Real-time Data Compression for Mass Spectrometry

Jose de Corral

2015 GPU Technology Conference

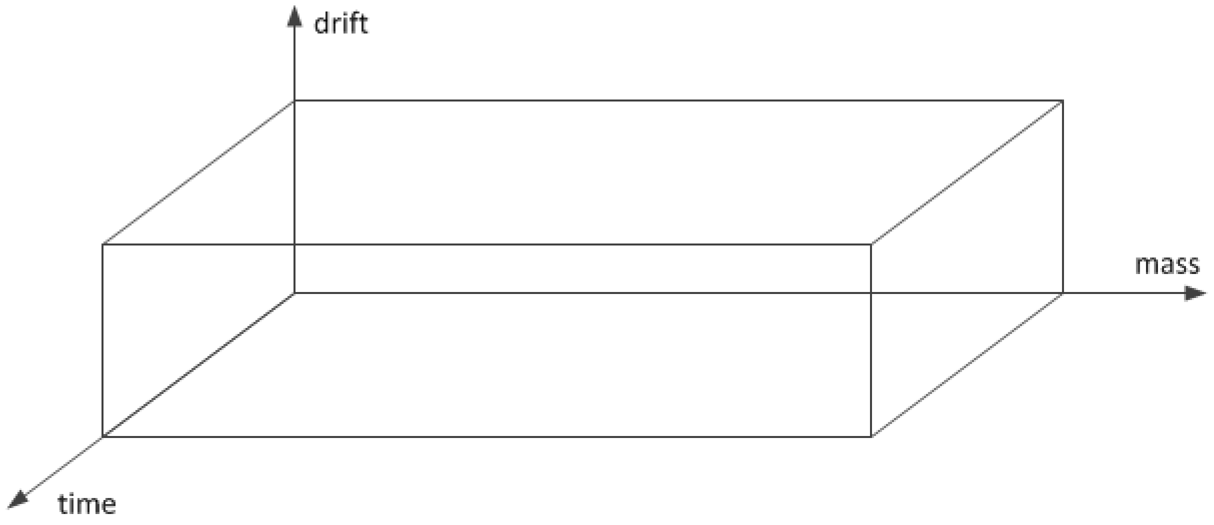
Introduction to LC/IMS/MS

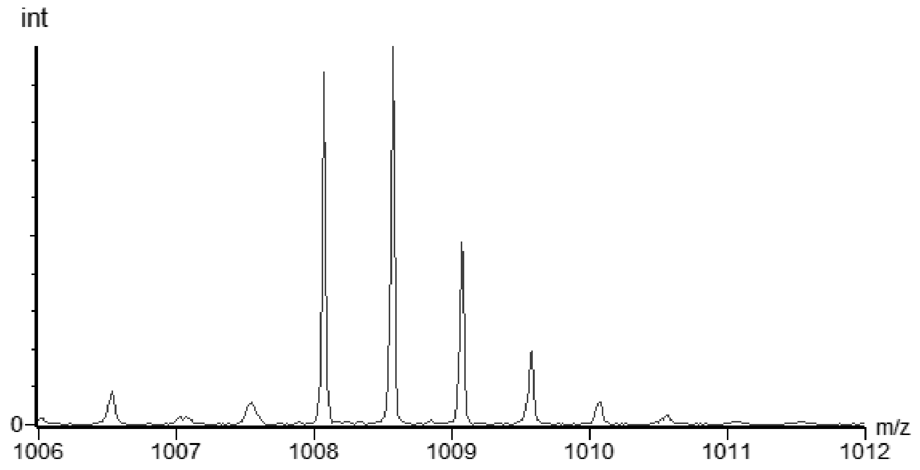
- LC/IMS/MS is the combination of three analytical techniques
  - Liquid Chromatography (LC)
  - Ion Mobility Separation (IMS)
  - Mass Spectrometry (MS)
- Main applications
  - Biopharmaceutical
  - Life Sciences
  - Food safety
  - Environmental Protection
  - Clinical
  - Chemical Materials

Introduction to LC/IMS/MS

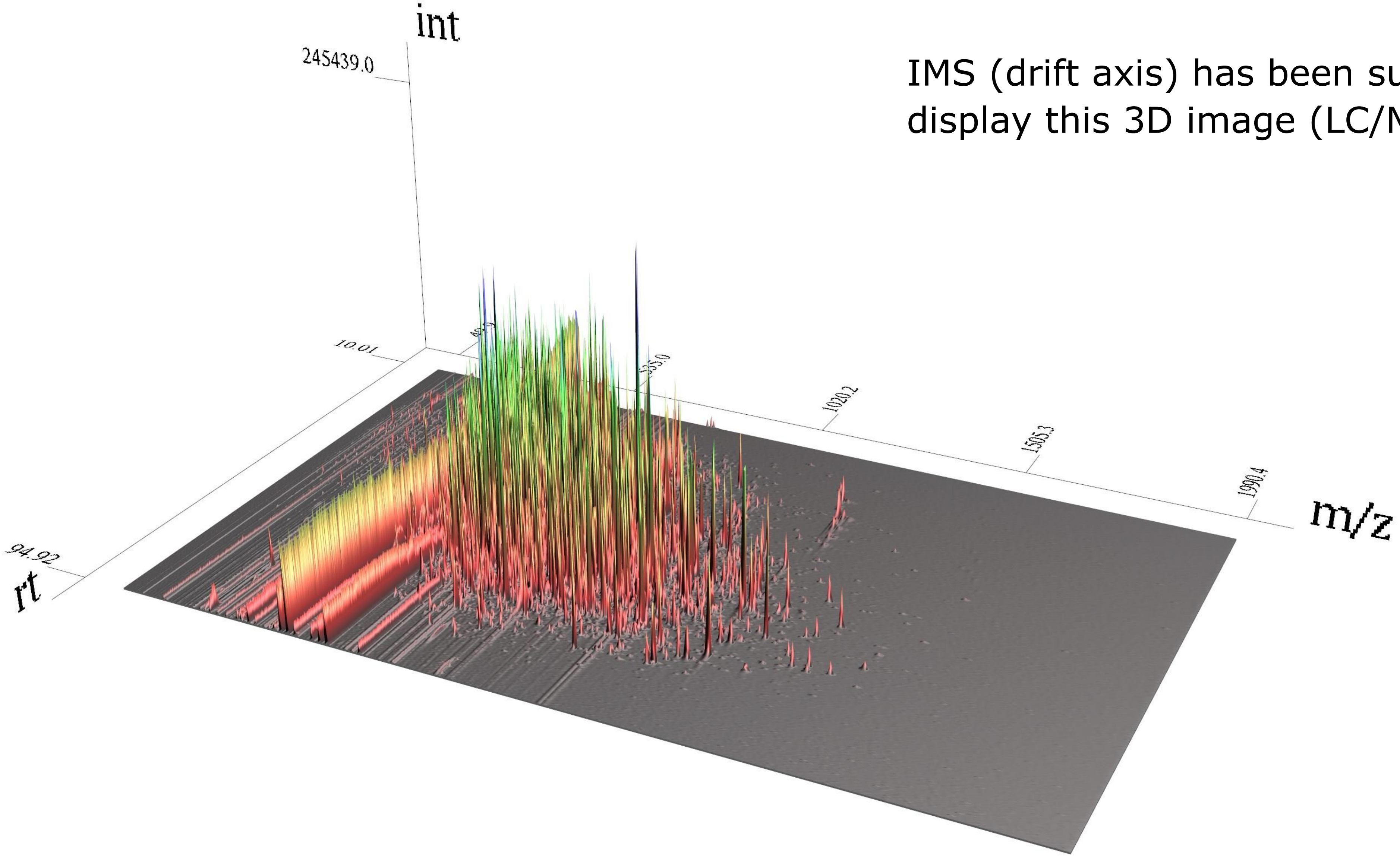
- LC/IMS/MS instruments produce ions and generates 3D data with these three axes
  - Mass to charge ratio of ions (m/z) (mass)
  - Ion mobility drift time (drift)
  - Chromatographic time (time)

- Ion intensity is recorded at each point in this volume
  - In each axis, intensity takes the form of peaks (mostly Gaussian)





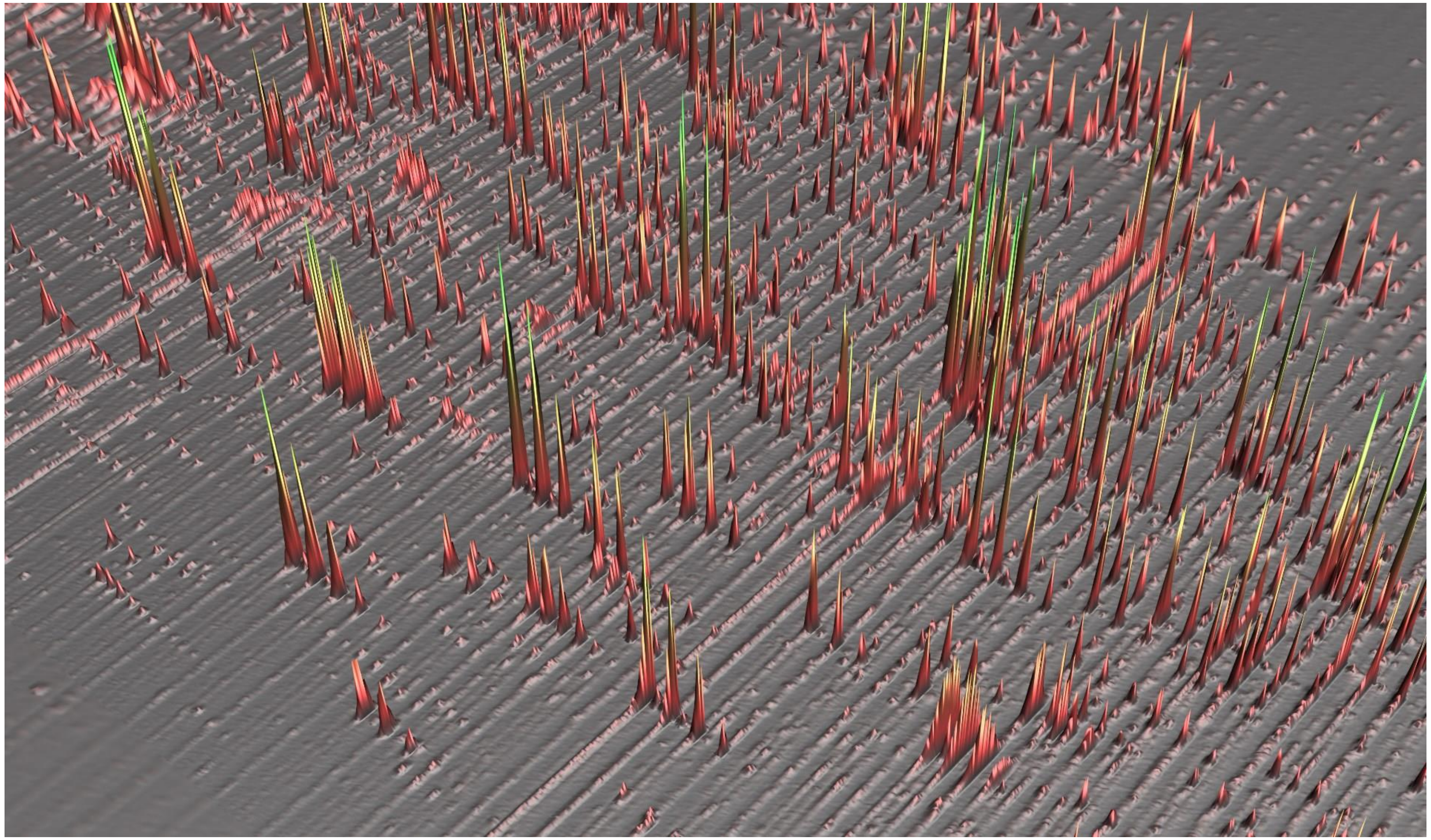
LC/IMS/MS Data Visual Example



IMS (drift axis) has been summed to display this 3D image (LC/MS data)

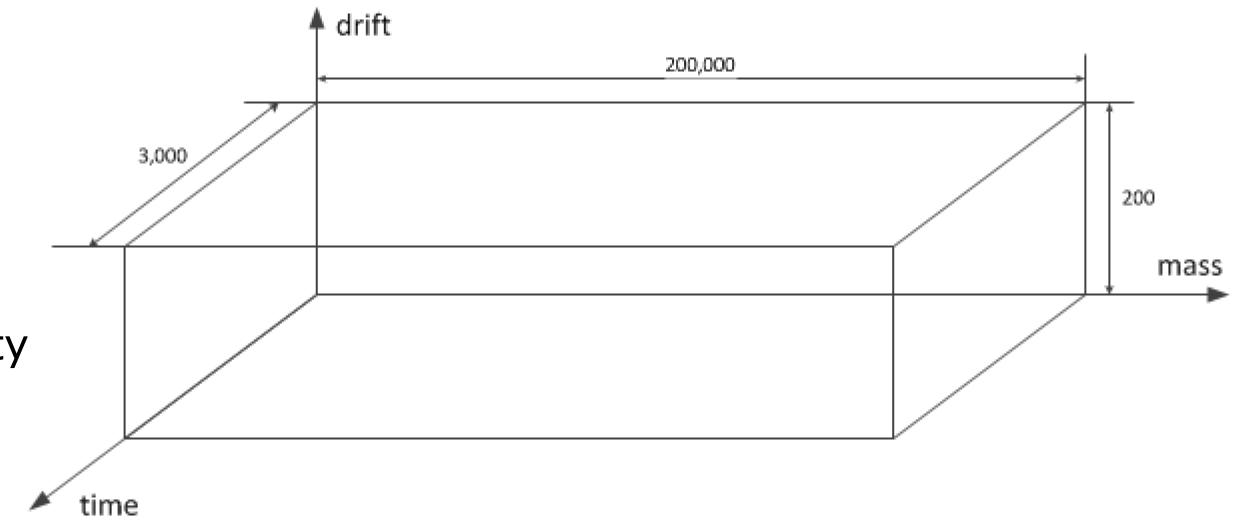
LC/IMS/MS Data Visual Example

A close up of same LC/MS data



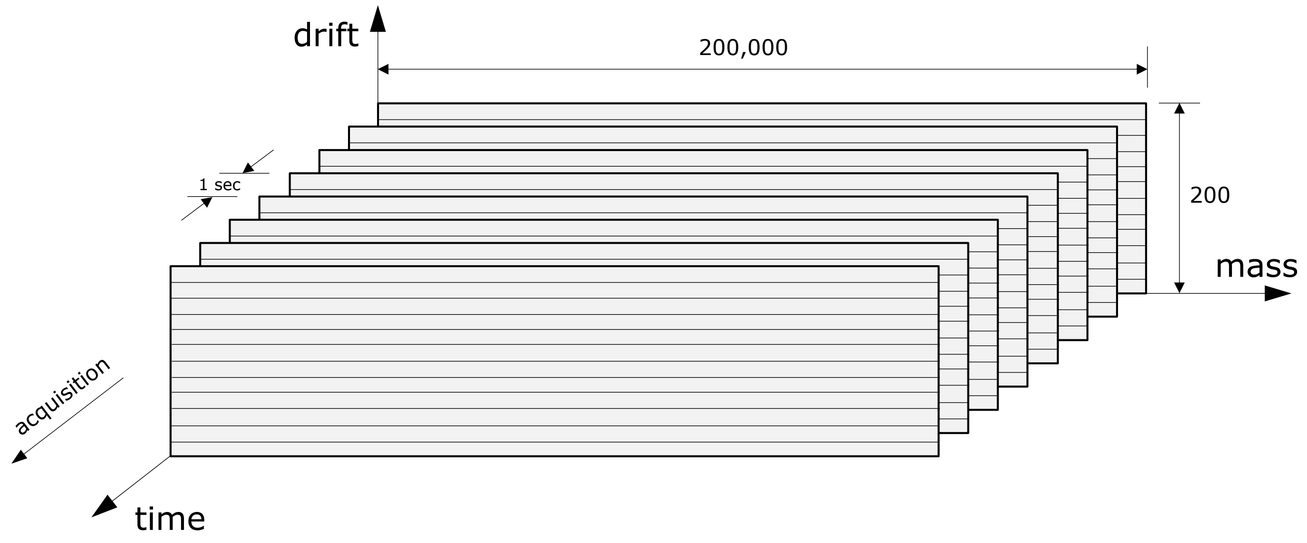
LC/IMS/MS Data Size

- Data volume from a single analysis is typically large (and growing). For example
  - About 200,000 data points in the mass axis
  - 200 data points in the drift axis
  - About 3,000 data points in the time axis
- Volume contains 120 billion data points
  - 480 GB of storage for a 4 byte recorded ion intensity
- Fortunately, the data is sparse
  - Only non-zero intensity values are recorded
  - Data size varies but still may be large (20GB ~ 30GB)
- Each instrument runs 10 or more analysis daily
  - Two data volumes per analysis



LC/IMS/MS Data Acquisition

- LC/IMS/MS instruments generate mass scans of m/z values (every ~5 msec)
- 200 consecutive scans represent 200 ion mobility values (drift) at a point in chromatographic time
- Acquisition time is the chromatographic time (time axis)
  - Each acquisition sampling time (~1 sec) we get a full plane of data (mass x drift)
  - Acquisition time 10 min to 2 hours



LC/IMS/MS Data Compression

- Specific for LC/IMS/MS data
  - Not a general purpose compression method
- Takes advantage of properties of the data
  - How the data is generated and its features (Gaussian peaks)
  - How the data is processed to get analytical results
- Lossy compression method based on relevance
  - Discards data points that do not contribute to the analytical results
  - Keeps all other data points unmodified
- Real-time compression
  - Data is compressed in real-time during acquisition

Compression Algorithm Overview

- The algorithm aims to remove certain points with non-zero intensity
  - Points deemed irrelevant for results
- The algorithm has four main steps
  - Before start, data must have all zero intensity data points restored (full data)

A. For each data point, compute the sum of all intensities inside a volume centered at the data point.

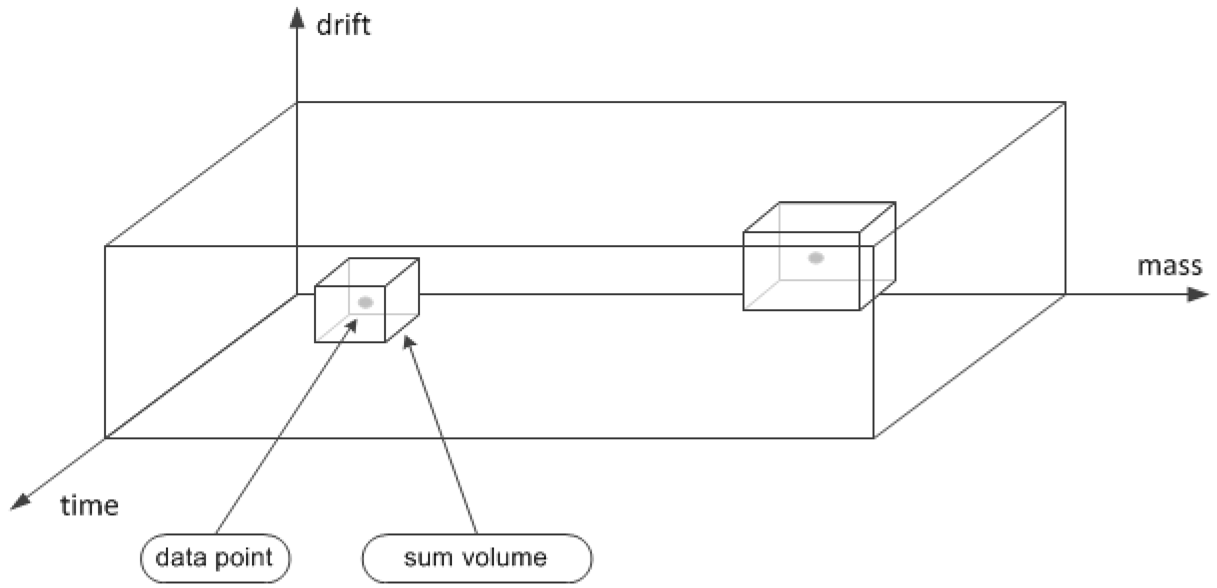
B. Compare each sum value to a given threshold.

C. If the sum is above the threshold, keep all points inside the volume. Otherwise, tag the points inside the volume as "candidates" to be discarded.

D. Once all data points have been processed as above, remove all points that remain tagged as candidates to be discarded.

Compression Algorithm Overview

- Step A sum volume dimensions are not constant
  - Computed from the Gaussian peak width of the data in each axis
  - Dimension in the mass and drift axes increase with the axis



Compression Algorithm Computation

- Simple algorithm but compute intense
- For example, lets assume a volume of just 5x9x7 (mass x drift x time) for step A
  - A naïve implementation would generate 314 sum operations per data point
  - In a data set with 120 billion data points it means 37.7 trillion sum operations
- Of course, the actual implementation uses far less sum operations per data point
  - But still close to a trillion sum operations
- Step B would require 120 billion compare operations in this data set example
- Step C requires a similar amount of computation as step A

Compression Algorithm Memory

- A large amount of device memory is needed to buffer real-time data
  - Consequence of requirement of full data (with zeros) to run the algorithm
  - At a minimum, need to buffer the dimension of the sum volume in the time axis
- Device memory requirements are reduced by tiling the data
  - Only one tile copied to device memory
- Not a trivial tiling
  - Different tile size and boundary criteria in each axis
  - Tiles in main memory have zero intensity data points removed
    - Faster transfers, less main memory
  - Zero intensity data points must be restored after copying each tile to device memory

Step A Implementation

A. For each data point, compute the sum of all intensities inside a volume centered at the data point.

- The sums computation is a box filter convolution in 3D with a variable box volume
- Convolution implemented as a separate convolution
  - A 1D convolution in each axis with a variable sum window
- Ring buffers in shared memory are used to speedup computation
  - Coalesced data is read from device memory only once
  - Make the computation independent of the window width
  - Enable computation “in-place” to save device memory

Step B Implementation

B. Compare each sum value to a given threshold.

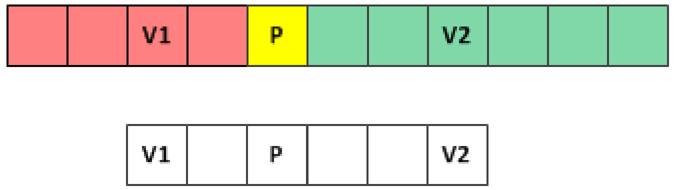
- The comparison result is stored in a single bit flag
  - Flag is set if sum is above the threshold
  - Effective use of the warp vote function __ballot()
  - Sets the data in adequate format for step C computation
  - Saves device memory

Step C Implementation

C. If the sum is above the threshold, keep all points inside the volume. Otherwise, tag the points inside the volume as “candidates” to be discarded.

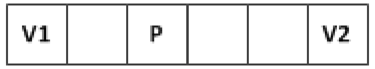
- Doing this computation traversing all volumes in parallel is inefficient
  - Only volumes with the comparison flag set do some work
  - All points inside volumes with the comparison flag set must be visited
  - Some points may be visited multiple times
- Instead, approach the problem from the perspective of the point to be “kept/discarded”
  - Check the comparison flag of all volumes the point contributes to.
  - Keep the point if it contributes to any volume with the comparison flag set. Discard it otherwise.

Step C Implementation



- A 1D example where point P contributes to six volumes
  - Point V1 is the center of a volume with a window width of 5 (the red volume)
  - Point V2 is the center of a volume with a window width of 7 (the green volume)
  - The red volume is the furthest volume to the left P contributes
  - The green volume is the furthest volume to the right P contributes
- P contributes to all volumes with center between V1 and V2
  - We have to examine only those six comparison flags

Step C Implementation



- The implementation is similar to the sums computation in step A
  - But summing “comparison flags” instead of ion intensities
  - This is called the “keeps” computation
- For each data point, compute the sum of all “comparison flags” inside a volume
  - The volume encloses the center point of all sum volumes the point contributes to.
  - If the sum of the flags is above zero, the point is kept. Discarded otherwise.

Step C Implementation

- The keeps computation is a box filter convolution in 3D with a variable box volume
- Convolution implemented as a separate convolution
  - A 1D convolution in each axis with a variable sum window
- Ring buffers in shared memory are used the same way as in step A
  - Flags are read from device memory only once
  - Computation “in-place” and independent of the keep window width
- The sum of flags in each axis is, in turn, stored in a single bit flag
  - Flag is set if sum is above zero
  - Next axis convolution uses this flag as input

Results

- As expected, data size reduction is data dependent
  - Varies between 30% and 90% (data reduced to 70% - 10% of original)

- High computation performance on a Tesla K40
  - Compressing simultaneously two data volumes totaling 164 billion points (36.6 GB) takes about 90 seconds
  - Well beyond the requirements for real-time compression (acquisition time was 75 min)
  - At least a speed up of 70x over a CPU implementation

Results

- Analytical results are essentially the same
  - With correct volume dimensions in step A and the right threshold in step B
  - Setting the threshold too high degrades the results

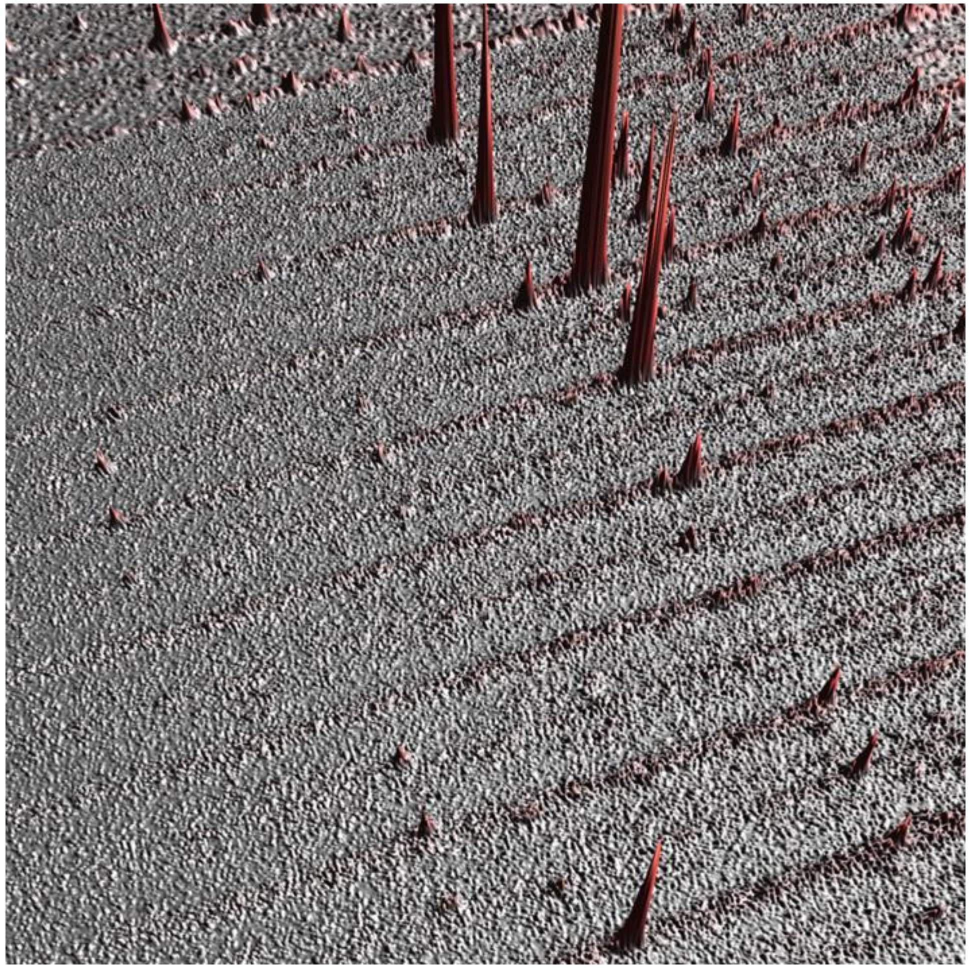
Data	Orig. size	Comp. size	Orig. result	Comp. result
500 ng Ecoli	14.5 GB	9.1 GB	969 proteins	969 proteins
100 ng Ecoli	8.6 GB	3.9 GB	840 proteins	836 proteins
Metabolomics set (10)	2.6 – 1.0 GB	1.2 – 0.06 GB	Indistinguishable PCA scores	

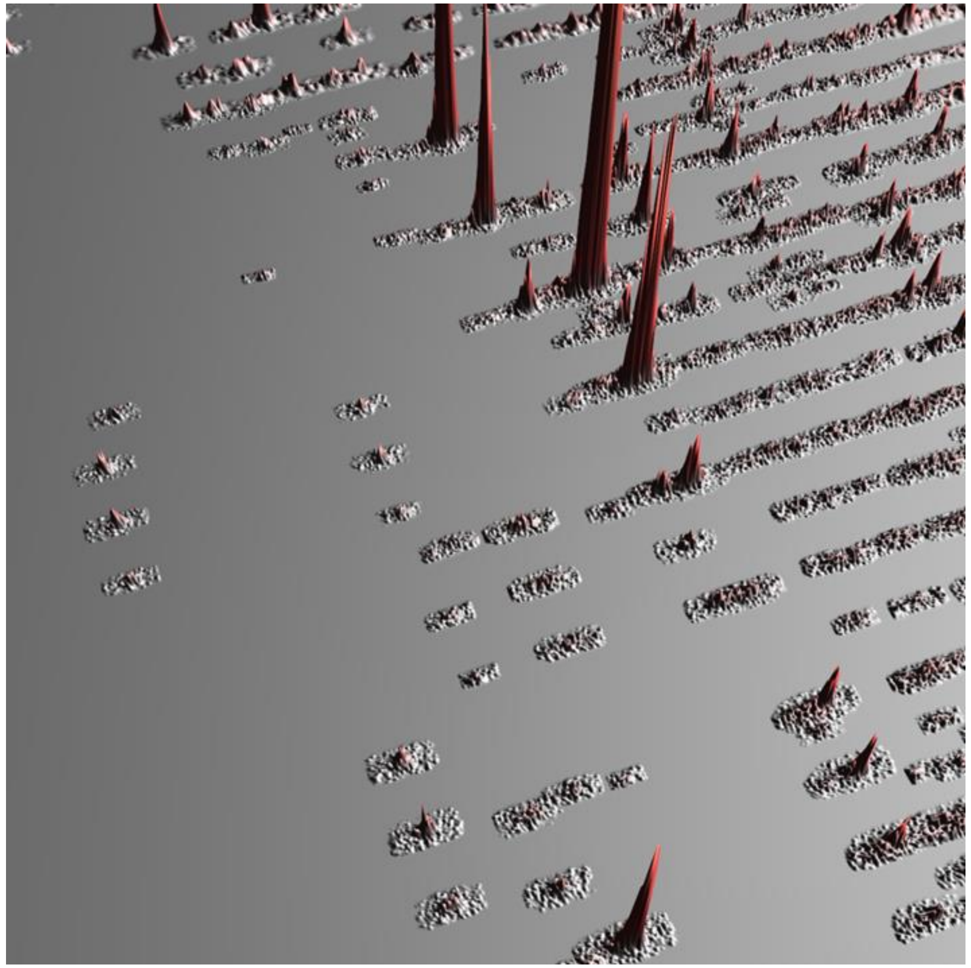
- Side benefit of compression algorithm
  - Post-acquisition data processing takes less time

Visual Results (LC/MS data)

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

Original data (11.1 GB)	Compressed data (3.4 GB)

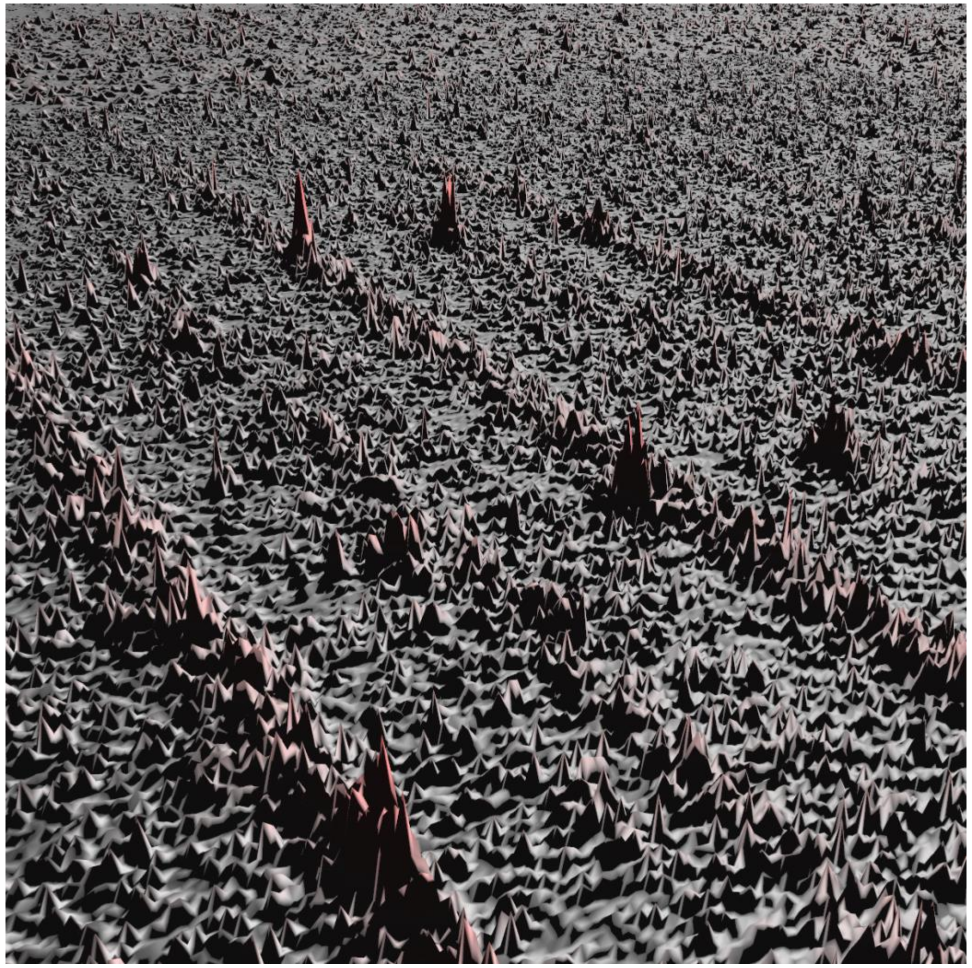


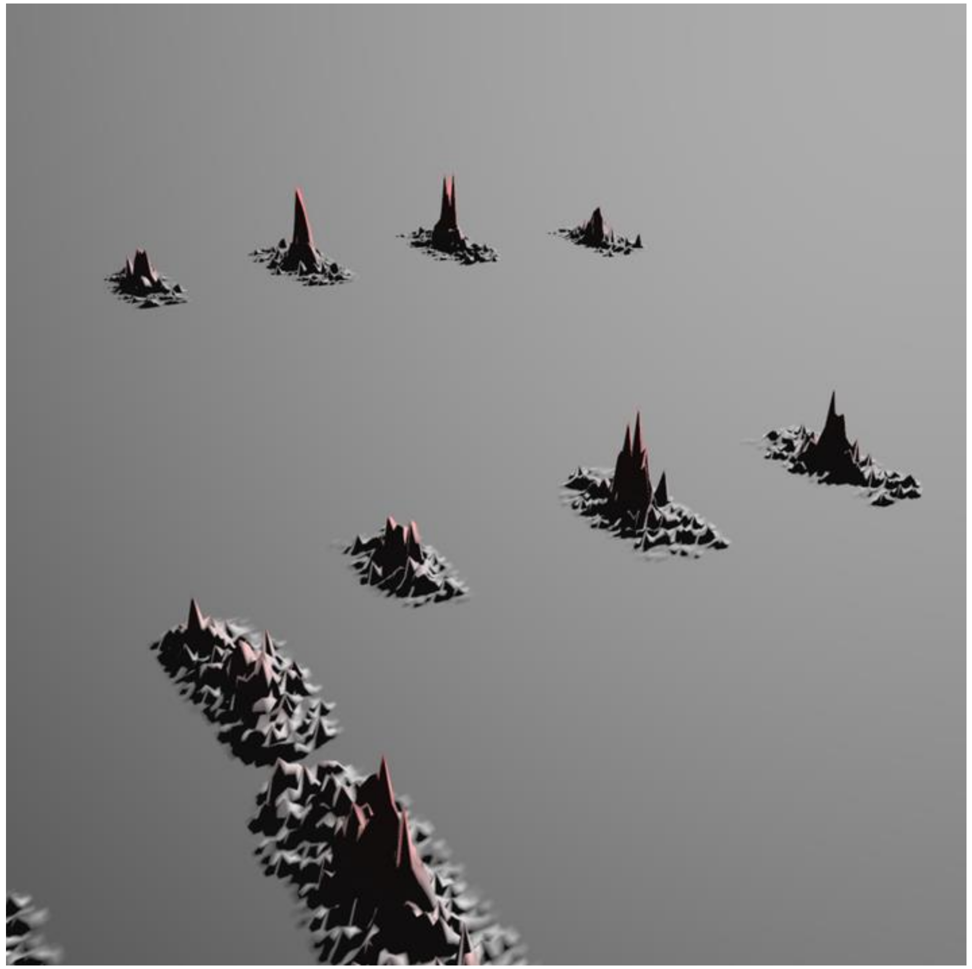


Visual Results (LC/MS data)

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

- Notice noise peaks near edge of larger peaks are preserved
  - A simple thresholding algorithm would have removed them too





Questions?

jose.de.corral@waters.com