Event reconstruction

Reconstruction is the operation of constructing physics quantities from the raw data collected in the experiment. As a software process, reconstruction is therefore the procedure of data reduction whose main client is data analysis.

The reconstruction process is seen as a collection of independent units, each one providing a set of corresponding objects as output. The reconstruction process can be divided into three steps, corresponding to local reconstruction within an individual detector module, global reconstruction within a whole detector, and combination of these to produce higher level objects.

The reconstruction units providing local reconstruction in a detector module use as input real data from the DAQ system or simulated data representing the real data. These data are in either case called digis. The output from the reconstruction units are RecHits, reconstructed hits which are typically position measurements in tracking type detectors and calorimetric clusters in calorimeter systems. The RecHits are used as the input for global reconstruction.

In the global reconstruction step information from the different modules of a subdetector are combined, although information from different subdetectors is not. For example, tracker RecHits are used to produce reconstructed charged particle tracks and muon system RecHits are used to produce candidate muon tracks.

The final reconstruction step combines reconstructed objects from individual subdetectors to produce higher level reconstructed objects suitable for high-level triggering or for physics analysis. For example, tracks in the tracker system and tracks in the muon system are combined to provide final muon candidates, and electron candidates from the calorimeter system are matched to tracks in the tracker system.

Local reconstruction

Local reconstruction leads to RecHits which contain information about the energy deposition and positions.

In the tracker detectors local reconstruction algorithms search for strips and pixels with a signal exceeding a threshold, and use them as seeds for clusters. Clusters are built by adding neighboring strips/pixels.

In the muon drift tubes, local reconstruction provides the position of a muon hit in a drift cell, determined from the drift time measurement and the effective drift velocity. Three-dimensional track segments within a superlayer are built from hits in each component layer.

In the muon cathode strip chambers, local reconstruction provides position and time of arrival of a muon hit from the distribution of charge induced on the cathode strips. Two dimensional hits are obtained in each layer, and these can be combined to create three-dimensional track segments within each chamber.

In the muon resistive plate chambers, local reconstruction gives the position of a muon hit from the position of clusters of hit strips.

In the electromagnetic calorimeter (ECAL), local reconstruction identifies the position, time of arrival, and energy of localized electromagnetic energy depositions.

In the hadron calorimeter (HCAL), local reconstruction identifies the position, time of arrival, and energy of localized electromagnetic energy depositions.

Global reconstruction

The global reconstruction algorithms use the object created in the local reconstruction within a single detector module, combining them with objects arising from other modules of the same sub-

Reconstruction

detector to produce further objects which represent the best measurement from that subdetector.

- Reconstruction in the tracker system: high/low p_T tracks, displaced vertices etc.
- Reconstruction in the calorimeter system: ECAL+HCAL tower linking to be used as a basis for jet reconstruction
- Reconstruction in the muon system: reconstruction of muons without inner tracker information in an inhomogeneous and nonuniform magnetic field.

Track reconstruction in a dense environment needs an efficient search for hits during the pattern recognition stage and a fast propagation of trajectory candidates. The track reconstruction is decomposed into five logical parts:

- hit reconstruction, which in turn consists of clustering of strips or pixels and estimating a position and its uncertainty.
- seed generation providing the initial trajectory candidates for the full track reconstruction.
- pattern recognition or trajectory building
- ambiguity resolution
- final track fit

Ambiguities in track finding arise because a given track may be reconstructed starting from different seeds, or because a given seed may result in more than one trajectory candidate.

Vertex reconstruction (interaction points) usually involves two steps, vertex finding and vertex fitting. Vertex finding involves grouping of tracks into vertex candidates. The vertex finding algorithms can be very different depending on the physics case (primary or secondary vertex, reconstruction of exclusive decays, etc.). Vertex fitting involves determining the best estimate of the vertex parameters (position, covariance matrix, track parameters constrained by the vertex position and their covariances) for a given set of tracks, as

well as indicators of the fit quality.

Combined reconstruction

The final stage of reconstruction combines input objects created in the global reconstruction within each subdetector, creating objects based on the complete detector.

Photon and electron identification. The global selection of electrons and photons proceed in three steps. The first step uses the calorimeter information only. The second step requires hits in the pixel detectors, consistent with an electron candidate. The success of the matching ECAL supercluster to hits in the pixel detector flags the candidate as an electron, otherwise the candidate is flagged as a photon. In the final step, the selection of electrons uses full track reconstruction, seeded from the pixel hits obtained by the matching step. The selection of photons can instead use isolation cuts and rejection of π^0's based on lateral shower shape and the reconstruction of converted photons.

Muon identification. Global muon identification starts from the standalone muon, adding associated silicon tracker hits and performing final fit to the track. Isolation criteria can be applied to the muon candidates to provide additional rejection against nonprompt muons from b, c and K decays.

Jet reconstruction. Jet reconstruction aims to reconstruct and identify jets arising from the hadronization of a scattered parton, in order to reconstruct its direction and energy. Many reconstruction algorithms exist in the literature, varying in speed, efficiency and resolution. Most algorithms use a clustering technique, in which calorimetric towers close in (η, φ) to a high $\mathrm{E_T}$ tower are summed together, subject to some constraints. For example in the cone algorithm, a seed tower is selected and then all objects sufficiently close in (η, φ) are used to form a proto-

jet. The process of association is iterated until the parameters of the proto-jet have stabilized. The procedure is repeated with the remaining unassociated towers, until no seeding tower with sufficiently high E_T remains.

MET reconstruction (missing E_T). Many channels of discovery at the LHC present as a clear signature for new physics a large missing transverse energy. Missing transverse energy is typically reconstructed by summing the energy depositions times $\sin\theta$ over the whole calorimeter system, and taking the transverse momentum carried by the muons into account.

b and τ tagging

Many physics channels produce b jets and τ jets in the final state. These need to be distinguished from more copious backgrounds containing only light flavoured jets. The top quark, for example, decays almost exclusively into a W boson and a b quark.

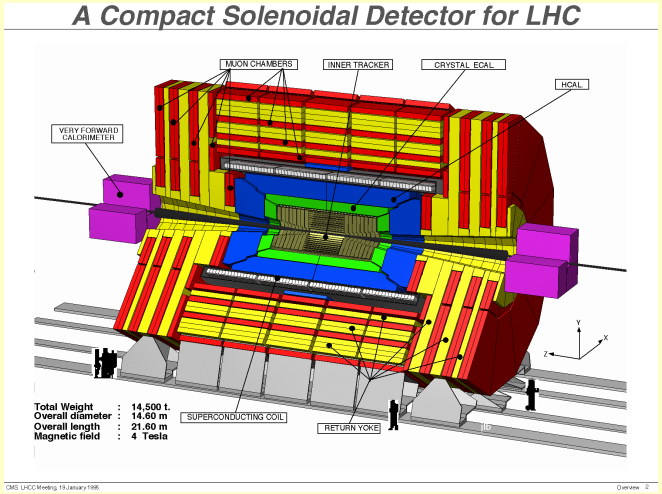
Inclusive tagging of b jets, as opposed to lighter flavour jets, mainly relies upon relatively distinct properties of b-hadrons such as proper lifetime (lifetime at rest), large mass, decays to final states with high charged track multiplicity, relatively large semileptonic branching ratios and a hard fragmentation function.

The b-tagging algoritms rely on the reconstruction of lower level physics objects. At LHC b tagging typically will be applied to jets. Most of the b hadron properties used for b tagging are exploited using charged particle tracks. To increase the signal-to-noise ratio, only tracks fulfilling certain quality criteria are selected.

For b tagging the track measurement precision close to the interaction point is most relevant. The simplest b tagging algorithm counts high quality tracks in a cone around the jet axis with high enough *impact parameter* significance. More

sophisticated algorithms are described in the literature.

Tau identification is based on τ jet properties such as lifetime, mass, small charged track multiplicity, collimation and isolation of τ decay products. Tau jet identification is important, since τ lepton decays hadronically about 65% of the time producing a τ jet. In general, the primary requirement for τ jet identification is the isolation of a collimated group of charged particles in the tracker.



Trigger

Trigger is the start of the physics event selection process. For the nominal LHC design luminosity of $10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$, an average of 17 events occurs at the beam crossing frequency of 25 ns. This input rate of 10^9 interactions every second must be reduced by a factor of at least 10^7 to 100 Hz, the maximum rate that can be absorbed by the on-line computer farm.

The CMS Level-1 (L1) trigger system is based on custom electronics. The High Level Trigger (HLT) system relies on software.

The CMS L1 trigger is based on the presence of local objects such as muons, electrons, photons and jets, and it employs global sums of E_T and MET. The requirements on the L1 trigger are chosen to provide a high efficiency for the hard scattering physics to be studied at the LHC. The L1 trigger operates without the benefit of full detector granularity and full-resolution data.

Reconstruction

The HLT is used to provide the selection of 1:1000 from the maximum average Level-1 output rate of 10^5 Hz to the final storage rate of 100 Hz. The HLT has a total processing time of up to $\sim$1 s, during which time interval the data are stored in memory.

Table 1: Example trigger thresholds

Trigger	Threshold (GeV)
Inclusive electron	29
Di-electrons	17
Inclusive photons	80
Di-photons	40,25
Inclusive muon	19
Di-muons	7
Inclusive τ-jets	86
Di-τ-jets	59
1-jet + MET	180 + 123
1-jet OR 3-jets OR 4-jets	657,247,113
Electron+τ-jet	19 + 45
Inclusive b-jets	237

To minimize the CPU requirement by the HLT, a key feature of the algorithms is to reconstruct the information in the CMS detector only partially. An example set of thresholds for some trigger objects are shown in Table 1.

The trigger algorithms can be simulated and their efficiences studied by simulation. To have a realistic physics analysis, the trigger simulation needs to be included in the event selection.

Event reconstruction software in CMS

Although the used reconstruction algorithms and methods are general and used widely in HEP, the reconstruction software is highly experiment dependent. The reconstruction sotfware is written by the physicists for their own use. The analysis software is even more specific: as the reconstruction is done the same way within an experiment, the analysis using the reconstructed objects varies from physics channel to physics channel.

Reconstruction

Here we do not go into the CMS reconstruction software, but describe only the basic principles how to reconstruct the physics objects for your own analysis.

The first place to start is the documentation. Although it may be in many cases almost non-existent, it is an obvious place to start.

Use tutorials and examples as working examples from which you can take the relevant part or enlarge to fulfill your needs.

Go to the source code and look for validation code and examples available. This is often a very fruitful approach.

. . . And most of all, ask help from the experts responsible for writing/maintaining the software in your experiment, or from other users who may have already solved your problem and who may have code available which you can easily adopt to your study.

How to browse the source code in CMS? The first way is using the GitHub browser, or the reference manual. The source can also be accessed directly via scram,

scram list

Here you see the paths to the source code, go to the source code directory and start digging with ls, grep, find | xargs grep etc. If you find something interesting, check out (git cms-addpkg) the code, and test it. Sometimes it doesn't work, and in such a case you did not find a *working* example, and you need to keep searching . . .

The config for running the DIGI and RECO steps in CMSSW are created with cmsDriver.py:

```
cmsDriver.py digi -s DIGI,L1,DIGI2RAW,HLT:GRun
--conditions auto:run2_mc_GRun --eventcontent RAWSIM
--datatier GEN-SIM-RAW -n -1 --no_exec --filein
file:fin.root --fileout fout.root


cmsDriver.py reco -s RAW2DIGI,RECO --filein

file:fin.root --fileout fout.root --conditions

auto:run2_mc_GRun --no_exec -n -1
```