# Tutorial Week

SWP 2012

# Timeline

- Tuesday:
  - Important websites
  - OpenMS User tutorial
  - C++ exercise
- Wednesday:
  - Install & Compile OpenMS dev
    - » https://open-ms.svn.sourceforge.net/svnroot/open-ms/branches/SWP2012
  - OpenMS: Library Structure, Important Classes, CMake
  - Dev tutorials
- Friday:
  - MapAlignment in detail

# User Tutorial Download

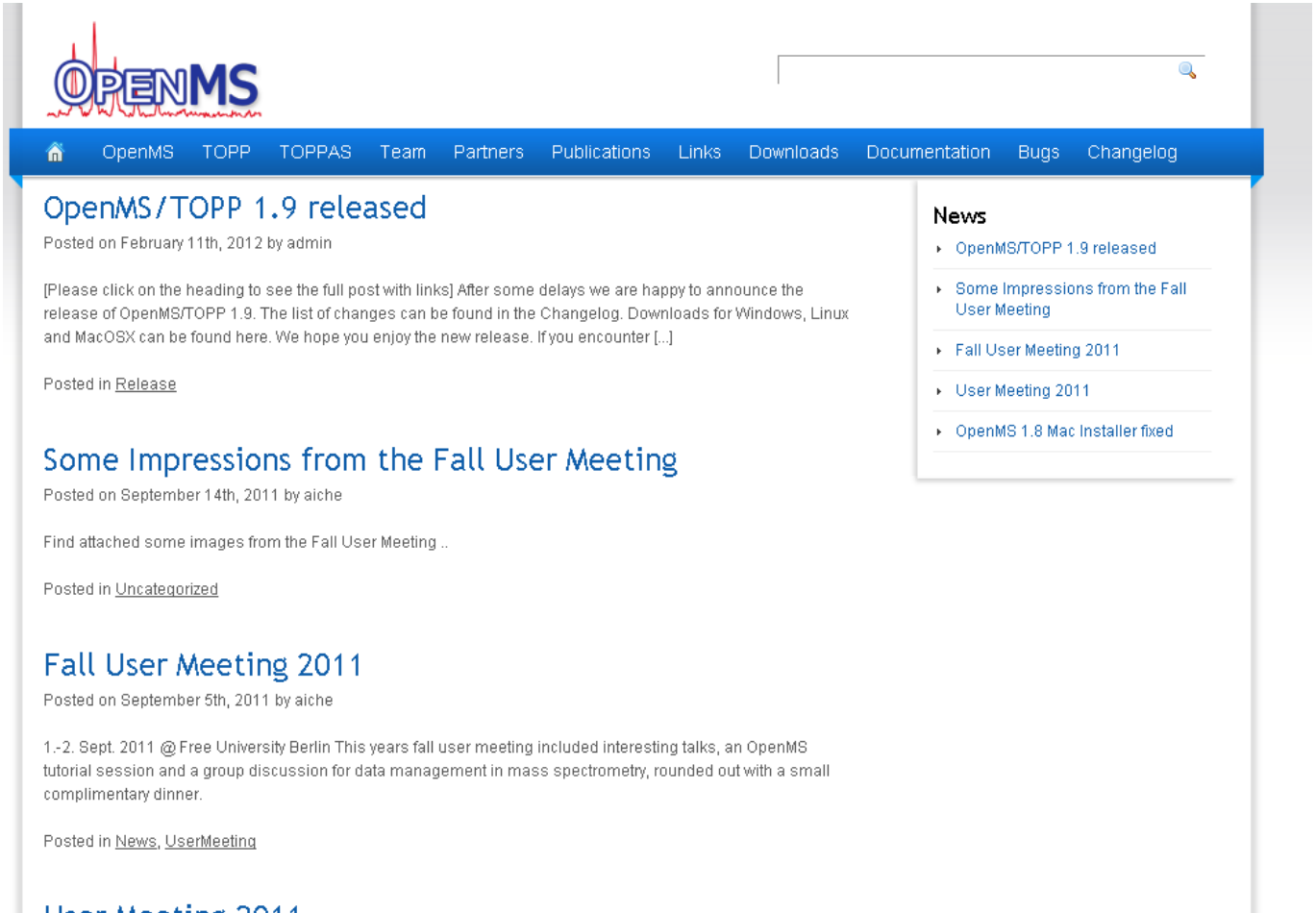
- http://sourceforge.net/apps/trac/open-ms/wiki

  → OpenMS tutorials: dowload tar.gz

- OMSSA: http://pubchem.ncbi.nlm.nih.gov/omssa/download.htm

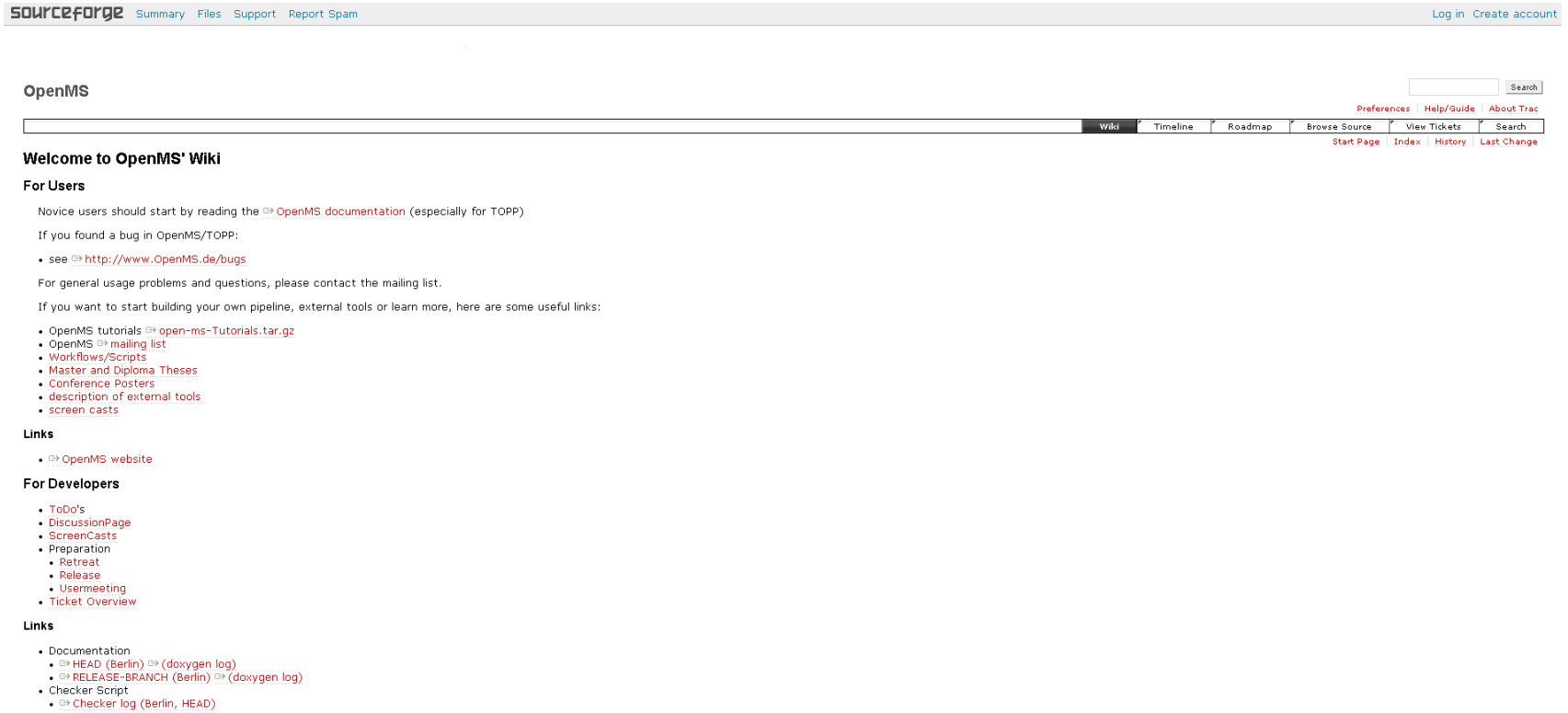
# Important Links

- http://OpenMS.de



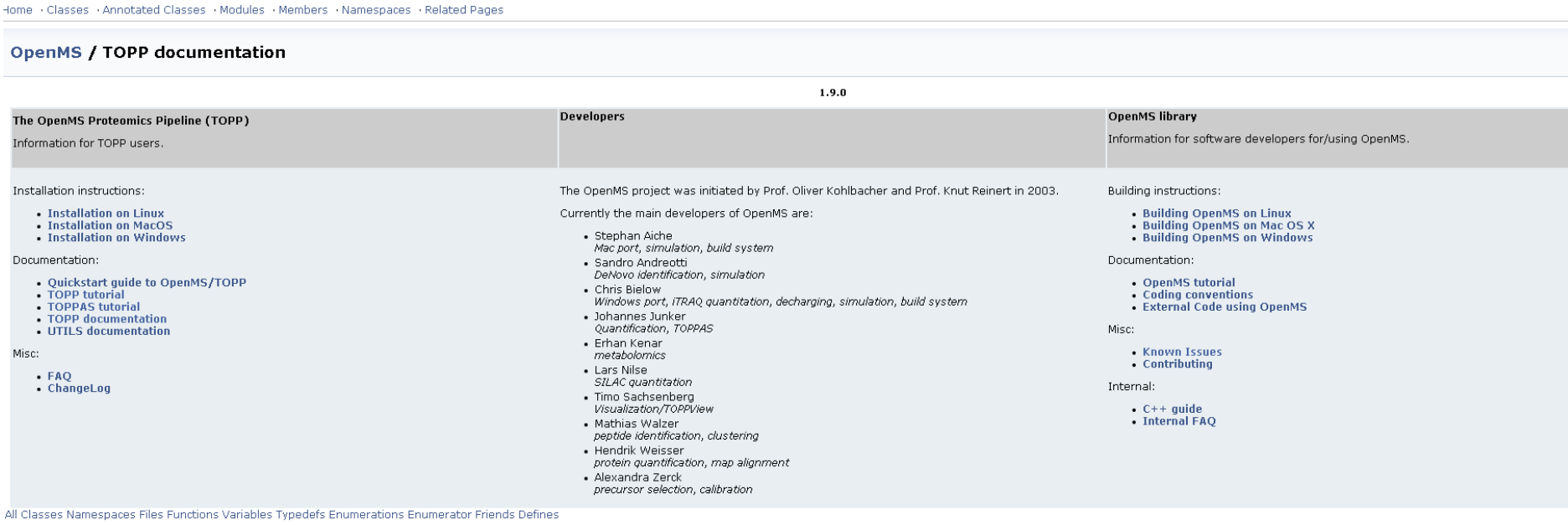
# Important Links

- http://sourceforge.net/apps/trac/open-ms/wiki



# Important Links

- http://ftp.mi.fu-berlin.de/OpenMS/release-documentation



Home · Classes · Annotated Classes · Modules · Members · Namespaces · Related Pages

**OpenMS / TOPP documentation**

1.9.0

| **The OpenMS Proteomics Pipeline (TOPP)**<br>Information for TOPP users. | **Developers** | **OpenMS library**<br>Information for software developers for/using OpenMS. |
|---|---|---|
| Installation instructions:<br>• Installation on Linux<br>• Installation on MacOS<br>• Installation on Windows<br>Documentation:<br>• Quickstart guide to OpenMS/TOPP<br>• TOPP tutorial<br>• TOPPAS tutorial<br>• TOPP documentation<br>• UTILS documentation<br>Misc:<br>• FAQ<br>• ChangeLog | The OpenMS project was initiated by Prof. Oliver Kohlbacher and Prof. Knut Reinert in 2003.<br>Currently the main developers of OpenMS are:<br>• Stephan Aiche<br>*Mac port, simulation, build system*<br>• Sandro Andreotti<br>*DeNovo identification, simulation*<br>• Chris Bielow<br>*Windows port, iTRAQ quantitation, decharging, simulation, build system*<br>• Johannes Junker<br>*Quantification, TOPPAS*<br>• Erhan Kenar<br>*metabolomics*<br>• Lars Nilse<br>*SILAC quantitation*<br>• Timo Sachsenberg<br>*Visualization/TOPPView*<br>• Mathias Walzer<br>*peptide identification, clustering*<br>• Hendrik Weisser<br>*protein quantification, map alignment*<br>• Alexandra Zerck<br>*precursor selection, calibration* | Building instructions:<br>• Building OpenMS on Linux<br>• Building OpenMS on Mac OS X<br>• Building OpenMS on Windows<br>Documentation:<br>• OpenMS tutorial<br>• Coding conventions<br>• External Code using OpenMS<br>Misc:<br>• Known Issues<br>• Contributing<br>Internal:<br>• C++ guide<br>• Internal FAQ |

All Classes Namespaces Files Functions Variables Typedefs Enumerations Enumerator Friends Defines

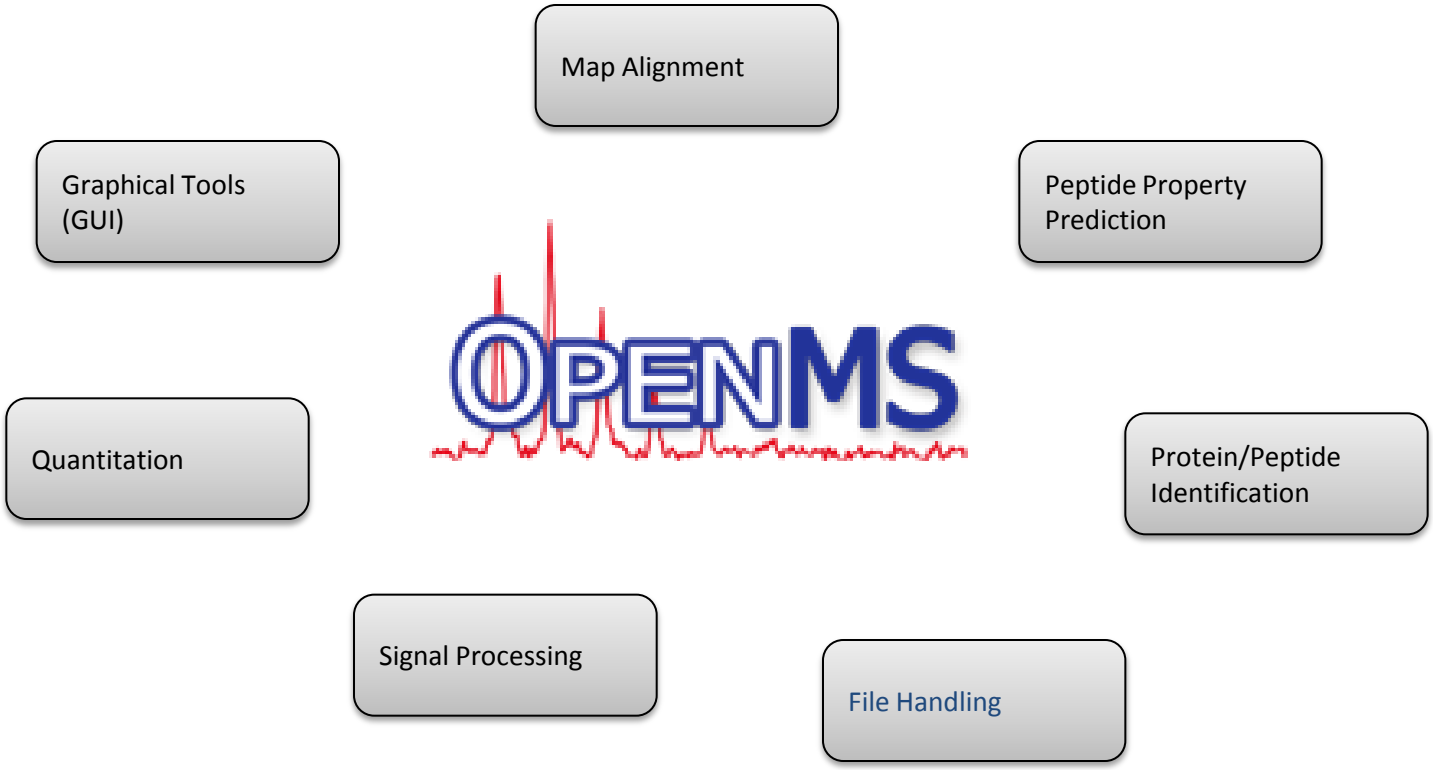
# User Tutorial

- Intro Talk
  - SimpleWorkflowsWithTOPPAS
  - **BasicIdentification**
- Quantitation
  - **LabelFree**
  - ITRAQ
  - SILAC

# Dev Tutorial

- Library Structure (svn)
- Coding Convention (slides)
- CMake (slides)
- FAQ
- Hands-on …

# Library Structure



# Dev Tutorial

- Tutorial_FileIO
  - Store the file in mzData, mzML and *MascotGenericFile* format
  - Compare the filesize; which one is the smallest?

# Dev Tutorial

- Tutorial_MSSpectrum
  - Templatize *MSSpectrum* using *RichPeak2D*
  - what extra information can you store?
  - If you annotate a user parameter (lets call it FWHM) and store the spectrum to disk (mzML). Once you load it back, is the information still there?

# Dev Tutorial

- Tutorial_SavitzkyGolayFilter
  - There is another smoothing filter in OpenMS: *GaussFilter*; use it to replace the SG filter
  - how can you find out what parameters are available for *GaussFilter* (name at least two)
  - Write the parameters to an INI file (see *Param* documentation) & view them using INIFileEditor
  - Compare two different parameter settings of *GaussFilter* & SG by storing the result as mzML file and viewing it using TOPPView

# Dev Tutorial

- Tutorial_RangeManager
  - What happens if you add another feature which extends the current range and call .getMin() or getMax() ?

# Dev Tutorial

- Tutorial_Unlabeled
  - Which other FeatureLinking algorithm is available for label-free data?
  - Is there a difference in results? (compare in terms of ConsensusFeatures and their size; hint: the FileInfo tool might be helpful).

# Dev Tutorial

- Tutorial_MapAlignment
  - Store the transformations (TransformationXMLFile)
  - Can you change the reference map? What happens to the transformations if you use the second map?
  - Use another map alignment algorithm. Can you use the „identification“ version? How does the result differ (see transformation).

# Dev Tutorial

- Set up an external project (see ***External Code using OpenMS*** in the OpenMS documentation).
  - Load an arbitrary consensusXML file and print the number of consensus features to std::out

# MapAlignment

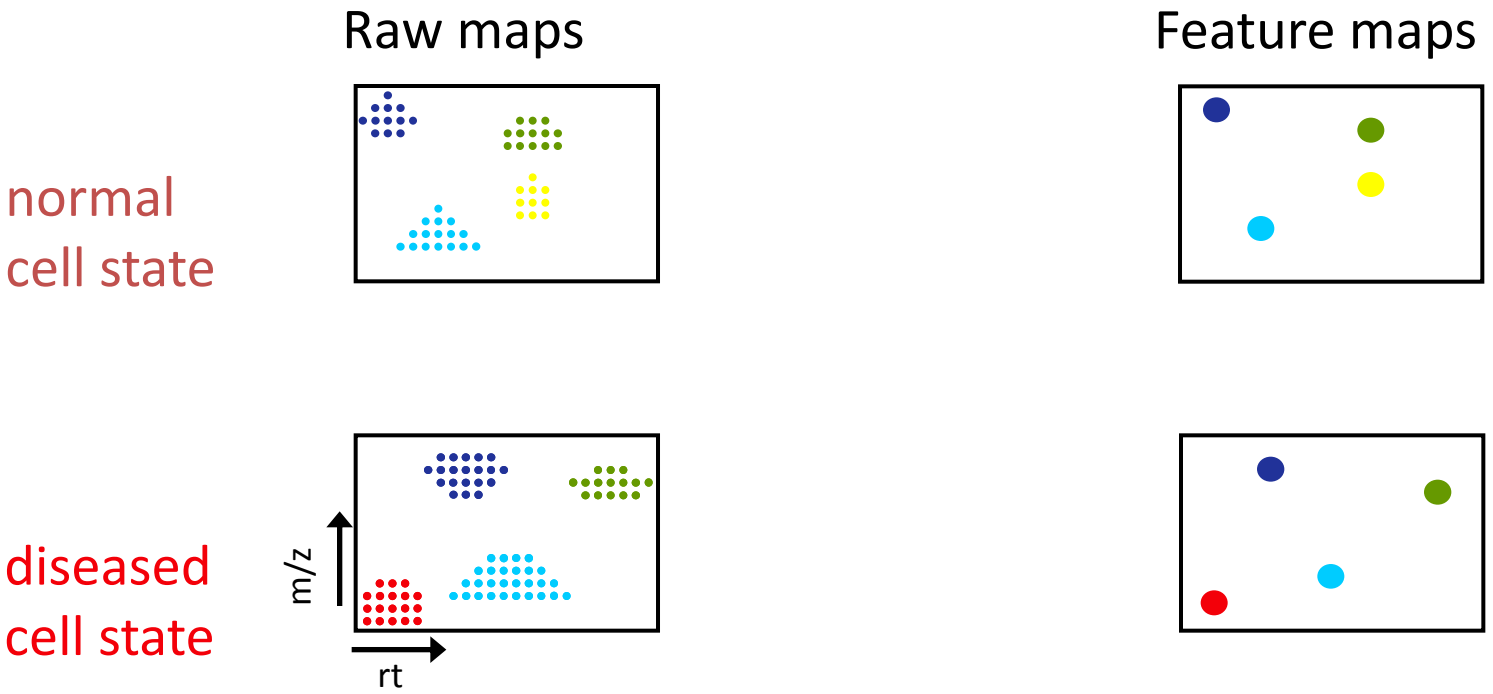
- PoseClustering aka GeometricHashing

Your task:

+ speed

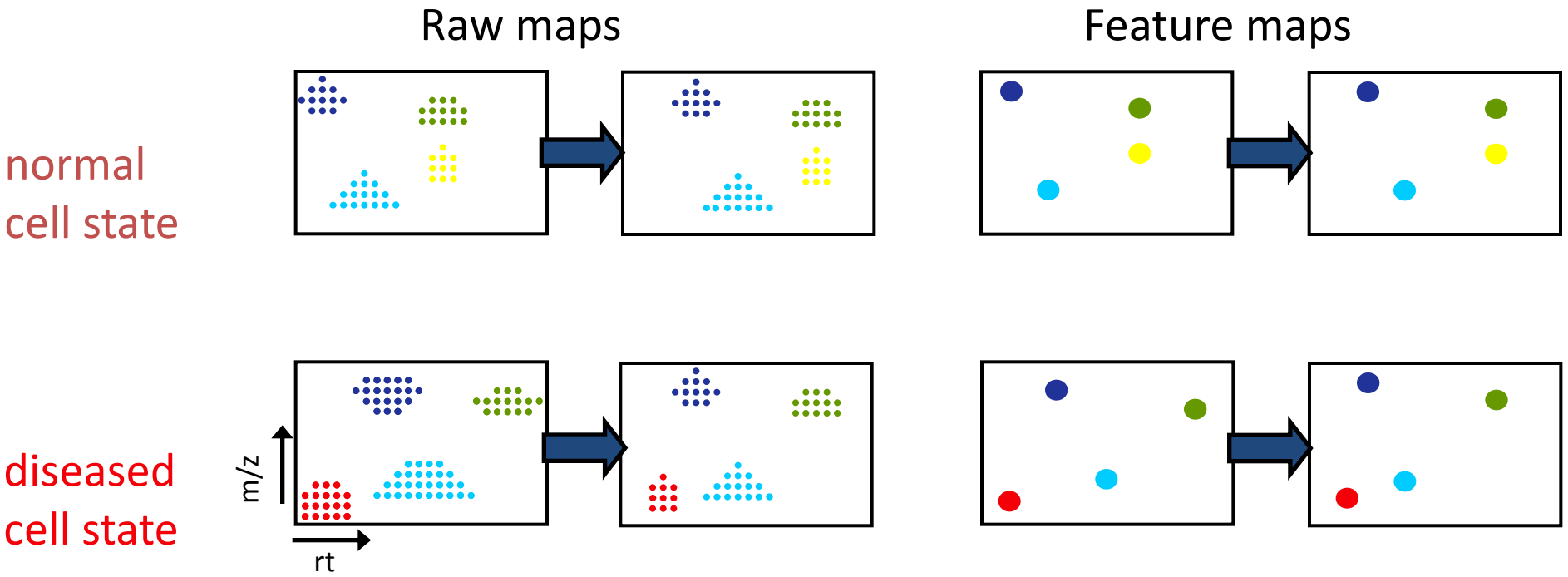
- memory

# Direct differential quantitation




- Both maps are used for differential quantitation

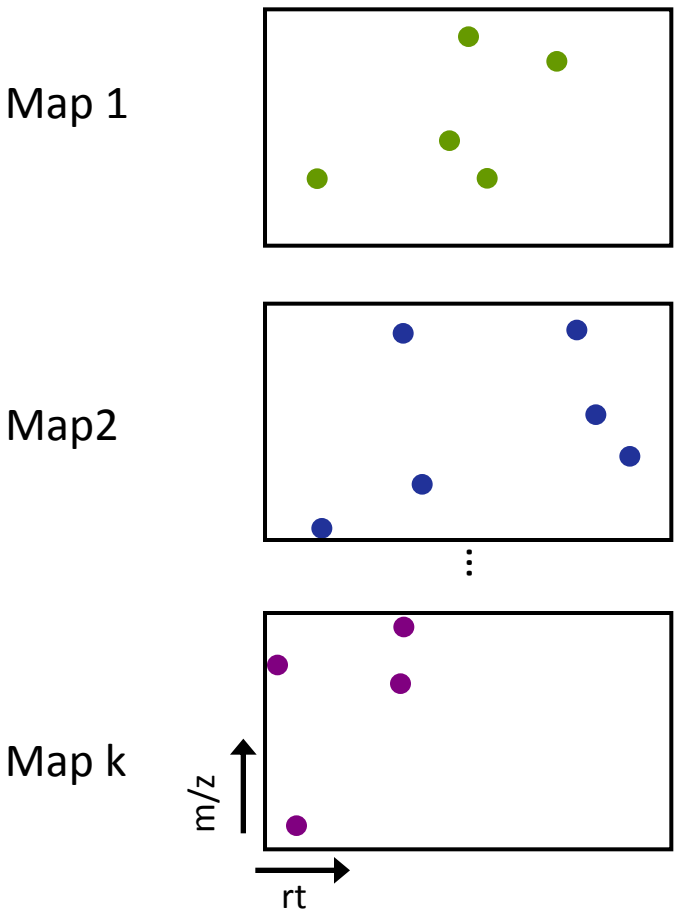
# Direct differential quantitation




- Both maps are used for differential quantitation
- Map transformation necessary beforehand

# Direct differential quantitation

**Given k feature maps**

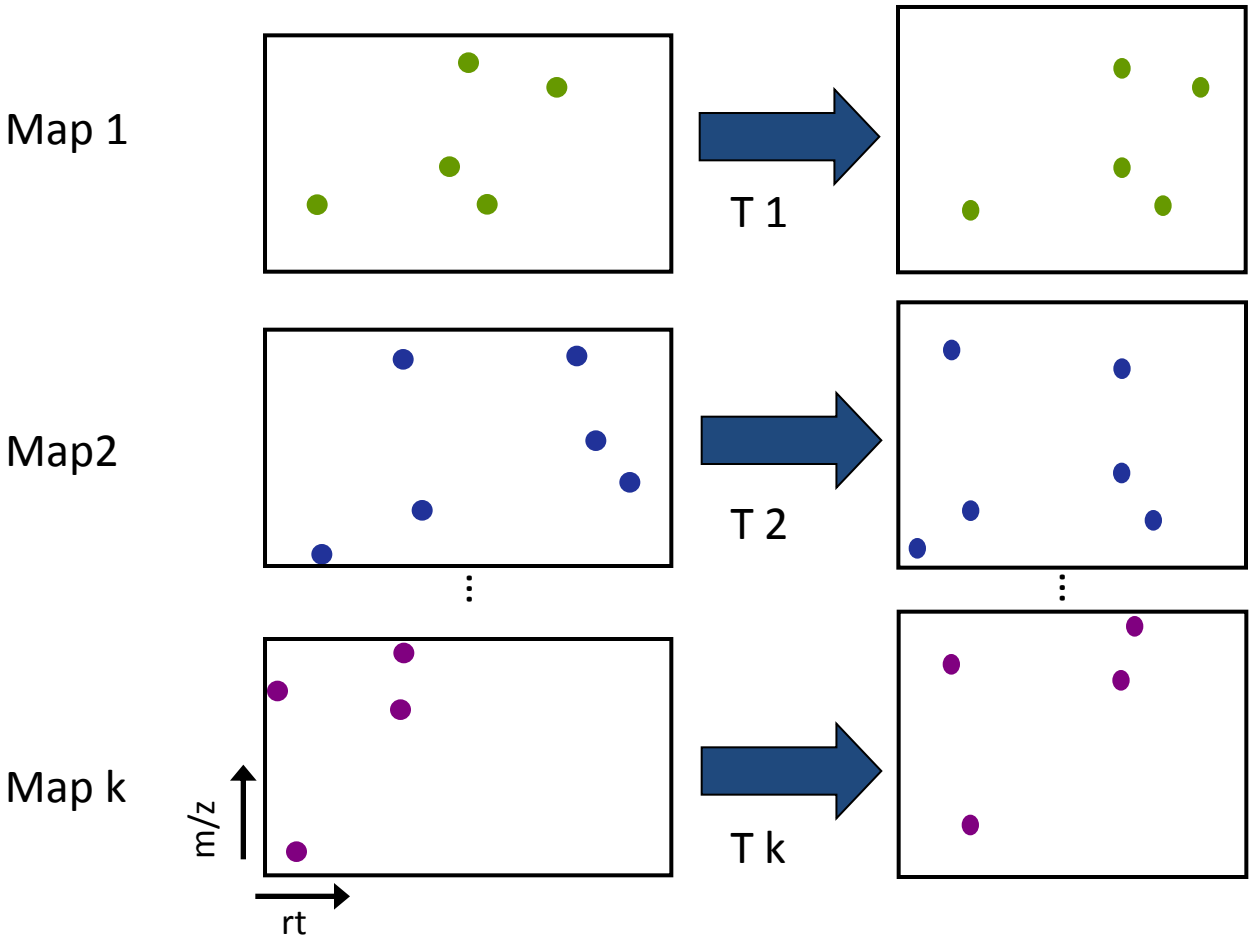



RT and m/z of a peptide vary between maps

$\Rightarrow$ **compute suitable mapping**

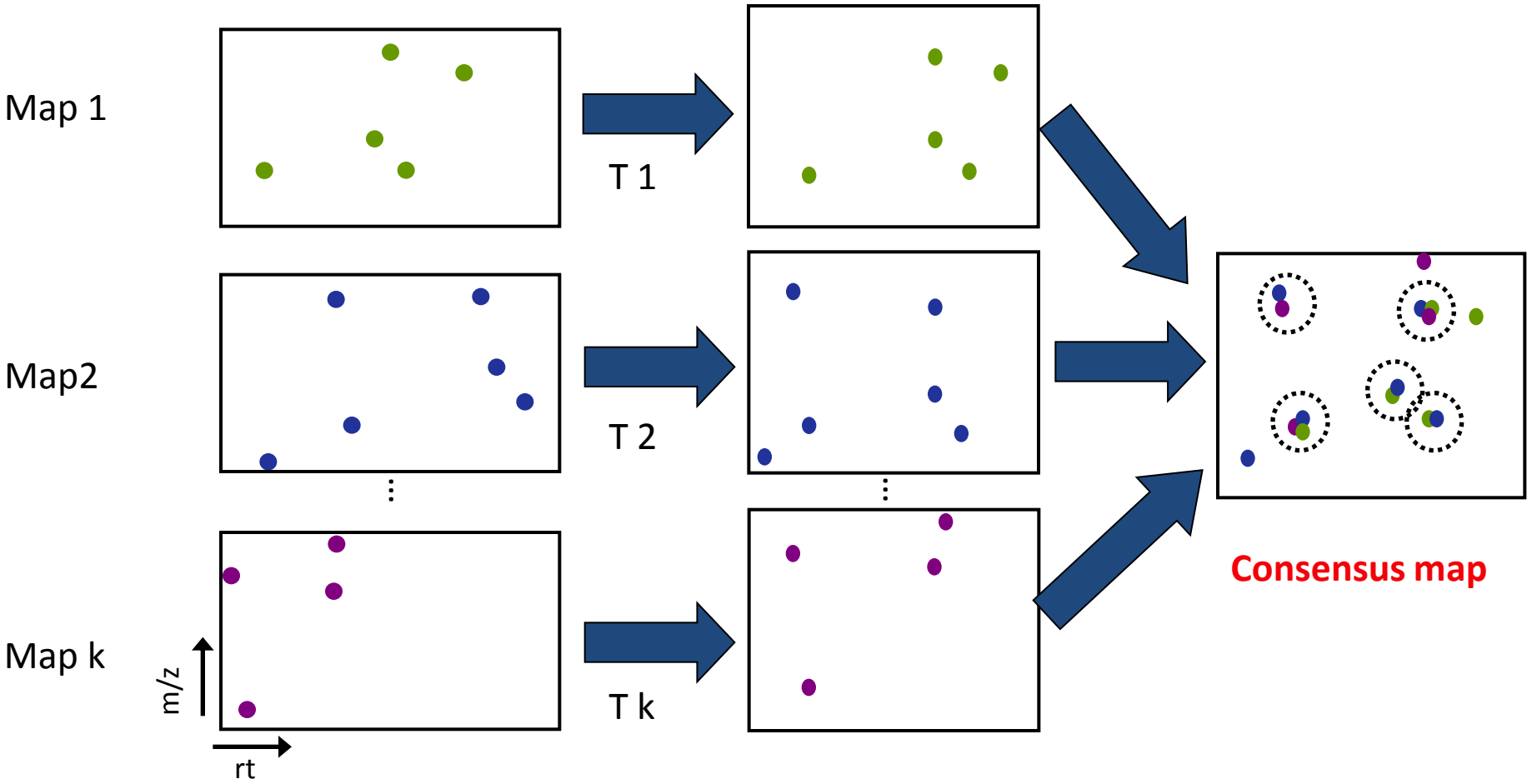
# Direct differential quantitation

**Bring the k maps onto a comparable coordinate system**



# Multiple feature map alignment

**Assign corresponding elements across k maps**



# Multiple map alignment

- **Main objectives:**
- Transform all maps onto comparable rt and m/z dimensions
- Run in real time
- **Compute a consensus map**

- **Main difficulties:**
- Distortion of the rt and m/z dimension
- Fraction of common elements
- **Time order of peptides changes**

# Capabilities

- Generic and modular
  - solves the multiple **raw** and **feature** map alignment problem
- Fast and accurate
- Robust
  - in the presence of noise
  - in dealing with time order changes of peptides

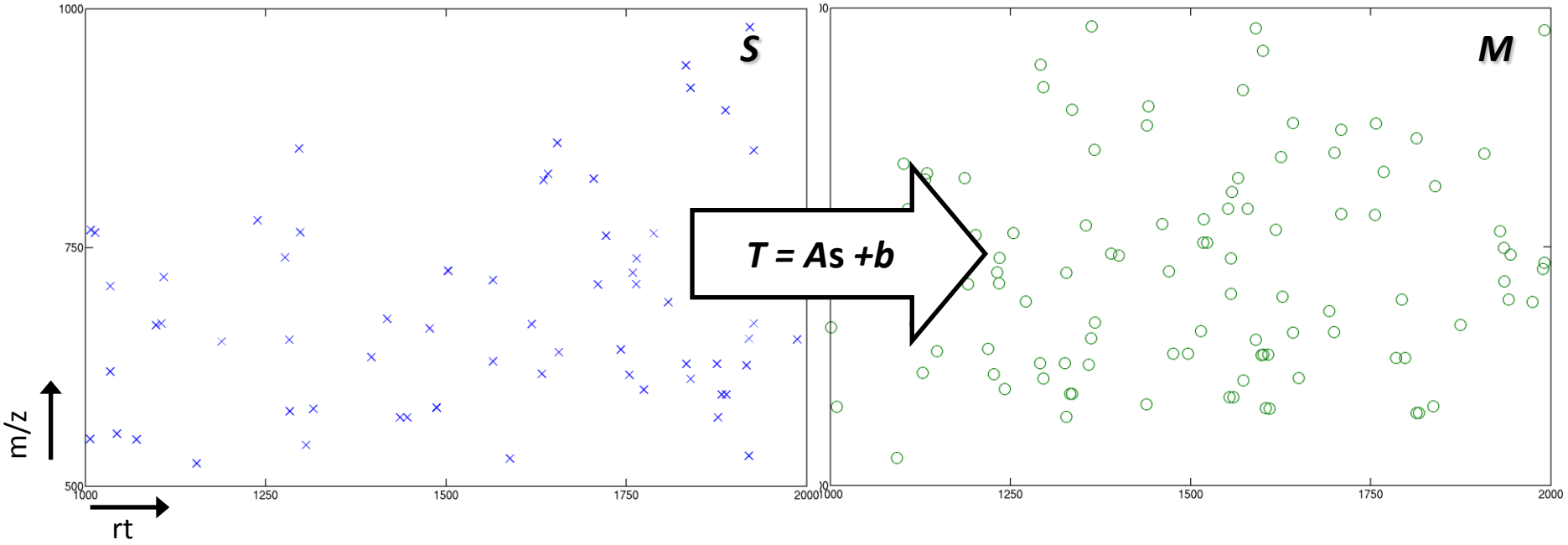
# Two-step algorithm

1. **Superposition phase:**
   transform all maps onto the coordinate system of a reference map

2. **Consensus phase:**
   successive grouping of corresponding elements

   *group elements in the transformed maps, which are nearest neighbors in a weighted Euclidean metric*

# Superposition phase

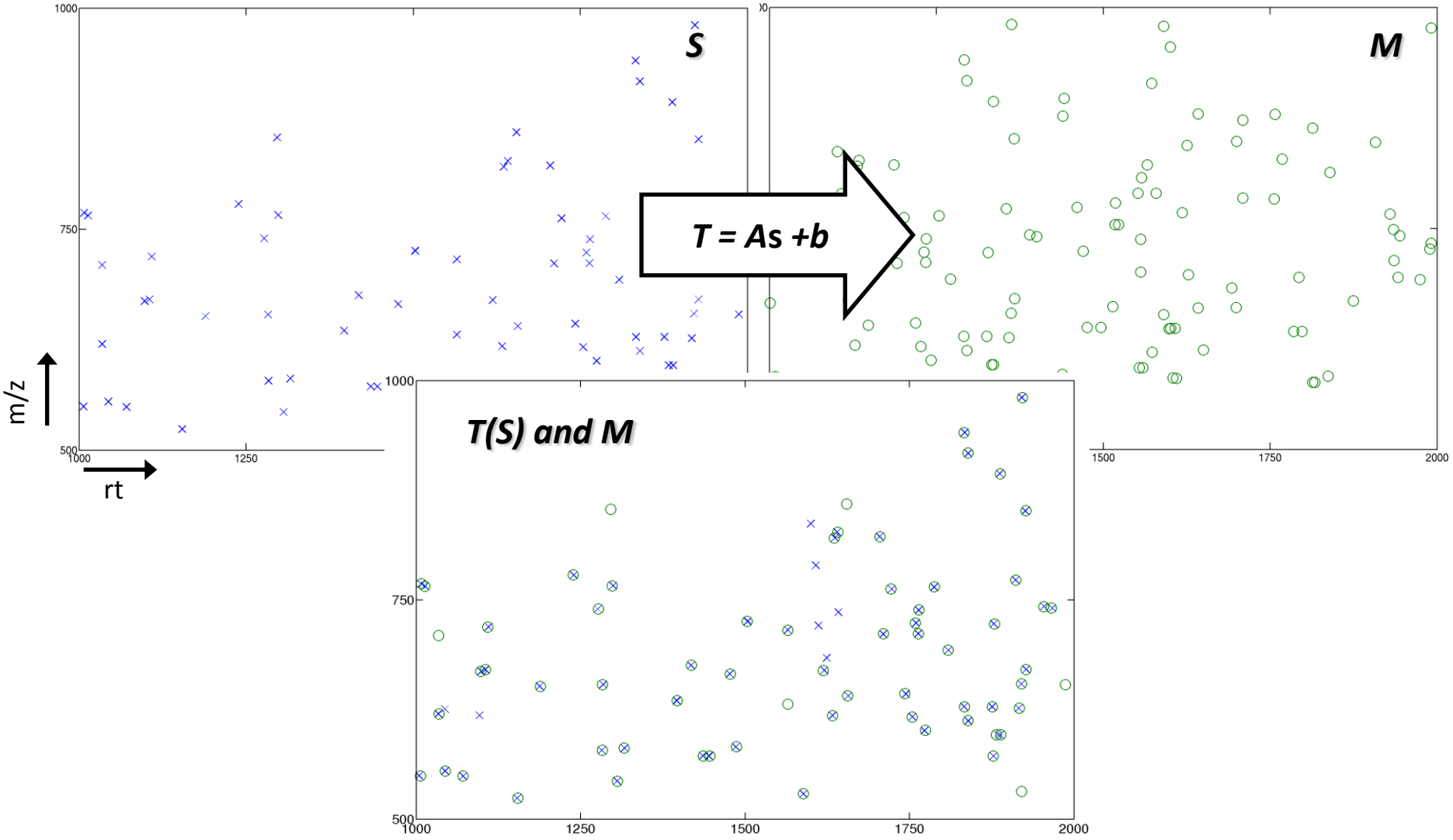



**The problem is to find the**

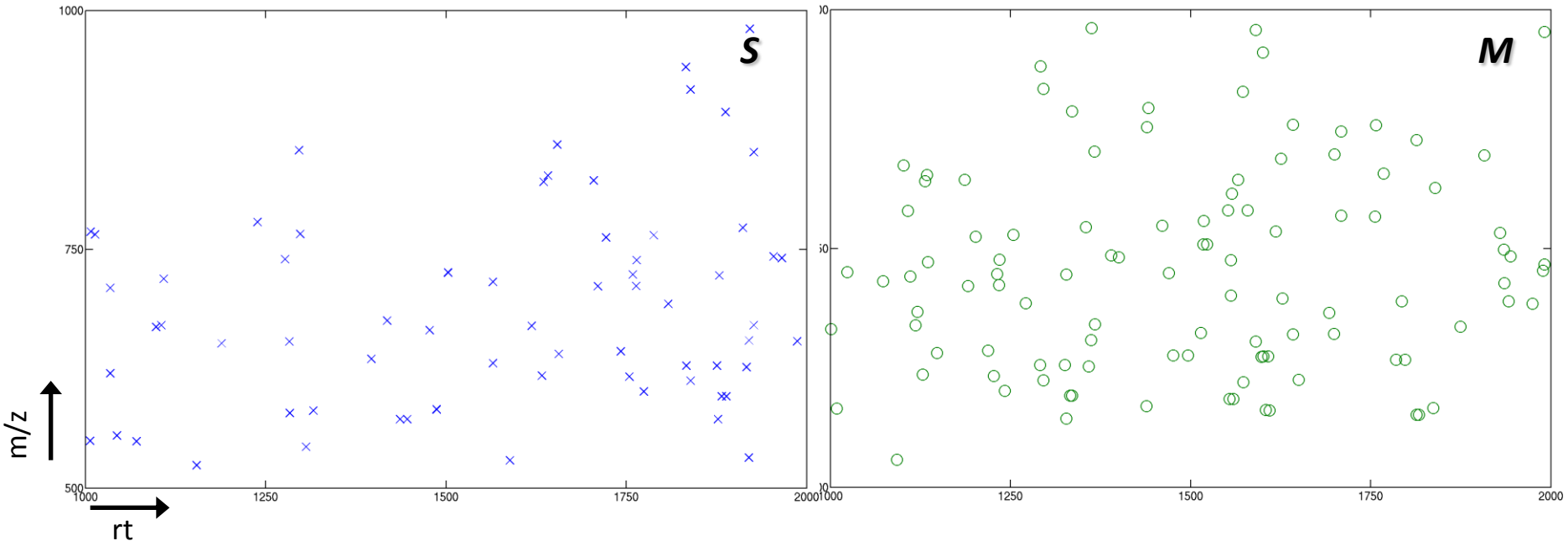
**affine transformation T, that**

**minimizes the distance between T(S) and M.**

# Superposition phase



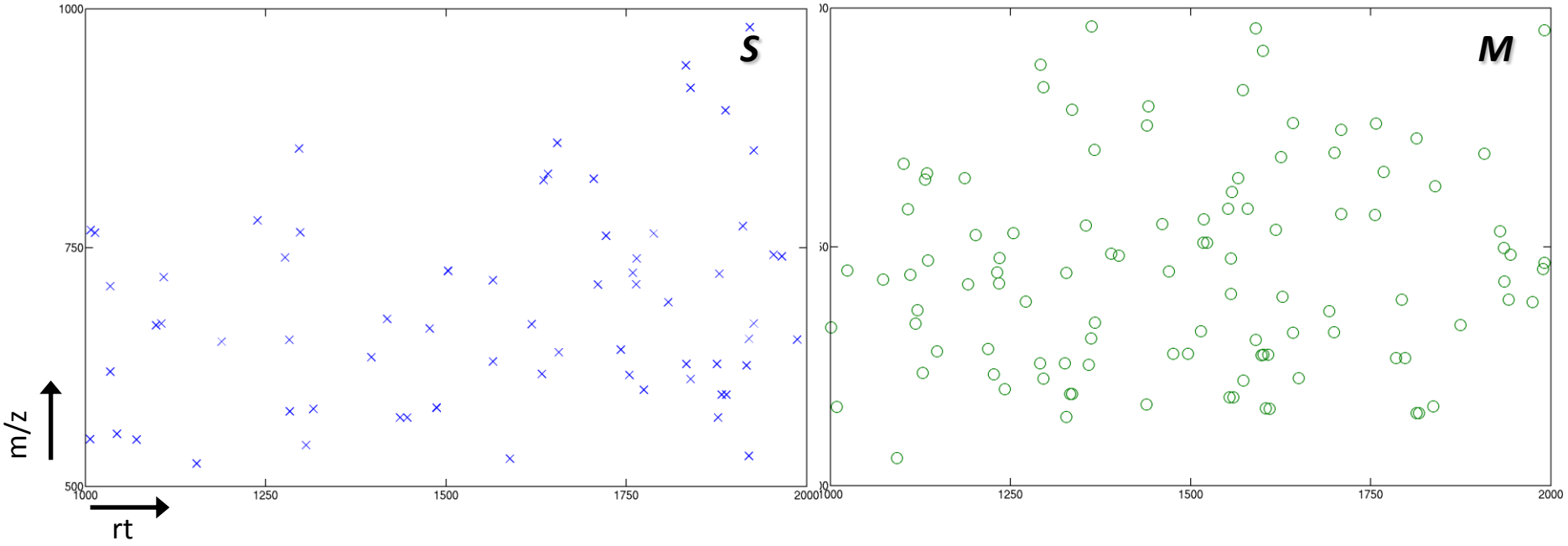
# Pose clustering approach




$$T_{rt}(s_{rt}) = a_{rt}s_{rt} + b_{rt}$$

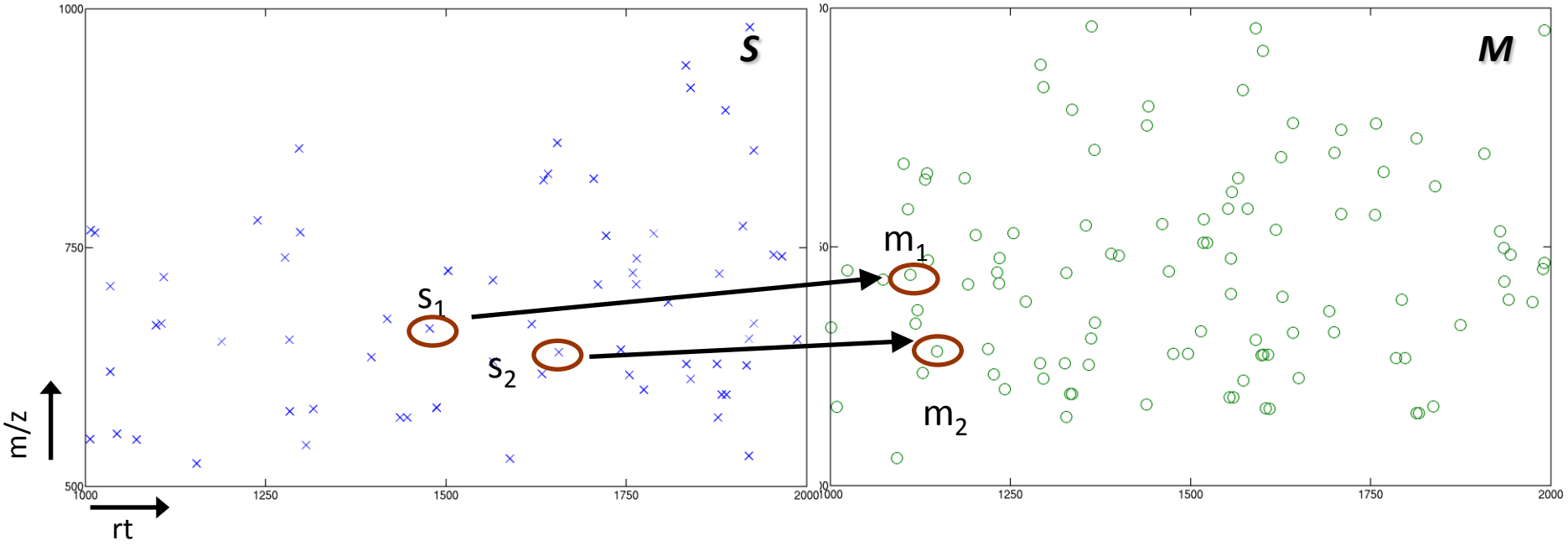
$$T_{m/z}(s_{m/z}) = a_{m/z}s_{m/z} + b_{m/z}$$

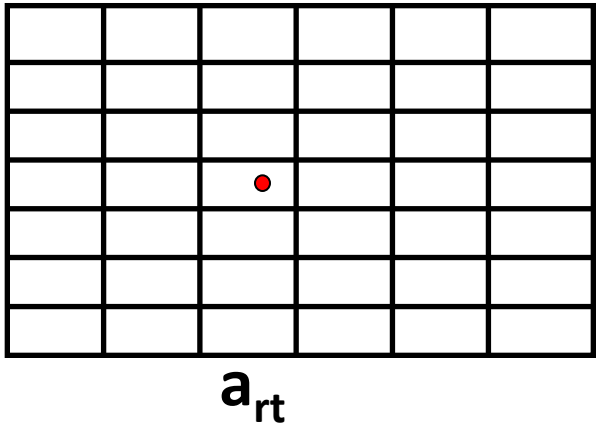
# Pose clustering approach




$$T_{rt}(s_{rt})=a_{rt}s_{rt}+b_{rt}$$

# Pose clustering approach



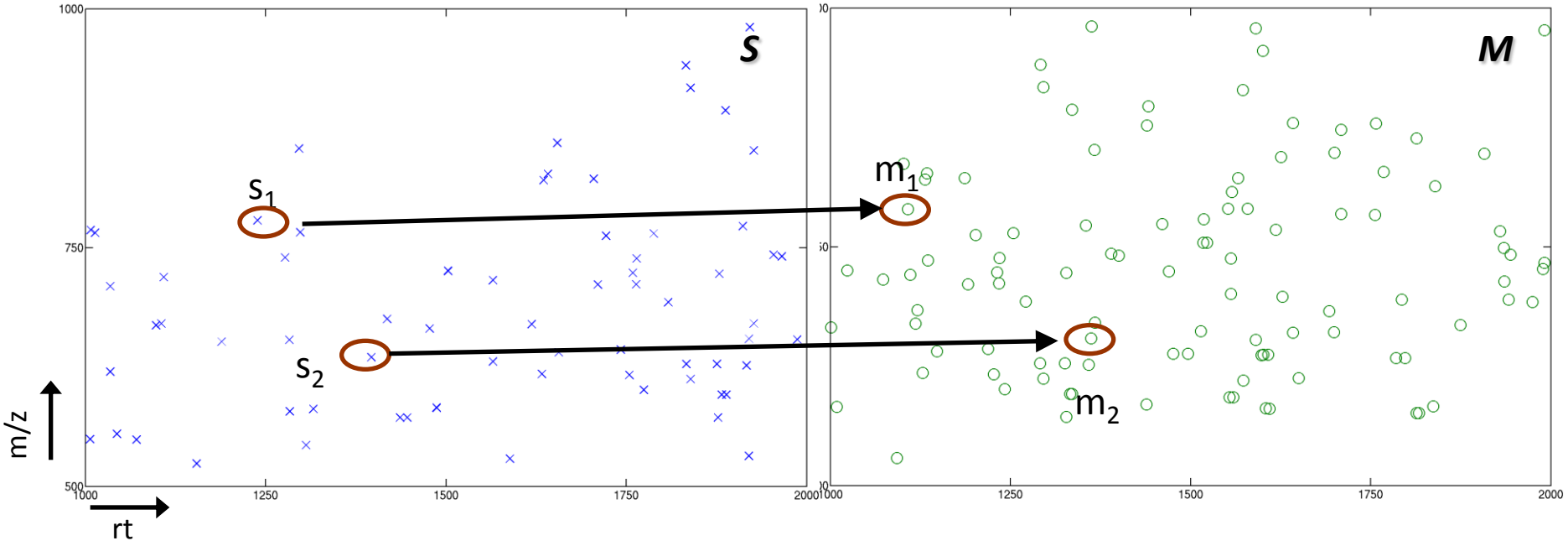
$a_{rt}$

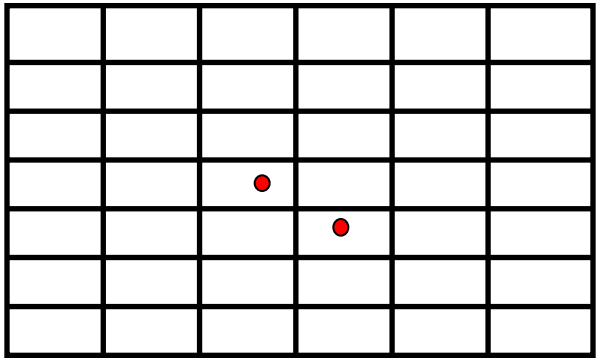
$b_{rt}$

$m_1 = a_{rt} s_1 + b_{rt}$

$m_2 = a_{rt} s_2 + b_{rt}$

# Pose clustering approach



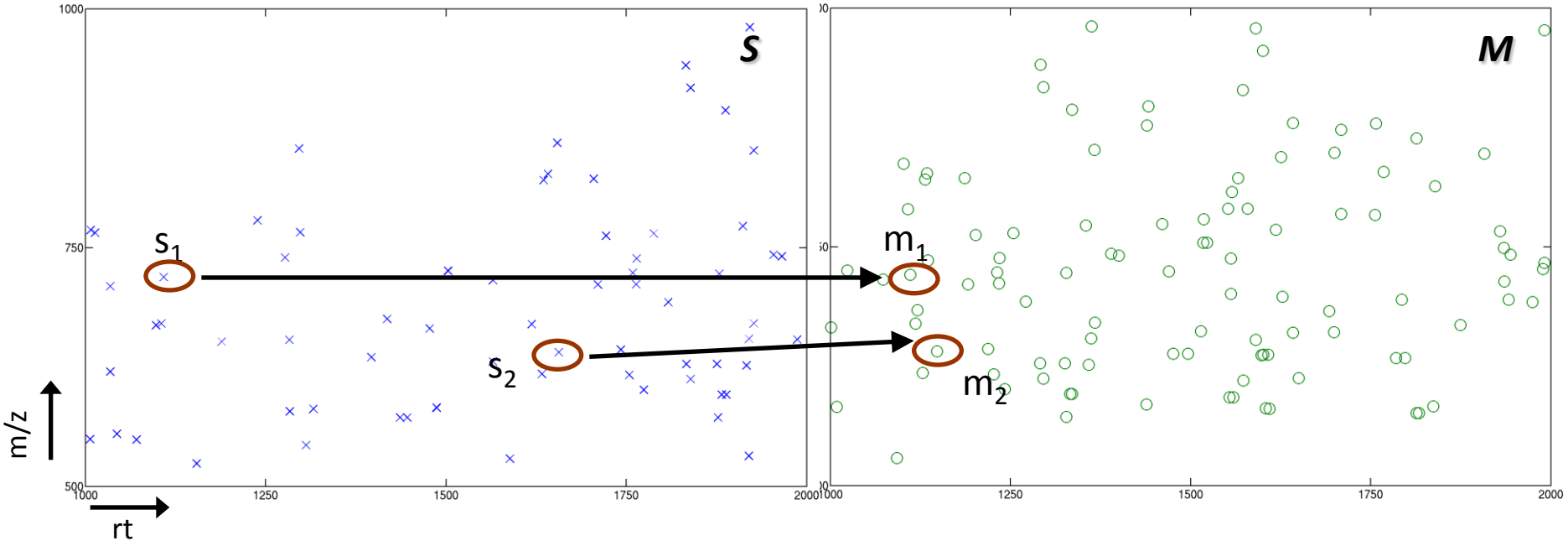
$a_{rt}$

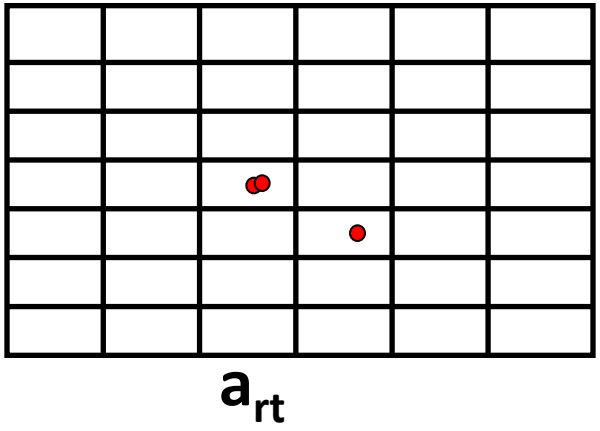
$b_{rt}$

$m_1 = a_{rt} s_1 + b_{rt}$

$m_2 = a_{rt} s_2 + b_{rt}$

# Pose clustering approach



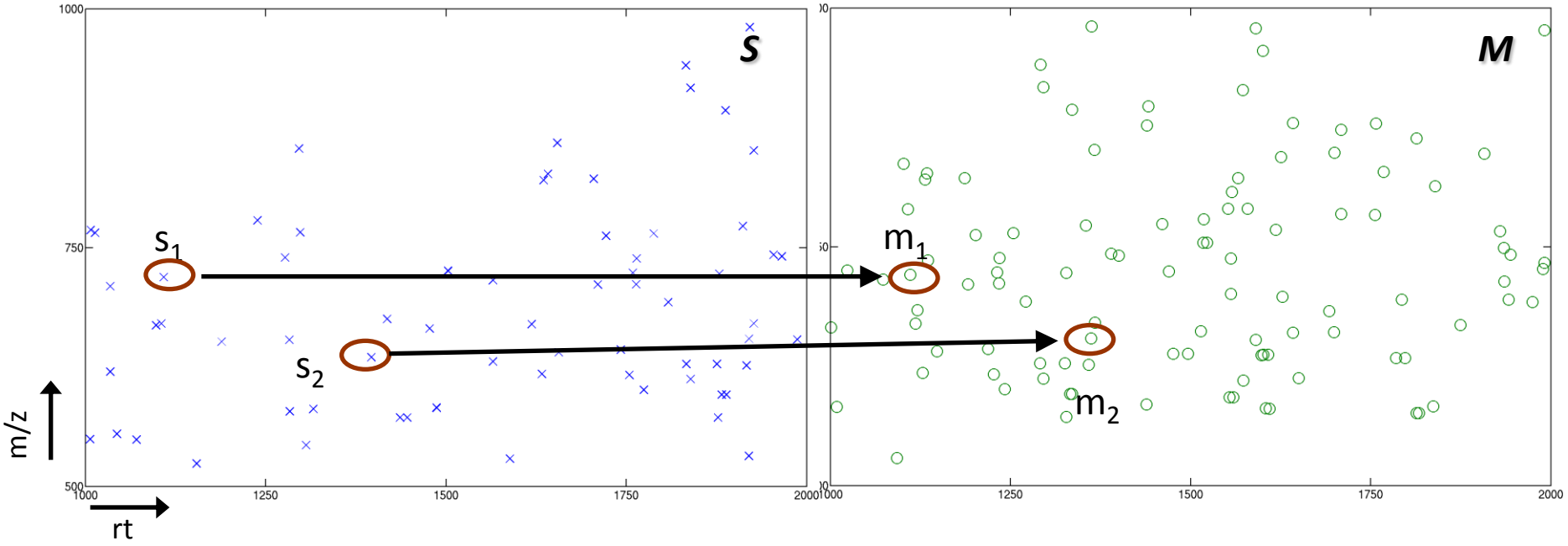
$a_{rt}$

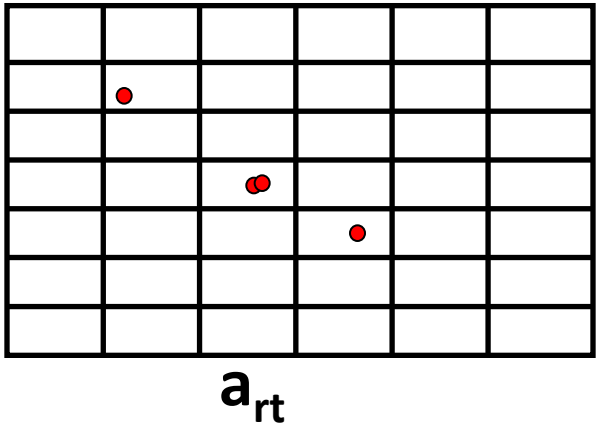
$b_{rt}$

$m_1 = a_{rt} s_1 + b_{rt}$

$m_2 = a_{rt} s_2 + b_{rt}$

# Pose clustering approach



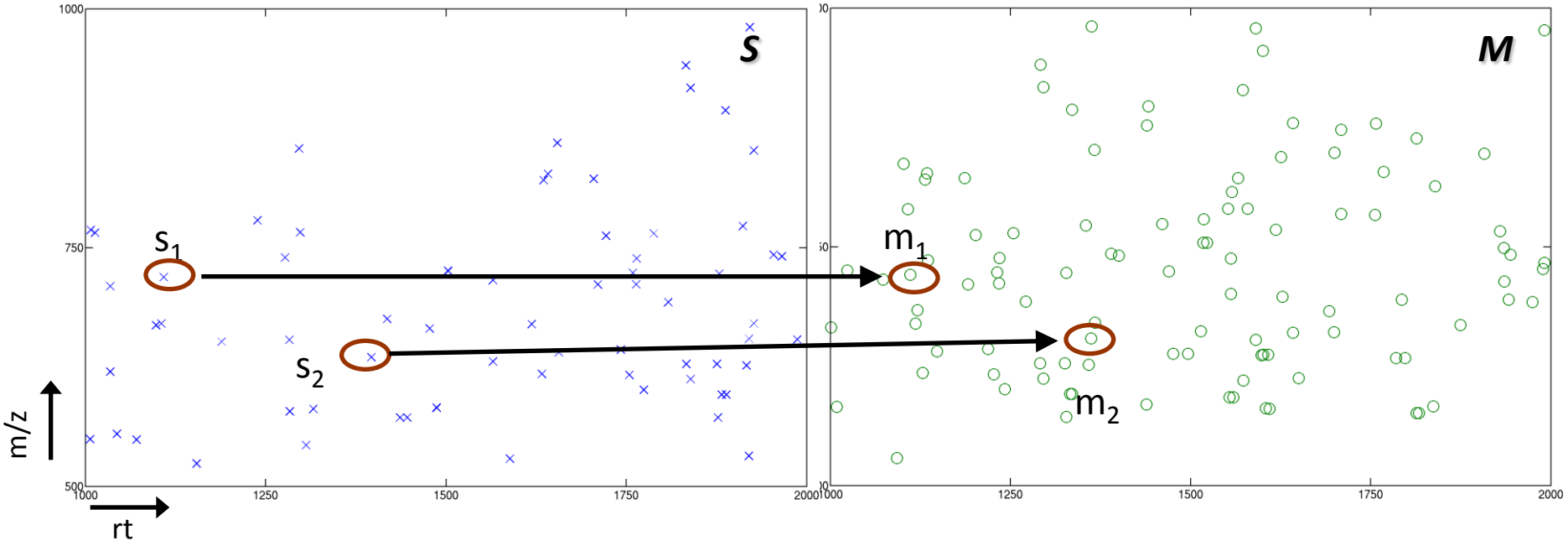
$a_{rt}$

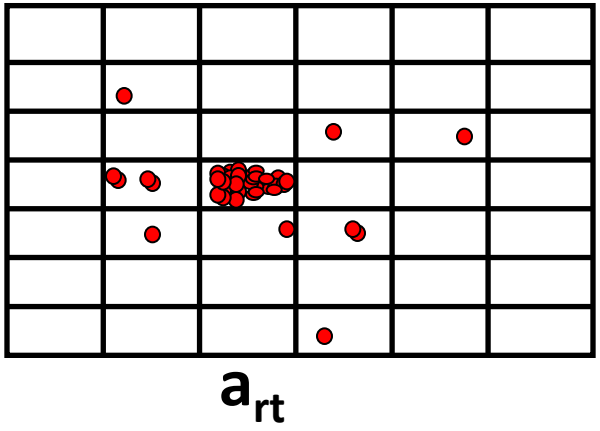
$b_{rt}$

$m_1 = a_{rt} s_1 + b_{rt}$

$m_2 = a_{rt} s_2 + b_{rt}$

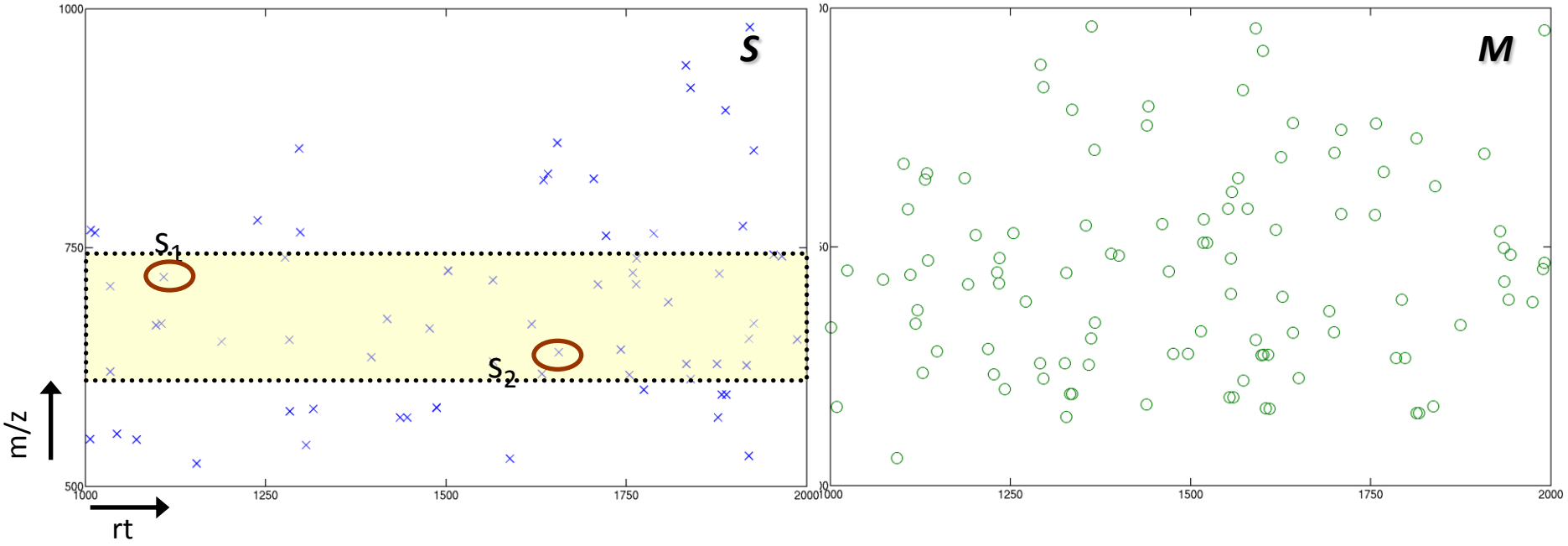
# Pose clustering approach



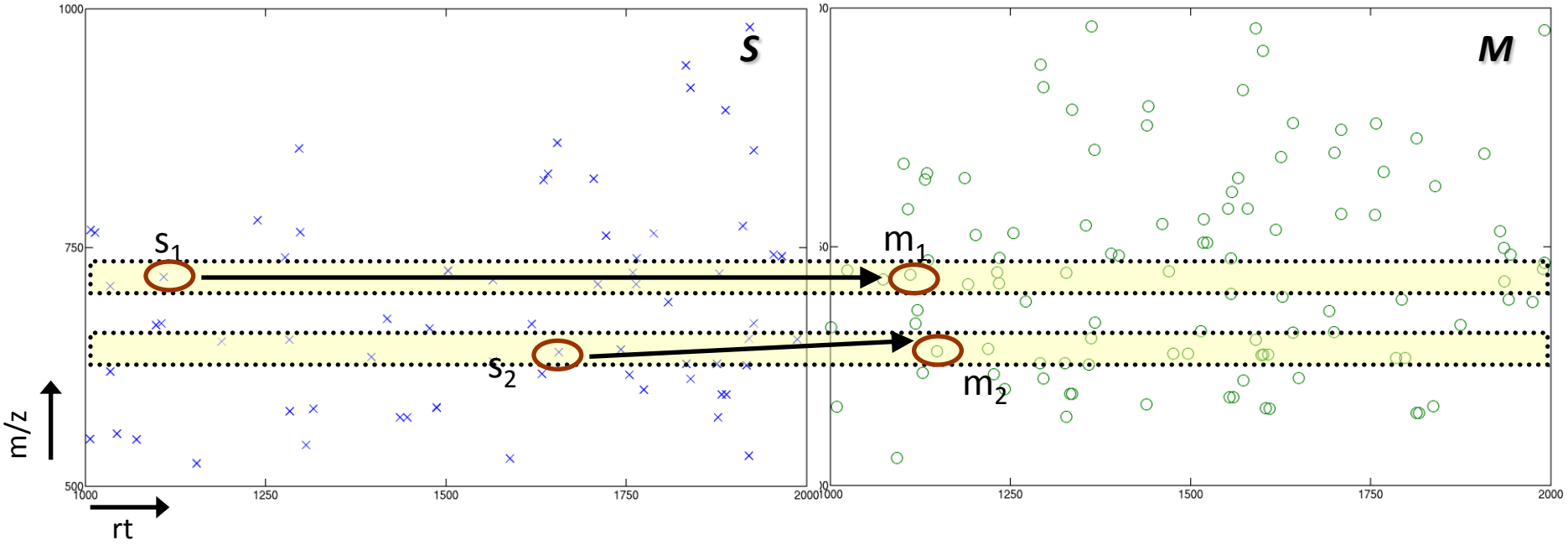

**Matching of corresponding tuples will result in the correct transformation**

# First speed-up



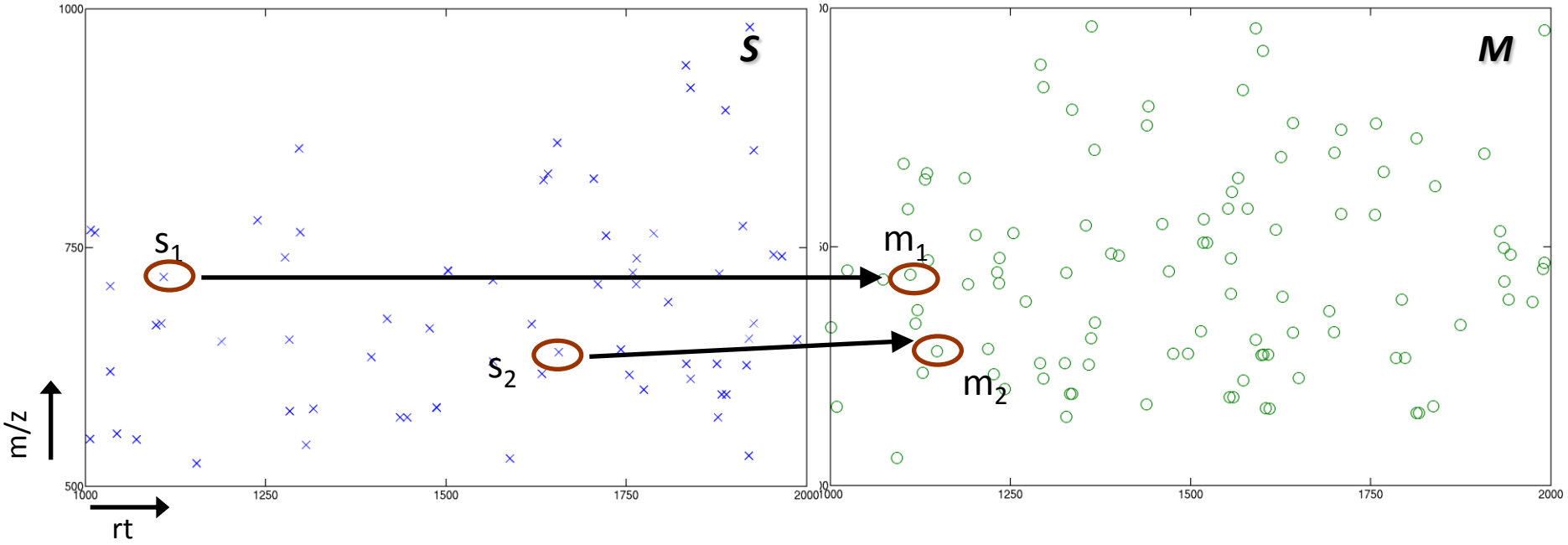

**Only consider tuples ($s_1$,$s_2$) in S with $s_1$ having a small distance to $s_2$ in m/z.**

# Second speed-up



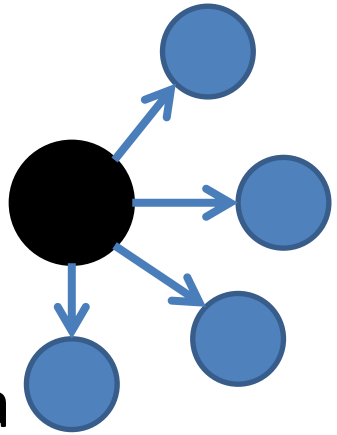

**Only match**
**tuple ($s_1$,$s_2$) onto tuple ($m_1$,$m_2$)**
**if $s_1$ and $m_1$ as well as $s_2$ and $m_2$**
**lie close together in m/z.**

# Improvement




**Normalize intensities in M and S:**
**Weight the vote of each transformation by the intensity similarities of the point matches ($s_1$,$m_1$) and ($s_2$,$m_2$).**

# Space Efficiency



- Current status:
  - star-wise alignment, BUT loading all data (featureXML) at the beginning
- Possible solutions:
  - Lazy loading (new interface)
    - Do NOT break other algorithms!
  - Data stripping: convex hulls, identifications
  - Sparseness: by intensity, quality

# Time Efficiency

- Current status:
  - Sequential reading of full featureXML and sequential alignment
- Possible solutions:
  - Parallelize: OpenMP
  - Internal optimizations (look at extreme pairings?)
  - SAX Parsing: featureXML loading
    - Skip convex hull, subfeatures, …

# OpenMP

```
#pragma omp parallel for
for(int n=0; n<size; ++n)
{
…
}
```

http://bisqwit.iki.fi/story/howto/openmp/#ExampleInitializingATableInParallel

# Xerces SAX Parser

- SAX (Simple API for XML)
- event-based sequential access parser API

```
virtual void endElement(const XMLCh* const uri,
                        const XMLCh* const local_name,
                        const XMLCh* const qname /*=tag*/);

virtual void startElement(const XMLCh* const uri,
                          const XMLCh* const local_name,
                          const XMLCh* const qname, /*=tag*/
                          const xercesc::Attributes& attrib

virtual void characters(const XMLCh* const chars,
                        const XMLSize_t length);
```

```
<html>
 <body style=„font:5px“>
  <H2>Heading</H2>

  some text ….

 </body>
</html>
```

# Xerces SAX Parser

```
template <typename MapType>
void MzMLHandler<MapType>::startElement(…)
{
  String tag = sm_.convert(qname);
  // do nothing until a new spectrum is reached
  if (skip_spectrum_) return;
  else if (tag==„spectrum“)
  {
     if (getAttributeAsInt (attributes, „RT“)<1000)
     {
       skip_spectrum = true;
       return;
     }
  }
  ...
}
```

```
template <typename MapType>
void MzMLHandler<MapType>::endElement(…)
{
  String tag = sm_.convert(qname);
  // leaving <spectrum>; activate parsing again
  if (tag=="spectrum”)
  {
    skip_spectrum = false;
    return;
  }
  ...
}
```