KDetSim
A root based 3D simulation tool for semiconductor detectors

Gregor Kramberger
Jožef Stefan Institute, Ljubljana, Slovenia

Outline

- Motivation
- Simulation of charge transport
- Structure of simulation library
- Examples of simulations
  - Pad detector
  - Strip detector
  - 3D detector
  - Pixel detector
- Silicon drift detector
- Conclusions and future work

Motivation – Why?

Why?

TCAD simulators (Synopsis,Silvaco) are excellent, proven and are key tools for design and process simulations.

More information on TCAD packages, *M. Benoit, 11th Trento Workshop, Paris, 2016*

BUT:

- Simulation of charge collection is a solution of differential equations
  - Very demanding in terms of CPU and time (4D problem)
  - Very difficult to do Monte Carlo approach for studying detector properties crucial to HEP (charge sharing, Lorentz angle, position resolution, …)
- Not so well suited for large multi-electrode system.
- Not easy to include data from other packages e.g. GEANT
- Don't allow fast modeling and fitting of the field/free parameters to the measurements.
- Not so flexible as custom made code.

Motivation – fast simulation tools

A solution for the “BUTs” of TCAD is a fast and lightweight ROOT based package for simulation of signal in semiconductor detectors:

- ROOT interface allows for an easy and standard GUI/IO interface, well integrated with other HEP tools (GEANT4, Fluka)
- C++ code in forms of class library is very fast and kind to the computer resources
- compile on all OS that run ROOT (Mac, Linux ,Windows, Unix)
- should be easily upgradable/extendable:
  - adding new physics models: mobility (presently 5), impact ionization, radiation damage…
  - new modules – electronics processing of the signal
- extensively used in TCT simulations – direct comparison of the measured and simulated detector response

There are other tools developed within RD50:
https://indico.cern.ch/event/456679/contributions/1126330/attachments/1199070/1744044/ComparissonOfSimulators.pdf

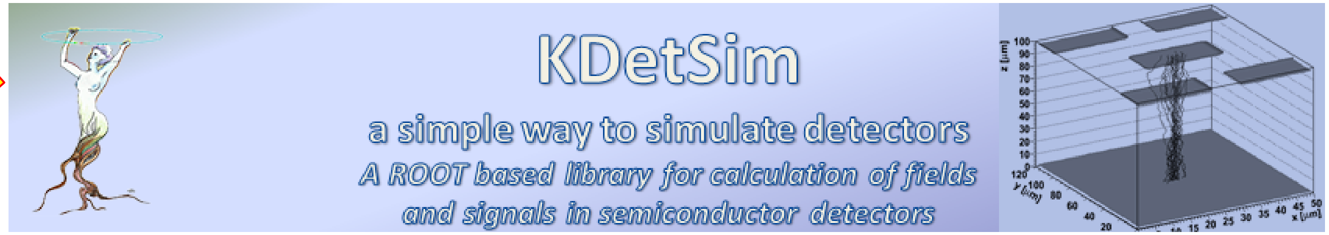
History and how to get it …

- Basic components of the simulation package done during my PhD. thesis. Over the years the package grew as the knowledge and requirements progressed (e.g. including multiplication, magnetic field, full 3D simulation …)
- Several publications were published using the software (by far not all…)
  - G. Kramberger . et al., Signals in non-irradiated and irradiated single sided silicon detectors, NIM A457 (2001) 550.
  - G. Kramberger, PhD. Thesis, University of Ljubljana, 2001.
  - G. Kramberger et al., Influence of trapping on silion strip detector design and performance, IEEE trans. nucl. sci., 2002, vol. 49(4), p. 1717 (PDF)
  - By: Mikuz, M; Studen, A; Cindro, V; et al., Timing in thick silicon detectors for a Compton camera, IEEE TRANSACTIONS ON NUCLEAR SCIENCE Volume: 49 Issue: 5 Pages: 2549-2557 Part: 2 Published: OCT 2002
  - D. Contarato, PhD Thesis, Universisty of Hamburg, 2005.
  - G. Kramberger and D. Contarato, Simulation of signal in irradiated silicon pixel detectors, NIMA 515 (2004)
  - G. Kramberger and D. Contarato, How to achieve highest charge collection efficiency in heavily irradiated position-sensitive silicon detector , NIM A 560 (2006) 98.
  - Kramberger, G.; Cindro, V.; Mandic, I.; et al. Modeling of electric field in silicon micro-strip detectors irradiated with neutrons and pions , JOURNAL OF INSTRUMENTATION Volume: 9 Article Number: P10016 Published: OCT 2014.
  - Mandic, Igor; Cindro, Vladimir; Gorisek, Andrej; et al. "TCT measurements with slim edge strip detectors" NUCLEAR INSTRUMENTS & METHODS IN PHYSICS RESEARCH SECTION A-ACCELERATORS SPECTROMETERS DETECTORS AND ASSOCIATED EQUIPMENT Volume: 751 Pages: 41-47 Published: JUL 1 2014 .

Link to the software:

http://kdetsim.org

HOME PAGE



KDetSim | Examples | Manual/Tutorial | Downloads | Class Index | Presentations

KDetSim

KDetSim introduction

KDetSim is a shared library (.dll under Windows .sl under Linux) which is dedicated to solving Poisson/Laplace equation in 2D and 3D and monte-carlo simulation of the charge transport inside semiconductor detectors. It is based on ROOT in the sense that it rellys heavily on its class libraries for all aspects of operation (visualization, IO , user interface...). The class library can be used to built executable code, but the primary use is within the root interpreter (CINT), where programs are executed in the form of macros.

Manual/Tutorial

An introduction and explanation of useage can be found in the manual, which is in fact a tutorial. Different functionalities of the classes are demonstrated on several examples.

Examples

A repository of several examples which can server as a starting point for the simulations.

Downloads

The distribution package with instructions to install on different OS platforms.

Class Index

A complete list of all classes defined in KDetSim can be found at above link. A complete hierarchy graph of all classes (Class Hierarchy), showing each class's base and derived classes can be found here and a complete list of data types here.

Who are we?

Gregor Kramberger, Jozef Stefan Institute, Ljubljana

» Last changed: 2012-10-23 15:06 » Last generated: 2012-10-23 15:06
This page has been automatically generated. For comments or suggestions regarding the documentation please send a mail to KDetSim developers.

download -> unzip -> run makefile -> compile

Basics – Calculation of Electric Field

- The package doesn't solve continuity/G-R equations in silicon, **but takes $N_{eff}(r)$** as an input

not solving the full set

$$\frac{\partial n}{\partial t} = \mu_e n \nabla \vec{E} + D_e \nabla^2 n + G_n - R_n$$

$$\frac{\partial p}{\partial t} = -\mu_h p \nabla \vec{E} + D_h \nabla^2 p + G_p - R_p$$

$$N_{eff} = p - n + N_D - N_A + \sum_{deep} Q_t \quad ,$$

Electric field

$$\nabla \vec{D} = e_0 N_{eff}(\vec{r}) \quad , \quad \vec{D} = \varepsilon\varepsilon_0 \vec{E} \quad , \quad \vec{E} = -\nabla U(\vec{r}) \quad ,$$

$$\boxed{\nabla(\varepsilon(\vec{r})\nabla U(\vec{r})) = -\frac{e_0 N_{eff}(\vec{r})}{\varepsilon_0}}$$

Boundary conditons:

$$\frac{\partial U}{\partial x} = 0 \quad , \frac{\partial U}{\partial y} = 0, \quad \frac{\partial U}{\partial z} = 0 \quad \text{at borders of simulated volume}$$

U = voltage at electrodes

Weigthing field

$$\Delta U_w(\vec{r}) = 0 \quad , \quad \vec{E}_w = \nabla U_w(\vec{r}) \quad ,$$

$$\frac{\partial U_w}{\partial x} = 0 \quad , \frac{\partial U_w}{\partial y} = 0, \quad \frac{\partial U_w}{\partial z} = 0 \quad \text{at borders of simulated volume}$$

$U_w = 1$ at readout electrode and $U_w = 0$ at all other electrodes

Current induced in electrodes:

$$I_{e,h} = \pm e_0 \vec{v}_{e,h} \vec{E}_w =$$

$$= \pm e_0 \mu_{e,h} \vec{E} \cdot \vec{E}_w$$

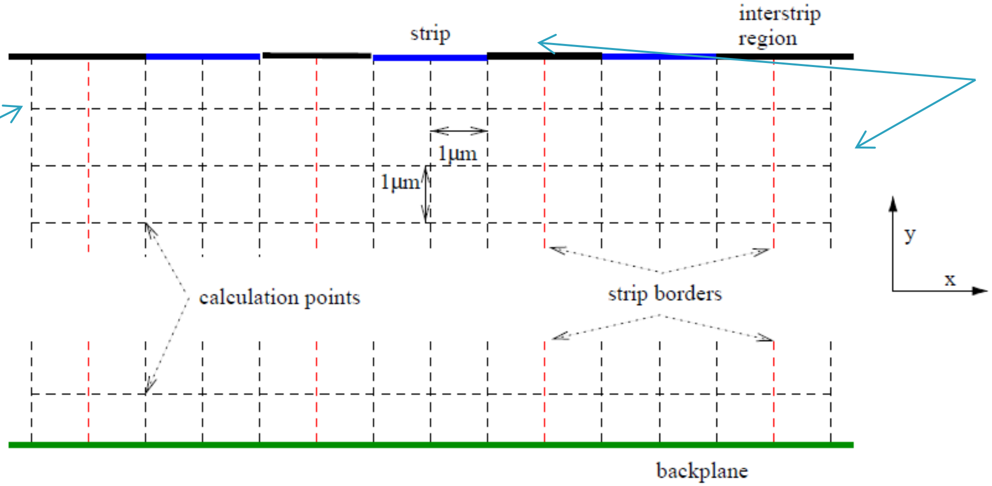
Ramo-Shockley theorem

Basics – Calculation of Electric Field

The partial differential equations are solved numerically by using finite difference equation on the mesh (FEM approach):

- 2D or 3D mesh with complex electrode arrangements/shapes (see examples).
- The mesh should be orthogonal but doesn't have to be equidistant (convergence of equation solver is the ultimate judge).

mesh density of order ¼ of the feature size



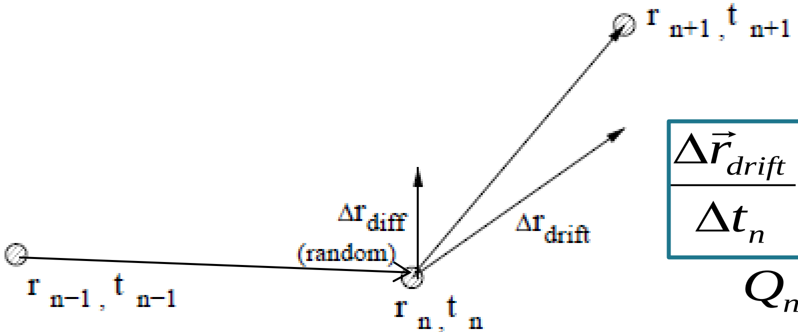
Reflective boundary condition – no field lines escape the simulated volume (von Neumann condition)

The differential equation translates to solving the system of equations for U:

- where every node represents an equation.
- The boundary conditions are essential as they determine the solution of the equations.
- The system of equations is solved by inverting the matrix: 3D structure (Nx*Ny*Nz). The matrix which should be inverted is sparse which significantly speeds up its inverse, so 10^6 node system is solved in the time scale of minutes on Core I7 portable CPU (8GB helps)

Basics – Simulation of charge transport

- Charge/current induced by a point-like charge q



$$\vec{r}_{n+1} - \vec{r}_n = \boxed{\Delta\vec{r}_{drift}} + \boxed{\Delta\vec{r}_{diff}}$$

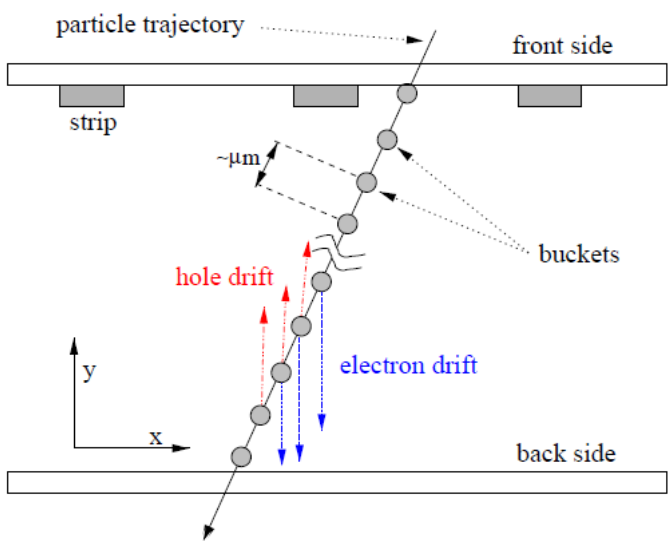
$$\vec{F} = q(\vec{E} + r_{H\,e,h}\ \vec{E} \times \vec{B})$$

$$\boxed{\frac{\Delta\vec{r}_{drift}}{\Delta t_n} = \mu_{e,h} \cdot \vec{F}(\vec{r}_n(t_n))} \quad , \quad \boxed{\Delta\vec{r}_{diff} = \vec{G}aus(\sqrt{2D\Delta t_n})}$$

$$Q_{n+1} - Q_n = q \cdot \underbrace{P(t_n, \tau_{eff})}_{} \cdot \underbrace{[U_w(\vec{r}_{n+1}) - U_w(\vec{r}_n)]}_{\text{difference in } U_w}$$

difference in U_w

P=0 or P=1 – trapping



$$I = \frac{Q_{n+1} - Q_n}{t_{n+1} - t_n}$$

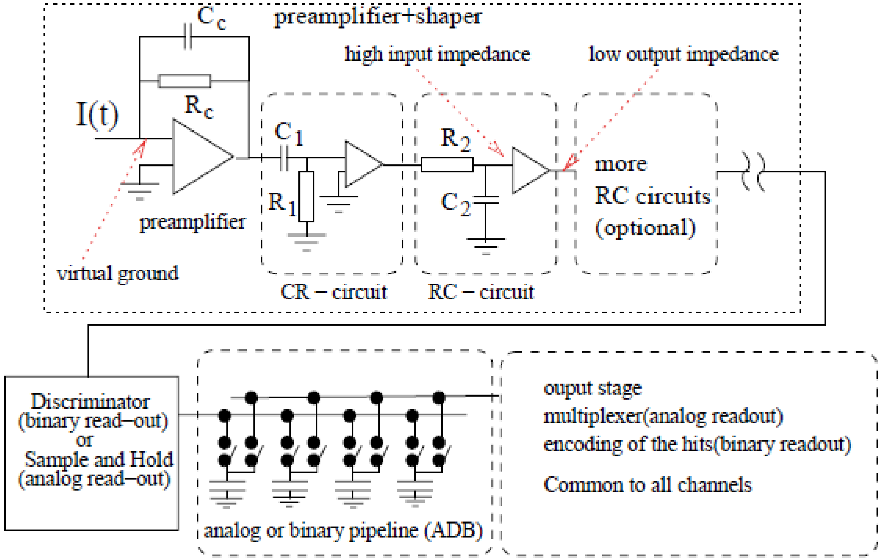
$$I(t) = \sum_{buckets} I_e(t) + I_h(t)$$

Different models are already included (Gaussian beam, exponential attenuation of beam, minimum ionizing particle), but you can easily make your own function which distributes buckets q around the sensor

Basics – electronics processing

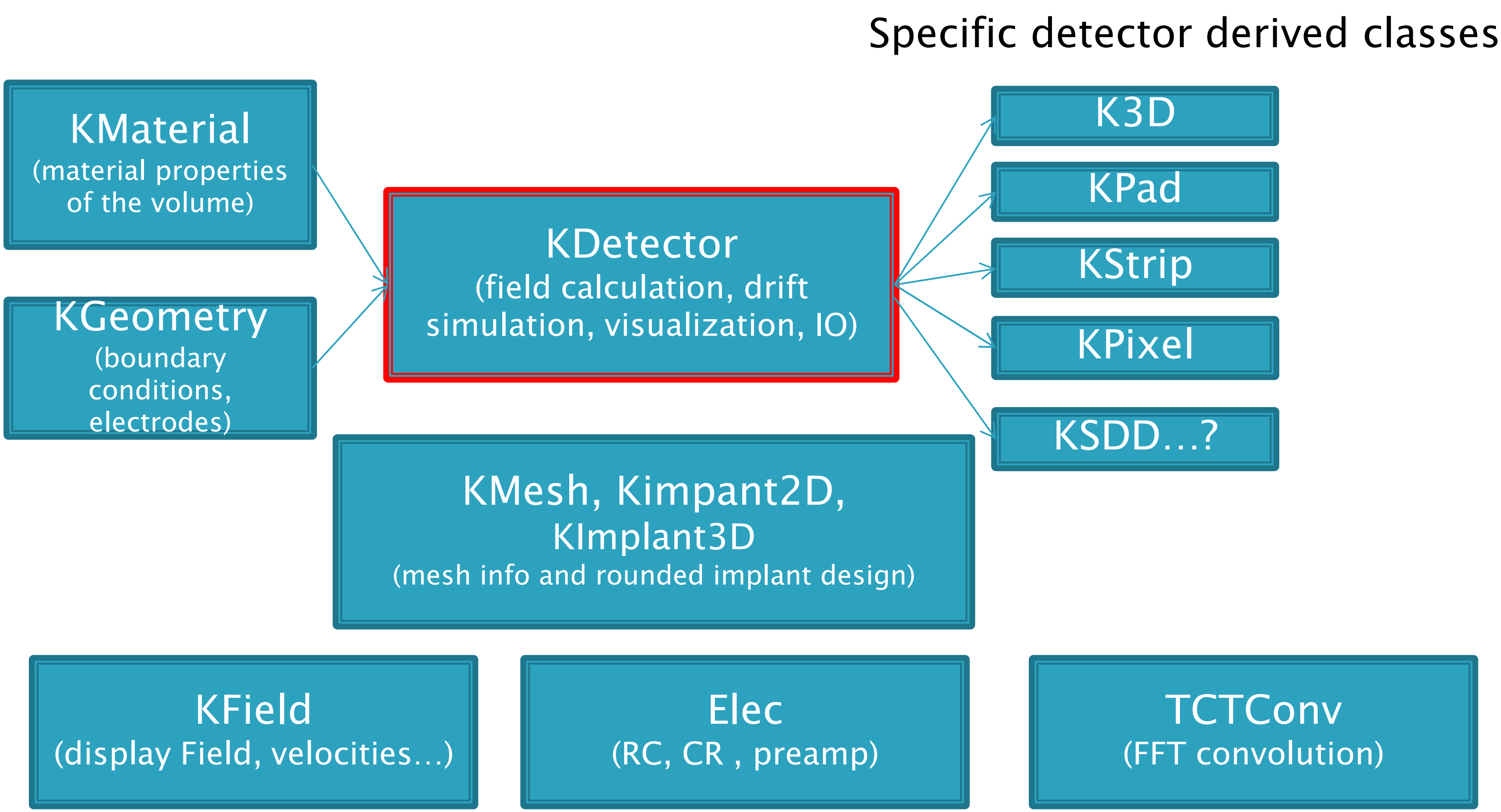
Simulated induced currents can be further processed by electronics:

- Basic electronics models are included:
  - preamp
  - CR, RC filtering / shaping
- Fast Fourier Transform Tools – Processing the induced current in the electrode by electronics (integration/amplification, shaping)



Structure of the simulation library

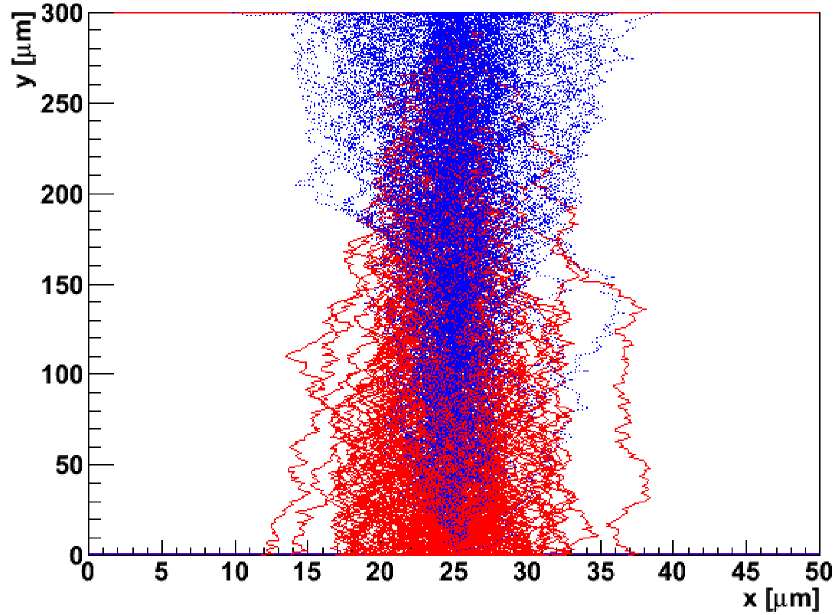
The library is a single .dll, .sl which is loaded in the ROOT framework

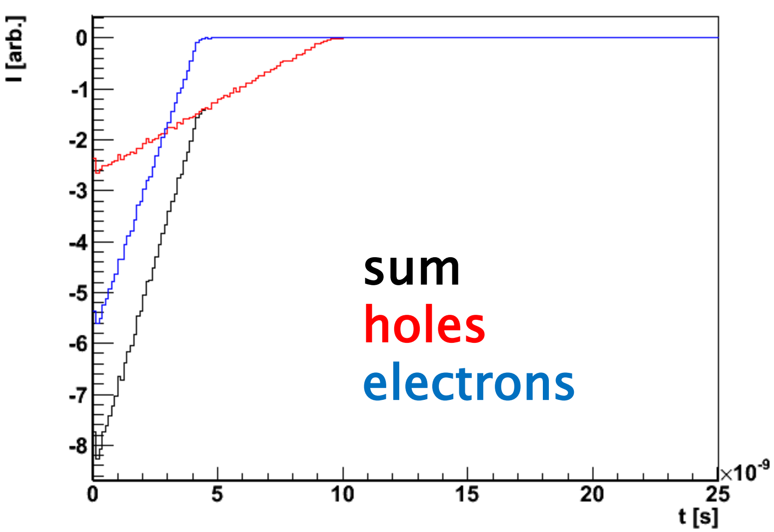


Examples of simulation – pad detectors

Simple example run as a root script:

```
{
  TF1 *neff=new TF1("neff","[0]",0,1000);
  neff->SetParameter(0,1);      // set Neff [1e12 cm-3]
  KPad det(50,300);             // dimensions of the sample
  det->Neff=neff;
  det->Voltage=-200;            // Set voltage
  det->SetUpVolume(1);          // Setup electrodes
  det->SetUpElectrodes();

  TCanvas c1;
  det->SetEntryPoint(25,299.9,0.5); // define track entry point
  det->SetExitPoint(25,1.,0.5);     // define track exit point
  det->Temperature=253;             // set temperature
  det->diff=1;                      // switch on diffusion
  det->ShowMipIR(200);              // draw drift paths for
                                    //  200 buckets

  TCanvas c2;
  det->MipIR(200,1);                 // simulate mip 200 bucket
  det->sum->Draw();                  // draw current
  det->pos->Draw("SAME");            // draw current for holes
  det->neg->Draw("SAME");            // draw current for electrons
}
```

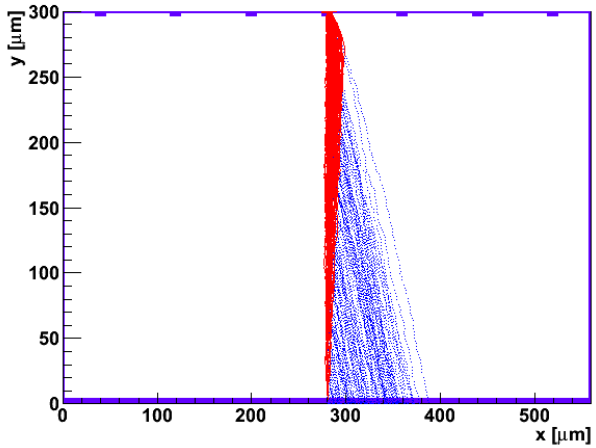


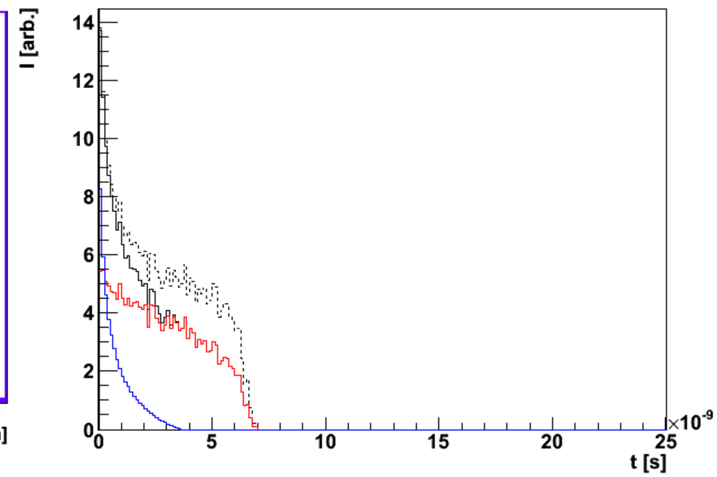


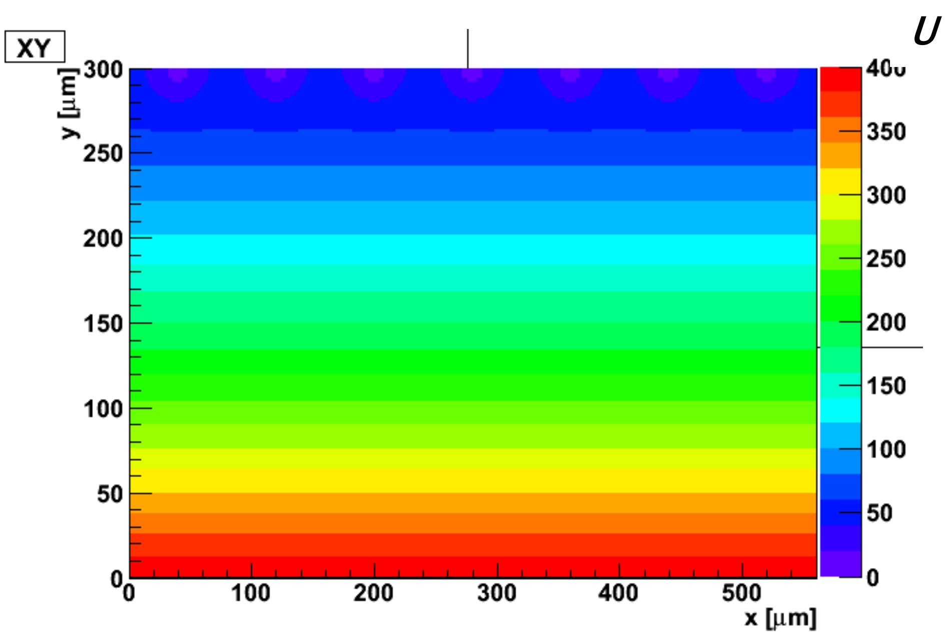
G. Kramberger, KDetSim – A root based 3D simulation tool for semiconductor detectors, SDD Fast Track Meeting, Trieste

Examples of simulation – strip detectors

```
{
 TF3 *f2=new TF3("f2","[0]",0,3000,0,3000,0,3000);
 f2->SetParameter(0,-2);

 KStrip *det=new KStrip(80,10,2,7,300);
// 80 micron pitch, 10 micron width, 300 microns
// 2 micron implant depth
// 7 strips

 det->Voltage=400;  // set voltage
 det->SetUpVolume(2);

 det->SetUpElectrodes();
 det->SetBoundaryConditions();
 det->SetUpMaterial(0); //Silicon

 det->NeffF=f2;
 det.SetDebug(0);
 det.CalField(0);  //calculate electric field
 det.CalField(1);  //calculate weigthing field

 det.B[0]=0;       // magnetic field
 det.B[1]=0;
 det.B[2]=1.8e-4;

 TCanvas c1;
  det->SetEntryPoint(200,299.9,0.5);  //entry point
  det->SetExitPoint(200,1.,0.5);       //exit point
  det->Temperature=253;                // set temperature
  det->diff=1;                         // diffusion is on
  det->ShowMipIR(100);                 // show drift paths

 TCanvas c2;
  det->MipIR(300,1);                   //simulated currents
  det->sum->DrawCopy();
  SimpleTrap(det.pos,10e-9,0,25e-9);   //do trapping
  SimpleTrap(det.neg,12e-9,0,25e-9);
det->sum->Reset();
det->sum->Add(det.pos); det->sum->Add(det.neg);
det->sum.Draw("SAME");   det->pos->Draw("SAME");
det->neg->Draw("SAME");
}
```



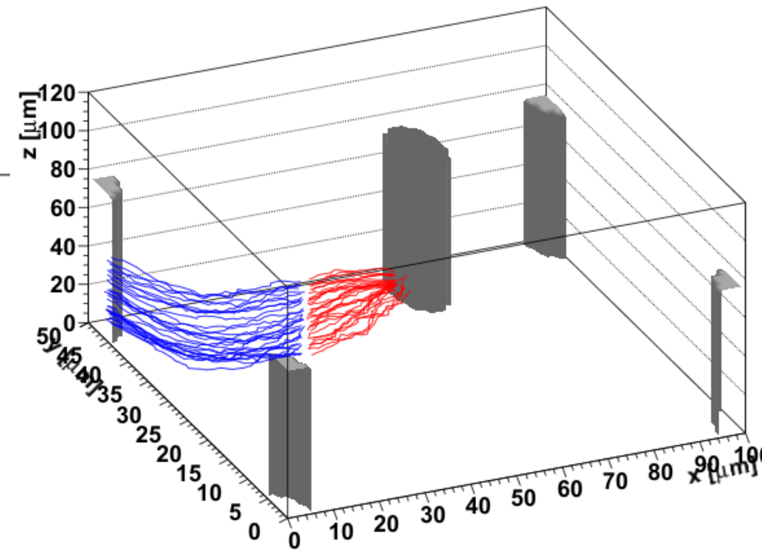


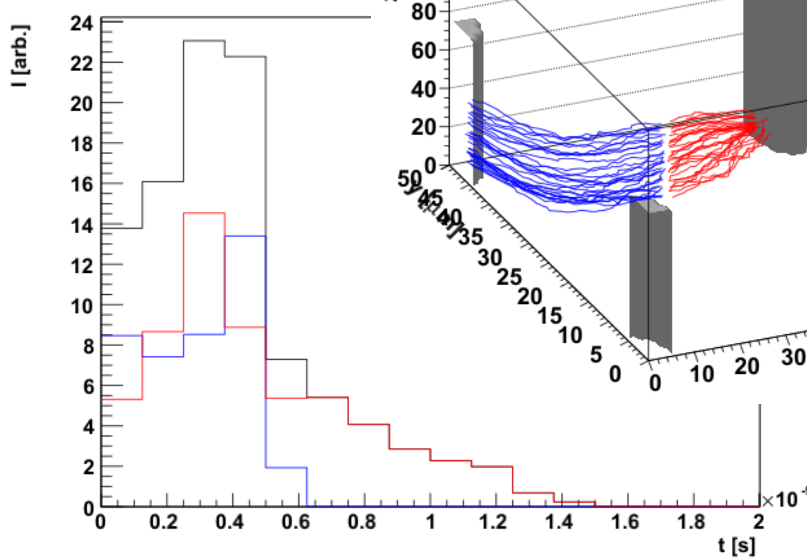


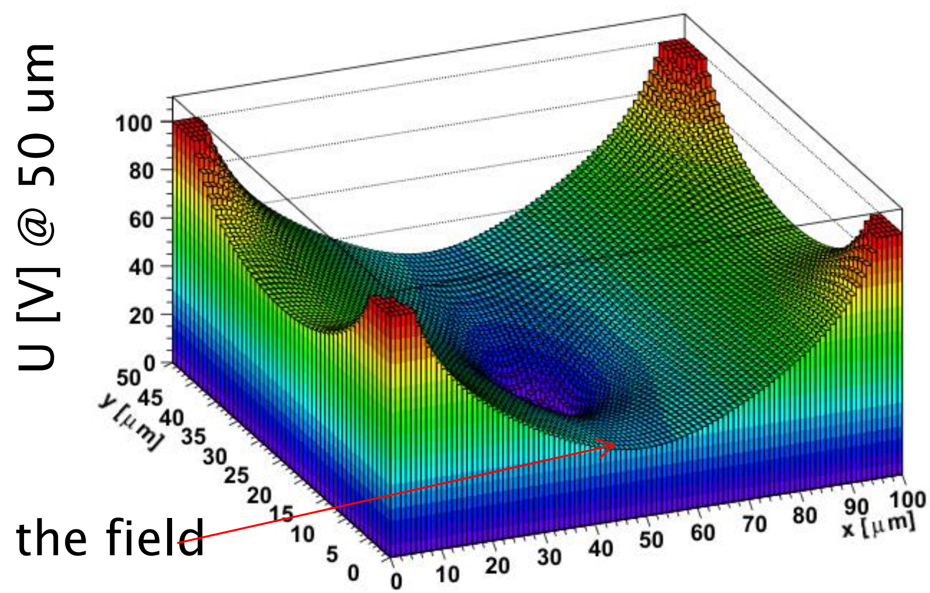
G. Kramberger, KDetSim – A root based 3D simulation tool for semiconductor detectors, SDD Fast Track Meeting, Trieste

3D detectors

```
{
 gStyle->SetCanvasPreferGL(kTRUE);

  // define a 3D detector with 5 electrodes
  // x=100 , y is 50 and thickness 120
  K3D *det=new K3D(5,100,50,120);
  // define the voltage
  det->Voltage=100;
  // define the drift mesh size and simulation mesh size in microns
  det->SetUpVolume(1,1);
  // define  columns #, postions, weigthing factor 2=0 , material Al=1
  det->SetUpColumn(0,0,0,5,75,2,1);
  det->SetUpColumn(1,100,0,5,75,2,1);
  det->SetUpColumn(2,0,50,5,75,2,1);
  det->SetUpColumn(3,100,50,5,75,2,1);
  det->SetUpColumn(4,50,25,5,-75,16385,1);
  Float_t Pos[3]={100,50,1};
  Float_t Size[3]={100,50,2};
  det->ElRectangle(Pos,Size,0,20);
  det->SetUpElectrodes();
  det->SetBoundaryConditions();
  //define the space charge
TF3 *f2=new TF3("f2","x[0]*x[1]*x[2]*0+[0]",0,3000,0,3000,0,3000);
  f2->SetParameter(0,-2);

  det->NeffF=f2;
  det->CalField(0); // calculate weigting field
  det->CalField(1); // calculate electric field

   // set entry points of the track
   det->enp[0]=30;   det->enp[1]=30;   det->enp[2]=50;
   det->exp[0]=30;   det->exp[1]=30;   det->exp[2]=10;

   // switch on the diffusion
   det->diff=1;
   // Show mip track
   TCanvas c1; c1.cd();
   det.ShowMipIR(30);
   // Show electric potential
   TCanvas c2; c2.cd();
   det.Draw("EPxy",60).Draw("COLZ");
   // calcualte induced current
   TCanvas c3; c3.cd();
   det.MipIR(100);
   det->sum.Draw();   det->neg.Draw("SAME");   det-
>pos.Draw("SAME");
}
```



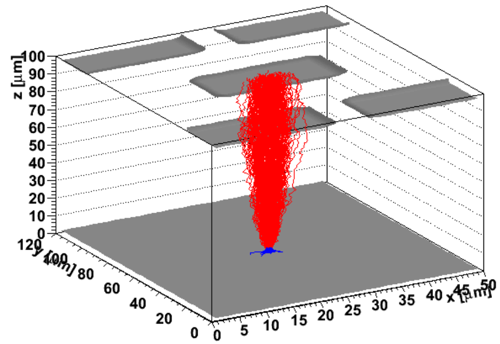


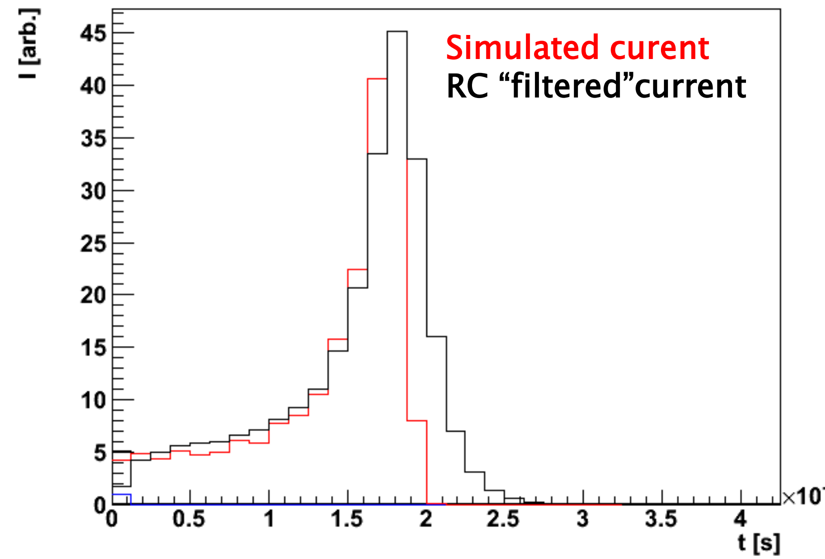


saddle in the field

Examples – Pixel sensor

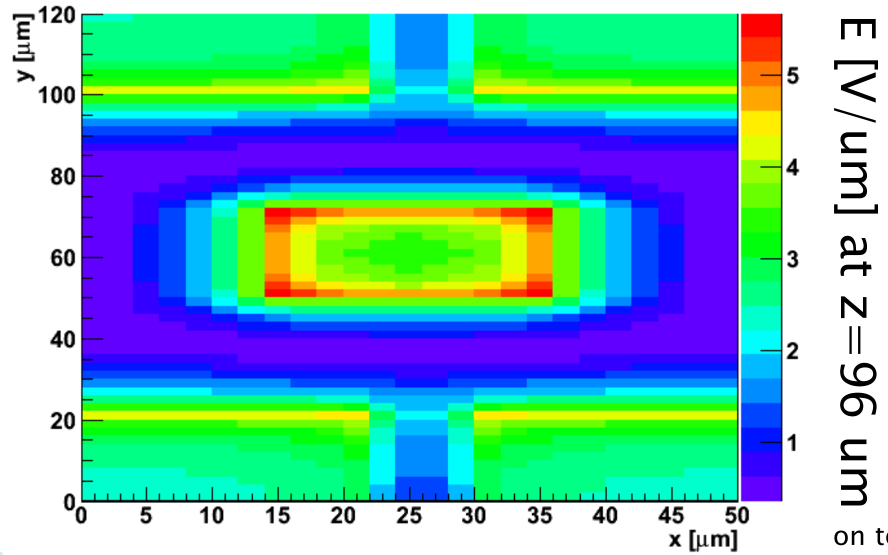
TCT generation of red light






```
{
 KPixel *det=new KPixel(5,50,120,100); // 5 pixels, x=50 um,y=120 um, z =100 um
 det->Voltage=200;                          //  setup the voltage
 det->SetUpVolume(2,2,1);                   //  2 um step for calculation of field in x,y and 1 um in z
 det->SetUpPixel(0,25,60,10,10,2,16385); // setup the center pixel for readout
 det->SetUpPixel(1,10,10,10,10,2,1);        // setup other pixels
 det->SetUpPixel(2,40,10,10,10,2,1);
 det->SetUpPixel(3,40,110,10,10,2,1);
 det->SetUpPixel(4,10,110,10,10,2,1);
 det->SetUpElectrodes();                    // init all electrodes
 det->SetBoundaryConditions();              // setup boundary conditions
  TF3 *f2=new TF3("f2","x[0]*x[1]*x[2]*0+[0]",0,3000,0,3000,0,3000);
  f2->SetParameter(0,-2);                   // space charge distribution
  det->NeffF=f2;
  det->CalField(0);  det->CalField(1);      //calculated fields
  det->diff=1;                              //diffusion is on
  det->enp[0]=25;   det->enp[1]=60;   det->enp[2]=1;
  det->exp[0]=25;   det->exp[1]=60;   det->exp[2]=3;
  det->MipIR(200,3);                        // mip simulations
  det.sum.DrawCopy(); det.neg.Draw("SAME"); det.pos.Draw("SAME");
  //electronics processing - RC filter
  Elec el(3e-12);                           // init electronics class
  el->preamp(det.sum);                      // RC filtering
  det->sum.Scale(det.pos.GetMaximum()/ det->sum.GetMaximum());
  det->sum.Draw("SAME");
}
```

Electric field strength



Silicon drift detectors

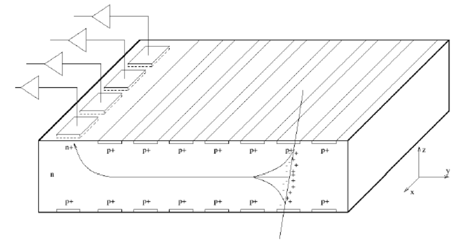
Special requirements to silicon strip/pixel detectors

- Time scale of signal formation (ns->μs).
- More steps required for charge collection – longer drift distance
- Several hundreds of different voltages (voltage divider) applied in the simulated volume.
- In order for the simulation to converge feature size is larger – few 10^6 points for the whole volume
  - 2D simulation – precise in the whole volume
  - 3D simulation – for details only a section of detector should be simulated

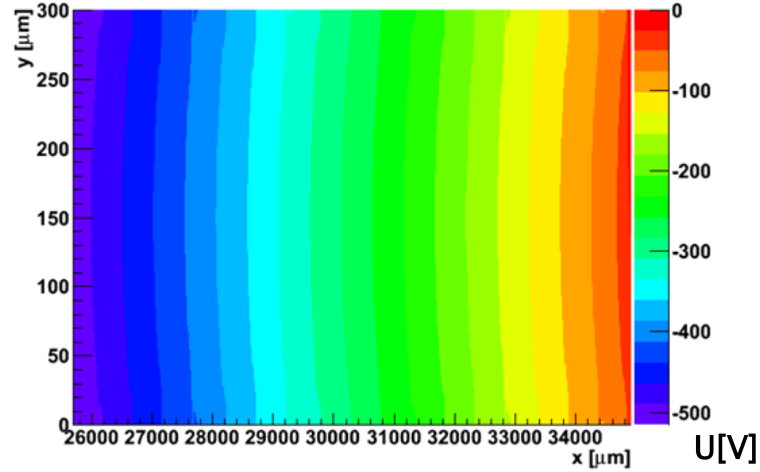
Disclaimer : I am not an expert on SDD…

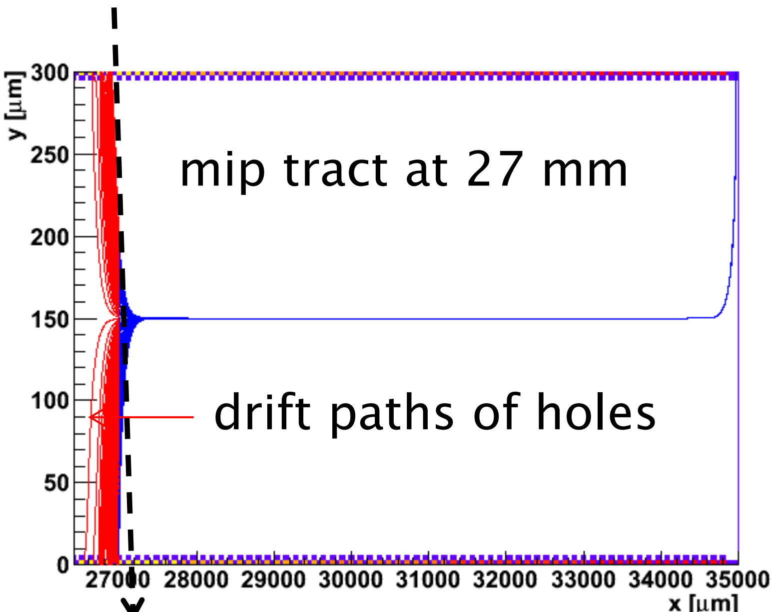
Most of the above adaptation was done at the request of Antonio.

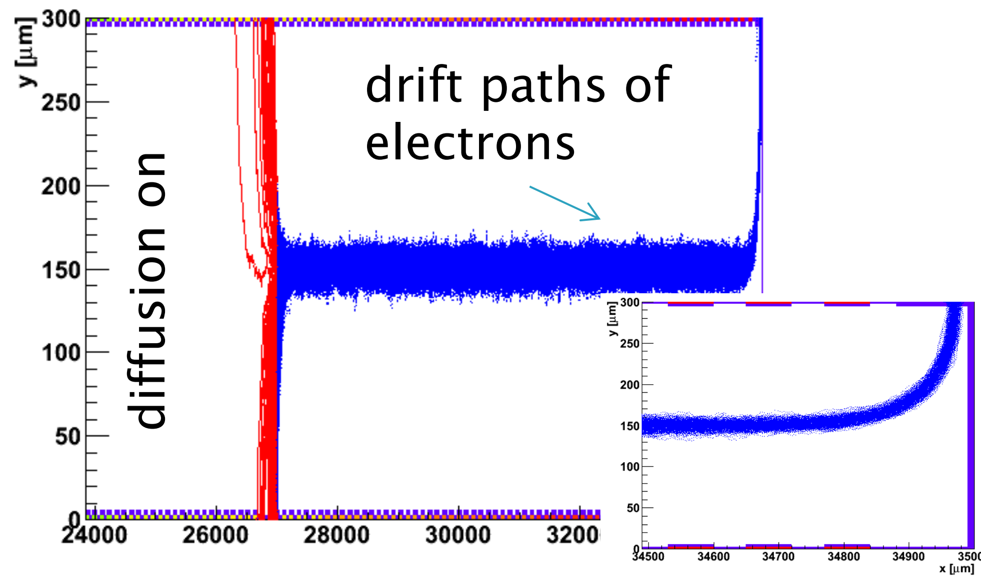
Silicon drift detector – 2D



- 120 μm field strip pitch
- 60 μm implant width (p^+)
- 300 μm thickness
- N_{eff}=5e11 cm^{-2}
- T=263 K
- ΔU=6V (–1800V@0μm) at p+ implants
- number of nodes=3e5



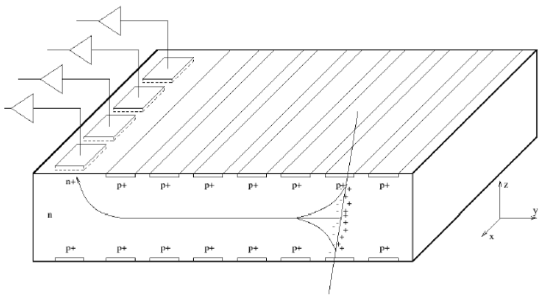


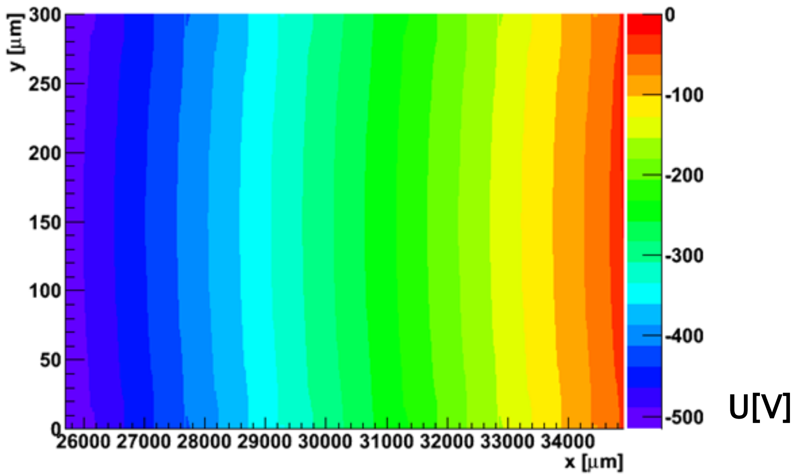


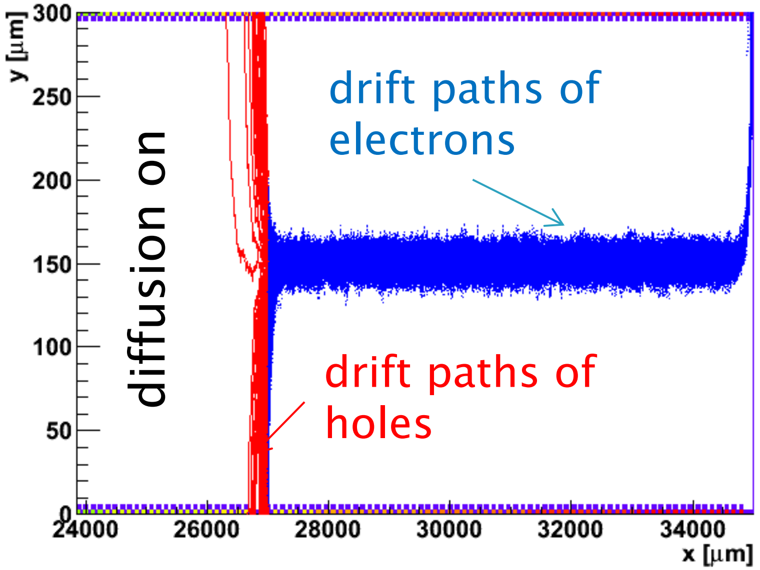
G. Kramberger, KDetSim – A root based 3D simulation tool for semiconductor detectors, SDD Fast Track Meeting, Trieste

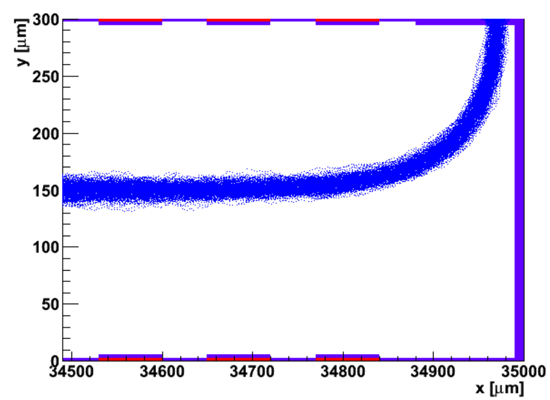
Silicon drift detector – 2D

- 120 μm field strip pitch
- 60 μm implant width (p^+)
- 300 μm thickness
- N_{eff}=5e11 cm^{-2}
- T=263 K
- ΔU=6V (–1800V@0μm) at p+ implants
- number of nodes=3e5





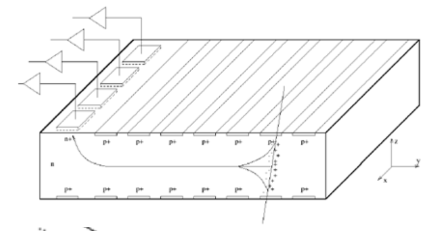


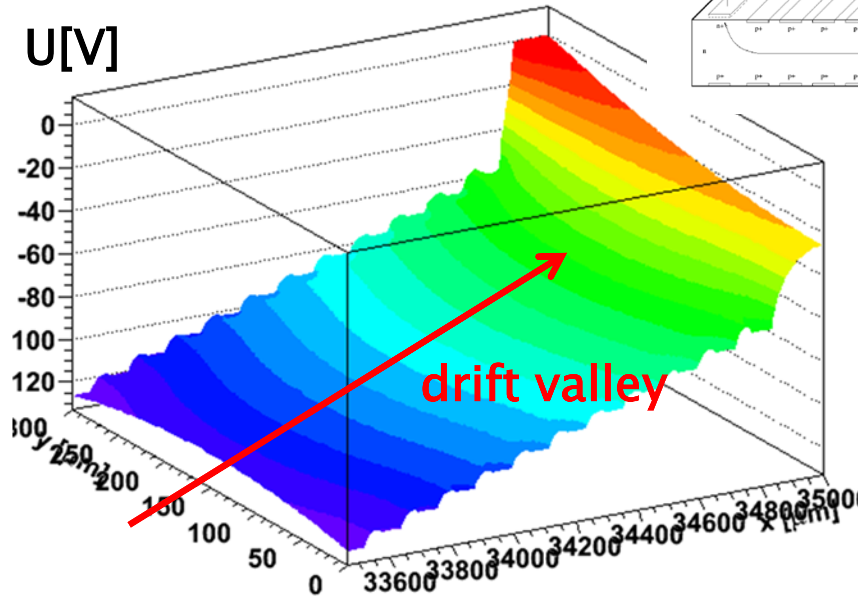


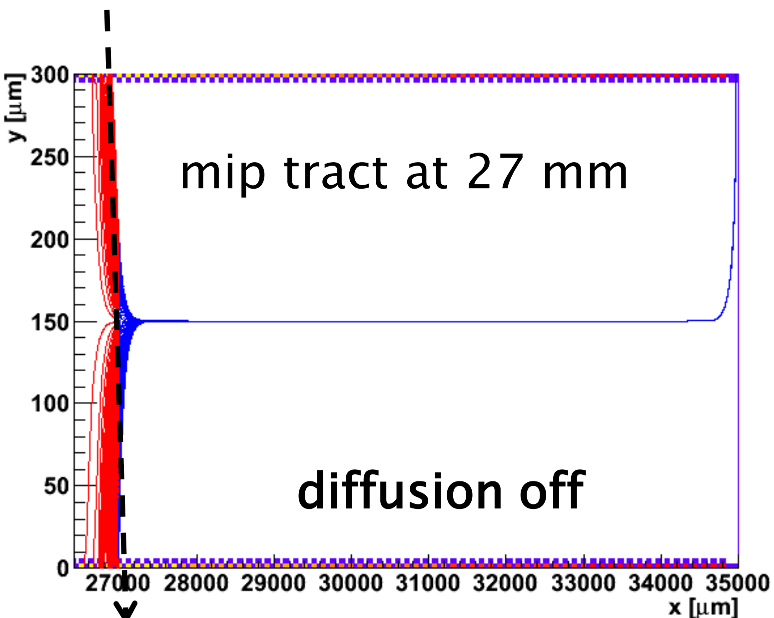
G. Kramberger, KDetSim – A root based 3D simulation tool for semiconductor detectors, SDD Fast Track Meeting, Trieste

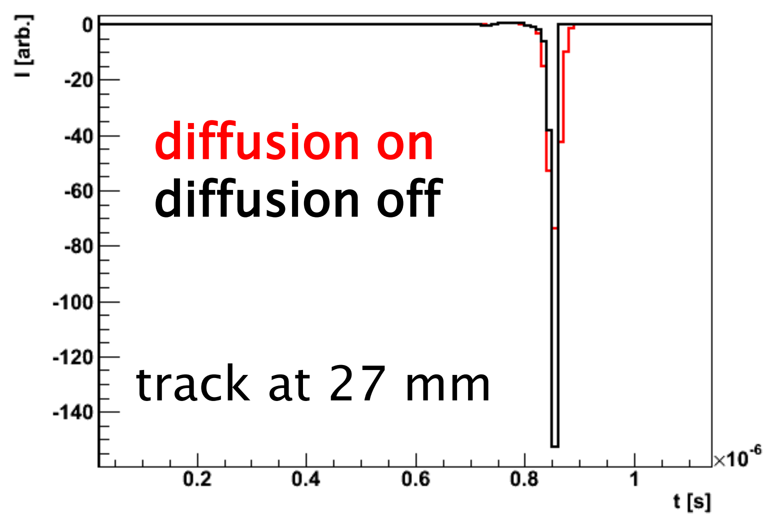
Silicon drift detector – 2D

- 120 μm pitch
- 60 μm implant
- 300 μm thickness
- Neff=5e11 cm^{-2}
- T=263 K
- ΔU=6V (-1800V@0μm)
- number of nodes=3e5





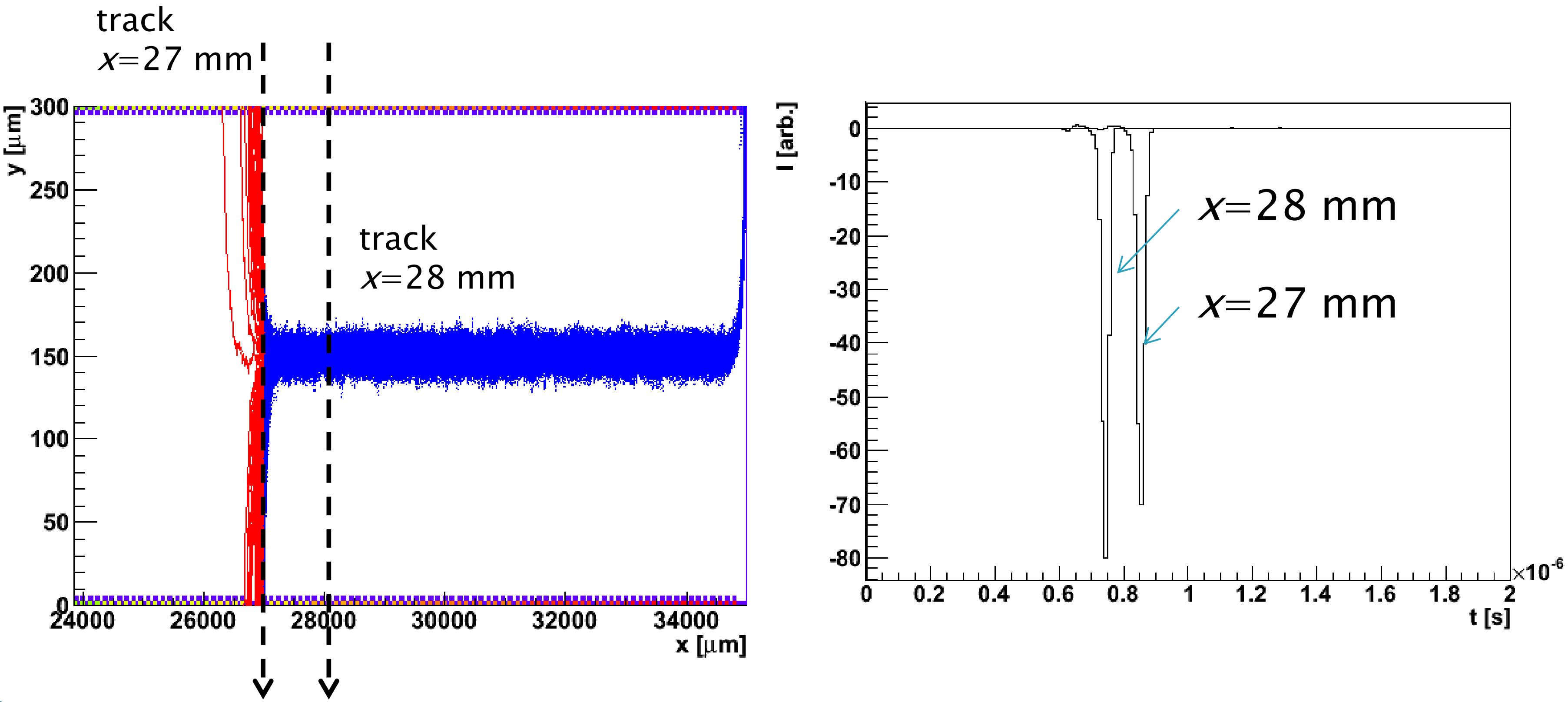




G. Kramberger, KDetSim – A root based 3D simulation tool for semiconductor detectors, SDD Fast Track Meeting, Trieste

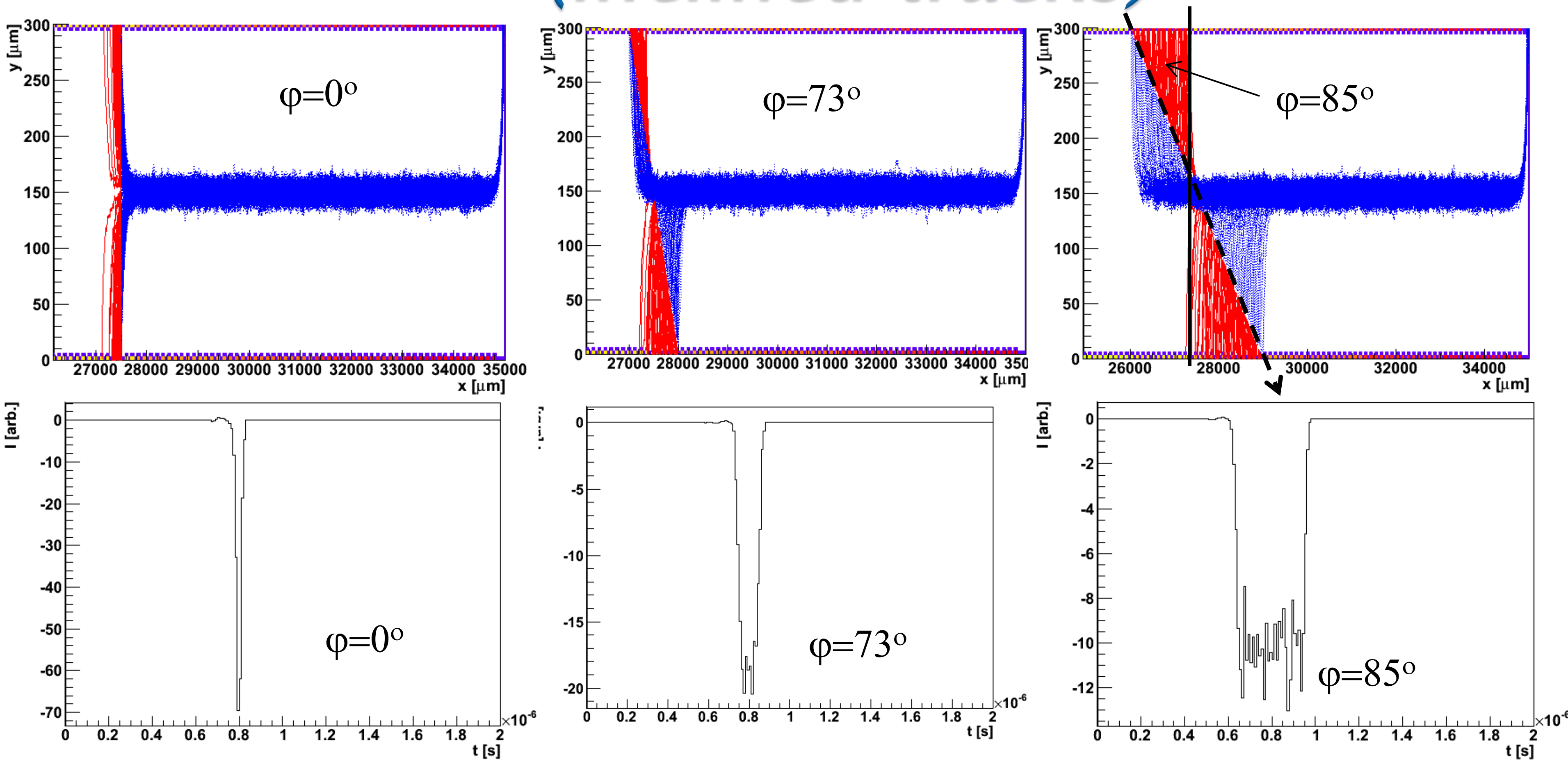
Silicon drift detector – 2D

- Example of simulated induced current – timing for different tracks
  - two positions 1 mm apart (~50 V difference) for perpendicular tracks
  - timing difference nicely seen in induced current signals
  - Some influence of diffusion on pulse shape (smaller, but wider)



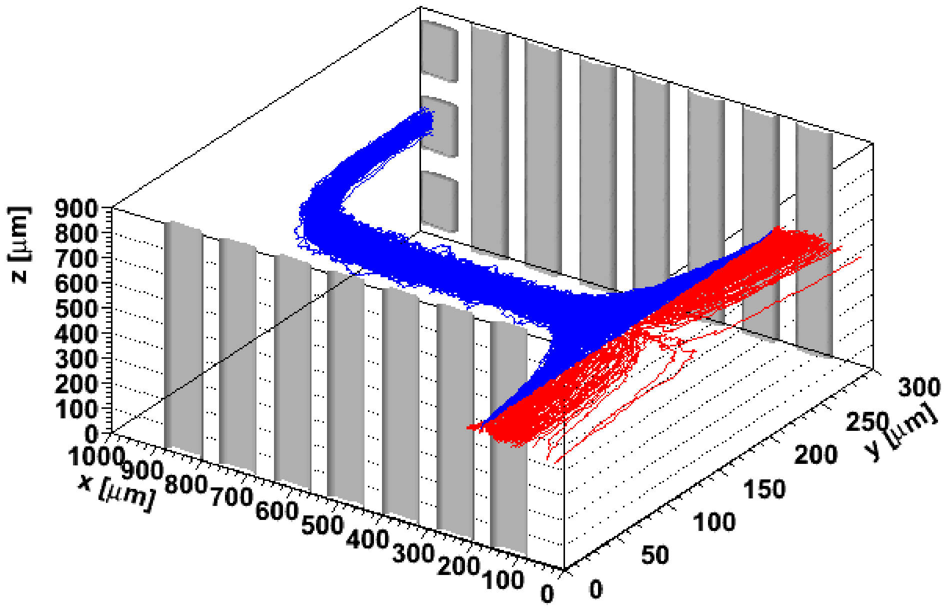
G. Kramberger, KDetSim – A root based 3D simulation tool for semiconductor detectors, SDD Fast Track Meeting, Trieste

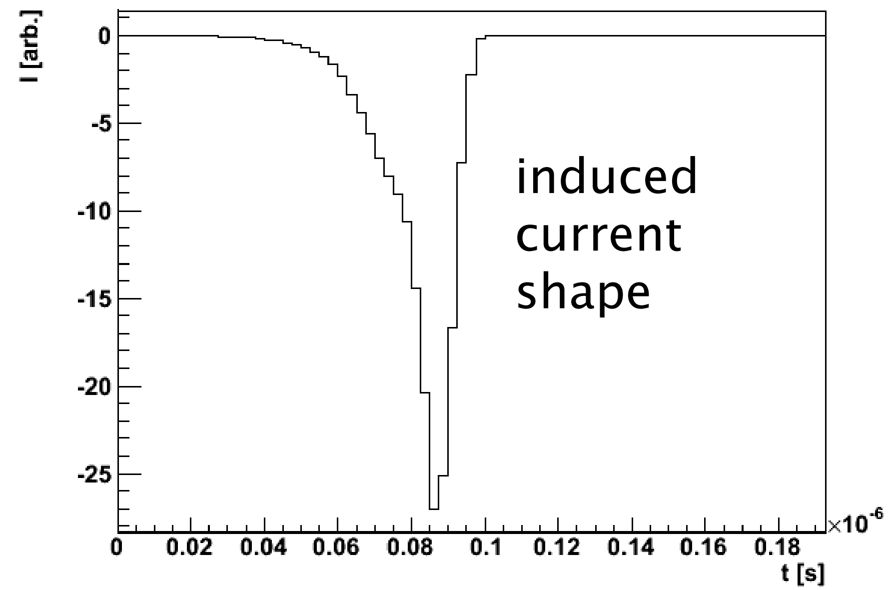
Silicon drift detectors –2D (inclined tracks)



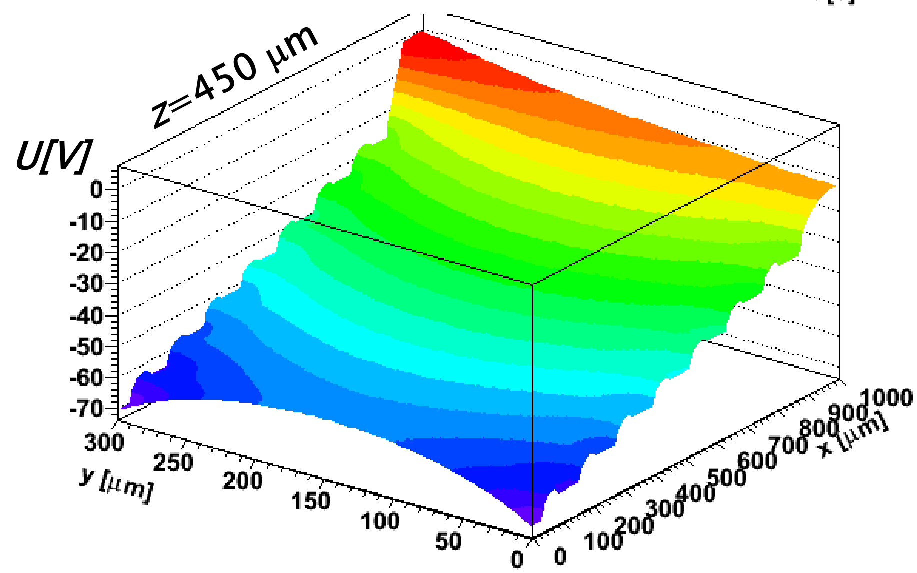
- Inclined tracks – wider pulses
- The incident angle can be estimated from the width of the pulse

Silicon drift detector – 3D



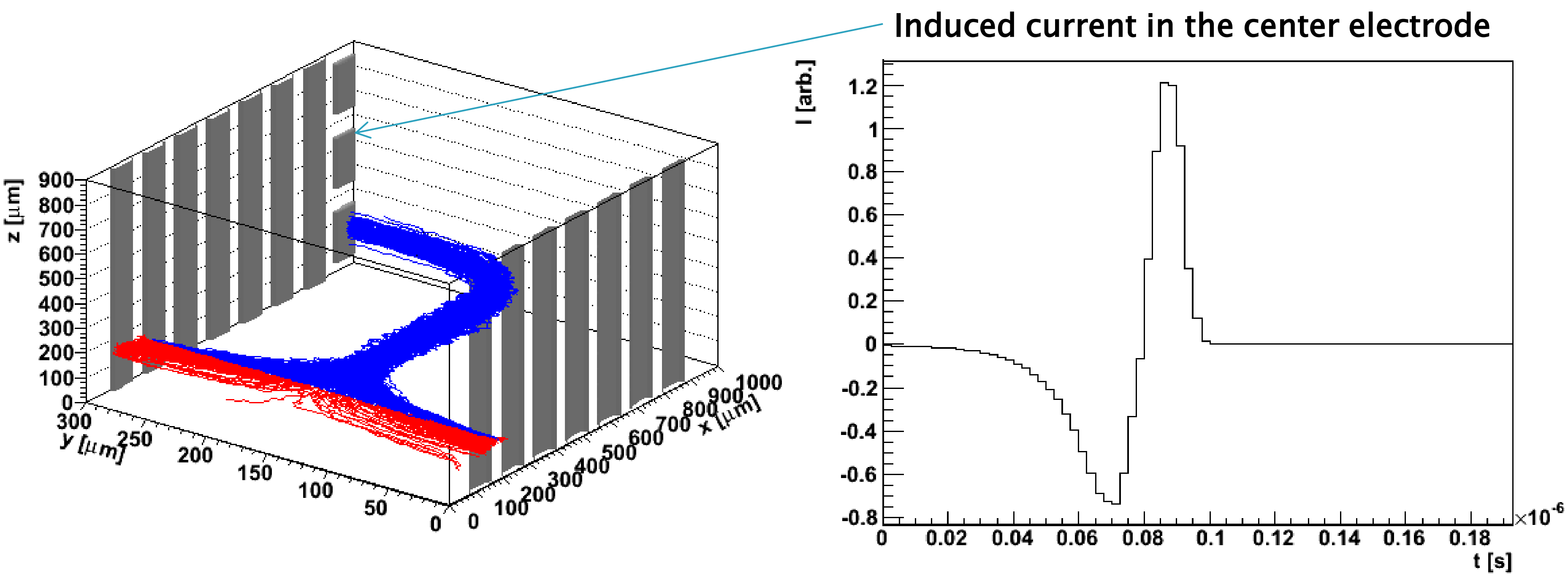


- 3D simulation of the same geometry
- A segment of three anodes
  - 300 µm pitch
  - 200 µm implant width
- Similar results as for 2D but some simulations require 3D simulation
- A map of potential along the drift plane at z=450 µm similar to 2D simulation.



G. Kramberger, KDetSim – A root based 3D simulation tool for semiconductor detectors, SDD Fast Track Meeting, Trieste

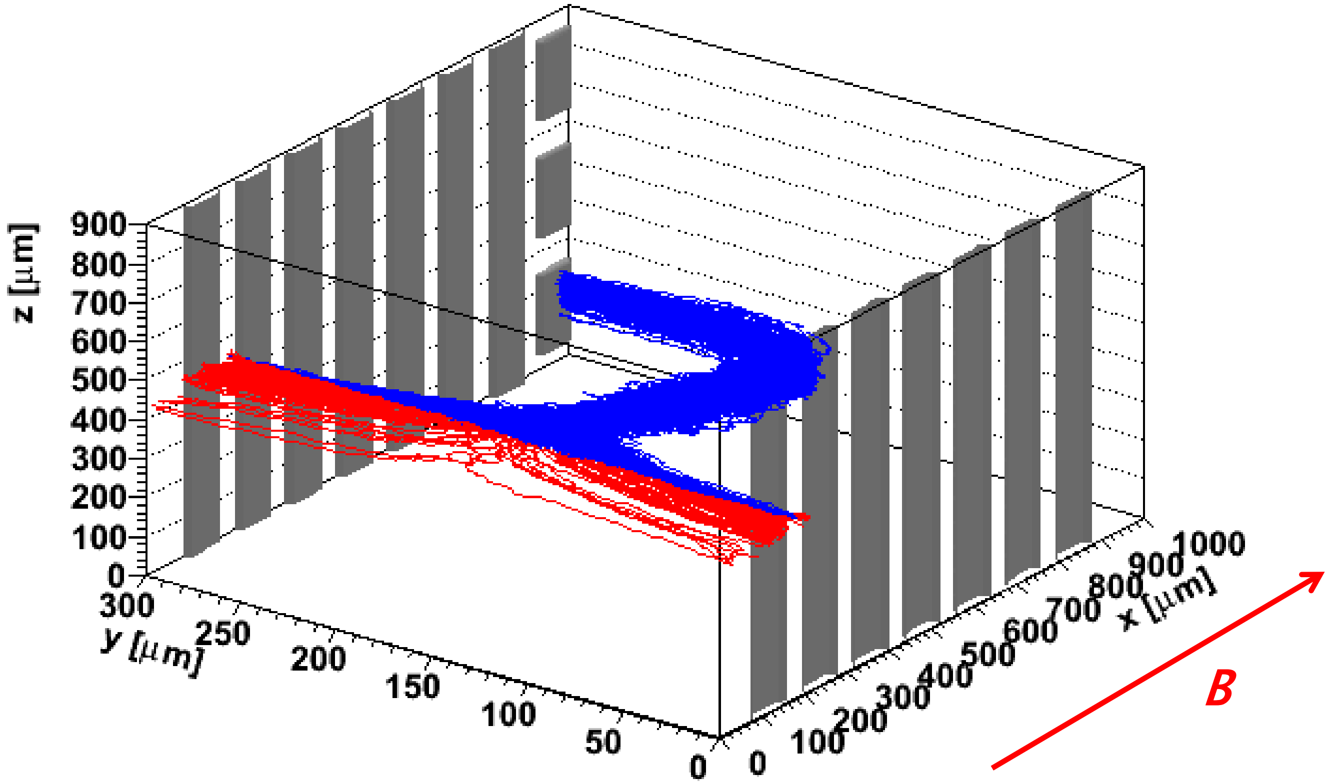
Silicon drift detector – 3D



- Example od induced current for electrons drifting to neighboring electrode (measurement electrode is the middle one)
- The current pulse is bipolar with vanishing charge (similar as in MWPC)
  - Trapping can influence that a lot – the carriers induced current while moving and not reaching the electrode

Silicon drift detector – 3D

- Magnetic field in x direction B_x=2T (the direction is chosen for illustrative reasons)
- Carriers drift to the neighboring anode, due to Lorentz force

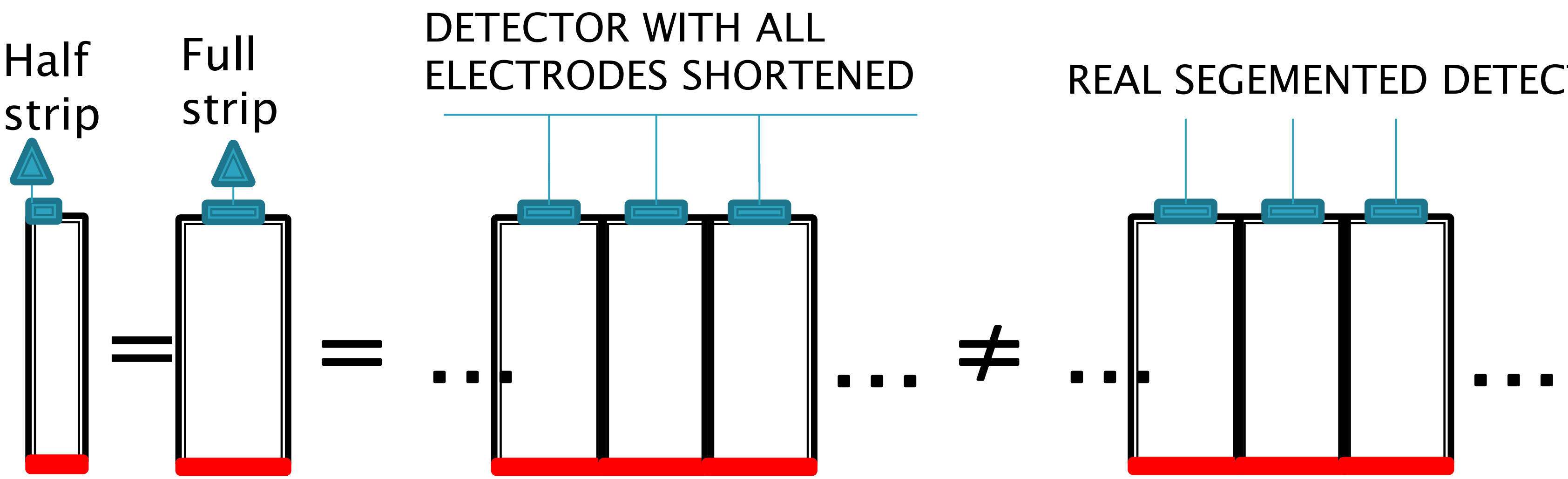


G. Kramberger, KDetSim – A root based 3D simulation tool for semiconductor detectors, SDD Fast Track Meeting, Trieste

Conclusions

- KDetSim is fast, lightweight and highly portable ROOT based code for simulation of signal in semiconductor detectors.
- It has been adopted also for simulation of SDD – both in 2D (large scale sensor) and 3D (smaller fractions of the sensor) enabling studies of
  - charge sharing/crosstalk
  - trapping – charge collecting efficiency
  - resolution for inclined tracks
  - …
- There are several tasks that are going on:
  - Import of E fields from TCAD into the simulator
  - Possible migration to FENICS meshing tools (not sure it required?)
  - Possible parallelization of the calculation (not sure if required?)
  - **Write a good manual (more users is certainly an incentive) – absolute necessity and priority**
- Anyone interested is welcome to join also not only for using but also to writing/debugging/improving code.

Choice of boundary conditions



All neighboring electrodes also with $U_w=1$ -> all strips shortened

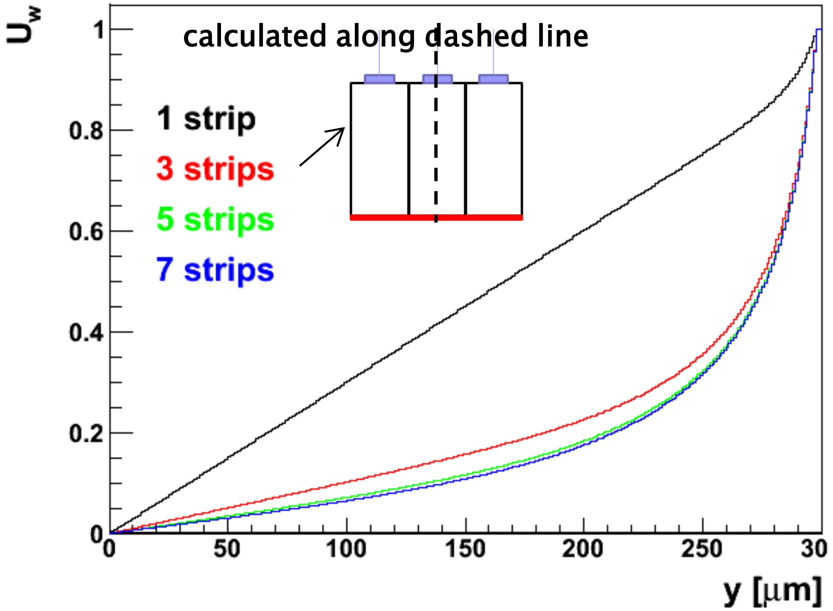
All neighboring electrodes at $U_w=0$ -> individual channels

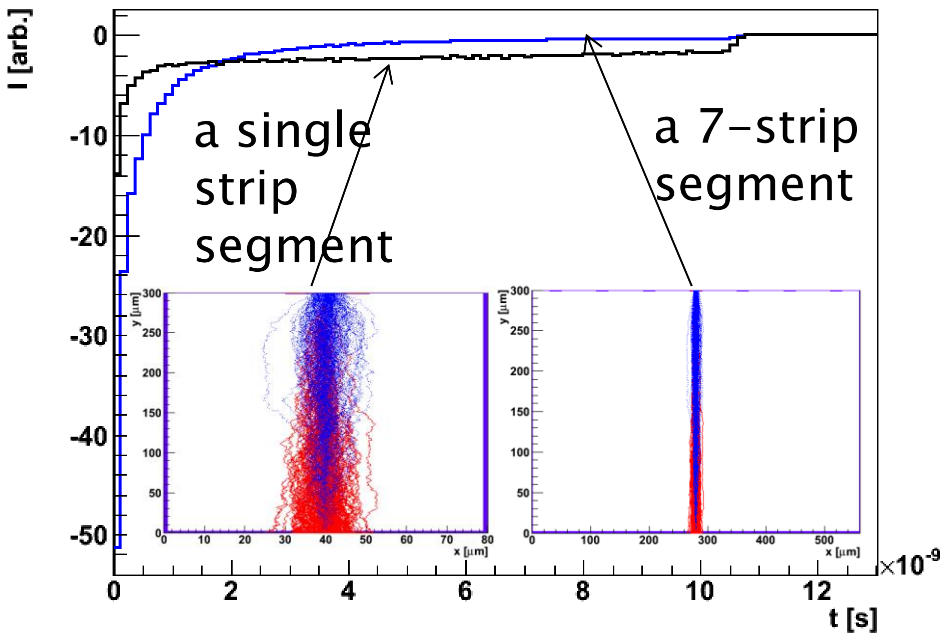
<u>Unlike for electric field</u> where for the symmetry reasons only a half strip can be used to calculate the field one should simulate a much larger section for the weighting field. Often not done in TCAD simulations.

A lot of effects in irradiated silicon detectors – such as e.g. "trapping induced charge sharing" can not be simulated without proper weighting field.

Example of simulated currents

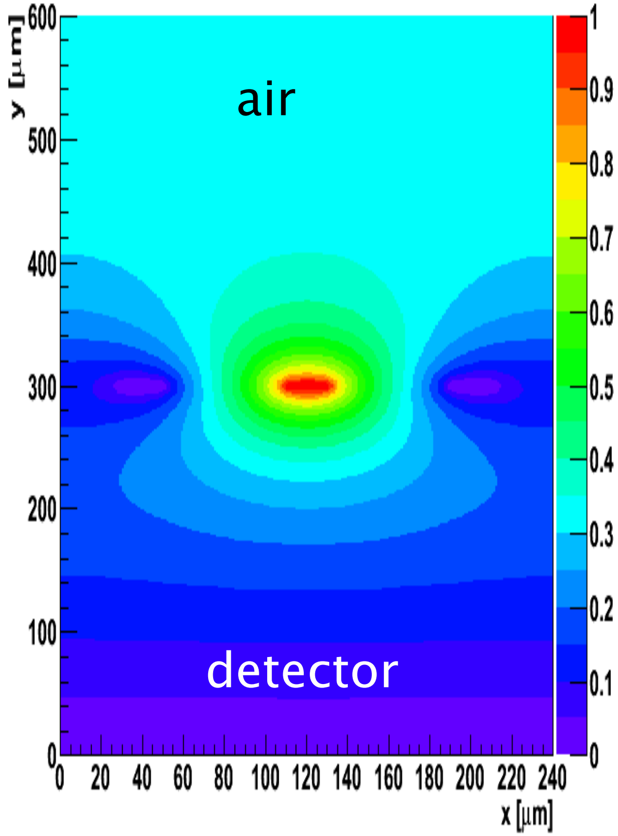
80 μm pitch, 20 μm width, 300 μm thick, V_{bias}=200 V, $N_{eff}=10^{12}$ cm^{-3}, n-on-p

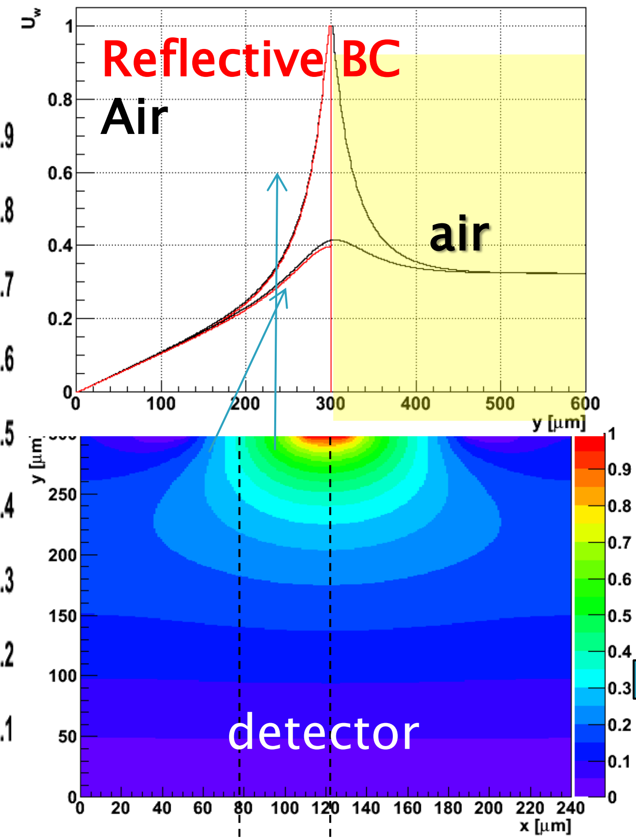




- It is clear that more strips should be taken into account: >3 should be enough
- any simulation tool that calculates the current induced in a sensors should include more strips than simply the minimum defined by symmetry!
  - Separate calculation of U_w and U is a good approach as it saves a lot of time, particularly for iterative approaches (modeling)

Choice of boundary conditions – U_w

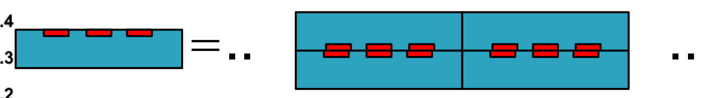




Reflective BC (von Neumann) at non-electrode surfaces

$$\nabla U_w = 0$$

No field lines escape the sensors – hence the structure is fully symmetrical in all directions



- For ATLAS geometry detectors the effect of reflective boundary conditions on the surface to weighting field is small – few % at most in the interstrip region. Should be looked individually for each structure.
- Same applys for electric field calculation.