PPIXELAV2 Operating Instructions

Ppixelav2 is the time-sliced version of pixelav2 and comes in 2 different versions (both concurrently supported) that perform 2 different simulation tasks. The version entitled "ppixelav2_rndm_trk_n_2f" generates a predefined number of clusters that have the track angles chosen randomly over predefined ranges. The version entitled "ppixelav2_list_trkpy_n_2f" generates clusters from user supplied track angles and momenta. Both versions use three common input files and one that is specific to that version. The common inputs are the Bichsel pion-electron cross section file "SIRUTH.SPR", an input file called "ppixel2.init" that contains sensor simulation parameters and also the E-field map for 1/4 of the pixel cell, and a file containing the weighting potential map for induction calculations "wgt_pot.init" for 1/4 pixel cell. The format of the ppixel2.init file is as follows:

A single line description of the simulation parameters and field profile

P(GeV/c)   Base_Run_Number

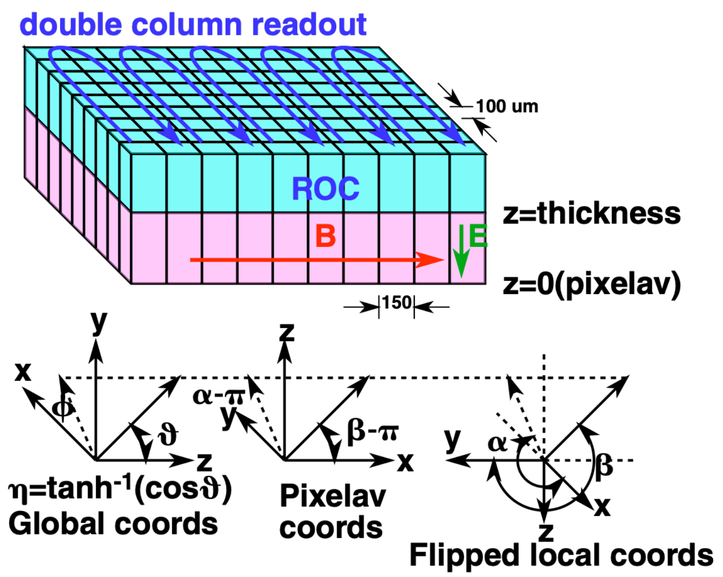
Bx(T) By(T) Bz(T)

thick(μm) sizex(μm) sizey(μm) temp(K) e fluence($10^{14} n_{eq}/cm^2$) h fluence($10^{14} n_{eq}/cm^2$) r_{He} r_{Hh} peaktime(ps) samptime(ps) eh drde nx ny nz

ix iy iz Ex(V/cm) Ey(V/cm) Ez(V/cm) [lots of these to describe the E-field map]

......

Where: P is the momentum vector of the incident pion for random track angle generation: if P<1.1(GeV/c) it is assumed that P=45 GeV/c (their entry points are generated uniformly over the area of one pixel); (Bx,By,Bz) is the magnetic field vector; "thick" is the thickness of the sensor, "sizex" is the x-dimension of the pixel, "sizey" is the y-dimension of the pixel, "temp" is the temperature, "e fluence and h fluence" set the trapping rates in irradiated sensors according to the rates given in Kramberger et al, r_{He} and r_{Hh} are the electron and hole Hall factors, peaktime is the peaking time of a CRRC preamp, samptime is the sampling time of the preamp, eh is an integer index to choose simulation of the n+ side signal (0) or the p+ side signal (1), drde is an integer to choose the original delta-ray range energy relation (0) or a newer one based on NIST Estar cross sections (1), and (nx,ny,nz) define the number of mesh nodes in each direction for the E-field map; there are then nx$\times$ny$\times$nz total lines defining the Efield map. The coordinate system is defined below where the track direction in pixelav coordinate frame is related to the local sensor coordinate frame as follows: $x_{pixelav}=-y_{local}$, $y_{pixelav}=-x_{local}$, $z_{pixelav}=-z_{local}$, $\cot(\alpha) = \Delta y_{pixelav}/\Delta z_{pixelav} = \Delta x_{local}/\Delta z_{local}$, $\cot(\beta) = \Delta x_{pixelav}/\Delta z_{pixelav} = \Delta y_{local}/\Delta z_{local}$



Note that the CRRC convolved output is currently disabled. The code outputs NCRRC time slices at intervals of samptime/NCRRC. NCRRC is a global definition that is compiled into the code.

The third common input file, wgt_pot.init, contains the weighting field map for 1/4 of a pixel cell. It begins with a single line describing the pixel size and the same mesh node count used in ppixel2.init [note that the meshes are required to have the same dimensions]. It is then followed with nx$\times$ny$\times$nz lines each with 9 elements of the 3x3 weighting potential matrix

thick(μm) sizex(μm) sizey(μm) nx ny nz

ix iy iz W00 W01 W02 W10 W11 W12 W20 W21 W22

……

The third input file for the cluster producing ppixelav_list_trkpy_n_2f code is called "track_list.txt" and reads a list of cot(α),cot(β),momentum, flipped, xlocal, ylocal, pT. The base run number given in ppixel2.init is used as a random number seed. xlocal, ylocal and pt are copied to the output file for use in network training and evaluation. pT and cot(β) are used to determine the momentum of the track.

Note that ppixelav always appends new events to cluster_events_dNNNN.out. This allows the program to be stopped and started without losing any events. **It is very important to delete these files before beginning any new simulations** (or the old and new events will both be present in the files).

Ppixelav always reads and writes files in the local directory. If you wish to run multiple instances of the code, they should be run from different directories (otherwise they will read/write the random number seed and append events to the same files causing very unstable results).

Ppixelav runs in two modes. Simply executing the program ./pixelav2_version will cause it to run from your shell. If you type ./pixelav2_version f , the program will fork a detached process and will run disconnected from the shell (you can close the shell or logoff and the process will run for the desired time). If you kill the running job, simply restarting it will begin from where you left-off. If you need additional statistics, you can just run it again with no changes. If you wish to run more than one instance of the code (on a dual processor machine) with the same initial conditions, you should make sure to use different random number seeds in the two initialization files (everything else can be the same) located in two different directories.

To begin a new simulation, there are four steps to follow: 1) remove the existing links to the input files (rm ppixel2.init, rm track_list.txt), 2) link the new input files to ppixel2.init, or track_list.txt, 3) remove old files (rm seedfile, rm cluster_events_dNNNN.out), 4) start the job (./ppixelav_version f).

The code takes up to 3 input parameters: index of first run (nth run counting from 1), number of runs (defaults to 1 if not specified), f (if present, it will fork the process). The process will produce a seed storage file "seedfileNNNN" named after the first run number and a series of "cluster_events_dNNNN.out" files as specified. This process is designed to run multiple jobs from the same directory to work on computing clusters.

All of the auxiliary code needed to run the processes is included in the source file so the local machine needs only a c-compiler and a valid mathlib.

Output File Format

See next page

```
Header [80 characters]
x-pitch [um]  y-pitch [um]  thickness [um]  time-slice-step [ps]
<cluster>
x-entry  y-entry  z-entry  n_x  n_y  n_z  number_eh_pairs y_module track_q_sign*pT
<time slice t_1>
pix_00_00 pix_00_01 pix_00_02 … pix_00_20
pix_01_00 pix_01_01 pix_00_02 … pix_01_20
.
.
.
pix_12_00 pix_12_01 pix_12_02 … pix_12_20
<time slice t_2>
.
.
.
<time slice t_16>
pix_00_00 pix_00_01 pix_00_02 … pix_00_20
pix_01_00 pix_01_01 pix_00_02 … pix_01_20
.
.
.
pix_12_00 pix_12_01 pix_12_02 … pix_12_20
<cluster>
x-entry  y-entry  z-entry  n_x  n_y  n_z  number_eh_pairs y_module track_q_sign*pT
.
.
.
```

(n_x, n_y, n_z) is the track direction. Due to sampling, the pixel charges pix_yy_xx must be multiplied by 10 to get total charge = number_eh_pairs. The track position at the pixel midplane (x-entry+0.5*t*n_x/n_z, y-entry+0.5*t*n_y/n_z, z-entry+0.5*t*sign(n_z)) is always in the 3x3 array about the center pixel pix_06_10 [it is uniformly distributed within the 3x3 pixel area]. y_module is the local y of the track midlpane coordinate in L1 [varies from -8.1 mm to +8.1 mm] and track_q_sign*pT is the product of the track pT and sign of the track charge. Flipped modules have z-entry=0 and $n_z > 0$. Unflipped modules have z-entry=100 [um] and $n_z < 0$.