Multimedia Computing: Algorithms, Systems, and Applications: Feature Extraction

By

Dr. Yu Cao

Department of Computer Science
The University of Massachusetts Lowell
Lowell, MA 01854, USA

Harris Corner Detector

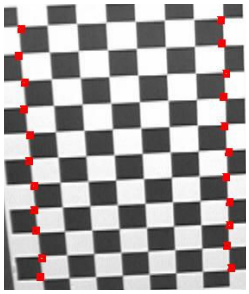
Finding Corners

Edge detectors perform poorly at corners.

Corners provide repeatable points for matching, so are worth detecting.

Idea:

- Exactly at a corner, gradient is ill defined.
- However, in the region around a corner, gradient has two or more different values.



3

The Harris corner detector

Form the second-moment matrix:

Sum over a small region around the hypothetical corner

Gradient with respect to x, times gradient with respect to y

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

Matrix is symmetric

4

Simple Case

First, consider case where:

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

This means dominant gradient directions align with x or y axis

If either λ is close to 0, then this is **not** a corner, so look for locations where both are large.

5

General Case

It can be shown that since C is symmetric:

$$C = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

So every case is like a rotated version of the one on last slide.

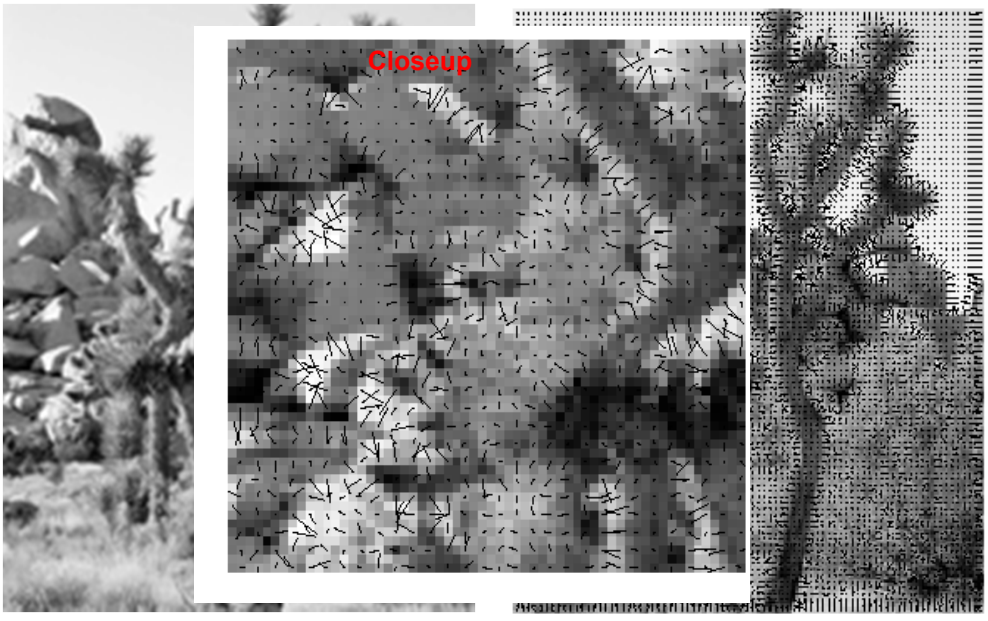
6

So, To Detect Corners

- Filter image
  - with Gaussian to reduce noise
- Compute magnitude of the gradient everywhere
  - magnitude of the x and y gradients at each pixel
- Construct C in a window
  - Construct C in a window around each pixel (Harris uses a Gaussian window – just blur)
- Use Linear Algebra to find $\lambda 1$ and $\lambda 2$
  - Solve for product of λs (determinant of C)
- If they are both big, we have a corner
  - If λs are both big (product reaches local maximum and is above threshold), we have a corner (Harris also checks that ratio of λs is not too high)

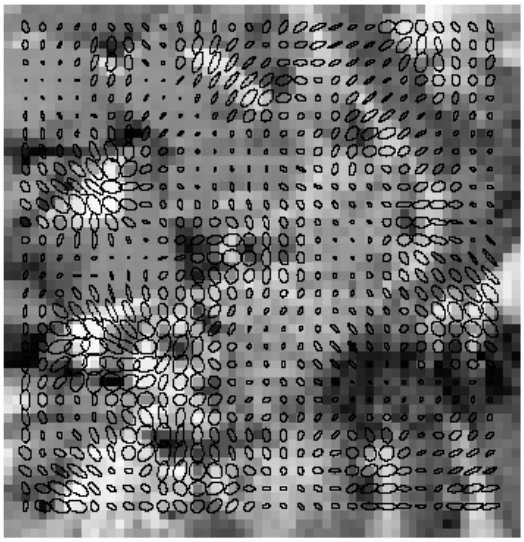
7

Gradient Orientation



8

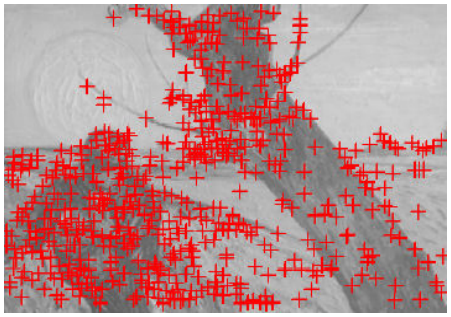
Corner Detection

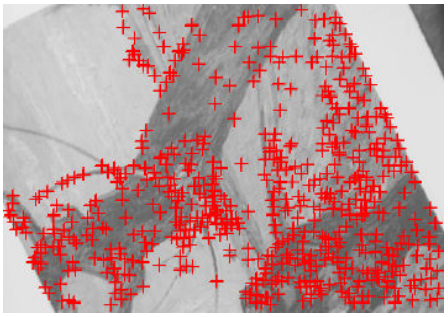


Corners are detected where the product of the ellipse axis lengths reaches a local maximum.

9

Harris Corners





- Originally developed as features for motion tracking
- Greatly reduces amount of computation compared to tracking every pixel
- Translation and rotation invariant (but not scale invariant)

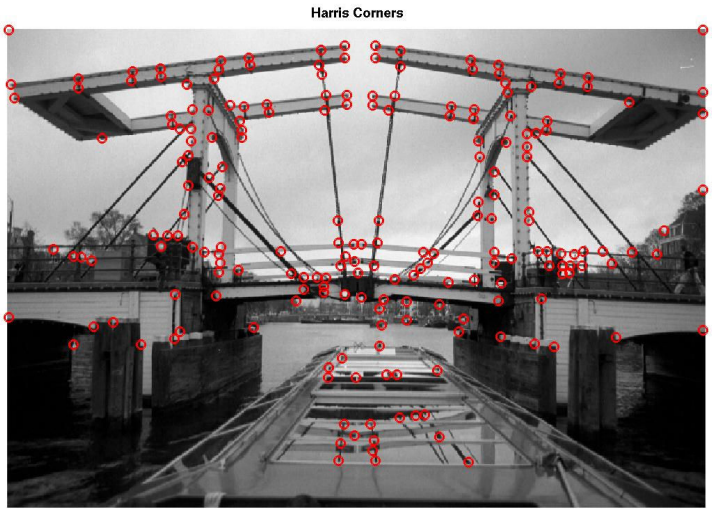
10

Harris Corner in Matlab

```
% Harris Corner detector - by Kashif Shahzad
sigma=2; thresh=0.1; sze=11; disp=0;

% Derivative masks
dy = [-1 0 1; -1 0 1; -1 0 1];
dx = dy'; %dx is the transpose matrix of dy

% Ix and Iy are the horizontal and vertical edges of image
Ix = conv2(bw, dx, 'same');
Iy = conv2(bw, dy, 'same');

% Calculating the gradient of the image Ix and Iy
g = fspecial('gaussian',max(1,fix(6*sigma)), sigma);
Ix2 = conv2(Ix.^2, g, 'same');                    % Smoothed squared image derivatives
Iy2 = conv2(Iy.^2, g, 'same');
Ixy = conv2(Ix.*Iy, g, 'same');

% My preferred  measure according to research paper
cornerness = (Ix2.*Iy2 - Ixy.^2)./(Ix2 + Iy2 + eps);

% We should perform nonmaximal suppression and threshold
mx = ordfilt2(cornerness,sze^2,ones(sze));        % Grey-scale dilate
cornerness = (cornerness==mx)&(cornerness>thresh); % Find maxima
[rws,cols] = find(cornerness);                    % Find row,col coords.

clf ; imshow(bw);
hold on;
p=[cols rws];
plot(p(:,1),p(:,2),'or');
title('\bf Harris Corners')
```

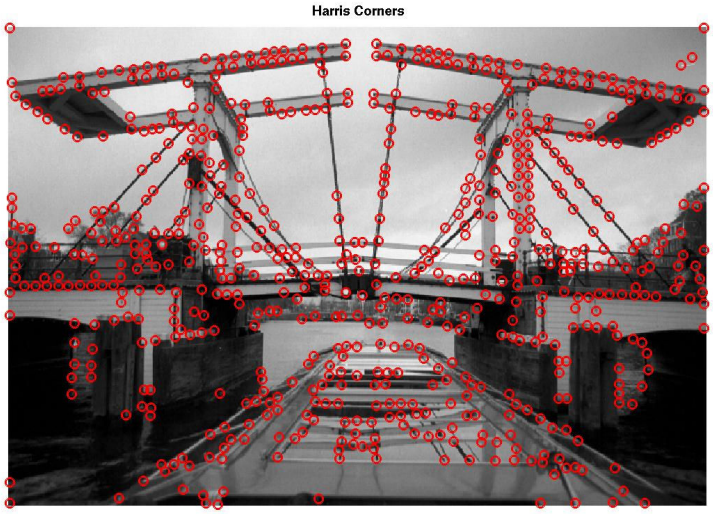
11

Example (σ=0.1)



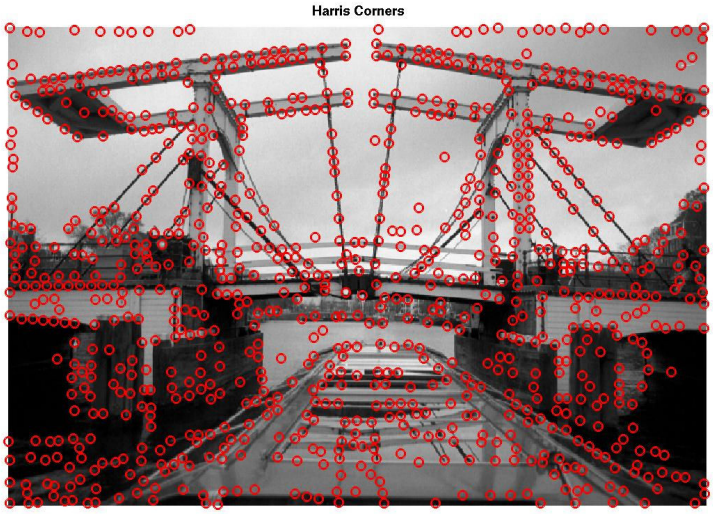
12

Example (σ=0.01)



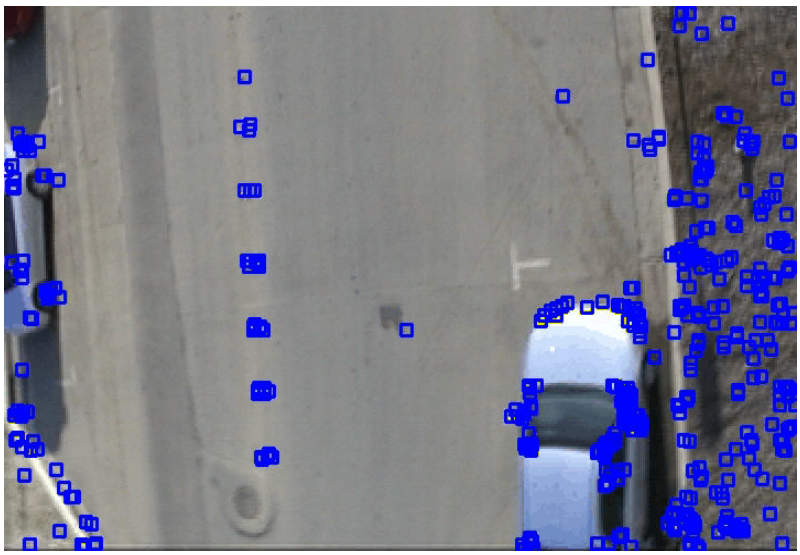
13

Example (σ=0.001)



14

Harris: OpenCV Implementation

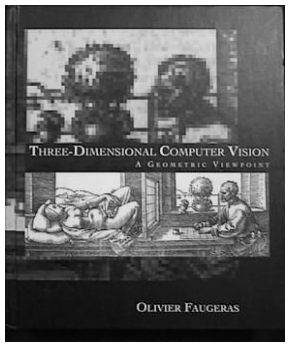


15

Template Matching

16

Problem: Features for Recognition



17

Templates

- Find an object in an image!
- Want Invariance!
  - Scaling
  - Rotation
  - Illumination
  - Deformation

18

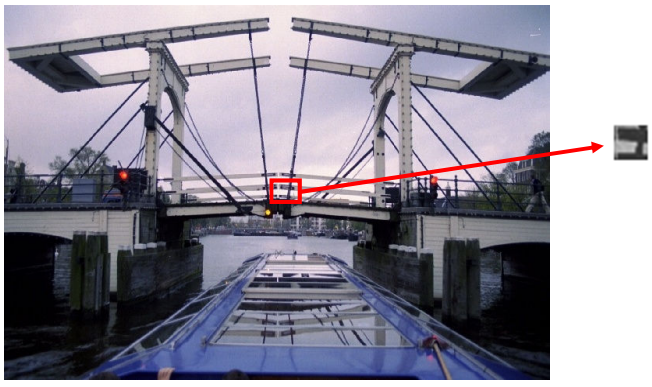
Convolution with Templates

```
% read image
im = imread('bridge.jpg');
bw = double(im(:,:,1)) ./ 256;
imshow(bw)

% apply FFT
FFTim = fft2(bw);
bw2 = real(ifft2(FFTim));
imshow(bw2)

% define a kernel
kernel=zeros(size(bw));
kernel(1, 1) = 1;
kernel(1, 2) = -1;
FFTkernel = fft2(kernel);

% apply the kernel and check out
    the result
FFTresult = FFTim .*
    FFTkernel;
result = real(ifft2(FFTresult));
imshow(result)
```

```
% select an image patch
patch = bw(221:240,351:370);
imshow(patch)
patch = patch - (sum(sum(patch)) / size(patch,1) /
    size(patch, 2));

kernel=zeros(size(bw));
kernel(1:size(patch,1),1:size(patch,2)) = patch;
FFTkernel = fft2(kernel);

% apply the kernel and check out the result
FFTresult = FFTim .* FFTkernel;
result = max(0, real(ifft2(FFTresult)));
result = result ./ max(max(result));
result = (result .^ 1 > 0.5);
imshow(result)

% alternative convolution
imshow(conv2(bw, patch, 'same'))
```

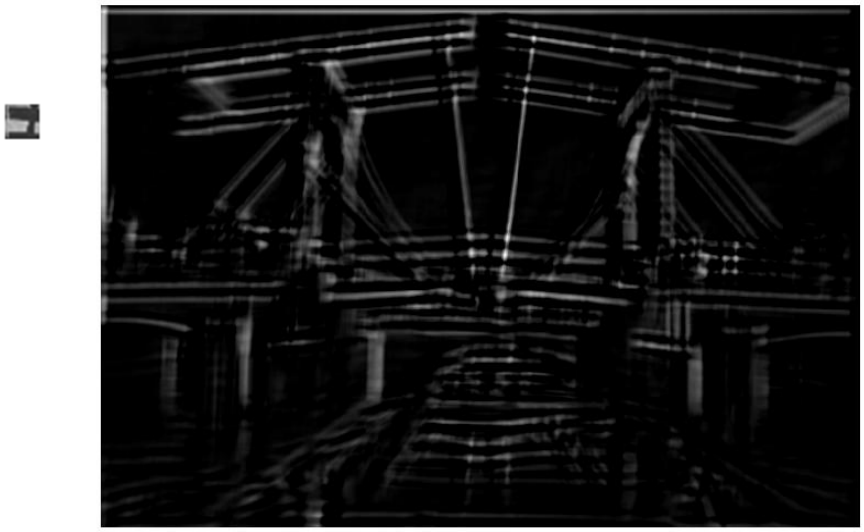
19

Template Convolution



20

Template Convolution



21

Recap: Convolution Theorem

$$F(I \otimes g) = F(I) \cdot F(g)$$

$$F(g(x,y))(u,v) = \int_{\Re^2} \int g(x,y) \ \exp\{-i2\pi(ux+vy)\} \ dx\,dy$$

22

Convolution with Templates

```
% read image
im = imread('bridge.jpg');
bw = double(im(:,:,1)) ./ 256;;
imshow(bw)

% apply FFT
FFTim = fft2(bw);
bw2 = real(ifft2(FFTim));
imshow(bw2)

% define a kernel
kernel=zeros(size(bw));
kernel(1, 1) =  1;
kernel(1, 2) = -1;
FFTkernel = fft2(kernel);

% apply the kernel and check out
    the result
FFTresult = FFTim .*
    FFTkernel;
result = real(ifft2(FFTresult));
imshow(result)
```

```
% select an image patch
patch = bw(221:240,351:370);
imshow(patch)
patch = patch - (sum(sum(patch)) / size(patch,1) /
    size(patch, 2));

kernel=zeros(size(bw));
kernel(1:size(patch,1),1:size(patch,2)) = patch;
FFTkernel = fft2(kernel);

% apply the kernel and check out the result
FFTresult = FFTim .* FFTkernel;
result = max(0, real(ifft2(FFTresult)));
result = result ./ max(max(result));
result = (result .^ 1 > 0.5);
imshow(result)

% alternative convolution
imshow(conv2(bw, patch, 'same'))
```

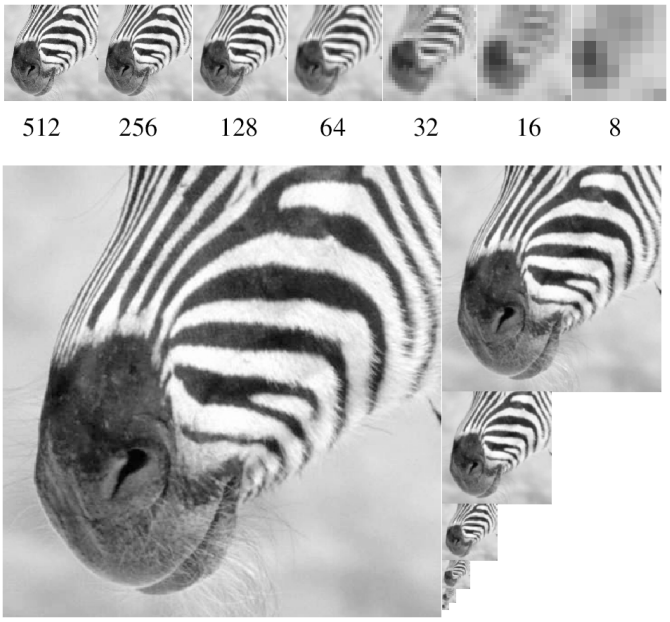
23

Convolution with Templates

- Invariances:
  - Scaling — No
  - Rotation — No
  - Illumination — No
  - Deformation — Maybe
- Provides
  - Good localization — No

24

Scale Invariance: Image Pyramid



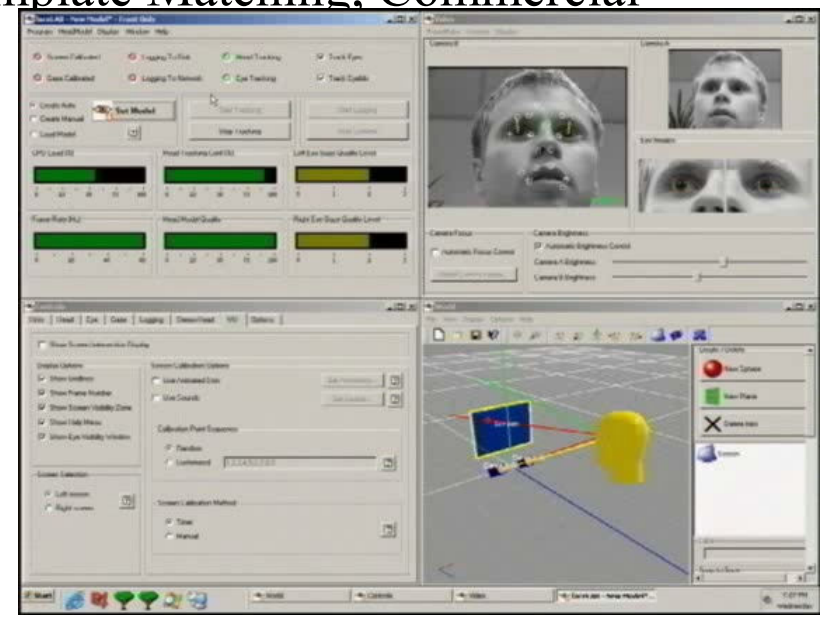
25

Templates with Image Pyramid

- Invariances:
  - Scaling — Yes
  - Rotation — No
  - Illumination — No
  - Deformation — Maybe
- Provides
  - Good localization — No

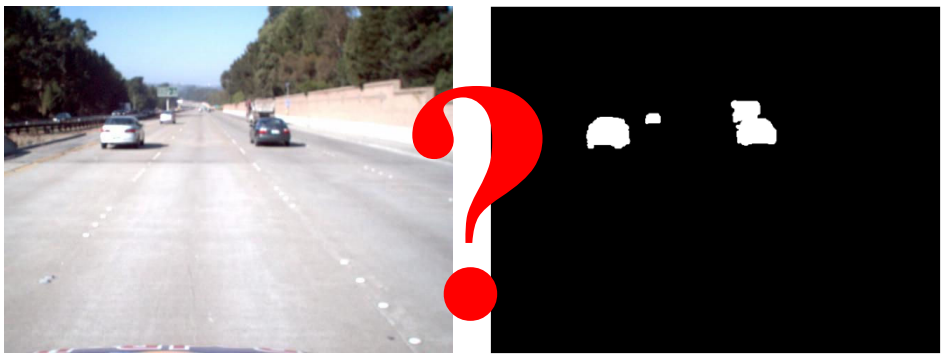
26

Template Matching, Commercial



27

Templates

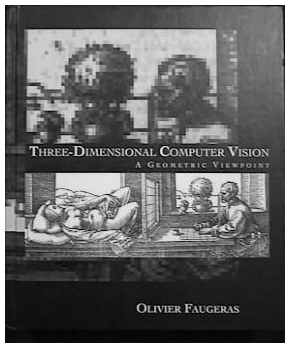


28

Scale-invariant feature transform (SIFT)

29

Let's Return to this Problem…



30

SIFT

- Invariances:
  - Scaling — Yes
  - Rotation — Yes
  - Illumination — Yes
  - Deformation — Maybe
- Provides
  - Good localization — Yes

31

SIFT Reference

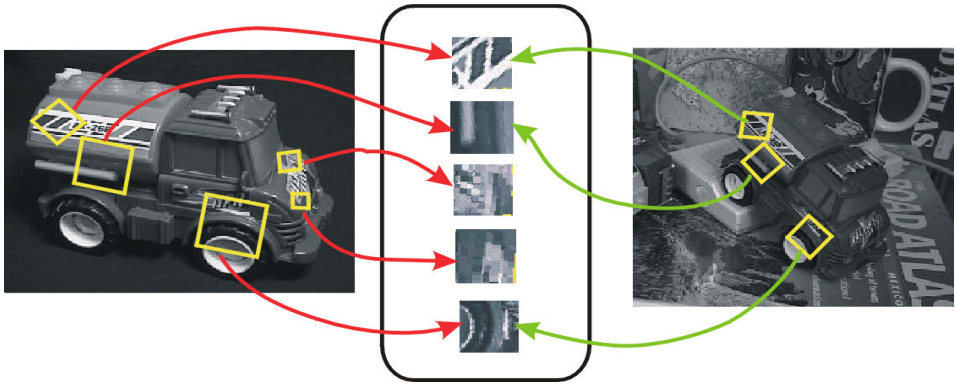
Distinctive image features from scale-invariant keypoints. David G. Lowe, International Journal of Computer Vision, 60, 2 (2004), pp. 91-110.

SIFT = Scale Invariant Feature Transform

32

Invariant Local Features

- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



33

Advantages of invariant local features

- **Locality:** features are local, so robust to occlusion and clutter (no prior segmentation)
- **Distinctiveness:** individual features can be matched to a large database of objects
- **Quantity:** many features can be generated for even small objects
- **Efficiency:** close to real-time performance
- **Extensibility:** can easily be extended to wide range of differing feature types, with each adding robustness

34

SIFT On-A-Slide

1. **Enforce invariance to scale: Compute Gaussian difference max, for may different scales; non-maximum suppression, find local maxima: keypoint candidates**
2. **Localizable corner: For each maximum fit quadratic function. Compute center with sub-pixel accuracy by setting first derivative to zero.**
3. **Eliminate edges: Compute ratio of eigenvalues, drop keypoints for which this ratio is larger than a threshold.**
4. **Enforce invariance to orientation: Compute orientation, to achieve scale invariance, by finding the strongest second derivative direction in the smoothed image (possibly multiple orientations). Rotate patch so that orientation points up.**
5. **Compute feature signature: Compute a "gradient histogram" of the local image region in a 4x4 pixel region. Do this for 4x4 regions of that size. Orient so that largest gradient points up (possibly multiple solutions). Result: feature vector with 128 values (15 fields, 8 gradients).**
6. **Enforce invariance to illumination change and camera saturation: Normalize to unit length to increase invariance to illumination. Then threshold all gradients, to become invariant to camera saturation.**

35

SIFT On-A-Slide

Step 1: Scale-space extrema detection

1. **Enforce invariance to scale: Compute Gaussian difference max, for may different scales; non-maximum suppression, find local maxima: keypoint candidates**
2. **Localizable corner: For each maximum fit quadratic function. Compute center with sub-pixel accuracy by setting first derivative to zero.**
3. **Eliminate edges: Compute ratio of eigenvalues, drop keypoints for which this ratio is larger than a threshold.**
4. **Enforce invariance to orientation: Compute orientation, to achieve scale invariance, by finding the strongest second derivative direction in the smoothed image (possibly multiple orientations). Rotate patch so that orientation points up.**
5. **Compute feature signature: Compute a "gradient histogram" of the local image region in a 4x4 pixel region. Do this for 4x4 regions of that size. Orient so that largest gradient points up (possibly multiple solutions). Result: feature vector with 128 values (15 fields, 8 gradients).**
6. **Enforce invariance to illumination change and camera saturation: Normalize to unit length to increase invariance to illumination. Then threshold all gradients, to become invariant to camera saturation.**

36

Find Invariant Corners

1. **Enforce invariance to scale: Compute Gaussian difference max, for may different scales; non-maximum suppression, find local maxima: keypoint candidates**

   - Keypoints are detected using scale-space extrema in difference-of-Gaussian function *D*
   - *D* definition:

$$D(x,y,\sigma) = (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y)$$
$$= L(x,y,k\sigma) - L(x,y,\sigma)$$

   - Efficient to compute

37

Finding "Keypoints" (Corners)

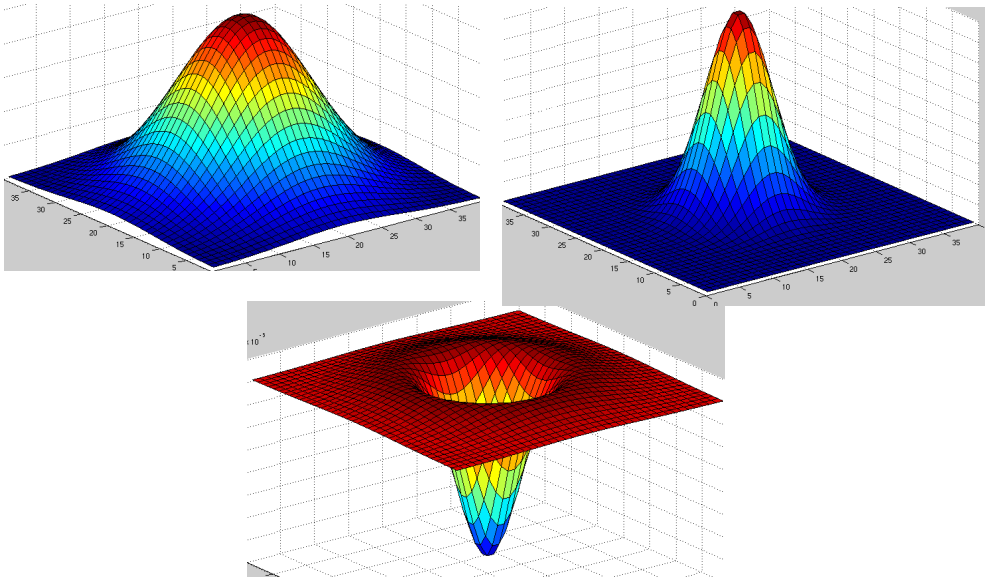
Idea: Find Corners, but scale invariance

Approach:

- Run linear filter (diff of Gaussians)
- At different resolutions of image pyramid

38

Difference of Gaussians



39

Difference of Gaussians

surf(fspecial('gaussian',40,4))
surf(fspecial('gaussian',40,8))
surf(fspecial('gaussian',40,8) - fspecial('gaussian',40,4))

40

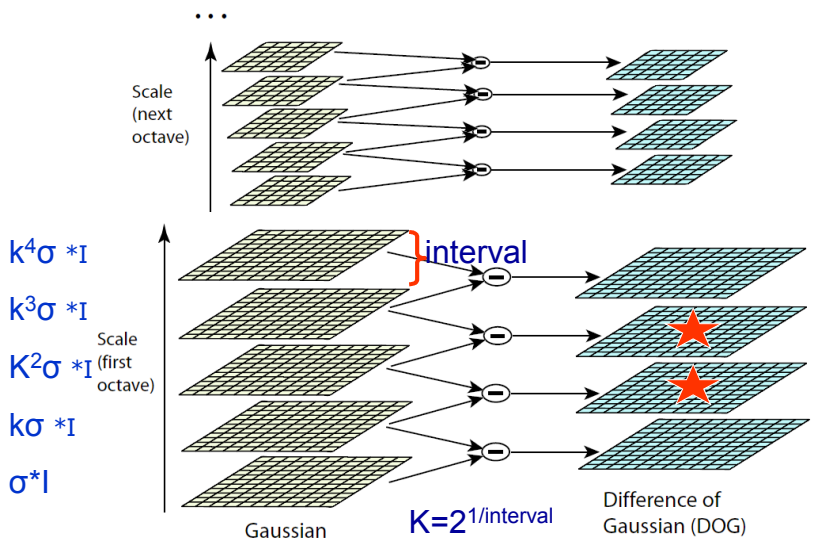
Find Corners with DiffOfGauss

```
im =imread('bridge.jpg');
bw = double(im(:,:,1)) / 256;

for i = 1 : 10
  gaussD = fspecial('gaussian',40,2*i) - fspecial('gaussian',40,i);
  res = abs(conv2(bw, gaussD,  'same'));
  res = res / max(max(res));
  imshow(res) ; title(['\bf i = ' num2str(i)]); drawnow
end
```

41

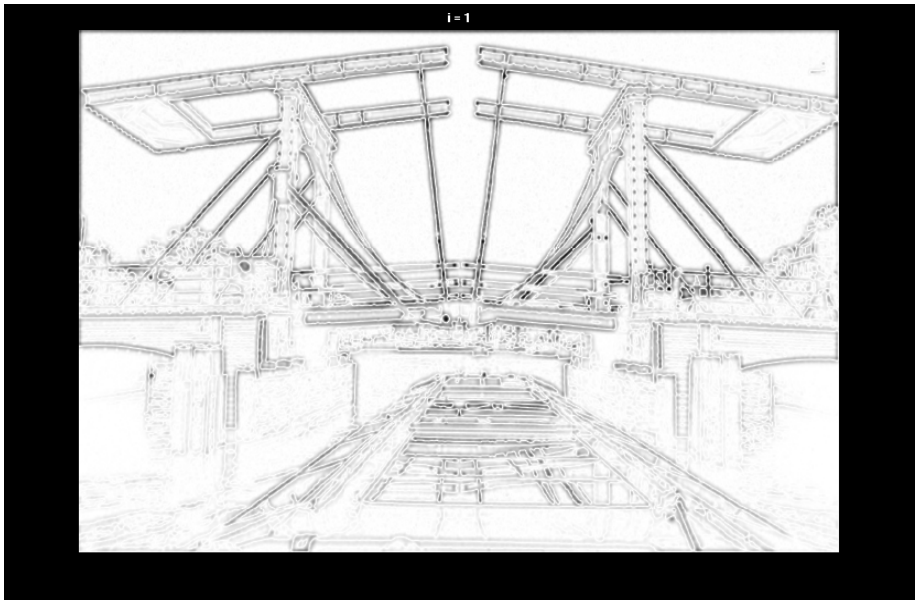
Scale-space extrema detection

- Scale space construction



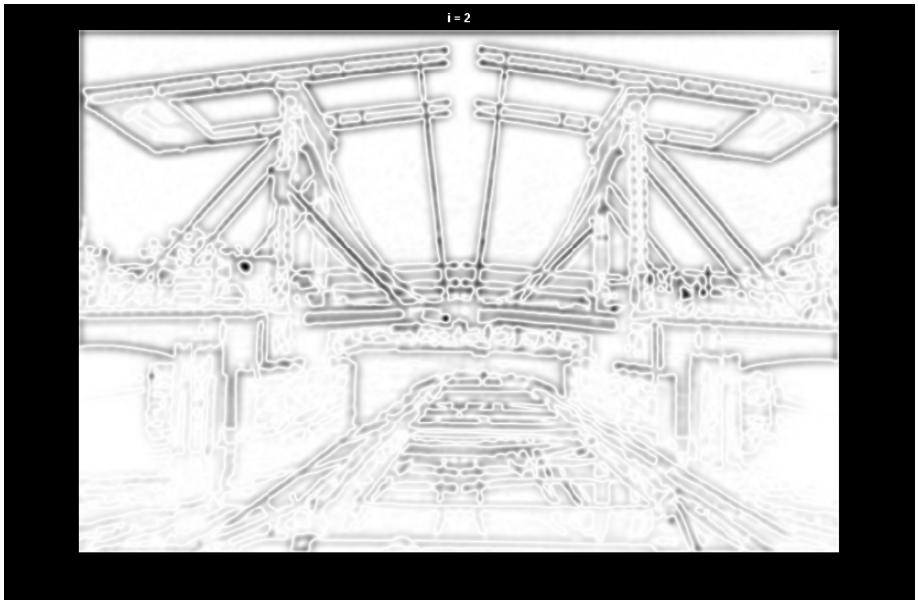
42

Gaussian Kernel Size i=1



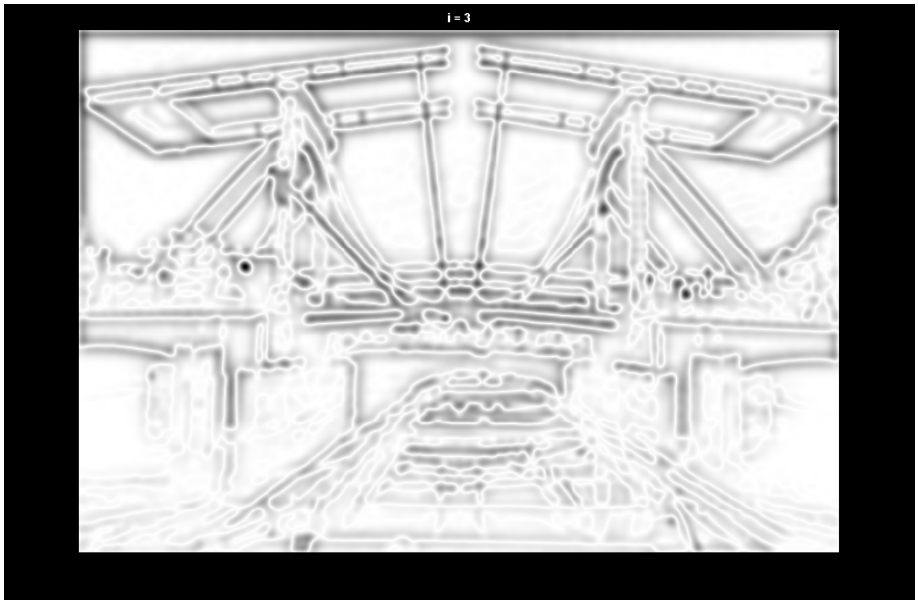
43

Gaussian Kernel Size i=2



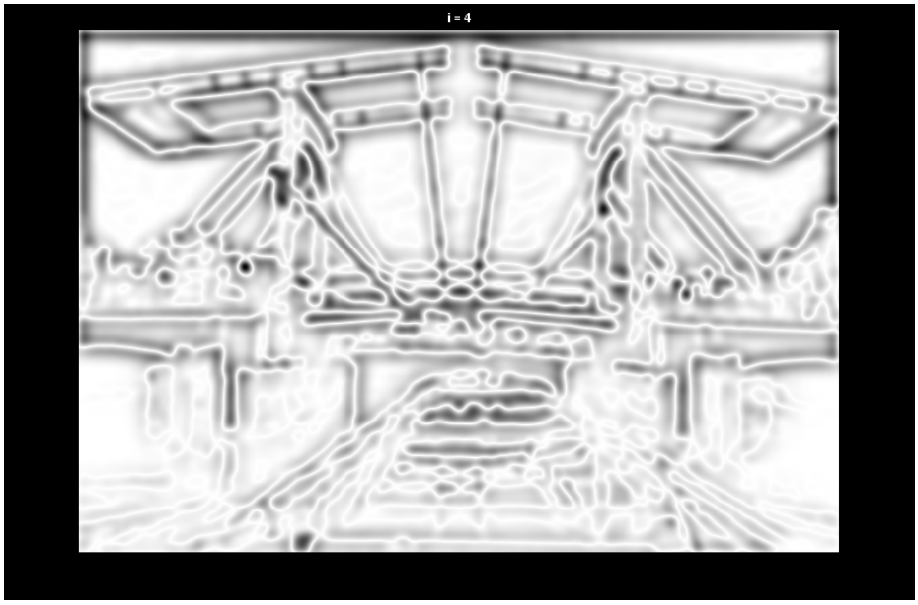
44

Gaussian Kernel Size i=3



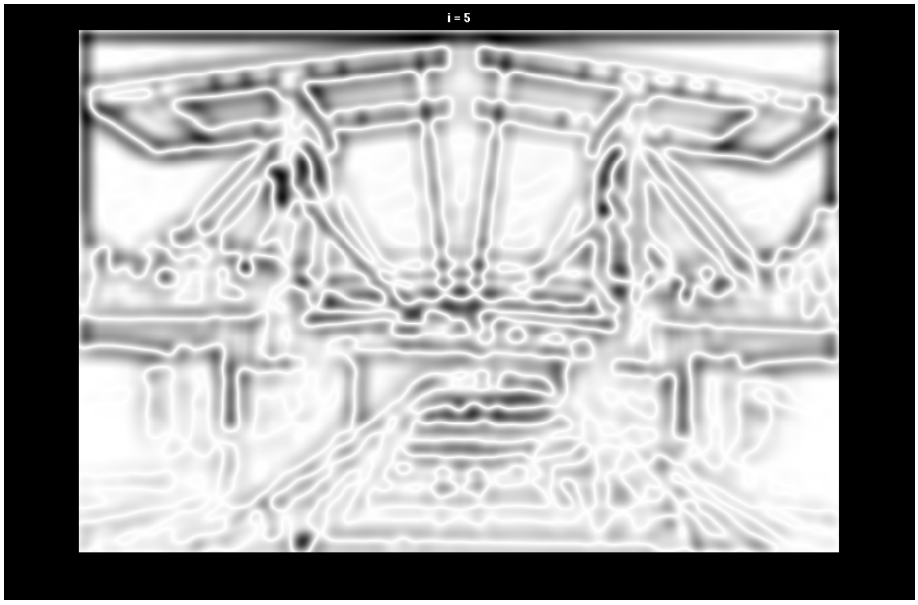
45

Gaussian Kernel Size i=4



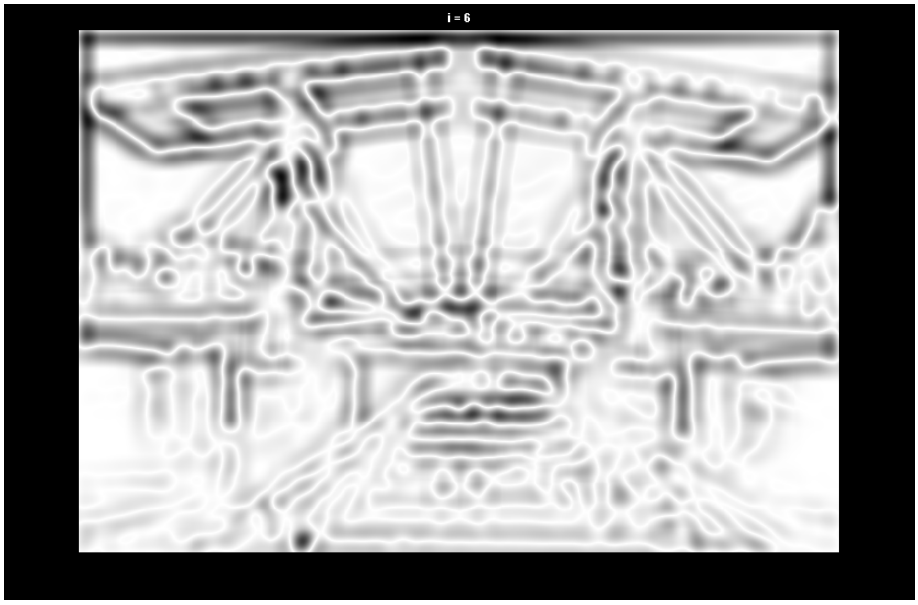
46

Gaussian Kernel Size i=5



47

Gaussian Kernel Size i=6



48

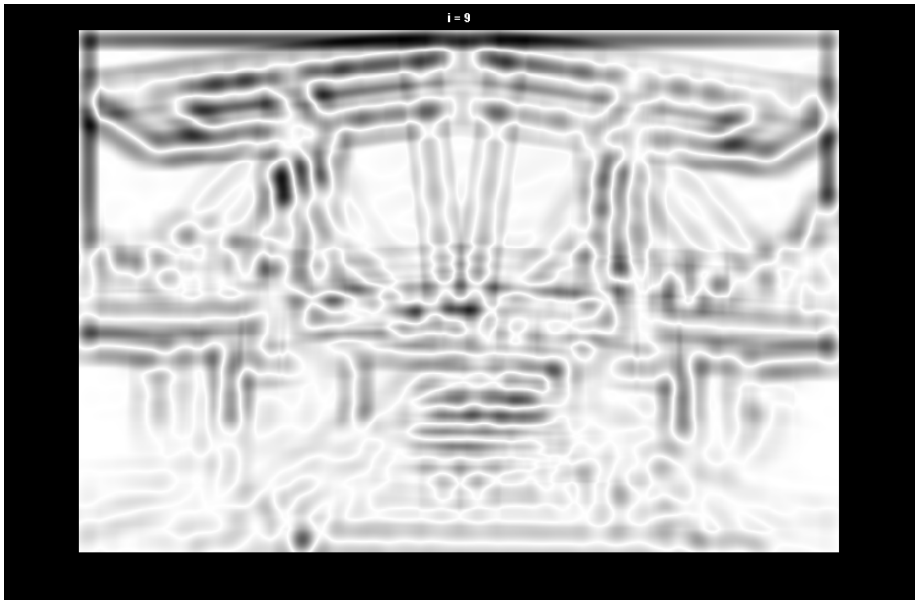
Gaussian Kernel Size i=7

49

Gaussian Kernel Size i=8

50

Gaussian Kernel Size i=9



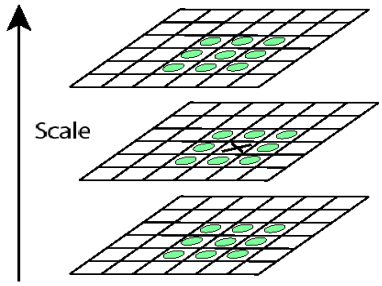
51

Gaussian Kernel Size i=10

52

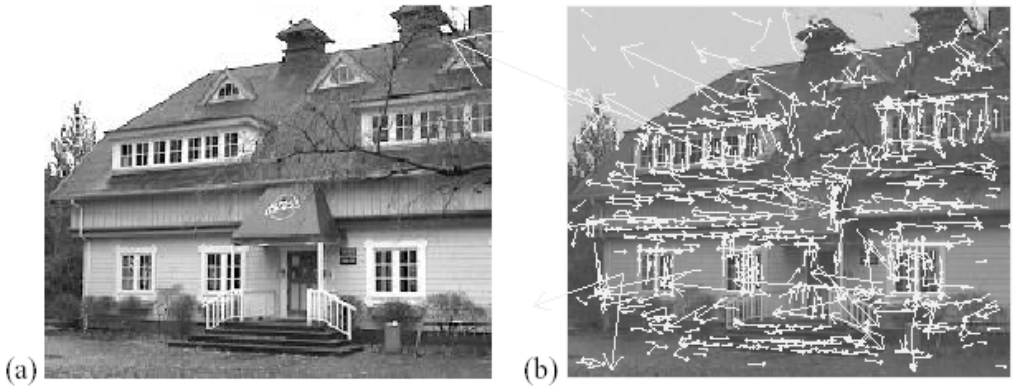
Key point localization

- Detect maxima and minima of difference-of-Gaussian in scale space
- Sample point is selected only if it is a minimum or a maximum of these points



53

Example of keypoint detection



(a) 233x189 image
(b) 832 DOG extrema
(c) 729 above threshold

54

SIFT On-A-Slide

Step 2: Key point localization

1. **Enforce invariance to scale: Compute Gaussian difference max, for may different scales; non-maximum suppression, find local maxima: keypoint candidates**
2. **Localizable corner: For each maximum fit quadratic function. Compute center with sub-pixel accuracy by setting first derivative to zero.**
3. **Eliminate edges: Compute ratio of eigenvalues, drop keypoints for which this ratio is larger than a threshold.**
4. **Enforce invariance to orientation: Compute orientation, to achieve scale invariance, by finding the strongest second derivative direction in the smoothed image (possibly multiple orientations). Rotate patch so that orientation points up.**
5. **Compute feature signature: Compute a "gradient histogram" of the local image region in a 4x4 pixel region. Do this for 4x4 regions of that size. Orient so that largest gradient points up (possibly multiple solutions). Result: feature vector with 128 values (15 fields, 8 gradients).**
6. **Enforce invariance to illumination change and camera saturation: Normalize to unit length to increase invariance to illumination. Then threshold all gradients, to become invariant to camera saturation.**

55

Localization

- 3D quadratic function is fit to the local sample points
- Start with Taylor expansion with sample point as the origin $D(\mathrm{X}) = D + \frac{\partial D^T}{\partial \mathrm{X}}\mathrm{X} + \frac{1}{2}\mathrm{X}^T\frac{\partial^2 D}{\partial \mathrm{X}^2}\mathrm{X}$
  - where $\hat{\mathrm{X}} = -\frac{\partial^2 D^{-1}}{\partial \mathrm{X}^2}\frac{\partial D}{\partial \mathrm{X}}$
- Take the derivative with respect to X, and set it to 0, giving $0 = \frac{\partial D}{\partial X} + \frac{\partial^2 D}{\partial X^2}\hat{X}$
- $\mathrm{X} = (x, y, \sigma)^T$ is the location of the keypoint
- This is a 3x3 linear system

56

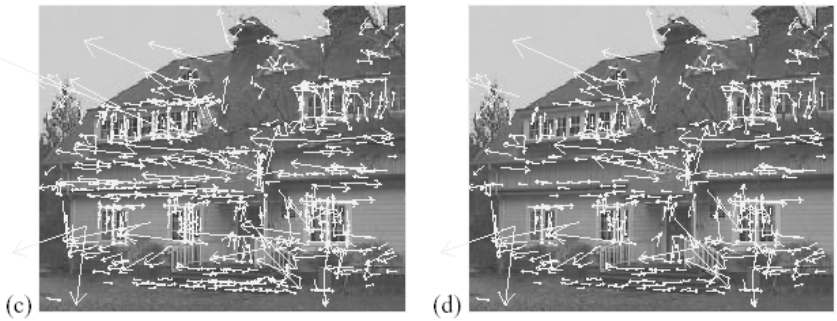
Filtering

- Contrast (use prev. equation): $D(\hat{X}) = D + \frac{1}{2}\frac{\partial D^T}{\partial X}\hat{X}$
  - If | *D(X)* | < 0.03, throw it out
- Edginess:
  - Use ratio of principal curvatures to throw out poorly defined peaks
  - Curvatures come from Hessian: $H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$
  - Ratio of *Trace(H)*^{2} and *Determinant(H)*

    $Tr(H) = D_{xx} + D_{yy}$

    $Det(H) = D_{xx}D_{yy} - (D_{xy})^2$
  - If ratio > $(r+1)^2/(r)$, throw it out (SIFT uses r=10)

Example of keypoint detection



(c) 729 left after peak value threshold (from 832)
(d) 536 left after testing ratio of principle curvatures

SIFT On-A-Slide

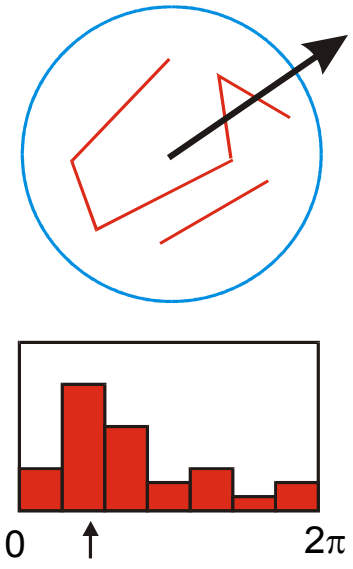
Step 3: Orientation assignment

1. **Enforce invariance to scale: Compute Gaussian difference max, for may different scales; non-maximum suppression, find local maxima: keypoint candidates**
2. **Localizable corner: For each maximum fit quadratic function. Compute center with sub-pixel accuracy by setting first derivative to zero.**
3. **Eliminate edges: Compute ratio of eigenvalues, drop keypoints for which this ratio is larger than a threshold.**
4. **Enforce invariance to orientation: Compute orientation, to achieve scale invariance, by finding the strongest second derivative direction in the smoothed image (possibly multiple orientations). Rotate patch so that orientation points up.**
5. **Compute feature signature: Compute a "gradient histogram" of the local image region in a 4x4 pixel region. Do this for 4x4 regions of that size. Orient so that largest gradient points up (possibly multiple solutions). Result: feature vector with 128 values (15 fields, 8 gradients).**
6. **Enforce invariance to illumination change and camera saturation: Normalize to unit length to increase invariance to illumination. Then threshold all gradients, to become invariant to camera saturation.**

59

Select canonical orientation

- Create histogram of local gradient directions computed at selected scale
- Assign canonical orientation at peak of smoothed histogram
- Each key specifies stable 2D coordinates (x, y, scale, orientation)



60

SIFT On-A-Slide

Step 4: Generation of key point descriptors

1. **Enforce invariance to scale: Compute Gaussian difference max, for may different scales; non-maximum suppression, find local maxima: keypoint candidates**
2. **Localizable corner: For each maximum fit quadratic function. Compute center with sub-pixel accuracy by setting first derivative to zero.**
3. **Eliminate edges: Compute ratio of eigenvalues, drop keypoints for which this ratio is larger than a threshold.**
4. **Enforce invariance to orientation: Compute orientation, to achieve scale invariance, by finding the strongest second derivative direction in the smoothed image (possibly multiple orientations). Rotate patch so that orientation points up.**
5. **Compute feature signature: Compute a "gradient histogram" of the local image region in a 4x4 pixel region. Do this for 4x4 regions of that size. Orient so that largest gradient points up (possibly multiple solutions). Result: feature vector with 128 values (15 fields, 8 gradients).**
6. **Enforce invariance to illumination change and camera saturation: Normalize to unit length to increase invariance to illumination. Then threshold all gradients, to become invariant to camera saturation.**

SIFT vector formation

- Thresholded image gradients are sampled over 16x16 array of locations in scale space
- Create array of orientation histograms
- 8 orientations x 4x4 histogram array = 128 dimensions

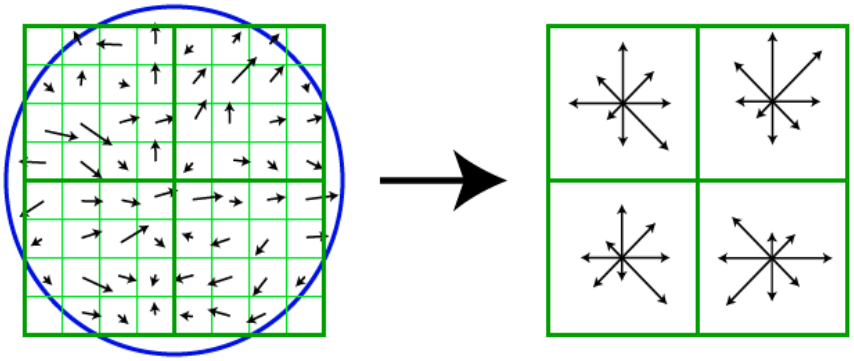


Image gradients

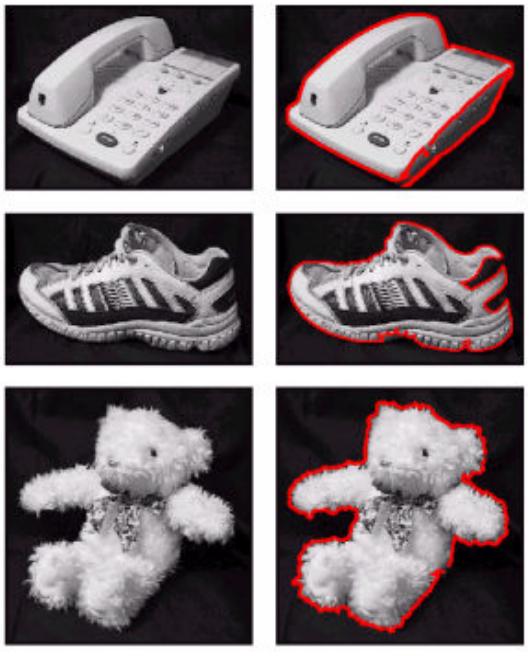
Keypoint descriptor

Nearest-neighbor matching to feature database

- Hypotheses are generated by **approximate nearest neighbor** matching of each feature to vectors in the database
  - SIFT use best-bin-first (Beis & Lowe, 97) modification to k-d tree algorithm
  - Use heap data structure to identify bins in order by their distance from query point
- **Result:** Can give speedup by factor of 1000 while finding nearest neighbor (of interest) 95% of the time

63

3D Object Recognition



- Extract outlines with background subtraction

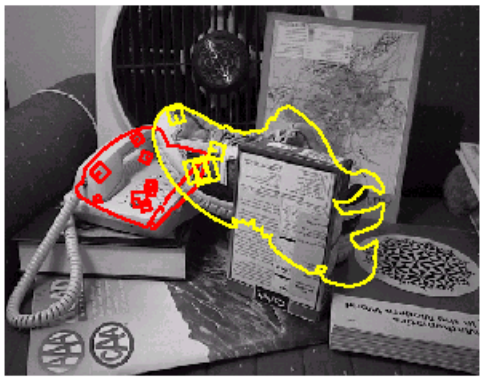
64

3D Object Recognition

- Only 3 keys are needed for recognition, so extra keys provide robustness
- Affine model is no longer as accurate

65

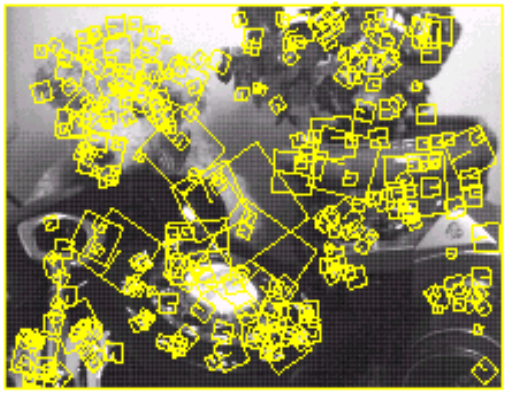
Recognition under occlusion



66

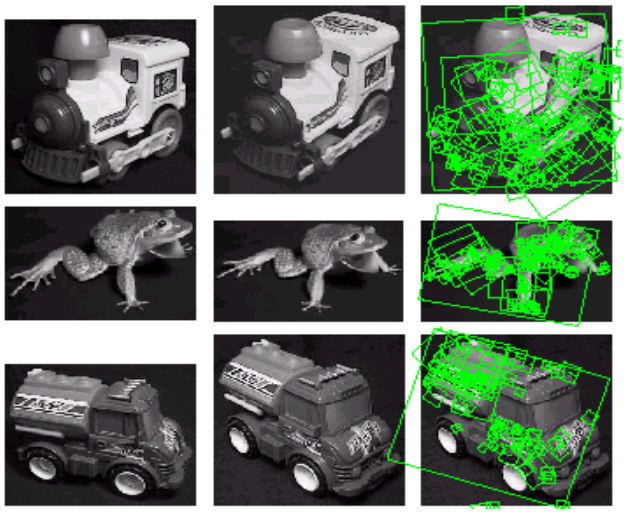
Test of illumination invariance

- Same image under differing illumination



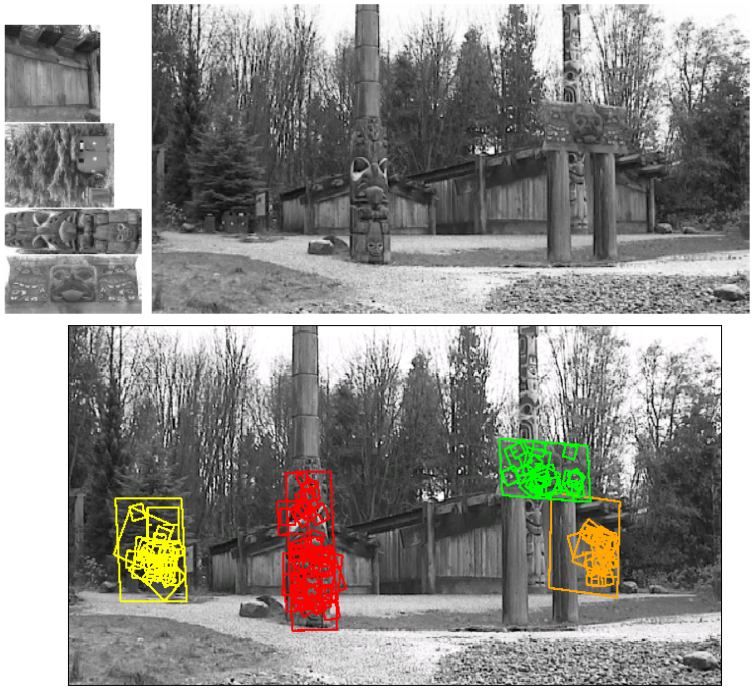
67

Examples of view interpolation



68

Location recognition



69

SIFT

- Invariances:
  - Scaling — Yes
  - Rotation — Yes
  - Illumination — Yes
  - Deformation — Maybe
- Provides
  - Good localization — Yes

70

Questions?

71