Introduction to Multi-Core Programming with OpenMP

Mario Fritz

Outline

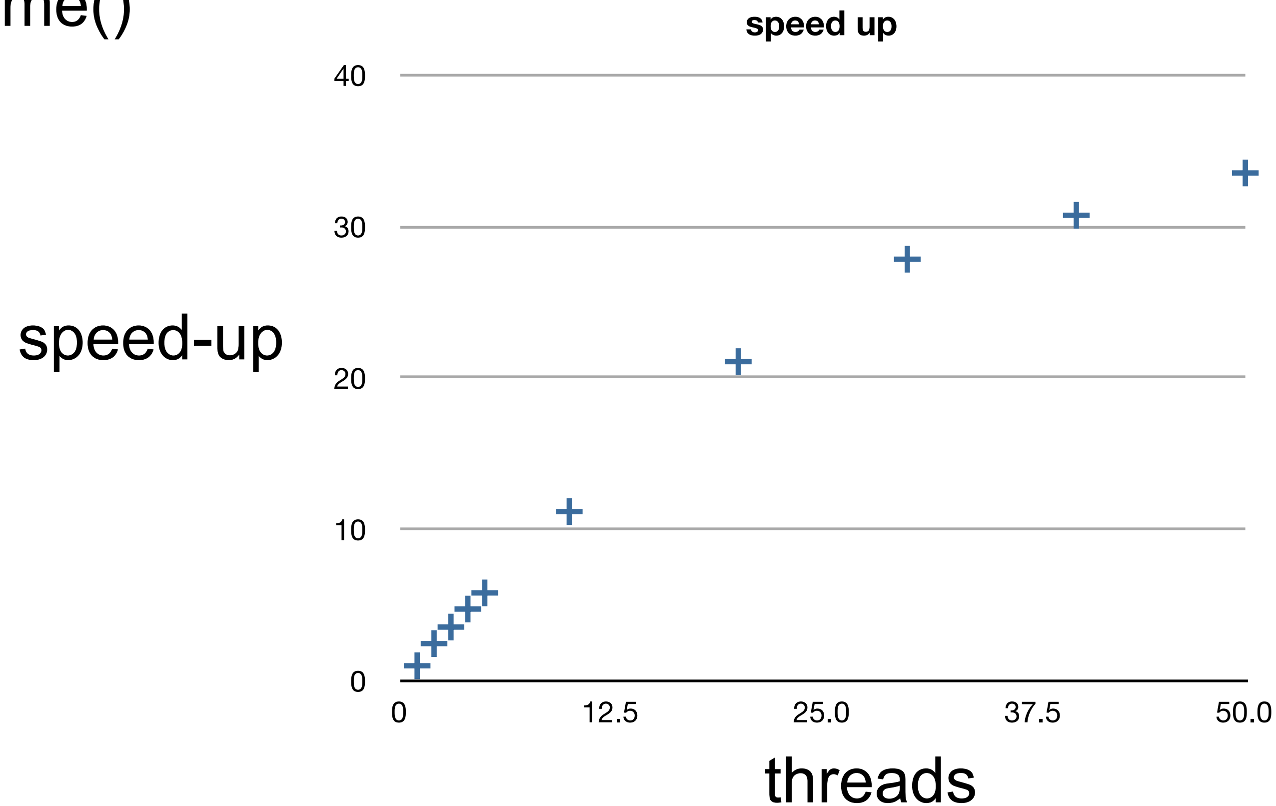
- Quick overview of multi-core parallelization
- First multi-core program
- Introduction to OpenMP
- Application: Local interest points
- Pitfalls

OpenMP

- cross-language cross platform parallelization
- little programming overhead
- not quite auto parallelization - but only comments for the compiler required
- maintains serial code
- many parameters can be changed at runtime

Simple Demo

- add #pragma omp parallel for
- -fopenmp
- export OMP_NUM_THREADS=10
- omp_get_wtime()



Simple Demo

- "#pragma omp parallel for" works for
  - signed integer loops
  - simple comparisons with loop invariant integers
  - 3.expression invariant increment/decrement
  - single entry / single exit loop

Private vs Shared Variables

- default:
  - all variables are shared among processes
  - expect: index variable in “parallel for” (if declared there)

- otherwise has to be specified manually e.g.:
  - #pragma omp parallel for private (x, y, z)

Some Special Commands

Function Name	Description
int omp_get_num_threads (void);	Returns the number of threads currently in use. If called outside a parallel region, this function will return 1.
int omp_set_num_threads (int NumThreads);	This function sets the number of threads that will be used when entering a parallel section. It overrides the OMP_NUM_THREADS environment variable.
int omp_get_thread_num (void);	Returns the current thread number between 0 (master thread) and total number of threads - 1.
int omp_get_num_procs (void);	Returns the number of available cores (or processors). A core or processor with Hyper-Threading Technology enabled will count as two cores (or two processors).

Some Environment Variables

Environment Variable	Description	Example
OMP_SCHEDULE	Controls the scheduling of the for-loop work-sharing construct.	set OMP_SCHEDULE = “guided, 2”
OMP_NUM_THREADS	Sets the default number of threads. The omp_set_num_threads() function call can override this value.	set OMP_NUM_THREADS = 4

Local Interest Points

Applications of Local Invariant Features

- Wide baseline stereo
- Motion tracking
- Panoramas
- Mobile robot navigation
- 3D reconstruction
- Recognition
  - Specific objects
  - Textures
  - Categories
- …

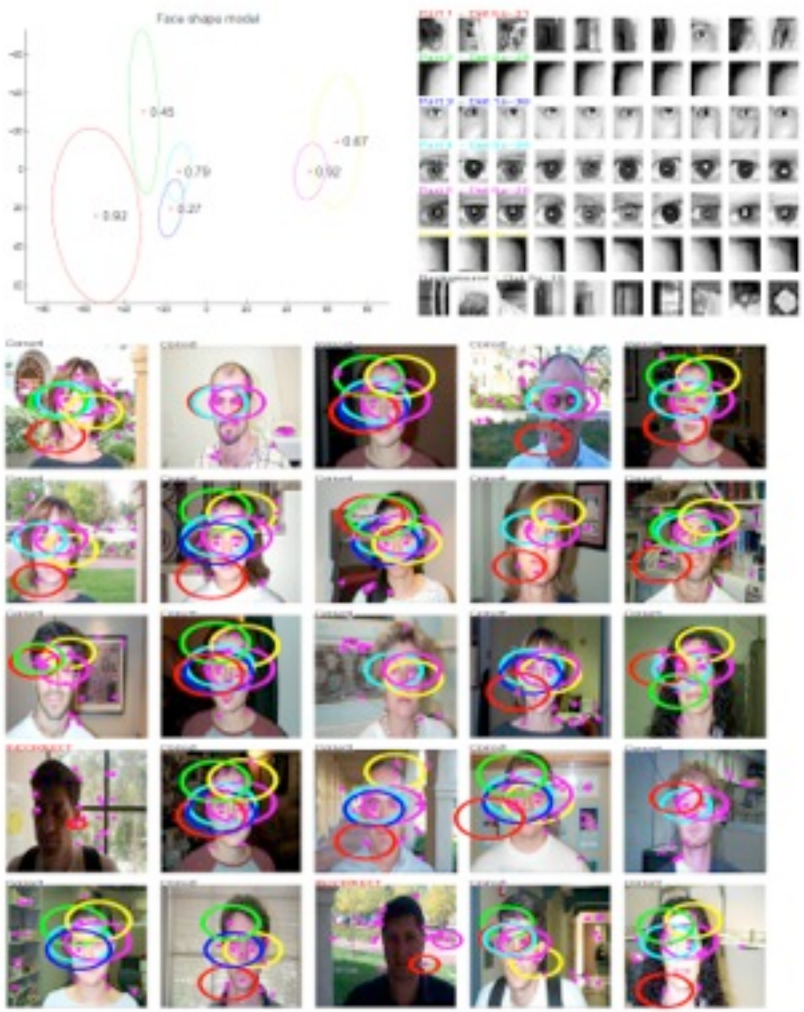
Slide credit: Kristen Grauman

Wide-Baseline Stereo

Image from T. Tuytelaars ECCV 2006 tutorial

Recognition of Categories

Constellation model



Weber et al. (2000)
Fergus et al. (2003)

Bags of words

Csurka et al. (2004)
Dorko & Schmid (2005)
Sivic et al. (2005)
Lazebnik et al. (2006), …

Application of Point Correspondence: Image Matching

by [Diva Sian]

by [swashford]

Slide credit: Steve Seitz

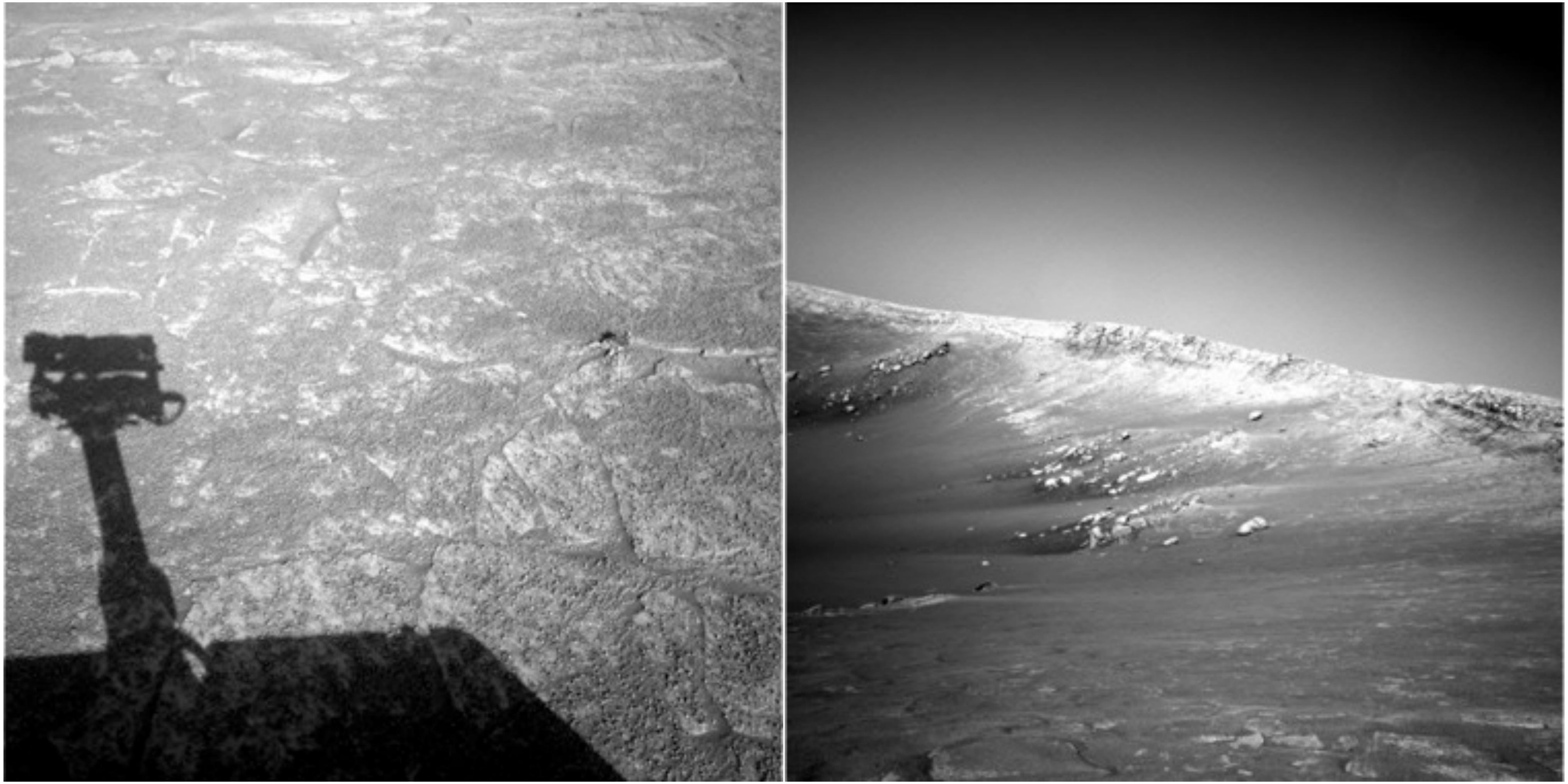
Harder Case

by Diva Sian

by scgbt

Slide credit: Steve Seitz

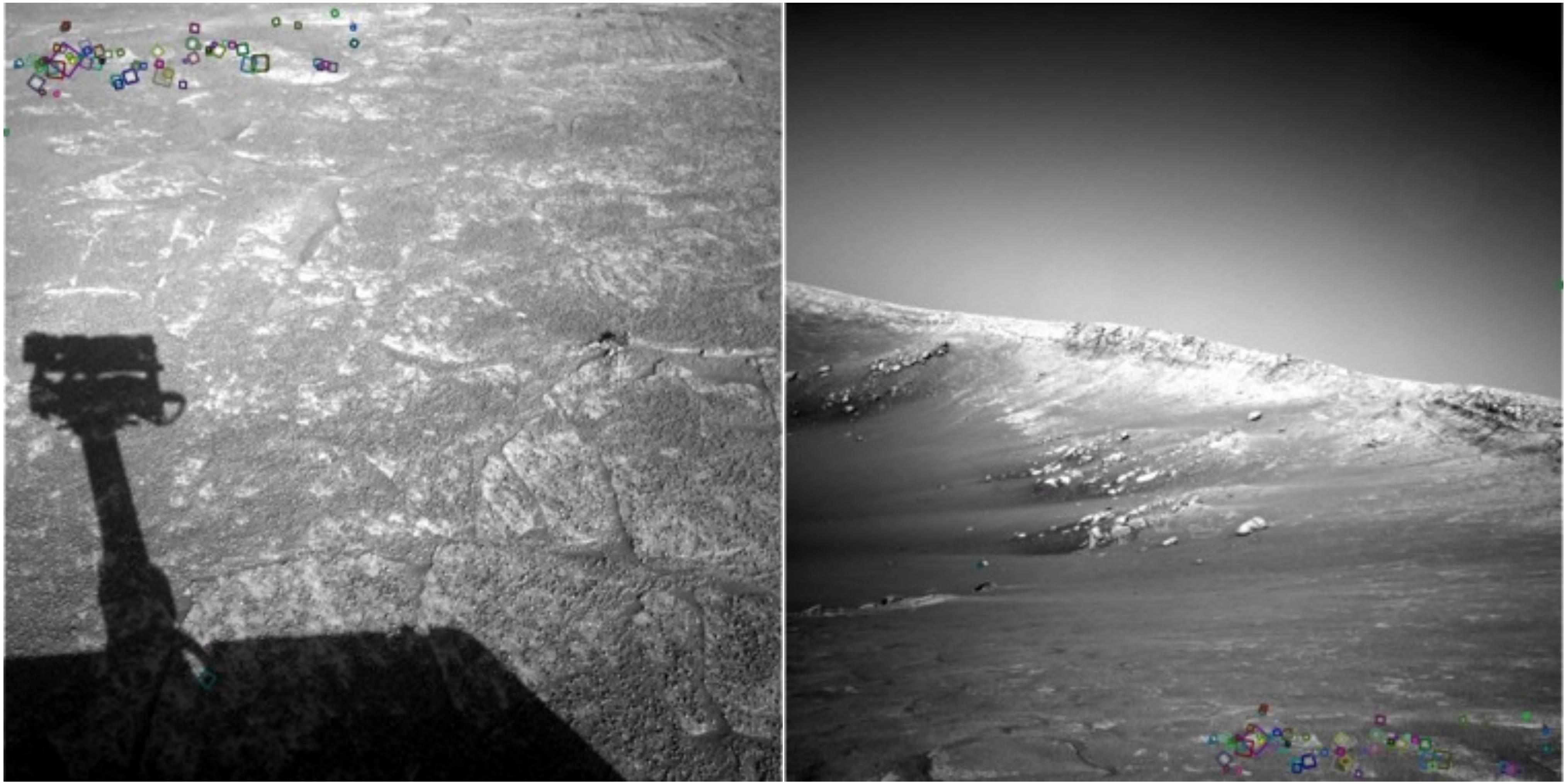
Harder Still?



NASA Mars Rover images

Slide credit: Steve Seitz

Answer Below (Look for tiny colored squares)



NASA Mars Rover images with SIFT feature matches
(Figure by Noah Snavely)

Slide credit: Steve Seitz

Application: Image Stitching

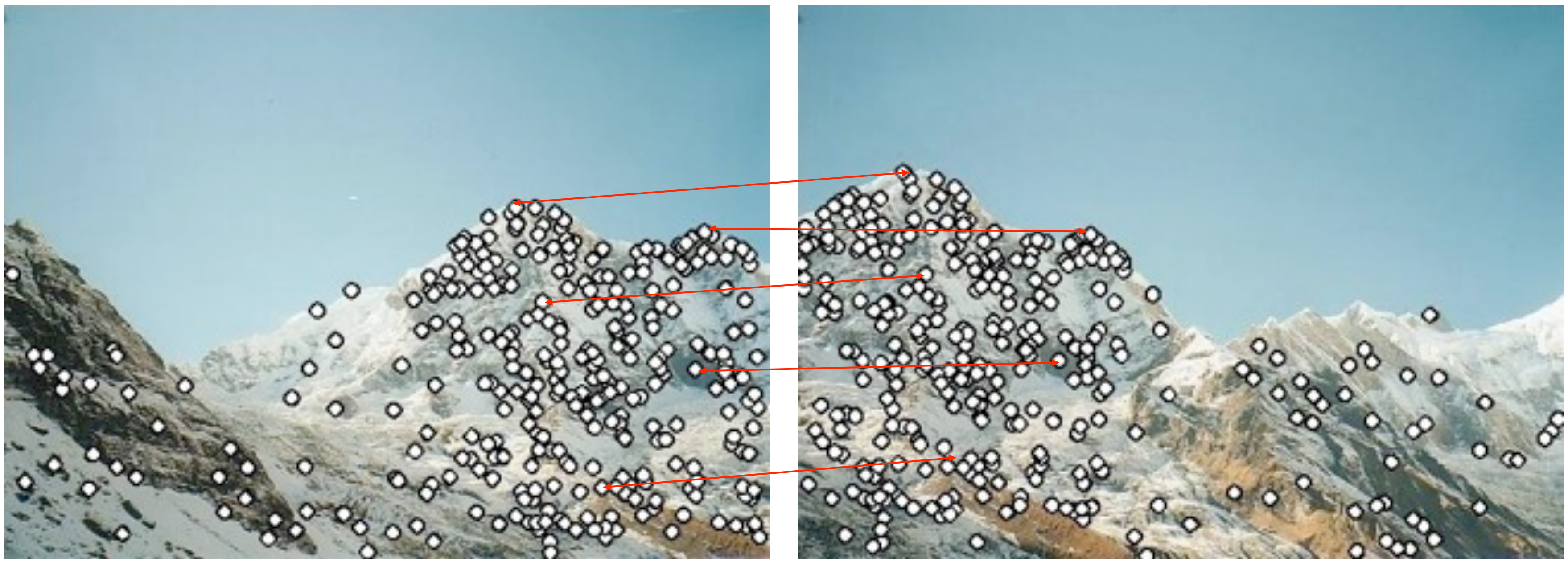
Slide credit: Darya Frolova, Denis Simakov

Application: Image Stitching


- Procedure:
  - Detect feature points in both images

Slide credit: Darya Frolova, Denis Simakov

Application: Image Stitching



- Procedure:
  - Detect feature points in both images
  - Find corresponding pairs

Slide credit: Darya Frolova, Denis Simakov

Application: Image Stitching

- Procedure:
  - Detect feature points in both images
  - Find corresponding pairs
  - Use these pairs to align the images

Slide credit: Darya Frolova, Denis Simakov

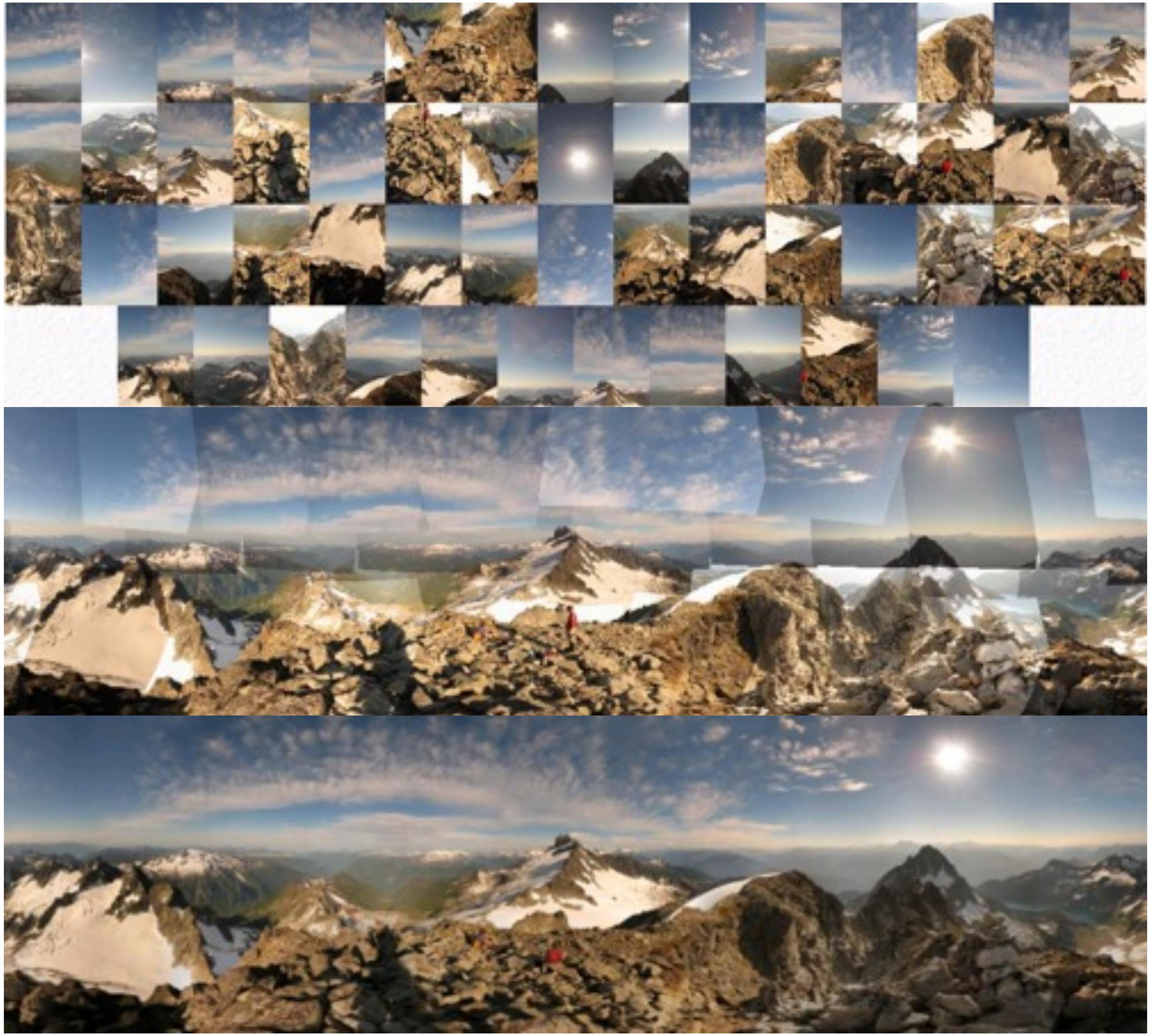
Application: Image Stitching

- Procedure:
  - Detect feature points in both images
  - Find corresponding pairs
  - Use these pairs to align the images

Slide credit: Darya Frolova, Denis Simakov

Automatic Mosaicing

[Brown & Lowe, ICCV'03]



Panorama Stitching

[Brown, Szeliski, and Winder, 2005]

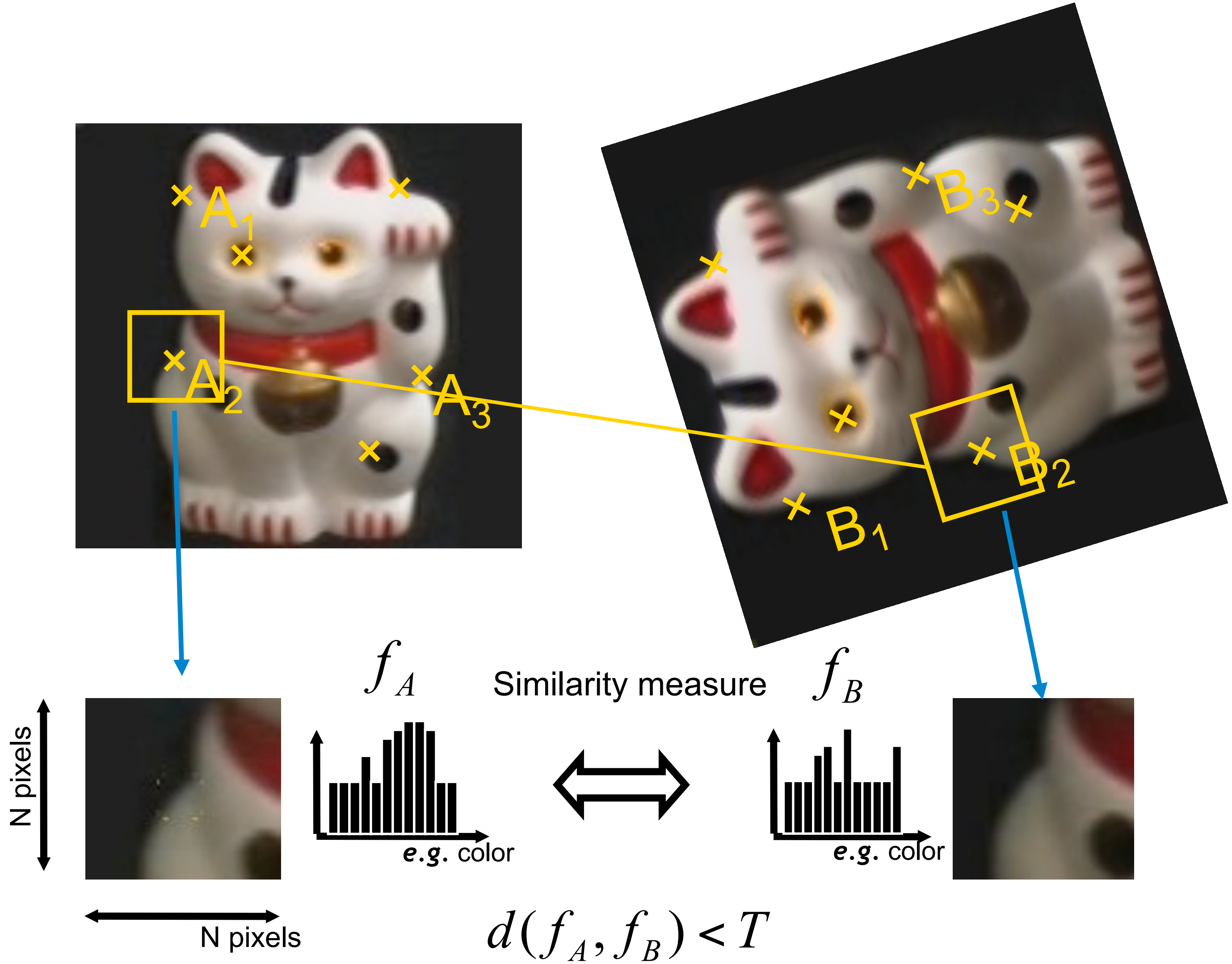
(a) Matier data set (7 images)

(b) Matier final stitch

http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html

iPhone version available

Point Correspondence for Object Instance Recognition: General Approach



1. Find a set of distinctive key-points
2. Extract and normalize the region content
3. Compute a local descriptor from the normalized region
4. Match local descriptors

Recognition of Specific Objects, Scenes

Schmid and Mohr 1997

Sivic and Zisserman, 2003

Rothganger et al. 2003

Lowe 2002

1. Interest Point Detection
Common Requirements

- Problem 1:
  - Detect the same point independently in both images

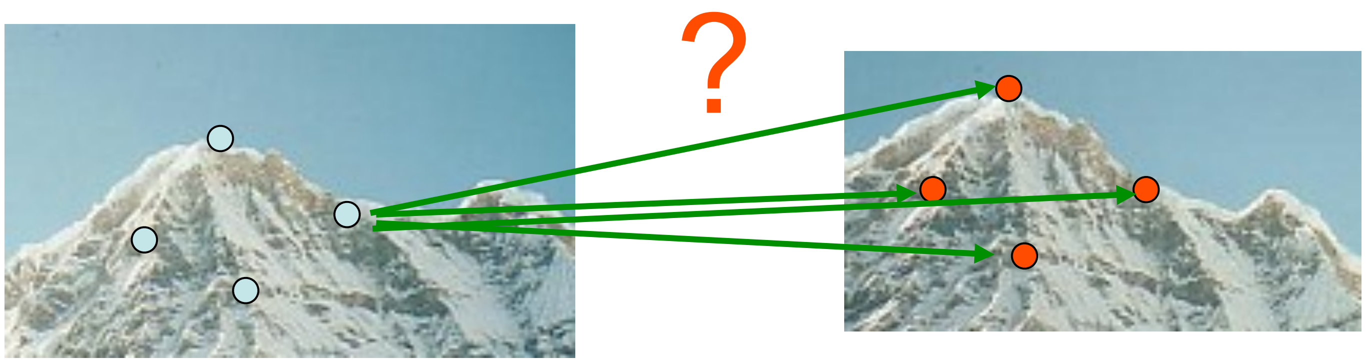
No chance to match!

We need a repeatable detector!

Slide credit: Darya Frolova, Denis Simakov

1. Interest Point Detection
Common Requirements

- Problem 1:
  - Detect the same point independently in both images
- Problem 2:
  - For each point correctly recognize the corresponding one



We need a reliable and distinctive descriptor!

Slide credit: Darya Frolova, Denis Simakov

1. Interest Point Detection Requirements

- Region extraction needs to be repeatable and accurate
  - Invariant to translation, rotation, scale changes
  - Robust or covariant to out-of-plane (≈affine) transformations
  - Robust to lighting variations, noise, blur, quantization
- Locality: Features are local, therefore robust to occlusion and clutter.
- Quantity: We need a sufficient number of regions to cover the object.
- Distinctiveness: The regions should contain “interesting” structure.
- Efficiency: Close to real-time performance.

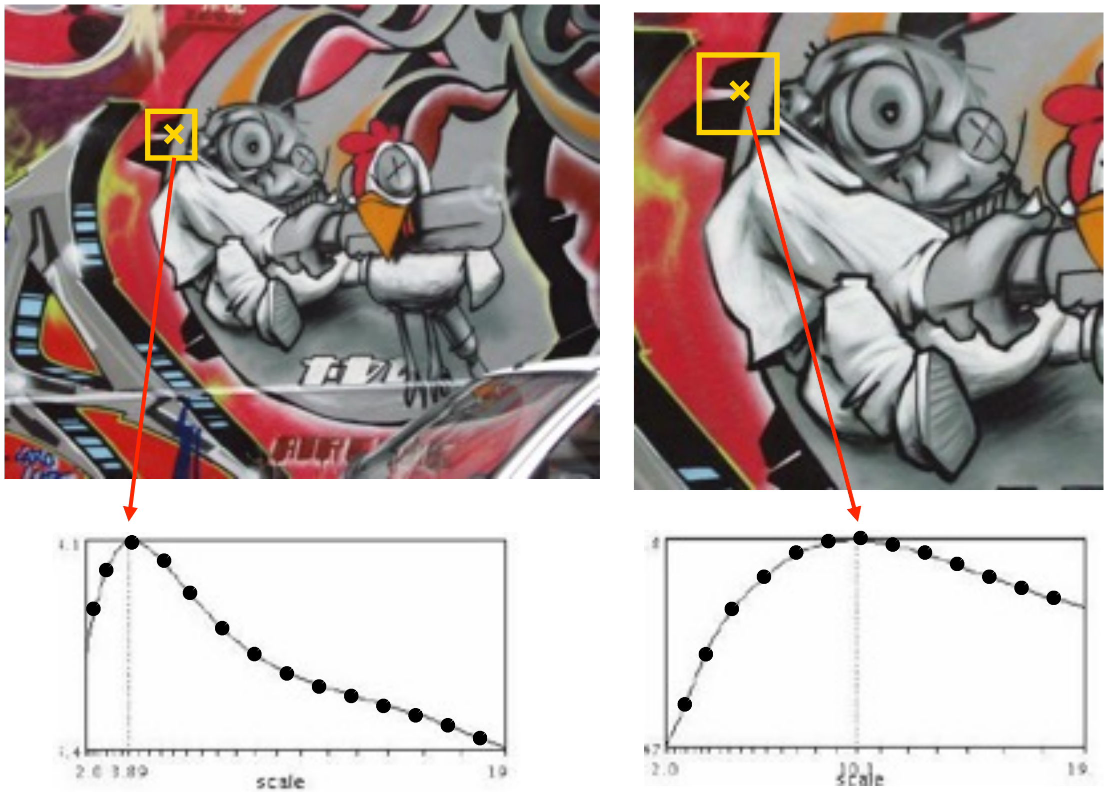
1. Interest Point Detection
Many Existing Detectors Available

- Hessian & Harris [Beaudet '78], [Harris '88]
- Laplacian, DoG [Lindeberg '98], [Lowe '99]
- Harris-/Hessian-Laplace [Mikolajczyk & Schmid '01]
- Harris-/Hessian-Affine [Mikolajczyk & Schmid '04]
- EBR and IBR [Tuytelaars & Van Gool '04]
- MSER [Matas '02]
- Salient Regions [Kadir & Brady '01]
- Others…

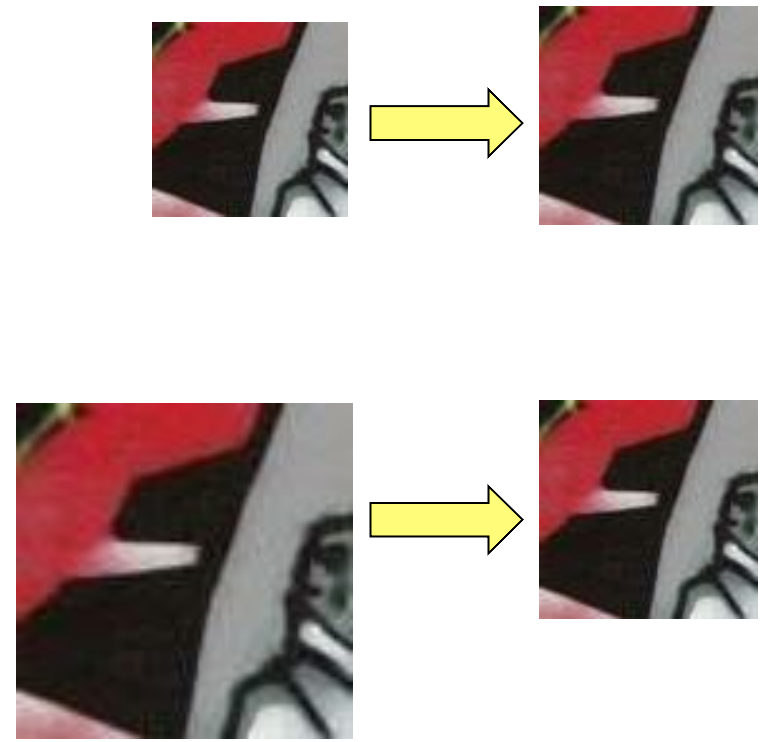
- ***Those detectors have become a basic building block for many recent applications in Computer Vision.***

1. Interest Point Detection
Automatic Scale Selection

- Normalize: Rescale to fixed size

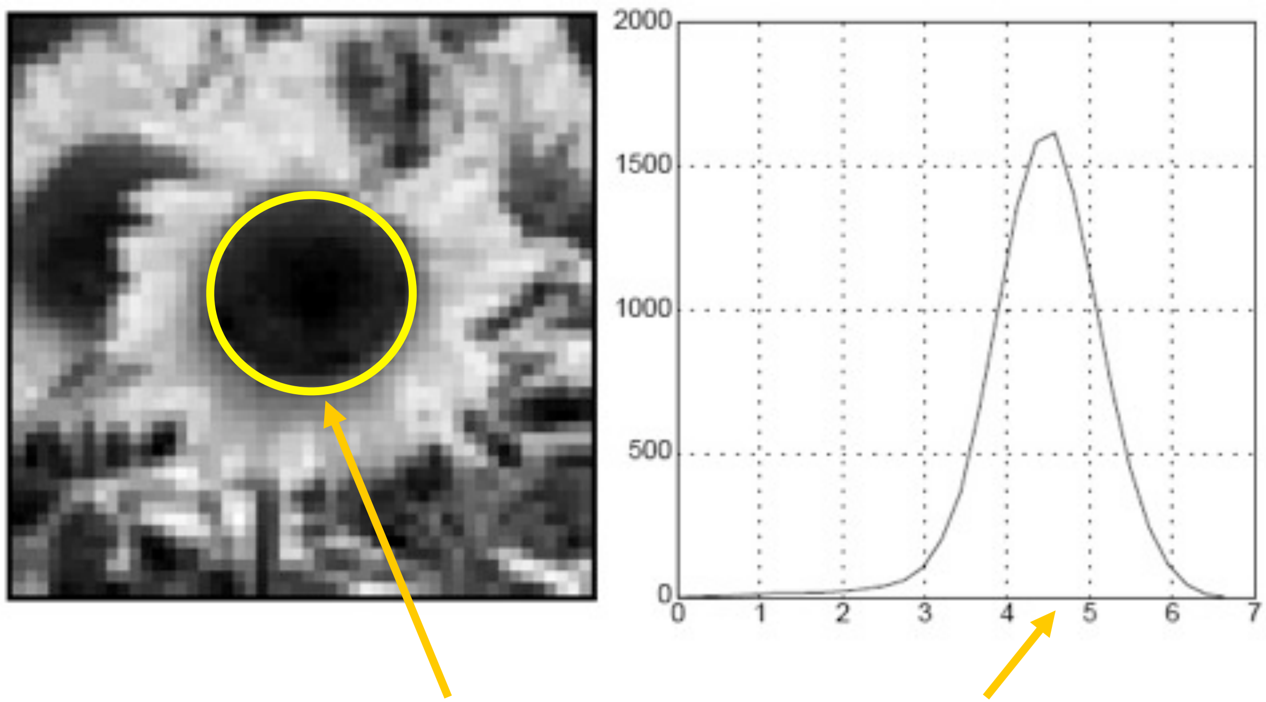


$f(I_{i_1 \dots i_m}(x, \sigma))$ $\qquad$ $f(I_{i_1 \dots i_m}(x', \sigma'))$



1. Interest Point Detection
Characteristic Scale

- We define the ***characteristic scale*** as the scale that produces peak of Laplacian response



Characteristic scale

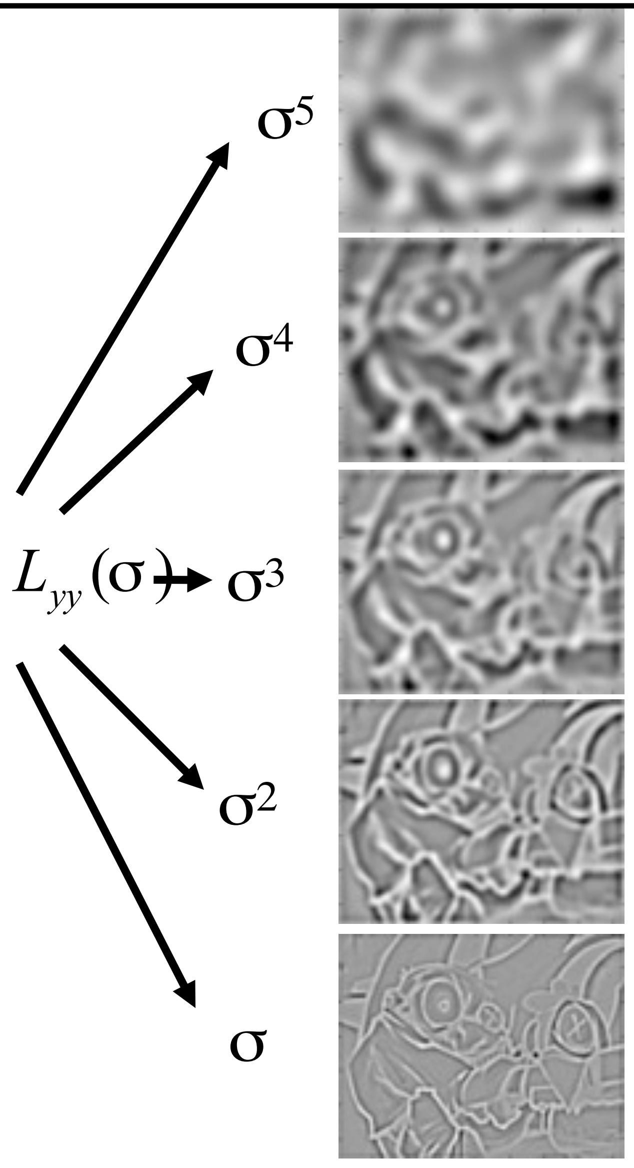
T. Lindeberg (1998). "Feature detection with automatic scale selection." ***International Journal of Computer Vision*** 30 (2): pp 77--116.

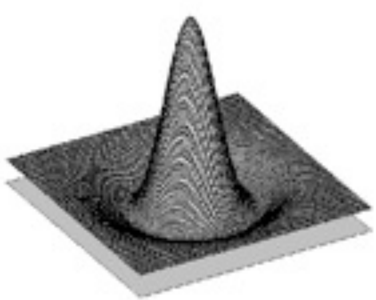
Slide credit: Svetlana Lazebnik

Laplacian-of-Gaussian (LoG)

- Interest points:
  - Local maxima in scale space of Laplacian-of-Gaussian

$$L_{xx}(\sigma) + L_{yy}(\sigma)$$

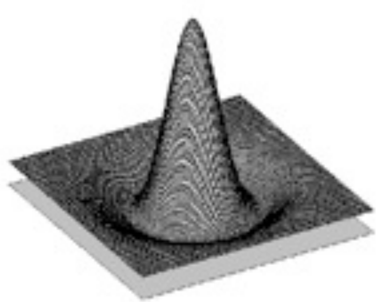


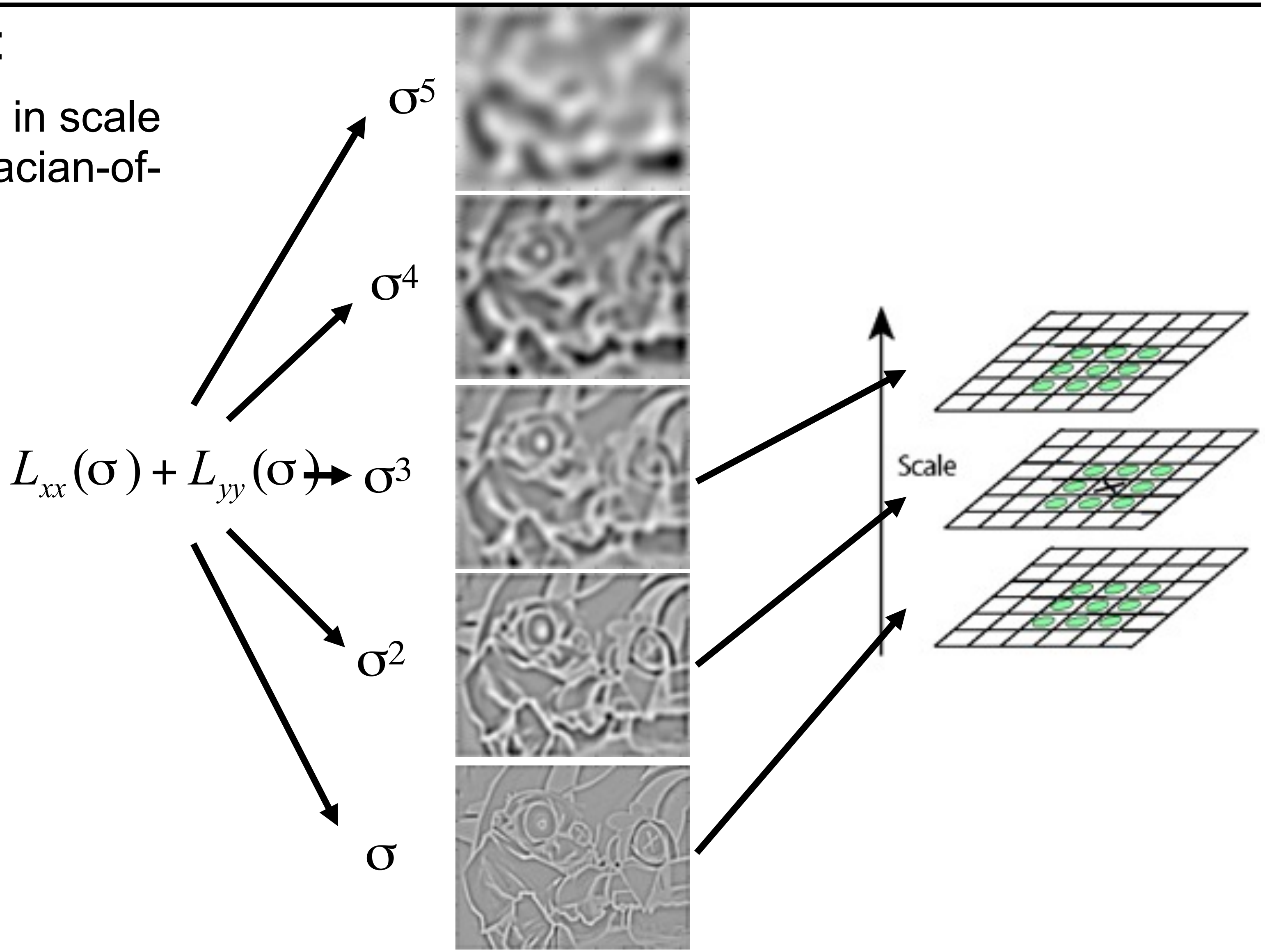


Slide adapted from Krystian Mikolajczyk

Laplacian-of-Gaussian (LoG)

- Interest points:
  - Local maxima in scale space of Laplacian-of-Gaussian

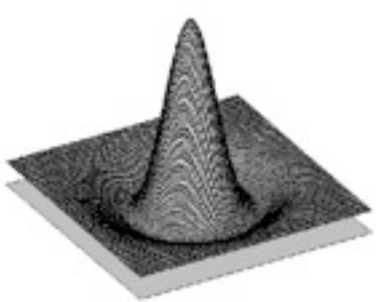


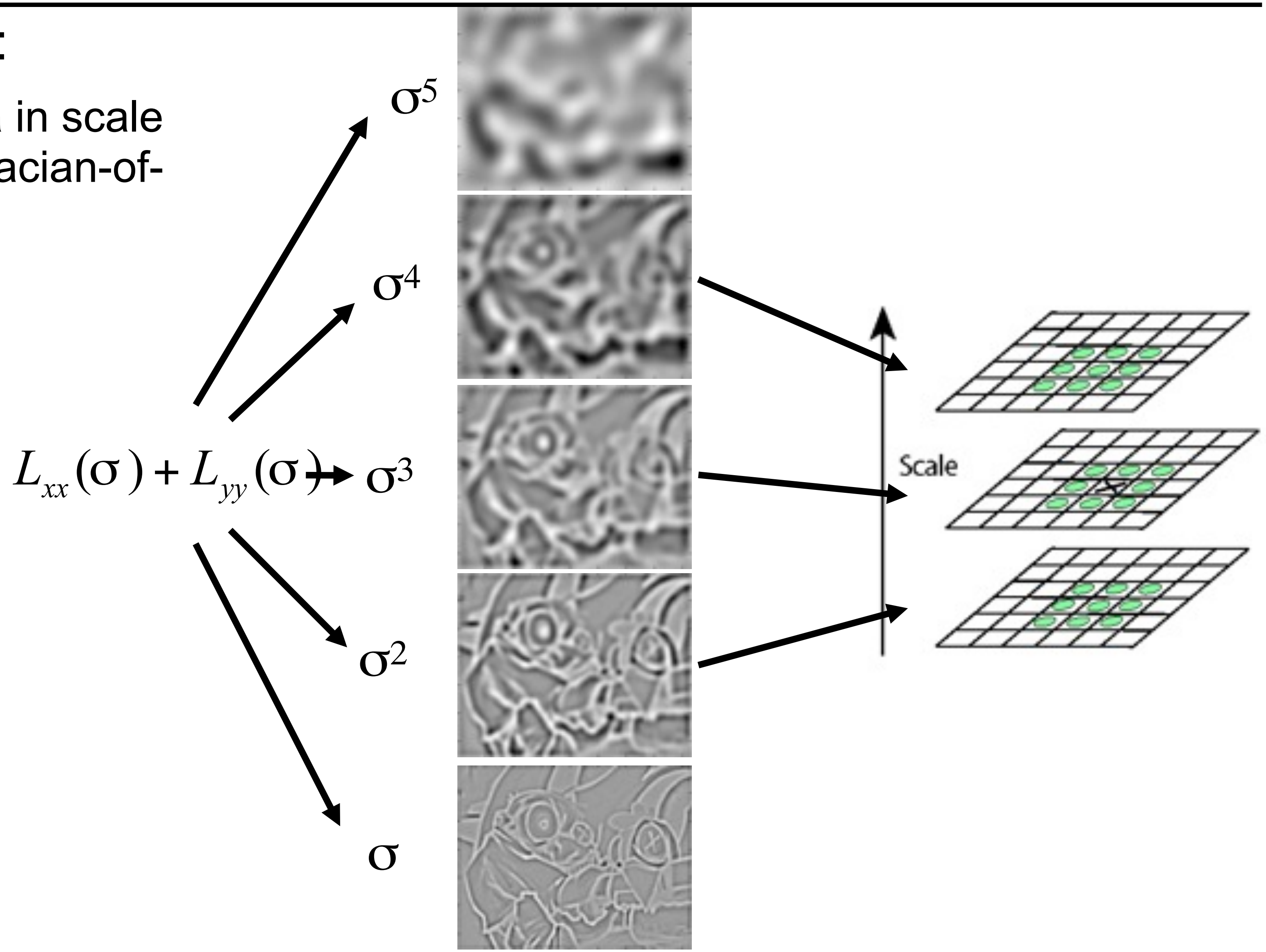


Slide adapted from Krystian Mikolajczyk

Laplacian-of-Gaussian (LoG)

- Interest points:
  - Local maxima in scale space of Laplacian-of-Gaussian

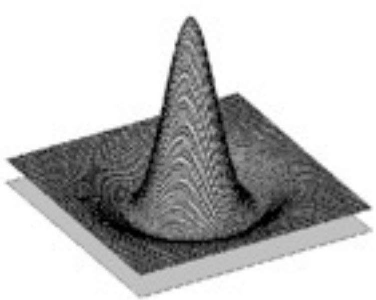


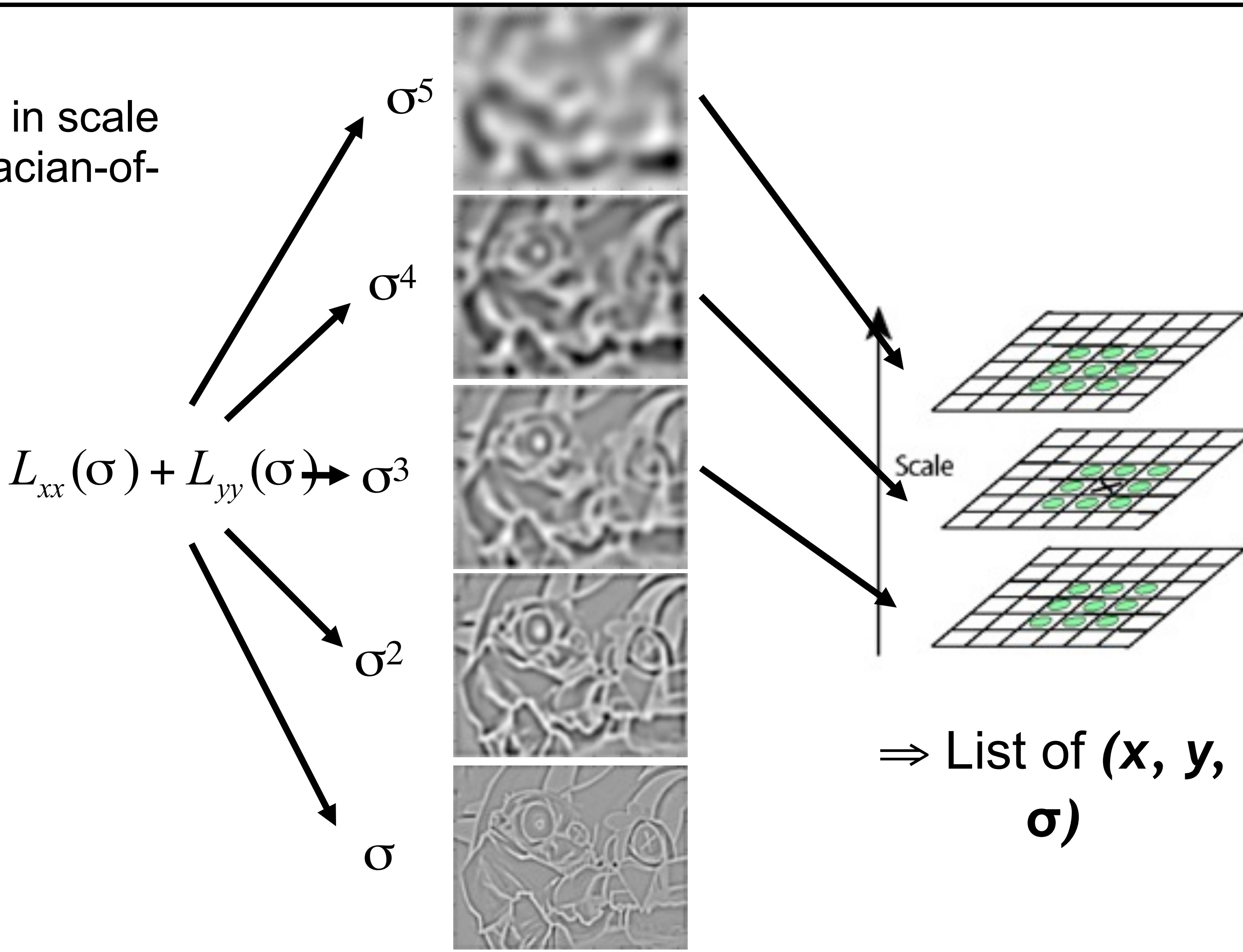


Slide adapted from Krystian Mikolajczyk

Laplacian-of-Gaussian (LoG)

- Interest points:
  - Local maxima in scale space of Laplacian-of-Gaussian





$\Rightarrow$ List of ***(x, y, σ)***

Slide adapted from Krystian Mikolajczyk

LoG Detector: Workflow

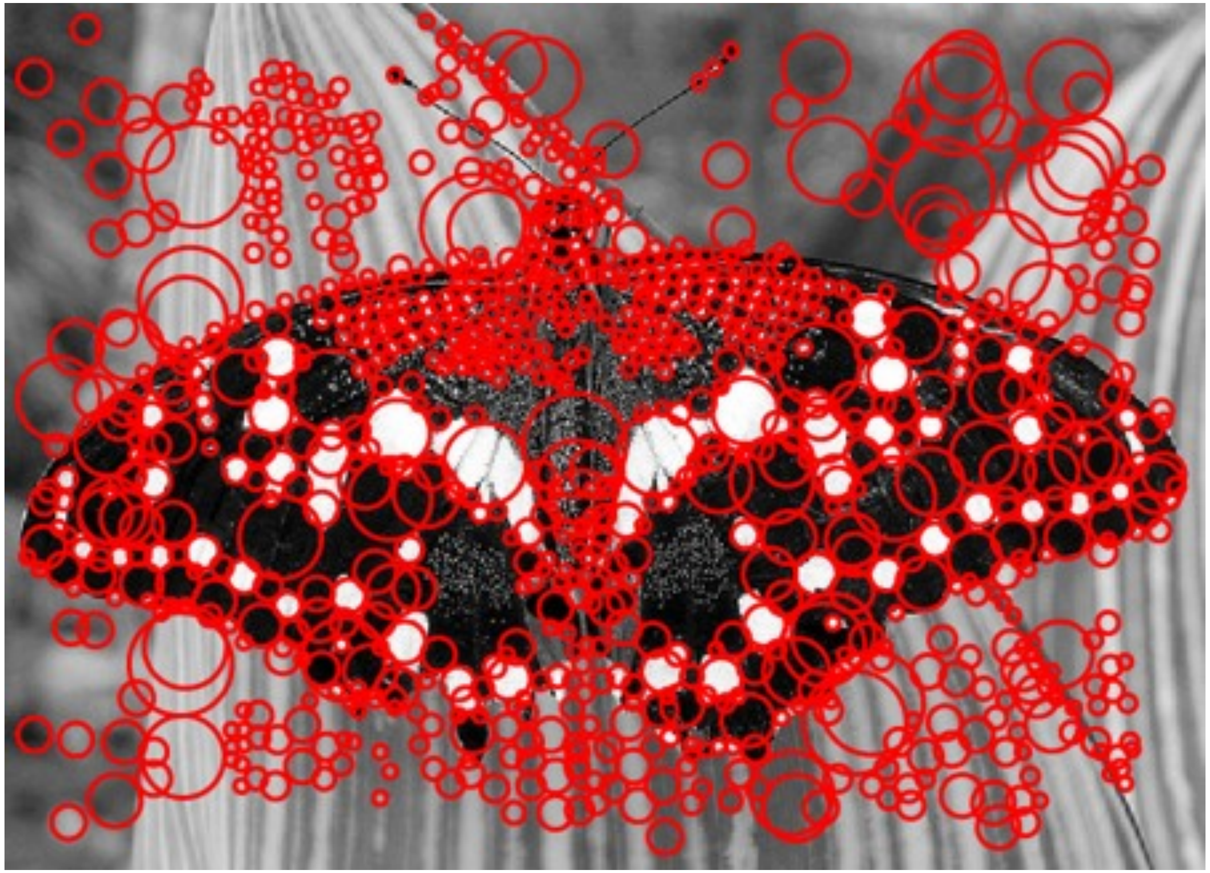
Slide credit: Svetlana Lazebnik

LoG Detector: Workflow

sigma = 11.9912

Slide credit: Svetlana Lazebnik

LoG Detector: Workflow



Slide credit: Svetlana Lazebnik

Technical Detail

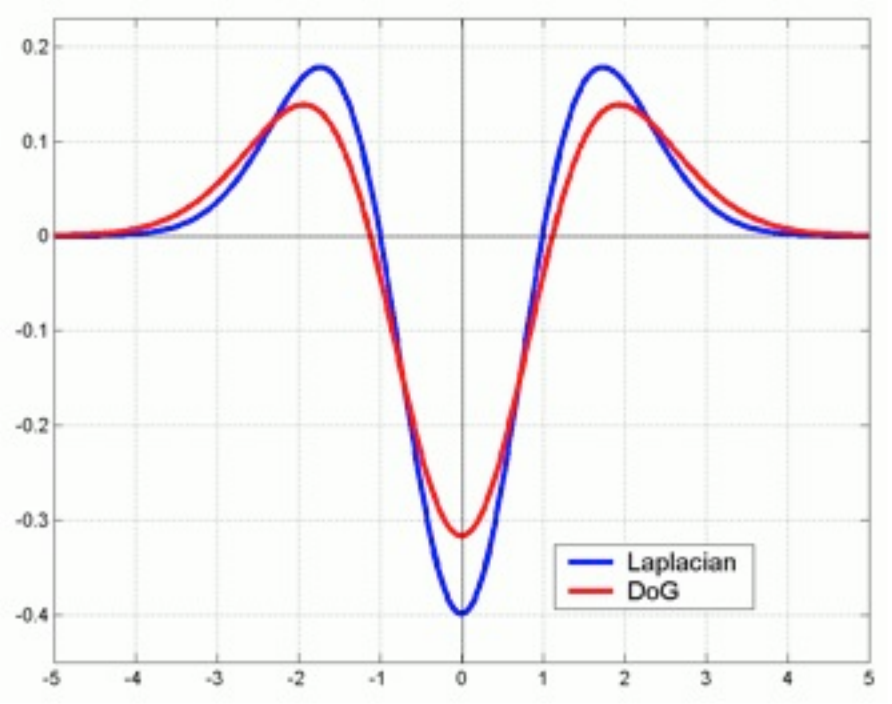
- We can efficiently approximate the Laplacian with a difference of Gaussians:

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

(Laplacian)

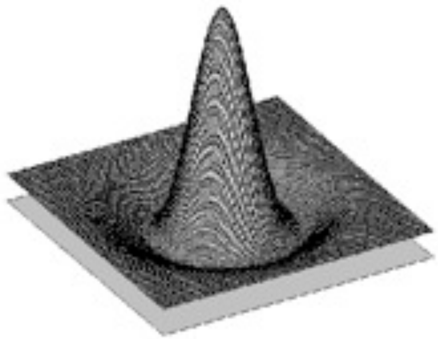
$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)



Difference-of-Gaussian (DoG)

- Difference of Gaussians as approximation of the LoG
  - This is used e.g. in Lowe's SIFT pipeline for feature detection.



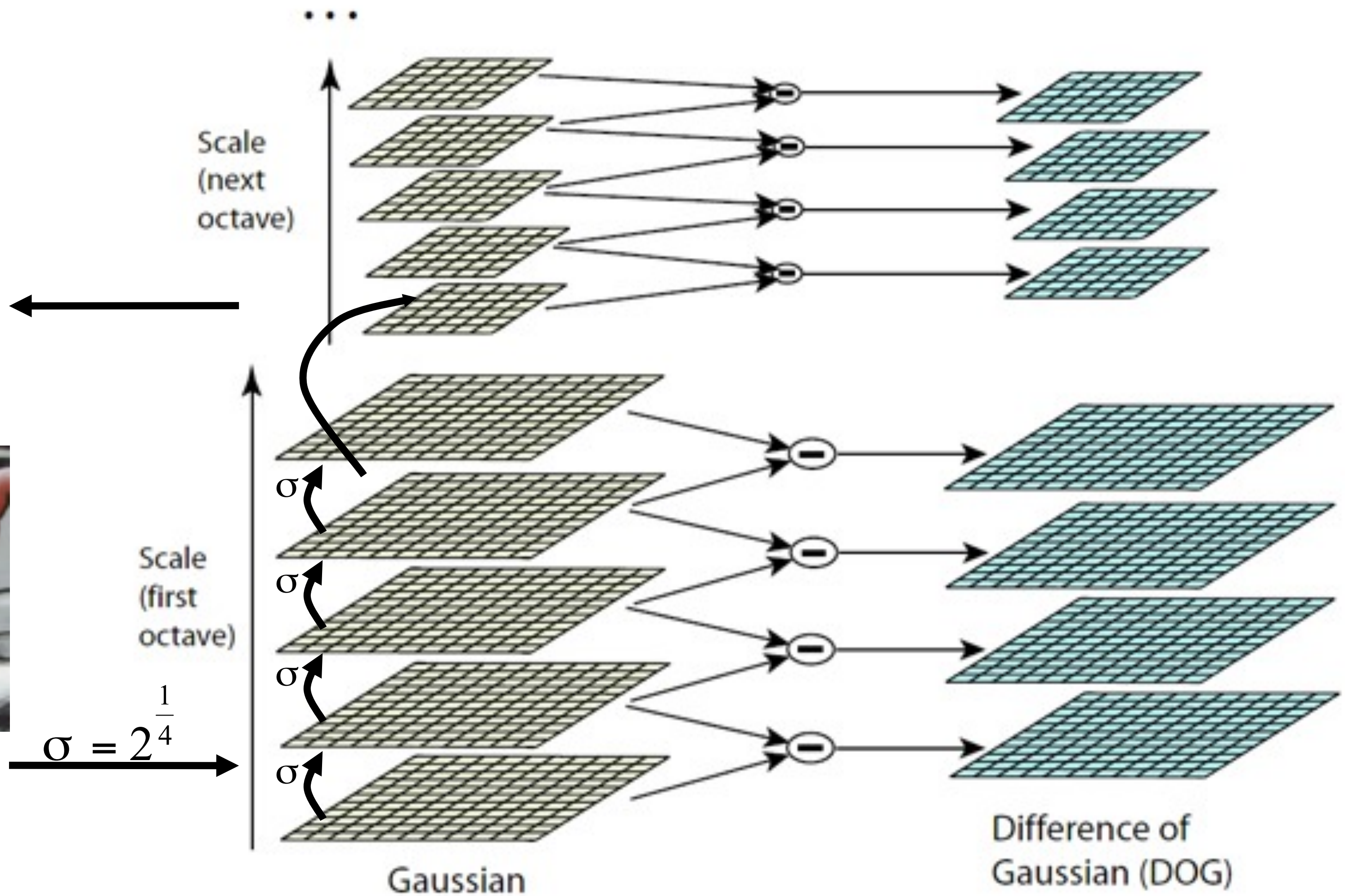
- Advantages
  - No need to compute 2nd derivatives
  - Gaussians are computed anyway, e.g. in a Gaussian pyramid.

DoG – Efficient Computation

- Computation in Gaussian scale pyramid

Sampling with step σ^4 *=2*

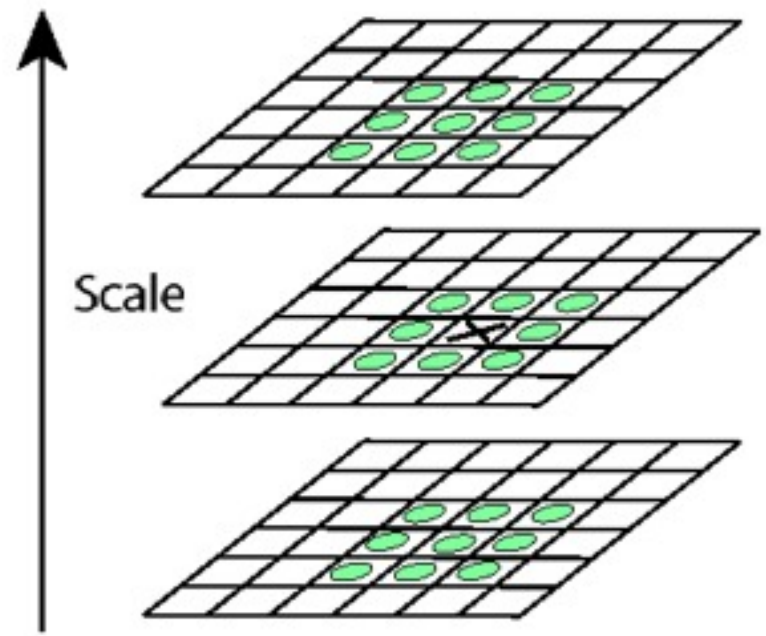
Original image



Slide adapted from Krystian Mikolajczyk

Key point localization with DoG (Lowe)

- Detect maxima of difference-of-Gaussian (DoG) in scale space
- Then reject points with low contrast (threshold)
- Eliminate edge responses

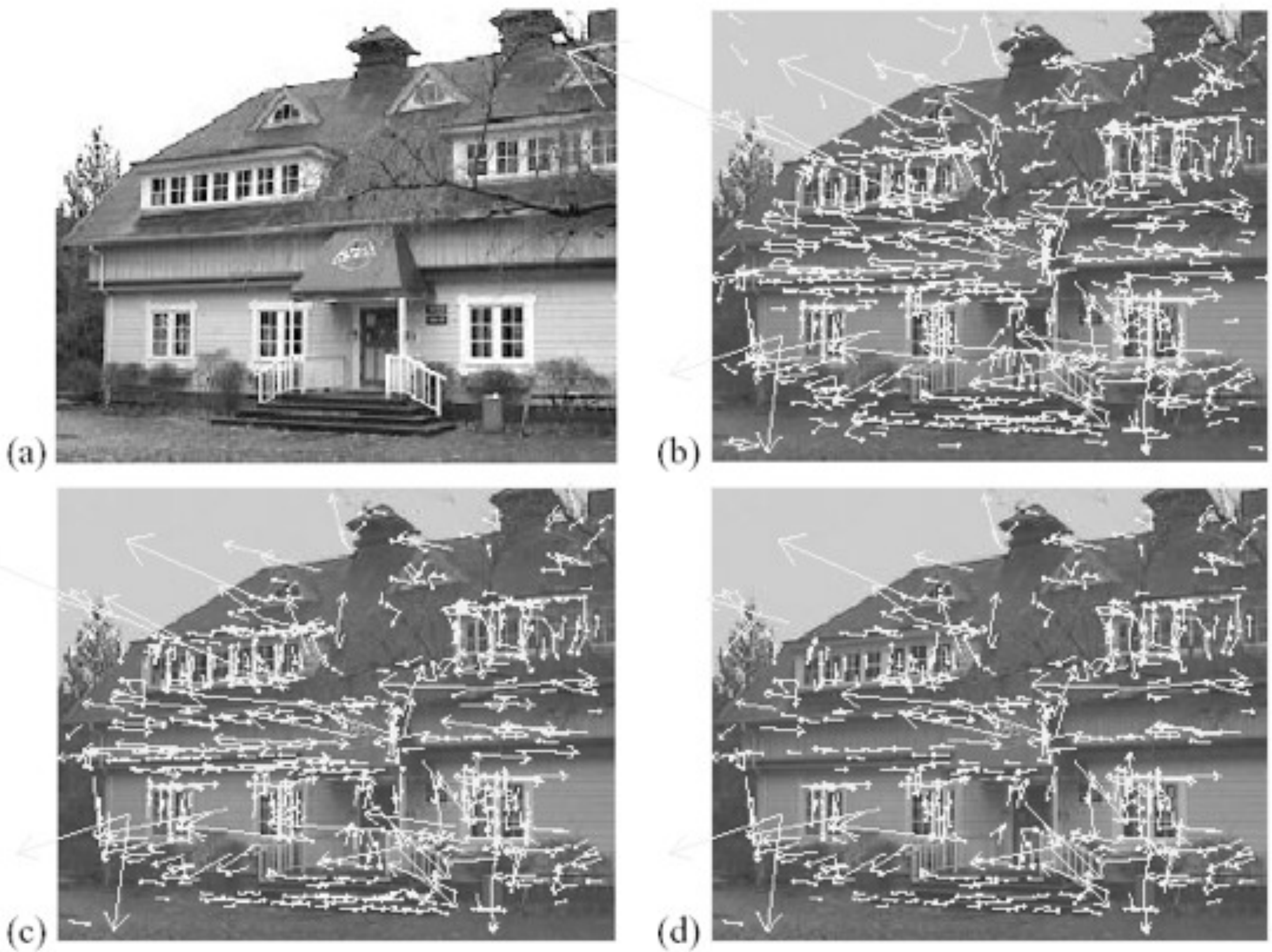


Candidate keypoints: list of (x,y,σ)

Slide credit: David Lowe

Results: Lowe's DoG

Example of Keypoint Detection



(a) 233x189 image

(b) 832 DoG extrema

(c) 729 left after peak value threshold

(d) 536 left after testing ratio of principle curvatures (removing edge responses)

Slide credit: David Lowe

Summary: Scale Invariant Detection

- Given: Two images of the same scene with a large ***scale difference*** between them.
- Goal: Find ***the same*** interest points ***independently*** in each image.
- Solution: Search for ***maxima*** of suitable functions in ***scale*** and in ***space*** (over the image).

- Two strategies
  - Laplacian-of-Gaussian (LoG)
  - Difference-of-Gaussian (DoG) as a fast approximation

  - ***These can be used either on their own, or in combinations with single-scale keypoint detectors (Harris, Hessian).***

DoG Code

Scheduling (parallelizing octaves)

- Load balancing problem
- Scheduling

Schedule Type	Description
static (default with no chunk size)	Partitions the loop iterations into equal-sized chunks or as nearly equal as possible in the case where the number of loop iterations is not evenly divisible by the number of threads multiplied by the chunk size. When chunk size is not specified, the iterations are divided as evenly as possible, with one chunk per thread. Set chunk to 1 to interleave the iterations.
dynamic	Uses an internal work queue to give a chunk-sized block of loop iterations to each thread as it becomes available. When a thread is finished with its current block, it retrieves the next block of loop iterations from the top of the work queue. By default, chunk size is 1. Be careful when using this scheduling type because of the extra overhead required.
guided	Similar to dynamic scheduling, but the chunk size starts off large and shrinks in an effort to reduce the amount of time threads have to go to the work queue to get more work. The optional chunk parameter specifies the minimum size chunk to use, which, by default, is 1.
runtime	Uses the OMP_SCHEDULE environment variable at runtime to specify which one of the three loop-scheduling types should be used. OMP_SCHEDULE is a string formatted exactly the same as it would appear on the parallel construct.

Cost of scheduling

Constructs	Cost (in microseconds)	Scalability
parallel	1.5	Linear
Barrier	1.0	Linear or O(log(n))
schedule(static)	1.0	Linear
schedule(guided)	6.0	Depends on contention
schedule(dynamic)	50	Depends on contention
ordered	0.5	Depends on contention
Single	1.0	Depends on contention
Reduction	2.5	Linear or O(log(n))
Atomic	0.5	Depends on data-type and hardware
Critical	0.5	Depends on contention
Lock/Unlock	0.5	Depends on contention

Overhead of threading

```
#pragma omp parallel for
for
( k = 0; k < m; k++ ) {
    fn1(k); fn2(k);
}

#pragma omp parallel for // adds unnecessary overhead
for ( k = 0; k < m; k++ ) {
    fn3(k); fn4(k);
}
```

1 x overhead

vs.

```
#pragma omp parallel
{
    #pragma omp for
    for ( k = 0; k < m; k++ ) {
        fn1(k); fn2(k);
    }

    #pragma omp for
    for ( k = 0; k < m; k++ ) {
        fn3(k); fn4(k);
    }
}
```

2 x overhead

More modifiers

- #pragma omp parallel for no wait
  - doesn’t wait for jobs to finish
- #pragma omp master
  - only master thread executes
- #pragma omp barrier
  - explicit synchonization
- #pragma omp single
  - only one thread executes
- #pragma omp critical
  - only one thread can enter (risk of deadlocks)
  - named critical sections #pragma omp critical (name)
- #pragma omp atomic
  - thread cannot be interrupted
- #pragma omp parallel if (n>=16)
  - parallel only for more than 16 loops

Additional resources:

- http://www.drdobbs.com/parallel/openmp-a-portable-solution-for-threading/225702895?pgno=1

- http://www.compunity.org/training/tutorials/2%20Basic_Concepts_Parallelization.pdf

- http://www.vlfeat.org/api/sift.html

- http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf