Uncalibrated Stereo and Feature Extraction

Computer Vision I

CSE 252A

Lecture 9

Announcements

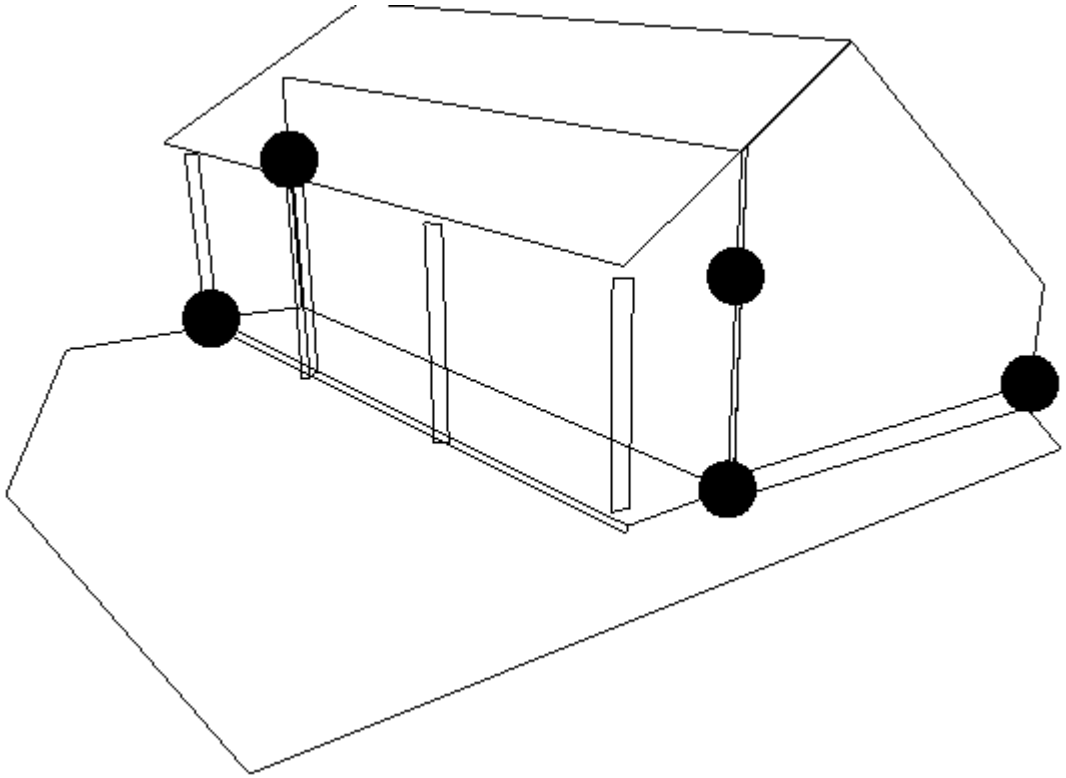
- Assignment 2 is due Nov 5, 11:59 PM

Calibrated stereo

- Offline
  - Calibration of stereo cameras
- Online
  1. Acquire stereo images
  2. Epipolar rectify stereo images
  3. Establish correspondence
  4. Estimate depth

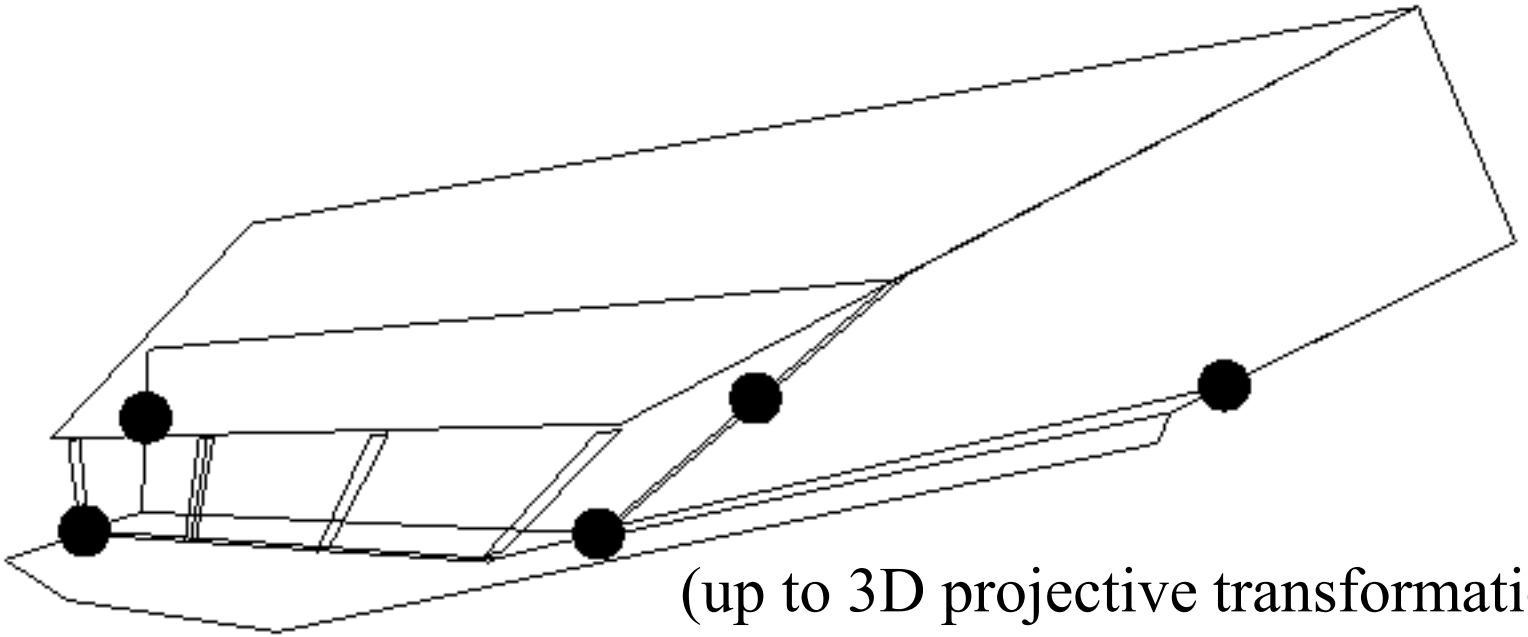
Calibrated stereo

- Known intrinsic camera parameters and extrinsic relationship between cameras
- Results in reconstructed 3D scene points



Uncalibrated stereo

- Unknown intrinsic camera parameters
- Unknown extrinsic relationship between cameras
- Results in reconstructed 3D scene points



(up to 3D projective transformation)

The essential and fundamental matrices

- Epipolar constraint using the essential matrix **E** and image points in **normalized coordinates**

$$\hat{\mathbf{x}}'^{\top}\mathtt{E}\hat{\mathbf{x}} = 0, \text{ where } \hat{\mathbf{x}} = \mathtt{K}^{-1}\mathbf{x} \text{ and } \hat{\mathbf{x}}' = \mathtt{K}'^{-1}\mathbf{x}'$$

$$(\mathtt{K}'^{-1}\mathbf{x}')^{\top}\mathtt{E}\mathtt{K}^{-1}\mathbf{x} = 0$$

$$\mathbf{x}'^{\top}\mathtt{K}'^{-\top}\mathtt{E}\mathtt{K}^{-1}\mathbf{x} = 0, \text{ where } \mathtt{K}'^{-\top} = (\mathtt{K}'^{-1})^{\top} = (\mathtt{K}'^{\top})^{-1}$$

$$\mathbf{x}'^{\top}\mathtt{F}\mathbf{x} = 0, \text{ where } \mathtt{F} = \mathtt{K}'^{-\top}\mathtt{E}\mathtt{K}^{-1}$$

- Epipolar constraint using the fundamental matrix **F** and image points in **pixel coordinates**

The essential and fundamental matrices

- In calibrated stereo, the essential matrix $\mathbf{E}$ is calculated from the rotation $\mathbf{R}$ and translation $\mathbf{t}$ of the second camera relative to the first

$$\mathtt{E} = [\mathbf{t}]_\times \mathtt{R}$$

- In uncalibrated stereo, the fundamental matrix $\mathbf{F}$ is calculated from point correspondences in pixel coordinates (covered next lecture)

$$\mathbf{x}'^\top \mathtt{F} \mathbf{x} = 0$$

The fundamental matrix

- Maps a point (in pixel coordinates) in the first image to its corresponding epipolar line (in pixel coordinates) in the second image

$$\boldsymbol{\ell}' = \mathbf{F}\mathbf{x}$$

- The epipolar line passes through the corresponding point in the second image

$$\mathbf{x}'^{\top}\boldsymbol{\ell}' = 0$$

$$\mathbf{x}'^{\top}\mathbf{F}\mathbf{x} = 0$$ The epipolar constraint

- Every epipolar line passes through the epipole

$$\mathbf{e}'^{\top}\boldsymbol{\ell}' = 0$$

The fundamental matrix

- Maps a point (in pixel coordinates) in the second image to its corresponding epipolar line (in pixel coordinates) in the first image

$$\boldsymbol{\ell} = \mathbf{F}^\top \mathbf{x}'$$

$$\boldsymbol{\ell}^\top = \mathbf{x}'^\top \mathbf{F}$$

- The epipolar line passes through the corresponding point in the first image

$$\boldsymbol{\ell}^\top \mathbf{x} = 0$$

$$\mathbf{x}'^\top \mathbf{F} \mathbf{x} = 0$$ The epipolar constraint

- Every epipolar line passes through the epipole

$$\boldsymbol{\ell}^\top \mathbf{e} = 0$$

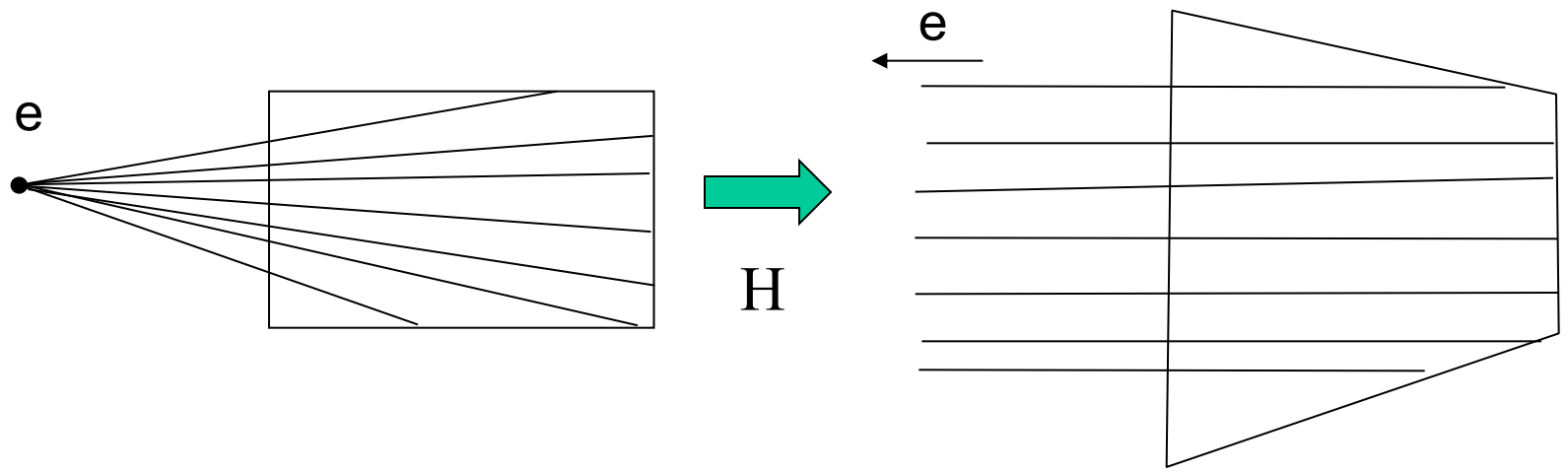
Example of using the fundamental matrix





Image pair rectification

Apply projective transformation so that epipolar lines correspond to horizontal scanlines



H should map epipole e to (1,0,0), a point at infinity on the x-axis

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = He$$

H should minimize image distortion

See Hartley and Zisserman, section 11.12, for algorithm

Note that rectified images are usually not rectangular

Epipolar rectification

The epipolar constraint must apply to both the original image pair (i.e., $\mathbf{x}'^{\top}\mathtt{F}\mathbf{x} = 0$) and rectified image pair (i.e., $\mathbf{x}'^{\top}_{\text{rectified}}\mathtt{F}_{\text{rectified}}\mathbf{x}_{\text{rectified}} = 0$).
Points in the original images map to points in the rectified images under the rectification transformations.

$$\mathbf{x}_{\text{rectified}} = \mathtt{H}\mathbf{x}$$
$$\mathbf{x}'_{\text{rectified}} = \mathtt{H}'\mathbf{x}'$$

Substitute this into $\mathbf{x}'^{\top}_{\text{rectified}}\mathtt{F}_{\text{rectified}}\mathbf{x}_{\text{rectified}} = 0$, then solve for $\mathtt{F}_{\text{rectified}}$.

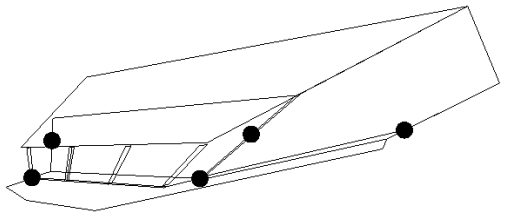
$$\begin{aligned}
\mathbf{x}'^{\top}_{\text{rectified}}\mathtt{F}_{\text{rectified}}\mathbf{x}_{\text{rectified}} &= 0 \\
(\mathtt{H}'\mathbf{x}')^{\top}\mathtt{F}_{\text{rectified}}\mathtt{H}\mathbf{x} &= 0 \\
\mathbf{x}'^{\top}\mathtt{H}'^{\top}\mathtt{F}_{\text{rectified}}\mathtt{H}\mathbf{x} &= 0 \\
\mathbf{x}'^{\top}\mathtt{F}\mathbf{x} &= 0, \text{ where } \mathtt{F} = \mathtt{H}'^{\top}\mathtt{F}_{\text{rectified}}\mathtt{H} \\
\mathtt{H}'^{-\top}\mathtt{F}\mathtt{H}^{-1} &= \mathtt{F}_{\text{rectified}}
\end{aligned}$$

Uncalibrated stereo

- **Intrinsic and extrinsic camera parameters are unknown**
- Input: Two images (or video frames)
- Detect features in images
- Determine sparse feature correspondences
- Compute the fundamental matrix (covered next lecture)
- Retrieve the relative uncalibrated camera projection matrices (up to 3D projective transformation) from the fundamental matrix
- Optional: epipolar rectify images and perform dense stereo matching using recovered epipolar geometry
- Triangulate corresponding 2D image points to estimate 3D scene points (up to 3D projective transformation)

Fundamental matrix

- 7 degrees of freedom
  - Two uncalibrated camera projection matrices (up to 3D projective transformation)
    - 24 elements, but 22 degrees of freedom
  - 3D projective transformation
    - 16 elements, 15 degrees of freedom
  - 22 – 15 = 7 degrees of freedom



The essential and fundamental matrices

Essential Matrix E

- Calibrated
- Normalized coordinates
- Rank 2
- 5 degrees of freedom
  - Camera rotation
  - Camera translation
  - Homogeneous matrix to scale
- Euclidean reconstruction

Fundamental Matrix F

- Uncalibrated
- Pixel coordinates
- Rank 2
- 7 degrees of freedom
  - Homogeneous matrix to scale
  - det F = 0
- Projective reconstruction

Uncalibrated stereo

- **Intrinsic and extrinsic camera parameters are unknown**
- Input: Two images (or video frames)
- Detect features in images
- Determine sparse feature correspondences
- Compute the fundamental matrix (covered next lecture)
- Retrieve the relative uncalibrated camera projection matrices (up to 3D projective transformation) from the fundamental matrix
- Optional: epipolar rectify images and perform dense stereo matching using recovered epipolar geometry
- Triangulate corresponding 2D image points to estimate 3D scene points (up to 3D projective transformation)

Feature extraction

- Feature extraction is comprised of
  - Feature detection
  - Feature description
    - A feature descriptor is
      - **Invariant** with respect to a set of transformations if its value remains unchanged after the application of any transformation from the family
      - **Covariant** with respect to a set of transformations if applying any transformation from the set produces the same result in the descriptor

Detection of corner-like features

- Corner
  - A rapid change of direction in a curve
  - A highly effective feature
    - Distinctive
    - Reasonably invariant to viewpoint

Detection of corner-like features



Slide credit: http://vims.cis.udel.edu/~chandra/

Corner-like features

- Minor eigenvalues of a gradient matrix are position and orientation covariant, but they are not scale covariant due to fixed Gaussian filter standard deviation σ and the window size

- They are largely invariant to scaling of intensity, except for thresholding

The importance of scale

- The features of many objects in an image are only important over a certain spatial extent
- The image scale of the object is critical to recognizing the object
  - However, the scale of objects in an image is typically unknown



Miniature model from *True Lies* (1994)

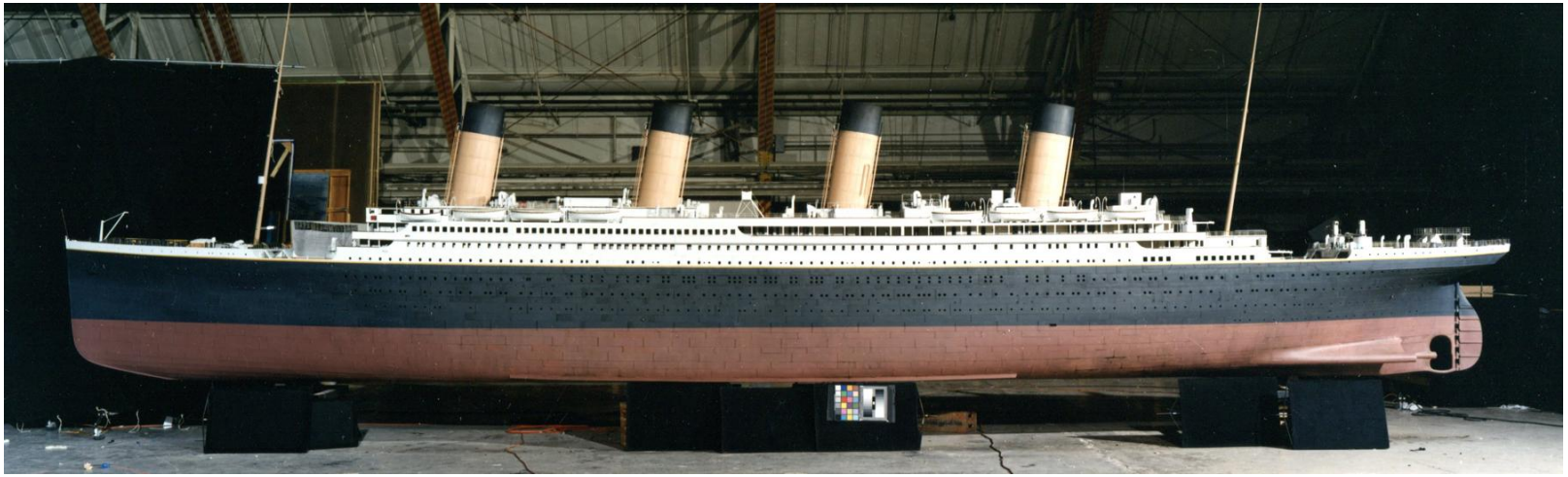
Smaller physical bridge, but more image pixels than image of real bridge on the right



Seven Mile Bridge, Florida Keys

The importance of scale

- The features of many objects in an image are only important over a certain spatial extent
- The image scale of the object is critical to recognizing the object
  - However, the scale of objects in an image is typically unknown



Miniature model from *Titanic* (1997)

The importance of scale

- The features of many objects in an image are only important over a certain spatial extent
- The image scale of the object is critical to recognizing the object
  - However, the scale of objects in an image is typically unknown

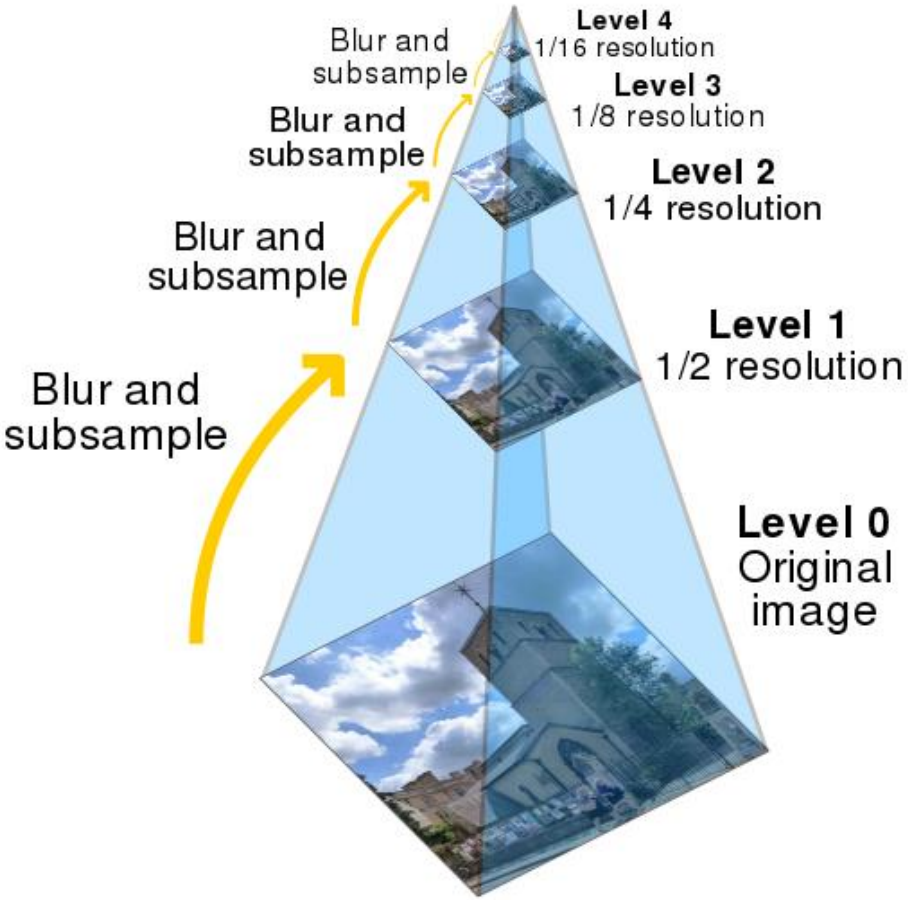
- So, how do we get the image of the object to be at (or near) the expected image scale?
  - Using multiscale image representations

Multiscale image representations

- Gaussian image pyramid
- Scale space

Gaussian pyramid

- Earliest (early 1980s) multiscale image representation
- Different levels of pyramid approximate the original image at different scales
- Width and height of next level is width and height of current level divided by rate
  - Typical rate is 2



Gaussian pyramid



Level 0 is original image



Collectively, remaining levels are 1/3 size of original image

Gaussian pyramid implementation

- Binomial filter kernel approximates kernel of Gaussian with variance = 1

Pascal's triangle

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1

(divided by 16)

- Filter (convolve) each level with this binomial kernel, then sample resulting every other pixel in every other row to produce next pyramid level
  - Convolution of two Gaussians is a Gaussian
  - Convolution is associative
    - Convolving Gaussian with image, subsampling, then convolving with result is equivalent to convolving two Gaussians (though with different variances), then convolving result with image

Scale space

- Pyramid representation is a predecessor to scale space representation
- Scale space theory is a formal theory for image structures at different scales
  - Image is represented by a one-parameter *family* of Gaussian low pass filtered images
  - A Gaussian filter meets all scale space axioms
    - Linearity, shift invariance, semi-group structure, non-creation of local extrema (zero-crossings), non-enhancement of local extrema, rotational symmetry, and scale invariance
- The *scale parameter* is the variance of the Gaussian filter
  - Note: use border mirror padding on input image when applying filter
- Image details significantly smaller than (two times) the standard deviation (square root of variance) are removed from the image at that scale parameter



Scale space vs Gaussian pyramid

1024 x 768



Standard deviation = 0

1024 x 768



Level 0

Scale space vs Gaussian pyramid

1024 x 768



Standard deviation = 1

512 x 384



Level 1

Scale space vs Gaussian pyramid

1024 x 768



Standard deviation = 2

256 x 192



Level 2

Scale space vs Gaussian pyramid

1024 x 768



Standard deviation = 4

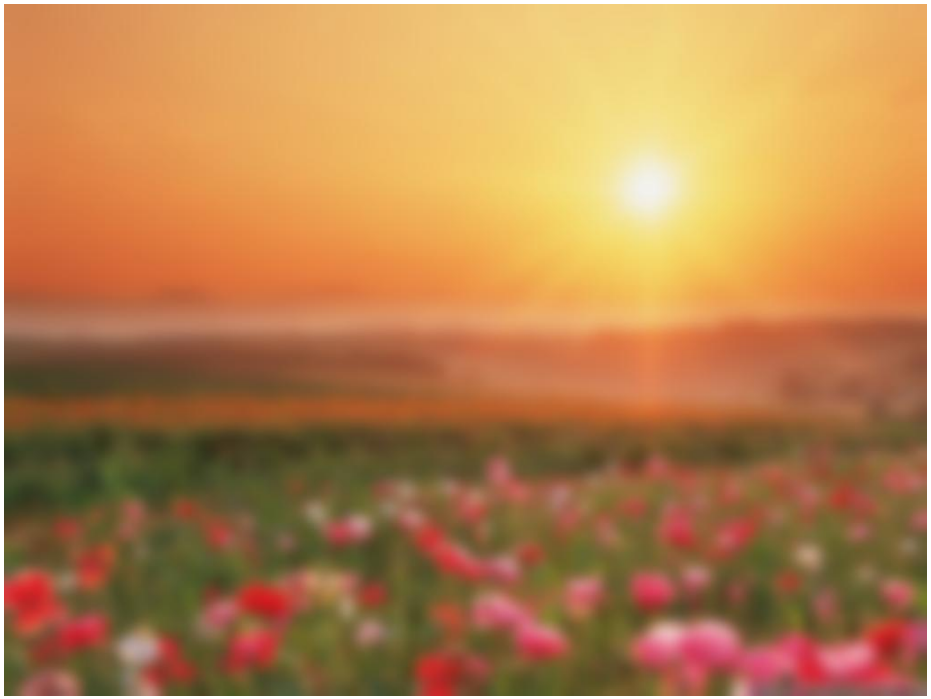
128 x 96



Level 3

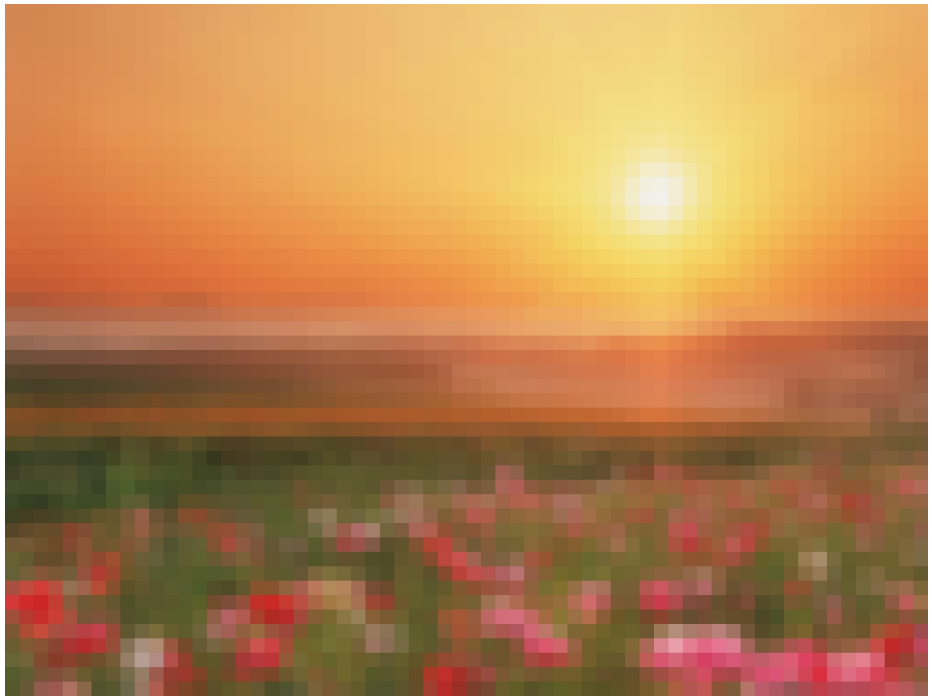
Scale space vs Gaussian pyramid

1024 x 768



Standard deviation = 8

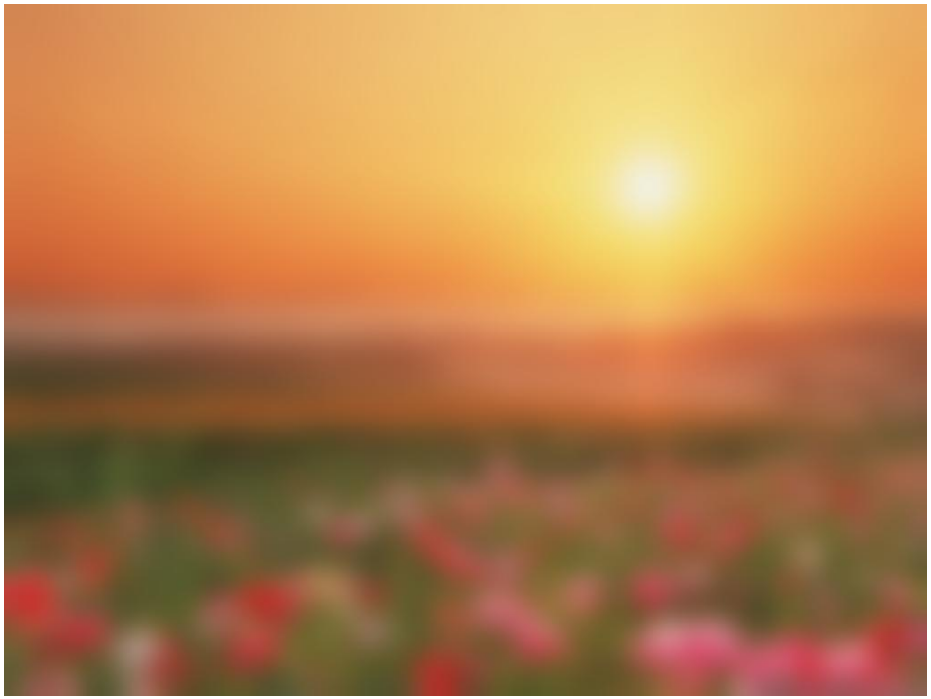
64 x 48



Level 4

Scale space vs Gaussian pyramid

1024 x 768



Standard deviation = 16

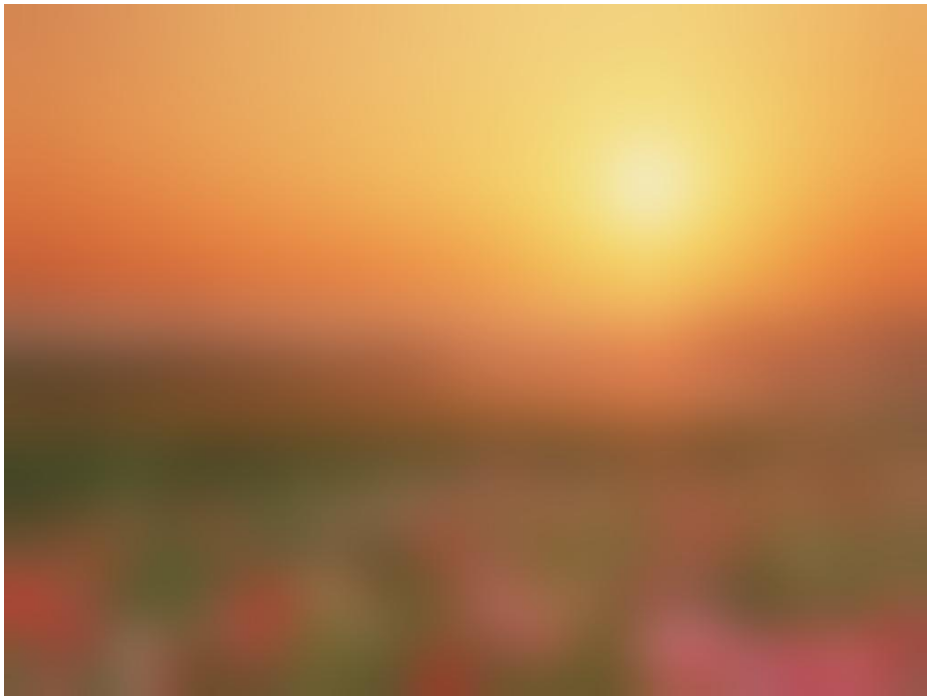
32 x 24



Level 5

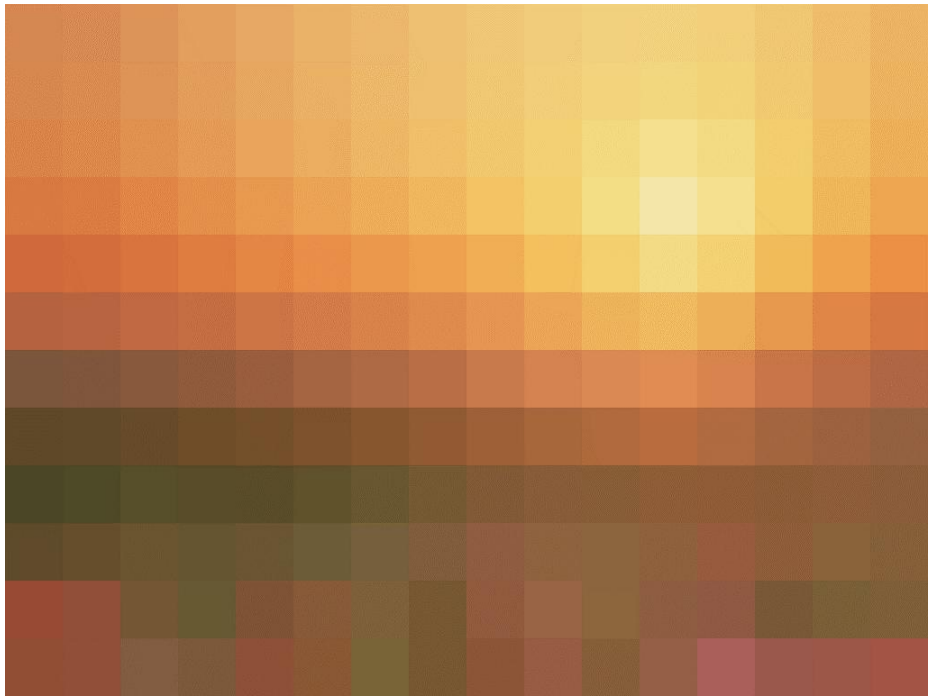
Scale space vs Gaussian pyramid

1024 x 768



Standard deviation = 32

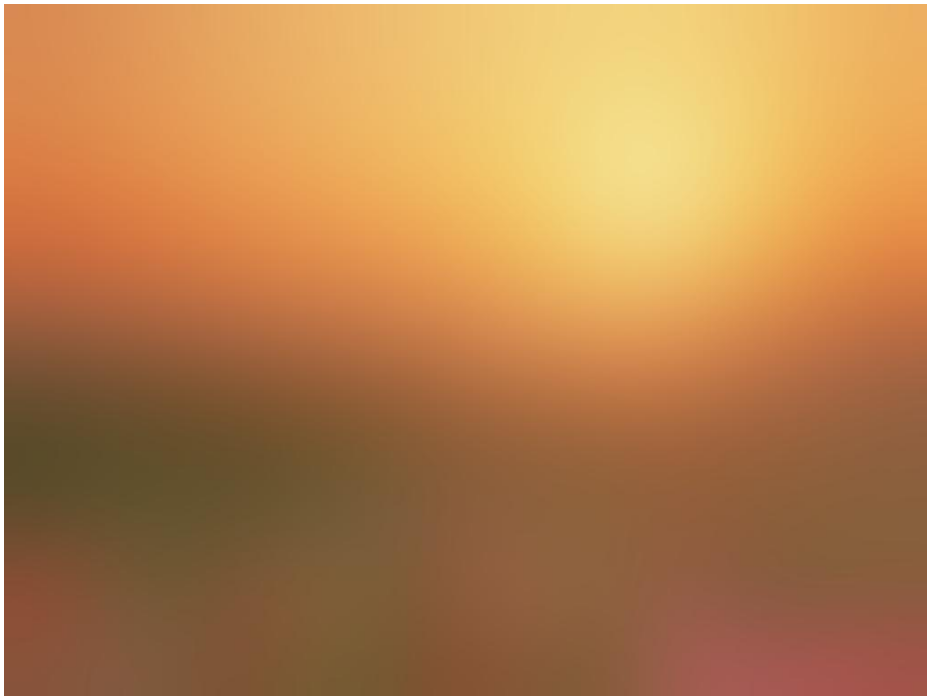
16 x 12



Level 6

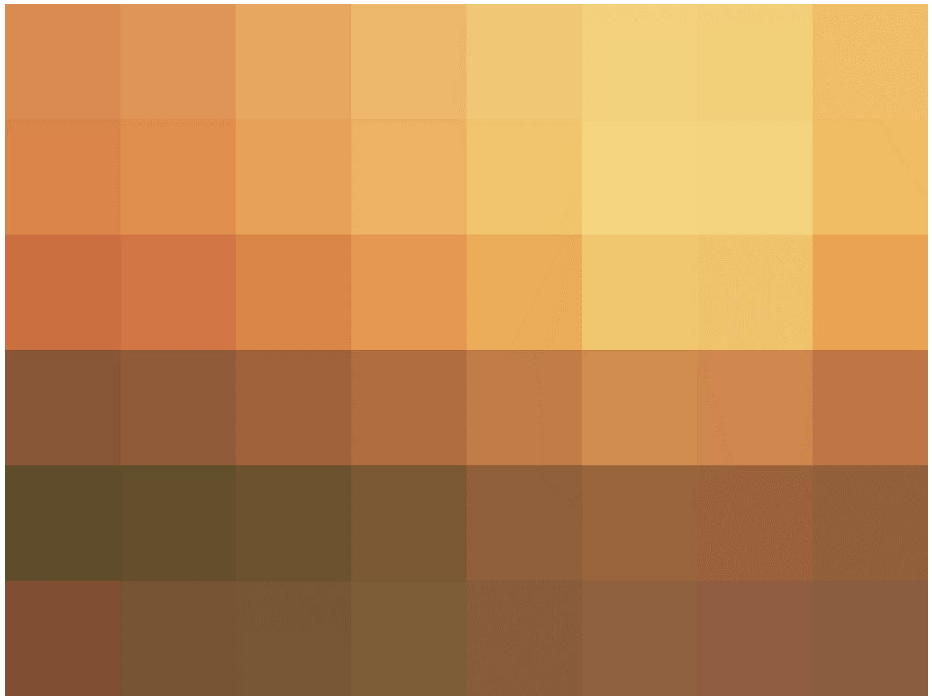
Scale space vs Gaussian pyramid

1024 x 768



Standard deviation = 64

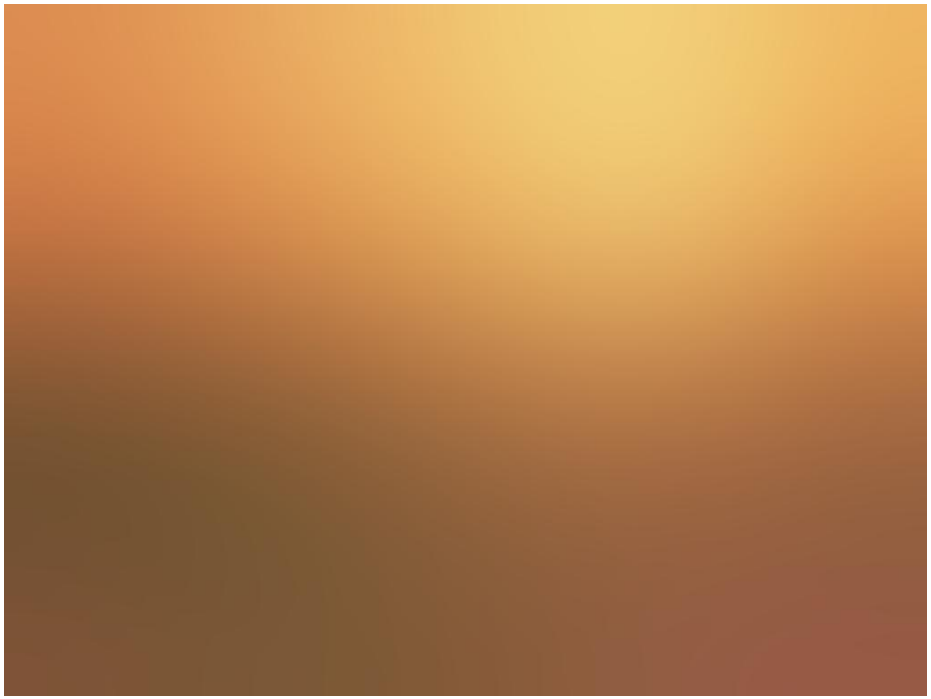
8 x 6



Level 7

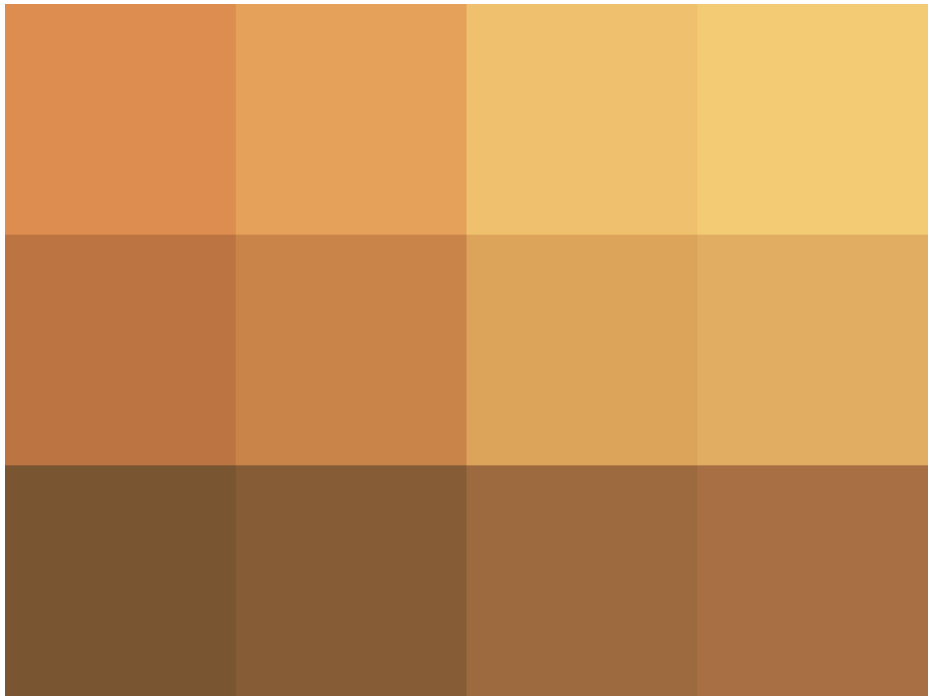
Scale space vs Gaussian pyramid

1024 x 768



Standard deviation = 128

4 x 3



Level 8

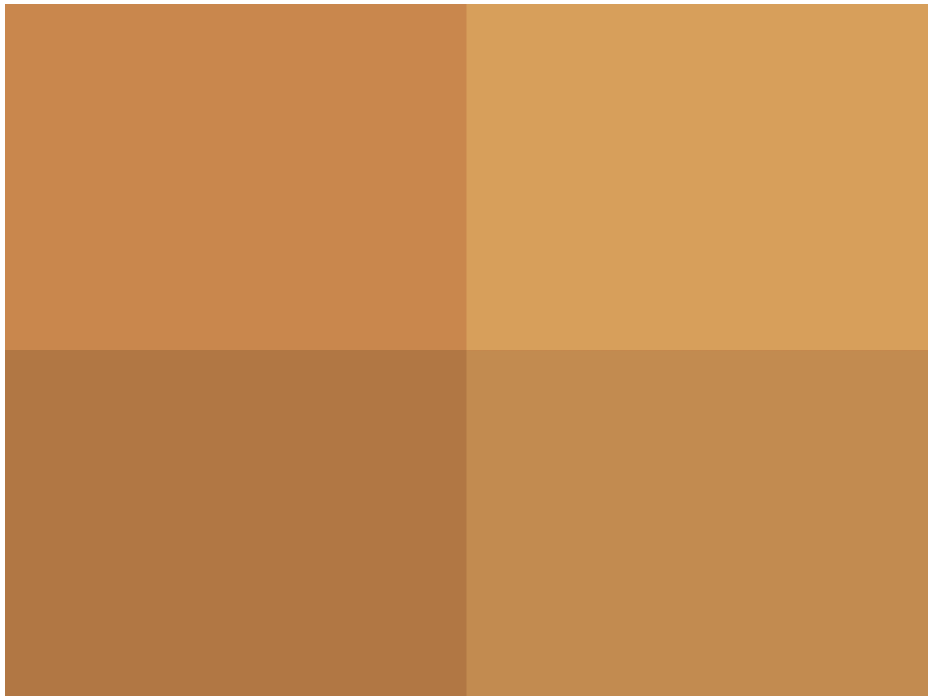
Scale space vs Gaussian pyramid

1024 x 768



Standard deviation = 256

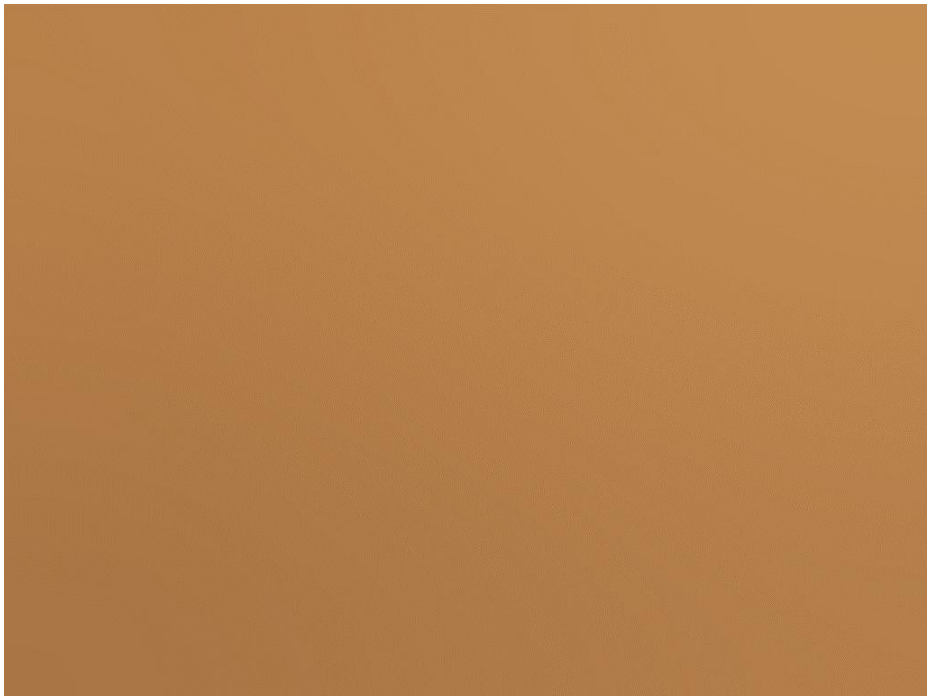
2 x 2



Level 9

Scale space vs Gaussian pyramid

1024 x 768



Standard deviation = 512

1 x 1



Level 10

Scale space vs Gaussian pyramid

- Note that Gaussian pyramid (of rate 2) levels corresponds to Gaussian kernel with standard deviations of 0, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512
  - Rate of Gaussian pyramid dictates standard deviations of levels
  - Scale space does not have this imposition; all standard deviations available
- In scale space, since all images are the same size, features are precisely located in the coordinates of the original image
  - Points in levels of Gaussian pyramid must be scaled up to coordinates of the original image, which is imprecise

Computing scale space

- For example, a Gaussian kernel with standard deviation of 512 requires a 3072x3072 kernel!
- Compute efficiently by filtering in frequency domain, not spatial domain
- The *scale parameter* is the variance of the Gaussian filter
  - Use border **mirror** padding on input image when applying filter

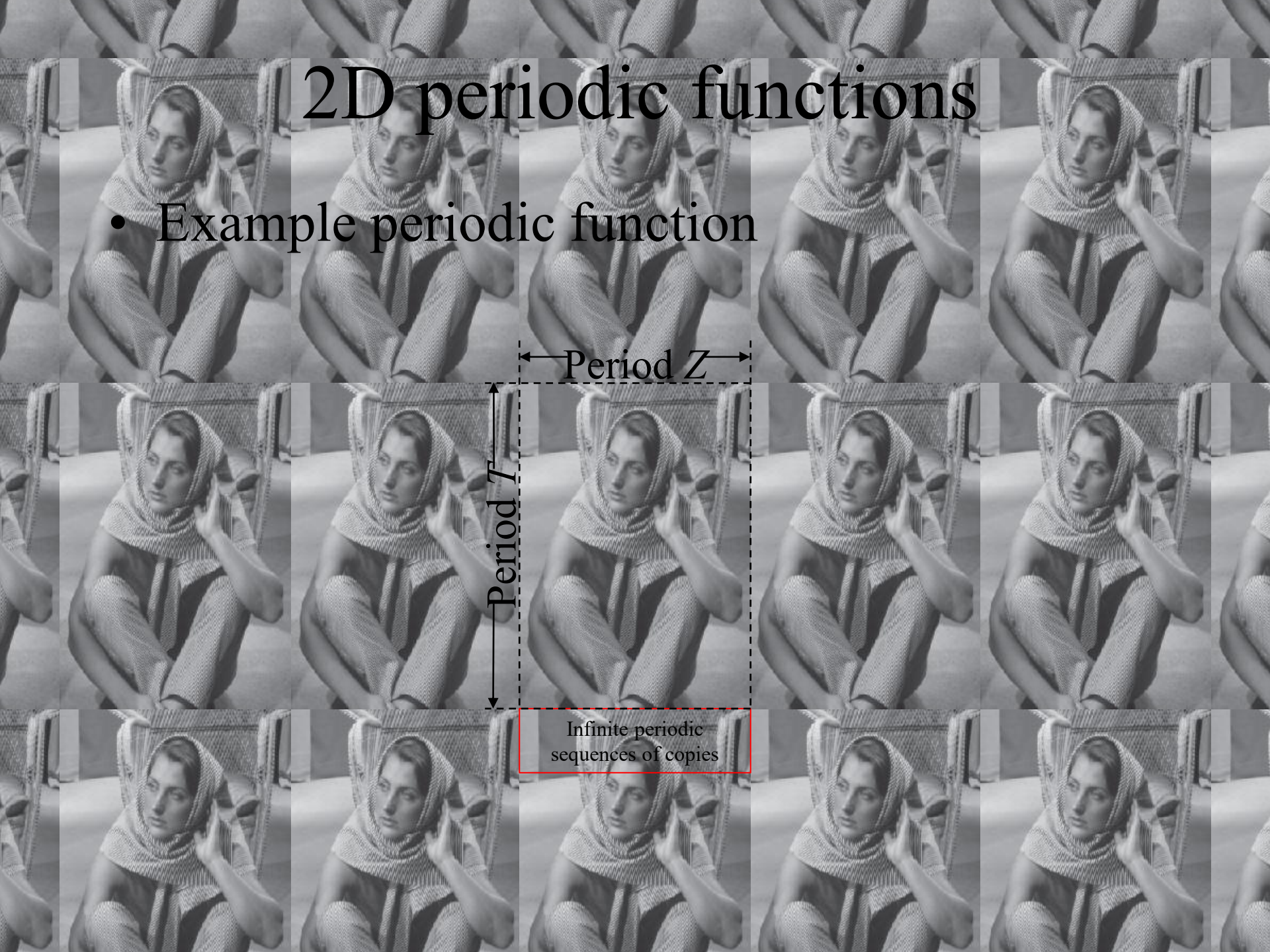
2D periodic functions

- Example periodic function



2D periodic functions

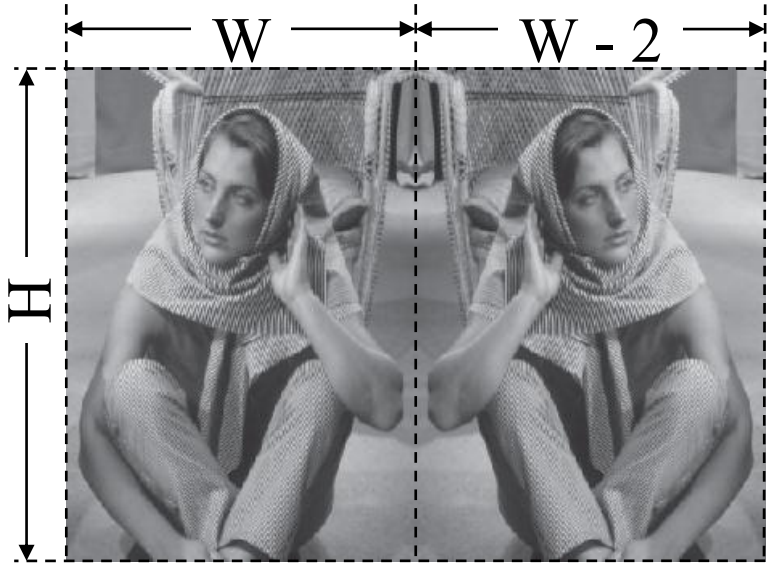
- Example periodic function



Scale space: use border **mirror** padding on input image when applying filter



Scale space: use border **mirror** padding on input image when applying filter



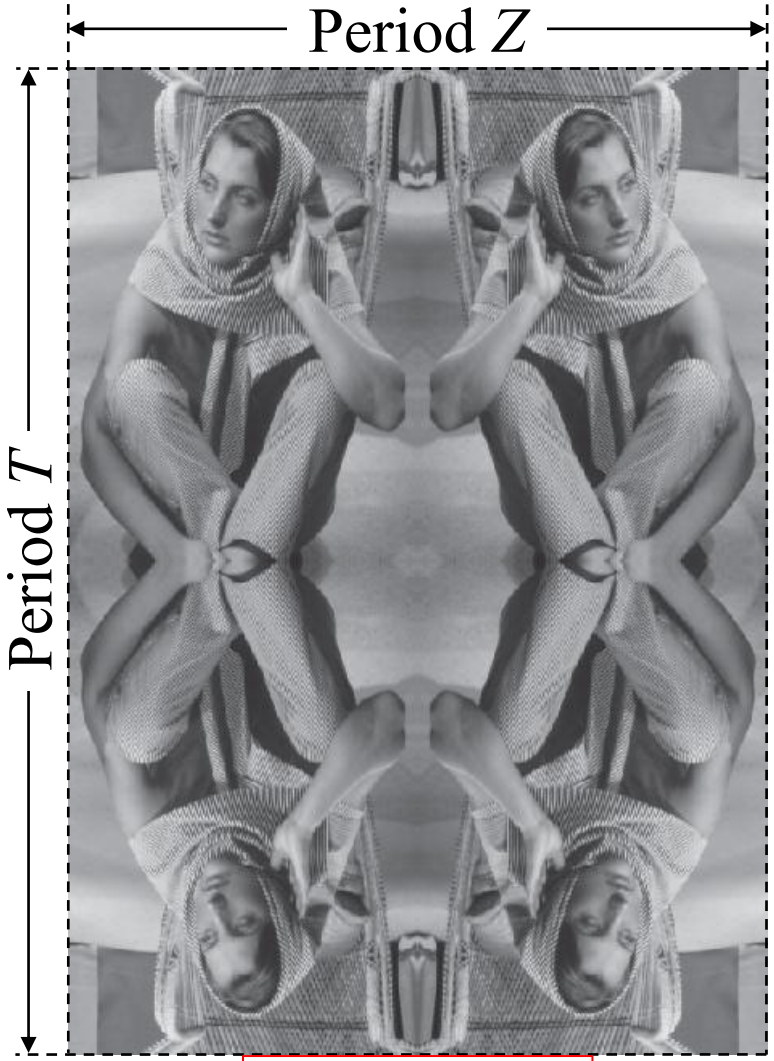
Mirror horizontally, **excluding** first and last *columns*

Scale space: use border **mirror** padding on input image when applying filter



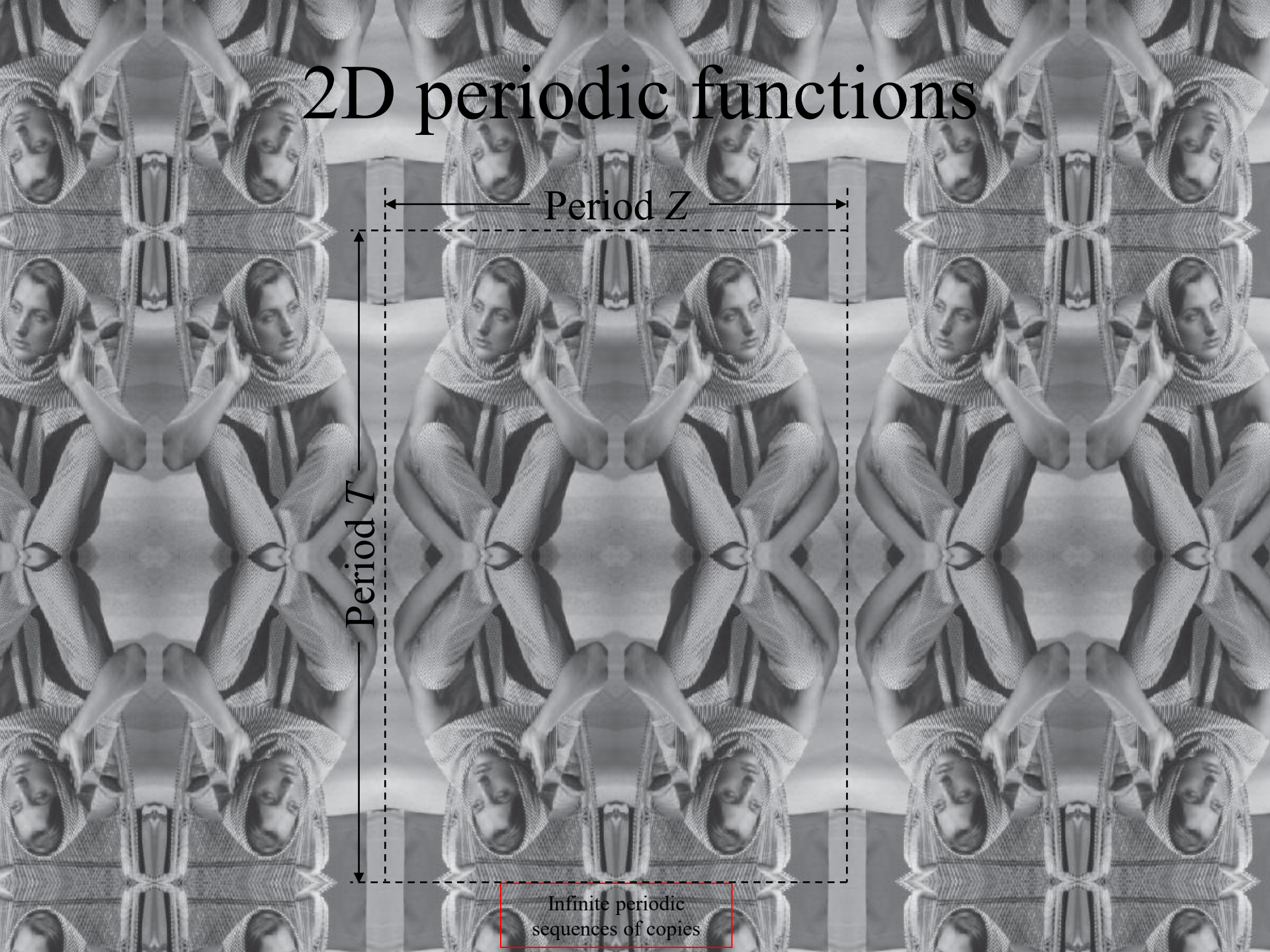
Mirror vertically, **excluding** first and last *rows*

2D periodic functions



Infinite periodic sequences of copies

2D periodic functions



Gaussian filter in frequency domain

Infinite periodic sequences of copies

$$\mathtt{H} = \mathbf{H}_v \mathbf{H}_u^\top$$

where

$$\mathbf{H}_u = (H(0), \ldots, H(\text{width} - 1)^\top$$

where

$$H(u) = e^{\sigma^2(\cos(2\pi u/\text{width})-1)} \text{ for } u = 0, \ldots, \lfloor \text{width}/2 \rfloor$$

then

$$H(u) = H(\text{width} - u) \text{ for } u = \lfloor \text{width}/2 \rfloor + 1, \ldots, \text{width} - 1$$

and where

$$\mathbf{H}_v = (H(0), \ldots, H(\text{height} - 1)^\top$$

where

$$H(v) = e^{\sigma^2(\cos(2\pi v/\text{height})-1)} \text{ for } v = 0, \ldots, \lfloor \text{height}/2 \rfloor$$

then

$$H(v) = H(\text{height} - v) \text{ for } v = \lfloor \text{height}/2 \rfloor + 1, \ldots, \text{height} - 1$$

Computing scale space

- Mirror pad image and forward Fourier transform

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M+vy/N)}$$

- For each scale parameter (variance of the Gaussian filter)
  - Design and apply Gaussian filter *H* in frequency domain

$$G(u,v) = F(u,v)H(u,v)$$

  - Inverse Fourier transform result

$$g(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} G(u,v) e^{j2\pi(ux/M+vy/N)}$$

  - Remove mirror padding

Use this Fourier transform pair

Multiscale image representations

- The features of many objects in an image are only important over a certain spatial extent
- The scale of objects in an image is typically unknown
- Use multiscale image representations to represent the image of the object at (or near) the expected image scale
  - Gaussian pyramid
  - Scale space

Scale invariant feature transform (SIFT)

- SIFT features are called *keypoints*
- Keypoints are invariant to
  - Scale
  - Rotation
- Keypoints are robust to
  - Changes in viewpoint
  - Changes in illumination
  - Noise

Scale invariant feature transform (SIFT)

- SIFT feature descriptors are *n*-dimensional feature vectors
  - Elements are invariant feature descriptors

Scale invariant feature transform (SIFT)

- Steps
  1. Construct the scale space
  2. Obtain the initial keypoints
  3. Improve the accuracy of the location of the keypoints
  4. Delete unsuitable keypoints
  5. Compute keypoint orientations
  6. Compute keypoint descriptors

SIFT, construct the scale space

- Search for stable features across all possible scales
  - Use a function of scale known as *scale space*
- Achieves scale invariance

SIFT, construct the scale space

- Each octave corresponds to doubling the standard deviation
  - First image at each new octave is downsampled third image (*octave image*) from previous octave

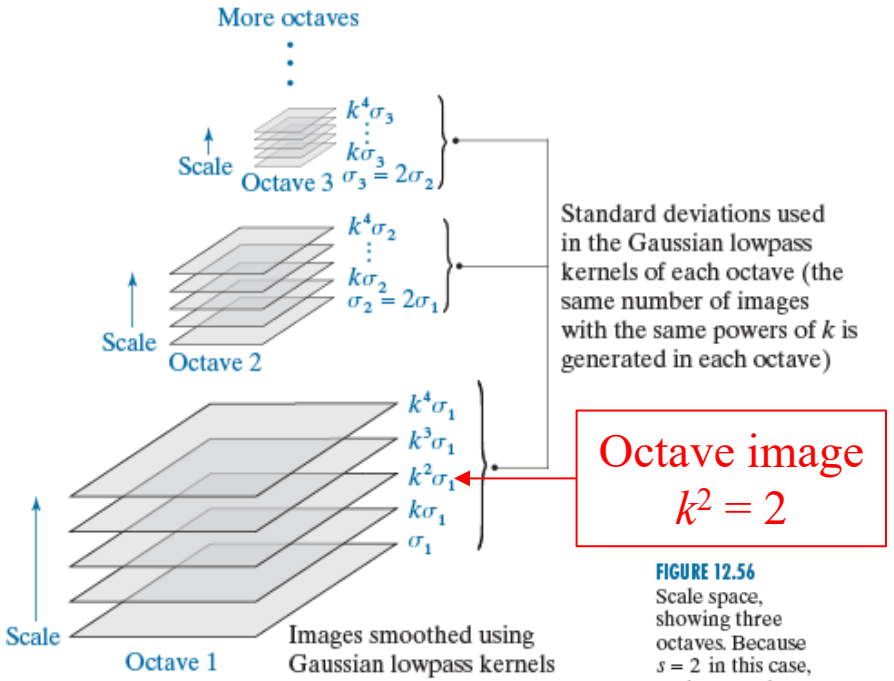


FIGURE 12.56 Scale space, showing three octaves. Because $s = 2$ in this case, each octave has five smoothed images. A Gaussian kernel was used for smoothing, so the space parameter is σ.

SIFT, construct the scale space

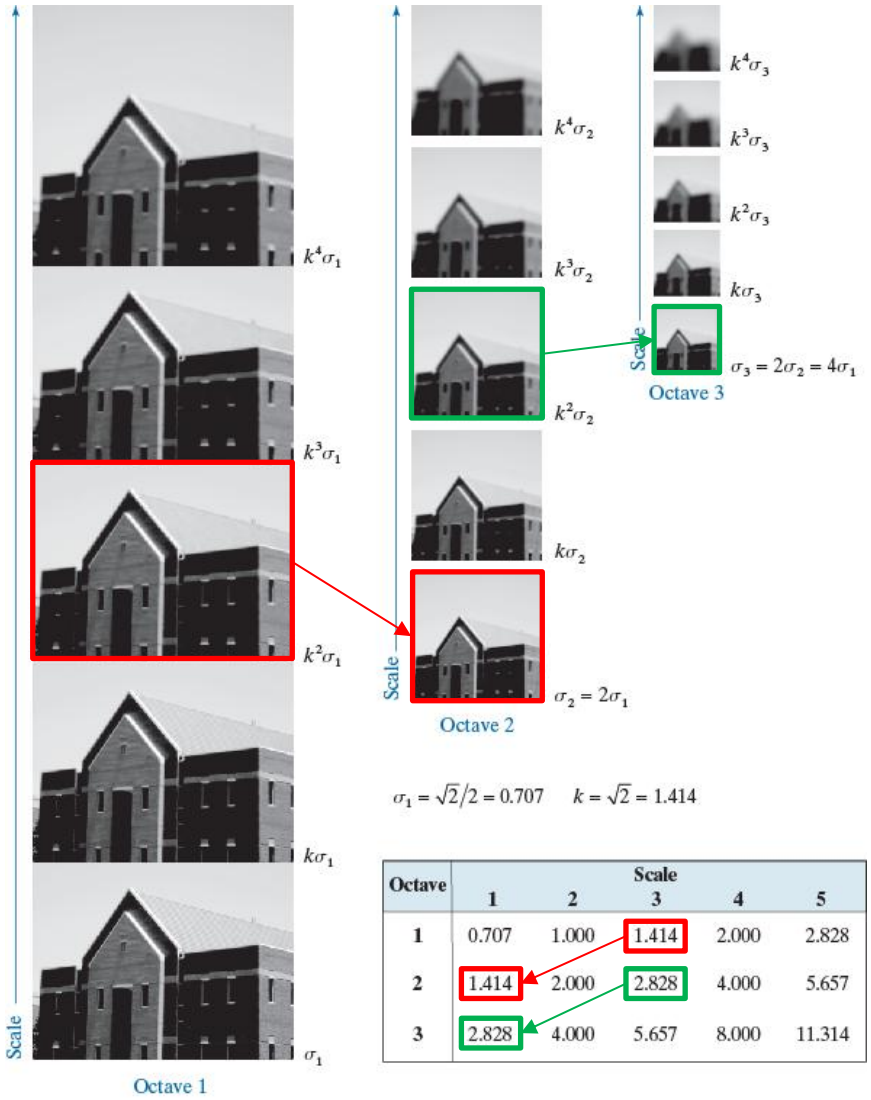


FIGURE 12.57 Illustration using images of the first three octaves of scale space in SIFT. The entries in the table are values of standard deviation used at each scale of each octave. For example the standard deviation used in scale 2 of octave 1 is $k\sigma_1$, which is equal to 1.0. (The images of octave 1 are shown slightly overlapped to fit in the figure space.)

$\sigma_1 = \sqrt{2}/2 = 0.707 \quad k = \sqrt{2} = 1.414$

Octave	Scale 1	2	3	4	5
1	0.707	1.000	1.414	2.000	2.828
2	1.414	2.000	2.828	4.000	5.657
3	2.828	4.000	5.657	8.000	11.314

SIFT, obtain the initial keypoints

- First, difference two adjacent scale-space images in an octave

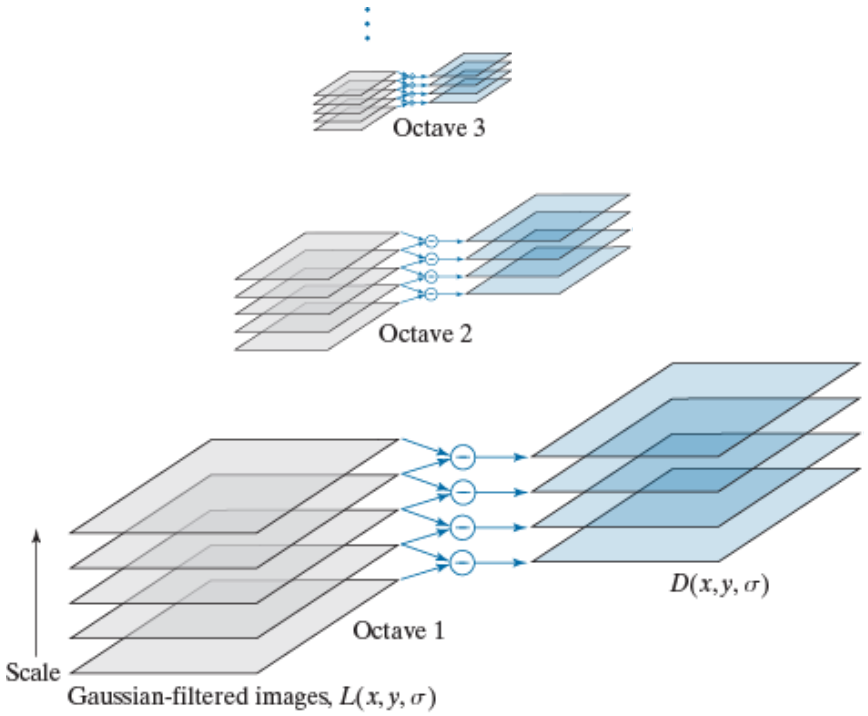


FIGURE 12.58 How Eq. (12-69) is implemented in scale space. There are $s+3$ $L(x, y, \sigma)$ images and $s+2$ corresponding $D(x, y, \sigma)$ images in each octave.

SIFT, obtain the initial keypoints

- Second, detect extrema in the differences

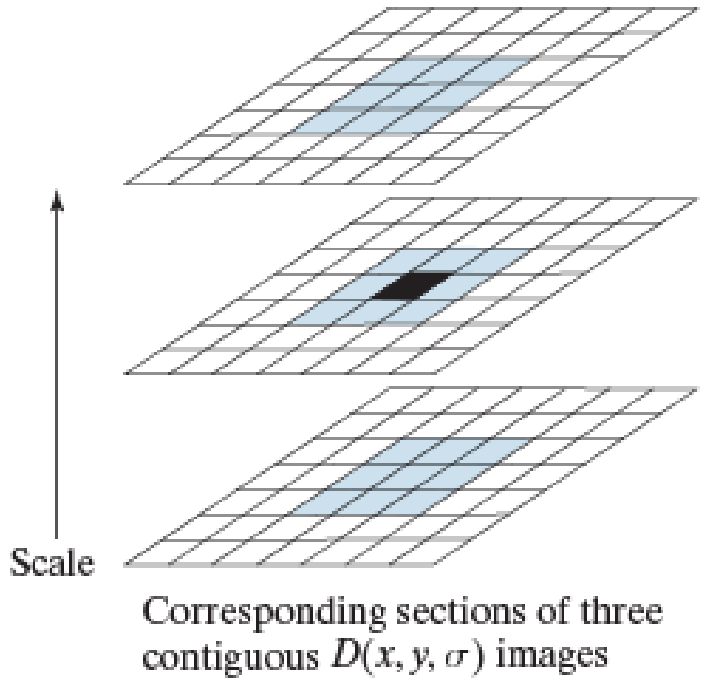


FIGURE 12.59 Extrema (maxima or minima) of the $D(x, y, \sigma)$ images in an octave are detected by comparing a pixel (shown in black) to its 26 neighbors (shown shaded) in 3×3 regions at the current and adjacent scale images.

The point (in black) is selected as an extremum point if its value is **larger** than the values of **all** its neighbors (in blue) *or* **smaller** than the values of **all** its neighbors (in blue)

SIFT, obtain the initial keypoints

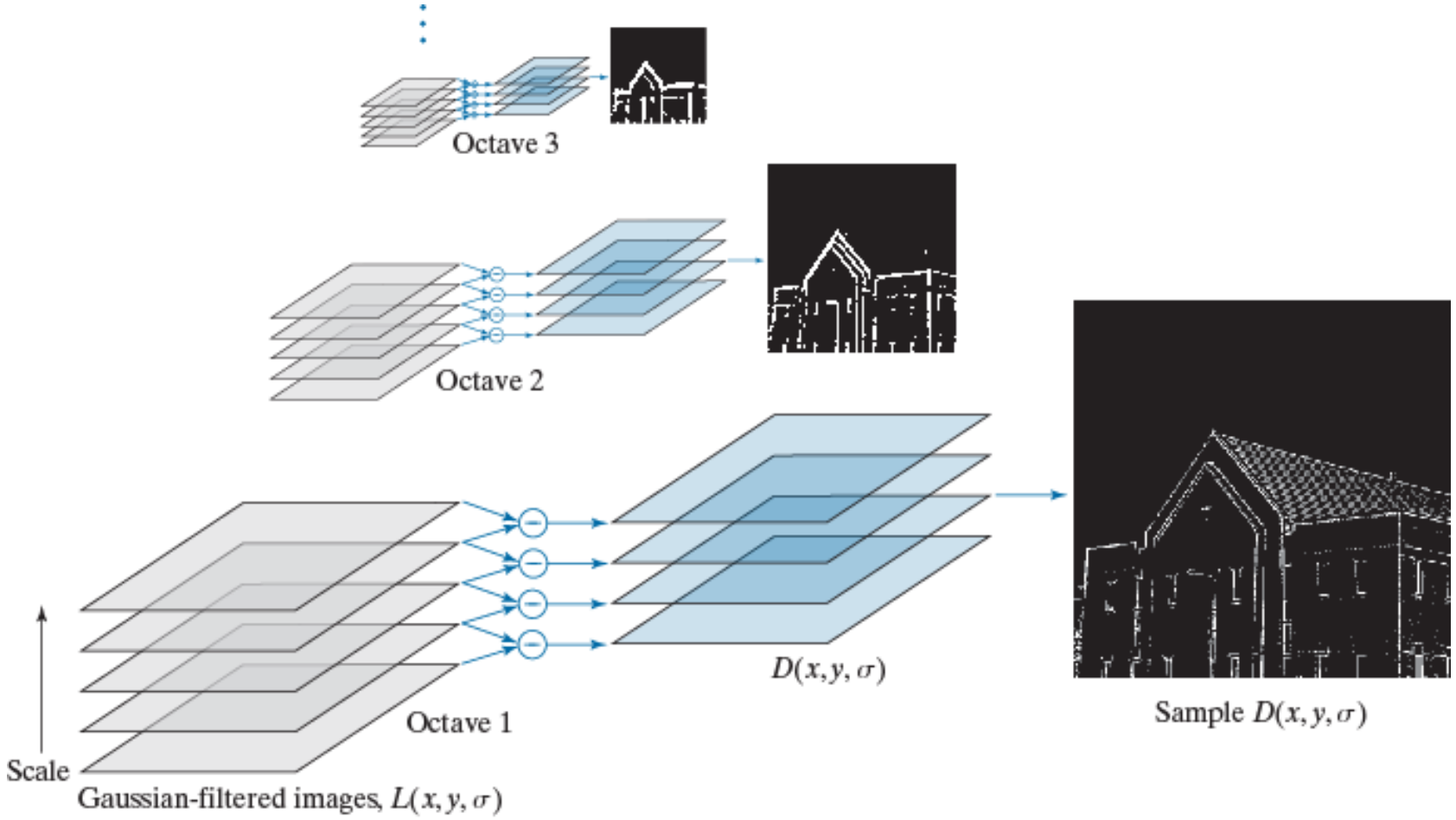


FIGURE 12.58 How Eq. (12-69) is implemented in scale space. There are $s+3$ $L(x, y, \sigma)$ images and $s+2$ corresponding $D(x, y, \sigma)$ images in each octave.

SIFT, improve the accuracy of the location of the keypoints

- Interpolate the values of the difference images about extrema
- Determine subpixel coordinates of extrema using interpolated values

SIFT, delete unsuitable keypoints

- Determine difference at subpixel keypoints
- Eliminate keypoints with low contrast and/or are poorly localized
- Additionally, delete keypoints associated with edges
  - Only keep corner-like features
    - Equivalent to thresholding the minor eigenvalue

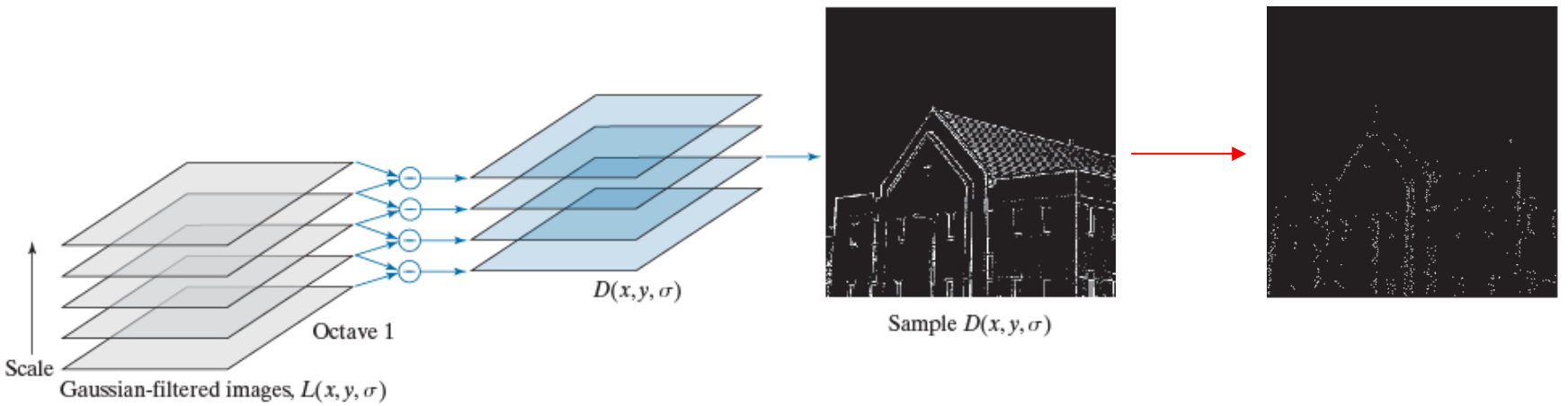


FIGURE 12.60 SIFT keypoints detected in the building image. The points were enlarged slightly to make them easier to see.

Scale invariant feature transform (SIFT)

- Steps
  1. Construct the scale space
  2. Obtain the initial keypoints
  3. Improve the accuracy of the location of the keypoints
  4. Delete unsuitable keypoints
  - So far, we have computed the location of each keypoint in scale space (i.e., location and scale of each keypoint)
    - Scale invariance
  - Next is rotation invariance
  5. Compute keypoint orientations
  6. Compute keypoint descriptors

SIFT, compute keypoint orientations

- For each keypoint
  - At its scale, compute the gradient magnitude and orientation of the keypoint
  - Form a **histogram of orientations**
    - 36 bins (10 degrees each)
    - Weight an orientation by its associated magnitude **and** a Gaussian, when adding it to an orientation bin
  - Initial orientation is largest bin
    - Create an additional keypoint for other bins within 80% the size of the largest bin
  - Improve orientation estimate using interpolation
    - Fit a parabola to values of the largest bin and its two neighboring bins

SIFT, compute keypoint orientations

- Keypoints
  - Location and scale (scale invariant)
  - Orientation (rotation invariant)
    - Length of arrow is histogram of orientations interpolated bin value
    - (Useful in matching keypoints across images)



FIGURE 12.60 SIFT keypoints detected in the building image. The points were enlarged slightly to make them easier to see.



FIGURE 12.61 The keypoints from Fig. 12.60 superimposed on the original image. The arrows indicate keypoint orientations.

Local regions

- The local region about each **oriented** keypoint is invariant to
  - Scale, orientation, illumination, and image viewpoint



FIGURE 12.61
The keypoints from Fig. 12.60 superimposed on the original image. The arrows indicate keypoint orientations.

SIFT, compute keypoint descriptors

- Feature descriptor
  - 16x16 region about keypoint
    - Gradient magnitude (Gaussian weighted) and direction at each point in region
    - Quantize gradient directions in each 4x4 subregion to 45 degree increments
      - Interpolate each of the 16 gradients directions to distribute it over all 8 bins (8 * 45 degrees = 360 degrees)
    - Concatenate the 16 8-directional histograms bins to form a 128-dimensional feature vector

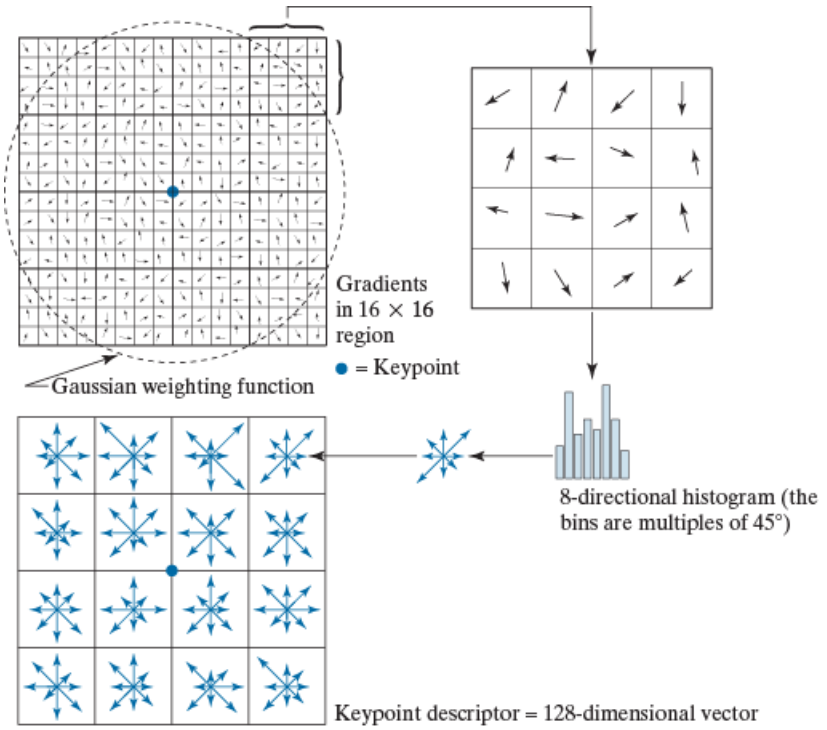
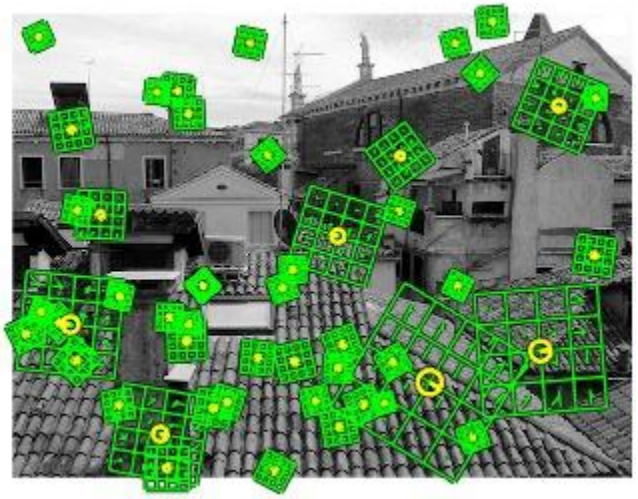


FIGURE 12.62
Approach used to compute a keypoint descriptor.

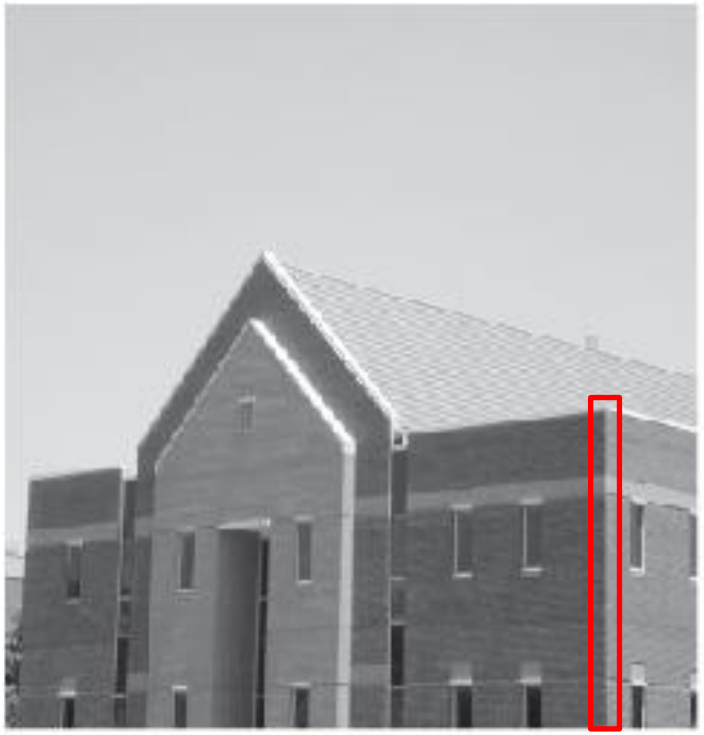
SIFT, compute keypoint descriptors

- Rotation invariance
  - Rotate the 8-directional histograms relative to the keypoint orientation
- Robustness to changes in illumination
  - Unitize the 128-dimensional feature vector
  - Threshold to reduce the influence of large gradient magnitudes
  - Unitize again



Matching keypoints across images

- First image is whole image
- Second image is darker version of red rectangle



Matching keypoints across images using SIFT features and feature descriptors



a b

FIGURE 12.63 (a) Keypoints and their directions (shown as gray arrows) for the building image and for a section of the right corner of the building. The subimage is a separate image and was processed as such. (b) Corresponding key points between the building and the subimage (the straight lines shown connect pairs of matching points). Only three of the 36 matches found are incorrect.

Next Lecture

- Structure from motion