Summer School on Symmetries,
Fundamental Interactions and Cosmology
September 2018, Fraueninsel/Chiemsee

Jets

Matteo Cacciari
LPTHE Paris and Université Paris Diderot

Lecture 1 - Jet algorithms

Lecture 2 - Jet substructure

[Includes material from
Gavin Salam and Grégory Soyez]

Outline

- Jet algorithms
  - How are jets made
- Jet substructure
  - What's inside them

What is a jet?

No, not this....

A **jet** is something that happens
in high energy events:
a collimated bunch of hadrons flying roughly in the same direction

Gluon 'discovery'

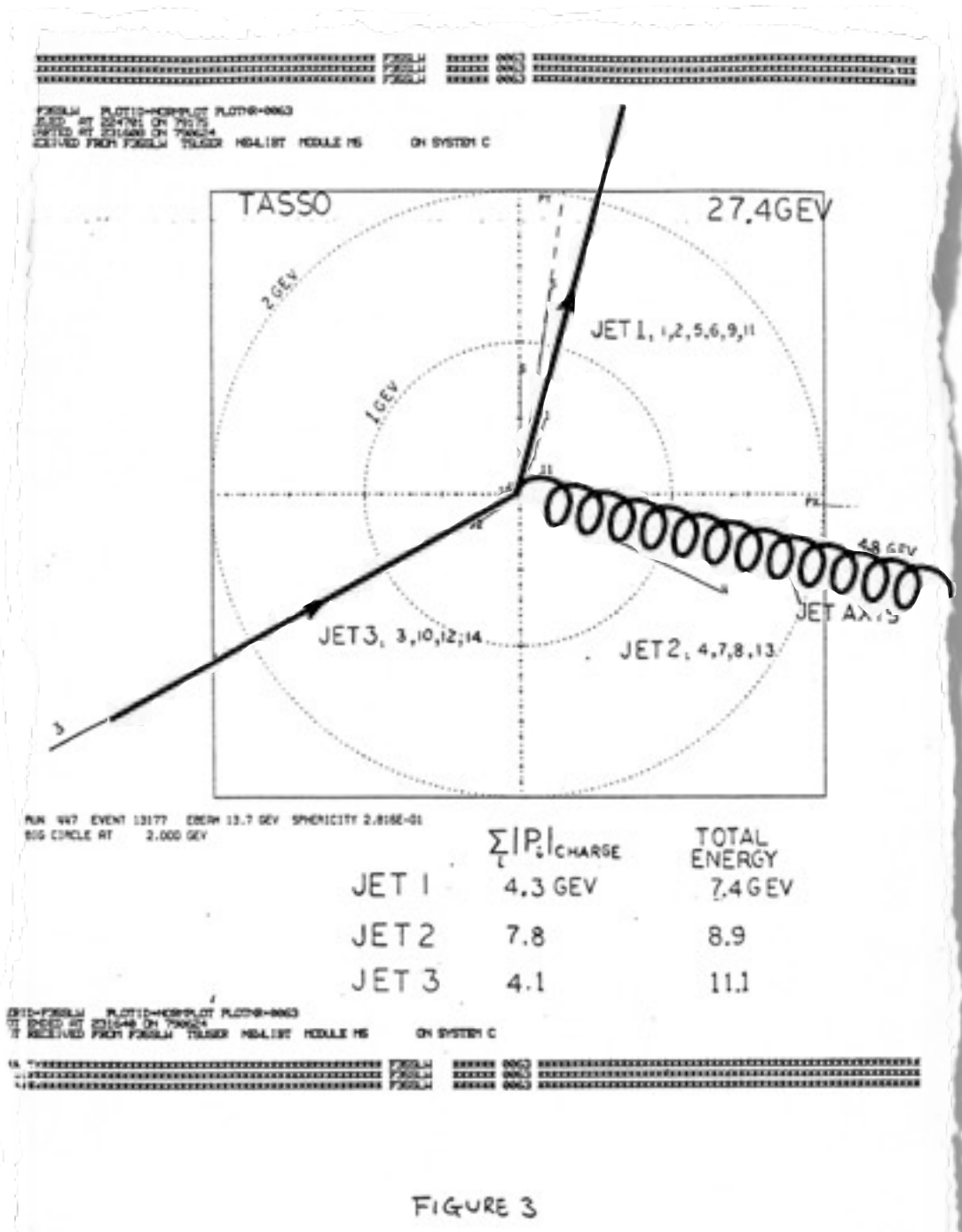


FIGURE 3

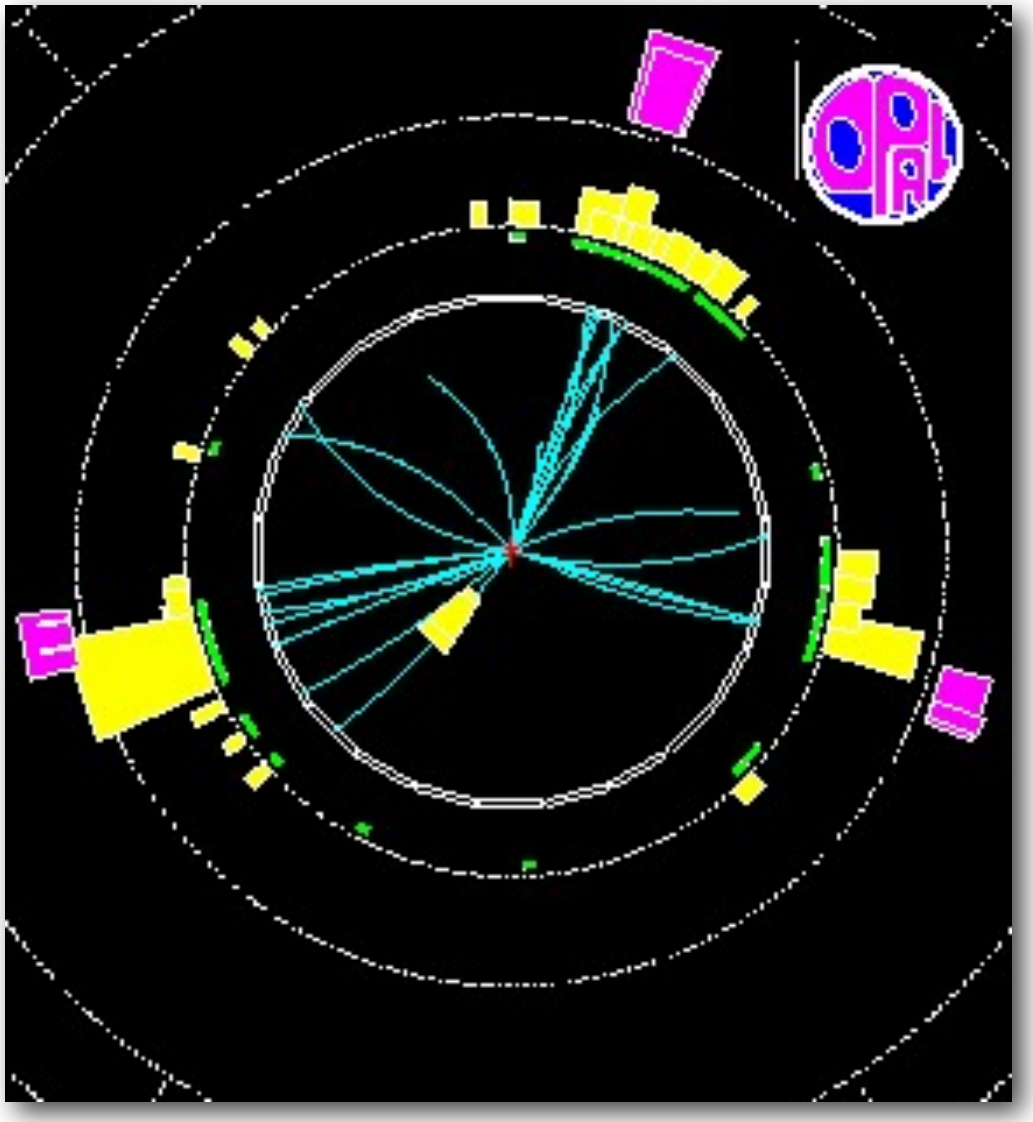
1979:

Three-jet events observed by TASSO, JADE, MARK J and PLUTO at PETRA in e^+e^- collisions at 27.4 GeV

Interpretation:
large angle emission of a hard gluon

Jets viewed as a proxy to the initial partons

Why jets



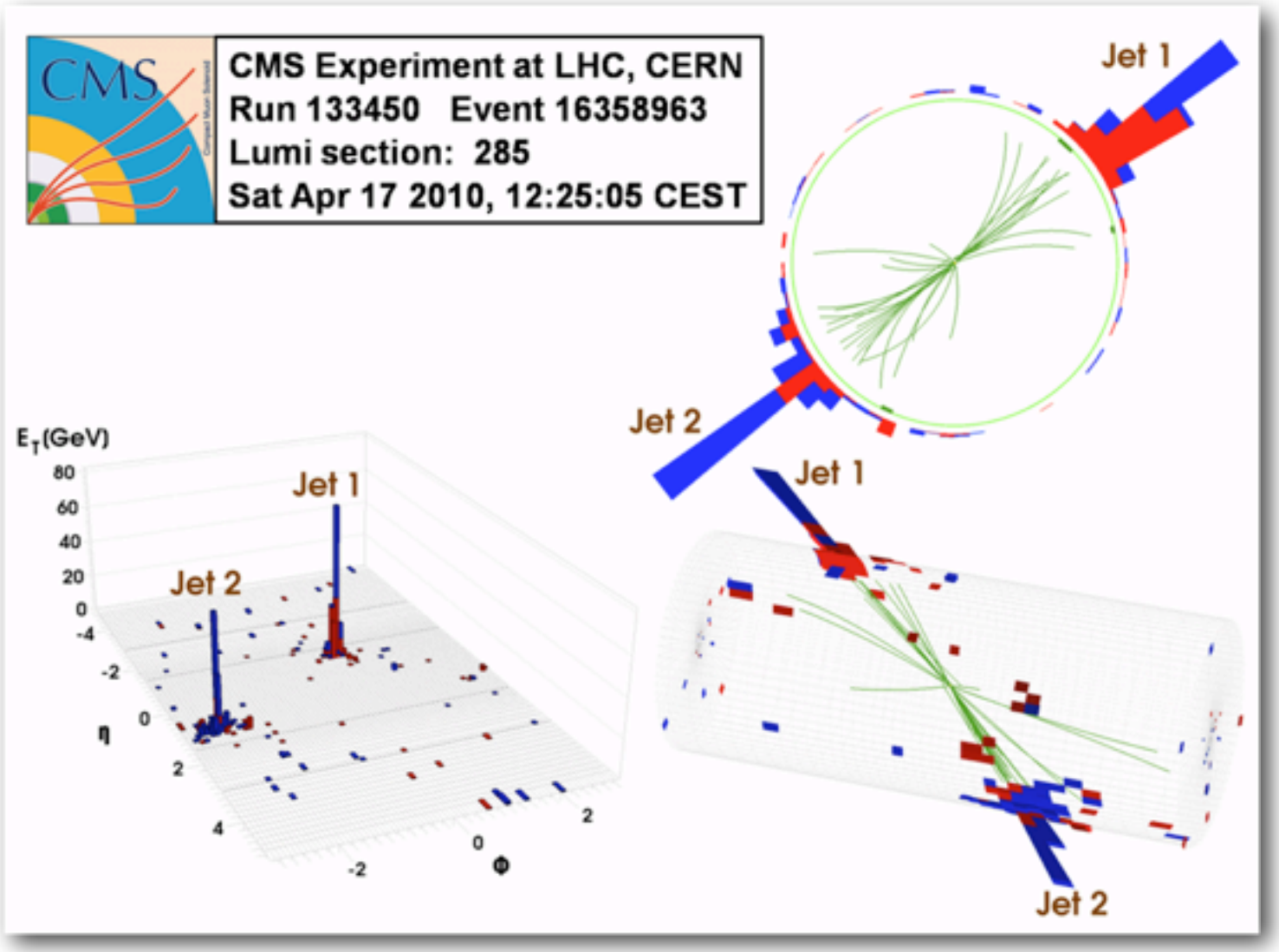
From PETRA to LEP

We could eyeball the collimated bunches, but it becomes impractical with millions of events

The classification of particles into jets is best done using a **clustering algorithm**

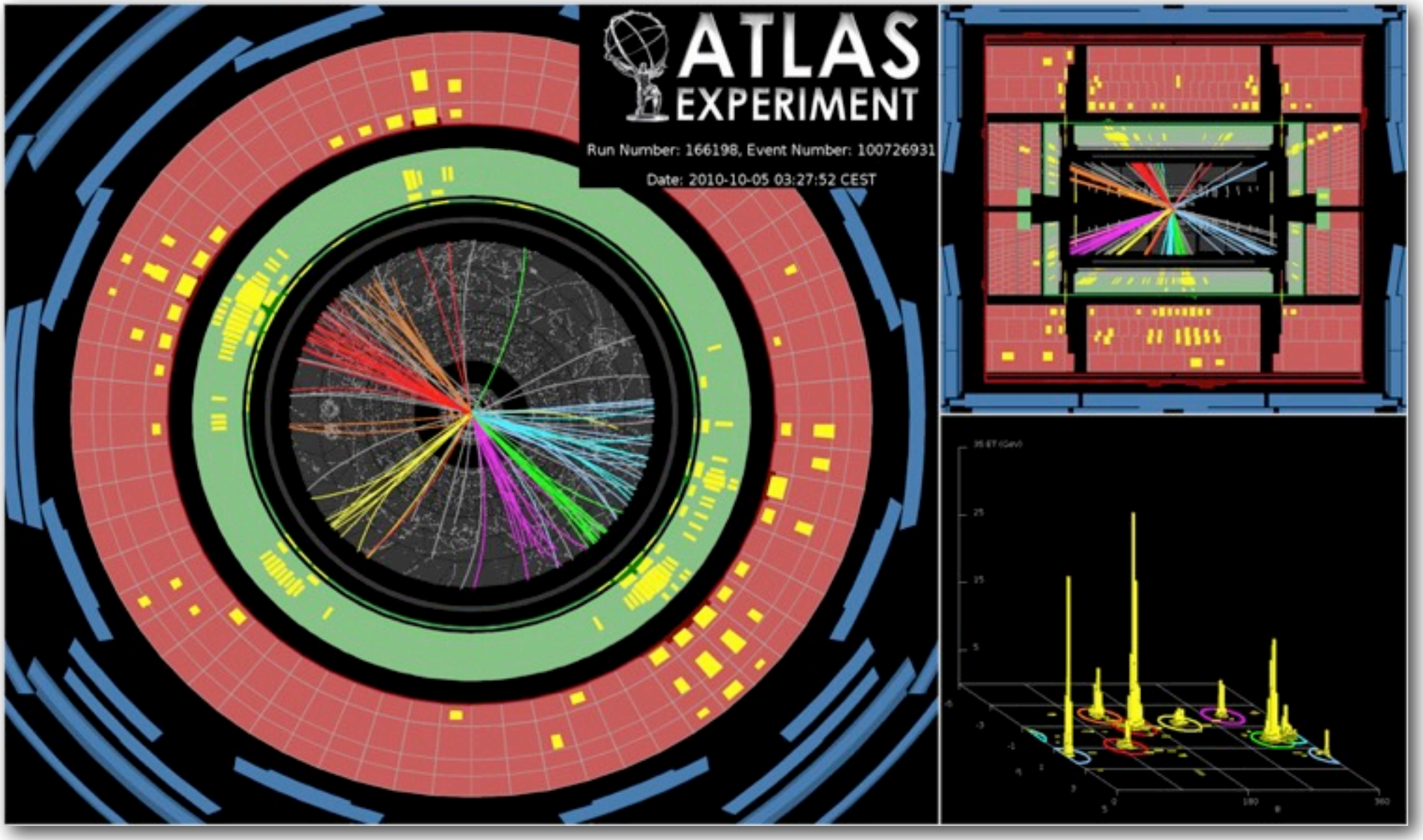
Jets @ LHC

A few decades after PETRA and LEP, the event displays got prettier, but jets are still pretty much the same



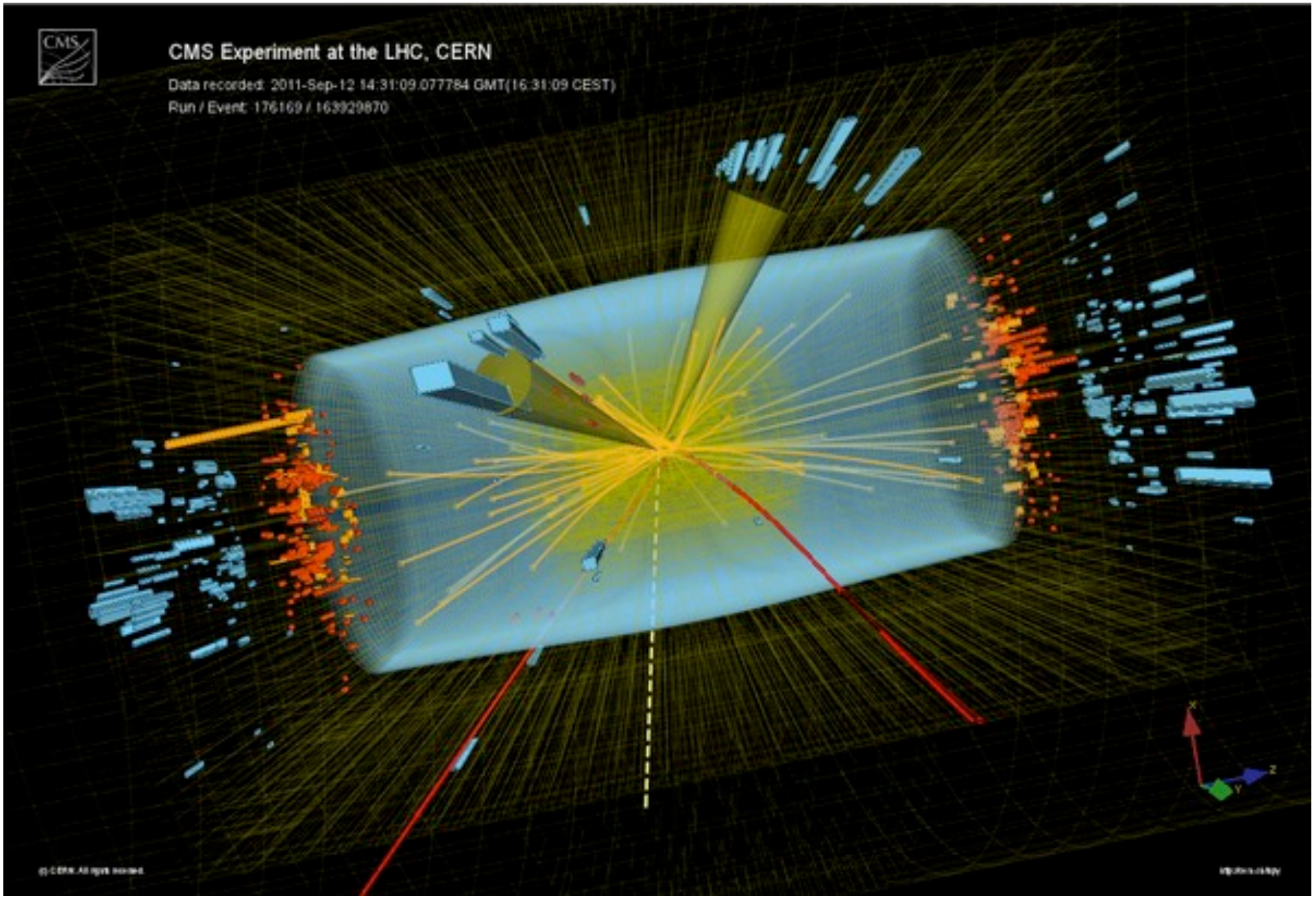
Dijet event from CMS

Jets @ LHC

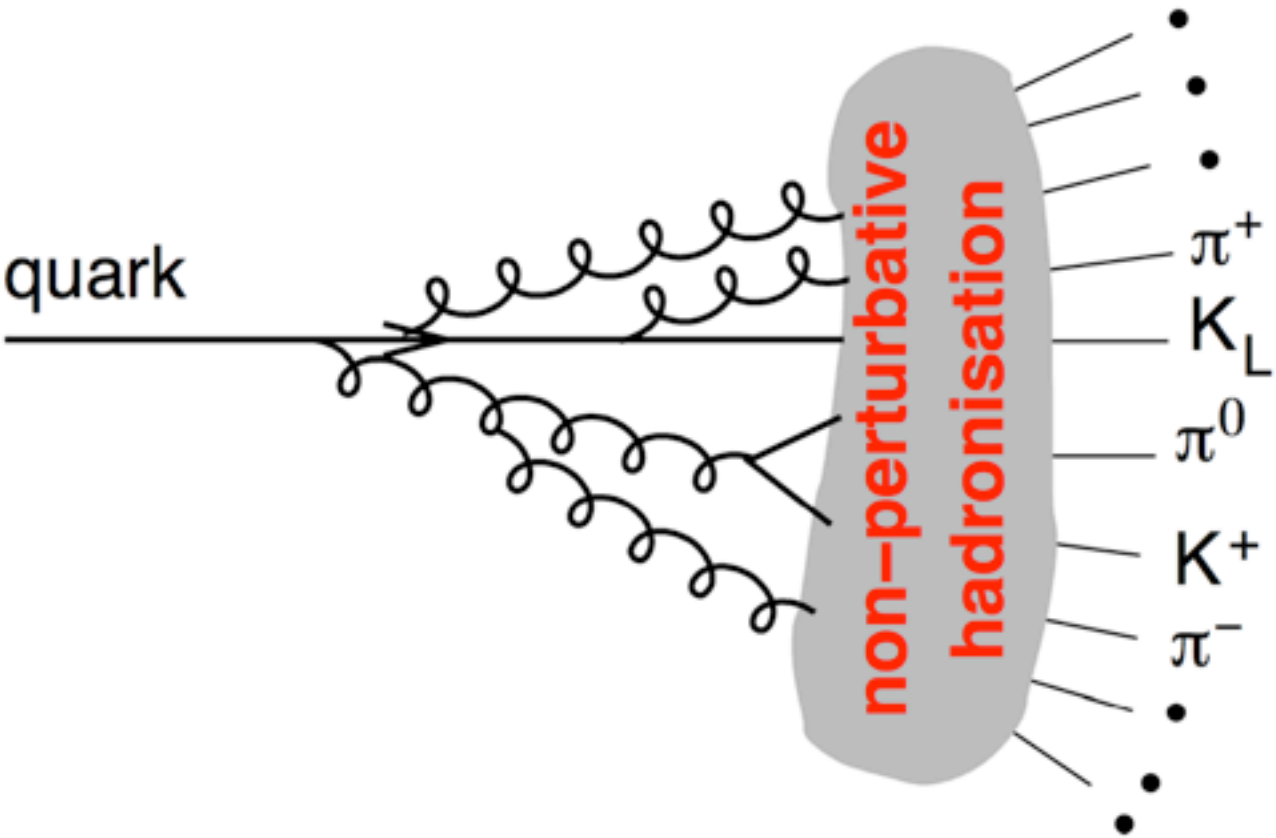


8(!) jets event from ATLAS

Jets @ LHC


CMS Experiment at the LHC, CERN
Data recorded: 2011-Sep-12 14:31:09.077784 GMT(16:31:09 CEST)
Run / Event: 176169 / 163929870

Why do jets happen?



Gluon emission

$$\int \alpha_s \frac{dE}{E} \frac{d\theta}{\theta} \gg 1$$

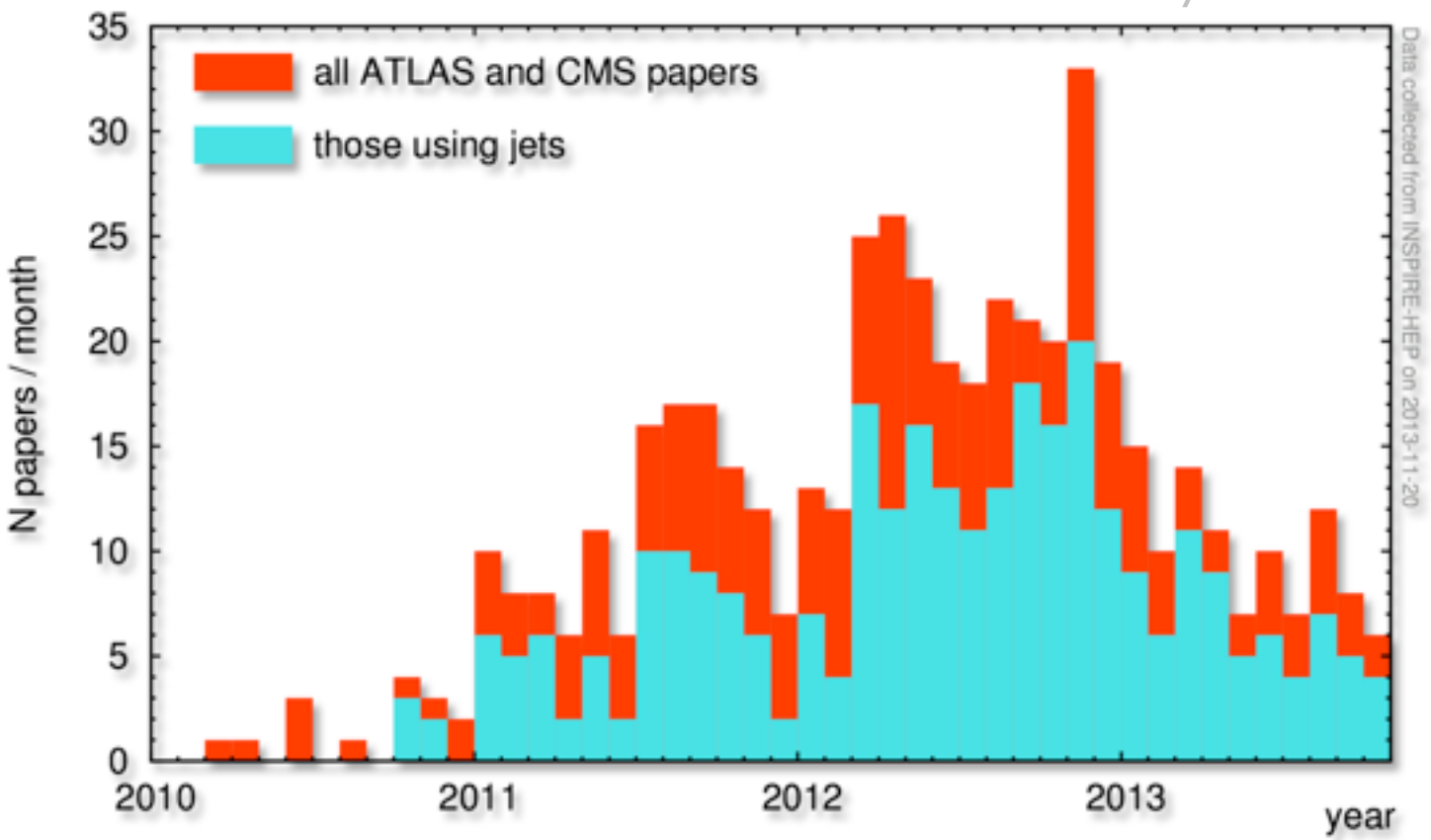
Non-perturbative physics

$$\alpha_s \sim 1$$

The pervasiveness of jets

- ATLAS and CMS have each published **400+** papers since 2010
  - More than **half** of these papers make use of **jets**
  - **60%** of the **searches** papers makes use of **jets**

Plot by G. Salam

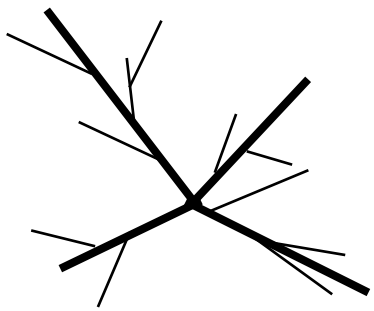


(Source: INSPIRE. Results may vary when employing different search keywords)

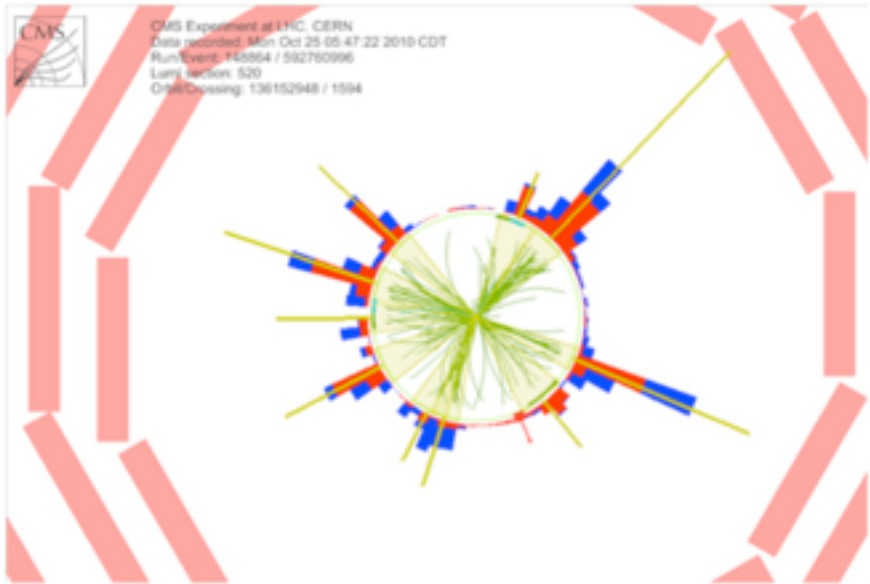
Why are jets so important?

Taming reality

Multileg + PS



??



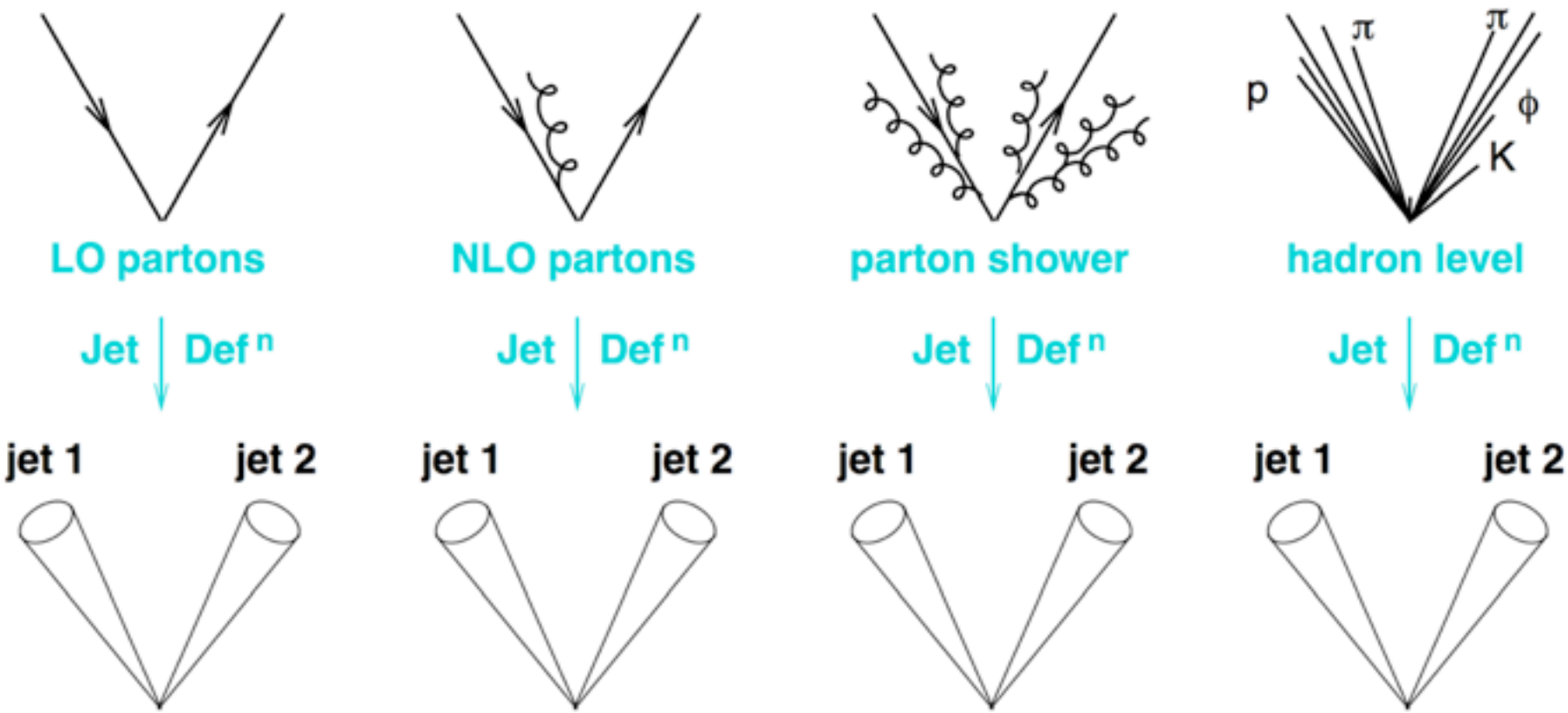
QCD predictions

Real data

Jets

One purpose of a 'jet clustering' algorithm is to
reduce the complexity of the final state, simplifying many hadrons
to **simpler objects** that one can hope to **calculate**

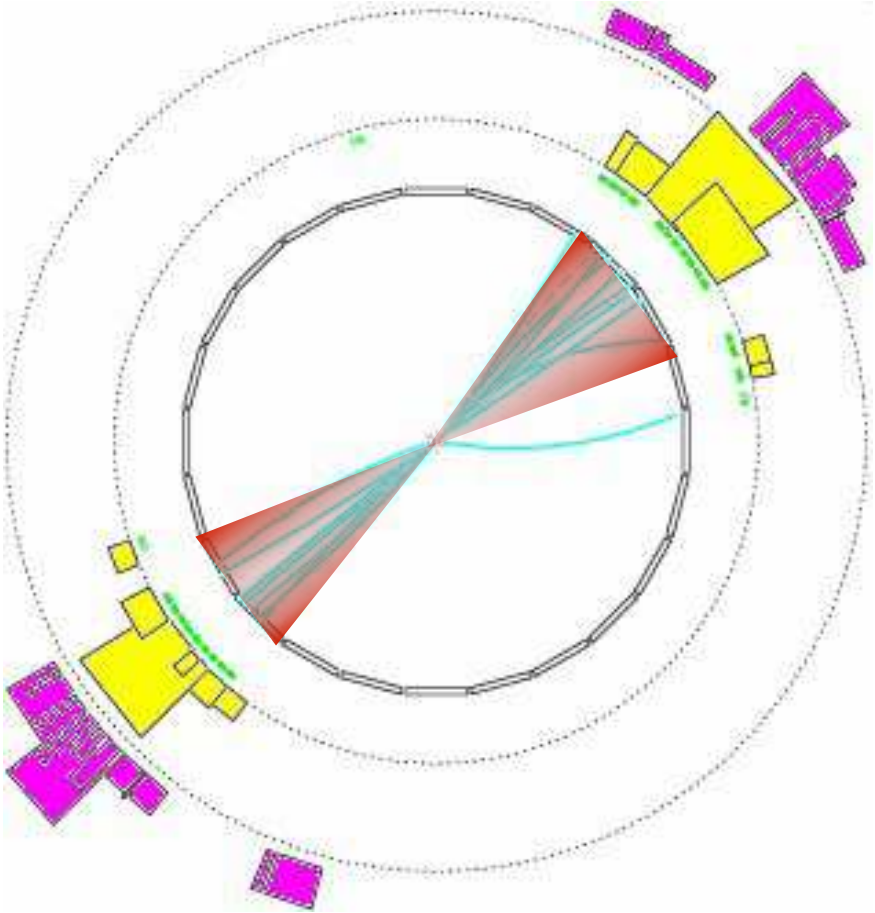
Jet definitions as projections



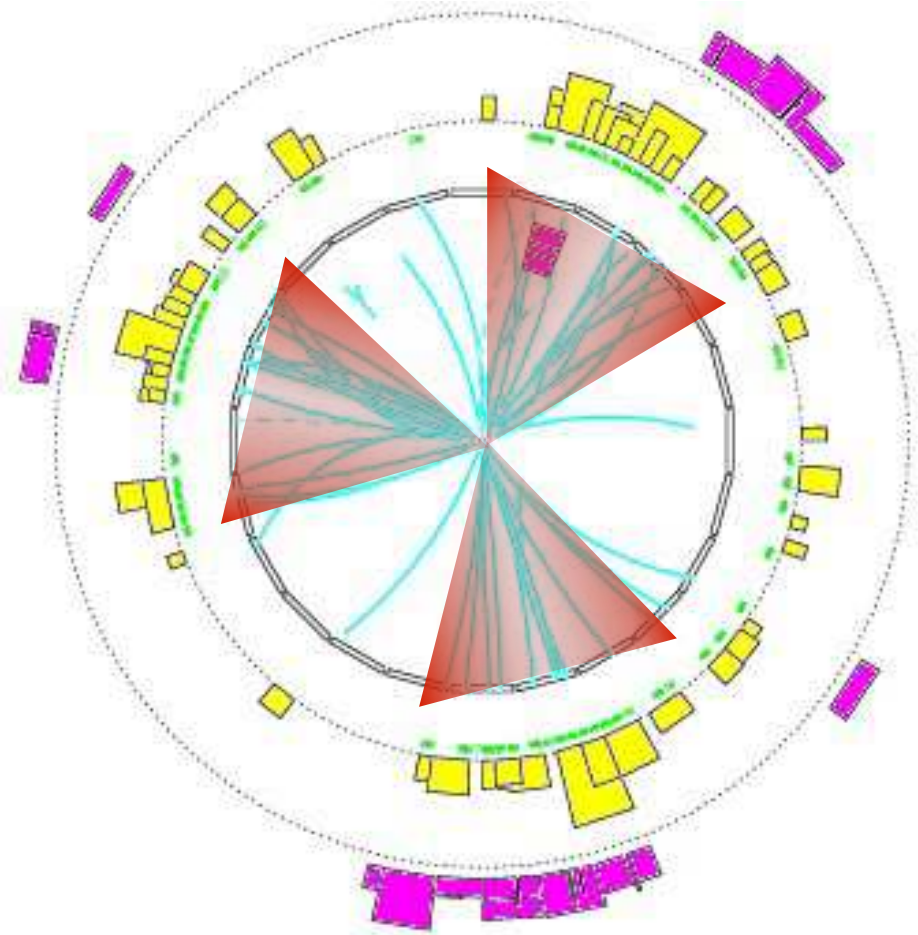
Projection to jets should be resilient to QCD effects

NB: projections are NOT unique:
a jet is NOT EQUIVALENT to a parton

Reconstructing jets is an ambiguous task

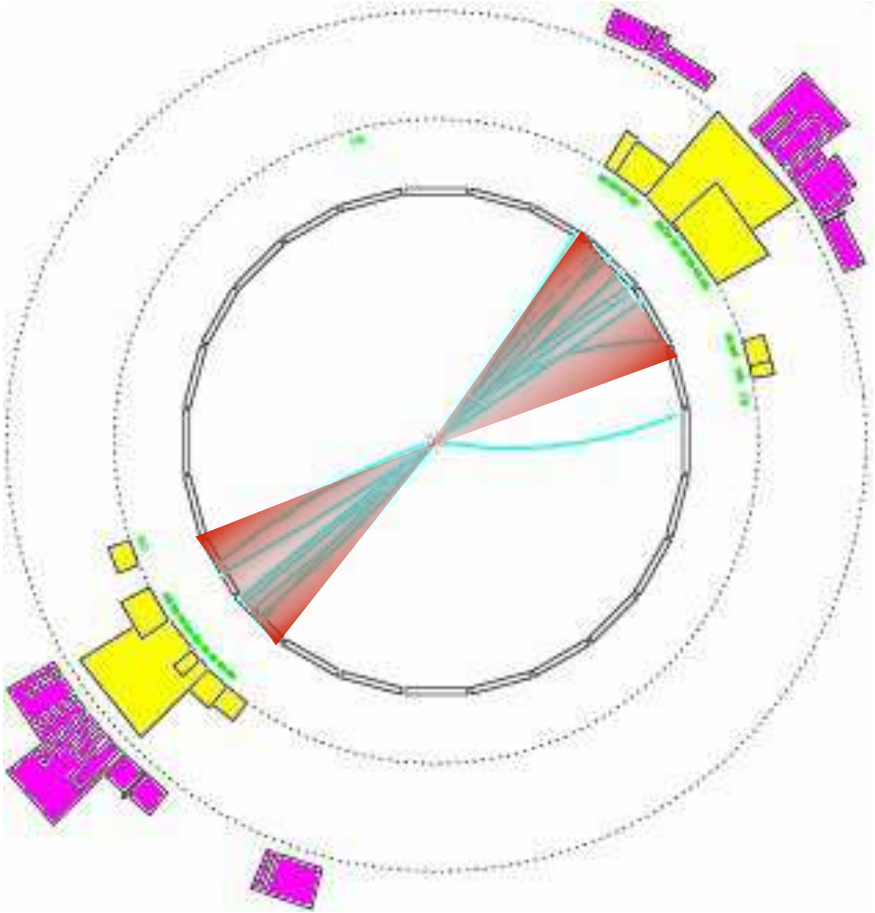


2 clear jets

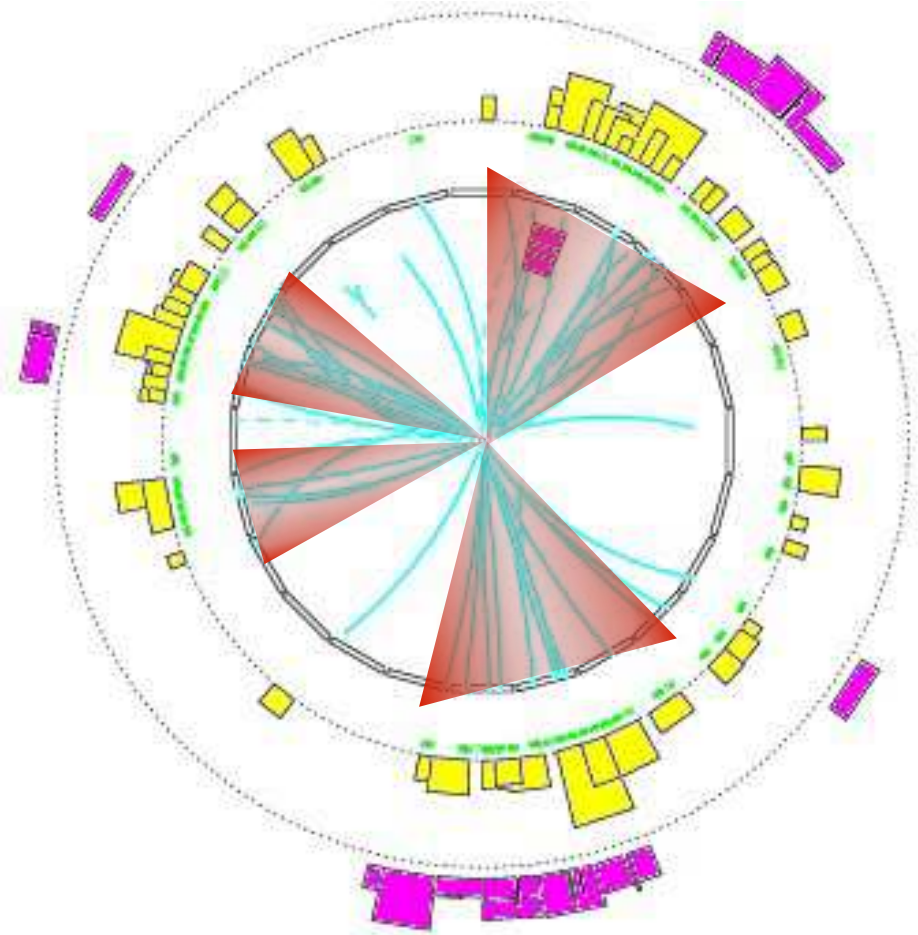


3 jets?

Reconstructing jets is an ambiguous task



2 clear jets

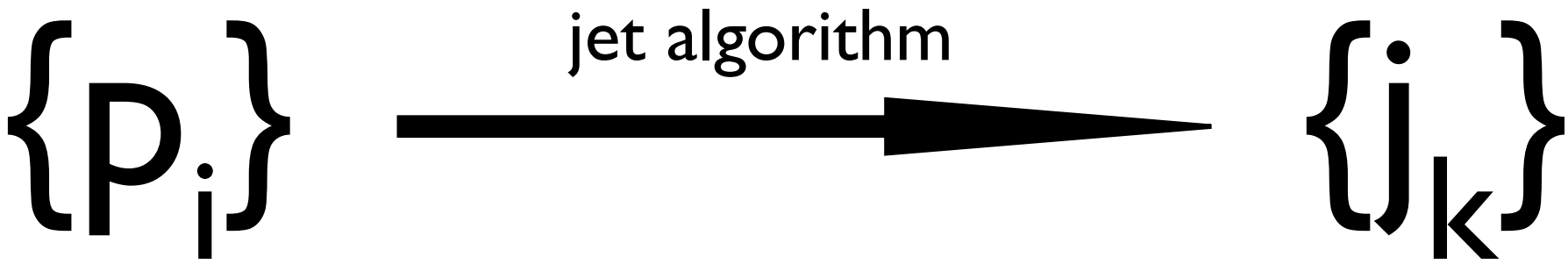


3 jets?

or 4 jets?

Jet clustering algorithm

A **jet algorithm** maps the momenta of the final state particles into the momenta of a certain number of jets:



particles,
4-momenta,
calorimeter towers, ...

jets

Most algorithms contain a resolution parameter, **R**, which controls the extension of the jet

"*Jet [definitions] are legal contracts between theorists and experimentalists*"
-- MJ Tannenbaum

Jets can serve two purposes

- They can be ***observables***, that one can measure and calculate
- They can be ***tools***, that one can employ to extract specific properties of the final state

Different clustering algorithms have different properties and characteristics that can make them more or less appropriate for each of these tasks

IRC safety

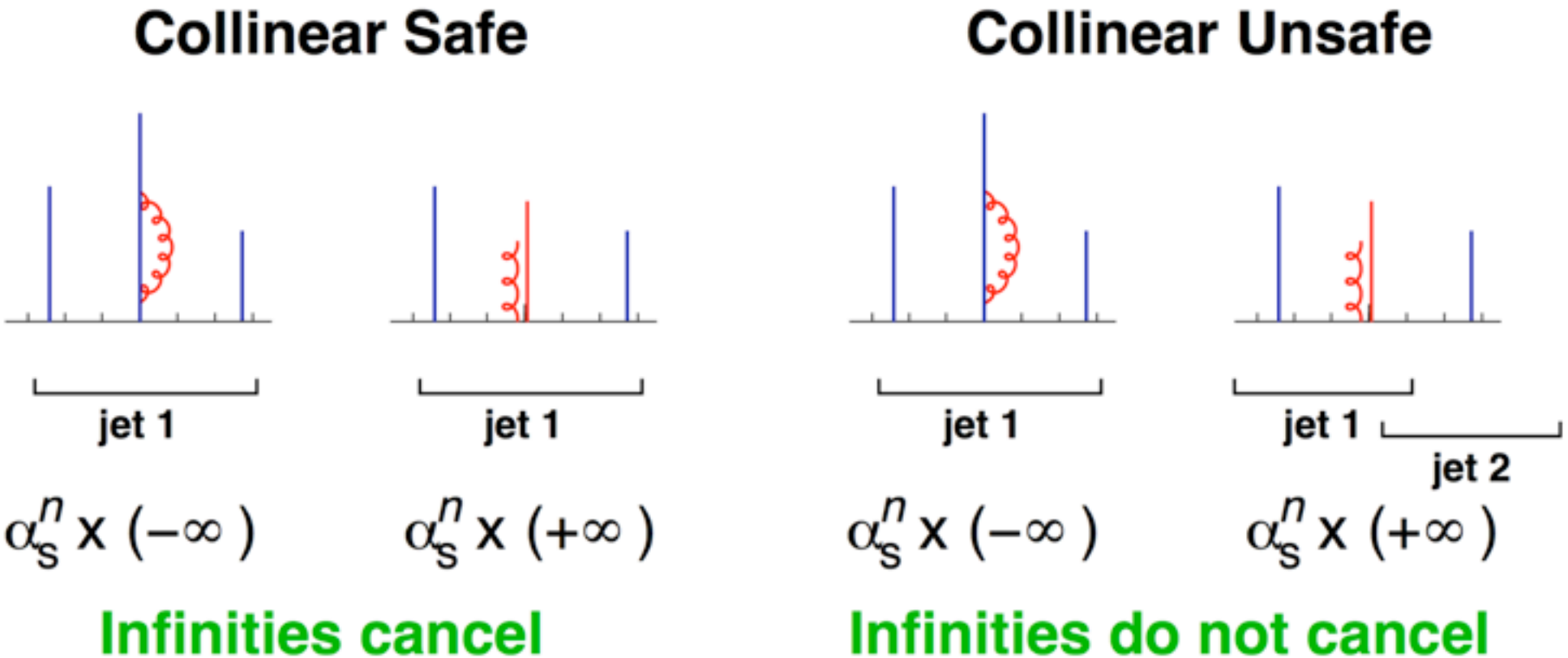
An observable is **infrared and collinear safe** if, in the limit of a **collinear splitting**, or the **emission of an infinitely soft** particle, the observable remains **unchanged:**

$$O(X; p_1, \ldots, p_n, p_{n+1} \to 0) \to O(X; p_1, \ldots, p_n)$$
$$O(X; p_1, \ldots, p_n \parallel p_{n+1}) \to O(X; p_1, \ldots, p_n + p_{n+1})$$

This property ensures cancellation of **real** and **virtual** divergences in higher order calculations

If we wish to be able to calculate a jet rate in perturbative QCD the jet algorithm that we use must be IRC safe:
soft emissions and collinear splittings must not change the hard jets

Reconstructing jets must respect rules



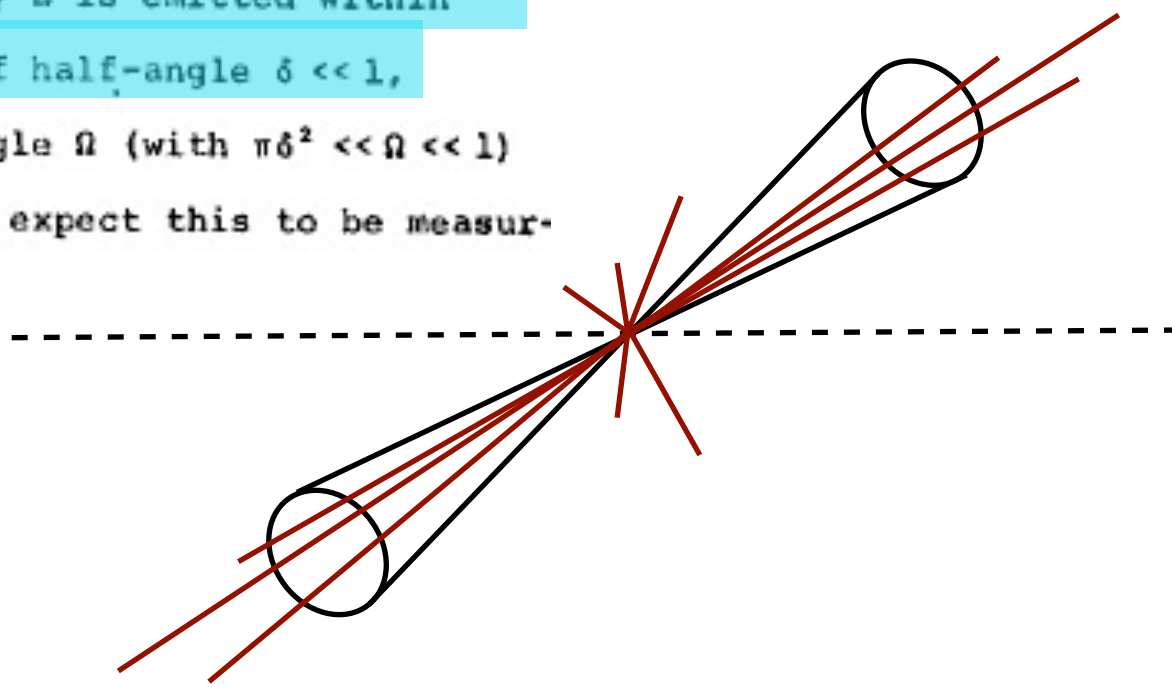
Perturbative calculations of jet observable will only be possible with collinear (and infrared) safe jet definitions

Cone algorithms

The first rigorous definition of cone jets in QCD is due to Sterman and Weinberg

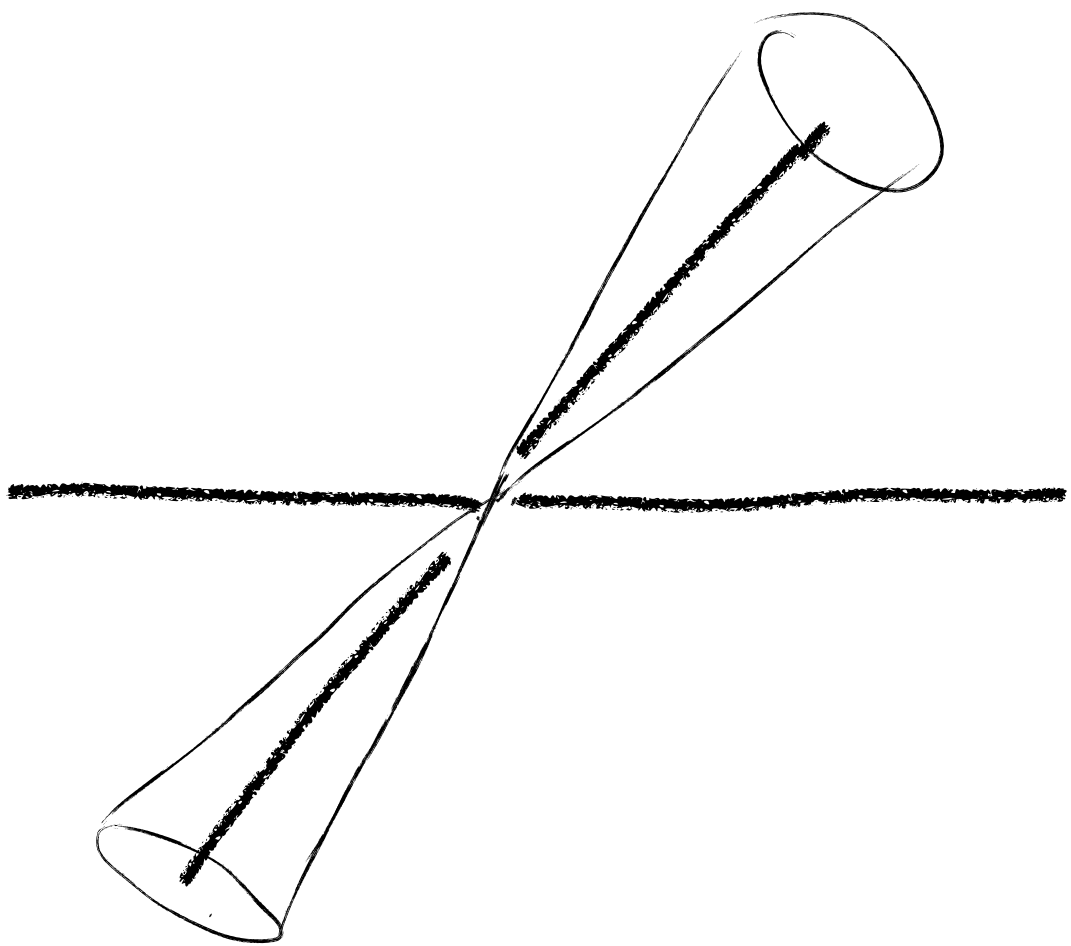
Phys. Rev. Lett. **39**, 1436 (1977)

To study jets, we consider the partial cross section $\sigma(E,\theta,\Omega,\epsilon,\delta)$ for e^+e^- hadron production events, in which all but a fraction $\epsilon \ll 1$ 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 Ω (with $\pi\delta^2 \ll \Omega \ll 1$) at an angle θ to the e^+e^- beam line. We expect this to be measur-

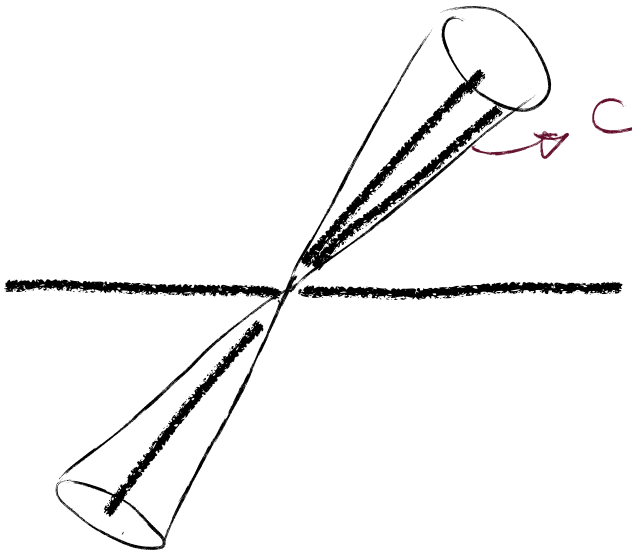


$$\sigma(E,\theta,\Omega,\epsilon,\delta) = (d\sigma/d\Omega)_0\,\Omega\left[1 - (g_E^2/3\pi^2)\left\{3\ln\delta + 4\ln\delta\,\ln 2\epsilon + \frac{\pi^3}{3} - \frac{5}{2}\right\}\right] \quad (6)$$

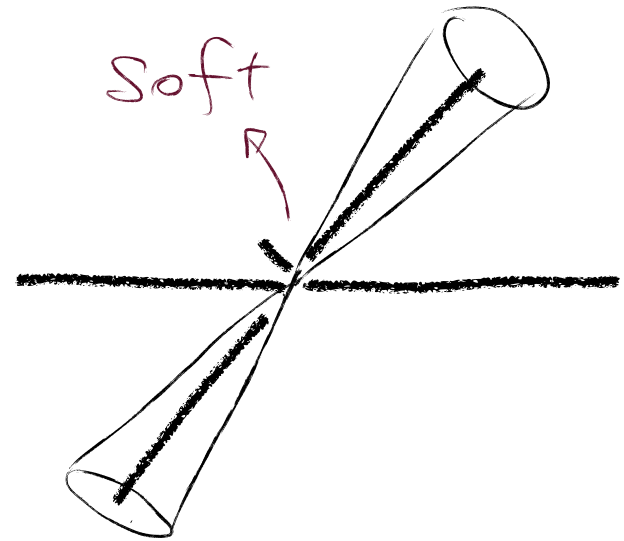
2 particles = 2 jets

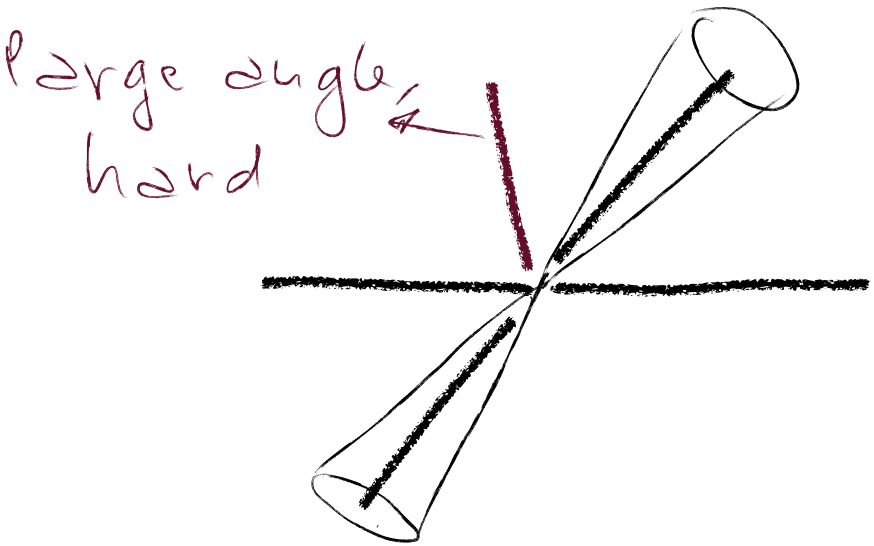


3 particles → ?



2 jets





3 jets

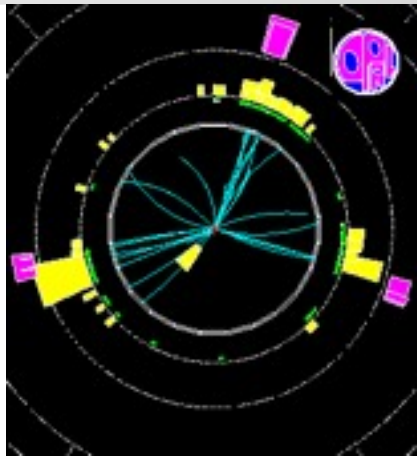
Jet algorithms

The Sterman-Weinberg definition is "inclusive enough" for IRC safety
Good for 2 jets and e^+e^- collisions

What happens in a more general case, where more than two jets are likely to exist?
Where do we place the cones? **How many**?

Iterative jet algorithms

Two main approaches to jet clustering



1. Find regions where a lot of energy flows

2. Decide which particles are "close", aggregate them

In HEP these are usually called **cone** and **sequential recombination** algorithms respectively

(in other fields they are often called partitional-type clustering and agglomerative hierarchical clustering)

Two main classes of jet algorithms

Sequential recombination algorithms

Bottom-up approach: combine particles starting from **closest ones**

How? Choose a **distance measure**, iterate recombination until few objects left, call them jets

Works because of mapping closeness ⇔ QCD divergence

Examples: Jade, k_t, Cambridge/Aachen, anti-k_t, …..

Usually trivially made IRC safe, but their algorithmic complexity scales like N^3

Cone algorithms

Top-down approach: find coarse regions of energy flow.

How? Find **stable cones** (i.e. their axis coincides with sum of momenta of particles in it)

Works because QCD only modifies energy flow on small scales

Examples: JetClu, MidPoint, ATLAS cone, CMS cone, SISCone…...

Can be programmed to be fairly fast, at the price of being complex and IRC unsafe

FERMILAB-Conf-90/249-E
[E-741/CDF]

Toward a Standardization of Jet Definitions *

* To be published in the proceedings of the 1990 Summer Study on High Energy Physics, *Research Directions for the Decade*, Snowmass, Colorado, June 25 - July 13, 1990.

Several important properties that should be met by a jet definition are [3]:

1. Simple to implement in an experimental analysis;
2. Simple to implement in the theoretical calculation;
3. Defined at any order of perturbation theory;
4. Yields finite cross section at any order of perturbation theory;
5. Yields a cross section that is relatively insensitive to hadronization.

Speed

Infrared and collinear safety

[Addition of a soft particle or a collinear splitting should not change final hard jets]

A little history

- Cone-type jets were introduced first in QCD in the 1970s (Sterman-Weinberg '77)

- In the 1980s cone-type jets were adapted for use in hadron colliders (SppS, Tevatron...) → iterative cone algorithms

- LEP was a golden era for jets: new algorithms and many relevant calculations during the 1990s
  - Introduction of the 'theory-friendly' k_t algorithm
    - sequential recombination type algorithm, IRC safe
    - it allows for all order resummation of jet rates
  - Several accurate calculations in perturbative QCD of jet properties: rates, jet mass, thrust, ...

Finding stable cones

In partitional-type algorithms (i.e. cones), one wishes to find the **stable configurations**:

axis of cones coincides with sum of 4-momenta of the particles it contains

The 'safe' way of doing so is to test
all possible combinations of N objects

Unfortunately, this takes $N2^N$ operations:
the time taken is the age of the universe for only 100 objects

An approximate way out is to use **seeds** (e.g. à la k-means)
However, the final result can depend on the choice of the seeds and, such jet algorithms usually turn out to be **IRC unsafe**

Finding cones

Different procedures for placing the cones lead to **different cone algorithms**

NB: their properties and behaviour can **vastly differ**:
there isn't **'a'** cone algorithm, but rather many of them

The main sub-categories of cone algorithms are:

- **Fixed** cone with **progressive removal** (FC-PR) (PyJet, CellJet, GetJet)
- **Iterative** cone with **progressive removal** (IC-PR) (CMS iterative cone)
- **Iterative** cone with **split-merge** (IC-SM) (JetClu, ATLAS cone)
- **IC-SM** with **mid-points** (IC_{mp}-SM) (CDF MidPoint, D0 Run II)
- **IC_{mp}** with **split-drop** (IC_{mp}-SD) (PxCone)
- **Seedless** cone with **split-merge** (SC-SM) (SISCone)

All, except SISCone, are infrared or collinear unsafe

Recombination algorithms

- First introduced in e^+e^- collisions in the '80s
- Typically they work by calculating a **'distance'** between particles, and then recombine them pairwise according to a given order, until some condition is met (e.g. no particles are left, or the distance crosses a given threshold)

IRC safety can usually be seen to be trivially guaranteed

JADE algorithm

Distance:

$$y_{ij} = \frac{2E_iE_j(1-\cos\theta_{ij})}{Q^2}$$

- Find the minimum y_{min} of all y_{ij}
- If y_{min} is below some jet resolution threshold y_{cut}, recombine i and j into a single new particle ('pseudojet'), and repeat
- If no $y_{min} < y_{cut}$ are left, all remaining particles are jets

Problem of this particular algorithm:
two **soft** particles emitted at **large angle** get easily recombined into a single jet: counterintuitive and perturbatively troublesome

e^+e^- k_t (Durham) algorithm

[Catani, Dokshitzer, Olsson, Turnock, Webber '91]

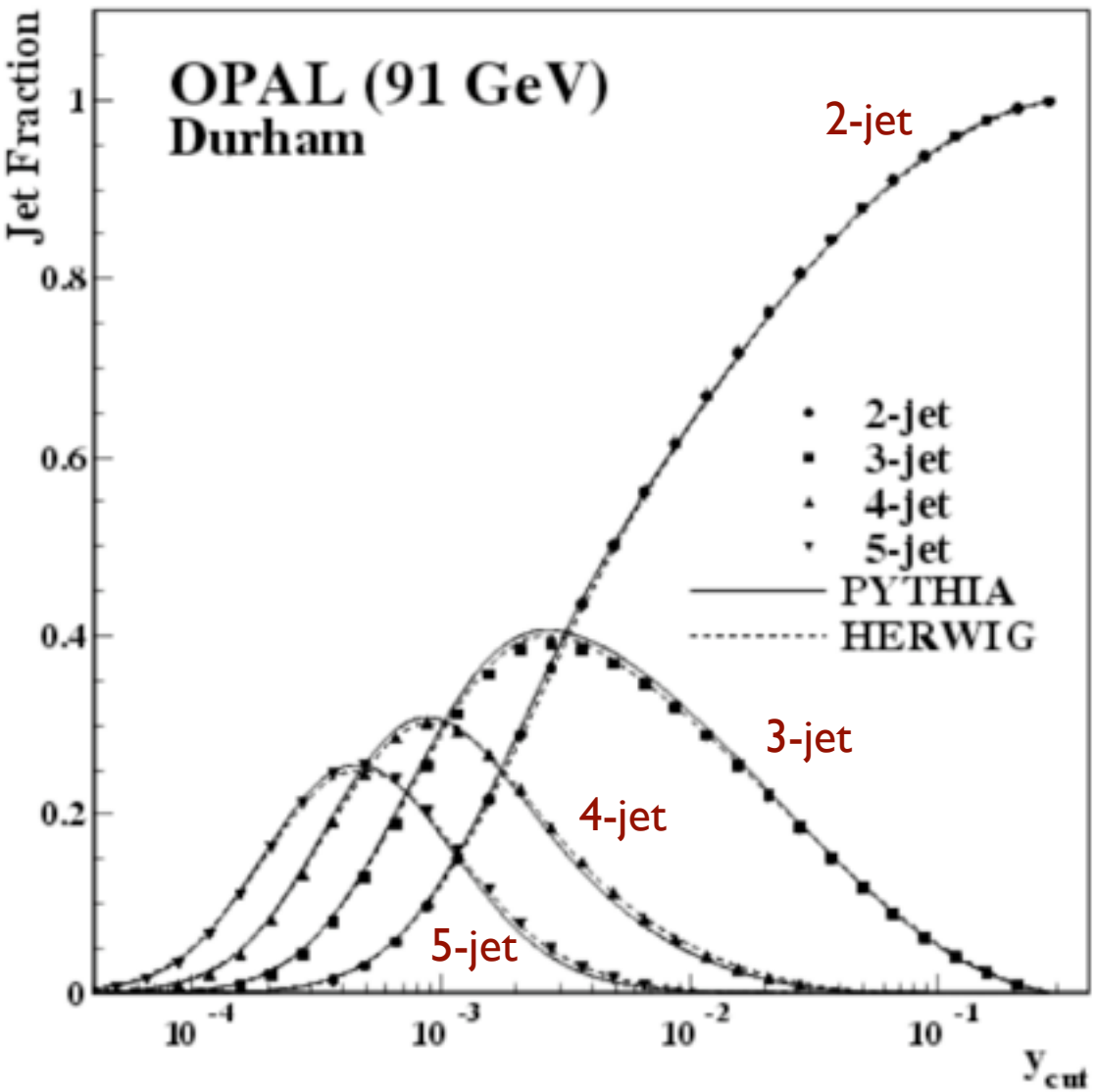
Distance:

$$y_{ij} = \frac{2\min(E_i^2, E_j^2)(1-\cos\theta_{ij})}{Q^2}$$

In the collinear limit, the numerator reduces to the **relative transverse momentum** (squared) of the two particles, hence the name of the algorithm

- Find the minimum y_{min} of all y_{ij}
- If y_{min} is below some jet resolution threshold y_{cut}, recombine i and j into a single new particle ('pseudojet'), and repeat
- If no $y_{min} < y_{cut}$ are left, all remaining particles are jets

e^+e^- k_t (Durham) algorithm in action



Characterise events
in terms of number of jets
(as a function of y_{cut})

Resummed calculations for distributions of y_{cut} doable with the k_t algorithm

e^+e^- k_t (Durham) algorithm v. QCD

k_t is a sequential recombination type algorithm

One key feature of the k_t algorithm is its relation to the structure of QCD divergences:

$$\frac{dP_{k\to ij}}{dE_i d\theta_{ij}} \sim \frac{\alpha_s}{\min(E_i, E_j)\theta_{ij}}$$

The y_{ij} distance is the inverse of the emission probability

- The k_t algorithm roughly inverts the QCD branching sequence (the pair which is recombined first is the one with the largest probability to have branched)
- The history of successive clusterings has physical meaning

The common wisdom circa 2005

- Cone algorithms are IRC unsafe
  - → because, to make them reasonably fast, they were usually implemented via approximate methods using seeds
- Sequential recombination algorithms (i.e. k_t) are slow and too susceptible to background contamination
  - → because they scale like N^3
  - → because they tend to collect soft particles up to large distances from centre
  - → because they were often run with R=1 and compared to cones with R=0.5!

The solution to the speed problem came from considering the clustering problem from a geometrical rather from a combinatorial point of view

- Sequential recombination algorithms could be implemented with $O(N^2)$ or even $O(N\ln N)$ complexity rather than $O(N^3)$
  [MC, Salam, 2006]

- Cone algorithms could be implemented exactly (and therefore made IRC safe) with $O(N^2\ln N)$ rather than $O(N2^N)$ complexity
  [Salam, Soyez 2007]

k_t algorithm in hadron collisions

(Inclusive and longitudinally invariant version)

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

- Calculate the distances between the particles: $\mathbf{d_{ij}}$
- Calculate the beam distances: $\mathbf{d_{iB}}$
- Combine particles with **smallest distance** d_{ij} or, if d_{iB} is smallest, call it a jet
- Find again smallest distance and repeat procedure until no particles are left (this stopping criterion leads to the *inclusive* version of the k_t algorithm)
- Only use jets with $p_t > p_{t,min}$

The speed 'problem'

Given N particles the k_t algorithm is, naively, an **$O(N^3)$ algorithm**: calculate N^2 distances, repeat for all N iterations

With 1000 particles, this takes 10^9 operations, i.e. **about a second**

Clustering such an event would take significantly more than generating it in a MonteCarlo, not to speak about trying to use it at the trigger level, where the time budges is of the order of **tens of milliseconds**

This, together with the tendency of the k_t algorithm to 'scoop up' soft radiation quite far from the hard partons, and to give jets with ragged borders, difficult to correct for, had led people to prefer cones in a hadronic environment

The FastJet lemma

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

Key lemma:

[MC, G.P. Salam, hep-ph/0512210]

If i and j form the **smallest** $d_{ij} = \min(p_{ti}^2, p_{tj}^2)\, \Delta R_{ij}^2$ **and** $p_{ti} < p_{tj}$

$$\Rightarrow \qquad \Delta R_{ij} < \Delta R_{ik} \qquad \forall k \neq j$$

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

Proof:

Suppose the lemma wrong: $\exists$ k such that $\Delta R_{ij} \geq \Delta R_{ik}$

then $d_{ik} = \min(p_{ti}^2, p_{tk}^2)\, \Delta R_{ik}^2 \leq p_{ti}^2\, \Delta R_{ik}^2 \leq p_{ti}^2\, \Delta R_{ij}^2 = d_{ij}$

which contradicts the initial statement that d_{ij} is the smallest one

The FastJet lemma

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 y and ϕ

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

The FastJet algorithm

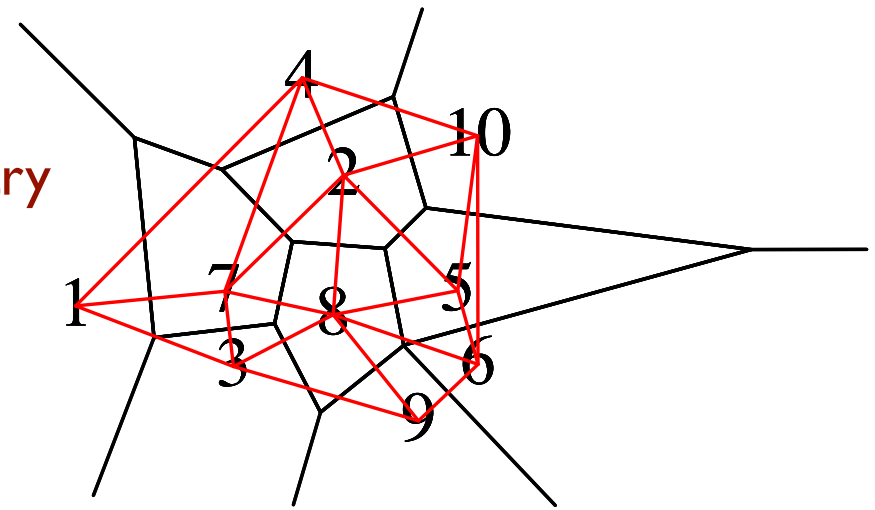
- For each particle i establish its **geometrical nearest neighbour G_i** and calculate the arrays d_{iG_i} and d_{iB}
- Find the minimal value d_{min} of the d_{iG_i} and d_{iB}, combine particles corresponding to it
- Update d_{iG_i} and d_{iB} if needed, continue until no particles left

This is already an **$O(N^2)$ algorithm**:
i.e. find O(N) neighbours for N particles

But we can do better

The FastJet algorithm

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)

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 lnN)**

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 lnN)**

Merge/eliminate particles appropriately

Update Voronoi diagram and distances' map — **O(lnN)**

repeat N times

Overall, an O(N ln N) algorithm

The k_t algorithm and its siblings

$$d_{ij} = \min(p_{ti}^{2p}, p_{tj}^{2p}) \frac{\Delta y^2 + \Delta\phi^2}{R^2} \qquad d_{iB} = p_{ti}^{2p}$$

p = 1 k_t algorithm

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

p = 0 Cambridge/Aachen algorithm

Y. Dokshitzer, G. Leder, S.Moretti and B. Webber, JHEP 08 (1997) 001
M. Wobisch and T. Wengler, hep-ph/9907280

p = -1 **anti-k_t algorithm**

MC, G. Salam and G. Soyez, arXiv:0802.1189

NB: in anti-kt pairs with a **hard** particle will cluster first: if no other hard particles are close by, the algorithm will give **perfect cones**

Quite ironically, a sequential recombination algorithm is the 'perfect' cone algorithm

IRC safety of generalised-k_t algorithms

$$d_{ij} = \min(p_{ti}^{2p}, p_{tj}^{2p}) \frac{\Delta y^2 + \Delta\phi^2}{R^2} \qquad d_{iB} = p_{ti}^{2p}$$

p > 0

New **soft** particle ($p_t \to 0$) means that d → 0 ⇒ clustered first, no effect on jets

New **collinear** particle ($\Delta y^2+\Delta\Phi^2 \to 0$) means that d → 0 ⇒ clustered first, no effect on jets

p = 0

New **soft** particle ($p_t \to 0$) can be new jet of zero momentum ⇒ no effect on hard jets

New **collinear** particle ($\Delta y^2+\Delta\Phi^2 \to 0$) means that d → 0 ⇒ clustered first, no effect on jets

p < 0

New **soft** particle ($p_t \to 0$) means d → ∞ ⇒ clustered last or new zero-jet, no effect on hard jets

New **collinear** particle ($\Delta y^2+\Delta\Phi^2 \to 0$) means that d → 0 ⇒ clustered first, no effect on jets

IRC safe algorithms

k_t	SR $d_{ij} = \min(p_{ti}^2, p_{tj}^2)\Delta R_{ij}^2/R^2$ hierarchical in rel p_t	Catani et al '91 Ellis, Soper '93	NlnN
Cambridge/ Aachen	SR $d_{ij} = \Delta R_{ij}^2/R^2$ hierarchical in angle	Dokshitzer et al '97 Wengler, Wobish '98	NlnN
anti-k_t	SR $d_{ij} = \min(p_{ti}^{-2}, p_{tj}^{-2})\Delta R_{ij}^2/R^2$ gives perfectly conical hard jets	MC, Salam, Soyez '08 (Delsart, Loch)	$N^{3/2}$
SISCone	Seedless iterative cone with split-merge gives 'economical' jets	Salam, Soyez '07	N^2lnN

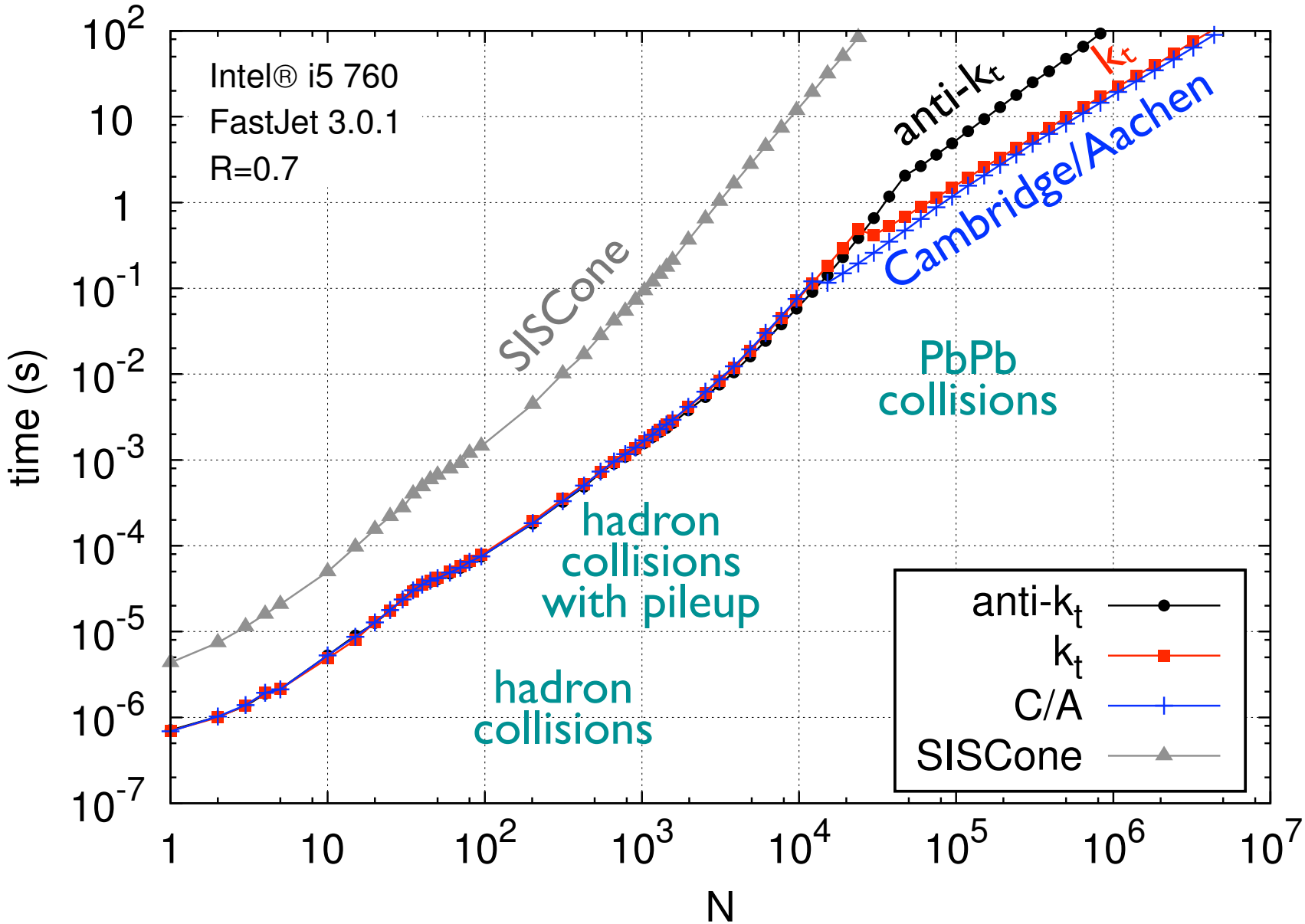
'second-generation' algorithms

All are available in FastJet, http://fastjet.fr

(As well as many IRC unsafe ones)

FastJet speed

Time needed to cluster an event with N particles



Anti-k_t in action

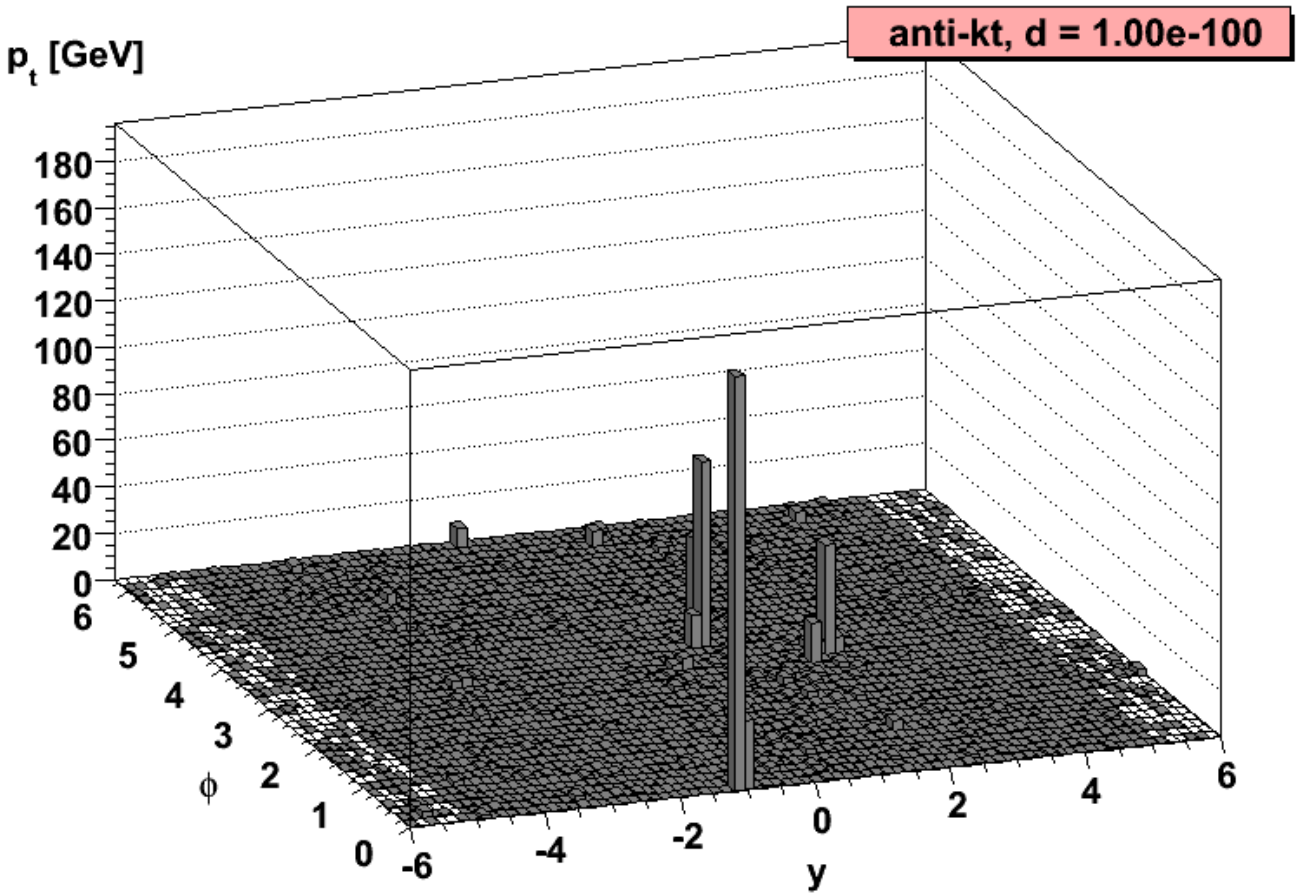
Clustering grows around hard cores

$$d_{ij} = \frac{1}{\max(p_{ti}^2, p_{tj}^2)} \frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = \frac{1}{p_{ti}^2}$$

Anti-k_t in action

Clustering grows around hard cores

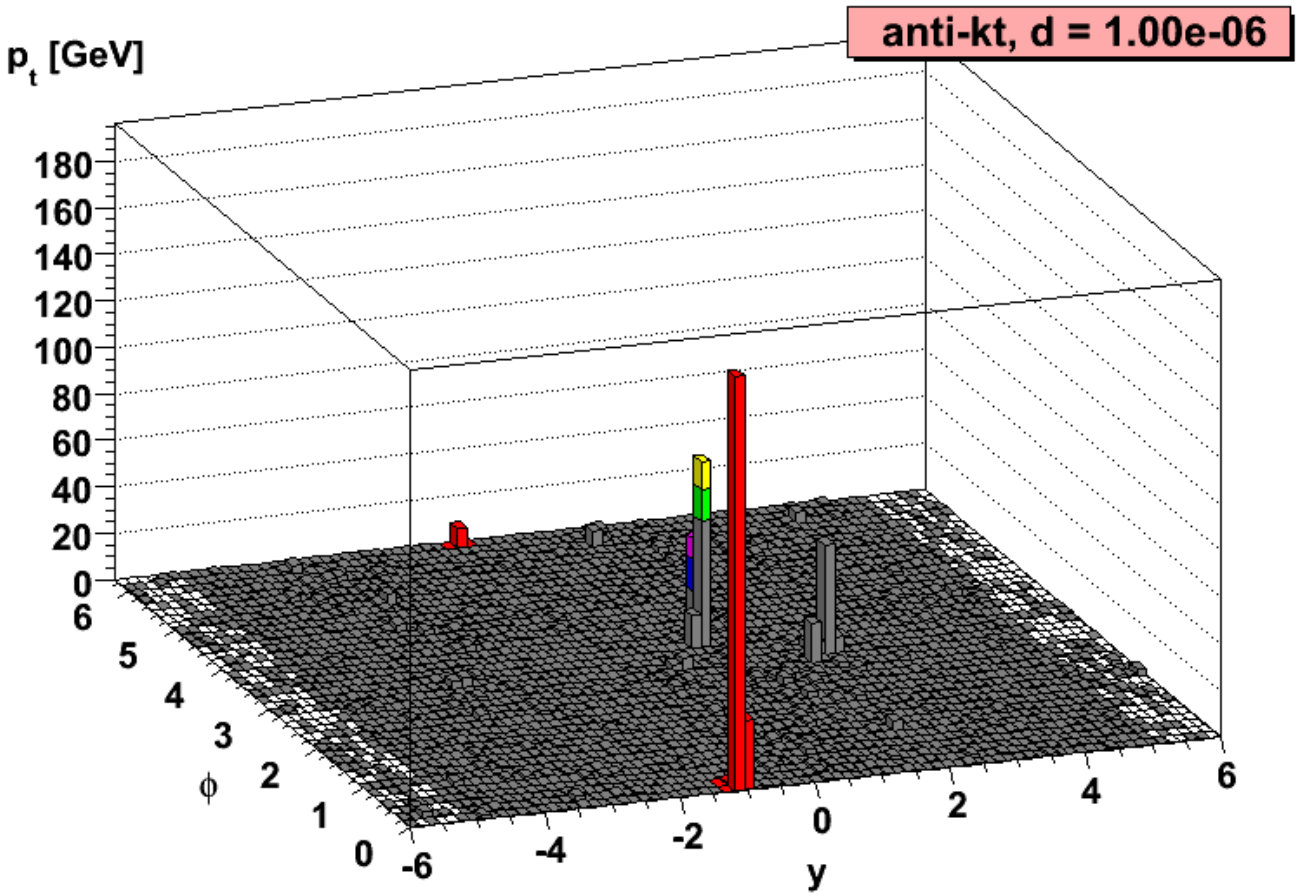
$$d_{ij} = \frac{1}{\max(p_{ti}^2, p_{tj}^2)} \frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = \frac{1}{p_{ti}^2}$$



Anti-k_t in action

Clustering grows around hard cores

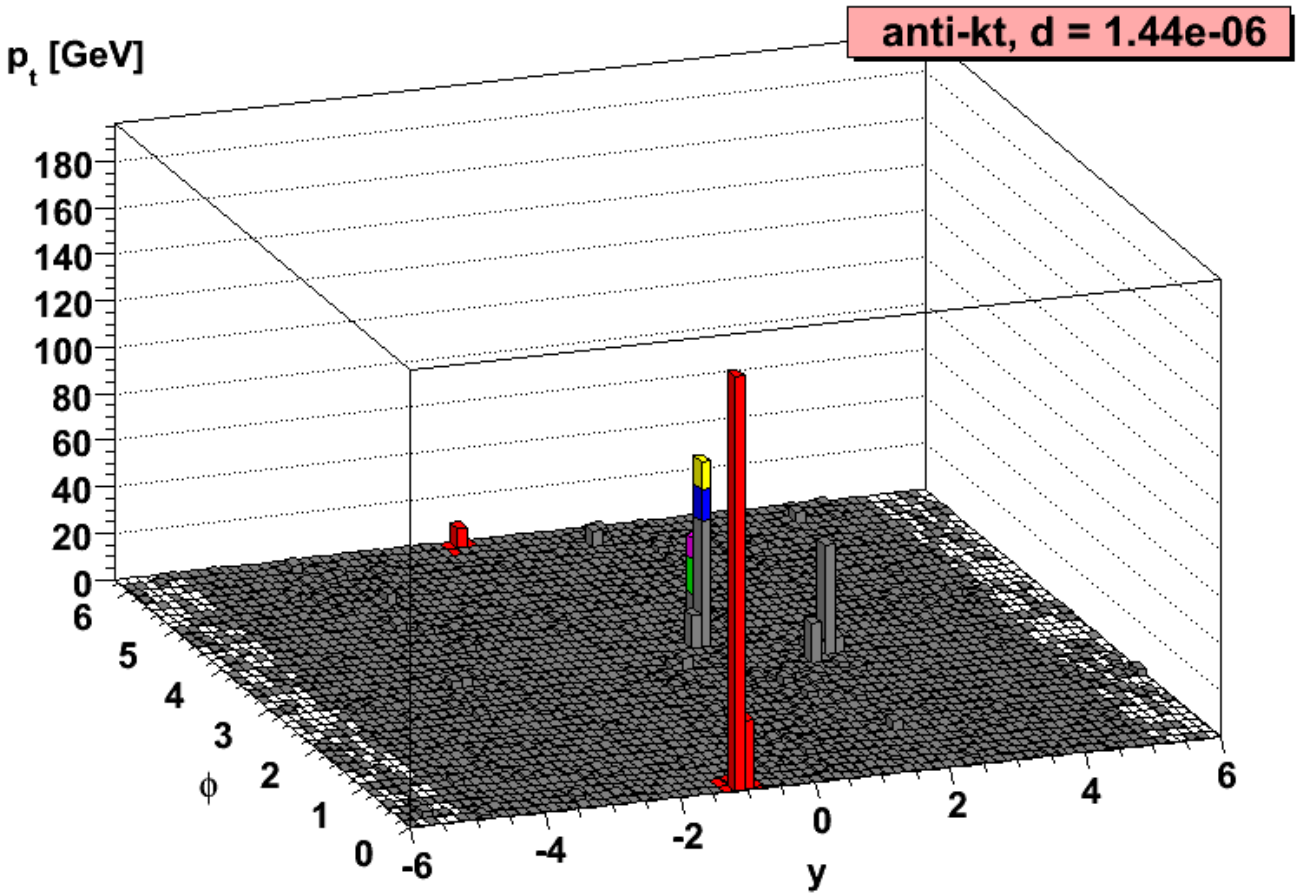
$$d_{ij} = \frac{1}{\max(p_{ti}^2, p_{tj}^2)} \frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = \frac{1}{p_{ti}^2}$$



Anti-k_t in action

Clustering grows around hard cores

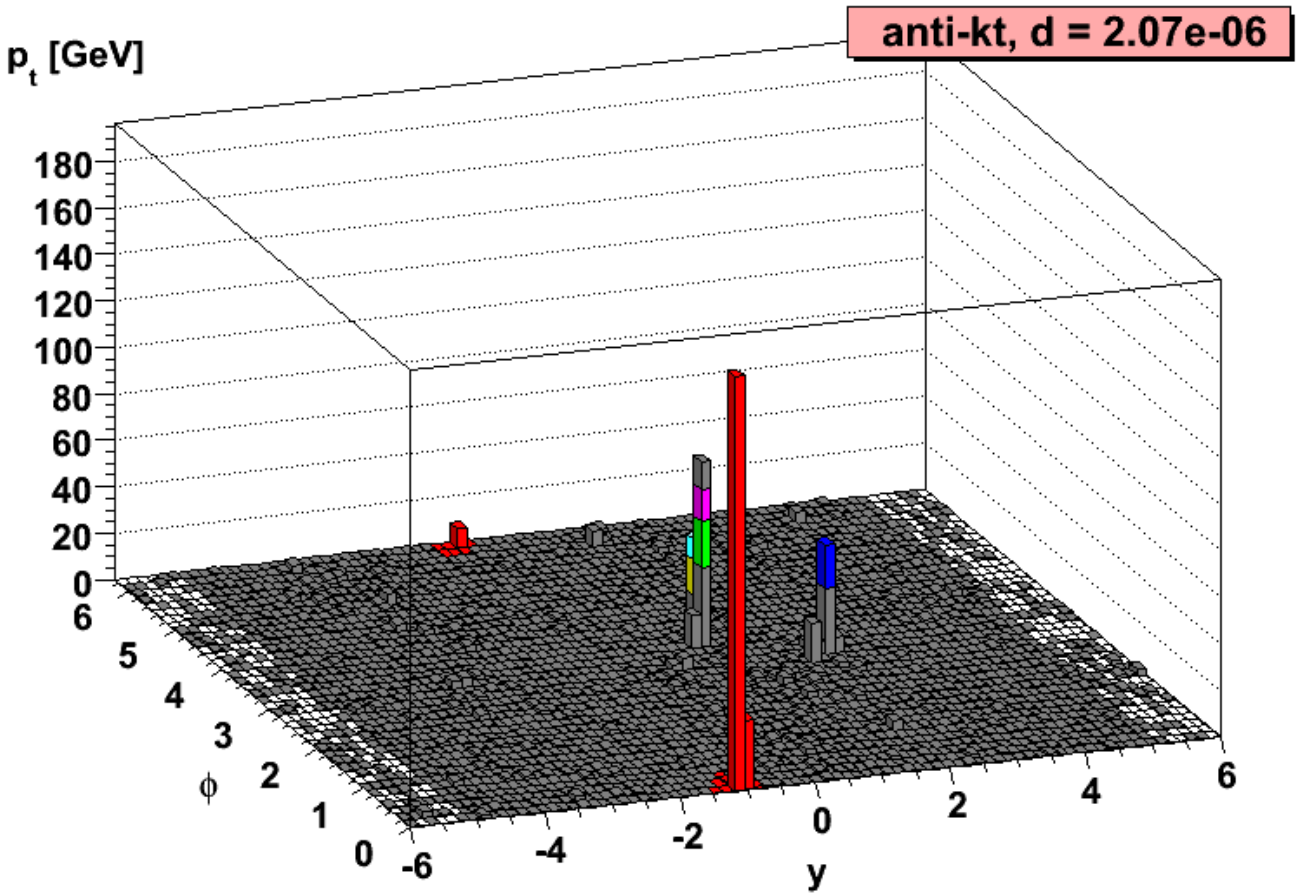
$$d_{ij} = \frac{1}{\max(p_{ti}^2, p_{tj}^2)} \frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = \frac{1}{p_{ti}^2}$$



Anti-k_t in action

Clustering grows around hard cores

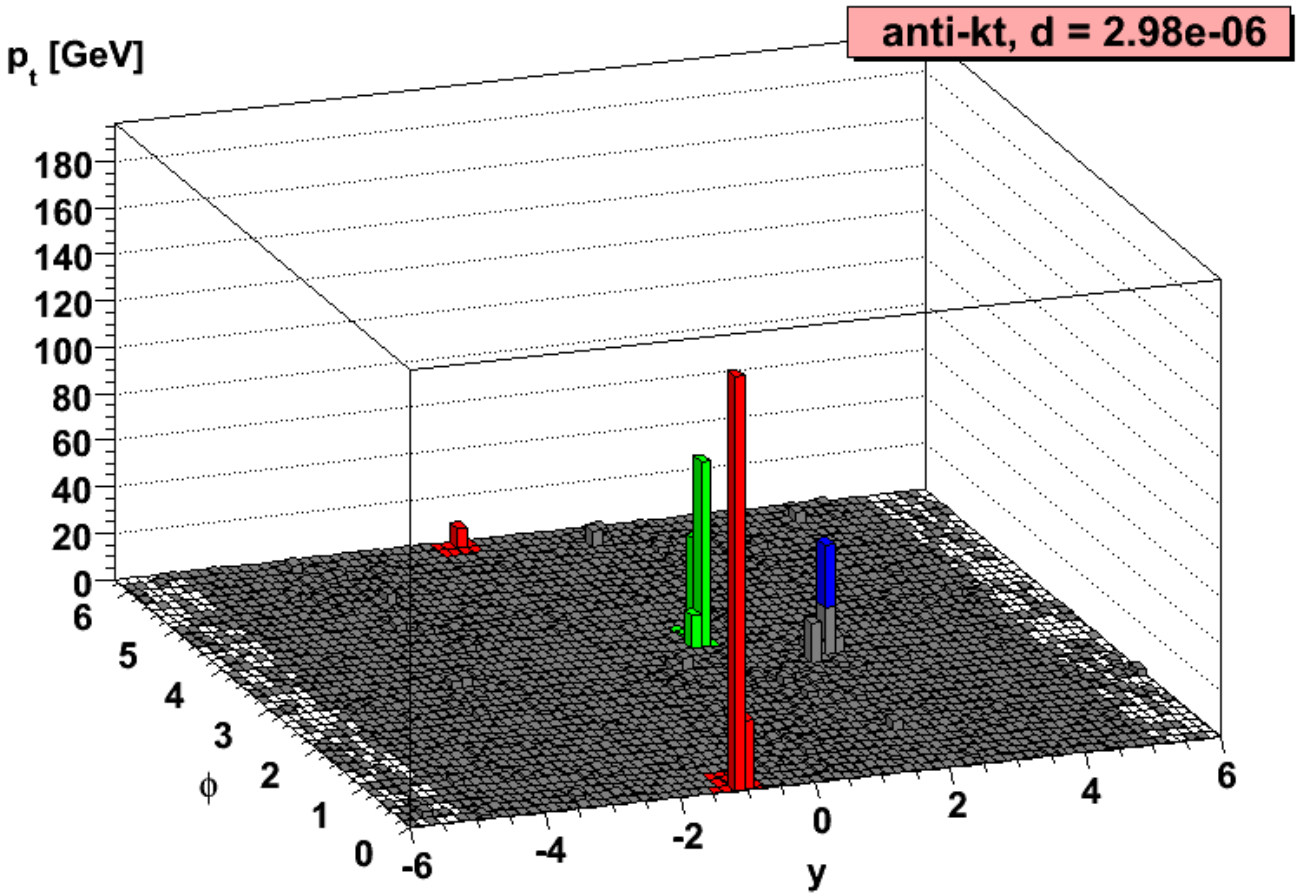
$$d_{ij} = \frac{1}{\max(p_{ti}^2, p_{tj}^2)} \frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = \frac{1}{p_{ti}^2}$$



Anti-k_t in action

Clustering grows around hard cores

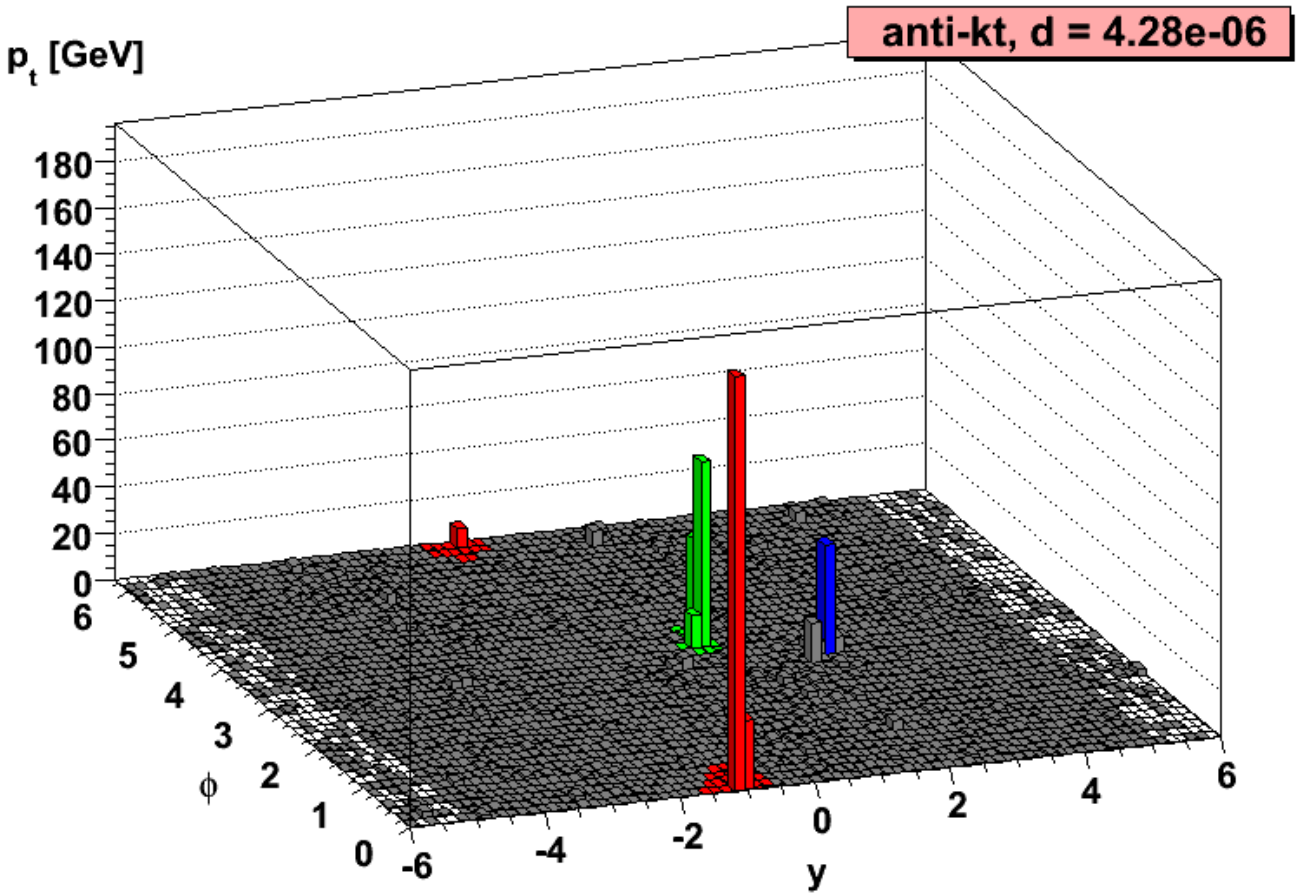
$$d_{ij} = \frac{1}{\max(p_{ti}^2, p_{tj}^2)} \frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = \frac{1}{p_{ti}^2}$$



Anti-k_t in action

Clustering grows around hard cores

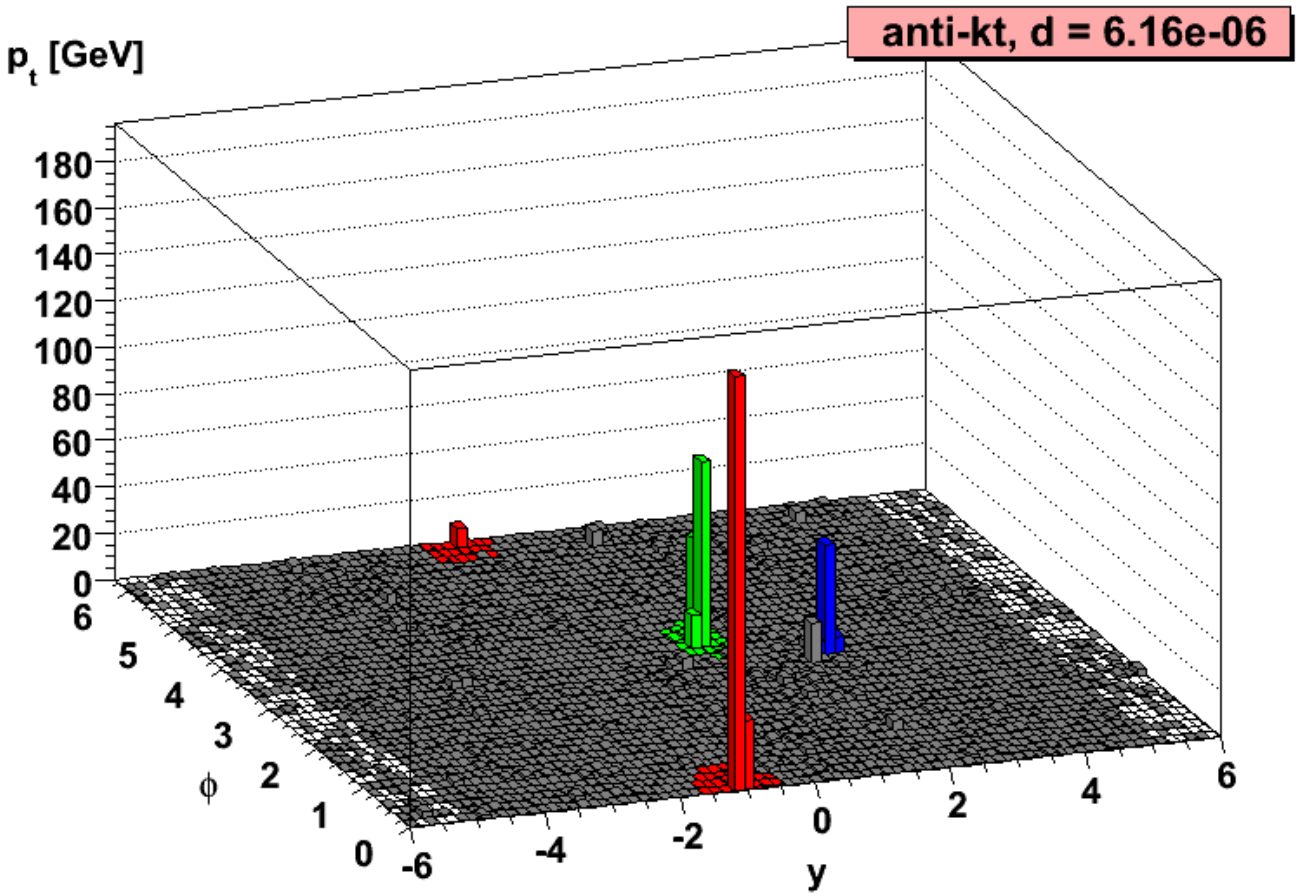
$$d_{ij} = \frac{1}{\max(p_{ti}^2, p_{tj}^2)} \frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = \frac{1}{p_{ti}^2}$$



Anti-k_t in action

Clustering grows around hard cores

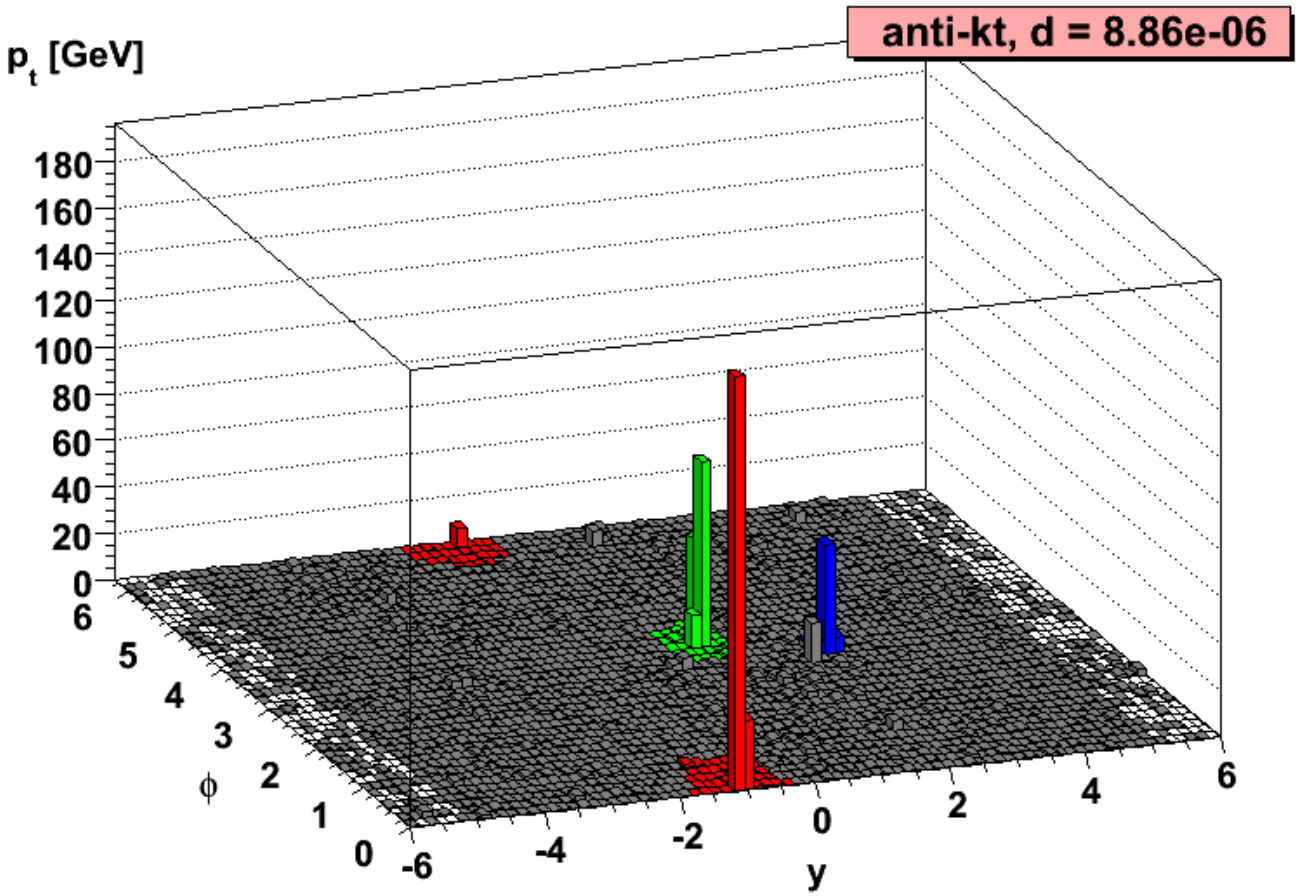
$$d_{ij} = \frac{1}{\max(p_{ti}^2, p_{tj}^2)} \frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = \frac{1}{p_{ti}^2}$$



Anti-k_t in action

Clustering grows around hard cores

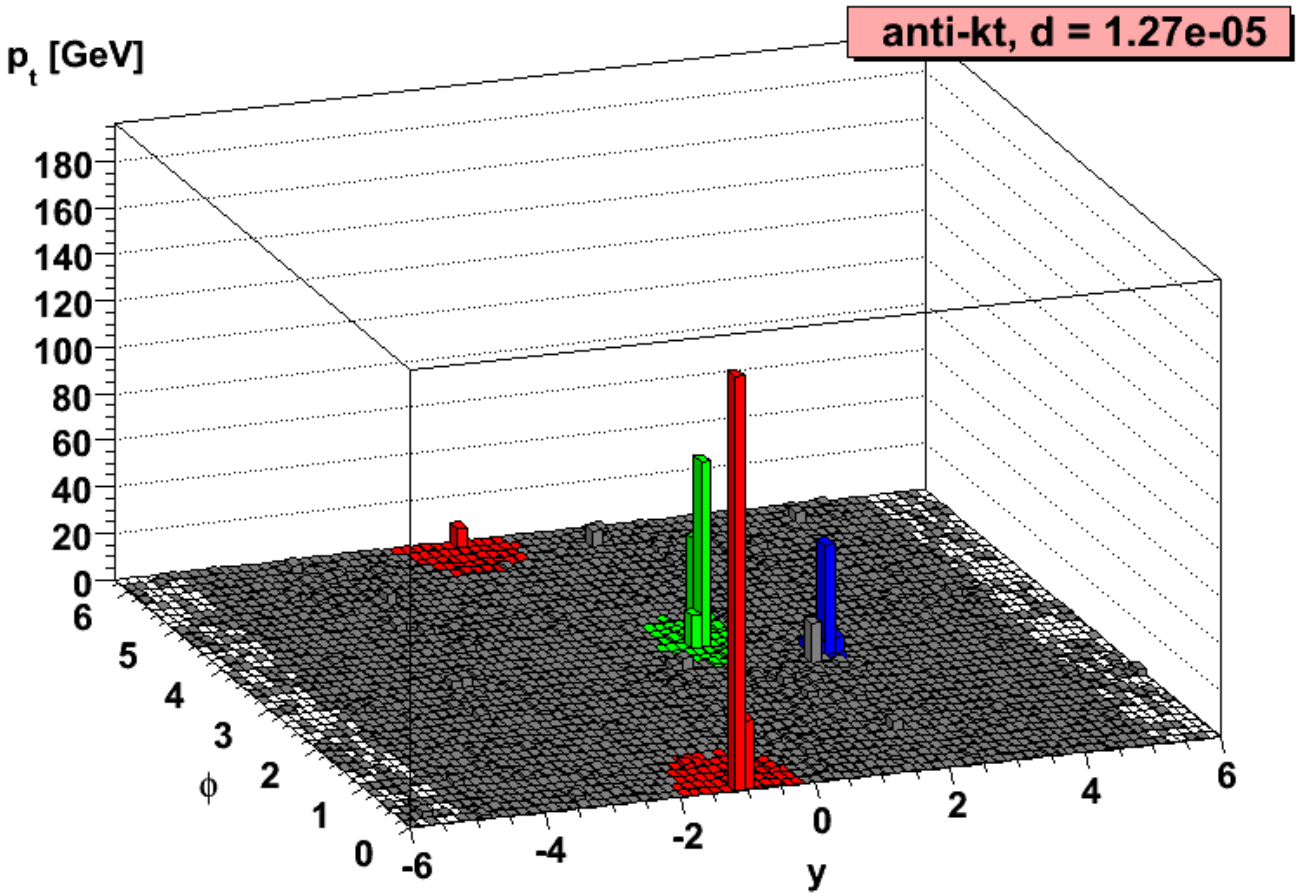
$$d_{ij} = \frac{1}{\max(p_{ti}^2, p_{tj}^2)} \frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = \frac{1}{p_{ti}^2}$$



Anti-k_t in action

Clustering grows around hard cores

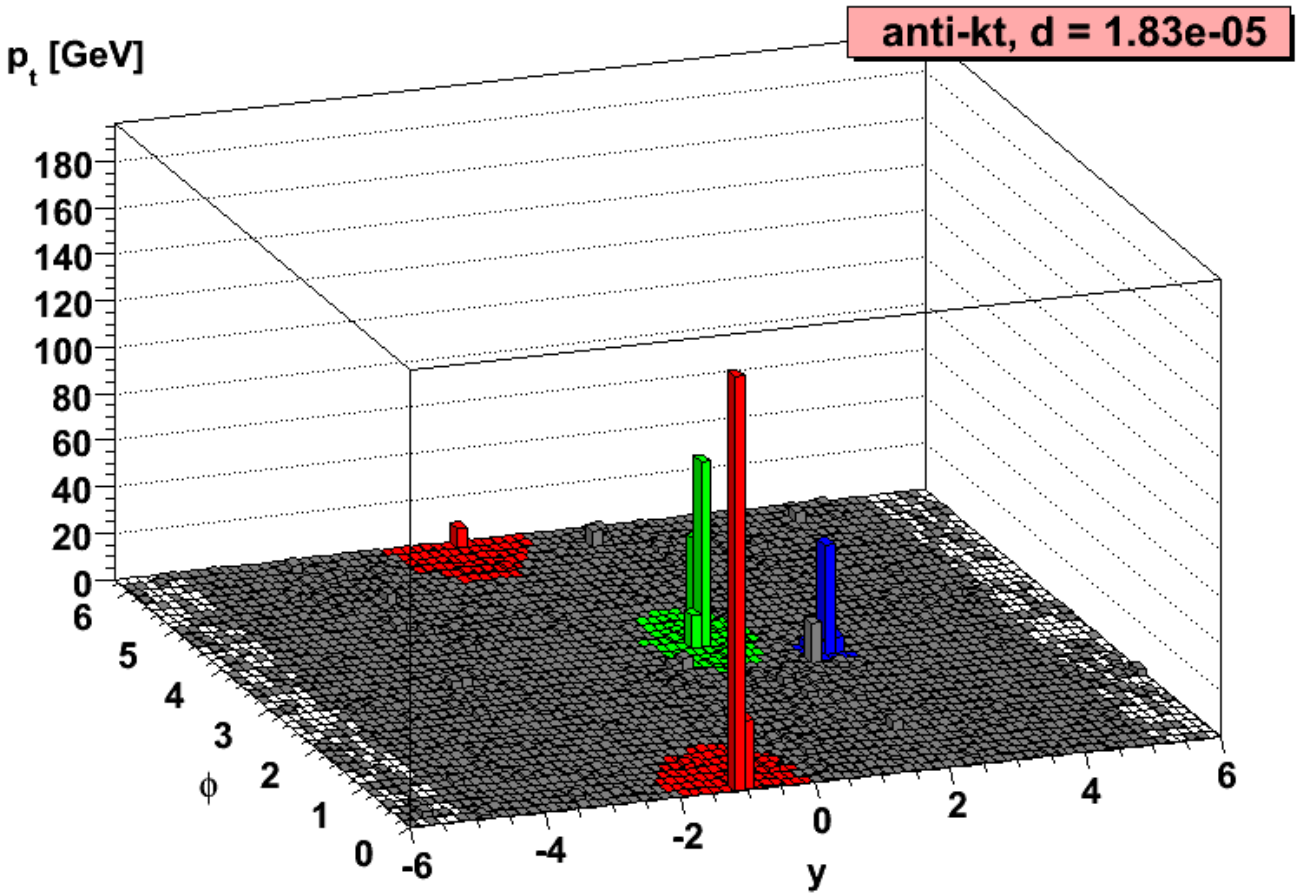
$$d_{ij} = \frac{1}{\max(p_{ti}^2, p_{tj}^2)} \frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = \frac{1}{p_{ti}^2}$$



Anti-k_t in action

Clustering grows around hard cores

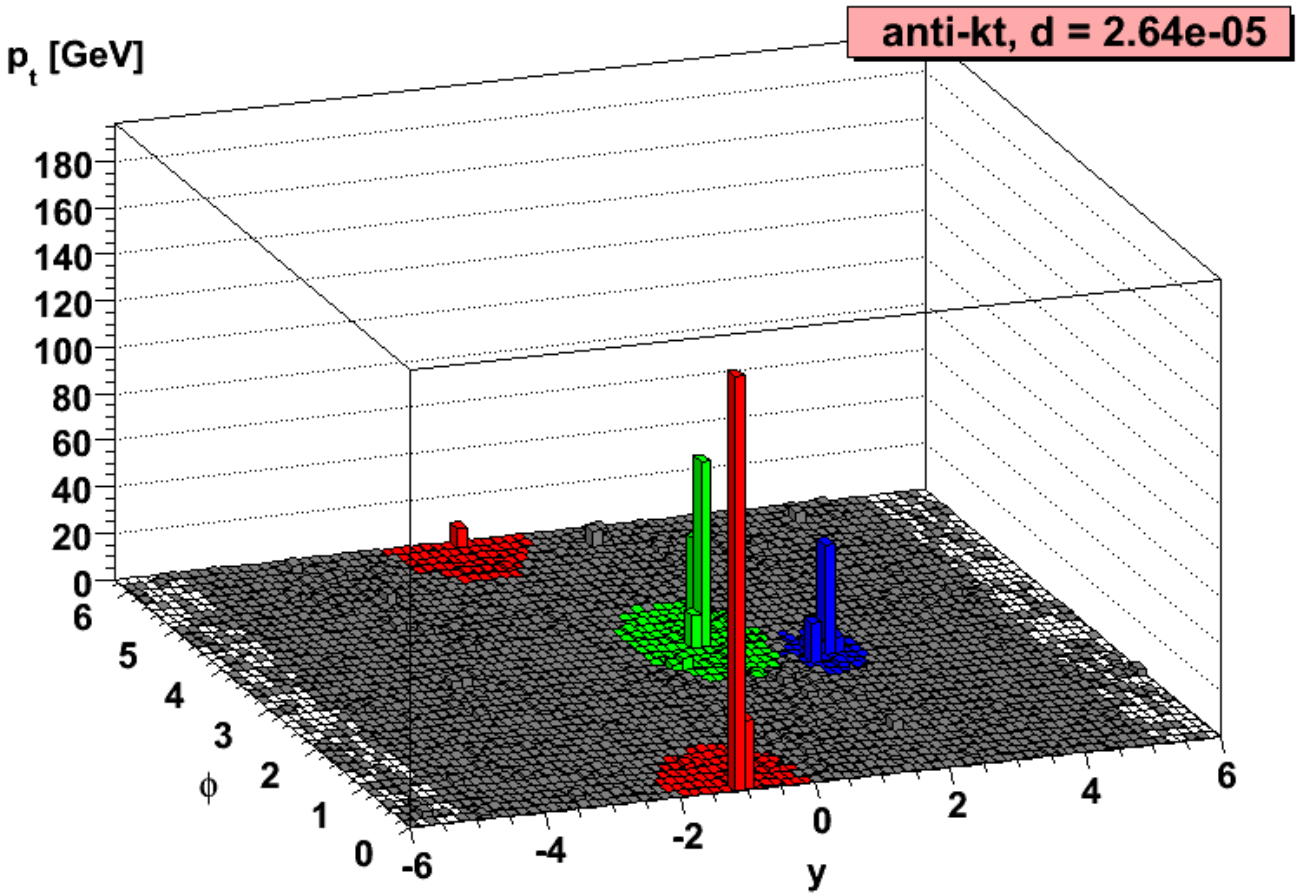
$$d_{ij} = \frac{1}{\max(p_{ti}^2, p_{tj}^2)} \frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = \frac{1}{p_{ti}^2}$$



Anti-k_t in action

Clustering grows around hard cores

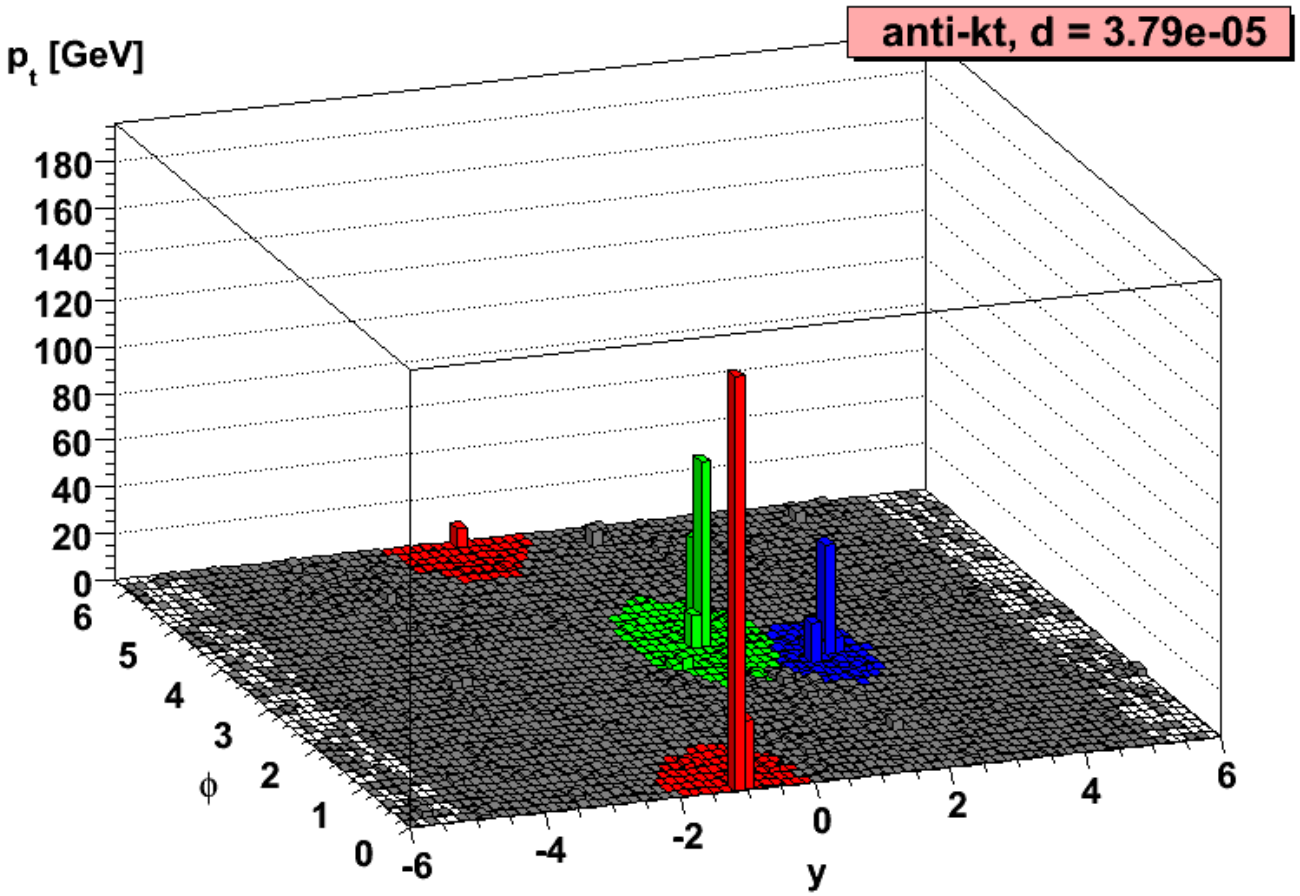
$$d_{ij} = \frac{1}{\max(p_{ti}^2, p_{tj}^2)} \frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = \frac{1}{p_{ti}^2}$$



Anti-k_t in action

Clustering grows around hard cores

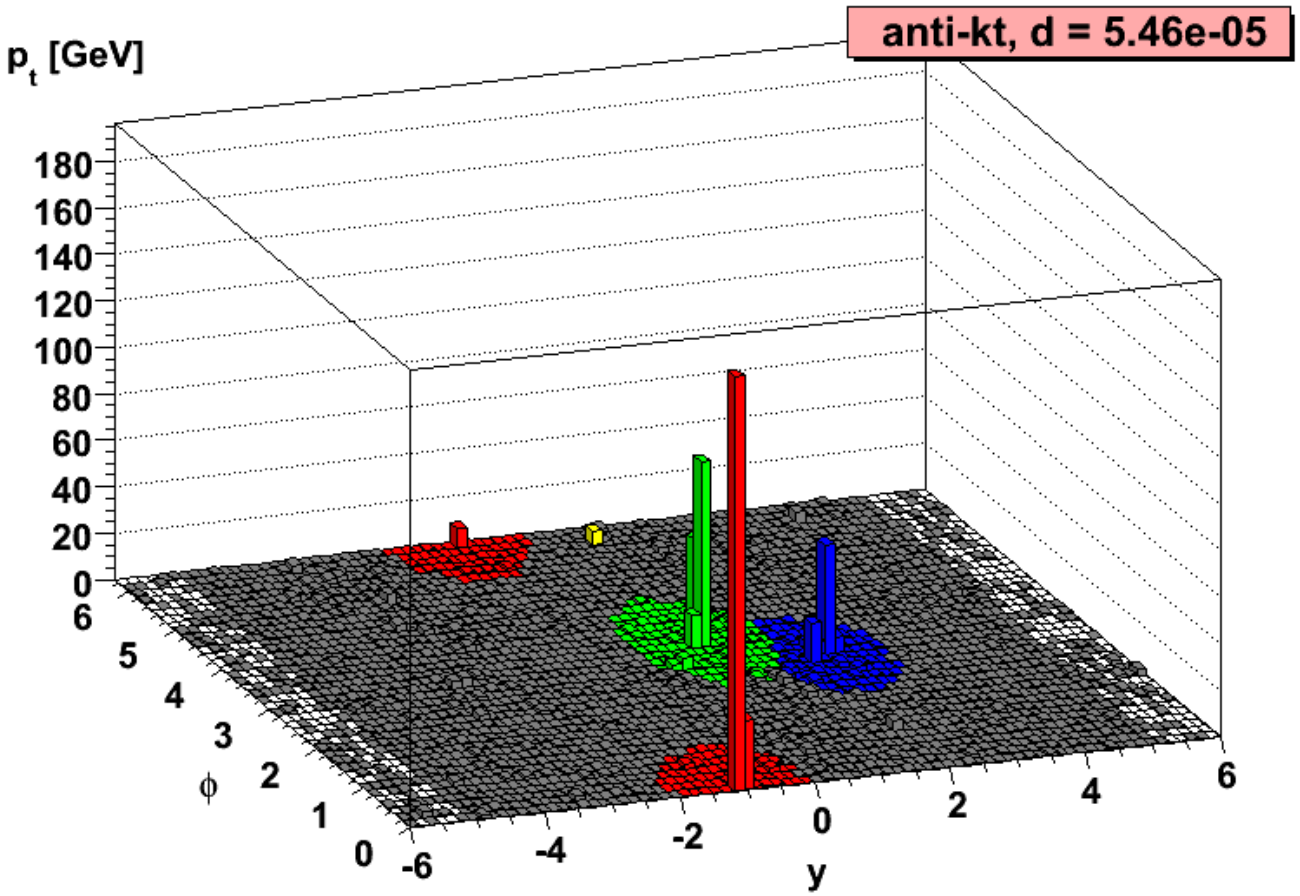
$$d_{ij} = \frac{1}{\max(p_{ti}^2, p_{tj}^2)} \frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = \frac{1}{p_{ti}^2}$$



Anti-k_t in action

Clustering grows around hard cores

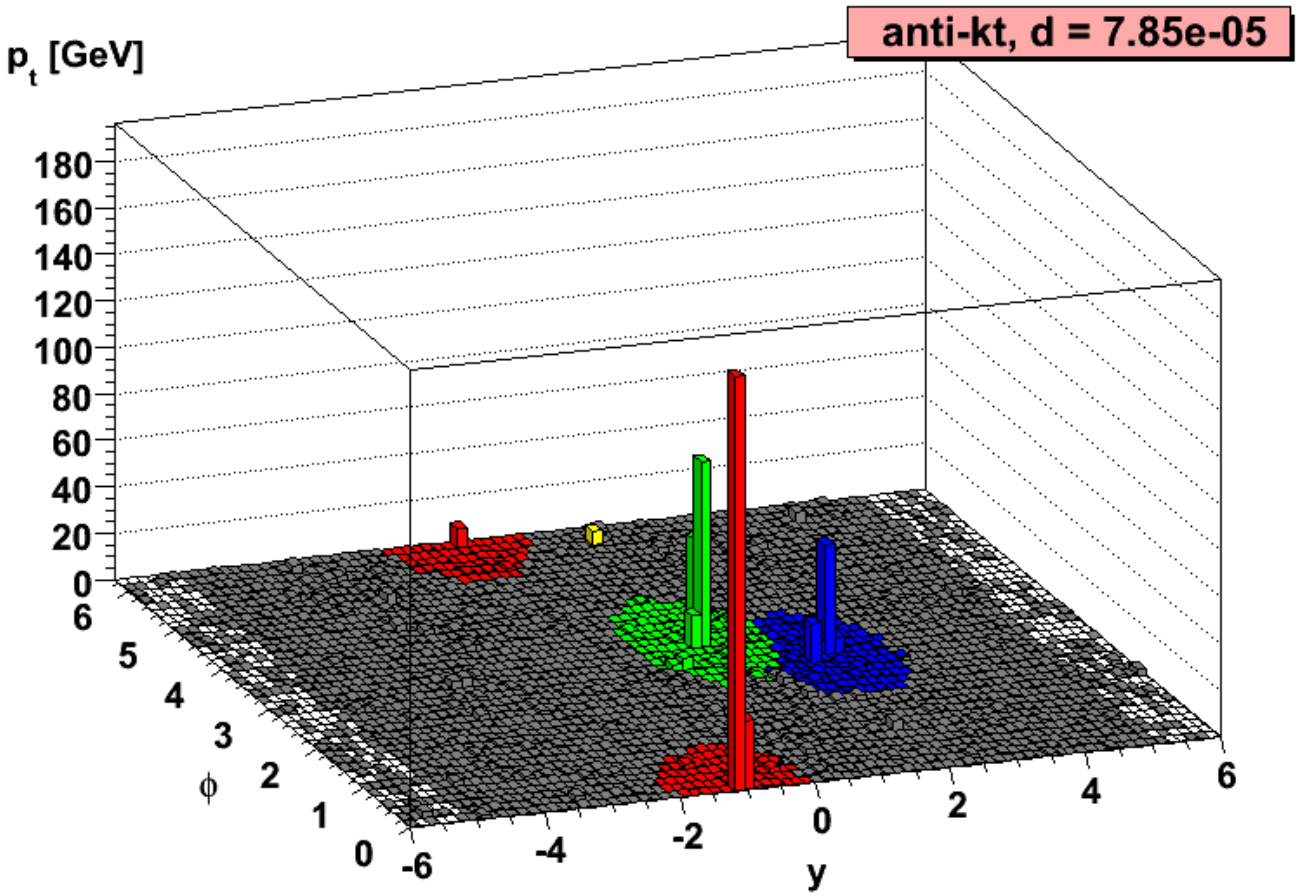
$$d_{ij} = \frac{1}{\max(p_{ti}^2, p_{tj}^2)} \frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = \frac{1}{p_{ti}^2}$$



Anti-k_t in action

Clustering grows around hard cores

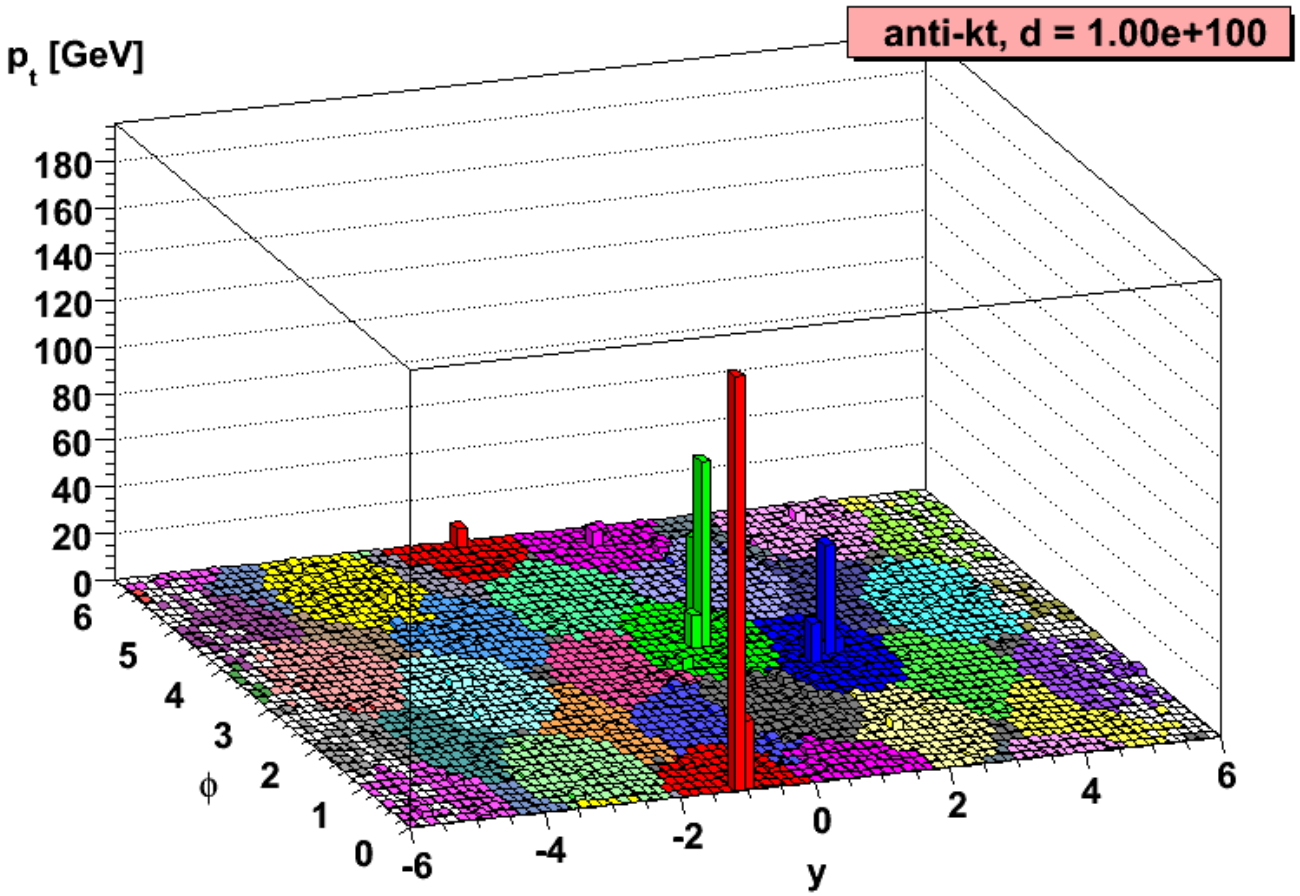
$$d_{ij} = \frac{1}{\max(p_{ti}^2, p_{tj}^2)} \frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = \frac{1}{p_{ti}^2}$$



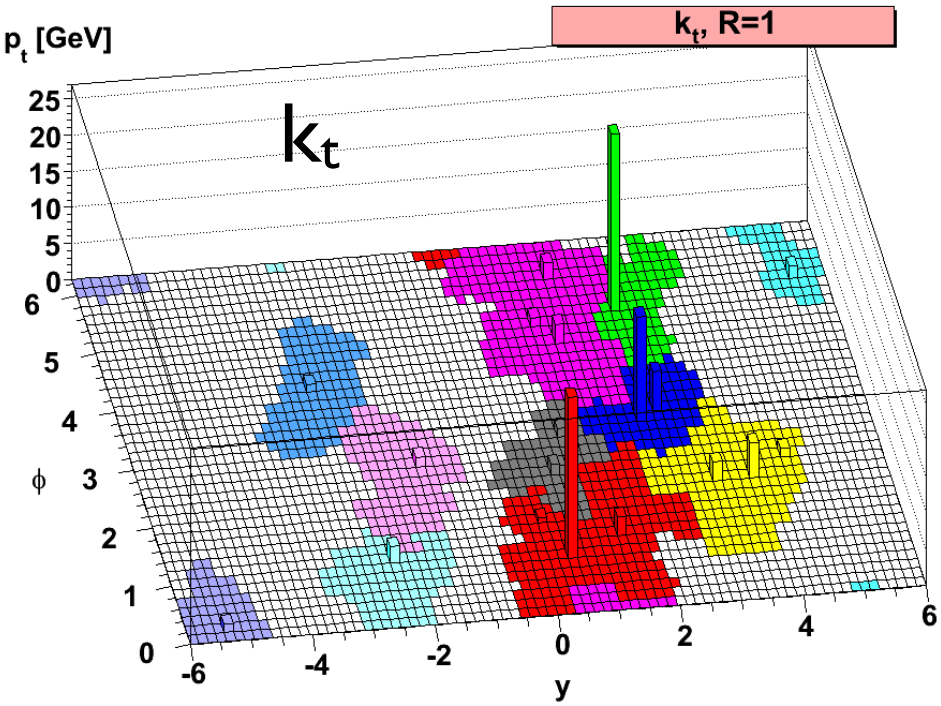
Anti-k_t in action

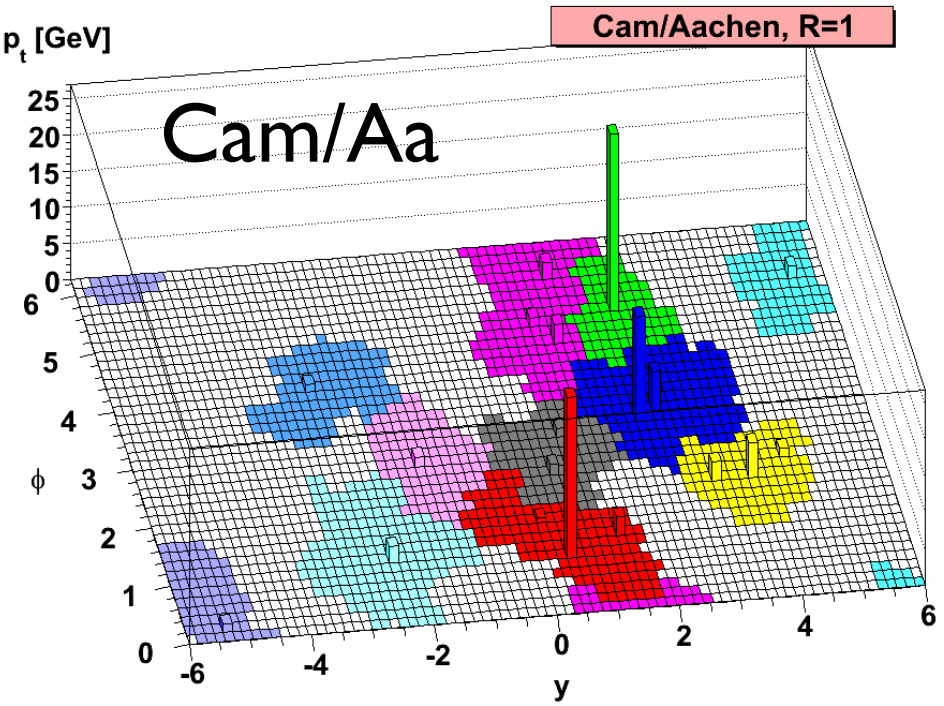
Clustering grows around hard cores

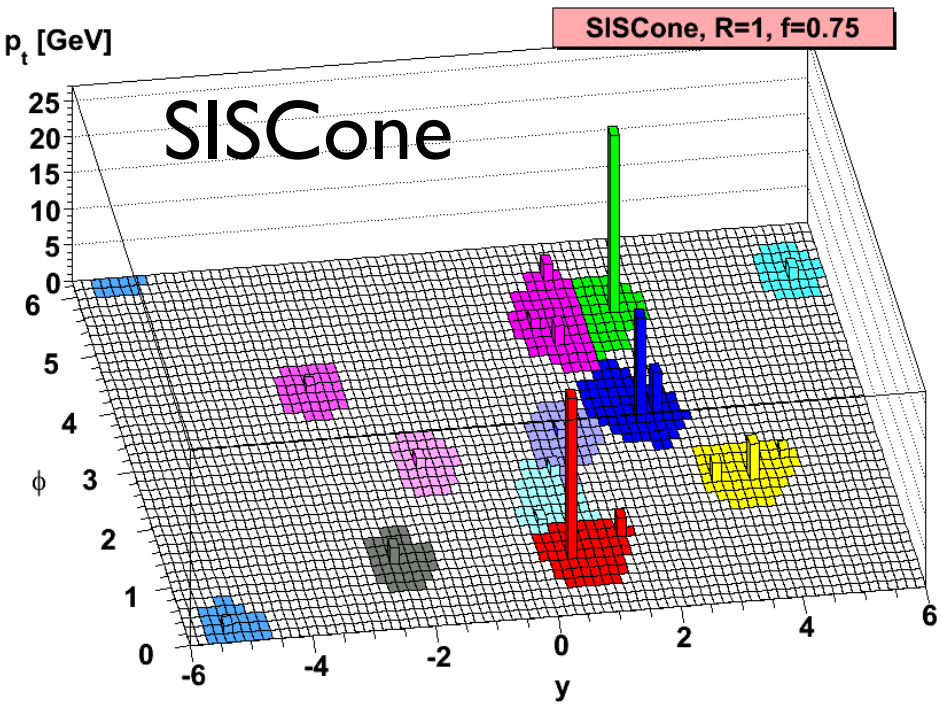
$$d_{ij} = \frac{1}{\max(p_{ti}^2, p_{tj}^2)} \frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = \frac{1}{p_{ti}^2}$$

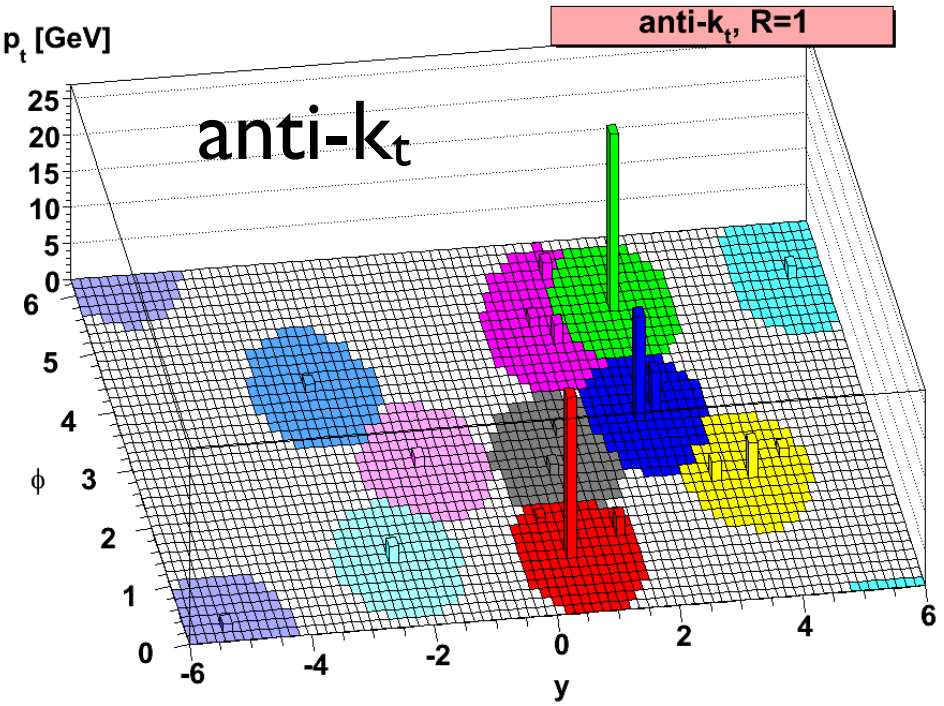


Anti-k_t gives circular jets ("cone-like") in a way that's infrared safe

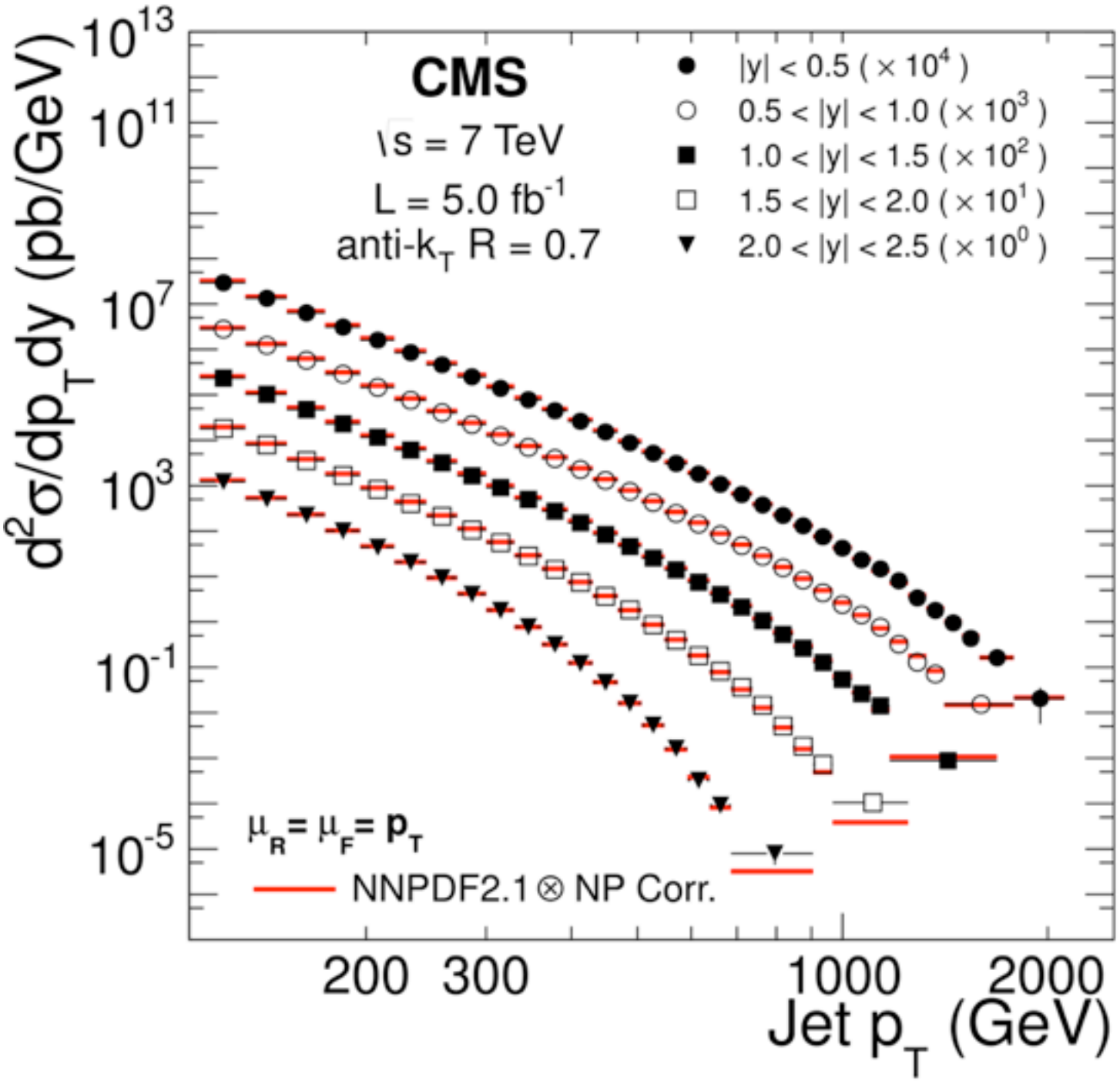

k_t, R=1
p_t [GeV]
k_t
y
ϕ


Cam/Aachen, R=1
p_t [GeV]
Cam/Aa
y
ϕ


SISCone, R=1, f=0.75
p_t [GeV]
SISCone
y
ϕ


anti-k_t, R=1
p_t [GeV]
anti-k_t
y
ϕ

Example of jet observable



Inclusive jet cross section

Excellent theory-data agreement over many orders of magnitude

Recap of Lecture 1

- A vast zoology of jet algorithms has been reduced in the past few years to **4 infrared and collinear safe algorithms**
  - All are implemented in an efficient and fast way
  - Of these, **anti-k_t** is used by all the LHC collaborations as their main algorithm for "finding" jets and measuring inclusive cross sections
- The four algorithms have quite different characteristics, which makes them non easily swappable when specific properties are needed for specific tasks. On the other hand, chances are that one can chose the algorithm which is most appropriate for a specific job