fiducial points and handles file I/O for fiducial data sets. Processes fiducial data into plane vectors to pass to `AlignmentFactory`.
- `SVTVolumeFactory`: Generates nested volume structure in GEANT4 format for simulation, and outputs it to file, along with some parameters for the PERL scripts used to generate the database entries called by GEANT4 functions in GEMC.
- `SVTStripFactory`: Generates start and end points of sensor strips for the Reconstruction, and layer corners for calibration and validation purposes.

The following general classes are used behind the scenes.

- `AlignmentFactory`: Calculates and applies alignment shifts to points and volumes.
- `Util`: Provides utility methods for visualizing vectors as volumes, manipulating volume properties, converting between the standard vector and GEANT4 rotation conventions, and file I/O.
- `Matrix`: Supports basic matrix algebra and general 3D rotation conversions.

The source code for the package is in a github repository at `https://github.com/drewkenjo/JCSG` with full package name `org.jlab.detector.geant4.v2.SVT`. A compiled version (December 16, 2016) of the binaries and documentation by the author are available to download from `https://userweb.jlab.org/~pdavies/dev/java/JCSG/`.

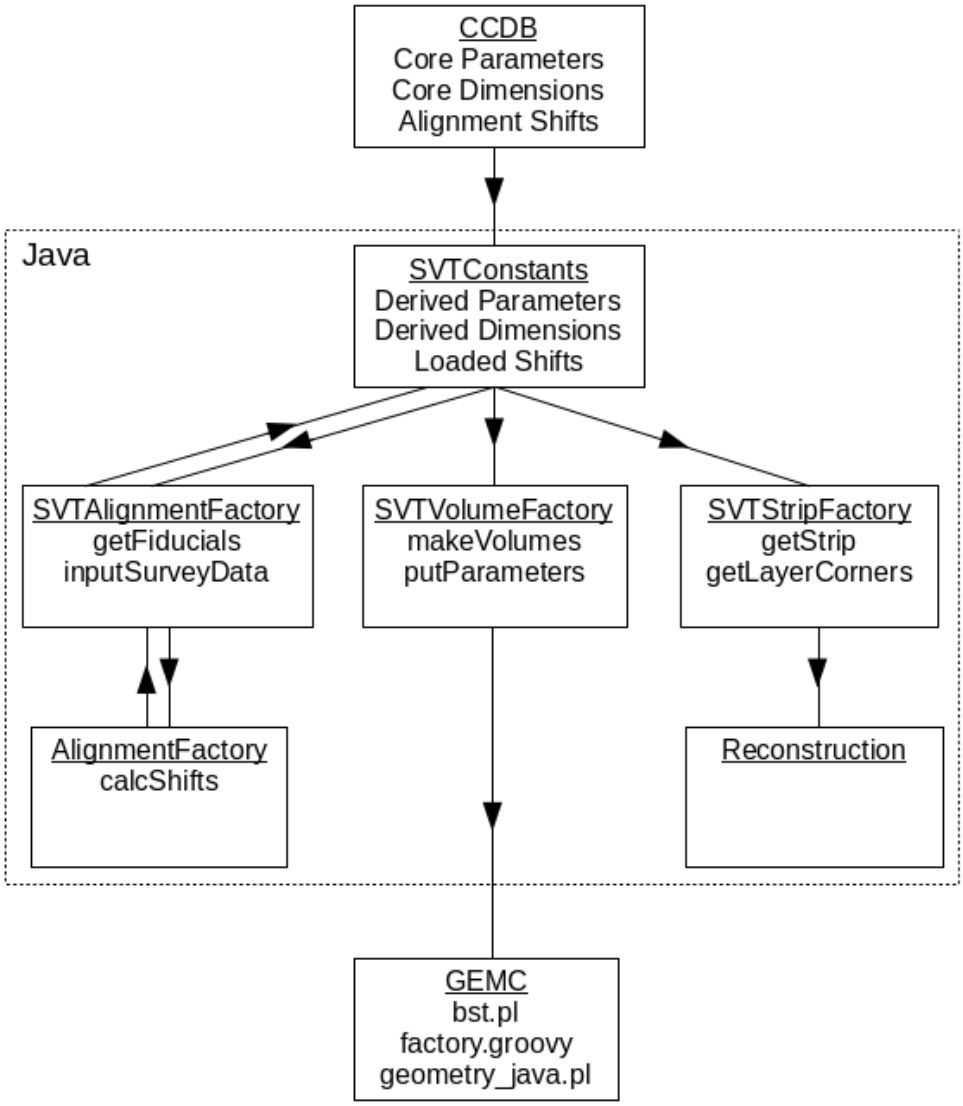

Figure 14: Flowchart of software packages.

2.3.1 SVTConstants :: Loading Parameters and Alignment Shifts

`SVTConstants` is a static class that handles parameters, conversions and transformations. The first step for any program using the service is to call `SVTConstants.connect` to setup the tables for the geometry parameters of the SVT in CCDB ready to be read. The `VERBOSE` switch prints out all the parameters. The other classes default to this setting and will print out their own debugging information accordingly.
Four indexing conventions are used by the software package.

1. `(Region, Sector, Module)[0:131]` These three indices are used for iterating the nested structure.
2. `(Region, Sector)[0:65]` Convert with `SVTConstants.convertRegionSector2Index` and `SVTConstants.convertIndex2RegionSector`.
3. `(Layer, Sector)[0:131]` Same as Convention 1, but combines regions and modules into layers for the alternative notation used by the Reconstruction. Convert with `SVTConstants.convertLayer2RegionModule` and `SVTConstants.convertRegionModule2Layer`.
4. `(Region, Sector, Fiducial)[0:197]` Similar to Convention 1, but iterates over the fiducial points instead of the modules for use by `SVTAlignmentFactory`. Convert with `SVTConstants.convertRegionSectorFiducial2Index` and `SVTConstants.convertIndex2RegionSectorFiducial`.

Listings 1-3 show examples of the different ways to load the core parameters for the ideal geometry and apply alignment shifts from the database or a custom file.

Listing 1 : Source code for initialising SVTConstants with ideal geometry.

```
SVTConstants.VERBOSE = true; // optional for debugging
DatabaseConstantProvider cp = new DatabaseConstantProvider( 10, "default");
cp = SVTConstants.connect( cp );
cp.disconnect();
```

Listing 2 : Source code for initialising SVTConstants using alignment shifts from CCDB.

```
SVTConstants.VERBOSE = true; // optional for debugging
DatabaseConstantProvider cp = new DatabaseConstantProvider( 10, "default");
cp = SVTConstants.connect( cp );
SVTConstants.loadAlignmentShifts( cp );
cp.disconnect();
```

Listing 3 : Source code for initialising SVTConstants using alignment shifts from file.

```
SVTConstants.VERBOSE = true; // optional for debugging
DatabaseConstantProvider cp = new DatabaseConstantProvider( 10, "default");
cp = SVTConstants.connect( cp );
cp.disconnect();
SVTConstants.loadAlignmentSectorShifts("shifts.dat");
```

The data files for alignment shifts use the format shown in Listing 4. The first column is a tag that uniquely identifies each of the 66 sector modules in the SVT using the RS index. The next three columns are the three components of the translation shift (x,y,z) in millimetres (mm). The last four columns define the components of the axis vector (x,y,z) and angle (in degrees) of the rotation shift, centred on the midpoint of the three ideal fiducial points for that sector. The axis does not need to be normalized.

Listing 4 : Format of alignment shifts in txt file.

```
R1S01    0.177    0.184    0.183    0.312    0.000    0.950    0.164
```

2.3.2 SVTVolumeFactory :: Generating Volumes

Listing 5 shows the use of `SVTVolumeFactory`, which is an object class that builds a tree of nested GEANT4 volumes used by the GEMC simulation. The first argument of the constructor takes the `ConstantProvider` used by `SVTConstants`. The second argument takes a boolean switch for whether the alignment shifts loaded in `SVTConstants` should be applied.

Listing 5 : Source code for using SVTVolumeFactory to generate the ideal geometry.

```
// for ideal geometry
SVTVolumeFactory svtIdealVolumeFactory = new SVTVolumeFactory( cp, false );
svtIdealVolumeFactory.VERBOSE = true; // optional for debugging
svtIdealVolumeFactory.makeVolumes();
System.out.println( svtIdealVolumeFactory.toString() ); // optional for debugging
```

Listing 6 shows the `setRange` method, which is used for debugging. It sets which regions, sectors and modules to generate in makeVolumes. The indices from start from 1, but 0 can be entered to use the previous or default value. There are methods to return the current minimum and maximum values for the indices: `getLayerMax`, `getLayerMin`, `getModuleMax`, `getModuleMin`, `getRegionMax`, `getRegionMin`, `getSectorMax`, `getSectorMin`.

Listing 6 : Source code for using setRange.

```
// example of default ranges: all regions, sectors, and modules
setRange( 1, SVTConstants.NREGIONS,
          new int[]{ 1, 1, 1, 1 }, SVTConstants.NSECTORS,
          1, SVTConstants.NMODULES );
// selecting only region 1 using overload (region, sectorMin, sectorMax)
svtIdealVolumeFactory.setRange( 1, 0, 0 );
```

2.3.3 SVTStripFactory :: Generating Strips

Listing 7 shows how to use `SVTStripFactory`, which is an object class that returns the start and end points of the sensor strips used by the Reconstruction. The first argument of the constructor takes the `ConstantProvider` used by `SVTConstants`. The second argument takes a boolean switch for whether the alignment shifts loaded in `SVTConstants` should be applied.

The following code snippet loops through each module, sector and region and uses `Util.createArrow` to display the strips as cylinders.

Listing 7 : Source code to visualise strips and layer corners from SVTStripFactory.

```
double stripArrowCapRadius = 0.5, // mm
    stripArrowPointerRadius = 0.25, // mm
    cornerDiscRadius = 0.075; // cm
SVTStripFactory svtIdealStripFactory = new SVTStripFactory( cp, false );
for( int region = svtIdealVolumeFactory.getRegionMin()-1;
         region < svtIdealVolumeFactory.getRegionMax(); region++ ){
    for( int sector = svtIdealVolumeFactory.getSectorMin()[region]-1;
             sector < svtIdealVolumeFactory.getSectorMax()[region]; sector++ ){
        for( int module = svtIdealVolumeFactory.getModuleMin()-1;
                 module < svtIdealVolumeFactory.getModuleMax(); module++ ){
            for( int strip = 0; strip < SVTConstants.NSTRIPS; strip+=16 ){
                Line3d stripLine = svtIdealStripFactory.getStrip(
                    region, sector, module, strip );
                Geant4Basic stripVol = Util.createArrow(
                     "strip"+strip+"_m"+module+"_s"+sector+"_r"+region, stripLine,
                     stripArrowCapRadius, stripArrowPointerRadius,
                     false, true, false ); // mm
                stripVol.setMother( svtIdealVolumeFactory.getMotherVolume() ); }
            Vector3d[] layerCorners = svtIdealStripFactory.getLayerCorners(
                region, sector, module );
            for( int i = 0; i < layerCorners.length; i++ ){
                Geant4Basic cornerDisc = new G4Tubs(
                    "cornerDisc"+i+"_m"+module+"_s"+sector+"_r"+region,
                    0, cornerDiscRadius, cornerDiscRadius, 0, 360 ); // cm
                cornerDisc.setPosition( layerCorners[i].times(0.1) ); // mm -> cm
                cornerDisc.setMother( svtIdealVolumeFactory.getMotherVolume() );
            }
        }
    }
}
```

Listing 8 shows how to calculate the endpoints of a strip using different methods and for different circumstances. As noted in the comments, the endpoints are calculated in the CLAS coordinate system and in the local coordinate system (associated with each module). There is also an example showing how to get the strip endpoints in the local coordinates and then transform to the CLAS coordinates for a particular layer and sector.

Listing 8 : Source code to calculate strip endpoints for different situations from SVTStripFactory.

```
import org.jlab.detector.calib.utils.DatabaseConstantProvider;
import org.jlab.detector.geant4.v2.SVT.*;
import org.jlab.geometry.prim.*;

// Connect to the database.
DatabaseConstantProvider cp = new DatabaseConstantProvider(11,"default");
SVTConstants.connect( cp );

// create the factory
SVTStripFactory svtIdealStripFactory = new SVTStripFactory( cp, false );

// set parameters.
int myStrip  = 3; // pick a strip, layer, sector
int myLayer  = 5;
int mySector = 6;
int myModule = 1; // corresponding module and region.
int myRegion = 2;

System.out.format("\nmyStrip = %1d, mySector = %1d, myModule = %1d, myRegion = %1d, myLayer =
    = %1d\n\n",myStrip,mySector,myModule,myRegion,myLayer);

// print the ideal endpoints in CLAS coordinates.
Line3d myCLASstripline = svtIdealStripFactory.getIdealStrip(myLayer, mySector, myStrip);
System.out.println("myLab    = "+myCLASstripline.toString());

// local coordinates for both layers in a layer+sector combination
Line3d myLocalstripline = svtIdealStripFactory.createIdealStrip(myStrip, 0);
System.out.println("myLocal0= "+myLocalstripline.toString());

// start from a strip in local coordinates and transform to CLAS coords.
Line3d myLocalstripline2 = svtIdealStripFactory.createIdealStrip(myStrip, myModule);
double myradius = SVTConstants.LAYERRADIUS[myRegion][myModule];
double myz0     = SVTConstants.Z0ACTIVE[myRegion];
```

```
Line3d myLocalstriplineTransformed = myLocalstripline2.transformed(SVTConstants.getLabFrame(
     myRegion, mySector, myradius, myz0 ));
System.out.println("myTrans␣=␣"+myLocalstriplineTransformed.toString());
System.out.println("myLab␣␣␣=␣"+myCLASstripline.toString());

// get the endpoints using the endpoints4CLAS-NOTE.groovy methods.
Line3d mystripline2 =  svtIdealStripFactory.getStrip(myRegion, mySector, myModule, myStrip);
System.out.println("myLab2␣␣=␣"+mystripline2.toString());
```

Listing 9 shows code analogous to Listing 8 except now the alignment shifts are added to the positions of the endpoints.

Listing 9 : Source code to calculate strip endpoints for different situations from SVTStripFactory with the alignment shifts included.

```
import org.jlab.detector.calib.utils.DatabaseConstantProvider;
import org.jlab.detector.geant4.v2.SVT.*;
import org.jlab.geometry.prim.*;

// connect to the CCDB
  SVTConstants.VERBOSE = true;
  DatabaseConstantProvider cp = new DatabaseConstantProvider();
  cp = SVTConstants.connect( cp );

// load alignment shifts;
  SVTConstants.loadAlignmentShifts( cp );

// create the factory
SVTStripFactory svtShiftedStripFactory = new SVTStripFactory( cp, false );

// set parameters.
int myStrip  = 3; // pick a strip, layer, sector
int myLayer  = 5;
int mySector = 6;
int myModule = 1; // corresponding module and region.
int myRegion = 2;

// shifted strip in lab coordinates
Line3d myCLASstriplineShifted  = svtShiftedStripFactory.getShiftedStrip(myLayer, mySector,
    myStrip);
System.out.println( "myLab␣␣␣=␣"+myCLASstriplineShifted.toString());
```

2.3.4 SVTAlignmentFactory :: Analyzing Fiducial Data

`SVTAlignmentFactory` is a static class that processes data of the fiducial points into alignment shifts, and applies alignment shifts to the ideal geometry. The data files of the fiducial survey points use the format shown in Listing 10. The first column is a tag to uniquely identify each of the 198 the fiducial points in the SVT. The next three columns are the Cartesian coordinates (x,y,z) of that point in the lab frame in millimetres.

Listing 10 : Format of fiducial data in txt file.

```
R1S01F1   -17.350   -68.283 -286.541
R1S01F2    17.350   -68.283 -286.541
R1S01F3     3.500   -68.283  122.333
```

2.3.5 AlignmentFactory :: Generating Shift Data

`AlignmentFactory` is a general static class that uses matrix algebra to compute the translation and rotation shifts between two given sets of normal vectors, and apply a given shift to a set of points or volumes.

An alignment shift consists of a translation and a rotation about the geometric mean point of the three ideal fiducials. Applying these to a point is trivial. For a volume however, the rotation requires a sequence of transformations and conversions between various representations, implemented in the `Util` static class.

Standard vector notation uses XYZ Extrinsic Tait-Bryan Euler angles (equivalent to $ZY'X''$ Intrinsic) [9, 10]. For a given pair elevation and azimuth angles, θ and ϕ respectively, first rotate about the X-axis by zero angle (for all vectors), then rotate about the Y-axis by θ, and finally rotate about the Z-axis by ϕ.

A `Geant4Basic` volume object stores a position in space and a rotation about its center of geometry. GEANT4 uses $XY'Z''$ Intrinsic (equivalent to ZYX Extrinsic). First rotate about the X axis, then the new Y axis, then the new Z axis. Euler angles do not lend themselves to manipulation, so they are converted to rotation matrices using the `Matrix` object class when applying a shift.

A rotation shift is defined by an axis and an angle centred on a point, and can be generated by comparing two sets of fiducial points using normal vectors. The axis/angle component is converted to a matrix, and applied to the initial rotation of a volume with a simple matrix multiplication.

2.3.6 Visualization Tool

A visualization tool was developed to check the geometry code using GDML and ROOT. The Geometry Description Markup Language (GDML) is a portable way to describe a volume structure for visualization and debugging purposes. It is an XML schema that defines a set of legal elements that are compatible with

GEANT4. A Java package was written to produce a tree of `Geant4Basic` objects to GDML file. Only a few volume types were included, such as boxes and cylinders. The source code for the package is available at `https://github.com/psq95/VolumeExporter`. ROOT is a modular scientific software framework based primarily on C++. It is used for big data processing, statistical analysis, visualization and storage. Part of its visualization capabilities includes a GDML interpreter. See Appendices B and C for examples of source code for using the GDML exporter class and ROOT, respectively.

2.3.7 Generating *gemc* Geometry Text Files

The flowchart in Figure 14 shows a leg that connects to the CLAS12 simulation package *gemc*. We have used the java code here to produce text files that contain the necessary parameters to describe the SVT geometry and are readily ported to other sites. Here we describe the procedure for generating those text files.

1. Download the *gemc* gitHub repositories `api` and `detectors` from `https://github.com/gemc`.

2. Define an environment variable named `myCOATJAVALIBS` that points to the location of the COATJAVA directory containing necessary libraries (*e.g.*, `~/coatjava/lib/clas`). At this writing that directory contains the files listed in Table 5.

Table 5: COATJAVA libraries

coat-libs-3.0-SNAPSHOT.jar	jcsg-0.3.2.jar
JEventViewer-1.1.jar	vecmath-1.3.1-2.jar

3. Define a second environment variable `gemcPERLMODS` that points to the location of the perl modules (*e.g.* `$GEMC/api-master/perl`) in the *gemc* distribution. This environment variable is used to actually tell perl where to find the appropriate libraries at runtime.

4. The main scripts for generating the SVT/BST geometry are in a directory such as `$GEMC/detectors-master/clas12/bst/`. In the file `config.dat` within this directory set the variation to `java` to obtain the version of the SVT geometry described here.

5. In the same directory (`$GEMC/detectors-master/clas12/bst/`), the main Perl script for generating the SVT/BST geometry is named `bst.pl`. It uses two other scripts internally `factory.groovy` and `geometry_java.pl`. The first one `factory.groovy` reads the SVT constants from the database, generates the volumes needed for Geant4, and writes them out in the file `bst_volumes.txt`. The second one, `geometry_java.pl`, is used in `bst.pl` to create the final, formatted output. Depending on how your directory structure is set up you may have to change one line (line 77 as of this writing)

from "`system('groovy -cp "../*" factory.groovy');`"

to the following.

"`system('<your path here>coatjava/bin/run-groovy factory.groovy'); `"

To run the main script `bst.pl` which generates the geometry text files use the command

```
perl -I $gemcPERLMODS bst.pl config.dat
```

where the files `bst.pl` and `config.dat` are in the local `bst` directory. A listing of the files produced by a successful run of the `bst.pl` script is shown in Table 6.

Table 6: SVT geometry output files.

bst__geometry_java.txt	bst__parameters_java.txt	bst__volumes_java.txt
bst__bank.txt	bst__hit_java.txt	bst__materials_java.txt

6. The scripts can be found at `https://github.com/gemc/detectors/tree/master/clas12/bst` Sample output files can be found at `https://userweb.jlab.org/~gilfoyle/svtGeometry/`. Additional documentation can be found at `https://userweb.jlab.org/~pdavies/dev/java/JCSG/`.

3 Validation

The previous SVT Geometry Document [7] was used as a preliminary basis for validating the parameters and conventions used by the GEMC and the Reconstruction for the SVT. However, it was discovered that many of the parameters in the document had become outdated since its publication as the SVT was built, so the finalized technical drawings from the Mechanical Engineering Department were used instead. [8]

Initially, the locations of the sensor layers in the SVT were not quite the same in both GEMC and the Reconstruction, and this resulted in poor resolution of the residuals. In addition, the residuals were also too large when reconstructing real cosmic data because there were differences between the simulated model and the physical detector.

3.1 Original Variation

Figures 15 and 16 show an early version of the geometry, called the `original` variation in GEMC. It was exported to a GDML file and visualized using ROOT to use custom transparencies to highlight volumes of interest. Figure 17 shows a comparison between the early version and the technical design drawings. It can be seen that the copper volume was misplaced, and the other passive volumes around the heatsink were not actually touching.

The `original` variation was built entirely in a Perl script called `geometry.pl`. It contained hard coded parameters that were discovered to be different to those used by the Reconstruction.

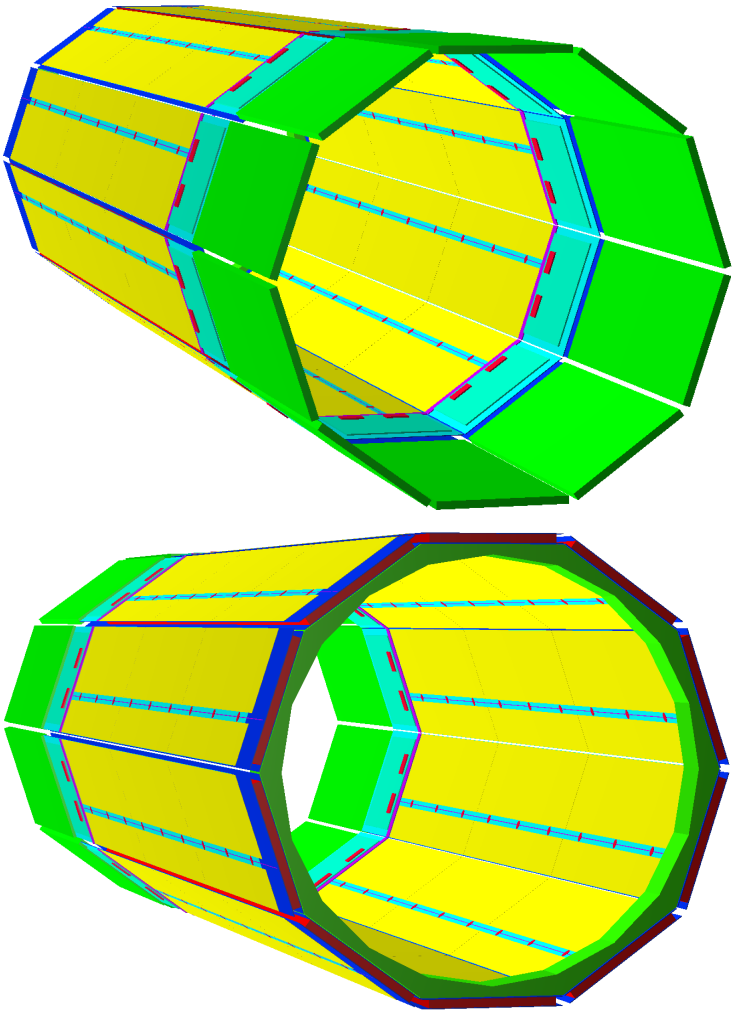

Figure 15: Early version of the SVT in GEMC with sensor modules removed.
Top: End view of the copper end, looking downstream.
Bottom: End view of the peek end, looking upstream.

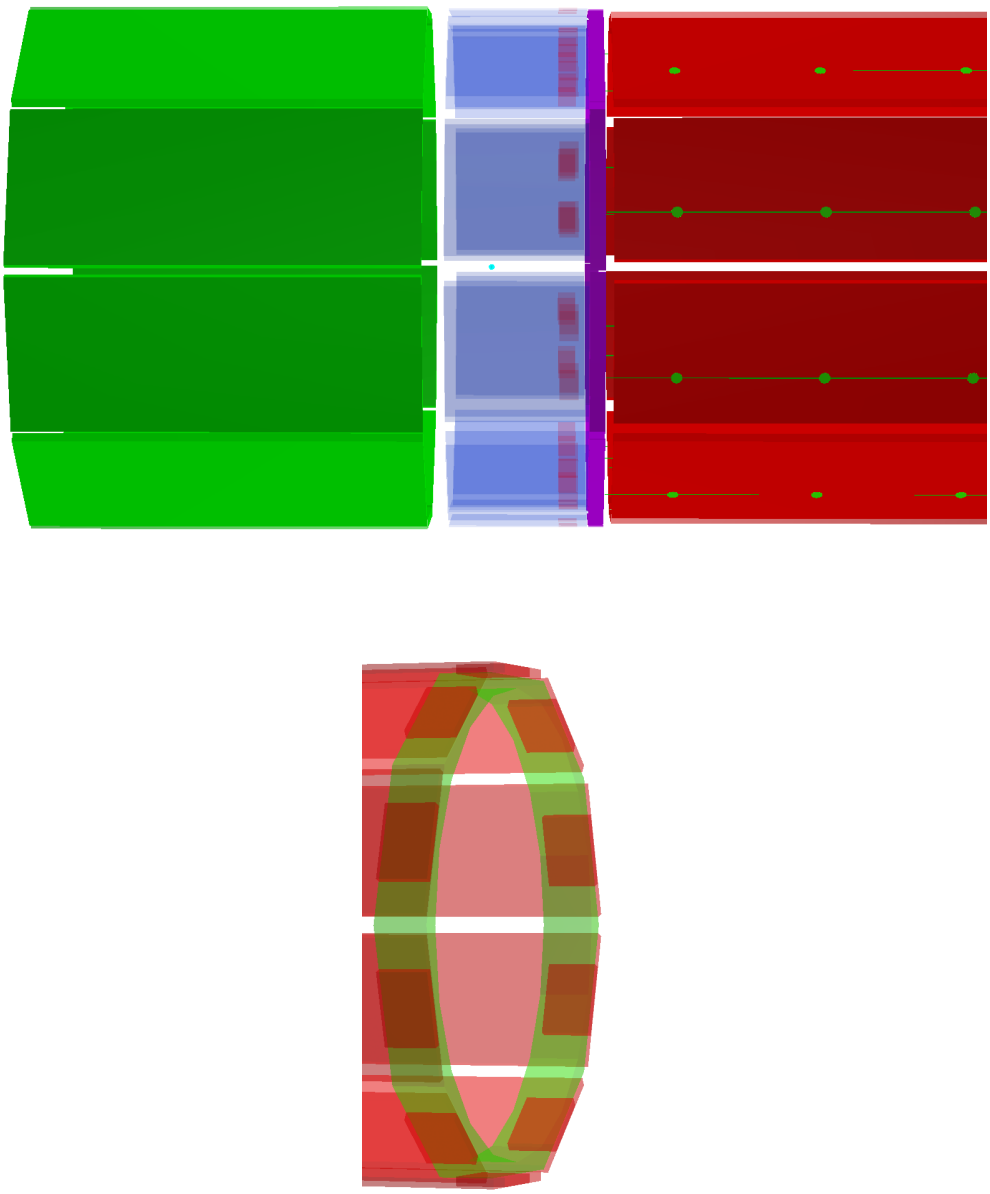

Figure 16: Early version of the SVT in GEMC with sensor modules removed.
Top: Side view of the copper end.
Bottom: Side view of the peek end.

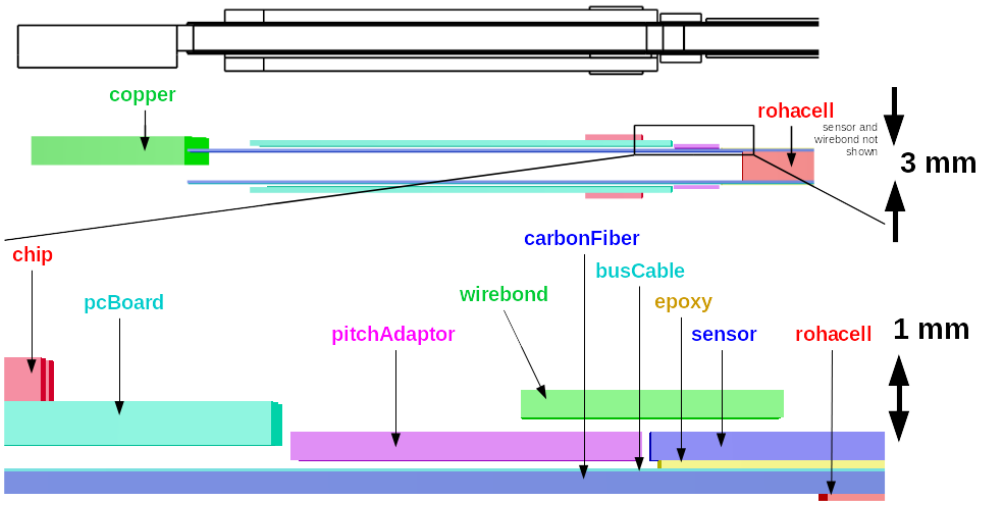

Figure 17: Comparison of early version of simulation to design drawings.
Top: Technical drawing showing side view of the copper support and HFCB (pcBoard) components of a sector module.
Bottom: Early version of the simulated geometry showing inconsistencies.

3.2 Java Variation

In order for both GEMC and the Reconstruction to use the same parameters from CCDB, a common geometry package was required. A Groovy script called `factorygroovy` first runs the Java code to output the nested volume structure and a few parameters to text files. A new Perl script called `geometry_javapl` then reads these files when building the detector volumes, and adds additional properties such as sensitivity and colour. Source code is available at `https://github.com/gemc/detectors/tree/master/clas12/bst`.

3.3 Fiducial Alignment

The Survey Group at Jefferson Lab recorded the actual positions of the fiducial points of each sector module as the SVT was being assembled. Two sets of data were received from the Group, called `Survey Ideal` and `Survey Measured`. The first was generated from an official Computer Aided Design (CAD) model of the SVT. Figure 18 shows how the second was derived from a fit of the experimental data to three circles for each region to minimize the overall shift of each module.

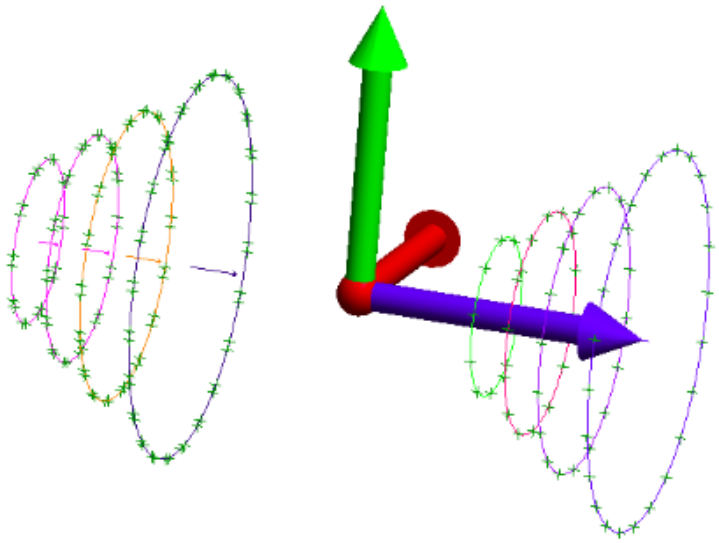

Figure 18: CAD preview of the fit used on the experimental fiducial survey data.

Another set of ideal fiducial points based on the core parameters, called `Factory Ideals`, was computed to determine more accurate values through consultation with the mechanical engineering team. Figure 19 shows the original state of the computed fiducial data, and figure 20 shows how the ideal geometry is now well defined within 2 μm resolution of the design specification. Notice the change in the vertical scale between Figures 19 and 20. Before the corrections deviations from the ideal values exceeded 300 μm in some places and were routinely off by 150 μm. After the corrections, the maximum deviation is about $2\mu m$ and is essentially zero is most places. We do not expect to be sensitive to variations below about 20 μm. Table 7 contains a translation of the Module# axis.

Table 7: The index used by the fiducial survey data is effectively the same as the `(Region, Sector)` index.

Module#	Region	Sector
1-10	1	1-10
11-24	2	1-14
25-42	3	1-18
43-66	4	1-24

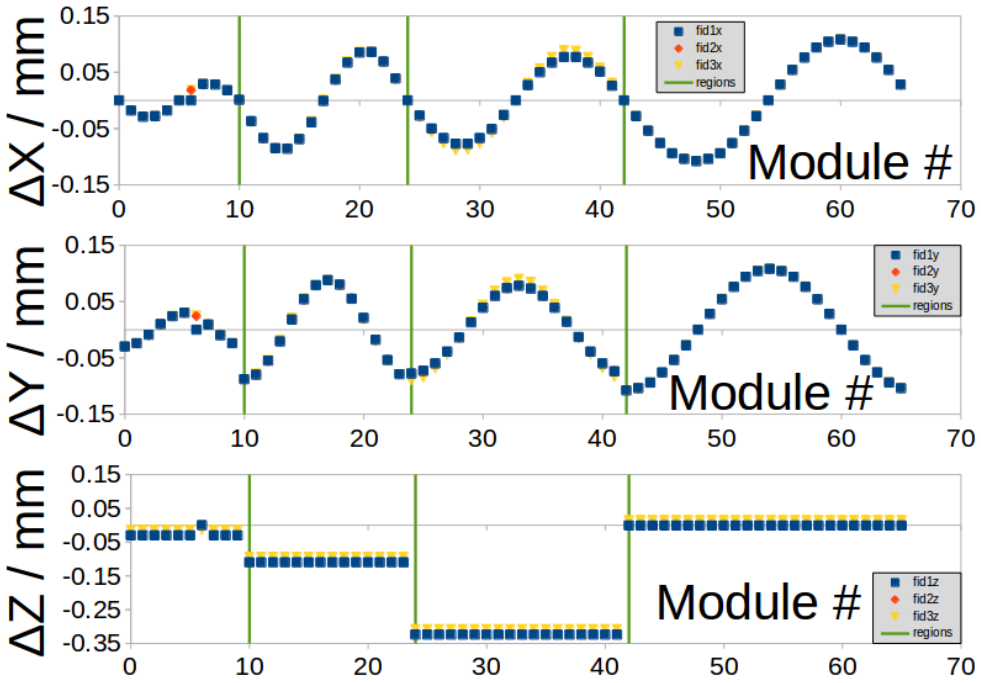

Figure 19: Comparison of fiducial data. `Factory Ideal` from `Survey Ideal` before corrections.

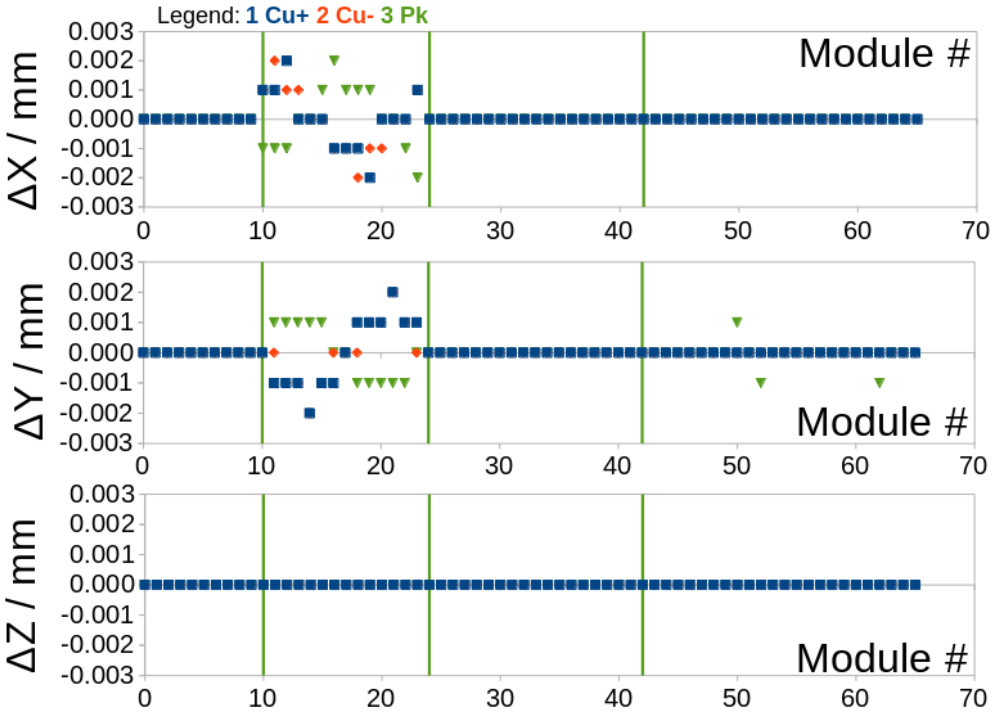

Figure 20: Comparison of fiducial data. `Factory Ideal` from `Survey Ideal` after corrections.

3.4 Fiducial Shifts

Figure 21 shows that, when compared to the ideal geometry, individual differences up to several hundred microns were observed between the measured data (`Survey Measured` and `Factory Ideal`). These differences were not due to flaws in the manufacturing process, but simply the difficulty of aligning macroscopic objects with great enough precision. Figure 22 shows the notation used for the distances between the fiducial points in one sector. Figure 23 shows the difference in those distances between the survey points (D1, D2, and D3) of the two data sets. The size of the disagreement between the two sets is smaller here (up to a hundred microns), but still significant. A set of alignment shifts was generated from these two fiducial data sets, and stored in CCDB in the `alignment` table. When these shifts are applied, the fiducial points do not line up exactly with the `Survey Measured` data resulting in the data seen in Figure 24. There is little improvement in the agreement when the shifts are applied. Compare Figures 21 and 24. In fact, the biggest difference between the two plots is a sign change in the difference since the difference is effectively reversed in the two plots. An explanation for the lack of improvement may lie with the fact that the `Survey Measured` data rely on a fit to the measured positions of the fiducial points (see previous section) so the distances D1, D2, and D3 in Figure 22 are no longer at their ideal values. The current algorithm to shift the `Factory Ideal` data maintains the original, ideal values for D1, D2, and D3 and calculates the the shifts with the center of rotation of the `Factory Ideal` triangle at the geometric mean of the three points defining that triangle (Cu+, Cu-, and Pk in Figure 22). It would be worthwhile to study the effect of applying the shifts to the geometric center of the `Survey Measured` data instead.

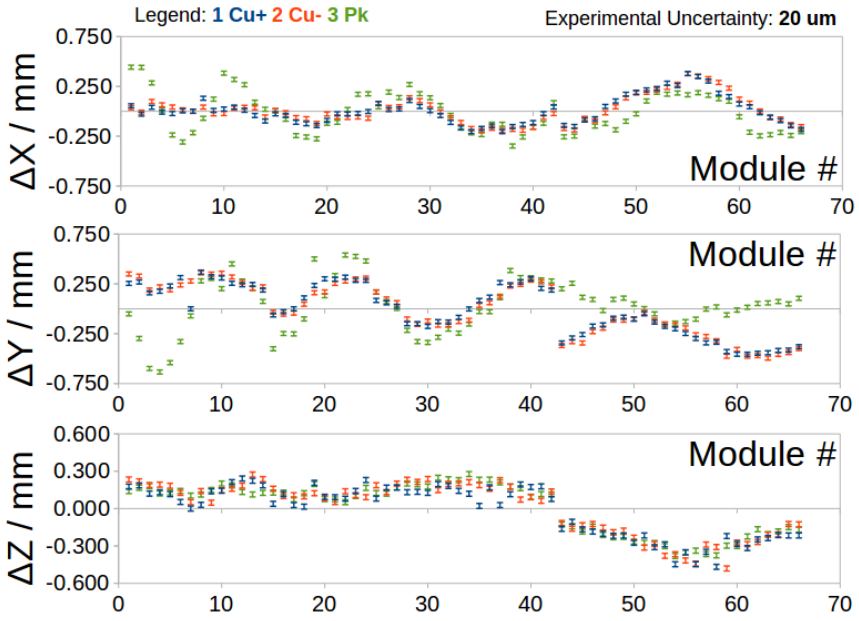

Figure 21: Comparison of fiducial data. `Survey Measured` from `Factory Ideal`.

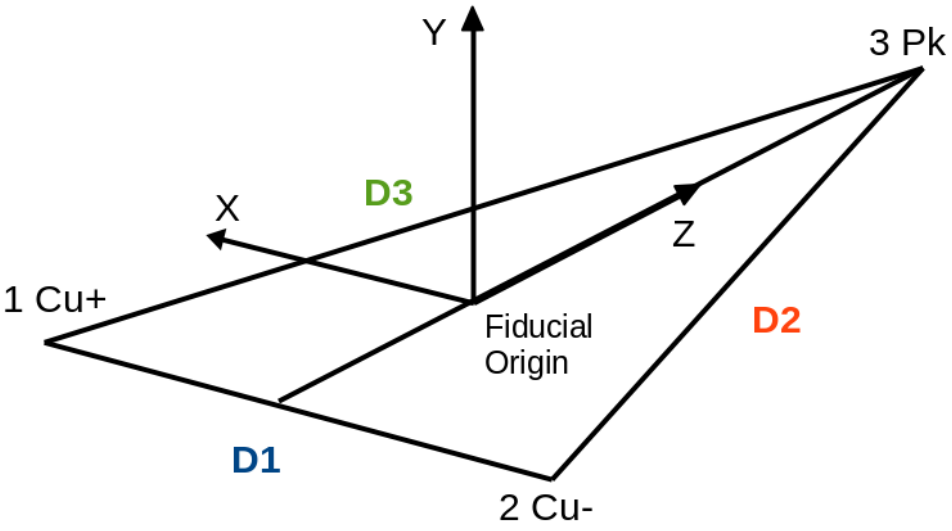

Figure 22: Diagram showing labeling of distances between fiducial points.

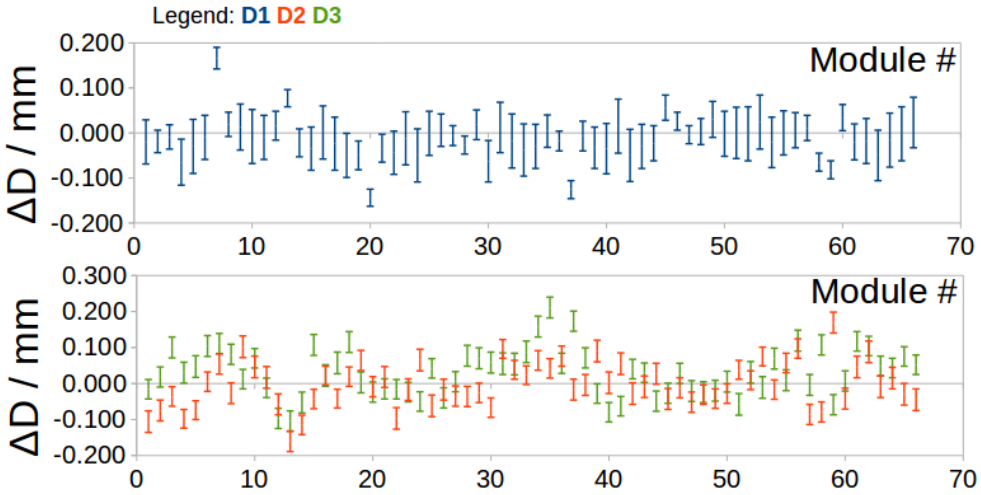

Figure 23: Comparison of distances between fiducial data. `Survey Measured` from `Factory Ideal`.

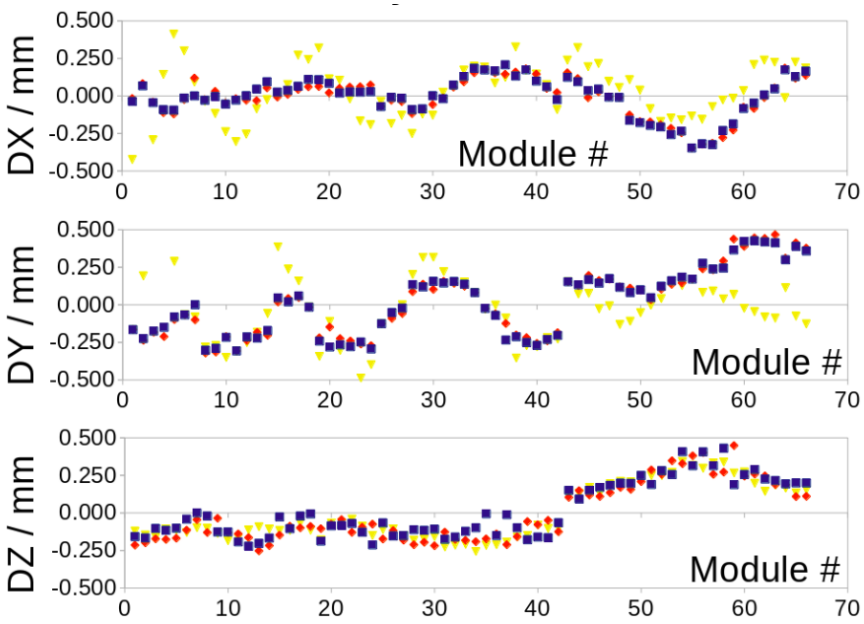

Figure 24: Comparison of fiducial data. `Factory Shifted` from `Survey Measured`.

4 Alignment Test Results

In addition to the validation procedures described above, the geometry package can also be tested using the CLAS12 alignment codes. Reaching the design goals for the hit position resolution σ_r of the SVT requires precise knowledge of the location and orientation of each sensor. That hit resolution is expected to be $\sigma_r \approx 55\ \mu m$ at the upstream end of each sensor and it reaches a maximum of $\sigma_r \approx 63\ \mu m$ at the downstream end. The upstream ends of the strips are all 156 μm apart while the downstream strip separation and resolution depends on the stereo angle of adjacent strips. Larger stereo angles means greater separation at the downstream end. The placement of the strips is expected to be within 20 μm of the design specifications. In this section we summarize the procedures to align the SVT and present our results to validate those procedures using the improved geometry packages described here. We test the alignment code first with simulated events from *gemc* and then use it to align the SVT using measured cosmic ray data. More detail on this topic will be in a future CLAS-NOTE.

To align the SVT we have used both simulated and measured cosmic ray tracks with no magnetic field (straight tracks) that are fitted with a linear, least-squares minimization procedure in a program called `millepede` [11]. The program has been applied to simulated and real cosmic ray tracks with a comparatively simple structure. Such a cosmic ray event is shown in Fig. 25 that we classify as a Type-1 track. Cosmic-ray events with isolated clusters of strips from all sixteen horizontal sensors are selected to simplify the interpretation of the results. Only the x position of an sensor was allowed to vary in the fit (see coordinates

in Fig. 25). Additional constraints on the angle of track to select nearly vertical ones were also used.

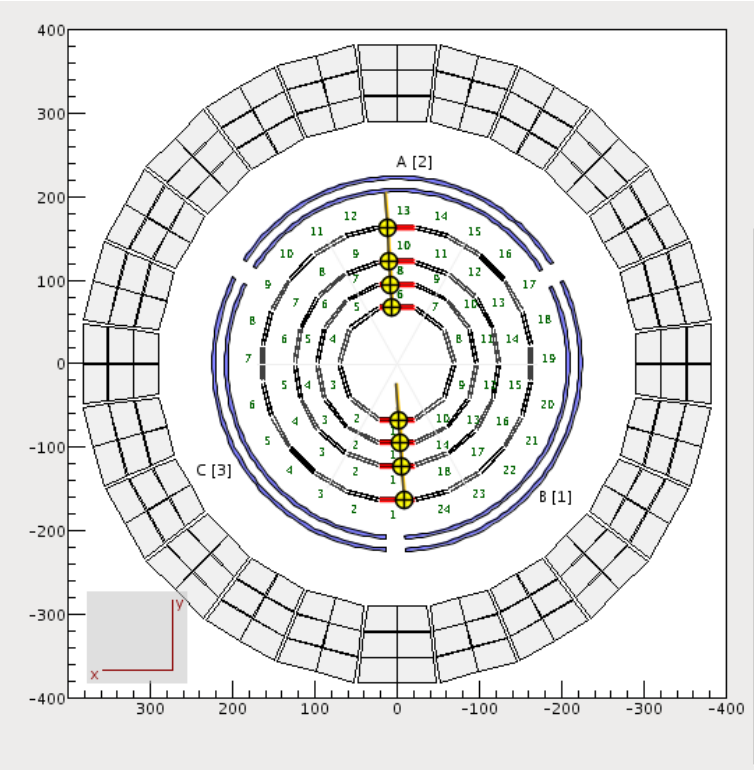

Figure 25: A Type 1, cosmic ray track in the SVT passing though all eight horizontal SVT layers.

The SVT `millepede` alignment was studied first with simulated events from *gemc*. The first case studied was cosmic ray events in the SVT using the ideal geometry - the SVT sensor positions calculated from the set of core parameters. These simulated, ideal-geometry events were reconstructed with the standard CLAS12 reconstruction package. The results are shown in Fig 26. The left-hand panel shows the residuals for Type-1 tracks from the reconstruction. The blue, open circles represent the centroid of the residual distribution for a single sensor (x position) plotted at the vertical position of the sensor (y value). Compare the vertical positions with Fig. 25. The uncertainties on each point represent the width of the residual distribution (not the uncertainty in the centroid) which is more closely related to the resolution. The residuals are close to, but not quite, aligned at zero. The observed shifts are all less than 0.5 μm (recall the expected position resolution resolution is 20 μm), but there is a systematic bias in the reconstruction which 'clocks' the residuals. The right-hand panel shows the misalignments extracted with `millepede`. To obtain reasonable results from `millepede` one must fix two of the misalignments otherwise there is an infinite number of possible solutions - the geometry parameters could all be shifted together by an arbitrary amount in x without changing the quality of the fit. In the results shown here, the alignments in the fourth layers from the center are set to zero. We expect the misalignments here to be similar in size and distribution to the residuals. In both cases,

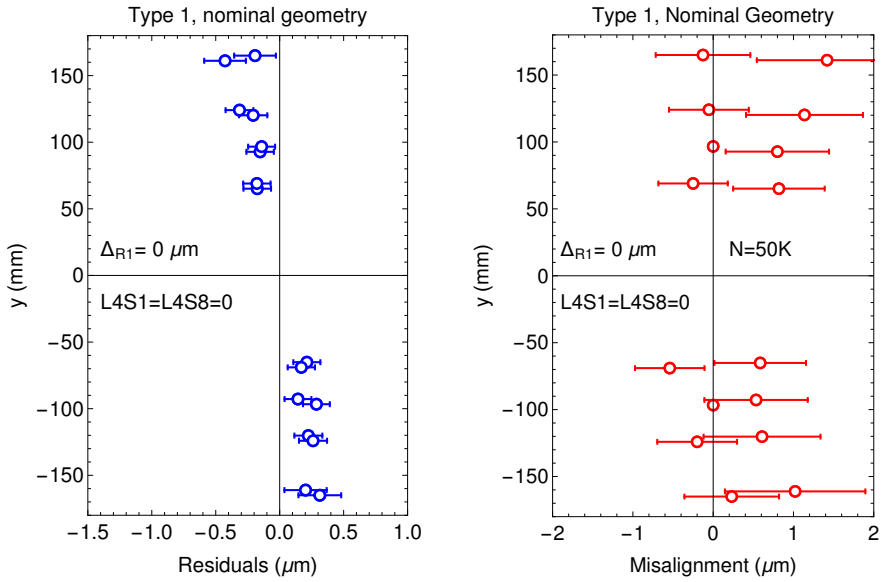

Figure 26: Simulated cosmic ray results using the ideal perfect geometry for the track fit residuals (left-hand panel) and `millepede` misalignments (right-hand panel).

the size of residuals/misalignments is similar ($< 2\ \mu m$), but the 'clocked' pattern in the residuals is not seen in the misalignments.

Known shifts in the x positions of the innermost region of sensors (layers 1-2) were inserted in the *gemc* simulation and then the misalignments extracted to see if the alignment code would reproduce the shifts. The results for a shift $\Delta x = 500\ \mu m$ are shown in Fig. 27 for 500,000 simulated events. The layout of the figure is the same as Fig. 26. The y axis is the vertical position of the sensors for Type-1 events and the horizontal position is the centroid of the residual/misalignment. The uncertainty is the width of the distribution for that sensor. The left-hand panel clearly shows the inserted shift in x for the innermost sensors. Note the fitted track averages the positive and negative residuals so the unshifted regions now have a large positive residual. The separation between the shifted and unshifted sensors is about $500 \mu m$ as expected. The misalignment from `millepede` shows the unshifted sensors at zero and the shifted sensors misaligned by 500 μm as expected.

The alignment code and the underlying geometry package have also been tested on real, measured cosmic rays collected with the SVT. The alignment code was used to extract the misalignments from the reconstructed cosmic rays. Rather than just comparing the residuals and misalignments from `millepede`, the extracted misalignments were used to correct the geometry and the reconstruction performed again. The results are shown in Fig. 28. It shows the residuals before (blue) and after (red) the correction. Before correction there are large residuals (in blue) which represent misalignmnents of order several hundred microns - much larger than the SVT specifications for position resolution. After correction, the residuals

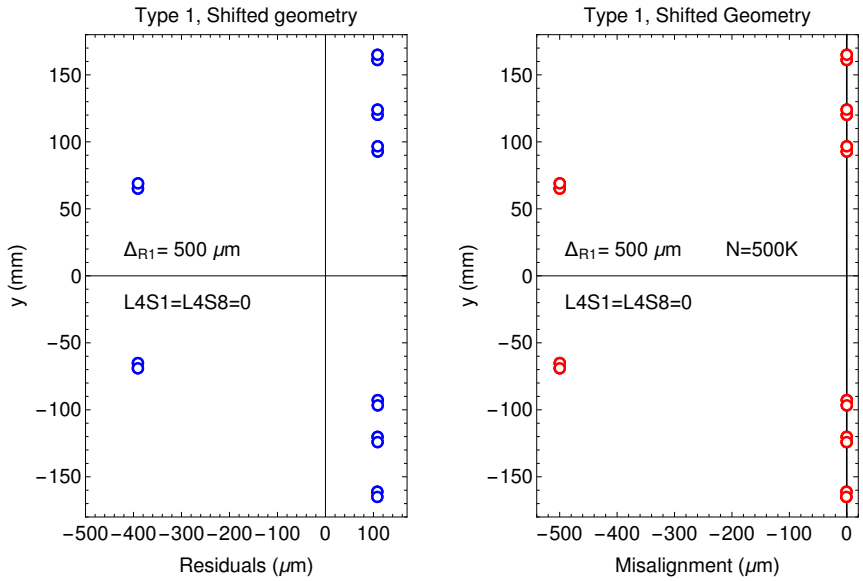

Figure 27: Simulated cosmic ray results using the ideal perfect geometry with a region 1 shift of 2 μm of all region 1 to positive x. The track fit residuals are shown in the left-hand panel and the `millepede` misalignments (are shown in the right-hand panel.

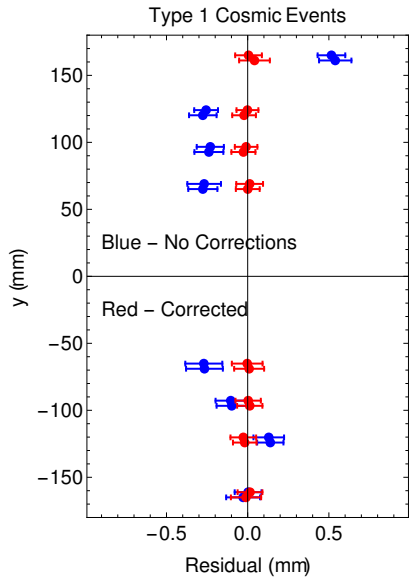

Figure 28: Residuals and misalignment correction results for measured SVT cosmic rays. Blue points show the initial residuals. Red points show the same residuals after correcting the reconstruction with the `millepede` results.

(in red) are now all close to zero. The standard deviation for all the residuals is about 25 μm.

The geometry package has been tested with the SVT alignment procedures using simulated, cosmic-ray events and real, measured cosmic rays in the SVT. The geometry package ensures that simulation and reconstruction obtain their geometry from the same source. The geometry and alignment packages show the expected improvements on both simulated and measured cosmic ray events.

5 Conclusion

The geometry of the SVT has been well defined according to the design specification, aligned using real cosmic data, and the simulation and reconstruction software now receive the same geometry from one source. Future work includes adding more volumes to the backing structure, such as the downstream support, the wirebond between sensor cards, and the kapton tape wrapped around the side of the sector modules, further alignment studies using non-Type-1 tracks, and testing of the common geometry using geantinos.

References

[1] Ziegler V. Status of Event Reconstruction in CLAS12. Presented at the CLAS Collaboration Meeting, Nov, 2016.

[2] CLAS Collaboration. EVIO file format. http://clasweb.jlab.org/clas12offline/docs/software/html/io/readingRawEvioFiles.html.

[3] CLAS Collaboration. HIPO file format. http://clasweb.jlab.org/clas12offline/docs/software/3.0/html/rec/inputfiles.html.

[4] Ungaro M. GEMC. https://gemc.jlab.org.

[5] Ungaro M. and Davies P. Source code for the SVT/BST in GEMC. https://github.com/psq95/detectors/tree/master/clas12/bst.

[6] Antonioli M.A. et al. Performance of the CLAS12 Silicon Vertex Tracker modules. *Nucl. Instrum. Meth. A*, 732:99–102, 2013.

[7] CLAS Collaboration. SVT Geometry Document. https://clasweb.jlab.org/wiki/index.php/CLAS12_Geometry,_Calibration,_Reconstruction_and_Monitoring_Documents.

[8] Mandal S. Technical Drawings of the SVT. https://userweb.jlab.org/~mandal/SVT/Drawings/Module_Drawings_July13/.

[9] Eric W. Weisstein. Euler Angles. http://mathworld.wolfram.com/EulerAngles.html.

[10] Euler Angles. https://en.wikipedia.org/wiki/Euler_angles.

[11] Volker Blobel, Claus Kleinwort, and Frank Meier. Fast alignment of a complex tracking detector using advanced track models. *Comput.Phys.Commun.*, 182:1760–1763, 2011.

A CCDB

Listing 11 : Example of CCDB interactive interface.

```
/> cd geometry/cvt/svt
/geometry/cvt/svt> ls
material
region
alignment
fiducial
svt
/geometry/cvt/svt> cat svt
+---------------------------------+
| nRegions       (int)| 4          |
| nModules       (int)| 2          |
| nSensors       (int)| 3          |
| nStrips        (int)| 256        |
| nFiducials     (int)| 3          |
| nPads          (int)| 8          |
| layerGapThk    (double)| 3.166   |
| readoutPitch   (double)| 0.156   |
| stereoAngle    (double)| 3.0     |
| phiStart       (double)| 270.0   |
| zRotationStart (double)| 90.0    |
| microGapLen    (double)| 0.112   |
| physSenWid     (double)| 42.000  |
| siliconThk     (double)| 0.320   |
| physSenLen     (double)| 111.625 |
| activeSenWid   (double)| 40.032  |
| activeSenLen   (double)| 109.955 |
| deadZnWid      (double)| 0.984   |
| deadZnLen      (double)| 0.835   |
| modulePosFac   (double)| 0.5     |
| stripStart     (double)| 0.048   |
+---------------------------------+
/geometry/cvt/svt> cat region
+-------------------------------------------------------------------------------+
| region | status | nSectors | zStart   | UlayerOuterRadius | CuSupportInnerRadius |
| int    | int    | int      | double   | double            | double               |
+-------------------------------------------------------------------------------+
```

```
| 1       | 1       | 10        | -219.826 | 65.447            | 65.403               |
| 2       | 1       | 14        | -180.380 | 93.047            | 93.005               |
| 3       | 1       | 18        | -141.206 | 120.482           | 120.435              |
| 4       | 0       | 24        | -83.405  | 161.362           | 161.315              |
+-------------------------------------------------------------------------------------+
/geometry/cvt/svt> cat fiducial
+---------------------------+
| CuX     (double)| 17.35   |
| PkX     (double)| 3.50    |
| OriginZ (double)| 62.13   |
| CuZ     (double)| 3.75    |
| PkZ0    (double)| 402.624 |
| PkZ1    (double)| 2.50    |
+---------------------------+
/geometry/cvt/svt> cat alignment
+-------------------------------------------------------------------+
| tag     | tx      | ty      | tz      | rx      | ry      | rz      | ra      |
| string | double | double | double | double | double | double | double |
+-------------------------------------------------------------------+
| R1S01   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   |
| R1S02   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   |
| R1S03   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   |
| R1S04   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   |
| R1S05   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   |
| R1S06   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   |
| R1S07   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   |
| R1S08   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   |
| R1S09   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   |
| R1S10   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   | 0.000   |
. .....  . ..... . ..... . ..... . .....  . .....  . .....  . ..... .

/geometry/cvt/svt> cd material
/geometry/cvt/svt/material> cat box
+-----------------------------------------------------------+
| name                | wid         | thk         | len         |
| string              | double      | double      | double      |
+-----------------------------------------------------------+
| heatSink            | 40.700      | 2.880       | 65.330      |
| heatSinkCu          | 40.700      | 2.500       | 65.330      |
| heatSinkRidge       | 40.700      | 0.380       | 15.600      |
| rohacell            | 41.000      | 2.500       | 352.925     |
```

```
| rohacellCu           | 41.000     | 2.500      | 338.425    |
| plastic              | 20.000     | 2.500      | 14.500     |
| plasticPk            | 12.000     | 0.380      | 12.500     |
| carbonFiber          | 42.000     | 0.190      | 401.640    |
| carbonFiberCu        | 36.750     | 0.190      | 46.580     |
| carbonFiberPk        | 41.000     | 0.190      | 355.060    |
| busCable             | 42.000     | 0.078      | 401.640    |
| busCableCu           | 36.750     | 0.078      | 46.580     |
| busCablePk           | 41.000     | 0.078      | 355.060    |
| pitchAdaptor         | 41.500     | 0.320      | 4.000      |
| pcBoardAndChips      | 36.730     | 1.000      | 41.440     |
| pcBoard              | 36.730     | 0.500      | 41.440     |
| chip                 | 7.500      | 0.500      | 5.000      |
| epoxyAndRailAndPads  | 41.000     | 0.063      | 401.640    |
| epoxyMajorCu         | 22.875     | 0.063      | 46.580     |
| epoxyMinorCu         | 10.875     | 0.063      | 46.580     |
| epoxyMajorPk         | 25.000     | 0.063      | 355.060    |
| epoxyMinorPk         | 13.000     | 0.063      | 355.060    |
| rail                 | 0.127      | 0.003      | 370.985    |
| wirebond             | 0.0        | 0.0        | 0.0        |
| kaptonWrapTapeSide   | 0.013      | 3.061      | 352.930    |
| kaptonWrapTapeCap    | 4.832      | 0.013      | 352.930    |
| kaptonWrapGlueSide   | 0.013      | 3.036      | 352.930    |
| kaptonWrapGlueCap    | 4.819      | 0.013      | 352.930    |
+-----------------------------------------------------------+

/geometry/cvt/svt/material> cat tube
+----------------------+
| name (string)| pad   |
| rmin (double)| 0.000 |
| rmax (double)| 1.500 |
| zlen (double)| 0.060 |
| phi0 (double)| 0.0   |
| dphi (double)| 360.0 |
+----------------------+
```

B Groovy Example

Listing 12 : Groovy script showing how to perform basic functions with each class.

```
import org.jlab.detector.calib.utils.DatabaseConstantProvider;
import org.jlab.detector.geant4.v2.SVT.*;
import org.jlab.geometry.exporter.GdmlExporter;
import org.jlab.geometry.exporter.VolumeExporterFactory;

// for ideal geometry
SVTConstants.VERBOSE = true; // optional for debugging
DatabaseConstantProvider cp = SVTConstants.connect();

SVTVolumeFactory svtIdealVolumeFactory = new SVTVolumeFactory( cp, false );
svtIdealVolumeFactory.makeVolumes();
//System.out.println( svtIdealVolumeFactory.toString() ); // optional for debugging

GdmlExporter gdmlFile = VolumeExporterFactory.createGdmlFactory();
gdmlFile.addTopVolume( svtIdealVolumeFactory.getMotherVolume() );

gdmlFile.addMaterialPreset("mat_hide", "mat_vacuum");
gdmlFile.addMaterialPreset("mat_half", "mat_vacuum");

gdmlFile.replaceVolumeMaterial( "vol_heatSink", "mat_hide");
//gdmlFile.replaceVolumeMaterial( "vol_heatSink", "mat_half");
gdmlFile.replaceVolumeMaterial( "vol_heatSinkCu", "mat_vacuum");
gdmlFile.replaceVolumeMaterial( "vol_heatSinkRidge", "mat_vacuum");

gdmlFile.replaceVolumeMaterial( "vol_carbonFiber", "mat_hide");
gdmlFile.replaceVolumeMaterial( "vol_carbonFiberCu", "mat_vacuum");
gdmlFile.replaceVolumeMaterial( "vol_carbonFiberPk", "mat_vacuum");

gdmlFile.replaceVolumeMaterial( "vol_busCable", "mat_hide");
gdmlFile.replaceVolumeMaterial( "vol_busCableCu", "mat_vacuum");
gdmlFile.replaceVolumeMaterial( "vol_busCablePk", "mat_vacuum");

gdmlFile.replaceVolumeMaterial( "vol_pcBoardAndChips", "mat_hide");

gdmlFile.replaceVolumeMaterial( "vol_epoxyAndRailAndPads", "mat_hide");

gdmlFile.replaceVolumeMaterial( "vol_module", "mat_hide");
```

```
gdmlFile.replaceVolumeMaterial( "vol_sensorPhysical", "mat_hide");

gdmlFile.replaceVolumeMaterial( "vol_sector", "mat_half");
//gdmlFile.replaceVolumeMaterial( "vol_region", "mat_half");
//gdmlFile.replaceVolumeMaterial( "vol_svt", "mat_half");
//gdmlFile.replaceVolumeMaterial( "vol_sector", "mat_hide");
gdmlFile.replaceVolumeMaterial( "vol_region", "mat_hide");
gdmlFile.replaceVolumeMaterial( "vol_svt", "mat_hide");

gdmlFile.replaceVolumeMaterial( "arrow0", "mat_hide");
//gdmlFile.replaceVolumeMaterial( "arrow0", "mat_half");

gdmlFile.writeFile("svt");
```

C ROOT

Listing 13 : ROOT script showing how to import a GDML file and display the volumes with custom transparency and colour.

```
#include <TColor.h>
void draw() {
    gSystem->Load("libGeom"); // library for geometry
    gSystem->Load("libGdml"); // library for GDML
    TGeoManager *geom = TGeoManager::Import("svt.gdml"); // filename here

    cout << "setting material transparencies\n";
    int transparencyHide = 100;
    int transparencyHalf = 50;
    int transparencyShow = 0;

    TList *matList = geom->GetListOfMaterials();
    TIter matNext( matList );
    while( mat = (TGeoMaterial*) matNext() )
    {
        TString *matName = new TString( mat->GetName() );
        if( matName->Contains("hide") )
        {
            cout << "hide " << transparencyHide;
            mat->SetTransparency( transparencyHide );
        }
        else if( matName->Contains("half") )
        {
            cout << "half " << transparencyHalf;
            mat->SetTransparency( transparencyHalf );
        }
        else
        {
            cout << "show " << transparencyShow;
            mat->SetTransparency( transparencyShow );
        }
        cout << " " << matName->Data() << "\n";
    }

    TGeoVolume *top = geom->GetTopVolume();
    //top->SetLineColor( kWhite );
    //geom->SetTopVisible();
```

```
cout << "setting␣volume␣colours\n";

TObjArray *volList = geom->GetListOfVolumes();
TIter volNext( volList );
while( vol = (TGeoVolume*) volNext() )
{
    vol->SetVisContainers(kTRUE); // for half-transparent volumes

    TString *volName = new TString( vol->GetName() );
    if( volName->Contains("module") )
    {
        vol->SetLineColor( kCyan-10 );
    }
    else if( volName->Contains("sensorActive") )
    {
        vol->SetLineColor( kBlue-1 );
    }
    else if( volName->Contains("sensorPhysical") )
    {
        vol->SetLineColor( kBlue );
    }
    else if( volName->Contains("fiducial") )
    {
        vol->SetLineColor( kCyan );
    }
    else if( volName->Contains("rohacell") )
    {
        vol->SetLineColor( kWhite );
    }
    else if( volName->Contains("heatSink") )
    {
        vol->SetLineColor( kOrange );
    }
    else if( volName->Contains("heatSinkCu") )
    {
        vol->SetLineColor( kOrange+1 );
    }
    else if( volName->Contains("carbonFiberCu") )
    {
        vol->SetLineColor( kGray+3 );
```

```
        }
        else if( volName->Contains("carbonFiberPk") )
        {
            vol->SetLineColor( kGray+3 );
        }
        else if( volName->Contains("busCable") )
        {
            vol->SetLineColor( kGray+2 );
        }
        else if( volName->Contains("pcBoard") )
        {
            vol->SetLineColor( kYellow-9 );
        }
        else if( volName->Contains("chip") )
        {
            vol->SetLineColor( kYellow+1 );
        }
        else if( volName->Contains("epoxy") )
        {
            vol->SetLineColor( kWhite );
        }
        else if( volName->Contains("arrow") )
        {
            vol->SetLineColor( kBlack );
        }
        else
        {
            vol->SetLineColor( kGray );
        }
        //cout << " " << volName->Data() << "\n";
    }

    cout << "VisLevel " << geom->GetVisLevel() << "\n";
    cout << "VisOption " << geom->GetVisOption() << "\n";
    //geom->SetVisLevel( 3 );
    //geom->SetVisOption( 1 );
    //cout << "new VisLevel " << geom->GetVisLevel() << "\n";
    //cout << "new VisOption " << geom->GetVisOption() << "\n";
    top->Draw("ogl");
}
```

D Strip Endpoints

The table below holds the three-dimensional, ideal endpoints of the first and last strips on the SVT modules in mm's constructed using the groovy script entitled `endpoints4CLAS-NOTE.groovy`.

Region	Sector	Module	Strip	x_u	y_u	z_u	x_d	y_d	z_d
1	1	1	1	-19.89000	-65.28700	-219.82600	-19.89000	-65.28700	113.60300
1	1	1	256	19.89000	-65.28700	-219.82600	20.01600	-65.28700	-217.42178
1	2	1	1	-54.46608	-41.12724	-219.82600	-54.46608	-41.12724	113.60300
1	2	1	256	-22.28339	-64.50934	-219.82600	-22.18145	-64.58340	-217.42178
1	3	1	1	-68.23797	-1.25828	-219.82600	-68.23797	-1.25828	113.60300
1	3	1	256	-55.94528	-39.09131	-219.82600	-55.90634	-39.21114	-217.42178
1	4	1	1	-55.94528	39.09131	-219.82600	-55.94528	39.09131	113.60300
1	4	1	256	-68.23797	1.25828	-219.82600	-68.27691	1.13845	-217.42178
1	5	1	1	-22.28339	64.50934	-219.82600	-22.28339	64.50934	113.60300
1	5	1	256	-54.46608	41.12724	-219.82600	-54.56802	41.05318	-217.42178
1	6	1	1	19.89000	65.28700	-219.82600	19.89000	65.28700	113.60300
1	6	1	256	-19.89000	65.28700	-219.82600	-20.01600	65.28700	-217.42178
1	7	1	1	54.46608	41.12724	-219.82600	54.46608	41.12724	113.60300
1	7	1	256	22.28339	64.50934	-219.82600	22.18145	64.58340	-217.42178
1	8	1	1	68.23797	1.25828	-219.82600	68.23797	1.25828	113.60300
1	8	1	256	55.94528	39.09131	-219.82600	55.90634	39.21114	-217.42178
1	9	1	1	55.94528	-39.09131	-219.82600	55.94528	-39.09131	113.60300
1	9	1	256	68.23797	-1.25828	-219.82600	68.27691	-1.13845	-217.42178
1	10	1	1	22.28339	-64.50934	-219.82600	22.28339	-64.50934	113.60300
1	10	1	256	54.46608	-41.12724	-219.82600	54.56802	-41.05318	-217.42178
1	1	2	1	19.89000	-68.76900	-219.82600	19.89000	-68.76900	113.60300
1	1	2	256	-19.89000	-68.76900	-219.82600	-20.01600	-68.76900	-217.42178
1	2	2	1	-24.33006	-67.32634	-219.82600	-24.33006	-67.32634	113.60300
1	2	2	256	-56.51275	-43.94424	-219.82600	-56.61469	-43.87018	-217.42178
1	3	2	1	-59.25686	-40.16730	-219.82600	-59.25686	-40.16730	113.60300
1	3	2	256	-71.54955	-2.33428	-219.82600	-71.58849	-2.21444	-217.42178
1	4	2	1	-71.54955	2.33428	-219.82600	-71.54955	2.33428	113.60300
1	4	2	256	-59.25686	40.16730	-219.82600	-59.21792	40.28714	-217.42178
1	5	2	1	-56.51275	43.94424	-219.82600	-56.51275	43.94424	113.60300
1	5	2	256	-24.33006	67.32634	-219.82600	-24.22812	67.40040	-217.42178
1	6	2	1	-19.89000	68.76900	-219.82600	-19.89000	68.76900	113.60300
1	6	2	256	19.89000	68.76900	-219.82600	20.01600	68.76900	-217.42178
1	7	2	1	24.33006	67.32634	-219.82600	24.33006	67.32634	113.60300
1	7	2	256	56.51275	43.94424	-219.82600	56.61469	43.87018	-217.42178
1	8	2	1	59.25686	40.16730	-219.82600	59.25686	40.16730	113.60300
1	8	2	256	71.54955	2.33428	-219.82600	71.58849	2.21444	-217.42178
1	9	2	1	71.54955	-2.33428	-219.82600	71.54955	-2.33428	113.60300
1	9	2	256	59.25686	-40.16730	-219.82600	59.21792	-40.28714	-217.42178
1	10	2	1	56.51275	-43.94424	-219.82600	56.51275	-43.94424	113.60300
1	10	2	256	24.33006	-67.32634	-219.82600	24.22812	-67.40040	-217.42178
2	1	1	1	-19.89000	-92.88700	-180.38000	-19.89000	-92.88700	153.04900
2	1	1	256	19.89000	-92.88700	-180.38000	20.01600	-92.88700	-177.97578
2	2	1	1	-58.22243	-75.05835	-180.38000	-58.22243	-75.05835	153.04900
2	2	1	256	-22.38189	-92.31824	-180.38000	-22.26837	-92.37291	-177.97578
2	3	1	1	-85.02319	-42.36347	-180.38000	-85.02319	-42.36347	153.04900
2	3	1	256	-60.22077	-73.46473	-180.38000	-60.14221	-73.56324	-177.97578

Region	Sector	Module	Strip	x_u	y_u	z_u	x_d	y_d	z_d
2	4	1	1	-94.98407	-1.27799	-180.38000	-94.98407	-1.27799	153.04900
2	4	1	256	-86.13219	-40.06062	-180.38000	-86.10415	-40.18346	-177.97578
2	5	1	1	-86.13219	40.06062	-180.38000	-86.13219	40.06062	153.04900
2	5	1	256	-94.98407	1.27799	-180.38000	-95.01211	1.15514	-177.97578
2	6	1	1	-60.22077	73.46473	-180.38000	-60.22077	73.46473	153.04900
2	6	1	256	-85.02319	42.36347	-180.38000	-85.10175	42.26496	-177.97578
2	7	1	1	-22.38189	92.31824	-180.38000	-22.38189	92.31824	153.04900
2	7	1	256	-58.22243	75.05835	-180.38000	-58.33595	75.00368	-177.97578
2	8	1	1	19.89000	92.88700	-180.38000	19.89000	92.88700	153.04900
2	8	1	256	-19.89000	92.88700	-180.38000	-20.01600	92.88700	-177.97578
2	9	1	1	58.22243	75.05835	-180.38000	58.22243	75.05835	153.04900
2	9	1	256	22.38189	92.31824	-180.38000	22.26837	92.37291	-177.97578
2	10	1	1	85.02319	42.36347	-180.38000	85.02319	42.36347	153.04900
2	10	1	256	60.22077	73.46473	-180.38000	60.14221	73.56324	-177.97578
2	11	1	1	94.98407	1.27799	-180.38000	94.98407	1.27799	153.04900
2	11	1	256	86.13219	40.06062	-180.38000	86.10415	40.18346	-177.97578
2	12	1	1	86.13219	-40.06062	-180.38000	86.13219	-40.06062	153.04900
2	12	1	256	94.98407	-1.27799	-180.38000	95.01211	-1.15514	-177.97578
2	13	1	1	60.22077	-73.46473	-180.38000	60.22077	-73.46473	153.04900
2	13	1	256	85.02319	-42.36347	-180.38000	85.10175	-42.26496	-177.97578
2	14	1	1	22.38189	-92.31824	-180.38000	22.38189	-92.31824	153.04900
2	14	1	256	58.22243	-75.05835	-180.38000	58.33595	-75.00368	-177.97578
2	1	2	1	19.89000	-96.36900	-180.38000	19.89000	-96.36900	153.04900
2	1	2	256	-19.89000	-96.36900	-180.38000	-20.01600	-96.36900	-177.97578
2	2	2	1	-23.89267	-95.45542	-180.38000	-23.89267	-95.45542	153.04900
2	2	2	256	-59.73321	-78.19552	-180.38000	-59.84673	-78.14085	-177.97578
2	3	2	1	-62.94311	-75.63572	-180.38000	-62.94311	-75.63572	153.04900
2	3	2	256	-87.74553	-44.53446	-180.38000	-87.82409	-44.43595	-177.97578
2	4	2	1	-89.52689	-40.83544	-180.38000	-89.52689	-40.83544	153.04900
2	4	2	256	-98.37877	-2.05280	-180.38000	-98.40681	-1.92996	-177.97578
2	5	2	1	-98.37877	2.05280	-180.38000	-98.37877	2.05280	153.04900
2	5	2	256	-89.52689	40.83544	-180.38000	-89.49885	40.95828	-177.97578
2	6	2	1	-87.74553	44.53446	-180.38000	-87.74553	44.53446	153.04900
2	6	2	256	-62.94311	75.63572	-180.38000	-62.86455	75.73423	-177.97578
2	7	2	1	-59.73321	78.19552	-180.38000	-59.73321	78.19552	153.04900
2	7	2	256	-23.89267	95.45542	-180.38000	-23.77915	95.51009	-177.97578
2	8	2	1	-19.89000	96.36900	-180.38000	-19.89000	96.36900	153.04900
2	8	2	256	19.89000	96.36900	-180.38000	20.01600	96.36900	-177.97578
2	9	2	1	23.89267	95.45542	-180.38000	23.89267	95.45542	153.04900
2	9	2	256	59.73321	78.19552	-180.38000	59.84673	78.14085	-177.97578
2	10	2	1	62.94311	75.63572	-180.38000	62.94311	75.63572	153.04900
2	10	2	256	87.74553	44.53446	-180.38000	87.82409	44.43595	-177.97578
2	11	2	1	89.52689	40.83544	-180.38000	89.52689	40.83544	153.04900
2	11	2	256	98.37877	2.05280	-180.38000	98.40681	1.92996	-177.97578
2	12	2	1	98.37877	-2.05280	-180.38000	98.37877	-2.05280	153.04900
2	12	2	256	89.52689	-40.83544	-180.38000	89.49885	-40.95828	-177.97578
2	13	2	1	87.74553	-44.53446	-180.38000	87.74553	-44.53446	153.04900
2	13	2	256	62.94311	-75.63572	-180.38000	62.86455	-75.73423	-177.97578
2	14	2	1	59.73321	-78.19552	-180.38000	59.73321	-78.19552	153.04900
2	14	2	256	23.89267	-95.45542	-180.38000	23.77915	-95.51009	-177.97578
3	1	1	1	-19.89000	-120.32200	-141.20600	-19.89000	-120.32200	192.22300
3	1	1	256	19.89000	-120.32200	-141.20600	20.01600	-120.32200	-138.80178
3	2	1	1	-59.84303	-106.26291	-141.20600	-59.84303	-106.26291	192.22300

Region	Sector	Module	Strip	x_u	y_u	z_u	x_d	y_d	z_d
3	2	1	256	-22.46206	-119.86848	-141.20600	-22.34366	-119.91157	-138.80178
3	3	1	1	-92.57811	-79.38695	-141.20600	-92.57811	-79.38695	192.22300
3	3	1	256	-62.10487	-104.95705	-141.20600	-62.00835	-105.03804	-138.80178
3	4	1	1	-114.14691	-42.93575	-141.20600	-114.14691	-42.93575	192.22300
3	4	1	256	-94.25691	-77.38625	-141.20600	-94.19391	-77.49536	-138.80178
3	5	1	1	-121.94790	-1.30587	-141.20600	-121.94790	-1.30587	192.22300
3	5	1	256	-115.04018	-40.48152	-141.20600	-115.01830	-40.60561	-138.80178
3	6	1	1	-115.04018	40.48152	-141.20600	-115.04018	40.48152	192.22300
3	6	1	256	-121.94790	1.30587	-141.20600	-121.96978	1.18178	-138.80178
3	7	1	1	-94.25691	77.38625	-141.20600	-94.25691	77.38625	192.22300
3	7	1	256	-114.14691	42.93575	-141.20600	-114.20991	42.82664	-138.80178
3	8	1	1	-62.10487	104.95705	-141.20600	-62.10487	104.95705	192.22300
3	8	1	256	-92.57811	79.38695	-141.20600	-92.67464	79.30596	-138.80178
3	9	1	1	-22.46206	119.86848	-141.20600	-22.46206	119.86848	192.22300
3	9	1	256	-59.84303	106.26291	-141.20600	-59.96144	106.21982	-138.80178
3	10	1	1	19.89000	120.32200	-141.20600	19.89000	120.32200	192.22300
3	10	1	256	-19.89000	120.32200	-141.20600	-20.01600	120.32200	-138.80178
3	11	1	1	59.84303	106.26291	-141.20600	59.84303	106.26291	192.22300
3	11	1	256	22.46206	119.86848	-141.20600	22.34366	119.91157	-138.80178
3	12	1	1	92.57811	79.38695	-141.20600	92.57811	79.38695	192.22300
3	12	1	256	62.10487	104.95705	-141.20600	62.00835	105.03804	-138.80178
3	13	1	1	114.14691	42.93575	-141.20600	114.14691	42.93575	192.22300
3	13	1	256	94.25691	77.38625	-141.20600	94.19391	77.49536	-138.80178
3	14	1	1	121.94790	1.30587	-141.20600	121.94790	1.30587	192.22300
3	14	1	256	115.04018	40.48152	-141.20600	115.01830	40.60561	-138.80178
3	15	1	1	115.04018	-40.48152	-141.20600	115.04018	-40.48152	192.22300
3	15	1	256	121.94790	-1.30587	-141.20600	121.96978	-1.18178	-138.80178
3	16	1	1	94.25691	-77.38625	-141.20600	94.25691	-77.38625	192.22300
3	16	1	256	114.14691	-42.93575	-141.20600	114.20991	-42.82664	-138.80178
3	17	1	1	62.10487	-104.95705	-141.20600	62.10487	-104.95705	192.22300
3	17	1	256	92.57811	-79.38695	-141.20600	92.67464	-79.30596	-138.80178
3	18	1	1	22.46206	-119.86848	-141.20600	22.46206	-119.86848	192.22300
3	18	1	256	59.84303	-106.26291	-141.20600	59.96144	-106.21982	-138.80178
3	1	2	1	19.89000	-123.80400	-141.20600	19.89000	-123.80400	192.22300
3	1	2	256	-19.89000	-123.80400	-141.20600	-20.01600	-123.80400	-138.80178
3	2	2	1	-23.65298	-123.14049	-141.20600	-23.65298	-123.14049	192.22300
3	2	2	256	-61.03395	-109.53492	-141.20600	-61.15235	-109.49183	-138.80178
3	3	2	1	-64.34305	-107.62441	-141.20600	-64.34305	-107.62441	192.22300
3	3	2	256	-94.81630	-82.05432	-141.20600	-94.91282	-81.97333	-138.80178
3	4	2	1	-97.27241	-79.12725	-141.20600	-97.27241	-79.12725	192.22300
3	4	2	256	-117.16241	-44.67675	-141.20600	-117.22541	-44.56764	-138.80178
3	5	2	1	-118.46928	-41.08617	-141.20600	-118.46928	-41.08617	192.22300
3	5	2	256	-125.37700	-1.91051	-141.20600	-125.39888	-1.78643	-138.80178
3	6	2	1	-125.37700	1.91051	-141.20600	-125.37700	1.91051	192.22300
3	6	2	256	-118.46928	41.08617	-141.20600	-118.44740	41.21025	-138.80178
3	7	2	1	-117.16241	44.67675	-141.20600	-117.16241	44.67675	192.22300
3	7	2	256	-97.27241	79.12725	-141.20600	-97.20941	79.23636	-138.80178
3	8	2	1	-94.81630	82.05432	-141.20600	-94.81630	82.05432	192.22300
3	8	2	256	-64.34305	107.62441	-141.20600	-64.24653	107.70540	-138.80178
3	9	2	1	-61.03395	109.53492	-141.20600	-61.03395	109.53492	192.22300
3	9	2	256	-23.65298	123.14049	-141.20600	-23.53457	123.18358	-138.80178
3	10	2	1	-19.89000	123.80400	-141.20600	-19.89000	123.80400	192.22300
3	10	2	256	19.89000	123.80400	-141.20600	20.01600	123.80400	-138.80178

Region	Sector	Module	Strip	x_u	y_u	z_u	x_d	y_d	z_d
3	11	2	1	23.65298	123.14049	-141.20600	23.65298	123.14049	192.22300
3	11	2	256	61.03395	109.53492	-141.20600	61.15235	109.49183	-138.80178
3	12	2	1	64.34305	107.62441	-141.20600	64.34305	107.62441	192.22300
3	12	2	256	94.81630	82.05432	-141.20600	94.91282	81.97333	-138.80178
3	13	2	1	97.27241	79.12725	-141.20600	97.27241	79.12725	192.22300
3	13	2	256	117.16241	44.67675	-141.20600	117.22541	44.56764	-138.80178
3	14	2	1	118.46928	41.08617	-141.20600	118.46928	41.08617	192.22300
3	14	2	256	125.37700	1.91051	-141.20600	125.39888	1.78643	-138.80178
3	15	2	1	125.37700	-1.91051	-141.20600	125.37700	-1.91051	192.22300
3	15	2	256	118.46928	-41.08617	-141.20600	118.44740	-41.21025	-138.80178
3	16	2	1	117.16241	-44.67675	-141.20600	117.16241	-44.67675	192.22300
3	16	2	256	97.27241	-79.12725	-141.20600	97.20941	-79.23636	-138.80178
3	17	2	1	94.81630	-82.05432	-141.20600	94.81630	-82.05432	192.22300
3	17	2	256	64.34305	-107.62441	-141.20600	64.24653	-107.70540	-138.80178
3	18	2	1	61.03395	-109.53492	-141.20600	61.03395	-109.53492	192.22300
3	18	2	256	23.65298	-123.14049	-141.20600	23.53457	-123.18358	-138.80178
4	1	1	1	-19.89000	-161.20200	-83.40500	-19.89000	-161.20200	250.02400
4	1	1	256	19.89000	-161.20200	-83.40500	20.01600	-161.20200	-81.00078
4	2	1	1	-60.93441	-150.56126	-83.40500	-60.93441	-150.56126	250.02400
4	2	1	256	-22.50988	-160.85709	-83.40500	-22.38818	-160.88970	-81.00078
4	3	1	1	-97.82625	-129.66003	-83.40500	-97.82625	-129.66003	250.02400
4	3	1	256	-63.37575	-149.55003	-83.40500	-63.26664	-149.61303	-81.00078
4	4	1	1	-128.05138	-99.92267	-83.40500	-128.05138	-99.92267	250.02400
4	4	1	256	-99.92267	-128.05138	-83.40500	-99.83358	-128.14048	-81.00078
4	5	1	1	-149.55003	-63.37575	-83.40500	-149.55003	-63.37575	250.02400
4	5	1	256	-129.66003	-97.82625	-83.40500	-129.59703	-97.93536	-81.00078
4	6	1	1	-160.85709	-22.50988	-83.40500	-160.85709	-22.50988	250.02400
4	6	1	256	-150.56126	-60.93441	-83.40500	-150.52865	-61.05612	-81.00078
4	7	1	1	-161.20200	19.89000	-83.40500	-161.20200	19.89000	250.02400
4	7	1	256	-161.20200	-19.89000	-83.40500	-161.20200	-20.01600	-81.00078
4	8	1	1	-150.56126	60.93441	-83.40500	-150.56126	60.93441	250.02400
4	8	1	256	-160.85709	22.50988	-83.40500	-160.88970	22.38818	-81.00078
4	9	1	1	-129.66003	97.82625	-83.40500	-129.66003	97.82625	250.02400
4	9	1	256	-149.55003	63.37575	-83.40500	-149.61303	63.26664	-81.00078
4	10	1	1	-99.92267	128.05138	-83.40500	-99.92267	128.05138	250.02400
4	10	1	256	-128.05138	99.92267	-83.40500	-128.14048	99.83358	-81.00078
4	11	1	1	-63.37575	149.55003	-83.40500	-63.37575	149.55003	250.02400
4	11	1	256	-97.82625	129.66003	-83.40500	-97.93536	129.59703	-81.00078
4	12	1	1	-22.50988	160.85709	-83.40500	-22.50988	160.85709	250.02400
4	12	1	256	-60.93441	150.56126	-83.40500	-61.05612	150.52865	-81.00078
4	13	1	1	19.89000	161.20200	-83.40500	19.89000	161.20200	250.02400
4	13	1	256	-19.89000	161.20200	-83.40500	-20.01600	161.20200	-81.00078
4	14	1	1	60.93441	150.56126	-83.40500	60.93441	150.56126	250.02400
4	14	1	256	22.50988	160.85709	-83.40500	22.38818	160.88970	-81.00078
4	15	1	1	97.82625	129.66003	-83.40500	97.82625	129.66003	250.02400
4	15	1	256	63.37575	149.55003	-83.40500	63.26664	149.61303	-81.00078
4	16	1	1	128.05138	99.92267	-83.40500	128.05138	99.92267	250.02400
4	16	1	256	99.92267	128.05138	-83.40500	99.83358	128.14048	-81.00078
4	17	1	1	149.55003	63.37575	-83.40500	149.55003	63.37575	250.02400
4	17	1	256	129.66003	97.82625	-83.40500	129.59703	97.93536	-81.00078
4	18	1	1	160.85709	22.50988	-83.40500	160.85709	22.50988	250.02400
4	18	1	256	150.56126	60.93441	-83.40500	150.52865	61.05612	-81.00078
4	19	1	1	161.20200	-19.89000	-83.40500	161.20200	-19.89000	250.02400

Region	Sector	Module	Strip	x_u	y_u	z_u	x_d	y_d	z_d
4	19	1	256	161.20200	19.89000	-83.40500	161.20200	20.01600	-81.00078
4	20	1	1	150.56126	-60.93441	-83.40500	150.56126	-60.93441	250.02400
4	20	1	256	160.85709	-22.50988	-83.40500	160.88970	-22.38818	-81.00078
4	21	1	1	129.66003	-97.82625	-83.40500	129.66003	-97.82625	250.02400
4	21	1	256	149.55003	-63.37575	-83.40500	149.61303	-63.26664	-81.00078
4	22	1	1	99.92267	-128.05138	-83.40500	99.92267	-128.05138	250.02400
4	22	1	256	128.05138	-99.92267	-83.40500	128.14048	-99.83358	-81.00078
4	23	1	1	63.37575	-149.55003	-83.40500	63.37575	-149.55003	250.02400
4	23	1	256	97.82625	-129.66003	-83.40500	97.93536	-129.59703	-81.00078
4	24	1	1	22.50988	-160.85709	-83.40500	22.50988	-160.85709	250.02400
4	24	1	256	60.93441	-150.56126	-83.40500	61.05612	-150.52865	-81.00078
4	1	2	1	19.89000	-164.68400	-83.40500	19.89000	-164.68400	250.02400
4	1	2	256	-19.89000	-164.68400	-83.40500	-20.01600	-164.68400	-81.00078
4	2	2	1	-23.41109	-164.22044	-83.40500	-23.41109	-164.22044	250.02400
4	2	2	256	-61.83562	-153.92462	-83.40500	-61.95733	-153.89201	-81.00078
4	3	2	1	-65.11675	-152.56553	-83.40500	-65.11675	-152.56553	250.02400
4	3	2	256	-99.56725	-132.67553	-83.40500	-99.67636	-132.61253	-81.00078
4	4	2	1	-102.38482	-130.51353	-83.40500	-102.38482	-130.51353	250.02400
4	4	2	256	-130.51353	-102.38482	-83.40500	-130.60262	-102.29572	-81.00078
4	5	2	1	-132.67553	-99.56725	-83.40500	-132.67553	-99.56725	250.02400
4	5	2	256	-152.56553	-65.11675	-83.40500	-152.62853	-65.00764	-81.00078
4	6	2	1	-153.92462	-61.83562	-83.40500	-153.92462	-61.83562	250.02400
4	6	2	256	-164.22044	-23.41109	-83.40500	-164.25305	-23.28938	-81.00078
4	7	2	1	-164.68400	-19.89000	-83.40500	-164.68400	-19.89000	250.02400
4	7	2	256	-164.68400	19.89000	-83.40500	-164.68400	20.01600	-81.00078
4	8	2	1	-164.22044	23.41109	-83.40500	-164.22044	23.41109	250.02400
4	8	2	256	-153.92462	61.83562	-83.40500	-153.89201	61.95733	-81.00078
4	9	2	1	-152.56553	65.11675	-83.40500	-152.56553	65.11675	250.02400
4	9	2	256	-132.67553	99.56725	-83.40500	-132.61253	99.67636	-81.00078
4	10	2	1	-130.51353	102.38482	-83.40500	-130.51353	102.38482	250.02400
4	10	2	256	-102.38482	130.51353	-83.40500	-102.29572	130.60262	-81.00078
4	11	2	1	-99.56725	132.67553	-83.40500	-99.56725	132.67553	250.02400
4	11	2	256	-65.11675	152.56553	-83.40500	-65.00764	152.62853	-81.00078
4	12	2	1	-61.83562	153.92462	-83.40500	-61.83562	153.92462	250.02400
4	12	2	256	-23.41109	164.22044	-83.40500	-23.28938	164.25305	-81.00078
4	13	2	1	-19.89000	164.68400	-83.40500	-19.89000	164.68400	250.02400
4	13	2	256	19.89000	164.68400	-83.40500	20.01600	164.68400	-81.00078
4	14	2	1	23.41109	164.22044	-83.40500	23.41109	164.22044	250.02400
4	14	2	256	61.83562	153.92462	-83.40500	61.95733	153.89201	-81.00078
4	15	2	1	65.11675	152.56553	-83.40500	65.11675	152.56553	250.02400
4	15	2	256	99.56725	132.67553	-83.40500	99.67636	132.61253	-81.00078
4	16	2	1	102.38482	130.51353	-83.40500	102.38482	130.51353	250.02400
4	16	2	256	130.51353	102.38482	-83.40500	130.60262	102.29572	-81.00078
4	17	2	1	132.67553	99.56725	-83.40500	132.67553	99.56725	250.02400
4	17	2	256	152.56553	65.11675	-83.40500	152.62853	65.00764	-81.00078
4	18	2	1	153.92462	61.83562	-83.40500	153.92462	61.83562	250.02400
4	18	2	256	164.22044	23.41109	-83.40500	164.25305	23.28938	-81.00078
4	19	2	1	164.68400	19.89000	-83.40500	164.68400	19.89000	250.02400
4	19	2	256	164.68400	-19.89000	-83.40500	164.68400	-20.01600	-81.00078
4	20	2	1	164.22044	-23.41109	-83.40500	164.22044	-23.41109	250.02400
4	20	2	256	153.92462	-61.83562	-83.40500	153.89201	-61.95733	-81.00078
4	21	2	1	152.56553	-65.11675	-83.40500	152.56553	-65.11675	250.02400
4	21	2	256	132.67553	-99.56725	-83.40500	132.61253	-99.67636	-81.00078

Region	Sector	Module	Strip	x_u	y_u	z_u	x_d	y_d	z_d
4	22	2	1	130.51353	-102.38482	-83.40500	130.51353	-102.38482	250.02400
4	22	2	256	102.38482	-130.51353	-83.40500	102.29572	-130.60262	-81.00078
4	23	2	1	99.56725	-132.67553	-83.40500	99.56725	-132.67553	250.02400
4	23	2	256	65.11675	-152.56553	-83.40500	65.00764	-152.62853	-81.00078
4	24	2	1	61.83562	-153.92462	-83.40500	61.83562	-153.92462	250.02400
4	24	2	256	23.41109	-164.22044	-83.40500	23.28938	-164.25305	-81.00078