

Jet Algorithms, Areas and Subtraction

Matteo Cacciari
LPTHE (Paris 6/7 & CNRS)

Work in collaboration with
Gavin Salam and Gregory Soyez

- Jet Algorithms, “cone v. kt”
- Infrared safety, performance
- Jet areas
- Subtraction

Jet Algorithm

 $\{P_i\}$

particles,
4-momenta,
calorimeter towers,

jet algorithm

 $\{j_k\}$

jets

Jet Algorithm requirements

A jet-finder **must** be

✓ infrared and collinear safe

soft emission shouldn't change jets
collinear splitting shouldn't change jets

(this allows one to use perturbation theory)

✓ identically defined at parton and hadron level

so that perturbative calculations can be compared to experiments

It is **nice** if a jet-finder is

✓ not too sensitive to hadronisation, underlying event, pile-up

(because we are not very good at modeling non-perturbative stuff)

✓ realistically applicable at detector level

(e.g. not too slow)

Jet Algorithms

Two main jet-finder classes: **cone algorithms** and **sequential clustering algorithms**

Cone

JetClu, Midpoint, SISCone...

Top-down:

Find coarse regions of energy flow (cones), and call them jets.

Works because *QCD only modifies energy flow on small scales*

Loved by *pp* and few(er) theorists

Sequential recombination

k_t , Jade, Cam/Aachen, ...

Bottom-up:

Cluster 'closest' particles repeatedly until few left \rightarrow jets.

Works because of mapping:
closeness \Leftrightarrow QCD divergence

Loved by e^+e^- , *ep* and theorists

Detailed definition can be messy.
Infrared/collinear safety must be carefully studied.

Simple definition,
infrared and collinear safe.

Until some time ago **cone** was infrared unsafe
and \mathbf{k}_t was slow

What happened next?

- k_t made fast (MC, Salam, hep-ph/0512210)
- cone made safe (Salam, Soyez, arXiv: 0704.0292)

Both implementations (and a lot more) available via Fastjet

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

Cone algorithms

A modern cone algorithm

How do I decide where to place the cones?

- 👉 try an initial location
- 👉 sum 4-momenta of particles inside cone, find axis
- 👉 use axis as a new trial location, and **iterate**
- 👉 stop when axis is stable
- 👉 merge overlapping cones, or split them into two

Issues:

💀 Where do I start?

Seedless (i.e. everywhere)?

Some particles above a threshold?

Calorimeter towers?

Very slow

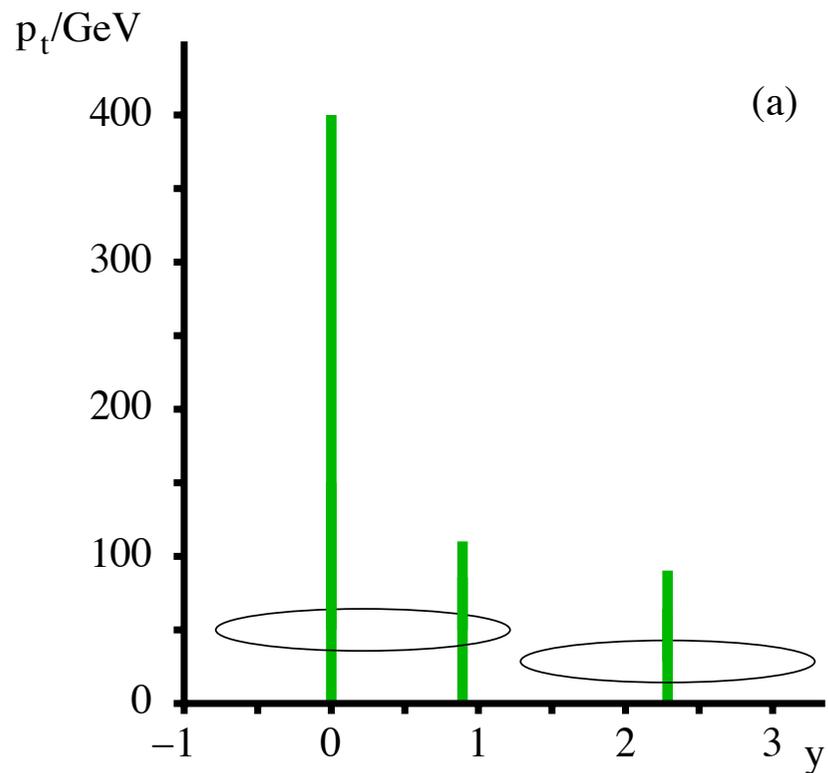
Collinear unsafe

Expt. dependent

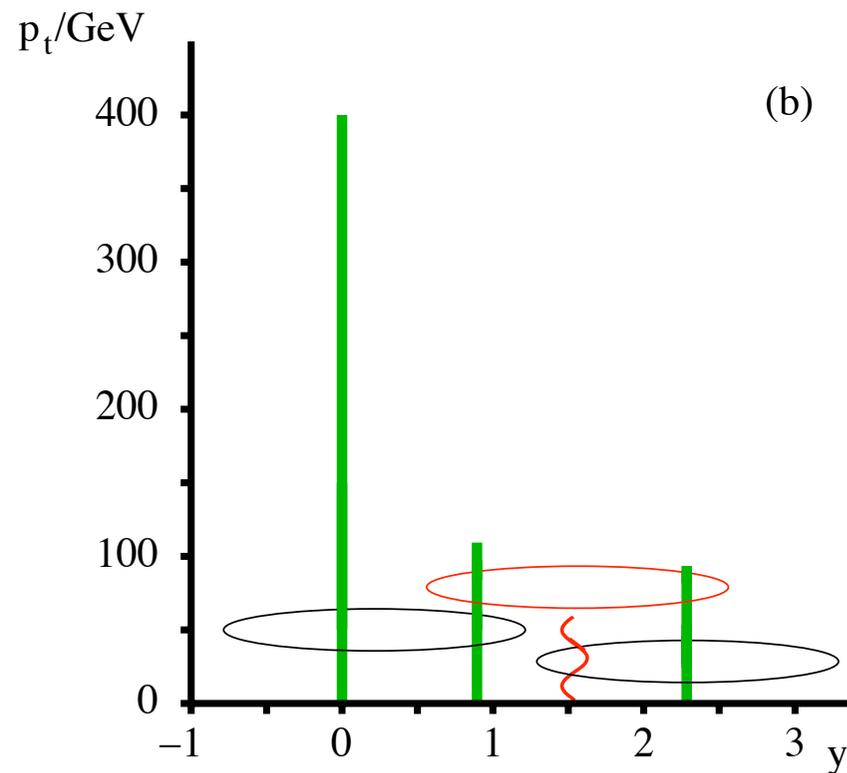
💀 How do I split/merge?

Complicated procedure, risky, not necessarily physical

MidPoint Infrared Unsafety



Three hard particles
clustered into two cones



Addition of a **soft** particles
changes the hard jets
configuration: three stable
cones are found

IR/Collinear unsafety is a serious problem for theorists!

- ▶ Invalidates theorems that ensure finiteness of perturbative QCD
 - Cancellation of real & virtual divergences
 - Makes results inherently non-perturbative
- ▶ ‘Pragmatically:’ limits accuracy to which it makes sense to calculate
 - Higher orders no longer form convergent series

Process	<i>Last meaningful order</i>	
	JetClu/Searchcone	MidPoint
Inclusive jets	LO	NLO [NNLO being worked on]
$W/Z + 1$ jet	LO	NLO
3 jets	none	LO [NLO in nlojet++]
$W/Z + 2$ jets	none	LO [NLO in MCFM]
jet masses in $2j + X$	none	none [LO in madgraph etc.]

Rather than define the cone alg. through the *procedure* you use to find cones, define it by the *result you want*:

A cone algorithm should find **all** stable cones

First advocated: Kidonakis, Oderda & Sterman '97

Guarantees IR safety of the set of stable cones

Only issue: you still need to find the stable cones in practice.

One known exact approach:

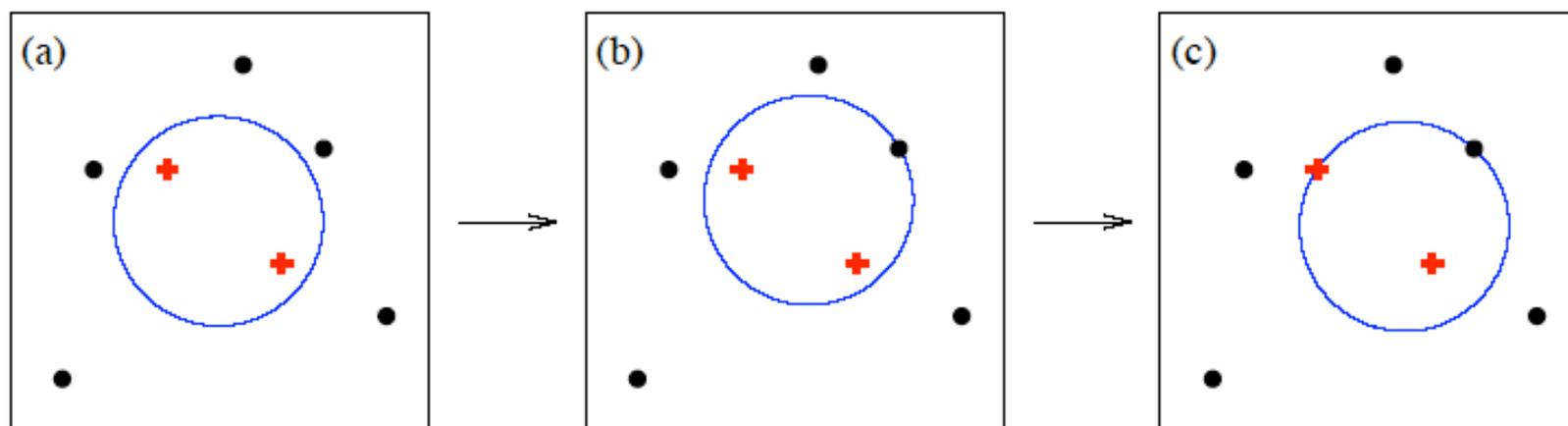
- ▶ Take each possible subset of particles and see if it forms a stable cone.
Tevatron Run II workshop, '00 (for fixed-order calcs.)
- ▶ There are 2^N subsets for N particles. Computing time $\sim N2^N$.
 10^{17} years for an event with 100 particles

Transform into geometrical problem

Cones are just *circles* in the $y - \phi$ plane. To find all stable cones:

1. Find all distinct ways of enclosing a subset of particles in a $y - \phi$ circle
2. Check, for each enclosure, if it corresponds to a stable cone

Finding all distinct circular enclosures of a set of points is *geometry*:



Any enclosure can be moved until a pair of points lies on its edge.

Result: Seedless Infrared Safe Cone algorithm (SISCone)

Runs in $N^2 \ln N$ time (\simeq midpoint's N^3)

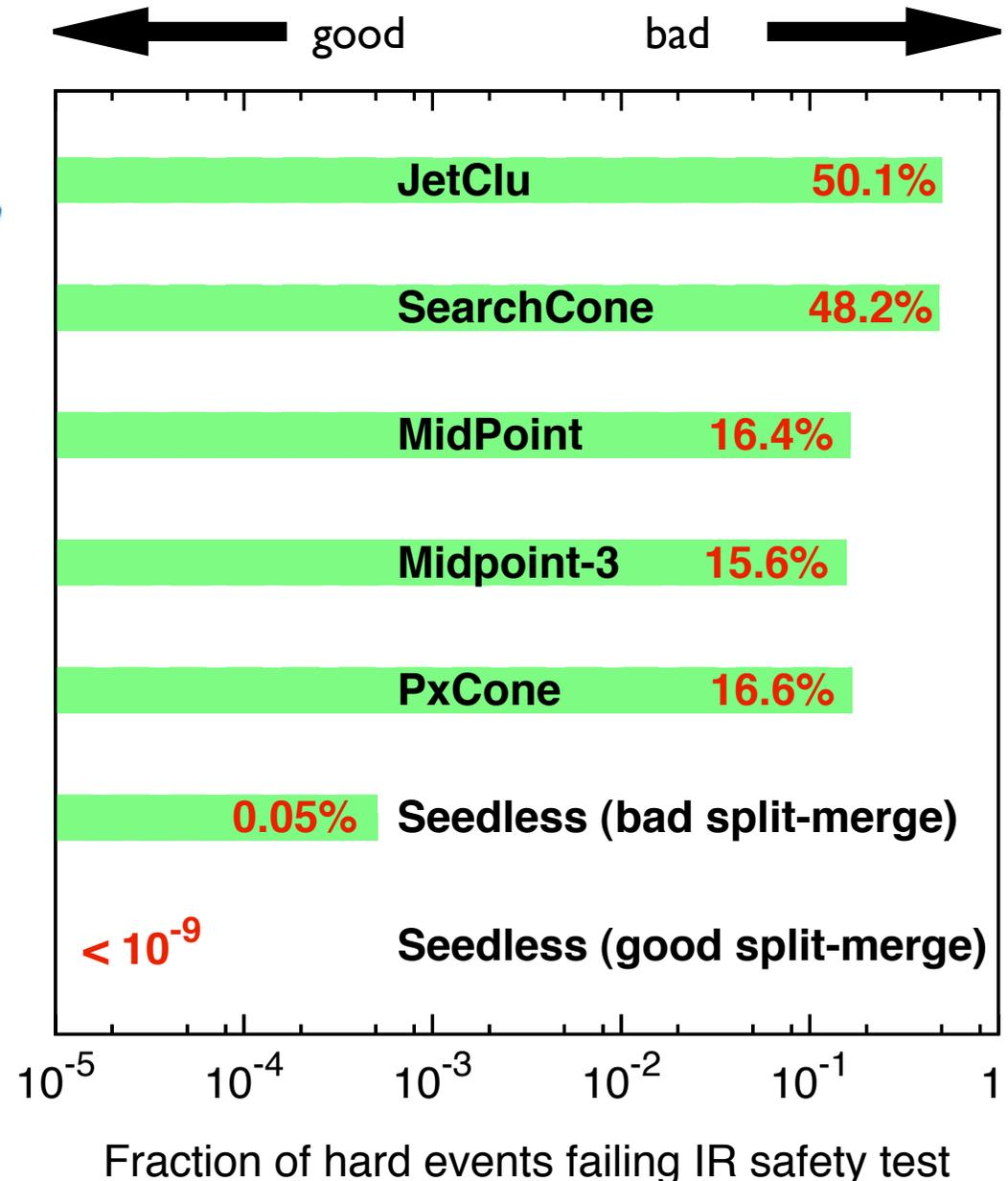
GPS & Soyez '07

Infrared (un)safety

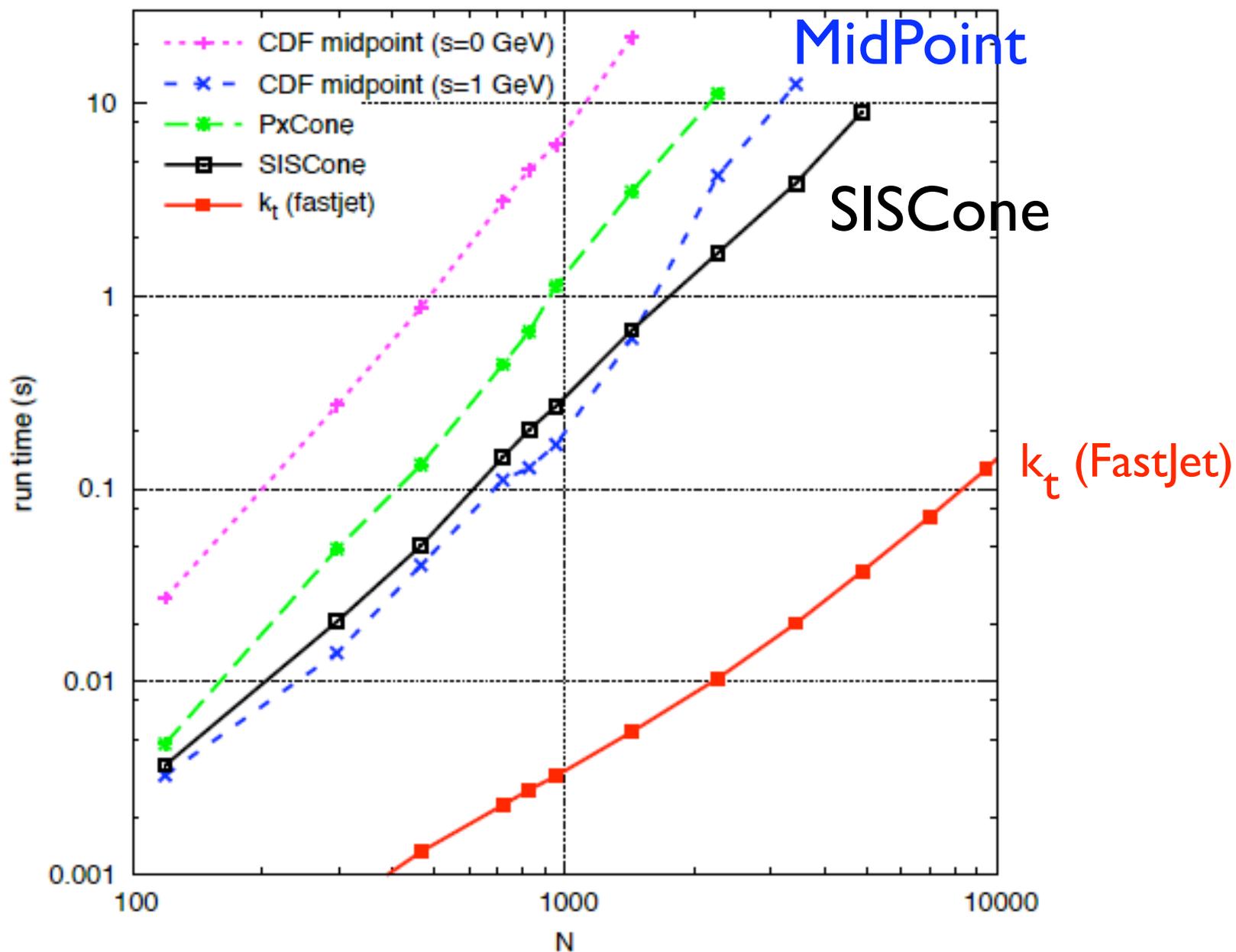
Q: How often are the hard jets changed by the addition of a soft particle?

- A:**
- ▶ Generate event with $2 < N < 10$ hard particles, find jets
 - ▶ Add $1 < N_{soft} < 5$ soft particles, find jets again [repeatedly]
 - ▶ If the jets are different, algorithm is IR unsafe.

SISCone



SISCone speed

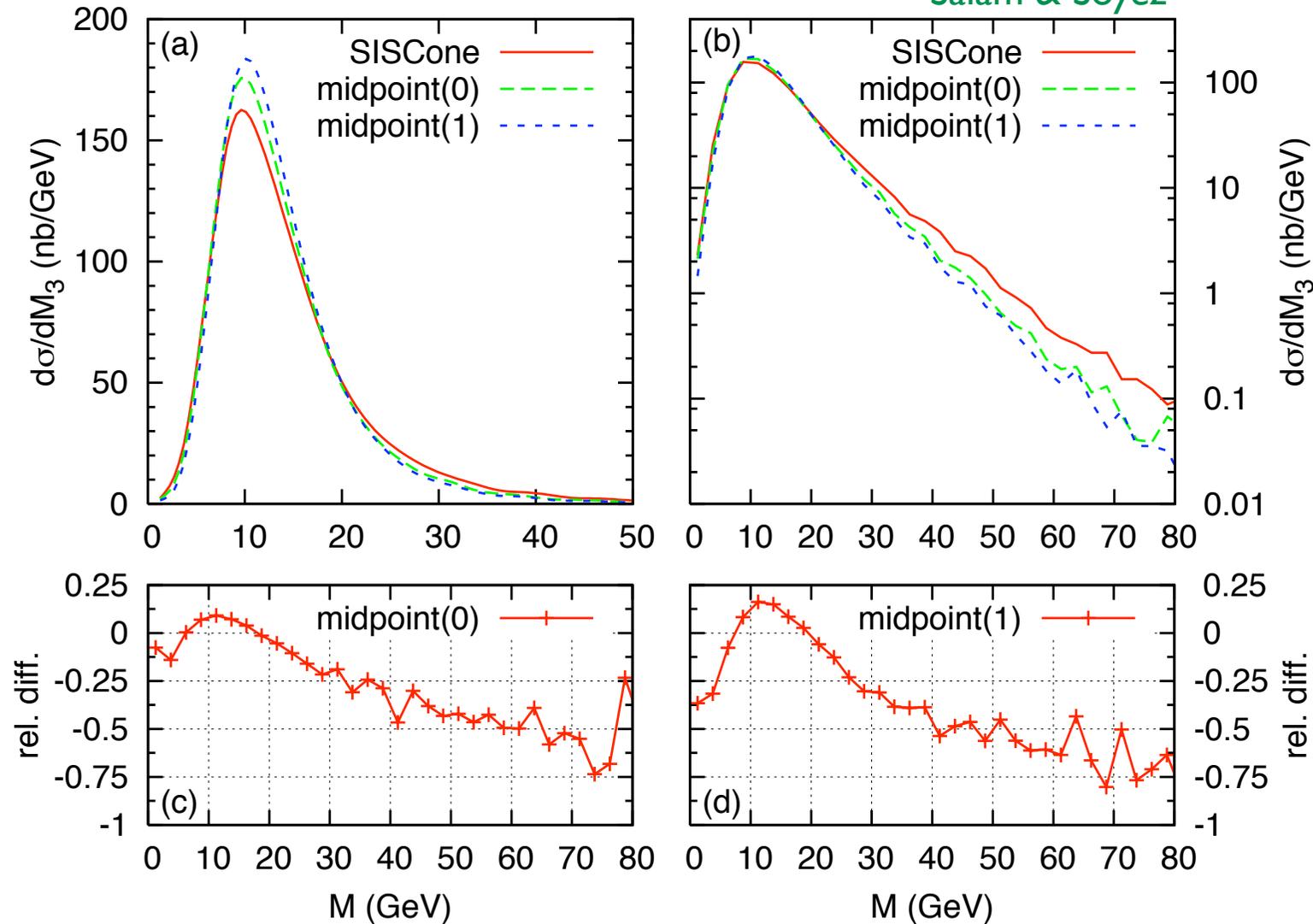


SISCone as fast as MidPoint → no penalty for infrared safety!

Jet mass

Infrared sensitivity is not just an annoying theorists' fixation

Salam & Soyez



Invariant
mass of a jet

$\frac{\text{MidPoint-SIScone}}{\text{SIScone}}$

Up to **70% difference** between MidPoint and SIScone

Recombination algorithms

k_t and Cambridge/Aachen

The definition of a sequential clustering algorithm is extremely simple.

For instance, take **the longitudinally invariant k_t** :

S. Catani, Y. Dokshitzer, M. Seymour and B. Webber, Nucl. Phys. B406 (1993) 187
S.D. Ellis and D.E. Soper, Phys. Rev. D48 (1993) 3160

- Calculate the distances between the particles: $d_{ij} = \min(k_{ti}^2, k_{tj}^2) \frac{\Delta\eta^2 + \Delta\phi^2}{R^2}$
- Calculate the beam distances: $d_{iB} = k_{ti}^2$
- Combine particles with smallest distance or, if d_{iB} is smallest, call it a jet
- Find again smallest distance and repeat procedure until no particles are left

This definition is infrared/collinear safe, has no artificial parameters, does not lead to dark towers or overlapping jets, can be applied equally well to data and theory

Variant: **Cambridge/Aachen**. Like k_t , but with $d_{ij} = \frac{\Delta\eta^2 + \Delta\phi^2}{R^2}$ and $d_{iB} = 1$

Clustering speed

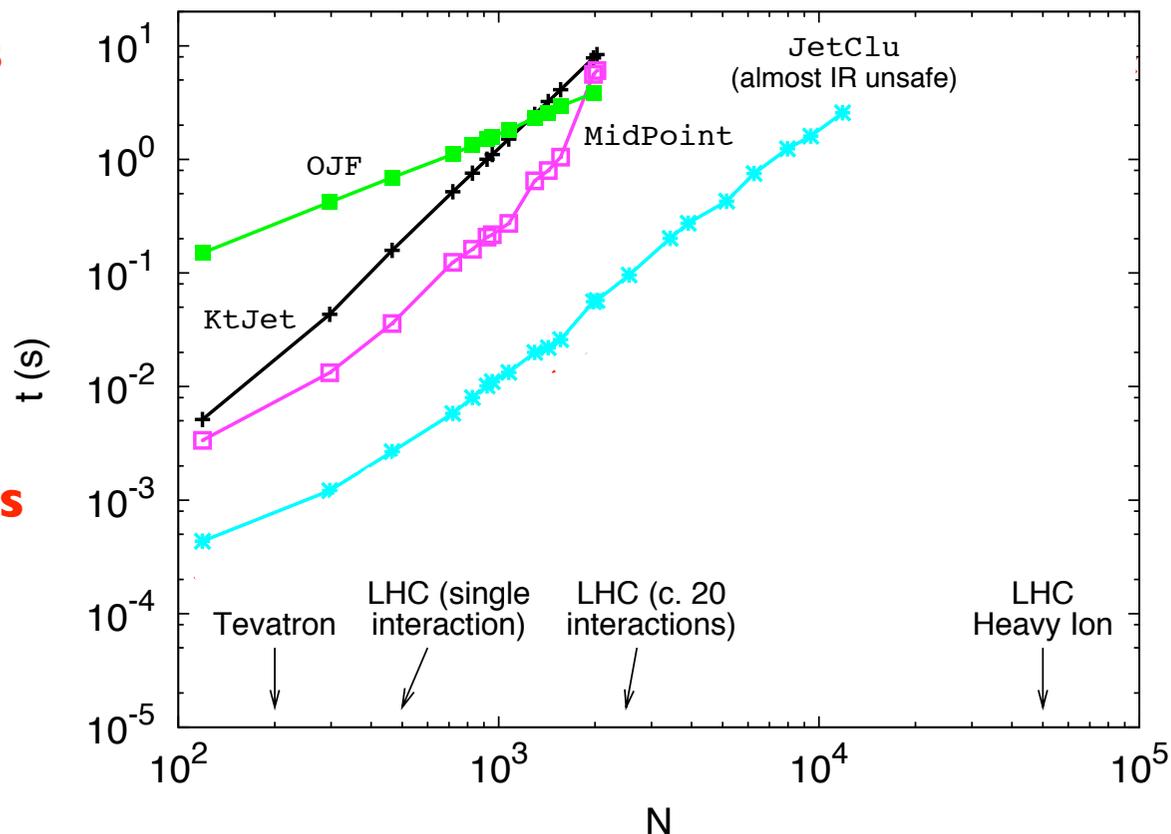
The k_t jet-finder has, however, an apparent drawback: finding all the distances is an N^2 operation, to be repeated N times

⇒ naively, the k_t jet-finder scales like N^3

Time taken to cluster N particles:

10 s

1 ms



Clustering quickly gets very slow: processing millions of events at LHC is simply not feasible with standard clustering algorithms

e.g. clustering a single heavy ion event at LHC would take 1 day of CPU!

FastJet

To improve the speed of the algorithm we must find more efficiently which particle is “close” to another and therefore gets combined with it

Observation (MC, G.P. Salam, hep-ph/0512210):

If i and j form the smallest d_{ij}

and

$$k_{ti} < k_{tj}$$



$$R_{ij} \leq R_{ik} \quad \forall k \neq j$$

i.e. j is the **geometrical** nearest neighbour of i

Translation from mathematics:

When a particle gets combined with another, and has the smallest k_t , its partner will be its **geometrical nearest neighbour** on the cylinder spanned by η and ϕ

This means that we need to look for partners only among the $O(N)$ nearest neighbours of all particles

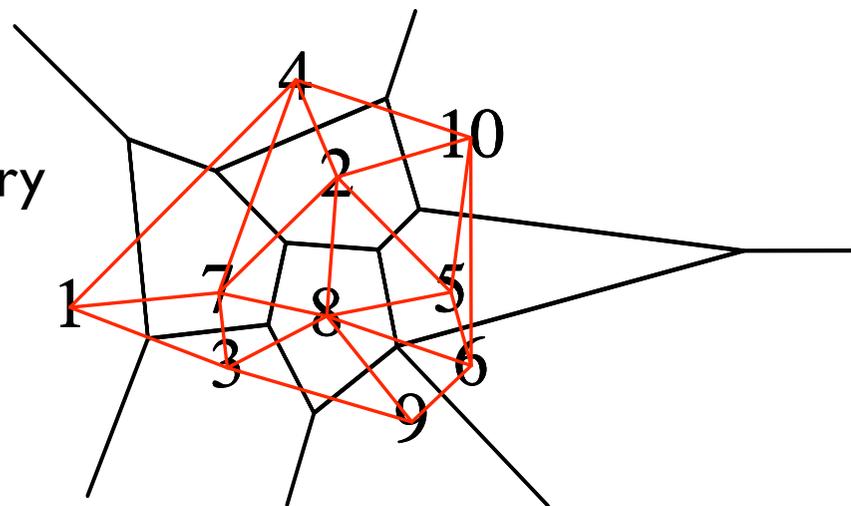
FastJet

Our problem has now become a **geometrical** one:
how to find efficiently the (nearest) neighbour(s) of a point

Widely studied problem in computational geometry

Tool: **Voronoi diagram**

Definition: each cell contains the locations which have the given point as nearest neighbour



The **dual** of a Voronoi diagram is a **Delaunay triangulation**

Key feature: once the Voronoi diagram is constructed, the nearest neighbour of a point will be in one of the $O(1)$ cells sharing an edge with its own cell

Example : the G(eometrical) N(earest) N(eighbour) of point 7 will be found among 1,4,2,8 and 3 (it turns out to be 3)

FastJet

The FastJet algorithm:

MC and G.P. Salam, hep-ph/0512210

Construct the Voronoi diagram of the N particles using the CGAL library

$O(N \ln N)$

Find the GNN of each of the N particles. Construct the d_{ij} distances, store the results in a priority queue (C++ map)

$O(N \ln N)$

Merge/eliminate particles appropriately

Update Voronoi diagram and distances' map

$O(\ln N)$



repeat N times

Overall, an $O(N \ln N)$ algorithm

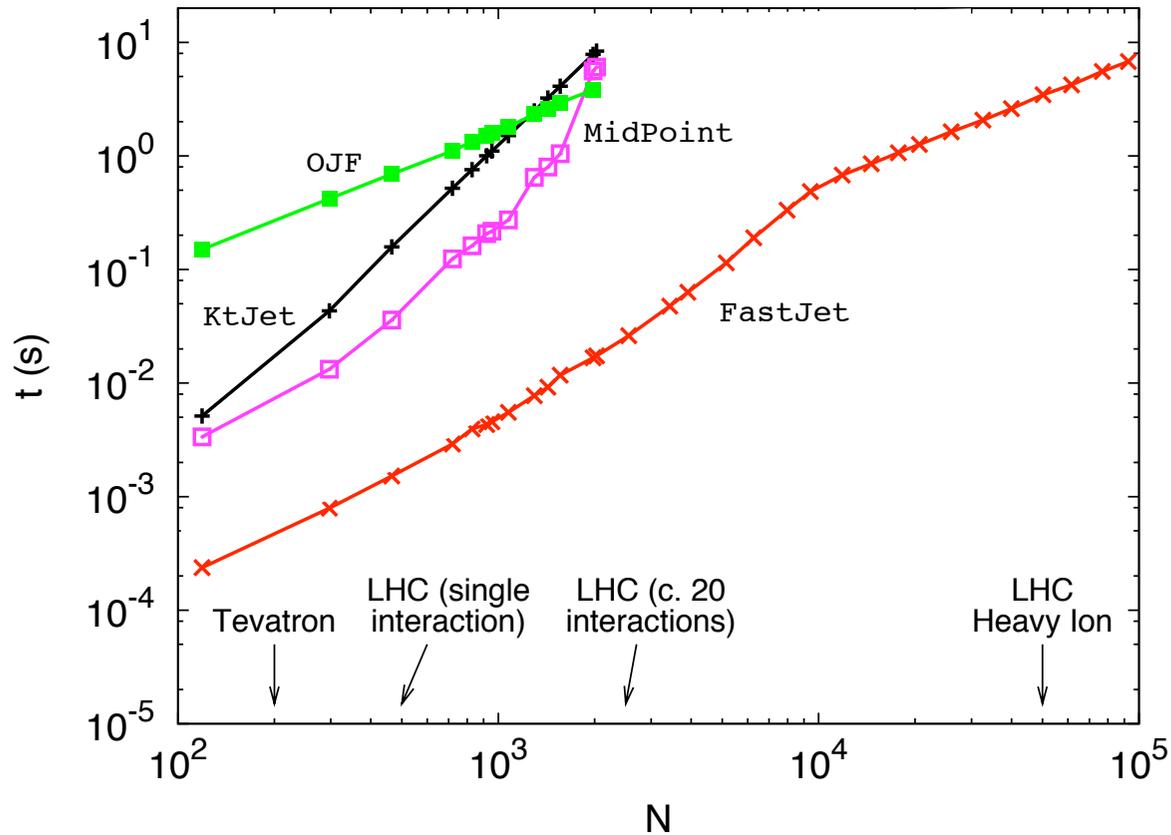
NB. Results identical to standard kt algorithm. This is NOT a new jet-finder.

FastJet performance

Time taken to cluster N particles:

10 s

1 ms



- Almost two orders of magnitude gain at small N (related $O(N^2)$ implementation)
- Large- N region now reachable

Jet areas and subtraction

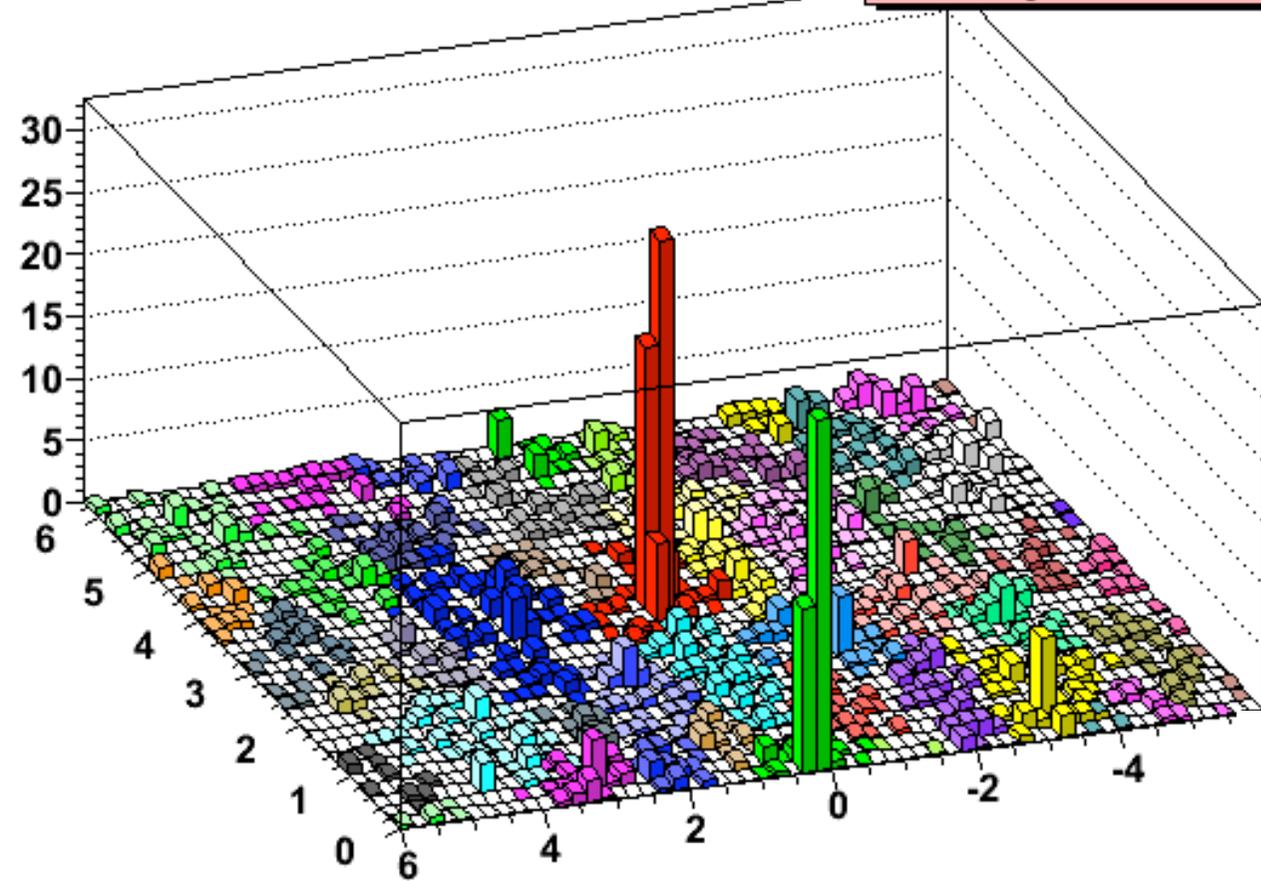
So far, old jet clustering, just better and/or faster

High speed and infrared safety allow for a **qualitatively new use** of jet clustering, through **new features**:

Jet areas

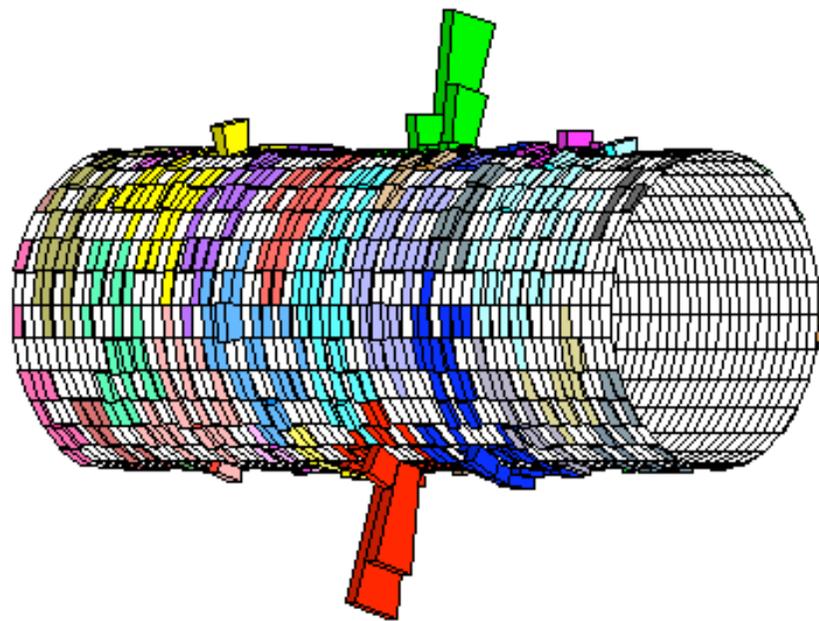
Jet areas

50GeV jets + minbias



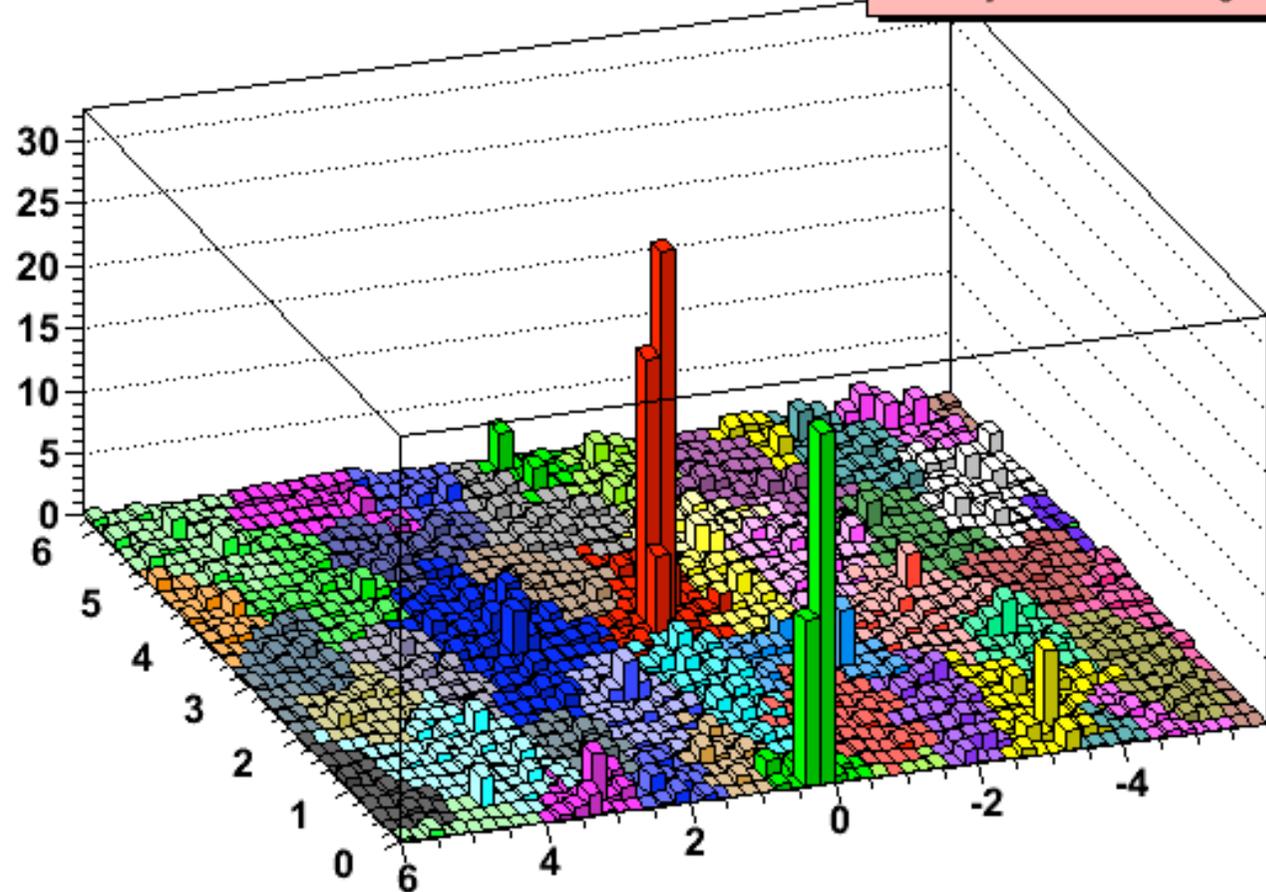
~ 2000 particles

Clustering takes $O(20\text{ s})$ with standard algorithms, but only $O(20\text{ ms})$ with FastJet



Jet areas

50GeV jets + minbias + ghosts

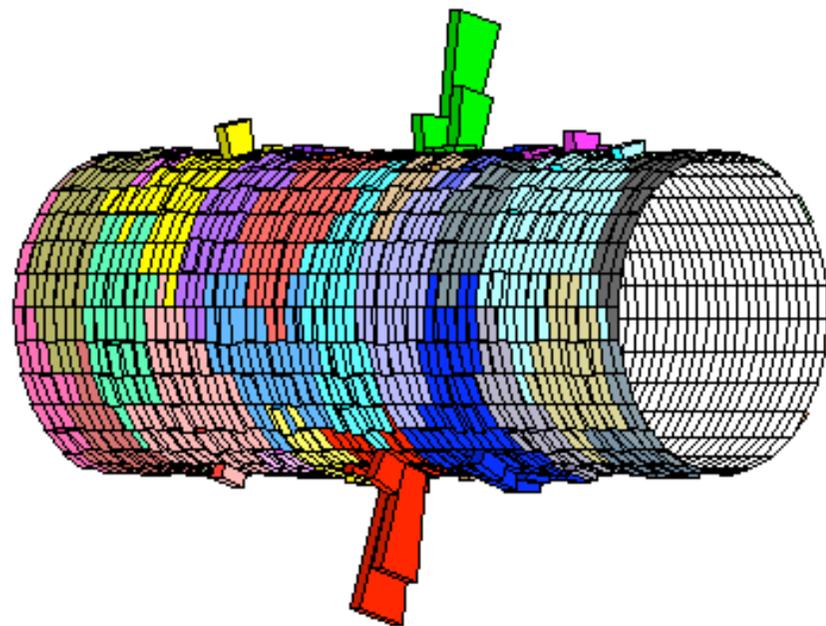


Try to estimate the **active area** of each jet
Fill event with many very soft particles, count how many are clustered into given jet

[NB. This is a **definition**]

~ 10000 particles

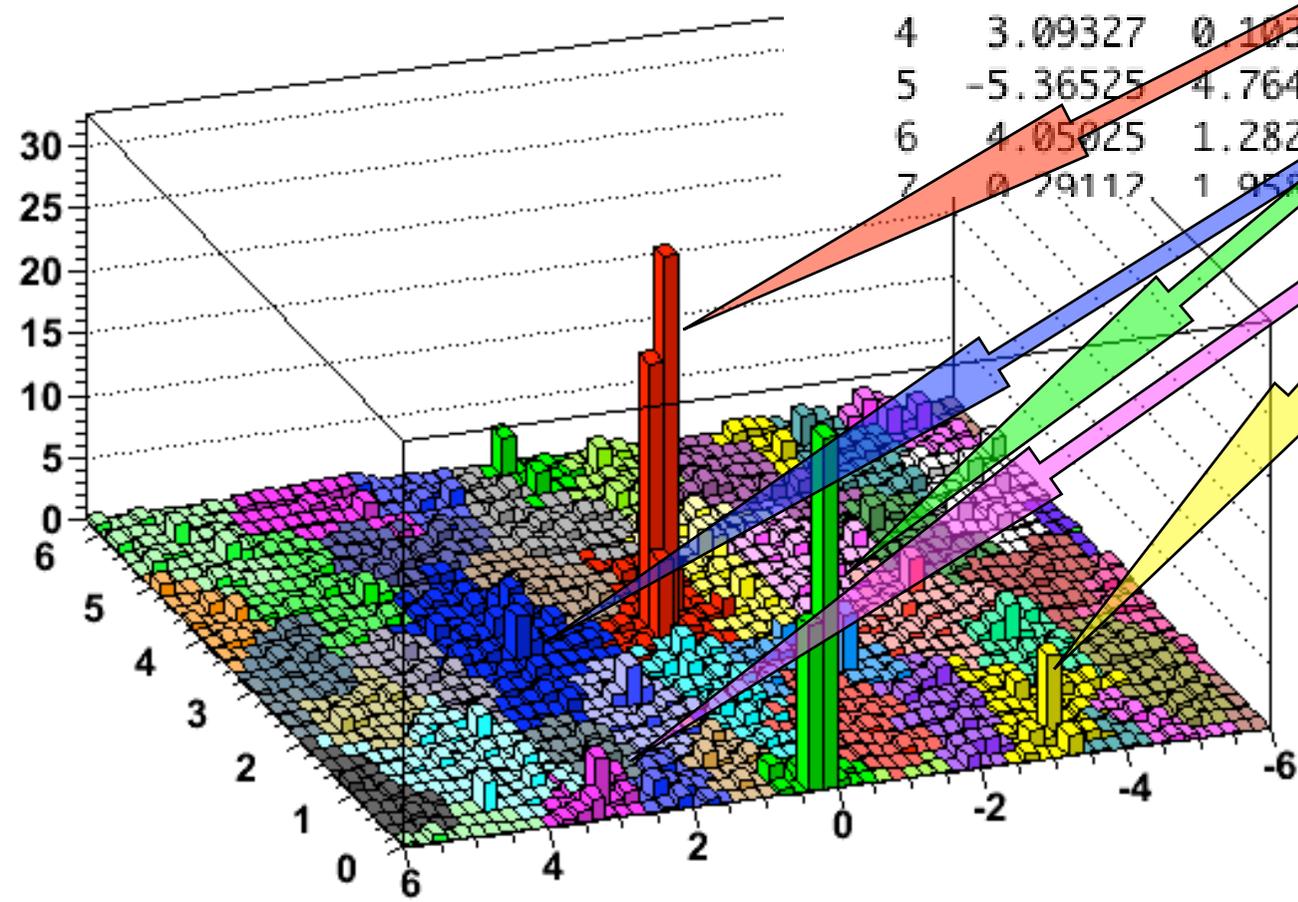
Don't even think about it with standard algorithms, $O(1\text{ s})$ with FastJet



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	26.585	3.084 +- 0.021	
4	3.09327	0.10350	20.072	2.688 +- 0.023	
5	-5.36525	4.76491	19.594	2.780 +- 0.012	
6	4.05025	1.28270	15.361	3.592 +- 0.028	
7	0.79117	1.95775	14.566	2.114 +- 0.018	



Jet areas are implemented in FastJet > v 2.0

```
// the input particles' 4-momenta
vector<fastjet::PseudoJet> input_particles;

// choose the jet algorithm
fastjet::JetDefinition jet_def(kt_algorithm,R);

// define the kind of area
fastjet::GhostedAreaSpec ghosted_area_spec(ghost_etamax);
fastjet::AreaDefinition area_def(ghosted_area_spec);

// perform the clustering
fastjet::ClusterSequence cs(input_particles,jet_def,area_def);

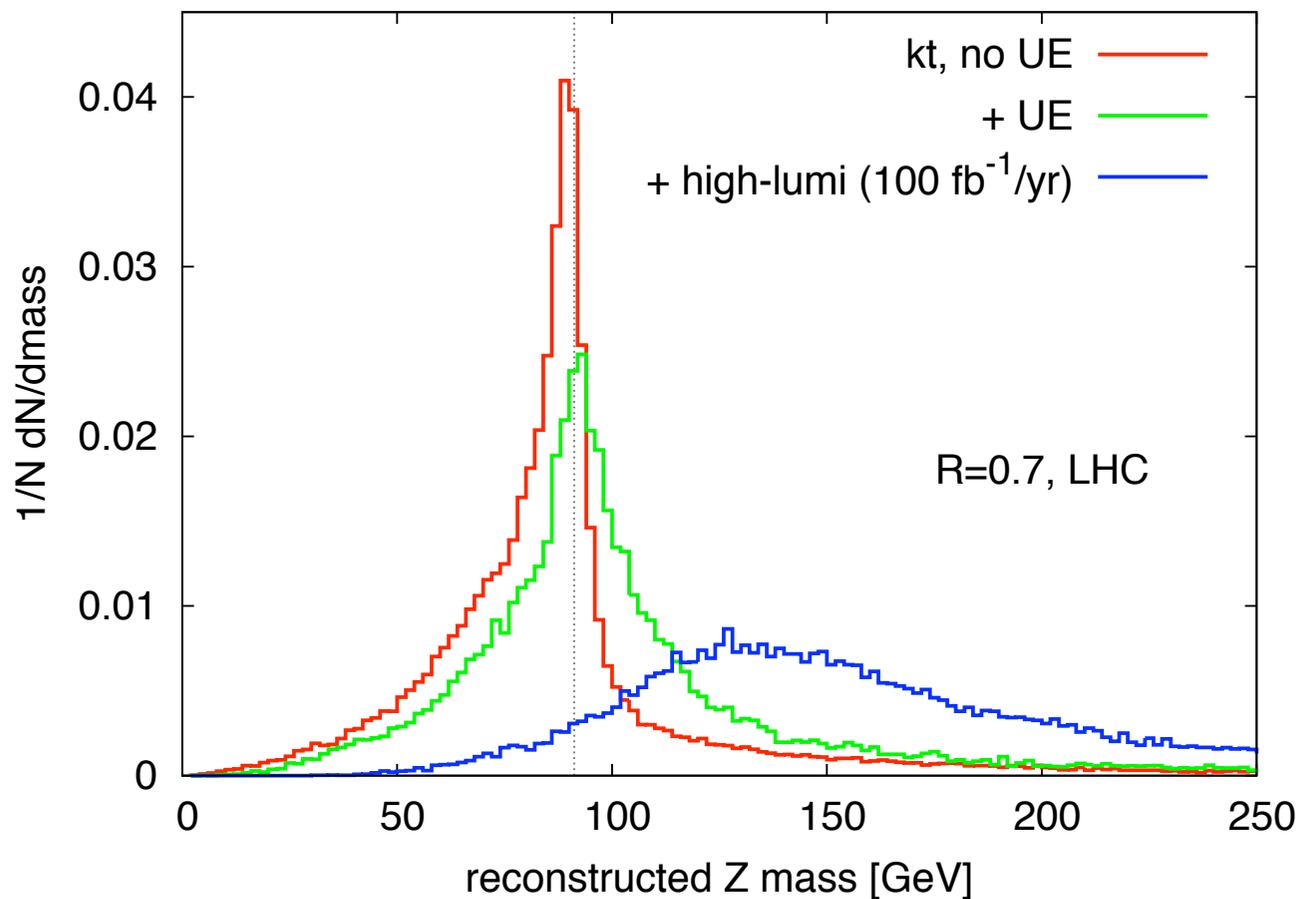
// get the jets with pt > 0
vector<fastjet::PseudoJet> jets = cs.inclusive_jets();

// a jet transverse momentum, area, and area 4-vector
double pt = jets[0].perp();
double area = cs.area(jets[0]);
fastjet::Pseudojet area_4vector = cs.area_4vector(jets[0]);
```

What do I need them for?

What are areas good for?

Challenge at high-luminosity machines:
reconstruct objects from jets when a lot of
spurious activity is present



You'd like to be able to
subtract this extra stuff
from the jets and get back
to the correct Z mass

Can knowledge of jet areas help?

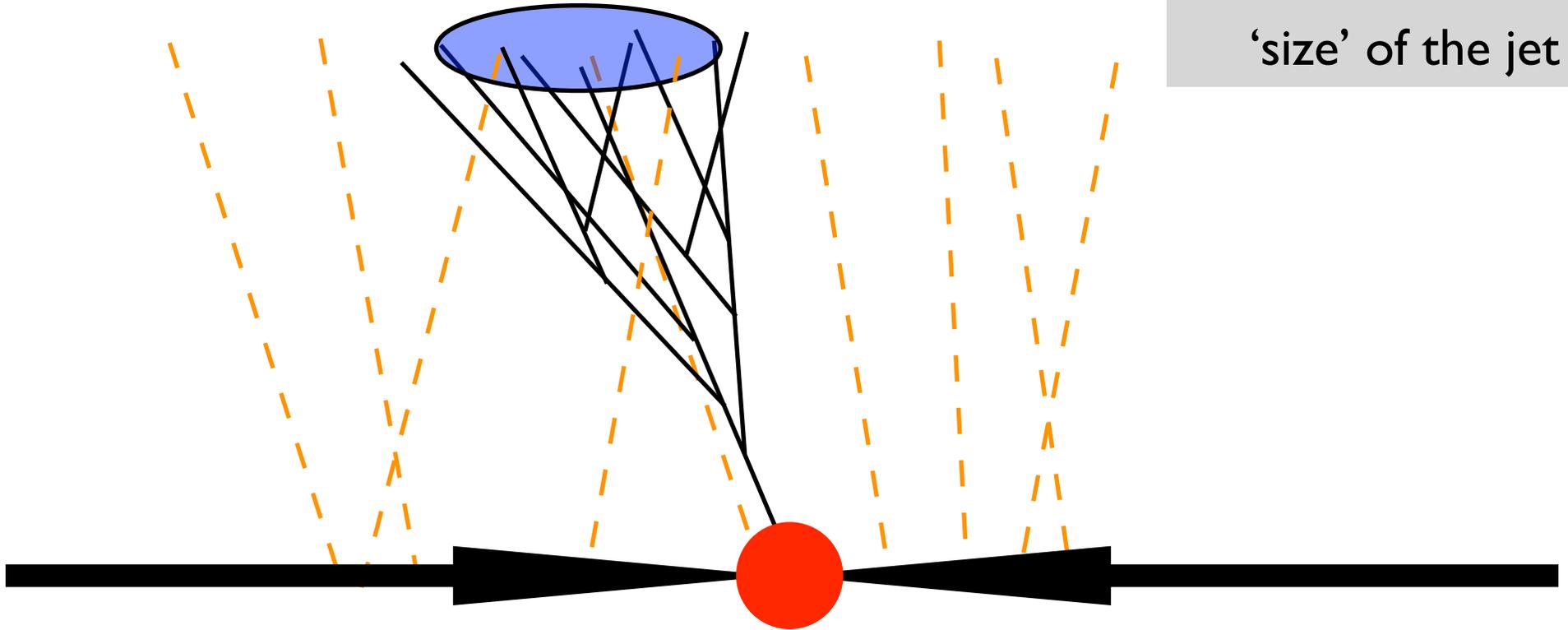
The intuitive picture

$$p_T(\text{jet}) \sim p_T(\text{parton}) +$$

Average underlying
momentum density

×

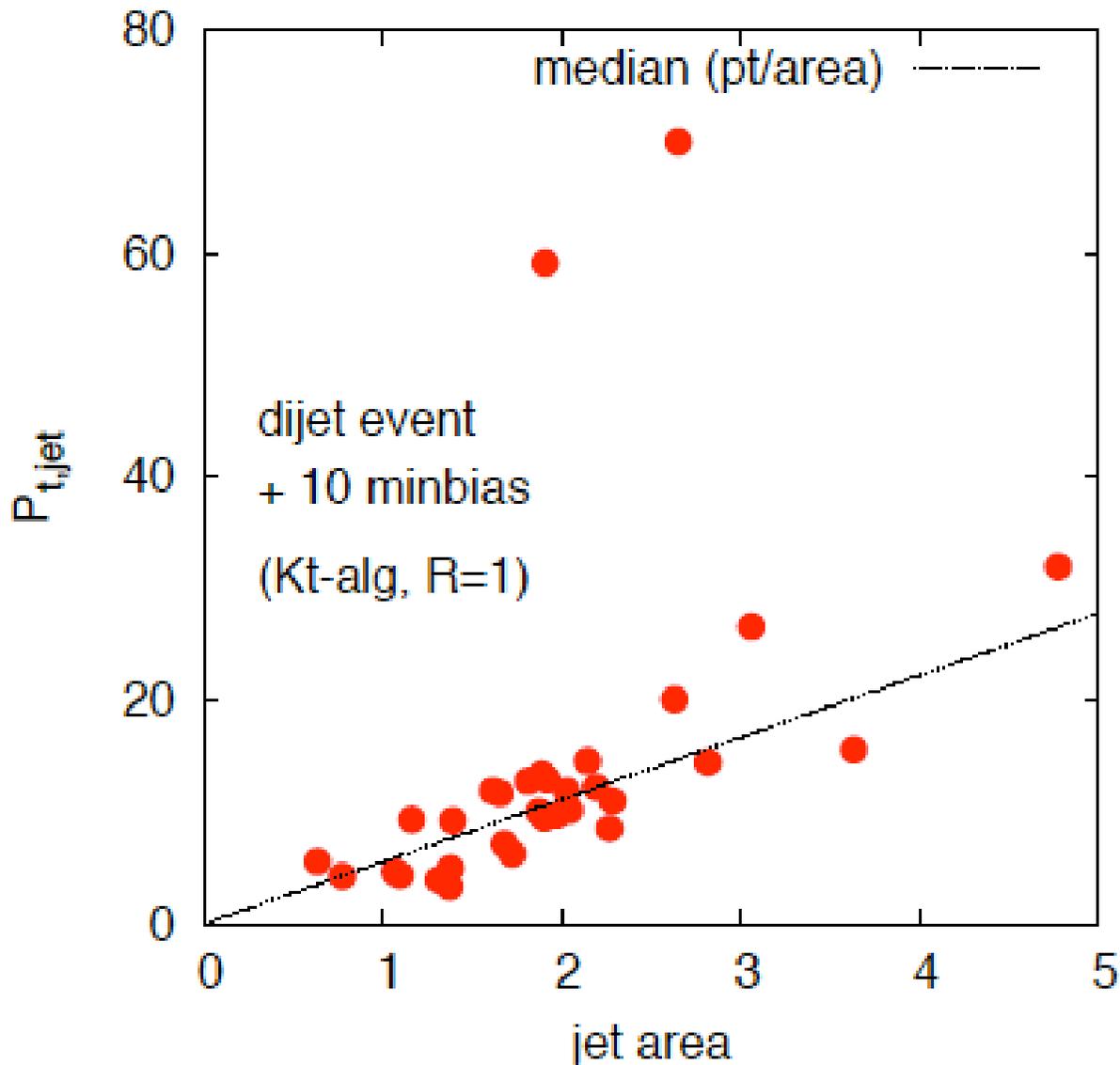
'size' of the jet



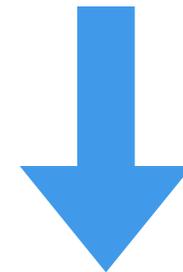
The 'size' of the jet can be the **active area** we just defined

But how do we get the **momentum density** of the radiation?

Areas distribution



The jets adapt to the surrounding environment

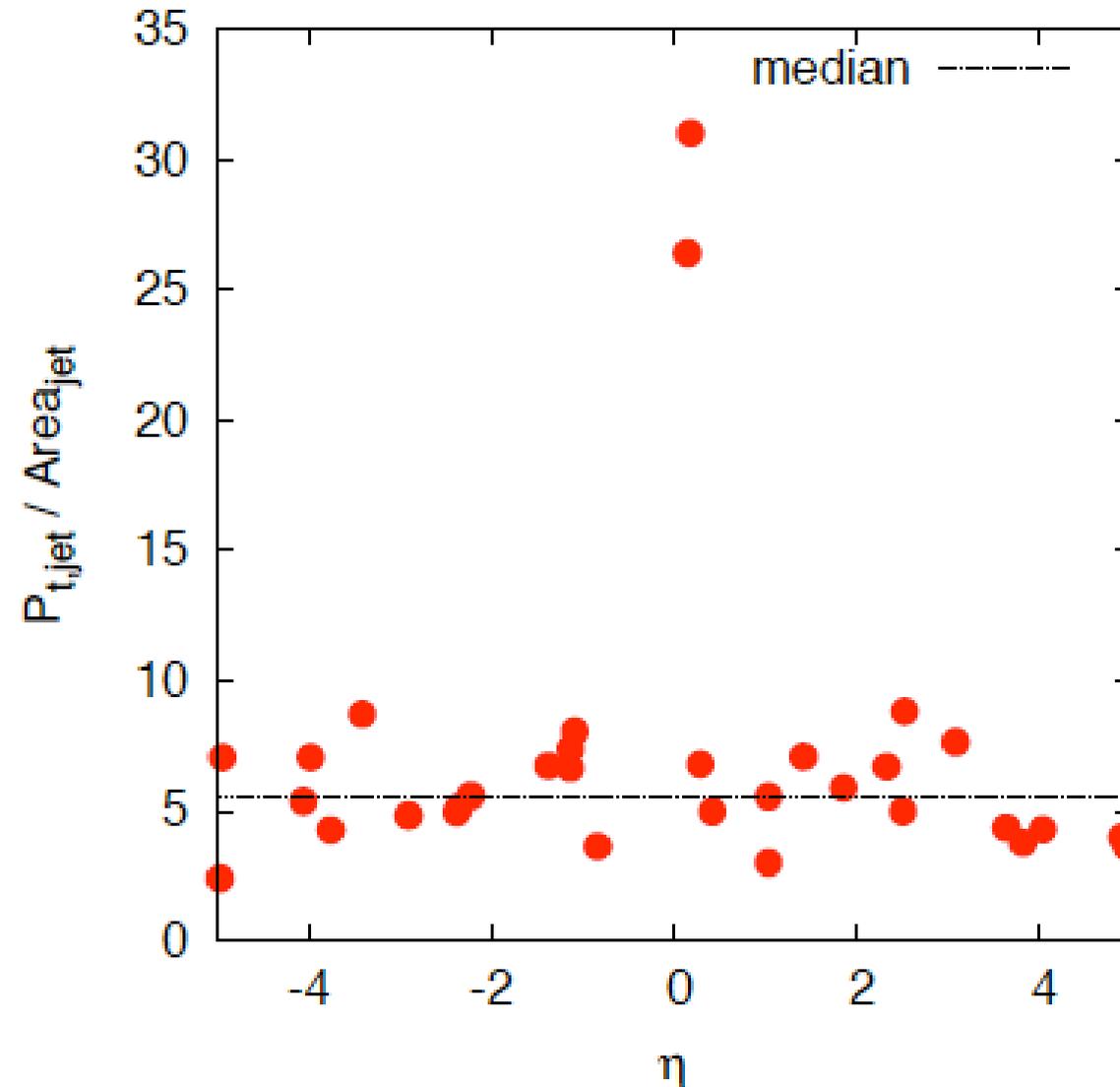


They can have very different areas

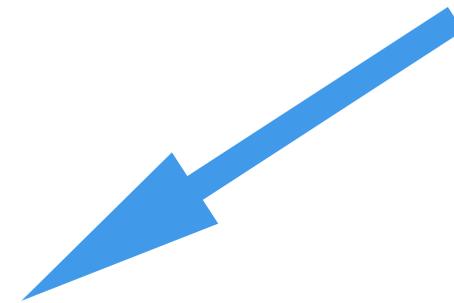
Area vs. p_T

Key observation:

p_T /Area is fairly constant, except for the hard jets



The distribution of background jets establishes its own average momentum density



(NB. this is true on an **event-by-event** basis)

Subtraction

- A proper operative definition of **jet area** can be given
- When a hard event is superimposed on a **roughly uniformly distributed background**, study of **transverse momentum/area** of each jet allows one to determine the noise density ρ (and its fluctuation) on an event-by-event basis
- Once measured, the background density can be used to correct the transverse momentum of the hard jets:

$$p_T^{\text{hard jet, corrected}} = p_T^{\text{hard jet, raw}} - \rho \times \text{Area}_{\text{hard jet}}$$

The subtraction

```
// the input particles' 4-momenta
vector<fastjet::PseudoJet> input_particles;

// choose the jet algorithm
fastjet::JetDefinition jet_def(kt_algorithm,R);

// define the kind of area
fastjet::GhostedAreaSpec ghosted_area_spec(ghost_etamax);
fastjet::AreaDefinition area_def(ghosted_area_spec);

// perform the clustering
fastjet::ClusterSequence cs(input_particles,jet_def,area_def);

// get the jets with pt > 0
vector<fastjet::PseudoJet> jets = cs.inclusive_jets();

// a jet transverse momentum, area, and area 4-vector
double pt = jets[0].perp();
double area = cs.area(jets[0]);
fastjet::Pseudojet area_4vector = cs.area_4vector(jets[0]);
```

```
// get the median, i.e. rho
double rho = cs.median_pt_per_unit_area(rapmax);
double rho_4v = cs.median_pt_per_unit_area_4vector(rapmax);

// subtract
double pt_sub = pt - rho * area;
fastjet::Pseudojet p_sub = jets[0] - rho_4v * area_4vector;
```

NB. The “_4vector” variants also correct jet directions, and are better for large R

Examples of UE/MB subtraction using FastJet and area method

Preliminary results (MC & GPS) for

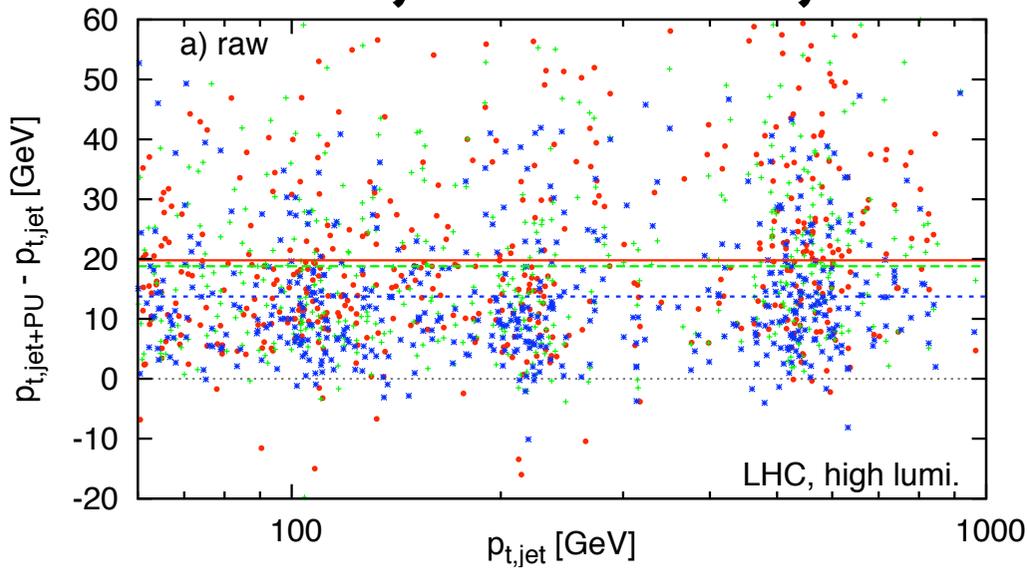
- ▶ High-lumi LHC
 - ▶ Z production
 - ▶ Z' (mass = 2 TeV)
 - ▶ W bosons in $t\bar{t}$ events
 - ▶ ...
- ▶ Heavy ion collisions
 - ▶ inclusive jet distribution in Pb-Pb collisions

Dijet subtraction

LHC: dijet + pileup

$R = 0.7$

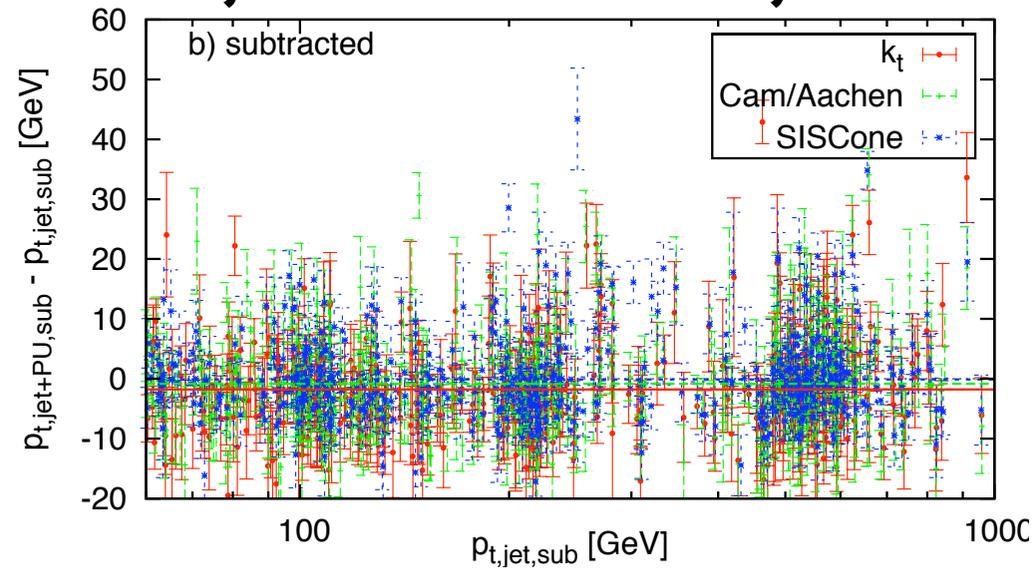
$P_{t,\text{jet+PU}} - P_{t,\text{jet}}$



raw

(on average 20 GeV added to jets)

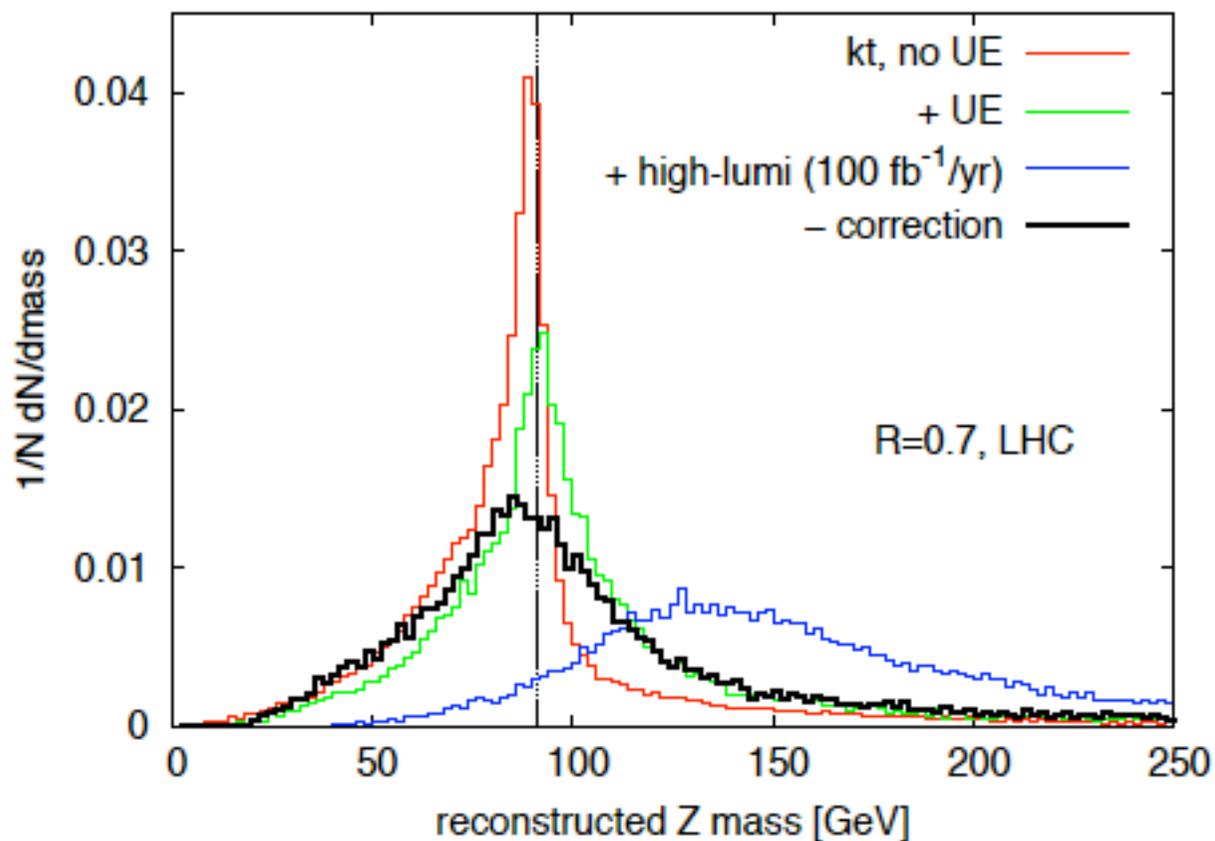
$P_{t,\text{jet+PU sub}} - P_{t,\text{jet sub}}$



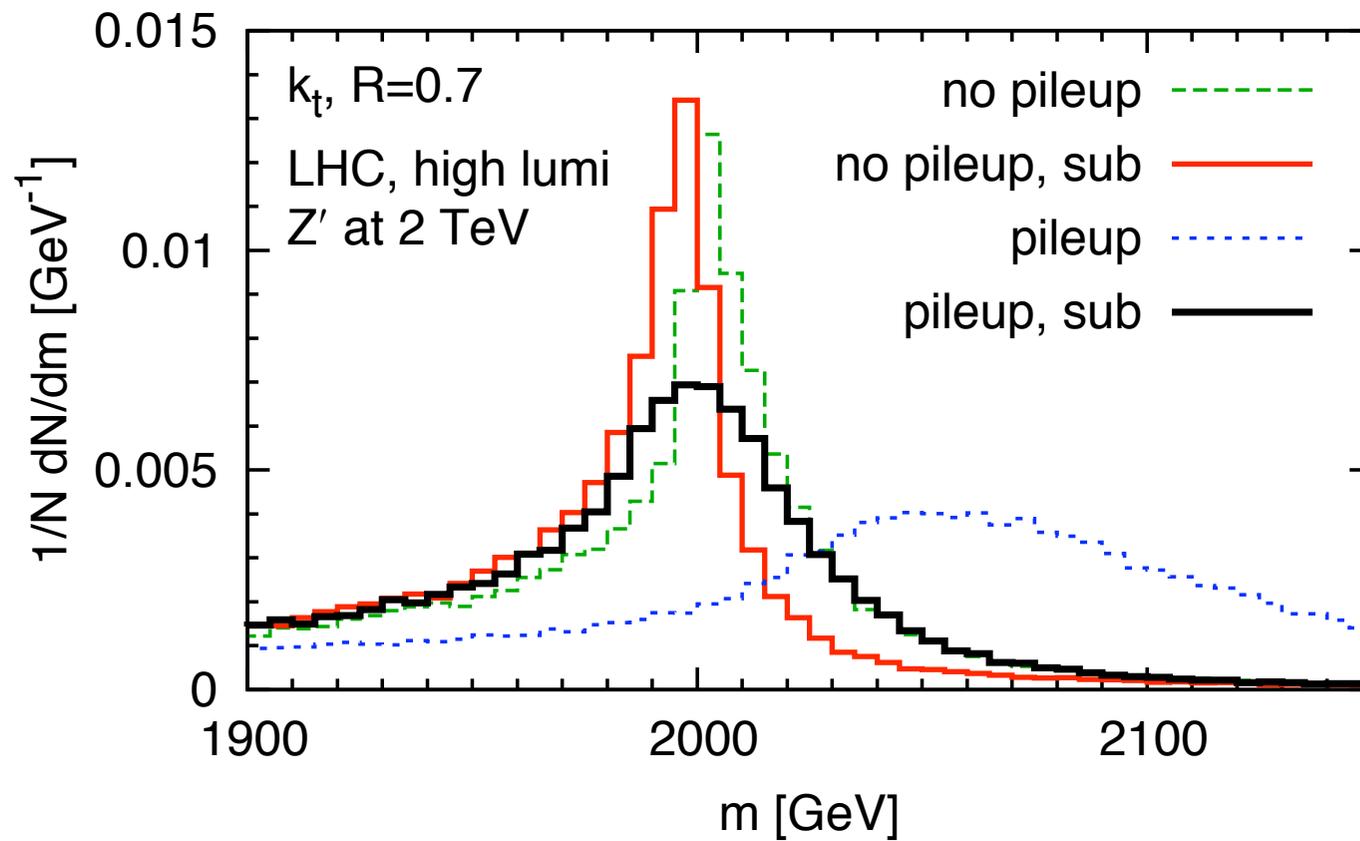
subtracted

Use jet areas to correct jet kinematics

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

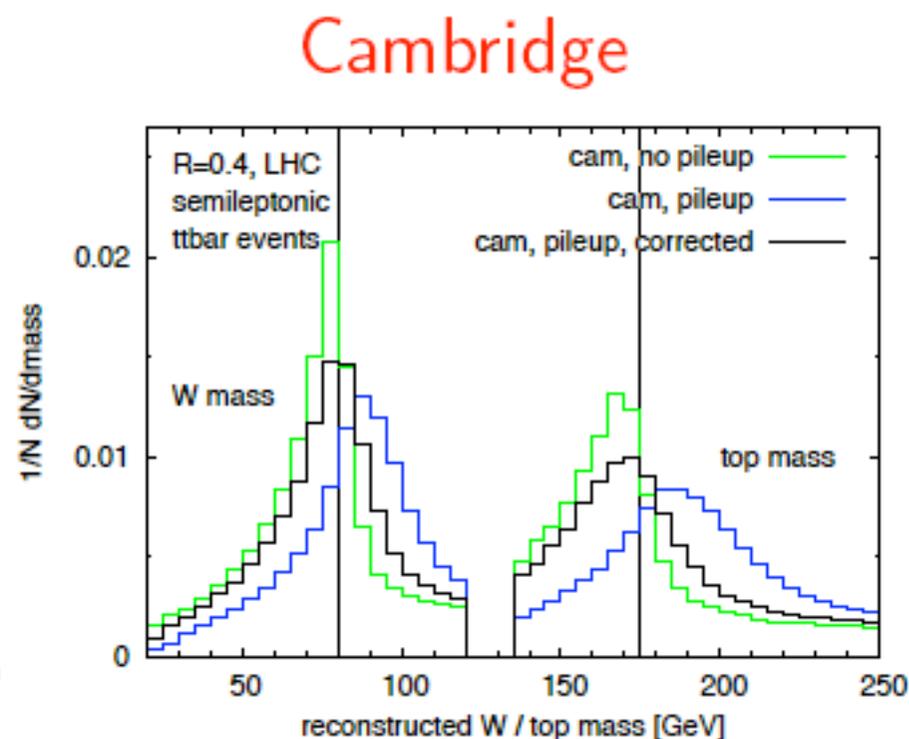
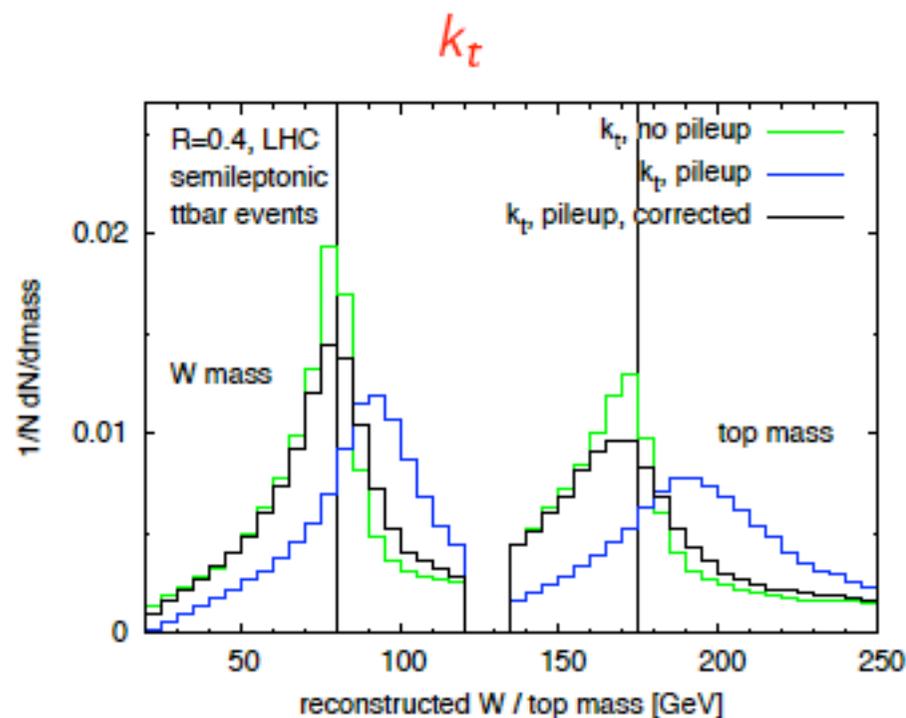


Reconstructed Z' mass



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

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

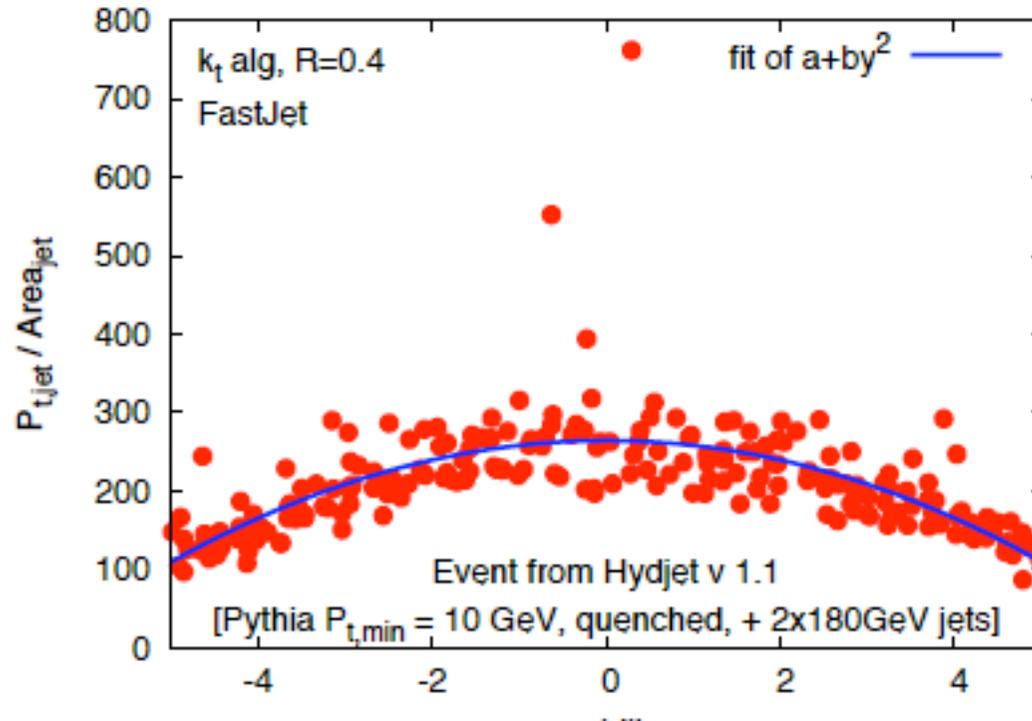


At LHC one expects ~ 30000 particles per Pb-Pb collisions

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

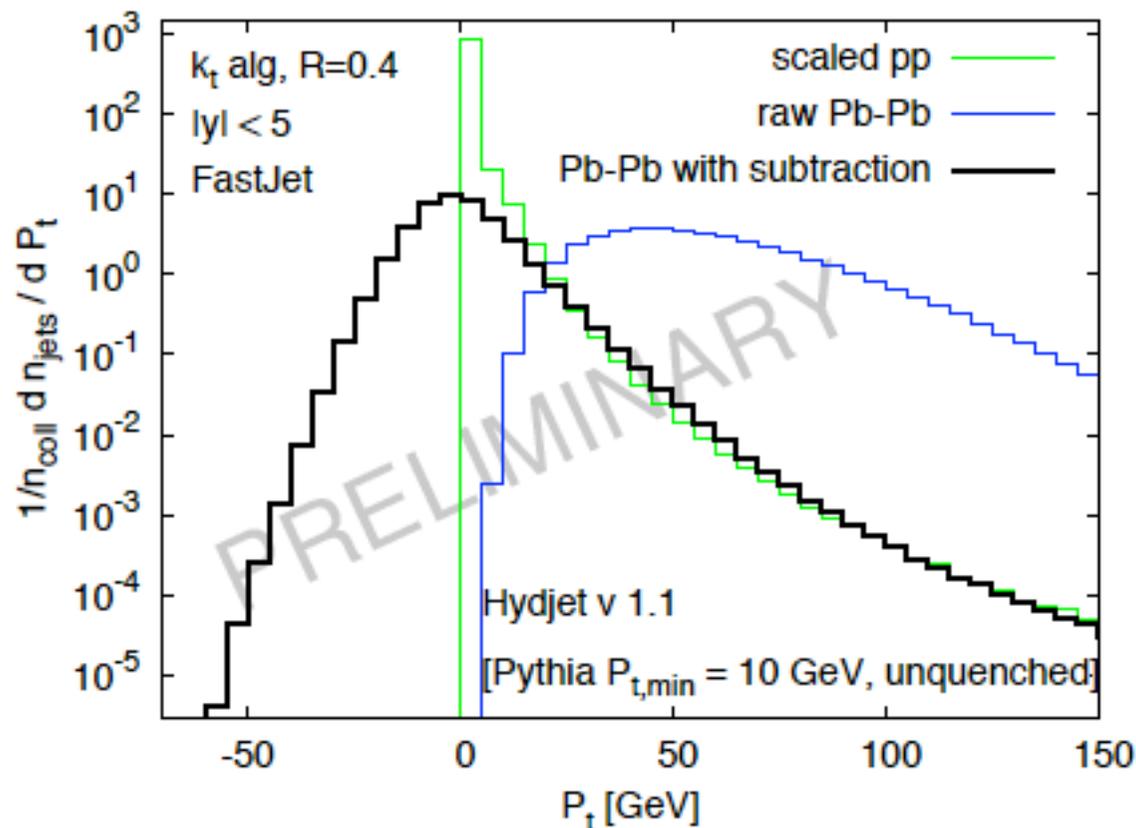


NB1: the simulation of a heavy ions collision suggests a parabolic fit of the background

NB2: no minimum p_T cut will ever be used

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

Conclusions

- Cone and recombination are alternative and complementary approaches to defining jets
- So far, cone algorithms were extremely messy and generally infrared unsafe. Now we finally have a really infrared safe (and reasonably fast) cone algorithm, **SISCone**. Phenomenology will have to follow
- Recombination algorithms like kt and Cambridge enjoy much simpler definitions. They are always infrared safe
- **FastJet** (<http://www.lpthe.jussieu.fr/~salam/fastjet>) resolves the speed issue, and allows one to calculate the area of jets
- The area of jets can be used for background subtraction, opening the way to a more widespread use of kt/cambridge clustering in high luminosity and heavy ions collisions environments