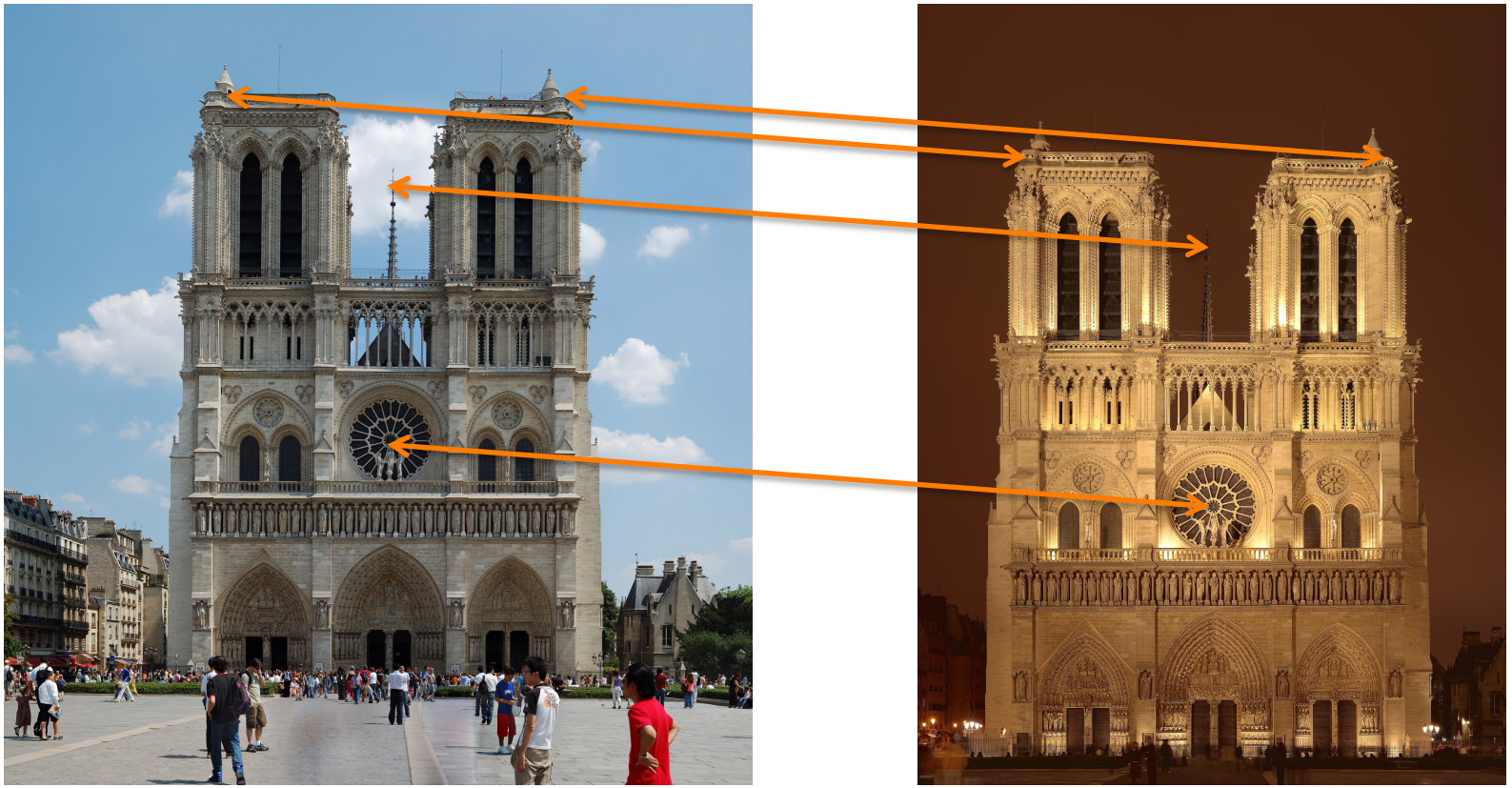
FEATURE REPRESENTATIONS

Why do we care about representation?



Discussion

- What information can we use to represent features?
- How do we make it robust to scale, rotation?

Proposed by David Lowe (2004)

SIFT: SCALE INVARIANT FEATURE TRANSFORM

Main parts of SIFT

1. Keypoint (i.e., Blobs) detection
   a. Scale-space extrema detection
   b. Keypoint localization
   c. Orientation assignment
2. Descriptor computation
   a. Keypoint descriptor

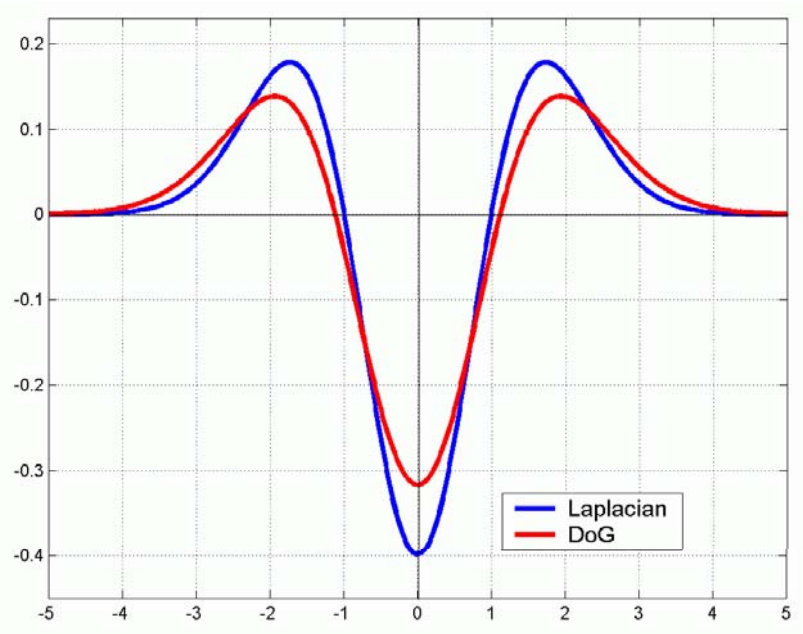
Approximating LoG with DoG

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

(Laplacian)

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

(Difference of Gaussians)



$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k - 1)\sigma^2 \nabla^2 G$$

Remember that blobs are detected using the Laplacian of Gaussian (LoG).

Image from http://www.cs.unc.edu/~lazebnik/spring11/lec08_blob.pdf

Efficient construction of the Difference of Gaussian

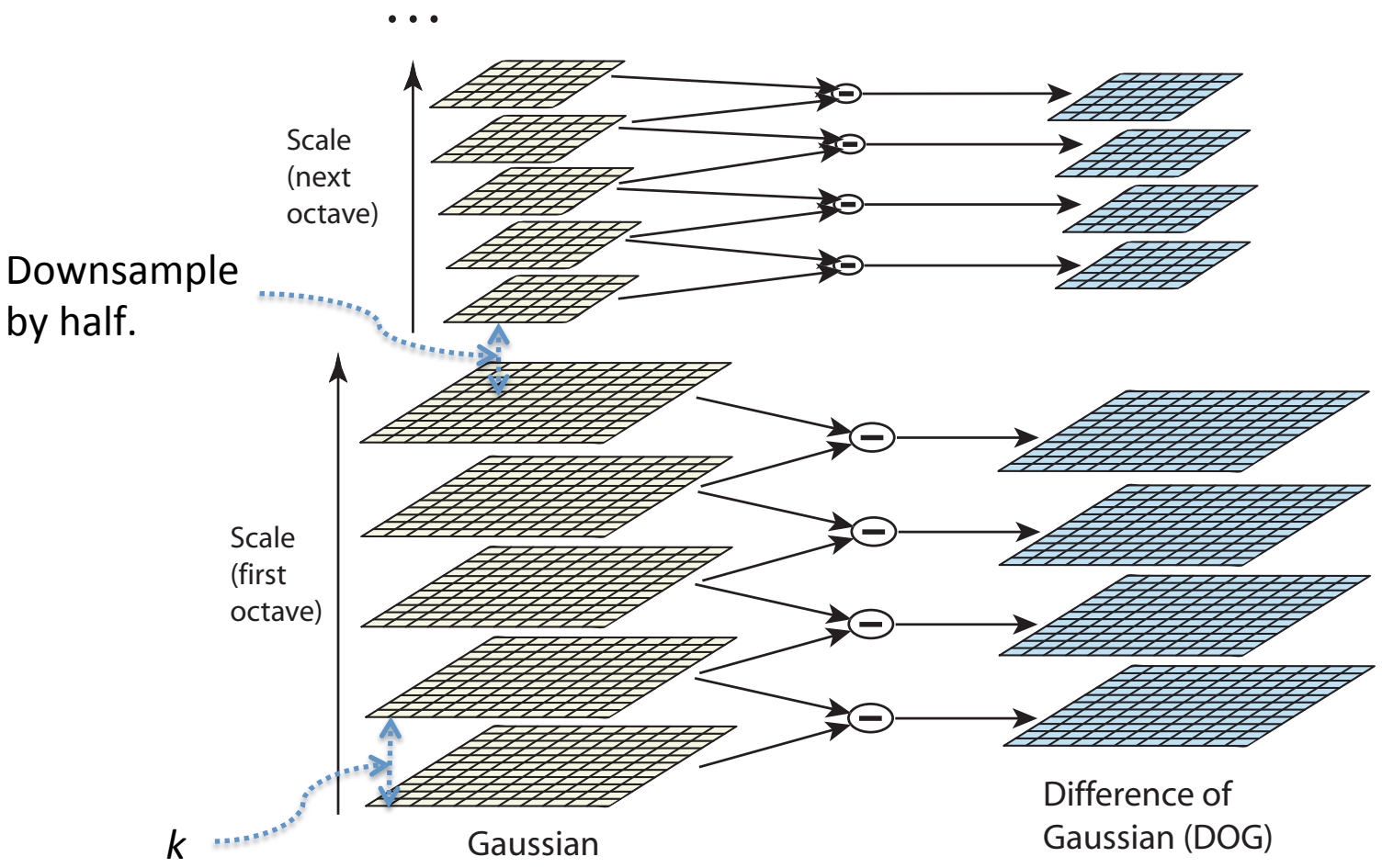
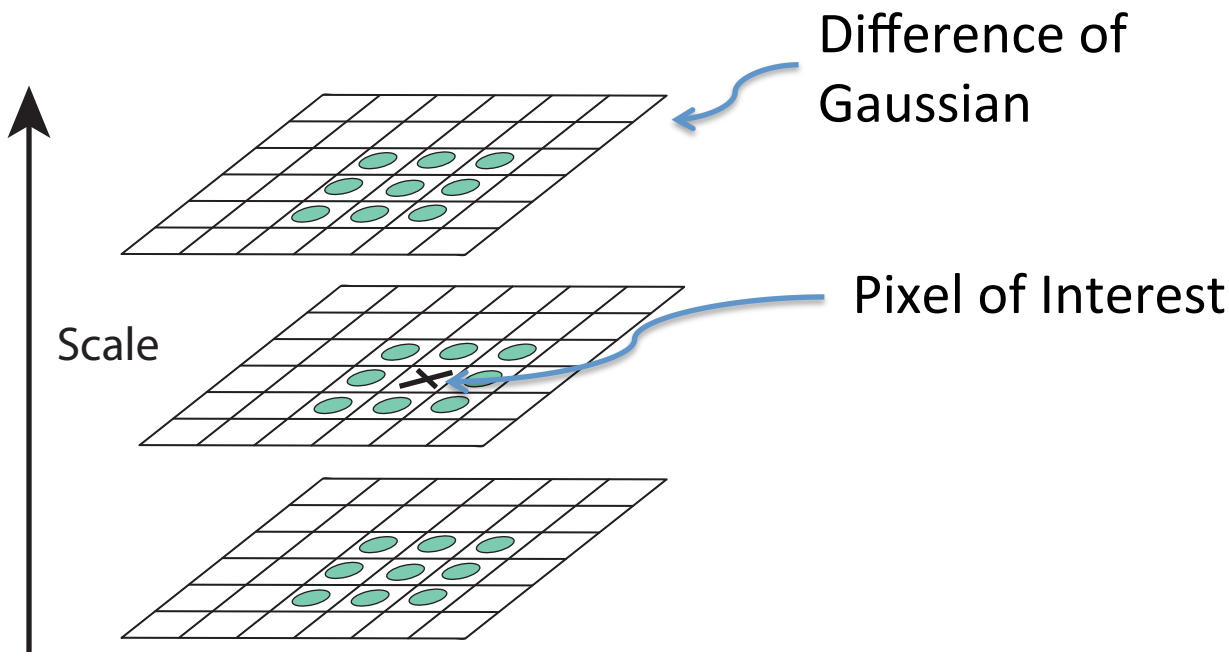


Image from D. Lowe. Distinctive Image features from scale-invariant keypoints. IJCV

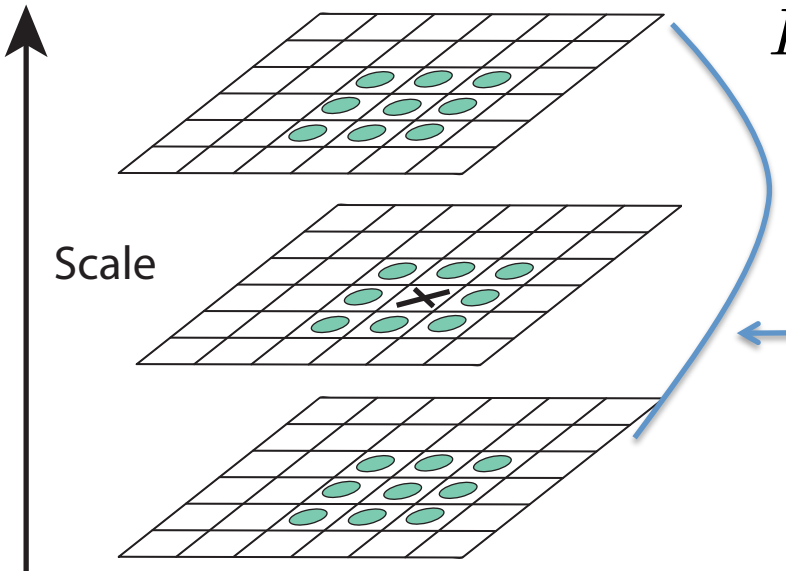
Local Extrema Detection



Maxima and minima of the difference-of-Gaussian images are detected by comparing a pixel (marked with X) to its 26 neighbors in 3x3 regions at the current and adjacent scales (marked with circles).

Image from D. Lowe. Distinctive Image features from scale-invariant keypoints. IJCV

Accurate keypoint localization



$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}$$

Quadratic interpolation of D(x, y, σ)

$$\hat{\mathbf{x}} = -\frac{\partial^2 D^{-1}}{\partial \mathbf{x}^2}\frac{\partial D}{\partial \mathbf{x}}$$

Keypoint refinement

Image from D. Lowe. Distinctive Image features from scale-invariant keypoints. IJCV

Filter Unstable Extrema

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D}{\partial \mathbf{x}}^T \hat{\mathbf{x}}.$$

Refined DoG response

Filter points with a low DoG response

Eliminating Edges

- Calculate the principal curvature from the Hessian matrix.
- Similar to Harris corner detector, we analyze the eigenvalues of *H*.

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

Hessian of the Difference of Gaussian

Eliminating Edges (cont'd)

Eigenvalues

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}$$

Largest Eigenvalue

$$\alpha = r\beta$$

Accept a keypoint if $\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}$

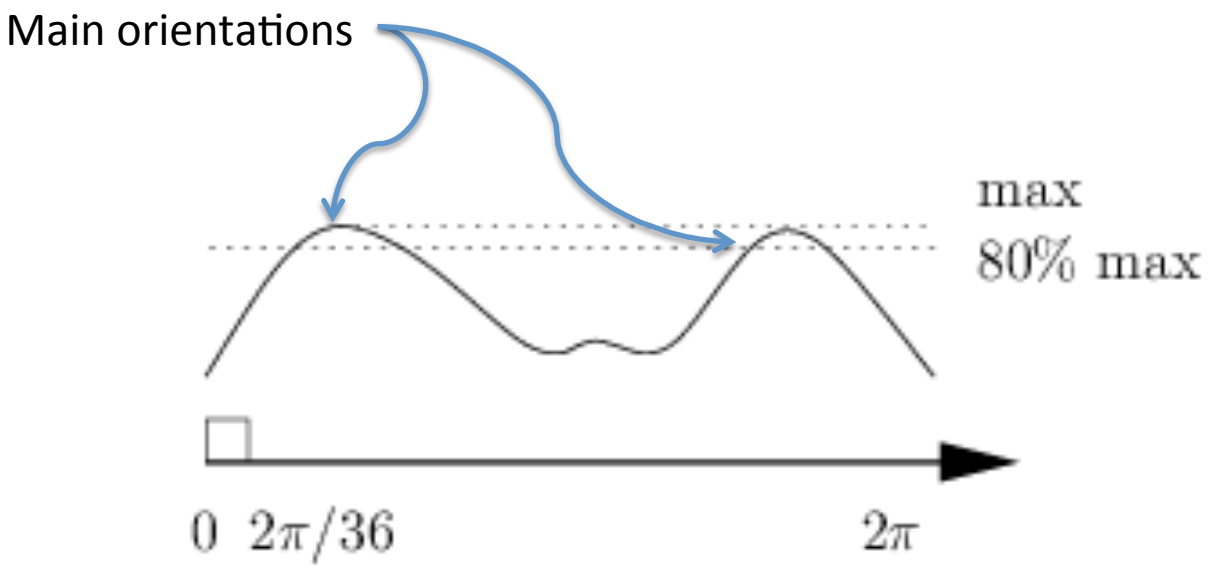
Discussion about invariance and covariance properties

- Laplacian of Gausian (i.e., blobs) response is invariant w.r.t scale and rotation.
- Blob location and scale are covariant w.r.t. Rotation and scaling.

Orientation Assignment

- For every keypoint (blob) do:
  1. Find the smoothed image (L) with the closest σ to the scale of the keypoint.
  2. Compute gradient magnitude and orientation of the selected image.
  3. Form an orientation histogram with 36 bins considering a region around the keypoint.
     I. Each sample added is weighted by its gradient magnitude and by a Gaussian window with $\sigma=1.5\ \sigma_{scale.}$
  4. Highest peak is detected.
     I. Add any other peaks if their height exceeds 80% of the highest peak.
     II. NOTE: A blob/keypoint might have multiple orientations assigned.
  5. Fit a parabola to each peak and interpolate to estimate the maximum orientation.

Orientation Assignment (cont'd)



Blobs can have multiple assigned orientations!

Image from http://www.vlfeat.org/api/sift.html
Credit: Andrea Vedaldi

Orientation Assignment (cont'd)

Keypoint
(Blob)

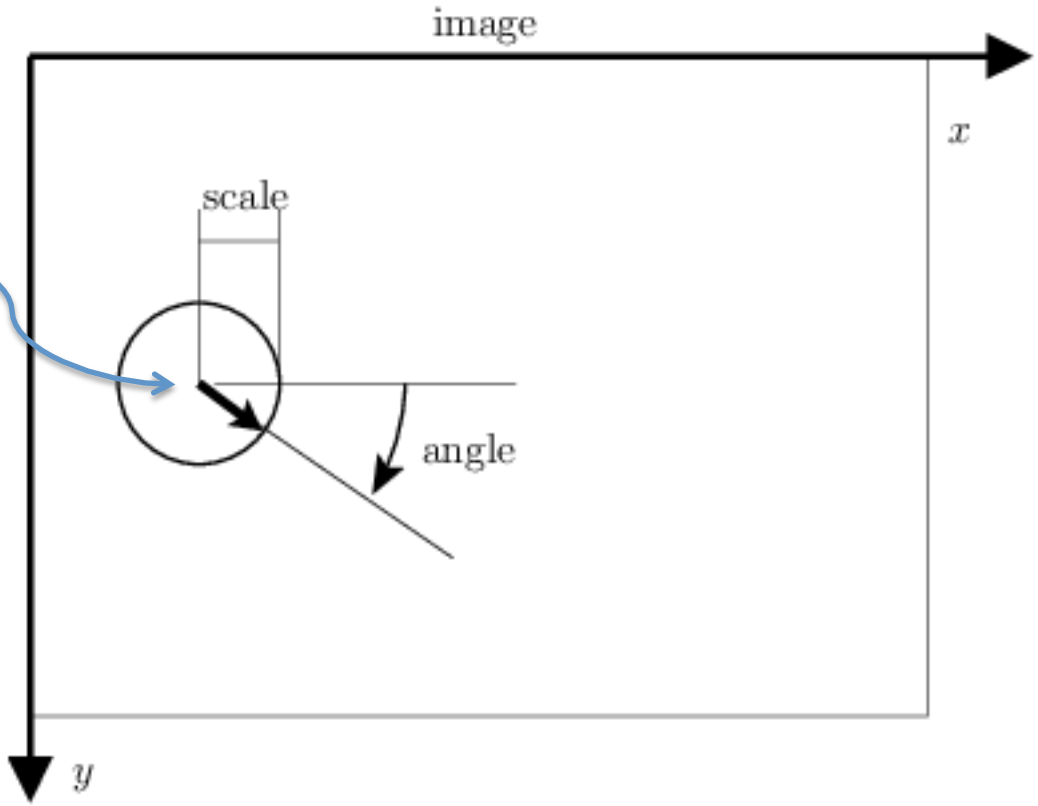
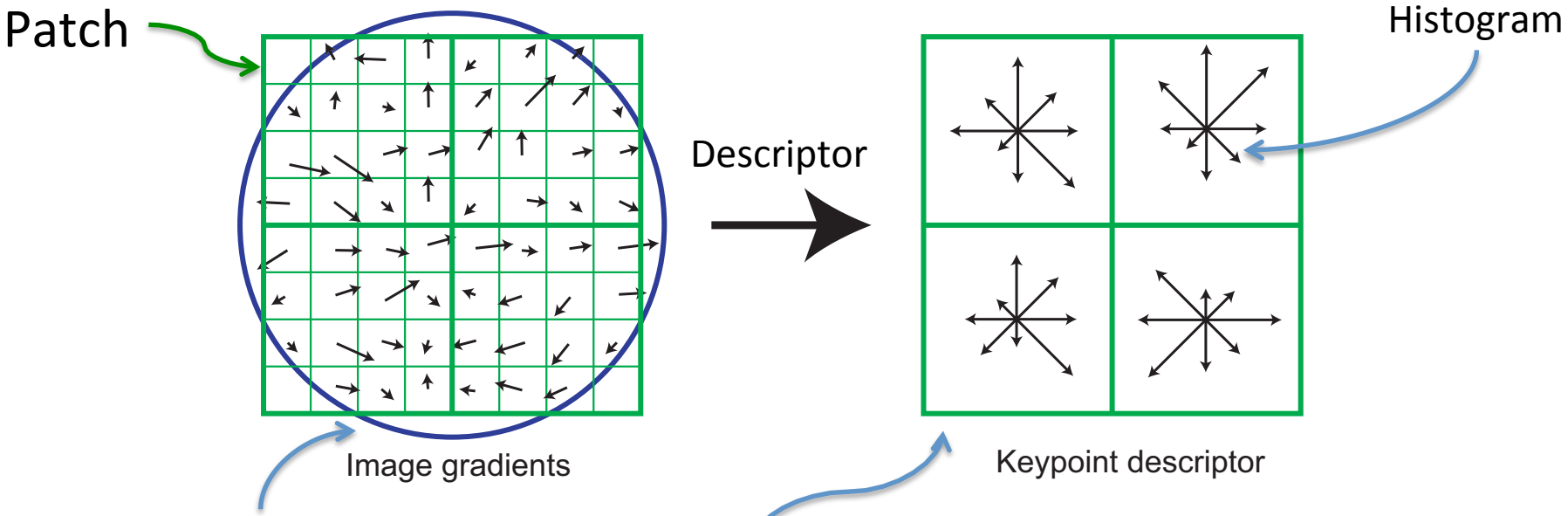


Image from http://www.vlfeat.org/api/sift.html
Credit: Andrea Vedaldi

Descriptor computation

Rotating first so that the orientation of the patch looks upright.



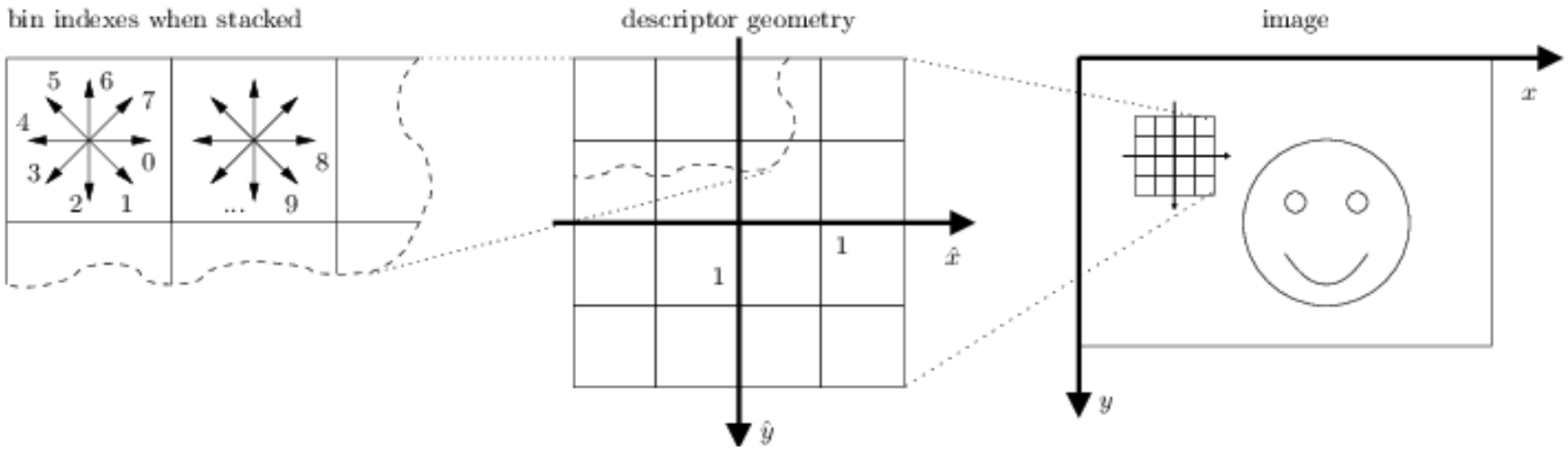
Gaussian window
(circle around patch)

4x4 subregion (in this figure)

A descriptor is computed per orientation.

Image from D. Lowe. Distinctive Image features from scale-invariant keypoints. IJCV

SIFT descriptor



- 4x4 bins/cells
- 8 orientation per bin
- The dimension of the descriptor thus is 128

Image from http://www.vlfeat.org/api/sift.html
Credit: Andrea Vedaldi

SIFT descriptor

- A weighting scheme (Gaussian window) is used to form the histograms.
- To reduce illumination effects in the descriptor, the feature vector is normalized.
- Threshold the normalized vector (threshold =0.2).
- Keep samples higher than the threshold and renormalize.

SIFT descriptors visualization

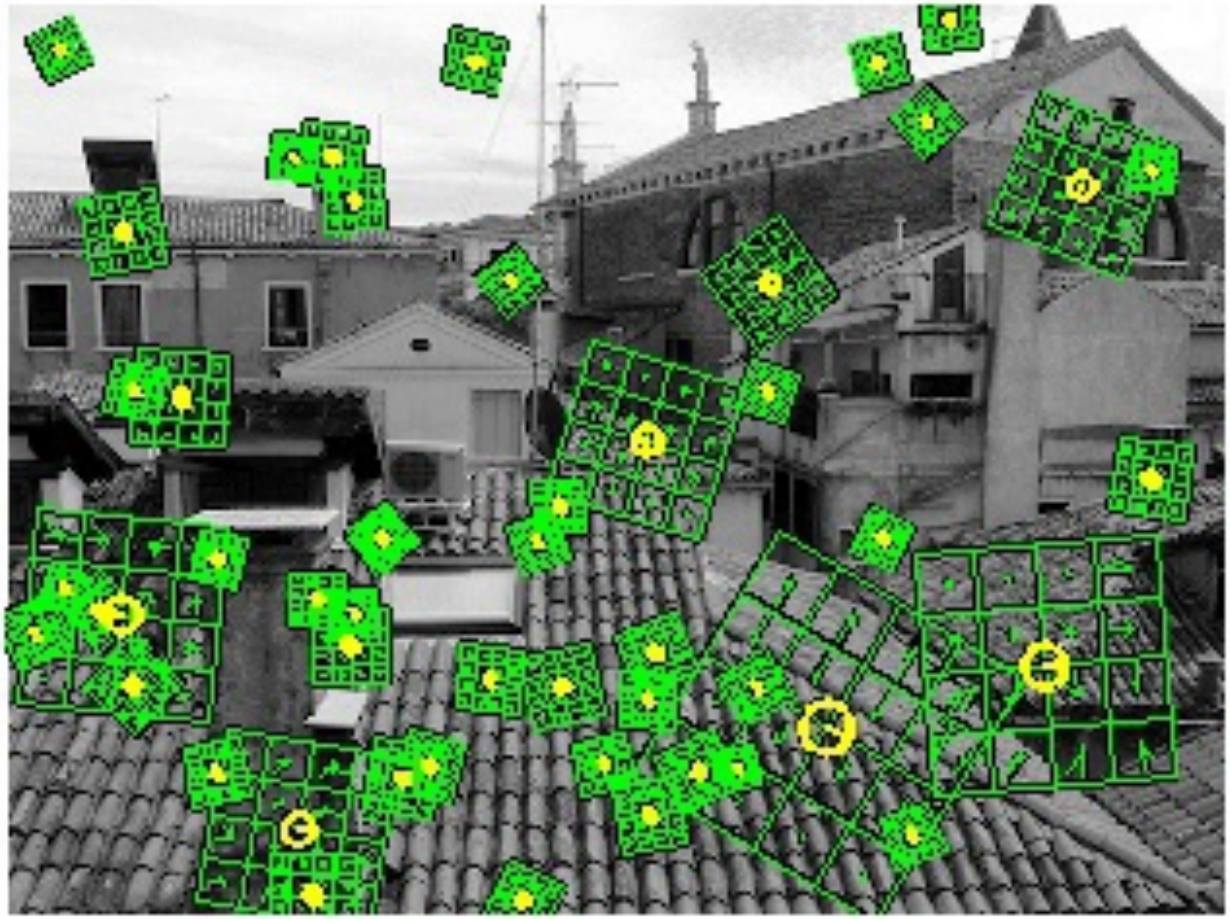


Image from http://www.vlfeat.org/api/sift.html
Credit: Andrea Vedaldi

SIFT implementations

- VLFeat: http://www.vlfeat.org/overview/sift.html
  - Matlab and C/C++
- OpenCV: http://docs.opencv.org/modules/nonfree/doc/feature_detection.html?highlight=sift
  - C/C++ and Python
- David Lowe: http://www.cs.ubc.ca/~lowe/keypoints/
  - Binary (SIFT is patented)

Discussion

- How do we match features?
  - Brute force Nearest-Neighbor search: Given a query feature find the reference feature that produces the smallest distance.
  - Approximate Nearest-Neighbor (ANN).

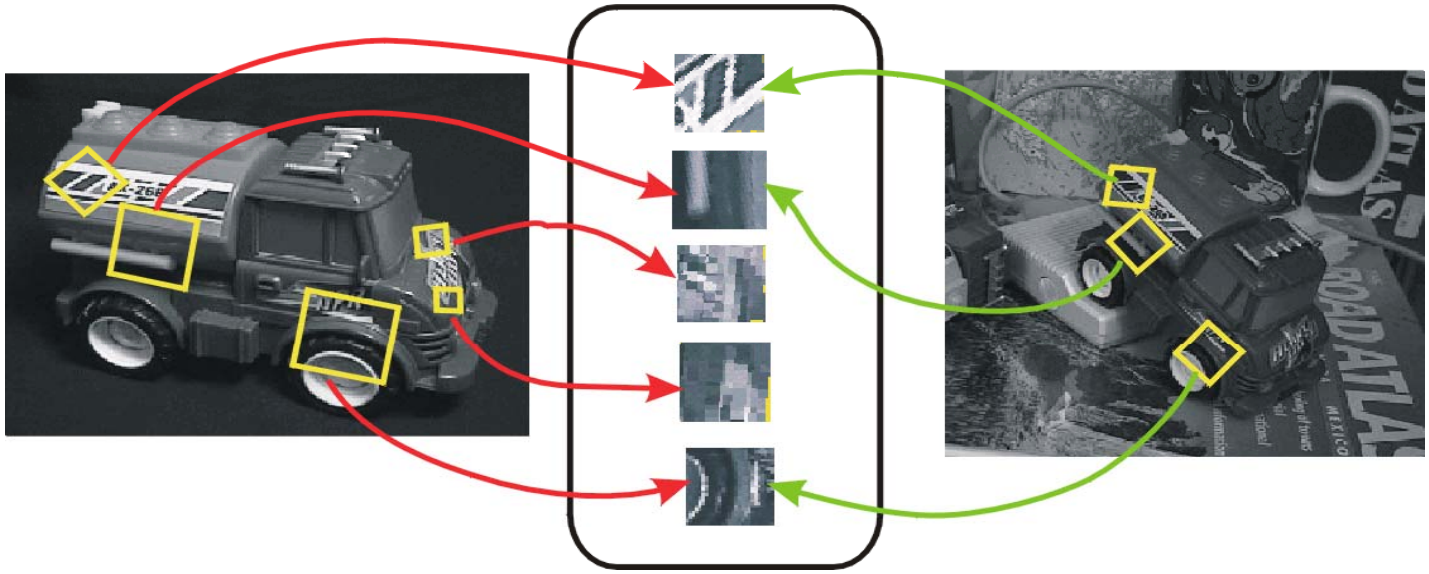
Invariance vs. Covariance

Invariance:

- features(transform(image)) = features(image)

Covariance:

- features(transform(image)) = transform(features(image))



Covariant detection => invariant description

Slide from http://www.cs.unc.edu/~lazebnik/spring11/lec08_blob.pdf
S. Lazebnik

Proposed by Bay et al. (2006)

SURF: SPEEDED UP ROBUST FEATURES

Blob detection via Hessian

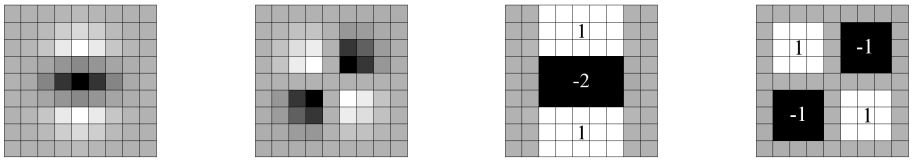
- Hessian matrix using LoG:

$$H = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix}$$

Point

Scale

- Problem: LoG is expensive to compute, why?
- Solution: Approximate with box filters!



Blob detection via Hessian (cont'd)

$$\det H = D_{xx}D_{yy} - (wD_{xy})^2$$

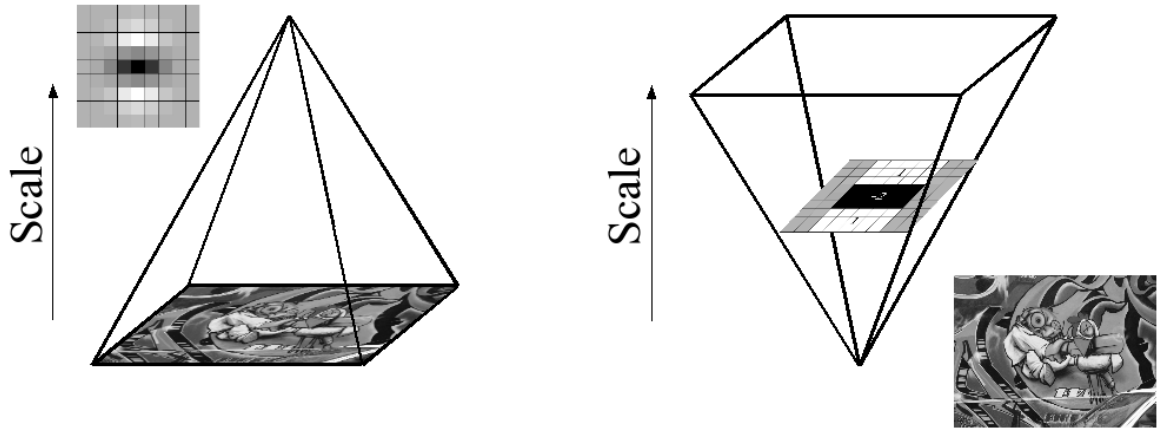
LoG approximations

Weight

w = 0.9

The determinant is used as the blob response.

Scale-space blob detection



- Instead of reducing the image size (left), Surf increase the filter size exploiting box filters.

Image from Bay et al. Speed-Up Robust Features. CVIU 2008

Blob detection via approximation of the determinant of the Hessian

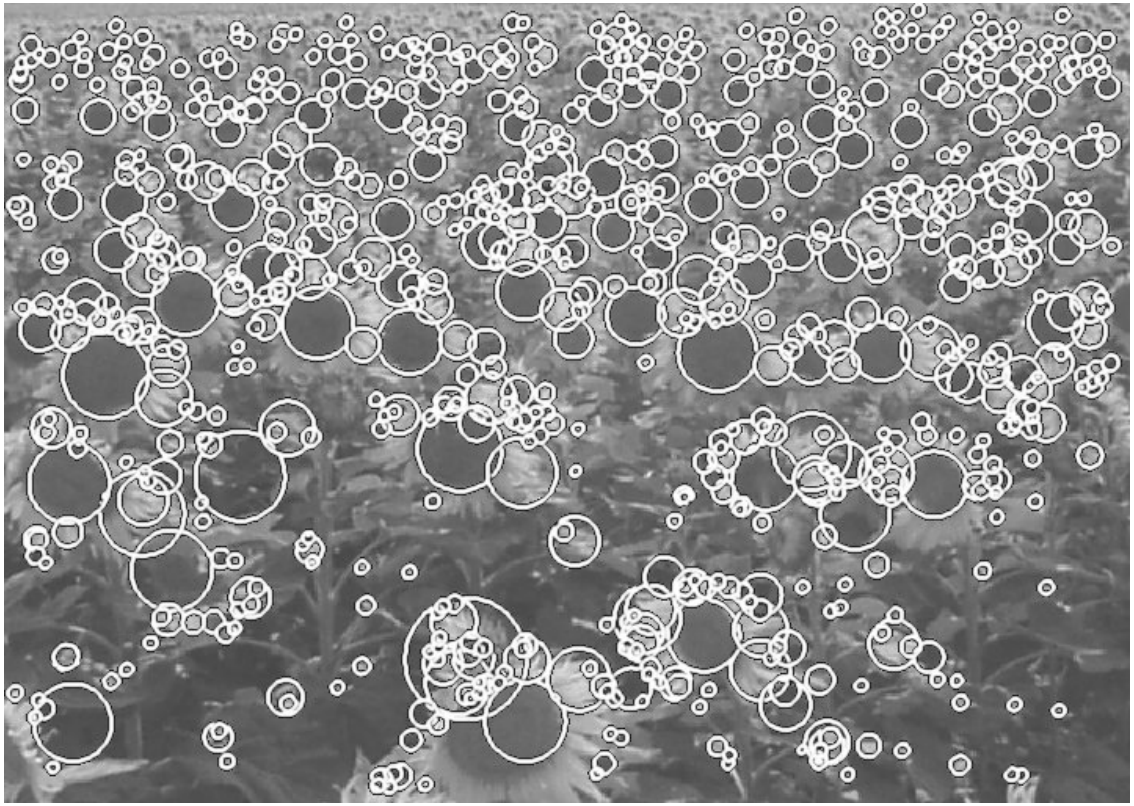
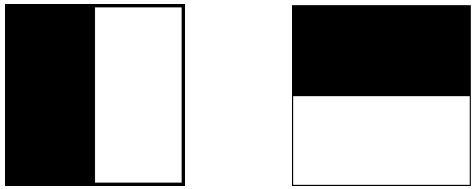


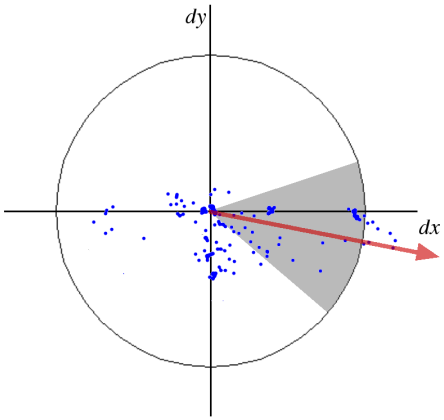
Image from Bay et al. Speed-Up Robust Features. CVIU 2008

Orientation Assignment

- Convolve image with Haar-like filters; estimate some sort of gradient.



- Represent the "gradients" as points, and calculate the orientation from a window.
  - Sum points over the angular window
  - The longest vector is kept as the main orientation



Images from Bay et al. Speed-Up Robust Features. CVIU 2008

Descriptor computation

- Grab an oriented patch around the blob.

- Compute the following sums at each sub-region, and concatenate them.

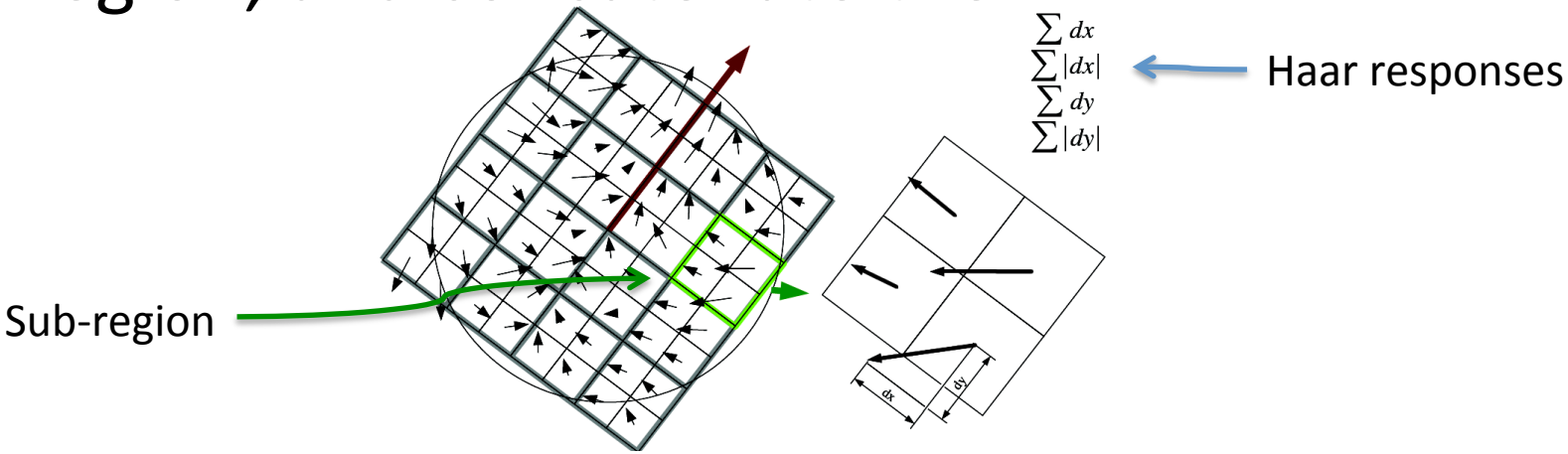


Image from Bay et al. Speed-Up Robust Features. CVIU 2008

BINARY DESCRIPTORS

BRIEF: Binary Robust Independent Elementary Features

- Proposed by Calonder et al. in ECCV 2010.
- SIFT and SURF work great, but the descriptor can be expensive to compute.
- Comparing SIFT or SURF descriptors involve Euclidean norms, which also add computational cost.

BRIEF Descriptor

- Idea: Compute cheaply a descriptor that can be compared efficiently.
- Binary Feature:

$$T(p; \mathbf{x}, \mathbf{y}) = \begin{cases} 1 & \text{if } \; p(\mathbf{x}) < p(\mathbf{y}) \\ 0 & \text{otherwise} \end{cases}$$

BRIEF Descriptor (cont'd)

- Concatenate the binary features to form the descriptor.
- Patterns used to form several descriptors:

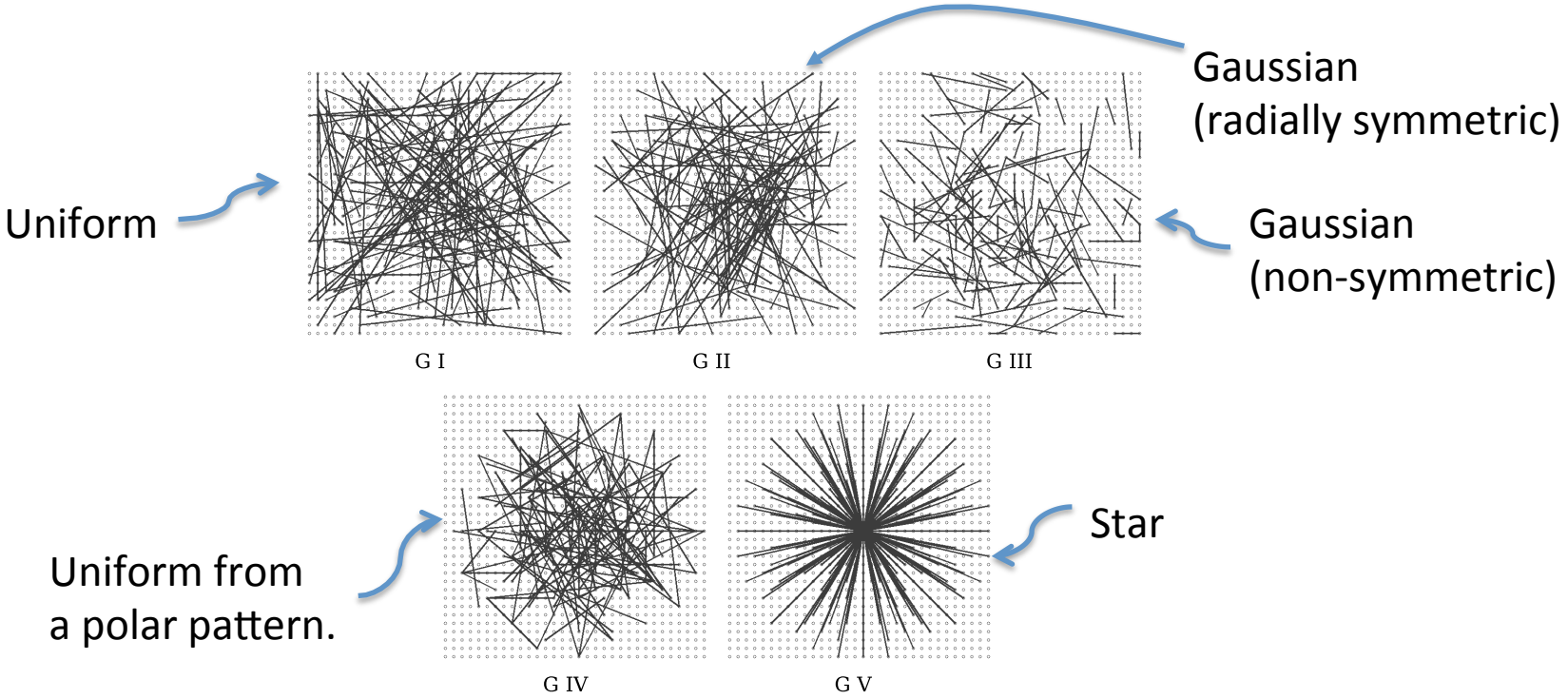
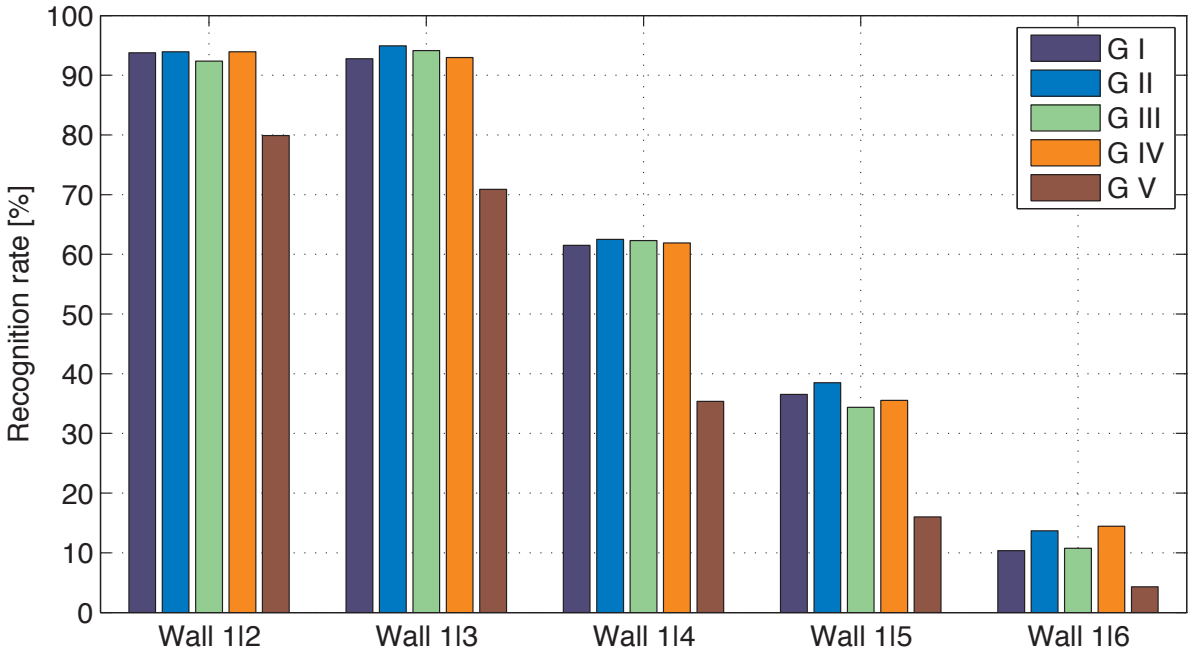


Image taken from Calonder et al. BRIEF: Binary Robust Independent Features. ECCV 2010.

What pattern works the best?



The random patterns work the best, according to experiments.

Image taken from Calonder et al. BRIEF: Binary Robust Independent Features. ECCV 2010.

Number of bits for BRIEF

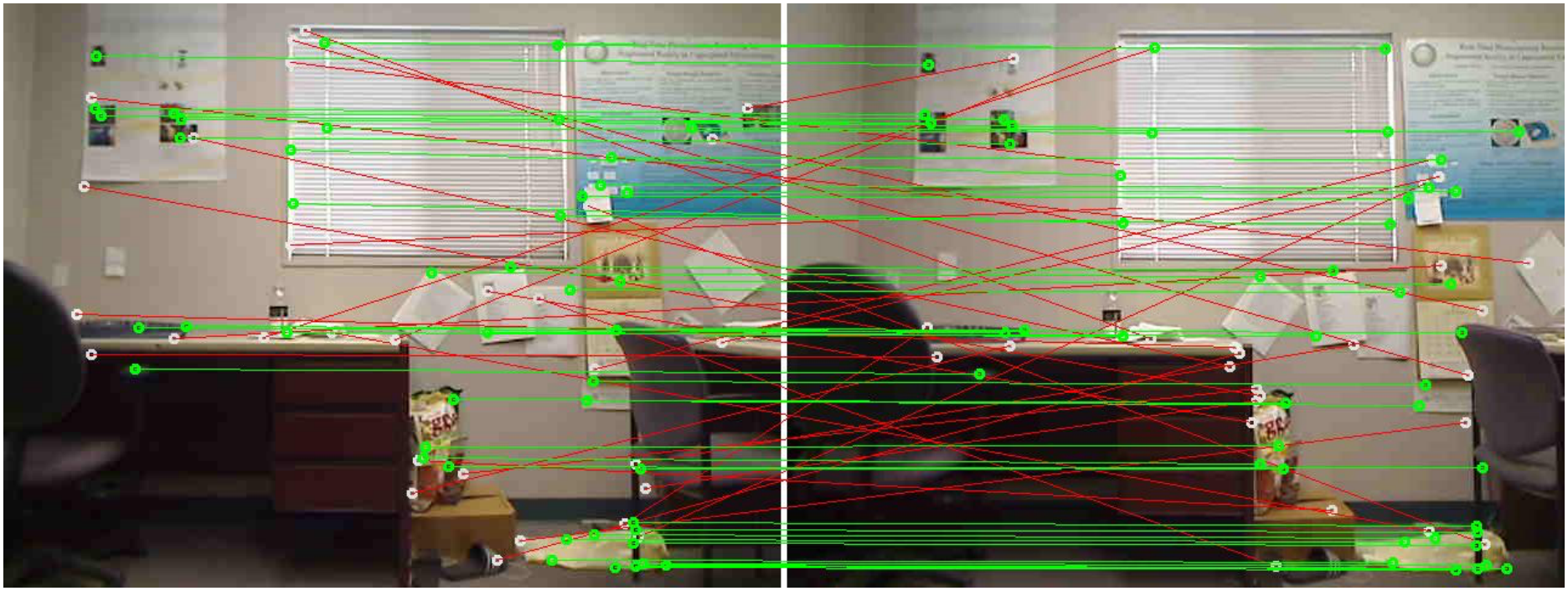
- Each bit is a binary feature.
- The sizes range from 16-512 bits.
- BRIEF is a descriptor, it needs keypoints to describe
  - FAST corners are used because they can be detected quickly.
  - CenSurE (a fast BLOB detector) is another good choice.

FEATURE MATCHING

Matching descriptors

- SIFT & SURF usually are matched with a nearest-neighbor (NN) classifier
  - Approximate NN methods also exist
- Binary descriptors are matched using a Hamming distance and a NN classifier.

Feature Matching Example



How do we fix those bad matches?

- We need geometric models to check consistency of the correspondences!