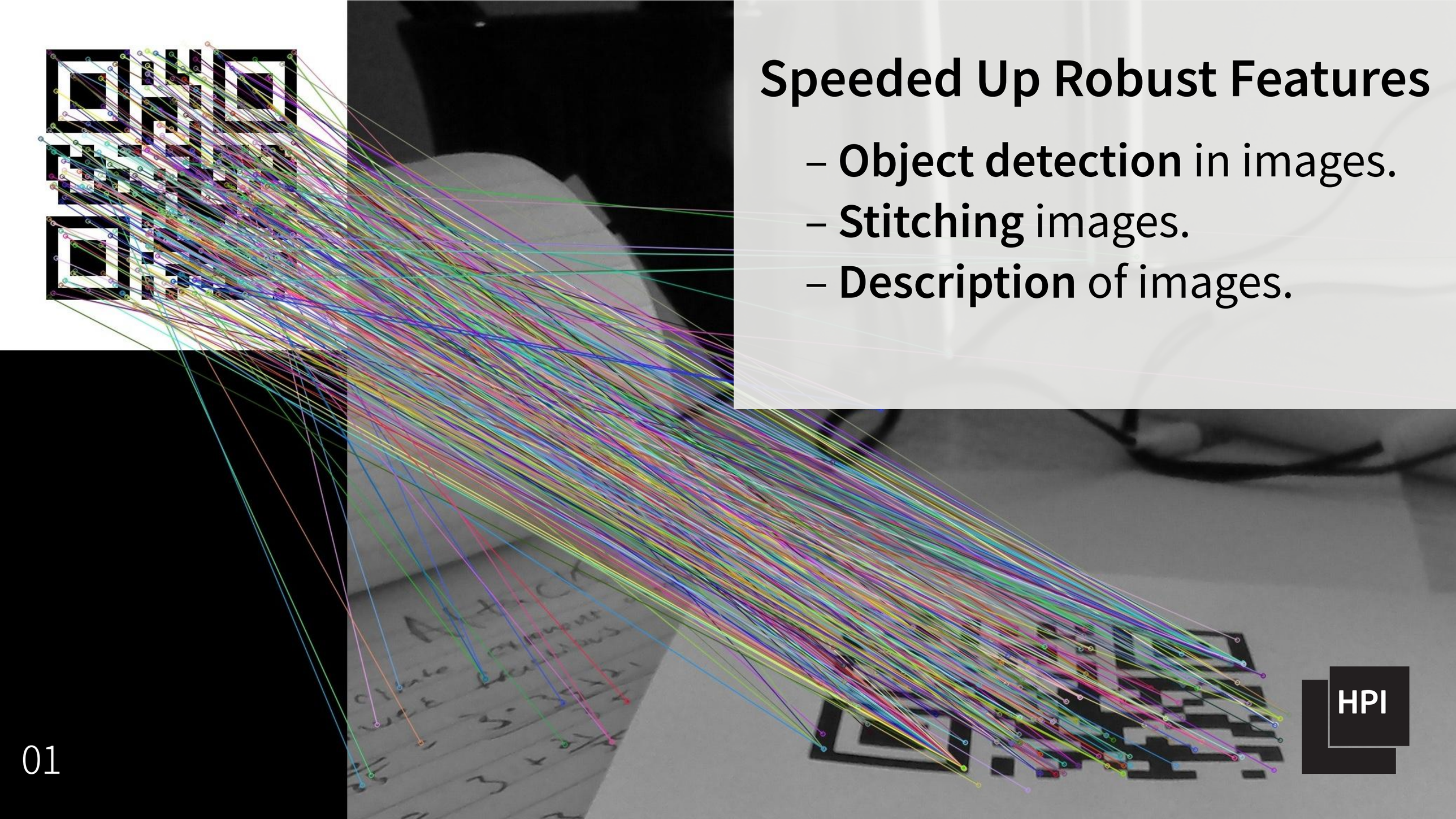
Patrick Schmidt, Christoph Sterz

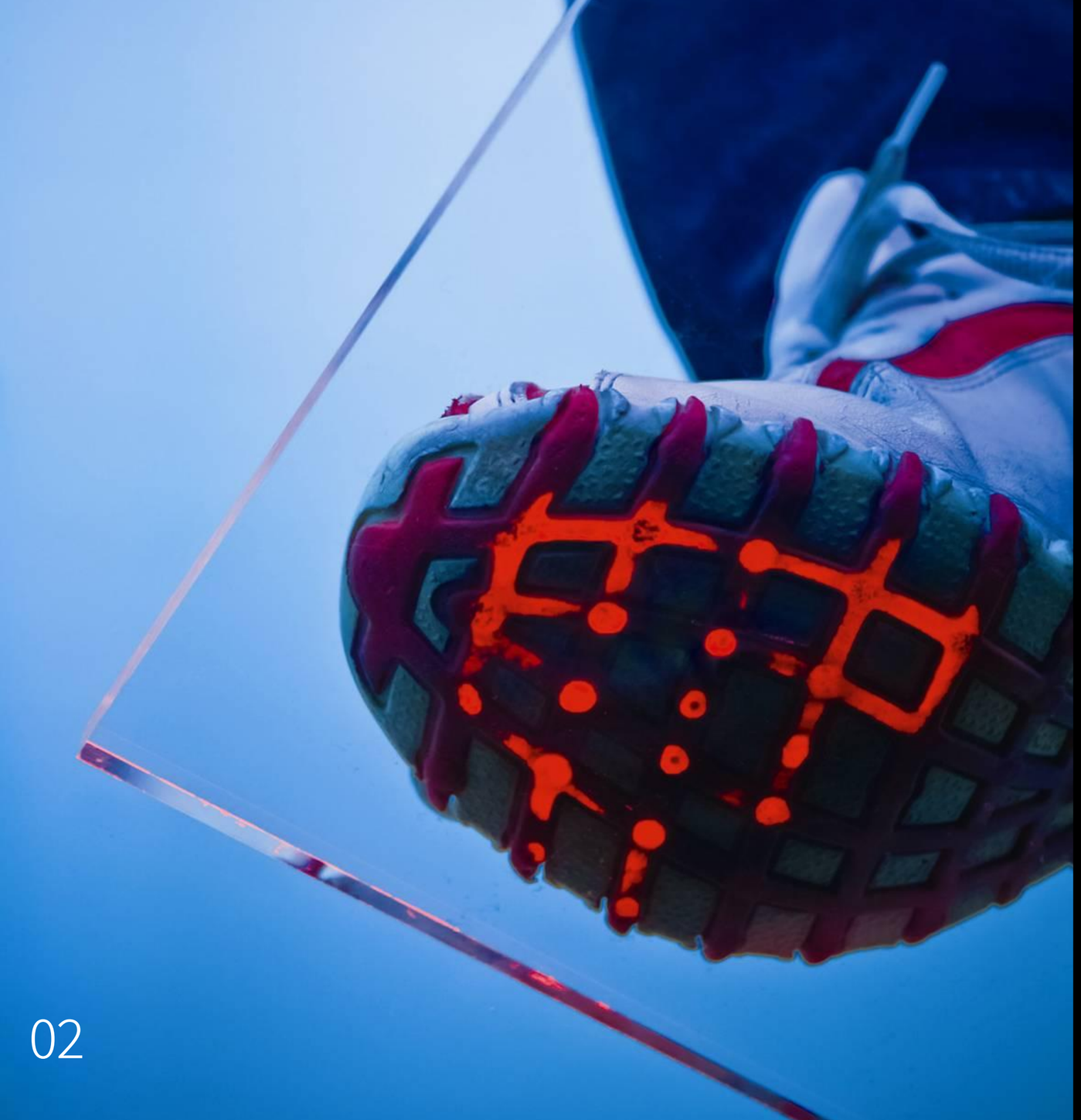
NUMA-aware SURF

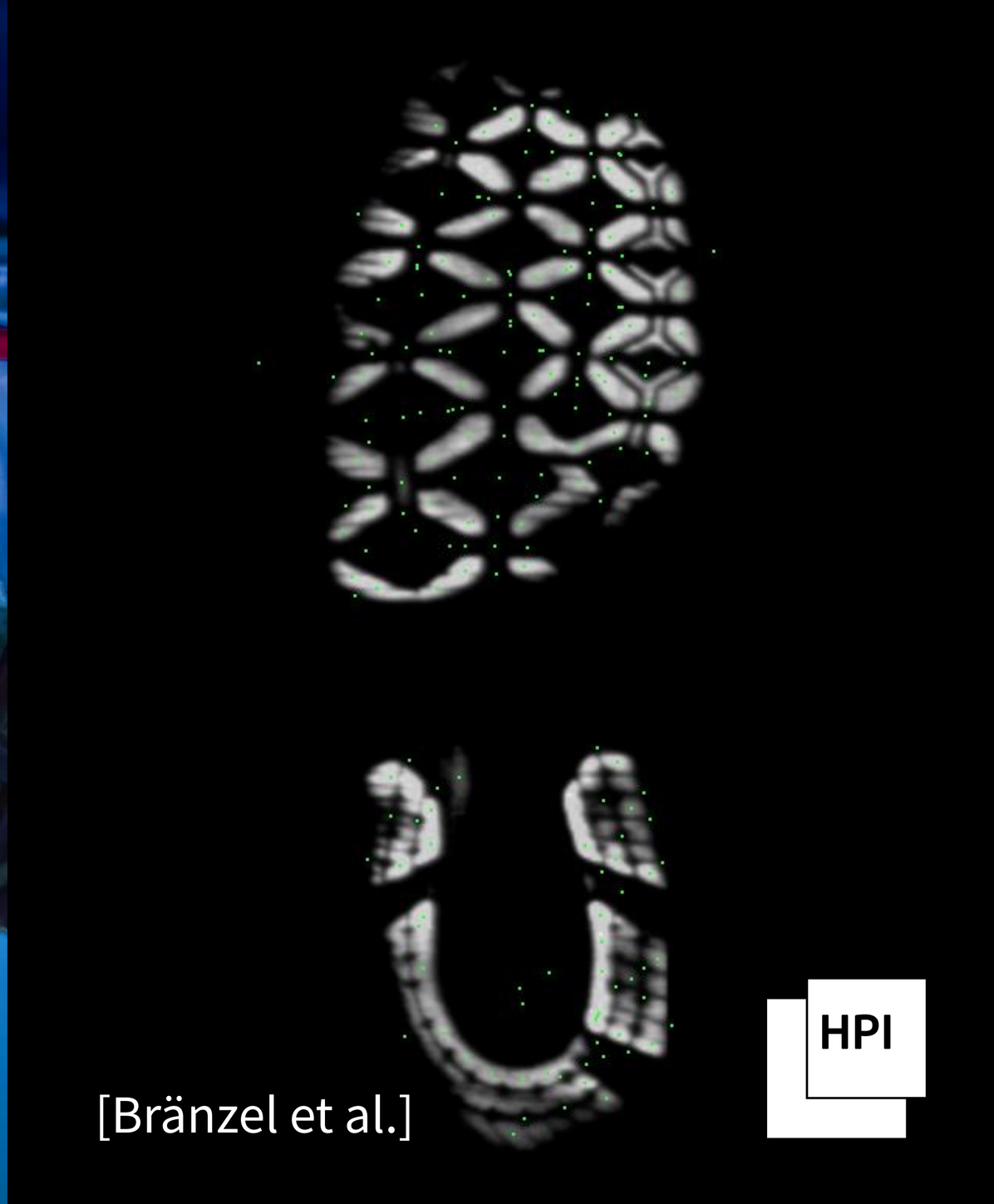
HPI



Speeded Up Robust Features

- **Object detection** in images.
- **Stitching** images.
- **Description** of images.





[Bränzel et al.]

SURF & NUMA
satellite images

Outline

I. SURF

Keypoint Extraction (our Focus):

– Wavelet Responses

– Approximation with Box-Filters

– Octaves and Scales

– Speeding up Filters with the **Integral Image**

Keypoint Description:

– Direction

– Results

Limitations

Outline

II. SURF & NUMA

Experiments:

– Time Performance

– Data Access Patterns

Implementation Proposal:

– Distributed Integral Image

– Ghost Cells within the Integral Image

Peformance Comparison:

– Single Thread vs. Multi Thread vs. Ours

Conclusion

Speeded-Up Robust Features (SURF)

Herbert Bay [a] , Andreas Ess [a] , Tinne Tuytelaars [b] , and Luc Van Gool [a,b]

[a] *ETH Zurich, BIWI*
Sternwartstrasse 7
CH-8092 Zurich
Switzerland

[b] *K. U. Leuven, ESAT-PSI*
Kasteelpark Arenberg 10
B-3001 Leuven
Belgium

Abstract

This article presents a novel scale- and rotation-invariant detector and descriptor, coined SURF (Speeded-Up Robust Features). SURF approximates or even outperforms previously proposed schemes with respect to repeatability, distinctiveness, and robustness, yet can be computed and compared much faster.

This is achieved by relying on integral images for image convolutions; by building on the strengths of the leading existing detectors and descriptors (specifically, using a Hessian matrix-based measure for the detector, and a distribution-based descriptor); and by simplifying these methods to the essential. This leads to a combination of novel detection, description, and matching steps.

The paper encompasses a detailed description of the detector and descriptor and then explores the effect of the most important parameters. We conclude the article with SURF's application to two challenging, yet converse goals: camera calibration as a special case of image registration, and object recognition. Our experiments underline SURF's usefulness in a broad range of topics in computer vision.

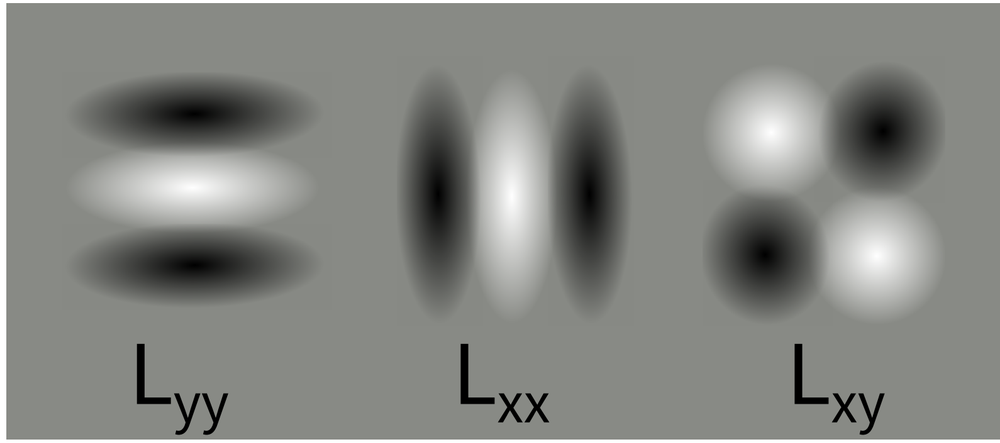
Key words: interest points, local features, feature description, camera calibration, object recognition
PACS:

1. Introduction

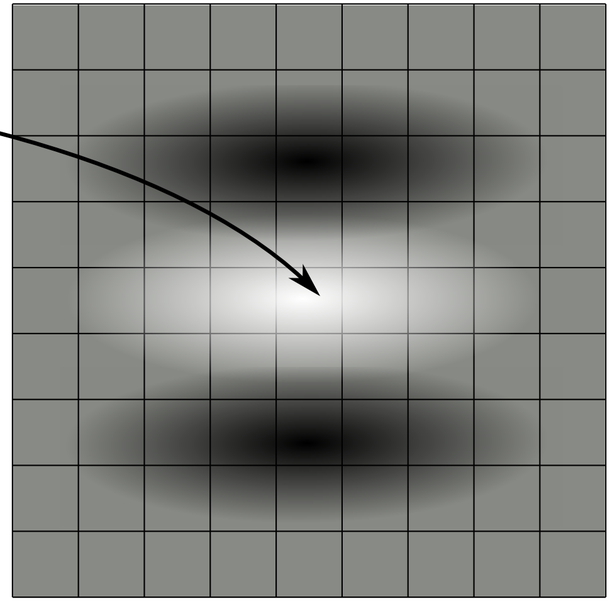
between the vectors, e.g. the Mahalanobis or Euclidean distance. The dimension of the descriptor has a direct impact

Wavelet Responses

- SURF tracks edges (≙gradient changes)
- gradient changes have high derivations in the image
- wavelets are used to calculate those derivations

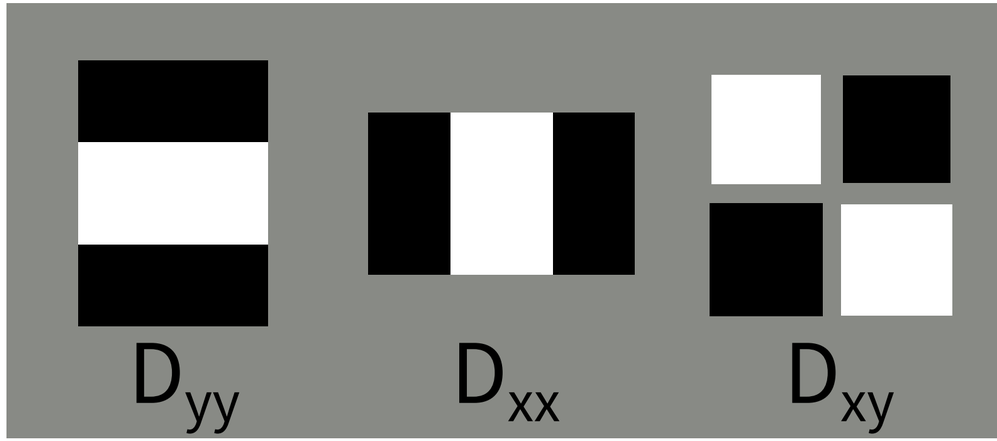


$$r_{yy} = \sum_{i,j} \text{Image}[i, j] \cdot L_{yy}[i, j]$$



Approximation with Box-Filters

– computation of wavelets is expensive
– let’s approximate them with box filters



– actually we want to compute the determinant of the Hessian
– with approximation we have to account for a bias $w \approx 0.9$

$$\mathcal{H} = \begin{bmatrix} r_{xx} & r_{xy} \\ r_{yx} & r_{yy} \end{bmatrix}$$

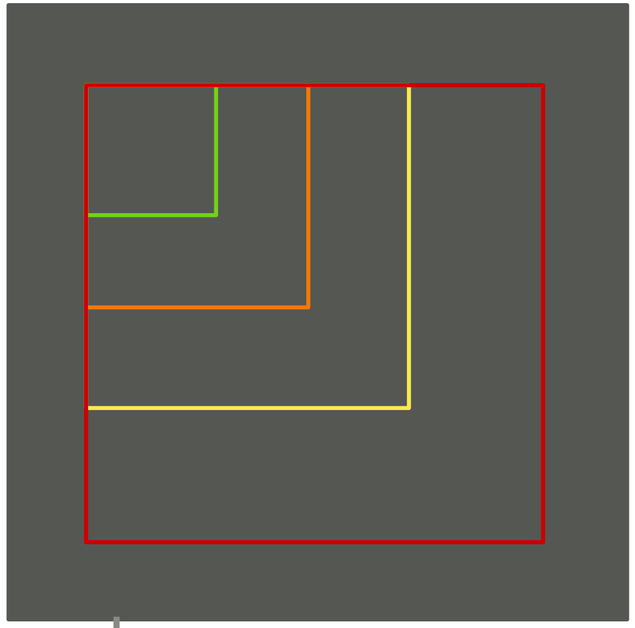
$$\det(\mathcal{H}) \approx D_{xx} \cdot D_{yy} - (w \cdot D_{xy})^2$$

Octaves and Scales

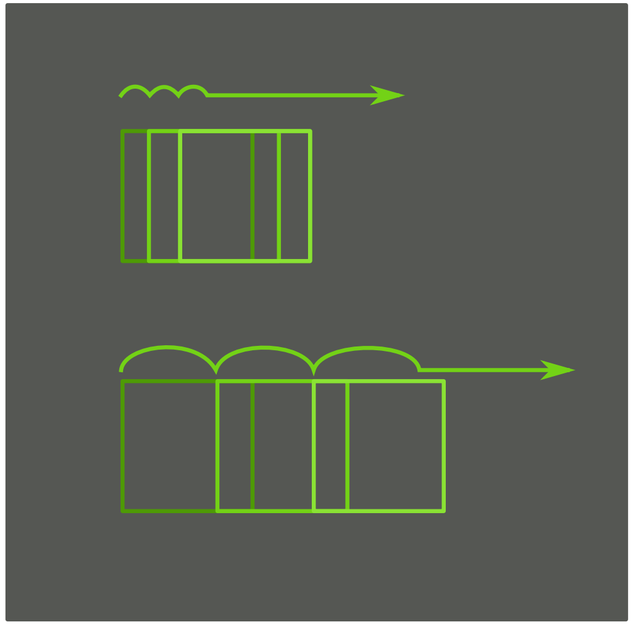
– objects can be differently sized in the image

→ let’s use different filter sizes with different step sizes

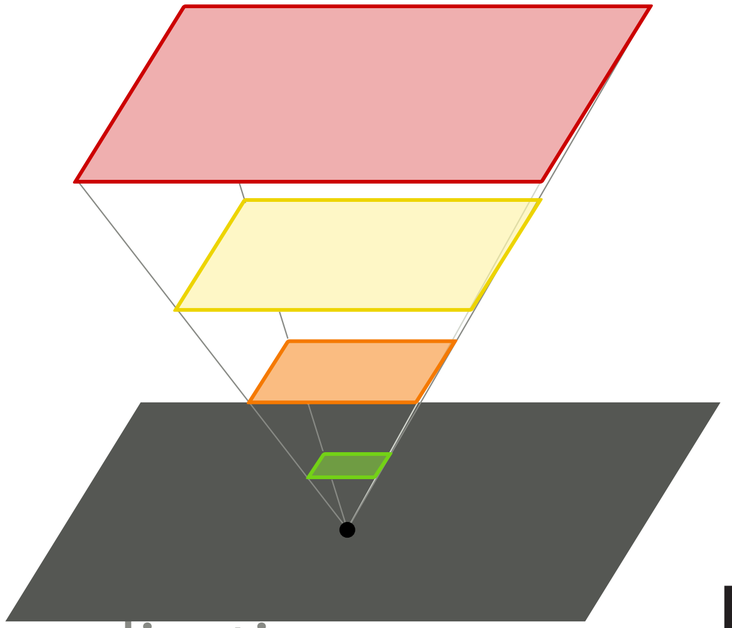
– each area is analyzed with multiple octaves and scales



scales



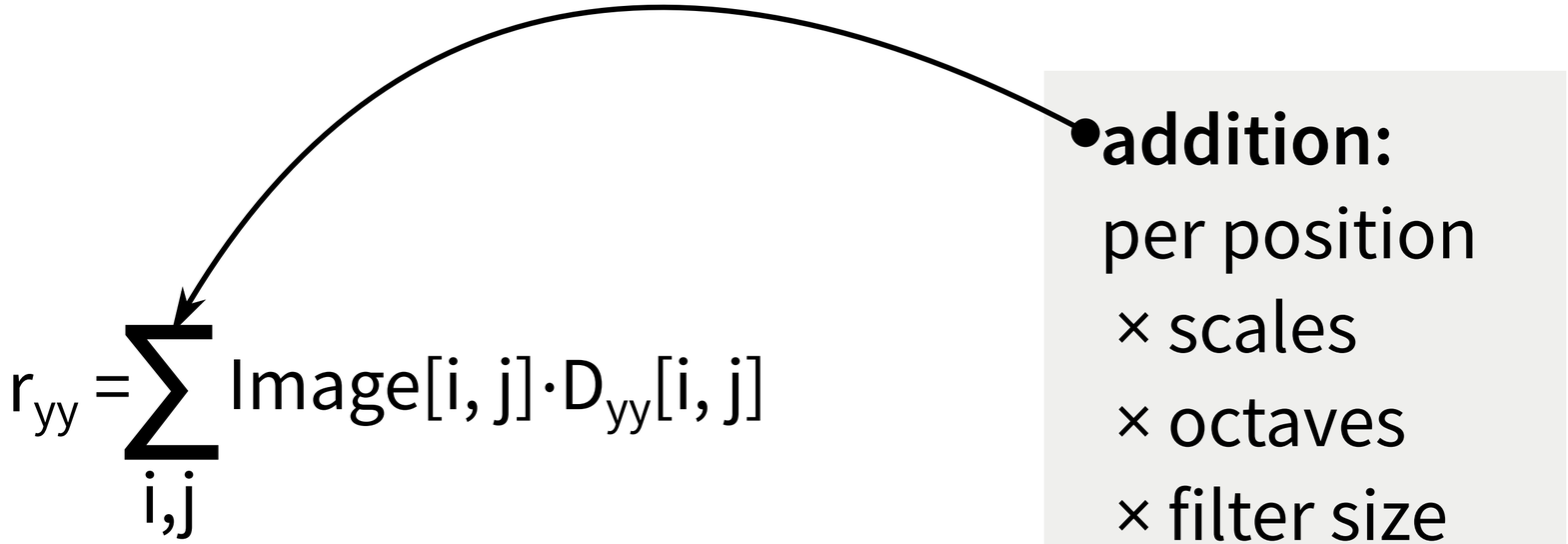
octaves



application

Speeding up Filters with the Integral Image

performance issue:



addition:
per position
× scales
× octaves
× filter size
× 3 box filters

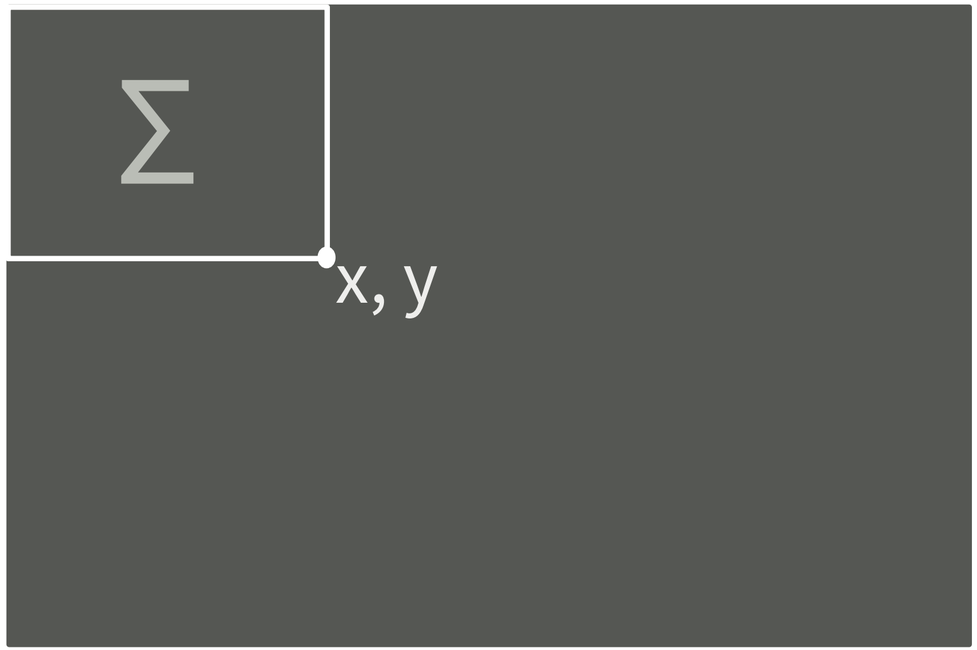
parallelsurf 0.96, naïve:

1 MByte greyscale image, just first octave

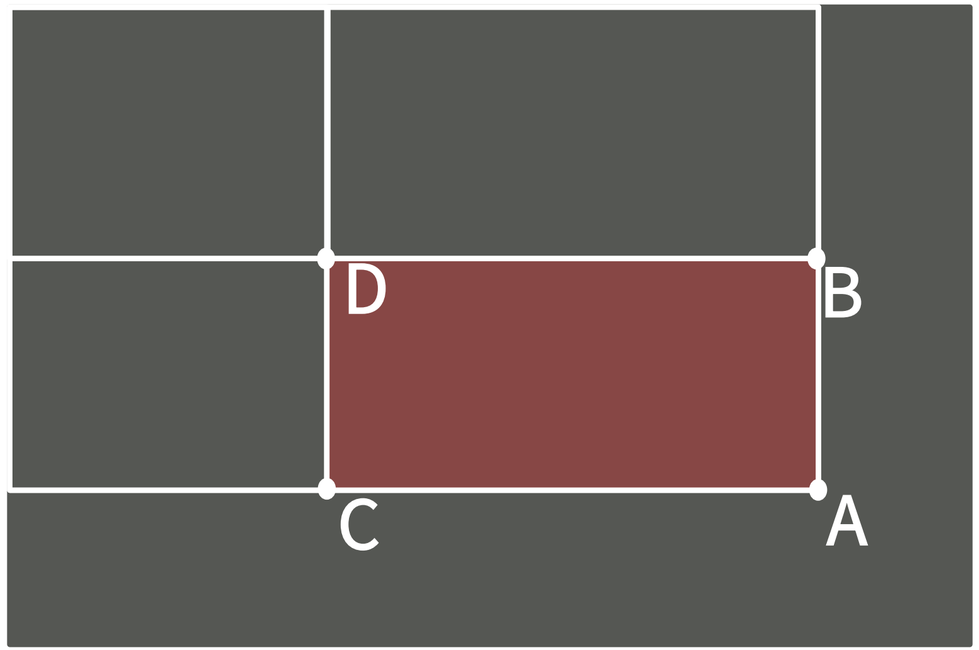
→ 7.05 GByte memaccess

The Integral Image

»Our Rescue« – Reducing memory acc. by 2 orders of magnitude



integral image



integral image

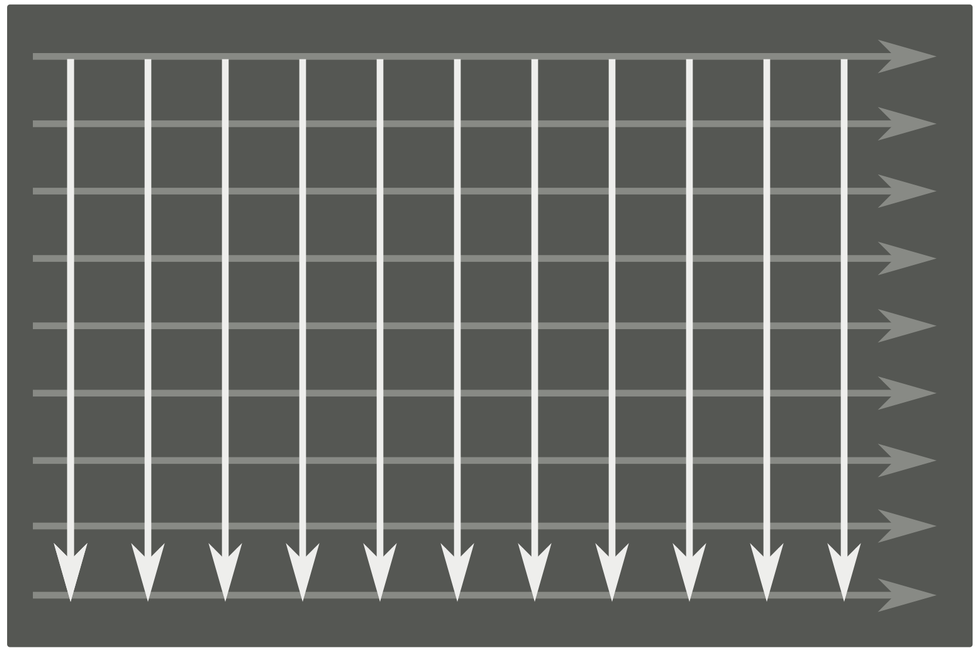
$\Sigma(\blacksquare) = \mathbf{A - B - C + D}$ (4 mem accesses)

first octave ≙ 70MB memaccess [Viola&Jones]

Computing the Integral Image

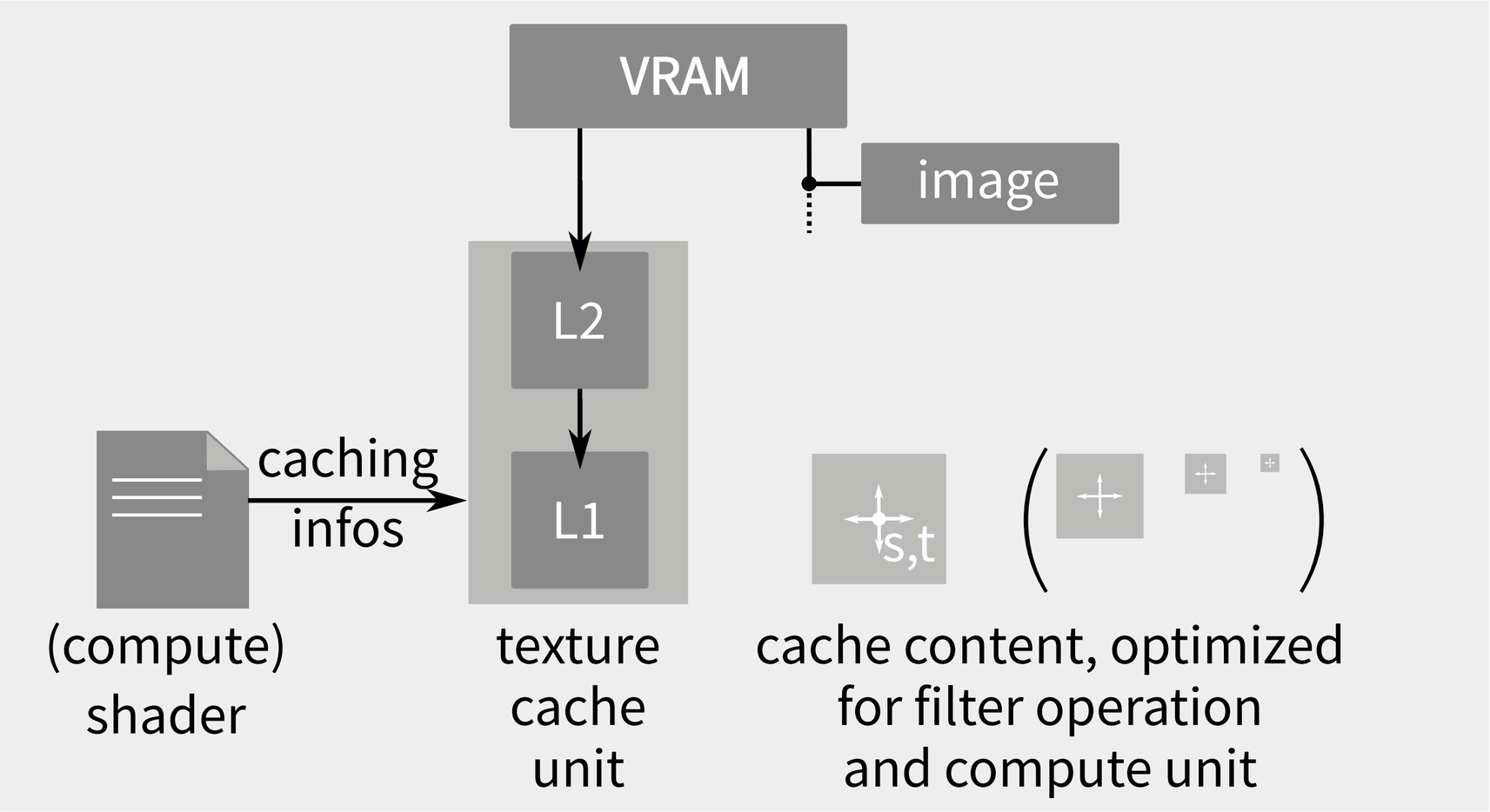
(in parallel)—Addition is commutative, associative!

embarassingly parallel
cache-friendly



embarassingly parallel
not cache-friendly (on CPUs)

Excursus: GPU Memory Caching {

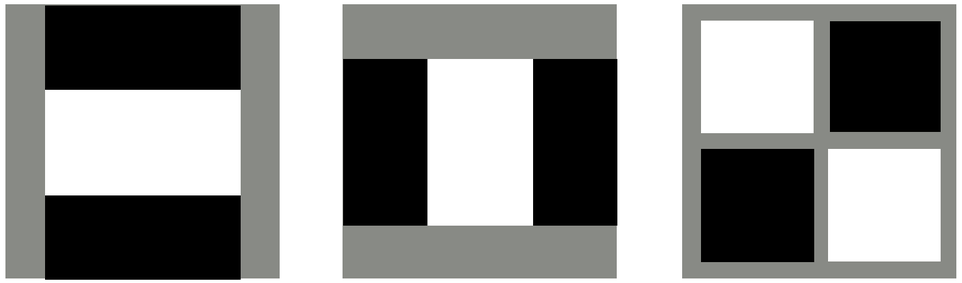


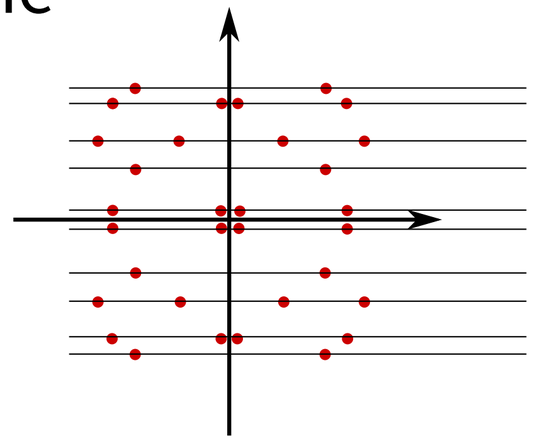
thanks to HPI3D

}

Back to CPU Caching: **Box Filters**

– it is good to compute all three filters in one pass!
→ improves cache hits in one line





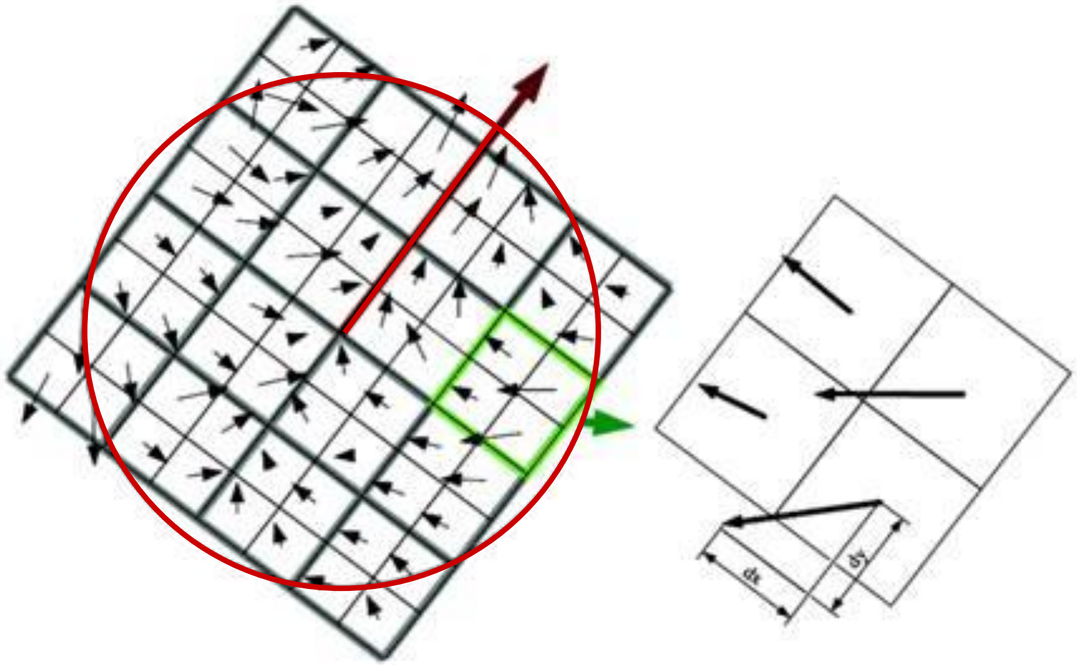
32 memory accesses
10 cache lines hit
(assuming small filter)

– implementations exist that try to also overlay access points of various filter scales!

[Terriberry et al.]

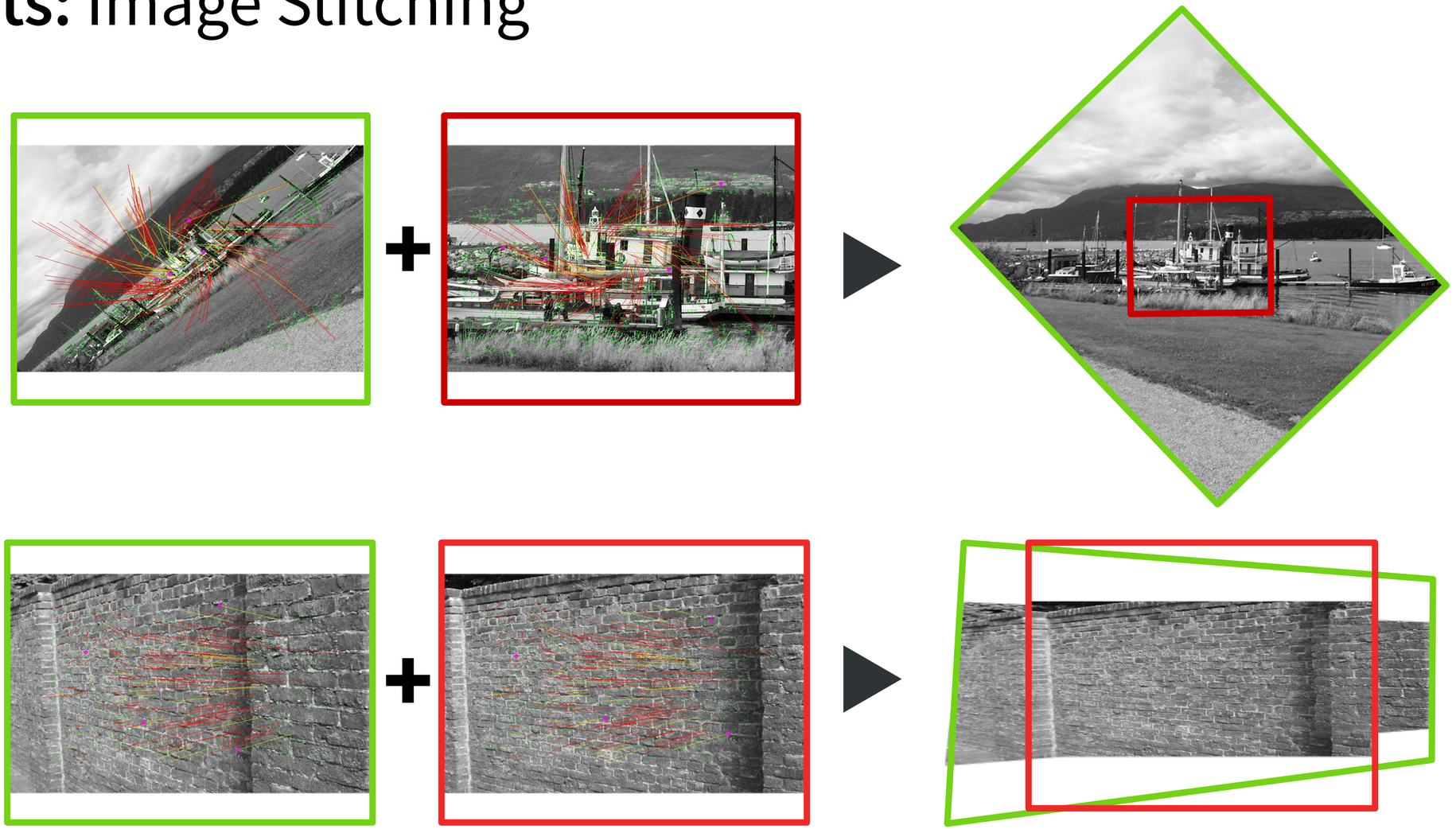
Last Step: **Feature Description**

– just features with det$(\mathcal{H})$ > threshold are processed further!
– the strongest direction is retrieved, and rotated filters are computed
– additionally, n×n sub-directions are obtained and stored as descriptor



[images: cs.wahsington.edu, docs.opencv.org]

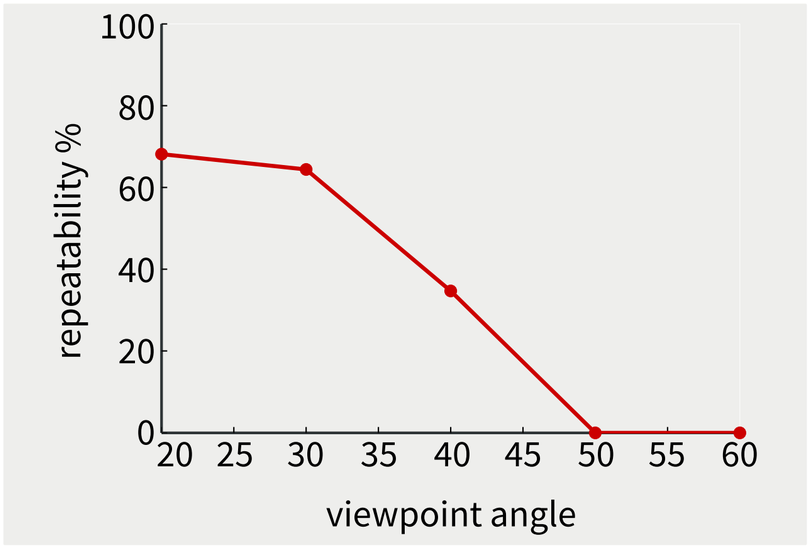
Results: Image Stitching



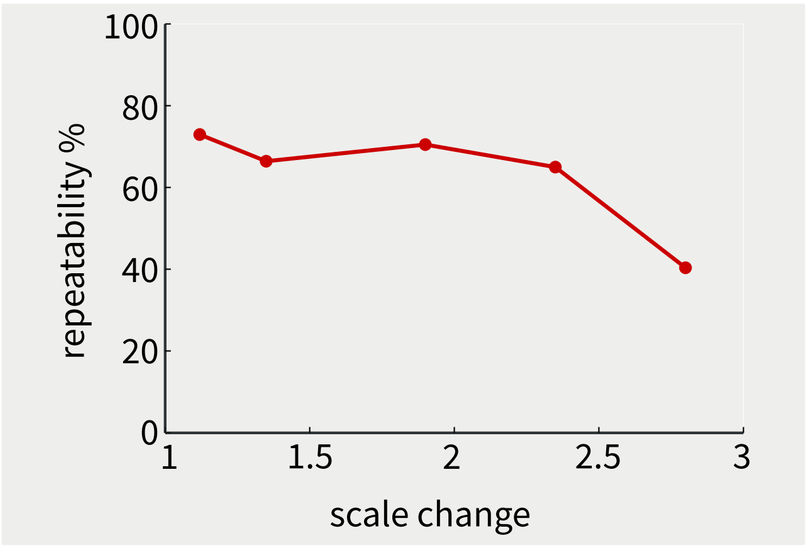
[images: TERRIBERRY et al.]

Qualitative Strengths & Limitations

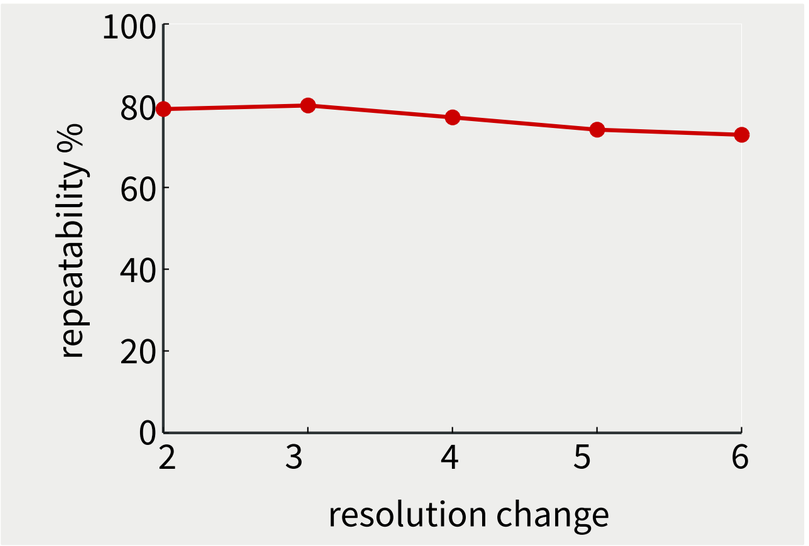
– SURF’s quality remains slightly inferior to SIFT
– rotational errors stem partly from pixel-grid combined with rotation



robustness (rotation)



robustness (scale)



robustness (resolution)

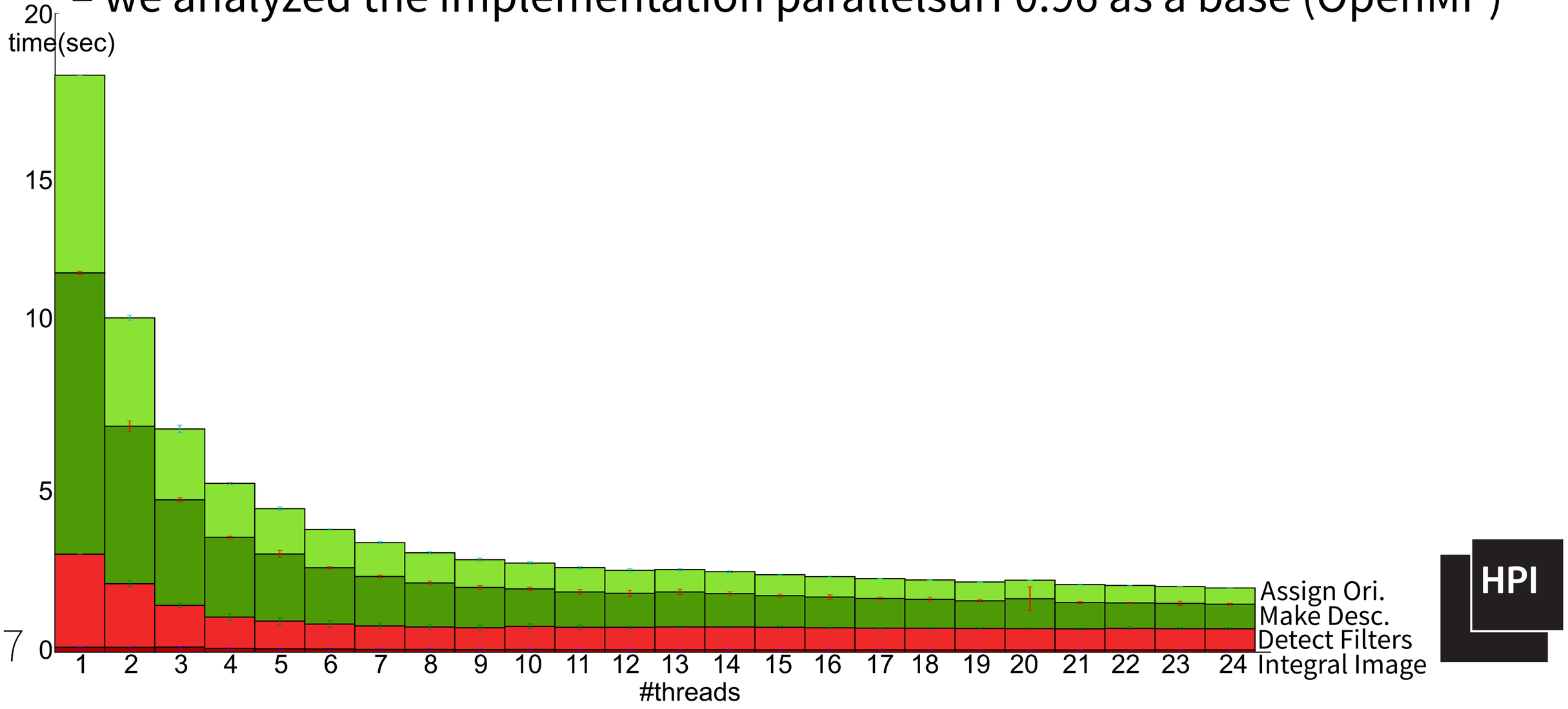
(images simplified)

[Bay et al. (SURF)]

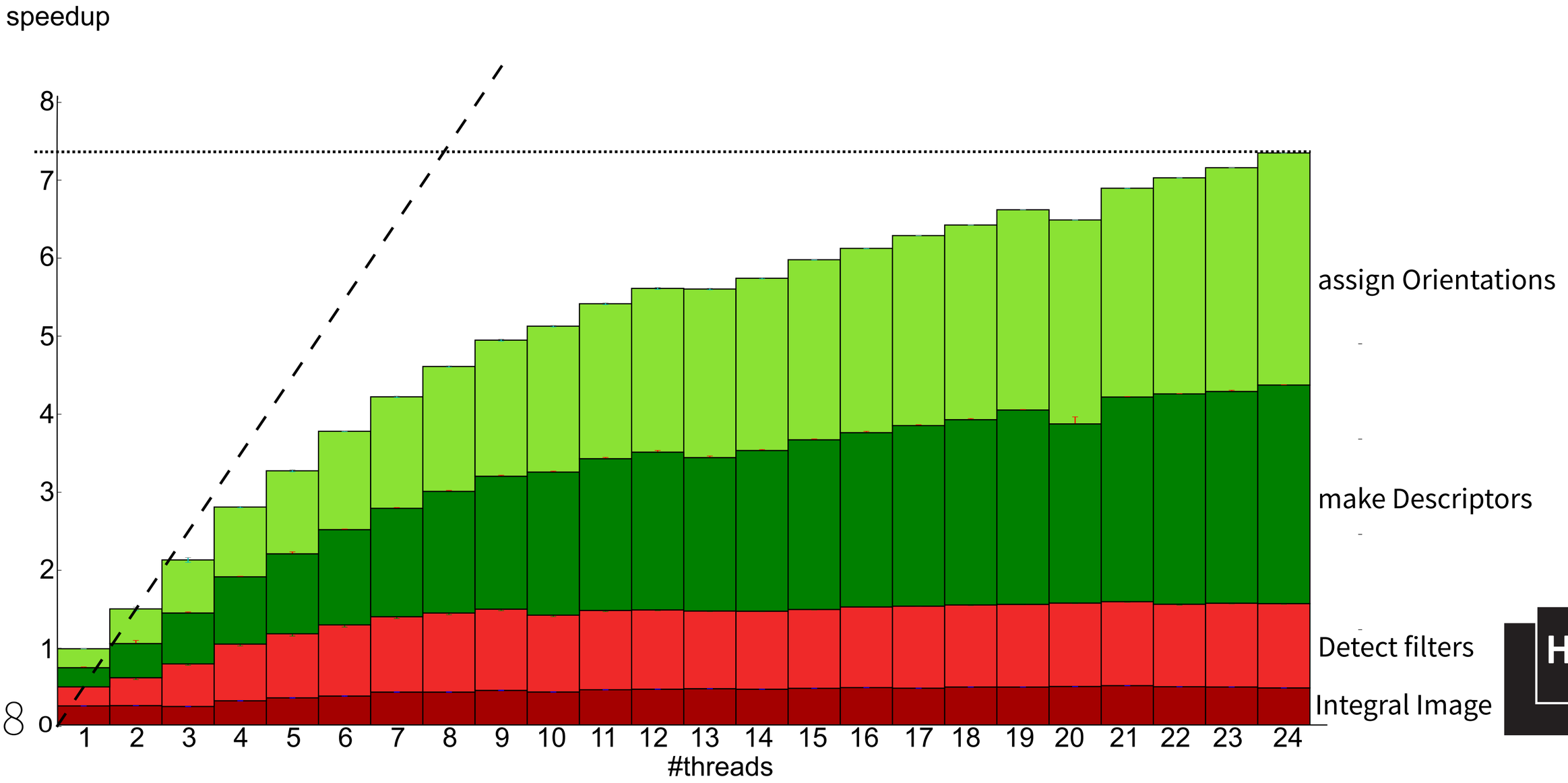
Part II: SURF & NUMA

Experiments: Time

– we analyzed the implementation parallelsurf 0.96 as a base (OpenMP)

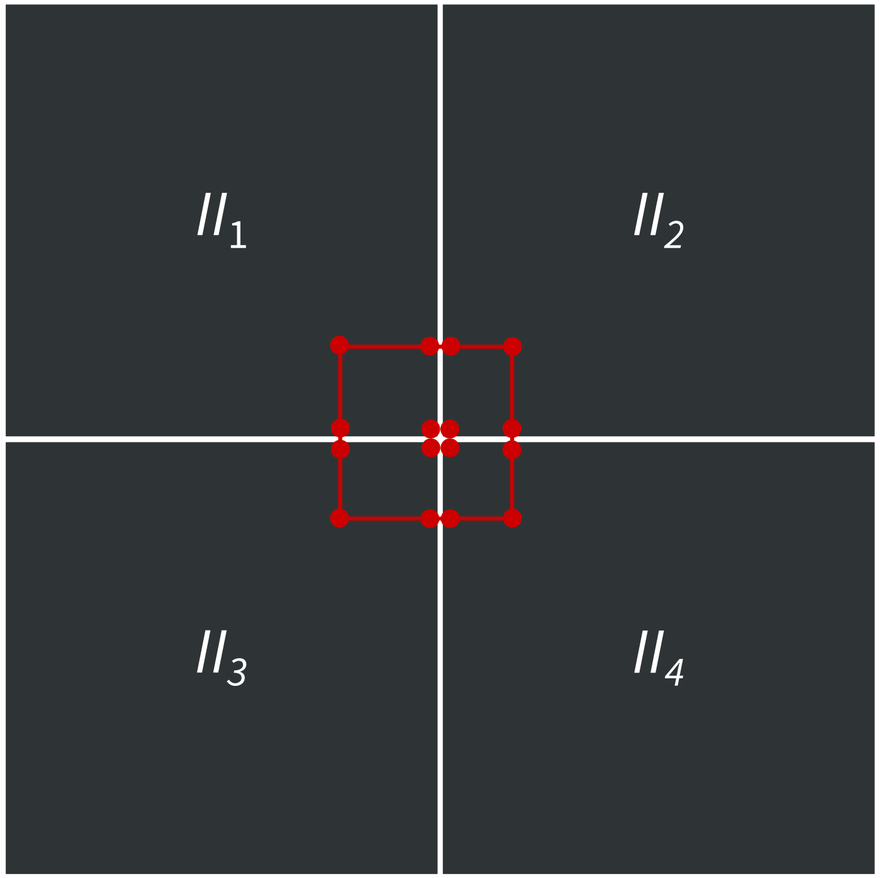


Experiments: Time (Speedup)

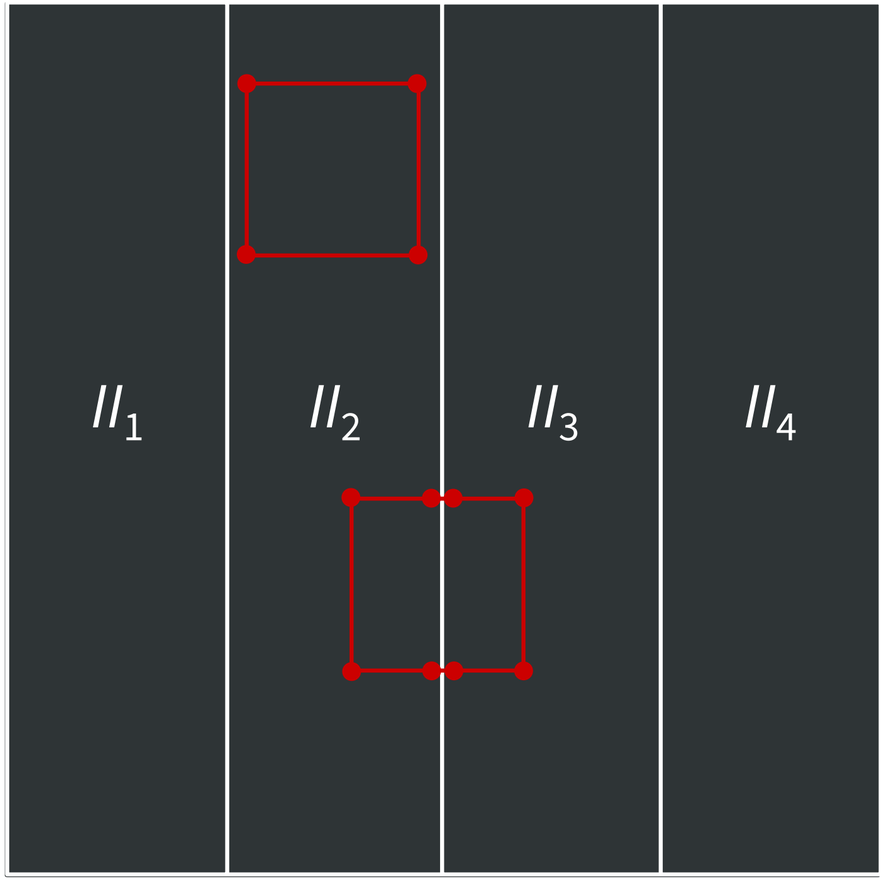


Idea: Calculate many Integral Images

– vertical is smarter if image is large *(if biggest filter < stripe)*



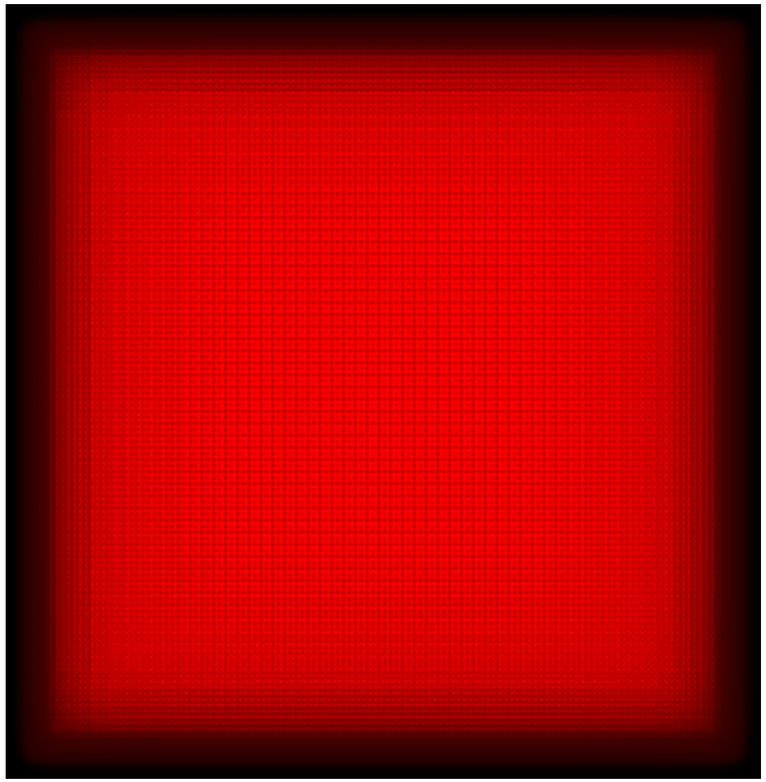
worst case: 4acc → 16acc



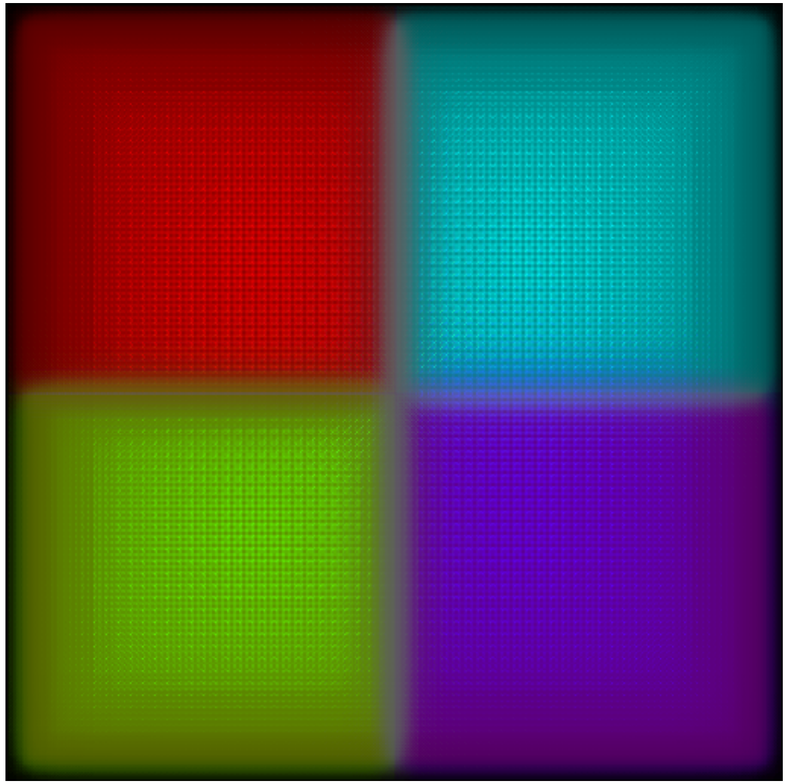
worst case: 4acc → 8acc, ‘partners’

Experiments: Memory Access

– we recorded the memory access pattern of first step (pre-thresholding)

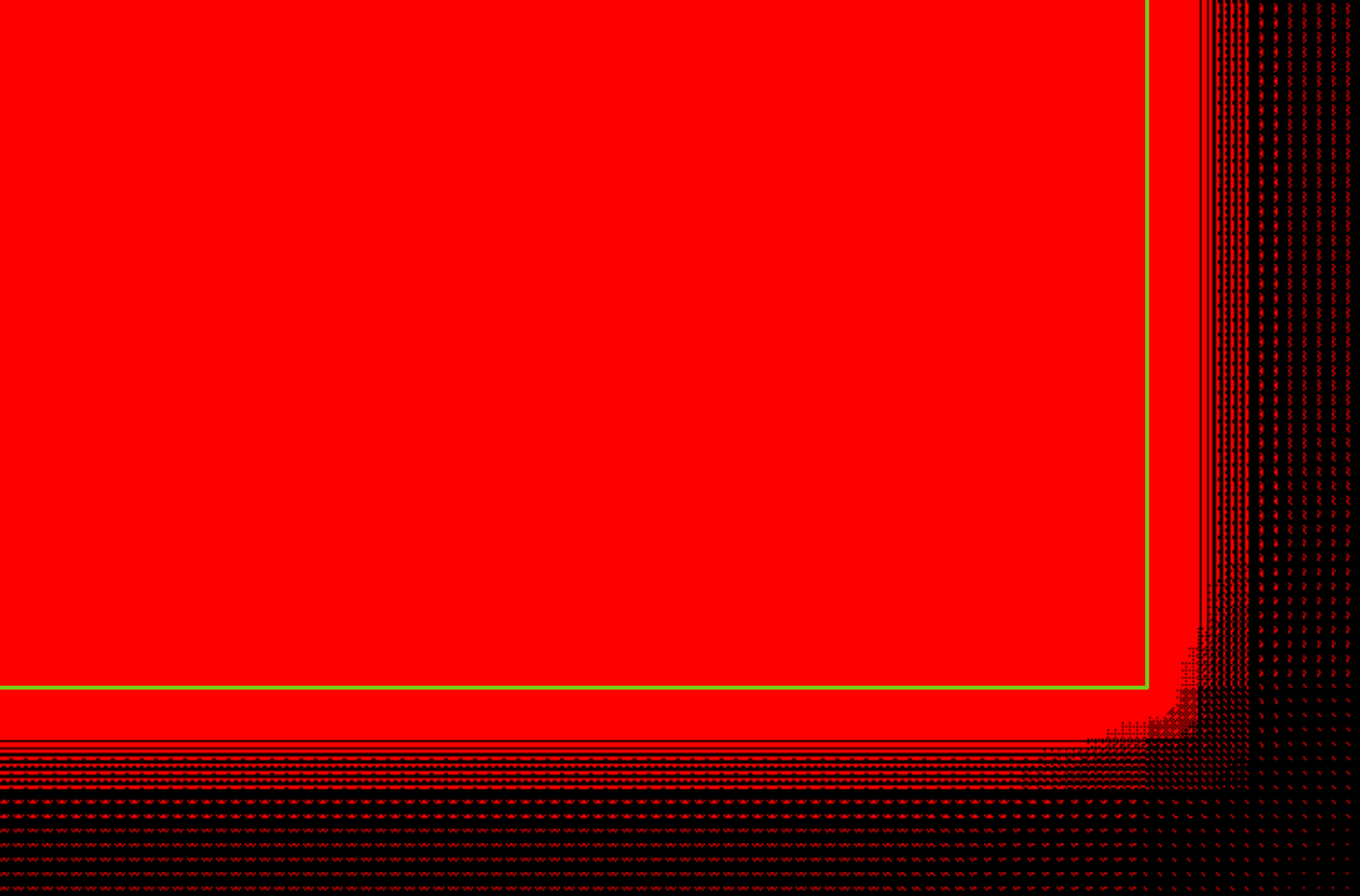


512×512, 1 part



512×512, 4 parts

(images visually enhanced)



Implementation: Algorithm & Locality

– Example: Detection

```
//Collect
FOR scales
  ALLOCATE scale_images
  FOR octaves
    #omp parallel for
    FOR filters
      FOR RANGE y
        FOR RANGE x
          scale_images[scale] ← Filter(x,y)
//Detect
FOR scales
  DetectFeatures(scale_images)
```

Implementation1: memcpy Integral-Images to all Nodes

– to test the performance of memory accesses, we consider the best scenario → every node does just local accesses

```
_ii2 = (double**) numa_alloc_onnode(
     width*height*sizeof(double),1);

if(!_ii2)
{
  std::cout << "[NUMA] Could not allocate Memory"
            << std::endl;
  return;
}
memcpy(_ii2, _ii, iWidth*iHeight*sizeof(double));
```


HPI

Implementation1: Memory Dispatch

– we once memecpy the integral image to other node(s)
– dispatch accesses based on thread locality

```
#include <utmpx.h>
#include <numa.h>

inline double **  getIntegralImage()
  {
   int cpuId = sched_getcpu();
   int nodeId = numa_node_of_cpu(cpuId);
   if(nodeId == 1) return _ii2;
   return _ii;
  }
```

slowdown!
time 10×

24 threads

HPI

buffered:
1.05×

24 threads

Side Note: Measuring Dispatch cost

– using std::chrono::high_resolution_clock

```
auto t1 = std::chrono::high_resolution_clock::now();
…
auto t2 = std::chrono::high_resolution_clock::now();
 std::cout << "Detect:"
    << std::chrono::duration_cast<std::chrono::nanoseconds>(t2-t1).count()
    << " ns"
    <<std::endl;
```

→ 79.96 µs

– called ~ 100m times. Extreme Overhead… not feasable

OMP PROC_BIND

– disallowing movement of threads between processors

→ might ensure more locality

significant speedup
of 5%

24 threads

Conclusion & Future Work

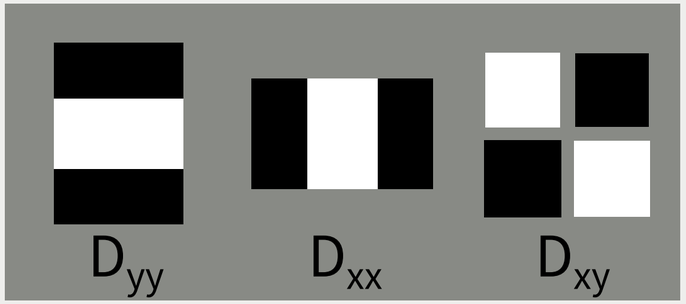
- **SURF is the *art of approximation*** applied to a mathematically complex task
- NUMA requires data locality, SURF allows for it
- parallelsurf does not respect locality *at all*

- parallelsurf already speeds up OK on NUMA machines using OMP
- memory access patterns *super-interesting* for further research
- micro-optimising OMP yields **~5%** speedup

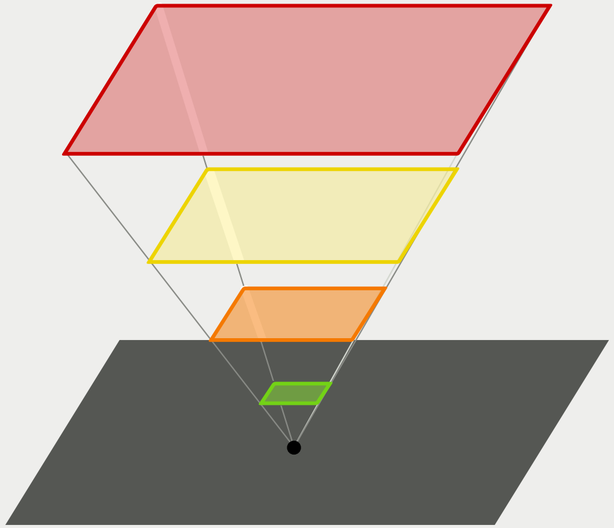
→ for further speedup full restructuring of code is needed!

Our Conclusion: Location, Location, Location!

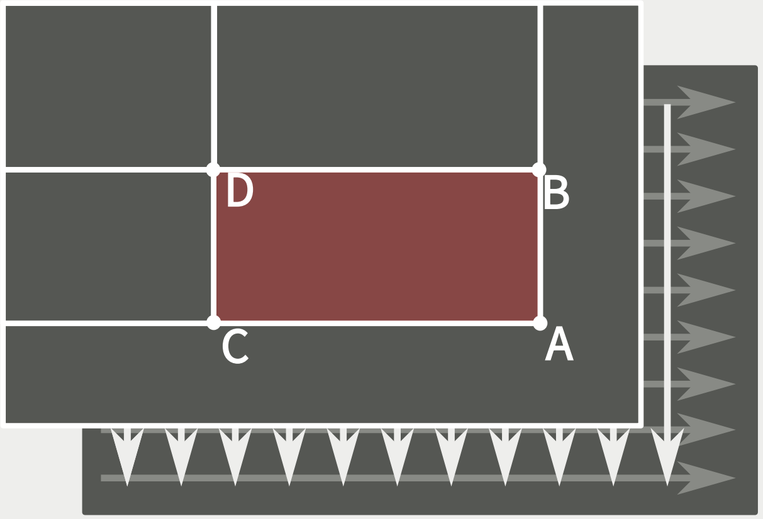
Thank you!


D_{yy}
D_{xx}
D_{xy}

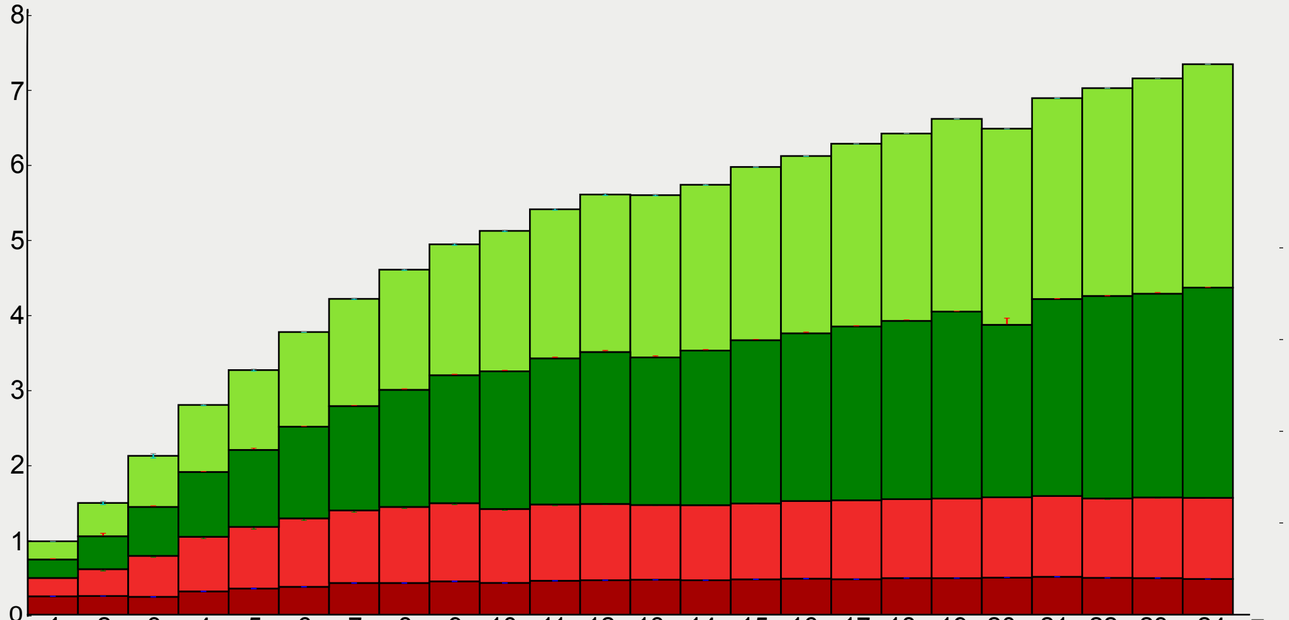
1



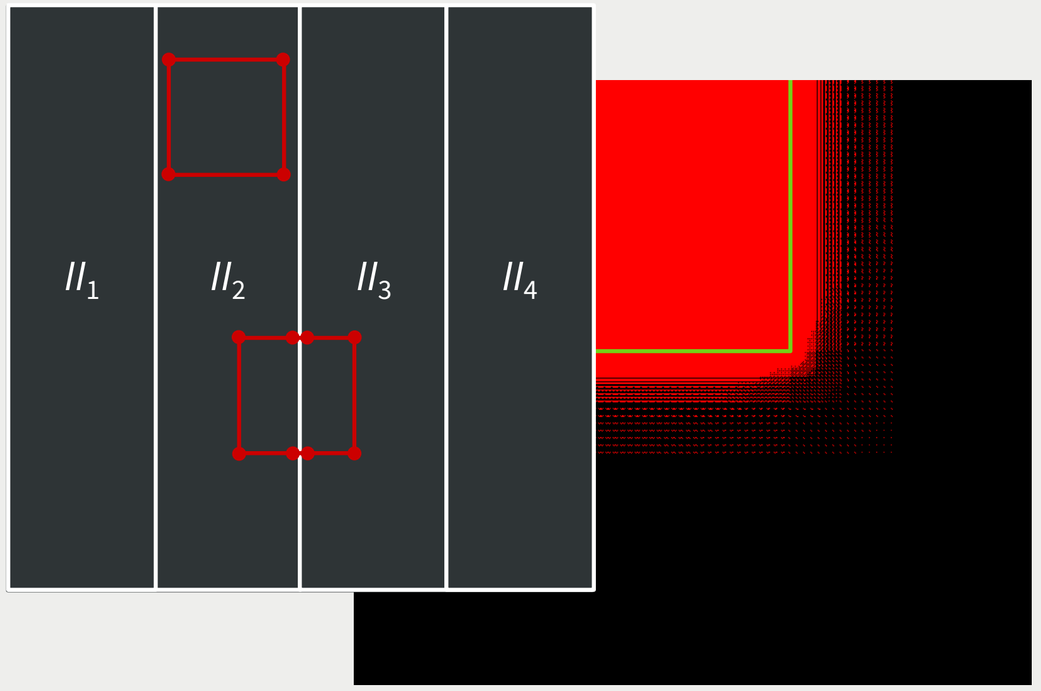
2


D
B
C
A

3


0
1
2
3
4
5
6
7
8
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24

4


II_1
II_2
II_3
II_4

5

SOURCES

[SURF paper] Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool, "SURF: Speeded Up Robust Features", Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346--359, 2008

[Viola & Jones] Viola, P.; Jones, M., "Rapid object detection using a boosted cascade of simple features," Computer Vision and Pattern Recognition, 2001. CVPR 2001.

[Bränzel et al.] Alan Bränzel, GravitySpace: tracking users and their poses in a smart room using a pressure-sensing floor. 2013. Proceedings of the SIGCHI(CHI '13).

SOURCES ctd.

[Terriberry et al.] Presentation: GPU Accelerating Speeded-Up Robust Features at Argon ST
http://people.xiph.org/ ~tterribe/pubs/gpusurf-talk.pdf, visited 02.02.15

[OpenMP] OpenMP Architecture Review Board, "OpenMP Application Program Interface, Version 3.1", July 2011. You can add "available from http://www.openmp.org

[parallelsurf] http://sourceforge.net/projects/parallelsurf/, visited 02.02.2015