Keypoints and Descriptors

Srikumar Ramalingam

School of Computing
University of Utah

Presentation Outline

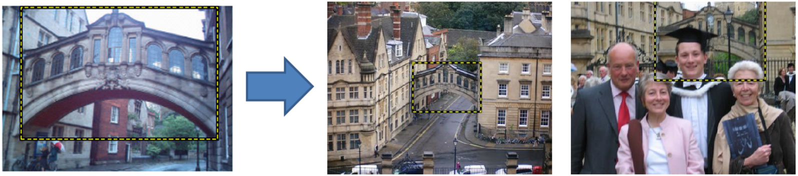
Main paper to be discussed

- David G. Lowe, Distinctive Image Features from Scale-Invariant Keypoints, IJCV, 2004.

How to find useful keypoints?

[0]Source: Kristen Grauman

How do you handle large scale changes?



[0]Source: Kristen Grauman

Problem Statement

- Extract distinctive invariant features from images.
- Perform reliable matching between different views of an object or scene.
- Provide invariance w.r.t scale changes, rotations, affine distortions, viewpoints, noise, illumination.

Uses of SIFT

- Fundamental matrix estimation
- Pose estimation
- 3D reconstruction
- Object recognition
- Image retrieval
- Visual odometry and robot navigation
- ..

How to perform object recognition?

- The invariant features can be matched against a large database of features of known objects from many images.
- The matching can be done using a fast nearest-neighbor algorithm.
- Pose estimation can be used for verification purposes.
- Can handle clutter and occlusion.

Searching for specific objects from movies

Example

retrieved shots

[0] Source: Kristen Grauman, Sivic, Zisserman

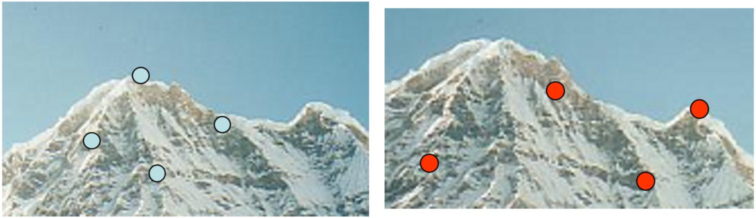
Searching for specific objects from movies

Query region

Retrieved frames

[0]Source: Kristen Grauman, Joseph Sivic and Andrew Zisserman

Repeatability



- What if we don't find the same features to match?

[0] Source: Kristen Grauman

Stages of generating SIFT features

- **Scale-space extrema detection:**
  - Searches over all scales and image locations.
  - Efficient implementation using difference-of-Gaussian function to detect points that are invariant to scale and orientation.
- **Keypoint localization:** A model is fit to determine location and scale. Keypoints are selected based on measures of their stability.
- **Orientation assignment:** Compute best orientation(s) for each keypoint region.
- **Keypoint descriptor:** Use local image gradients at selected scale and rotation to describe each keypoint region.

Presentation Outline

Detection of scale-space extrema

- Search for stable features across all possible scales, using a continuous function of scale known as scale space (Witkin, 1983).
- An image is represented as a one-parameter family of smoothed images, the scale-space representation, parameterized by the size of the smoothing kernel used for suppressing fine-scale structures.
- Does it just correspond to the size of the image?

Why do we use the Gaussian function as the scale-space kernel?

- We sometimes want to look at a leaf and other times an entire tree. If it's a tree, we get rid of small details such as leaves and twigs. While getting rid of these small details, we must ensure that we do not introduce new false details.
- Koenderink (1984) and Lindeberg (1994) showed that the only way to ensure this under reasonable assumptions is by using Gaussian function as the only scale-space kernel.

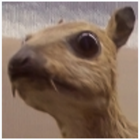
[0] Source: aishack.in

What is an image kernel?

- A kernel, convolution matrix, or mask is a small matrix that is useful for blurring, sharpening, embossing, edge detection, and more.
- This is accomplished by means of convolution between a kernel and an image.

[0] Source: wikipedia.org

Identity kernel

Operation	Kernel	Image result
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	

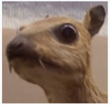
[0]Source: wikipedia.org

Edge kernel

Operation	Kernel	Image result
 Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

[0] Source: wikipedia.org

Box blur and Gaussian kernels

Operation	Kernel	Image result
Box blur (normalized) 	$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$	

[0] Source: wikipedia.org

Scale Space using Gaussian kernel- Witkin 1983

- An edge position may shift with increasing scale
- Two edges may merge with increasing scale
- An edge may not split into two with increasing scale

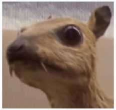
[0] Source: Noah Snavely

Scale Space

- The scale space of an image is a function $L(x, y, \sigma)$.
- We obtain $L(x, y, \sigma)$ by convolution of a variable-scale Gaussian $G(x, y, \sigma)$ with the input image $I(x, y)$: $L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$, where $*$ is the convolution operator in x and y.
- The Gaussian $G(x, y, \sigma)$ is given below:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

Image kernel - Gaussian function

Operation	Kernel	Image result
 **Gaussian blur 5 × 5** (approximation)	$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

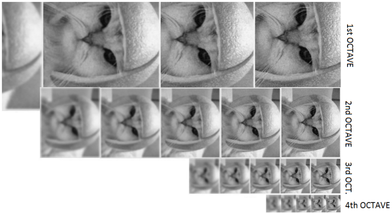
[0] Source: wikipedia.org

Image kernel - Gaussian function

- To create a scale space, progressively blur out images using Gaussian kernel

[0]Source: aishack.in

SIFT - Scale Space and downsampling



- Scale space images are obtained for different octaves. This is done by progressively blurring out images using Gaussian kernels, and then repeating the same after downsampling the original image by half to create the next octave.

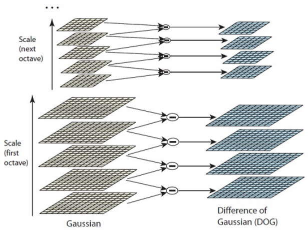
[0]Source: aishack.in

Difference of Gaussians

- Stable keypoint locations are detected using scale-space extrema in the difference-of-Gaussian function convolved with the image $D(x, y, \sigma)$.
- The diference of Gaussian is computed at two nearby scales separated by a constant multiplicative factor k.
- $$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned}$$

[0]Source: aishack.in

Difference of Gaussians



- For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left.
- Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right.
- After each octave, the Gaussian image is down-sampled by a factor of 2, and the process repeated.

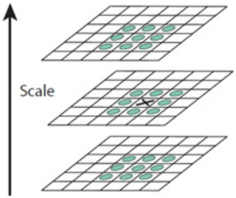
Laplacian of a Gaussian

- We take an image and blur it a little using Gaussian function.
- Calculate the second order derivatives or the Laplacian - locates edges and corners that are good for detecting keypoints.
- Computation of the second order derivative is also extremely sensitive to noise, and the blurring helps.

Laplacian of a Gaussian

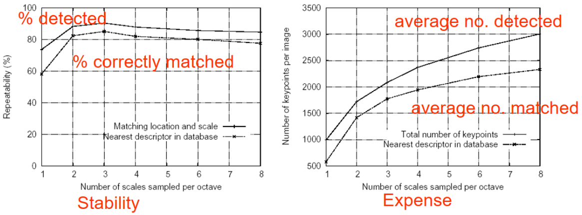
- The laplacian of a Gaussian is represented as $\nabla^2 G$.
- The scale-invariant laplacian of a Gaussian would be $\sigma^2 \nabla^2 G$.
- The difference of Gaussian images is approximately equivalent to the laplacian of the Gaussian.

Locate maxima-minima in DoG images



- Maxima and minima of the DoG images are detected by comparing a pixel (marked with X) to its 26 neighbors.
- The marked points are the approximate maxima and minima because the maxima/minima almost never lies exactly on a pixel.

Scale-space extrema detection

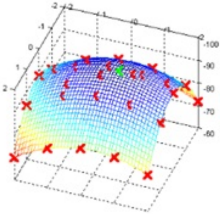


32 images were synthetically transformed and noise added.

- Sampling in scale for efficiency
- How many scales should be used per octave? S=?
- More scales evaluated, more keypoints found
  - $S < 3$, stable keypoints increased too
  - $S > 3$, stable keypoints decreased
  - $S = 3$, maximum stable keypoints found

[0] Source: Kristen Grauman

Accurate keypoint localization



- Fit a 3D quadratic function to the local sample points to determine the interpolated location of the maximum.
- The idea is to use the Taylor expansion of the scale-space function:

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

- Here D and its derivates are evaluated at the sample point and $\mathbf{x} = (x, y, \delta)^T$ is the offset from this point.

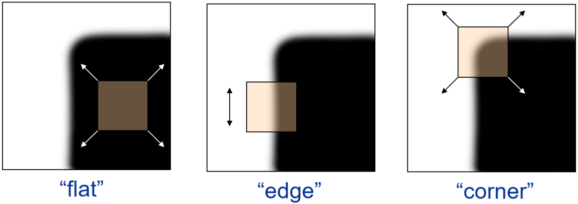
Removing low contrast features

- If the intensity value (i.e., without sign) at the current pixel in the DoG image (that is being checked for minima/maxima) is less than a certain value, it is rejected.
- The subpixel intensity value is computed using Taylor expansion and if it is less than a threshold, it is rejected.

[0] Source: aishack.in

Presentation Outline

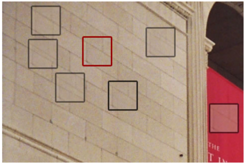
Corners as keypoints



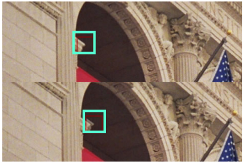
- **A flat region:** Both the gradients are small.
- **An edge:** One gradient will be big (perpendicular to the edge) and the other will be small (along the edge).
- **A corner:** Both gradients are large. If both the gradients are large, we allow it to be treated as keypoints.

[0]Source: Kristen Grauman

Harris corner detector



Small window motions produce small variations



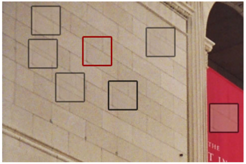
Small window motions produce large variations

- Let us consider the intensity change by moving a window by a small displacement:

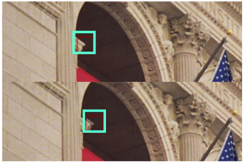
$$E(x,y) = \sum_{x,y} w(x,y)[I(x+u,y+v) - I(x,y)]^2$$

[0]Source:aishack.in

Harris corner detector



Small window motions produce small variations

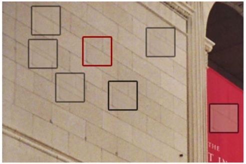


Small window motions produce large variations

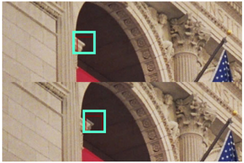
- $$E(x, y) = \sum_{x,y} w(x, y)[I(x + u, y + v) - I(x, y)]^2$$
- $E(x, y)$ is the difference between the original and the moved window.
- (u, v) is the window's displacement.
- $I(x, y)$ is the image intensity at (x, y).

[0]Source:aishack.in

Harris corner detector



Small window motions produce small variations



Small window motions produce large variations

- Let us consider the intensity change by moving a window by a small displacement:

$$E(x,y) = \sum_{x,y} w(x,y)[I(x+u,y+v) - I(x,y)]^2$$

- We are looking for windows that produce large E value. Maximize E:

$$E(u,v) \approx \sum_{x,y} w(x,y)[I(x,y) + uI_x + vI_y - I(x,y)]^2$$

Harris corner detector

$$E(u,v) \approx \sum_{x,y} w(x,y)[uI_x + vI_y]^2$$

$$E(u,v) \approx \sum_{x,y} w(x,y)[u^2 I_x^2 + v^2 I_y^2 + 2uvI_x I_y]$$

$$E(u,v) \approx \sum_{x,y} w(x,y) \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

[0] Source:aishack.in

Harris corner detector

$$E(u,v) \approx \sum_{x,y} w(x,y) \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \mathrm{M} \begin{bmatrix} u \\ v \end{bmatrix}$$

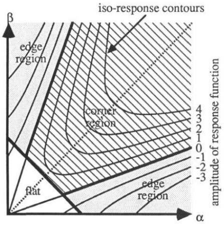
where

$$\mathrm{M} = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

---

[0] Source:aishack.in

Harris Corner Detector

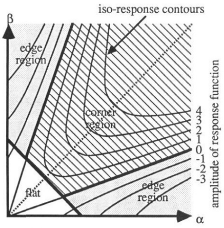


- Let α and β be the Eigenvalues of the matrix M. The Eigenvalues determine the stability of a matrix.

$$\begin{aligned} R &= det(M) - k(trace(M))^2 \\ det(M) &= \alpha \times \beta \\ trace(M) &= \alpha + \beta \end{aligned}$$

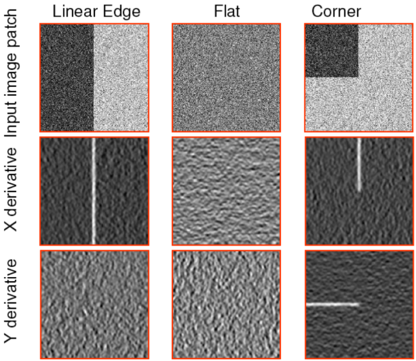
[0] Source: Harris and Stevens, 1988

Harris Corner Detector



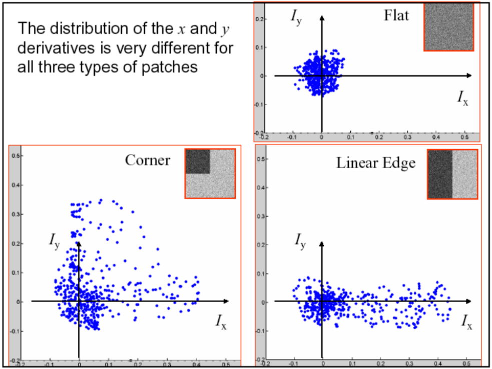
- $$\begin{aligned} R &= det(M) - k(trace(M))^2 \\ det(M) &= \alpha \times \beta \\ trace(M) &= \alpha + \beta \end{aligned}$$
- If both eigenvalues are small, then the pixel is "flat" (the white region).
- If one eigenvalue is large, and the other is small, then the pixel is an edge (the gray region).

Images and 2D Derivatives



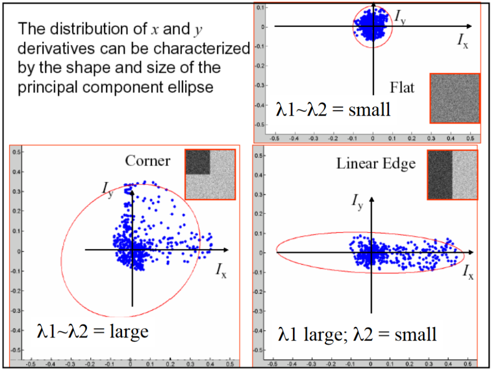
[0]Source: Robert Collins

2D Derivatives as points



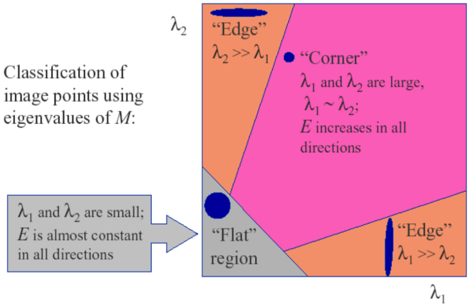
[0] Source: Robert Collins

Fit ellipses to the 2D derivatives



[0]Source: Robert Collins

Classification based on Eigen Values



[0] Source: Robert Collins

Corner Response Measure

Measure of corner response:

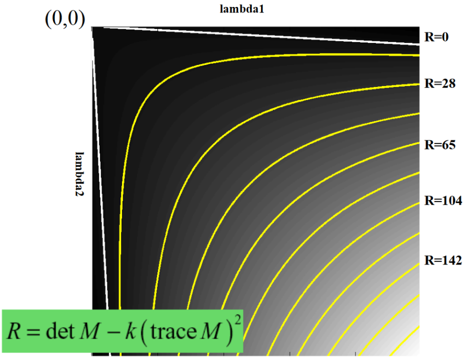
$$R = \det M - k\left(\operatorname{trace} M\right)^2$$

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

(k is an empirically determined constant; $k = 0.04$ - 0.06)

[0] Source: Robert Collins

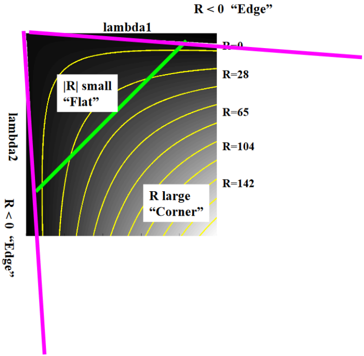
Corner Response Measure



[0] Source: Robert Collins

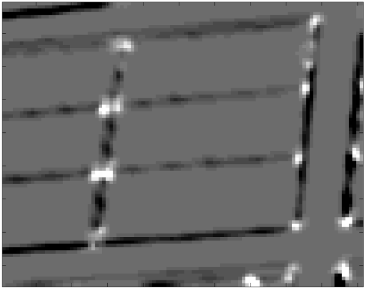
Corner Response Measure

- R depends only on eigenvalues of M
- R is large for a corner
- R is negative with large magnitude for an edge
- $|R|$ is small for a flat region



[0]Source: Robert Collins

Corner Response Measure

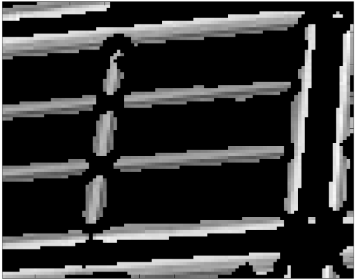


Harris R score.

Ix, Iy computed using Sobel operator
Windowing function w = Gaussian, sigma=1

[0] Source: Robert Collins

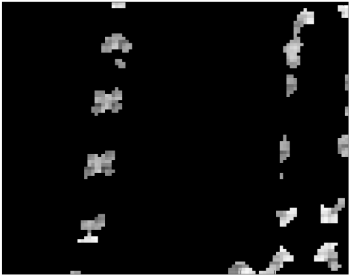
Corner Response Measure



Threshold: R < -10000
(edges)

[0]Source: Robert Collins

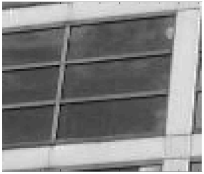
Corner Response Measure

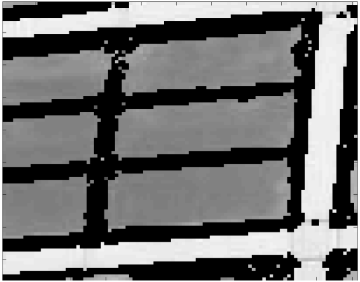


Threshold: > 10000
(corners)

[0]Source: Robert Collins

Corner Response Measure



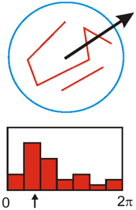


Threshold: -10000 < R < 10000
(neither edges nor corners)

[0] Source: Robert Collins

Presentation Outline

Orientation assignment

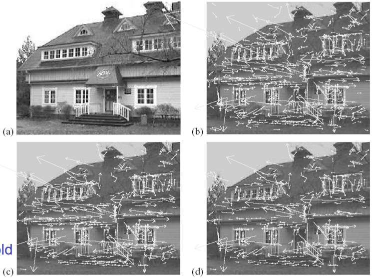


- Create histogram of local gradient directions at selected scale.
- Assign canonical orientation at peak of smoothed histogram.
- Each key specifies stable 2D coordinates (x,y,scale, orientations)
- If there are two major orientations, then use both.

[0]Source: Kristen Grauman

Keypoint localization with orientation

233x189



832

initial keypoints

729

keypoints after gradient threshold

536

keypoints after ratio threshold

[0] Source: Kristen Grauman

Keypoint Descriptors

- At this point, each keypoint has
  - location
  - scale
  - orientation
- Next is to compute a descriptor for the local image region about each keypoint that is
  - highly distinctive
  - invariant as possible to variations such as changes in viewpoint and illumination

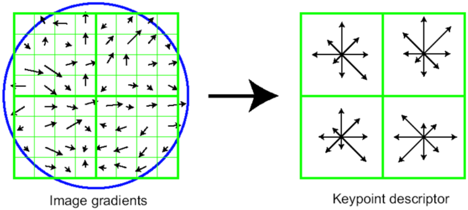
[0] Source: Kristen Grauman

Normalization

- Rotate the window to standard orientation.
- Scale the window size based on the scale at which the point was found.

[0] Source: Kristen Grauman

Lowe's Keypoint Descriptor



- Shown with 2x2 descriptors over 8x8.
- In experiments, 4x4 arrays of 8 bin histograms is used, a total of 128 features for one keypoint.

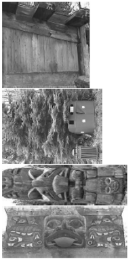
[0]Source: Kristen Grauman

Lowe's Keypoint Descriptor

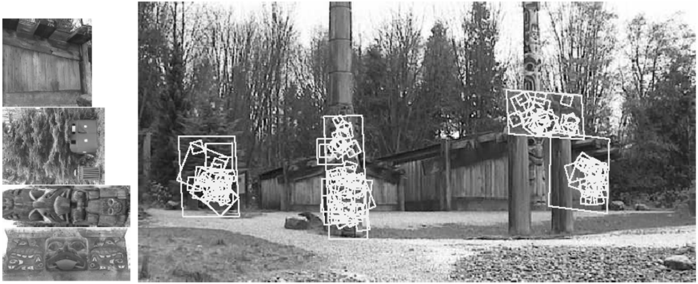
- Use the normalized region about the keypoint.
- Compute gradient magnitude and orientation at each point in the region.
- Weight them by a Gaussian window overlaid on the circle.
- Create an orientation histogram over the 4x4 subregions of the window.
- 4x4 descriptors over 16x16 sample arrays were used in practice. 4x4 times 8 directions gives a vector of 128 values.

[0] Source: Kristen Grauman

SIFT for matching objects



SIFT for matching objects



Acknowledgments

Some presentation slides are adapted from David Lowe's landmark paper, Kristen Grauman, Andrew Zisserman, Joseph Sivic, wikipedia.org, Robert Collins, and Utkarsh Sinha (aishack.in)