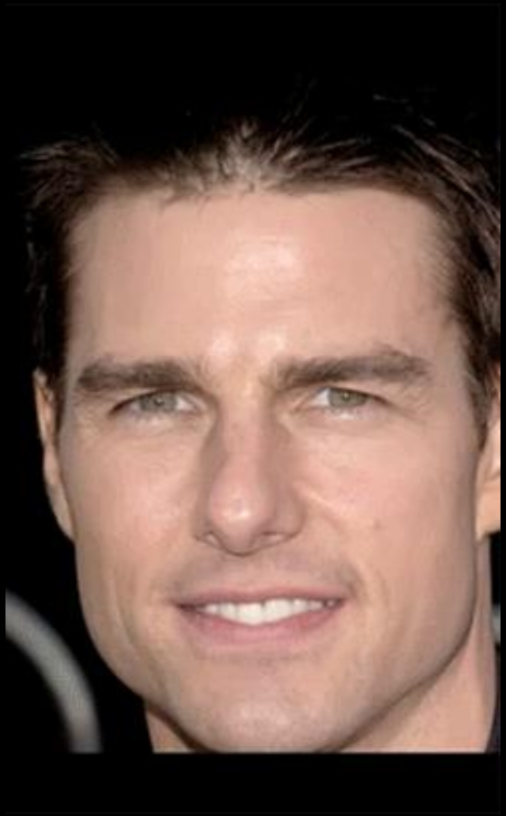
Read Szeliski 7.1.2 and 7.1.3

Local Image Features

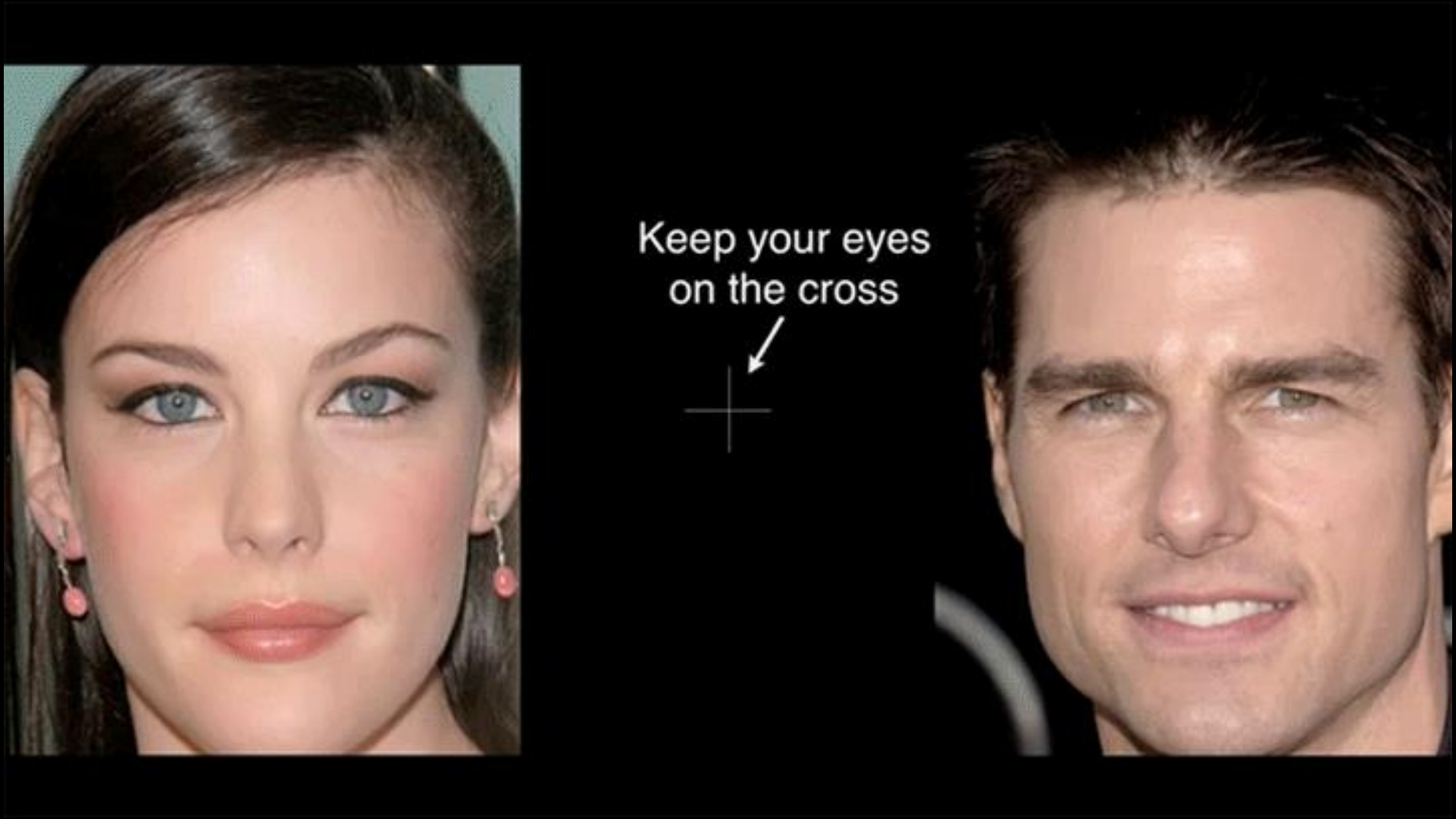
Computer Vision

James Hays

Acknowledgment: Many slides from Derek Hoiem and Grauman&Leibe 2008 AAAI Tutorial

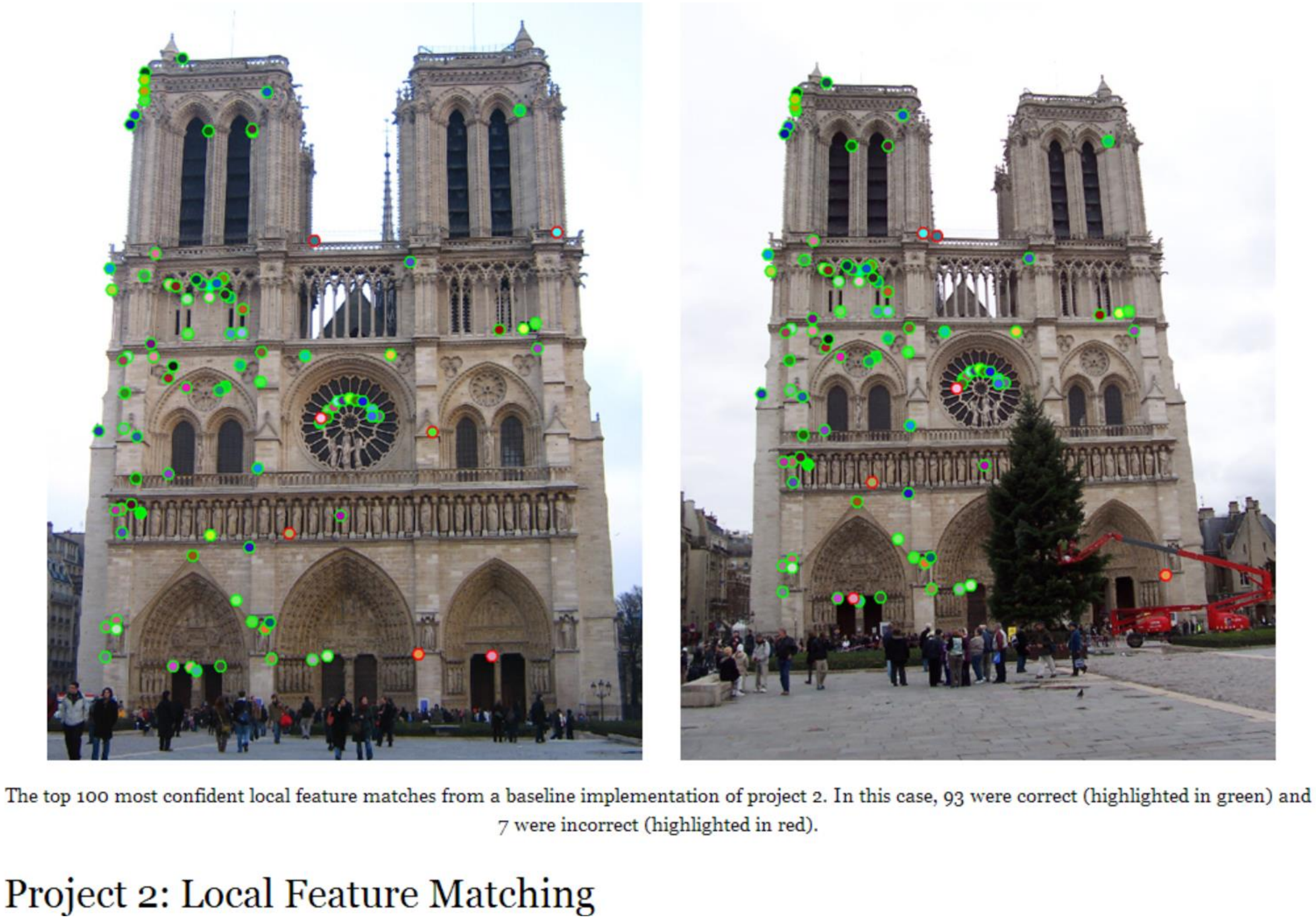


“Flashed Face Distortion”
2nd Place in the 8th Annual
Best Illusion of the Year
Contest , VSS 2012



Keep your eyes on the cross

Project 2

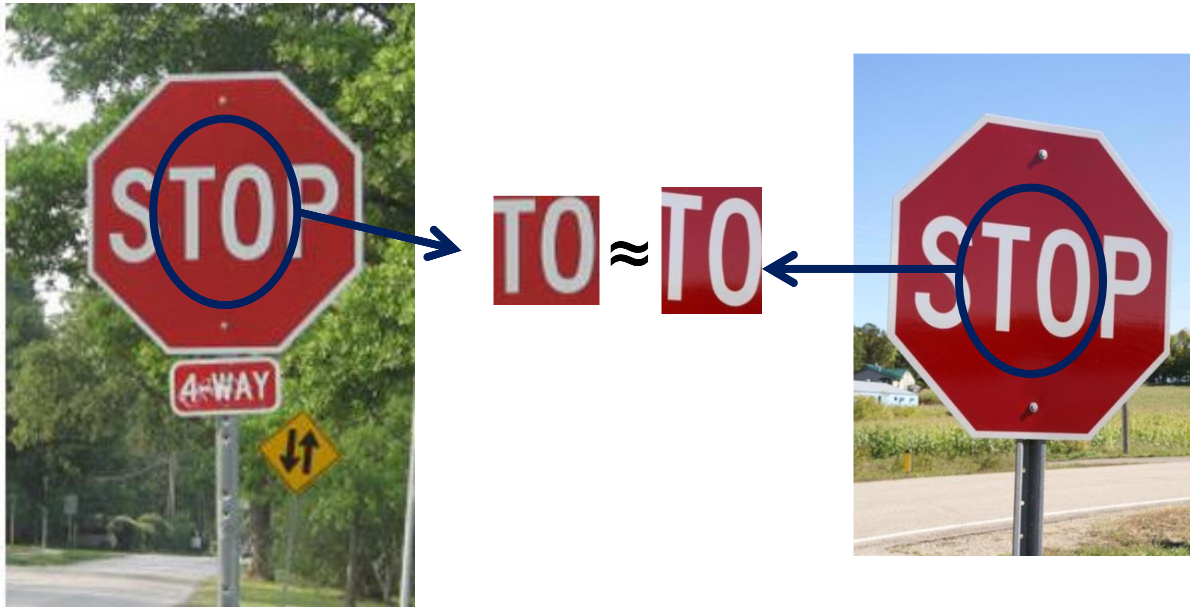


The top 100 most confident local feature matches from a baseline implementation of project 2. In this case, 93 were correct (highlighted in green) and 7 were incorrect (highlighted in red).

Project 2: Local Feature Matching

This section: correspondence and alignment

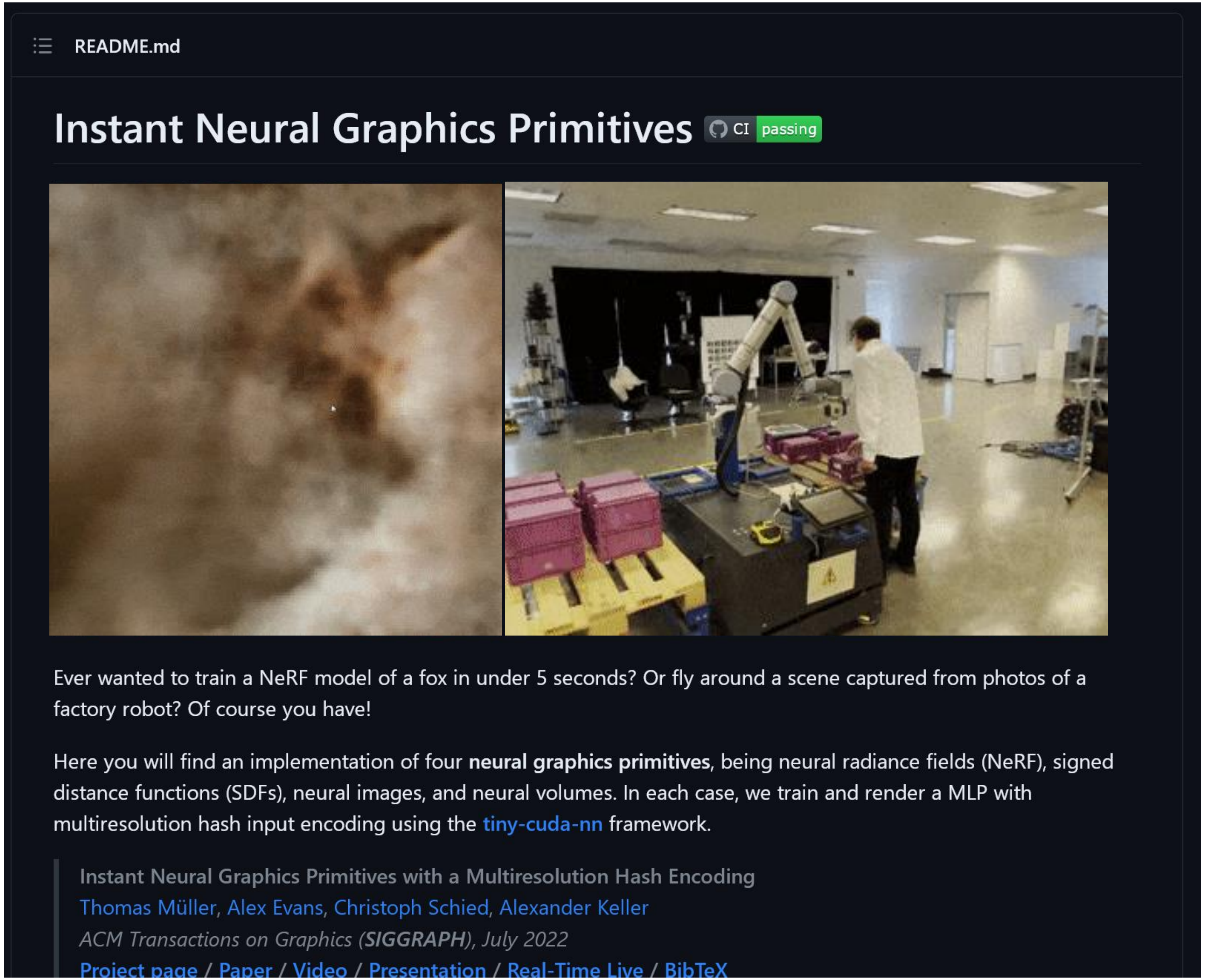
- Correspondence: matching points, patches, edges, or regions across images



SIFT is 20+ years old. Is it still useful?

SIFT is 20+ years old. Is it still useful?

- Let’s look at some trendy research on Neural Radiance Fields (NERF)



Instant Neural Graphics Primitives

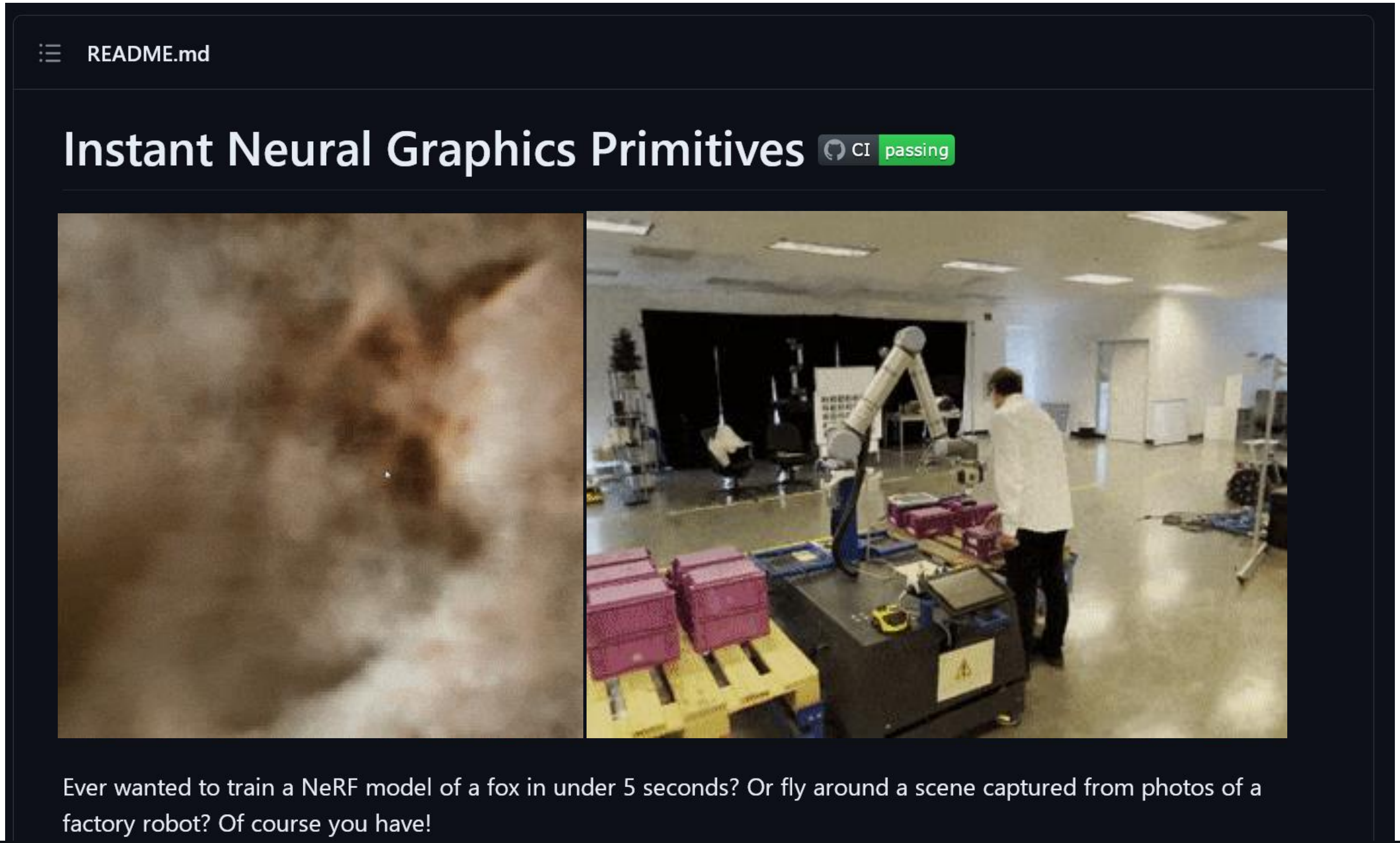
Ever wanted to train a NeRF model of a fox in under 5 seconds? Or fly around a scene captured from photos of a factory robot? Of course you have!

Here you will find an implementation of four **neural graphics primitives**, being neural radiance fields (NeRF), signed distance functions (SDFs), neural images, and neural volumes. In each case, we train and render a MLP with multiresolution hash input encoding using the tiny-cuda-nn framework.

> Instant Neural Graphics Primitives with a Multiresolution Hash Encoding
> Thomas Müller, Alex Evans, Christoph Schied, Alexander Keller
> *ACM Transactions on Graphics (**SIGGRAPH**), July 2022*
> Project page / Paper / Video / Presentation / Real-Time Live / BibTeX

SIFT is 20+ years old. Is it still useful?

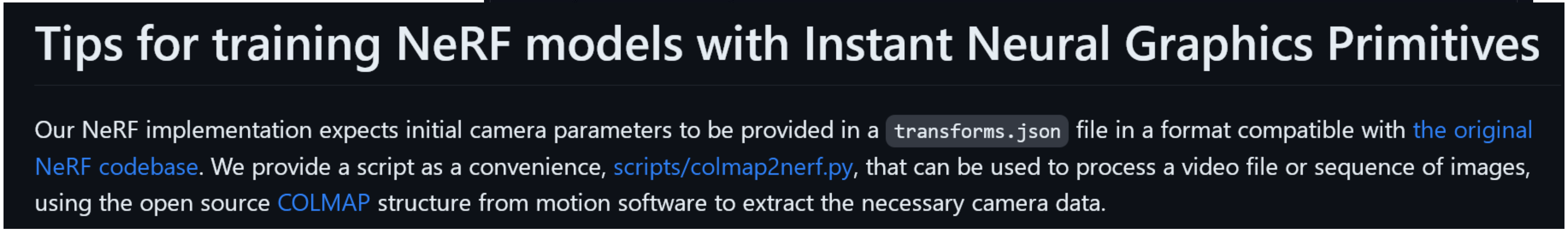
- Let’s look at some trendy research on Neural Radiance Fields (NERF)
- Let’s look under the hood



README.md

Instant Neural Graphics Primitives

Ever wanted to train a NeRF model of a fox in under 5 seconds? Or fly around a scene captured from photos of a factory robot? Of course you have!

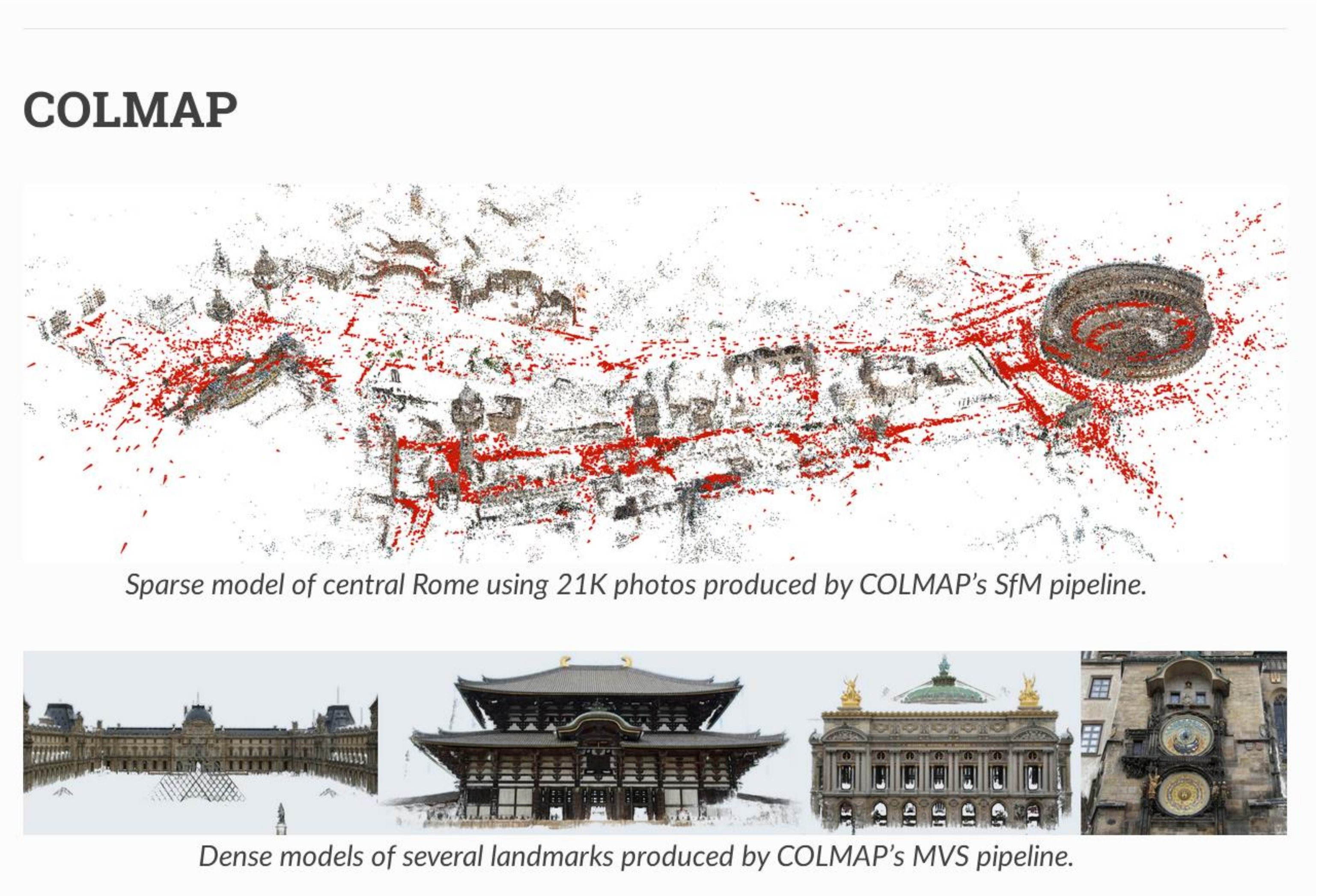


Tips for training NeRF models with Instant Neural Graphics Primitives

Our NeRF implementation expects initial camera parameters to be provided in a `transforms.json` file in a format compatible with the original NeRF codebase. We provide a script as a convenience, scripts/colmap2nerf.py, that can be used to process a video file or sequence of images, using the open source COLMAP structure from motion software to extract the necessary camera data.

SIFT is 20+ years old. Is it still useful?

- COLMAP is the “standard” way to do structure from motion these days



COLMAP

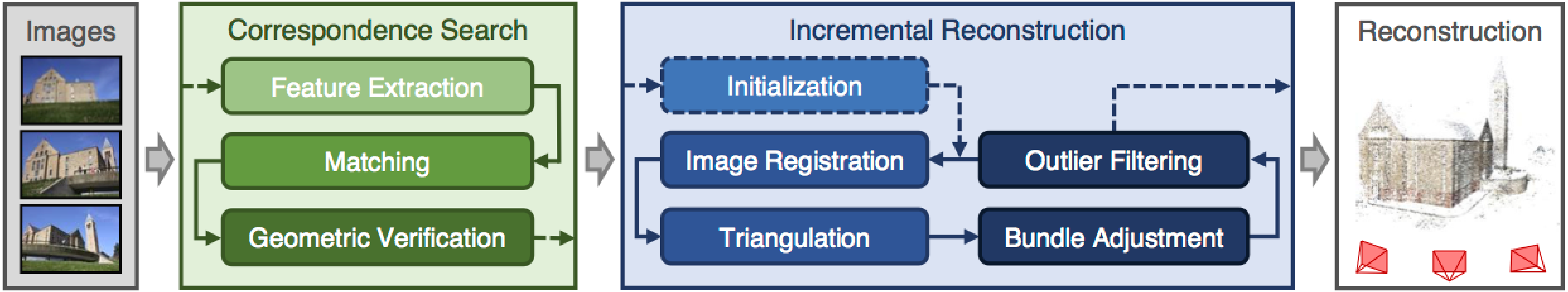
Sparse model of central Rome using 21K photos produced by COLMAP’s SfM pipeline.

Dense models of several landmarks produced by COLMAP’s MVS pipeline.

“Structure-From-Motion Revisited”. Johannes L. Schonberger, Jan-Michael Frahm; CVPR 2016
3k+ citations

SIFT is 20+ years old. Is it still useful?

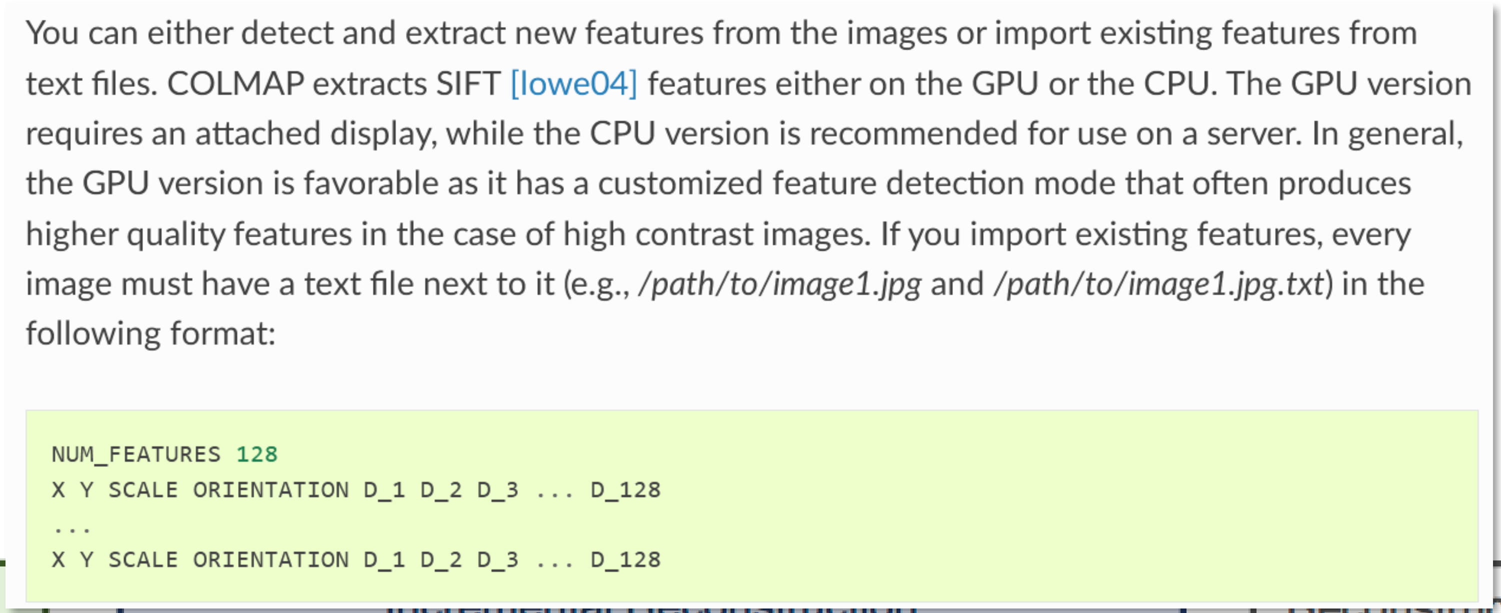
- COLMAP is the “standard” way to do structure from motion these days



“Structure-From-Motion Revisited”. Johannes L. Schonberger, Jan-Michael Frahm; CVPR 2016
3k+ citations

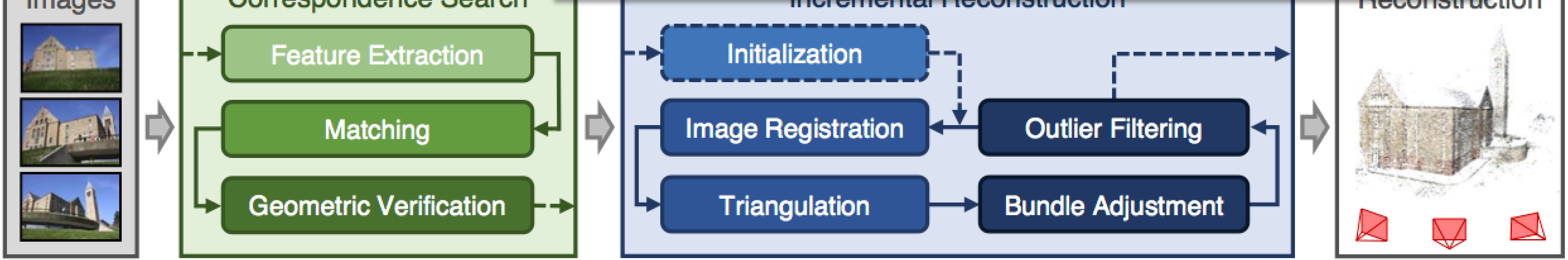
SIFT is 20+ years old. Is it still useful?

- COLMAP is the "standard" way to do structure from motion these days



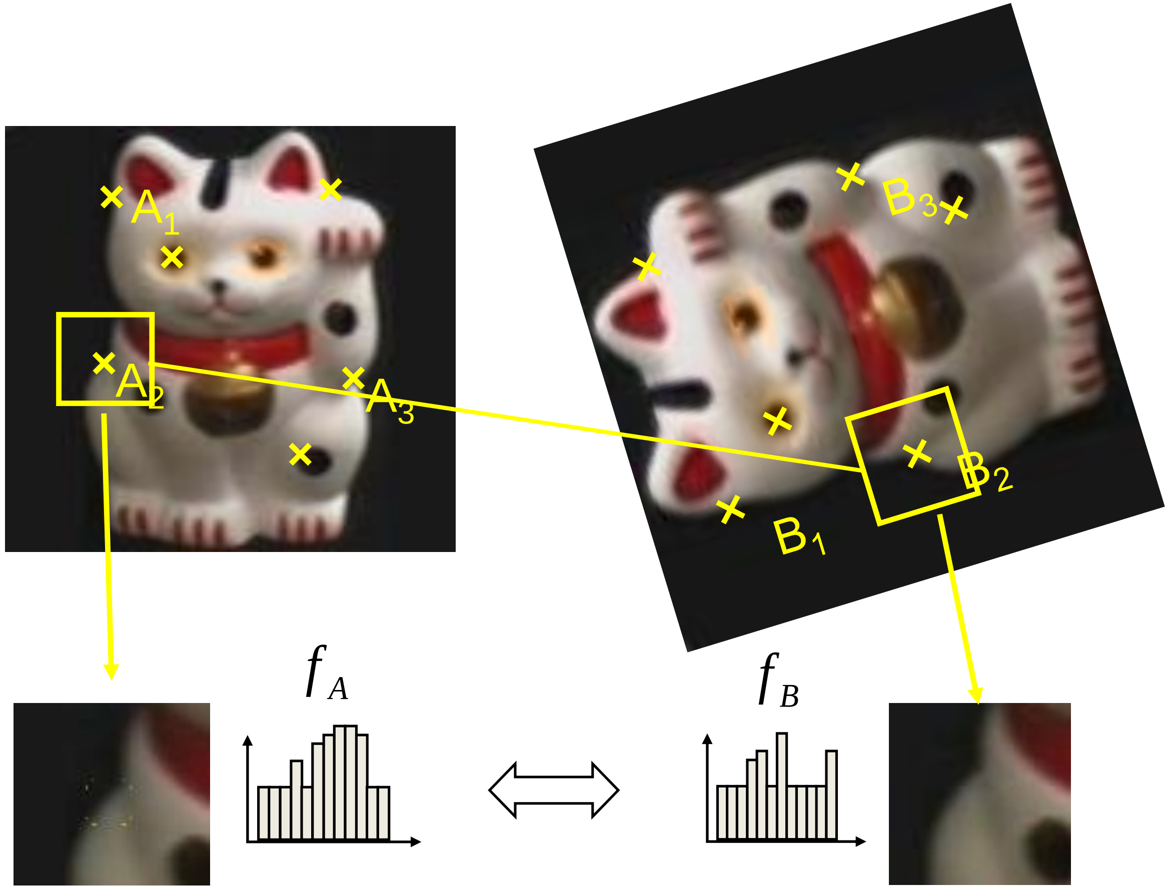
You can either detect and extract new features from the images or import existing features from text files. COLMAP extracts SIFT [lowe04] features either on the GPU or the CPU. The GPU version requires an attached display, while the CPU version is recommended for use on a server. In general, the GPU version is favorable as it has a customized feature detection mode that often produces higher quality features in the case of high contrast images. If you import existing features, every image must have a text file next to it (e.g., */path/to/image1.jpg* and */path/to/image1.jpg.txt*) in the following format:

```
NUM_FEATURES 128
X Y SCALE ORIENTATION D_1 D_2 D_3 ... D_128
...
X Y SCALE ORIENTATION D_1 D_2 D_3 ... D_128
```



"Structure-From-Motion Revisited". Johannes L. Schonberger, Jan-Michael Frahm; CVPR 2016
3k+ citations

Overview of Keypoint Matching



$$d(f_A, f_B) < T$$

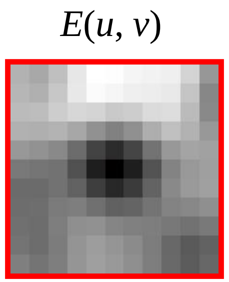
1. Find a set of distinctive keypoints

2. Compute a local descriptor from the region around each keypoint

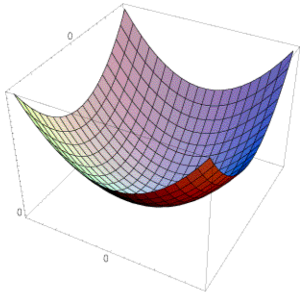
3. Match local descriptors

K. Grauman, B. Leibe

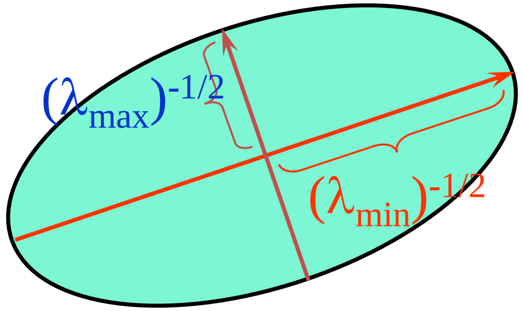
Review: Harris corner detector



- Define distinctiveness by local auto-correlation.
- Approximate local auto-correlation by second moment matrix



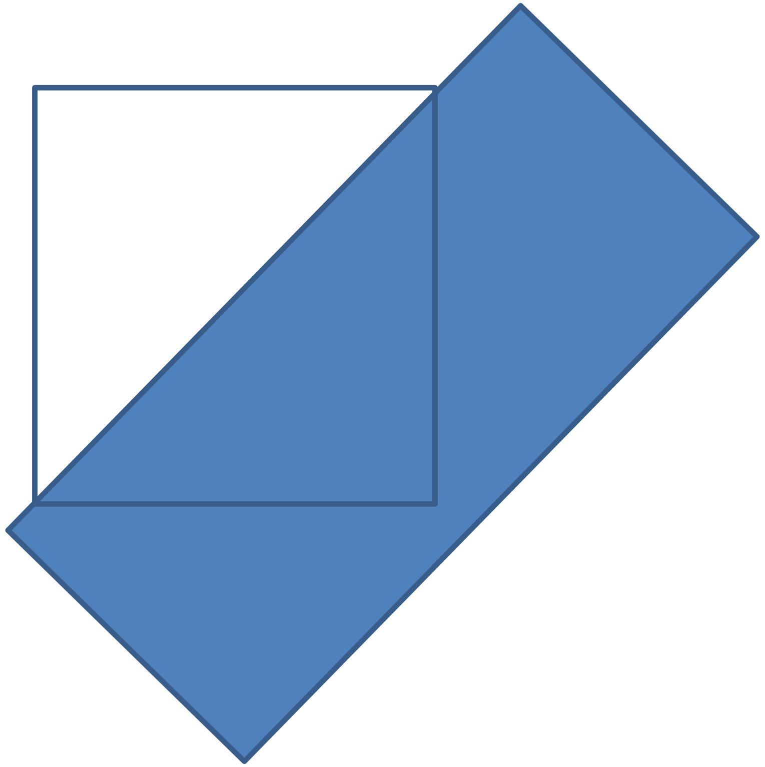
- Quantify distinctiveness (or cornerness) as function of the eigenvalues of the second moment matrix.
- But we don’t actually need to compute the eigenvalues. Instead, we use the determinant and trace of the second moment matrix.



Review: Harris corner detector

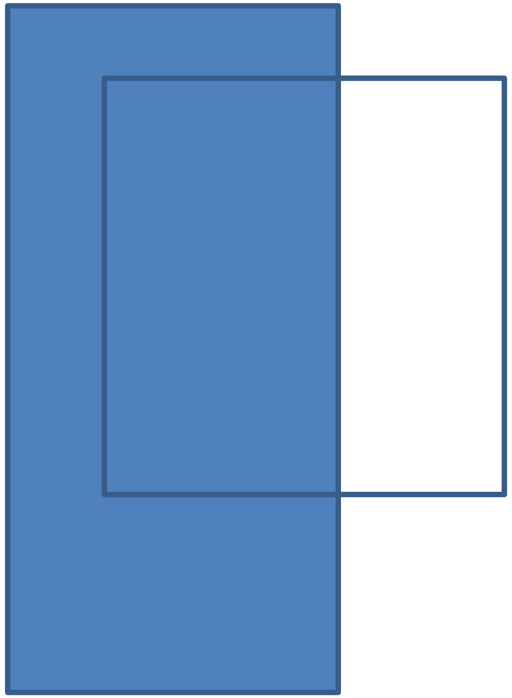
- We want to find *distinctive* patches that don’t look self-similar to neighboring patches
- If there are *gradients* in a patch, those gradients indicate distinctiveness in a particular direction.
- We want to check that we have strong, independent gradients in all directions.
- The eigenvalues of a the collection of gradients in a patch tell us this.

What do the gradients / structure matrix look like?



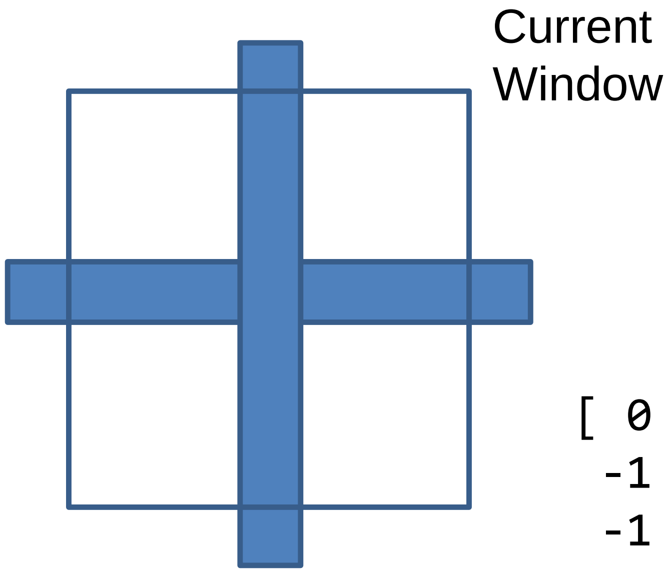
```
[ 0 0
 -1 1
 -1 1
  0 0
  0 0
 -1 1
  0 0
  …   ]
```

$$\begin{bmatrix} C & -C \\ -C & C \end{bmatrix}$$



```
[ 0 0
  1 0
  1 0
  0 0
  0 0
  0 0
  1 0
  …   ]
```

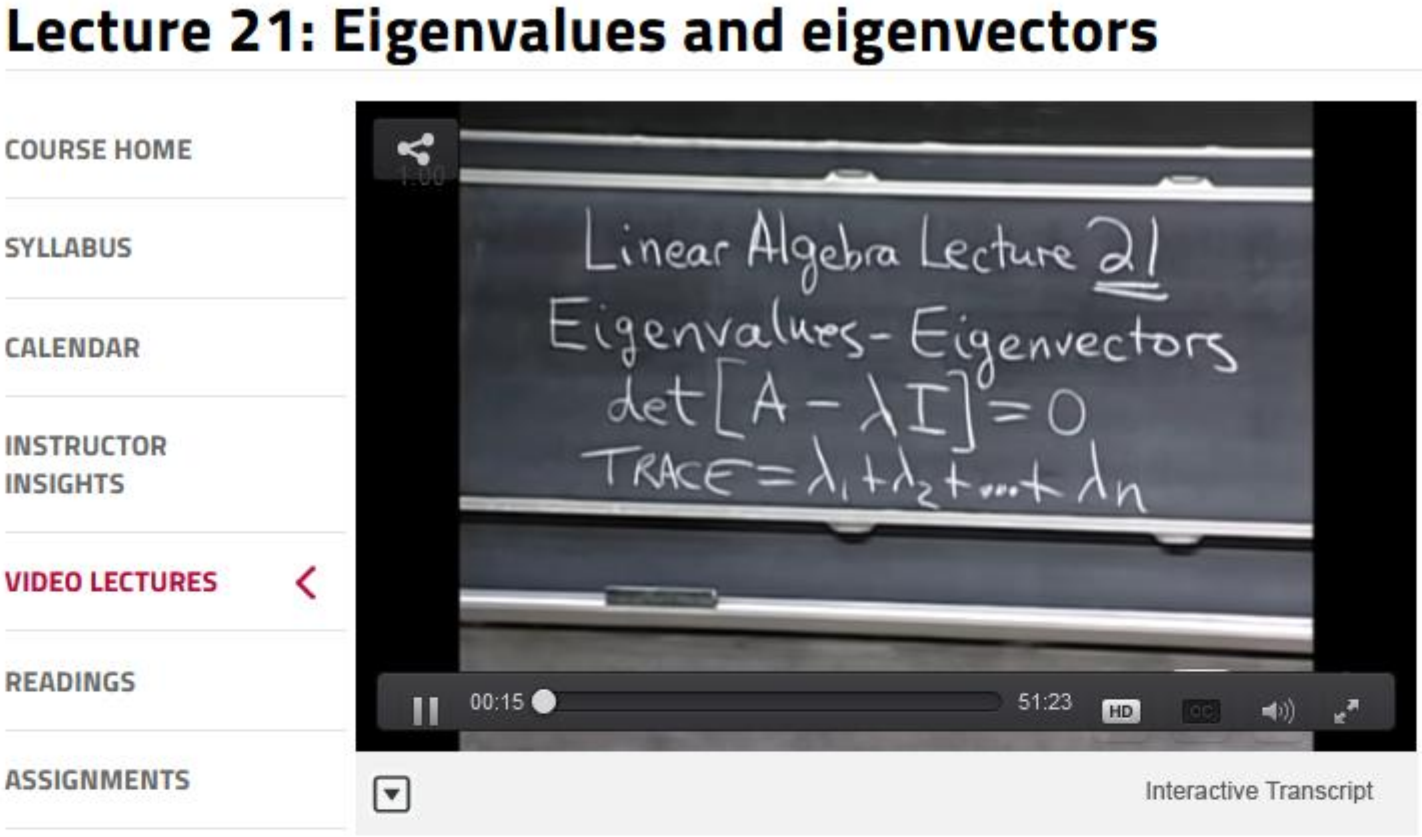
$$\begin{bmatrix} C & 0 \\ 0 & 0 \end{bmatrix}$$



```
[ 0 0
 -1 0
 -1 0
  0 0
  0 0
  0 1
  1 0
  …   ]
```

$$\begin{bmatrix} C & 0 \\ 0 & C \end{bmatrix}$$

If you’re not comfortable with Eigenvalues and Eigenvectors, Gilbert Strang’s linear algebra lectures are linked from the course homepage

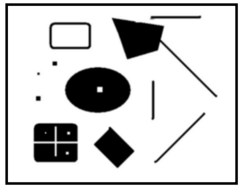


Lecture 21: Eigenvalues and eigenvectors

COURSE HOME
SYLLABUS
CALENDAR
INSTRUCTOR INSIGHTS
VIDEO LECTURES
READINGS
ASSIGNMENTS
Linear Algebra Lecture 21
Eigenvalues - Eigenvectors
$\det[A - \lambda I] = 0$
TRACE $= \lambda_1 + \lambda_2 + \dots + \lambda_n$
00:15
51:23
HD
CC
Interactive Transcript

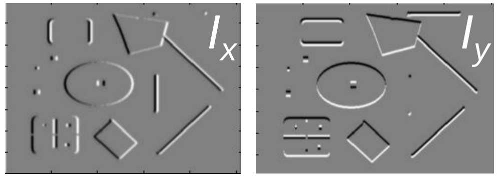
Harris Detector [Harris88]

- Second moment matrix

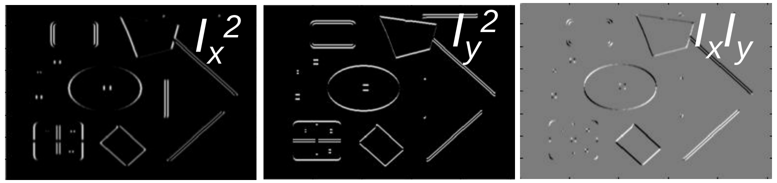


$$\mu(\sigma_I,\sigma_D)=g(\sigma_I)*\begin{bmatrix} I_x^2(\sigma_D) & I_xI_y(\sigma_D) \\ I_xI_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

1. Image derivatives (optionally, blur first)



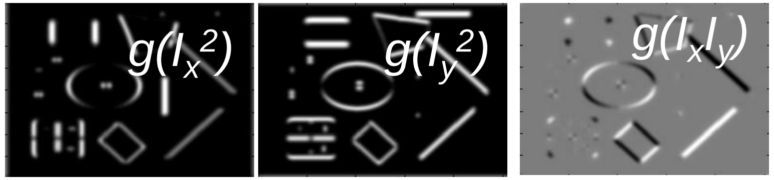
2. Square of derivatives



$$\det M = \lambda_1\lambda_2$$

$$\operatorname{trace} M = \lambda_1+\lambda_2$$

3. Gaussian filter *g(σ_I)*

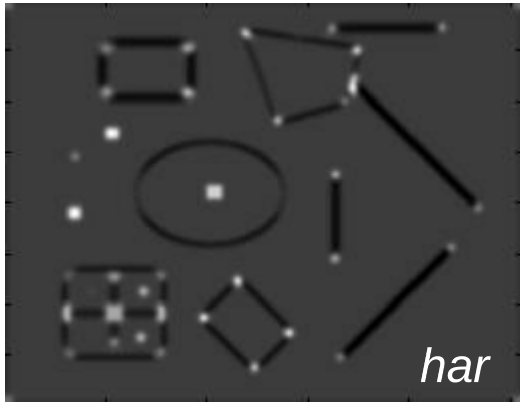


4. Cornerness function – both eigenvalues are strong

$$har=\det[\mu(\sigma_I,\sigma_D)]-\alpha[\operatorname{trace}(\mu(\sigma_I,\sigma_D))^2]=$$

$$g(I_x^2)g(I_y^2)-[g(I_xI_y)]^2-\alpha[g(I_x^2)+g(I_y^2)]^2$$

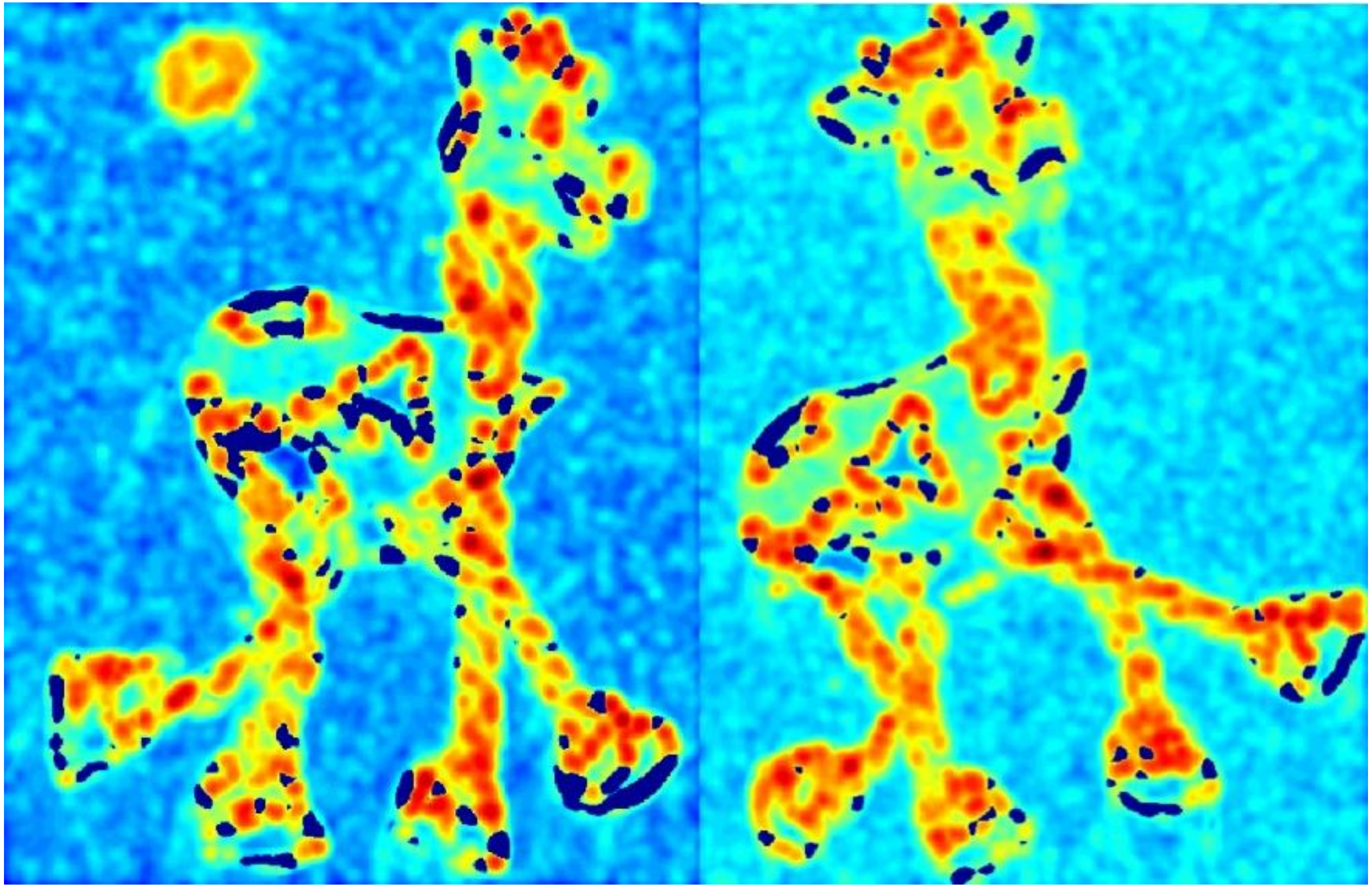
5. Non-maxima suppression



Harris Detector: Steps

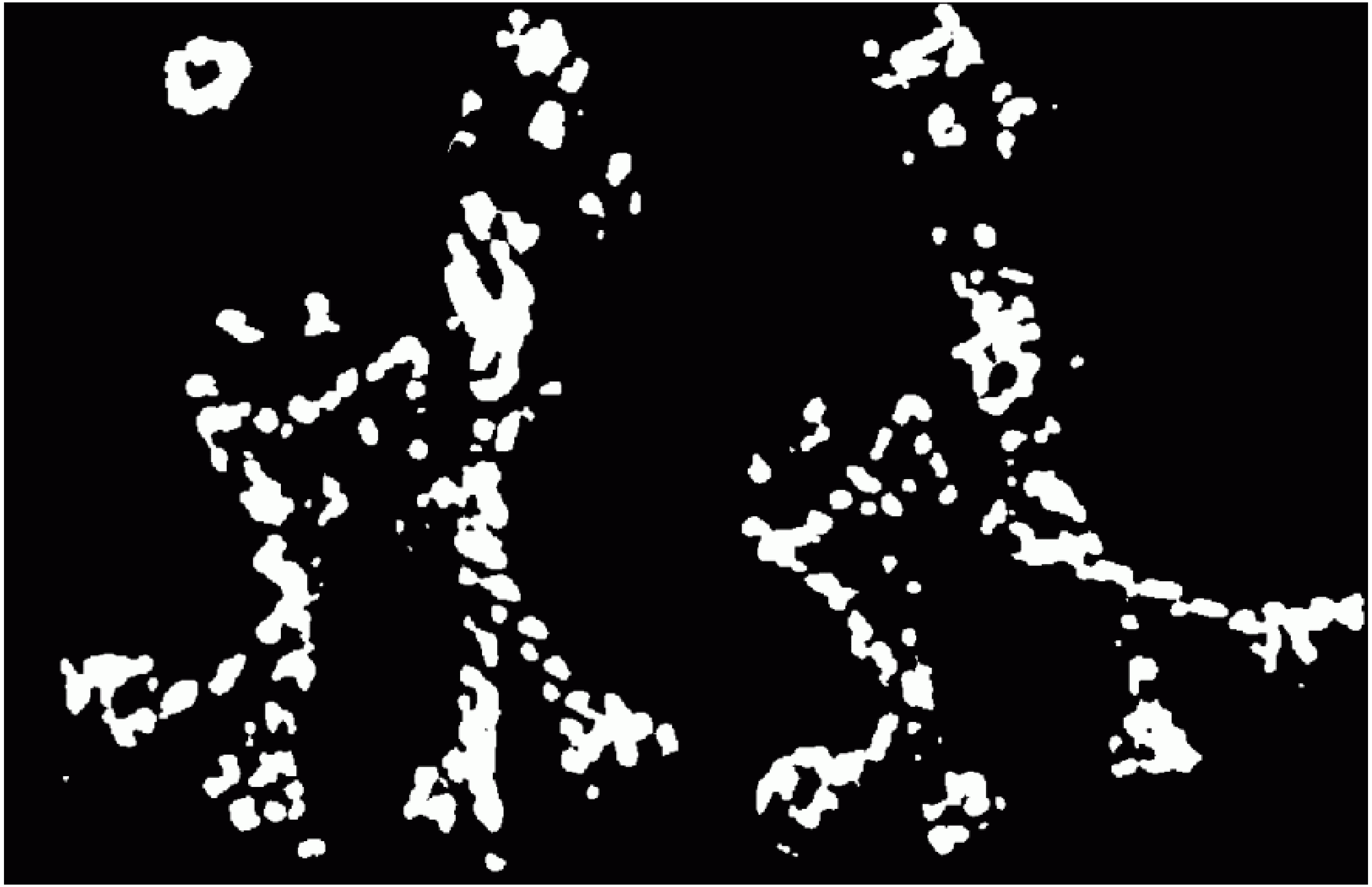
Harris Detector: Steps

Compute corner response R



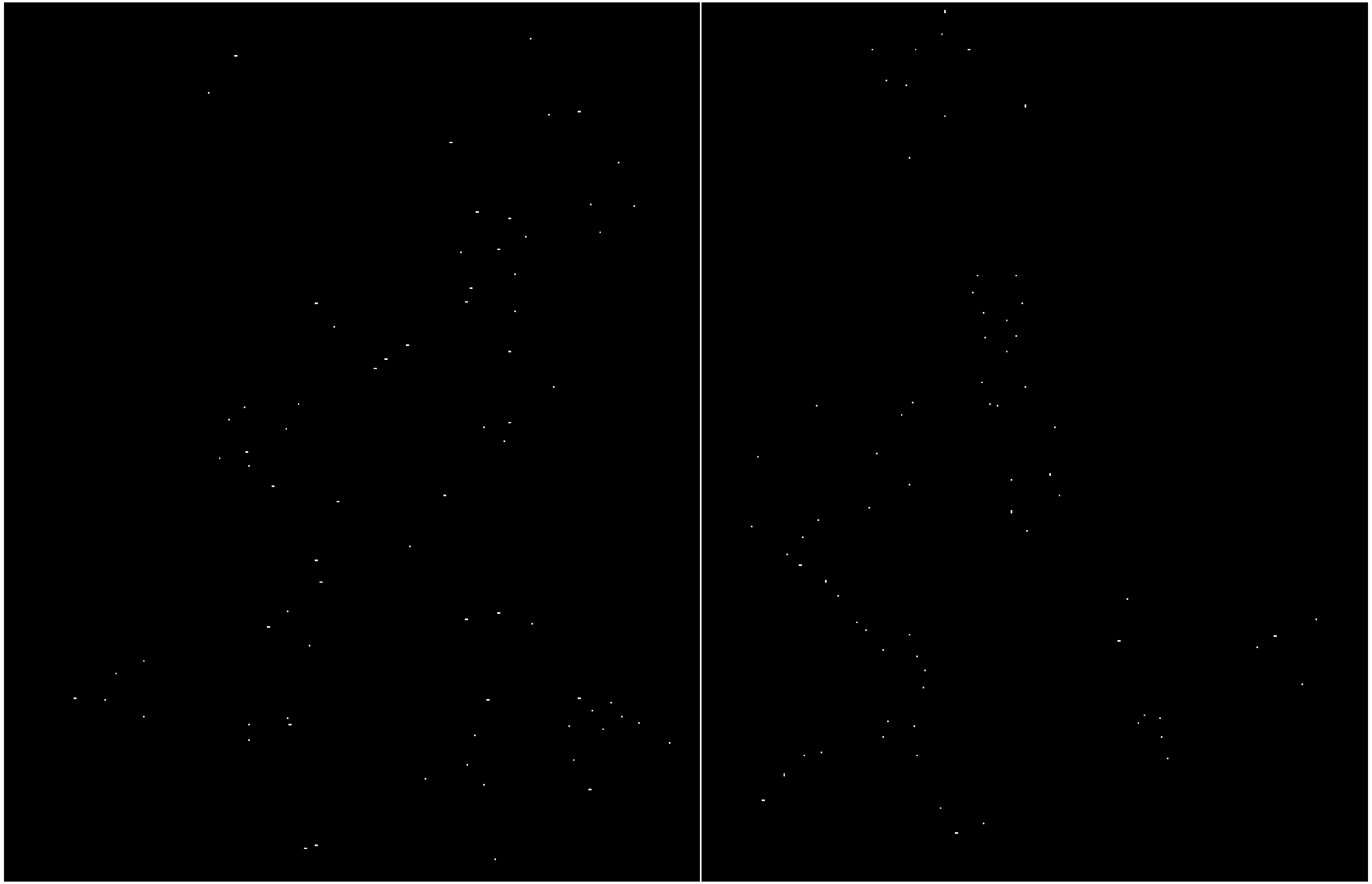
Harris Detector: Steps

Find points with large corner response: R>threshold

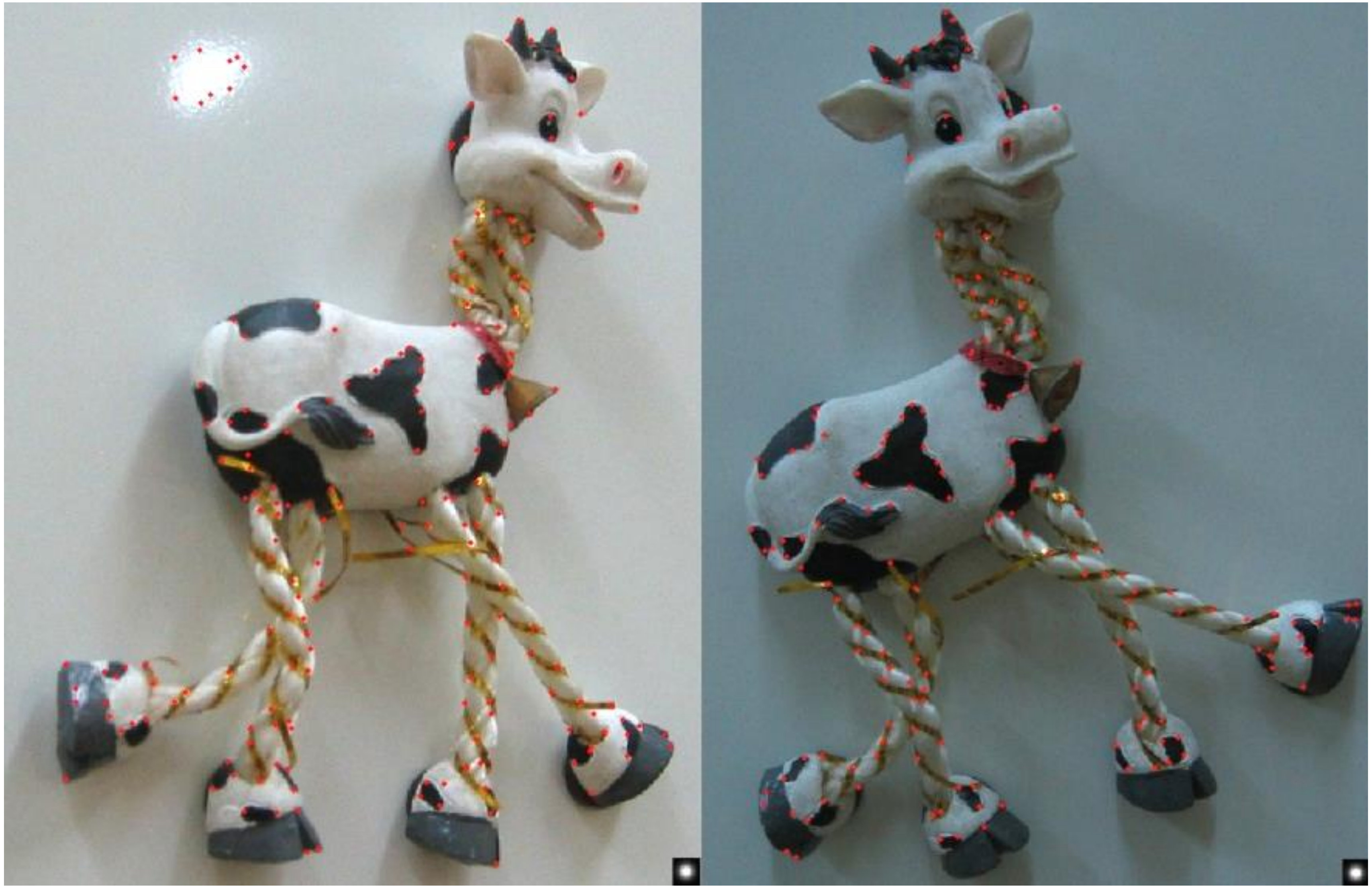


Harris Detector: Steps

Take only the points of local maxima of R

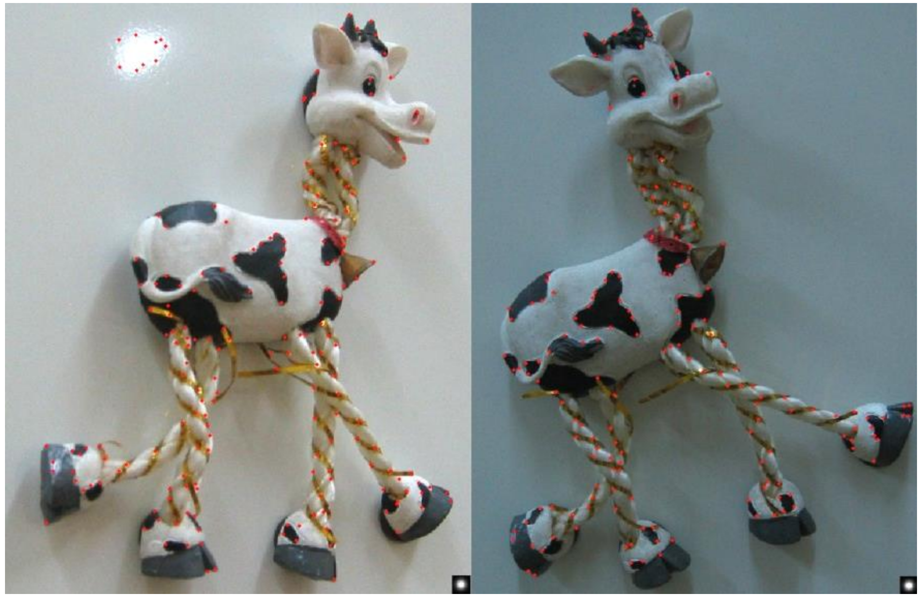


Harris Detector: Steps

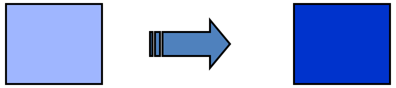


Invariance and covariance

- We want corner locations to be *invariant* to photometric transformations and *covariant* to geometric transformations
  - **Invariance:** image is transformed and corner locations do not change
  - **Covariance:** if we have two transformed versions of the same image, features should be detected in corresponding locations



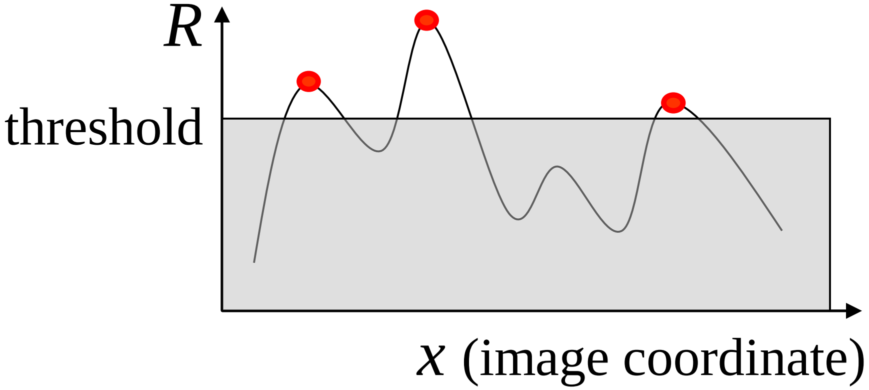
Affine intensity change

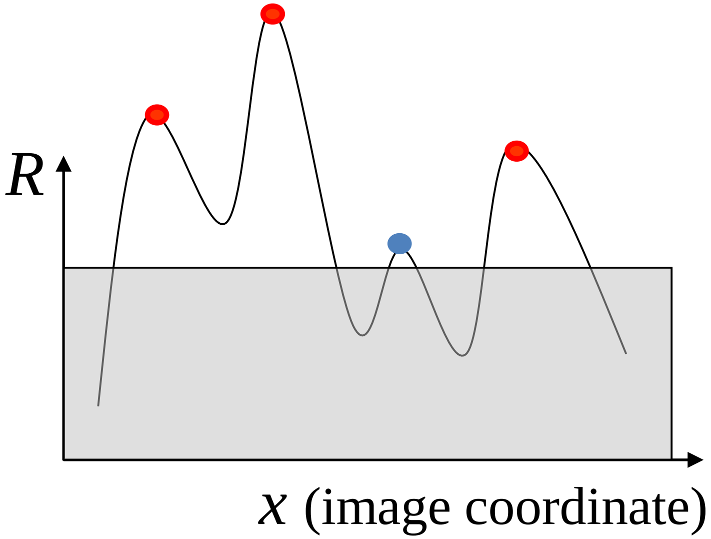


$$I \rightarrow a\,I + b$$

- Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$

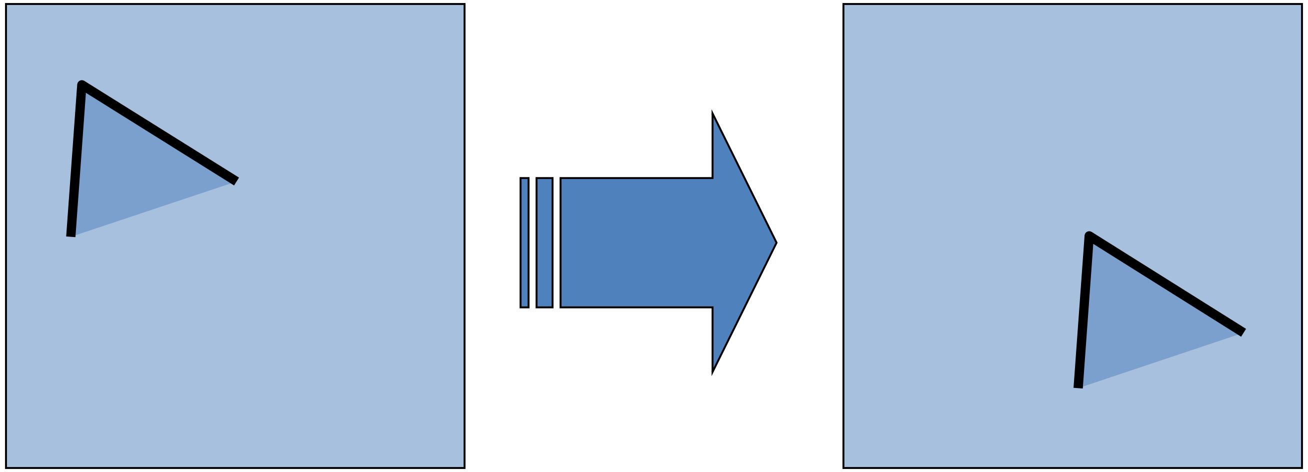
- Intensity scaling: $I \rightarrow a\,I$





Partially invariant to affine intensity change

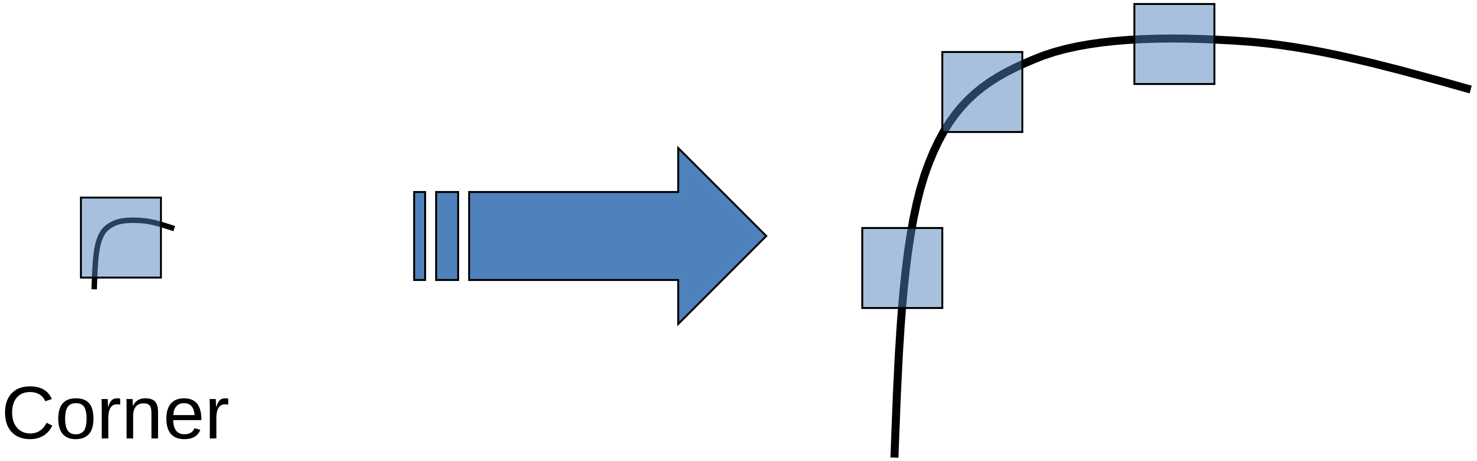
Image translation



- Derivatives and window function are shift-invariant

Corner location is covariant w.r.t. translation

Scaling

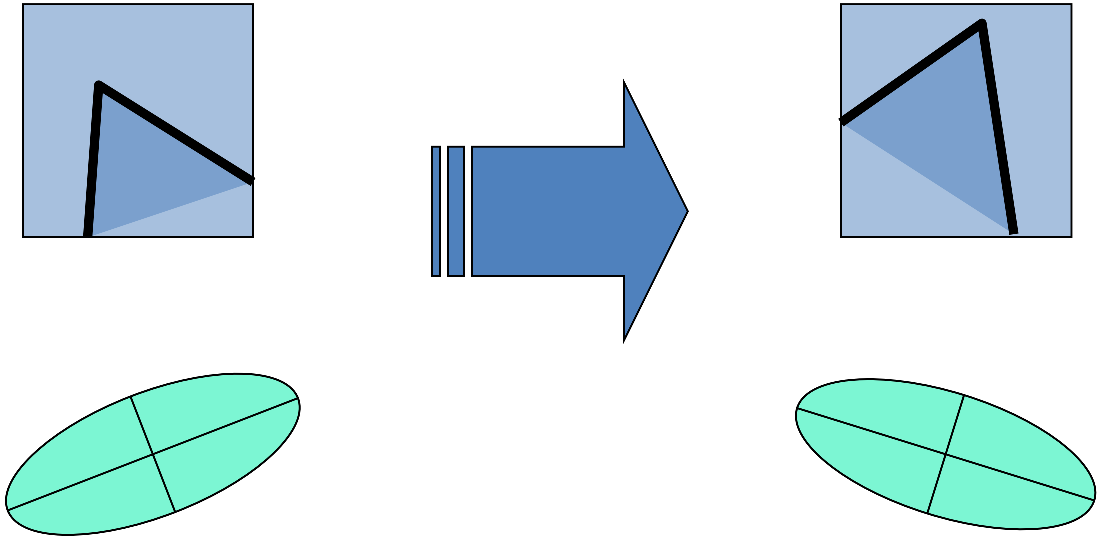


Corner

All points will
be classified
as edges

Corner location is not covariant to scaling!

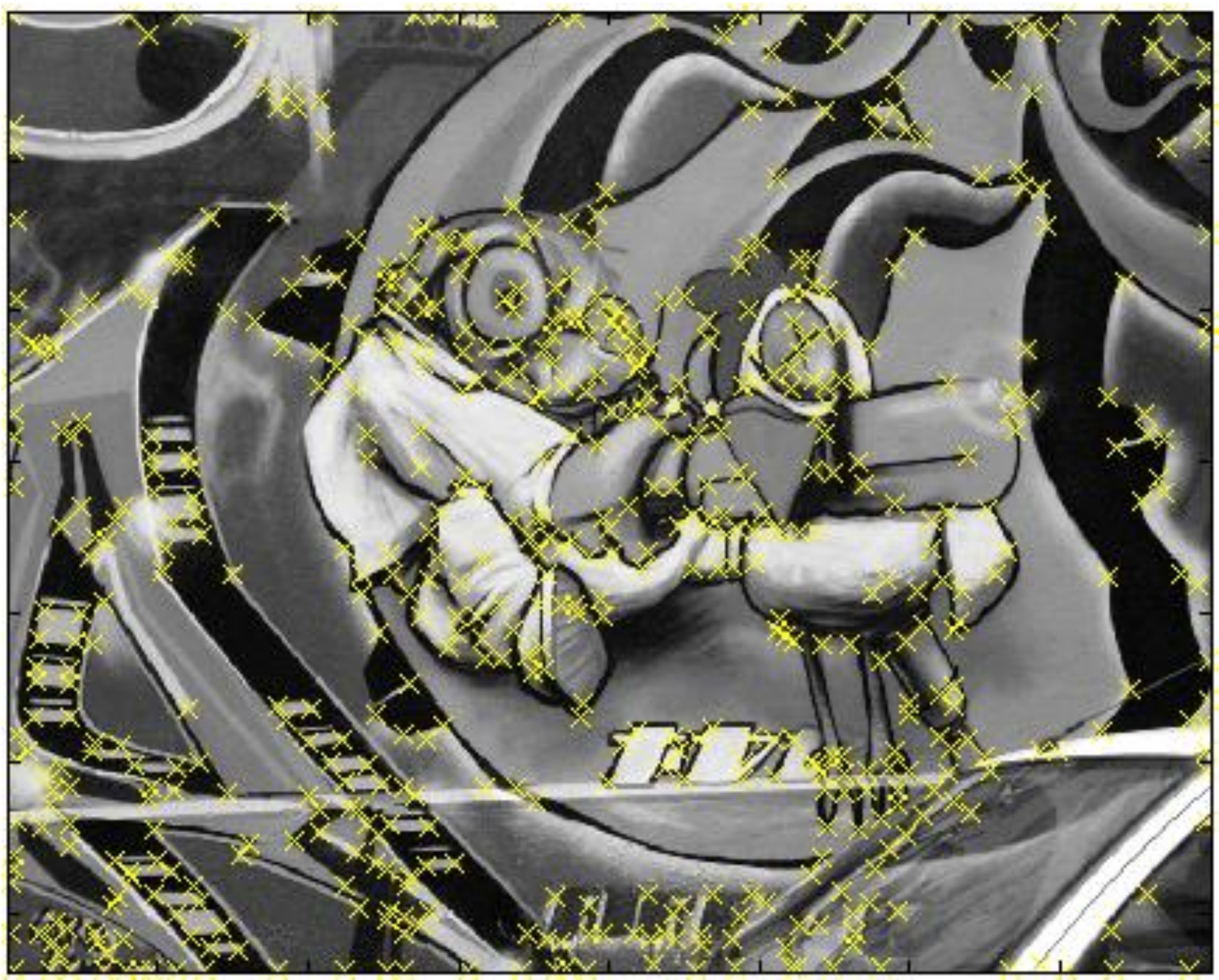
Image rotation



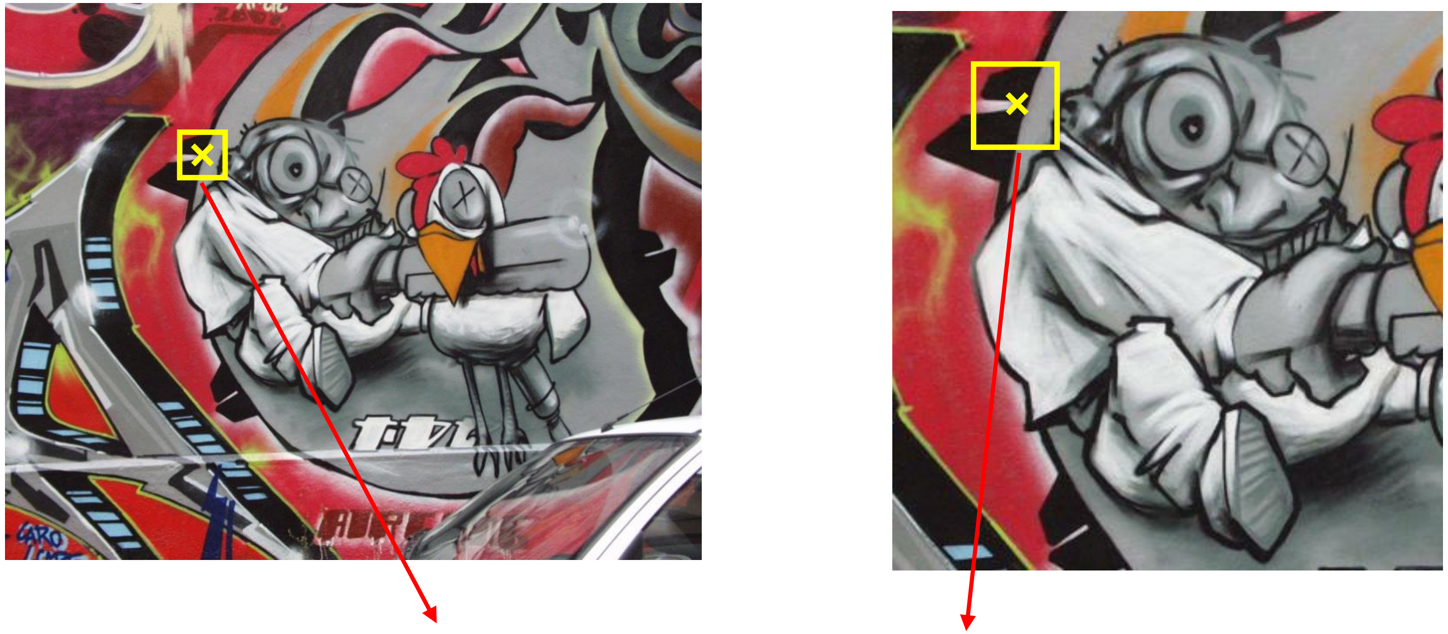
Second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner location is covariant w.r.t. rotation

So far: can localize in x-y, but not scale



Automatic Scale Selection



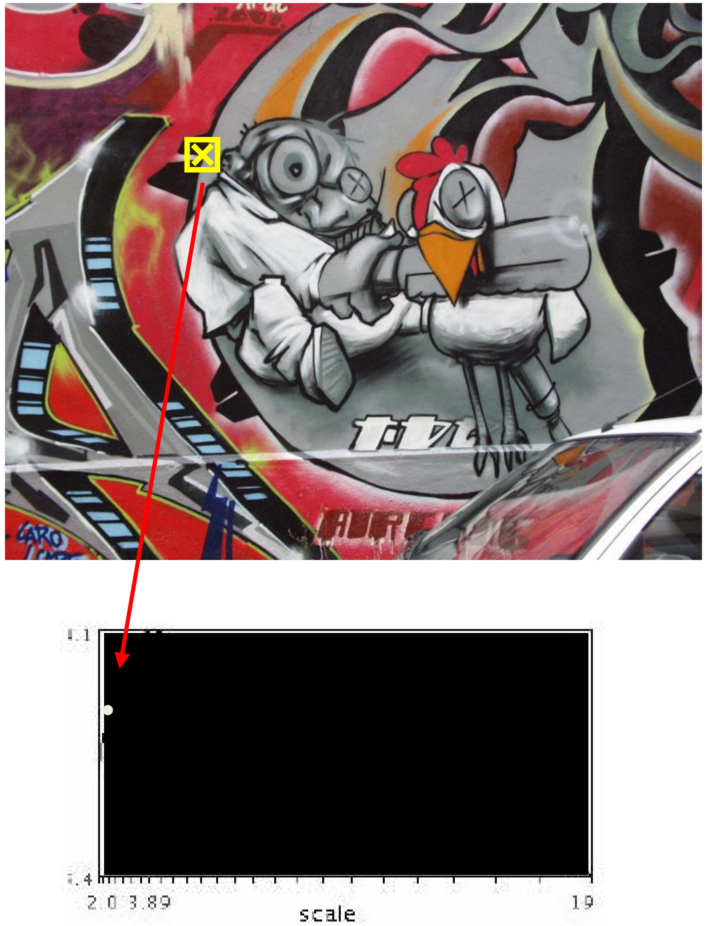
$$f(I_{i_1\ldots i_m}(x,\sigma)) \;=\; f(I_{i_1\ldots i_m}(x',\sigma'))$$

How to find corresponding patch sizes?

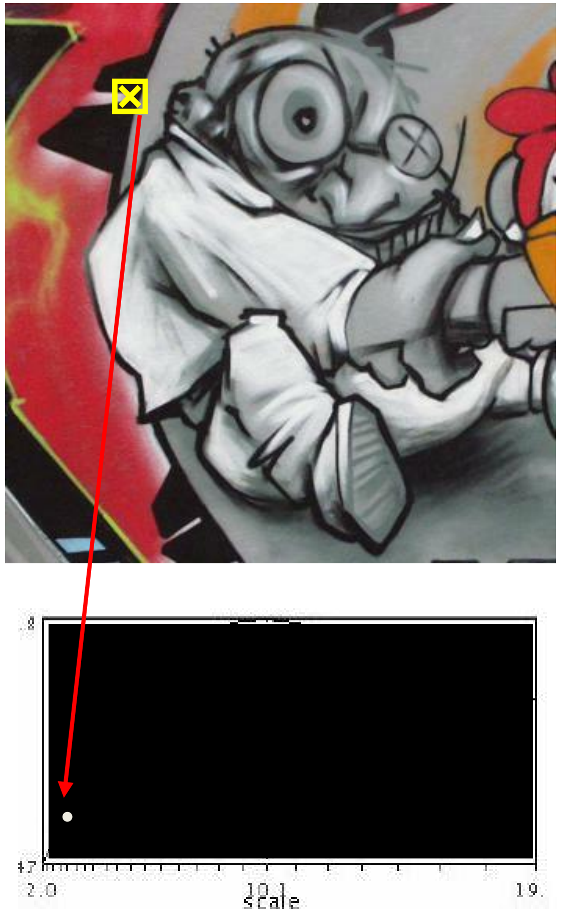
K. Grauman, B. Leibe

Automatic Scale Selection

- Function responses for increasing scale (scale signature)



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

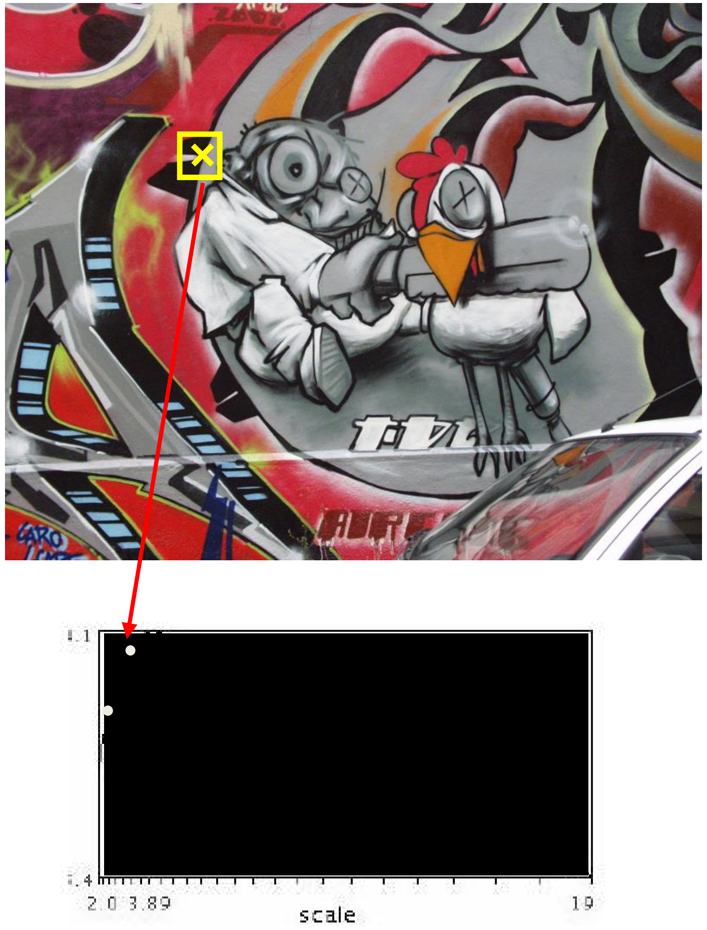


$f(I_{i_1 \dots i_m}(x', \sigma))$

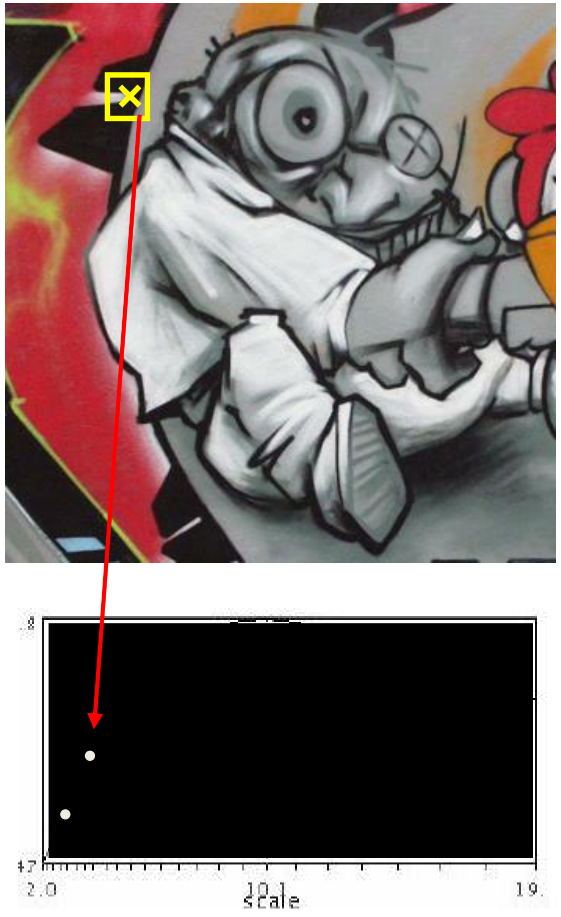
K. Grauman, B. Leibe

Automatic Scale Selection

- Function responses for increasing scale (scale signature)



$f(I_{i_1 \ldots i_m}(x,\sigma))$

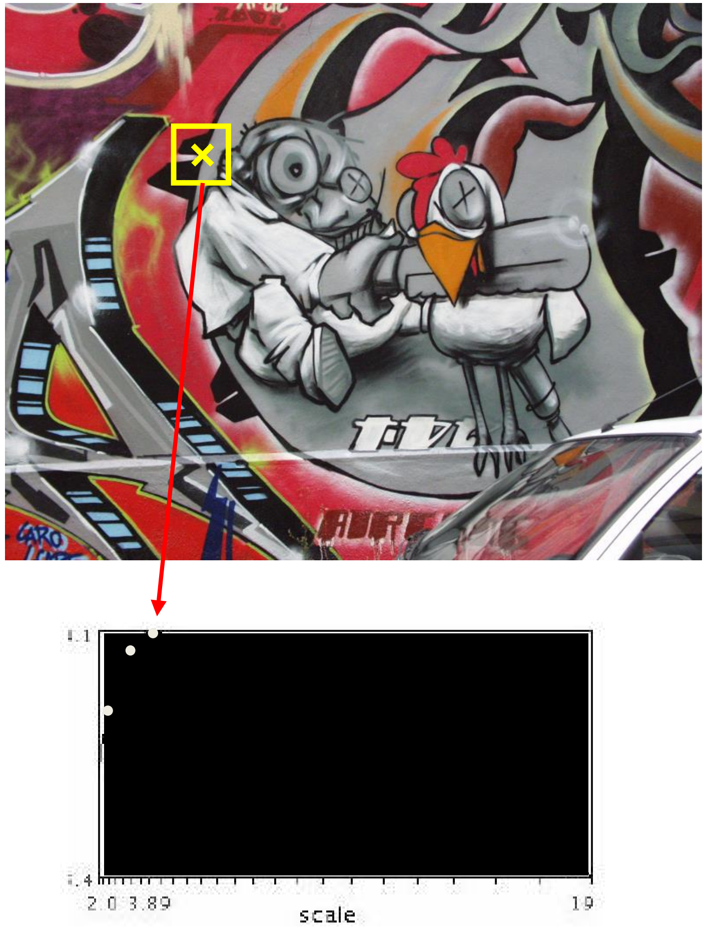


$f(I_{i_1 \ldots i_m}(x',\sigma))$

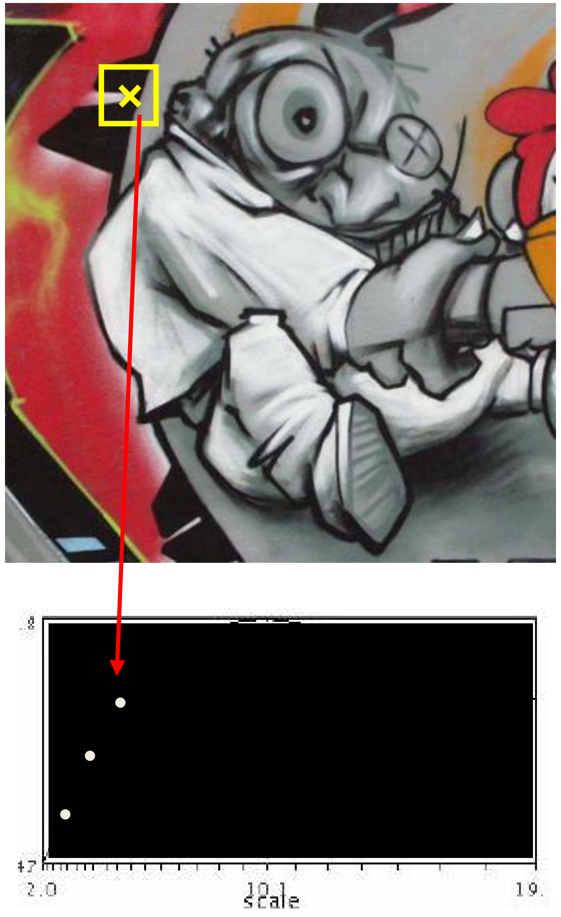
K. Grauman, B. Leibe

Automatic Scale Selection

- Function responses for increasing scale (scale signature)



$f(I_{i_1\ldots i_m}(x,\sigma))$

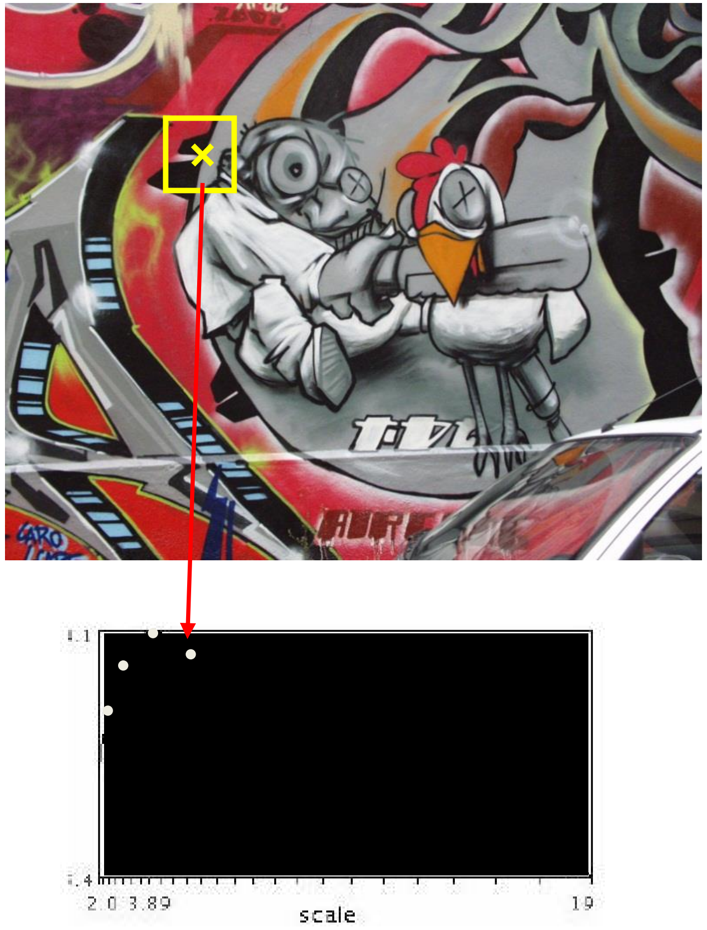


$f(I_{i_1\ldots i_m}(x',\sigma))$

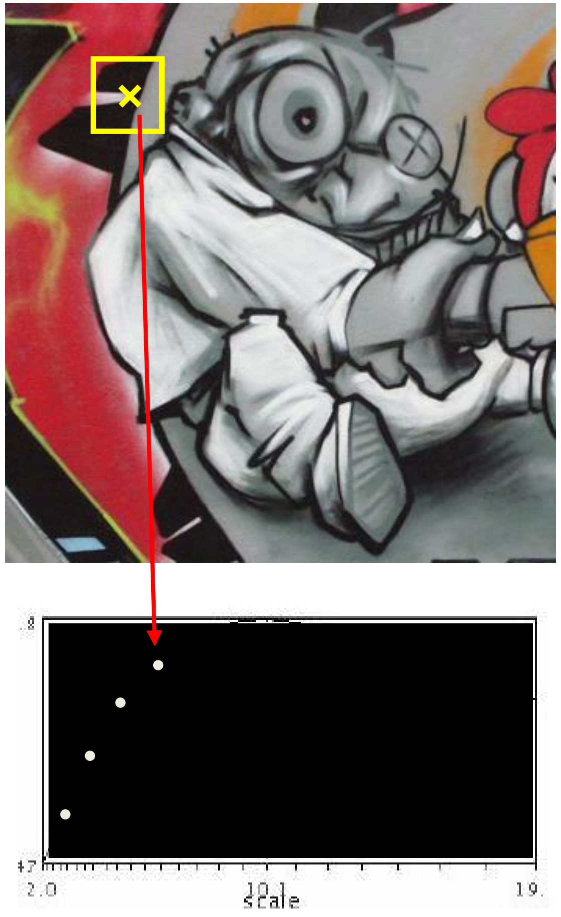
K. Grauman, B. Leibe

Automatic Scale Selection

- Function responses for increasing scale (scale signature)



$f(I_{i_1 \ldots i_m}(x, \sigma))$

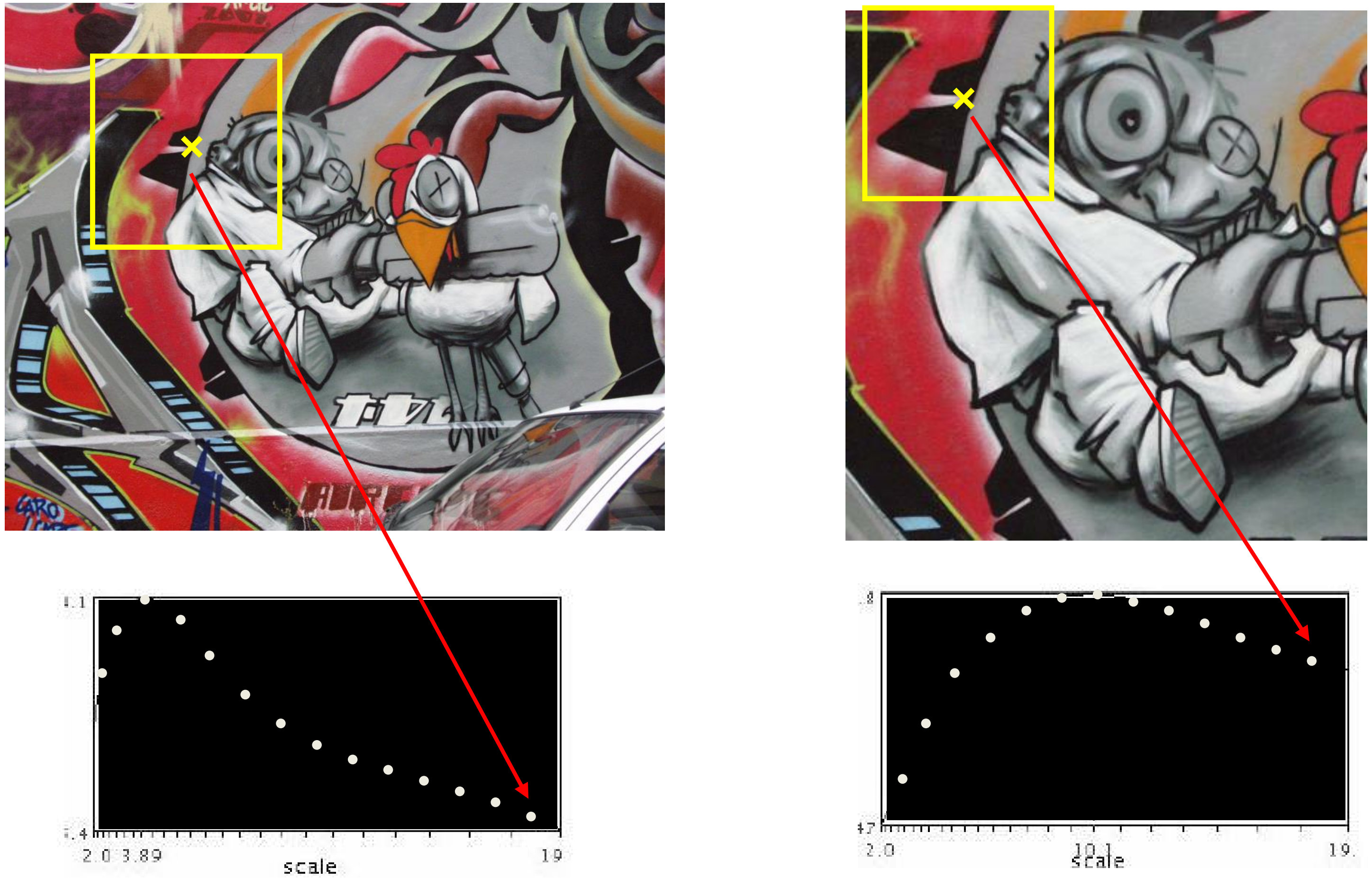


$f(I_{i_1 \ldots i_m}(x', \sigma))$

K. Grauman, B. Leibe

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

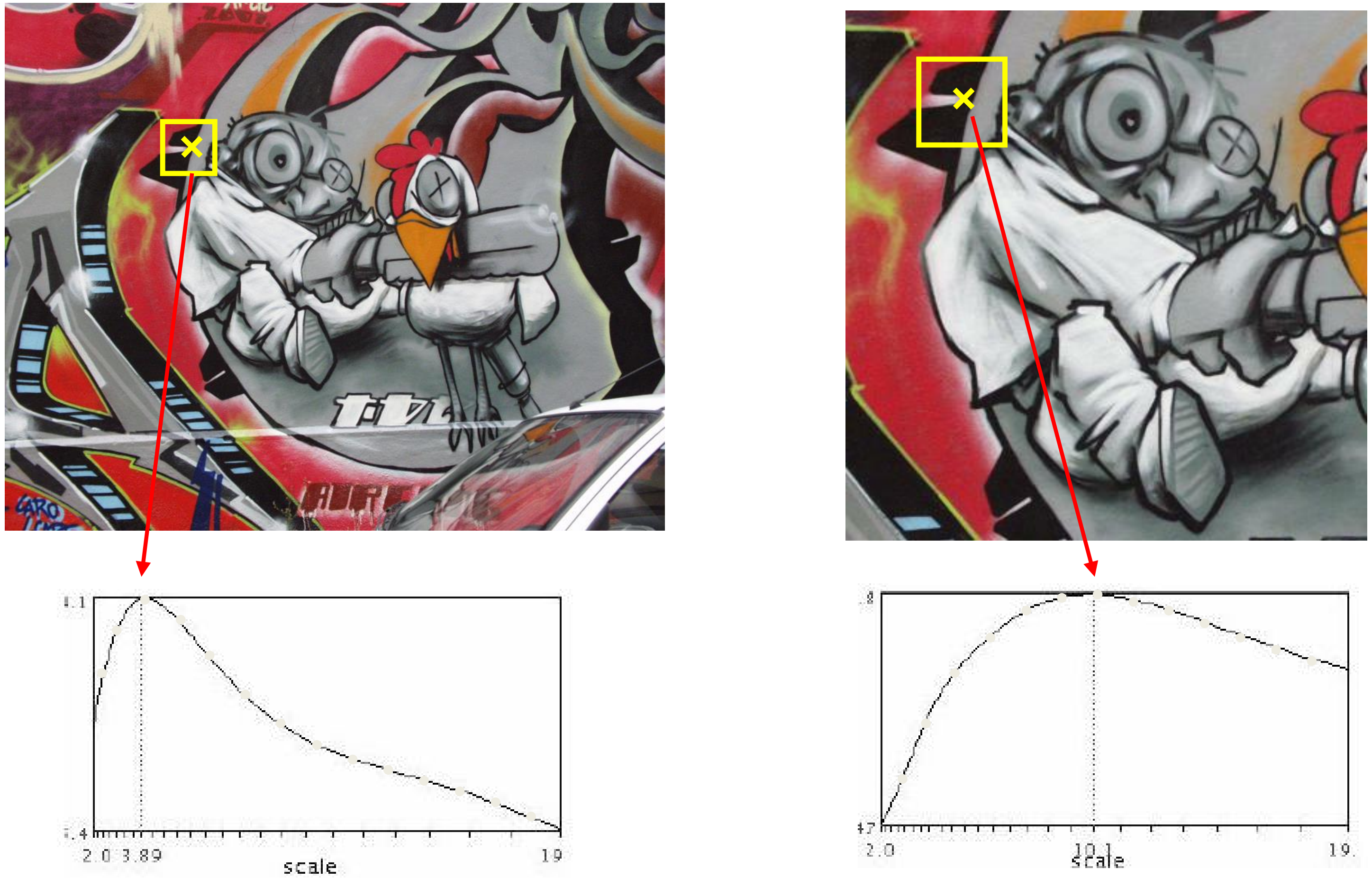


$f(I_{i_1 \ldots i_m}(x,\sigma))$ $f(I_{i_1 \ldots i_m}(x',\sigma))$

K. Grauman, B. Leibe

Automatic Scale Selection

- Function responses for increasing scale (scale signature)



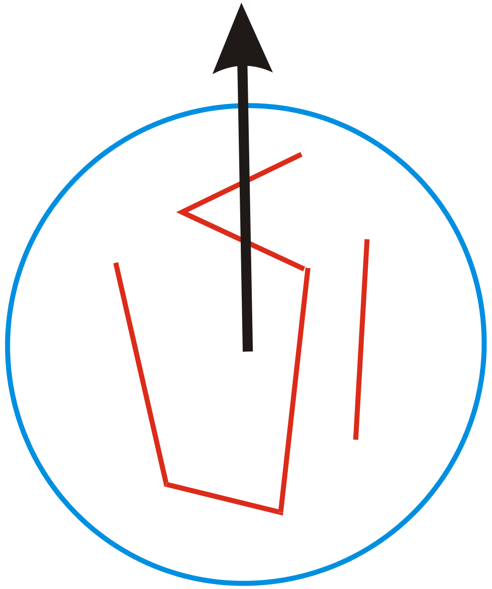
$f(I_{i_1 \ldots i_m}(x, \sigma))$ $f(I_{i_1 \ldots i_m}(x', \sigma'))$

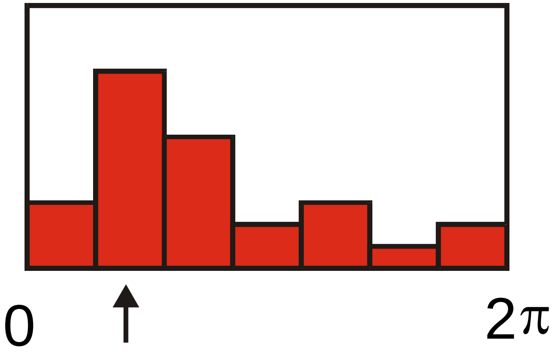
K. Grauman, B. Leibe

Orientation Normalization

- Compute orientation histogram
- Select dominant orientation
- Normalize: rotate to fixed orientation

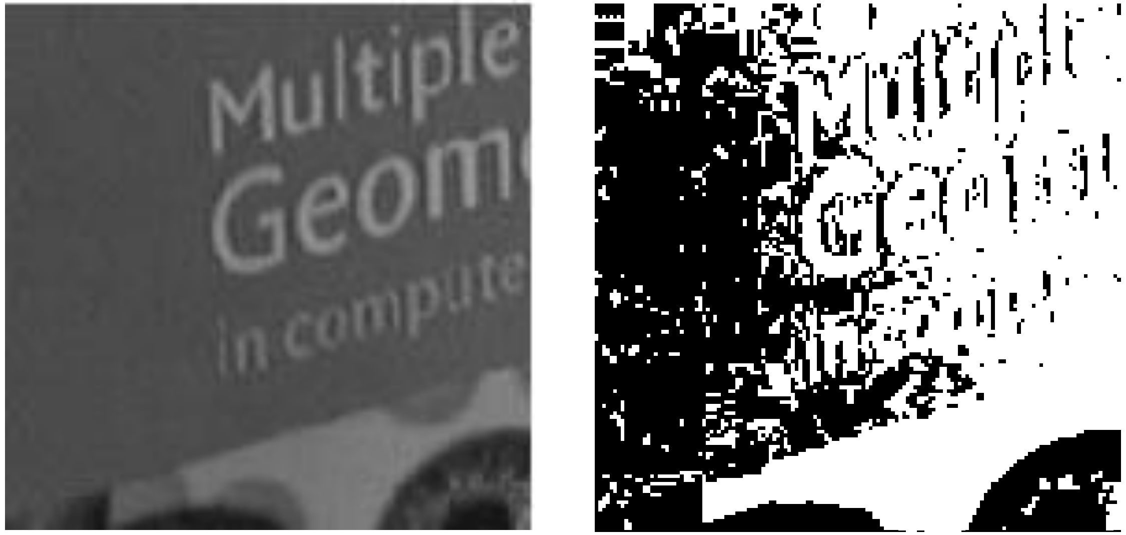
[Lowe, SIFT, 1999]





Maximally Stable Extremal Regions

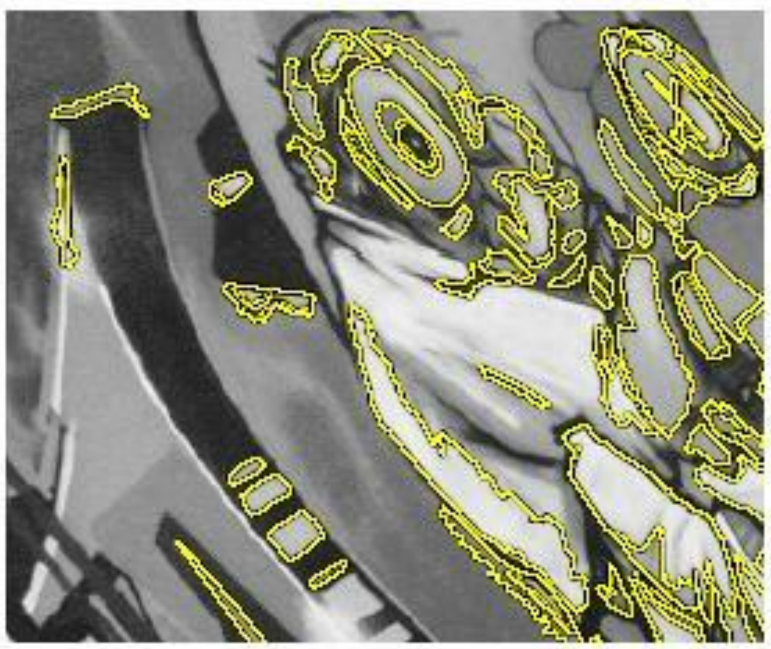
- Based on Watershed segmentation algorithm
- Select regions that stay stable over a large parameter range

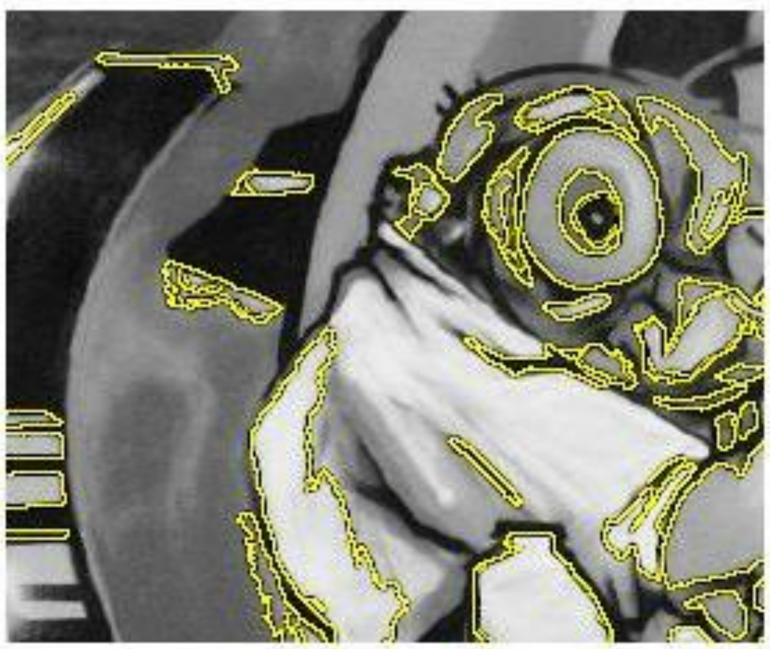


"Robust Wide Baseline Stereo from Maximally Stable Extremal Regions",
Matas, Chum, Urban, and Pajdla, BMVC 2002
6k+ citations

K. Grauman, B. Leibe

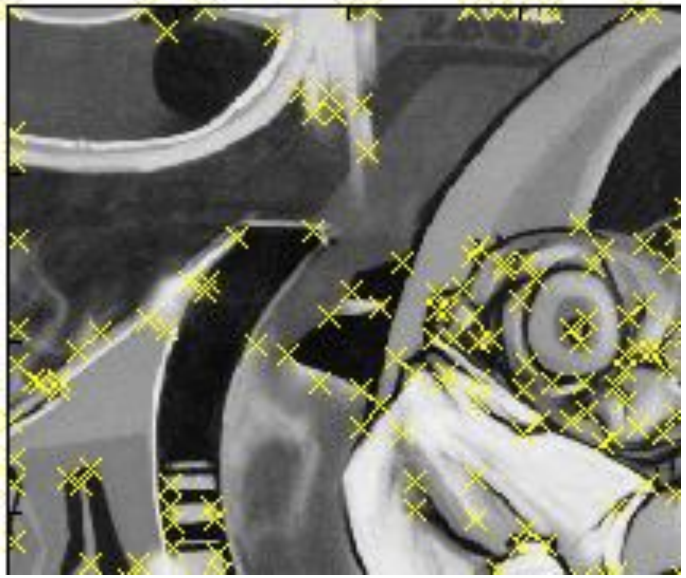
Example Results: MSER

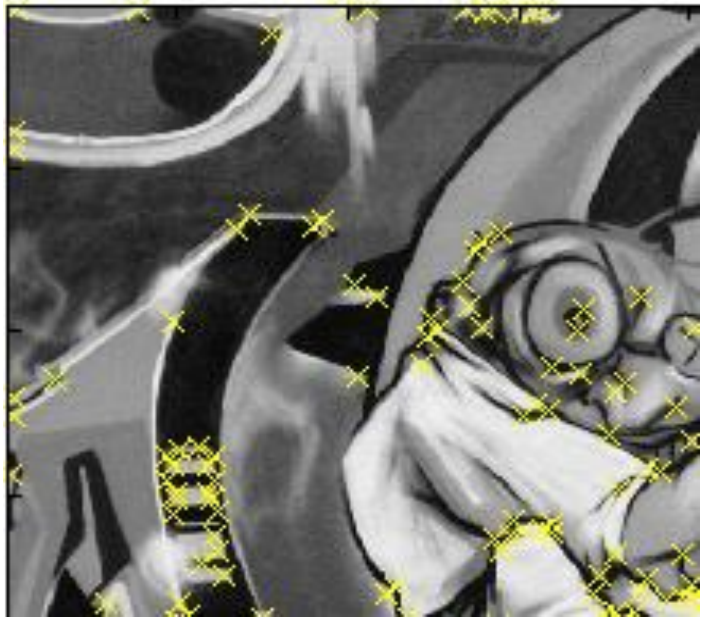


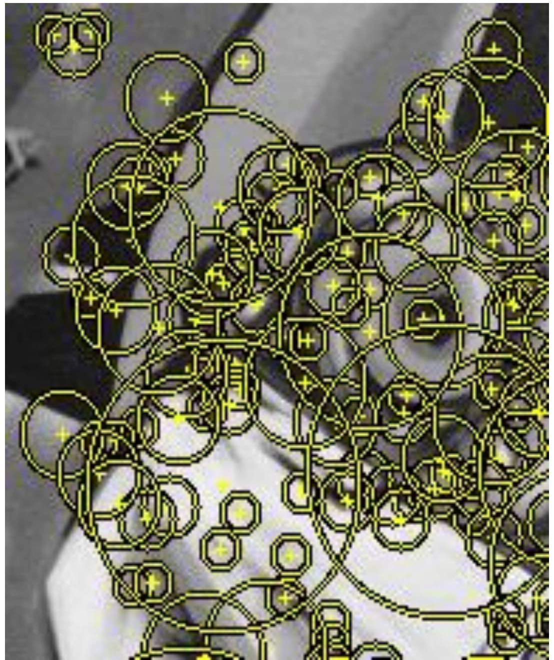


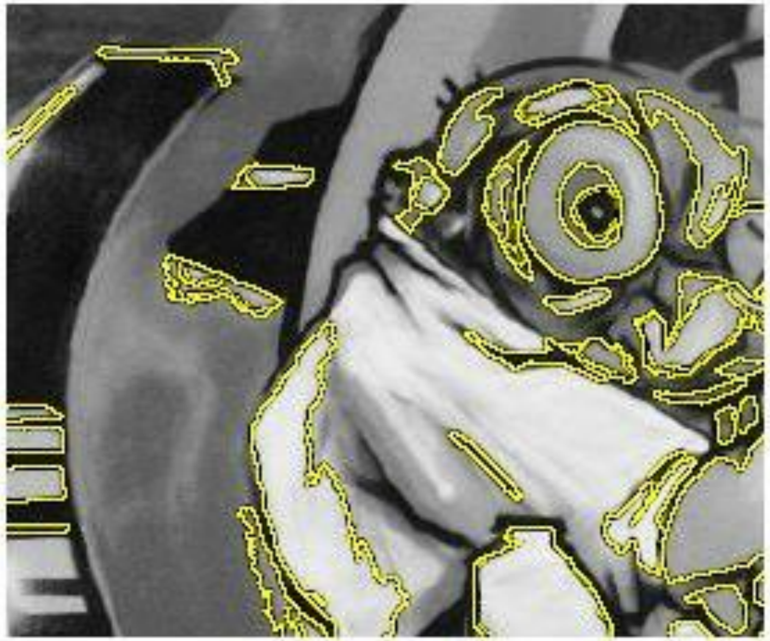
K. Grauman, B. Le

Comparison


Harris

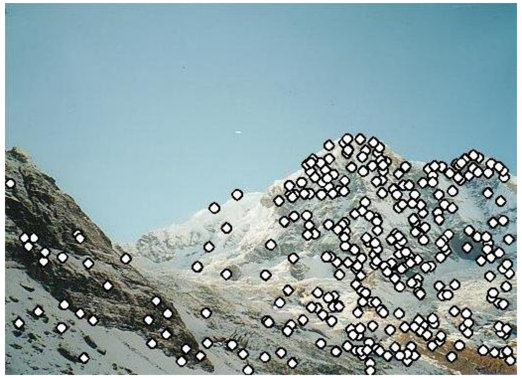

Hessian


LoG

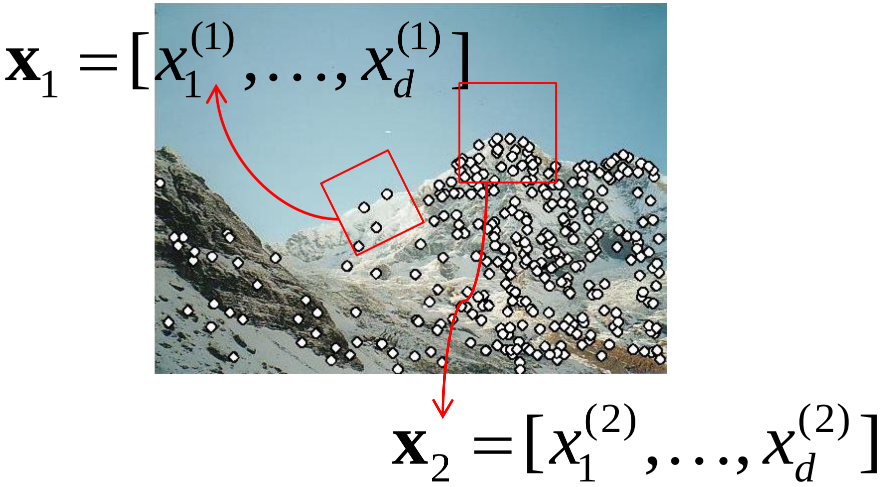

MSER

Local features: main components

1) **Detection:** Identify the interest points



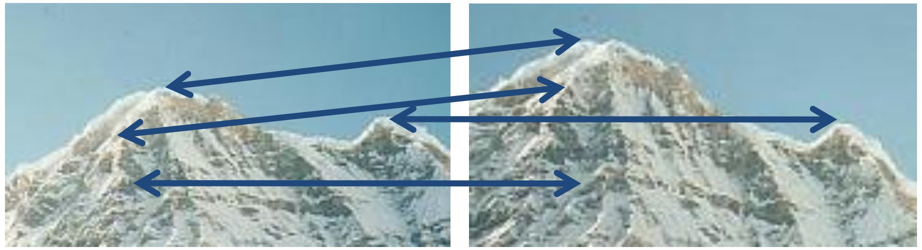
2) **Description**: Extract vector feature descriptor surrounding each interest point.



$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$

$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

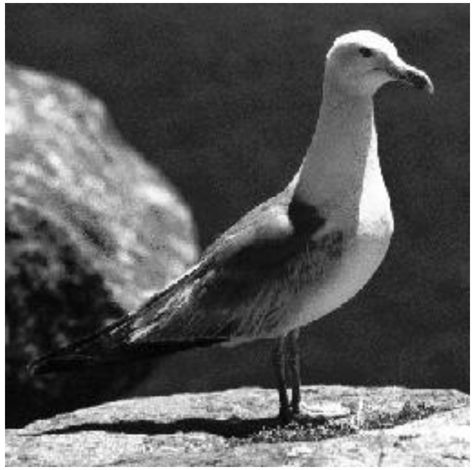
3) **Matching:** Determine correspondence between descriptors in two views

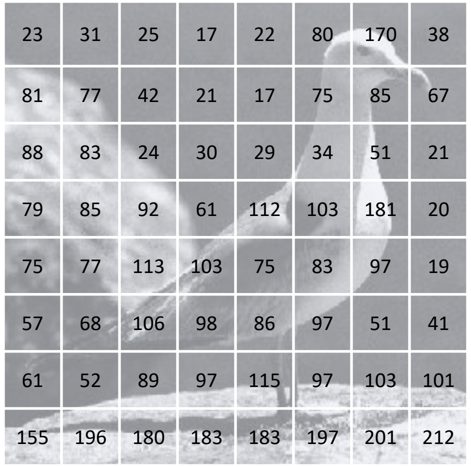


Kristen Grauman

Image representations

- Templates
  - Intensity, color, gradients, etc.
  - Keeps spatial layout





23	31	25	17	22	80	170	38
81	77	42	21	17	75	85	67
88	83	24	30	29	34	51	21
79	85	92	61	112	103	181	20
75	77	113	103	75	83	97	19
57	68	106	98	86	97	51	41
61	52	89	97	115	97	103	101
155	196	180	183	183	197	201	212

- Histograms
  - Distribution of intensity, color, texture, SIFT descriptors, etc.
  - Discards spatial layout

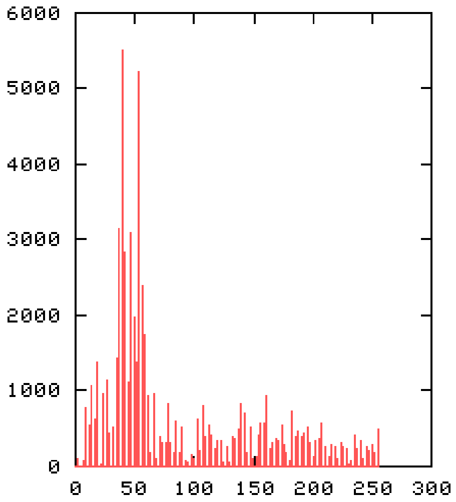
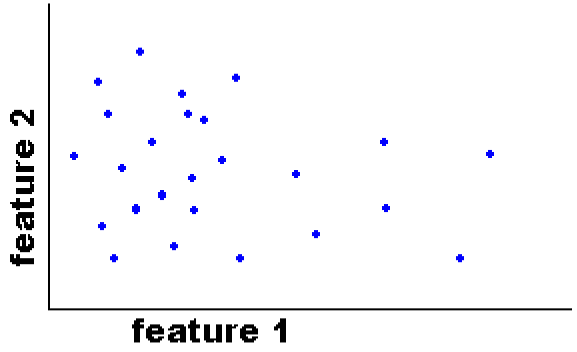
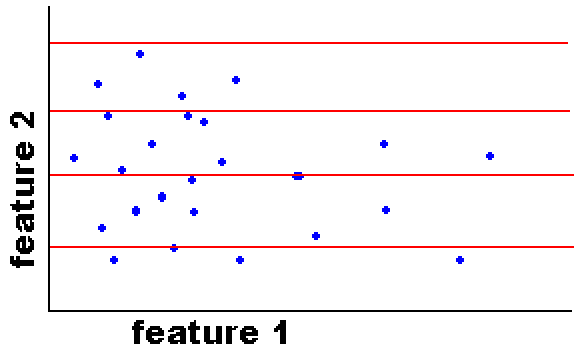
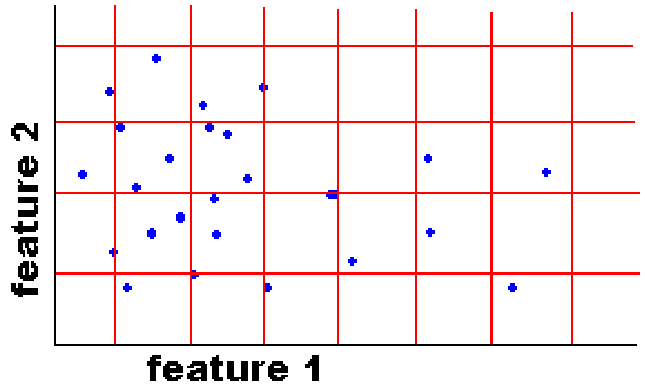


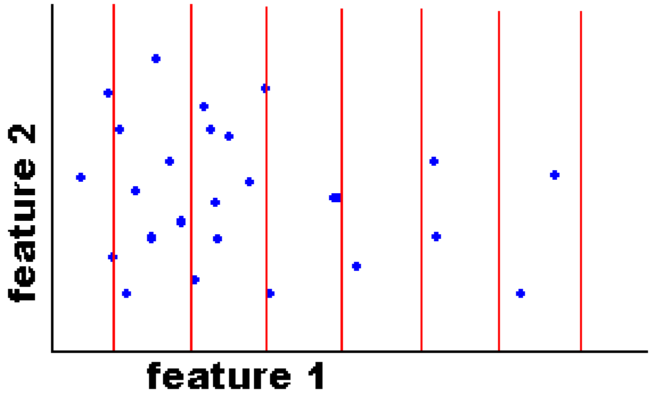
Image Representations: Histograms

Histogram: Probability or count of data in each bin









- Joint histogram
  - Requires lots of data
  - Loss of resolution to avoid empty bins

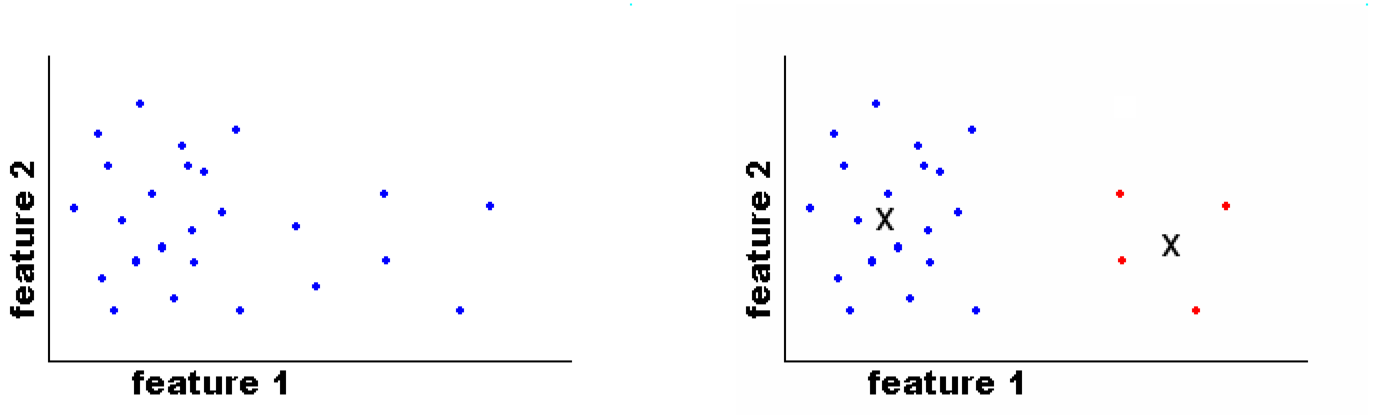
Marginal histogram

- Requires independent features
- More data/bin than joint histogram

Images from Dave Kauchak

Image Representations: Histograms

Clustering



Use the same cluster centers for all images

Images from Dave Kauchak

Computing histogram distance

$$\text{histint}(h_i, h_j) = 1 - \sum_{m=1}^{K} \min\left(h_i(m), h_j(m)\right)$$

Histogram intersection (assuming normalized histograms)

$$\chi^2(h_i, h_j) = \frac{1}{2} \sum_{m=1}^{K} \frac{[h_i(m) - h_j(m)]^2}{h_i(m) + h_j(m)}$$

Chi-squared Histogram matching distance

Cars found by color histogram matching using chi-squared

Histograms: Implementation issues

- Quantization
  - Grids: fast but applicable only with few dimensions
  - Clustering: slower but can quantize data in higher dimensions

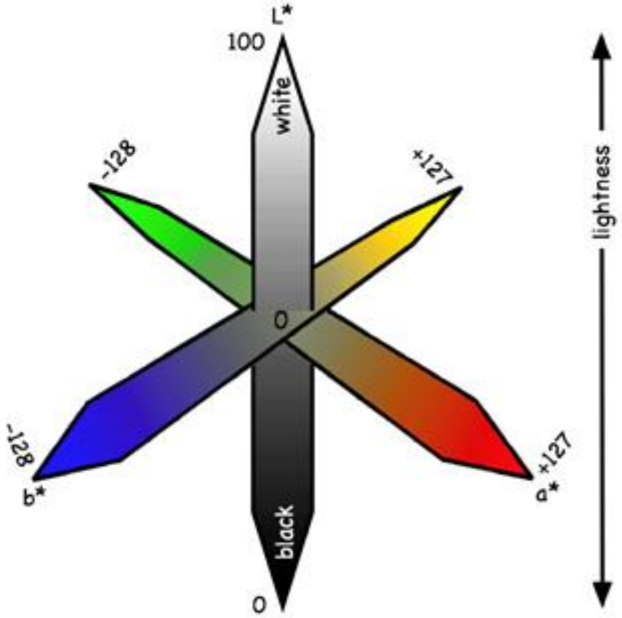
Few Bins
Need less data
Coarser representation

Many Bins
Need more data
Finer representation

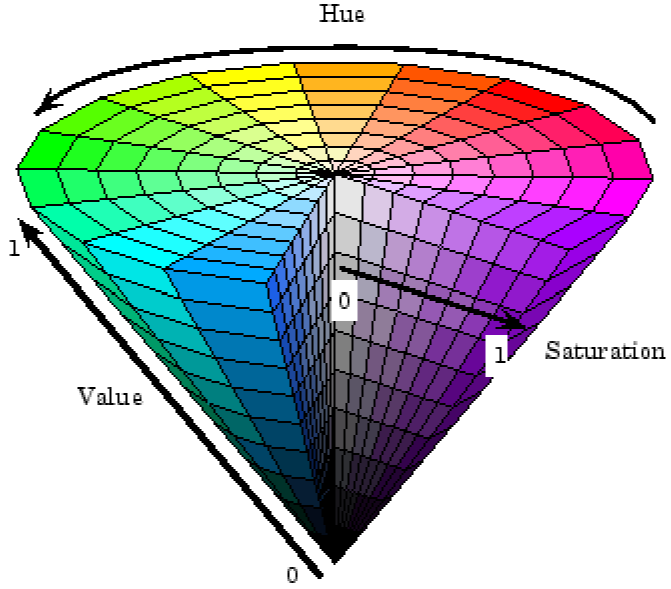
- Matching
  - Histogram intersection or Euclidean may be faster
  - Chi-squared often works better
  - Earth mover’s distance is good for when nearby bins represent similar values

What kind of things do we compute histograms of?

- Color



L*a*b* color space

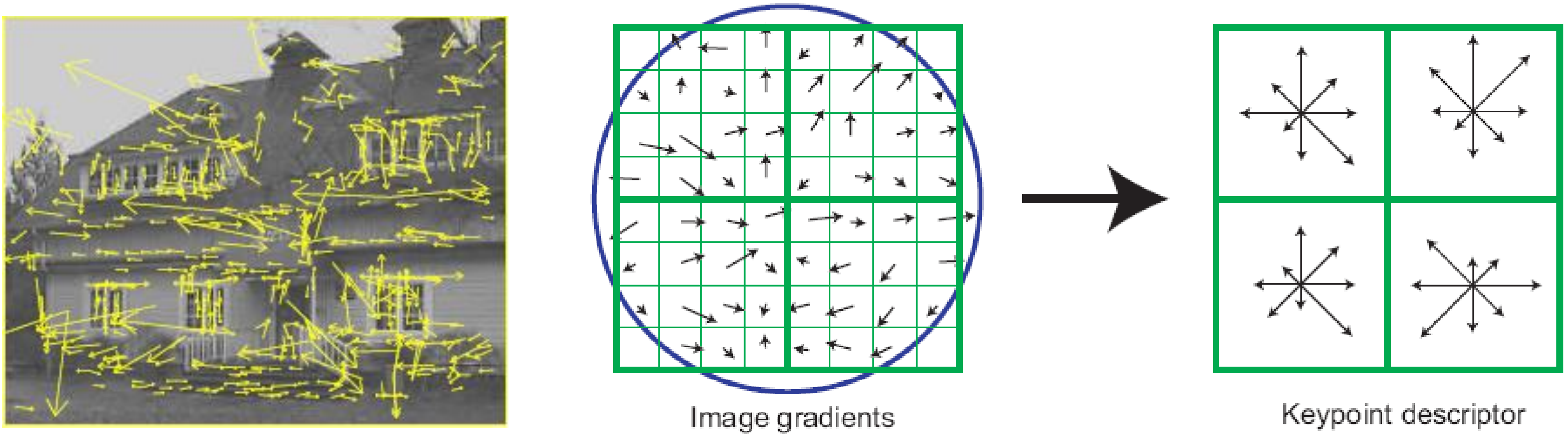


HSV color space

- Texture (filter banks or HOG over regions)

What kind of things do we compute histograms of?

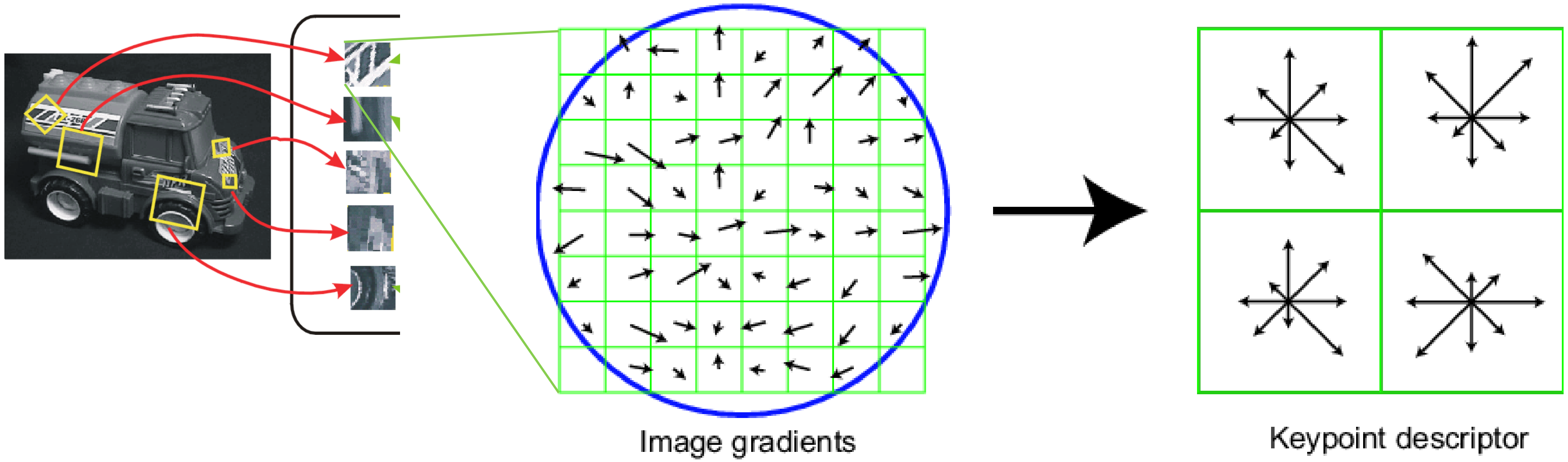
- Histograms of oriented gradients



SIFT – Lowe IJCV 2004

SIFT vector formation

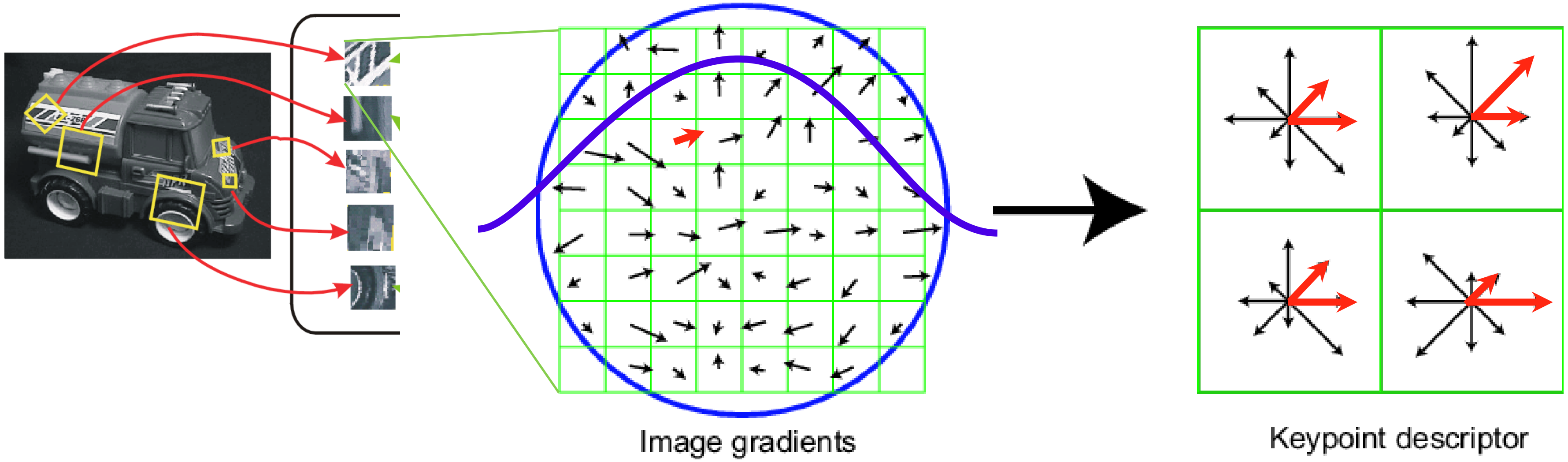
- 4x4 array of gradient orientation histogram weighted by magnitude
- 8 orientations x 4x4 array = 128 dimensions
- Motivation: some sensitivity to spatial layout, but not too much.



showing only 2x2 here, but typical feature would be 4x4

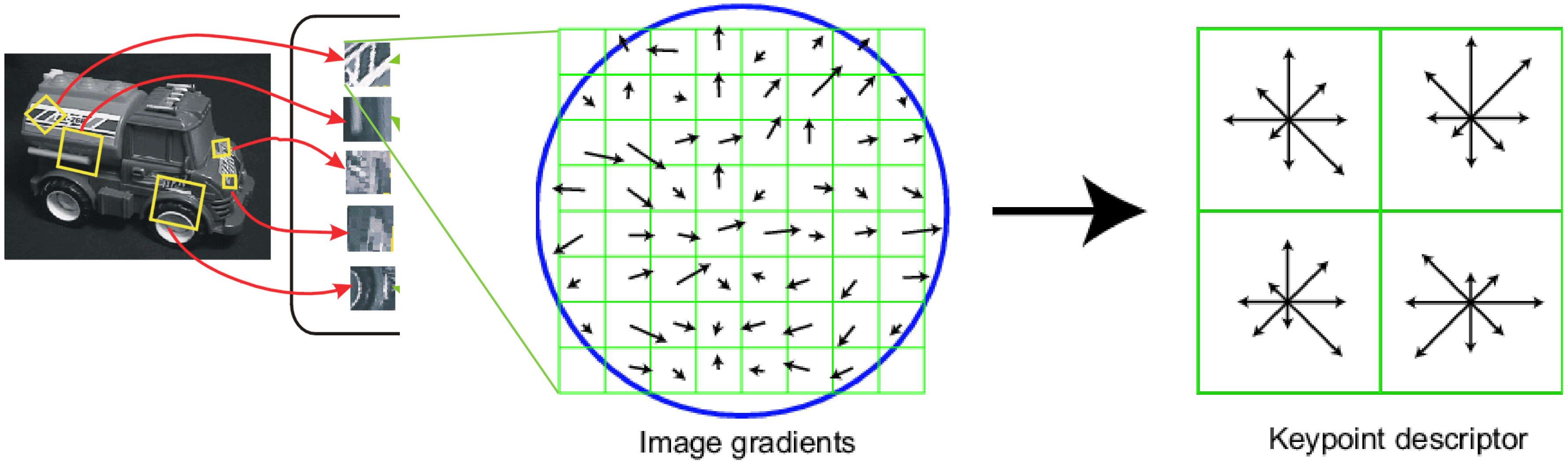
Ensure smoothness

- Gaussian weight
- Interpolation
  - a given gradient contributes to 8 bins: 4 in space times 2 in orientation

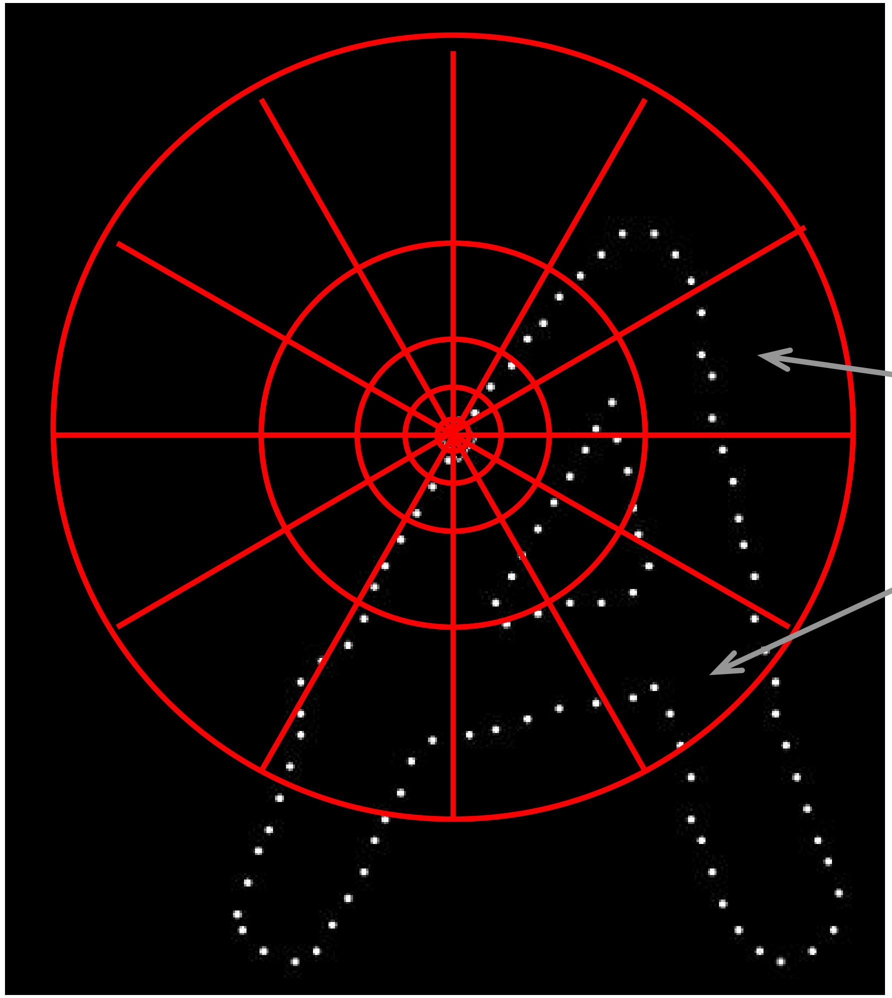


Reduce effect of illumination

- 128-dim vector normalized to 1
- Optionally, threshold gradient magnitudes to avoid excessive influence of high gradients
  - – after normalization, clamp gradients >0.2
  - – renormalize



Local Descriptors: Shape Context



Count the number of points inside each bin, e.g.:

Count = 4

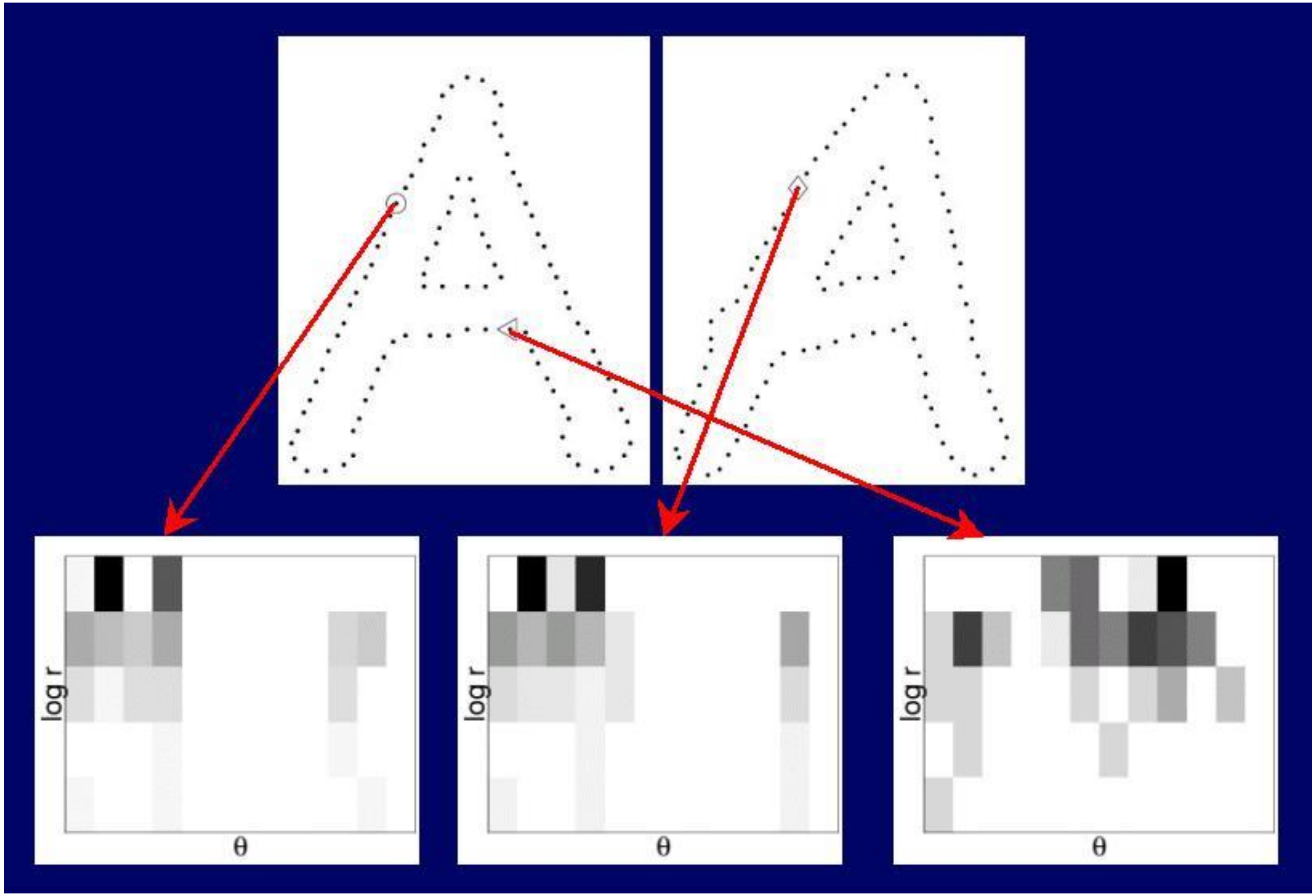
⋮

Count = 10

Log-polar binning: more precision for nearby points, more flexibility for farther points.

Belongie & Malik, ICCV 2001

Shape Context Descriptor



Self-similarity Descriptor

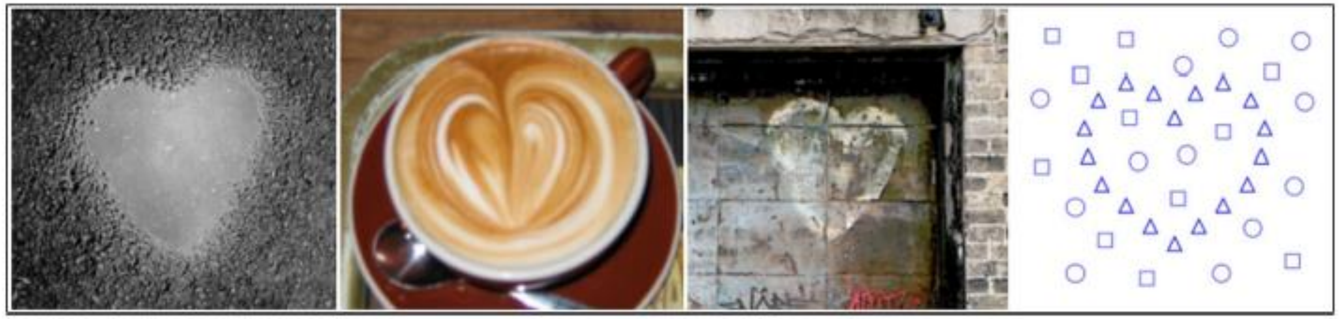
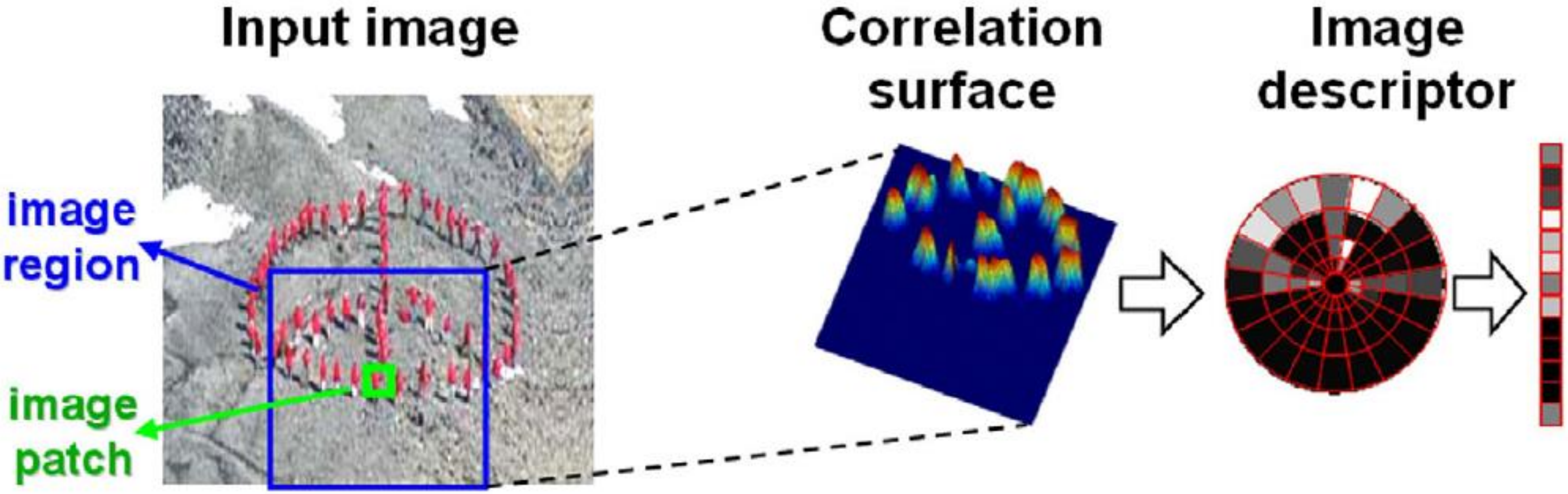


Figure 1. *These images of the same object (a heart) do NOT share common image properties (colors, textures, edges), but DO share a similar geometric layout of local internal self-similarities.*

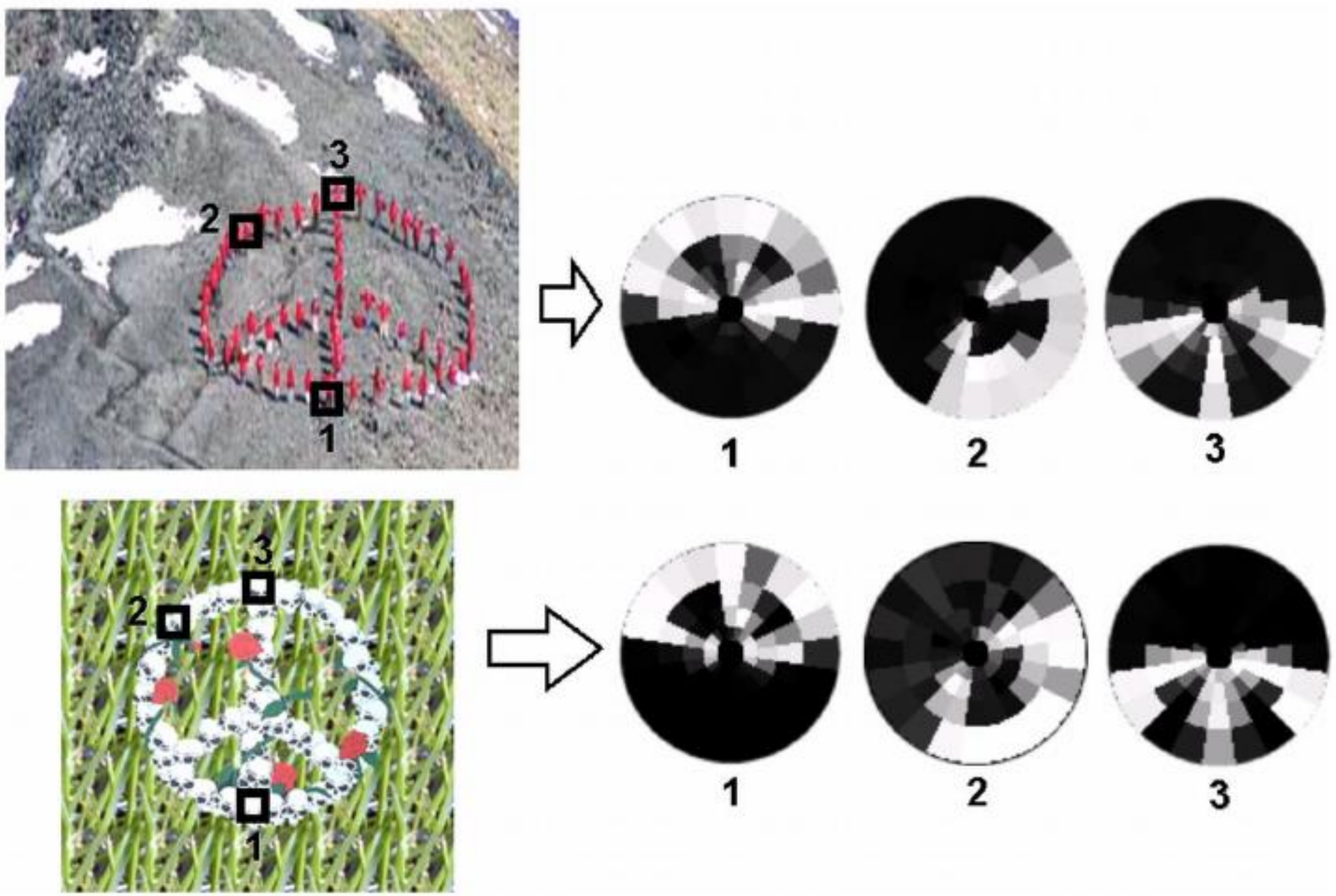
Matching Local Self-Similarities across Images and Videos, Shechtman and Irani, 2007

Self-similarity Descriptor



Matching Local Self-Similarities across Images and Videos, Shechtman and Irani, 2007

Self-similarity Descriptor



Matching Local Self-Similarities across Images and Videos, Shechtman and Irani, 2007

Learning Local Image Descriptors, Winder and Brown, CVPR 2007

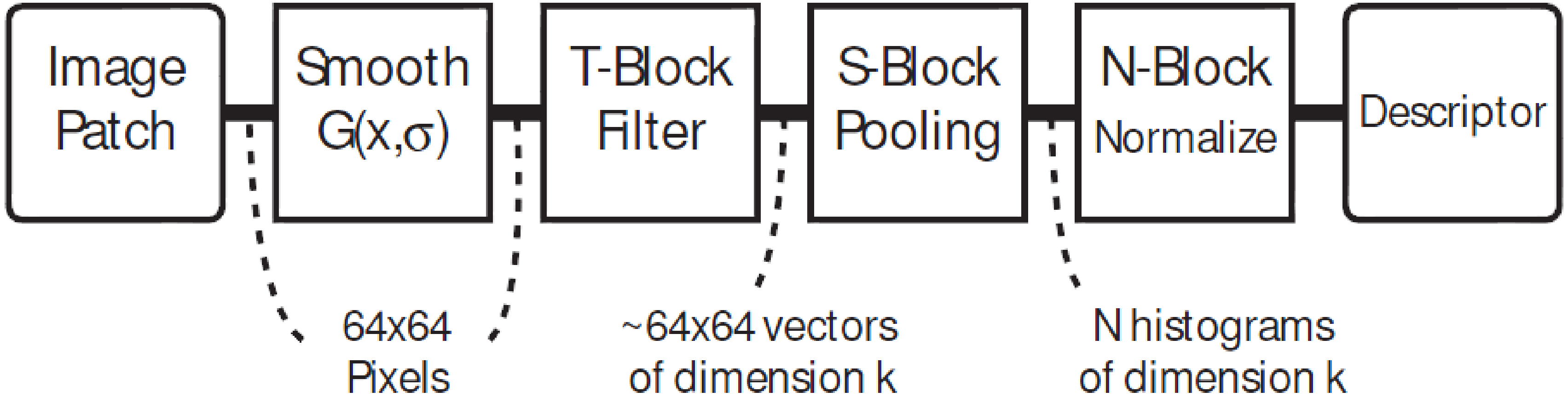


Image Patch

Smooth G(x,σ)

T-Block Filter

S-Block Pooling

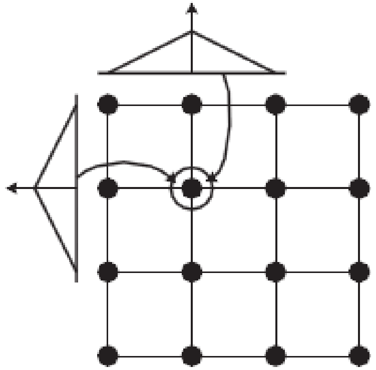
N-Block Normalize

Descriptor

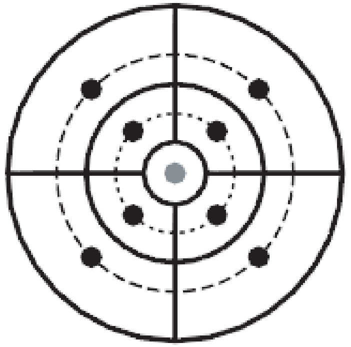
64x64 Pixels

~64x64 vectors of dimension k

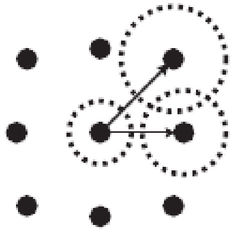
N histograms of dimension k



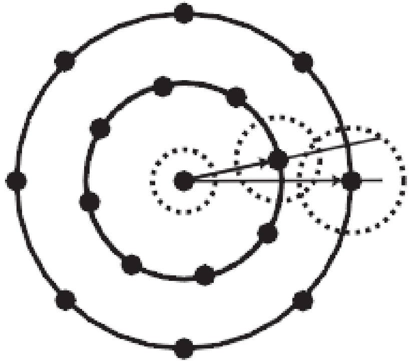
S1: SIFT grid with bilinear weights



S2: GLOH polar grid with bilinear radial and angular weights



S3: 3x3 grid with Gaussian weights



S4: 17 polar samples with Gaussian weights

Learning Local Image Descriptors, Winder and Brown, CVPR 2007

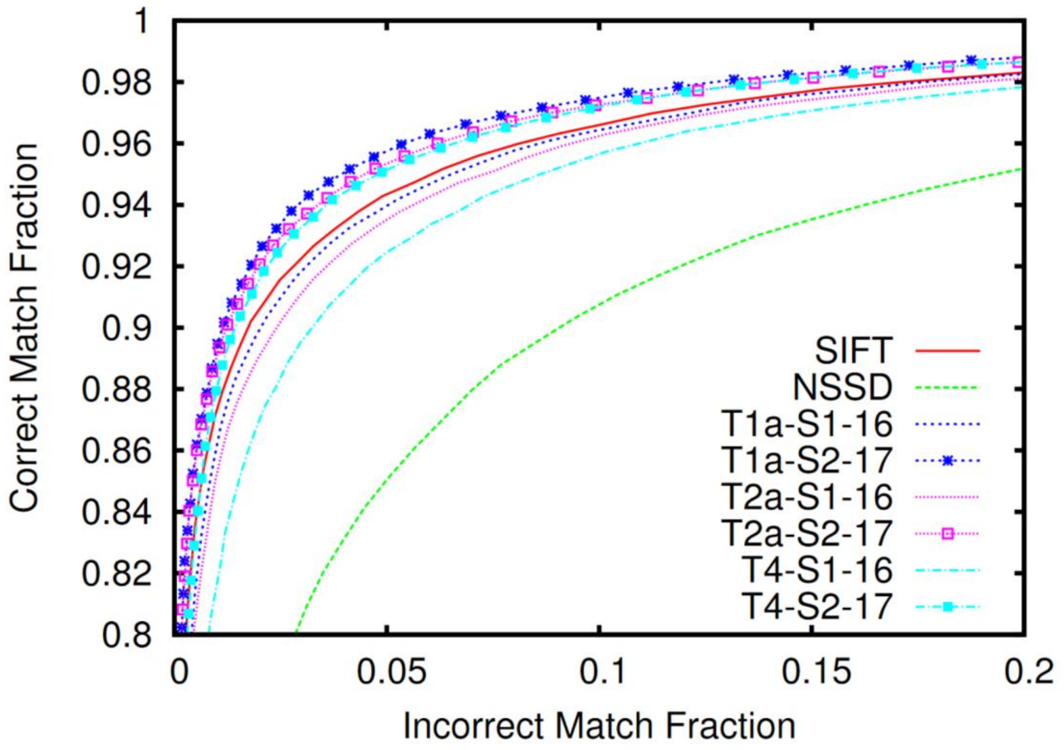


Figure 5. Selected ROC curves for the trained descriptors with four dimensional T-blocks ($k = 4$). Those that perform better than SIFT all make use of the S2 log-polar summation stage. See Table 4 for details.

We obtained a mixed training set consisting of tourist photographs of the Trevi Fountain and of Yosemite Valley (920 images), and a test set consisting of images of Notre Dame (500 images). We extracted interest points and matched them between all of the images within a set using the SIFT detector and descriptor [9]. We culled candidate matches using a symmetry criterion and used RANSAC [5] to estimate initial fundamental matrices between image pairs. This stage was followed by bundle adjustment to reconstruct 3D points and to obtain accurate camera matrices for each source image. A similar technique has been described by [17].

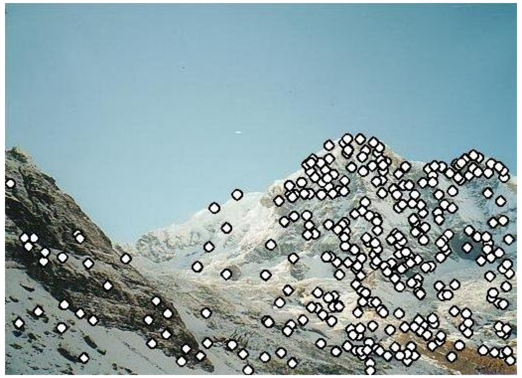
Local Descriptors

- Most features can be thought of as templates, histograms (counts), or combinations
- The ideal descriptor should be
  - Robust
  - Distinctive
  - Compact
  - Efficient
- Most available descriptors focus on edge/gradient information
  - Capture texture information
  - Color rarely used

K. Grauman, B. Leibe

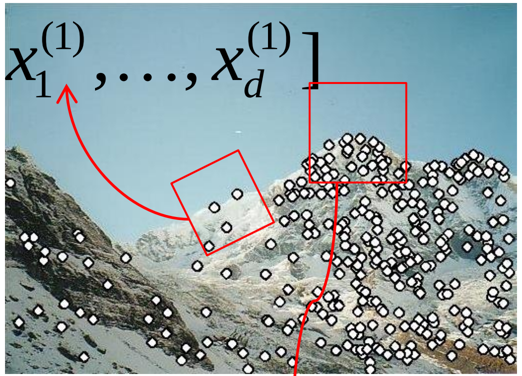
Local features: main components

1) **Detection:** Identify the interest points



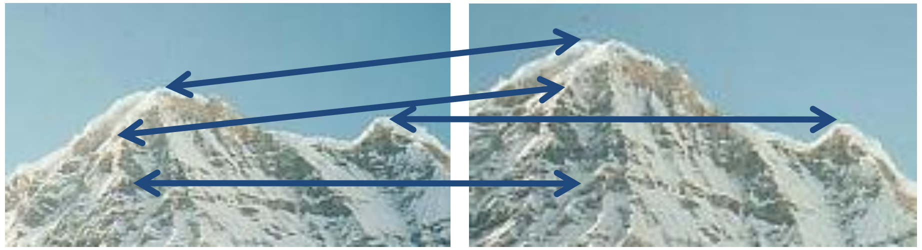
2) **Description:** Extract vector feature descriptor surrounding each interest point.

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

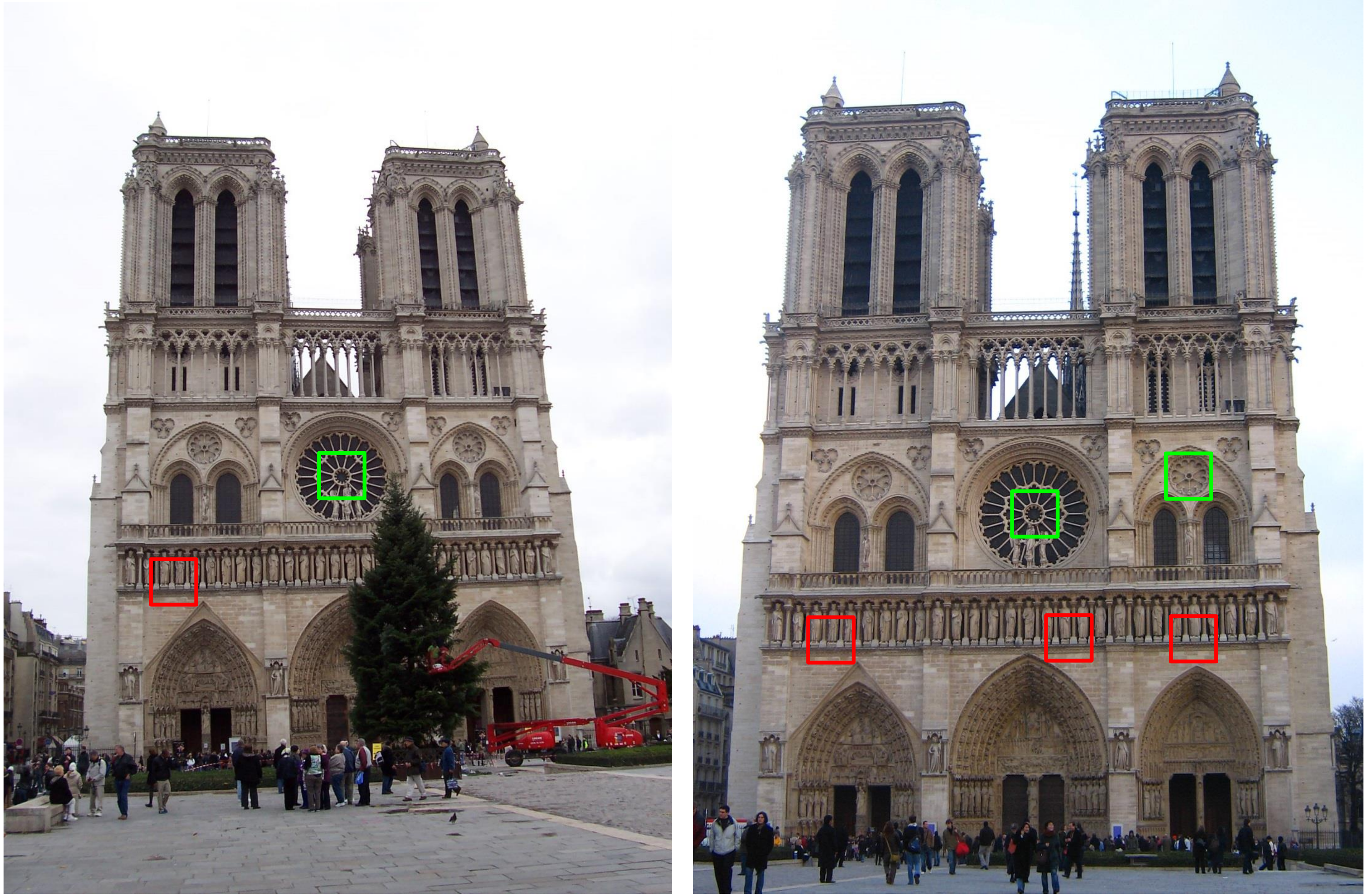
3) **Matching:** Determine correspondence between descriptors in two views



Kristen Grauman

Matching

- Simplest approach: Pick the nearest neighbor. Threshold on absolute distance
- Problem: Lots of self similarity in many photos



Distance: 0.34, 0.30, 0.40

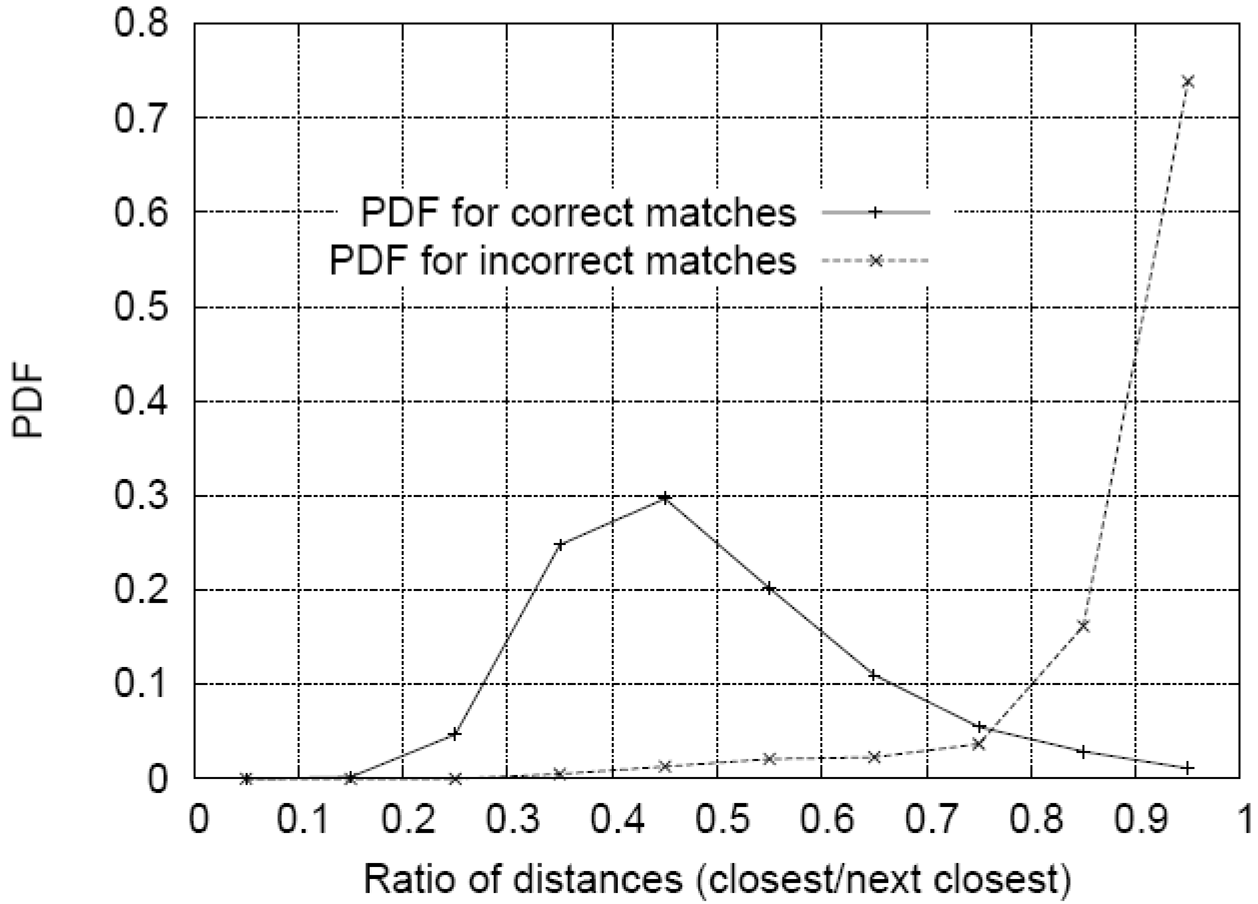
Distance: 0.61
Distance: 1.22

Nearest Neighbor Distance Ratio

- $\frac{NN1}{NN2}$ where NN1 is the distance to the first nearest neighbor and NN2 is the distance to the second nearest neighbor.
- Sorting by this ratio (into ascending order) puts matches in order of confidence (in descending order of confidence).

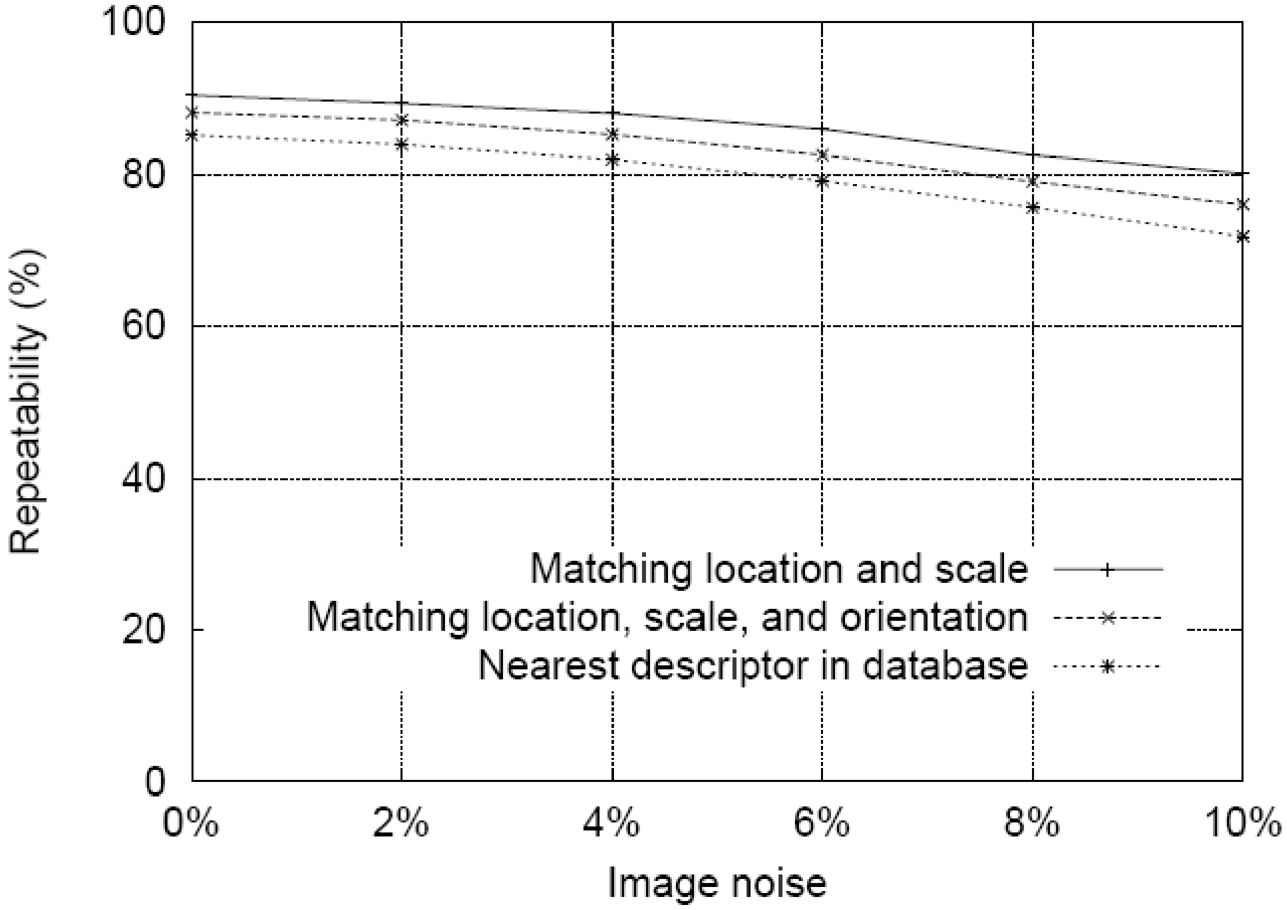
Matching Local Features

- Nearest neighbor (Euclidean distance)
- Threshold ratio of nearest to 2nd nearest descriptor

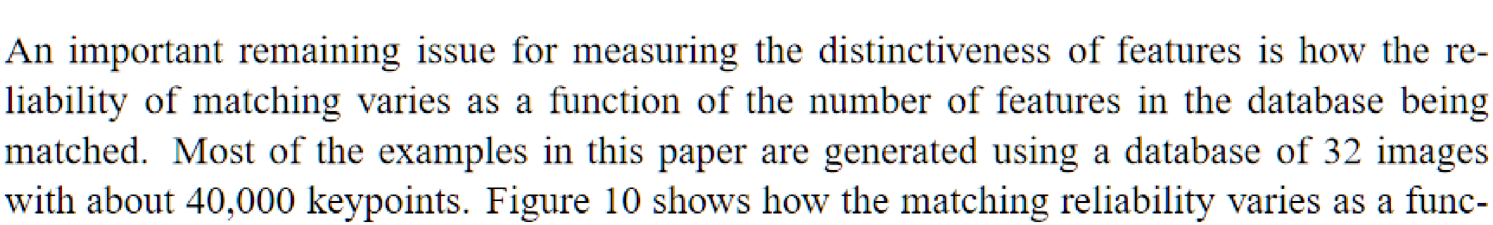


Lowe IJCV 2004

SIFT Repeatability



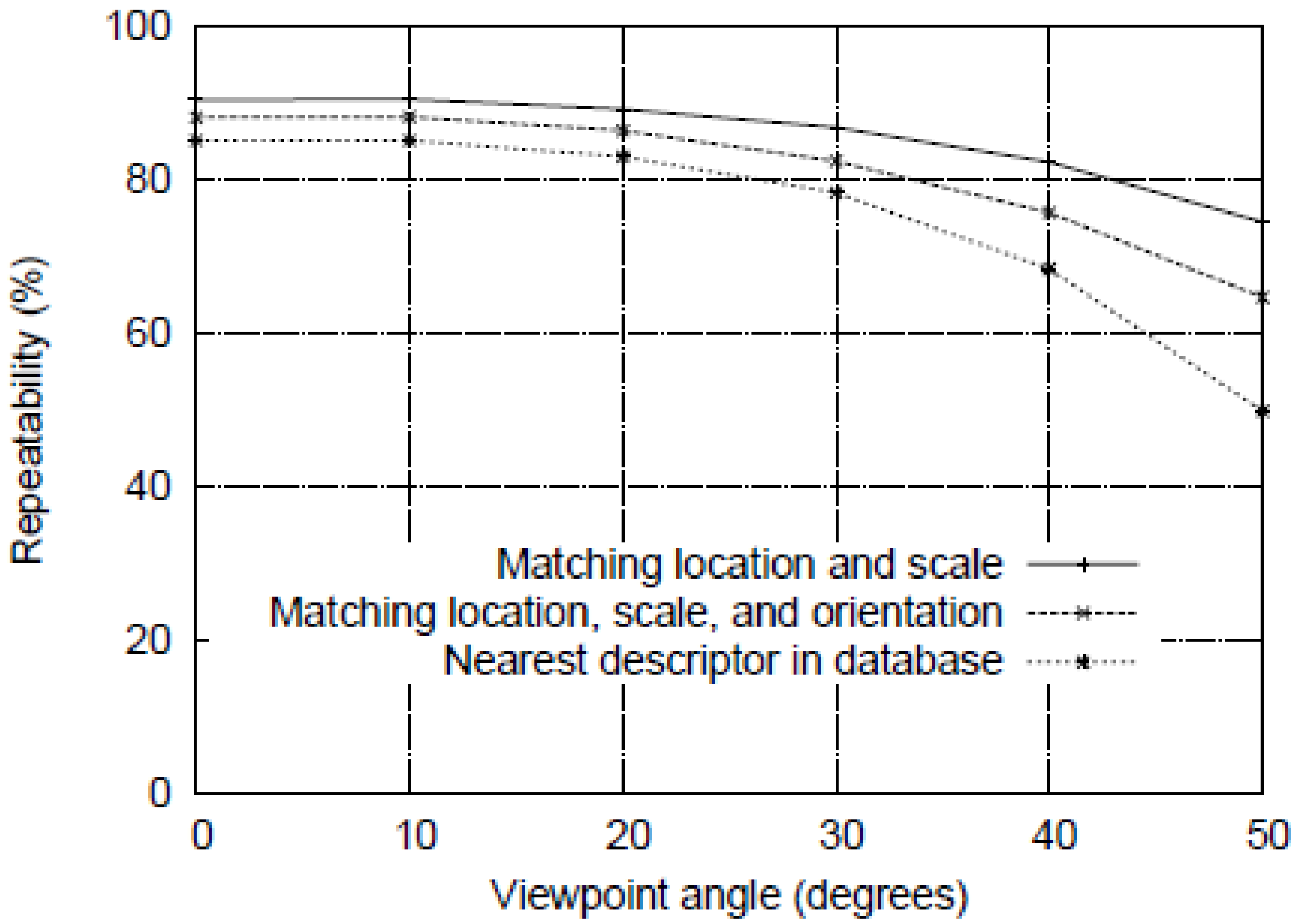
6.4 Matching to large databases



An important remaining issue for measuring the distinctiveness of features is how the reliability of matching varies as a function of the number of features in the database being matched. Most of the examples in this paper are generated using a database of 32 images with about 40,000 keypoints. Figure 10 shows how the matching reliability varies as a func-

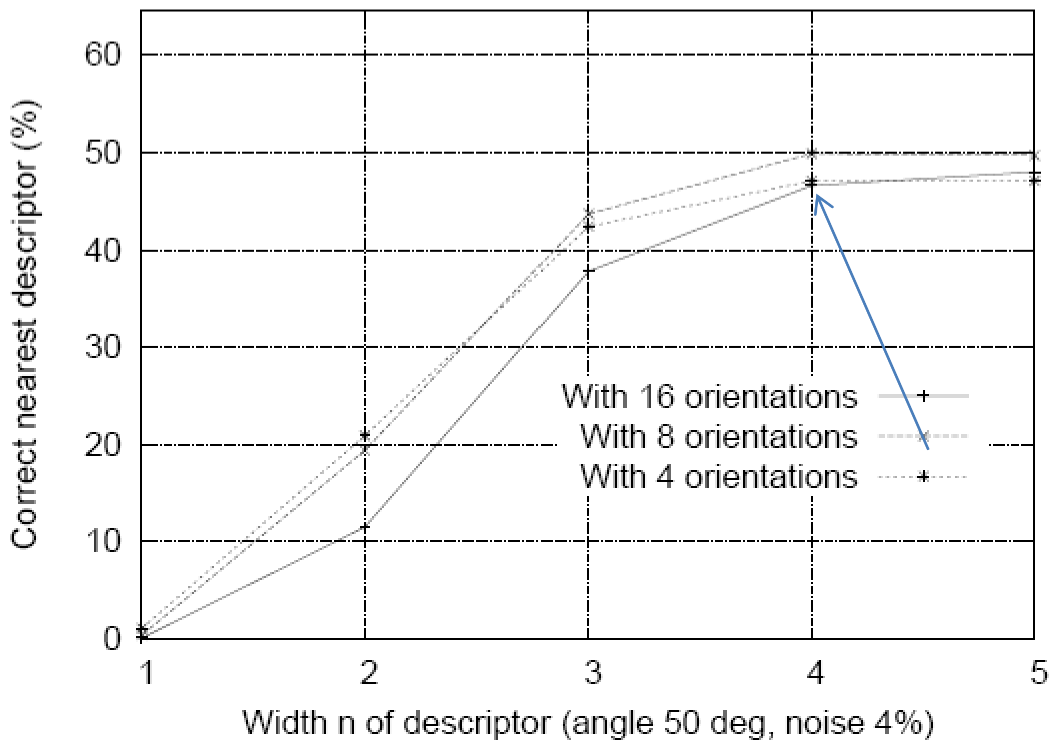
Lowe IJCV 2004

SIFT Repeatability



Lowe IJCV 2004

SIFT Repeatability



Lowe IJCV 2004

Choosing a detector

- What do you want it for?
  - Precise localization in x-y: Harris
  - Good localization in scale: Difference of Gaussian
  - Flexible region shape: MSER

- Best choice often application dependent
  - Harris-/Hessian-Laplace/DoG work well for many natural categories
  - MSER works well for buildings and printed things

- Why choose?
  - Get more points with more detectors

- There have been extensive evaluations/comparisons
  - [Mikolajczyk et al., IJCV’05, PAMI’05]
  - All detectors/descriptors shown here work well

Comparison of Keypoint Detectors

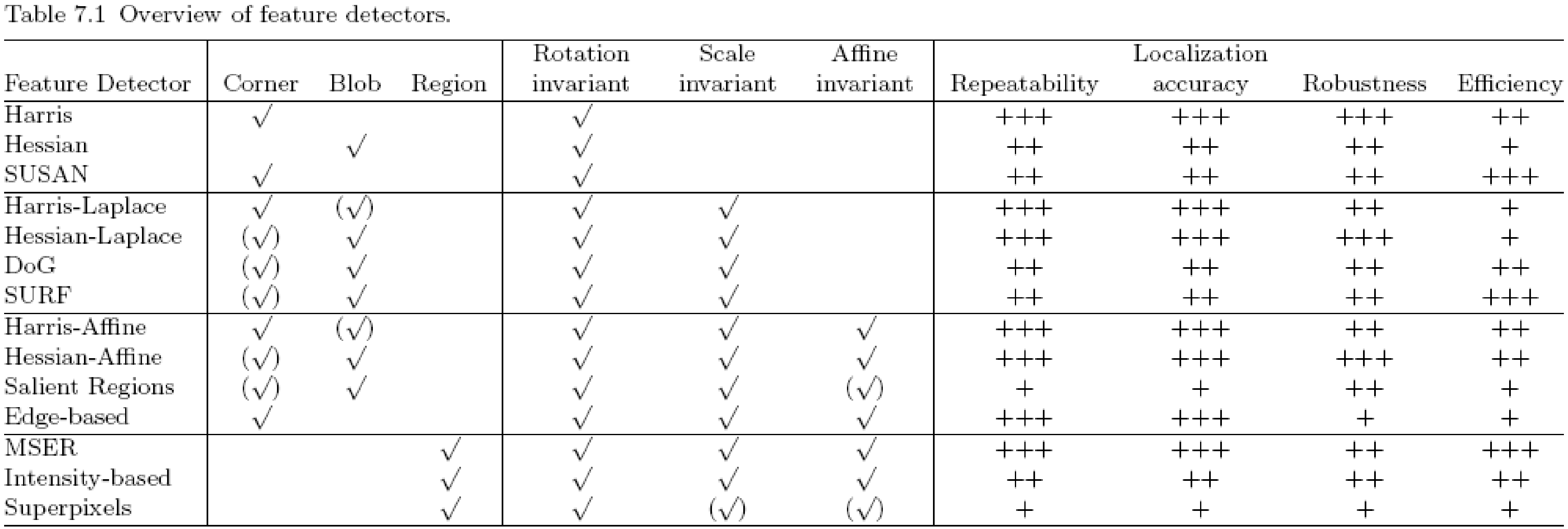


Table 7.1 Overview of feature detectors.

				Rotation	Scale	Affine		Localization		
Feature Detector	Corner	Blob	Region	invariant	invariant	invariant	Repeatability	accuracy	Robustness	Efficiency
Harris	√			√			+++	+++	+++	++
Hessian		√		√			++	++	++	+
SUSAN	√			√			++	++	++	+++
Harris-Laplace	√	(√)		√	√		+++	+++	++	+
Hessian-Laplace	(√)	√		√	√		+++	+++	+++	+
DoG	(√)	√		√	√		++	++	++	++
SURF	(√)	√		√	√		++	++	++	+++
Harris-Affine	√	(√)		√	√	√	+++	+++	++	++
Hessian-Affine	(√)	√		√	√	√	+++	+++	+++	++
Salient Regions	(√)	√		√	√	(√)	+	+	++	+
Edge-based	√			√	√	√	+++	+++	+	+
MSER			√	√	√	√	+++	+++	++	+++
Intensity-based			√	√	√	√	++	++	++	++
Superpixels			√	√	(√)	(√)	+	+	+	+

Tuytelaars Mikolajczyk 2008

Choosing a descriptor

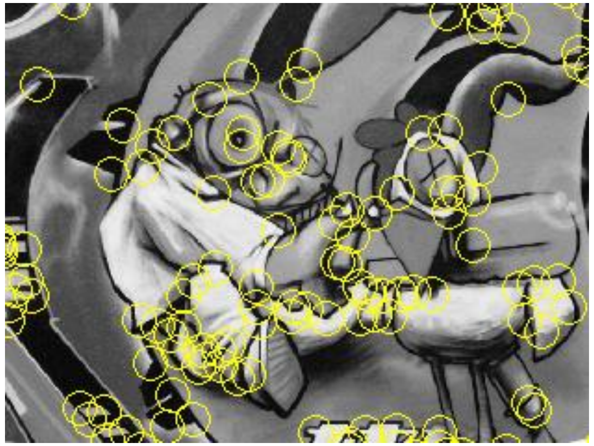
- Again, need not stick to one

- For object instance recognition or stitching, SIFT or variant is a good choice

- Learning-based methods are taking over this space, although not as quickly as one might expect.

Things to remember

- Keypoint detection: repeatable and distinctive
  - Corners, blobs, stable regions
  - Harris, DoG



- Descriptors: robust and selective
  - spatial histograms of orientation
  - SIFT

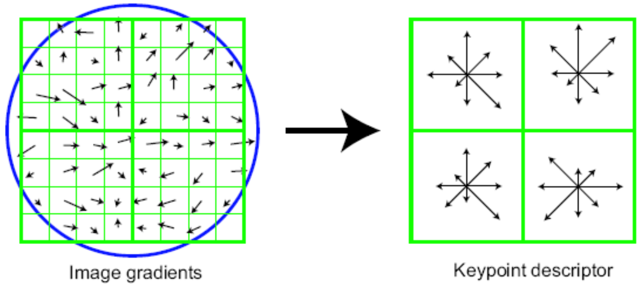


Image gradients
Keypoint descriptor