# Table of Contents

- Code Checkout.................................................................................................................................1
- Background Information on bTagging..............................................................................................2
- Primary Vertex Reconstruction.........................................................................................................3
- Impact Parameter Tagging....................................................................................................................6
- Secondary Vertex Reconstruction (Time Permitting)...........................................................................9
Code Checkout

It is assumed you are running on the cmslpc cluster at FNAL and that you know some basic C++/ROOT. A working grid certificate is also assumed for the third section of the tutorial. To get the code for this tutorial, follow the command-line instructions below.

```
source /uscms1/prod/sw/cms/cshrc prod
mkdir ejterm2010
cd ejterm2010
cmsrel CMSSW_3_3_6_patch3
cd CMSSW_3_3_6_patch3/src
cmsenv
cmscvsroot CMSSW
cvs login (For the password, enter 98passwd)
cvs co -d LPCbIDUserCode/kellerjd/LPCbID
scramv1 b -j4
```

Once the compilation has completed (no errors), you should be ready to go with the tutorial.
Background Information on bTagging

Jets are a very common occurrence at high energy colliders (especially hadron colliders). Most jets are created from light partons (up, down, strange quarks and gluons, or udsg). Charm (c) and bottom (b) quarks also create jets, but at lower rates. b-Jets are especially useful because they are signs of interesting physics (such as top quarks, higgs bosons, supersymmetry, and other new physics phenomena). Thus, it is very useful to differentiate between b-Jets and other jets. Luckily, b-quarks are distinct, having a longer lifetime than c-quarks or light partons. b-Tagging, the identifying of b-Jets, uses this property as an advantage by looking for tracks which originate from a vertex away from the primary vertex (a secondary vertex), or by looking for tracks with a large distance of closest approach (dca). These are signs of b-Jets in an event. This tutorial will walk you through some of the important aspects of b-Tagging (Question: Given that the b-quark has a long lifetime, would you expect a b-hadron to decay by a strong, weak, or electromagnetic interaction? Answer: Typically, particles with long lifetimes decay weakly. So, it should be expected for the b-hadron to decay weakly.).
Primary Vertex Reconstruction

In order to reconstruct secondary vertices, it is important to reconstruct the primary vertex. Primary vertex reconstruction in CMS is performed by clustering tracks together and performing fits to determine the likelihood these tracks originated from a common vertex. The reconstructed vertex with the largest total pT is considered the primary vertex. Many tracks should originate from this point. So, this assumption seems reasonable. The first part of this tutorial will look at properties of primary vertex reconstruction.

Exercise 1: Plot primary vertex variables using CMS.MinBias MC

The code that you checked out and compiled previously contains a class called CMS.PrimaryVertexAnalyzer, which obtains data describing the primary vertex and the beamspot. To run this on MC, follow the instructions below.

1. Move into the directory which contains the CMS.PrimaryVertexAnalyzer configuration file:

   cd LPCbID/EJTERMExercises/test

2. Open the file primaryVertexAnalyzer_cfg.py using your favorite test editor.

3. Go to the DBS data discovery page and look for the 900 GeV CMS.MinBias data sample.
   ♦ The CMSSW release is 3_3_6_patch3.
   ♦ The primary dataset is CMS.MinBias
   ♦ The dataset has the name /CMS.MinBias/Summer09-D6T_STARTUP3X_V8I_900GeV_ReReco_336p3_v1/GEN-SIM-RECO

4. Choose a handful of files to run the CMS.PrimaryVertexAnalyzer on, modify the primaryVertexAnalyzer_cfg.py file, and perform a cmsRun over the files.

5. From the resulting TTree, make some plots for real primary vertices
   ♦ Open the resulting TFile using

     root -l pvAnalyzer.root

   Examine the contents of the TTree using

     pvTree->Print()

   ♦ Make a 3D plot of the primary vertex position.
   Hint

     In ROOT, it is possible to make a 3D plot of the primary vertex position with the following line in interactive ROOT.

     pvTree->Draw("pvX:pvY:pvZ", "!pvIsFake")

   The first set of quotation marks indicate the variables you wish to draw (the colon allows you to plot the variables on different axes). The second set of quotation marks is a cut implemented on the TTree data. Here we want to draw only real (not fake) primary vertices.

   ♦ Plot the x position versus the y position, and x position versus the z position of the primary vertex.
   ♦ Based on these plots, how would you describe the primary vertex shape/symmetry.

6. Make plots comparing the primary vertex to the beamspot. For instance, plot pvX - bsX0, pvY - bsY0, and pvZ - bsZ0 for real primary vertices only.
The primary vertex has a cigar-shaped distribution, with the distribution in x and y being symmetric.

1. What are the characteristics of a fake primary vertex? When do you think a fake primary vertex would be found? A fake primary vertex is the same as the beam spot. A fake primary vertex will be found when there are not enough quality tracks to reconstruct a vertex, or no sensible vertex is reconstructed.

Exercise 2: Repeat Exercise 1 with real CMS.MinBias data

The data can again be found using DBS. The CMSSW release is 3_3_6_patch3 and the primary dataset is MinimumBias. A suggested dataset is /MinimumBias/BeamCommissioning09-BSCNOBEAMHALO-Dec19thSkim_336p3_v1/RAW-RECO. Pick approximately 10-15 files to run on.
Impact Parameter Tagging

Now that we have examined primary vertex reconstruction, let's move on to looking at the impact parameter of tracks in jets. First, we will examine the effect jet flavor has on impact parameter.

Exercise 3: Use MC to examine impact parameter vs. jet flavor

1. Within the test directory resides a file called impactParameterAnalyzerMC_cfg.py. Open this file with your favorite text editor.
2. Use DBS to find some Release Validation TTbar samples to run the analyzer on.
   - CMSSW Release: 3_3_6
   - primary dataset is RelValTTbar
   - dataset is /RelValTTbar/CMSSW_3_3_6-MC_3XY_V9A-v1/GEN-SIM-RECO
   - Run over the whole dataset.
3. Use the resulting TTree to plot the impact parameter significance (the value divided by its uncertainty) of the first three tracks of the jet for both the 2D (the transverse plane) and 3D impact parameters with respect to the primary vertex. It is important to include the cut
   \[
   \text{abs}(\text{ipXdTrkn}/\text{ipXdErrTrkn}) < 30.0
   \]
   where X = 2,3 (the dimension of the impact parameter) and n = 1,2,3 (the track number) to get a closer look at the region of interest. For instance, your draw commands in ROOT should look like
   ```
   \text{ipTree->Draw("ip2dTrk1/ip2dErrTrk1", "abs(ip2dTrk1/ip2dErrTrk1) < 30.0");}
   \text{ipTree->Draw("ip2dTrk2/ip2dErrTrk2", "abs(ip2dTrk2/ip2dErrTrk2) < 30.0");}
   ...
   ```
   
   and so forth.

Plot the 3D impact parameter significance of the 2nd track for the different jet flavors and compare them.
   - udsg-jets have flavor codes 1-3 and 21, c-jets have flavor code 4, b-jets have flavor code 5.
Hint

The TTree::Project() function is the easiest way to do this. It allows you to project TTree data into an already created histogram. Below is an example of how to do this for b-jets (jetFlavor == 5):

```c
TH1F* b3DIP2 = new TH1F("b3DIP2", "The 3D Impact Parameter Significance of Trk 2 for Different Jet Flavors", 100, -10.0, 30.0);
ipTree->Project("b3DIP2", "ip3dTrk2/ip3dErrTrk2", "jetFlavor == 5");
```

Create two other histograms for c-Jets (called c3DIP2), and for udsg-Jets (called udsg3DIP2).

♦ It may be helpful to normalize the histograms, and overlay them on one canvas.

Hint

The TH1::Scale() and TH1::Integral() functions are useful for this. It is also helpful to make the histograms have different line colors with the SetLineColor() function. For instance, the line color for the histogram in the above example can be set to red and the histogram normalized by

```c
b3DIP2->SetLineColor(kRed);
b3DIP2->Scale(1.0/b3DIP2->Integral());
```

Other options for the SetLineColor() function are kBlue and kGreen, or any number in the color palette. Multiple histograms can be drawn on the same canvas using the THStack class, as shown below.

```c
THStack* stack = new THStack("stack", "The 3D Impact Parameter Significance of Trk 2 for Different Jet Flavors");
stack->Add(b3DIP2);
stack->Add(c3DIP2);
stack->Add(udsg3DIP2);
stack->Draw("nostack")
```

The 3D Impact Parameter Significance of Trk 2 for Different Jet Flavors

1. Does the impact parameter significance plot confirm what you have been told about the b-quark lifetime? Particles with longer lifetimes are expected to have larger impact parameters. This confirms
the long lifetime of b-hadrons.

2. Based on the distribution, what can you say about the lifetime of c-quarks? c-hadrons also have a long lifetime, though not as long as b-hadrons.

3. What does the shape of the distribution for udsg-jets tell you about them? Light hadrons decay very quickly near the primary vertex.

Exercise 4: Repeat Exercise 3 with the MinimumBias Data set in Exercise 2

1. Use the impactParameterAnalyzerData_cfg.py for this. Just run over the 10-15 files used in the previous exercise. Skip the jet flavor stuff as well (unfortunately, data has no monte carlo truth:).

Based on the shapes of the impact parameter significance distributions, what would you say is the predominant flavor of the jets in this data sample?

Based on the shape of the distributions, udsg-Jets are the predominant jet flavors in the Minimum Bias dataset.
Secondary Vertex Reconstruction (Time Permitting)

Apart from using the impact parameter of tracks within a jet, it is possible to take a more active approach and attempt to reconstruct the decay vertex of the b-hadron. Remember the b secondary vertex is expected to be displaced some distance away from the primary vertex. However, b-hadrons are not the only particles with a displaced secondary vertex. The K-Short meson, Lambda baryon, and even photon conversion to electrons also produce large displaced vertices. Such particles occur predominately in light-flavor jets, and are thus a major background to b-Tagging. This section of the tutorial will use secondary vertex information to reconstruct K-Shorts.

Exercise 5: Use secondary vertex information to reconstruct K-Shorts in MinimumBias MC

1. Within the test directory, there is a cfg file called secondaryVertexAnalyzer_CMS.MinBias_900GeV_cfg.py. Create a CRAB file which uses this cfg file to run over the whole set of MinimumBias MC.
   • To access the CRAB environment on cmslpc, do the following.
     
     ```bash
     source /uscmst1/prod/grid/CRAB/crab.csh
     ```
   • There are a couple of places you can find example crab.cfg files to use as a guide. One is located at
     
     ```bash
     $CRABDIR/python/crab.cfg
     ```
   The other can be found on the DBS page under any of the dataset listings. All of the datasets you will be running on should be stored at FNAL. So, to speed up the running of jobs, modify the scheduler in the crab.cfg file with
     
     ```bash
     scheduler = condor
     ```
   to have the jobs run over the cmslpc batch system. Also, to get your output from the CRAB jobs, make sure you set
     
     ```bash
     output_file = svAnalyzer.root
     ```
   • To run your crab jobs, do
     
     ```bash
     crab -create
     crab -submit
     ```
   • To check on the status of your crab jobs, do
     
     ```bash
     crab -status
     ```
   • To retrieve output from your crab jobs, do
     
     ```bash
     crab -getoutput
     ```

2. The resulting output files contain a TTREE which has very basic information about the secondary vertex (vertex chi-squared, number of tracks, pT of each track, etc.). Use this information to reconstruct the K-Shorts by creating a K-Short mass peak (Hint: how do K-Shorts predominantly decay? Draw the Feynman diagram for this. How can the info contained in these files be used for this? The K-Short decays predominately to oppositely charged pions. Look for a vertex which has two
oppositely charged tracks, and assume they are pions.

- Your output will be contained in TTrees spread out over several output files. To combine this output, there is a ROOT macro called svTree_merger.C in the LPCbID/EJTERMExercises/test directory that will do this for you. Copy this file to the same directory as your CRAB working directory and run it using

```c
root -l svTree_merger.C
```

and follow the instructions as they appear. When finished type .q to exit ROOT.

- To help get you started, there is a ROOT macro in LPCbID/EJTERMExercises/test called svTree_Looper.C, which contains some basic code that will loop over the entries of a TTree in a given file. There are some suggestions in this macro on how to use this information to create a Lorentz vector and how to use this vector to reconstruct particles. However, it is up to you to fill in the code to reconstruct the K-Shorts, create the histograms of the invariant mass, and fit the histogram with a function. Once finished, run the macro by performing

```c
root -l .L svTree_Looper.C+; svTree_Looper("filename.root");
```

3. Fit your invariant mass peak with a functional model, taking the background into account in the model. What value does your model give for the K-Short mass? How well does this mass agree with the accepted value from the PDG? 

![Ditrack Mass](image)

<table>
<thead>
<tr>
<th>Entries</th>
<th>4029</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.5458</td>
</tr>
<tr>
<td>RMS</td>
<td>0.1187</td>
</tr>
<tr>
<td>Underflow</td>
<td>95</td>
</tr>
<tr>
<td>Overflow</td>
<td>1111</td>
</tr>
<tr>
<td>Integral</td>
<td>2823</td>
</tr>
<tr>
<td>$\chi^2 / \text{ndf}$</td>
<td>147.3 / 95</td>
</tr>
<tr>
<td>Prob</td>
<td>0.0004664</td>
</tr>
<tr>
<td>Peak Factor</td>
<td>5.53 ± 0.19</td>
</tr>
<tr>
<td>Mass</td>
<td>0.4985 ± 0.0002</td>
</tr>
<tr>
<td>Width</td>
<td>0.00744 ± 0.00040</td>
</tr>
<tr>
<td>BKG Slope</td>
<td>23.4 ± 2.8</td>
</tr>
<tr>
<td>BKG Intercept</td>
<td>2.883 ± 1.531</td>
</tr>
</tbody>
</table>

1. Based on your functional model, estimate how many K-Shorts were reconstructed. Using the integral of the histogram in the range of the peak and the integral of the linear background function, approximately 1300 K-Shorts are present.

Exercise 6: Repeat Exercise 5 above for K-Shorts in MinimumBias Data
Approximately 340 K-Shorts were reconstructed with this method.

