Available on CMS information server **CMS NOTE-2007/033**

The Compact Muon Solenoid Experiment

CMS Note

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland

16 July 2007

A new technique for the reconstruction, validation, and simulation of hits in the CMS Pixel Detector

M. Swartz, D. Fehling, G. Giurgiu, P. Maksimovic, V. Chiochia

Abstract

This note describes new techniques for the reconstruction/validation and the simulation of pixel hits. The techniques are based upon the use of pre-computed projected cluster shapes or "templates". A detailed simulation called Pixelav that has successfully described the profiles of clusters measured in beam tests of radiation-damaged sensors is used to generate the templates. Although the reconstruction technique was originally developed to optimally estimate the coordinates of hits after the detector became radiation damaged, it also has superior performance before irradiation. The technique requires a priori knowledge of the track angle which makes it suitable for the second in a two-pass reconstruction algorithm. However, the same modest angle sensitivity allows the algorithm to determine if the sizes and shapes of the cluster projections are consistent with the input angles. This information may be useful in suppressing spurious hits caused by secondary particles and in validating seeds used in track finding. The seed validation is currently under study but has the potential to significantly increase the speed of track finding in the offline reconstruction. Finally, a new procedure that uses the templates to re-weight clusters generated by the CMSSW simulation is described. The first tests of this technique are encouraging and when fully implemented, the technique will enable the fast simulation of pixel hits that have the characteristics of the much more CPU-intensive Pixelav hits. In particular, it may be the only practical technique available to simulate hits from a radiation damaged detector in CMSSW.

1 Introduction

The standard CMSSW [1] pixel hit reconstruction algorithm [2] works very well. It is relatively insensitive to the knowledge of the track angles and functions well at the earliest stages of the track finding. It is quite insensitive to charge fluctuations caused by delta rays and has nearly optimal resolution for unirradiated detectors. It is quite simple and therefore quite fast which enables it to be used in the High Level Trigger (HLT). However, after the pixel detector becomes radiation-damaged, charge trapping will cause the projected cluster shapes to become distorted and the standard technique will develop large biases especially along the global z-direction.

This note describes a new technique that was originally intended to be “calibratable”: to be driven by parameters that could model the changing detector characteristics and to maintain optimal resolution after irradiation. The technique is based upon the fitting of the cluster projections to pre-computed shapes or “templates”. A detailed simulation called Pixelav [4] that has successfully described the profiles of clusters measured in beam tests of radiation-damaged sensors is used to generate the templates. Although the template technique was designed to function optimally after the detector became radiation-damaged, it also has superior performance before irradiation. Unlike the standard technique, the template technique requires a priori knowledge of the track angle which makes it suitable for the second in a two-pass reconstruction algorithm. However, the same modest angle sensitivity allows the algorithm to determine if the sizes and shapes of the cluster projections are consistent with the input angles. This information is useful in suppressing spurious hits caused by secondary particles and potentially in validating seeds used in track finding. Finally, a spin-off of the new reconstruction procedure is a parametric simulation procedure that uses the templates to re-weight clusters generated by the CMSSW simulation. This technique permits the fast simulation of pixel hits that have the characteristics of the much more CPU-intensive Pixelav hits. In particular, it may be the only technique that has the potential to simulate hits from a radiation damaged detector in CMSSW.

This note is organized as follows: a brief description of the pixel geometry is given in Section 2, the characteristics of pixel clusters are described in Section 3, the Pixelav simulation is summarized in Section 5, the new “Template Algorithm” is described in Section 6, the use of the templates in simulation is described in Section 7, and some concluding remarks are given in Section 8.

2 CMS Pixel Geometry

The CMS pixel tracking system [5] is shown in Fig. 1. It consists of 3 cylindrical 0.5-m long barrels at radii of 4 cm, 7 cm, and 11 cm and endcaps of 2 forward disks having sensitive regions between radii 5.8 cm and 14.5 cm. The disks are located at 32.5 cm and 48.5 cm from the interaction point which provides 3 hit coverage over the region of pseudorapidity $|\eta| \leq 2.5$. A complete description of the pixel system geometry is given on the pixel geometry TWiki page [6].

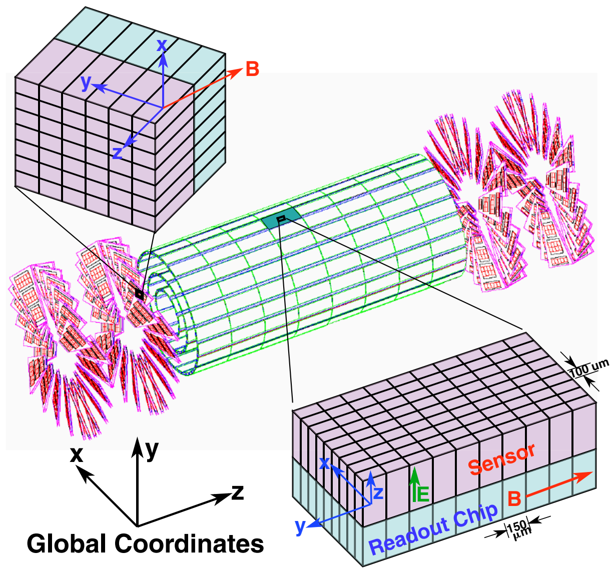


Figure 1: Layout of the CMS pixel tracking system and examples of local coordinate systems.

The barrels and disks are composed of planar detector elements which have the sandwich structure shown in Fig. 1. Barrel modules or forward plaquettes are rectangular arrays of (rectangular) silicon diodes that are bump-bonded to readout chips. Each 150×100 μm diode is connected to an individual readout circuit having exactly the same transverse dimensions. The readout chip, which provides analog signals to an 8-bit encoder, is described in detail in Ref. [5]. Each module or plaquette has a local coordinate system defined by the intra-pixel electric field direction (the z-direction) and the direction of first-order Lorentz drift $\vec{E}\times\vec{B}$ (the x-direction). The x-directions of all local frames are sampled by the 100 μm pitch of the pixels and the y-directions are sampled by the 150 μm pitch of the pixels. As is shown in Fig. 1, the x-direction for barrel modules that have the sensors mounted on the outside radius of the support structure ("unflipped" modules) is approximately in the negative azimuthal direction ($-\phi$-direction) of the global coordinate system. The x-direction for "flipped" modules that have the sensors mounted on the inside radius of the support structure is in the positive azimuthal direction. The y-axes for flipped and unflipped modules point in the negative z-direction in the global coordinate system. As is shown in Fig. 1, forward pixel local coordinate systems have their x-axes in the positive radial direction on the blades facing away from the interaction point (the 3-plaquette sides of the blades). The forward local frames have their x-axes pointing radially inward on the sides of the blades that face the interaction point (4-plaquette sides) such that the y-directions of the frames on either side of a blade are coincident.

3 Pixel Clusters

The deposition of charge by a track having angles α and β with respect to the local x- and y-axes of a barrel module is shown in Fig. 2. The primary track deposits approximately 25,000 electron-hole pairs per 300 μm of track length more or less uniformly in the y-direction. For highly inclined tracks, about 12,500 pairs are deposited in each 150 μm wide pixel column. The n-in-n sensors collect electrons which have a large Lorentz angle ($\sim 23^\circ$ at 150V bias [7]) in the 4 T magnetic field of CMS. The charge from the larger local z-side of the sensor typically drifts by more than a pixel x-width into the adjacent row of pixels producing clusters with the typical shape shown in Fig. 3. The track projection is shown as the dashed red line on the cluster. Note that track center, shown as the cross, is contained in a pixel that does not have enough charge (the threshold is approximately 2.5k electrons) to trigger its readout. The primary ionization process produces large fluctuations in charge along the track. Note that any pixel signal larger than the most probable one for a full track-traversal of the pixel does not contain useful position information. Energetic delta rays often cross pixel cell boundaries causing strong charge correlations between adjacent pixel cells and sometimes causing unusual cluster shapes.

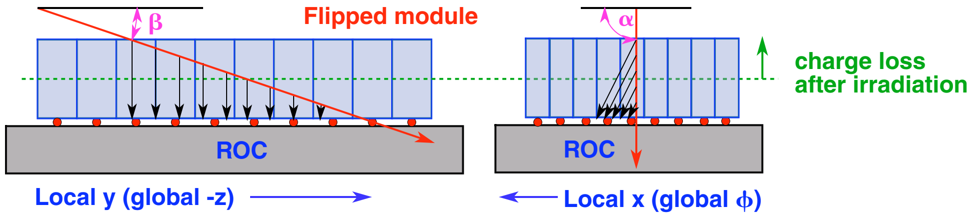


Figure 2: Geometrical and Lorentz-drift induced charge sharing in a "flipped" pixel barrel module.

The single most important feature of pixel clusters is that the shape of the x-projection of the cluster is independent of the y-position of the hit and the y-size of the cluster (independent of the angle β). Similarly, the shape of the y-projection of the cluster is independent of the x-position of the hit and the x-size of the cluster (independent of the angle α). This x-y factorization is a consequence of the facts that the field configurations in the pixels don't couple the two coordinates except perhaps in the corners of the cells where there are small 2-d focusing effects and that the pixels have a periodic structure. This property of the system is heavily exploited by the standard reconstruction algorithm [2] and by the template algorithm described in this note. They sum the x- and y- charge projections of the two-dimensional clusters and treat the projections or profiles independently. The y-profiles for a large sample of $\beta = 15^\circ$ tracks that were measured from several test sensors are shown in Fig. 4. Note that the unirradiated sensor (fluence $\Phi = 0$) has a rectangular profile with well defined edges. The average signal in the interior pixels of the projection is constant as expected. These (projected) pixels contain no position information. A simple analysis shows that if one assumes that the statistical uncertainty on a signal $s(y)$ is proportional to $\sqrt{s}$, then the uncertainty on the parameter y, δy, is given by the following expression,

$$\delta y = C\frac{\sqrt{s(y)}}{ds/dy} \tag{1}$$

were C is a constant. This suggests that all of the position information is contained in the smaller signals near the cluster edges where the slope is largest. After irradiation to fluence $\Phi = 8 \times 10^{14}$ $\mathrm{n_{eq}/cm^2}$, charge trapping causes the cluster to have a bias-voltage-dependent shape. Note that although charge is preferentially lost from the "far" end of the cluster, the interior pixels now contain position information. A summary of the key features of the pixel clusters follows:

- The shapes of x- and y-projections of the two-dimensional pixel clusters are independent.
- There is no position information in very large pixel signals. Once the maximum signal is exceeded, one only learns about the likelihood of energetic delta ray emission (still useful information).
- The best position information is contained in the small pixel signals near the cluster ends.

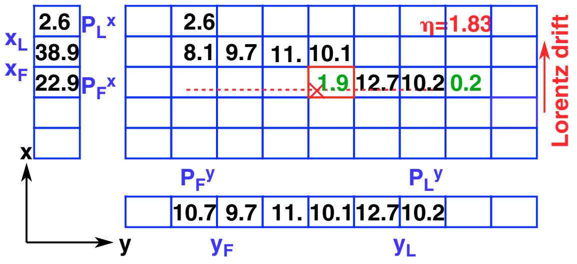


Figure 3: Cluster shape for the barrel hit shown in Fig. 2. The signals in each pixel are given in kilo-electrons. Those shown in green are below the readout threshold. The track projection is shown as the dashed red line. The x- and y-projections are also shown as one-dimensional arrays. The coordinates of the boundaries between the first and second pixels (x_F/y_F) and the next-to-last and last pixels (x_L/y_L) and the charges of the first and last pixels $P^{x/y}_{F/L}$ are also shown.

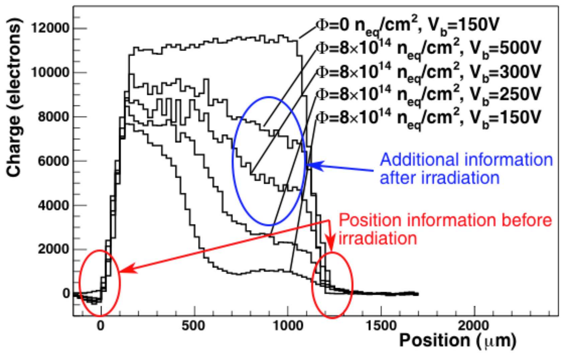


Figure 4: Charge collection profiles for 125x125μm^2 test sensors illuminated by a $\beta = 15°$ test beam. An unirradiated sensor (fluence $\Phi = 0$) is compared with a heavily irradiated sensor (fluence $\Phi = 8 \times 10^{14}$ $\mathrm{n_{eq}/cm^2}$) operated at several bias voltages.

4 Standard Reconstruction Technique

The standard technique for the reconstruction of pixel hits in CMSSW is an "eta-like" technique that uses the signals in the first and last pixels of the x and y cluster projections $P^{x/y}_{F/L}$. The use of the first and last projected pixel charges reduces the sensitivity of the procedure to delta ray emission which becomes quite likely in long clusters. The reconstructed hit coordinates in each projection are given by the following expressions [3],

$$x_{rec} = \frac{x_F + x_L}{2} + \frac{P^x_L - P^x_F}{P^x_L + P^x_F} \cdot \frac{W^x_{\text{eff}}(\cot\alpha)}{2} - \frac{\Delta_x}{2} \tag{2}$$

$$y_{rec} = \frac{y_F + y_L}{2} + \frac{P^y_L - P^y_F}{P^y_L + P^y_F} \cdot \frac{W^y_{\text{eff}}(\cot\beta)}{2} - \frac{\Delta_y}{2} \tag{3}$$

where: $x_{F/L}$ and $y_{F/L}$ are the coordinates of the first/second and last/next-to-last pixel boundaries (defined in Fig. 3), W^x_{eff} and W^y_{eff} are the total charge widths in the end pixels, and Δ_x and Δ_y are the maximum Lorentz-drift in the x- and y-directions. Note that Δ_y vanishes in the pixel barrel but is non-zero in the pixel endcaps. The effective charge widths in the two projections are given by the following expression

$$W^x_{\text{eff}}(\cot\alpha) = |T\cot\alpha + \Delta_x| - (x_L - x_F) \tag{4}$$

$$W^y_{\text{eff}}(\cot\beta) = |T\cot\beta + \Delta_y| - (y_L - y_F) \approx \frac{\text{pitch}^y_F + \text{pitch}^y_L}{2} \tag{5}$$

where: T is the sensor thickness, and $\mathrm{pitch}^y_{F/L}$ are the pitches of the first and last pixels in the y-projection. These expressions are valid when the cluster contains the double-size pixels that are present at the edges of the readout chips. The use of the average pitch sizes to approximate W^y_{eff} makes it insensitive to the track direction and appropriate for the first pass of a two pass hit reconstruction algorithm without sacrificing much resolution. Problems do arise, however, when equations 2 and 3 are used to reconstruct hits in a radiation damaged detector. After an exposure of 6×10^{14} $\mathrm{n_{eq}}$/cm^2, the residual distributions develop biases of 30-50 μm and the resolutions are significantly worsened. To overcome these difficulties, a new technique that uses a priori information to fit the entire projected cluster shapes was developed. It is based upon a detailed simulation that was developed to interpret several beam test measurements. The following sections describe the simulation and the new simulation-based reconstruction technique.

5 Pixelav Simulation

The detailed sensor simulation, Pixelav [4], incorporates the following elements: an accurate model of charge deposition by primary hadronic tracks (in particular to model delta rays) [8]; a realistic electric field map resulting from the simultaneous solution of Poisson's Equation, carrier continuity equations, and various charge transport models; an established model of charge drift physics including mobilities, Hall Effect, and 3-D diffusion; a simulation of charge trapping and the signal induced from trapped charge; and a simulation of electronic noise, response, and threshold effects.

Several of the Pixelav details described in [4] have changed since they were published. The commercial semiconductor simulation code now used to generate a full three dimensional electric field map is the ISE TCAD package [9]. The charge transport simulation originally integrated the position and velocity equations which required very small step sizes to maintain stability. It was modified to integrate only the position equation by using the fully-saturated drift velocity,

$$\frac{d\vec{r}}{dt} = \frac{\mu\left[q\vec{E} + \mu r_H \vec{E}\times\vec{B} + q\mu^2 r_H^2(\vec{E}\cdot\vec{B})\vec{B}\right]}{1+\mu^2 r_H^2|\vec{B}|^2} \tag{6}$$

where $\mu(\vec{E})$ is the mobility, $q = \pm 1$ is the sign of the charge carrier, $\vec{E}$ is the electric field, $\vec{B}$ is the magnetic field, and r_H is the Hall factor of the carrier. The use of the fully-saturated drift velocity permits much larger integration steps and significantly increases the speed of the code. A final speed enhancement results from the implementation of adaptive step sizing in the Runge-Kutta integrations using the Cash-Karp embedded 5th-order technique [10]. Pixelav was developed to use the vector (SIMD) processing on the PowerPC G4 and G5 families of processors. A port to the less capable Intel SSE architecture has recently been performed. Early testing indicates that the speed of the ported code running on a 2.8 GHz Xeon is approximately 50% of the speed achieved on a 2.5 GHz G5 processor.

The simulation was originally written to interpret beam test data from several unirradiated and irradiated sensors. It was extremely successful in this task, demonstrating that simple type inversion is unable to describe the measured charge collection profiles in irradiated sensors and yielding unambiguous observations of doubly-peaked electric fields in those same sensors [11]. In these studies, charge collection across the sensor bulk was measured using the "grazing angle technique" [12]. As is shown in Fig. 5, the surface of the test sensor was oriented by a small angle (15°) with respect to the pion beam. Several samples of data were collected with zero magnetic field and at temperature of -10°C. The charge measured by each pixel along the y direction sampled a different depth z in the sensor. Precise entry point information from the beam telescope was used to produce finely binned charge collection profiles.

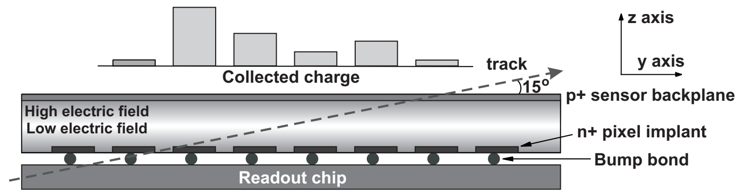


Figure 5: The grazing angle technique for determining charge collection profiles. The charge measured by each pixel along the y direction samples a different depth z in the sensor.

The charge collection profiles for a sensor irradiated to a fluence of $\Phi = 5.9 \times 10^{14}\ \mathrm{n_{eq}/cm^2}$ and operated at bias voltages of 150V, 200V, 300V, and 450V are presented in Fig 6. The measured profiles are shown as solid dots and thePixelav- simulated profiles are shown as histograms. The Pixelav simulations based upon the electric field produced by a tuned two-trap model [11].

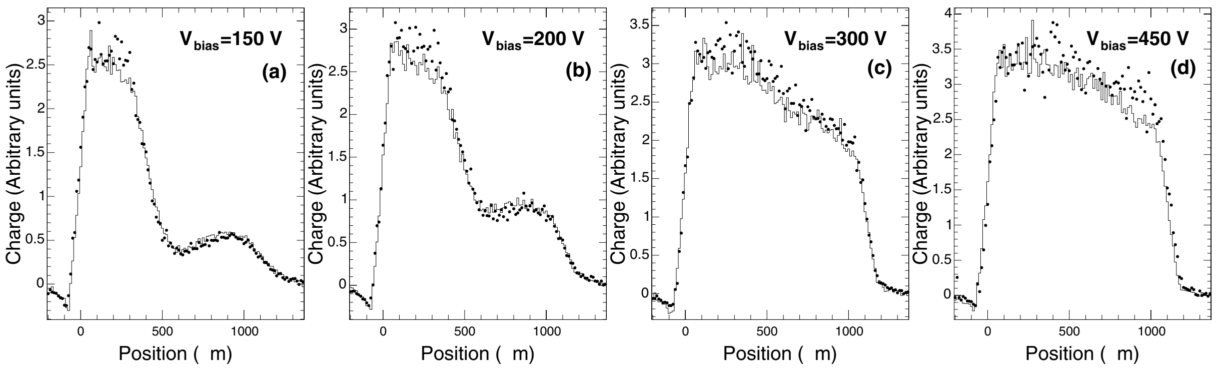


Figure 6: The measured charge collection profiles at a temperature of $-10°$C and bias voltages of 150V, 200V, 300V, and 450V are shown as solid dots for a fluence of 5.9×10^{14} $\mathrm{n_{eq}/cm^2}$. The two-trap double junction simulation is shown as the solid histogram in each plot.

The simulation describes the measured charge collection profiles well both in shape and normalization (the charge scale of the data is uncertain at the 10% level). The apparently unphysical "wiggle" observed at low bias voltages is actually the signature of a doubly peaked electric field having a minimum near the midplane of the sensor and maxima at the n+ and p+ implants. The relative signal minimum near $y = 700$ μm corresponds to the minimum of the electric field z-component, E_z, where both electrons and holes travel only short distances before trapping. This small separation induces only a small signal on the n^+ side of the detector. At larger values of y, E_z increases causing the electrons drift back into the minimum where they are likely to be trapped. However, the holes drift into the higher field region near the p^+ implant and are more likely to be collected. The net induced signal on the n^+ side of the detector therefore increases and creates the local maximum seen near $y = 900$ μm.

6 Template Reconstruction Algorithm

The template-based reconstruction algorithm is a procedure that translates pre-stored cluster projection shapes, also called "templates", across measured cluster projections to find the best fit and hence an estimate of the hit position in both x and y. The Pixelav simulation is used to generate the templates which are stored as functions of $\cot\alpha$ and $\cot\beta$ along with large quantities of auxiliary information in a template object. The following sections describe this procedure.

6.1 Motivation

One of the original motivations for the template-based reconstruction algorithm was the realization that radiation damage would significantly change the charge sharing functions of the detector during large portions of its useful life. Any reconstruction algorithm that was not tunable would become biased and non-optimal as the detector ages. Another motivation was the observation [shown in Fig. 4] that the interior pixel signals in the y-projections of long barrel clusters would acquire position sensitivity as the detector ages. The "Standard" algorithm uses only the end pixels of the projections which is nearly optimal before aging but becomes less so after irradiation. The implementation of an algorithm that uses all of the (projected) pixel information was an obvious choice. Since Pixelav had demonstrated that it could describe the behavior of a heavily irradiated detector and since we had demonstrated that we could tune that description, it seemed obvious to base a more capable algorithm on the detailed simulation. This has numerous advantages in implementation over a purely data-driven approach. Once the detailed simulation has been tuned, it can generate cluster shapes, predict resolutions, and provide goodness-of-fit normalizations for a large range of track angles and cluster charges independently of other detector subsystems and their state of operation (ie alignment). In effect, Pixelav becomes a "software test beam" replacing the very limited pixel beam test data available.

6.2 Coordinate System

The pixel system local coordinate systems were chosen to simplify the pixel software downstream of the event unpacking (Raw2Digi step) as much as possible. The barrel and forward pixel systems are fundamentally different because they have different orientations of the magnetic field in their local frames. The 4T magnetic field is perpendicular to the electric fields of the barrel modules producing a Lorentz angle in the local x-z plane of approximately 23° at 150V bias [7]. In the forward local frames, the magnetic field direction deviates fom the electric field directions by 20° producing a much smaller Lorentz angle of approximately 7.2° in the local x-z plane. In addition, because the angle between the electric and magnetic fields is between 0° and 90°, a second-order Lorentz drift also occurs in the local y-z plane [see equation 6] having an angle of approximately 4.5°. The direction of the second-order Lorentz drift changes sign on opposite sides of the forward blades (the local y-axes on opposite sides of the blades are coincident). After irradiation, signal loss caused by carrier trapping makes the clusters measured by both detectors asymmetric. These effects are shown in Figs. 7 and 8 for the barrel and forward detectors, respectively.

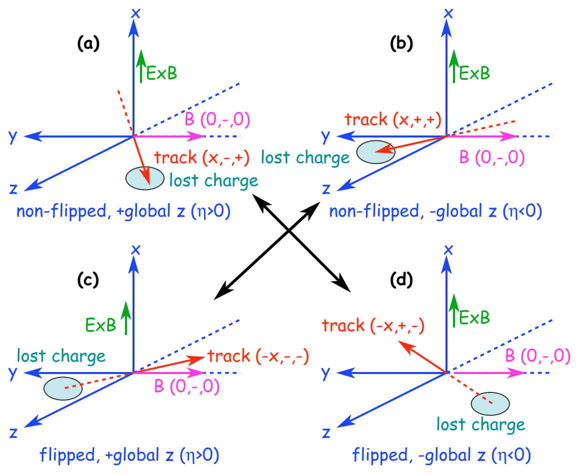


Figure 7: The flipped and non-flipped local barrel coordinate systems for tracks having positive and negative pseudorapidity. The octants with charge lost from trapping are also shown.

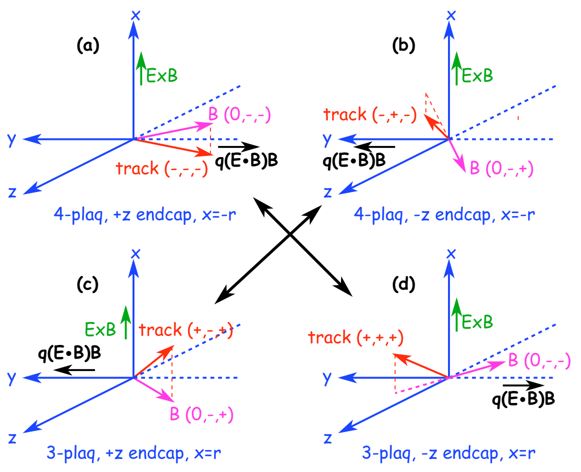


Figure 8: The 4-plaquette and 3-plaquette local coordinate systems in the forward and backward encaps. The y-components of the second-order Lorentz drift are indicated by the black arrows.

The track direction, B-field direction, Lorentz-drift direction, and octant with reduced signal from trapping are shown for positive and negative pseudorapidity (η) tracks in the flipped and non-flipped local barrel frames in Fig. 7. A bit of scrutiny will reveal that the charge drift in the non-flipped frame at positive η and the flipped frame at negative η is identical. They differ only in that the track direction is inverted. Note that the same is true of the other two cases. This leads to the convenient conclusion that the charge drift and cluster shape can be entirely described in terms of $\cot\alpha = n_x/n_z$ and $\cot\beta = n_y/n_z$ where (n_x, n_y, n_z) is the direction of the track. [Note that $\cot\alpha$ and $\cot\beta$ are invariant under the inversion of the track direction but differ in sign between cases (a)-(d) and (b)-(c) in Fig. 7.] These are signed quantities that are related to the lengths of the x- and y-projections of the cluster, L_x and L_y,

$$L_x = |T\cot\alpha + \Delta_x|, \qquad L_y = |T\cot\beta + \Delta_y| \tag{7}$$

where T, Δ_x, and Δ_y were defined in Section 4.

A similar analysis in the forward local frames is shown in Fig. 8 where the second-order Lorentz drift along the y-axes is shown and the charge loss from trapping is suppressed for clarity. One can again see that the positive-η 4-plaquette panel frame and the negative-η 3-plaquette panel frame differ only in the inversion of the track direction. Similarly, the negative-η 4-plaquette panel frame and the positive-η 3-plaquette panel frame are equivalent. The cluster shape is again described entirely by the two parameters $\cot\alpha$ and $\cot\beta$. This choice of frames implies that knowledge of detector IDs is completely unnecessary to reconstruct or simulate pixel hits. One needs only to know $\cot\alpha$, $\cot\beta$, and whether a barrel for forward detector is involved.

The factorization of the x- and y-projections of the pixel clusters discussed in Section 3 implies that the shape of the x-projection of a cluster depends only upon $\cot\alpha$ and that the shape of the y-projection depends only upon

$\cot\beta$. One final simplification is to note that the y-projections of the clusters in the (a)-(d) and (b)-(c) cases are mirror images. This will simplify the storage of the templates discussed in the following section.

6.3 First Pass Template Generation

The template algorithm requires a-priori knowledge of the projected cluster shapes as functions of $\cot\alpha$, $\cot\beta$, and the hit position. This information is extracted from 30000-event samples simulated by Pixelav at fixed track angle and random hit position. The charge distribution of a sample of $\cot\beta = 1.97$ barrel clusters from an unirradiated sensor is shown in Fig. 9. Note that the significant Landau tail is caused by energetic delta ray emission. Since delta-rays distort the cluster shapes, the template generation procedure utilizes only those events having less than the average cluster charge Q_{avg}. This retains approximately 64% of the (asymmetrically-distributed) sample and yields an accurate determination of the projected cluster shapes as caused by the geometrical, charge drift, trapping, and charge induction effects. Note that the determination of the average cluster shapes is quite insensitive to the exact value of the cluster charge requirement. The average RMS template resolutions in x and y are shown in Table 1 as functions of the maximum cluster charge used to calibrate the technique. The resolutions and sample fractions are averaged over all cluster lengths from 0 to 11.5 pixels ($\eta = 0 - 2.5$). Note that the resolutions are improved by the cluster charge requirement but are also extremely insensitive to its exact value.

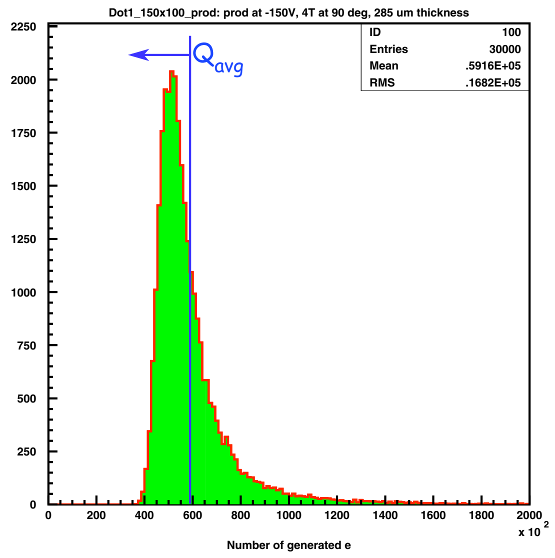
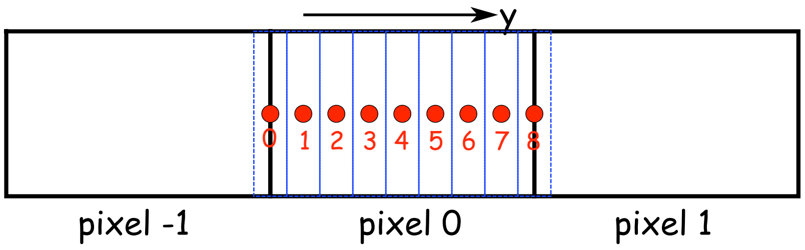


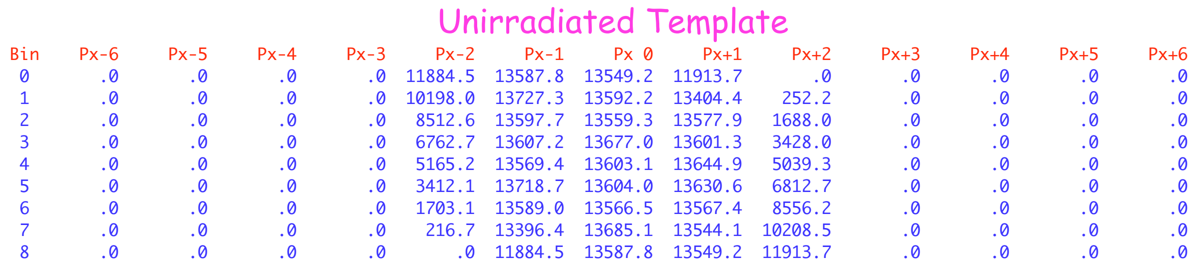
Figure 9: The charge distribution of a sample of $\cot\beta = 1.97$ barrel clusters in electrons. Removing events with cluster charges larger than the 59,000 electron average retains about 64% of the sample.

Table 1: Comparison of the average RMS template resolution in x and y as functions of the maximum cluster charge used to generate the template shapes and some auxiliary information. The average fraction of the calibration sample that is retained is also listed. The averages include all cluster lengths from 0 to 11.5 pixels ($\eta = 0 - 2.5$).

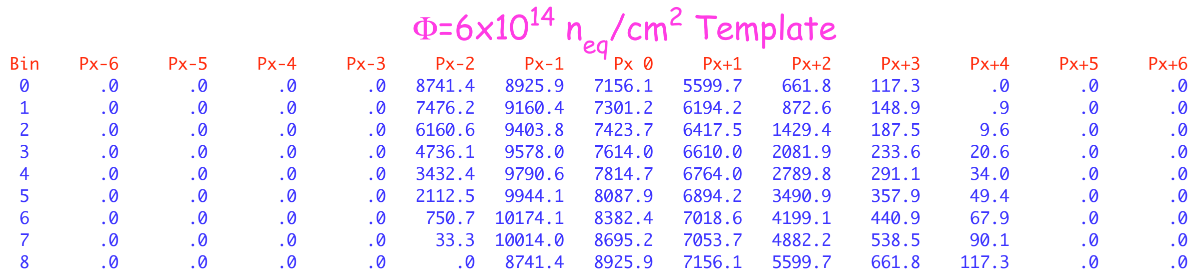
Maximum Cluster Charge	Fraction of Events	x RMS (μm)	y RMS (μm)
$0.85Q_{avg}$	0.29	8.60	20.03
$1.00Q_{avg}$	0.69	8.61	20.32
$1.15Q_{avg}$	0.87	8.59	20.68
$10.0Q_{avg}$	1.00	9.10	22.51

The template generation is done in two passes. The first pass processing is described in this section and the second pass is described in section 6.4. During the first pass, the x- and y-projections for each simulated cluster are summed into respective 7-pixel and 21-pixel arrays. By construction, the struck pixel is the center pixel in each projection. The x and y coordinates of the hit are each binned in bins of width 0.125 pixel pitch. The bins are chosen so that the middle bin is centered on the pixel center and the end bins are centered of the pixel boundaries. This yields 9 bins spanning the central pixel (pixel 0) where the end pixels differ by a full pixel pitch as shown in Fig. 10 for the y-projections of unirradiated and heavily irradiated (fluence $\Phi = 6 \times 10^{14}\mathrm{n_{eq}}/\mathrm{cm}^2$) $\cot\beta = 1.97$ samples. The total charge, square of the charge, and number of entries are summed for each of the 9 (8 independent) bins. The template consists of the average signal $S^{y/x}_{i,j}$ in each projected pixel i and bin j. The resulting templates for the unirradiated and irradiated sensors are shown in Fig. 10. Note that trapping reduces the projected signals but produces apparently larger clusters from charge induction. The application of the 2500 electron readout threshold actually reduces the observed cluster size.




Unirradiated Template

Bin	Px-6	Px-5	Px-4	Px-3	Px-2	Px-1	Px 0	Px+1	Px+2	Px+3	Px+4	Px+5	Px+6
0	.0	.0	.0	.0	11884.5	13587.8	13549.2	11913.7	.0	.0	.0	.0	.0
1	.0	.0	.0	.0	10198.0	13727.3	13592.2	13404.4	252.2	.0	.0	.0	.0
2	.0	.0	.0	.0	8512.6	13597.7	13559.3	13577.9	1688.0	.0	.0	.0	.0
3	.0	.0	.0	.0	6762.7	13607.2	13677.0	13601.3	3428.0	.0	.0	.0	.0
4	.0	.0	.0	.0	5165.2	13569.4	13603.1	13644.9	5039.3	.0	.0	.0	.0
5	.0	.0	.0	.0	3412.1	13718.7	13604.0	13630.6	6812.7	.0	.0	.0	.0
6	.0	.0	.0	.0	1703.1	13589.0	13566.5	13567.4	8556.2	.0	.0	.0	.0
7	.0	.0	.0	.0	216.7	13396.4	13685.1	13544.1	10208.5	.0	.0	.0	.0
8	.0	.0	.0	.0	.0	11884.5	13587.8	13549.2	11913.7	.0	.0	.0	.0


Φ=6x10^{14} n$_{eq}$/cm^2 Template

Bin	Px-6	Px-5	Px-4	Px-3	Px-2	Px-1	Px 0	Px+1	Px+2	Px+3	Px+4	Px+5	Px+6
0	.0	.0	.0	.0	8741.4	8925.9	7156.1	5599.7	661.8	117.3	.0	.0	.0
1	.0	.0	.0	.0	7476.2	9160.4	7301.2	6194.2	872.6	148.9	.9	.0	.0
2	.0	.0	.0	.0	6160.6	9403.8	7423.7	6417.5	1429.4	187.5	9.6	.0	.0
3	.0	.0	.0	.0	4736.1	9578.0	7614.0	6610.0	2081.9	233.6	20.6	.0	.0
4	.0	.0	.0	.0	3432.4	9790.6	7814.7	6764.0	2789.8	291.1	34.0	.0	.0
5	.0	.0	.0	.0	2112.5	9944.1	8087.9	6894.2	3490.9	357.9	49.4	.0	.0
6	.0	.0	.0	.0	750.7	10174.1	8382.4	7018.6	4199.1	440.9	67.9	.0	.0
7	.0	.0	.0	.0	33.3	10014.0	8695.2	7053.7	4882.2	538.5	90.1	.0	.0
8	.0	.0	.0	.0	.0	8741.4	8925.9	7156.1	5599.7	661.8	117.3	.0	.0

Figure 10: The signal averages $S^{y}_{i,j}$ of 13 of the pixels in the y-projection of $\cot\beta = 1.97$ barrel clusters for each of 9 bins in the y hit position. They are shown for unirradiated and heavily irradiated sensors ((fluence $\Phi = 6 \times 10^{14}\mathrm{n_{eq}/cm^2}$).

The same procedure is also used to calculate the expected rms, $\Delta S^{y/x}_{i,j}$ of the average signals. The $\Delta S^{y/x}_{i,j}$ for the unirradiated $\cot\beta = 1.97$ sample are plotted vs the projected signals in Fig. 11. The signal/rms points for pixels from either side of the cluster projection are shown as different colors. The signals from the "near" side, the side with the shorter carrier drift path to the readout chip, are shown as red points. The signals from the "far" side, the side with the longer carrier drift path, are shown as blue points. The two sets of points are fit to independent functions of the form

$$\Delta S^{y/x}_{i,j} = \sqrt{a + bS^{y/x}_{i,j} + c(S^{y/x}_{i,j})^2 + d(S^{y/x}_{i,j})^3 + e(S^{y/x}_{i,j})^4} \qquad (8)$$

where a-e are constants. The best fits to the near and far side data are shown as red and blue solid curves in Fig. 11. Note that there are no differences between the cluster ends for an unirradiated sensor and that the RMSs scale dominantly as $\Delta S^{y/x}_{i,j} \propto \sqrt{S^{y/x}_{i,j}}$. The same information is shown for the irradiated sensor in Fig. 12. Note that the fluctuations of the far end are significantly reduced by charge trapping and that the scaling of $\Delta S^{y/x}_{i,j}$ is approximately linear in $S^{y/x}_{i,j}$.

An identical procedure is applied to the x-projection of each cluster at each set of track angles. As was discussed in Section 3, the shapes of the x-projections are independent of $\cot\beta$ and depend upon $\cot\alpha$ only. The normalization of the projected x-signals does depend upon $\cot\beta$, however, the fitting algorithm discussed in Section 6.4 is insensitive to the normalization. Therefore, a single set of x-projections spanning the relevant range in $\cot\alpha$ is sufficient to fit all clusters. The predicted RMS uncertainties of the signals do depend upon $\cot\beta$, however, they do so in a scalable way. This is shown in Fig. 13 where the rms and average x-signals are plotted for three values of $\cot\beta$ corresponding to the three values of pseudorapidity: 0.5, 1.5, and 2.0. The best fits for the near and far cluster ends at $\eta = 2.0$ are scaled by the factor $\sqrt{Q_{avg}(\eta)/Q_{avg}(2.0)}$ and shown as the dashed ($\eta = 1.5$) and

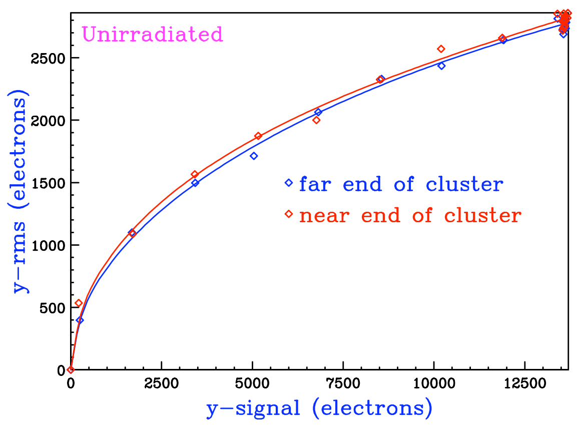


Figure 11: The rms versus signal for the y-projected pixels of the unirradiated $\cot\beta = 1.97$ sample. The signals from the near side, the side with the shorter carrier drift path to the readout chip, are shown as red points. The signals from the far side, the side with the longer carrier drift path, are shown as blue points. The solid curves are best fits to equation 8.

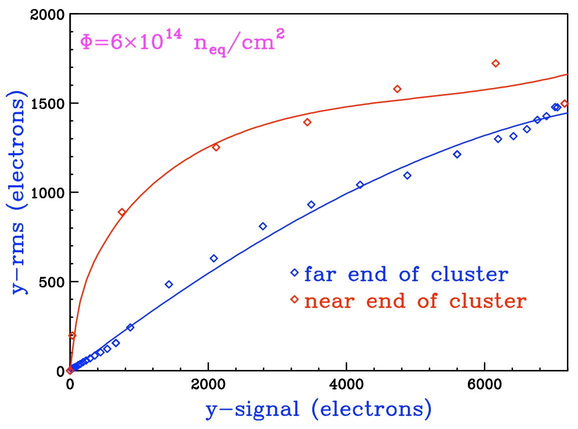


Figure 12: The rms versus signal for the y-projected pixels of the $\cot\beta = 1.97$ sample irradiated to $\Phi = 6\times10^{14}\mathrm{n_{eq}/cm^2}$.The signals from the near side are shown as red points and the signals from the far side are shown as blue points. The solid curves are best fits to equation 8.

dotted ($\eta = 0.5$) curves. It is clear that a single set of rms functions can be scaled to other values of $\cot\beta$.

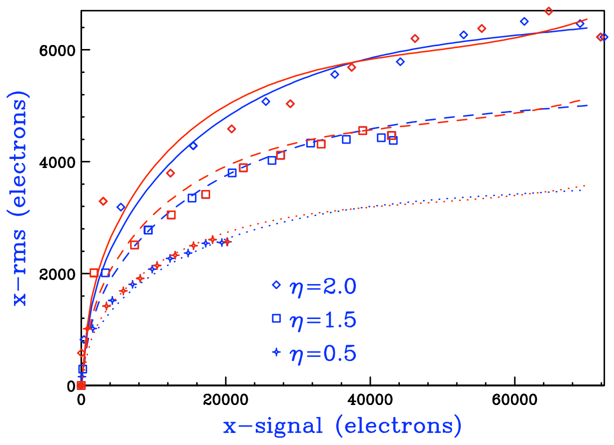


Figure 13: The rms and average x-signals are plotted for three values of $\cot\beta$ corresponding to the three values of pseudorapidity: 0.5, 1.5, and 2.0. The best fits for the near and far cluster ends at $\eta = 2.0$ are scaled by the factor $\sqrt{Q_{avg}(\eta)/Q_{avg}(2.0)}$ and shown as the dashed ($\eta = 1.5$) and dotted ($\eta = 0.5$) curves.

The first pass of the template generation algorithm produces 9-bin templates in both x and y; 5-parameter descriptions of the x-rms and y-rms functions for both near and far ends of the clusters; the average charge Q_{avg}; and maximum signals for the x- and y-projections, S^x_{max} and S^y_{max}. These are stored in individual files for each each set of track angles. The barrel track angles are chosen to sample the y-cluster length $T\cot\beta$ in 0.25 pixel increments from 0 pixels ($\eta = 0$) to 11.5 pixels ($\eta = 2.5$) [it was found that coarser 0.5 pixel sampling lead to interpolation errors and resolution loss at the 5% level for the worst cases (midway between the points)]. Since displaced vertices produce acceptance tails to $\eta = 2.9$ and the long clusters in this region are very expensive computationally, coarser 0.5 pixel sampling was chosen from 11.5 to 18 pixel y-cluster lengths. The $\cot\alpha$ values are chosen to sample $\alpha' = \alpha - \pi/2$ in 0.075 radian increments from -0.225 to 0.225 radians in the barrel. Even though a single set of $\cot\alpha$ angles at a common value of $\cot\beta$ is adequate to describe the x-shapes of all clusters, some x-related information generated in the second-pass of the template generation procedure does require additional sampling in $\cot\beta$. Therefore, five $\cot\alpha$ sweeps at $T\cot\beta$ values of 0, 1.75, 3.5, 7.0, and 11.0 pixels are performed. All generated ($\cot\alpha$, $\cot\beta$) points are shown graphically in Fig. 14. For the forward pixels, the acceptance is much smaller so that $T\cot\beta$ is sampled in three 0.25-pixel increments between 0.45 and 0.95 pixels. The α' acceptance in the forward pixels is enlarged by displaced vertices and is sampled in 0.075 radian increments from 0.075 to 0.675 radians. These points are sampled at the extreme values of $\cot\beta$ and are shown in Fig. 14.

The templates are chosen to sample only positive $\cot\beta$ because, as was mentioned at the end of Section 6.2, negative $\cot\beta$ y-projections are mirror images of the positive $\cot\beta$ projections in the barrel and forward systems. The following expression is used to create 9-bin templates at negative $\cot\beta$,

$$S^y_{i,j}(-\cot\beta) = S^y_{20-i,8-j}(\cot\beta) \tag{9}$$

where 20 is the maximum pixel index in a y-template, and 8 is the maximum bin index.

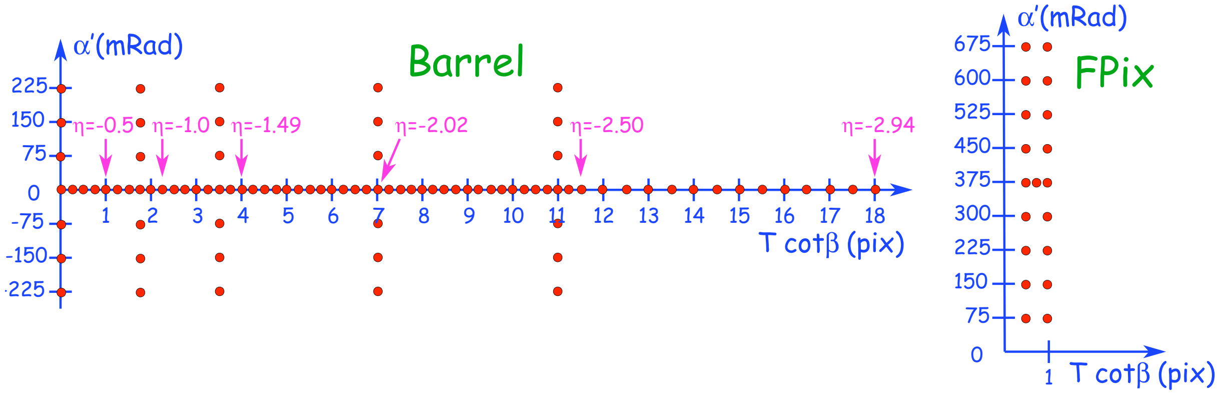


Figure 14: The values of $\alpha' = \alpha - \pi/2$ and $T\cot\beta$ used to generate a set of templates are plotted as red points for the barrel and forward detectors. The values of pseudorapidity η corresponding to various values of $T\cot\beta$ are also shown for reference.

6.4 Second Pass Template Generation: Reconstruction Algorithm

The second pass of the template generation uses the pre-stored results of the first pass to apply the actual template reconstruction algorithm to the same data samples used to generate the 9-bin templates. The second pass generates information on biases, errors, corrections, and goodness-of-fit that are combined with the results of the first pass to build a 448 kB ascii template summary file that represents a given set of operating conditions as simulated by Pixelav.

6.4.1 Philosophy and Strategy

A simple description of the template algorithm is that it translates and fits the pre-tabluated projected cluster shapes to the measured projections of real data clusters to estimate the best hit position. This is a loaded statement because the measured signals have large fluctuations caused by delta rays. The delta rays also produce strong correlations in the fluctuations of adjacent pixels. A correct statistical treatment involves considerable technical complexity. Luckily, we can appeal to the observation made in Section 3 that the hit position information is contained primarily in the small signals. Large signals carry no information and are also likely to involve the large fluctuations that complicate any analysis. Our use of the low Q events to make the 9-bin templates avoided the effects of the fluctuations on the template shapes. We will try to do the same on the entire sample by limiting the size of individual pixel signals. The analysis in Section 6.3 also produced expected signal rms's. These are obviously highly biased quantities that apply only to the smaller signals, those that carry position information. We will therefore use them to weight our chisquare function. To avoid the complexity of correlations between projected pixel neighbors, we choose to define a simple diagonal chisquare function. We don't expect it to be correctly normalized even for clusters with truncated pixel charges, but we can normalize our goodness-of-fit criterion to account for our ignorance. This is a somewhat academic discussion because the performance of the algorithm is quite insensitive to the weighting of projected pixel signals in the chisquare function.

Finally, we note that the template reconstruction algorithm makes the implicit assumption that there is prior knowledge of the track direction before the algorithm is invoked. The algorithm is therefore suitable for a second pass to refine the estimates of hit position and its uncertainty. This is not a major constraint because of the high granularity of the pixel detector and the relatively large spacing between pixel planes. Any multi-plane pixel-based reconstruction algorithm should be able to establish a sufficiently accurate knowledge of the track direction to achieve the full resolution of this algorithm.

6.4.2 Description of the Template Algorithm

The following is a description of the template algorithm. The reader should note that the template-based approach implicitly incorporates all of the relevant detector physics into the templates themselves. Lorentz drift manifests itself as an offset in the projected cluster shapes with respect to bin number. Non-uniformity of the Lorentz drift modifies the shapes of the templates. Charge loss and trapping makes them asymmetric. This implies that although the templates themselves depend upon which projection is being analyzed, the actual procedure does not depend upon projection. The following description applies to the general reconstruction of a pixel cluster. Not all steps are needed for the second-pass template processing. The differences between these cases are noted.

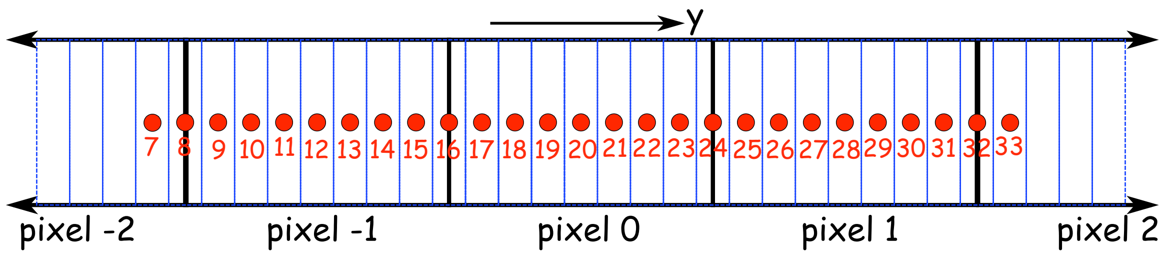
Preliminary Template Processing: The first step is to interpolate the templates and auxiliary information in $\cot\alpha$ and $\cot\beta$. Simple linear interpolation in $\cot\beta$ is used for all y-related quantities. The x-template is interpolated linearly in $\cot\alpha$ only whereas other x-related quantities are interpolated in both $\cot\alpha$ and $\cot\beta$. Parameterized quantities are not interpolated until after the entire function has been evaluated at each ($\cot\alpha$, $\cot\beta$) point. The interpolation step is unnecessary for the second-pass template processing because the requisite information was prepared during the first-pass processing.

The 9-bin templates in x and y are shifted by ± 1 and ± 2 pixels to span the 5 central pixels of the cluster for the possible locations of the x- and y-hit coordinates. This is illustrated in Fig. 15 for the unirradiated y-template shown in Fig. 10. The resulting templates now have 41 bins so that bins 4, 12, 20, 28, and 36 correspond to hit positions at the centers of pixels -2, -1, 0, 1, and 2, respectively. The templates are also padded with zeros to increase their lengths to 25 pixels in y and 11 pixels in x to match the size of the working buffers used to contain the cluster data.

Preliminary Cluster Processing: The total charge of the two-dimensional input cluster is calculated before the individual pixel charges are truncated to a maximum size given by the angle-interpolated value of S^y_{max}. After this truncation (also called "decapitation") step, the 1-d projections $P_i^{y/x}$ are calculated. These working buffers have lengths 11 in x and 25 in y to accommodate the following processing procedure. Any double pixels are expanded to occupy 2 adjacent elements in the projection arrays where each contains one half of the total double-pixel charge. The first and last pixels of the projections are identified and the clusters are shifted to center them in the projection arrays (the shifts $\mathrm{shift}_{y/x}$ are stored for later use). A set of double pixel flags is also shifted to track the locations of the expanded double pixels. These flags are then used to modify the interpolated templates by replacing the contents of the corresponding adjacent single pixels by their average value. The entire procedure of replacing a single double-pixel with a pair of half-signal single pixels has exactly the same pull in the final chisquare analysis as would have a single entry for a double pixel in the limit that the rms uncertainty on the pixel signals $\Delta P_i^{y/x}$ scale as $\sqrt{P_i^{y/x}}$. Note that the second-pass template processing assumes that all pixels are single size.

A key idea in the template-based algorithm is the recognition that there is important information in the absence of information. Since the readout chip is zero-suppressed, all pixels at the periphery of a cluster must have signals less than the readout threshold P_{min}. To force the fitting procedure to recognize this fact, the two pixels adjacent to either end of of the projected clusters are set equal to $P_{min}/2$ and they are assigned uncertainties $P_{min}/2$. These "pseudo-pixels" improve the resolution of the algorithm. The use of doubled pseudo-pixels helps to ensure that misaligned clusters and templates always have large values of chisquare even when the input clusters are small.

Initial Chisquare Minimization: The basic goal of the procedure is to translate the expected cluster shape until it best matches the observed cluster shape. This is shown in Fig. 16 where the y-projection of a cluster is shown as the set of magenta data points. We note that the signal at pixel +2 falls below the readout threshold and is replaced by a green pseudo-pixel. The basic cluster shape, encoded in 1/8 pixel bins is shown as the blue histogram. It is clear that translating the cluster to the left by 2/8 bins would produce much better agreement and suggests that true hit is likely to be at -2 bins in pixel 0. To allow for less than perfect signal height calibration, we allow the overall normalization of the template or of the data to float. This is accomplished by evaluating the following chisquare



Bin	Px-6	Px-5	Px-4	Px-3	Px-2	Px-1	Px 0	Px+1	Px+2	Px+3	Px+4	Px+5	Px+6
7	.0	.0	216.7	13396.4	13685.1	13544.1	10208.5	.0	.0	.0	.0	.0	.0
8	.0	.0	.0	11884.5	13587.8	13549.2	11913.7	.0	.0	.0	.0	.0	.0
9	.0	.0	.0	10198.0	13727.3	13592.2	13404.4	252.2	.0	.0	.0	.0	.0
10	.0	.0	.0	8512.6	13597.7	13559.3	13577.9	1688.0	.0	.0	.0	.0	.0
11	.0	.0	.0	6762.7	13607.2	13677.0	13601.3	3428.0	.0	.0	.0	.0	.0
12	.0	.0	.0	5165.2	13569.4	13603.1	13644.9	5039.3	.0	.0	.0	.0	.0
13	.0	.0	.0	3412.1	13718.7	13604.0	13630.6	6812.7	.0	.0	.0	.0	.0
14	.0	.0	.0	1703.1	13589.0	13566.5	13567.4	8556.2	.0	.0	.0	.0	.0
15	.0	.0	.0	216.7	13396.4	13685.1	13544.1	10208.5	.0	.0	.0	.0	.0
16	.0	.0	.0	.0	11884.5	13587.8	13549.2	11913.7	.0	.0	.0	.0	.0
17	.0	.0	.0	.0	10198.0	13727.3	13592.2	13404.4	252.2	.0	.0	.0	.0
18	.0	.0	.0	.0	8512.6	13597.7	13559.3	13577.9	1688.0	.0	.0	.0	.0
19	.0	.0	.0	.0	6762.7	13607.2	13677.0	13601.3	3428.0	.0	.0	.0	.0
20	.0	.0	.0	.0	5165.2	13569.4	13603.1	13644.9	5039.3	.0	.0	.0	.0
21	.0	.0	.0	.0	3412.1	13718.7	13604.0	13630.6	6812.7	.0	.0	.0	.0
22	.0	.0	.0	.0	1703.1	13589.0	13566.5	13567.4	8556.2	.0	.0	.0	.0
23	.0	.0	.0	.0	216.7	13396.4	13685.1	13544.1	10208.5	.0	.0	.0	.0
24	.0	.0	.0	.0	.0	11884.5	13587.8	13549.2	11913.7	.0	.0	.0	.0
25	.0	.0	.0	.0	.0	10198.0	13727.3	13592.2	13404.4	252.2	.0	.0	.0
26	.0	.0	.0	.0	.0	8512.6	13597.7	13559.3	13577.9	1688.0	.0	.0	.0
27	.0	.0	.0	.0	.0	6762.7	13607.2	13677.0	13601.3	3428.0	.0	.0	.0
28	.0	.0	.0	.0	.0	5165.2	13569.4	13603.1	13644.9	5039.3	.0	.0	.0
29	.0	.0	.0	.0	.0	3412.1	13718.7	13604.0	13630.6	6812.7	.0	.0	.0
30	.0	.0	.0	.0	.0	1703.1	13589.0	13566.5	13567.4	8556.2	.0	.0	.0
31	.0	.0	.0	.0	.0	216.7	13396.4	13685.1	13544.1	10208.5	.0	.0	.0
32	.0	.0	.0	.0	.0	.0	11884.5	13587.8	13549.2	11913.7	.0	.0	.0
33	.0	.0	.0	.0	.0	.0	10198.0	13727.3	13592.2	13404.4	252.2	.0	.0

Figure 15: The signal averages $S^y_{i,j}$ of 13 of the pixels in the y-projection of $\cot\beta = 1.97$ unirradiated barrel clusters for 27 of the 41 bins in the y-hit position after shifting the 9-bin template by ± 1 and ± 2 pixels.

function for some or all of the template bins,

$$\begin{aligned}
\chi^2(j) &= \sum_i \frac{(P_i^{y/x} - N_j S_{i,j}^{y/x})^2}{(\Delta P_i^{y/x})^2} \qquad (10)\\
N_j &= \sum_i \frac{P_i^{y/x}}{(\Delta P_i^{y/x})^2} \Bigg/ \sum_i \frac{S_{i,j}^{y/x}}{(\Delta P_i^{y/x})^2}
\end{aligned}$$

where the projected pixel uncertainties $\Delta P_i^{y/x}$ are calculated using equation 8 from the pre-stored parameters and interpolated in $\cot\alpha$ and $\cot\beta$ if appropriate (not needed in second-pass template generation). The actual χ^2 minimization search can be performed in several ways that trade-off speed for robustness. The slowest and most robust search evaluates equation 10 for each of the 41 bins and finds the absolute minimum. A faster and still secure alternative is to limit the search to the central 25 bins if there are no double-pixels at the ends of the projected cluster and to use the central 33 bins if there is an end double-pixel. Still faster but having slightly less than optimal resolution is to search every fourth bin for a minimum and then to expand the search just to the second nearest neighbors to find another minimum and then to the nearest neighbors of until a group of 3 consecutive bins has been evaluated and a minimum established. This minimization scheme is roughly four times faster than the slowest one but relies on the smooth parabolic shape of the χ^2 function. It works well for most clusters with single-size pixels but must be started at the finer step size for those with double-size pixels.

Position Estimation for Single Pixel Projections: The procedure described to this point is applied to all cluster projections and always results in the bin number and value of the chisquare minimum. For single pixel cluster projections, the chisquare value is stored and a simplified position estimation is performed. The reconstructed position of the hit, y_{rec} or x_{rec}, is determined by correcting the position given by the pixel center for the centering

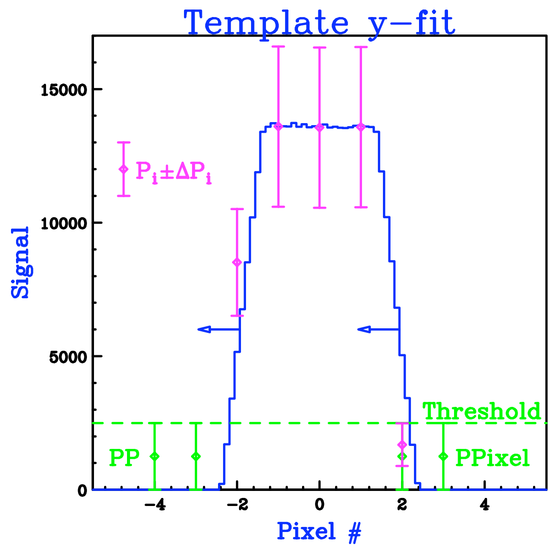


Figure 16: The y-projection of a cluster is shown as the set of magenta data points. The signal at pixel +2 falls below the readout threshold and is replaced by a green pseudo-pixel. The basic cluster shape, encoded in 1/8 pixel bins is shown as the blue histogram.

step ($\text{shift}_{y/x}$) and for the bias $D^{y/x}_{1/2}$ where the subscript indicates single and double pixels separately,

$$y_{rec} = y_{pix} - \text{shift}_y - D^y_k \tag{11}$$
$$x_{rec} = x_{pix} - \text{shift}_x - D^x_k. \tag{12}$$

The bias is determined from the average residual of all one-pixel clusters during the second-pass template generation. It is also calculated separately for single double-pixel clusters by merging adjacent rows and columns of the Pixelav events. This is done with two adjacent pixel pairings to span all possible situations. The bias calculation automatically corrects single pixel clusters for Lorentz-drift and for bias caused by radiation damage which can cause two-pixel clusters to become single pixel clusters. The same procedure is also used to calculate the rms spreads in hit residual for single pixel clusters, $\Delta^{y/x}_{1/2}$. These can differ significantly from the usual product of $(12)^{-1/2}$ and the pixel pitch because single pixel clusters often occur only in limited regions hit position depending upon incident track angles and Lorentz drift. The uncertainties on the reconstructed pixel hits, σ_y and σ_x, are taken to be $\Delta^y_{1/2}$ and $\Delta^x_{1/2}$, respectively, for one-pixel projections.

Note that the quantities $D^{y/x}_{1/2}$ and $\Delta^{y/x}_{1/2}$ are interpolated in $\cot\beta$ and $\cot\alpha$ in ordinary hit processing whereas they are actually generated during second-pass template processing.

Position Estimation for Multiple Pixel Projections: For multiple pixel projections, the bin number of the chisquare minimum is used to seed a two-bin interpolation calculation to refine the knowledge of the chisquare minimum in terms of a continuous parameter. This is done by defining the bins adjacent to the minimum bin as bins l and h as is shown in Fig. 17. The chisquare function is then redefined in terms of a linear combination of the functions $S^{y/x}_{i,l}$ and $S^{y/x}_{i,h}$. For simplicity, we drop the x/y superscripts from the quantities and then express the chisquare function for each projection as

$$\chi^2 = \sum_i \frac{\{P_i - N\left[(1-r)S_{i,l} + rS_{i,h}\right]\}^2}{\Delta P_i^2} \tag{13}$$
$$r = \frac{\sum_i P_i(S_{i,h}-S_{i,l})/\Delta P_i^2 \sum_i P_i S_{i,l}/\Delta P_i^2 - \sum_i P_i^2/\Delta P_i^2 \sum_i S_{i,l}(S_{i,h}-S_{i,l})/\Delta P_i^2}{\sum_i P_i^2/\Delta P_i^2 \sum_i (S_{i,h}-S_{i,l})^2/\Delta P_i^2 - [\sum_i P_i(S_{i,h}-S_{i,l})/\Delta P_i^2]^2} \tag{14}$$

where N is a common normalization factor and r is a dimensionless ratio that is bounded by 0 and 1 and determines the position of the χ^2 minimum between the centers of bins l and h. The resulting estimates of the hit position are given by the following expressions,

$$y_{rec} = y_{bin}[l] + r\,(y_{bin}[h] - y_{bin}[l]) - \text{shift}_y \tag{15}$$
$$x_{rec} = x_{bin}[l] + r\,(x_{bin}[h] - x_{bin}[l]) - \text{shift}_x \tag{16}$$

where $y_{bin}[i]$ and $x_{bin}[i]$ are the x and y positions of bin i. The templates corresponding to bins l and h for the example shown in Fig 16 are shown as the blue solid and red dashed histograms in Fig. 18. It is clear that a something close to an $r = 0.5$ combination of the templates will yield the best fit.

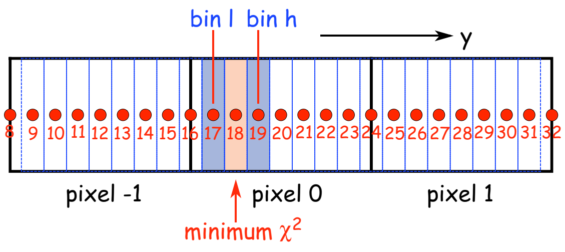


Figure 17: The definition of bins l and h for the example shown in Fig 16.

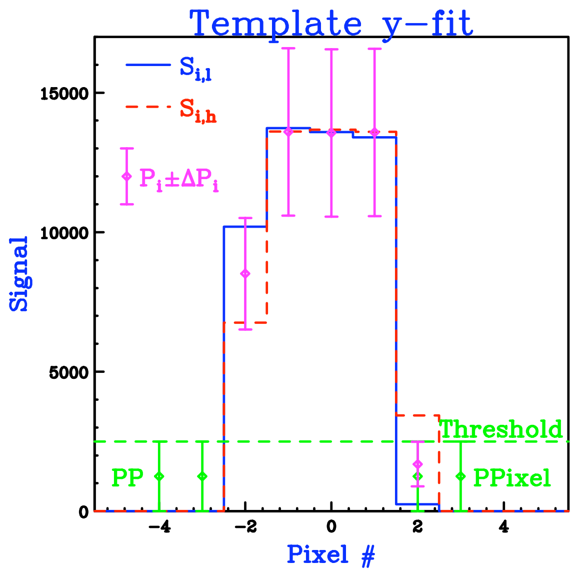


Figure 18: The templates corresponding to bins l and h for the example shown in Fig 16 are shown as the blue solid and red dashed histograms.

Residual Q_{FL} Corrections: There are a number of processes that can cause the observed topology of a cluster to differ from its "true" topology. Signals can fluctuate above and below the readout threshold or charge can migrate to neighboring pixels, or trapping can effectively shorten clusters. These effects are more likely when one of the edge pixels has a small charge. To study and correct for them (on average), the average residuals $D^y = y_{rec} - y_{hit}$ and $D^x = x_{rec} - x_{hit}$ measured during the second processing pass are plotted against the quantity $Q_{FL} = (P_F - P_L)/(P_F + P_L)$ constructed from the first and last pixel charges, P_F and P_L, of the projected cluster. This is done for y- and x-projections at each value of $(\cot\beta, \cot\alpha)$. Like most quantities produced in the second template generation pass, these are sliced into four bins of Q/Q_{avg}: $Q/Q_{avg} > 1.5$, $1.5 > Q/Q_{avg} > 1$, $1 > Q/Q_{avg} > 0.85$, and $0.85 > Q/Q_{avg}$. These ranges are chosen to contain roughly 30% of the sample in each of the three lower bins and ~5% of the sample in the largest bin (contains mostly delta rays). As an example, the distributions of D^x versus Q_{FL} for $\cot\beta = 1.97$ and $1 > Q/Q_{avg} > 0.85$ are shown in Fig. 19 for new and heavily irradiated sensors. Note that the corrections for the unirradiated sensor are approximately odd in Q_{FL} and are small except at large values of $|Q_{FL}|$ where they approach 50% of the resolution. Since they contribute in quadrature to the resolution, the Q_{FL} corrections do not significantly improve the resolution of the unirradiated sensors. The corrections for the irradiated sensor are quite different: they are generally larger even have the opposite sign for the same Q_{FL}! During the second pass, these corrections are parameterized by fifth-order polynomials for each set of track angles and charge bin (see Fig. 19 for an example),

$$D^{y/x}_{FL}(Q_{FL}) = d_0 + d_1 Q_{FL} + d_2 Q^2_{FL} + d_3 Q^3_{FL} + d_4 Q^4_{FL} + d_5 Q^5_{FL} \tag{17}$$

where d_0-d_5 are stored for each set of angles and each of 4 charge bins. During the second-pass template generation, these are applied to a "third pass" that stores the $Q^{y/x}_{FL}$ values and results of equations 15 and 16 from the second pass.

In actual operation, the corrections for the observed charge bin are interpolated in track angle and then applied to the reconstructed hit positions estimated from equations 15 and 16,

$$y_{rec} = y_{bin}[l] + r\,(y_{bin}[h] - y_{bin}[l]) - \text{shift}_y - \bar{D}^y_{FL}(Q_{FL}, \cot\beta, Q) \tag{18}$$
$$x_{rec} = x_{bin}[l] + r\,(x_{bin}[h] - x_{bin}[l]) - \text{shift}_x - \bar{D}^x_{FL}(Q_{FL}, \cot\alpha, \cot\beta, Q), \tag{19}$$

where the quantities $\bar{D}^{y/x}_{FL}$ are interpolated over track angles.

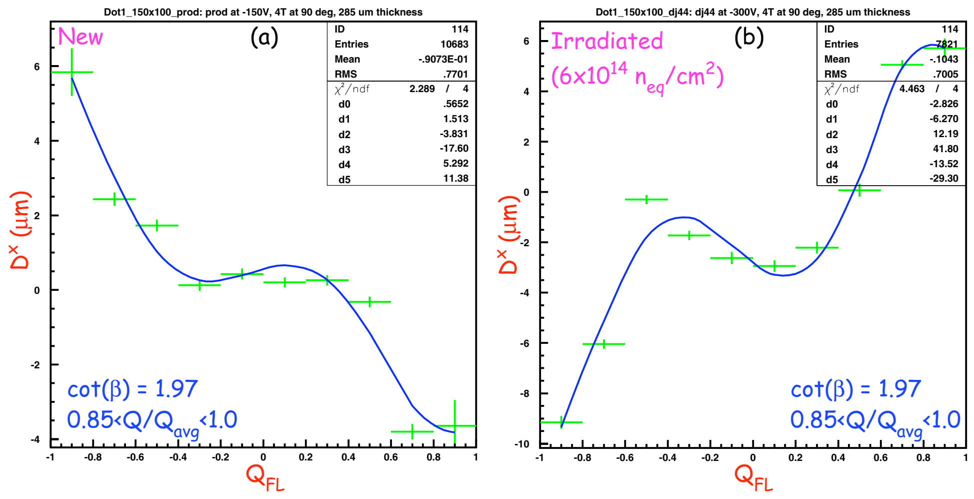


Figure 19: The average residuals $D^x = x_{rec} - x_{hit}$ are shown versus Q_{FL} for unirradiated (a) and irradiated (b) samples of ($\cot\alpha$, $\cot\beta$)=(0, 1.97) clusters in the $1 > Q/Q_{avg} > 0.85$ charge bin. The results of fits to fifth-order polynomials are also shown as solid blue lines.

Final Error and Bias Estimates: The second template pass also generates and stores final bias and resolution information for each set of track angles and charge bin. The residuals from the application of equation 17, $D^{y/x}_{final}$, are accumulated for each Q bin and final Gaussian fits are performed as shown in Fig. 20 for the D^x_{final} distributions of unirradiated (a) and irradiated (b) samples of ($\cot\alpha$, $\cot\beta$)=(0, 1.97) clusters in the $1 > Q/Q_{avg} > 0.85$ charge bin. The mean, rms, Gaussian center, and Gaussian sigma are stored for each set of angles and charge bin. In actual operation, the means are interpolated in the track angles and are used to correct the final position estimates

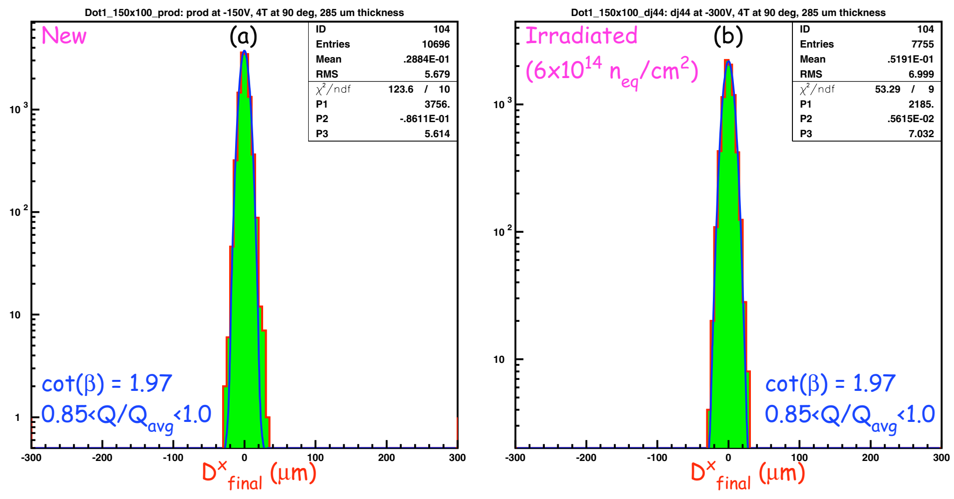


Figure 20: The residuals D^x_{final} are shown for unirradiated (a) and irradiated (b) samples of ($\cot\alpha$, $\cot\beta$)=(0, 1.97) clusters in the $1 > Q/Q_{avg} > 0.85$ charge bin.

given by equations 18 and 19. The interpolated rms widths are used to estimate the uncertainties of the position estimates. This choice includes any non-Gaussian tails that may be present and represents a better estimate of the true resolution than the Gaussian fit parameters.

Chisquare Probabilities: The second-pass of the template generation stores the averages of the minimum y- and x-chiqsquare functions for later use in the calculation of goodness-of-fit probabilities. This single parameter is adequate to reproduce the actual distributions as is shown in Fig. 21 for the y-chisquare distribution. The histograms show the actual distributions in the $1 > Q/Q_{avg} > 0.85$ bin. The solid blue curves show the differential chisquare function for the observed mean value. While not perfect, it is clear that the single parameter is adequate to describe the differential distribution and by extension, it should be adequate to describe the integrated probability distribution.

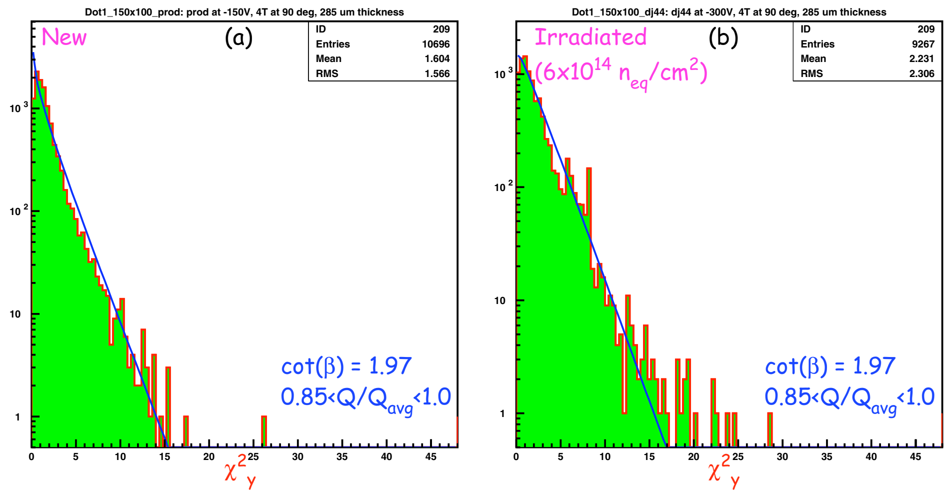


Figure 21: The y-chisquare χ^2_y distribution for unirradiated (a) and irradiated (b) samples of ($\cot\alpha$, $\cot\beta$)=(0, 1.97) clusters in the $1 > Q/Q_{avg} > 0.85$ charge bin. The theoretical differential distribution having the correct mean value is plotted as the solid blue curves.

In actual operation, the chisquare minima are converted to probabilities to test the projected cluster shapes against the a-priori expectations provided by the templates.

Template Information: The second-pass of the template generation produces a considerable quantity of information. Except for the single-pixel projection information, all of it is stored in the four bins of Q/Q_{avg}: $Q/Q_{avg} > 1.5$, $1.5 > Q/Q_{avg} > 1$, $1 > Q/Q_{avg} > 0.85$, and $0.85 > Q/Q_{avg}$. A final step in the second pass writes each ($\cot\alpha$, $\cot\beta$) entry into an ascii summary file. Although this discussion has been careful to consistently use the local pixel coordinate system defined in Section 6.2, the Pixelav uses a different coordinate system. These are shown in Fig. 22. The information is transformed from Pixelav coordinates to the local pixel coordinates in the final step. The total information generated in the two passes for each set of track angles constitutes 301 4-byte integer and floating point words. An additional 20 4-byte words is reserved for possible growth, yielding a single track angle entry size of 331 words. Since each full template constitutes 116 separate ($\cot\alpha$, $\cot\beta$) points, the total size of a template is 153.6 kbytes. The ascii summary file is somewhat larger and requires 448 kB of storage although it does compress to 88 kB after processing with gzip. In the future, it will be stored in a database.

6.5 Performance

Unfortunately, very limited beam test data are available for the final pixel geometry and readout chip. All of the available data were collected at normal incidence or near normal incidence so that all clusters have one or two pixel sizes. Therefore, the characterization of the template algorithm depends almost entirely upon simulated data. The native performance of template algorithm was studied by reconstructing large samples clusters generated by Pixelav with random positions and track angles. This work is summarized in Section 6.5.1. There are a number of effects that are more easily studied with the full CMSSW simulation than with the standalone code. These include the effects of double-size pixels, the effects of detector edges, and the effects secondary particle production upstream the pixel sensors. These are discussed in Section 6.5.2. The production of secondary particles yields cluster shapes that are inconsistent with the track angles and reconstructed hit coordinates that are not well correlated with the position of the primary track. These can be suppressed by the use of the goodness-of-fit information generated by the template algorithm. This is discussed in Section 6.5.3. Finally, the impact of the template reconstruction on CMSSW tracking is discussed in Section 6.5.4. The reader should note that all of the

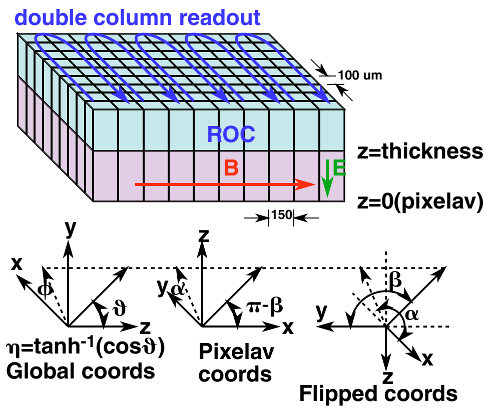


Figure 22: The definitions of the flipped-barrel local coordinate system and the Pixelav coordinate system.

resolution plots shown in Section 6.5 are root-mean-square (rms) quantities and include the effects of tails. The exception occurs in Section 6.5.4 which reports the Gaussian sigma results generated by the tracking validation process.

6.5.1 Native Performance

The performance of the template algorithm is compared with that of the standard algorithm by plotting the rms y- and x-residuals for a sample of reconstructed barrel clusters generated by Pixelav as shown in Fig. 23. The residuals are plotted as functions of pseudorapidity for the two cluster charge bands $1.5 > Q/Q_{avg} > 1$ (30% of all clusters) and $1 > Q/Q_{avg}$ (70% of all clusters). The clusters were simulated for an unirradiated physical sensor (includes focusing effects near the n+ implants) operated at 150V bias. The rms residuals are used to measure the effects of non-Gaussian tails on the performance of the algorithms. Note that the template and standard algorithms perform similarly in the lower charge band which has less delta-ray activity. Near $\eta = 0$, the projected y-clusters consist of single pixels and have poor resolution. Near $\eta = 0.5$, the y-projections consist of two-pixel clusters and the y-resolution is quite good. It then worsens at larger η where the template algorithm has approximately 10% better resolution. The x-resolutions for the lower charge band improve with increasing η (and increasing Q). The algorithms perform comparably at low η and diverge a bit at large η where the template resolution is about 20% better than the standard resolution. In the larger charge band where there is increased delta-ray activity, the template algorithm has significant advantages over the standard algorithm at nearly all pseudorapidities.

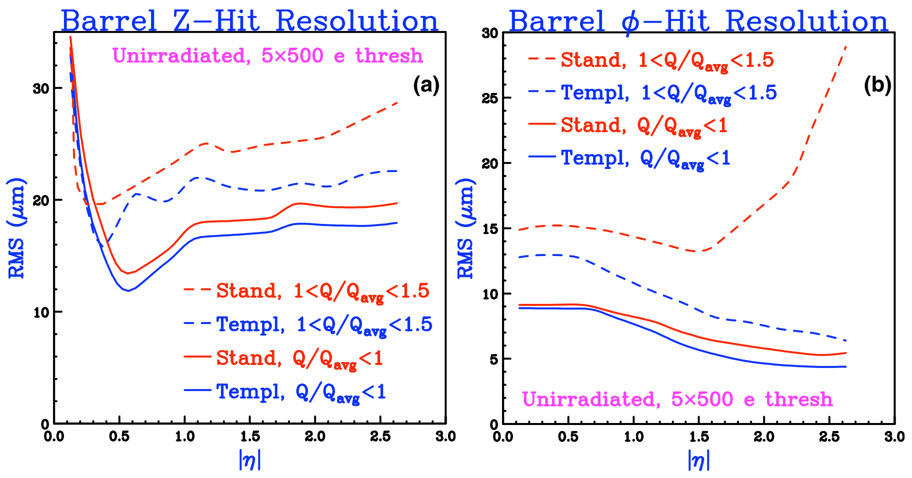


Figure 23: The rms y-residuals (a) and x-residuals (b) of a sample of reconstructed barrel clusters for the template (blue) and standard (red) algorithms are plotted versus pesudorapidity for the cluster charge bands $1.5 > Q/Q_{avg} > 1$ (dashed lines) and $1 > Q/Q_{avg}$ (solid lines). The event sample is generated by Pixelav and models an unirradiated detector operated at 150V bias.

The two algorithms were also compared using a Pixelav-generated sample of clusters from a heavily irradiated physical sensor ($\Phi = 6 \times 10^{14}$ $\mathrm{n_{eq}/cm^2}$) operated at 300V bias. A calibrated template is used to reconstruct these events. The Lorentz-shift used by the standard algorithm is reduced from 121 μm to 75.3 μm to account for the higher operating bias and the loss of charge sharing caused by trapping. The resulting rms residuals are plotted versus η in Fig. 24 for the cluster charge bands $1.5 > Q/Q_{avg} > 1$ and $1 > Q/Q_{avg}$. We note that the resolutions of both algorithms are degraded, but template algorithm is less affected (as it was designed to be). In particular, the standard algorithm develops large η-dependent bias in the y-direction after irradiation which is reflected in the significant degradation of the y-resolution. The template algorithm has a much smaller intrinsic bias that is automatically corrected.

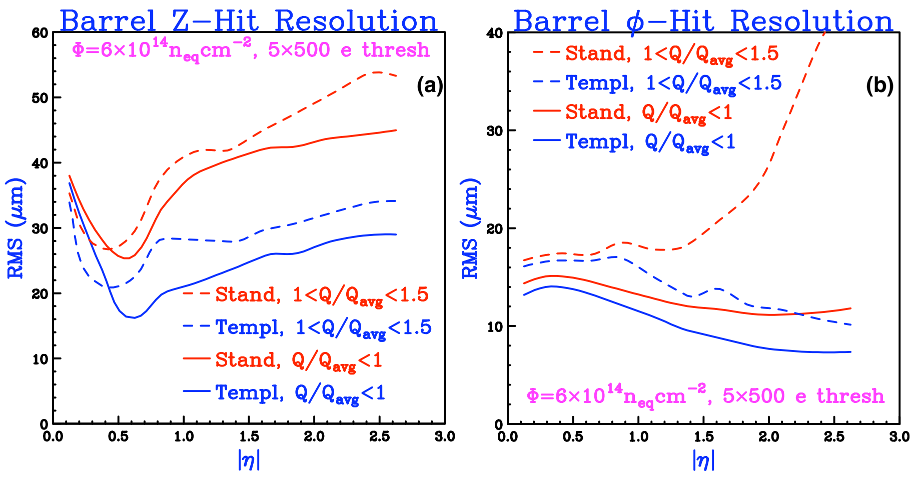


Figure 24: The rms y-residuals (a) and x-residuals (b) of a sample of reconstructed barrel clusters for the template (blue) and standard (red) algorithms are plotted versus pesudorapidity for the cluster charge bands $1.5 > Q/Q_{avg} > 1$ (dashed lines) and $1 > Q/Q_{avg}$ (solid lines). The event sample is generated by Pixelav and models a detector with significant radiation-damage ($\Phi = 6 \times 10^{14}$ $\mathrm{n_{eq}/cm^2}$) operated at 300V bias.

6.5.2 Performance in CMSSW

The template algorithm was also tested using samples of events with six 20 GeV muons generated by the CMSSW simulation. Special templates corresponding to the simpler sensor physics in the CMSSW simulation were generated using a highly simplified electric field map that is uniform and does not include focusing effects near the n+ implants. The simplified templates and event samples predict performance that is similar to the more physical ones as shown in Fig. 25. The predicted rms resolutions are shown for the two cases. Note that even the “crossed” cases of physical templates analyzing simplified events or vice-versa predict the same resolutions. These distributions justify the simplifications used in the CMSSW simulation to model overdepleted, unirradiated sensors. The templates corresponding to the simplified Pixelav modeling were then used to reconstruct clusters generated by the CMSSW simulation. A comparison of the performance of the template algorithm operating on CMSSW clusters and simplified Pixelav clusters is shown in Fig. 26. The rms resolutions are shown in the two charge bands. We note that they quite similar and that the Pixelav-calibrated template procedure produces the expected results when applied to CMSSW-generated clusters.

Although the CMSSW simulation has a simplified model of the sensor physics, it does correctly model the geometry of the entire tracking system and the spatial distribution of vertices. The resolution of the pixel tracking system is affected by the presence of double-size pixels at readout chip boundaries and by the sensor edges. Additionally, the CMSSW simulation includes showering by the primary charged particles as they transit the detector. The secondary charge particles produced by the showering are visible as low charge clusters for larger pseudorapidities in the barrel and present a potential background for the tracking. These effects can be quantified by comparing the rms template resolutions for a sample of CMSSW-generated barrel clusters before (dashed red lines) and after (solid blue lines) the removal of low-charge clusters and clusters containing double pixels and edge pixels as shown in Fig. 27. Note that additional effects produce on a small increase in the rms residuals for the charge band $1.5 > Q/Q_{avg} > 1$ but significantly affect the rms residuals in the lower charge band $1 > Q/Q_{avg}$. This effect is directly traceable to low-charge clusters produced by secondary charged particles. These clusters have shapes that differ from those produced by the primary charged particles and also have small charges that become distinguishable at large pseudorapidity where the primary clusters become large.

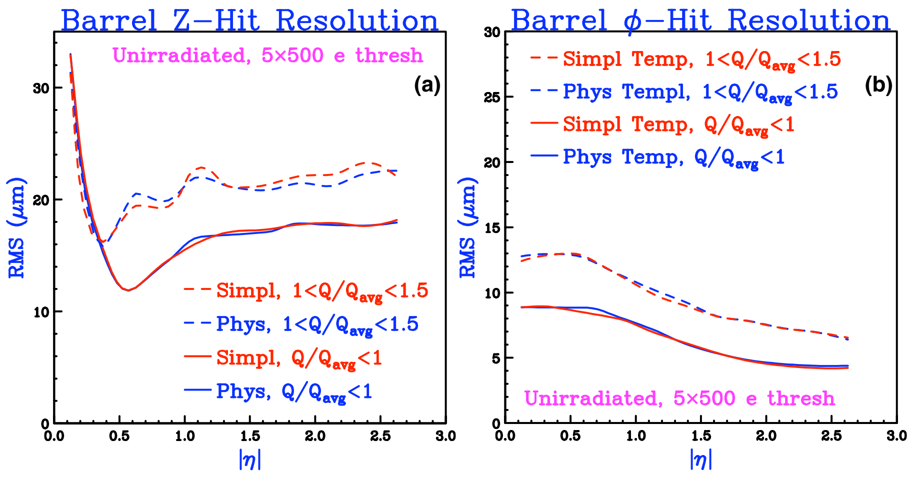


Figure 25: The rms y-residuals (a) and x-residuals (b) of a sample of template-reconstructed barrel clusters for the physical (blue) and simplified (red) electric field maps are plotted versus pesudorapidity for the cluster charge bands $1.5 > Q/Q_{avg} > 1$ (dashed lines) and $1 > Q/Q_{avg}$ (solid lines). Both event samples are generated by Pixelav and model an unirradiated detector operated at 150V bias.

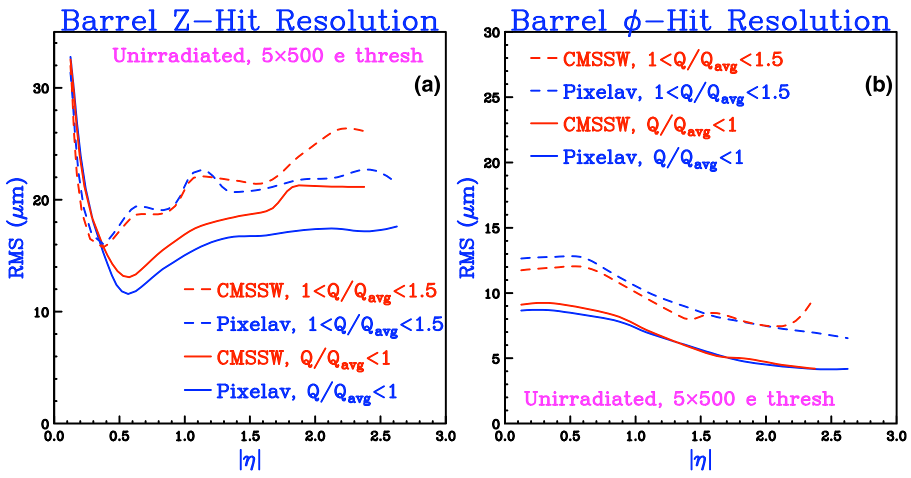


Figure 26: The rms y-residuals (a) and x-residuals (b) of Pixelav-generated (blue) and CMSSW-generated (red) samples of template-reconstructed barrel clusters are plotted versus pesudorapidity for the cluster charge bands $1.5 > Q/Q_{avg} > 1$ (dashed lines) and $1 > Q/Q_{avg}$ (solid lines). Both samples model an unirradiated detector operated at 150V bias.

This problem is illustrated in Fig. 28 which shows the CMSSW-generated cluster charge distribution for 10 0.25-slices of pseudorapidity. The charge distributions for primary muons are shown in red and black and the charge distribution for secondary electrons is shown in magenta. Edge clusters appear as muons with low cluster charge and become more prominent in the larger-η bins. The secondary electrons are present in all slices but are particularly pronounced at large η where they comprise nearly 20% of all hits. The presence of reconstructed secondary clusters on tracks is somewhat smaller but still comprises approximately 6% of hits in the largest η slice. To help suppress the secondary clusters, the template code stores a minimum charge Q_{min} for each $\cot\beta$ entry. If the cluster charge if found to be less than Q_{min}, the qbin flag returned from PixelTempReco2D is set to the value 4. This allows the rejection or flagging of high-η, low-charge clusters with no loss of efficiency but does not suppress the low-η secondaries. A more powerful discriminant based upon template probabilities is discussed in Section 6.5.3.

A comparison of the template and standard algorithms operating on a sample of CMSSW-generated clusters with all effects is shown in Fig. 29. The rms resolutions in the two charge bands are shown in the standard algorithm (red lines), the template algorithm (blue lines), and the template algorithm after the low charge clusters have been removed with a simple charge cut (green lines). Note that the template algorithm still outperforms the standard one with all effects present and improves further when the low charge clusters are removed.

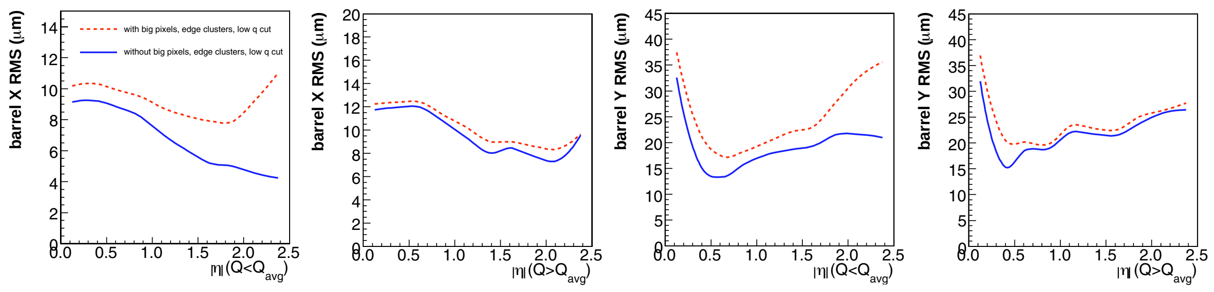


Figure 27: The rms y-residuals and x-residuals of a CMSSW-generated sample of template-reconstructed barrel clusters are plotted versus pseudorapidity for the cluster charge bands $1.5 > Q/Q_{avg} > 1$ and $1 > Q/Q_{avg}$. The dashed red curves represent the entire sample whereas the solid blue curves represent a sub-sample that does not contain the low-q clusters from secondary charged particles or clusters with double-pixels or edge pixels.

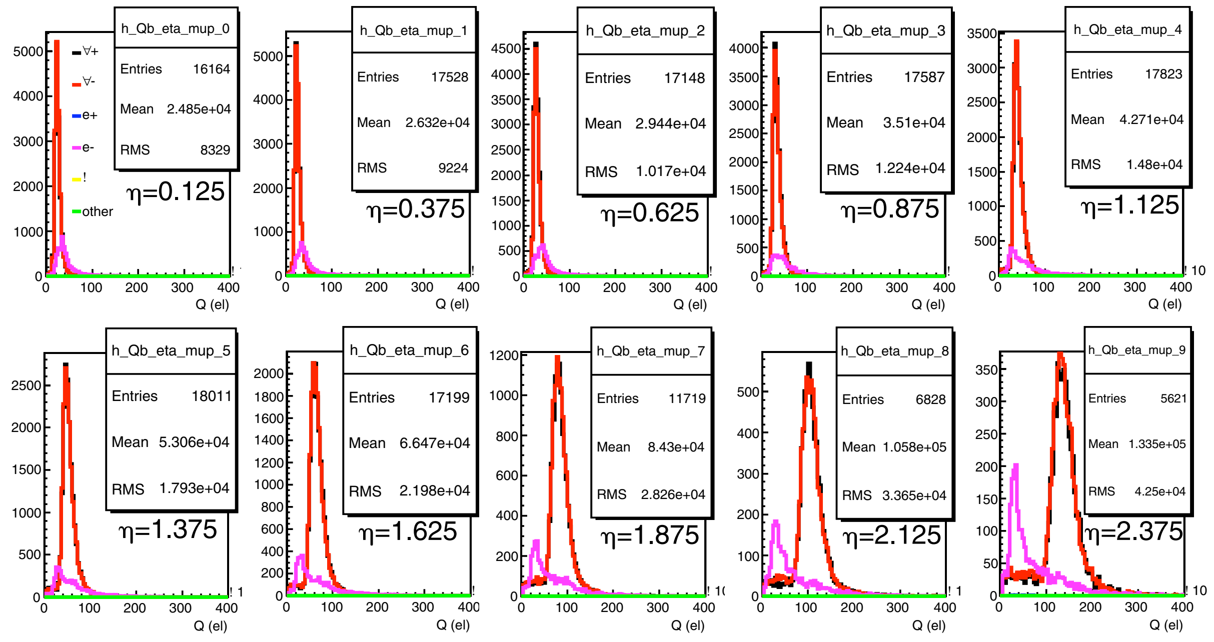


Figure 28: The charge distribution of clusters (not necessarily track associated) in 10 slices of η for primary muons (red/black) and secondary electrons (magenta) from a sample of CMSSW-generated muon events.

The performance of the template reconstruction in the forward pixel system with CMSSW-simulated clusters is shown in Fig. 30. The dashed red lines show the the rms template resolutions for all clusters and the solid blue lines show the same quantities after the removal of clusters containing double pixels and edge pixels. Note that the additional effects produce only a small increase in the rms x-residuals but significantly affect the rms y-residuals. It is not possible to remove clusters produced by secondary particles because tracks transiting the forward detectors have nearly normal incidence angles and produce small y-clusters and low cluster charge. The presence of double-size pixels moves some clusters from the well-resolved two pixel size into the very poorly resolved one double pixel size worsening the rms y-resolution. Although the Lorentz-drift is suppressed by a factor of $\sin 20° = 0.34$ and the x-projections of the clusters are smaller than similar $\cot\alpha$ tracks would produce in the barrel, they are generally larger than the y-projections and are less affected by the presence of double-size pixels.

A comparison of the template and standard algorithms operating on a sample of CMSSW-generated clusters with all effects is shown in Fig. 31. The rms resolutions in the two charge bands are shown for: the standard algorithm (red lines), the template algorithm (blue lines), and the template algorithm after the low charge clusters have been removed (green lines). Note that the template algorithm still outperforms the standard one in the x-projection with all effects present. There is essentially no difference between the algorithms in the y-reconstruction.

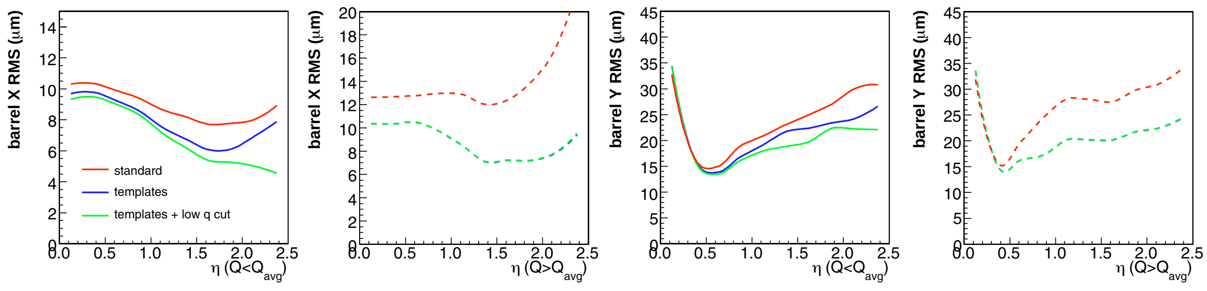


Figure 29: The rms y-residuals and x-residuals of a CMSSW-generated sample of standard- and template-reconstructed barrel clusters are plotted versus pseudorapidity for the cluster charge bands $1.5 > Q/Q_{avg} > 1$ and $1 > Q/Q_{avg}$. The red curves show the standard algorithm, the blue curves show the template algorithm and the green curves show the template algorithm after the removal of the low charge clusters. Note that the green and blue curves are coincident for the $1.5 > Q/Q_{avg} > 1$ band.

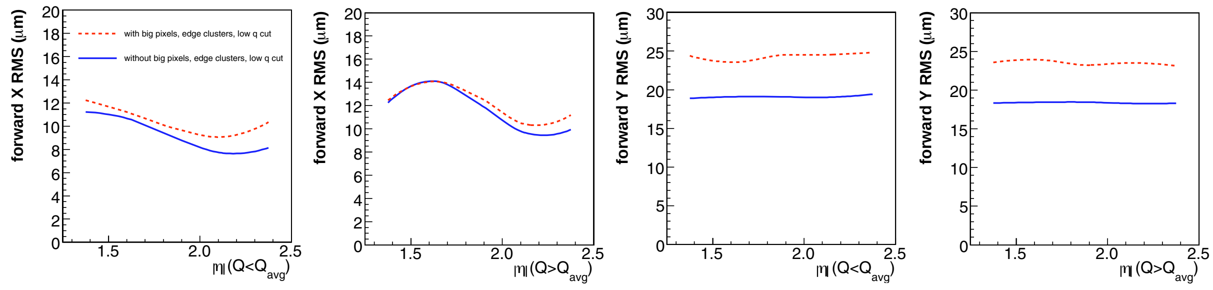


Figure 30: The rms y-residuals and x-residuals of a CMSSW-generated sample of template-reconstructed forward pixel clusters are plotted versus pseudorapidity for the cluster charge bands $1.5 > Q/Q_{avg} > 1$ and $1 > Q/Q_{avg}$. The dashed red curves represent the entire sample whereas the solid blue curves represent a sub-sample that does not contain clusters with double-pixels or edge pixels.

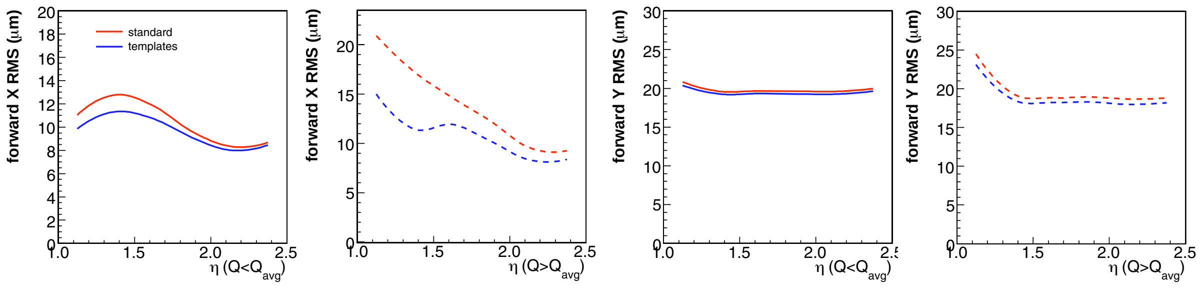


Figure 31: The rms y-residuals and x-residuals of a CMSSW-generated sample of standard- and template-reconstructed forward clusters are plotted versus pseudorapidity for the cluster charge bands $1.5 > Q/Q_{avg} > 1$ and $1 > Q/Q_{avg}$. The red curves show the standard algorithm and the blue curves show the template algorithm.

6.5.3 Cluster Shape Information

The template algorithm was designed to use the a-priori cluster shape information generated by Pixelav to optimize the resolution of the hit reconstruction. As part of the process, it minimizes the chisquare function defined by equations 10 and 13. This also provides information that can be used to test the compatibility of the observed cluster shapes with the shapes expected for the input track angles. In order to interpret the minimum chisquare information, the simple charge-bin-dependent one-parameter description discussed in Section 6.4.2 and illustrated in Fig. 21 was implemented. These are interpolated in the track angles and are applied to the problem in the usual way to estimate the chisquare tail probabilities in each projection,

$$\text{Prob}_{y/x} = 1 - \Gamma(\bar{\chi}^2_{y/x}/2, \chi^2_{y/x}/2) \tag{20}$$

where: Γ is the incomplete Gamma function, $\bar{\chi}^2_{y/x}$ is the expected average of the distribution, and $\chi^2_{y/x}$ is the minimum determined from the template algorithm. This description is not perfect and the resulting y- and x-probability distributions shown in Fig. 32 for a sample of Pixelav-generated clusters are not uniform (note that the peaks in the y-probability distribution are caused by pseudo-pixel contribution to single pixel clusters and an arbitrary normalization choice). Nevertheless, the probabilities can be used to test the consistency of the observed clusters with the expected template shapes.

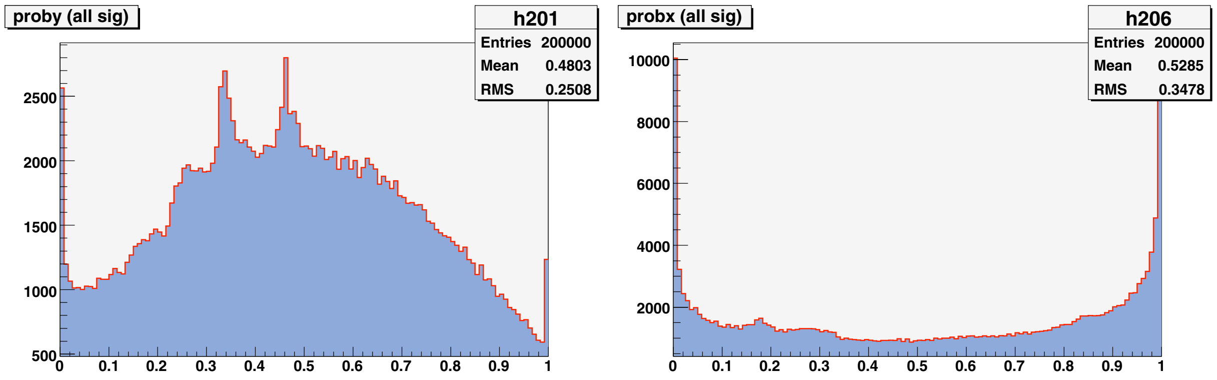


Figure 32: The y- and x-probabilities for a large sample of Pixelav-generated hits.

There are two distinct uses for the goodness-of-fit information. The first is to validate that the cluster is likely to have been produced by the transit of primary charged particle. It was shown in Section 6.5.2 that some of the clusters observed in the CMSSW simulation are produced by secondary showers. These can have the wrong cluster charge or shape and should not be included in reconstructed tracks. The template probabilities are a useful tool to reject these. The second application is to test the compatibility of the cluster with the track angle hypotheses. In principle, this could be a powerful constraint in track seeding and is discussed in Section 6.8. In either use case, it is essential to understand the inefficiencies of y- or x-probability requirements. These are estimated from a large sample of Pixelav clusters and are plotted as functions of the base 10 logarithm of the minimum probability in Fig. 33. The inefficiencies arising from the clusters with large delta ray activity in the largest charge band, $Q/Q_{avg} > 1.5$, are shown as dashed curves. Since these events comprise only 4.5% of the entire sample, it is clear that poorly measured events with large delta rays are disproportionately removed by reasonable values of the minimum probabilities. Furthermore, as one might expect given the factorization of the y- and x-projections, the inefficiencies associated with y- and x-probabilities are largely independent. The total inefficiency of separate y-probability and x-probability requirements is quite close to the sum of the two functions.

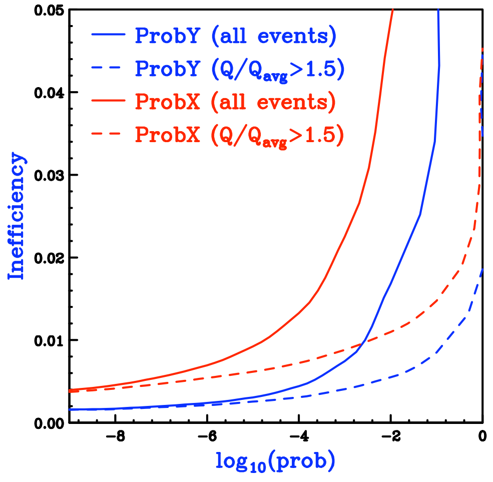


Figure 33: The inefficiencies of y- (blue) and x-probability (red) are shown as functions of the base 10 logarithm of the minimum probability as solid lines. The inefficiencies arising from the loss of poorly measured clusters in the largest charge band, $Q/Q_{avg} > 1.5$, are shown as dashed curves.

The utility of a minimum probability requirement is illustrated in Fig. 34 which shows the same CMSSW-generated cluster charge distributions that were shown in Fig. 28 after the application of the requirements $\mathrm{Prob}_y > 10^{-3}$, $\mathrm{Prob}_x > 10^{-3}$. We note that the secondary electron induced clusters are eliminated at large η and are suppressed at smaller η. The low charge distribution of edge clusters is also removed. The particular probability requirements shown in Fig. 34 are not optimized. It is clear that one would like to remove hits due to secondaries at the earliest stage of track finding. This suggests that the probabilities should be used in track seed finding and that any probability cuts must be optimized for that purpose. This is discussed in Section 6.8.

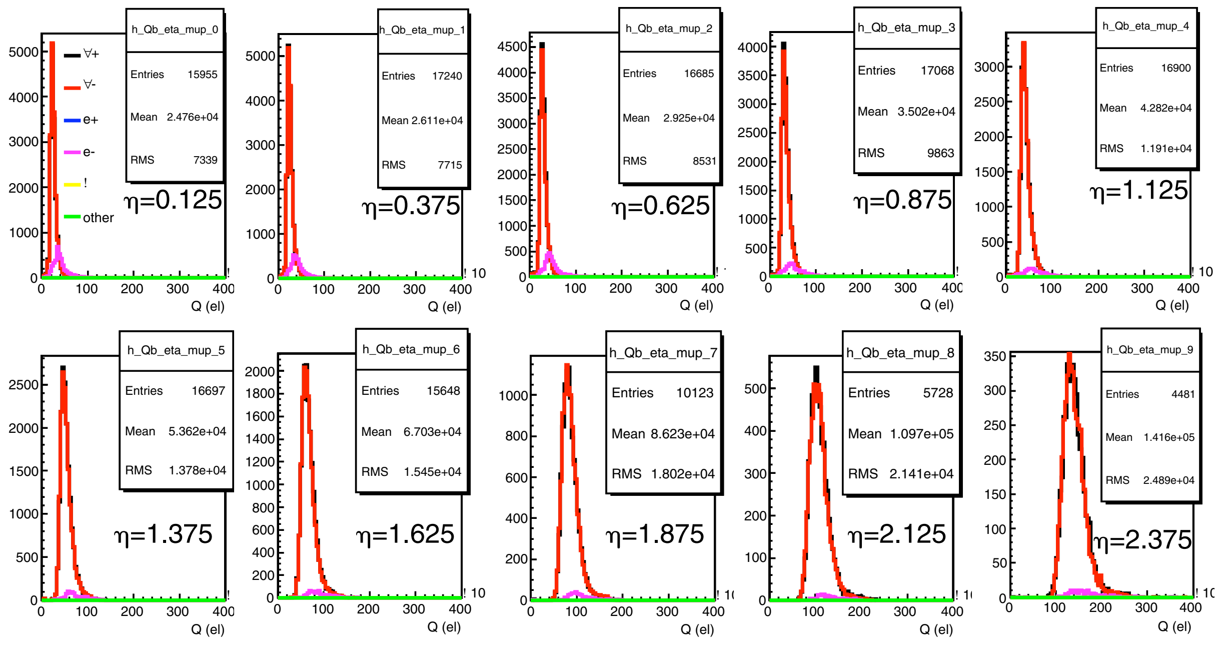


Figure 34: The charge distribution of clusters (not necessarily track associated) in 10 slices of η for primary muons (red/black) and secondary electrons (magenta) from a sample of CMSSW-generated muon events after the application of minimum y-probability and x-probability requirements of 10^{-3}.

6.5.4 Impact on Tracking

The impact of the second-pass template reconstruction upon the final track parameters was studied using a sample of 14,600 simulated 10 GeV muon tracks and version 1_6_0 of the tracking-validation code. The inclusion of templates does not affect the efficiency of the Kalman Filter tracking and the mean number of hits as is shown in Fig. 35 where the template (blue) and standard (red) algorithm results are shown in red as functions of pseudorapidity. The inclusion of templates does improve the quality of the track fitting at all pseudorapidities as is shown in Fig. 36(a) where the track fit chisquare is plotted as a function of η for the template reconstruction (blue) and the standard reconstruction (red). The improvement in track quality also manifests itself in the improved resolution in a number of track parameters. Note that the tracking-validation code characterizes the track resolution with Gaussian fits to residual distributions. The Gaussian resolution of the d_0 parameter, the transverse distance of closest approach to the beam axis, is shown as a function of η in Fig. 36(b) for the template reconstruction (blue) and the standard reconstruction (red). Note that the improvement is significant especially at large η. The Gaussian resolutions of z_0, the longitudinal displacement of the point of closest approach, and $\cot\theta$, the track polar direction at the point of closest approach are plotted as functions of η in Fig. 37 for the template reconstruction (blue) and the standard reconstruction (red). The Gaussian resolutions of the p_T and ϕ parameters are plotted as functions of η in Fig. 38 for the template reconstruction (blue) and the standard reconstruction (red). It is clear that the Gaussian resolution of all parameters is improved by the use of the second-pass template reconstruction especially at large η.

It is also instructive to examine the effect of the template algorithm on the rms pulls for each of the five track parameters. Unlike the Gaussian resolutions, they are sensitive to the presence of tails in the residual distributions and they also test the accuracy of the error estimates returned by the reconstruction algorithms. The rms pulls are listed in Table 2 for the two algorithms. It is clear that the template algorithm does somewhat better for d_0 and ϕ. It is likely that the rms resolutions of these quantities are significantly improved by the template algorithm. It is quite

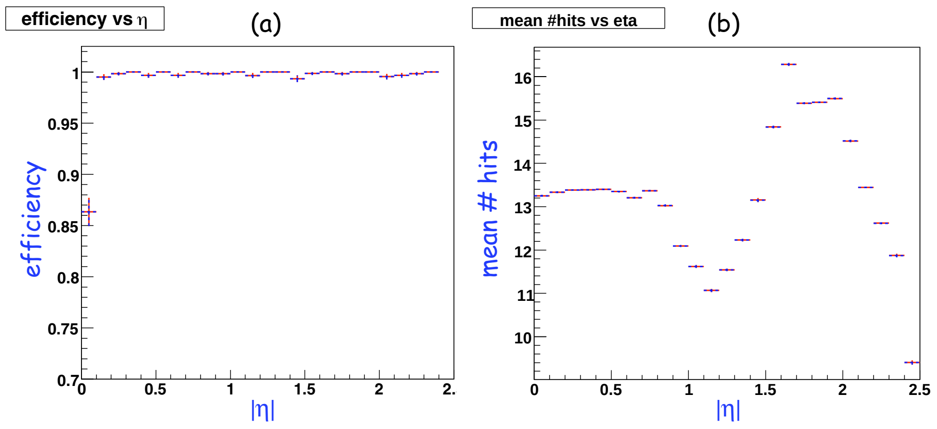


Figure 35: The tracking efficiency (a) and the mean number of hits (b) are plotted versus pseudorapidity for second-pass pixel hit reconstruction using the template algorithm (blue) and the standard algorithm (red). Note that the blue and red points overlap completely.

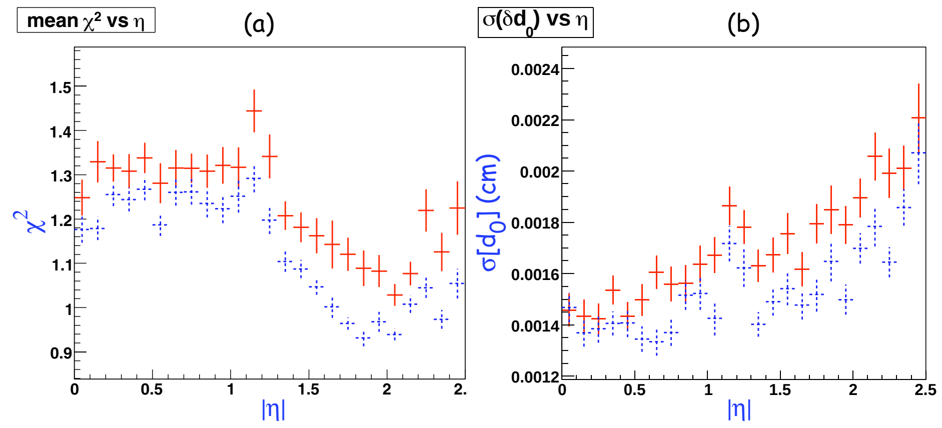


Figure 36: The chisquare (a) and Gaussian d_0 resolution (b) are plotted as functions of η for the template reconstruction (blue) and the standard reconstruction (red).

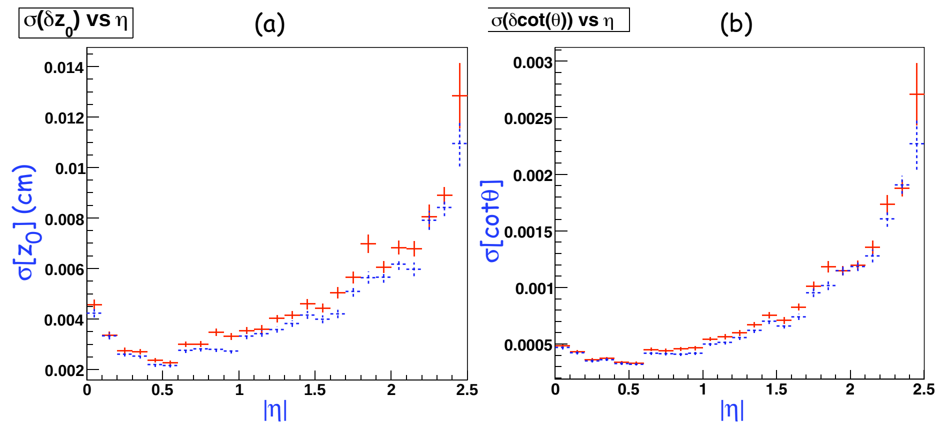


Figure 37: The Gaussian resolutions of the z_0 (a) and $\cot\theta$ (b) parameters are plotted as functions of η for the template reconstruction (blue) and the standard reconstruction (red).

possible that these improvements will manifest themselves in the form of improved tagging efficiency/purity for b-quarks and τ-leptons. Finally, we note that the use of the template algorithm in pixel seeding (see Section 6.8) would avoid the association of secondary electrons with tracks and might improve the track parameter resolution even more than has been shown in this section. Additional study is needed to determine if this is the case.

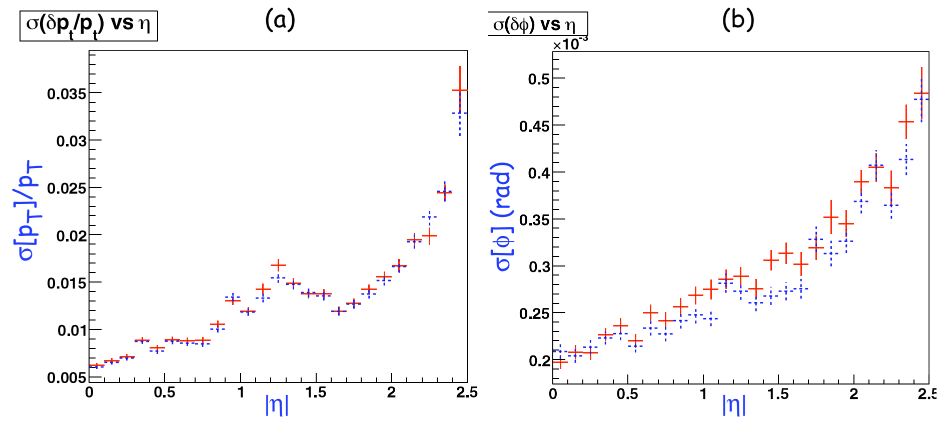


Figure 38: The Gaussian resolutions of the p_T (a) and ϕ (b) parameters are plotted as functions of η for the template reconstruction (blue) and the standard reconstruction (red).

Table 2: RMS track parameter pulls for a sample of simulated 10 GeV muon tracks.

Parameter	Standard Reco	Template Reco
d_0	1.81	1.16
z_0	1.65	1.35
$\cot\theta$	1.43	1.27
p_T	1.31	1.26
ϕ	1.54	1.16

6.6 Tuning and Calibration

The template algorithm is based upon the premise that the Pixelav simulation can be tuned to accurately describe the pixel sensors as they are gradually damaged by exposure to the large LHC radiation field. This premise rests upon the demonstrated success in the modeling of charge collection profiles measured with test sensors irradiated to several fluences [11]. The parameters of the double-junction model were tuned by hand until the simulation reproduced the profiles measured at the fluence $\Phi_0 = 6 \times 10^{14}\mathrm{n_{eq}/cm^2}$ as is shown in Fig. 6. Although this was extremely tedious, it was also shown that the parameters of the model could then be scaled to lower fluences using separate scale factors for the acceptor density N_A, donor density N_D, and trapping rates $\Gamma_{e/h}$:

$$N_A(\Phi) = R_A(\Phi)N_A(\Phi_0), \qquad N_D(\Phi) = R_D(\Phi)N_D(\Phi_0), \qquad \Gamma_{e/h}(\Phi) = R_\Gamma(\Phi)\Gamma_{e/h}(\Phi_0) \tag{21}$$

where the scale factors R are given by the following expressions,

$$R_\Gamma(\Phi) = \frac{\Phi}{\Phi_0}, \qquad R_A(\Phi) = R_\Gamma(\Phi)(1+\alpha), \qquad R_D(\phi) = R_\Gamma(\Phi)(1-\alpha) \tag{22}$$

and where α depends upon the fluence. The scale factors that were determined at the fluences $2 \times 10^{14}\mathrm{n_{eq}/cm^2}$ and $0.5 \times 10^{14}\mathrm{n_{eq}/cm^2}$ are plotted in Fig. 39. These provide essentially a one-parameter prescription to tune the model to intermediate fluences and should greatly expedite the calibration process.

The actual calibration procedure is to repeat the beam test measurements in-situ in CMS. This requires that samples of large-η tracks and pixel clusters be recorded at a series of pixel bias voltages. It has already been shown [13] that due to displaced primary vertices, it is possible to acquire such samples even in the central barrel modules. In principle, the in-situ measurement could acquire large statistics in only a few hours of dedicated operation at several bias voltages. Since it will require several years of operation to reach fluences comparable to Φ_0, the calibration procedure will not have to be performed frequently. However, because the readout chip is zero-suppressed, the very useful information in the small tails of the charge collection profiles will not be available. This is illustrated in Fig. 40 which shows the effect of the readout threshold upon the charge collection profile that was measured in the beam test with an un-suppressed prototype readout chip. The essential tail information is visible at only one bias setting. This implies that finer, carefully-targeted voltage scans will be required to calibrate the sensor model.

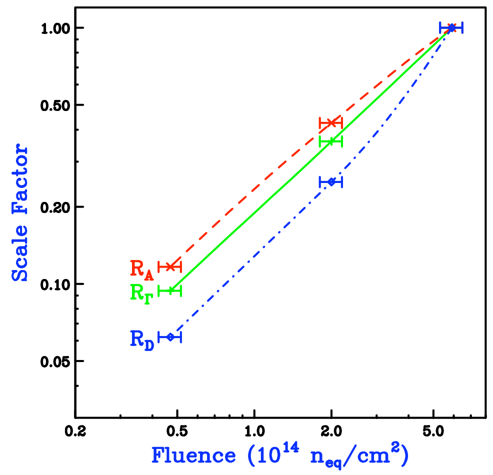


Figure 39: The scale factors R_A, R_D, and R_Γ as defined in equations 21 and 22 are plotted as functions of fluence.

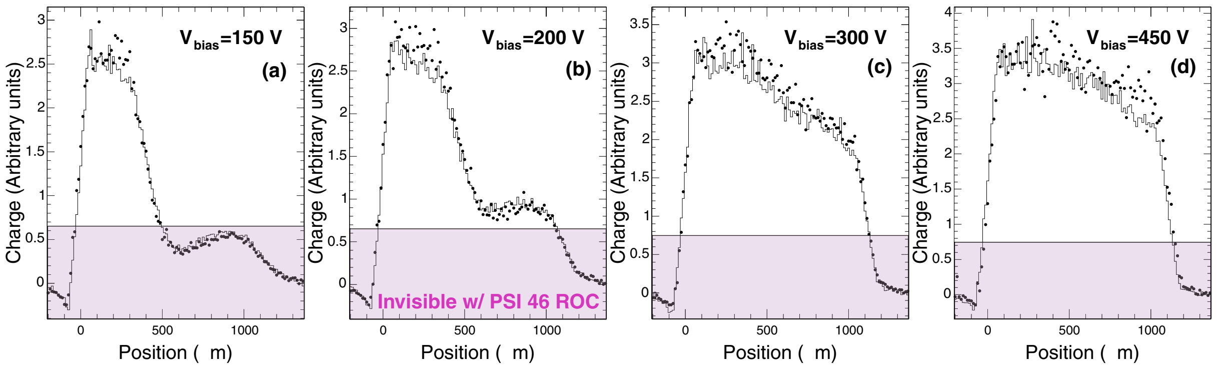


Figure 40: The measured charge collection profiles at a temperature of $-10°$C and bias voltages of 150V, 200V, 300V, and 450V are shown as solid dots for a fluence of 5.9×10^{14} n_{eq}/cm^2. The two-trap double junction simulation is shown as the solid histogram in each plot and the shaded region shows the effect of the readout threshold. Any signal dot inside the shaded region will be invisible in-situ in CMS.

Unfortunately, the template calibration cannot be automated. The calibration procedure will always require some iteration and hand adjustment of the modeling parameters. We are planning to develop a suite of CMSSW software packages to facilitate this. After the implementation of the template-based simulation package in CMSSW (see Section 7) it will be possible to develop this software using data from simulated irradiated sensors.

6.7 Alignment Sensitivity

Although the template reconstruction algorithm is moderately sensitive to the knowledge of the track direction, the sensitivity is far below the angle resolution of the tracker and is less than any likely angle offsets caused by misalignment of the pixel system even in the earliest running. To demonstrate this, we assume that the local track direction is entirely determined by two pixel planes separated by 4 cm. The effects of y- and x-misalignments upon the resolutions are shown in Fig. 41. It is clear that a 2 mm misalignment in y and a 1 mm misalignment in x have negligible effects on the resolutions. Note that the angle offsets caused by the misalignments do not bias the residual distributions.

6.8 Track Seeding

The sensitivity of the template algorithm to the cluster shapes was discussed in Section 6.5.3 in the context of identifying secondary electron backgrounds. The template probabilities test the consistency of the observed cluster shapes with the shapes expected for the input angle hypotheses. This technology can also be used to test the consistency of the observed shapes with the angle hypotheses. Pixel doublets and triplets are used to seed the Kalman Filter track finding algorithm. Each pair of pixel hits defines the β-direction of a possible track and each

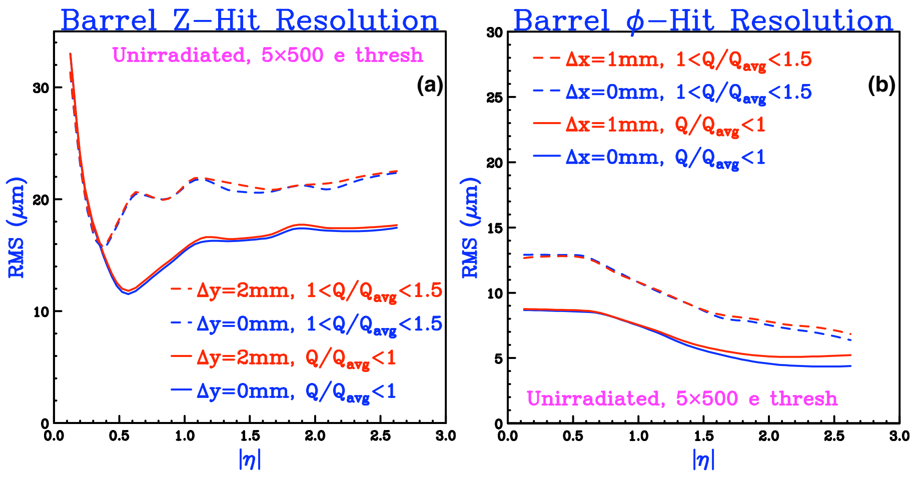


Figure 41: The rms y-residuals and x-residuals of a Pixelav-generated sample of template-reconstructed barrel clusters are plotted versus pseudorapidity for the cluster charge bands $1.5 > Q/Q_{avg} > 1$ (dashed curves) and $1 > Q/Q_{avg}$ (solid curves). The blue curves assume that two adjacent planes are aligned and the red curves assume that the planes are misaligned by 2 mm in the y-direction and 1 mm in the x-direction.

triplet of hits defines the α-direction of a possible track. The cluster shapes crudely measure both of these angles and can, in principle, be used to “validate” possible track seeds. This idea is sketched schematically in Fig. 42 which shows (not to scale) the y- and x-projections of a possible triplet of pixel hits. We note that the lengths of both cluster projections in the middle layer are inconsistent with the triplet hypothesis. Rejecting inconsistent track seeds before the Kalman Filter is invoked can significantly reduce the track-finding time.

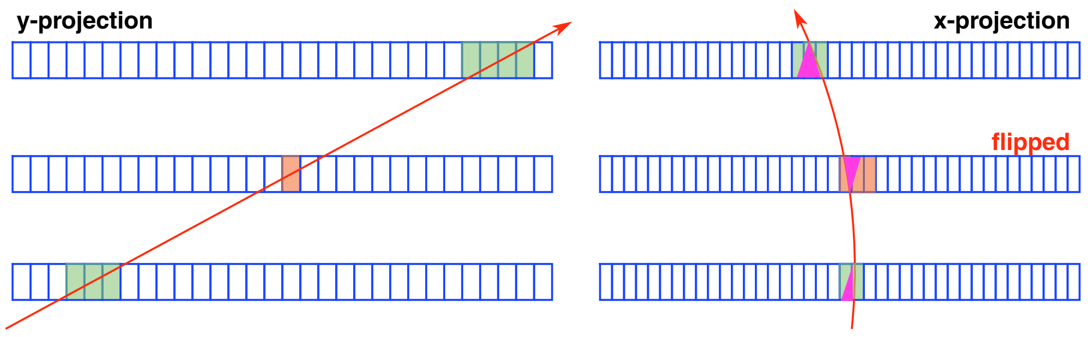


Figure 42: A schematic diagram of a pixel triplet seed and a track hypothesis. Note that the projected sizes of the clusters (shaded green) in the outer layers are consistent with the track angles whereas the projected sizes of the cluster (shaded red) in the middle layer are not consistent with the track angles.

A simpler version of this procedure that compares the y-lengths of pairs of barrel clusters was developed by the Nebraska Group [14] and is implemented in CMSSW/RecoTracker/PixelStubs. The intrinsic $\cot\beta$ resolution of the y-length technique is compared with that of the template algorithm in Fig. 43. A special version of the template code that searched a range of $\cot\beta$ hypotheses for a chisquare minimum was implemented for this study. The rms widths of the $\cot\beta$ residuals are plotted as functions of the track direction in $\cot\beta$ bins (note that $\cot\beta = 6$ is approximately $\eta = 2.5$) in the charge bands $1.5 > Q/Q_{avg} > 1$ (dashed lines) and $1 > Q/Q_{avg}$ (solid lines). They are compared with the rms resolutions of the $\cot\beta$ derived from the cluster length in pixels according to the following expression,

$$\cot\beta = (y_L - y_F)/T, \tag{23}$$

where: y_L and y_F are the y-coordinates of the first and last pixels in the cluster and T is the sensor thickness. Note that the intrinsic template resolution is better by a factor of two.

The procedure used to extract $\cot\beta$ information from cluster shapes searches a fairly large space for the chisquare minimum and is fairly slow. A more elegant approach is to use the reconstructed coordinates of pixel doublets or triplets to generate angle hypotheses and to cut on template probabilities to test consistency with those angles.

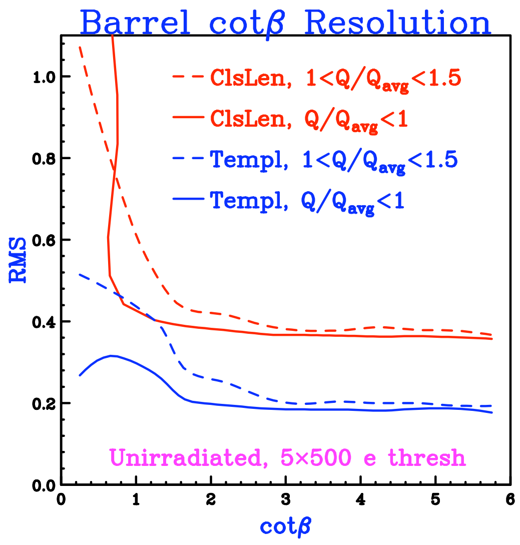


Figure 43: The rms $\cot\beta$ resolutions of a special version of the template algorithm (blue lines) are compared with the resolutions of a simple cluster length estimator (shown as red lines, see equation 23) as functions of $\cot\beta$ in the charge bands $1.5 > Q/Q_{avg} > 1$ (dashed lines) and $1 > Q/Q_{avg}$ (solid lines).

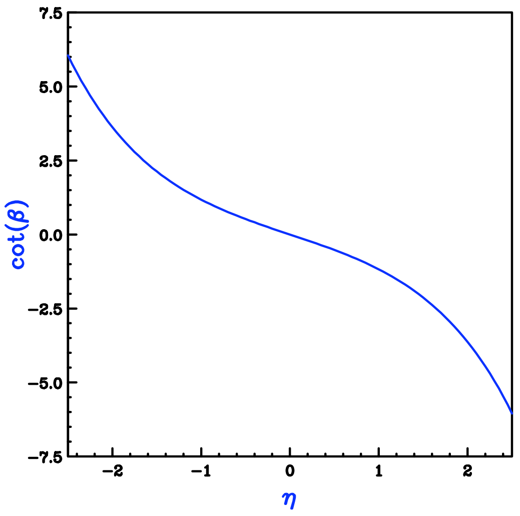


Figure 44: The relationship between $\cot\beta$ and pseudorapidity η in the pixel barrel. Note that the cluster lengths (proportional to $\cot\beta$) increase rapidly with η.

The template reconstruction procedure can be applied to doublets to test just the $\cot\beta$/y-projection consistency ($\cot\alpha$ can be set to zero) and it can be applied to triplets to test consistency in both projections. The efficiencies of angle hypotheses that are displaced by $\Delta\cot\beta$ and $\Delta\cot\alpha$ are shown in Fig. 45 for several values of template probability cuts. We note that the relationship between $\cot\beta$ and pseudorapidity in the pixel barrel is quite simple,

$$\cot\beta = \frac{e^{-\eta} - e^{\eta}}{2} = -\sinh\eta, \tag{24}$$

and is plotted in Fig. 44. The acceptance of the detector is approximately $|\cot\beta| < 6$ and $|\cot\alpha| < 0.225$. This suggests that the y-probability requirement could significantly reduce the number of seeds especially at large η where tracks that are relatively close in η can deviate in $\cot\beta$, but that an x-probability requirement will have a less dramatic effect.

The development of a template-based "seed cleaner" is in progress. The Modified Pixel Seeder processes pixel doublet seeds generated by the Global Pixel Seeder by applying the template algorithm to both pixel hits and by requiring that the y-probabilities exceed 10^{-3}. The results of the application of the seeder and the seeder/cleaner combination to a sample of 750 simulated $t\bar{t}$ events are summarized in Table 3. The cleaner reduces the number of seeds by more than a factor of two and reduces the tracking time by almost a factor of two. Taking into account the additional overhead of the template algorithm, the total time for seeding and tracking is reduced by 40%. The template-based procedure loses about 1.6% of the tracks. The quality of the lost tracks is unknown and is currently under study.

The use of the template algorithm for seed cleaning has the additional advantage that it can be calibrated to match the degrading performance of the pixel detector. At a minimum, this should keep the efficiencies of the seed-cleaner reasonably constant in time even if the rejection power for poor seeds declines. In actuality, the reverse may be true. As the detector ages, the template algorithm will acquire the ability to distinguish between positive and negative $\cot\beta$ due to the asymmetry of charge trapping. This may actually improve the ability to reject some backgrounds. The use of the template algorithm to validate pixel seeds would also improve the resolution of track parameters even in the first pass of the track finding since it would automatically reject non-primary particles and would improve the resolution of RecHits used in the first-pass track fitting.

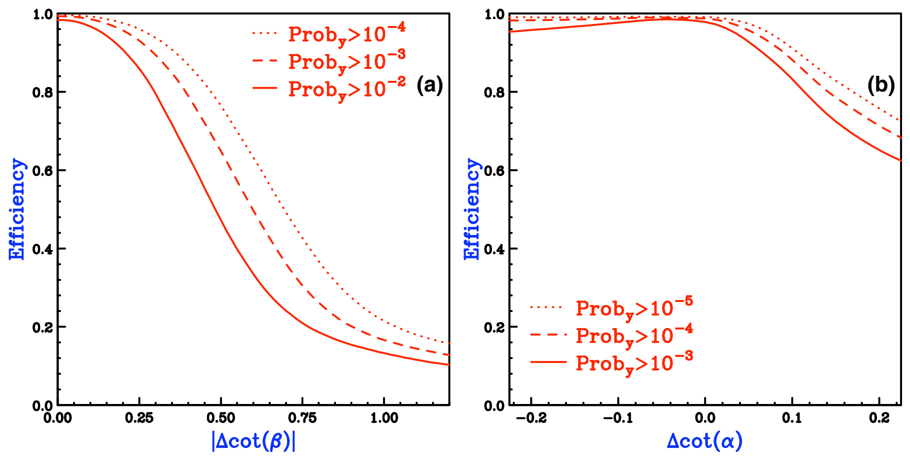


Figure 45: The average efficiencies of several y-probability and x-probability requirements for samples of tracks that have angle hypotheses shifted by $\Delta\cot\beta$ and $\Delta\cot\alpha$. Narrower distributions imply greater angle sensitivities of the probability functions.

Table 3: Comparison of the normal pixel seed builder (Global Pixel Seeder) and a template-based seed cleaner (Modified Pixel Seeder) for a sample of 750 simulated $t\bar{t}$ events.

Quantity	Global Pixel Seeder	Modified Pixel Seeder
Total Seeds (10^3)	1085	476
Total Tracks (10^3)	37.6	37.0
Seeding Time	0.13 s/event	0.19 s/event
Tracking Time	1.80 s/event	0.96 s/event
Total Time	1.92 s/event	1.15 s/event

7 Template-Based Simulation Algorithm

It should be clear from the discussion in Section 6.5.1 that radiation damage will significantly degrade the performance of the pixel system. It is obviously desirable to reproduce the changes in detector response in the CMSSW simulation. The modeling of irradiated sensors by the vectorized Pixelav code requires approximately 1.5 s per hit on a 2.5 GHz G5 processor and more than 3 s per hit on a 2.8 GHz Xeon processor. This code is obviously much too slow to be integrated into the CMSSW pixel simulation. It is clear that another approach is required. This section describes a technique that is based upon the same template object used in hit reconstruction. The template-based simulation technique has the advantages that it utilizes the same calibration and modeling developed for the reconstruction thereby eliminating a separate calibration for the simulation and it ensures that the simulation and reconstruction processes remain synched as the detector ages.

The template-based simulation technique is less developed than is the template-based reconstruction technique but a proof of principle has recently been developed and tested. A c++ procedure that is analogous to PixelTempReco2D has been developed but the full details of its implementation in CMSSW have not been designed and implemented.

7.1 Philosophy

There are three observations upon which the template-based simulation is built. The first is that the existing CMSSW simulation works well enough and is fast enough to describe pixel detector before the detector becomes radiation damaged. The second is that it is not possible to model the physics of a radiation-damaged sensor rapidly enough for use in CMSSW. The Pixelav simulation does model the important physics and after years of effort and orders of magnitude of improvement, it is still much too slow. The third observation is that the effects of radiation-damage are already encoded in the projected cluster shapes that are stored in the template object. The obvious idea is to try to modify the clusters generated by the standard CMSSW simulation to exhibit the effects of radiation-damage using the information stored in the templates. In principle, this is a straightforward procedure. It is complicated by the fact that the templates store one-dimensional projections of the two-dimensional clusters but the simulation generates two-dimensional clusters. A procedure is needed that can modify the 2-D clusters to achieve the changes in the 1-D projections predicted by the ratios of the templates for the generated and desired events.

7.2 Description of the Template-Based Simulation Algorithm

The template-based simulation algorithm re-weights individual pixel signals to modify the one dimensional projections as suggested by the ratios of the one-dimensional templates. This is possible because the number of pixels in a typical cluster N is usually (97% of all cases) less than or equal to the number constraints M provided by the one-dimensional projections. The following algorithm is designed to identify and re-weight the "core" of the cluster. Additional pixels from delta rays are treated in an ad-hoc manner.

Cluster Preparation: The inputs to the algorithm are the CMSSW-generated two-dimensional cluster, track angles, and hit position. The first step is to prepare the input cluster based upon information from templates corresponding to the physics model of the CMSSW simulation. Using the generated track angles and hit position, y- and x-templates corresponding to the "generating" model are interpolated and labeled G_i^y, G_j^x. The columns and rows having template signal larger than 50% of the readout threshold are identified. This defines the "inside" region where the re-weighting problem will be formulated. The pixels of the CMSSW input cluster are then categorized and sequentially labeled at inside I_k or outside O_l pixels as shown in Fig. 46. In order to avoid the effects of large signal fluctuations on the re-weighting procedure, the inside signals are then truncated at the same angle-dependent maximum signal $I_{max}(\cot\beta)$ used in the reconstruction procedure. The truncated signals $\tilde{I}_k$ are then summed into y- and x-projections P_i^y and P_j^x as shown in Fig. 47.

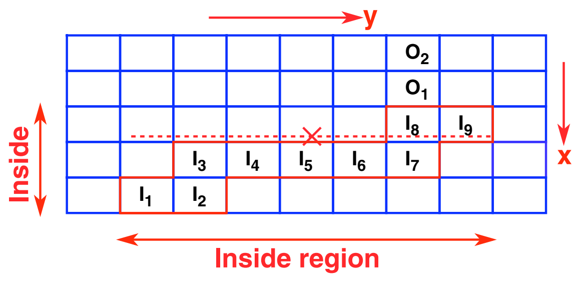


Figure 46: The categorization of the inside I_k and outside O_l pixels in the input cluster .

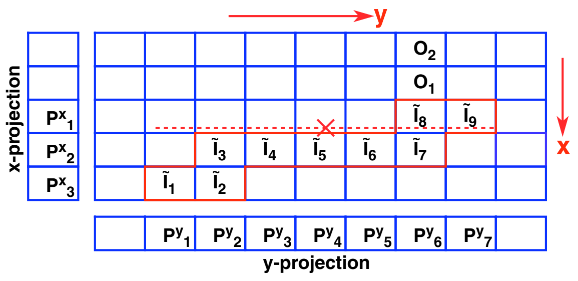


Figure 47: The truncated inside pixels are summed into y- and x-projections.

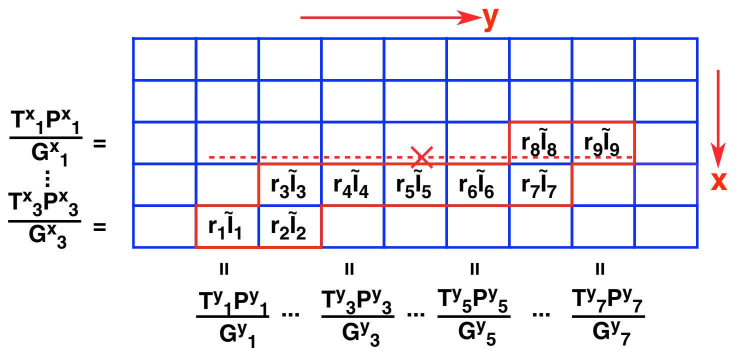
Formulation of Re-weighting Problem: The next step is to use the input track angles and hit position to interpolate the y- and x-templates, T_i^y and T_j^x, which correspond to the "target" physical model. These templates would normally correspond to the Pixelav model of an irradiated detector. The goal of the procedure is to find the N pixel weights r_k that modify the truncated pixel signals so that the re-weighted signals $r_k\tilde{I}_k$ have the y- and x-projections $P_i^y T_i^y/G_i^y$ and $P_j^x T_j^x/G_j^x$ as shown in Fig. 48.

The cluster re-weighting problem is therefore a linear problem in N unknown r_i with M conditions (M is the sum of the numbers of columns and rows in the inside region) and can be expressed in matrix form

$$\mathbf{A}\cdot\mathbf{r}=\mathbf{b} \tag{25}$$

where the $M\times N$ matrix $\mathbf{A}$ is composed of truncated signals $\tilde{I}_k$, the N-vector $\mathbf{r}=(r_k)$, and the M-vector $\mathbf{b}$ contains the re-weighted projections P_iT_i/G_i. Unfortunately, standard techniques for the solution of this problem like Singular Value Decomposition (SVD) often yield unphysical (negative) values for the weights. Note however, that when $M>N$, SVD actually minimizes the least square difference $|\mathbf{A}\cdot\mathbf{r}-\mathbf{b}|^2$. Clearly, we would like to do exactly this but with the additional constraints that all r_k be positive. This is a standard problem in the field known as Quadratic Programming. Our problem can be cast into standard form by subtracting a constant from the least square difference and minimizing the new function L,

$$L=|\mathbf{A}\cdot\mathbf{r}-\mathbf{b}|^2-|\mathbf{b}|^2=\mathbf{r}^T\cdot\mathbf{A}^T\cdot\mathbf{A}\cdot\mathbf{r}-2\mathbf{b}^T\cdot\mathbf{A}\cdot\mathbf{r}=\mathbf{r}^T\cdot\mathbf{Q}\cdot\mathbf{r}+2\mathbf{c}^T\cdot\mathbf{r} \tag{26}$$



y
x
$\frac{T^x{}_1 P^x{}_1}{G^x{}_1}$ =
$\frac{T^x{}_3 P^x{}_3}{G^x{}_3}$ =
$r_1\tilde{I}_1$
$r_2\tilde{I}_2$
$r_3\tilde{I}_3$
$r_4\tilde{I}_4$
$r_5\tilde{I}_5$
$r_6\tilde{I}_6$
$r_7\tilde{I}_7$
$r_8\tilde{I}_8$
$r_9\tilde{I}_9$
$\frac{T^y{}_1 P^y{}_1}{G^y{}_1}$ ...
$\frac{T^y{}_3 P^y{}_3}{G^y{}_3}$ ...
$\frac{T^y{}_5 P^y{}_5}{G^y{}_5}$ ...
$\frac{T^y{}_7 P^y{}_7}{G^y{}_7}$

Figure 48: Formulation of the pixel re-weighting problem.

where the $N \times N$ symmetric matrix $\mathbf{Q}$ is given by $\mathbf{Q} = \mathbf{A}^T \cdot \mathbf{A}$ and the N-vector $\mathbf{c}$ is given by $\mathbf{c} = -\mathbf{A}^T \cdot \mathbf{b}$. There are a number of commercial codes that solve this problem numerically. We were able to find a non-commercial code written mostly in c++ called OOQP for Object-Oriented Quadratic Programming [15]. It is available under a GPL-like license from the University of Chicago and relies upon the BLAS3 linear algebra package and also upon the fortran-coded MA27 sparse linear solver from the HSL archive. This code is freely available and can be compiled with f2c and gcc. OOQP can also work with the faster MA57 sparse linear solver but a commercial license is needed (except to UK academics who can receive it free from the RAL Numerical Computation Group).

Note that double pixels are treated differently in the re-weighting procedure than in the reconstruction procedure. The expansion of double pixels into pairs of single-size pixels simplified the coding of the reconstruction procedure. A similar treatment would only complicate the re-weighting procedure. Therefore, the generated and target templates are modified to model the double pixels by merging appropriate adjacent columns or rows. This is done starting at the central "struck" pixel and proceeds in both directions away from the central pixel.

Final Re-Weighting The actual re-weighting of the cluster depends upon the integers M and N. If M is larger than or equal to N (97% of all cases), OOQP is used to solve for r and the weights are applied to the un-truncated inside pixels. Any outside pixels are reweighted using the weight applied to the nearest inside pixel. This procedure is illustrated in Fig. 49. If M is less than N or if OOQP fails to find a solution (approximately 0.15% of all cases), a simpler re-weighting is peformed. If the number of columns is equal to or larger than the number of rows (normal case), the columns of cluster are re-weighted using the weights T_i^y/G_i^y. If the number of columns is less than the number of rows, the rows are reweighted using the weights T_j^x/G_j^x.

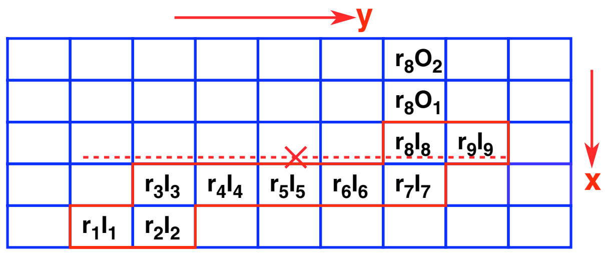


Figure 49: Application of weights to un-truncated inside pixels and outside pixels.

7.3 First Tests: Performance and Speed the Template-Based Simulation Algorithm

The simulation algorithm was tested using clusters generated by Pixelav with a simplified electric field map corresponding to the CMSSW simulation. They were re-weighted to model a sensor that was irradiated to a fluence of 6×10^{14} $\mathrm{n_{eq}/cm^2}$. The re-weighted clusters were reconstructed using the template algorithm and the residuals were compared with those produced by reconstructing full Pixelav simulations of the irradiated sensor. Initial testing showed that attempts to re-weight these CMSSW-like events lead to residual distributions with offsets in the 10-11 μm range. It was noted that the input clusters were generated assuming that the detector was operated at the nominal 150V bias whereas the output template corresponded to an irradiated sensor operated at 300V bias.

The different bias voltage leads to a different average Lorentz drift and therefore different typical topologies of the output clusters. To overcome this problem, another set of CMSSW-like events was generated with Pixelav that corresponded to the uniform field approximation for a detector operated at 300V bias. The Lorentz angle was reduced from 23° to 16.1° making the topologies of the generated clusters closer to those of the irradiated detector. Using the new event stream and its corresponding template, the residual offsets of the re-weighted and reconstructed events were reduced to values less than 1 μm. The resulting rms residuals from this second attempt are compared with those from the full Pixelav simulation of the irradiated sensor as shown in Fig. 50. The local y (global z) residuals for the two simulations are very similar. The local x (global ϕ) residuals are similar but exhibit a somewhat different η dependence. It is clear that the simulation of irradiated sensors will require that the parameters of the CMSSW simulation be "matched" with individual templates to achieve the best results.

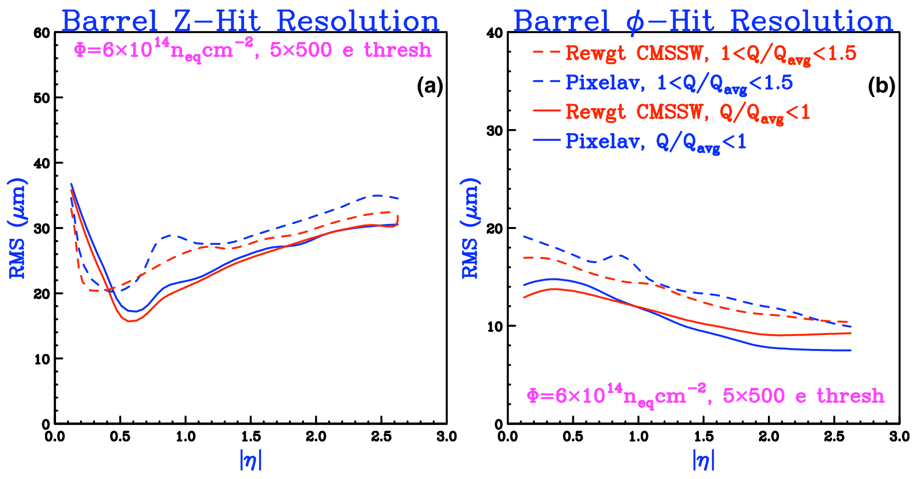


Figure 50: The rms y-residuals (a) and x-residuals (b) of Pixelav-generated (blue) and reweighted CMSSW-like (red) samples of reconstructed barrel clusters using the template algorithm are plotted versus pesudorapidity for the cluster charge bands $1.5 > Q/Q_{avg} > 1$ (dashed lines) and $1 > Q/Q_{avg}$ (solid lines). Both samples model an irradiated detector ($\Phi = 6 \times 10^{14}$ $\mathrm{n_{eq}/cm^2}$) operated at 300V bias.

The speed of the algorithm is dominated by the time needed to solve the quadratic programming problem. As tested on a 2.5 GHz G5 processor, the simulation can process approximately 3300 clusters per second which yields 0.3 ms/cluster. The CMSSW digitizer has a tested speed of 12 ms/cluster on a standard Xeon processor [16]. Even allowing for 10-20% differences in processor speed and overhead from the unwritten CMSSW interface, the speed of the re-weighting procedure is easily sufficient for use in CMSSW and is much faster than the 1.5(3) s/cluster speed of the vectorized Pixelav simulation operating on available G5(Intel) processors.

7.4 Future Development

This technique is currently the only alternative for the simulation of the Pixel system after it has become radiation-damaged. We regard the results shown in Fig. 50 as an encouraging proof of principle, but we intend to study and develop the algorithm further before final implementation. Additionally, the CMSSW interface for the template simulation has not yet been designed. Since the algorithm re-weights already-generated clusters, there are a number of possible implementation schemes. It could be invoked from within the pixel digitizer. This has the advantage that it could be applied before the response of the readout electronics and electronic noise are simulated. It is also possible to create an independent module that could re-weight "noisy" clusters after generation and before clusterization. The performance of this scheme is slightly worse than the first, but it simplifies the implementation.

8 Conclusions

This note has described new techniques for the reconstruction/validation and the simulation of pixel hits. The techniques are based upon the use of pre-computed projected cluster shapes or "templates". A detailed simulation called Pixelav that has successfully described the profiles of clusters measured in beam tests of radiation-damaged sensors is used to generate the templates. Although the reconstruction technique was originally developed to optimally estimate the coordinates of hits after the detector became radiation damaged, it also has superior performance before irradiation. The technique requires a priori knowledge of the track angle which makes it suitable for the second in a two-pass reconstruction algorithm. However, the same modest angle sensitivity allows the algorithm to determine if the sizes and shapes of the cluster projections are consistent with the input angles. This information may be useful in suppressing spurious hits caused by secondary particles and in validating seeds used in track finding and has the potential to significantly increase the speed of track finding in the offline reconstruction. The improved tracking performance shown in Section 6.5.4 is likely underestimated. The characterization of the track parameter resolutions by Gaussian fit sigmas ignores the effects of tails which are well controlled by the template algorithm. It also uses the template algorithm in a second-pass which does not eliminate hits caused by secondary particles from the reconstructed tracks. The use of template algorithm at the seeding level would remove these and might further reduce resolution tails. Nevertheless, it seems quite possible that the already-demonstrated improvements in tracking resolution will manifest themselves in the form of improved tagging efficiency/purity for b-quarks and τ-leptons.

The implementation of the template reconstruction algorithm in CMSSW is well advanced. It can already be used to reconstruct simulated data. A suite of calibration tools needs to be developed and some additional but straightforward implementation enhancements are needed before the template algorithm could be used to reconstruct real data. Investigations of its use in seeding and track finding are just beginning.

Finally, a new procedure that uses the templates to re-weight clusters generated by the CMSSW simulation was described. The first tests of this technique are encouraging and when fully implemented, the technique will enable the fast simulation of pixel hits that have the characteristics of the much more CPU-intensive Pixelav hits. In particular, it may be the only practical technique available to simulate hits from a radiation damaged detector in CMSSW. Additional work is required to finish the algorithm development and to integrate it into CMSSW.

9 Acknowledgements

The work presented in this note has benefited from conversations with Danek Kotlinski and this manuscript has benefited from comments by Sal Rappoccio. The authors would like to thank them for their input. This work was supported by the National Science Foundation under Grant No. PHY-0457374.

10 Appendix

10.1 Implementation of the Template Reconstruction Algorithm

The template code separates into two main categories: template production and pixel hit processing. Only the latter task is integrated into CMSSW. This section first gives a brief overview of the template production processing and then describes the CMSSW implementation of the hit processing in more detail.

10.1.1 Template Production

Template production begins with the generation of the electric field map for 1/4 of a pixel cell using TCAD 9.0 [9]. The user must prepare boundary description and command files that define the structure of the pixel cell. These are processed using a TCAD utility called **MESH** that generates the mesh, an array of nodes, interconnections and doping profiles that describe the boundary value problem. This file and another command file describing the physics options and solution techniques are then processed by the code **DESSIS** which actually solves the coupled partial differential equations on the mesh. As the solution is iteratively solved for the ramping bias voltage, DESSIS stores the field information at a series of preselected bias voltages. It typically takes about 8 hours of computing on a multiprocessor workstation to calculate a complete set of bias voltages. Although TCAD is a commercial product, JHU and several CMS institutions hold licenses. Luckily, this step must be performed only when some change in temperature or irradiation-level is required.

The mesh and electric field output files from the TCAD step are processed by a small standalone c-program called **gen_efield**. This code interpolates the field values on the very non-uniform mesh to produce the regular rectangular grid of electric field points that is needed for fast field lookups in Pixelav. It also transforms the fields from the TCAD coordinate system to the Pixelav coordinate system. The ascii output file from gen_efield is augmented to include Pixelav control information. Slightly different control information is used for the Pixelav processes that simulate multiple $\cot\beta$ increments and multiple α' increments. These Pixelav processes generate samples of pixel hits at fixed angle but random position on the "struck" pixel. A total of seven Pixelav jobs generate 116 sequentially numbered output files. The first-pass template processing is performed by a standalone fortran program called

gen_xy_templateq that sequentially processes the 116 Pixelav-produced files and outputs an x-template file, a y-template file, and two rms-signal plots for each input file. The second pass processing is performed by a standalone fortran program called **gen_zp_templateqp7** that reads all 116 Pixelav, x-template, and y-template files and then produces a single file called "template_ summary_zpIJKL.out" where IJKL are four integer digits. This file contains all of the information for a single template.

The time needed for the generation of a full template is significant and is determined entirely by the speed of Pixelav. The four 2.5 GHz and two 2.0 GHz G5 processors currently available at JHU require about 2 days to generate a template corresponding to the simplified physics in the CMSSW pixel simulation. The generation of a template for an unirradiated "physical" sensor with all electrostatic effects requires about 5 days and the generation of a template for a heavily irradiated physical sensor requires about 12 days.

10.1.2 Pixel Hit Processing in CMSSW

The implementation of the template reconstruction algorithm in CMSSW is incorporated into three c++ components: the template object which is a single instance of the class **SiPixelTemplate**, the reconstruction procedure **SiPixelTemplateReco::PixelTempReco2D**, and the class **PixelCPETemplateReco** that invokes the procedure and serves as an interface to the CMSSW infrastructure. This architecture was chosen to encapsulate the template reconstruction procedure so that it could be developed and tested using Pixelav-simulated data inside simple standalone codes and could then be seamlessly incorporated in the CMSSW framework with no alterations. It has proven to be an extremely powerful design since the standalone codes can process 200,000 simulated hits and generate large numbers of diagnostic plots in only 30 seconds. In addition, since the CMSSW simulation does not describe all of the sensor physics that is relevant to actual operation, it is the only way to test the reconstruction algorithm to its full potential.

SiPixelTemplate: The class **SiPixelTemplate** is stored in "CondFormats/SiPixelObjects" to facilitate its inclusion in the pixel hit reconstruction and also in the pixel hit simulation. The latter task is in development (see Section 7) and has not been deployed yet. The class stores and interpolates all of the information contained in each template_ summary_ zpIJKL.out file needed by the reconstruction process. This information is stored in a structure of structures that is too complex to be described here (see SiPixelTemplate.h for documentation). The single instance of this class is initialized by invoking the method **pushfile(int filenum)** that reads the summary file ending in digits IJKL = filenum and then adds the structure to a private vector of structures. It can store and manage an arbitrary number of templates. The method **interpolate(int id, bool fpix, float cotalpha, float cotbeta)** causes the template object to interpolate the barrel (fpix=false) or forward detector (fpix=true) template having identity id. All of the information is interpolated in $\cot\beta$ and x-related information is also interpolated in $\cot\alpha$. The interpolated values are stored in private variables which are accessed by a series of methods. The interpolation of the parameterized functions for $(\Delta P_i^{y/x})^2$ and $D_{FL}^{y/x}$ require arguments before the interpolations can be carried out. The interpolate method initializes the appropriate pointers and interpolation ratios for these quantities so that calls to the methods **ysigma2**, **xsigma2**, **yflcorr**, and **xflcorr** can calculate the interpolated quantities.

SiPixelTemplateReco::PixelTempReco2D: The template algorithm as described in Section 6.4.2 is implemented in the procedure **PixelTempReco2D(int id, bool fpix, float cotalpha, float cotbeta, array_ 2d cluster, std::vector$\langle$bool$\rangle$ ydouble, std::vector$\langle$bool$\rangle$ xdouble, SiPixelTemplate& templ, float& yrec, float& sigmay, float& proby, float& xrec, float& sigmax, float& probx, int& qbin, int speed)** which is stored in "RecoLocalTracker/SiPixelRecHits". The first four quantities of the argument list are identical to those of SiPixelTemplate::interpolate. Additional inputs are the actual 2-dimensional cluster "cluster" which is stored in a 2-d Boost multiarray container, two vectors of boolean flags to indicate which of the rows and columns of the cluster are double-size pixels, the template object, and an integer to choose between speed and robustness in the chisquare search algorithm. The output quantities are the reconstructed y- and x-coordinates, their estimated uncertainties, the chisquare probabilities in the two projections, and an integer to indicate the cluster charge bin which is useful for diagnostic purposes. The hit coordinates are returned in micron units with respect to the origin located at the center of the multiarray element cluster[0][0].

PixelCPETemplateReco: The class **PixelCPETemplateReco** provides the infrastructure needed to support the use of PixelTempReco2D in the CMSSW environment. The template is stored in a private class variable. The method **localPosition(const SiPixelCluster& cluster, const GeomDetUnit& det)** unpacks the cluster and decodes the region (barrel or forward). It then calls PixelTempReco2D and returns the coordinates of the hit. The coordinate uncertainties, probabilities, and cluster charge bin are stored in private class variables. The method **localError()** must be called after localPosition to access the errors. The template probabilities are accessed by the methods **probabilityX()** and **probabilityY()**. Finally the method **qBin()** returns an integer from 0 to 4 where 0-3 define the physical charge bands: $Q/Q_{avg} > 1.5$, $1.5 > Q/Q_{avg} > 1$, $1 > Q/Q_{avg} > 0.85$, and $0.85 > Q/Q_{avg} > Q_{min}/Q_{avg}$. A value of 4 indicates that Q is less than the minimum physical charge Q_{min} and that the cluster likely to have been generated by secondary particle.

10.1.3 Speed

The template algorithm involves considerably more processing than does the standard CMSSW algorithm. To improve its speed, the original coding was modified to avoid memory allocation and deallocation by placing most of the working variables in static memory. Additionally, the original extremely robust algorithm was modified to implement several different search schemes for the initial minimum χ^2 bin (see Section 6.4.2. These can varied from the original scheme (speed=0) to the most aggressive one (speed=3). The performance of the algorithm is quite insensitive to this choice. The actual timing of the hit reconstruction scales with the number of pixel clusters in an event. The timing of the standard algorithm, and the template algorithm with speed=0 and speed=3 are listed in Table 4. We note that the template algorithm varies from three (speed=3) to four (speed=0) times slower than the standard algorithm but that it is still quite fast as compared with many other CMSSW tracking processes. Using the templates in the second pass hit reconstruction module "ctfWithMaterialTracks" increases its processing time by only about 5%. Since the actual track finding is very slow (nine times slower than the second pass hit reconstruction), the processing-time impact of second-pass template use is less than 1% of the total tracking chain time. Note that the application of the template algorithm to "seed cleaning" as discussed in Section 6.8 could significantly reduce the total tracking chain time.

Table 4: Tested Speed of the Template Reconstruction Algorithm (2.5 GHz Xeon)

Algorithm	Time/Cluster (μs/hit)
Standard	12.4
Template (speed 0)	52.1
Template (speed 3)	37.9

10.1.4 Future Improvements

Database Storage of Template Information: As of this writing, the template information is loaded from ascii files when the template object is initialized. Each file contains header information and 116 entries each of which contains the information to initialize a c++ struct. An improvement planned for the near term is to store this information using CMS database technology so that it is properly archived and distributed to potential offline and HLT clients.

Detector to Template Matching: At the current time, a single template is used to reconstruct all of the data from the simulated detector. In actual operation, different parts of the detector may be operated under different conditions (eg. bias voltages and temperatures may vary) and will age at different rates. This suggests that several templates will be needed to reconstruct data at any given time. The template object is designed to store multiple templates and the reconstruction procedure is designed to use them. The clusters processed by PixelCPETemplateReco object are already associated with detector elements. A mechanism to match detector elements to templates must implemented to achieve this functionality.

10.2 Implementation of the Template Simulation Algorithm

The CMSSW implementation of the template reconstruction algorithm in the form of standalone and CMSSW-interface components has been very successful. The template object and reconstruction procedure can be tested from Pixelav-generated events with a standalone code and then they can be included with no changes directly into CMSSW. This same model is being developed for the template simulation algorithm. The algorithm is encapsulated into a procedure called **SiPixelTemplateRewgt::PixelTempRewgt** which uses a single instance of the template class **SiPixelTemplate**. This procedure has been tested in standalone mode using Pixelav-generated events. A CMSSW interface has not yet been implemented.

SiPixelTemplateRewgt::PixelTempRewgt: The template-based simulation algorithm as described in Section 7.2 is implemented in the procedure **PixelTempRewgt(int id0, int id1, bool fpix, std::vector⟨float⟩ track, array_2d& cluster, std::vector⟨bool⟩ ydouble, std::vector⟨bool⟩ xdouble, SiPixelTemplate& templ, double& lsq)**. The quantities **id0** and **id1** are the identifiers of the templates that correspond to the input and output models. Usually, the id0 template corresponds to the simplified CMSSW physics used to generate the clusters and id1 template corresponds to the radiation-damaged detector. The vector **track** contains the six parameters of the track in the pixel local frame: three coordinates in microns and three direction cosines. It is assumed that $z = 0$ defines the midplane of the sensor and that the origin of the coordinate system in at the center of cluster[3][10]. The quantity lsq returns the total square deviation from the solution of the quadratic programming problem in units of square charges. All of the other quantities in the argument list are identical to those used for PixelTempReco2D (see Section 10.1.2) except that the Boost multitarray **cluster** contains the 7x21 pixel input cluster and returns a 7x21 pixel re-weighted output cluster.

References

[1] The CMSSW software project is described at http://cmsdoc.cern.ch/Releases/CMSSW/latest_nightly/doc/html/.

[2] S. Cucciarelli and D. Kotlinski, “Pixel Hit Reconstruction, CMS IN 2004/014, May 2004.

[3] The expressions given here are generalizations of similar ones found in Ref. [2] to accommodate the presence of double-size pixels in some clusters.

[4] M. Swartz, “CMS pixel simulations”, *Nucl. Instr. Meth.*, **A511**, pp. 88-91, 2003.
M. Swartz, “A Detailed Simulation of the CMS Pixel Sensor”, CMS Note 2002/027, July 2002. [Online]. Available: http://cmsdoc.cern.ch/doc/notes/docs/NOTE2002_027.

[5] D. Bortoletto (CMS Collaboration) “The CMS pixel system”, Nucl. Instrum. Meth. A **579**, 669 (2007).

[6] The CMS pixel geometry is described at https://twiki.cern.ch/twiki/bin/view/CMS/PixelGeometry .

[7] A. Dorokhov *et al.*, “Tests of silicon sensors for the CMS pixel detector,” Nucl. Instrum. Meth. A **530**, 71 (2004) [arXiv:physics/0311050].

[8] H. Bichsel, Nucl. Instrum. Meth. A **562**, 154 (2006); H. Bichsel, *Rev. Mod. Phys.* **60**, 663 (1988); and private communication (see http://faculty.washington.edu/hbichsel/MCCsi.html).

[9] TCAD 9.0 User’s Manual, Synopsys, Inc., Mountain View CA. USA, http://www.synopsys.com.

[10] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, “Numerical Recipes in C”, pg 717; J.R. Cash, and A.H. Karp, *ACM Transactions on Mathematical Software*, **16**, 201 (1990).

[11] V. Chiochia, *et al.*, “Simulation of heavily irradiated silicon pixel sensors and comparison with test beam measurements,” IEEE Trans. Nucl. Sci. **52-4**, 1067 (2005), e-print: arXiv:physics/0411143; M. Swartz *et al.*, “Observation, modeling, and temperature dependence of doubly peaked electric fields in irradiated silicon pixel sensors,” Nucl. Instrum. Meth. A **565**, 212 (2006), e-print: arXiv:physics/0510040.

[12] B. Henrich, et al., CMS Note 1997/021, March 1997. http://cmsdoc.cern.ch/documents/97/ note97_021.pdf

[13] L. Wilke, et al., private communication.

[14] A. Dominguez, et al., private communication.

[15] E. M. Gertz and S. J. Wright, ”Object-Oriented Software for Quadratic Programming”, ACM Transactions on Mathematical Software **29**, 58 (2003); online resources: http://pages.cs.wisc.edu/šwright/ooqp/.

[16] D. Kotlinski, private communication.