# Cone vs. $k_t$, FastJet, and UE/MB subtraction in $k_t$ and Cambridge/Aachen clustering

Matteo Cacciari and Gavin Salam

LPTHE, Universities of Paris VI and VII and CNRS

MCWS, Frascati, 23 October 2006

# Outline

- Brief review of clustering algorithms: Cone vs. $k_t$
  - Overview of **iterative cone** algorithms (& what's wrong with them)
  - **Clustering** algorithms
    - How they work
    - Where they've been criticised (speed, underlying-event (UE) sensitivity)
- How to solve the speed problem.
  Fast algorithm for $k_t$ clustering: FastJet

  A brief presentation of the $\mathcal{O}(N \ln N)$ algorithm
- Underlying event and minimum bias/pile-up subtraction using FastJet and jet areas

  Some preliminary plots and results

# Outline

- Brief review of clustering algorithms: Cone vs. $k_t$
  - Overview of **iterative cone** algorithms (& what's wrong with them)
  - **Clustering** algorithms
    - How they work
    - Where they've been criticised (speed, underlying-event (UE) sensitivity)
- How to solve the speed problem.
  Fast algorithm for $k_t$ clustering: `FastJet`

  A brief presentation of the $\mathcal{O}(N \ln N)$ algorithm
- Underlying event and minimum bias/pile-up subtraction using `FastJet` and jet areas

  Some preliminary plots and results

# Outline

- Brief review of clustering algorithms: Cone vs. $k_t$
  - Overview of **iterative cone** algorithms (& what's wrong with them)
  - **Clustering** algorithms
    - How they work
    - Where they've been criticised (speed, underlying-event (UE) sensitivity)

- How to solve the speed problem.
  Fast algorithm for $k_t$ clustering: `FastJet`

  A brief presentation of the $\mathcal{O}(N \ln N)$ algorithm

- Underlying event and minimum bias/pile-up subtraction using `FastJet` and jet areas

  Some preliminary plots and results

# Jet algorithm requirements

What is **needed** of a jet algorithm

- Must be infrared and collinear (IRC) safe

  soft emissions shouldn't change jets

  collinear splitting shouldn't change jets
- Must be identical procedure at parton level, hadron-level

  So that theory calculations can be compared to experimental measurements

What is *nice* for a jet algorithm

- Shouldn't be too sensitive to hadronisation, underlying event, pileup

  Because we can only barely model them
- Should be realistically applicable at detector level

  Not too slow, not too complex to correct
- Should behave 'sensibly'

  e.g. don't want it to spuriously ignore large energy deposits

# Jet algorithm types

## Mainstream jet-algorithms

- Iterative cone algorithms (JetClu, ILCA/Midpoint, . . . )

  Searches for cones centred on regions of energy flow

  Dominant at hadron colliders

- Sequential recombination algorithms ($k_t$, Cambridge/Aachen, Jade)

  Recombine closest pair of particles, next closest, etc.

  Dominant at $e^+e^-$ and *ep* colliders

## Other approaches

- 'Optimal Jet Finder', Deterministic Annealing

  Fit jet axes (and #) so as to minimise a weight function

  [forms of '*k*-means' clustering]

- . . .

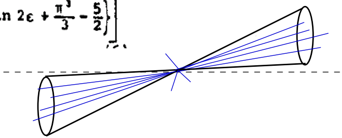
As LHC startup approaches it's important for the choice of jet algorithm to be well-motivated.

# Cone Origins

First 'cone algorithm' dates back to Sterman and Weinberg (1977) — the original infrared-safe cross section:

To study jets, we consider the partial cross section $\sigma(E,\theta,\Omega,\varepsilon,\delta)$ for $e^+e^-$ hadron production events, in which all but a fraction $\varepsilon \ll 1$ of the total $e^+e^-$ energy $E$ is emitted within some pair of oppositely directed cones of half-angle $\delta \ll 1$, lying within two fixed cones of solid angle $\Omega$ (with $\pi\delta^2 \ll \Omega \ll 1$) at an angle $\theta$ to the $e^+e^-$ beam line. We expect this to be measur-

$$\sigma(E,\theta,\Omega,\varepsilon,\delta) = (d\sigma/d\Omega)_0\,\Omega\left[1 - (g_E^2/3\pi^2)\left\{3\ln\delta + 4\ln\delta\,\ln 2\varepsilon + \frac{\pi^2}{3} - \frac{5}{2}\right\}\right]$$



# Modern cones address various issues

Where do you put the cones?

- Place a cone at some trial location
- Sum four-momenta of particles in cone – find corresponding axis
- Use that axis as a new trial location, and *iterate*
- Stop when you reach a stable axis [or when you get bored]

What are the initial trial locations?

- 'Seedless' — i.e. everywhere But too slow on computer
- Use locations with energy flow above some threshold as *seeds*

Issue: is seed threshold = parton energy, hadron energy (collinear unsafe)?
Or calorimeter tower energy (experiment and $\eta$-dependent)?

# ILCA

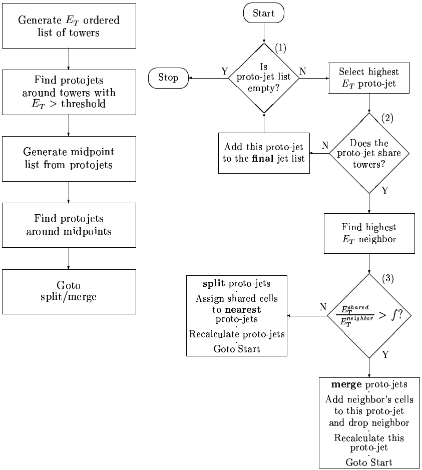
Consideration of many of these issues led to the formulation of the *Improved Legacy Cone Algorithm* (ILCA), a.k.a. *Midpoint* algorithm.

hep-ex/0005012

Quite complex and has several parameters:

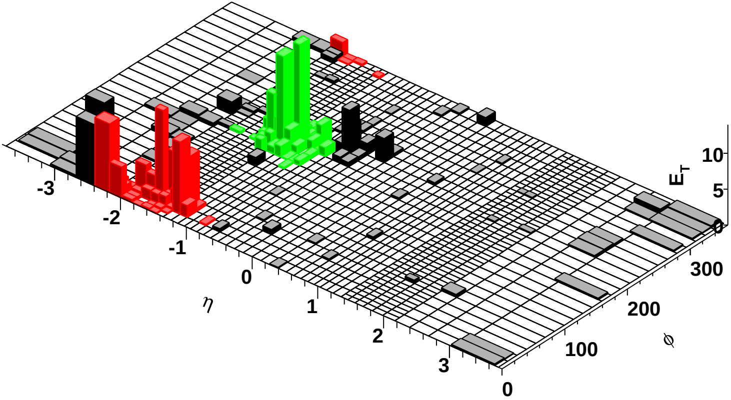
| |
|---|
| cone radius ($R$) |
| seed threshold ($E_0$) |
| $f_{\text{overlap}}$ |

Only one of these is remotely physical: $R$.




2/3 of ILCA flowchart

# ILCA has "Dark Towers"

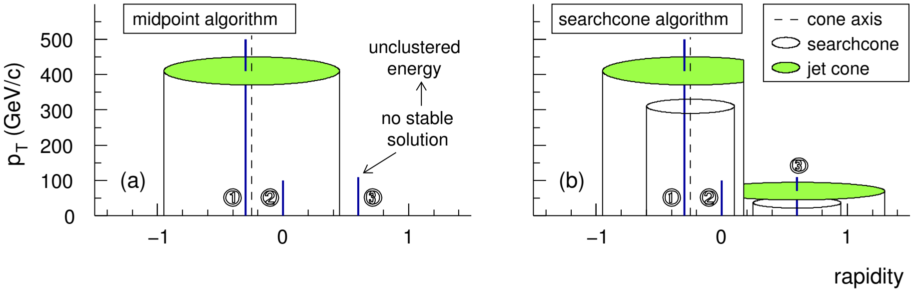



Considerable energy can be left out of jets ≡ **Dark Towers**

S. Ellis, Huston & Tönnesmann '01

# Search Cone

Dark towers are consequence of particles that are never in stable cones:




Ellis, Huston and Tönnesmann suggest *iterating a smaller 'search-cone'* and then drawing final cone around it.

Searchcone adopted by CDF (to confuse issue they still call it 'midpoint'...)

hep-ex/0505013, hep-ex/0512020

...but it looks like it's not infrared safe

Wobisch, '06

# Cone summary

- Cone algorithms are complicated beasts.
- So much so, it's often not clear *which* cone algorithm is being used!
- They often behave in unforeseen ways.
- *Patching* them makes them more complex and error-prone.

Didn't even mention the hacks people put into cone theory calculations to 'tune' them to hadron level: (cf. $R_{sep}$, which breaks the NLO jet X-section).

**LHC experiments should be wary of cone algorithms**

# Sequential Recombination Algorithms

Best known is **$\mathbf{k_t}$ algorithm:**

1. Calculate (or update) distances between all particles $i$ and $j$, and between $i$ and beam:

$$d_{ij} = \min(k_{ti}^2, k_{tj}^2)\frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = k_{ti}^2\,, \qquad \Delta R_{ij}^2 = \Delta y_{ij}^2 + \Delta\phi_{ij}^2$$

2. Find smallest of $d_{ij}$ and $d_{iB}$
   - If $d_{ij}$ is smallest, recombine $i$ and $j$ (add result to particle list, remove $i$, $j$)
   - if $d_{iB}$ is smallest call $i$ a jet (remove it from list of particles)

3. If any particles are left, repeat from step 1.

Catani, Dokshitzer, Olsson, Turnock, Seymour & Webber '91–93

S. Ellis & Soper, '93

Variant: **Cambridge / Aachen algorithm**
Like $k_t$ with but $d_{ij} = \Delta R_{ij}^2/R^2$ and $d_{iB} = 1$.

Dokshitzer, Leder, Moretti & Webber '97; Wobisch '00

# Why $k_t$?

$k_t$ distance measures

$$d_{ij} = \min(k_{ti}^2, k_{tj}^2)\frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = k_{ti}^2$$

are closely related to structure of divergences for QCD emissions

$$[dk_j]|M^2_{g\to g_i g_j}(k_j)| \sim \frac{\alpha_s C_A}{2\pi}\frac{dk_{tj}}{\min(k_{ti}, k_{tj})}\frac{d\Delta R_{ij}}{\Delta R_{ij}}, \qquad (k_{tj} \ll k_{ti}, \;\; \Delta R_{ij} \ll 1)$$

and

$$[dk_i]|M^2_{Beam\to Beam+g_i}(k_i)| \sim \frac{\alpha_s C_A}{\pi}\frac{dk_{ti}}{k_{ti}}\, d\eta_i, \qquad (k_{ti}^2 \ll \{\hat{s}, \hat{t}, \hat{u}\})$$

**$k_t$ algorithm attempts approximate inversion of branching process**

One parameter: $R$ (like cone radius), whose natural value is 1

Optional second parameter: stopping scale $d_{cut}$ — 'exclusive' $k_t$ algorithm

# $k_t$ v. cone

$k_t$ algorithm seems better than cone

- it's simpler, safer and better-defined (IRC safe to all orders)
- exclusive variant is more flexible (allows cuts on momentum scales)
- less sensitive to hadronization
- In MC studies $k_t$ alg. is systematically as good as, or better than cone algorithms for typical reconstruction tasks

Seymour '94

Butterworth, Cox & Forshaw '02

Benedetti et al (Les Houches) '06

But seldom used at Tevatron. **Why?**

1. Because it's slow?
2. Because it includes more underlying event?
3. Because it's harder to understand/correct for detector effects/noise?

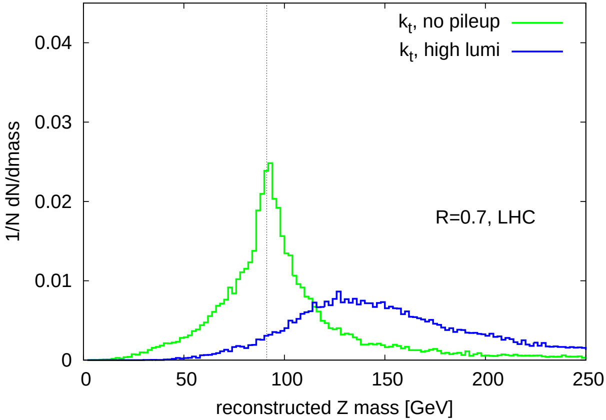
But all LEP and HERA experiments managed fine

And as of '05, CDF too

# $k_t$ v. cone: $Z$ mass (uncorrected)

Try reconstructing $M_Z$ from $Z \to 2$ jets [Use inv. mass of two hardest jets]

On same events, compare uncorrected $k_t$ v. ILCA (midpoint) cone




$k_t$ allegedly more sensitive to min-bias.
*Is this true?*

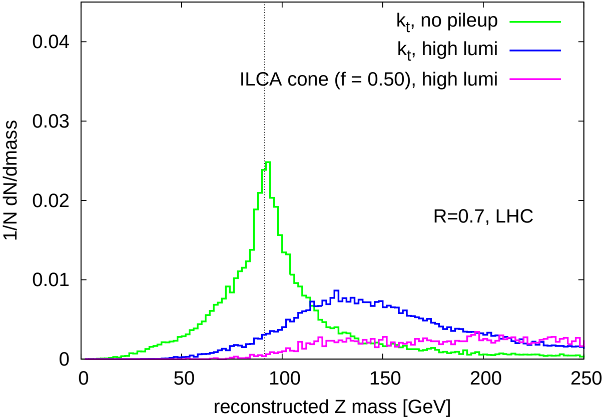
ILCA with standard parameters ($f_{overlap} = 0.5$) fares *very poorly*

ILCA with modified params. is no better than $k_t$.

# $k_t$ v. cone: $Z$ mass (uncorrected)

Try reconstructing $M_Z$ from $Z \to 2$ jets [Use inv. mass of two hardest jets]

On same events, compare uncorrected $k_t$ v. ILCA (midpoint) cone




$k_t$ allegedly more sensitive to min-bias.
*Is this true?*

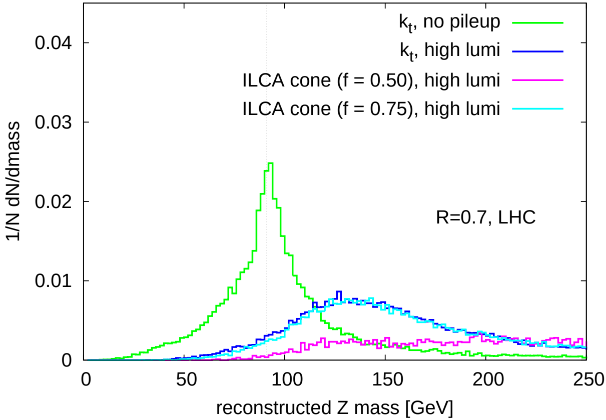
ILCA with standard parameters ($f_{overlap} = 0.5$) fares *very poorly*

ILCA with modified params. is no better than $k_t$.

# $k_t$ v. cone: $Z$ mass (uncorrected)

Try reconstructing $M_Z$ from $Z \to 2$ jets [Use inv. mass of two hardest jets]

On same events, compare uncorrected $k_t$ v. ILCA (midpoint) cone

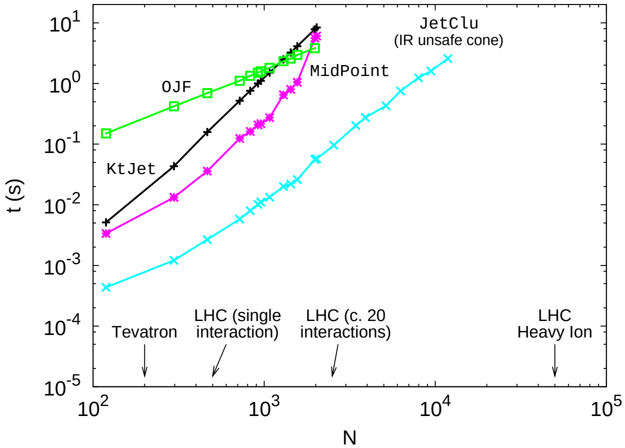



$k_t$ allegedly more sensitive to min-bias.
*Is this true?*

ILCA with standard parameters ($f_{overlap} = 0.5$) fares *very poorly*

ILCA with modified params. is no better than $k_t$.

# Time to cluster $N$ particles




Standard C++ (and fortran) $k_t$-clustering takes time $\sim N^3$.

a Pb-Pb event takes 1 day!

IR-unsafe cone (JetClu) is *much faster*.

IR-safe cone (Midpoint) is as bad as $k_t$

**Jet-clustering speed is an issue** for high-luminosity *pp* ($\sim 10^8$ events) and Pb-Pb ($\sim 10^7$ events) collisions at LHC.

NB: want to rerun jet-alg. with a range of parameter choices

+ want to run on multiple MC samples of similar size

# Why is $k_t$ an $N^3$ algorithm?

1. Given the initial set of particles, construct a table of all the $d_{ij}$, $d_{iB}$.
   [$\mathcal{O}(N^2)$ *operations, done once*]
2. Scan the table to find the minimal value $d_{\min}$ of the $d_{ij}$, $d_{iB}$.
   **[$\mathcal{O}(\mathbf{N}^2)$ operations, done N times]**
3. Merge or remove the particles corresponding to $d_{\min}$ as appropriate.
   [$\mathcal{O}(1)$ operations, done $N$ times]
4. Update the table of $d_{ij}$, $d_{iB}$ to take into account the merging or removal, and if any particles are left go to step 2.
   [$\mathcal{O}(N)$ operations, done $N$ times]

This is the "brute-force" or "naive" method

# Can we do better than $N^2$?

There are $N(N-1)/2$ distances $d_{ij}$ — surely we have to calculate them all in order to find smallest?

$k_t$ distance measure is partly *geometrical:*

- Consider smallest $d_{ij} = \min(k_{ti}^2, k_{tj}^2) R_{ij}^2$
- Suppose $k_{ti} < k_{tj}$
- Then: $R_{ij} <= R_{i\ell}$ for any $\ell \neq j$. [If $\exists\ \ell$ s.t. $R_{i\ell} < R_{ij}$ then $d_{i\ell} < d_{ij}$]

*In words:*

**if $i, j$ form smallest $d_{ij}$ then $j$ is geometrical nearest neighbour (GNN) of $i$.**

$\Rightarrow$ $k_t$ distance need only be calculated between GNNs

Each point has 1 GNN $\rightarrow$ need only calculate $N$ $d_{ij}$'s

# Can we do better than $N^2$?

There are $N(N-1)/2$ distances $d_{ij}$ — surely we have to calculate them all in order to find smallest?

$k_t$ distance measure is partly *geometrical:*

- Consider smallest $d_{ij} = \min(k_{ti}^2, k_{tj}^2)R_{ij}^2$
- Suppose $k_{ti} < k_{tj}$
- Then: $R_{ij} <= R_{i\ell}$ for any $\ell \neq j$. [If $\exists\ \ell$ s.t. $R_{i\ell} < R_{ij}$ then $d_{i\ell} < d_{ij}$]

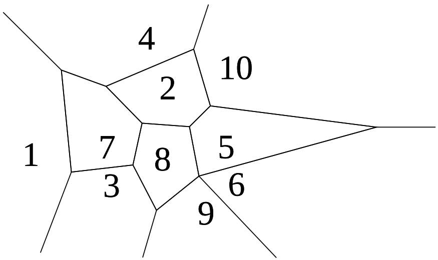
*In words:*

**if $i, j$ form smallest $d_{ij}$ then $j$ is geometrical nearest neighbour (GNN) of $i$.**

$\Rightarrow$ $k_t$ distance need only be calculated between GNNs

Each point has 1 GNN $\rightarrow$ need only calculate $N$ $d_{ij}$'s

# Finding Geom Nearest Neighbours




Given a set of vertices on plane (1...10) a *Voronoi diagram* partitions plane into cells containing all points closest to each vertex

Dirichlet '1850, Voronoi '1908

A vertex's nearest other vertex is always in an adjacent cell.

E.g. GNN of point 7 will be found among 1,4,2,8,3 (it turns out to be 3)

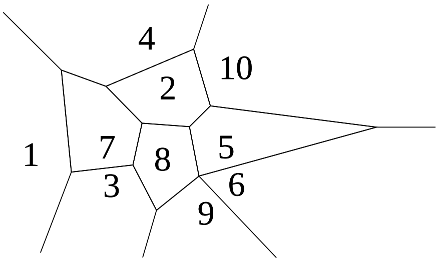
Construction of Voronoi diagram for $N$ points: $N \ln N$ time    Fortune '88

Update of 1 point in Voronoi diagram: $\ln N$ time

Devillers '99 [+ related work by other authors]

Convenient C++ package available: **CGAL**    http://www.cgal.org

# Finding Geom Nearest Neighbours




Given a set of vertices on plane (1...10) a *Voronoi diagram* partitions plane into cells containing all points closest to each vertex

Dirichlet '1850, Voronoi '1908

A vertex's nearest other vertex is always in an adjacent cell.

E.g. GNN of point 7 will be found among 1,4,2,8,3 (it turns out to be 3)

Construction of Voronoi diagram for $N$ points: $N \ln N$ time — Fortune '88

Update of 1 point in Voronoi diagram: $\ln N$ time

Devillers '99 [+ related work by other authors]

Convenient `C++` package available: **CGAL** — `http://www.cgal.org`

# Assembling fast $k_t$ clustering

The `FastJet` algorithm:

Construct the Voronoi diagram of the $N$ particles with CGAL **$\mathcal{O}(N \ln N)$**

Find the GNN of each of the $N$ particles, calculate $d_{ij}$ store result in a *priority queue* (`C++ map`) **$\mathcal{O}(N \ln N)$**

Repeat following steps **N** times:

- Find smallest $d_{ij}$, merge/eliminate $i, j$ **$N \times \mathcal{O}(1)$**
- Update Voronoi diagram and distance map **$N \times \mathcal{O}(\ln N)$**

**Overall an $\mathcal{O}(N \ln N)$ algorithm**

MC & GPS, hep-ph/0512210

http://www.lpthe.jussieu.fr/~salam/fastjet/

Results **identical** to standard $N^3$ implementations:
this is **NOT** a new $k_t$ jet-finder

# Assembling fast $k_t$ clustering

The `FastJet` algorithm:

Construct the Voronoi diagram of the $N$ particles with CGAL $\mathcal{O}(\mathbf{N} \ln \mathbf{N})$

Find the GNN of each of the $N$ particles, calculate $d_{ij}$ store result in a *priority queue* (`C++ map`) $\mathcal{O}(\mathbf{N} \ln \mathbf{N})$

Repeat following steps **N** times:

- Find smallest $d_{ij}$, merge/eliminate $i, j$ $\mathbf{N} \times \mathcal{O}(1)$
- Update Voronoi diagram and distance map $\mathbf{N} \times \mathcal{O}(\ln \mathbf{N})$

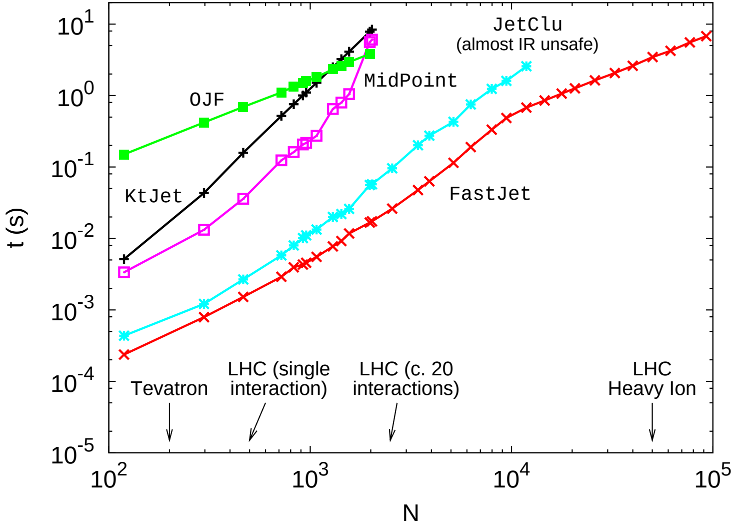
**Overall an $\mathcal{O}(\mathbf{N} \ln \mathbf{N})$ algorithm**

MC & GPS, hep-ph/0512210

`http://www.lpthe.jussieu.fr/~salam/fastjet/`

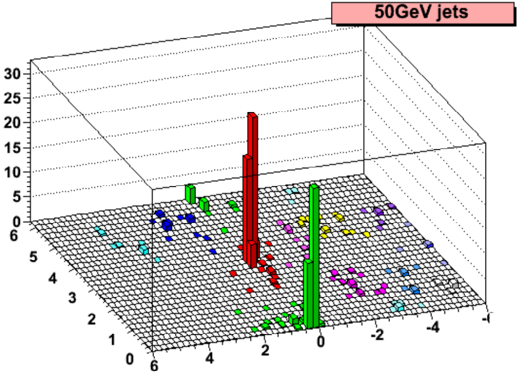
Results **identical** to standard $N^3$ implementations:
this is **NOT** a new $k_t$ jet-finder

# FastJet performance

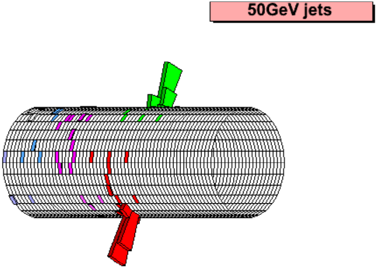



NB: for $N < 10^4$, FastJet switches to a related geometrical $N^2$ alg.

# What is speed good for?



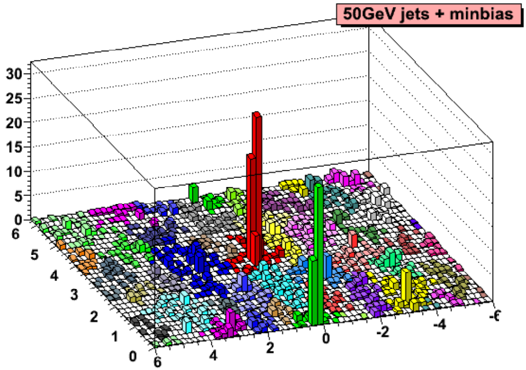

'Standard hard' event
Two well isolated jets



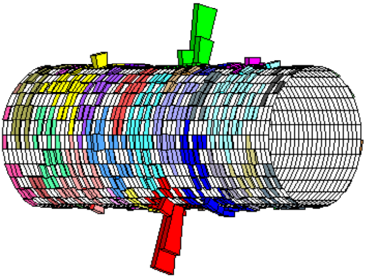

$\sim$ 200 particles

Easy even with old methods

# What is speed good for?




Add 10 min-bias events (moderately high lumi)

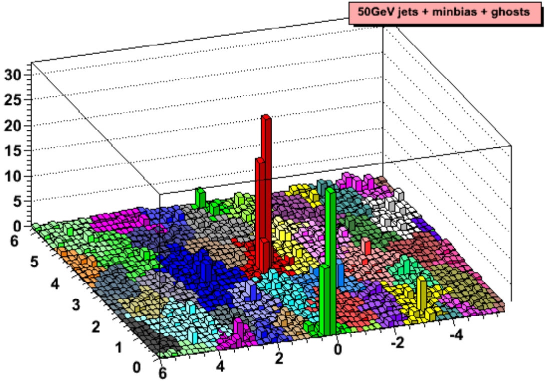


$\sim$ 2000 particles

Clustering takes $\mathcal{O}(10s)$ with old methods.

20ms with `FastJet`.

# What is speed good for?



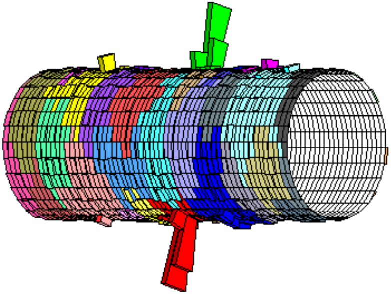

Add dense coverage of infinitely soft *“ghosts”*

See how many end up in jet to measure jet area

~ 10000 particles

Clustering takes ~ 20 minutes with old methods.

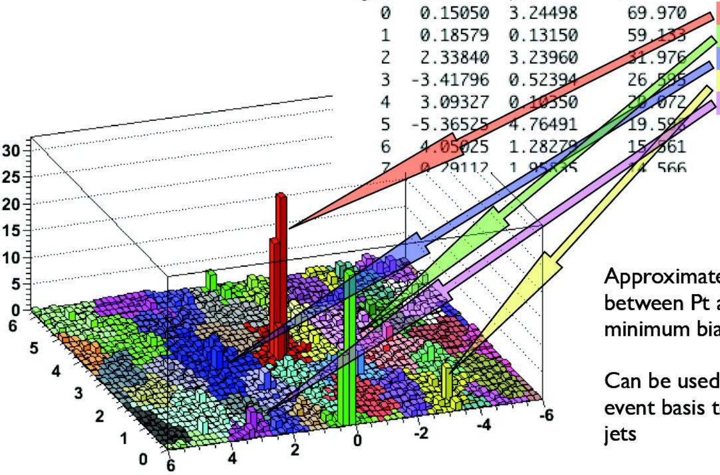
0.6s with `FastJet`.



# Jet areas

iev 0 (irepeat 24): number of particles = 1428
strategy used = NlnN
number of particles = 9051
Total area: 76.0265
Expected area: 76.0265

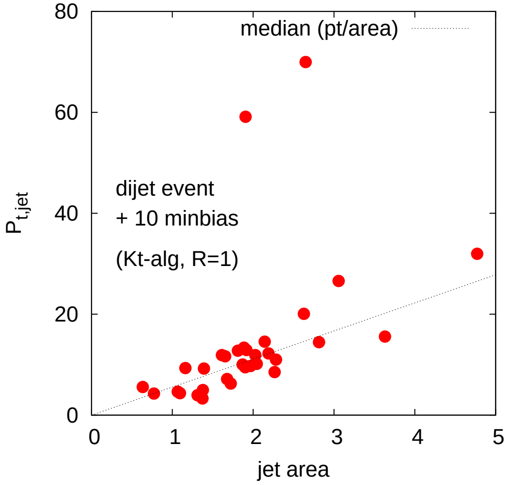
| ijet | eta | phi | Pt | area | +- | err |
|---|---|---|---|---|---|---|
| 0 | 0.15050 | 3.24498 | 69.970 | 2.625 | +- | 0.020 |
| 1 | 0.18579 | 0.13150 | 59.133 | 1.896 | +- | 0.020 |
| 2 | 2.33840 | 3.23960 | 31.976 | 4.749 | +- | 0.028 |
| 3 | -3.41796 | 0.52394 | 26.595 | 3.084 | +- | 0.021 |
| 4 | 3.09327 | 0.10350 | 20.072 | 2.688 | +- | 0.023 |
| 5 | -5.36525 | 4.76491 | 19.593 | 2.780 | +- | 0.012 |
| 6 | 4.05025 | 1.28279 | 15.561 | 3.592 | +- | 0.028 |
| 7 | 0.29112 | 1.95835 | 14.566 | 2.114 | +- | 0.018 |




Approximate linear relation between Pt and area for minimum bias jets.

Can be used on an event-by-event basis to correct the hard jets

# Jet areas




Jet areas in $k_t$ algorithm are quite varied

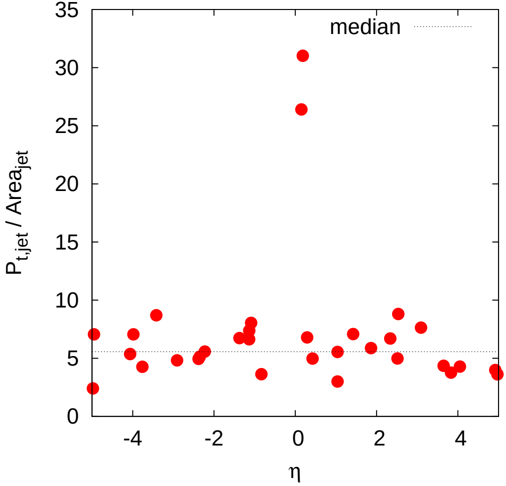
Because $k_t$-alg adapts to the jet structure

- Contamination from min-bias $\sim$ area

Complicates corrections: min-bias subtraction is different for each jet.

Cone supposedly simpler
Area $= \pi R^2$?

# Subtraction using areas




Key observation: $p_T$/area is quite uniform, except for the hard jets

**Correction procedure:**

1. Measure area $A$ of each jet using ghost particles
2. Find median $p_t/A = Q_0$
3. Subtract $\Delta p_t = A \times Q_0$ from each jet.

NB. This is an event-by-event correction, which also provides its own uncertainty

NB: cone much harder to correct this way — too slow to add $10^4$ ghosts

# UE/MB subtraction using areas

Examples of UE/MB subtraction using `FastJet` and area method

Preliminary results (MC & GPS) for

- High-lumi LHC
  - $Z$ production
  - $Z'$ (mass = 2 TeV)
  - $W$ bosons in $t\bar{t}$ events
  - ...
- Heavy ion collisions
  - inclusive jet distribution in Pb-Pb collisions

# Disclaimer

I'm trying to sell the idea (and the `FastJet` code),
not the work itself

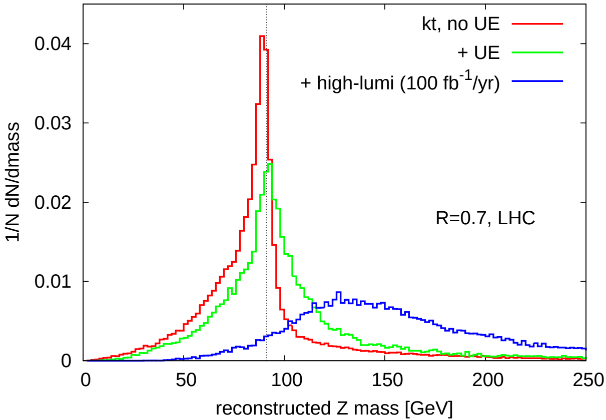
The "analyses" I'll be showing are certainly naive (theorists' work) and miss many refinements:

- We use all hadrons, neutral and charged alike, to construct the jets
- No detector effects are included
- ...

Finally, the value of these results can only be judged by comparing them to similar ones obtained with different techniques and/or jet-finders

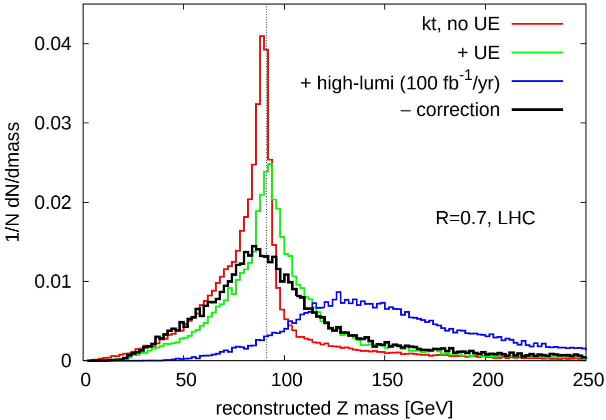
# Use jet areas to correct jet kinematics

Try reconstructing $M_Z$ from $Z \to 2$ jets, with subtraction of UE/MB

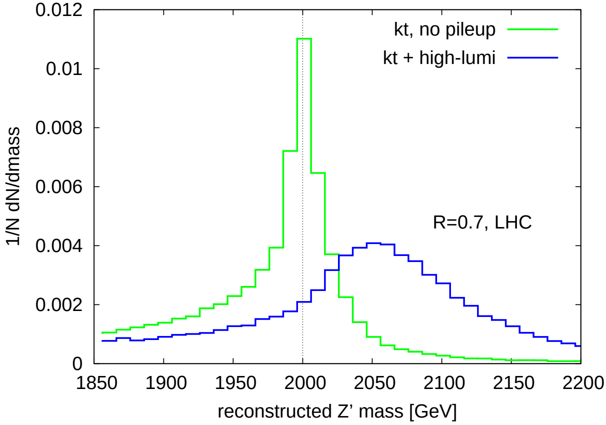


# Use jet areas to correct jet kinematics

Try reconstructing $M_Z$ from $Z \to 2$ jets, with subtraction of UE/MB



# Reconstruct $Z'$ mass [2 TeV]

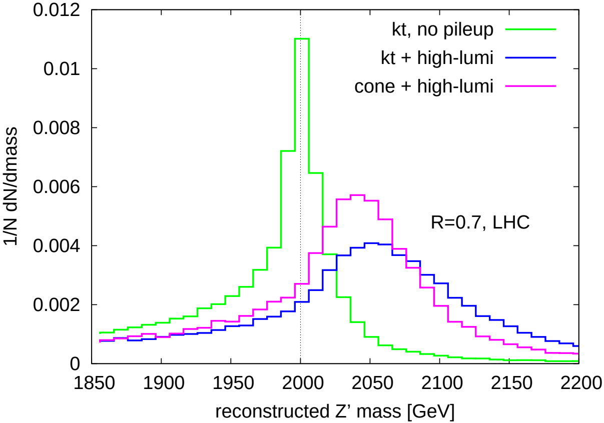



Uncorrected cone better than $k_t$.

Cam is intermediate ($\langle A_{cam} \rangle \simeq \langle A_{cone} \rangle$, but fluctuations larger)

Corrected Cam (and $k_t$) is best.

# Reconstruct $Z'$ mass [2 TeV]

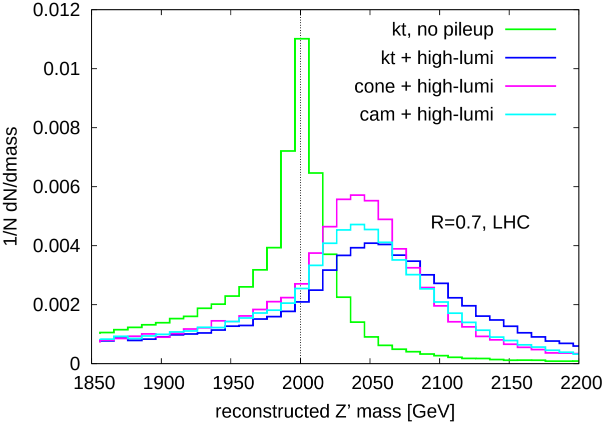



Uncorrected cone better than $k_t$.

Cam is intermediate ($\langle A_{cam} \rangle \simeq \langle A_{cone} \rangle$, but fluctuations larger)

Corrected Cam (and $k_t$) is best.

# Reconstruct $Z'$ mass [2 TeV]

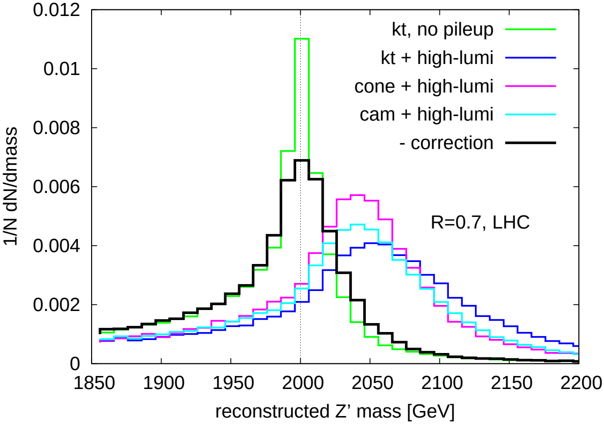



Uncorrected cone better than $k_t$.

Cam is intermediate ($\langle A_{cam} \rangle \simeq \langle A_{cone} \rangle$, but fluctuations larger)

Corrected Cam (and $k_t$) is best.

# Reconstruct $Z'$ mass [2 TeV]




Uncorrected cone better than $k_t$.

Cam is intermediate ($\langle A_{cam} \rangle \simeq \langle A_{cone} \rangle$, but fluctuations larger)

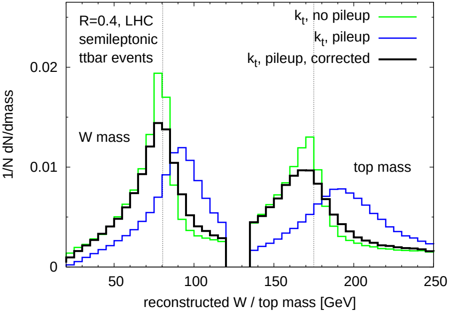
Corrected Cam (and $k_t$) is best.

# $t\bar{t}$ at LHC

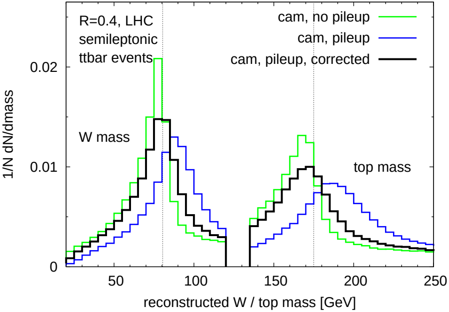
$t\bar{t}$ production in high-lumi $pp$ collisions at LHC

$W$ mass reconstruction via dijet mass in semileptonic decay with $b$-tagging

## $k_t$




## Cambridge



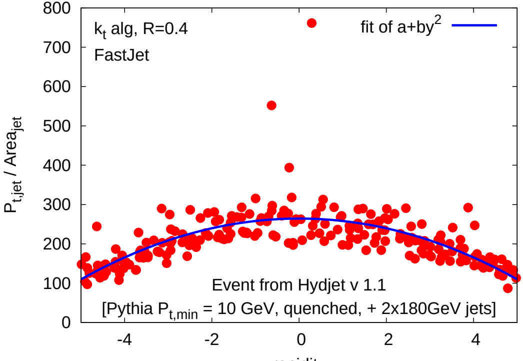
# Heavy ions at LHC

At LHC one expects $\sim$ 30000 particles per Pb-Pb collisions

Very few will be hard (e.g. a dijet event), most will be very soft (10 GeV or less).

Easy way of decluttering the event: a minimum $p_T$ cut. However, this is not an infrared safe procedure, and the result must then be artificially corrected back to the ‘real’ one.

Alternative: same kind of subtraction used in high-lumi *pp* events

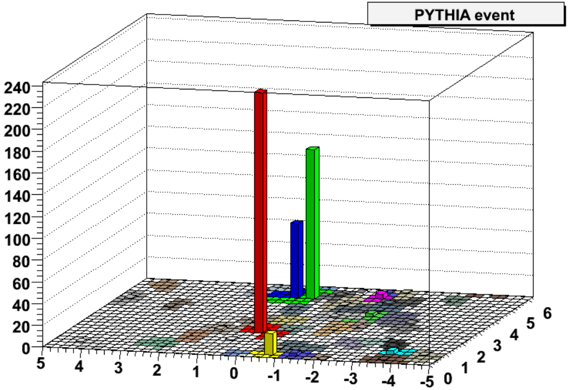



NB1: the simulation of a heavy ions collision suggests a parabolic fit of the background

NB2: no minimum $p_T$ cut will ever be used

# An easy piece

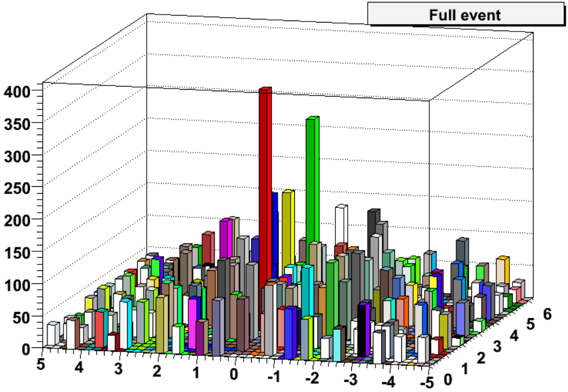
A **200 GeV** dijet PYTHIA event embedded in a HYDJET Pb-Pb one at the LHC




1. The *pp* hard event generated by PYTHIA only
2. The same event embedded in the whole Pb-Pb collisions
3. The result of the subtraction of the background

# An easy piece

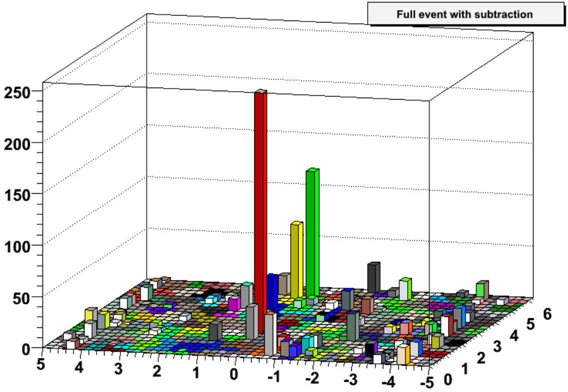
A **200 GeV** dijet PYTHIA event embedded in a HYDJET Pb-Pb one at the LHC




1. The *pp* hard event generated by PYTHIA only
2. The same event embedded in the whole Pb-Pb collisions
3. The result of the subtraction of the background

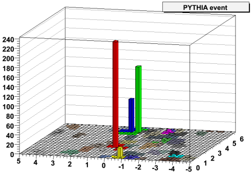
# An easy piece

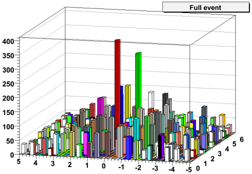
A **200 GeV** dijet PYTHIA event embedded in a HYDJET Pb-Pb one at the LHC

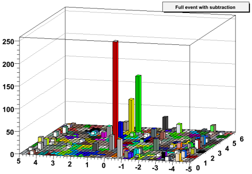



1. The *pp* hard event generated by PYTHIA only
2. The same event embedded in the whole Pb-Pb collisions
3. The result of the subtraction of the background

# How does it work?




# ACM-SIAM Symposium on Discrete A
#-----------------------------------

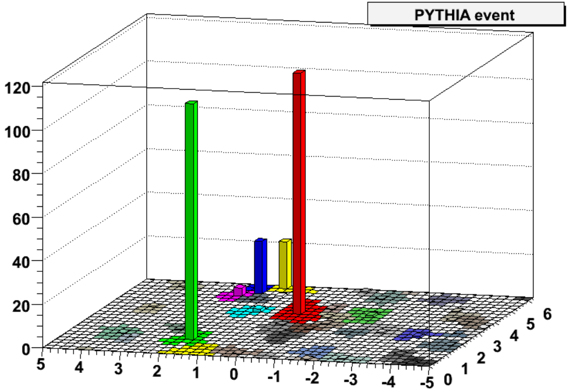
| # ijet | rap | phi | Pt |
|---|---|---|---|
| 0 | 0.350 | 2.356 | 220.847 |
| 1 | 0.407 | 5.362 | 136.889 |
| 2 | 0.820 | 5.436 | 68.398 |
| 3 | -0.726 | 0.383 | 19.615 |
| 4 | -1.452 | 5.632 | 4.385 |
| 5 | -3.613 | 1.036 | 3.563 |
| 6 | -1.787 | 3.970 | 2.940 |
| 7 | -1.346 | 0.340 | 2.643 |
| 8 | -0.781 | 5.829 | 2.061 |
| 9 | 1.332 | 2.241 | 1.932 |
| 10 | -3.429 | 1.415 | 1.833 |
| 11 | -1.800 | 5.162 | 1.671 |
| 12 | -0.287 | 5.705 | 1.362 |

## full event subtraction

| # ijet | rap | phi | Pt | area | [illegible] | (rap corr | phi corr | Pt corr)ext |
|---|---|---|---|---|---|---|---|---|
| 0 | 0.365 | 2.384 | 373.691 | 0.508 | [illegible] | 0.362 | 2.350 | 234.278 |
| 1 | 0.319 | 5.344 | 279.249 | 0.572 | [illegible] | 0.362 | 5.312 | 124.396 |
| 2 | 0.893 | 4.389 | 177.530 | 0.546 | [illegible] | 0.940 | 4.216 | 31.866 |
| 3 | 0.854 | 5.286 | 163.356 | 0.343 | [illegible] | 0.820 | 5.402 | 71.867 |
| 4 | 1.723 | 3.549 | 147.817 | 0.546 | [illegible] | 1.729 | 4.007 | 11.831 |
| 5 | -1.654 | 0.663 | 134.394 | 0.470 | [illegible] | -1.608 | 0.739 | 15.476 |
| 6 | -2.071 | 1.217 | 134.374 | 0.546 | [illegible] | -1.999 | 1.880 | 4.367 |
| 7 | 0.649 | 3.035 | 132.372 | 0.445 | [illegible] | 0.716 | 2.911 | 11.869 |
| 8 | -0.067 | 6.259 | 130.080 | 0.546 | [illegible] | 100000.000 | 0.000 | 0.000 |
| 9 | 1.370 | 5.331 | 128.190 | 0.432 | [illegible] | 1.245 | 5.730 | 18.678 |
| 10 | -0.983 | 6.081 | 126.292 | 0.368 | [illegible] | -0.953 | 6.154 | 27.413 |
| 11 | 1.006 | 1.321 | 124.928 | 0.406 | [illegible] | 1.087 | 1.437 | 16.194 |
| 12 | -1.913 | 2.834 | 122.455 | 0.521 | [illegible] | 100000.000 | 0.000 | 0.000 |
| 13 | -1.327 | 5.567 | 116.838 | 0.419 | [illegible] | -1.476 | 5.805 | 6.334 |
| 14 | 0.313 | 1.751 | 115.682 | 0.305 | [illegible] | 0.326 | 1.732 | 31.802 |
| 15 | 1.609 | 4.179 | 112.815 | 0.356 | [illegible] | 1.609 | 4.392 | 22.361 |
| 16 | -1.506 | 4.920 | 111.001 | 0.419 | [illegible] | -1.701 | 4.707 | 2.443 |

# A less easy piece

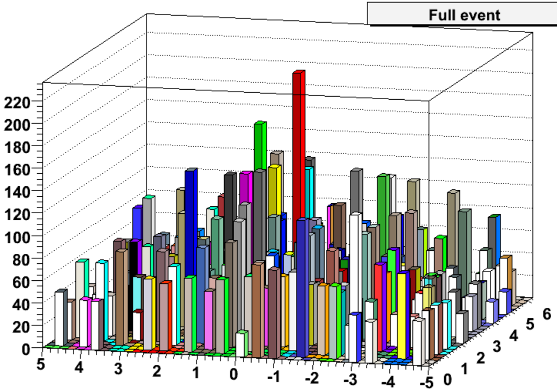
A **100 GeV** dijet PYTHIA event embedded in a HYDJET Pb-Pb one at the LHC




1. The *pp* hard event generated by PYTHIA only
2. The same event embedded in the whole Pb-Pb collisions
3. The result of the subtraction of the background

# A less easy piece

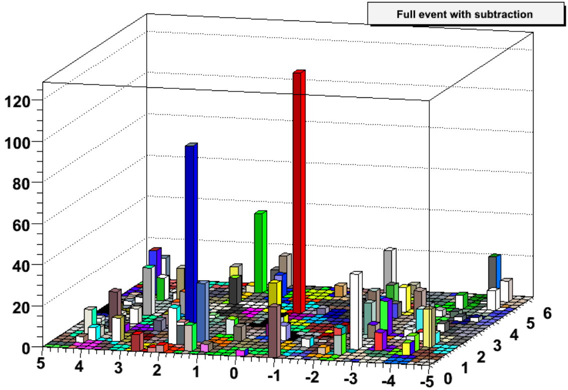
A **100 GeV** dijet PYTHIA event embedded in a HYDJET Pb-Pb one at the LHC




1. The *pp* hard event generated by PYTHIA only
2. The same event embedded in the whole Pb-Pb collisions
3. The result of the subtraction of the background

# A less easy piece

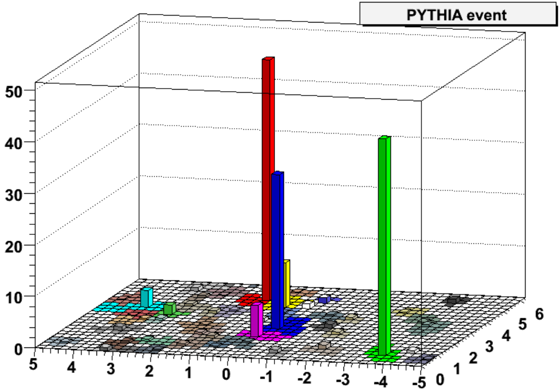
A **100 GeV** dijet PYTHIA event embedded in a HYDJET Pb-Pb one at the LHC




1. The *pp* hard event generated by PYTHIA only
2. The same event embedded in the whole Pb-Pb collisions
3. The result of the subtraction of the background

# A difficult piece

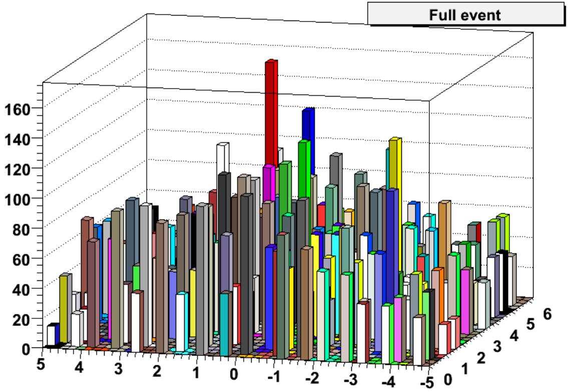
A **40 GeV** dijet PYTHIA event embedded in a HYDJET Pb-Pb one at the LHC




1. The *pp* hard event generated by PYTHIA only
2. The same event embedded in the whole Pb-Pb collisions
3. The result of the subtraction of the background

# A difficult piece

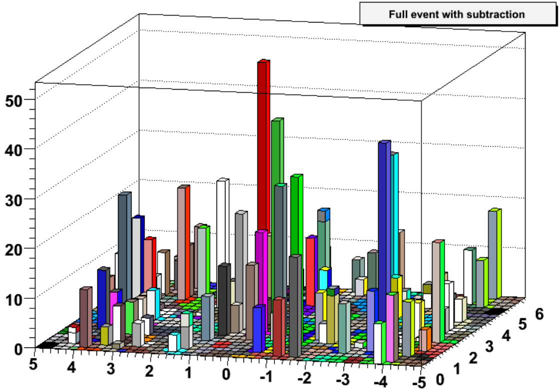
A **40 GeV** dijet PYTHIA event embedded in a HYDJET Pb-Pb one at the LHC




1. The *pp* hard event generated by PYTHIA only
2. The same event embedded in the whole Pb-Pb collisions
3. The result of the subtraction of the background

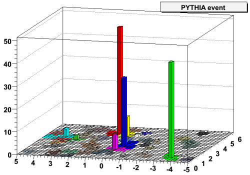
# A difficult piece

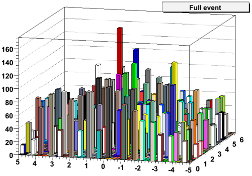
A **40 GeV** dijet PYTHIA event embedded in a HYDJET Pb-Pb one at the LHC

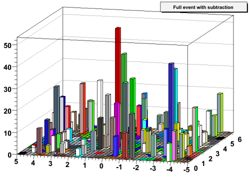



1. The *pp* hard event generated by PYTHIA only
2. The same event embedded in the whole Pb-Pb collisions
3. The result of the subtraction of the background

# How does it work?



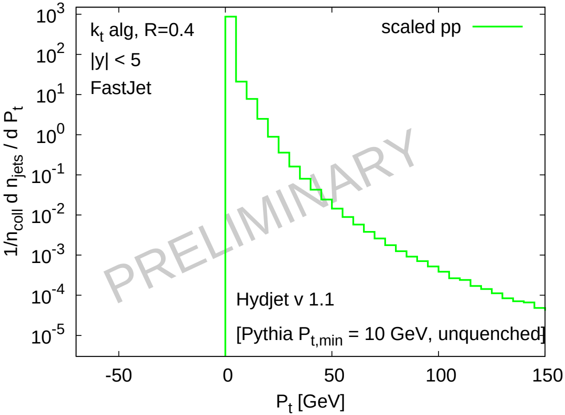




```
## hard event
# ijet     rap     phi        Pt     area
    0    1.138   4.990    46.696    0.807
    1   -3.693   0.982    41.942    0.813
    2   -0.166   2.638    29.912    1.143
    3    0.599   4.654     8.716    0.934
    4    0.054   1.967     6.157    0.553
    5    3.880   3.941     3.238    1.073
    6    3.001   3.589     1.840    0.622
    7   -0.256   5.169     1.126    0.413

## full event subtraction
# ijet     rap     phi        Pt     area   [illegible]   (rap corr phi corr Pt corr)ext
    0    1.094   4.945   160.668    0.521   [illegible]      1.073    4.937    48.411
    1   -1.179   1.530   134.888    0.470   [illegible]     -1.203    1.539    33.736
    2    0.259   5.016   127.540    0.635   [illegible]  100000.000    0.000     0.000
    3   -2.602   3.506   123.007    0.635   [illegible]     -2.546    3.974     8.290
    4    0.081   2.138   111.034    0.406   [illegible]      0.109    2.211    20.413
    5   -2.250   4.253   110.653    0.406   [illegible]     -2.308    4.313    32.292
    6   -0.156   2.741   109.869    0.305   [illegible]     -0.143    2.696    41.763
    7   -3.620   0.856   109.479    0.457   [illegible]     -3.689    0.987    42.644
    8    2.107   4.399   107.934    0.419   [illegible]      2.074    4.369    25.428
    9    0.588   1.614   107.771    0.470   [illegible]      0.773    2.281     5.529
   10    1.082   1.989   106.716    0.432   [illegible]      1.023    2.011    13.914
   11   -0.165   0.441   105.569    0.495   [illegible]  100000.000    0.000     0.000
   12   -1.498   0.482   104.687    0.406   [illegible]     -1.463    0.427    20.104
   13   -0.489   4.917   100.438    0.406   [illegible]     -0.276    5.166    10.850
   14    0.949   0.054    99.661    0.445   [illegible]      0.685    5.813     4.184
   15    0.649   2.721    98.143    0.343   [illegible]      0.664    2.617    23.367
   16    1.406   5.541    95.984    0.457   [illegible]      1.290    0.769     4.452
   17    2.521   0.778    93.890    0.483   [illegible]      2.503    0.641     4.706
```

# Inclusive jets in Pb-Pb collisions

Apply subtraction procedure allows to the *pp* single inclusive jet distribution from Pb-Pb collisions:

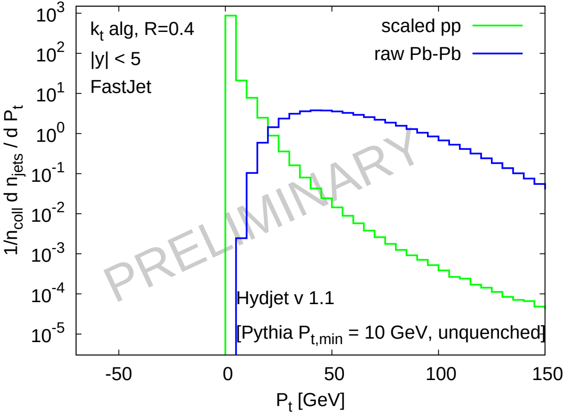



Good agreement with 'hard' distribution after subtraction of huge background

Even this rough subtraction seems able to allow one measuring jets down to low $p_T$

# Inclusive jets in Pb-Pb collisions

Apply subtraction procedure allows to the *pp* single inclusive jet distribution from Pb-Pb collisions:

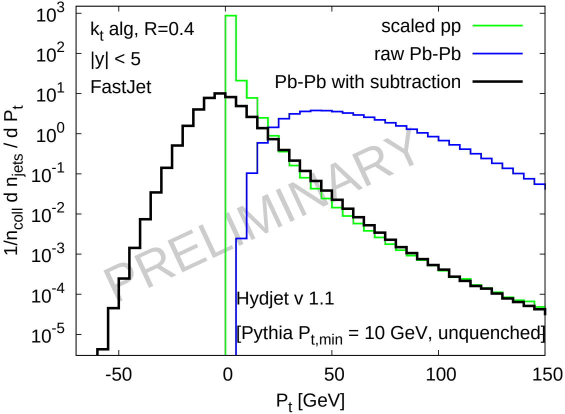



Good agreement with 'hard' distribution after subtraction of huge background

Even this rough subtraction seems able to allow one measuring jets down to low $p_T$

# Inclusive jets in Pb-Pb collisions

Apply subtraction procedure allows to the *pp* single inclusive jet distribution from Pb-Pb collisions:

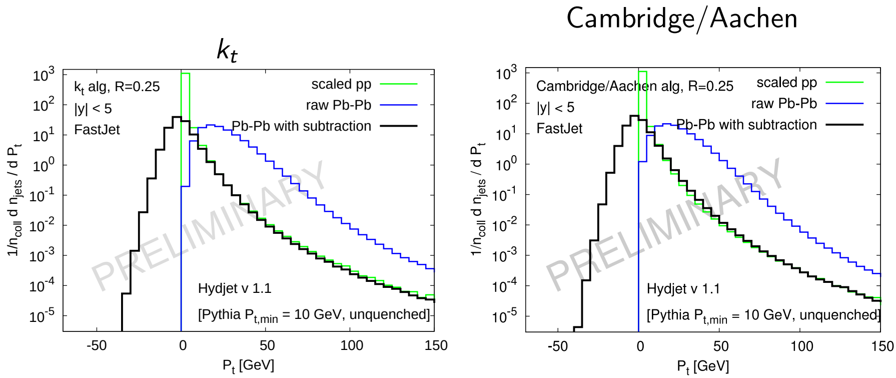



Good agreement with 'hard' distribution after subtraction of huge background

Even this rough subtraction seems able to allow one measuring jets down to low $p_T$

# Inclusive jets in Pb-Pb collisions

Similar results from Cambridge/Aachen, and with $R = 0.25$:




⇒ With smaller $R$ we seem to be able to go to even lower $p_T$

# Summary

- **Speed**
  - $k_t$ alg. can be **fast** — key observation is geometrical reformulation
    Get code from `http://www.lpthe.jussieu.fr/~salam/fastjet`
- **Jet areas and subtractions**
  - Jet areas ($\rightarrow$ min. bias. contributions) do fluctuate
  - But areas can (should) be **measured** and **used for correction** on jet-by-jet basis.
    Preliminary studies seem promising
    Version 2.0 of `FastJet` includes the subtraction
  - $k_t$ is part of a class of algorithms — other example deserving more attention is *Cambridge/Aachen* alg.
    It too can be made fast

# Summary

- **Speed**
  - $k_t$ alg. can be **fast** — key observation is geometrical reformulation
    Get code from `http://www.lpthe.jussieu.fr/~salam/fastjet`
- **Jet areas and subtractions**
  - Jet areas (→ min. bias. contributions) do fluctuate
  - But areas can (should) be **measured** and **used for correction** on jet-by-jet basis.
    Preliminary studies seem promising
    Version 2.0 of `FastJet` includes the subtraction
  - $k_t$ is part of a class of algorithms — other example deserving more attention is *Cambridge/Aachen* alg.
    It too can be made fast