Concurrent Computation of Connected Pattern Spectra for Very Large Information Mining

Michael H. F. Wilkinson\textsuperscript{1}, Ugo Moschini\textsuperscript{1}, Georgios K. Ouzounis\textsuperscript{2}, and Martino Pesaresi\textsuperscript{2}

\textsuperscript{1} Johann Bernoulli Instituut, Rijksuniversiteit Groningen
\textsuperscript{2} Global Security and Crisis Management Unit, Institute for the Security and Protection of the Citizen, Joint Research Centre, European Commission
How connected pattern spectra are useful in image information mining: an example.
How connected pattern spectra are useful in image information mining: an example.

Brief Max-Tree overview.
How connected pattern spectra are useful in image information mining: an example.

Brief Max-Tree overview.

Shared-memory parallel algorithm for computing connected pattern spectra from the Max-Tree structure.
How connected pattern spectra are useful in image information mining: an example.

Brief Max-Tree overview.

Shared-memory parallel algorithm for computing connected pattern spectra from the Max-Tree structure.

Testings and timings on Gpixel input imagery.
Area pattern spectra are traditionally computed by:
Area pattern spectra are traditionally computed by:

- applying a series of ‘openings’ at different scales.
Area pattern spectra are traditionally computed by:

- applying a series of ‘openings’ at different scales.
- taking difference images between those.
Area pattern spectra are traditionally computed by:

- applying a series of ‘openings’ at different scales.
- taking difference images between those.
- summing grey levels of all pixel in the difference image.
Area pattern spectra are traditionally computed by:

- applying a series of ‘openings’ at different scales.
- taking difference images between those.
- summing grey levels of all pixel in the difference image.

They are feature vectors in which each bin tells how much image content lies in a particular size class.
Area pattern spectra are traditionally computed by:

- applying a series of ‘openings’ at different scales.
- taking difference images between those.
- summing grey levels of all pixel in the difference image.

They are feature vectors in which each bin tells how much image content lies in a particular size class.

The goal is to derive the bin entries that describe best the target ROIs.
Key features:
Key features:

- Connected pattern spectra are pattern spectra based on connected attribute filters.
Key features:

- Connected pattern spectra are pattern spectra based on connected attribute filters.
- Connected attribute filters work on connected components of the image.
Key features:

- Connected pattern spectra are pattern spectra based on connected attribute filters.
- Connected attribute filters work on connected components of the image.
- Shape information is added with connected pattern spectra (e.g. compactness, elongation).
Key features:

- Connected pattern spectra are pattern spectra based on connected attribute filters.
- Connected attribute filters work on connected components of the image.
- Shape information is added with connected pattern spectra (e.g. compactness, elongation).
- Multivariate pattern spectra: consider both size and shape information at the same time.
The Sana’a example

Sana’s example shows a pattern spectrum computed using non-compactness as shape information and area.
Pattern spectra can be computed efficiently from hierarchical image representation like Max-Tree.
Pattern spectra can be computed efficiently from hierarchical image representation like Max-Tree.
Max-Tree structure

- Each node represents a *peak component*. 
Each node represents a *peak component*.

An image is a set of nested peak components.
Max-Tree structure

- Each node represents a *peak component*.

- An image is a set of nested peak components.

- Edges are parent-child relationships.
- Each node represents a *peak component*.

- An image is a set of nested peak components.

- Edges are parent-child relationships.

- Every node keeps the attribute values for that connected component.
Filtering rules allow to decide which nodes must preserved and their new grey values.
Filtering rules allow to decide which nodes must be preserved and their new grey values.

Build the tree once, filter many times.
Filtering rules allow to decide which nodes must preserved and their new grey values.

Build the tree once, filter many times.

Univariate or multivariate pattern spectra can be worked out in a single pass.
Filtering rules allow to decide which nodes must preserved and their new grey values.

Build the tree once, filter many times.

Univariate or multivariate pattern spectra can be worked out in a single pass.

*Image sizes are increasing with increasing sensor resolution: a parallel solution is needed.*
Assign disjoint sections of the image to different threads.
Max-Tree Parallelization

- Assign disjoint sections of the image to different threads.

- Build different Max-Trees for every disjoint section.
Max-Tree Parallelization

- Assign disjoint sections of the image to different threads.

- Build different Max-Trees for every disjoint section.

- Stitch them together.
Max-Tree Parallelization

- Assign disjoint sections of the image to different threads.

- Build different Max-Trees for every disjoint section.

- Stitch them together.

The following changes to the structure were made so to give every pixels unique and precise labels, indicating their node-membership:
The following changes to the structure were made so to give every pixels unique and precise labels, indicating their node-membership:

- Each pixel is considered as a node.
The following changes to the structure were made so to give every pixels unique and precise labels, indicating their node-membership:

- Each pixel is considered as a node.

- All members of a component point to a single member of the component called *level root*. 
The following changes to the structure were made so to give every pixels unique and precise labels, indicating their node-membership:

- Each pixel is considered as a node.
- All members of a component point to a single member of the component called \textit{level root}.
- The level root has the pointer to the parent node.
The following changes to the structure were made so to give every pixels unique and precise labels, indicating their node-membership:

- Each pixel is considered as a node.

- All members of a component point to a single member of the component called *level root*.

- The level root has the pointer to the parent node.

- Only the level roots really represent the Max-Tree.
Compute size and non-compactness attributes instead of a filtering phase.
Adapt to pattern spectra

- Compute size and non-compactness attributes instead of a filtering phase.

- Insert the pattern spectrum computation.
Adapt to pattern spectra

- Compute size and non-compactness attributes instead of a filtering phase.

- Insert the pattern spectrum computation.

- Allocate as many private pattern spectra as threads to avoid mutexes mechanisms.
Adapt to pattern spectra

- Compute size and non-compactness attributes instead of a filtering phase.

- Insert the pattern spectrum computation.

- Allocate as many private pattern spectra as threads to avoid mutexes mechanisms.

- A single thread is in charge of summing the partial spectra at the end.
The Algorithm

\textbf{process} \textit{CombinedConstructionAndPatternSpectrum}(p)

\textit{LocalMaxTreeBuild}(V^p) ;

\texttt{var} \ i := 1 \ , \ q := p ;

\textbf{while} \ p + i < K \ \land \ q \ \text{mod} \ 2 = 0 \ \textbf{do}

\hspace{1em} P(sa[p + i]) \ (\ast \text{wait to glue with}
\hspace{1em} \hspace{1em} \hspace{1em} \hspace{1em} \hspace{1em} \hspace{1em} \text{right-hand neighbour} \ast) ;

\hspace{1em} \textit{FuseTrees}(p, (p + i)) ;

\hspace{1em} i := 2 \ast i \ ; \ q := q/2 ;

\textbf{end} ;

\textbf{if} \ p \neq 0 \ \textbf{then}

\hspace{1em} V(sa[p]) \ (\ast \text{signal left-hand neighbour} \ast) ;

\textbf{end} ;

\textit{Barrier}(p) \ (\ast \text{use Barrier synchronisation type} \ast) ;

\textit{MaxTreePatSpectrum2D}(p, \text{nodes}, \text{threadPatSpec}[p]) ;

\textit{Barrier}(p) \ (\ast \text{use Barrier synchronisation type} \ast) ;

\textbf{if} \ p = 0 \ \textbf{then}

\hspace{1em} \textbf{for} \ i := 1 \ \textbf{to} \ K - 1 \ \textbf{do}

\hspace{2em} \textit{SumPatSpecs}(0, i) ;

\hspace{1em} \textbf{end} ;

\textbf{end} \ \textit{CombinedConstructionAndPatternSpectrum} .
Tests

Algorithm

- Implemented in C.

- Tested on a Dell R815, four 16-core AMD Opteron, 512 GB RAM.

Images tested

- ~1 and 1.2 Gpixel Sana’a and Port-au-Prince images.

- 1.2 Gpixel astronomical images.
Results: Total Speed

![Graph showing the relationship between number of threads and speed (10^6 pixels s^-1)]

- X-axis: Number of Threads
- Y-axis: Speed (10^6 pixels s^-1)
Results: Speed Up

Number of Threads vs. Speed Up
Wall clock times reduce from 272 - 447s to 8.13 - 15s.
Wall clock times reduce from 272 - 447s to 8.13 - 15s.

80 - 90% of the time is for building the Max Tree on 64 threads.
Results: Summary

- Wall clock times reduce from 272 - 447s to 8.13 - 15s.

- 80 - 90% of the time is for building the Max Tree on 64 threads.

- Speed increases rapidly between 1 and 32 threads.
Wall clock times reduce from 272 - 447s to 8.13 - 15s.

80 - 90% of the time is for building the Max Tree on 64 threads.

Speed increases rapidly between 1 and 32 threads.

Good efficiency up to about 32 threads.
Results: Summary

- Wall clock times reduce from 272 - 447s to 8.13 - 15s.
- 80 - 90% of the time is for building the Max Tree on 64 threads.
- Speed increases rapidly between 1 and 32 threads.
- Good efficiency up to about 32 threads.
- 32 threads drop: presence of only 32 FPU is not the reason.
Conclusions

Processing times from several minutes to 8 - 15s for Gpixel images.
Conclusions

- Processing times from several minutes to 8 - 15s for Gpixel images.
- Pattern spectrum can be recalculated interactively in 3s.
Conclusions

- Processing times from several minutes to 8 - 15s for Gpixel images.

- Pattern spectrum can be recalculated interactively in 3s.

- Change of speed up beyond 32 threads: perhaps due to costly attributes. Optimization might help.
Conclusions

- Processing times from several minutes to 8 - 15s for Gpixel images.

- Pattern spectrum can be recalculated interactively in 3s.

- Change of speed up beyond 32 threads: perhaps due to costly attributes. Optimization might help.

- High memory usage: ~90GB. Scales linearly with the increase of bits per pixels.
Conclusions

- Processing times from several minutes to 8 - 15s for Gpixel images.

- Pattern spectrum can be recalculated interactively in 3s.

- Change of speed up beyond 32 threads: perhaps due to costly attributes. Optimization might help.

- High memory usage: ~90GB. Scales linearly with the increase of bits per pixels.

- Develop code for distributed memory machines to deal with tera and petapixel images.
Questions