Thinking Beyond the Block

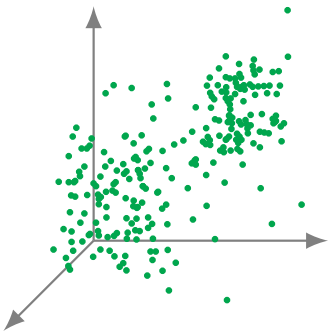
Block Matching for Copy–Move Forgery Detection Revisited

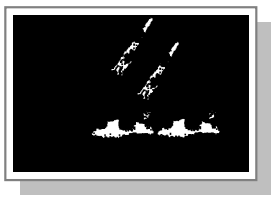
Matthias Kirchner — Binghamton University
Pascal Schöttle — University of Münster
Christian Riess — Stanford University

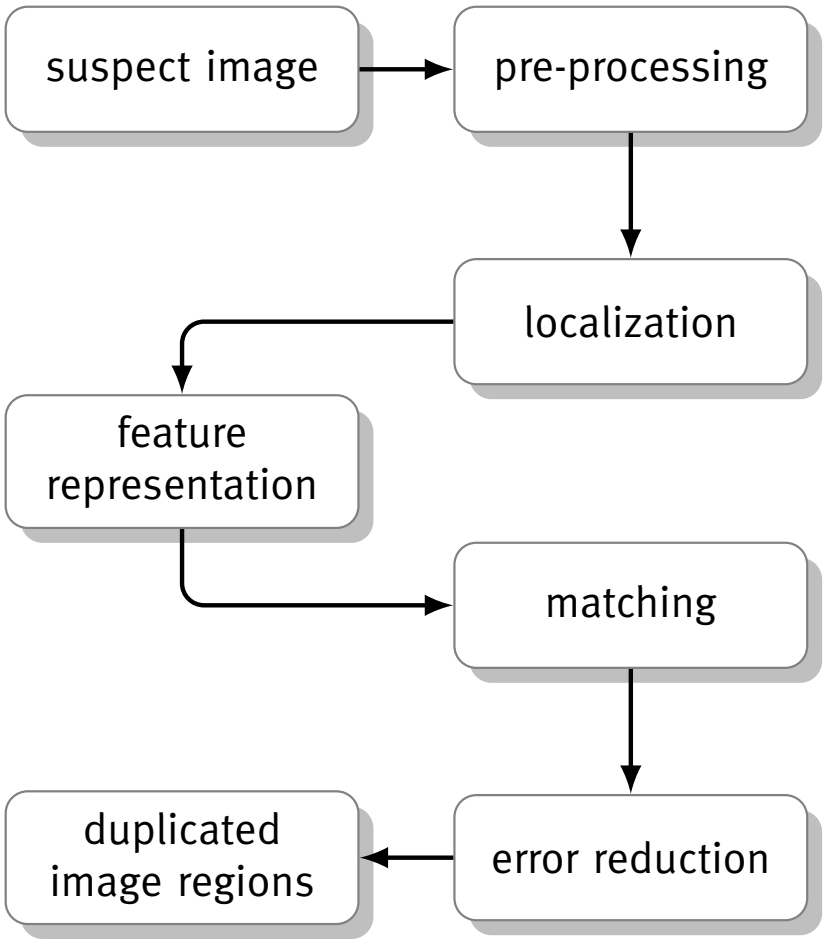
IS&T/SPIE Media Watermarking, Security, and Forensics
San Francisco, CA | February 9, 2015

Yet Another Paper CMFD Paper?

- general detection/localization procedure since 2003 [Fridrich et al. 2003]





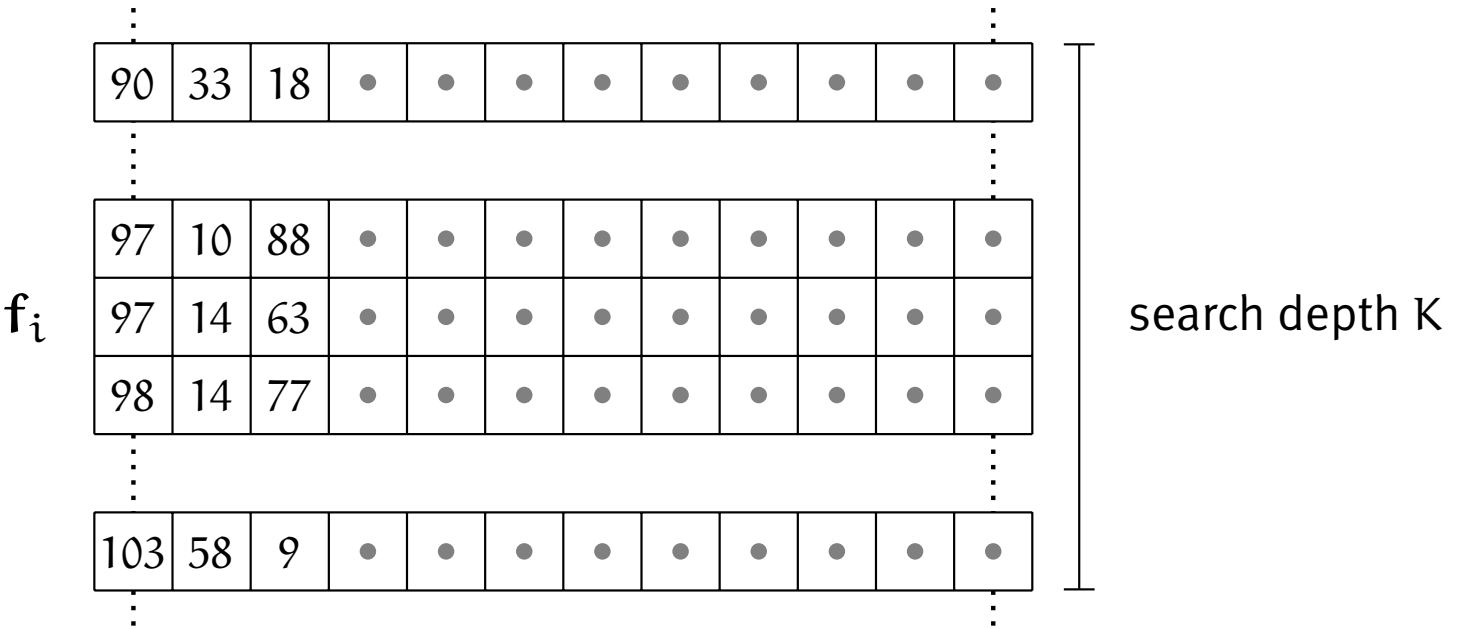


- color space transforms
- saturated regions
- overlapping blocks
- key points (SIFT, ...)
- find similar blocks / key points in the feature space
- remove false positives

Lexicographic Matching

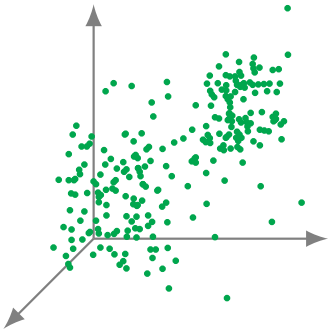
- lexicographic sorting of feature vectors
- compare i-th sorted vector to K neighbors

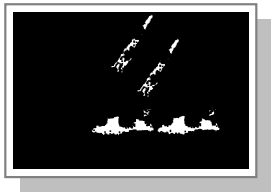
[Fridrich et al. 2003]

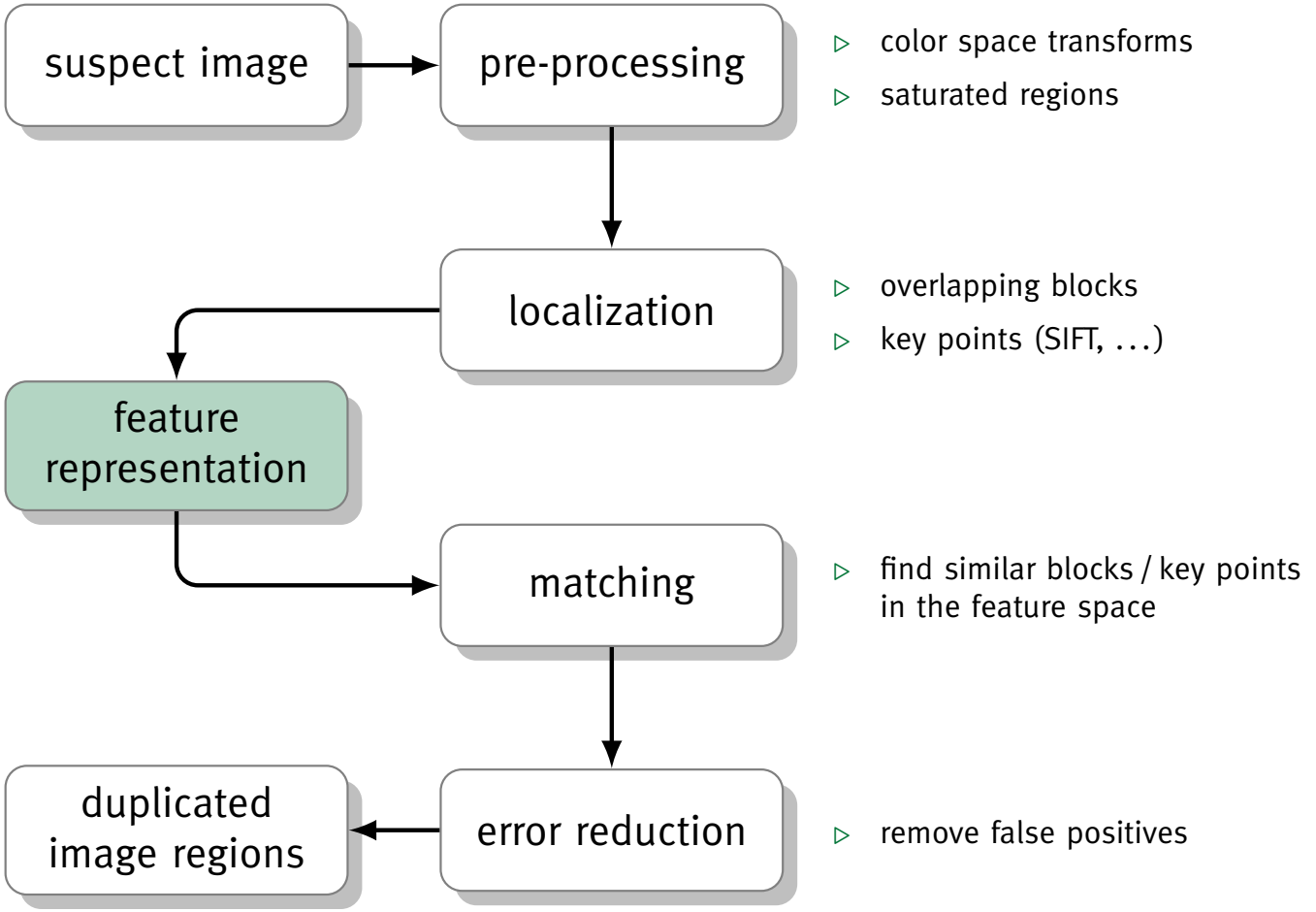


Yet Another Paper CMFD Paper?

- general detection/localization procedure since 2003 [Fridrich et al. 2003]

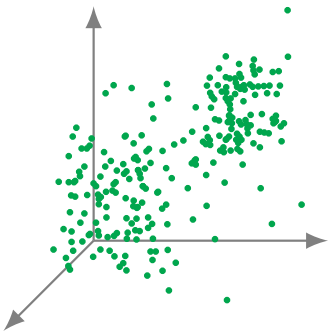


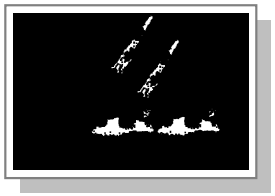


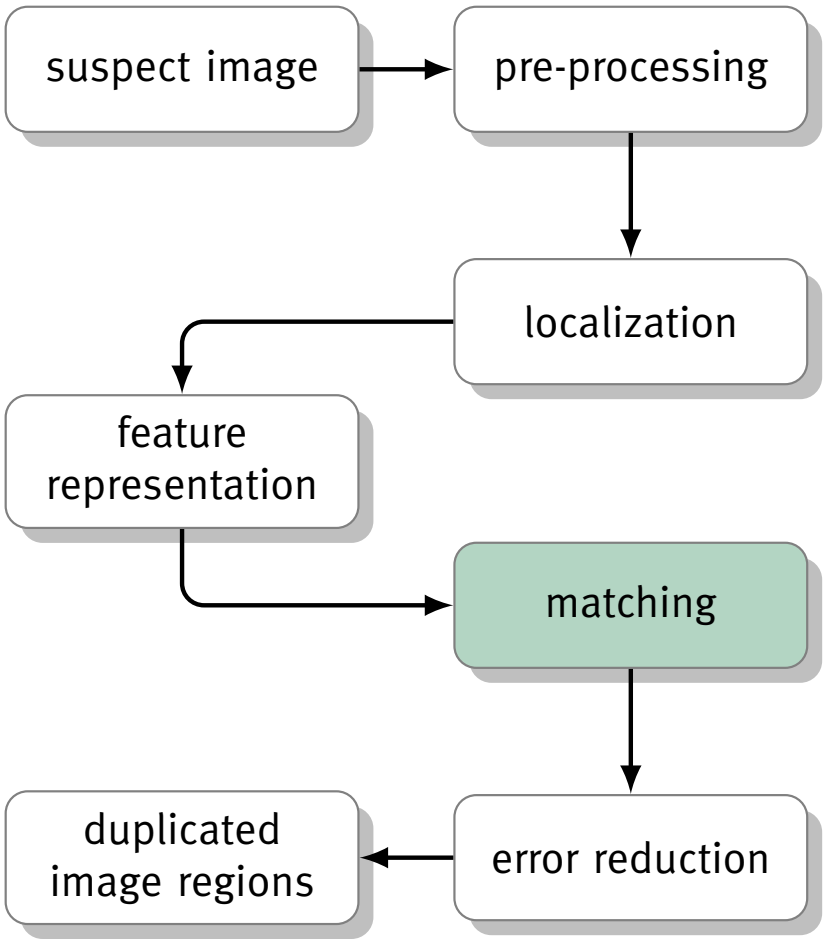


Yet Another Paper CMFD Paper?

- general detection/localization procedure since 2003 [Fridrich et al. 2003]



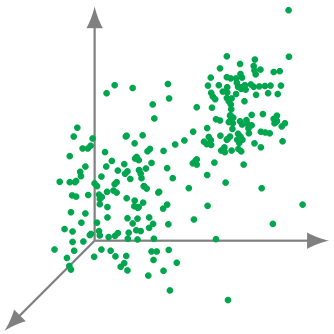


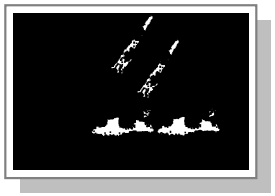


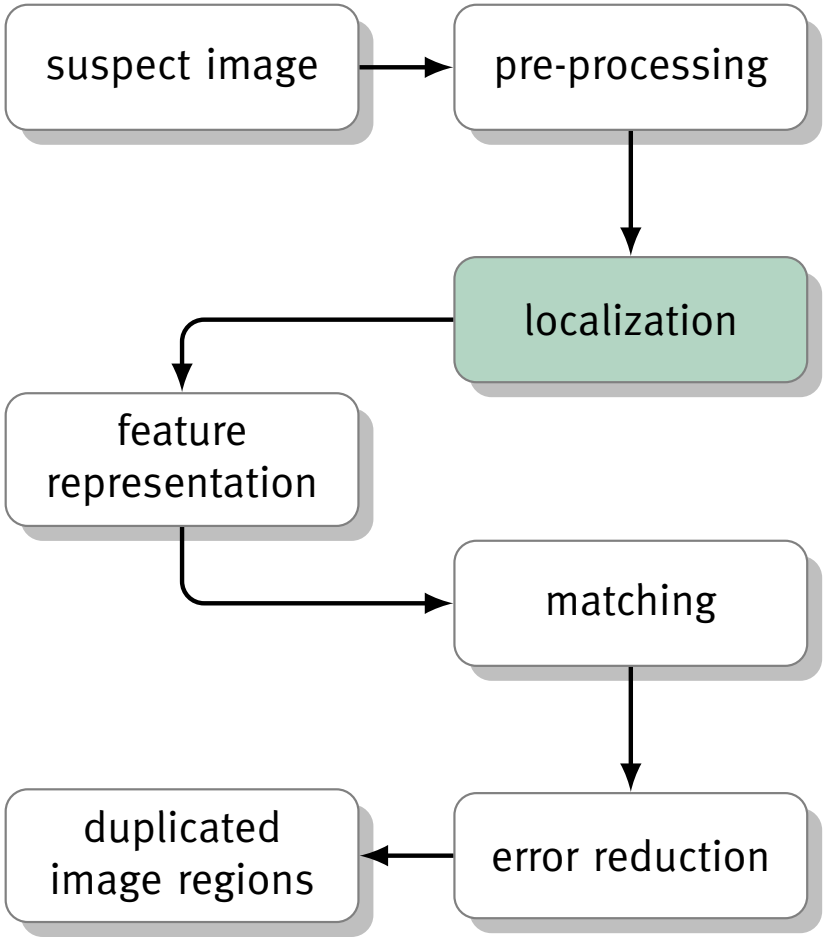
- color space transforms
- saturated regions
- overlapping blocks
- key points (SIFT, ...)
- find similar blocks / key points in the feature space
- remove false positives

Yet Another Paper CMFD Paper?

- general detection/localization procedure since 2003 [Fridrich et al. 2003]







- color space transforms
- saturated regions
- overlapping blocks
- key points (SIFT, ...)
- find similar blocks / key points in the feature space
- remove false positives

Block-Based?

- reported typical runtimes (single-thread)

	blocks (Zernike moments)	SIFT	
GRIP database (0.75 MP)	54 s	4 s	[Cozzolino et al. 2014]
Erlangen database (8 MP)	4900 s	126 s	[Christlein et al. 2012], matching step only

Block-Based?

- reported typical runtimes (single-thread)

	blocks Zernike moments	SIFT	
GRIP database (0.75 MP)	54 s	4 s	[Cozzolino et al. 2014]
Erlangen database (8 MP)	4900 s	126 s	[Christlein et al. 2012], matching step only

GRIP DB image TP_C01_013

blocks

SIFT

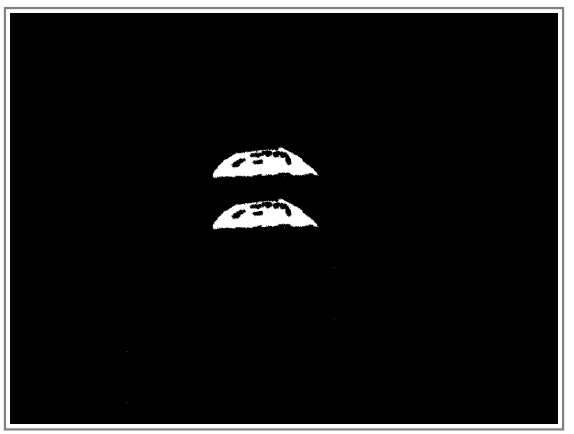
Block-Based?

- reported typical runtimes (single-thread)

	blocks Zernike moments	SIFT	
GRIP database (0.75 MP)	54 s	4 s	[Cozzolino et al. 2014]
Erlangen database (8 MP)	4900 s	126 s	[Christlein et al. 2012], matching step only

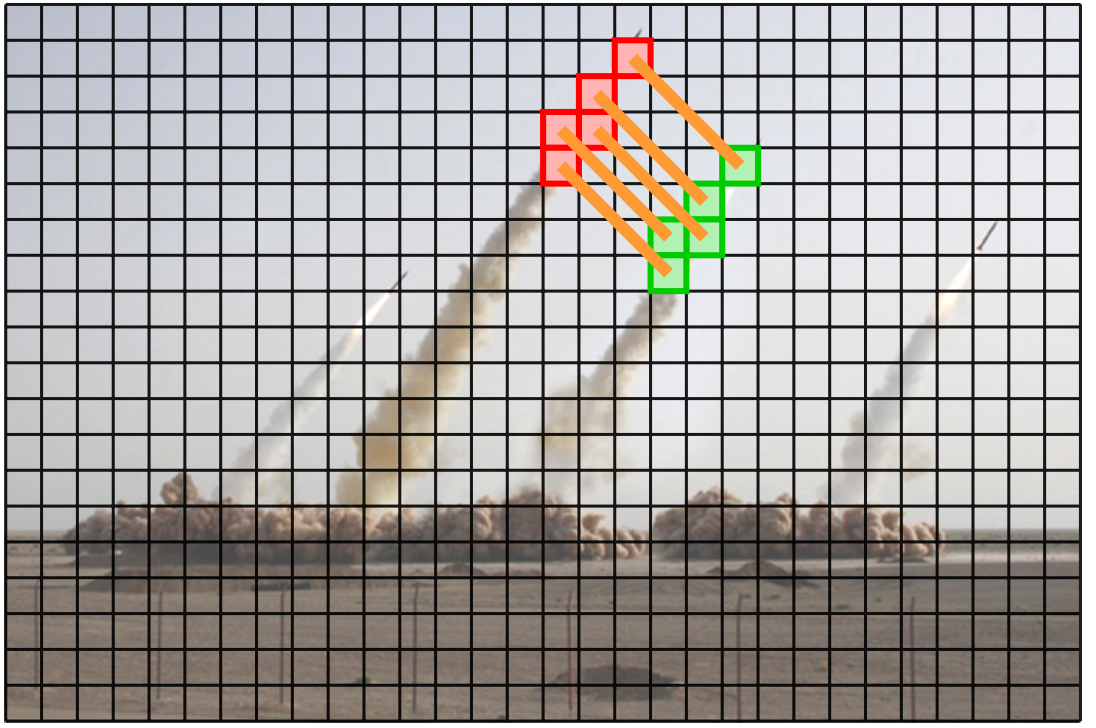
GRIP DB image TP_C01_013

blocks



block-based kD tree matching is the method of choice for most accurate CMFD

Working Assumptions

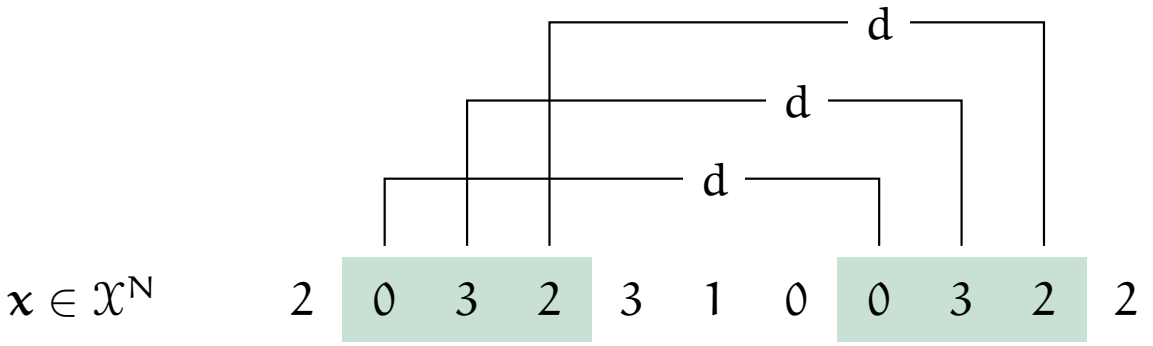


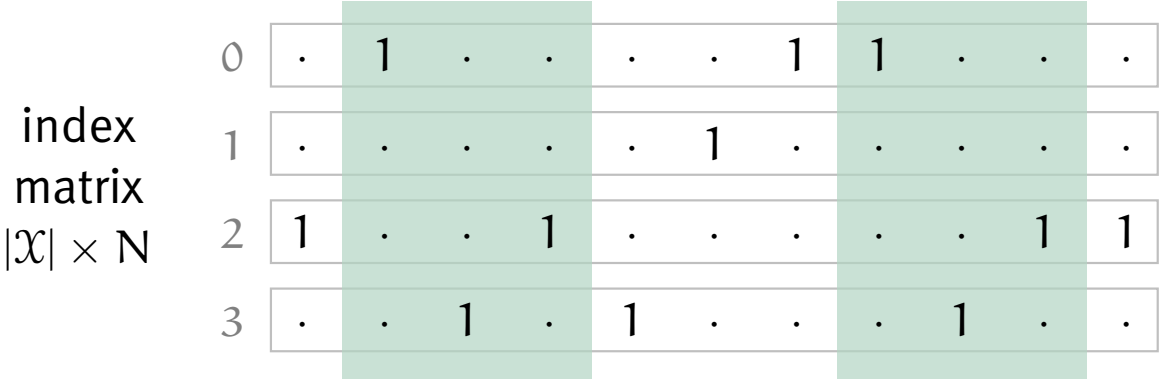
- copied regions are spatially connected subsets of feature vectors $\{\mathbf{f}_i\}$
- we focus on rigid translations only
- feature vectors can be quantized to integers, $\mathbf{f}_i \mapsto \mathbf{x}_i \in \mathbb{N}$

goal

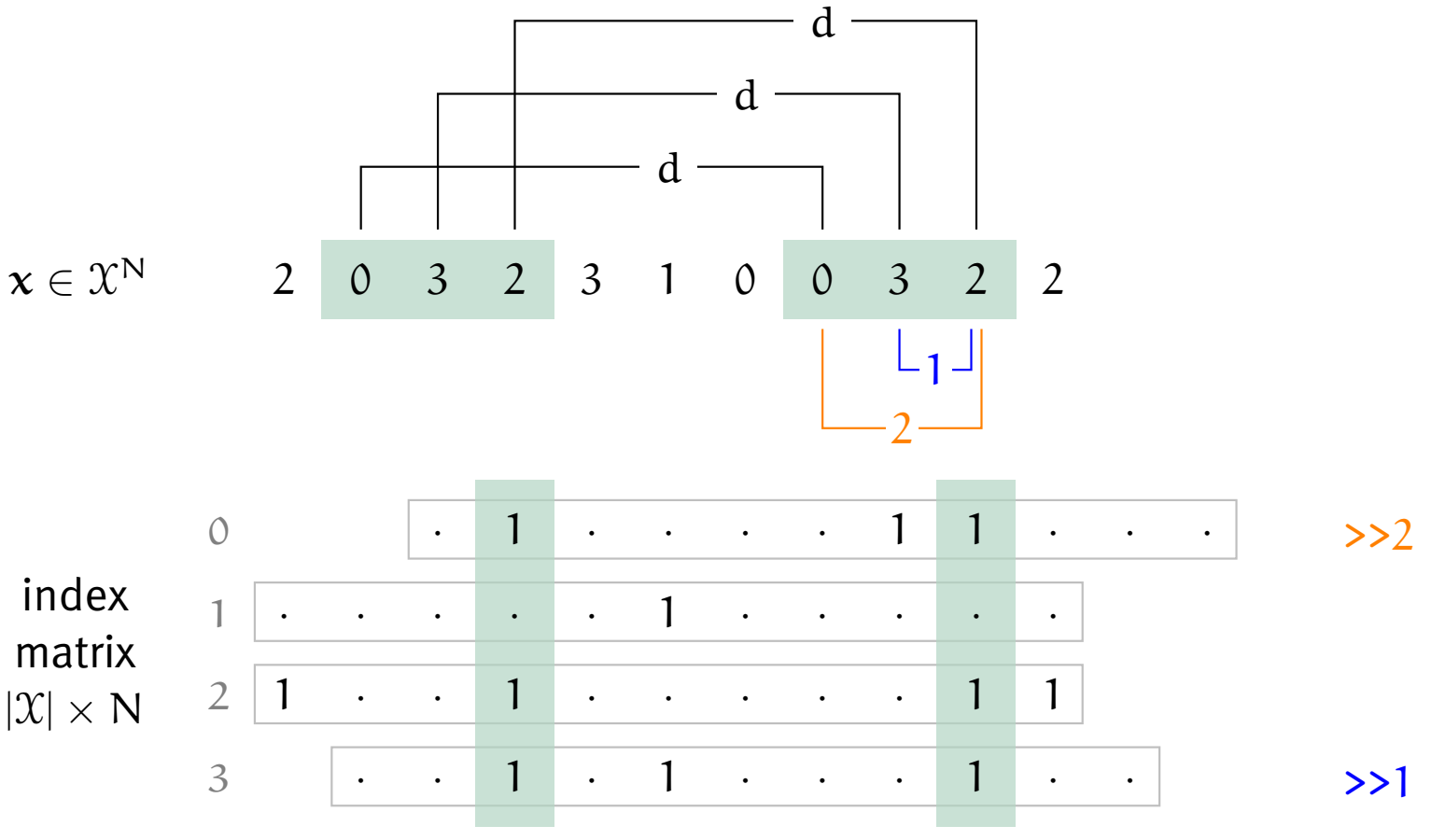
- find *all* repeated occurrences of connected patches of integers

Bit Twiddling For One Dimension

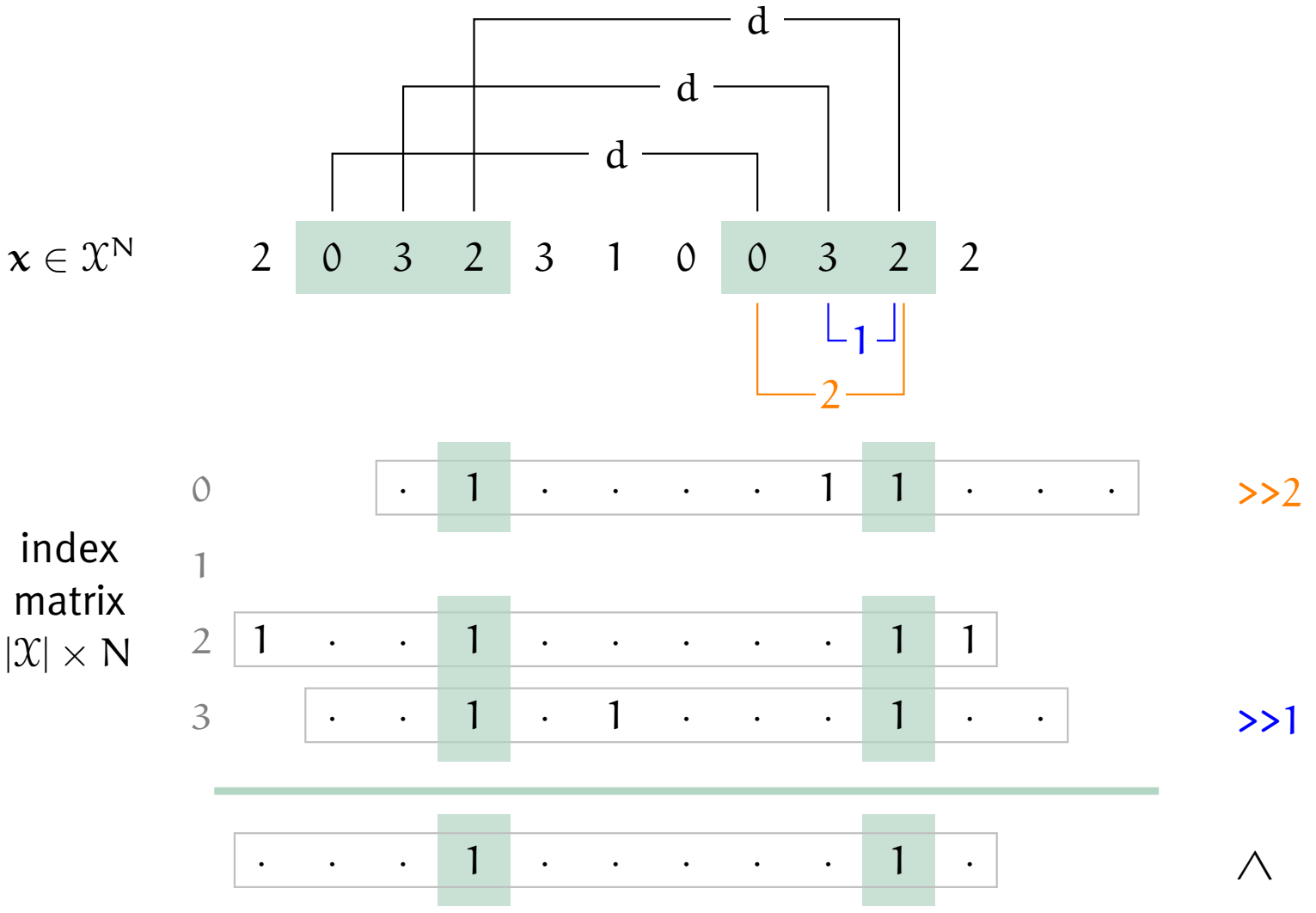




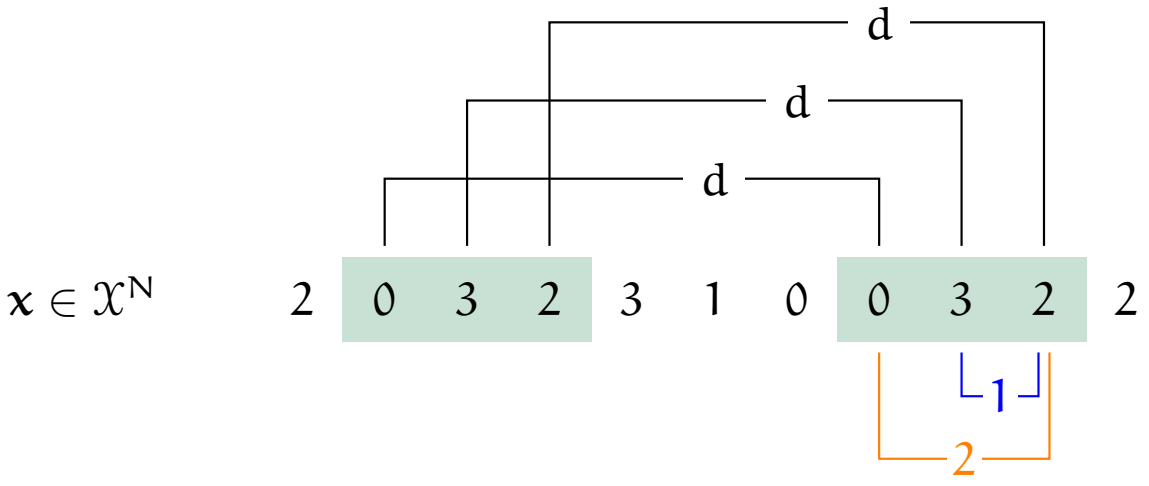
Bit Twiddling For One Dimension



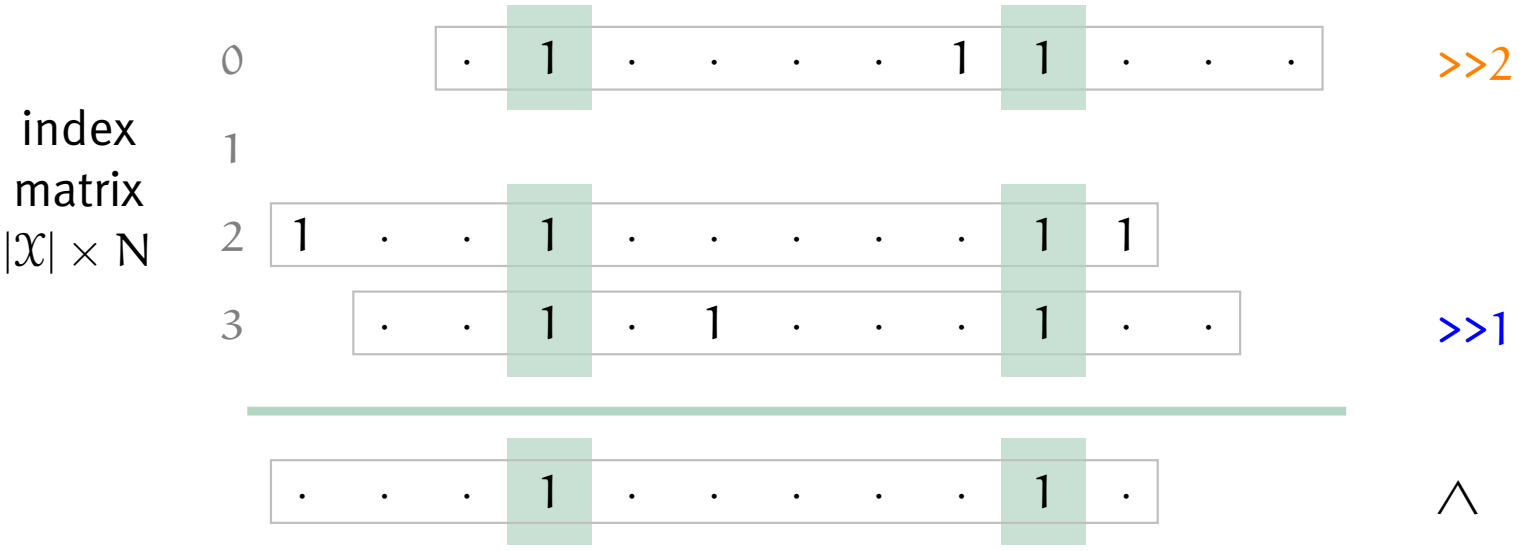
Bit Twiddling For One Dimension



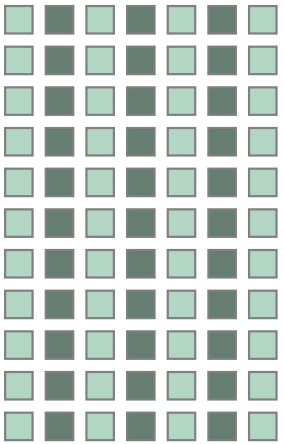
Bit Twiddling For One Dimension



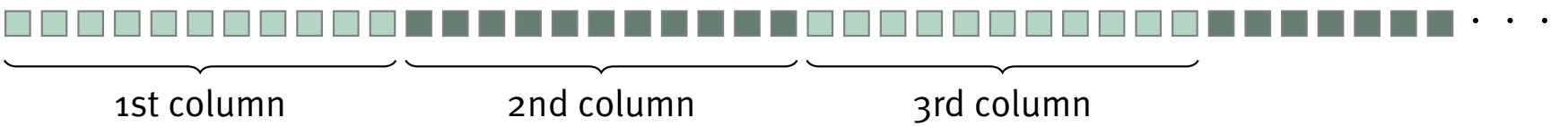
- shift/AND for all blocks
- re-use earlier shifts
- buffer columns for non-unique elements



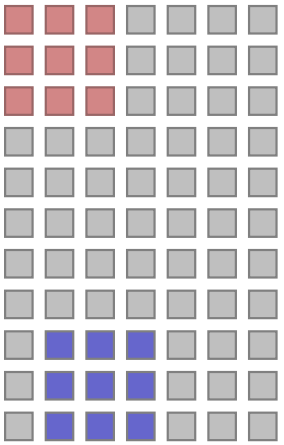
Extension to 2D



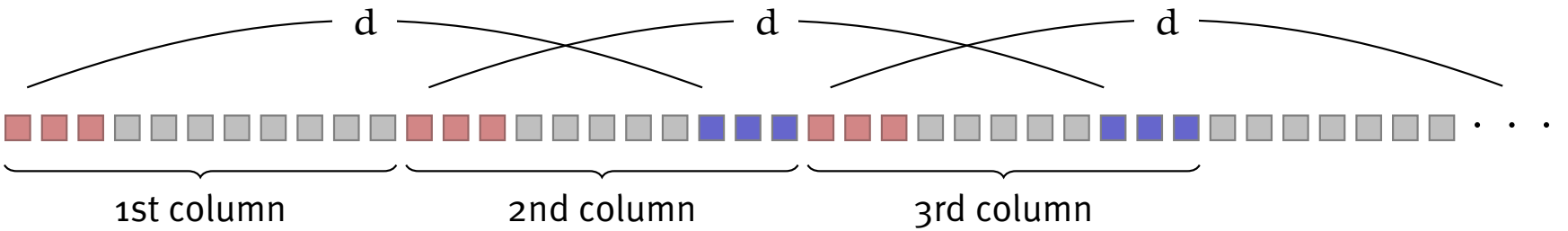
- column-major vectorization of the input



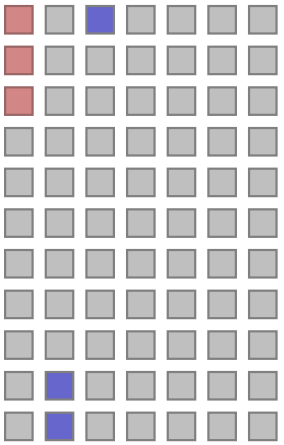
Extension to 2D



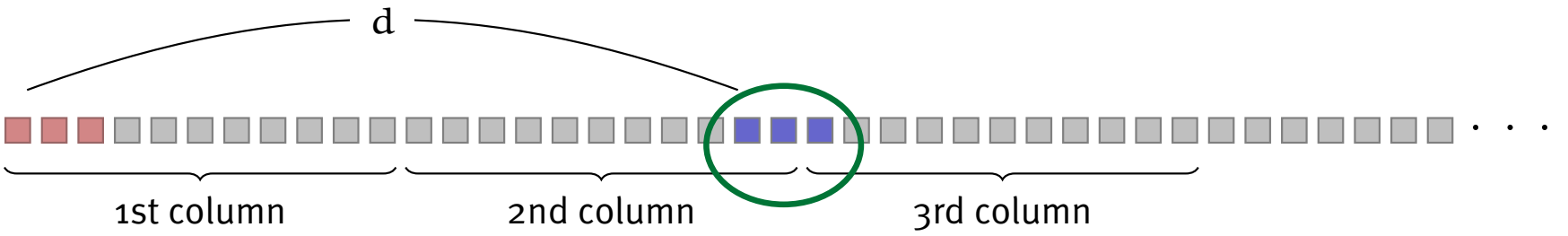
- column-major vectorization of the input to find duplicate $w \times 1$ patches



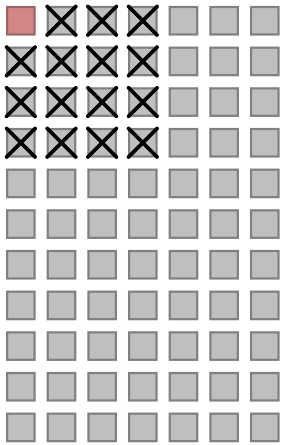
Extension to 2D



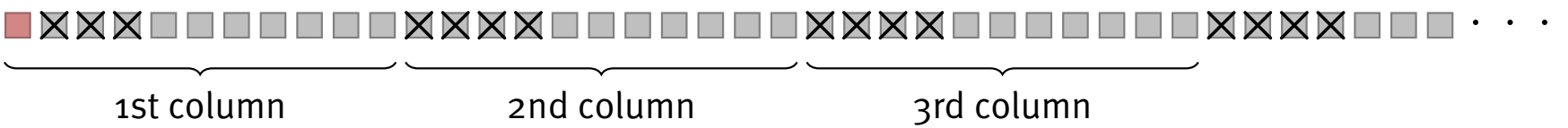
- column-major vectorization of the input to find duplicate $w \times 1$ patches
- special treatment of border elements



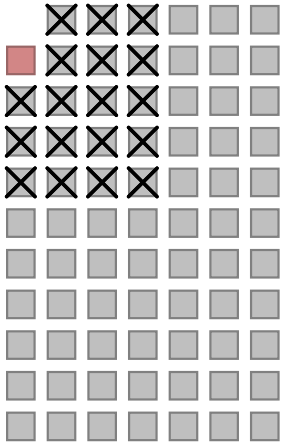
Extension to 2D



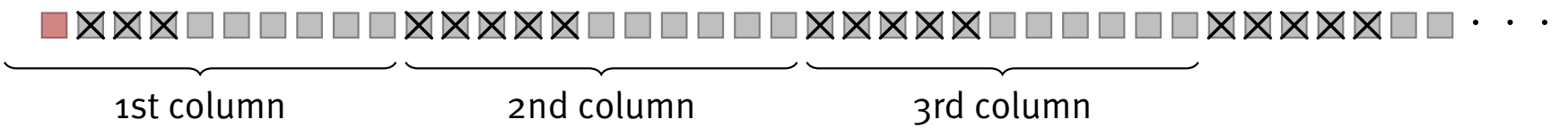
- column-major vectorization of the input to find duplicate $w \times 1$ patches
- special treatment of border elements
- varying masks for spatially inadmissible patches



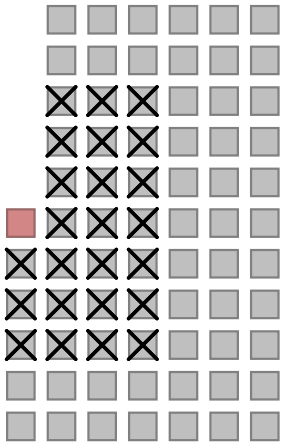
Extension to 2D



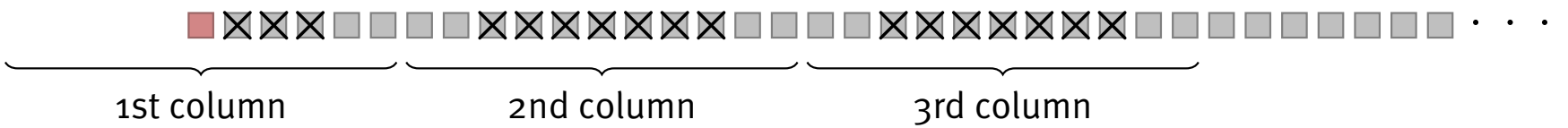
- column-major vectorization of the input to find duplicate $w \times 1$ patches
- special treatment of border elements
- varying masks for spatially inadmissible patches



Extension to 2D

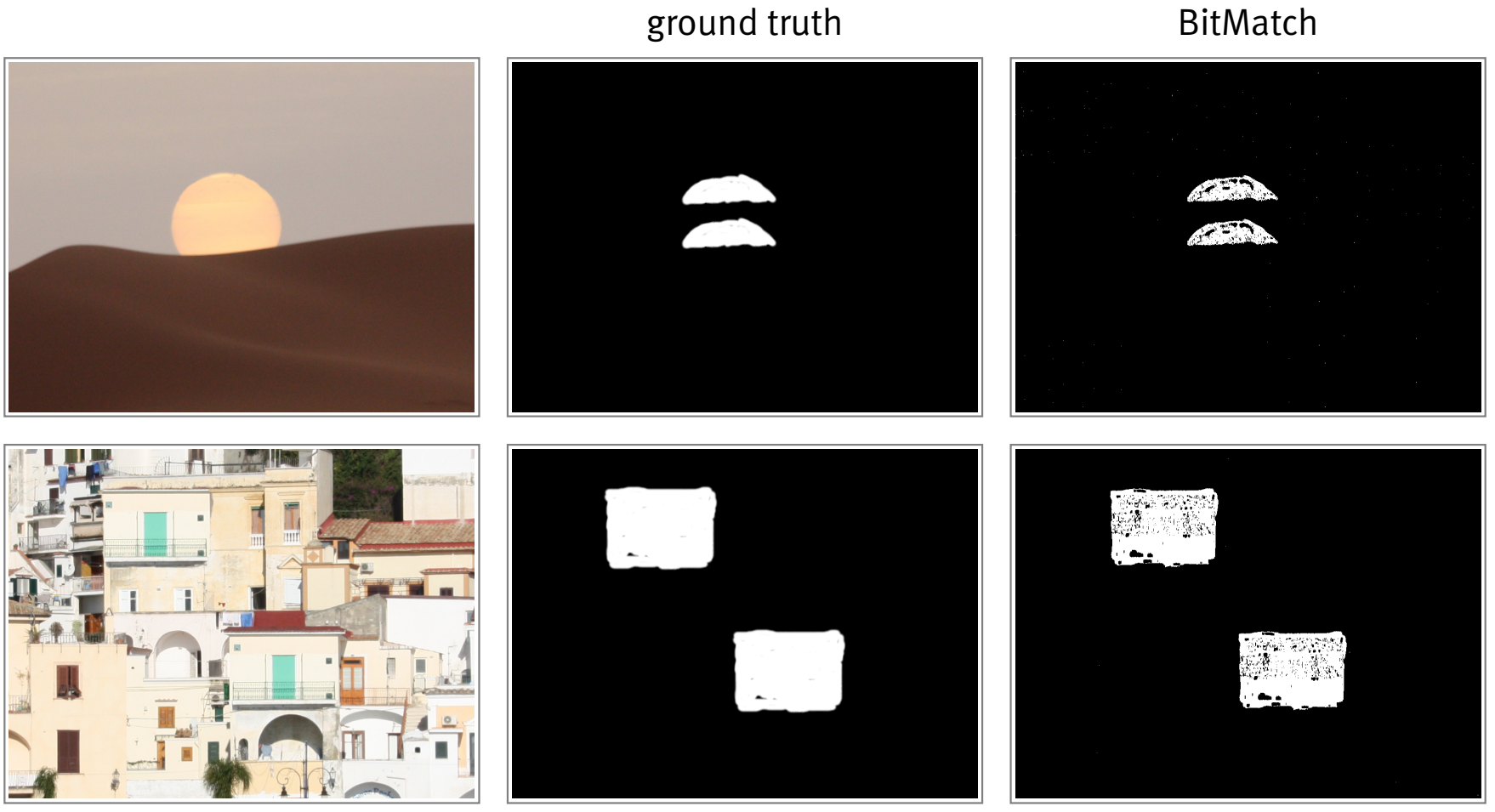


- column-major vectorization of the input to find duplicate $w \times 1$ patches
- special treatment of border elements
- varying masks for spatially inadmissible patches



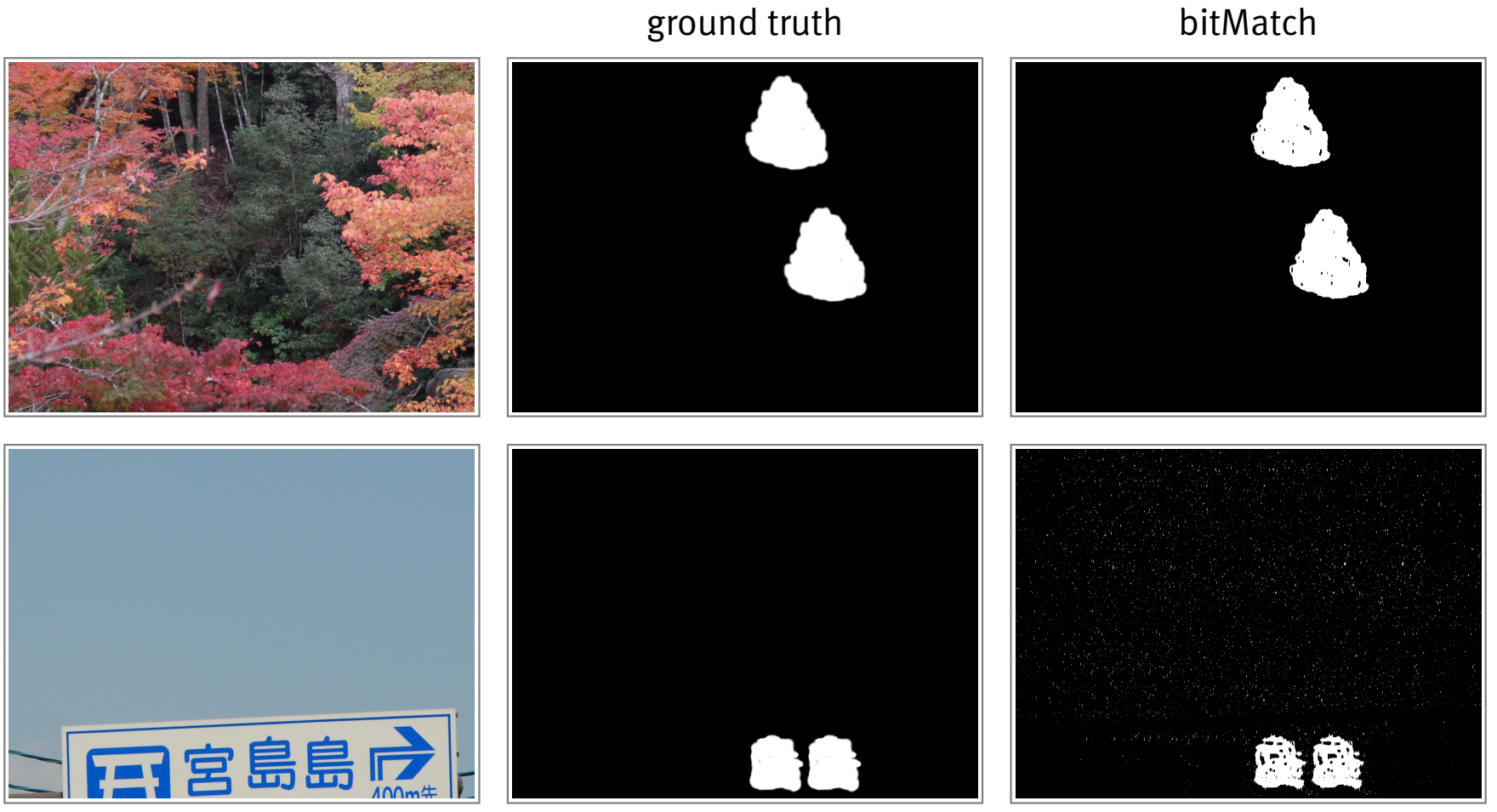
Typical Results

GRIP-UNINA Database, intensity-based exact matching, 5×1 patches



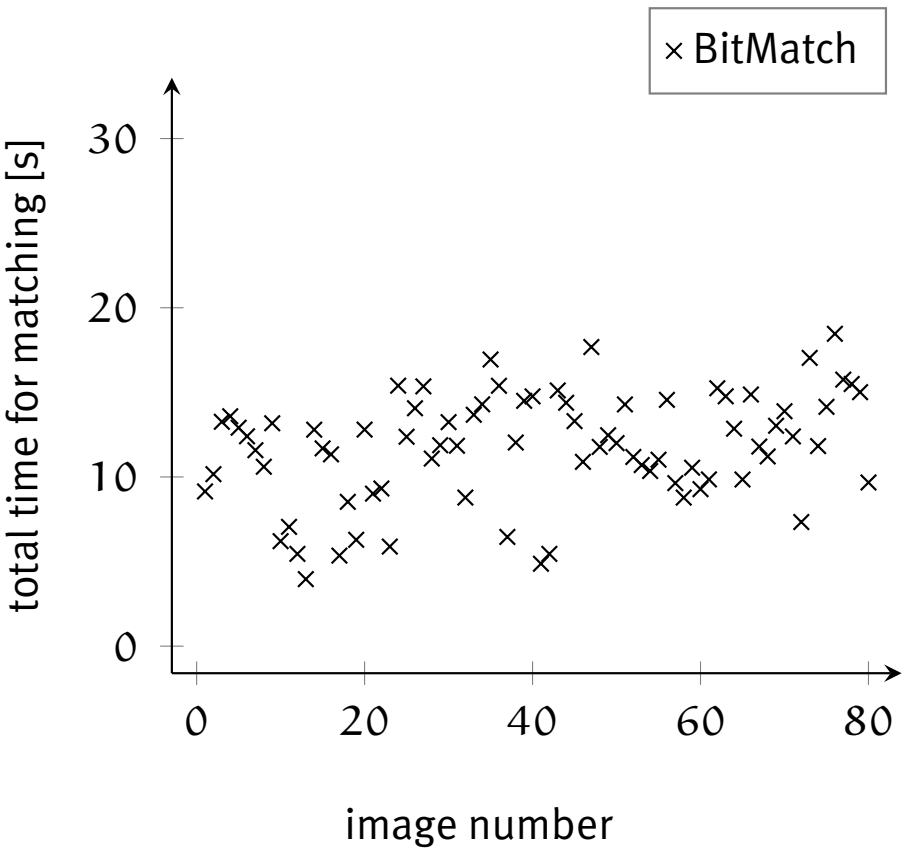
Typical Results

GRIP-UNINA Database, intensity-based exact matching, 5×1 patches



Quantitative Results

The Rise of Lexicographic Matching



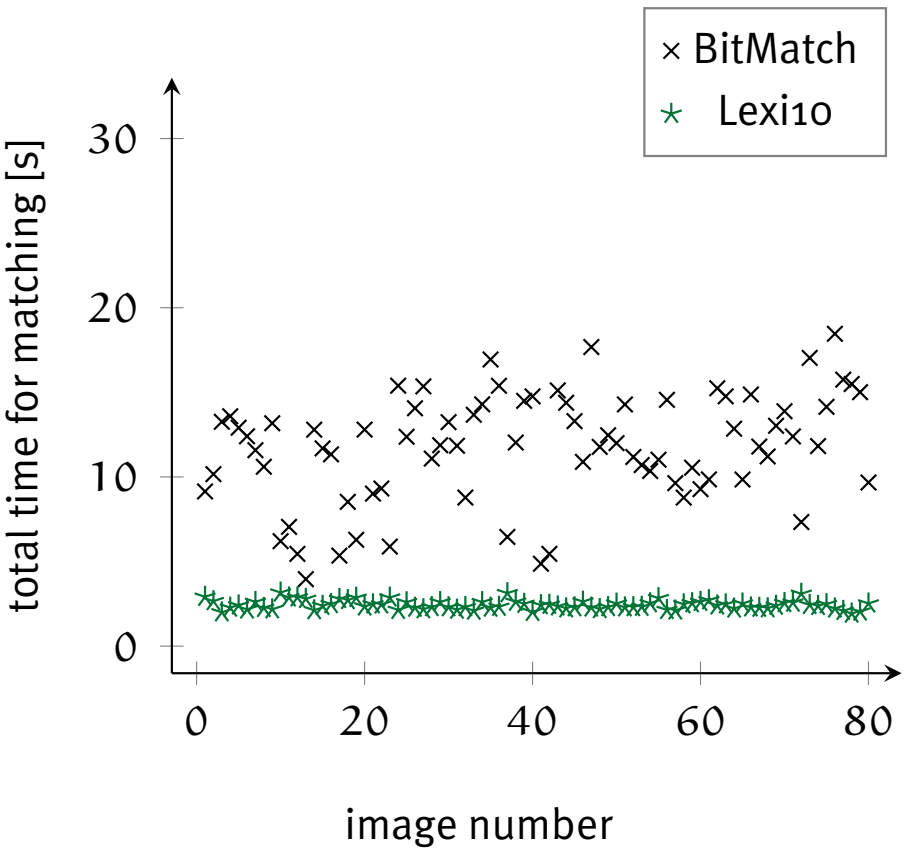
GRIP-UNINA Database

F-measure: $F = \frac{2TP}{2TP + FP + FN}$

	F-measure	runtime
BitMatch	0.9487	11.71598

Quantitative Results

The Rise of Lexicographic Matching



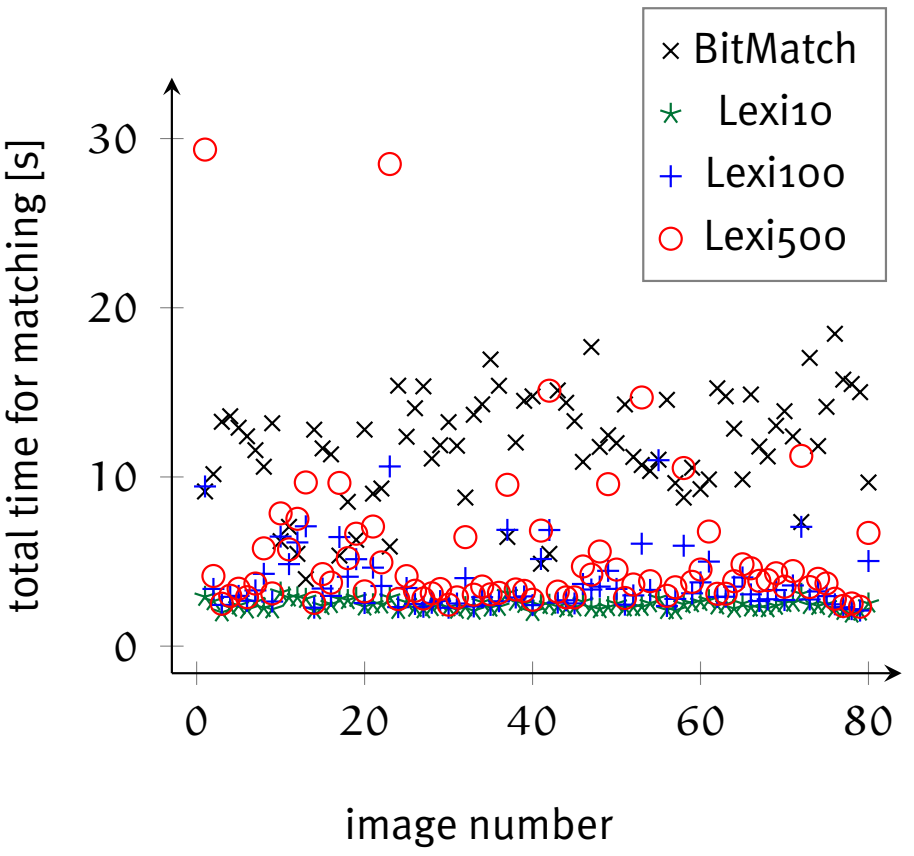
GRIP-UNINA Database

F-measure: $F = \frac{2TP}{2TP + FP + FN}$

	F-measure	runtime
BitMatch	0.9487	11.71598
Lexi10	0.8617	2.43631

Quantitative Results

The Rise of Lexicographic Matching



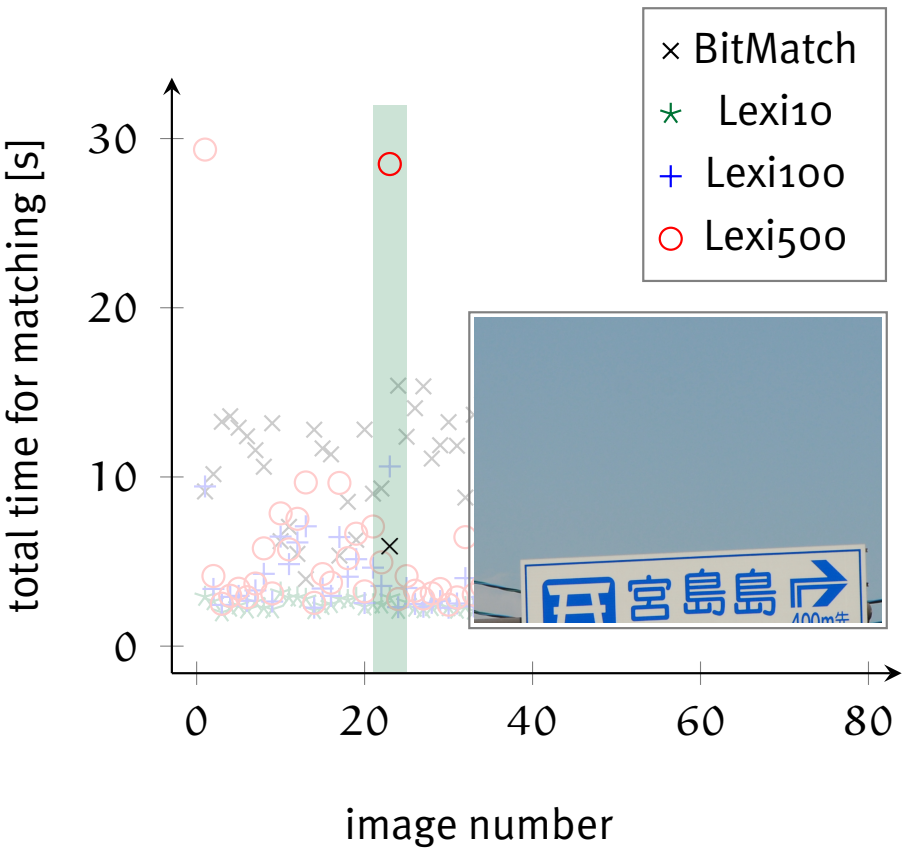
GRIP-UNINA Database

F-measure: $F = \frac{2TP}{2TP + FP + FN}$

	F-measure	runtime
BitMatch	0.9487	11.71598
Lexi10	0.8617	2.43631
Lexi100	0.9343	3.78639
Lexi500	0.9477	5.73957

Quantitative Results

The Rise of Lexicographic Matching



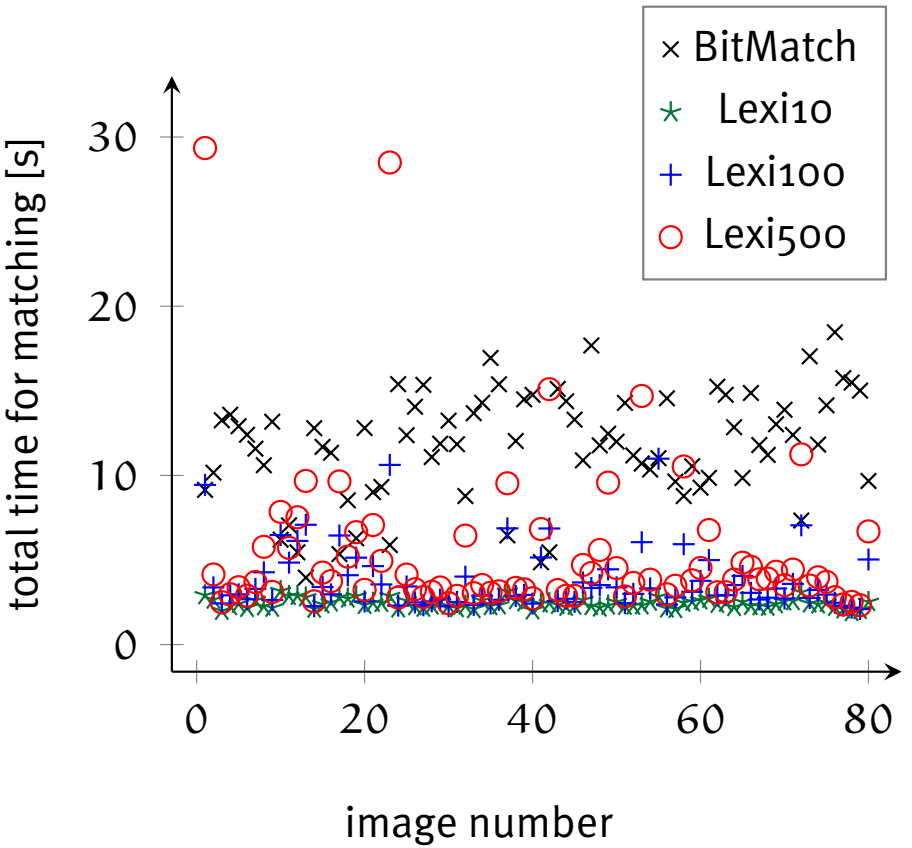
GRIP-UNINA Database

$$\text{F-measure:} \quad F = \frac{2TP}{2TP + FP + FN}$$

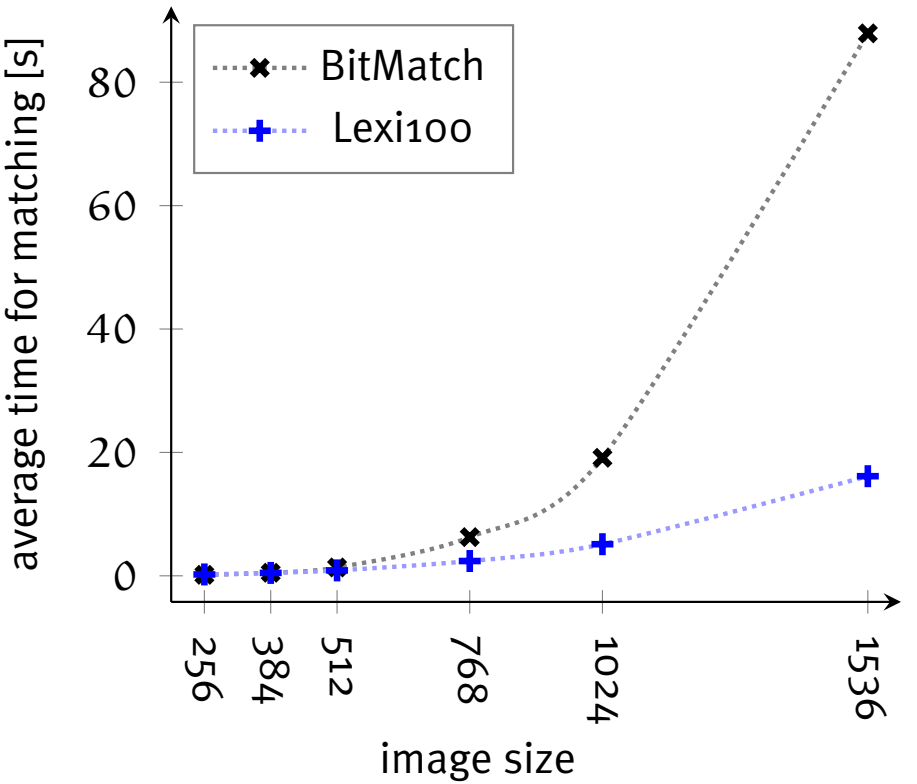
	F-measure	runtime
BitMatch	0.9487	11.71598
Lexi10	0.8617	2.43631
Lexi100	0.9343	3.78639
Lexi500	0.9477	5.73957

Quantitative Results

The Fall of BitMatch

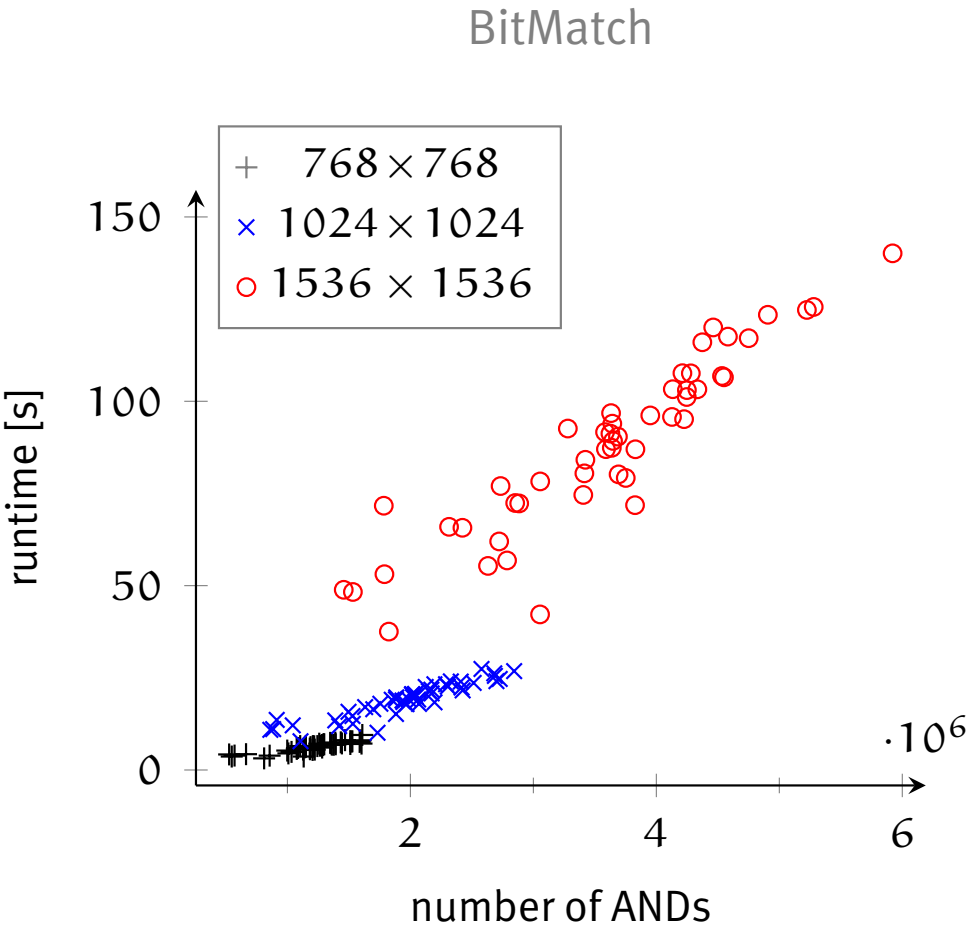


GRIP-UNINA Database

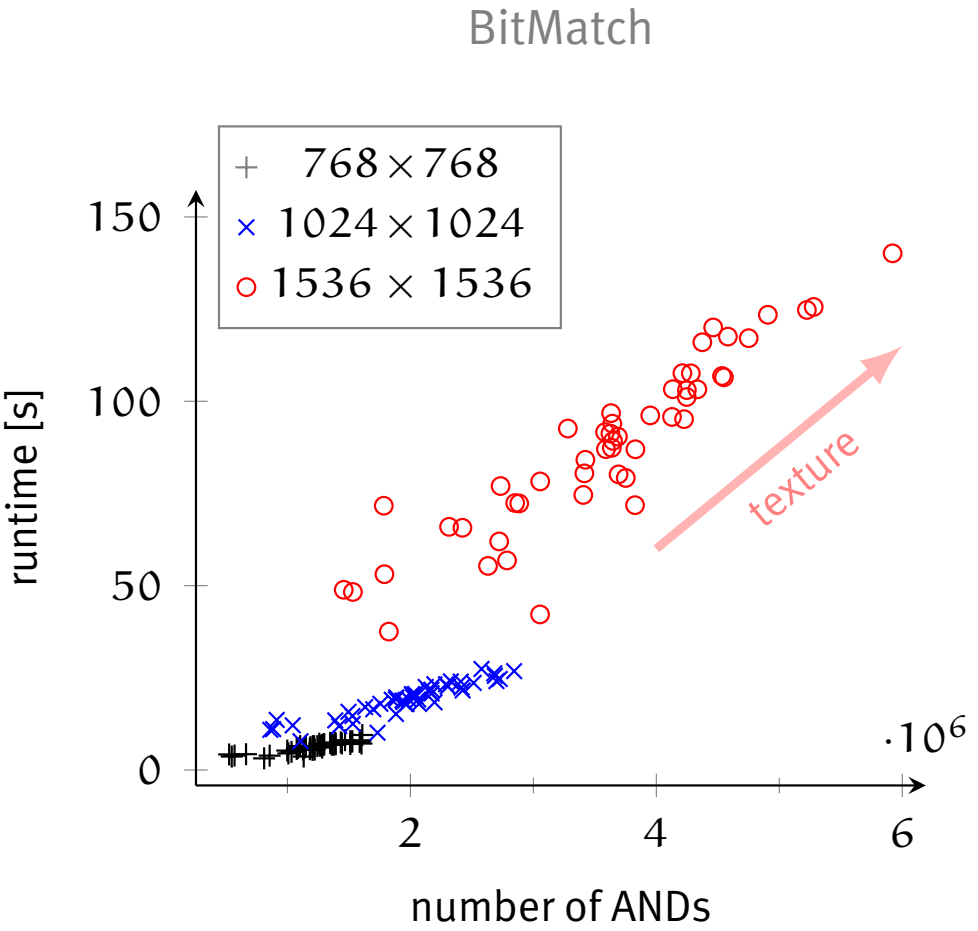


Dresden Image Database

Bottlenecks

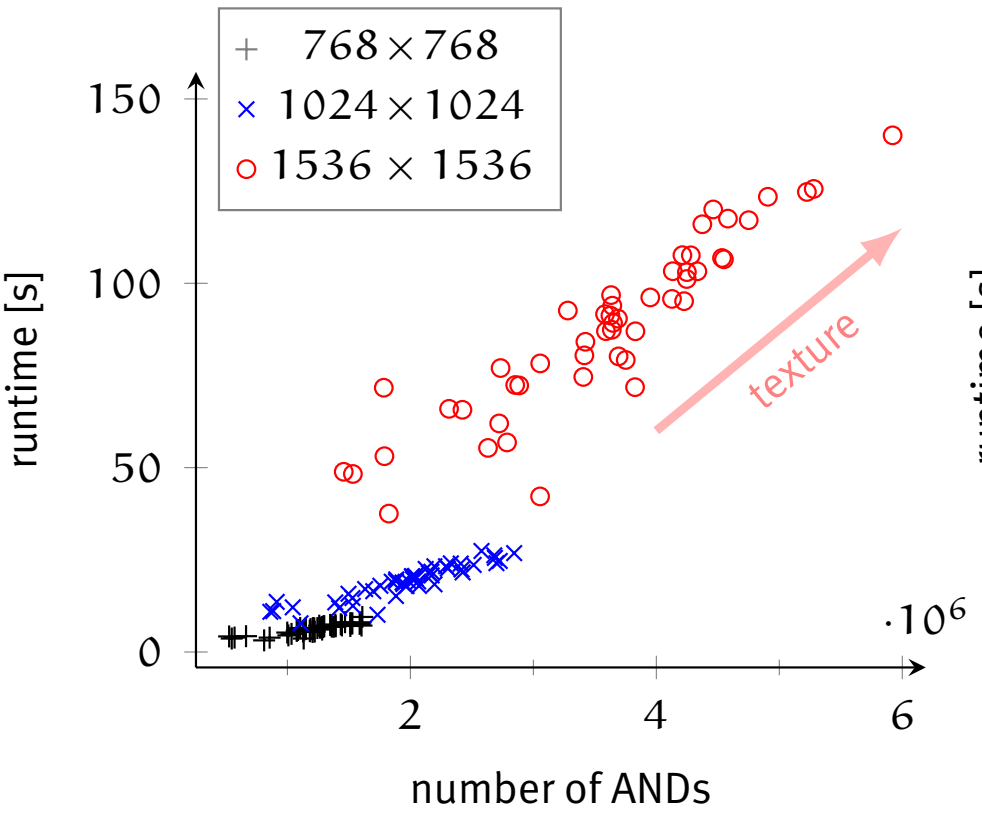


Bottlenecks

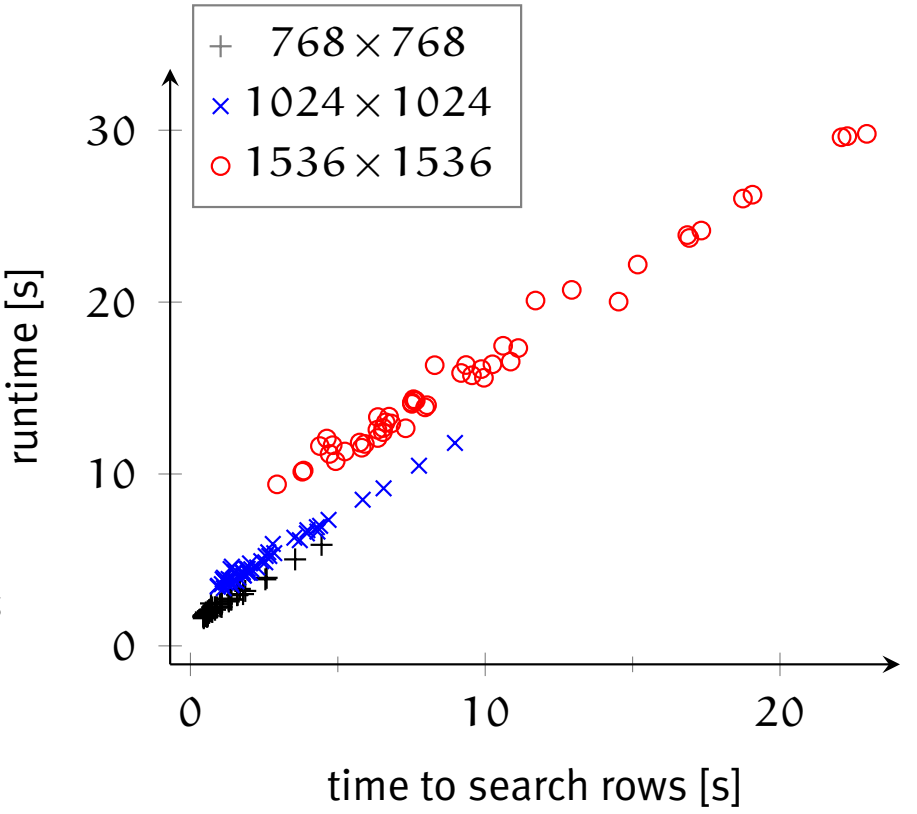


Bottlenecks

BitMatch

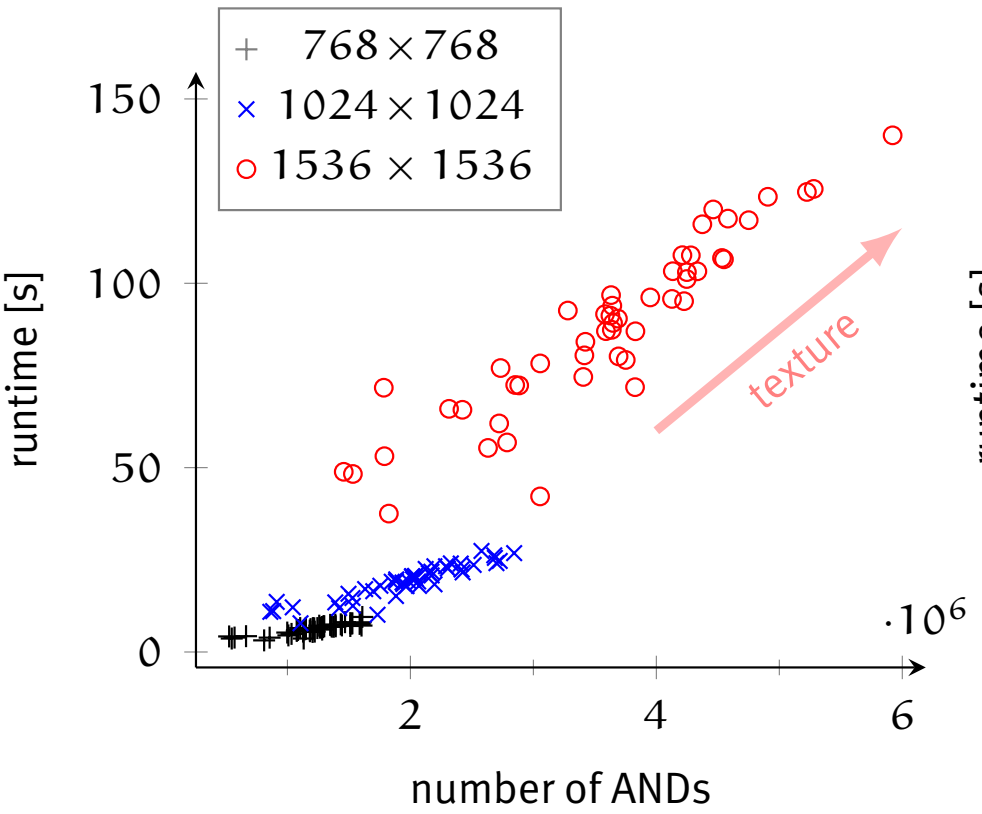


lexicographic matching

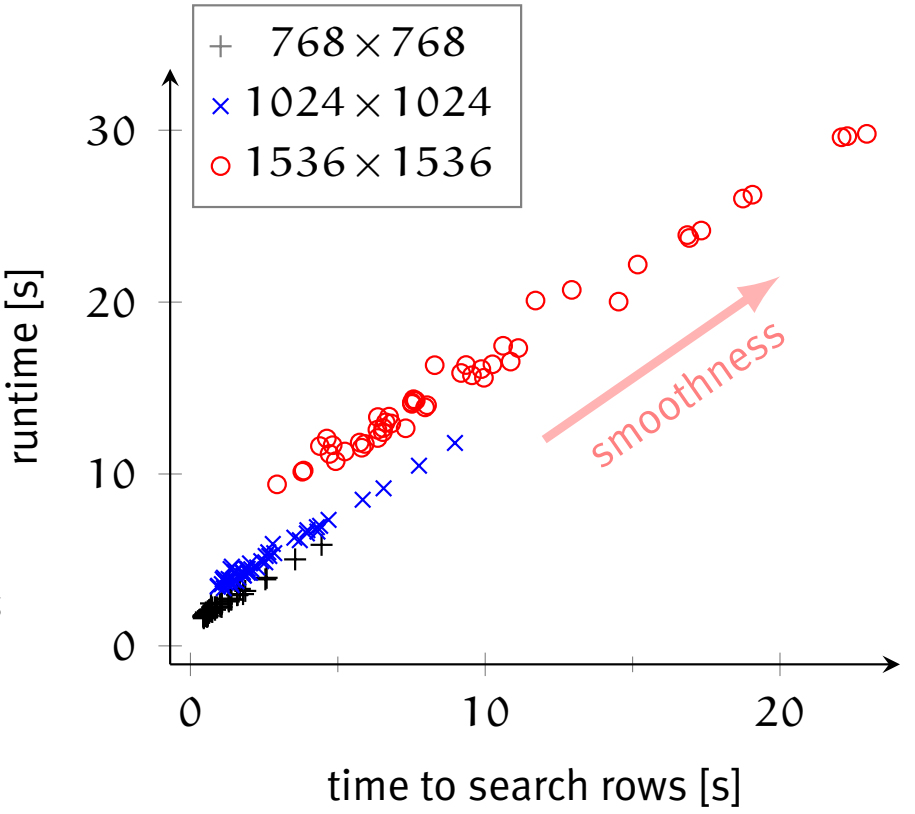


Bottlenecks

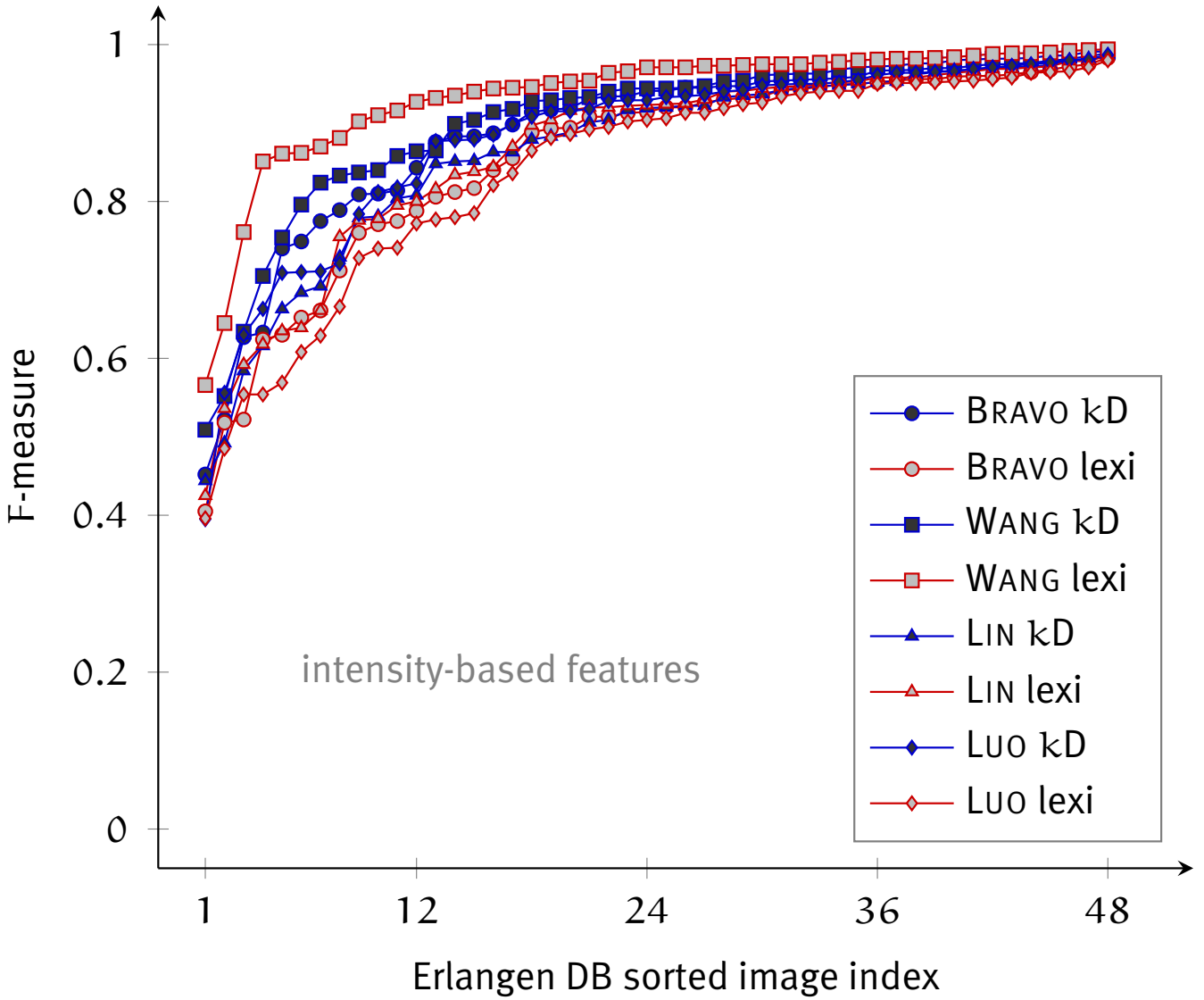
BitMatch



lexicographic matching

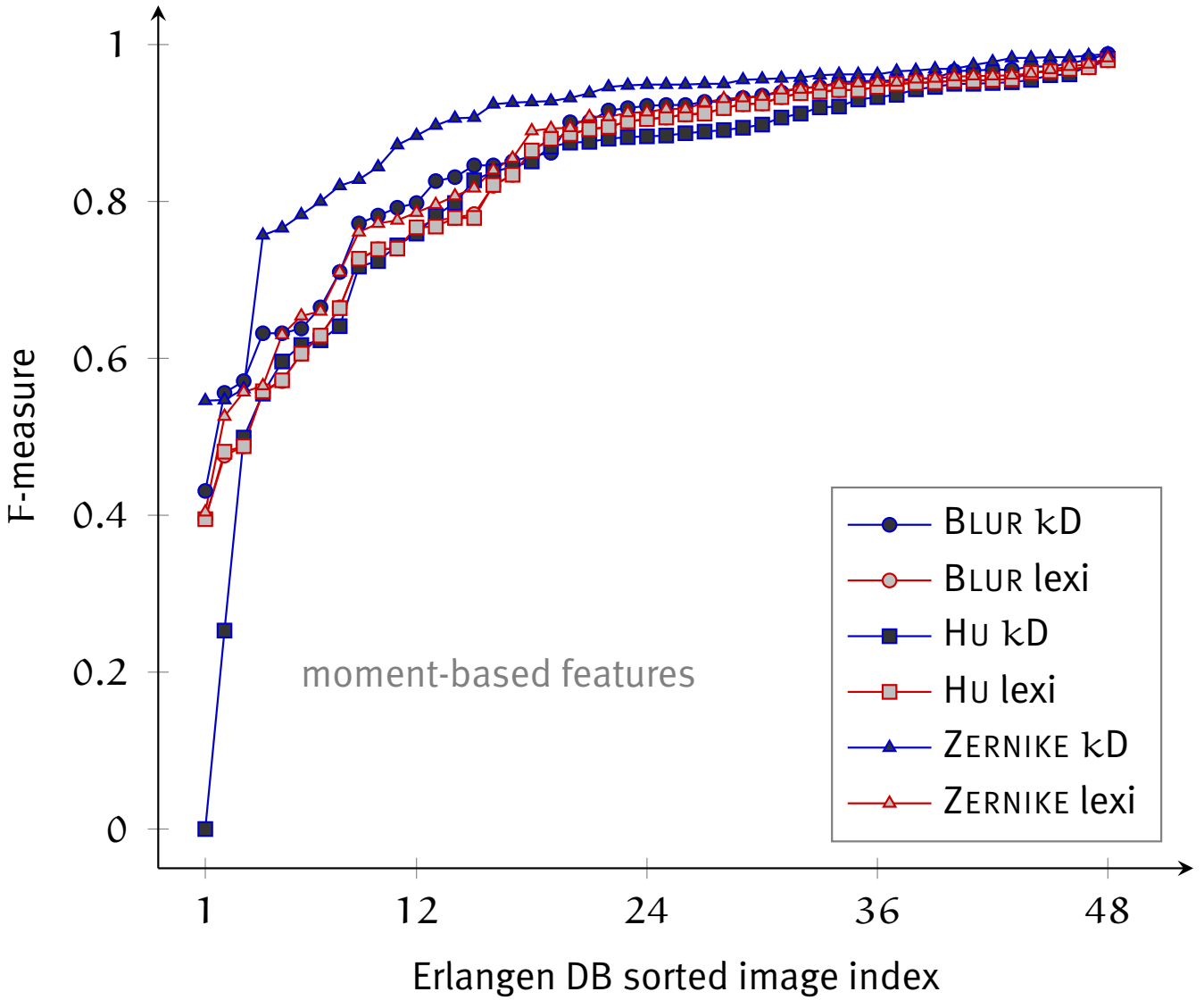


Back to the Roots



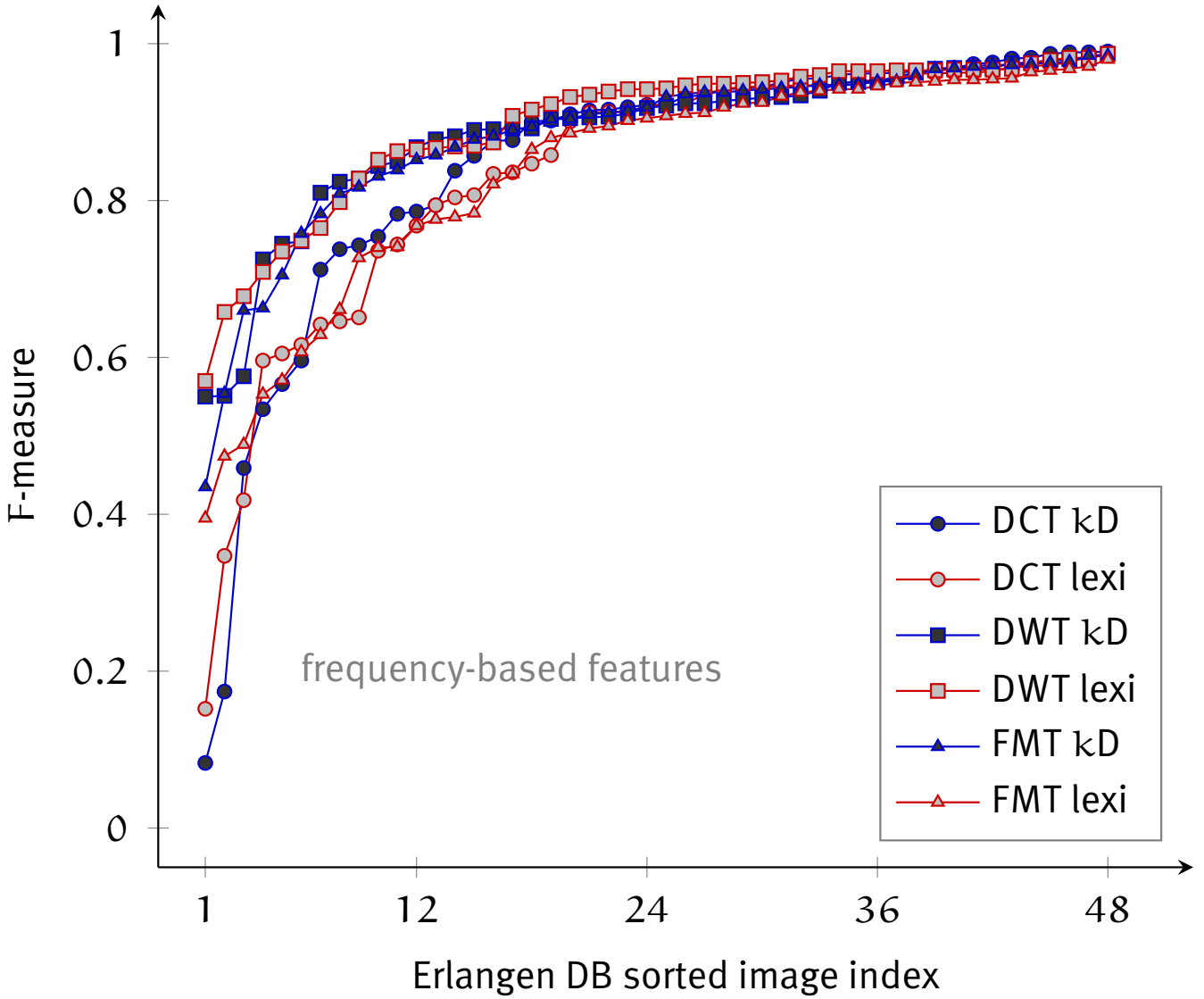
16×16 blocks
$\geqslant 800$ equal shifts

Back to the Roots



16×16 blocks
$\geqslant 800$ equal shifts

Back to the Roots



16×16 blocks
$\geqslant 800$ equal shifts

Summary

BitMatch

- replace feature-matching with index-based bit-wise operations to find *all exact* duplications

Summary

BitMatch

- replace feature-matching with index-based bit-wise operations to find *all exact* duplications
- current implementation has some (limited) benefits, but in general lexicographic matching works *surprisingly* well

Summary

BitMatch

- replace feature-matching with index-based bit-wise operations to find *all exact* duplications
- current implementation has some (limited) benefits, but in general lexicographic matching works *surprisingly* well

Back to the Roots?

- re-evaluation of kD-tree predominance highly recommended

Summary

BitMatch

- replace feature-matching with index-based bit-wise operations to find *all exact* duplications
- current implementation has some (limited) benefits, but in general lexicographic matching works *surprisingly* well

Back to the Roots?

- re-evaluation of kD-tree predominance highly recommended

Future Work: Exact Matching?

- make working assumption of scalar feature-vector representations work

Summary

BitMatch

- replace feature-matching with index-based bit-wise operations to find *all exact* duplications
- current implementation has some (limited) benefits, but in general lexicographic matching works *surprisingly* well

Back to the Roots?

- re-evaluation of kD-tree predominance highly recommended

Future Work: Exact Matching?

- make working assumption of scalar feature-vector representations work

- source code available: `ws.binghamton.edu/kirchner`

Thinking Beyond the Block

Block Matching for Copy–Move Forgery Detection Revisited

Matthias Kirchner — Binghamton University
Pascal Schöttle — University of Münster
Christian Riess — Stanford University

IS&T/SPIE Media Watermarking, Security, and Forensics
San Francisco, CA | February 9, 2015