Lecture 11

Features 2: SIFT

Wed. Oct. 14, 2020

Features

- edge

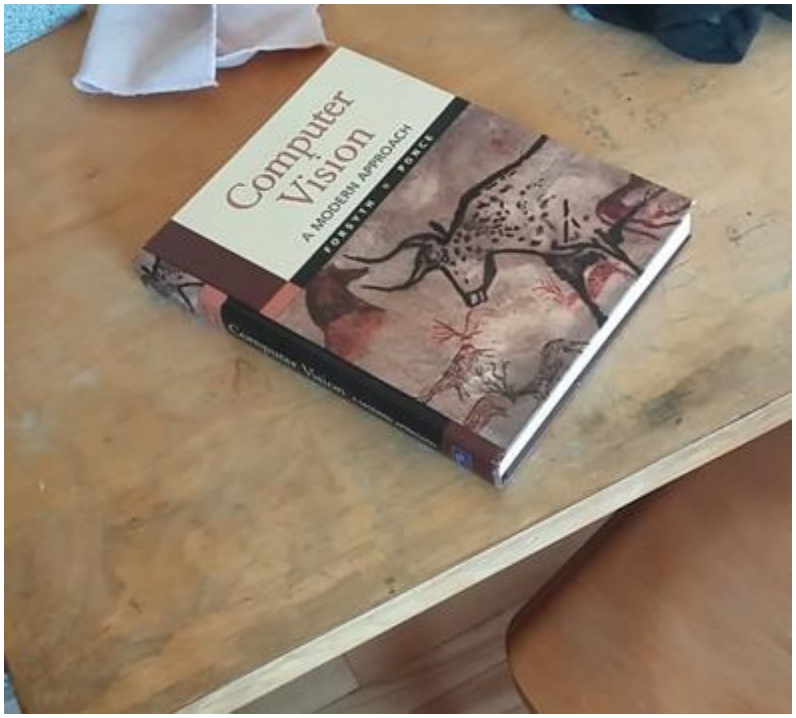
- “corner” -- locally distinctive point
  (local maximum of Harris-Stevens operator)

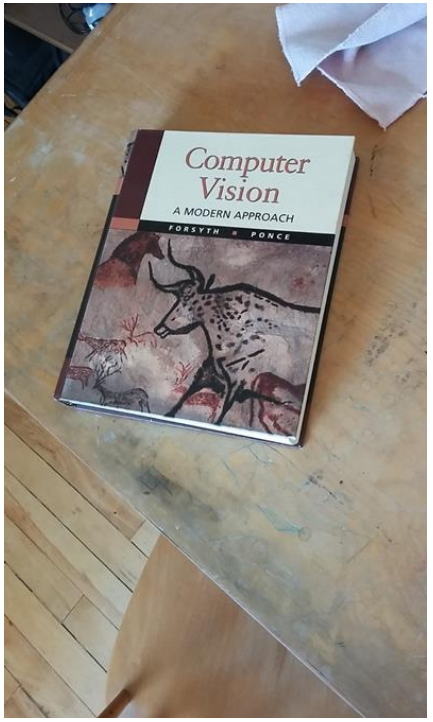
- RGB weighted histogram (last lecture)

Many other features have been proposed.
Today we look at a classic: “SIFT”.

Motivation: image matching

For example, here we have two images of the same scene, viewed from two different perspectives. We would like to *match local neighborhoods* in the two images. We could then use these matches to decide if there are *matching objects*.





We would like to extract features in each image, which describe a local neighborhood.

The features may have several properties:

- position
- scale
- orientation
- image intensities

SIFT [Lowe 1999, 2004]

SIFT stands for "*scale invariant feature transform*"

A SIFT feature consists of

- position (x, y)
- scale σ
- orientation θ
- image gradient descriptor

Let's address position and scale first.

SIFT step 1: position & scale

Recall from lecture 9: a “box” will produce a local minimum (bright box) or a local maximum (dark box) in a Laplacian of a Gaussian scale space.

These local maxima or minima will define the position and scale of SIFT ‘keypoints’.

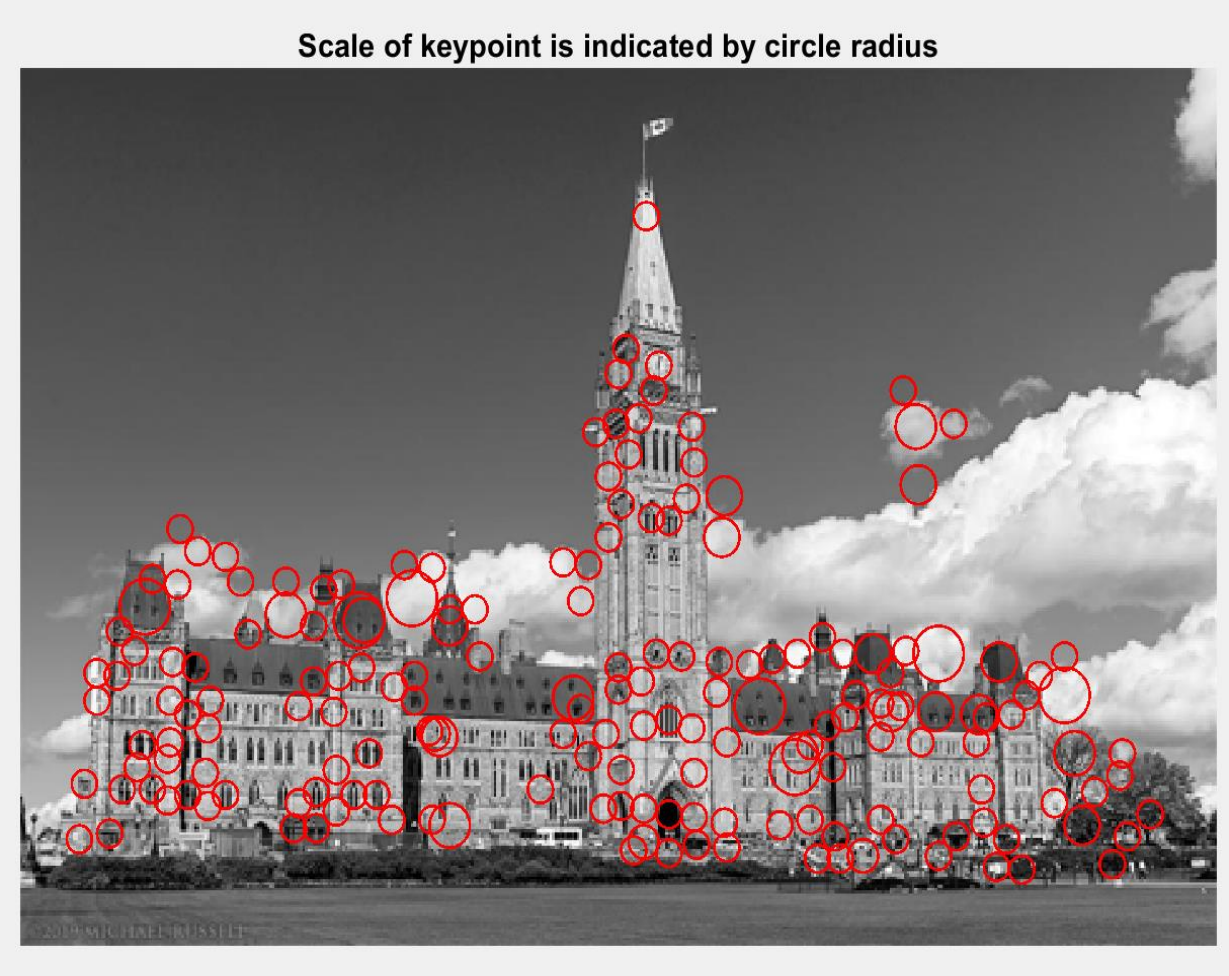
Most maxima and minima detected in a Laplacian of a Gaussian real images are not boxes. This is analogous to “corner detection”, where the actual goal was to find locally distinctive points.

With SIFT, the goal is to find points that produce minima or maxima in scale space (“scale invariant feature transform”).

Example

Canadian Parliament Buildings in Ottawa.

Example



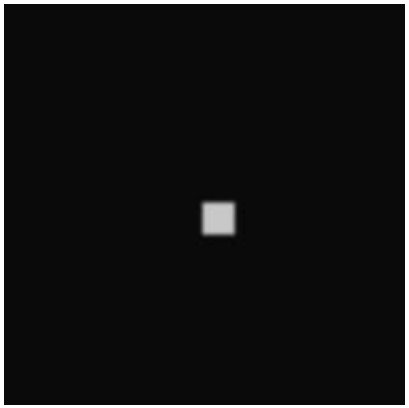
Admittedly it is not obvious why the maxima and minima occur at these particular positions and scales!

For the next several slides, we will work with the example of a box image.

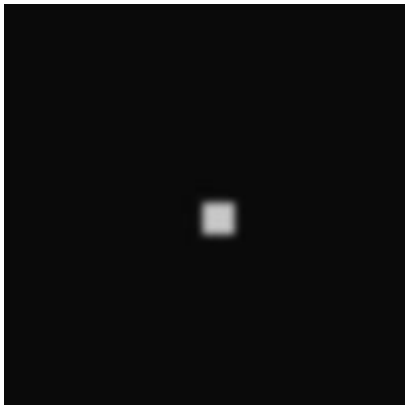
As we will see, even a box can have some subtle behaviors!

Recall lecture 9: Gaussian Scale Space

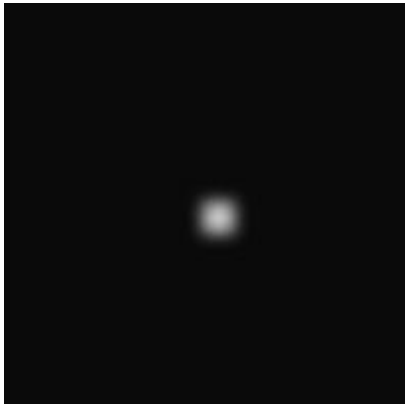
$$I\left(x, y, \sigma = 2^k\right) = I(x, y) * G(x, y; \sigma = 2^k)$$



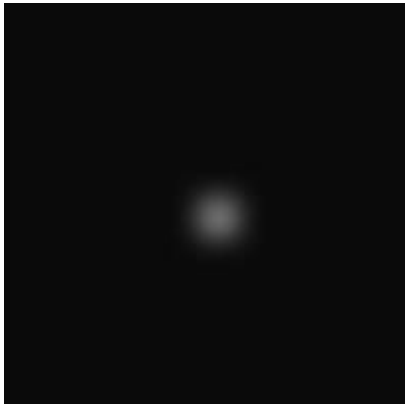
$\sigma = 1$



$\sigma = 2$

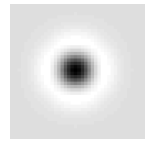


$\sigma = 4$



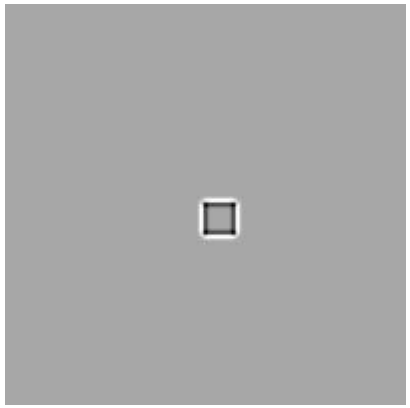
$\sigma = 8$

Example of a “box”.

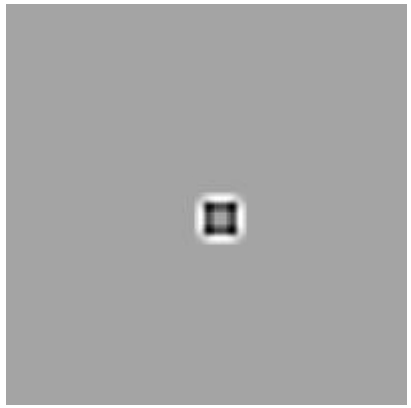


Recall lecture 9: Normalized Laplacian of Gaussian Scale Space

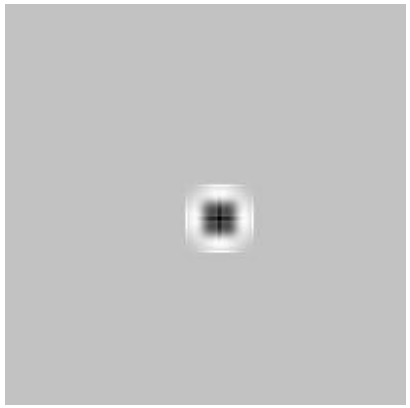
$$I(x, y, \sigma = 2^k) = I(x, y) * \sigma^2 \nabla^2 G(x, y; \sigma = 2^k)$$



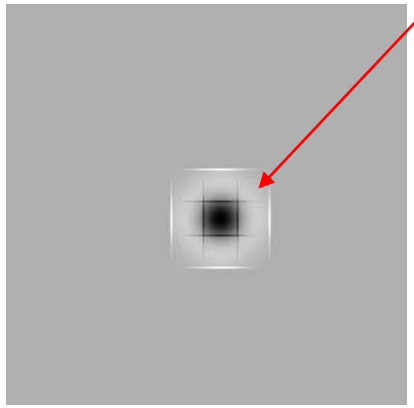
$\sigma = 1$



$\sigma = 2$



$\sigma = 4$



$\sigma = 8$

The response to the bright box yields a minimum in scale space (x, σ) with x at the center of the box and scale σ that is the half width of the box.

SIFT does not use a Laplacian of a Gaussian scale space.

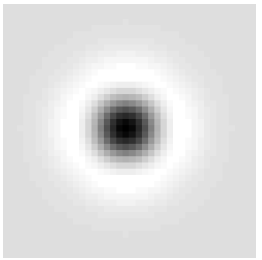
Instead it uses something almost equivalent…

I mention this mainly because if you read further about SIFT, then you will surely need to know about this detail.

“Difference of Gaussians” (DOG)

Normalized Laplacian of a Gaussian:

$$\sigma^2 \ \nabla^2 \ G(x, y; \sigma)$$



Difference of Gaussians:

$$G(x, y; \sigma_1) - G(x, y; \sigma_2)$$

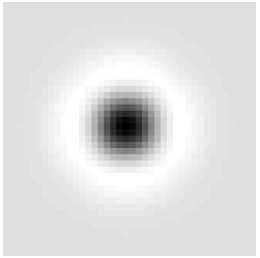
It has long been known (Marr and Hildreth ‘79) that the two have approximately the same shape when $\sigma_1 \approx \sigma_2$ (and $\sigma_1 \neq \sigma_2$).

“Difference of Gaussians” (DOG)

One can show (see lecture notes) using basic Calculus:

if κ is a real number slightly greater than 1, then

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



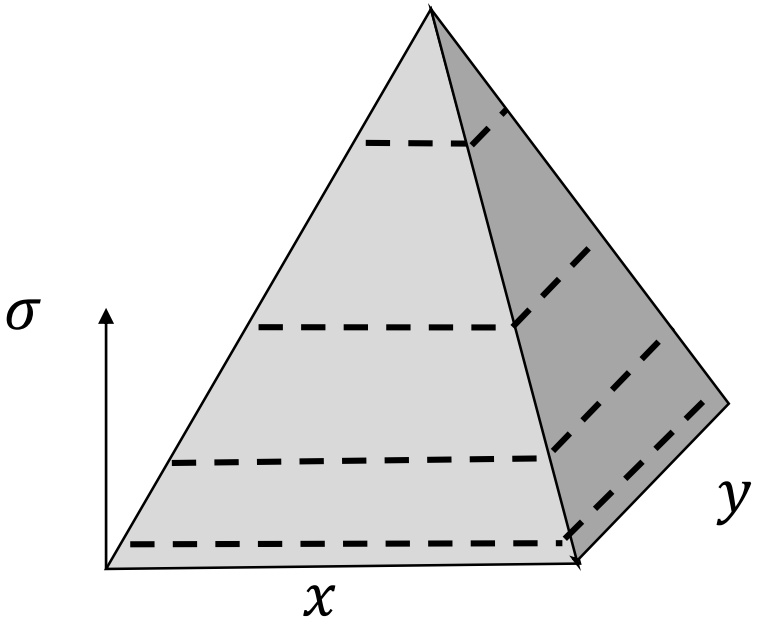
To find the position (x, y) and scale σ of a keypoint, SIFT uses a Gaussian pyramid.

Let's review…

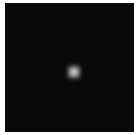
Recall lecture 9: Gaussian Pyramid

Create image at layer k by *subsampling* the blurred version of the image.

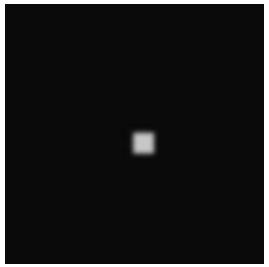
(I discussed two ways to do it.)



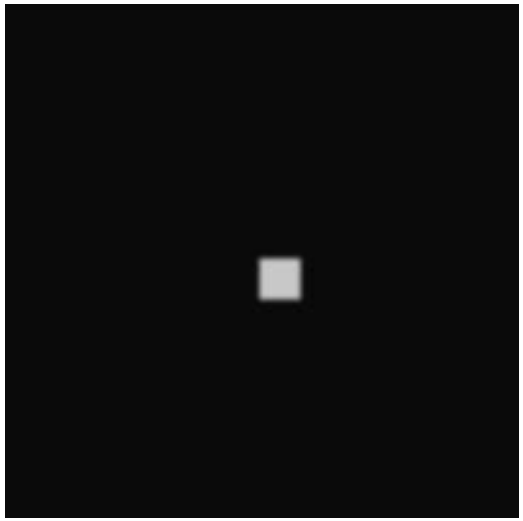
$\sigma = 8$



$\sigma = 4$

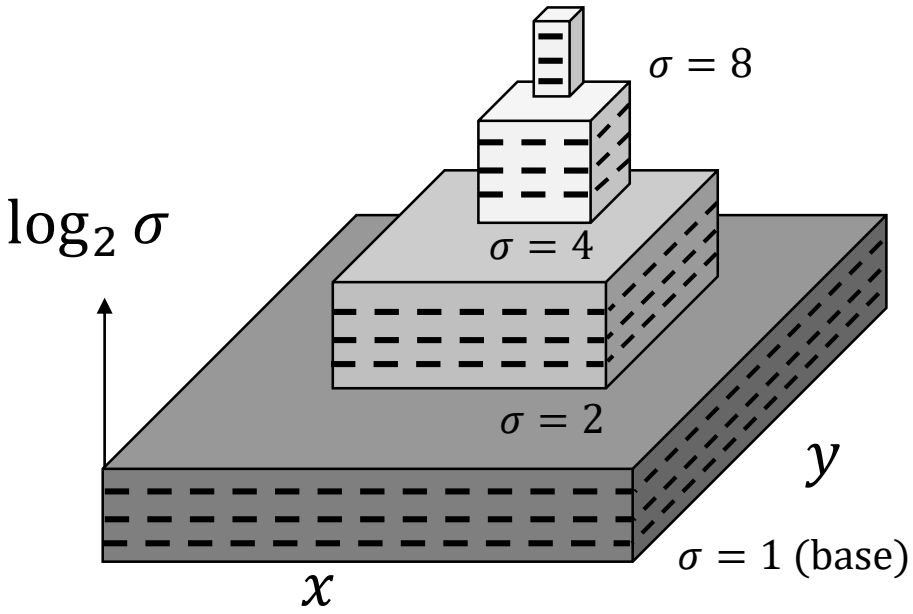


$\sigma = 2$



$\sigma = 1$

Sampling the Gaussian Pyramid



As on the previous slide, we subsample with a stride of 2 each time σ doubles.

(A doubling of size is called an *octave*.)

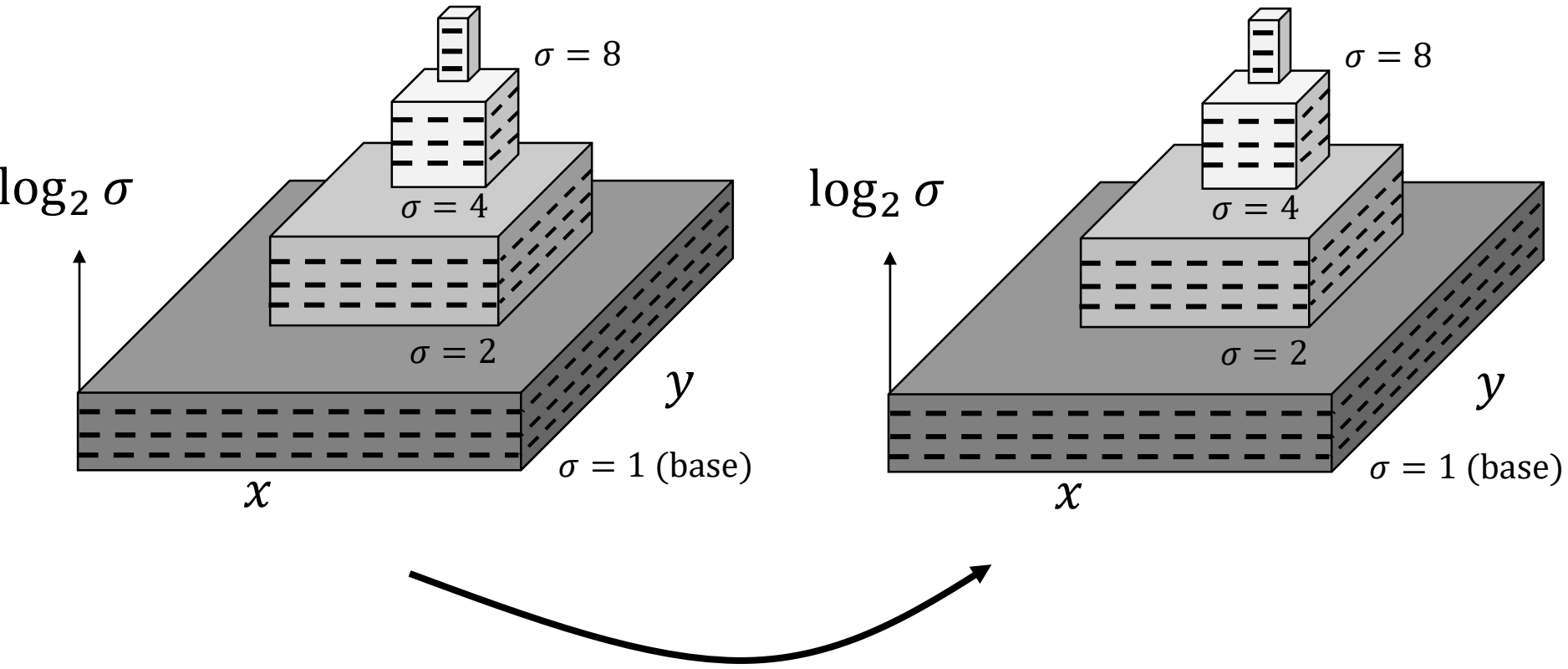
However, SIFT also samples m scales within each octave. (Here $m = 4$.)

It use the same size image and sampling rate within each octave.

$$\sigma = 1, 2^{\frac{1}{m}}, 2^{\frac{2}{m}}, \ldots, 2^{\frac{m-1}{m}}, 2, 2^{1+\frac{1}{m}}, \ldots, 4, \ldots, 8, etc$$

Sampled
Gaussian pyramid

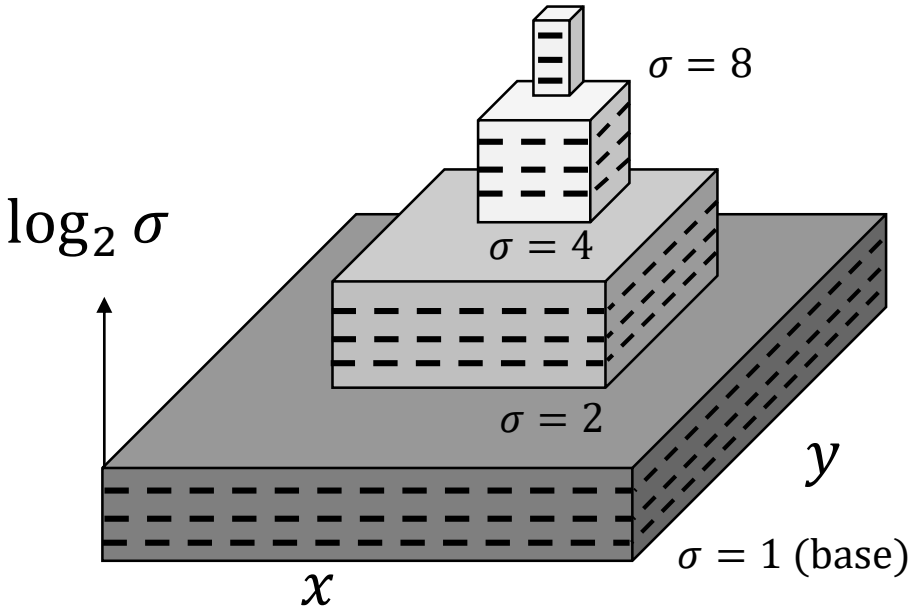
Sampled
Difference of Gaussian pyramid

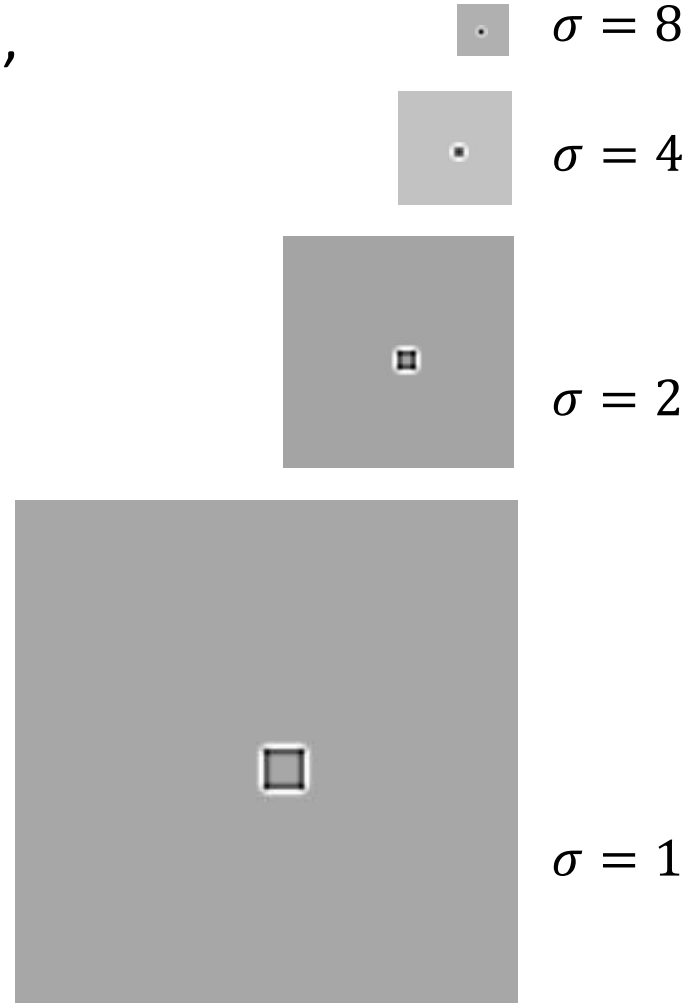


Easy to compute: just take differences of two neighboring levels (within each octave). Subsample with stride 2 every octave.

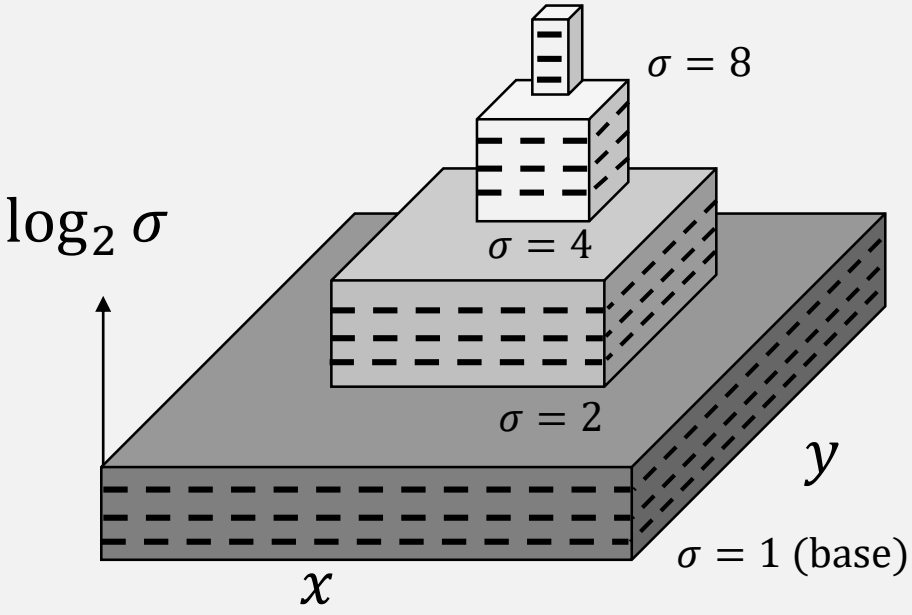
`Difference of Gaussian' Pyramid

The images are lined up within each octave, which *somewhat* simplifies finding minima and maxima in scale space.

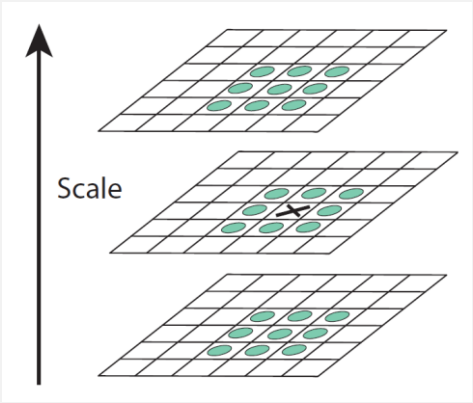




Sampled
Difference of Gaussian
pyramid

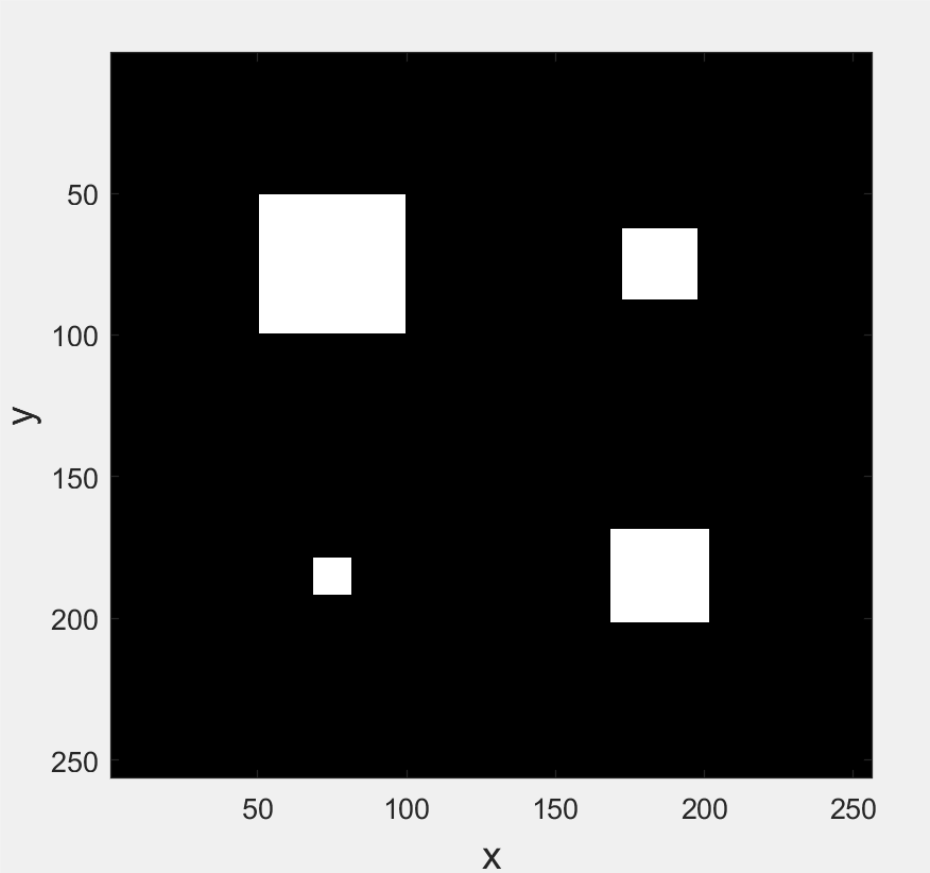


The figure below is from Lowe (2004).



Example

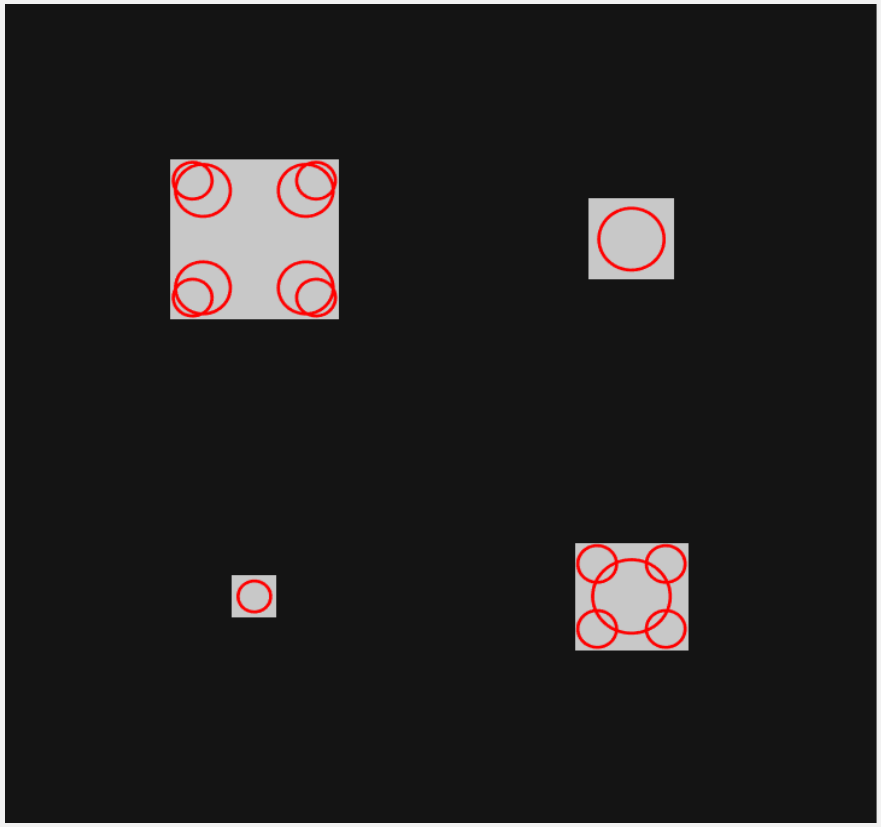
In Assignment 2, you will implement a simplified version of part of SIFT. Here is an example in which I will mark their position and scale with a red circle (radius = keypoint scale).



Example

Results are less clean that I had hoped. (The implementation ignores some details specified in Lowe's paper.)

One of the four boxes is not found. Some of the keypoints found are not boxes.



ASIDE: The red circles were drawn using Matlab's `viscircles`. The radius of circles was supposed to be the scale of the keypoint. However, this is not what happened. (I will further investigate why and possibly change this slide later.)

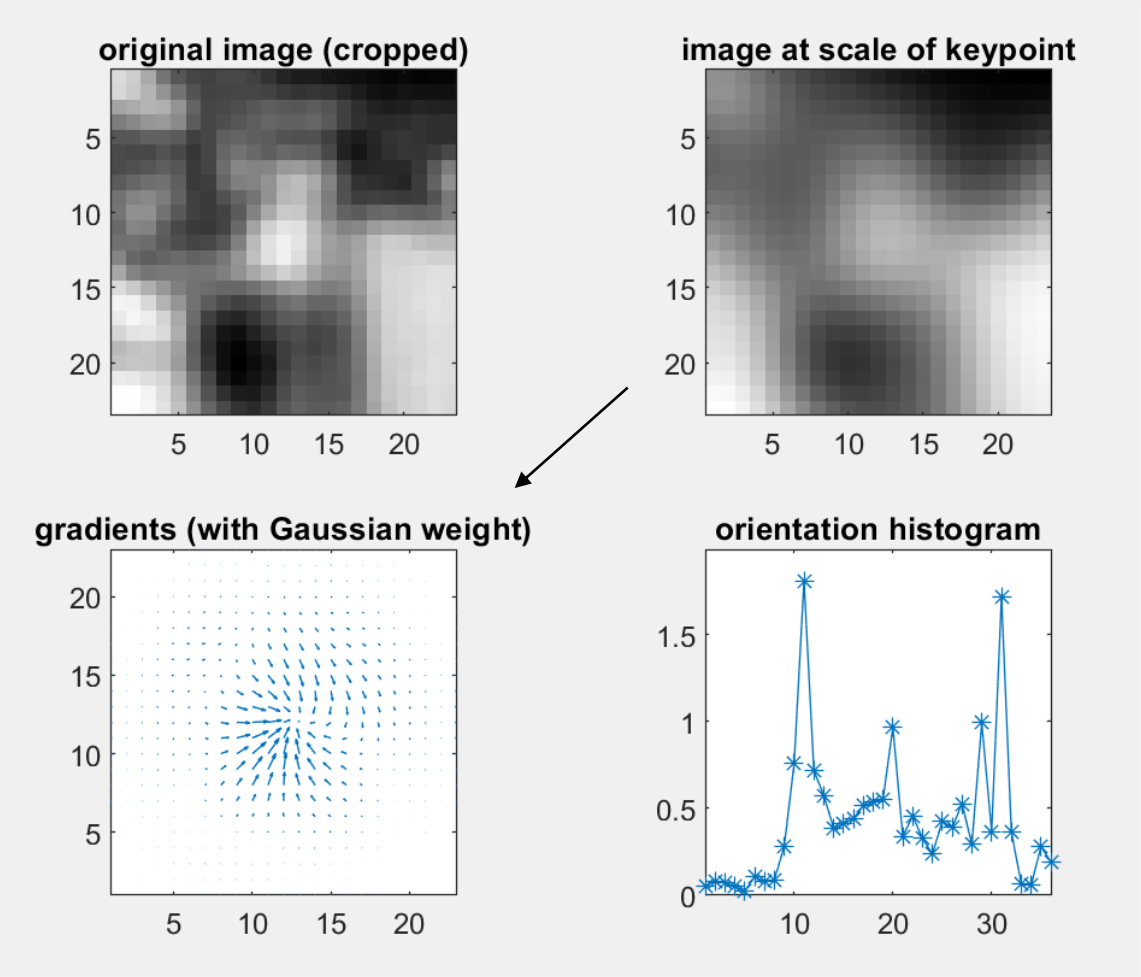
SIFT overview

A SIFT keypoint consists of

- position (x, y)
- scale σ
- orientation θ
  - Choose a dominant orientation of the image gradients in the neighborhood of the keypoint
- image gradient descriptor (128-d vector)
  - summarize the spatial structure of the image gradients

SIFT step 2: orientation

Make a "orientation histogram" out of the intensity gradient vectors in the (x, y) neighborhood and at the scale σ of the keypoint.

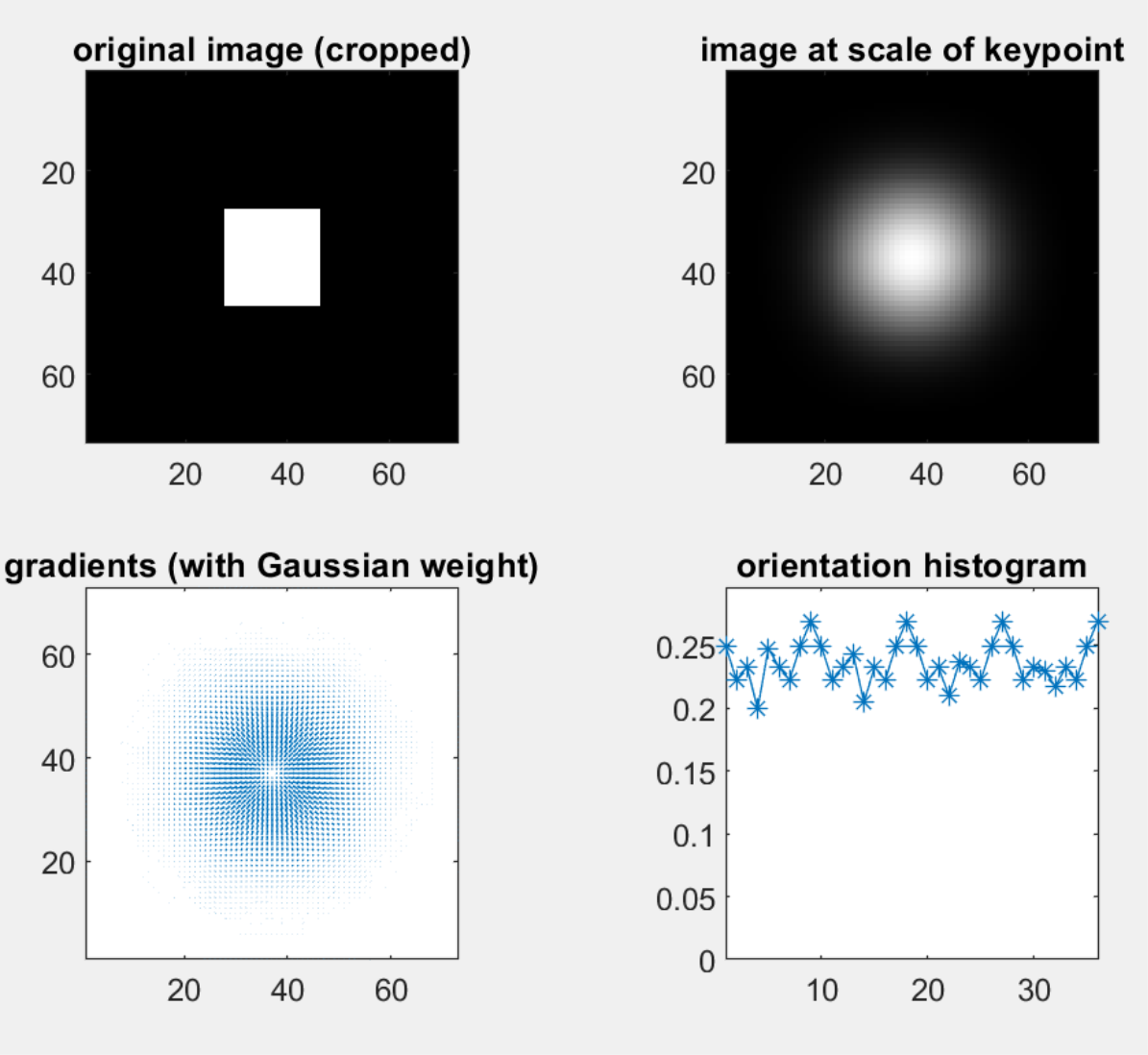


The peak of the histogram defines the *orientation* of a SIFT keypoint.

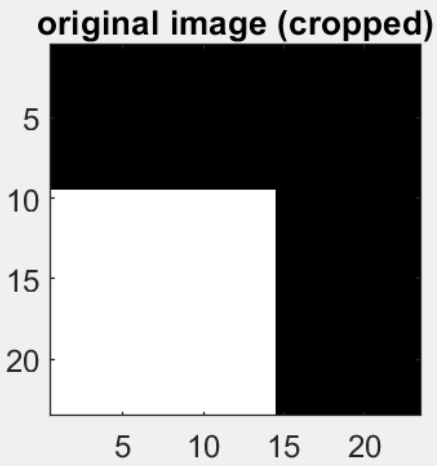
If there is more than one peak, then multiple SIFT keypoints will be created.

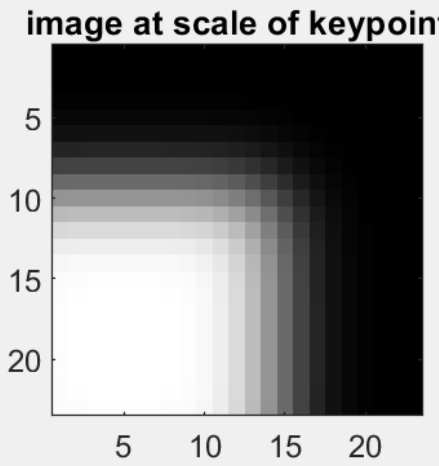
Lowe suggested using all peaks within 80% of max.

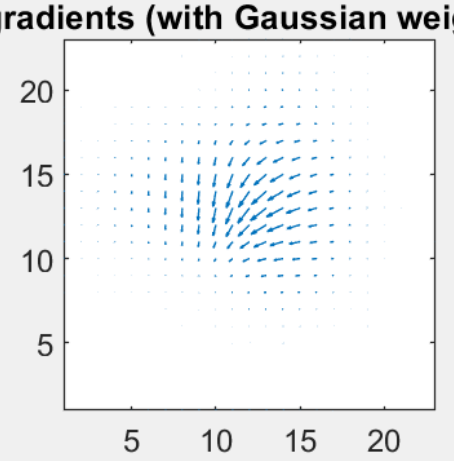
Example (box)

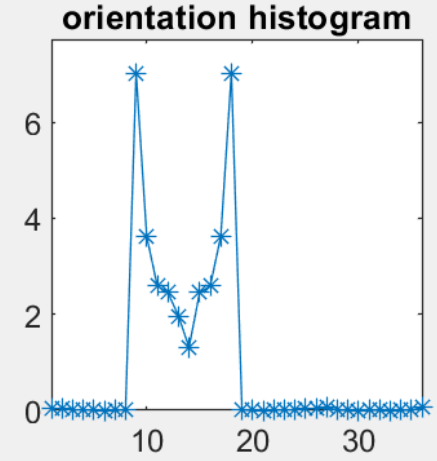

original image (cropped)
20
40
60
20
40
60
image at scale of keypoint
20
40
60
20
40
60
gradients (with Gaussian weight)
60
40
20
20
40
60
orientation histogram
0.25
0.2
0.15
0.1
0.05
0
10
20
30

Example (corner of a box)

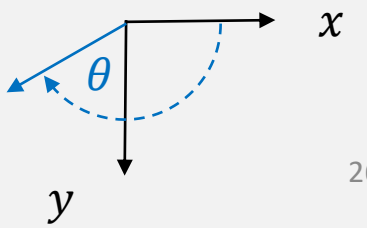








For this example, two peaks occur, namely near 90 degrees and 180 degrees.



SIFT overview

A SIFT keypoint consists of

- position (x, y)
- scale σ
- orientation θ
- image gradient descriptor (128-d vector)
  - summarize the *spatial structure* of the image gradients

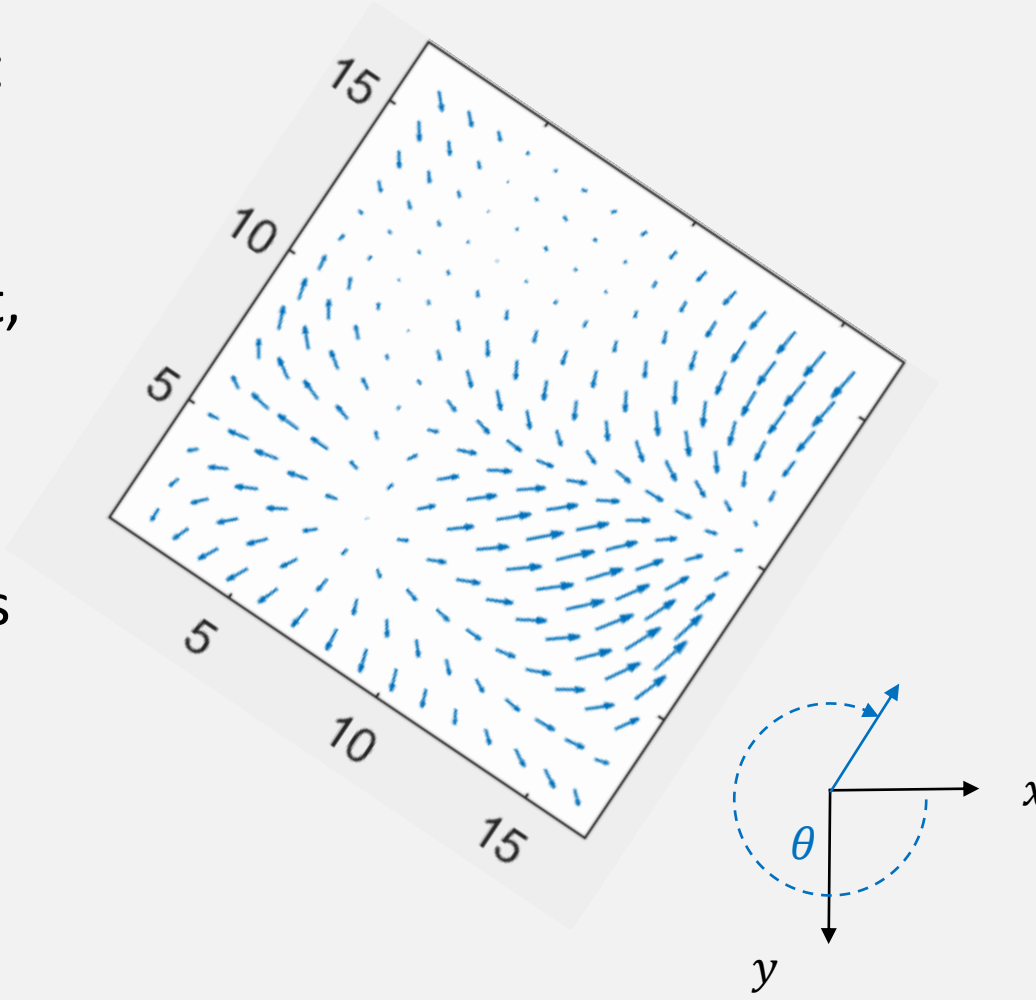
SIFT step 3a: image gradient descriptor

Given the intensity gradients at the scale of the keypoint, …

compute a 16×16 square grid that is centered at the keypoint, and whose vertical axis is aligned with the orientation θ of the keypoint.

Interpolate the image gradients on this grid.

And then …



SIFT step 3b: image gradient descriptor

Partition the 16×16 square grid into 4×4 cells.

For each cell, make an orientation "histogram" out the intensity gradients within that cell. (Weight by distance from center not shown here!)

These 16 histograms use 8 orientations each. This gives a SIFT descriptor vector that is 128 (16*8) dimensional.

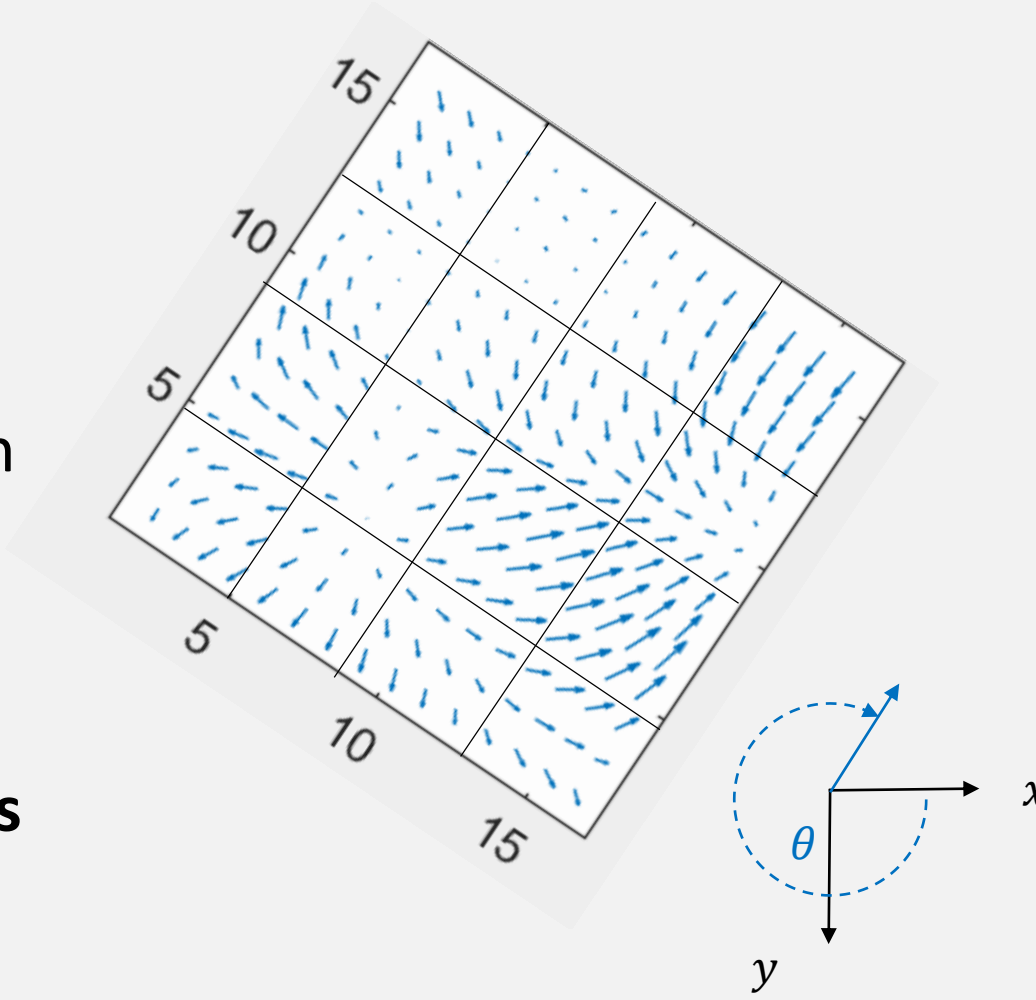


Figure below is from [Lowe, 2004]

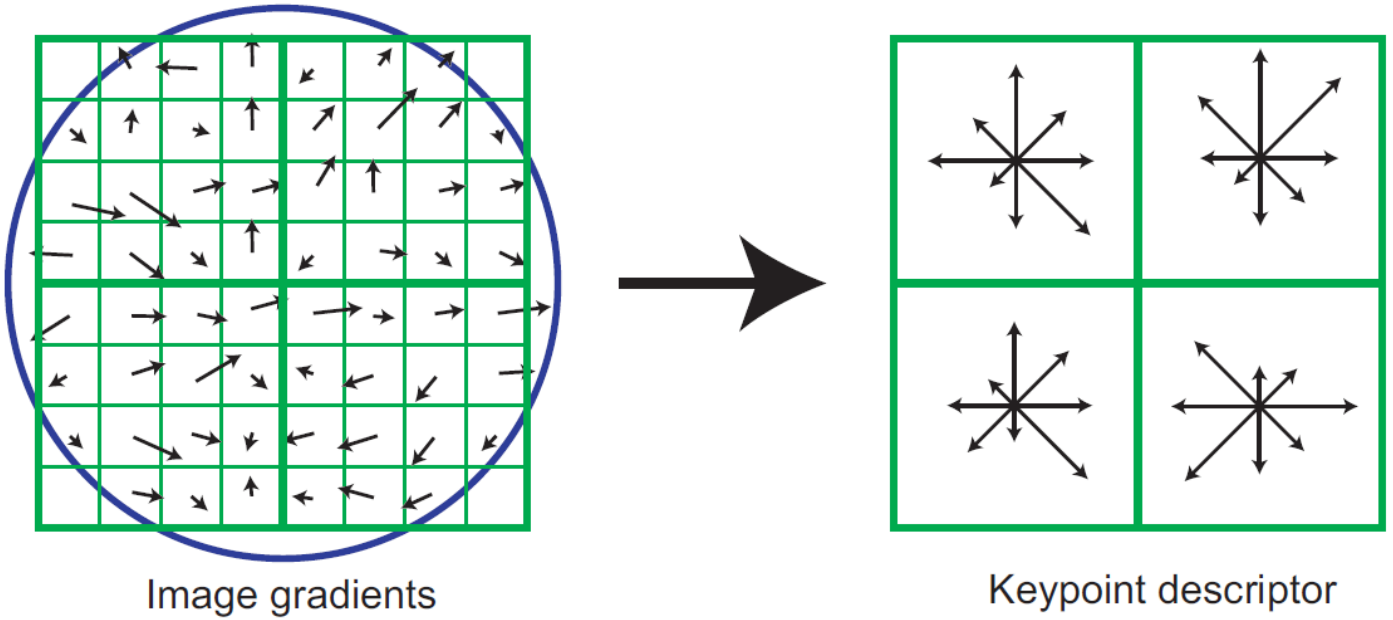


Figure 7: A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4x4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This figure shows a 2x2 descriptor array computed from an 8x8 set of samples, whereas the experiments in this paper use 4x4 descriptors computed from a 16x16 sample array.

"Design details": How many cells? How many orientations per cell? ….

In the 2000's, there were many other proposals for features: SURF, MSER, HOG …

and many other techniques for solving matching problems e.g. "bag of words".

BTW, Matlab doesn't have a SIFT feature.

https://www.mathworks.com/help/vision/ug/local-feature-detection-and-extraction.html

Applications: image matching

1. Image indexing and recognition

2. Image stitching (panoramas)

Here I will only loosely sketch the idea.

I will not examine you on this part.

Image Indexing and Recognition

How to find a match between two images containing the same object?

For example, which (if any) of the three objects shown on left are present in the image on right? (example is from Lowe 1999)

Image Indexing and Recognition
("training phase")

1. We are given a set of "training images". Each image has an ID.

These may of particular objects (cats, dogs, cars, bicycles, cereal boxes, …).

2. For each image, compute SIFT keypoints

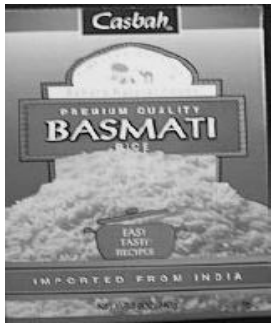
$$\{\ (ID_i, x_i, y_i, \sigma_i, \theta_i, \boldsymbol{d}_i)\ \}$$

3. Build a database for representing *all* keypoints of *all* the training images.

   (Details omitted. There are many ways to do this.)

Image Indexing and Recognition
("testing phase")

Given a new image such as below right, decide if it contains any of the objects in the training set. To do so, we somehow must compare the SIFT keypoints in the new image with keypoints in the database.



?

Image Indexing and Recognition ("testing phase")

Given a new image, decide if "contains" an object in the training set.

One early approach [Lowe, 1999 & 2004]:

- compute the SIFT features of the new image
- Then what ??

Image Indexing and Recognition ("testing phase")

[Lowe, 1999 & 2004]:

- compute the SIFT keypoints of the new image
- for each SIFT feature $(ID, x, y, \sigma, \theta, \boldsymbol{d})$ in the new image ID, find the SIFT keypoints $(ID_i, x_i, y_i, \sigma_i, \theta_i, \boldsymbol{d}_i)$ in the training set such that $\boldsymbol{d}$ is similar to $\boldsymbol{d}_i$

This is a "k nearest neighbor" problem, where we are searching in a 128-d descriptor space.

Then what ?

Image Indexing and Recognition ("testing phase")

[Lowe, 1999 & 2004]:

- compute the SIFT keypoints of the new image
- for each SIFT feature descriptor $\boldsymbol{d}$ in the new image,
  find the SIFT keypoints $(ID_i, x_i, y_i, \sigma_i, \theta_i, \boldsymbol{d}_i)$ in the training set such that $\boldsymbol{d}$ is similar to $\boldsymbol{d}_i$
  cast a vote for these images (ID_i)

The image ID with the most votes is selected.

How can we improve this? What info are we not using?

Image Indexing and Recognition ("testing phase")

How can we improve this? What info are we not using?

We can also use the $(x_i, y_i, \sigma_i, \theta_i)$ information for each keypoint. There is a strong geometric relationship between these parameters in the new image and the training image.

In Lowe's implementation, one estimates a 2D transformation (translation, rotation, shear, scaling) that relates the new image to the candidate image ID in the database. (Details omitted!)

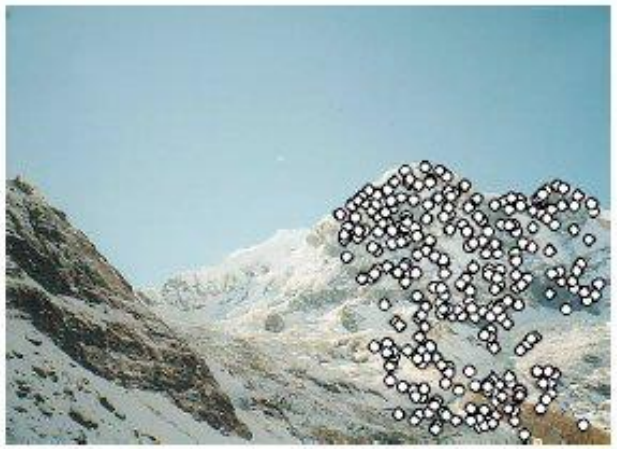
Examples of Training Images

Test image

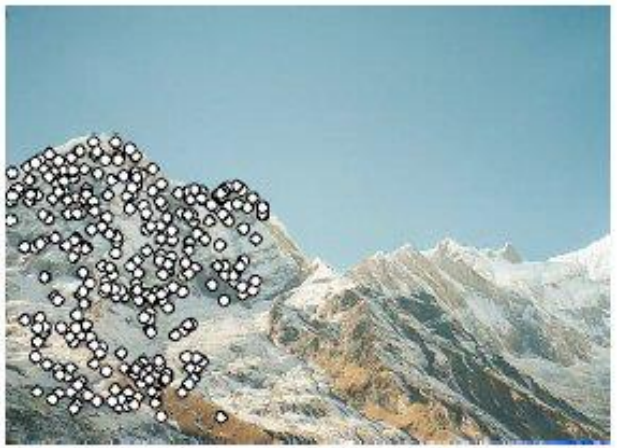
Application 2: Panoramas

(a) Image 1

(b) Image 2



(c) SIFT matches 1



(d) SIFT matches 2

Brown and Lowe, ICCV03

Application: Panoramas

The set of corresponding features is used to estimate a more general 2D mapping (called a “homography”) which can then be used to render a panorama. We will learn about this a few lectures from now.

Coming up….

- Quiz 2 today
- Assignment 2 *should* be posted end of week
- I will reorder the upcoming lectures
  - next week I will start the 3D vision topics
  - lectures on CNN's and detection/identification will be moved to later in the course