Analysis memo
S-2S TOF detector

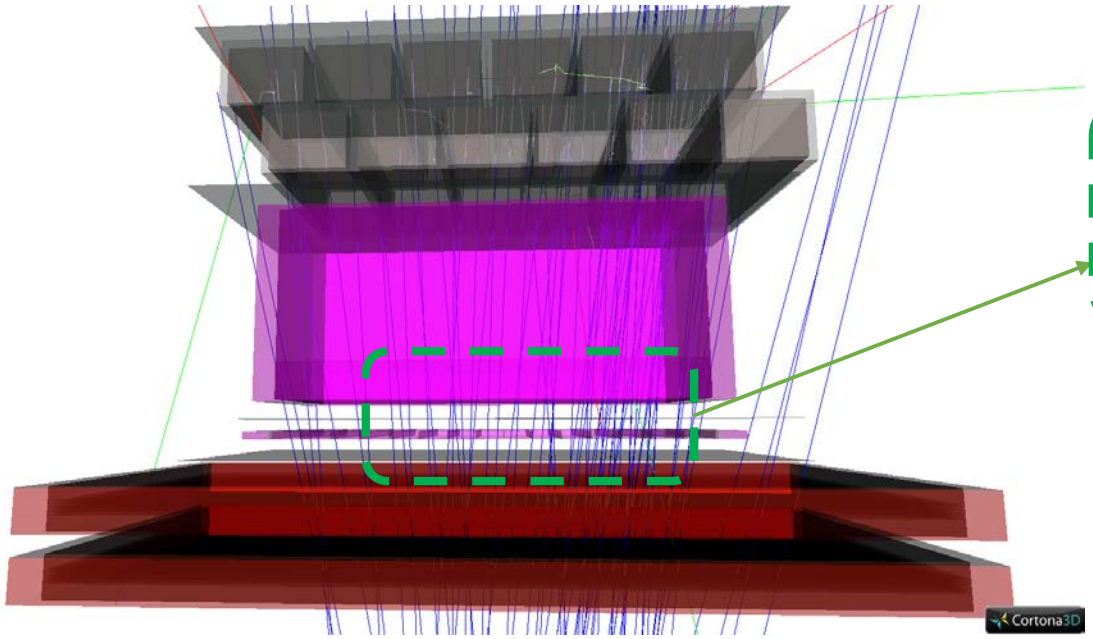
Toshiyuki Gogami

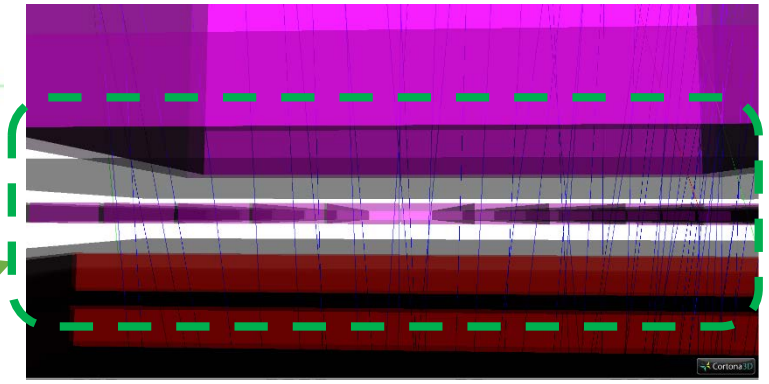
10Mar2015

Contents

S-2S TOF:
Plane → zigzag configuration

Gap problem of TOF detector

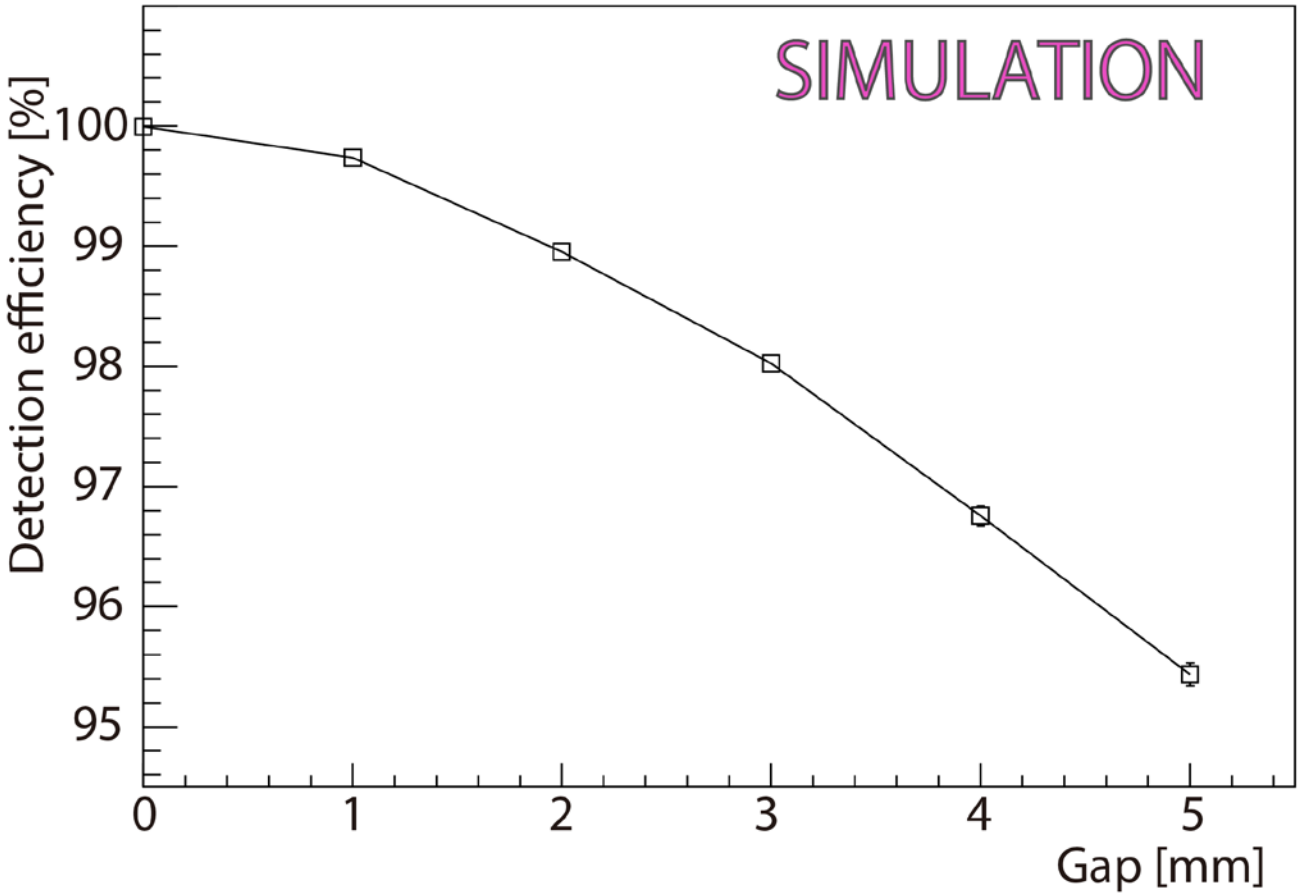




≥1 mm gaps between scintillators (due to black sheet)

➡ EVENT LOSS …

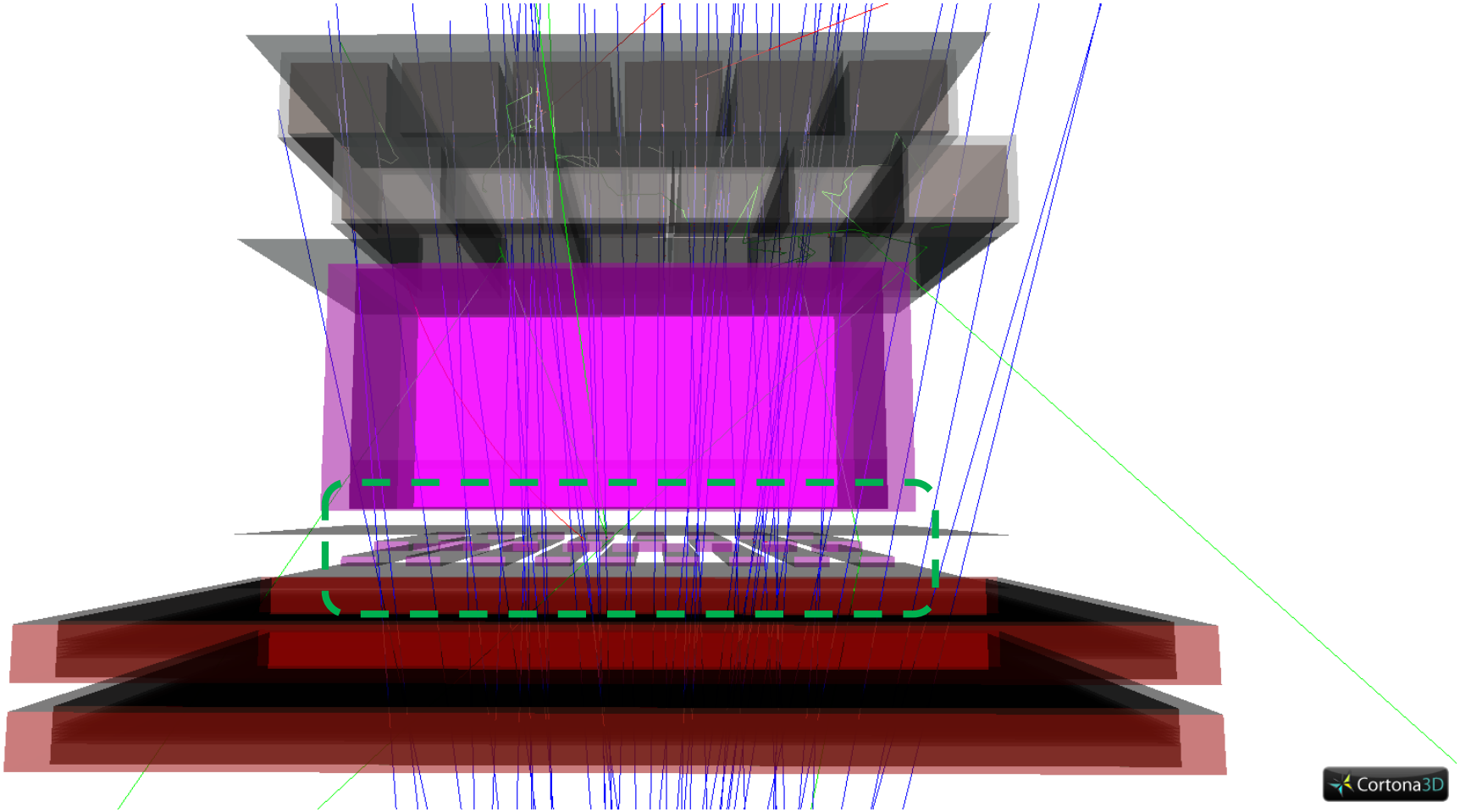
Gap vs. detection efficiency



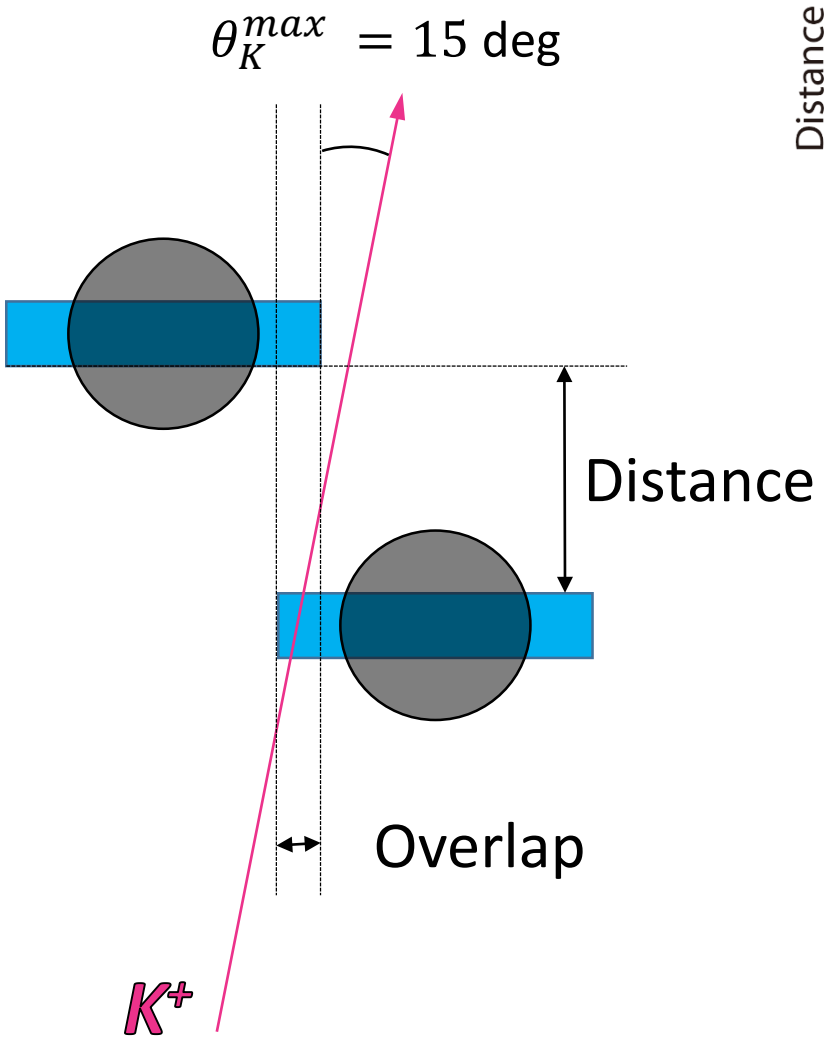
Only required geometrical hit in this plot.

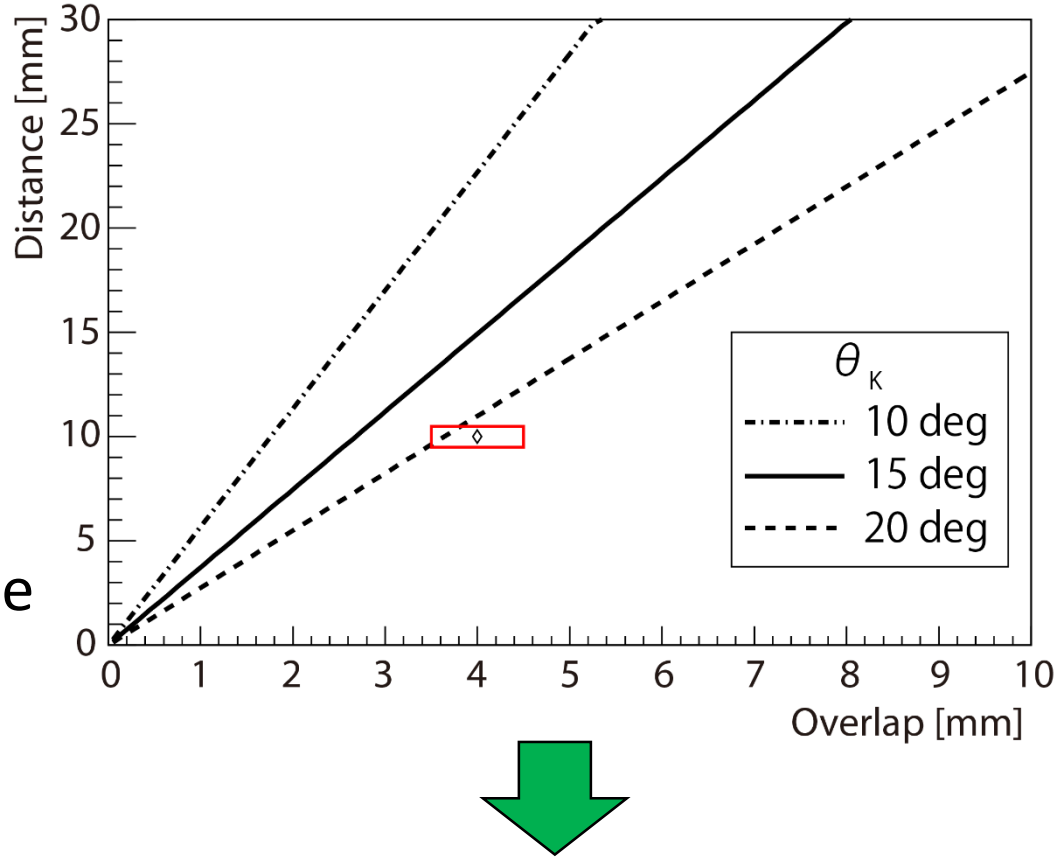
→ If ADC cut is applied, the efficiency is expected to be decreased more.

Plane → Zigzag configuration



Overlap and distance (TOF detector)

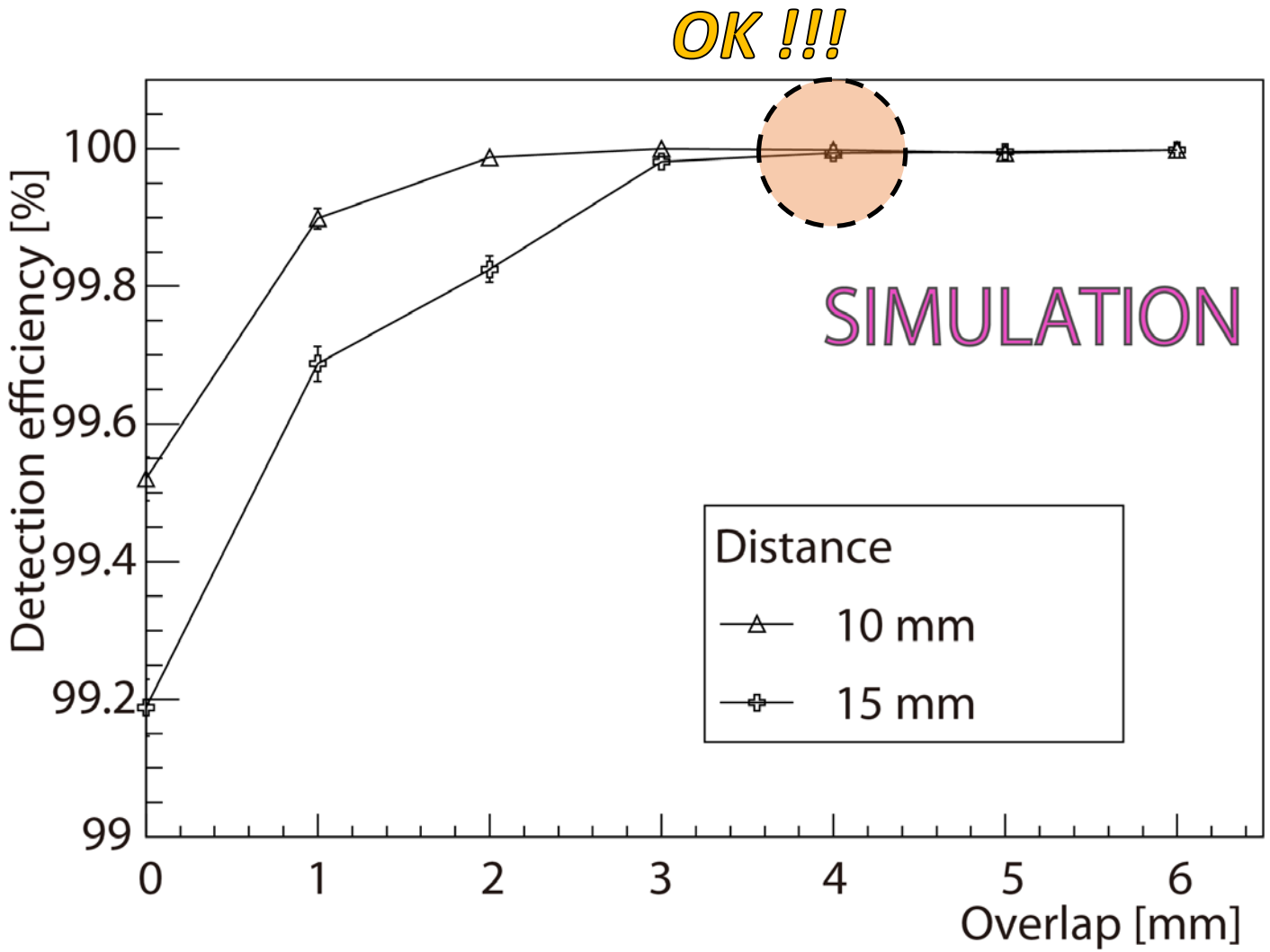




(Overlap, Distance) = (4 mm, 10 mm)

手計算では十分これでカバーできる

Overlap vs. detection efficiency

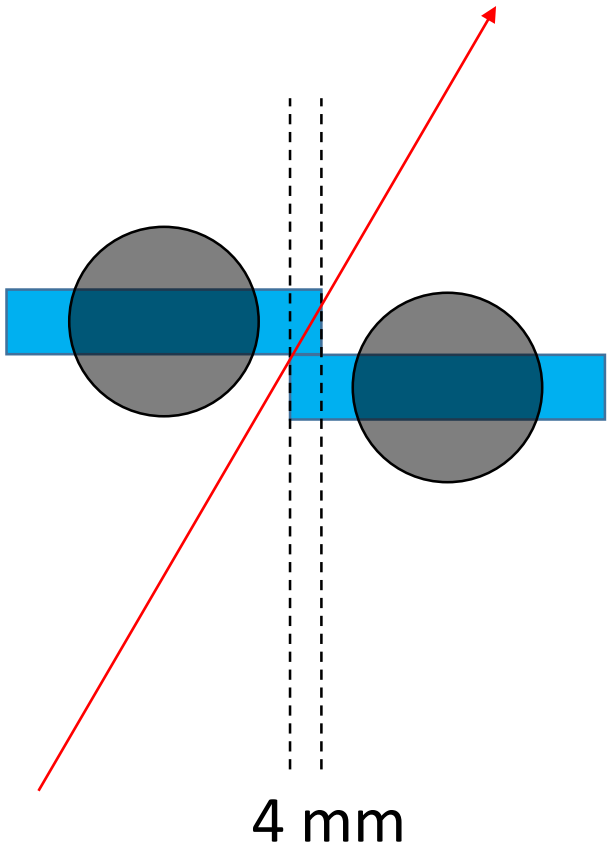


But,

Not necessary to have a distance between scintillators.

→ Then, the distance is determined to be zero.

Final design



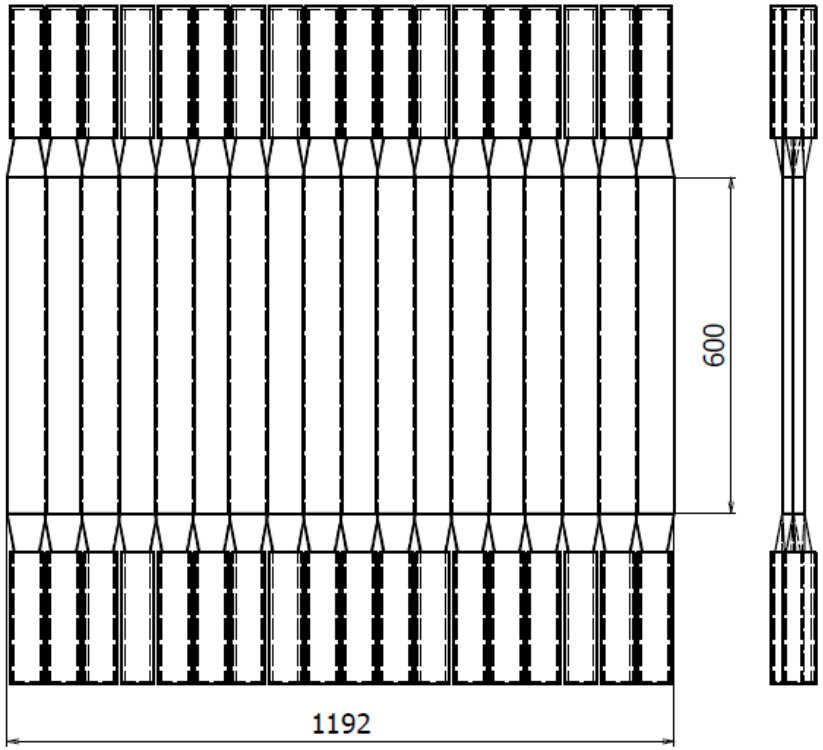
Final design →

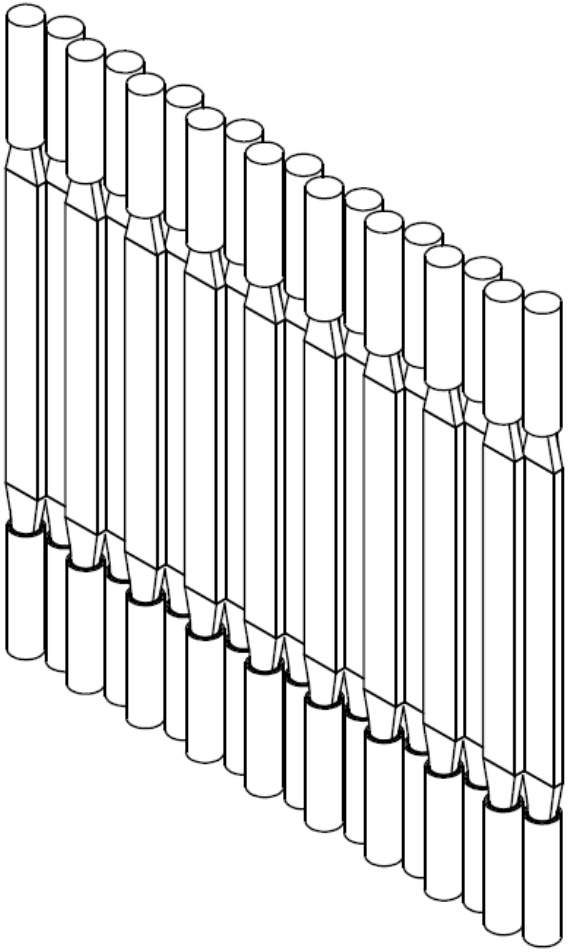
- Distance : 0 mm
- Overlap: 4 mm

$$R_{min} = \frac{4.0}{\sin(15°)} = 15.4\ mm$$

$$\frac{15.4}{20.0} = 0.77$$

Any particles with $x' \leq 15°$ pass through ≥ 0.77 segment

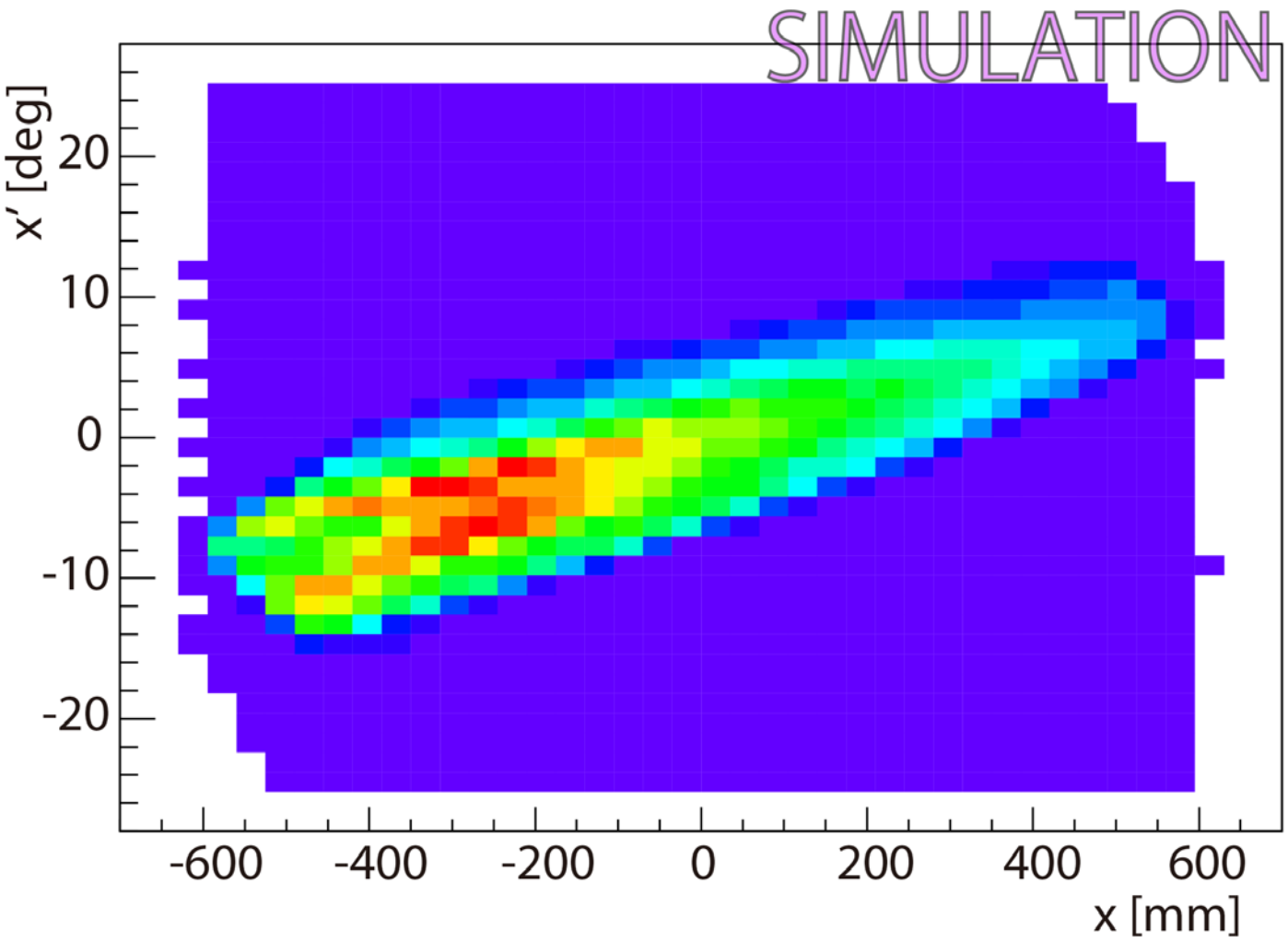




				Nuclear and Hadronic Physics Group, Kyoto University
		名前	日付	S-2S TOF detector
		T.Gogami	2015/02/18	
				tof_one_layer_stagger

Backup

x vs. x’ distribution



loss.cc

```
/*
 "loss.cc"

 Toshi Gogami , 28Nov2014
*/

void loss(){
 // ======== General conditions ================
 gROOT->SetStyle("Plain");
 gStyle->SetOptStat(0);

 // ======== Open ROOT file ====================
 //TFile* f1 = new TFile("tof_line_gap-0mm.root"); // gap-0mm (NO zig-zag)
 //TFile* f1 = new TFile("tof_line_gap-1mm.root"); // gap-2mm (NO zig-zag)
 //TFile* f1 = new TFile("tof_line_gap-2mm.root"); // gap-2mm (NO zig-zag)
 //TFile* f1 = new TFile("tof_line_gap-3mm.root"); // gap-3mm (NO zig-zag)
 //TFile* f1 = new TFile("tof_line_gap-4mm.root"); // gap-4mm (NO zig-zag)
 //TFile* f1 = new TFile("tof_line_gap-5mm.root"); // gap-4mm (NO zig-zag)
 //TFile* f1 = new TFile("tof_dist-10mm_over-0mm.root"); //
 //TFile* f1 = new TFile("tof_dist-10mm_over-1mm.root"); //
 //TFile* f1 = new TFile("tof_dist-10mm_over-2mm.root"); //
 //TFile* f1 = new TFile("tof_dist-10mm_over-3mm.root"); //
 //TFile* f1 = new TFile("tof_dist-10mm_over-4mm.root"); //
 //TFile* f1 = new TFile("tof_dist-10mm_over-5mm.root"); //
 //TFile* f1 = new TFile("tof_dist-10mm_over-6mm.root"); //
 //TFile* f1 = new TFile("tof_dist-15mm_over-0mm.root"); //
 //TFile* f1 = new TFile("tof_dist-15mm_over-1mm.root"); //
 //TFile* f1 = new TFile("tof_dist-15mm_over-2mm.root"); //
 //TFile* f1 = new TFile("tof_dist-15mm_over-3mm.root"); //
 //TFile* f1 = new TFile("tof_dist-15mm_over-4mm.root"); //
 //TFile* f1 = new TFile("tof_dist-15mm_over-5mm.root"); //
 TFile* f1 = new TFile("tof_dist-15mm_over-6mm.root"); //
 TTree* t1 = (TTree*)f1->Get("tree");

 //TCut cutcommon = "abs(vdx[7])<500.0 && abs(vdy[7])<200.0";
 //TCut cutcommon = "abs(vdx[7])<500.0 && abs(vdy[7])<200.0 && abs(vdx[8])<500.0 && abs(vdy[8])<200.0";
 TCut cutcommon = "abs(vdx[7])<500.0 && abs(vdy[7])<200.0 && WCTrig";
 //TCut cutcommon = "WCTrig";

 // ======== Create Histogram ================
 double xmin = -800.0 , xmax = 800.0;
 double ymin = -500.0 , ymax = 500.0;
 int xbin = 100;
 int ybin = 100;
 TH2F* hist1 = new TH2F("hist1","",xbin,xmin,xmax,ybin,ymin,ymax);

 TH2F* hist2 = (TH2F*)hist1->Clone("hist2");
 hist2->SetMarkerColor(2);
 hist2->SetMarkerStyle(1);

 t1->Project("hist1","vdy[7]:vdx[7]","VDTrig"&&cutcommon);
 t1->Project("hist2","vdy[7]:vdx[7]","TOFTrig"&&cutcommon);

 // ========= Draw histograms ==================
 TCanvas* c1 = new TCanvas("c1","c1");
 hist1->Draw();
 hist2->Draw("same");

 // ===== Comments =====
 double n1 = 0.0 , n2 = 0.0;
 double eff, effer;
 n1 = hist1->GetEntries();
 n2 = hist2->GetEntries();
 eff = n2/n1;
 //effer = eff * sqrt(1./n1 + 1./n2);
 if(n1!=0 && n2!=0){
  //effer = eff * sqrt(n1-n2)/(n1-n2);
  effer = eff * sqrt(n1-n2)/n1;
 }
 else effer = 0.0;

 cout << endl;
 cout << " n1=" << n1 << " +/- " << sqrt(n1) << endl;
 cout << " n2=" << n2 << " +/- " << sqrt(n2) << endl;
 cout << "  --> Efficiency = " << eff*100.0
    << " +/- " << effer*100 << " %" << endl;

 cout << " " <<  f1->GetName() << " " << eff*100.0
    << " " << effer*100 << endl;
}
```

plot.cc

```
/*
 plot.cc

 Toshiyuki Gogami , 6Mar2015
*/

void plot(){
 // ========== General conditions ============
 gROOT->SetStyle("Plain");
 gStyle->SetOptStat(0);

 // ========== Read data ====================
 ifstream* ifs = new ifstream("result_6Mar2015.dat");
 char tempc[800];
 const int n1=6;
 double x1[n1]={0.0 , 1.0 , 2.0 , 3.0 , 4.0 , 5.0};
 double xer1[n1];
 double y1[n1];
 double yer1[n1];

 const int n2=7;
 double x2[n2]={0.0 , 1.0 , 2.0 , 3.0 , 4.0 , 5.0, 6.0};
 double xer2[n2];
 double y2[n2];
 double yer2[n2];

 const int n3=7;
 double x3[n3]={0.0 , 1.0 , 2.0 , 3.0 , 4.0 , 5.0, 6.0};
 double xer3[n3];
 double y3[n3];
 double yer3[n3];

 for(int i=0 ; i<n1 ; i++){
  *ifs >> tempc >> y1[i] >> yer1[i];
  xer1[i] = 0.0;
 }

 for(int i=0 ; i<n2 ; i++){
  *ifs >> tempc >> y2[i] >> yer2[i];
  xer2[i] = 0.0;
 }
 for(int i=0 ; i<n3 ; i++){
  *ifs >> tempc >> y3[i] >> yer3[i];
  xer3[i] = 0.0;
 }
 ifs->close();

 // ========= Create empty histograms ==========
 TH2F* h_emp = new TH2F("h_emp","",
                                                 100,0.0,5.5,
                                                 100,94.5,101.);
 h_emp->GetXaxis()->SetTitle("Gap [mm]");
 h_emp->GetYaxis()->SetTitle("Detection efficiency [%]");

 TH2F* h_emp2 = new TH2F("h_emp2","",
                                                 100,0.0,6.5,
                                                 100,99.0,100.1);
 h_emp2->GetXaxis()->SetTitle("Overlap [mm]");
 h_emp2->GetYaxis()->SetTitle("Detection efficiency [%]");


 // ========== Create graphs ============
 TGraphErrors* gr1 = new TGraphErrors(n1,x1,y1,xer1,yer1);
 gr1->SetName("gr1");
 gr1->GetXaxis()->SetTitle("gap [mm]");
 gr1->GetYaxis()->SetTitle("Detection efficiency [%]");

 TGraphErrors* gr2 = new TGraphErrors(n2,x2,y2,xer2,yer2);
 gr2->SetName("gr2");
 gr2->GetXaxis()->SetTitle("Overlap [mm]");
 gr2->GetYaxis()->SetTitle("Detection efficiency [%]");

 TGraphErrors* gr3 = new TGraphErrors(n3,x3,y3,xer3,yer3);
 gr3->SetName("gr3");
 gr3->GetXaxis()->SetTitle("Overlap [mm]");
 gr3->GetYaxis()->SetTitle("Detection efficiency [%]");

 // ======== Legend ========================
 TLegend* dragon_leg = new TLegend(0.5,0.2,0.8,0.45,"Distance");
 dragon_leg->AddEntry(gr2,"10 mm","pl");
 dragon_leg->AddEntry(gr3,"15 mm","pl");
 dragon_leg->SetFillStyle(0);

 // ========= Draw graphs ==================
 TCanvas* c1 = new TCanvas("c1","c1");
 h_emp->Draw();
 gr1->Draw("l*same");

 TCanvas* c2 = new TCanvas("c2","c2");
 h_emp2->Draw();
 gr2->Draw("l*same");
 gr3->Draw("l*same");
 dragon_leg->Draw();

 // ======== Graph options =================
 gr1->SetMarkerStyle(25);
 gr2->SetMarkerStyle(26);
 gr3->SetMarkerStyle(28);

 // ======= Print ========
 //c1->Print("tofeff_gap_plane.eps","eps");
 //c1->Print("tofeff_overlap_zigzag.eps","eps");


}
```