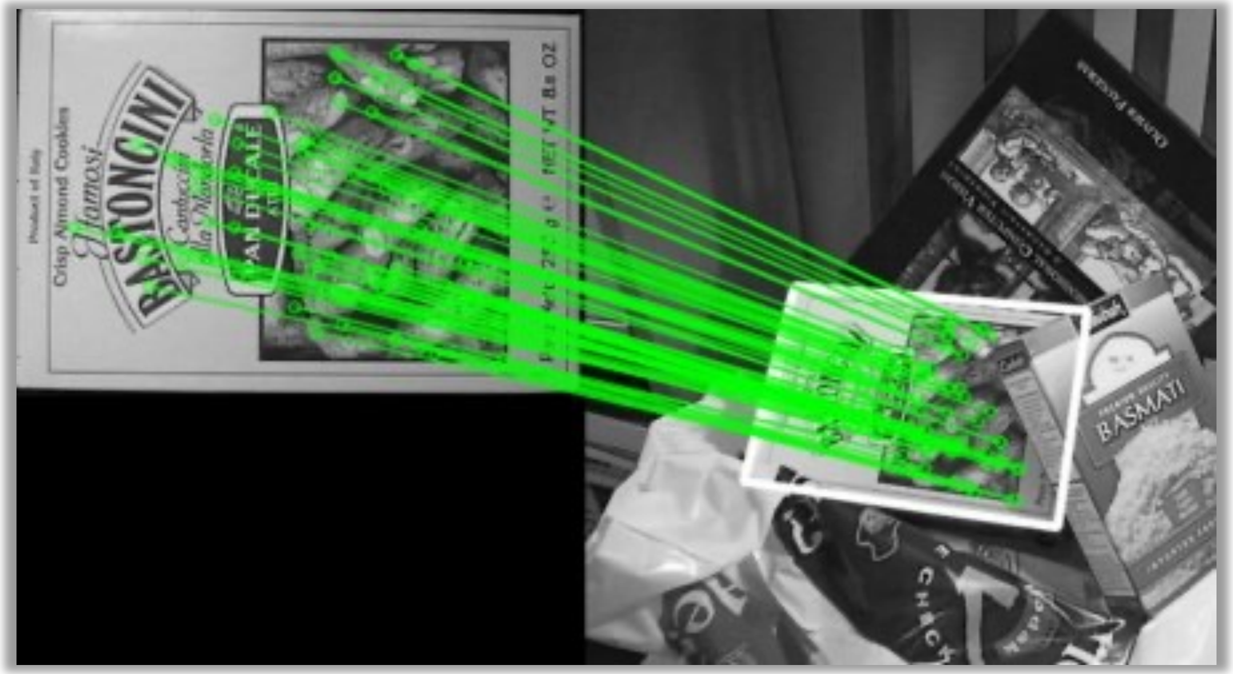
Scale Invariant Keypoints & Feature Descriptors



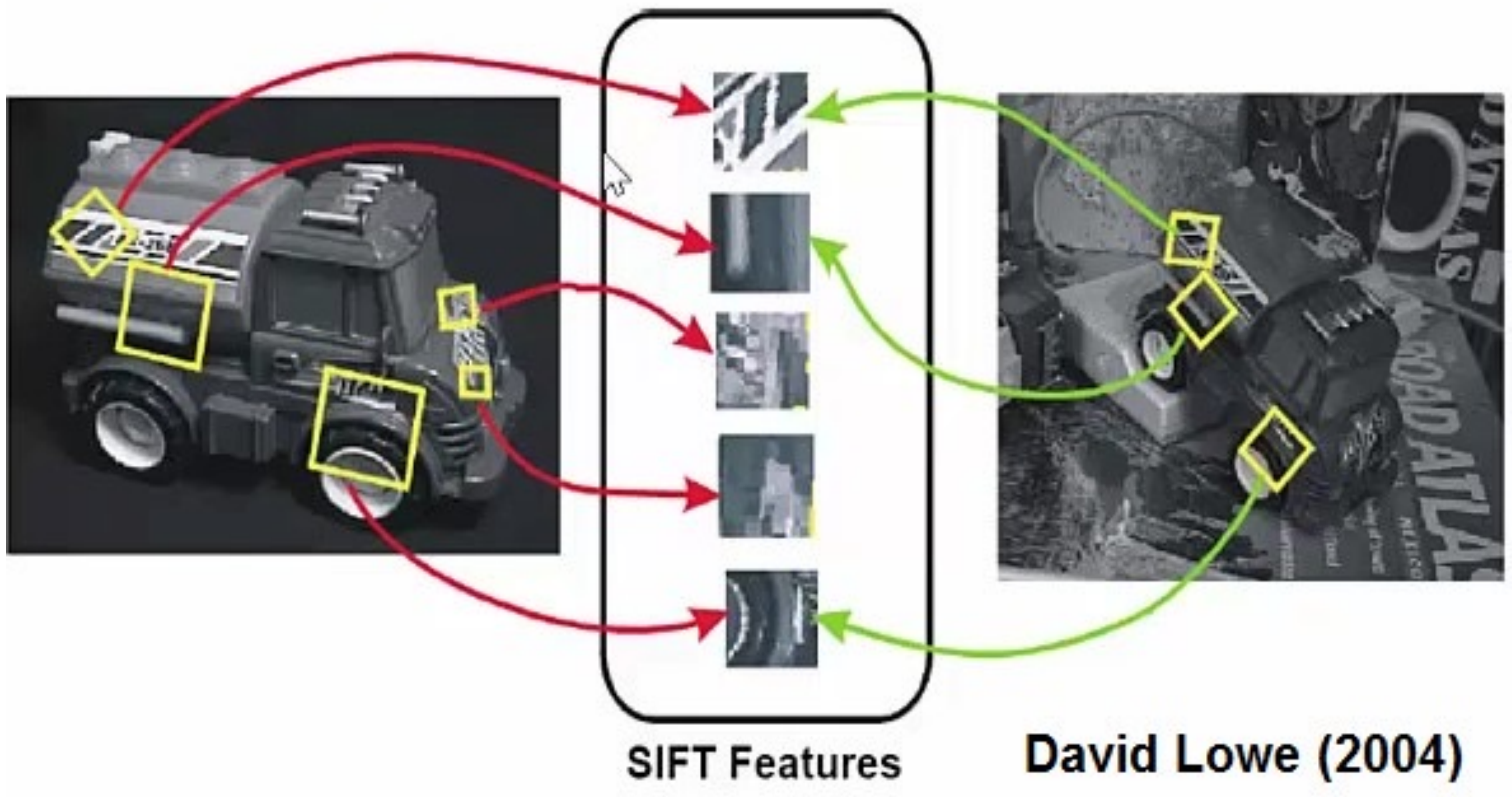
CSC420
David Lindell
University of Toronto
cs.toronto.edu/~lindell/teaching/420
Slide credit: Babak Taati ←Ahmed Ashraf ←Sanja Fidler

TCIG
Toronto Computational
Imaging Group

Overview

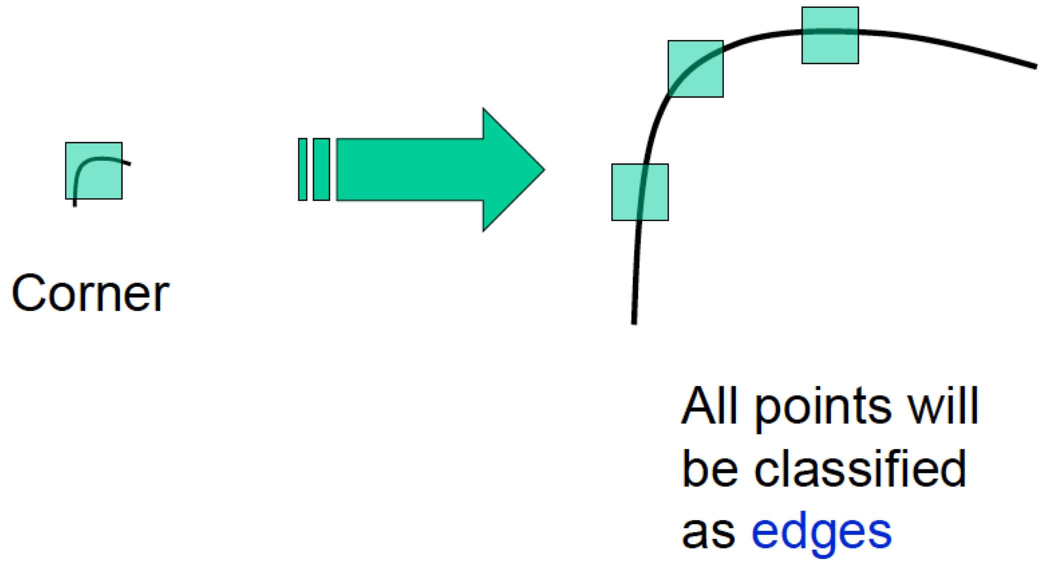
- motivation
- scale invariant keypoint detection
- learned keypoint detection
- image features
- matching

Scale Invariant Feature Transform (SIFT)



Properties of Harris Corner Detector

- Scale?



- Corner location is not scale invariant/covariant!

[Source: J. Hays]

Our Goal: Matching Objects / Images

- Our goal is to be able to match an object in different images where the object appears in different scale, rotation, viewpoints, etc.

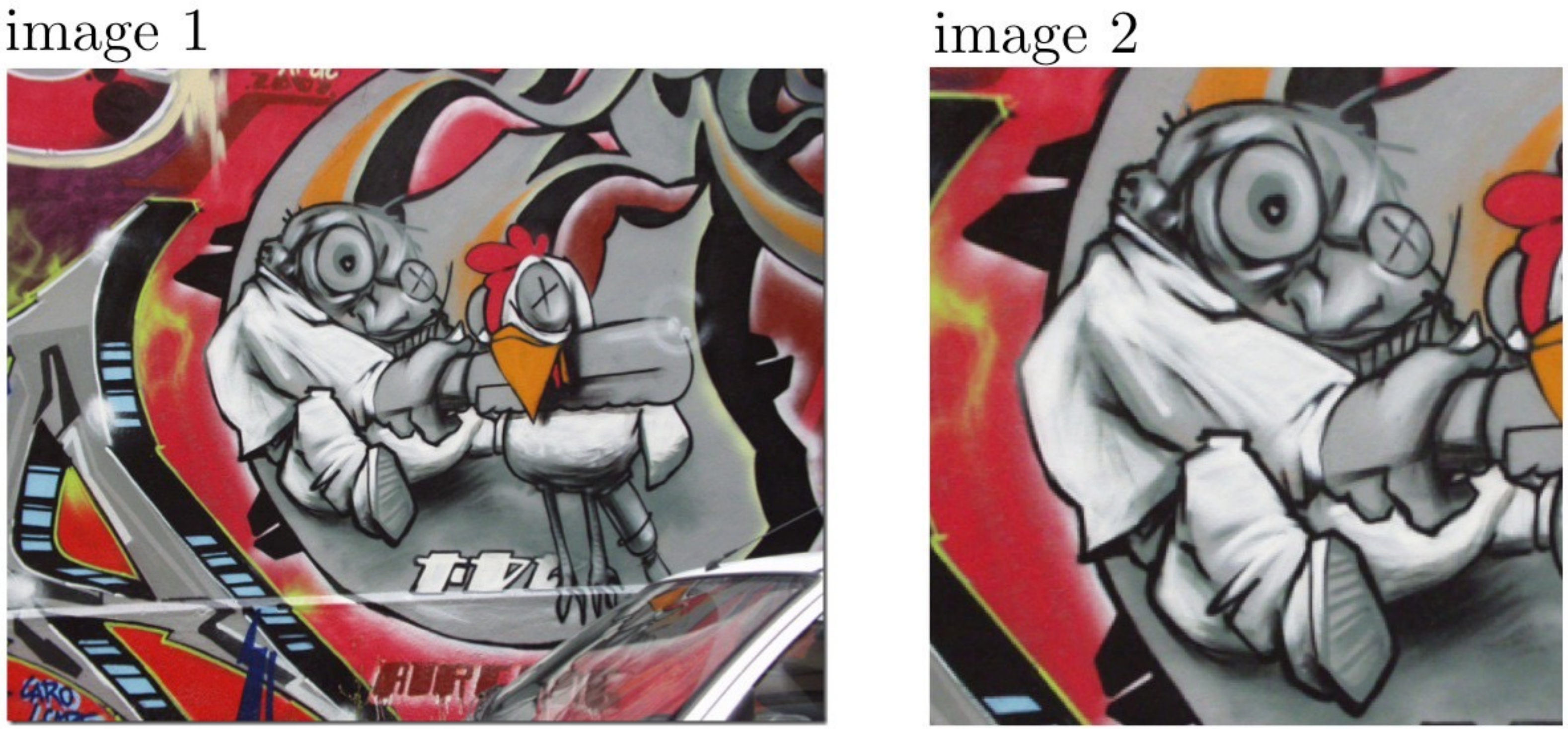


Figure: We want to be able to match these two objects / images

Our Goal: Matching Objects / Images

- Our goal is to be able to match an object in different images where the object appears in different scale, rotation, viewpoints, etc. How?

image 1

image 2

Figure: But these shouldn't be matched!

Our Goal: Matching Objects / Images

- Our goal is to be able to match an object in different images where the object appears in different scale, rotation, viewpoints, etc. How?
  - Find interest points on each image

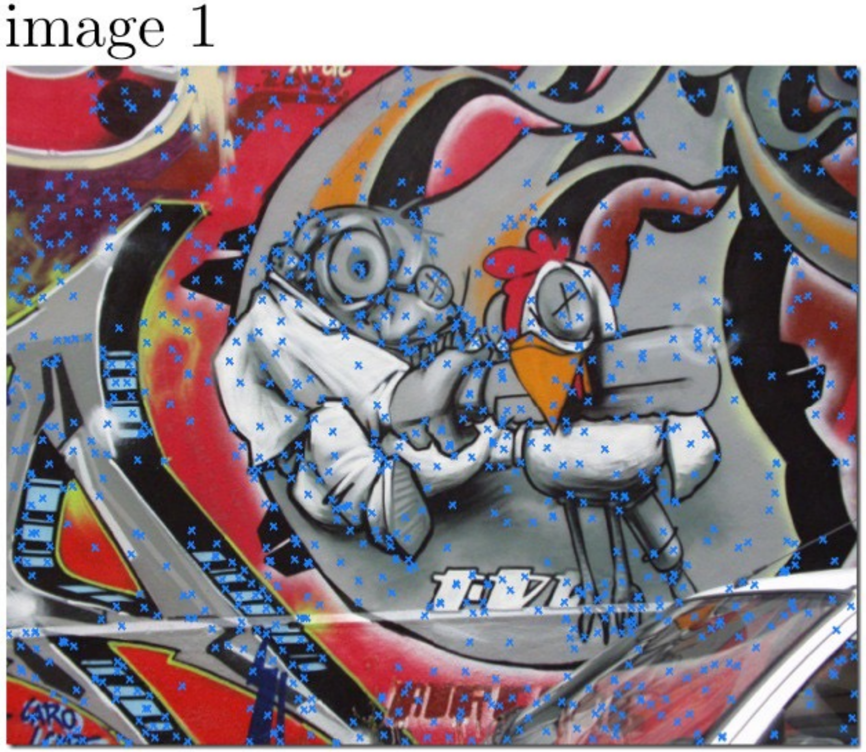


Figure: Find some interest points in an image

Our Goal: Matching Objects / Images

- Our goal is to be able to match an object in different images where the object appears in different scale, rotation, viewpoints, etc. How?
  - Find interest points on each image

Figure: And independently in other images

Our Goal: Matching Objects / Images

- Our goal is to be able to match an object in different images where the object appears in different scale, rotation, viewpoints, etc. How?
  - Find interest points on each image

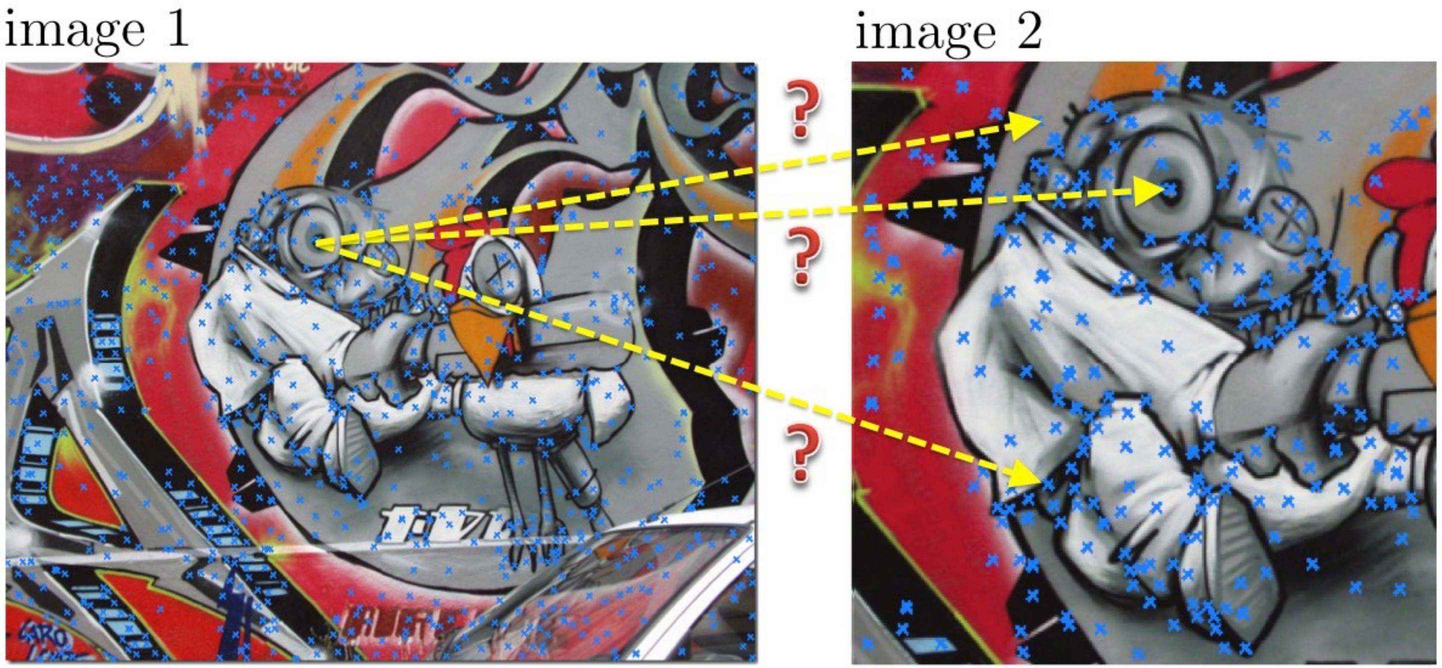


Figure: How can we match points??

Our Goal: Matching Objects / Images

- Our goal is to be able to match an object in different images where the object appears in different scale, rotation, viewpoints, etc. How?
  - Find interest points on each image
  - Form a vector description of each point. How? What size? Length?

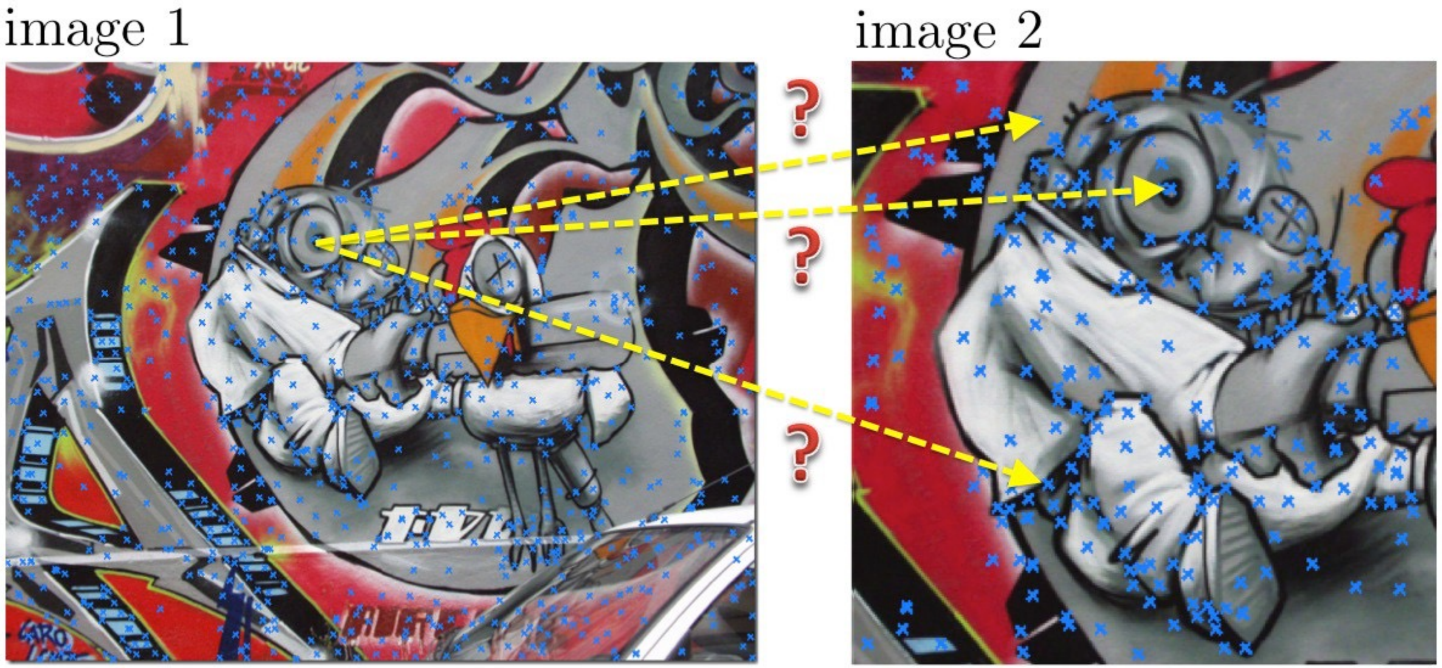


Figure: We could match if we took a patch around each point, and describe it with a feature vector (we know how to compare vectors)

Our Goal: Matching Objects / Images

- Our goal is to be able to match an object in different images where the object appears in different scale, rotation, viewpoints, etc. How?
  - Find interest points on each image
  - Form a vector description of each point. How? What size? Length?

image 1

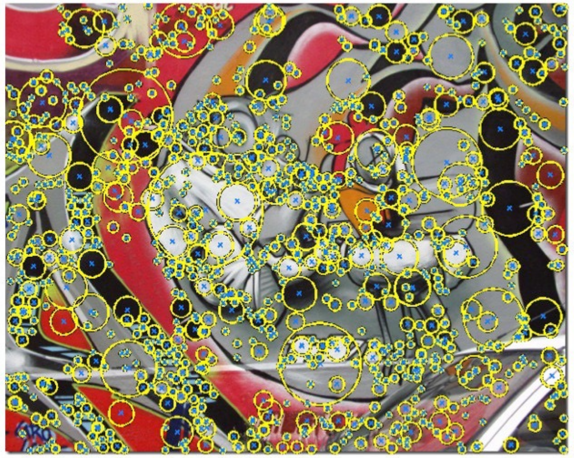


Figure: What if my interest point detector tells me the size (scale) of the patch? We are hoping that this "canonical" size somehow reflects size of the object.

Our Goal: Matching Objects / Images

- Our goal is to be able to match an object in different images where the object appears in different scale, rotation, viewpoints, etc. How?
  - Find interest points on each image
  - Form a vector description of each point. How? What size? Length?

image 1

Figure: And then we can form our feature vectors with respect to this size (how?)

Our Goal: Matching Objects / Images

- Our goal is to be able to match an object in different images where the object appears in different scale, rotation, viewpoints, etc. How?
  - Find interest points on each image
  - Form a vector description of each point. How? What size? Length?
  - Matching

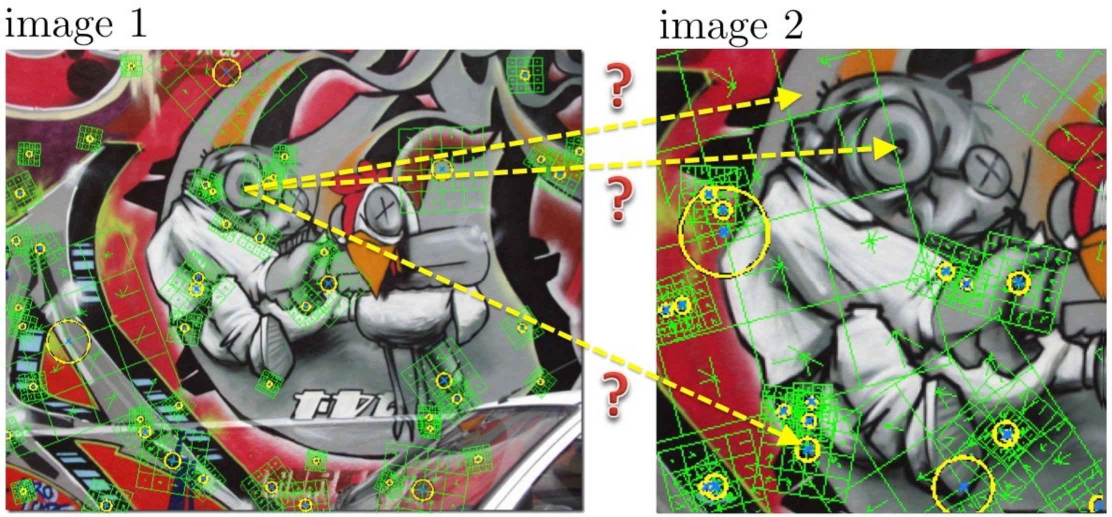


Figure: Then life is easy: we find the best matches and compute a transformation (scale, rotation, etc) of the object – in a later lecture

Our Goal: Matching Objects / Images

- Our goal is to be able to match an object in different images where the object appears in different scale, rotation, viewpoints, etc. How?
  - Find interest points on each image
  - Form a vector description of each point. How? What size? Length?
  - Matching

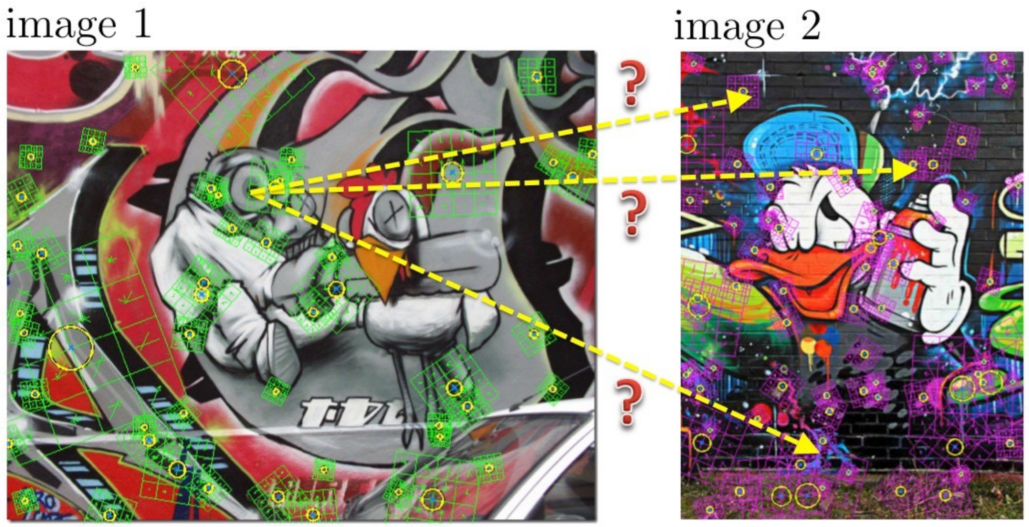


Figure: And we are hoping that our feature vectors and our matching algorithm will be able to say that this image does not contain our object!

Our Goal: Matching Objects / Images

- Our goal is to be able to match an object in different images where the object appears in different scale, rotation, viewpoints, etc. How?
  - Find interest points on each image   Let's do this first!
  - Form a vector description of each point. How? What size? Length?
  - Matching

Overview

- motivation
- scale invariant keypoint detection
- learned keypoint detection
- image features
- matching

Scale Invariant Interest Points

- How can we **independently** select interest points in each image, such that the detections are repeatable across different scales?

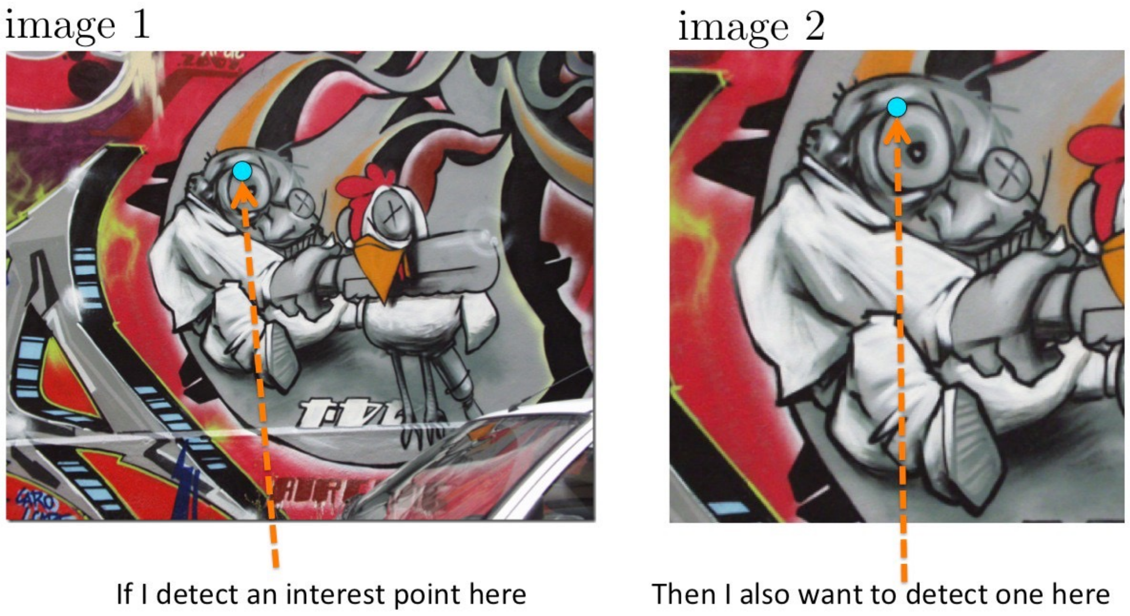
image 1

image 2

[Source: K. Grauman]

Scale Invariant Interest Points

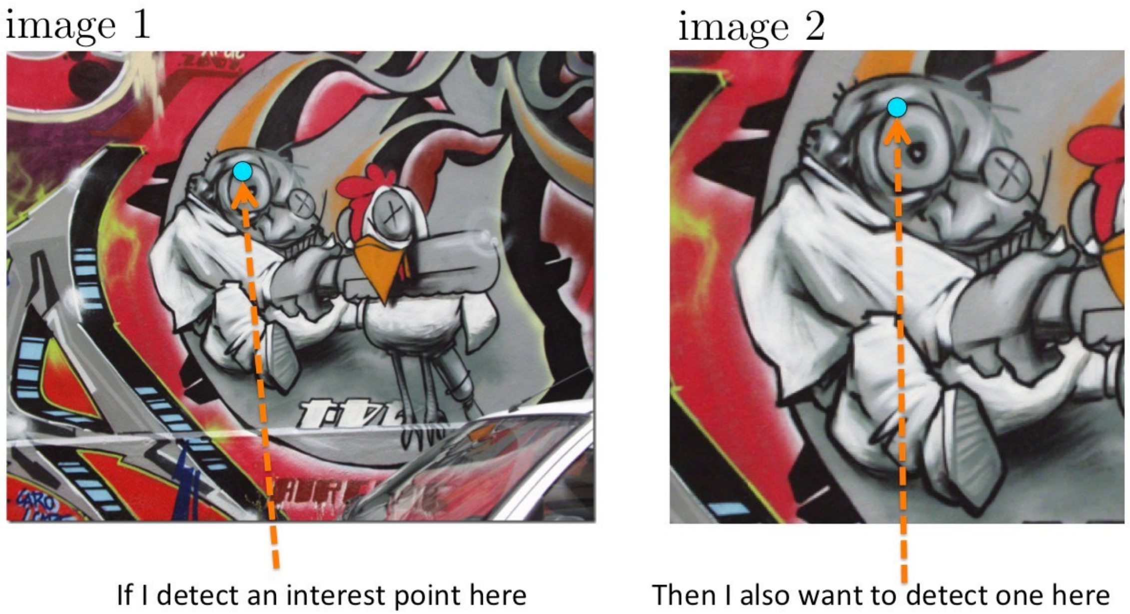
- How can we **independently** select interest points in each image, such that the detections are repeatable across different scales?



[Source: K. Grauman]

Scale Invariant Interest Points

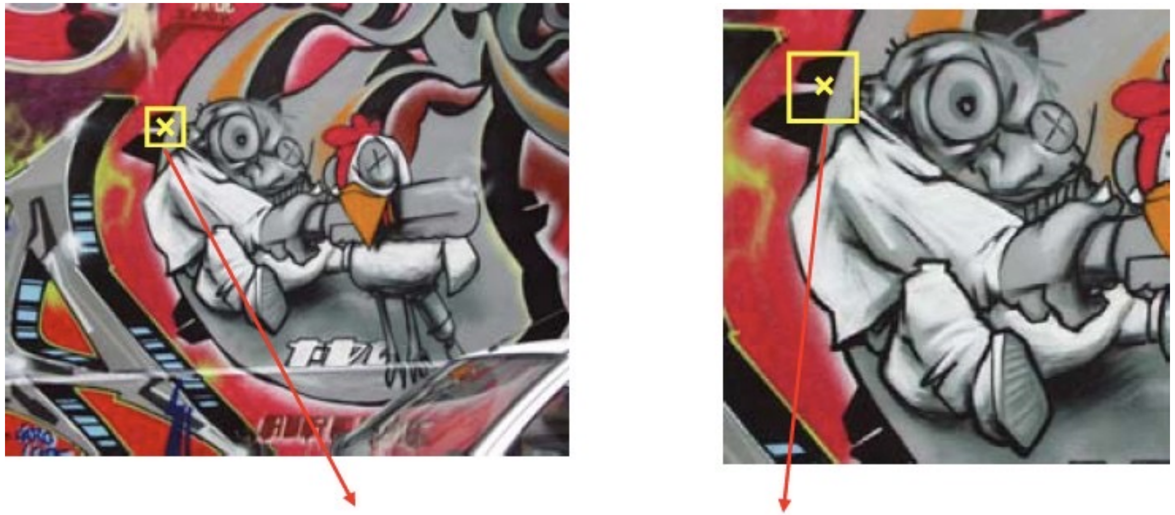
- How can we **independently** select interest points in each image, such that the detections are repeatable across different scales?
  - Extract features at a variety of scales, e.g., by using multiple resolutions in a pyramid, and then matching features at the "corresponding" level.
  - When does this work?



[Source: K. Grauman]

Scale Invariant Interest Points

- How can we **independently** select interest points in each image, such that the detections are repeatable across different scales?
  - More efficient to extract features that are stable in both location and scale.

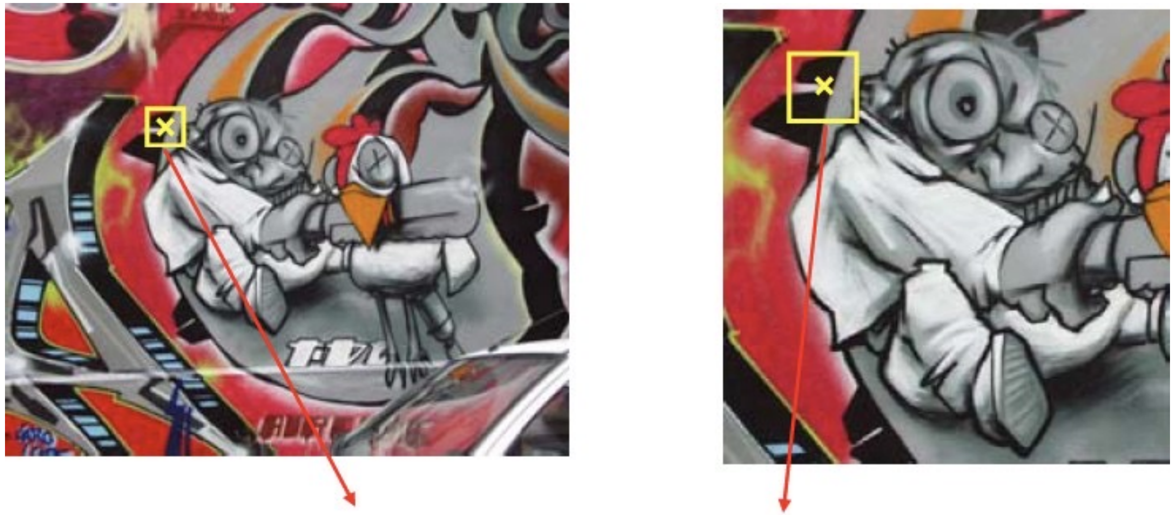


$$f(I_{i_1 \ldots i_m}(x, \sigma)) = f(I_{i_1 \ldots i_m}(x', \sigma'))$$

[Source: K. Grauman]

Scale Invariant Interest Points

- How can we **independently** select interest points in each image, such that the detections are repeatable across different scales?
  - With the Harris corner detector we found a maxima in a spatial search window
  - Find scale that gives local maxima of a function f in both position and scale.



$$f(I_{i_1 \ldots i_m}(x, \sigma)) = f(I_{i_1 \ldots i_m}(x', \sigma'))$$

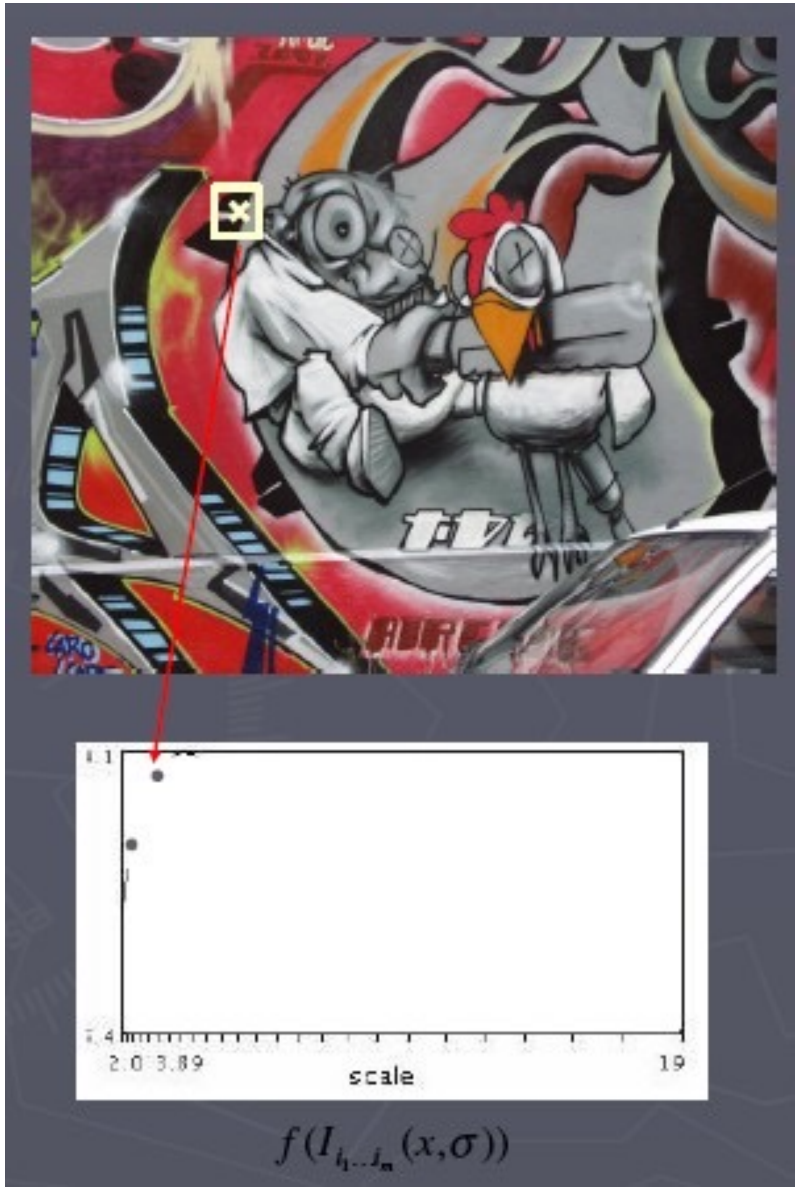
[Source: K. Grauman]

Automatic Scale Selection

- Function responses for increasing scale (scale signature).

Automatic Scale Selection

- Function responses for increasing scale (scale signature).



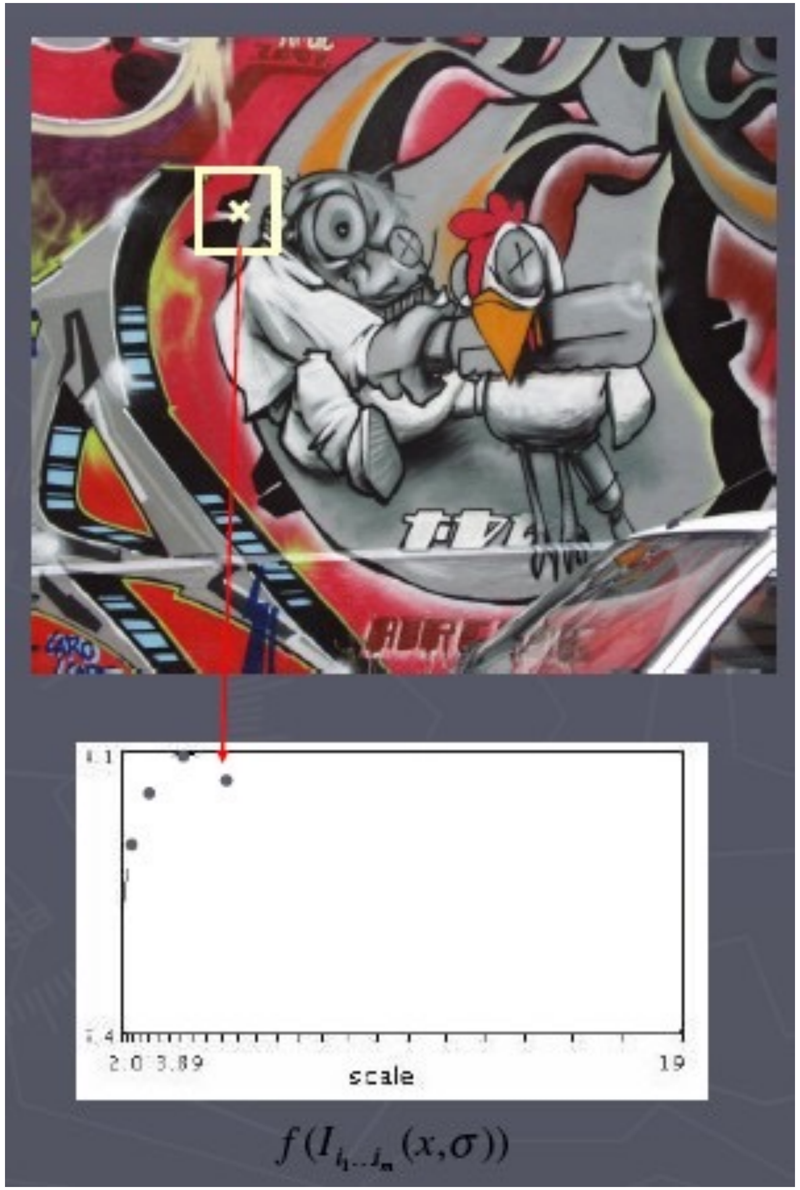
Automatic Scale Selection

- Function responses for increasing scale (scale signature).

$f(I_{i_1 \dots i_m}(x, \sigma))$

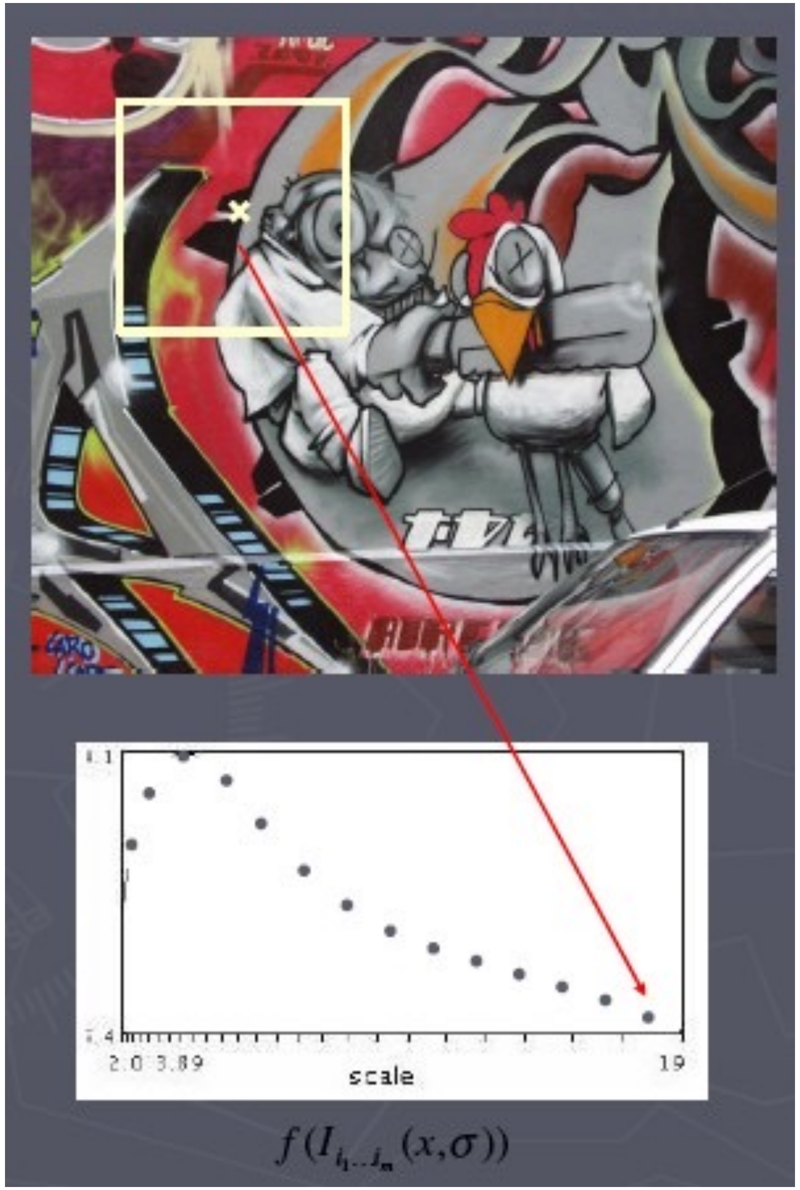
Automatic Scale Selection

- Function responses for increasing scale (scale signature).



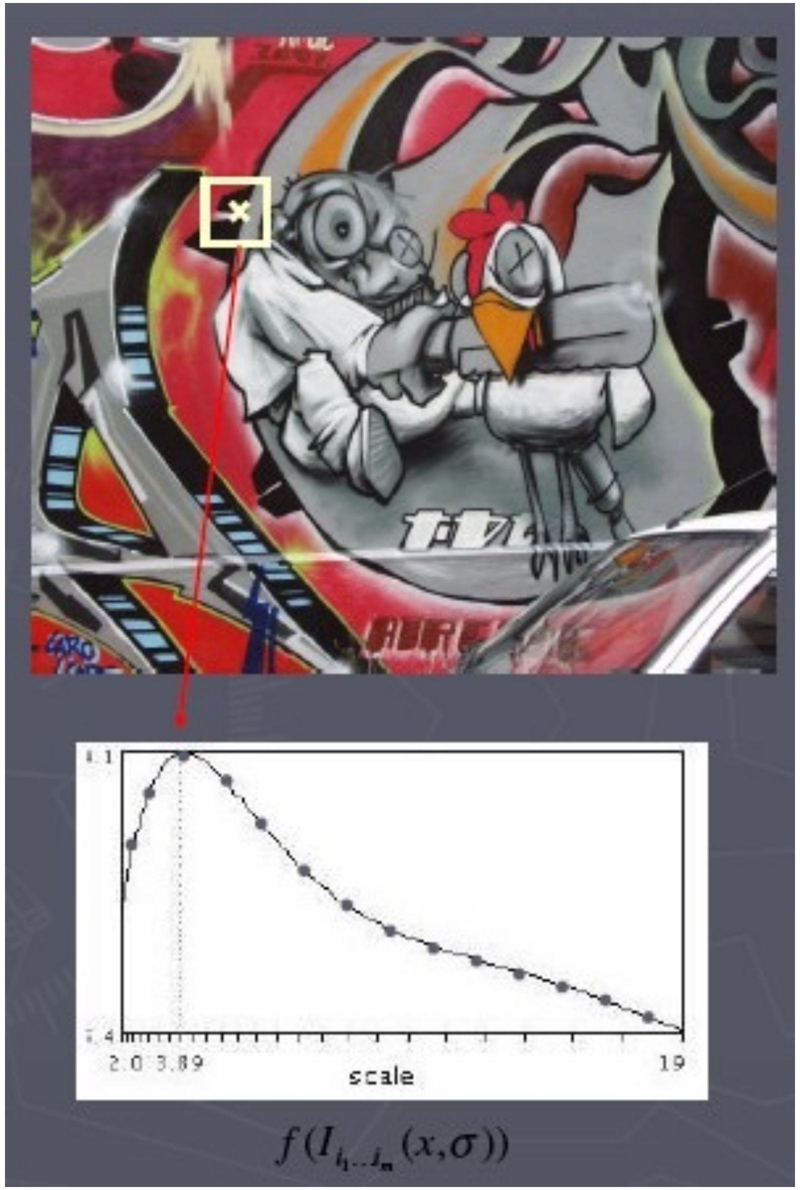
Automatic Scale Selection

- Function responses for increasing scale (scale signature).



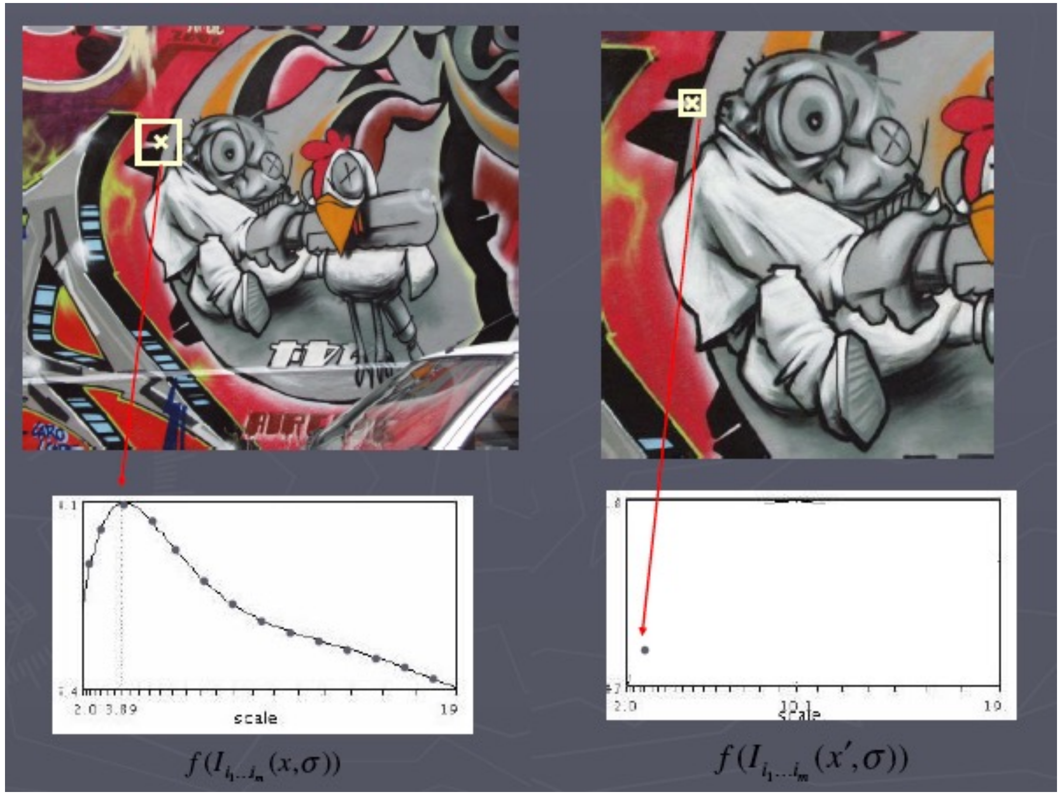
Automatic Scale Selection

- Function responses for increasing scale (scale signature).



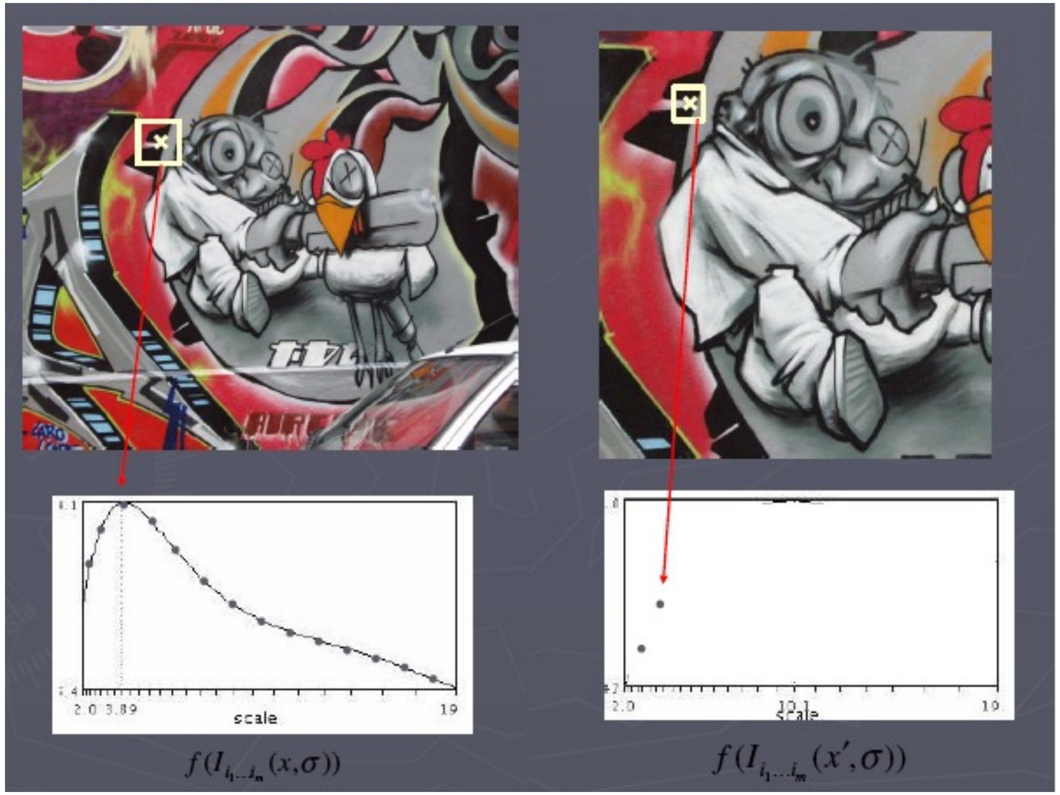
Automatic Scale Selection

- Function responses for increasing scale (scale signature).



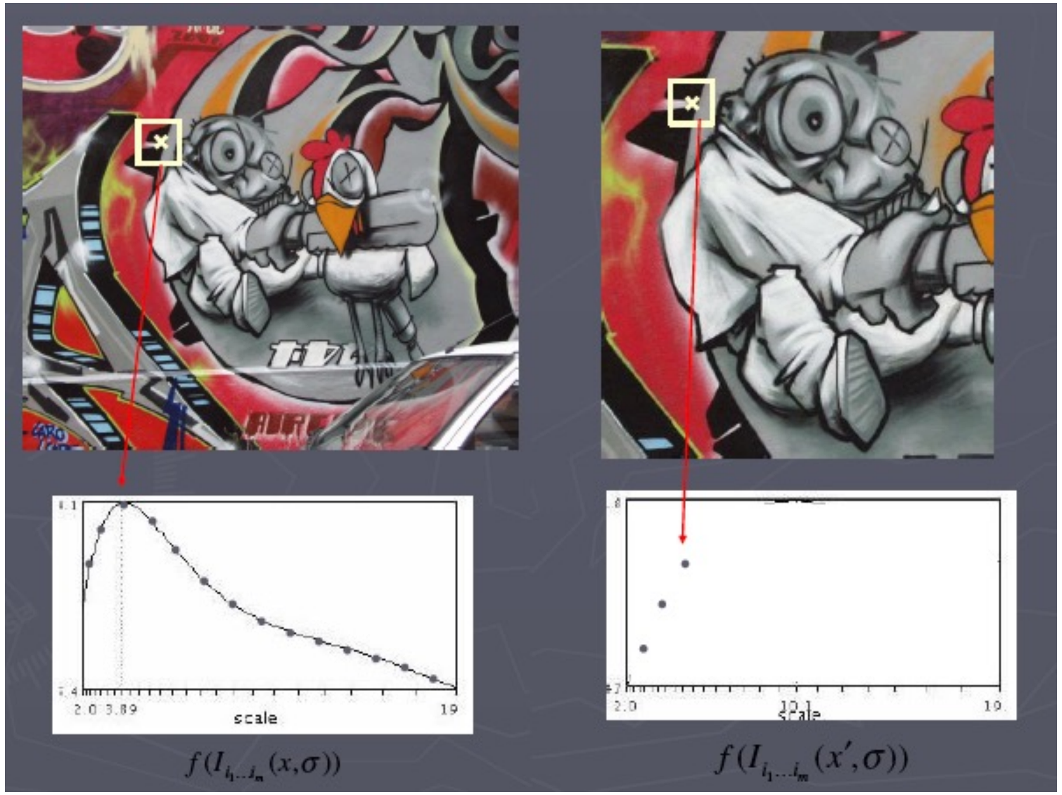
Automatic Scale Selection

- Function responses for increasing scale (scale signature).



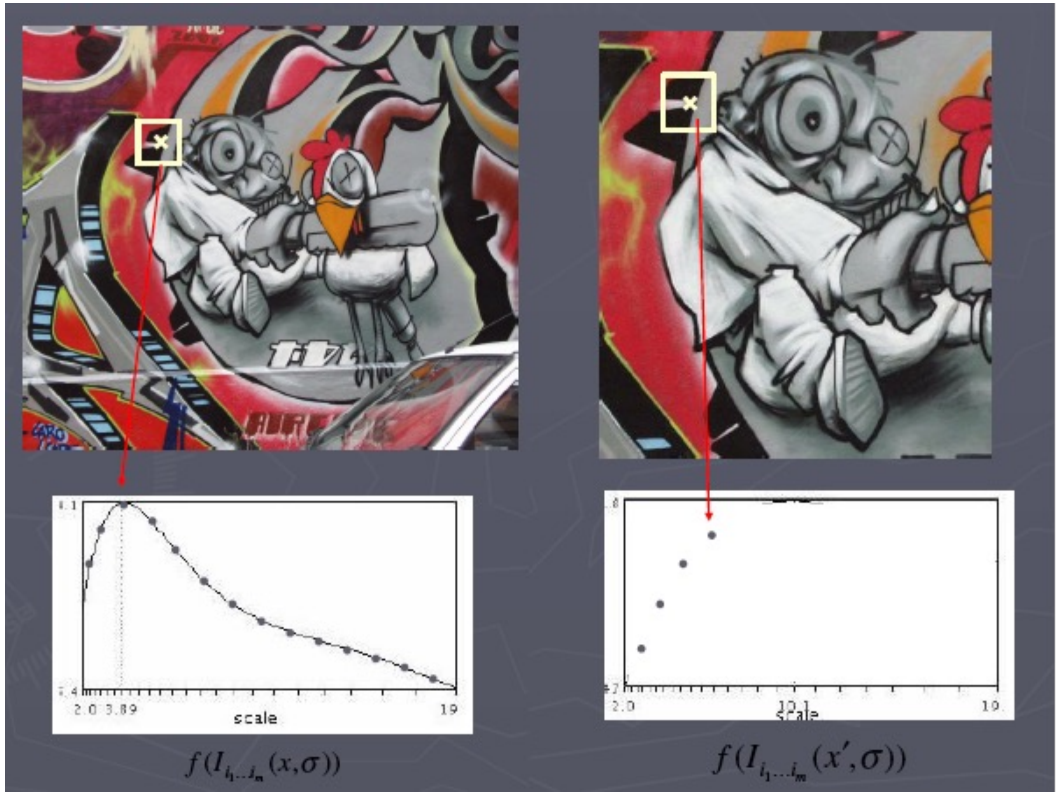
Automatic Scale Selection

- Function responses for increasing scale (scale signature).



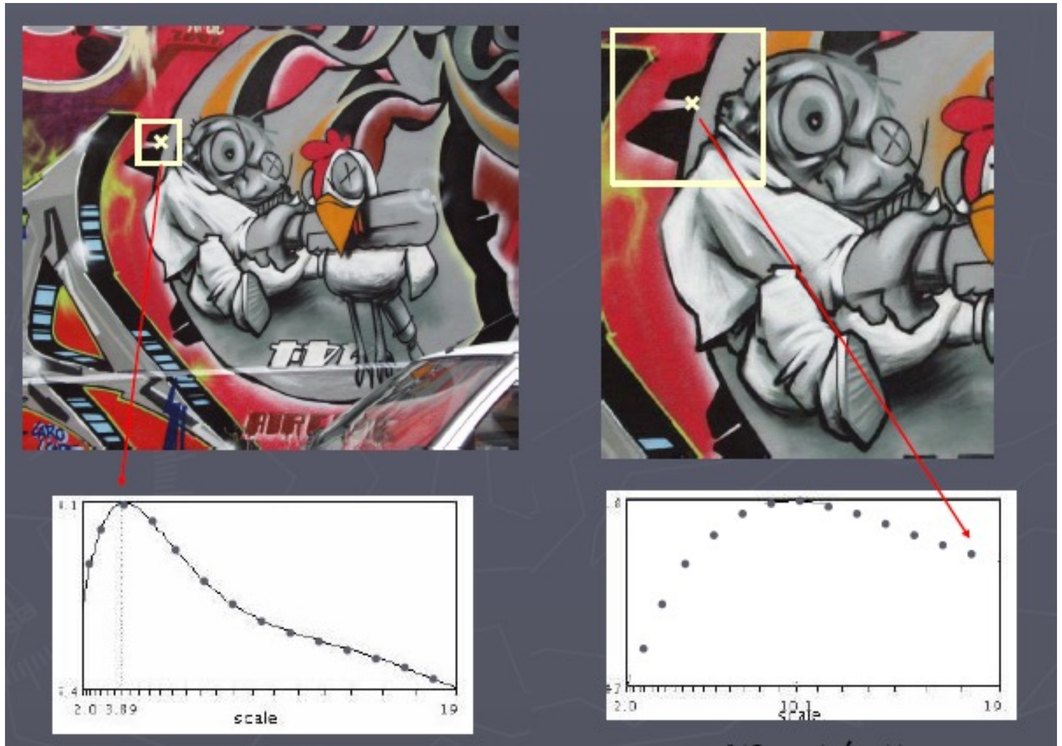
Automatic Scale Selection

- Function responses for increasing scale (scale signature).



Automatic Scale Selection

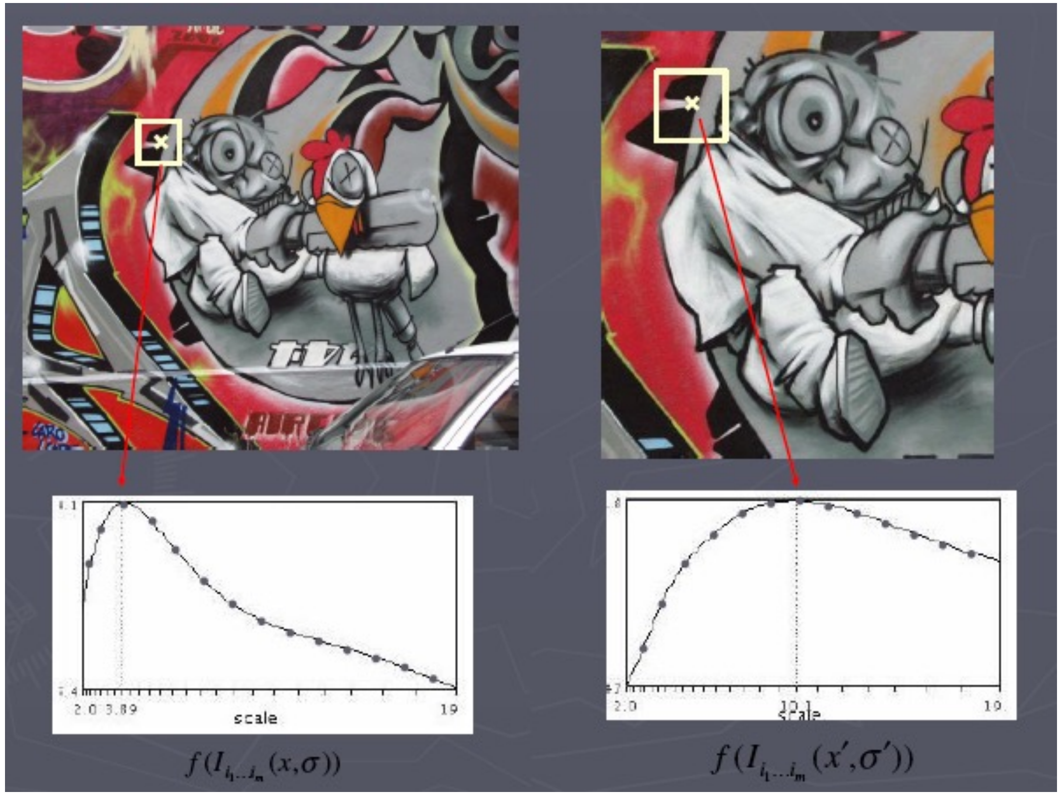
- Function responses for increasing scale (scale signature).



$f(I_{i_1 \ldots i_m}(x,\sigma))$ $f(I_{i_1 \ldots i_m}(x',\sigma))$

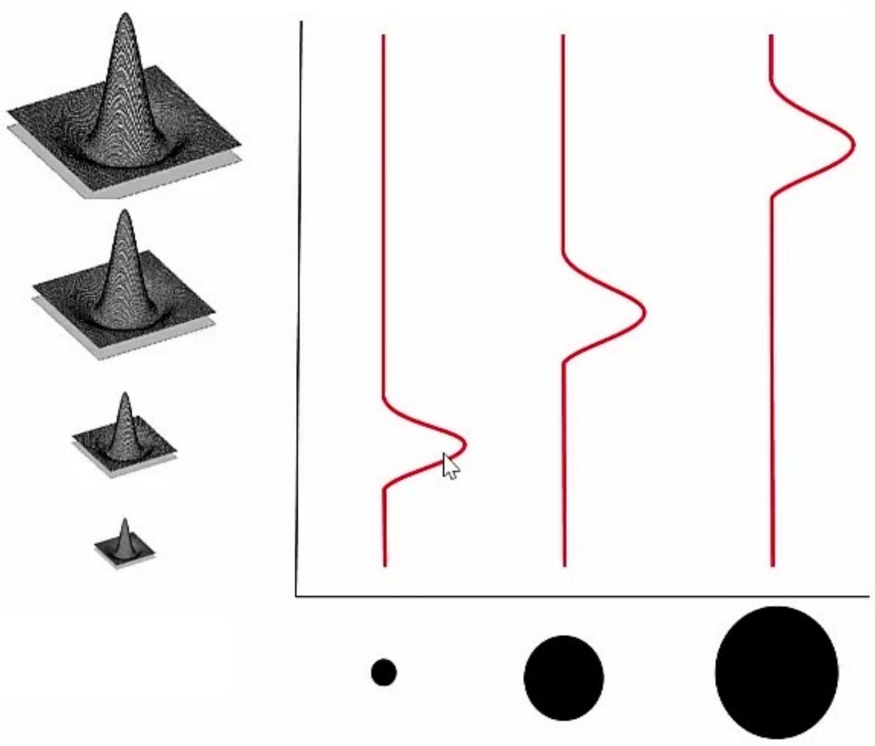
Automatic Scale Selection

- Function responses for increasing scale (scale signature).

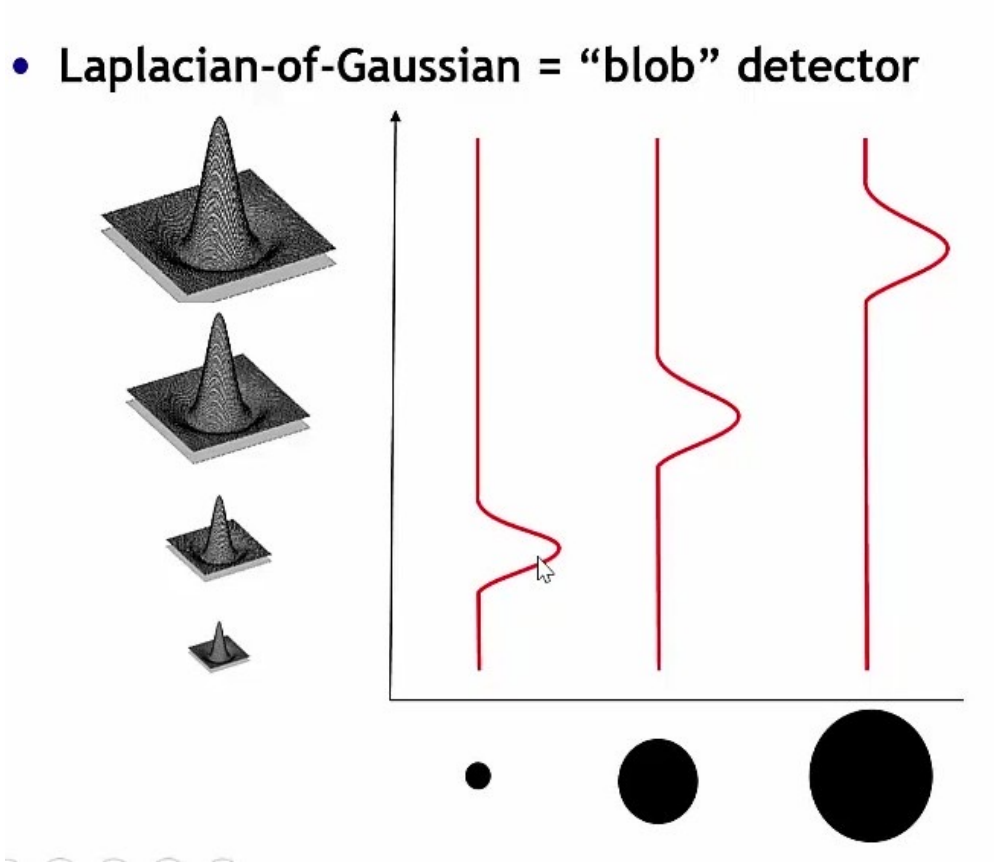


What Can the Signature Function Be?

what does this function detect?

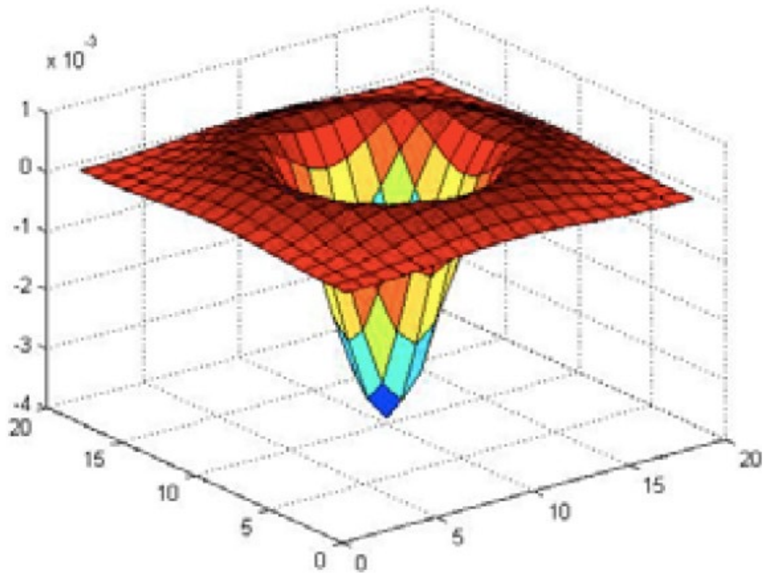


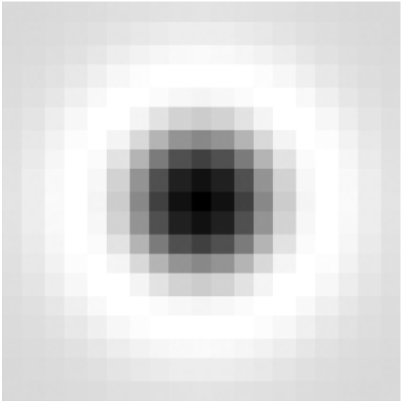
What Can the Signature Function Be?



Blob Detection – Laplacian of Gaussian

- Laplacian of Gaussian: We mentioned it for edge detection
  - $\nabla_g^2(x, y, \sigma) = \frac{\partial^2 g(x,y,\sigma)}{\partial x^2} + \frac{\partial^2 g(x,y,\sigma)}{\partial y^2}$ where G is Gaussian

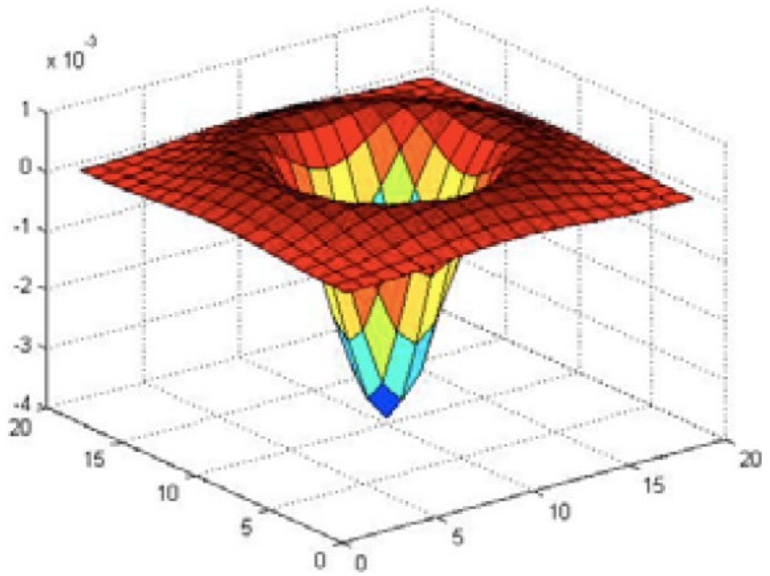


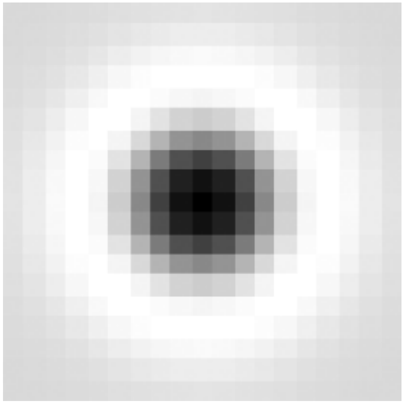


[Source: K. Grauman]

Blob Detection – Laplacian of Gaussian

- Laplacian of Gaussian: We mentioned it for edge detection
  - $\nabla_g^2(x, y, \sigma) = \frac{\partial^2 g(x,y,\sigma)}{\partial x^2} + \frac{\partial^2 g(x,y,\sigma)}{\partial y^2}$ where G is Gaussian
  - $\nabla_g^2(x, y, \sigma) = -\frac{1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right) \exp - \frac{x^2+y^2}{2\sigma^2}$

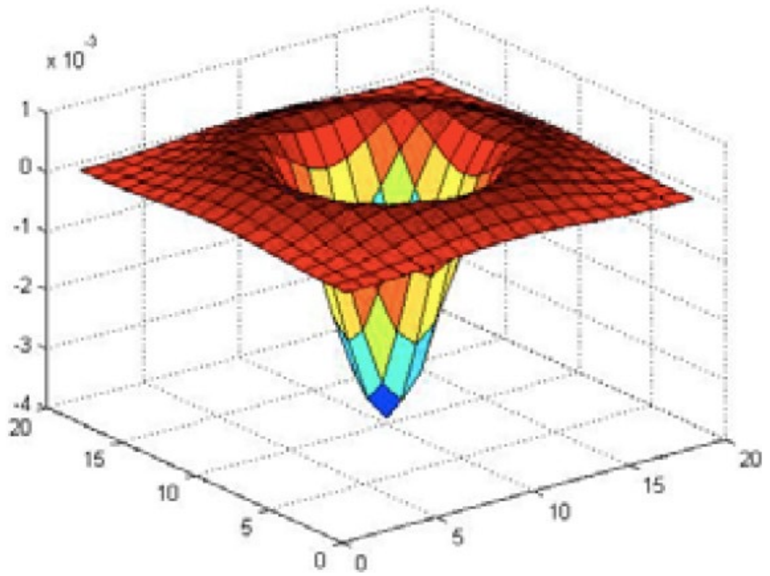


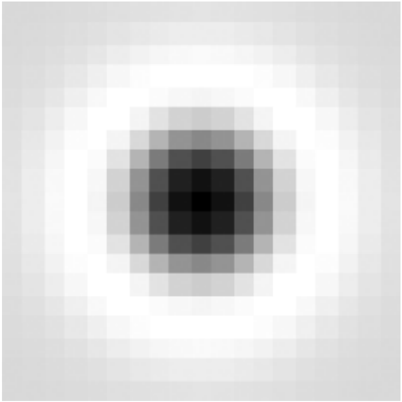


[Source: K. Grauman]

Blob Detection – Laplacian of Gaussian

- Laplacian of Gaussian: We mentioned it for edge detection
  - $\nabla_g^2(x, y, \sigma) = \frac{\partial^2 g(x,y,\sigma)}{\partial x^2} + \frac{\partial^2 g(x,y,\sigma)}{\partial y^2}$ where G is Gaussian
  - $\nabla_g^2(x, y, \sigma) = -\frac{1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right) \exp - \frac{x^2+y^2}{2\sigma^2}$

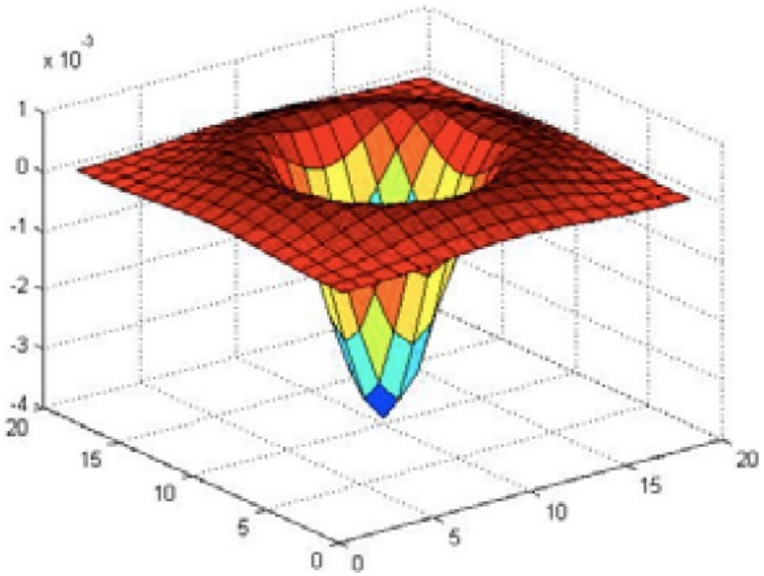


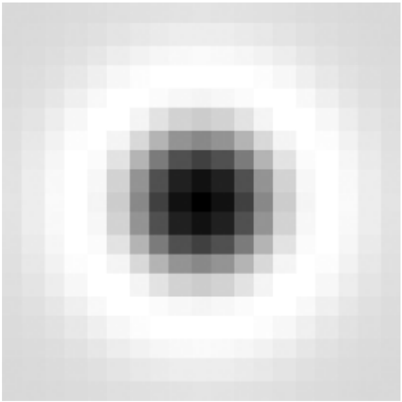


[Source: K. Grauman]

Blob Detection – Laplacian of Gaussian

- Laplacian of Gaussian: We mentioned it for edge detection
- It is a circularly symmetric operator (finds difference in all directions)
- It can be used for 2D blob detection! How?



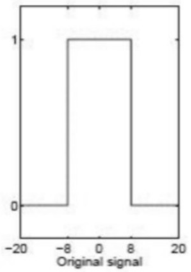


[Source: K. Grauman]

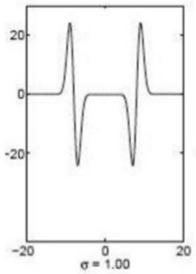
Blob Detection – Laplacian of Gaussian

- It can be used for 2D blob detection! How?

Original signal



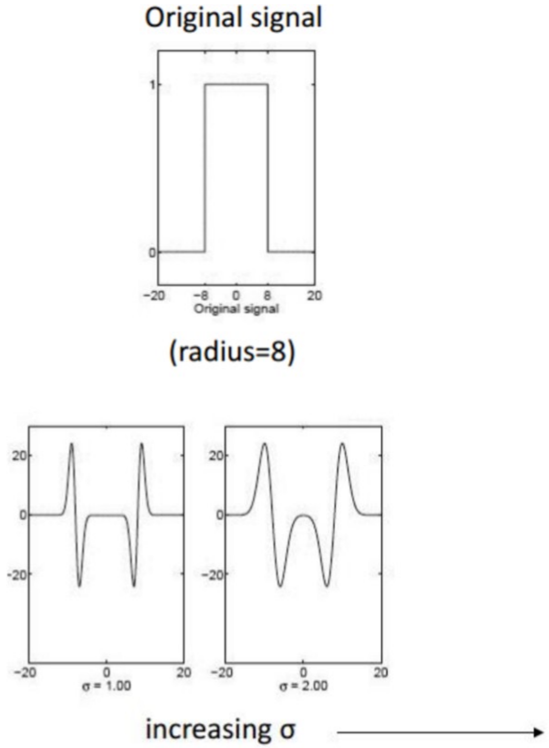
(radius=8)

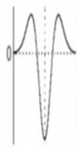


[Source: F. Flores-Mangas]

Blob Detection – Laplacian of Gaussian

- It can be used for 2D blob detection! How?

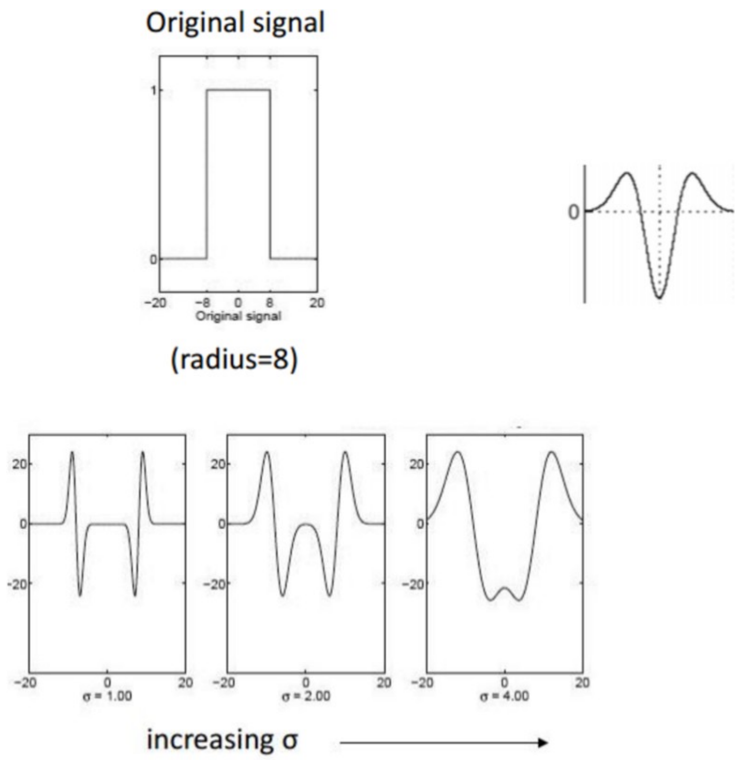




[Source: F. Flores-Mangas]

Blob Detection – Laplacian of Gaussian

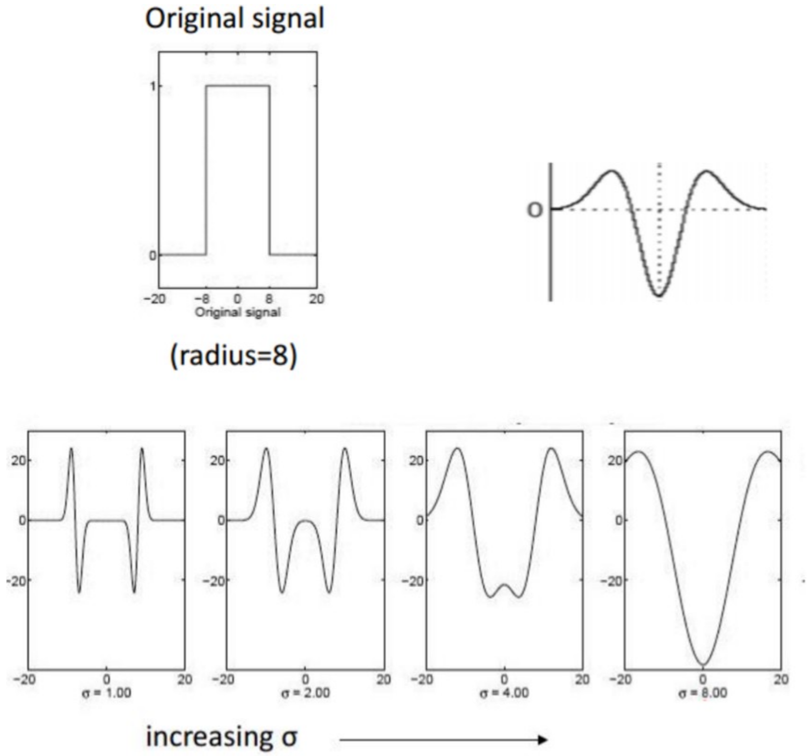
- It can be used for 2D blob detection! How?



[Source: F. Flores-Mangas]

Blob Detection – Laplacian of Gaussian

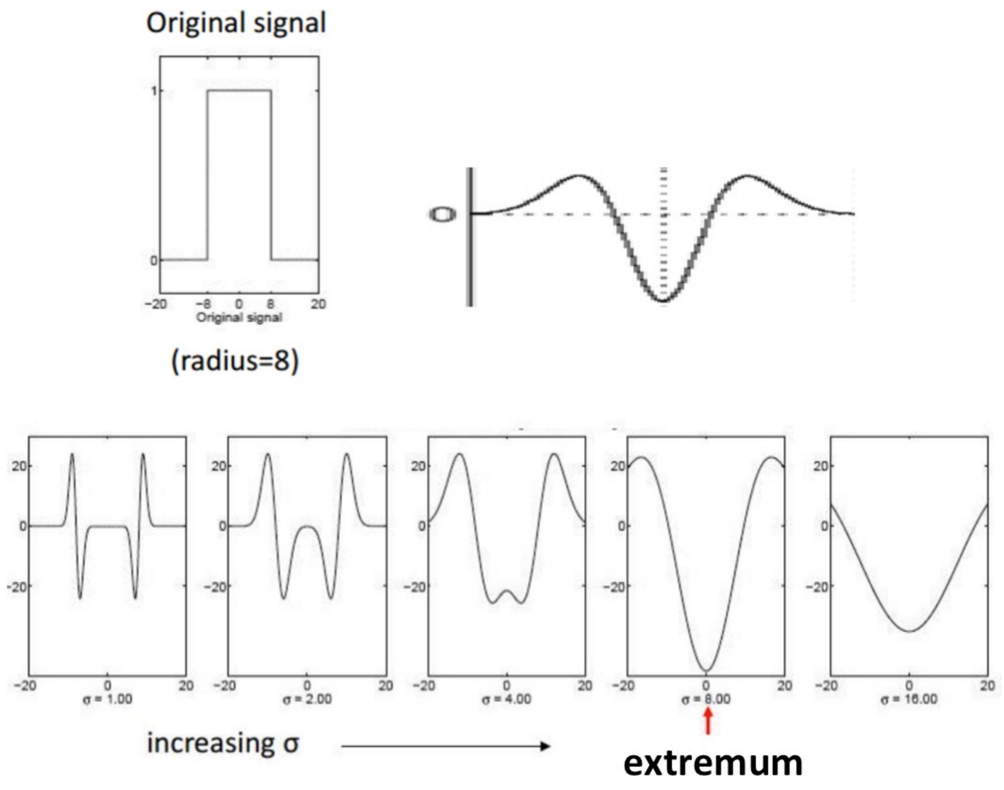
- It can be used for 2D blob detection! How?



[Source: F. Flores-Mangas]

Blob Detection – Laplacian of Gaussian

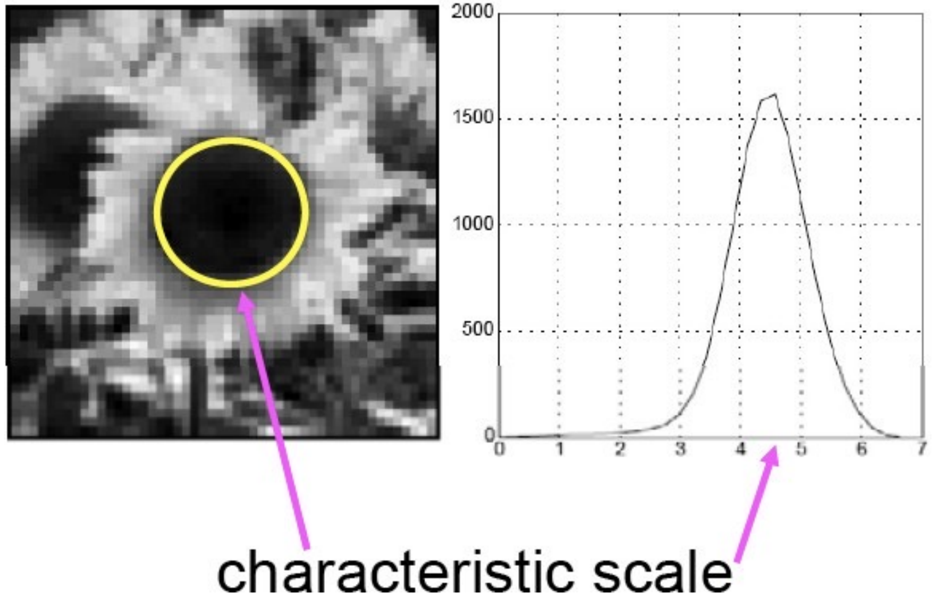
- It can be used for 2D blob detection! How?



[Source: F. Flores-Mangas]

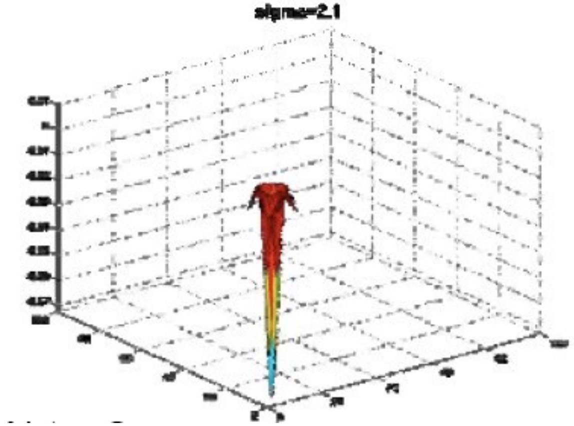
Characteristic Scale

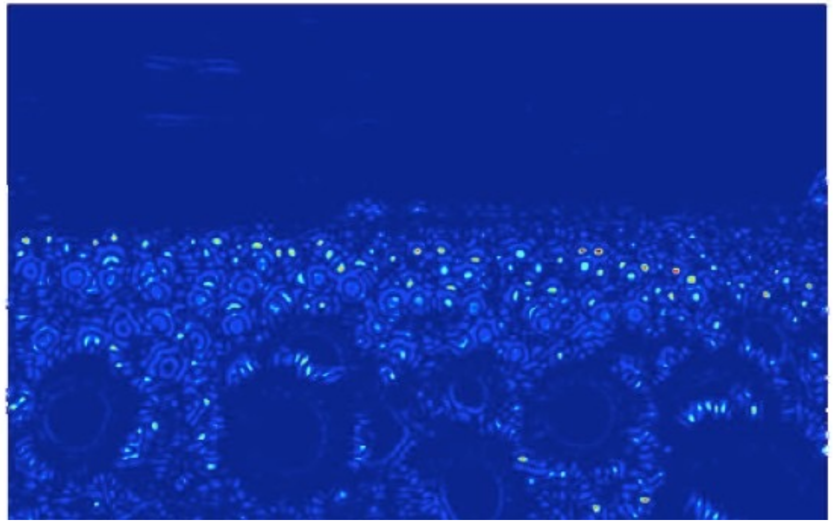
- We define the characteristic scale as the scale that produces peak (minimum or maximum) of the Laplacian response



[Source: S. Lazebnik]

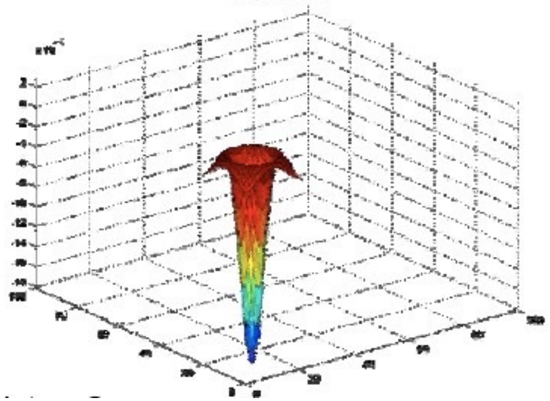
Example

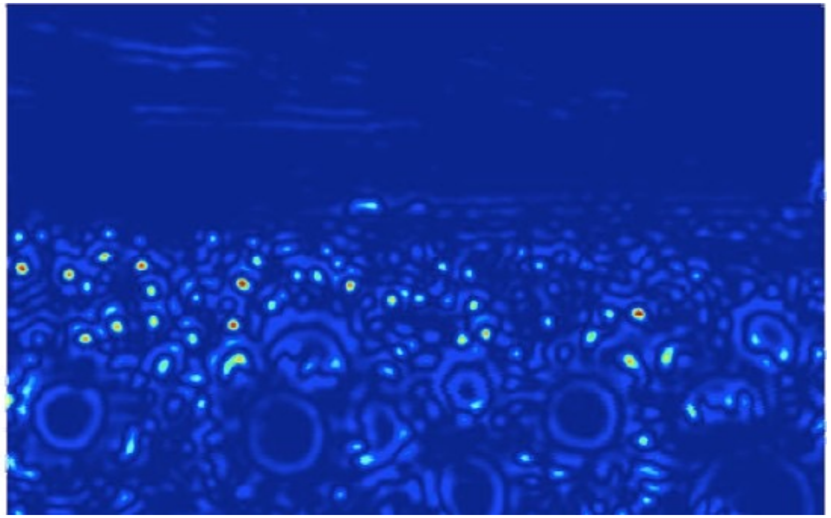




[Source: K. Grauman]

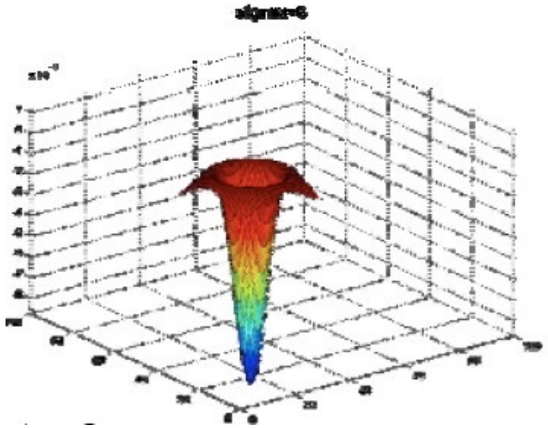
Example

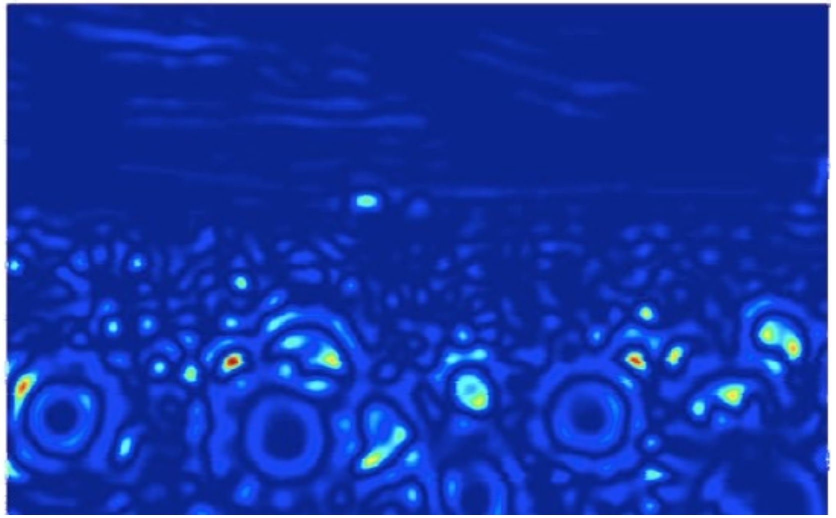




[Source: K. Grauman]

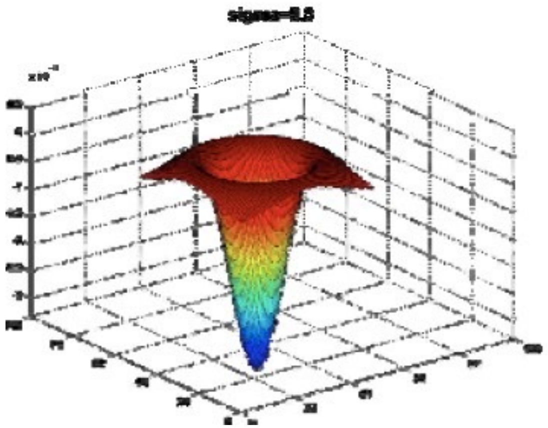
Example

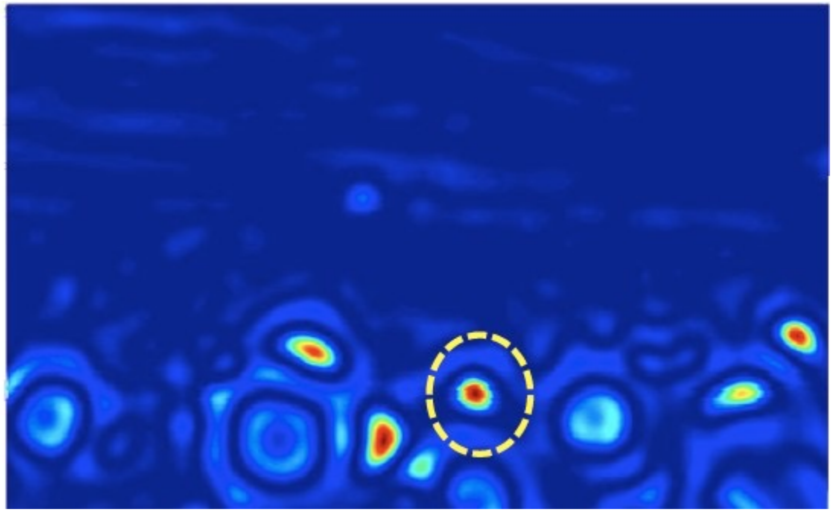




[Source: K. Grauman]

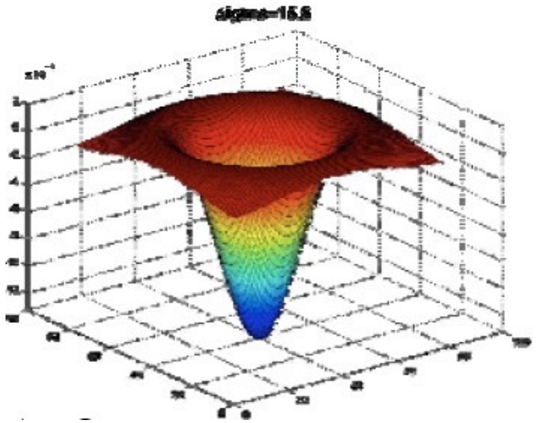
Example

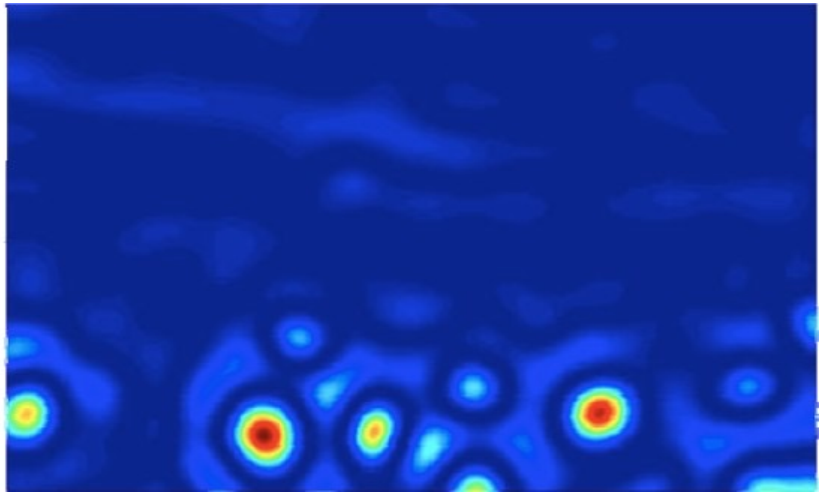




[Source: K. Grauman]

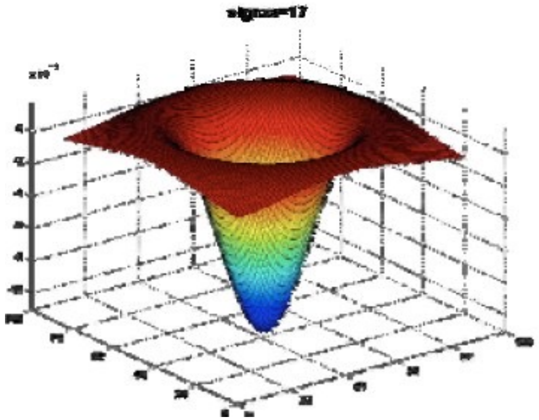
Example

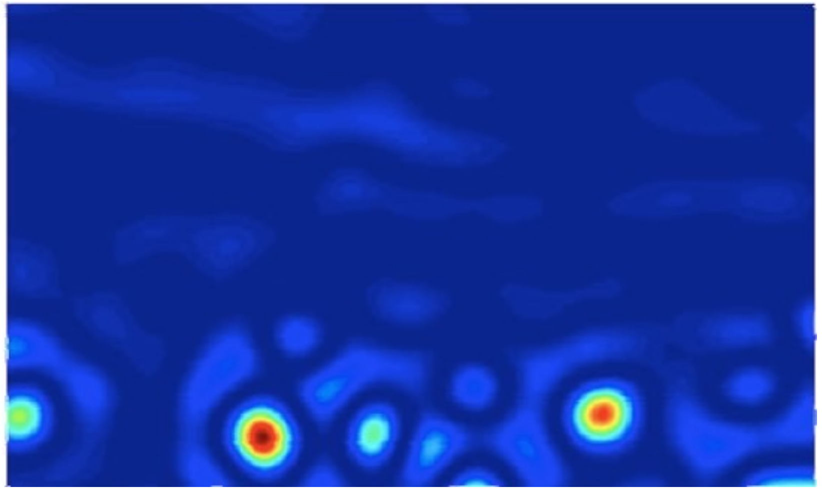




[Source: K. Grauman]

Example

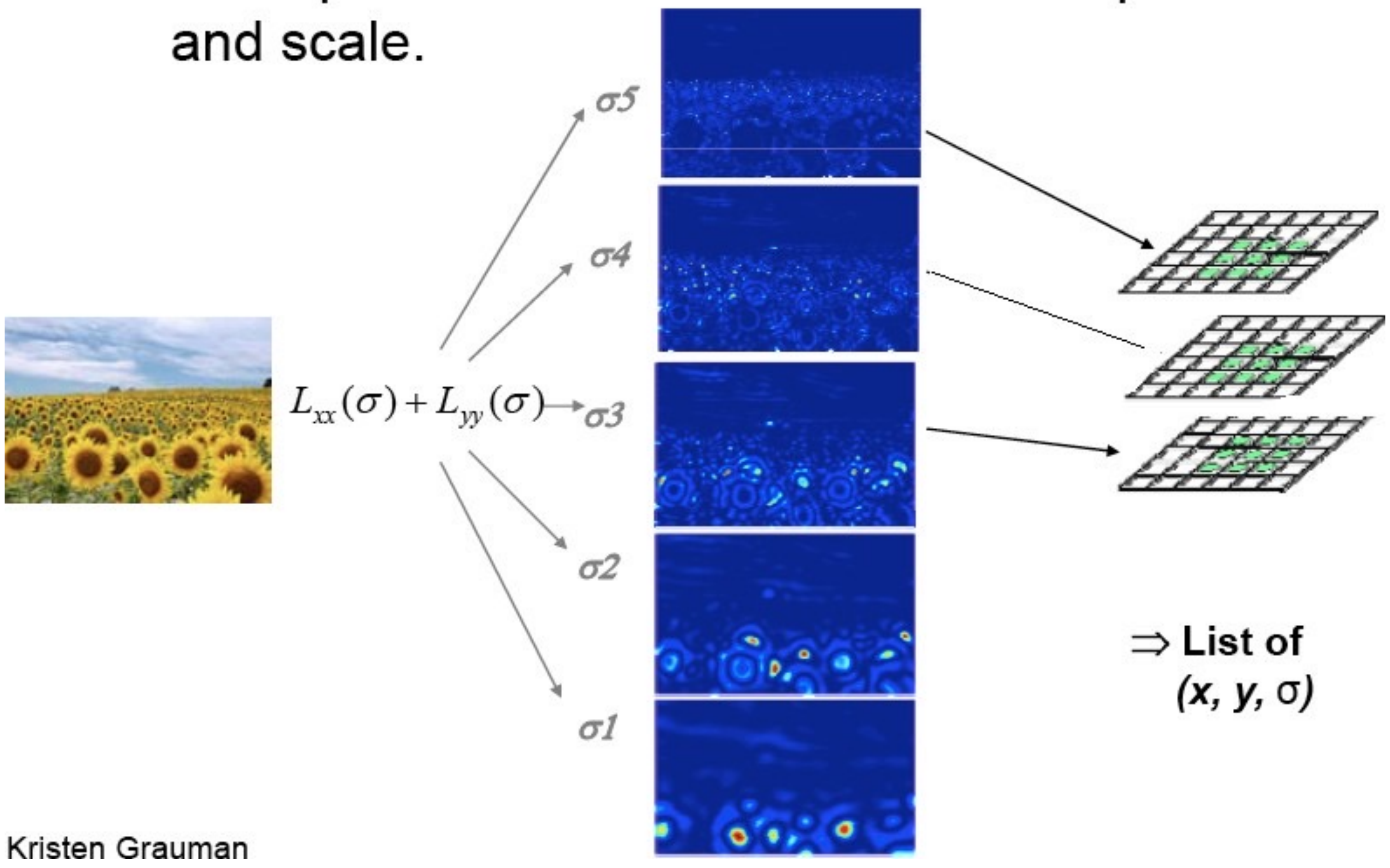




[Source: K. Grauman]

Scale Invariant Interest Points

Interest points are local maxima in both position and scale.



[Source: K. Grauman]

Example

[Source: S. Lazebnik]

Blob Detection – Laplacian of Gaussian

- That's nice. But can we do faster?
- Remember again the Laplacian of Gaussian:

$$\nabla_g^2(x, y, \sigma) = \frac{\partial^2 g(x, y, \sigma)}{\partial x^2} + \frac{\partial^2 g(x, y, \sigma)}{\partial y^2} \; where\ g\ is\ gaussian$$

Blob Detection – Laplacian of Gaussian

- That's nice. But can we do faster?
- Remember again the Laplacian of Gaussian:

$$\nabla_g^2(x, y, \sigma) = \frac{\partial^2 g(x, y, \sigma)}{\partial x^2} + \frac{\partial^2 g(x, y, \sigma)}{\partial y^2} \ where\ g\ is\ gaussian$$

$$\nabla_g^2(x, y, \sigma) = -\frac{1}{\pi\sigma^4}\left(1 - \frac{x^2 + y^2}{2\sigma^2}\right) \exp -\frac{x^2 + y^2}{2\sigma^2}$$

- Is this separable?
- Larger scale (σ), larger the filters (more work for convolution)
- Can we do it faster?

Approximate the Laplacian of Gaussian

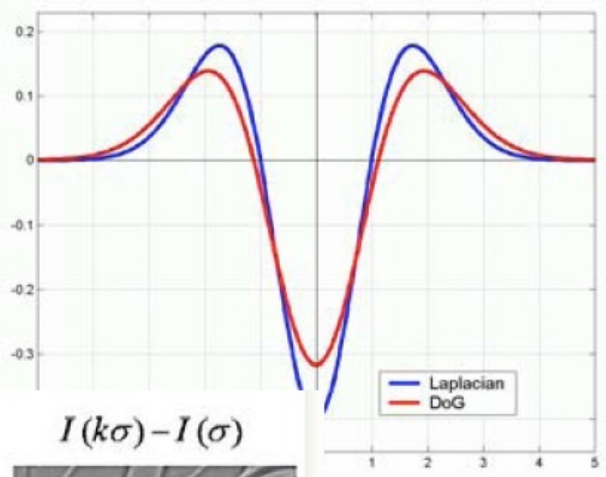
- We can approximate the Laplacian with a difference of Gaussians; and use separable convolution.

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

(Laplacian)

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

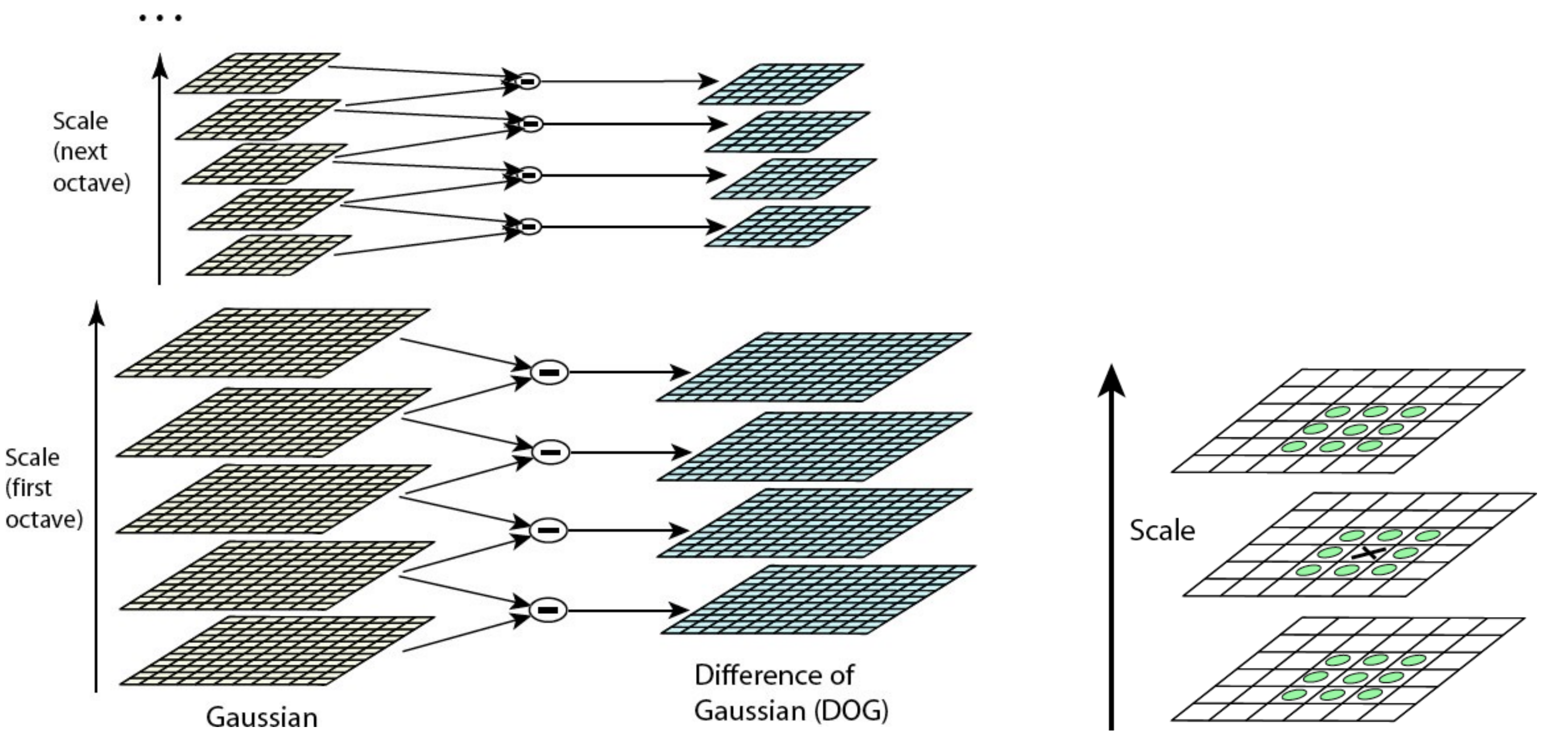
(Difference of Gaussians)



[Source: K. Grauman]

Lowe's DoG

- Lowe (2004) proposed computing a set of sub-octave Difference of Gaussian filters looking for 3D (space+scale) maxima in the resulting structure



[Source: R. Szeliski]

Lowe's DoG

- First compute a Gaussian image pyramid

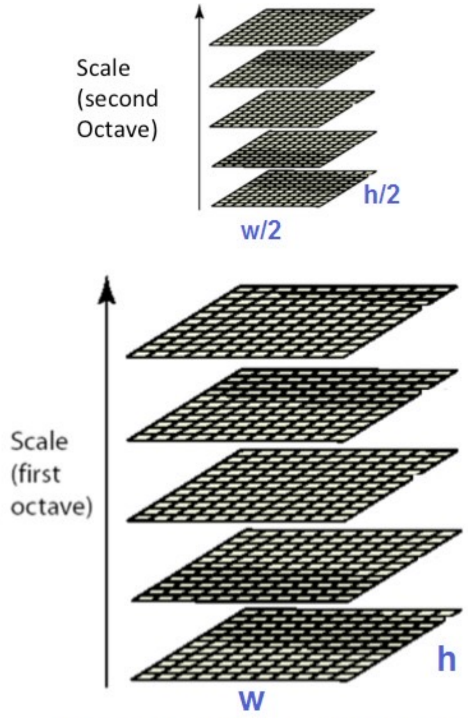
$$I_s = I * G_{k^s\sigma}$$

$$\vdots$$

$$I_2 = I * G_{k^2\sigma}$$

$$I_1 = I * G_{k\sigma}$$

$$I_0 = I * G_{\sigma}$$



Gaussian Smoothed Images

Each image is smoothed by a factor of k more than the image below

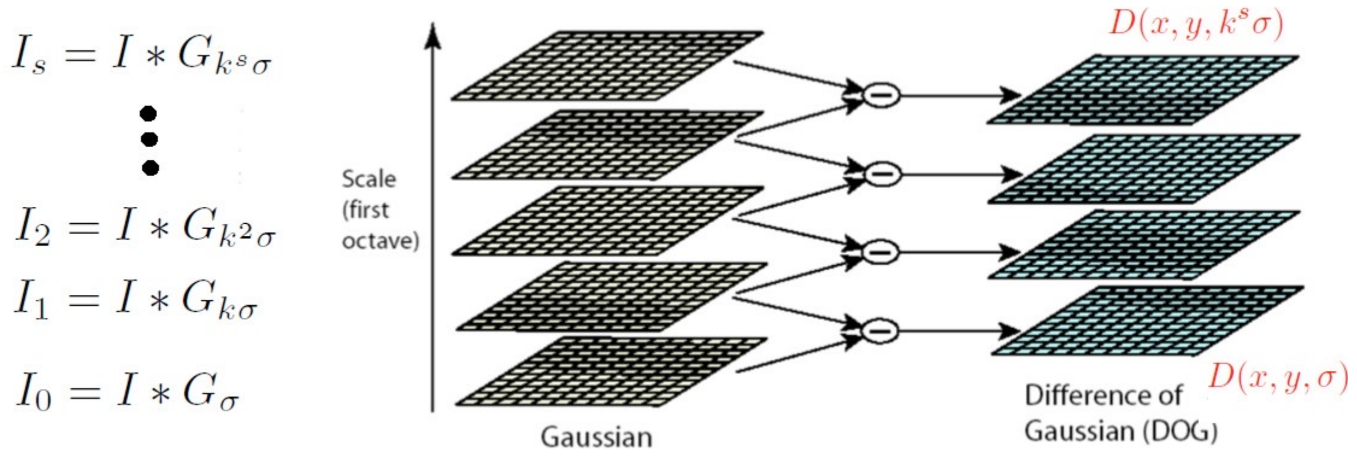
[Source: F. Flores-Mangas]

Lowe's DoG

- First compute a Gaussian image pyramid
- Compute Difference of Gaussians

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

$$\text{for } \rho = \{\sigma,\ k\sigma,\ k^2\sigma, \ldots, k^{s-1}\sigma\}, \quad k = 2^{1/s}$$



[Source: F. Flores-Mangas]

Lowe's DoG

- First compute a Gaussian image pyramid
- Compute Difference of Gaussians
- At every scale

[Source: F. Flores-Mangas]

Lowe's DoG

- First compute a Gaussian image pyramid
- Compute Difference of Gaussians
- At every scale
- Find local maxima in scale
- A bit of pruning of bad maxima and we're done!

[Source: F. Flores-Mangas]

Examples

Figure: Let's first try out some synthetic images

Examples

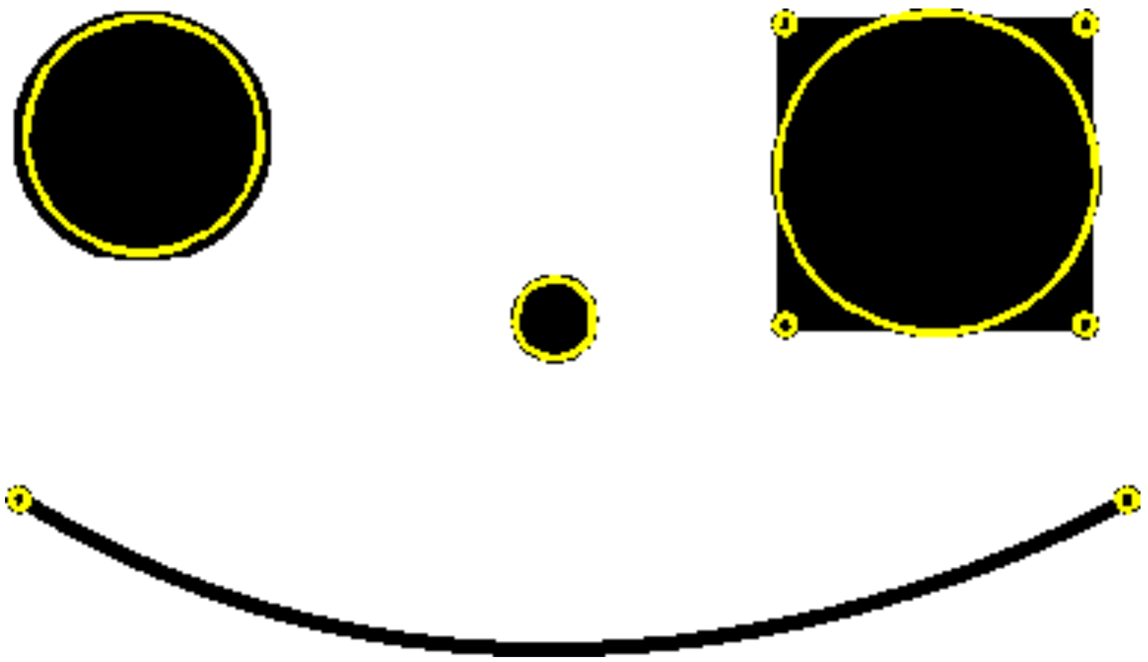


Figure: Detected interest points (kind of make sense)

Examples

Figure: Other roundy objects

Examples

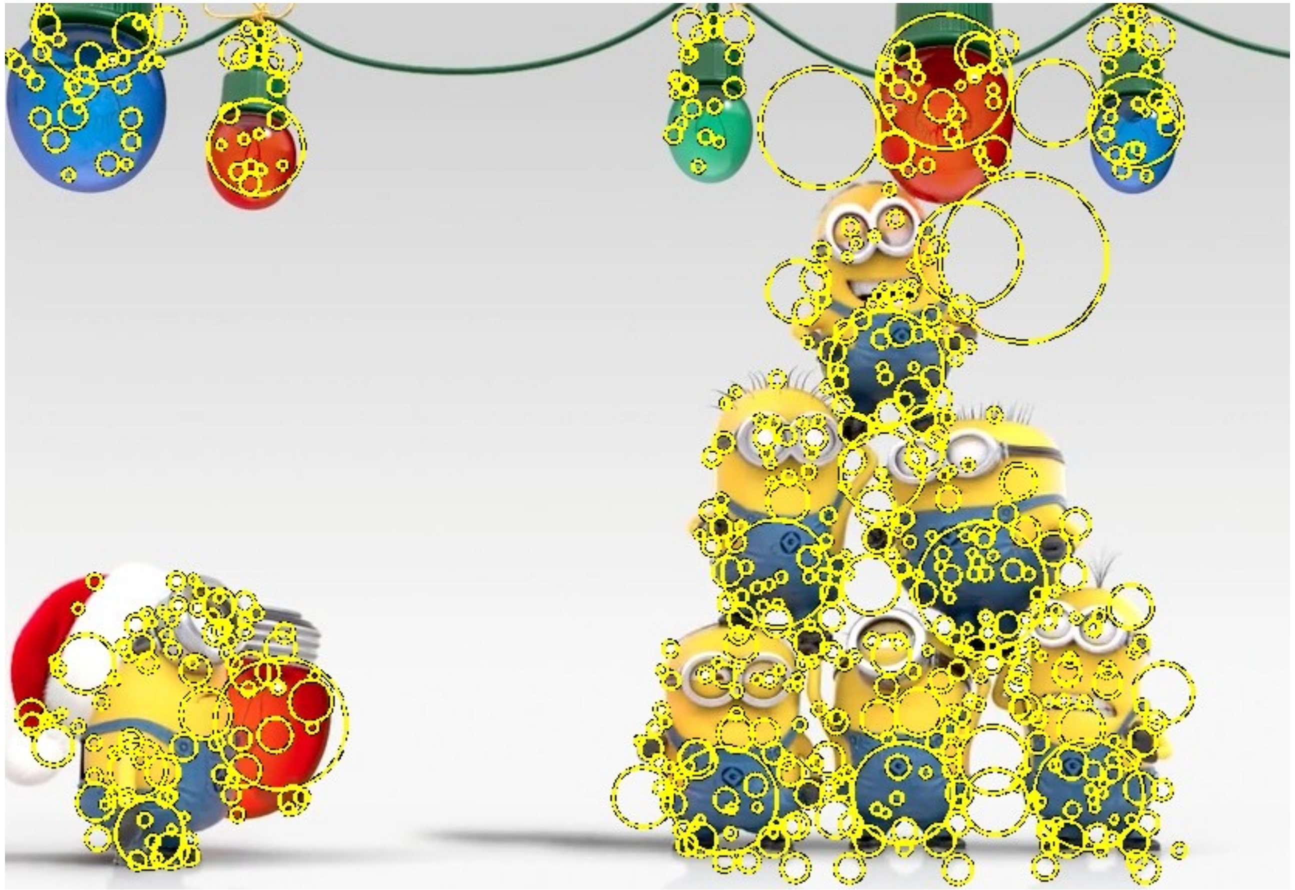


Figure: Detected interest points

Examples

Figure: Real images

Examples

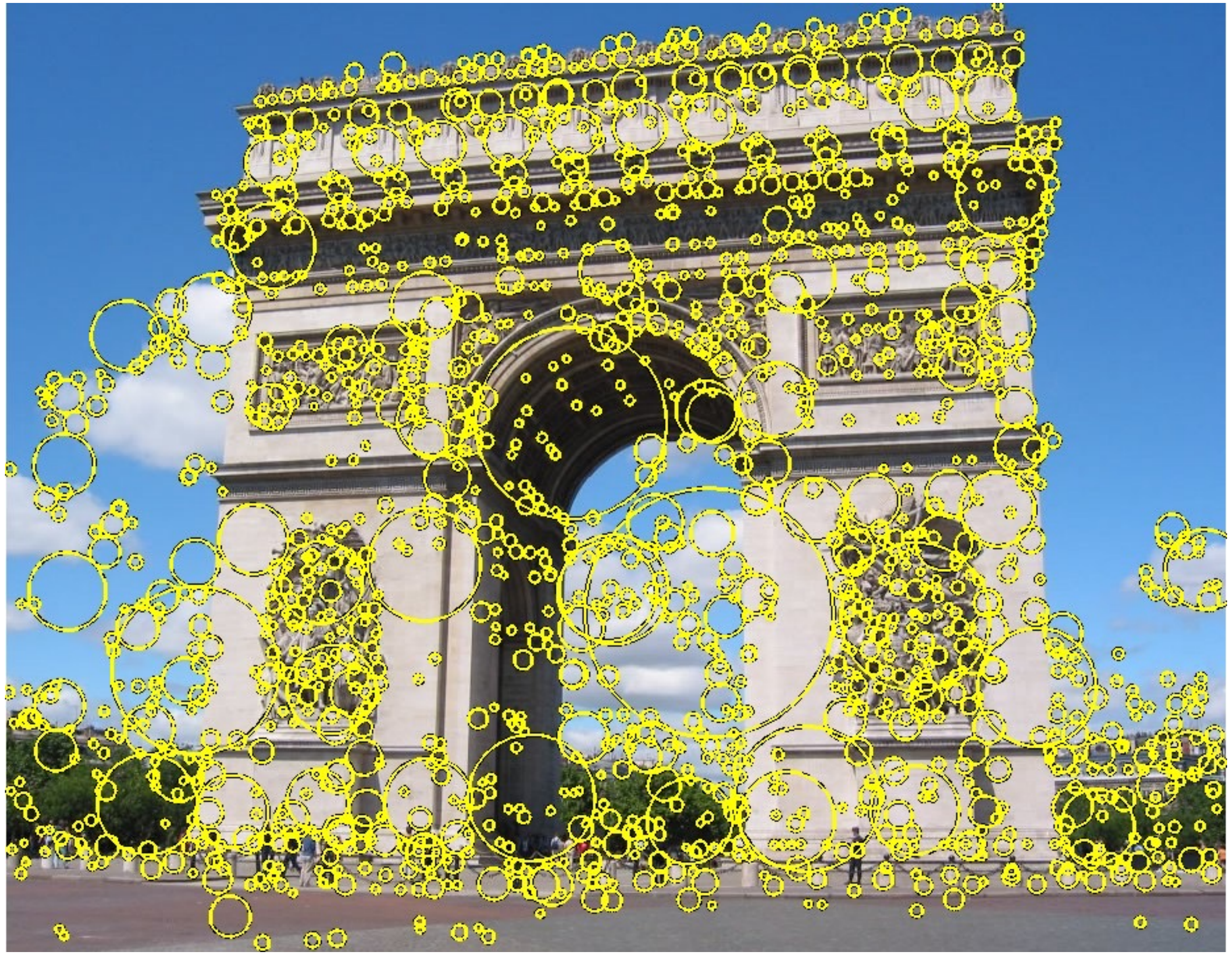
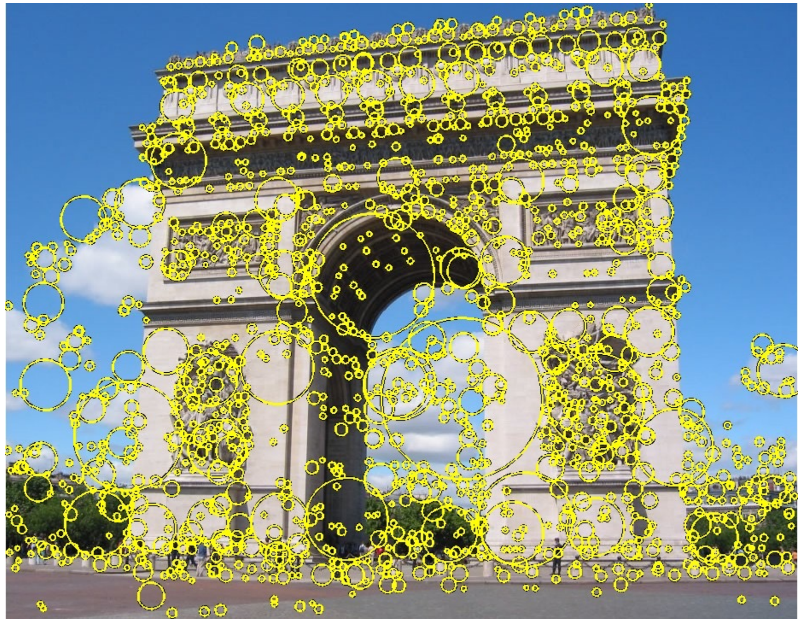
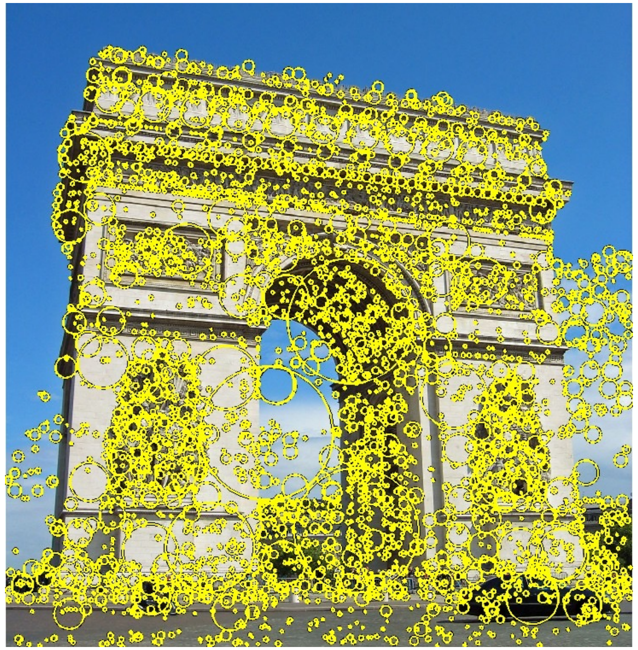


Figure: Detected interest points

Examples





Other Interest Point Detectors (Many Good Options!)

- Lindeberg: Laplacian of Gaussian
- Lowe: DoG (typically called the SIFT interest point detector)
- Mikolajczyk & Schmid: Hessian/Harris-Laplacian/Affine
- Tuyttelaars & Van Gool: EBR and IBR
- Matas: MSER
- Kadir & Brady: Salient Regions

Summary – Stuff You Should Know

- To match the same scene or object under different viewpoint, it's useful to first detect interest points (keypoints)

Summary – Stuff You Should Know

- To match the same scene or object under different viewpoint, it's useful to first detect interest points (keypoints)
- We looked at these interest point detectors:
  - Harris corner detector: translation and rotation but not scale invariant
  - Scale invariant interest points: Laplacian of Gaussians and Lowe's DoG

Summary – Stuff You Should Know

- To match the same scene or object under different viewpoint, it's useful to first detect interest points (keypoints)
- We looked at these interest point detectors:
  - Harris corner detector: translation and rotation but not scale invariant
  - Scale invariant interest points: Laplacian of Gaussians and Lowe's DoG
- Harris' approach computes I_X^2 , I_Y^2 and $I_X . I_y$ and blurs each one with a gaussian.

Summary – Stuff You Should Know

- To match the same scene or object under different viewpoint, it's useful to first detect interest points (keypoints)
- We looked at these interest point detectors:
  - Harris corner detector: translation and rotation but not scale invariant
  - Scale invariant interest points: Laplacian of Gaussians and Lowe's DoG
- Harris' approach computes I_X^2 , I_Y^2 and $I_X . I_y$ and blurs each one with a gaussian.
  - Denote with: $A = g * I_x^2$, $B = g * (I_x I_y)$ and $C = g * I_y^2$. Then
  - $M_{xy} = \begin{bmatrix} A\,(x,y) & B\,(x,y) \\ B\,(x,y) & C\,(x,y) \end{bmatrix}$ characterizes the shape of E_{wssd} for a window around (x,y).

Summary – Stuff You Should Know

- To match the same scene or object under different viewpoint, it's useful to first detect interest points (keypoints)
- We looked at these interest point detectors:
  - Harris corner detector: translation and rotation but not scale invariant
  - Scale invariant interest points: Laplacian of Gaussians and Lowe's DoG
- Harris' approach computes I_X^2 , I_Y^2 and $I_X . I_y$ and blurs each one with a gaussian.
  - Denote with: $A = g * I_x^2$, $B = g * (I_x I_y)$ and $C = g * I_y^2$. Then
  - $M_{xy} = \begin{bmatrix} A\,(x,y) & B\,(x,y) \\ B\,(x,y) & C\,(x,y) \end{bmatrix}$ characterizes the shape of E_{wssd} for a window around (x,y).
  - Compute "cornerness" score for each (x, y) as
    $R(x, y) = \det(M) - \alpha\, \mathrm{trace}(M)^2$. Find $R(x, y) >$ threshold and do non-maxima suppression to find corners.

Summary – Stuff You Should Know

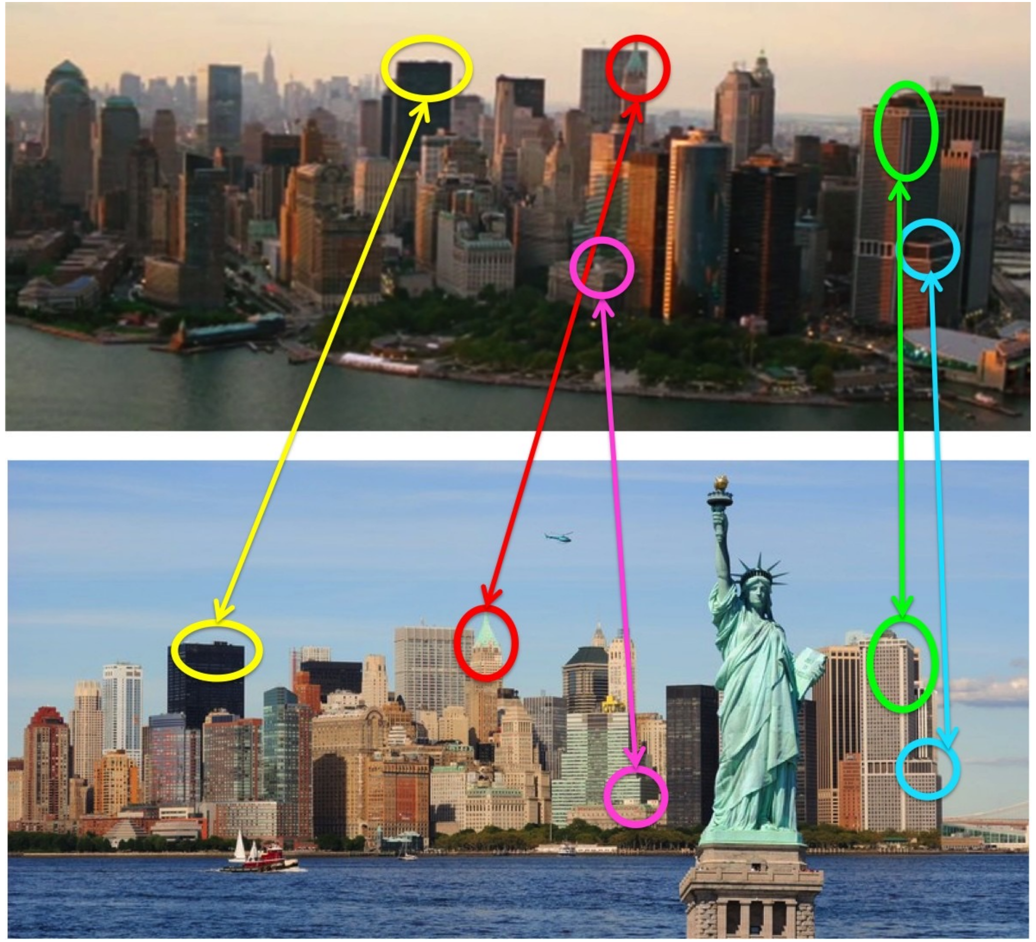
- To match the same scene or object under different viewpoint, it's useful to first detect interest points (keypoints)
- We looked at these interest point detectors:
  - Harris corner detector: translation and rotation but not scale invariant
  - Scale invariant interest points: Laplacian of Gaussians and Lowe's DoG
- Harris' approach computes I_X^2 , I_Y^2 and $I_X . I_y$ and blurs each one with a gaussian.
  - Denote with: $A = g * I_x^2$, $B = g * (I_x I_y)$ and $C = g * I_y^2$. Then
  - $M_{xy} = \begin{bmatrix} A\,(x,y) & B\,(x,y) \\ B\,(x,y) & C\,(x,y) \end{bmatrix}$ characterizes the shape of E_{wssd} for a window around (x,y).
  - Compute "cornerness" score for each (x, y) as $R(x, y) = \det(M) - \alpha\, \mathrm{trace}(M)^2$. Find $R(x, y) >$ threshold and do non-maxima suppression to find corners.
- Lowe's approach creates a Gaussian pyramid with "s" blurring levels per octave, computes difference between consecutive levels, and finds local extrema in space and scale

Overview

- motivation
- scale invariant keypoint detection
- learned keypoint detection
- image features
- matching

Let's Remember How Interest Point Stuff Started

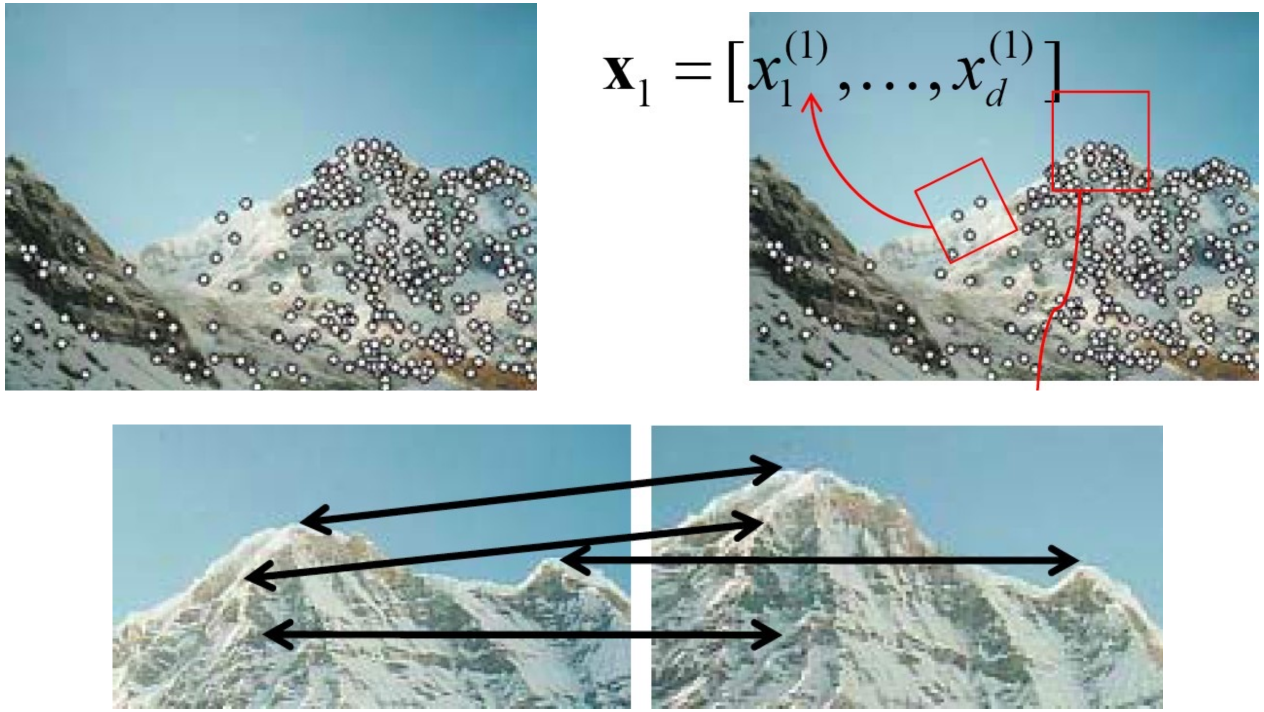
- Which city is in the photo above?



New York City

Local Features

- **Detection**: Identify the interest points.
- **Description**: Extract **feature vector** descriptor around each interest point.
- **Matching**: Determine correspondence between descriptors in two views.



[Source: K. Grauman]

SIFT Interest Points

- Works pretty well in variety of settings

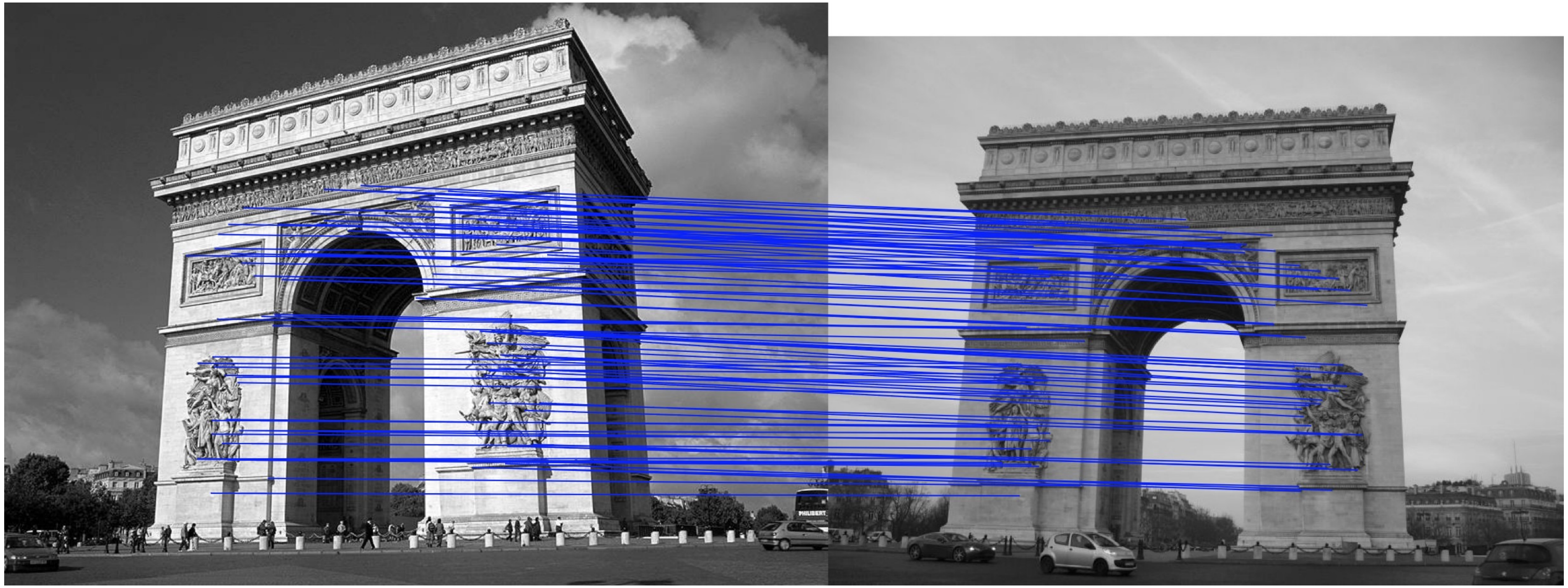


Figure: Lowe's interest point detector finds scale-invariant points that can be reliably matched across different images. (We will talk about how to do matching soon)

SIFT Interest Points

- Works pretty well in variety of settings

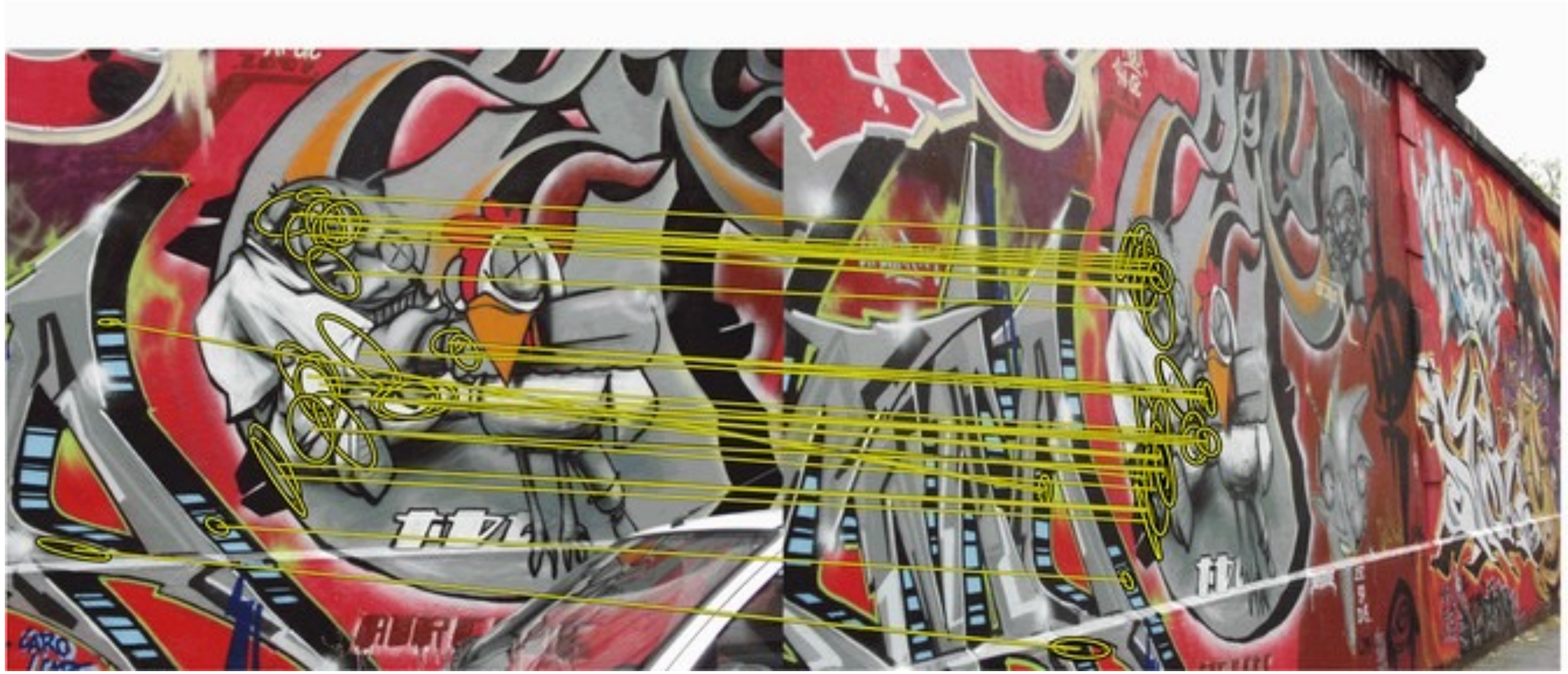


Figure: Lowe's interest point detector finds scale-invariant points that can be reliably matched across different images. (We will talk about how to do matching soon)

SIFT Interest Points

- Works pretty well in variety of settings

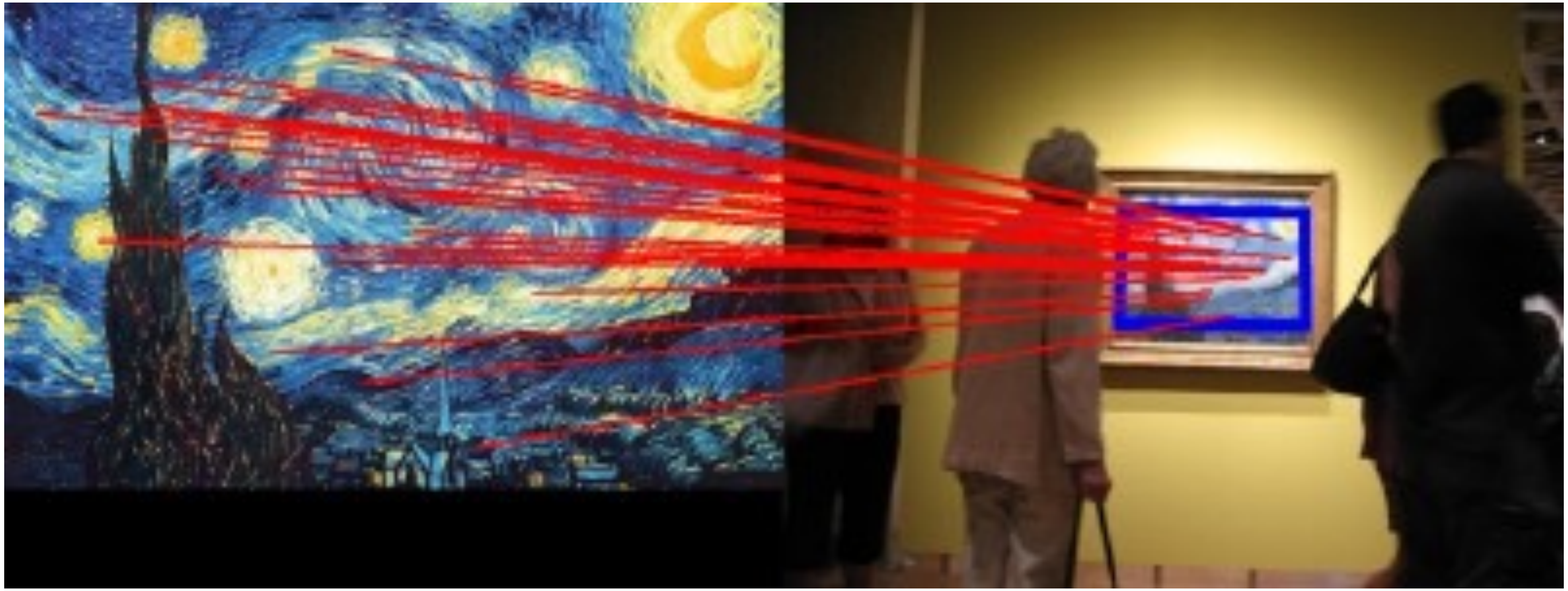


Figure: Lowe's interest point detector finds scale-invariant points that can be reliably matched across different images. (We will talk about how to do matching soon)

SIFT Interest Points

- What about in different lighting/weather conditions?




SIFT Interest Points

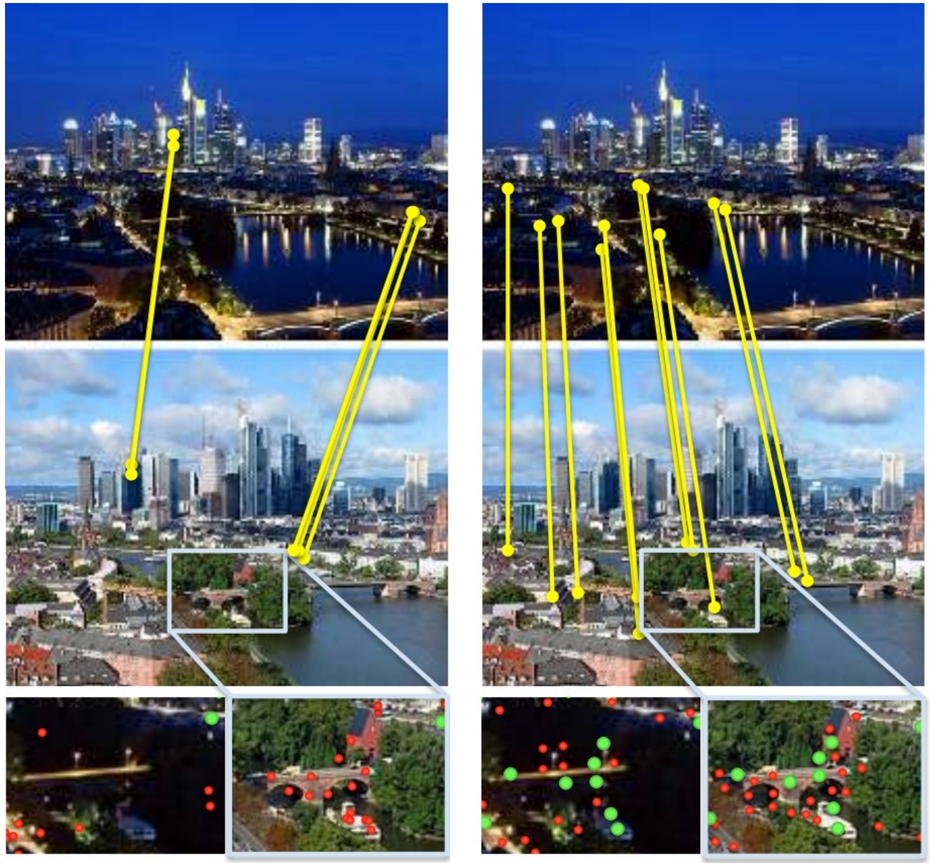
- Fails in very different lighting conditions

Figure: Green point(s) are repeatable interest points, red are non-repeatable

[Pic from: Y. Verdie, K. M. Yi, P. Fua and V. Lepetit. TILDE: A Temporally Invariant Learned DEtector. CVPR'15]

SIFT Interest Points

- Can we use Machine Learning to detect interest points more reliably?



SIFT Learned Interest Point Detector?

[Pic from: Y. Verdie, K. M. Yi, P. Fua and V. Lepetit. TILDE: A Temporally Invariant Learned DEtector. CVPR'15]

Training Data

- What can we use?

Training Data

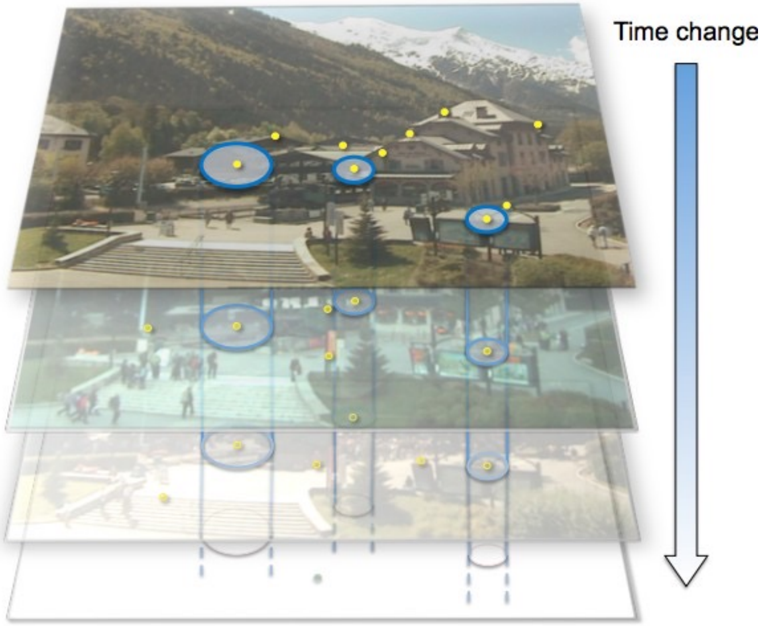
- What can we use? Data from Webcam

Training Data

- Now that we have training images, how shall we train the detector?

Training the Detector

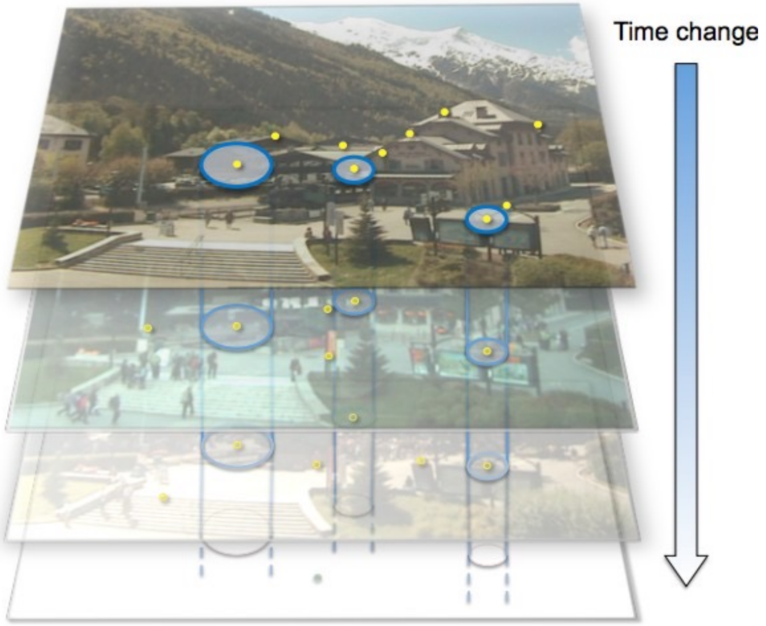
- Detect e.g. SIFT Interest Points in images across time
- Keep only those that are repeatable across time.
- These are our (super reliable) positive training examples. What about negative examples?



(a) Stack of training images

Training the Detector

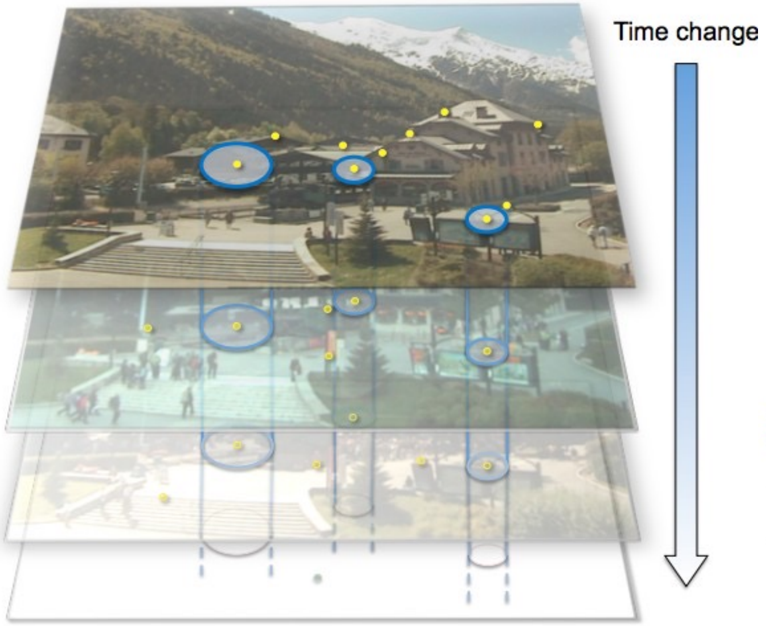
- Detect e.g. SIFT Interest Points in images across time
- Keep only those that are repeatable across time.
- These are our (super reliable) positive training examples. What about negative examples? All other points with some distance wrt positive points



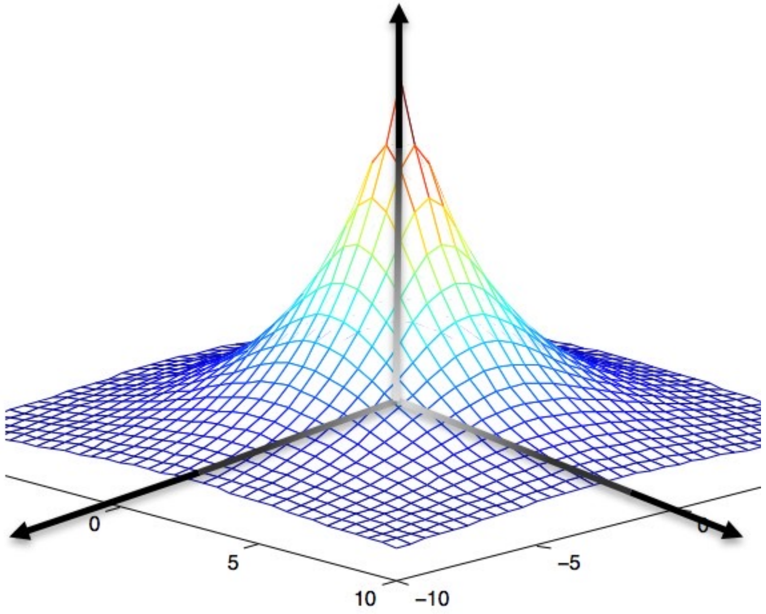
(a) Stack of training images

Training the Detector

- Detect e.g. SIFT Interest Points in images across time
- Keep only those that are repeatable across time.
- These are our (super reliable) positive training examples. What about negative examples? All other points with some distance wrt positive points
- Take a patch around each point, extract some features on it. Train a classifier/regressor



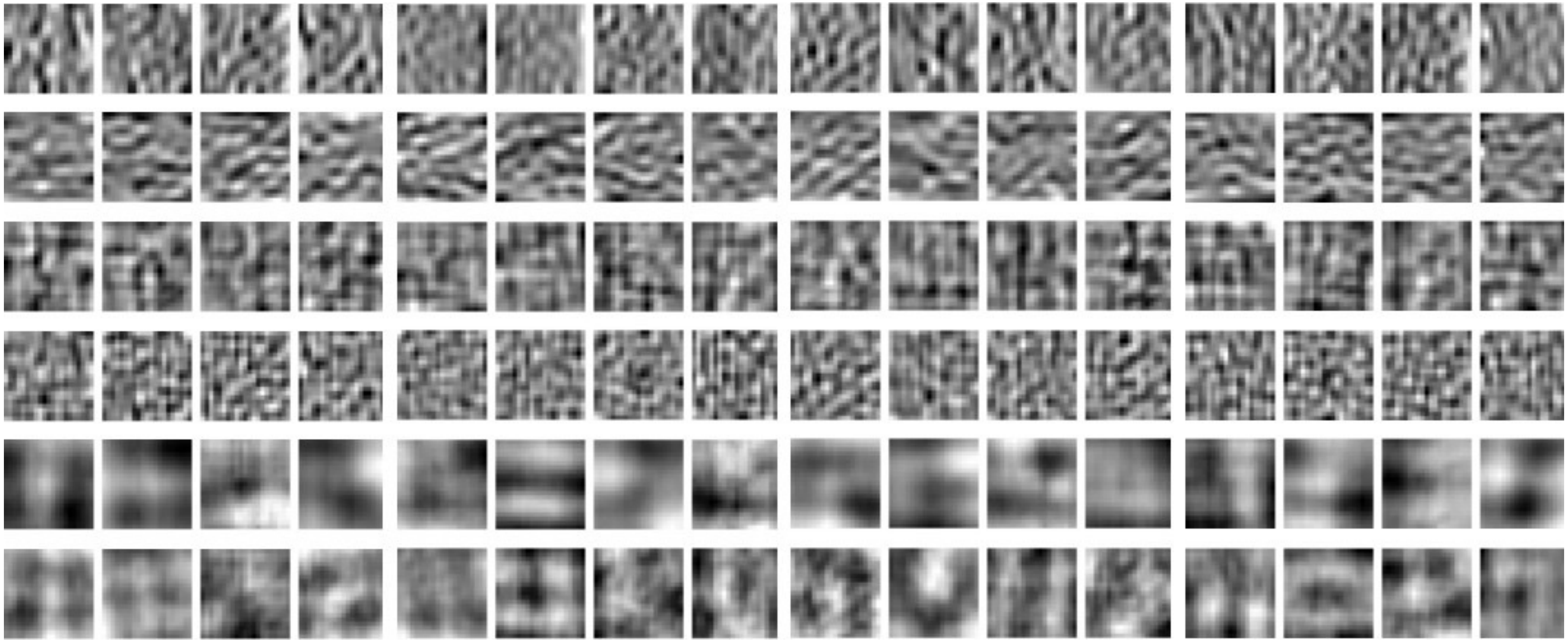
(a) Stack of training images



(b) Desired response on positive samples

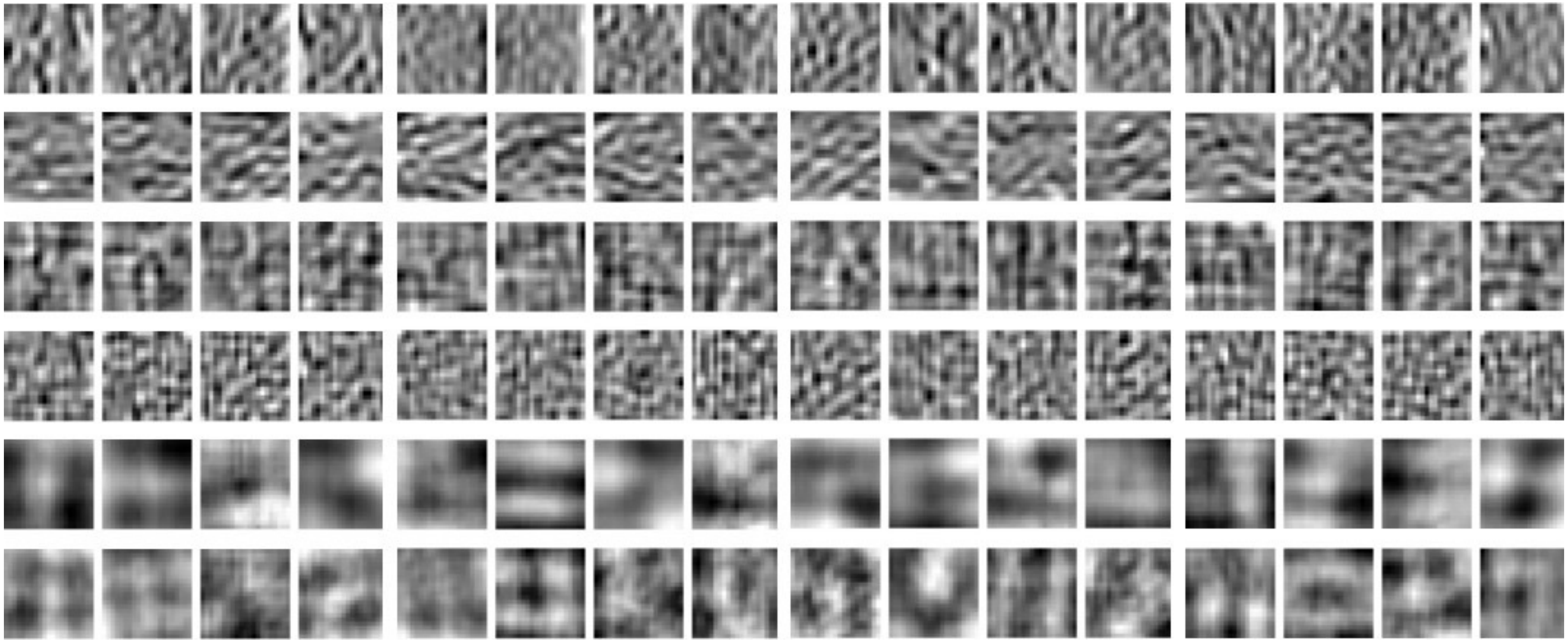
Trained Filters

- Remember from the lecture where we trained a classifier to detect edges: If we train a linear classifier on a patch, it can be seen as a filter



Trained Filters

- Remember from the lecture where we trained a classifier to detect edges: If we train a linear classifier on a patch, it can be seen as a filter



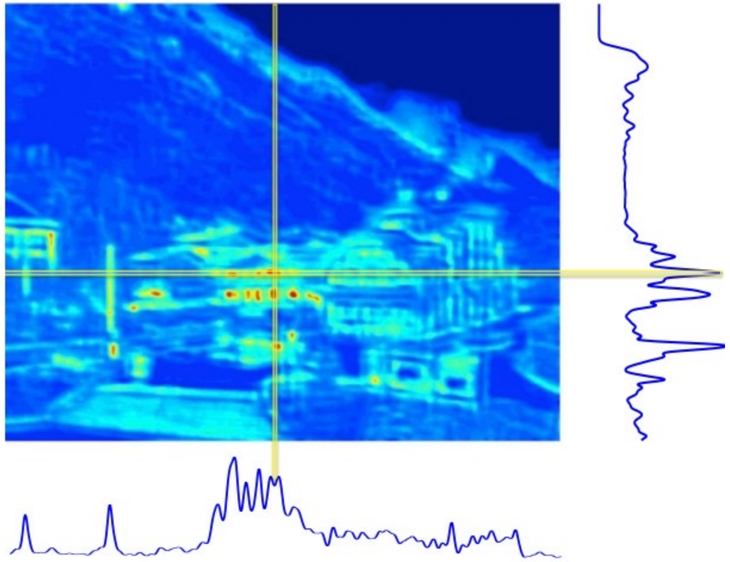
Tiny lesson learned: Sometime our intermediate results (filters in this case) don’t look interpretable at all, but they still do the job

Using the Learned Interest Point Detector

- Now that we trained our detector, how can we use it on new images?

Using the Learned Interest Point Detector

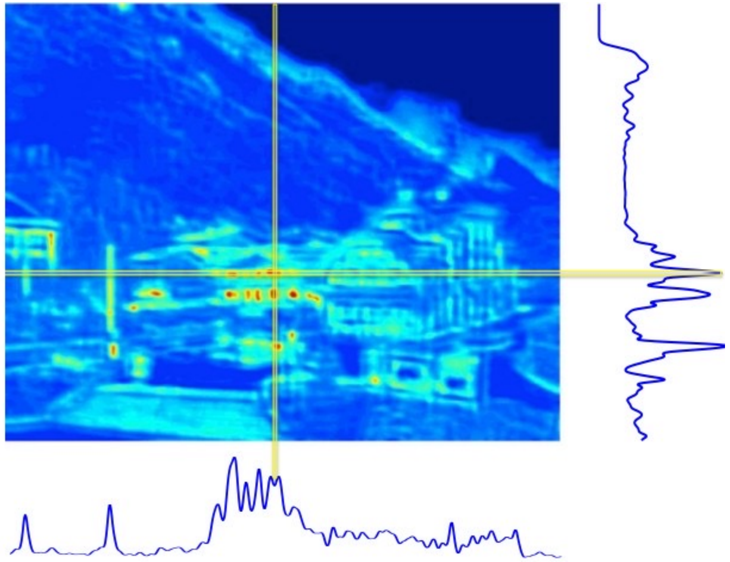
- Apply our filter on each image patch (convolution, if it's a linear classifier)



(c) Regressor response for a new image

Using the Learned Interest Point Detector

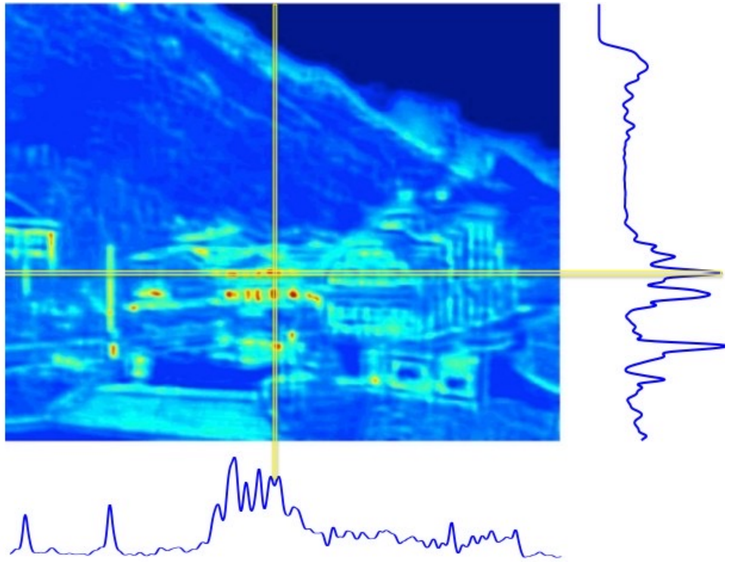
- Apply our filter on each image patch (convolution, if it's a linear classifier)
- This has response everywhere. How can we find the actual interest points?



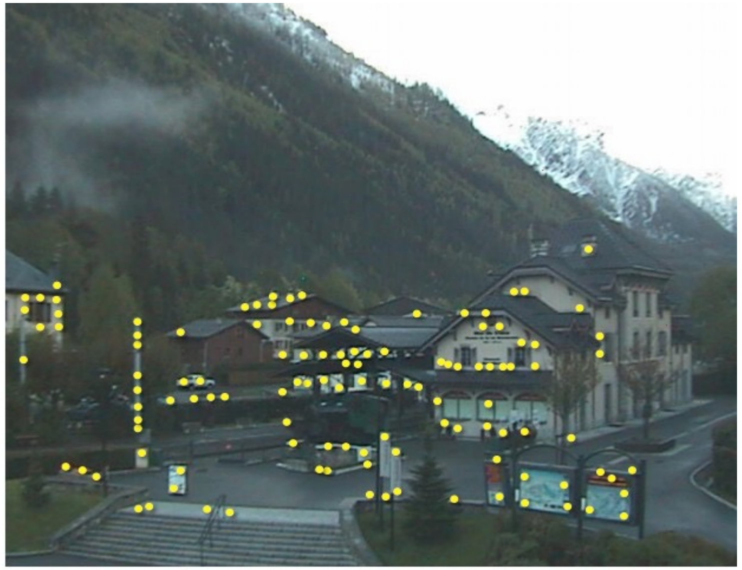
(c) Regressor response for a new image

Using the Learned Interest Point Detector

- Apply our filter on each image patch (convolution, if it's a linear classifier)
- This has response everywhere. How can we find the actual interest points?
- Non-maxima suppression (keep only points that are local maxima)



(c) Regressor response for a new image



(d) Keypoints detected in the new image

Results

- Visually check how well we can now match with new interest points

(a) Original images (b) SIFT (c) SURF (d) FAST-9 (e) Our keypoints

- SIFT, SURF are hand-designed interest point detectors
- FAST is trained to detect corners fast: First employs a slow method to detect corners, then trains decision trees to detect them really fast

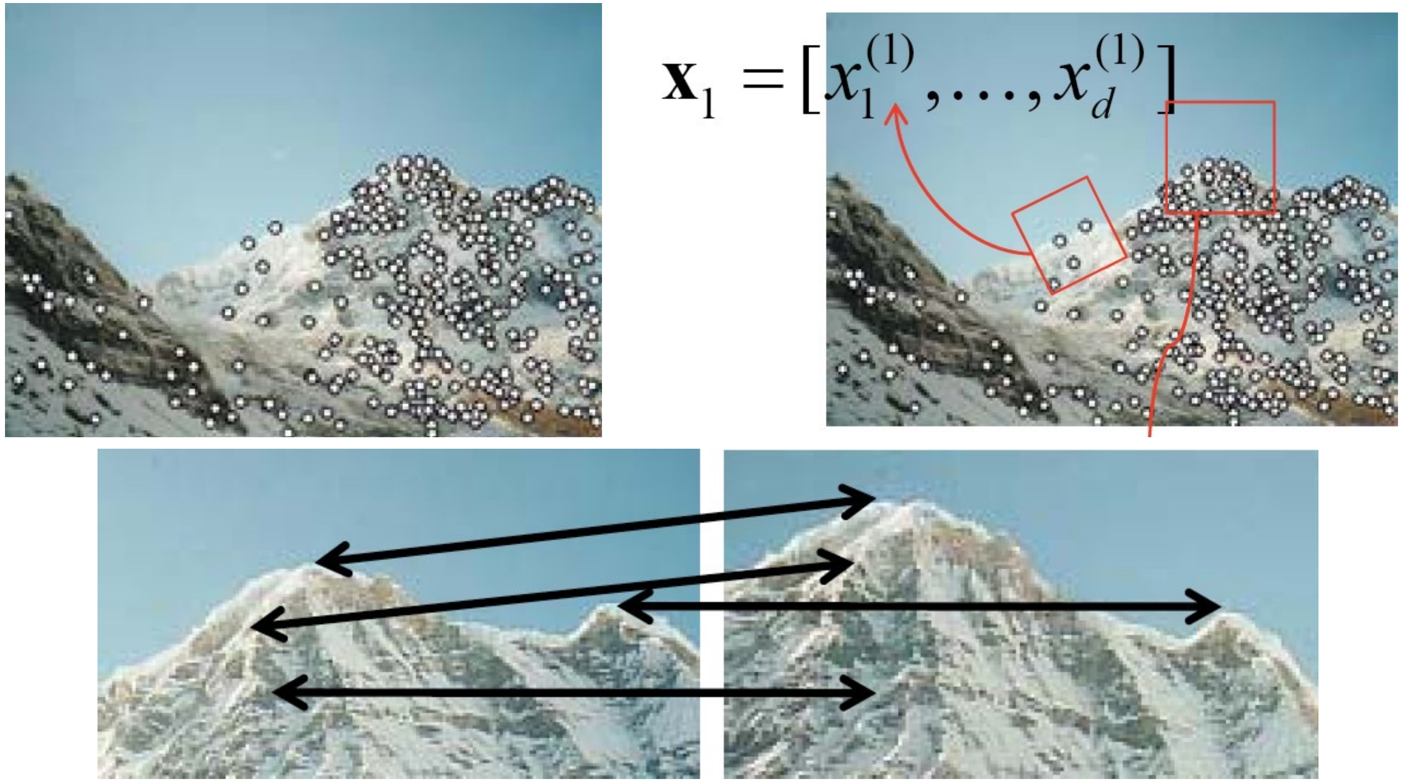
[E. Rosten and T. Drummond. Machine Learning for High Speed Corner Detection. ECCV 2006]
[Verdie et al. TILDE: A Temporally Invariant Learned Detector. CVPR 2015]

Overview

- motivation
- scale invariant keypoint detection
- learned keypoint detection
- image features
- matching

Local Features

- Detection: Identify the interest points.
- **Description: Extract a feature descriptor around each interest point.**
- Matching: Determine correspondence between descriptors in two views.



[Source: K. Grauman]

The Ideal Feature Descriptor

- Repeatable: Invariant to rotation, scale, photometric variations

The Ideal Feature Descriptor

- Repeatable: Invariant to rotation, scale, photometric variations
- Distinctive: We will need to match it to lots of images/objects!

The Ideal Feature Descriptor

- Repeatable: Invariant to rotation, scale, photometric variations
- Distinctive: We will need to match it to lots of images/objects!
- Compact: Should capture rich information yet not be too high-dimensional (otherwise matching will be slow)

The Ideal Feature Descriptor

- Repeatable: Invariant to rotation, scale, photometric variations
- Distinctive: We will need to match it to lots of images/objects!
- Compact: Should capture rich information yet not be too high-dimensional (otherwise matching will be slow)
- Efficient: We would like to compute it (close-to) real-time

Invariances

e.g. scale,
translation,
rotation

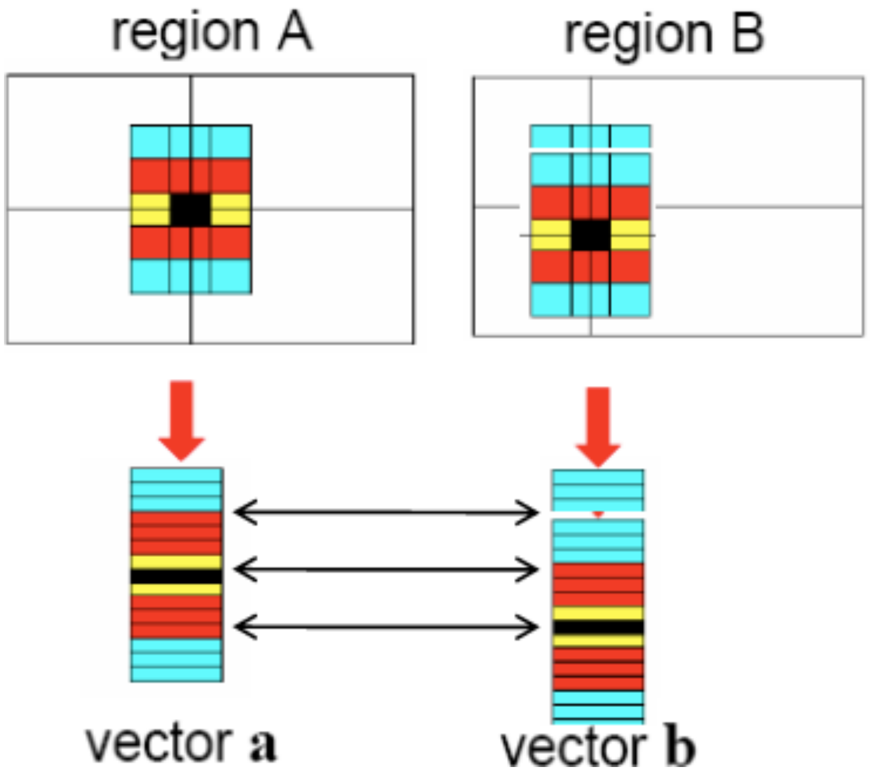
[Source: T. Tuytelaars]

Invariances

[Source: T. Tuytelaars]

What If We Just Took Pixels?

- The simplest way is to write down the list of intensities to form a feature vector, and normalize them (i.e., mean 0, variance 1).
- Why normalization?
- But this is very sensitive to even small shifts, rotations and any affine transformation.



[Source: K. Grauman]

Tons Of Better Options

- SIFT
- PCA-SIFT
- GLOH
- HOG
- SURF
- DAISY
- LBP
- Shape Contexts
- Color Histograms

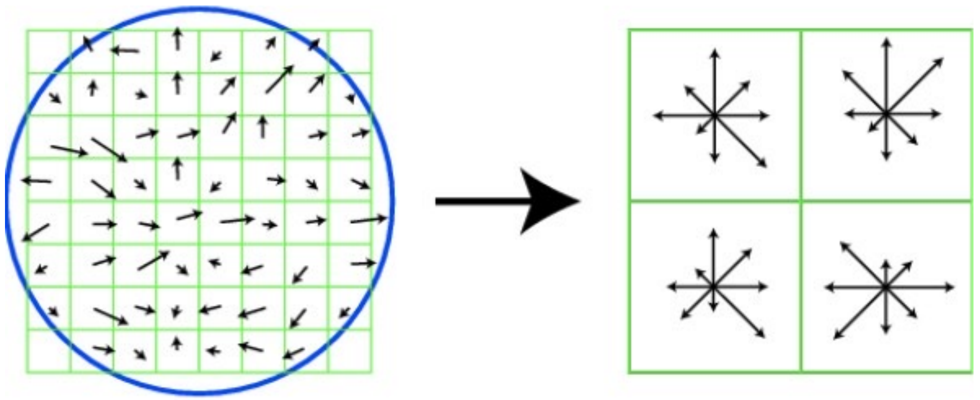
Tons Of Better Options

- SIFT TODAY
- PCA-SIFT
- GLOH
- HOG
- SURF
- DAISY
- LBP
- Shape Contexts
- Color Histograms

SIFT Descriptor [Lowe 2004]

- SIFT stands for Scale Invariant Feature Transform
- Invented by David Lowe, who also did DoG scale invariant interest points
- Actually in the same paper, which you should read:

David G. Lowe
Distinctive image features from scale-invariant keypoints
International Journal of Computer Vision, 2004
Paper: http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf



(a) image gradients (b) keypoint descriptor

SIFT Descriptor

- Our scale invariant interest point detector gives scale ρ for each keypoint

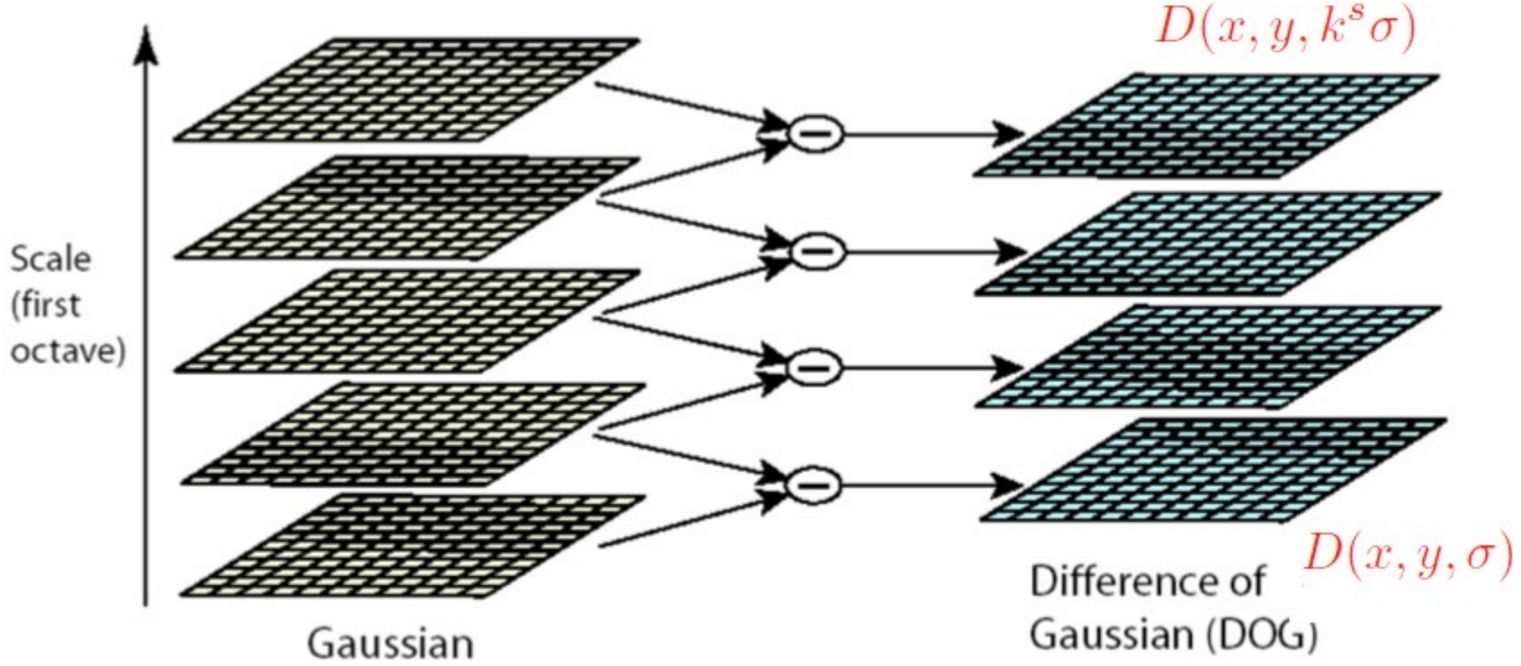
$$I_s = I * G_{k^s\sigma}$$

$$\vdots$$

$$I_2 = I * G_{k^2\sigma}$$

$$I_1 = I * G_{k\sigma}$$

$$I_0 = I * G_{\sigma}$$



[Adopted from: F. Flores-Mangas]

SIFT Descriptor

- For each keypoint, we take the Gaussian-blurred image at corresponding scale ρ

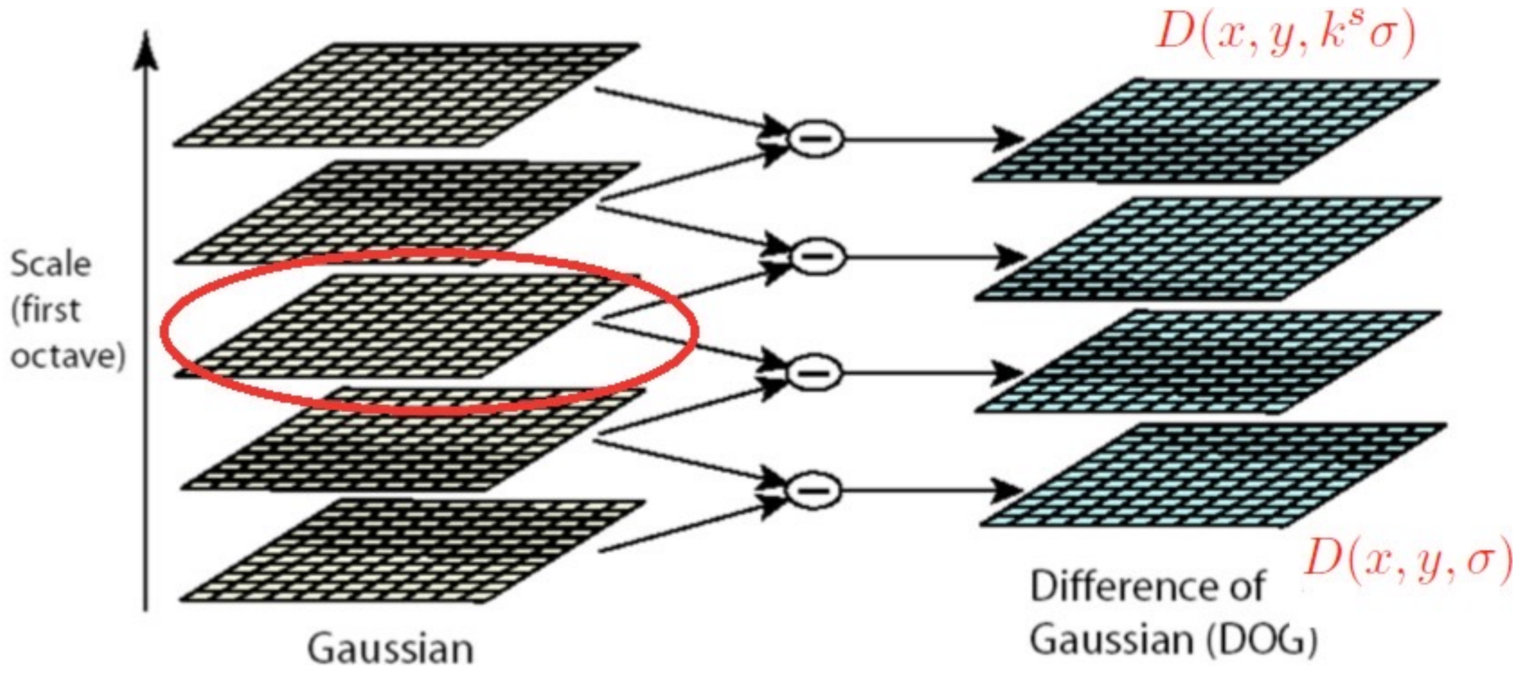
$$I_s = I * G_{k^s\sigma}$$

$$\vdots$$

$$I_2 = I * G_{k^2\sigma}$$

$$I_1 = I * G_{k\sigma}$$

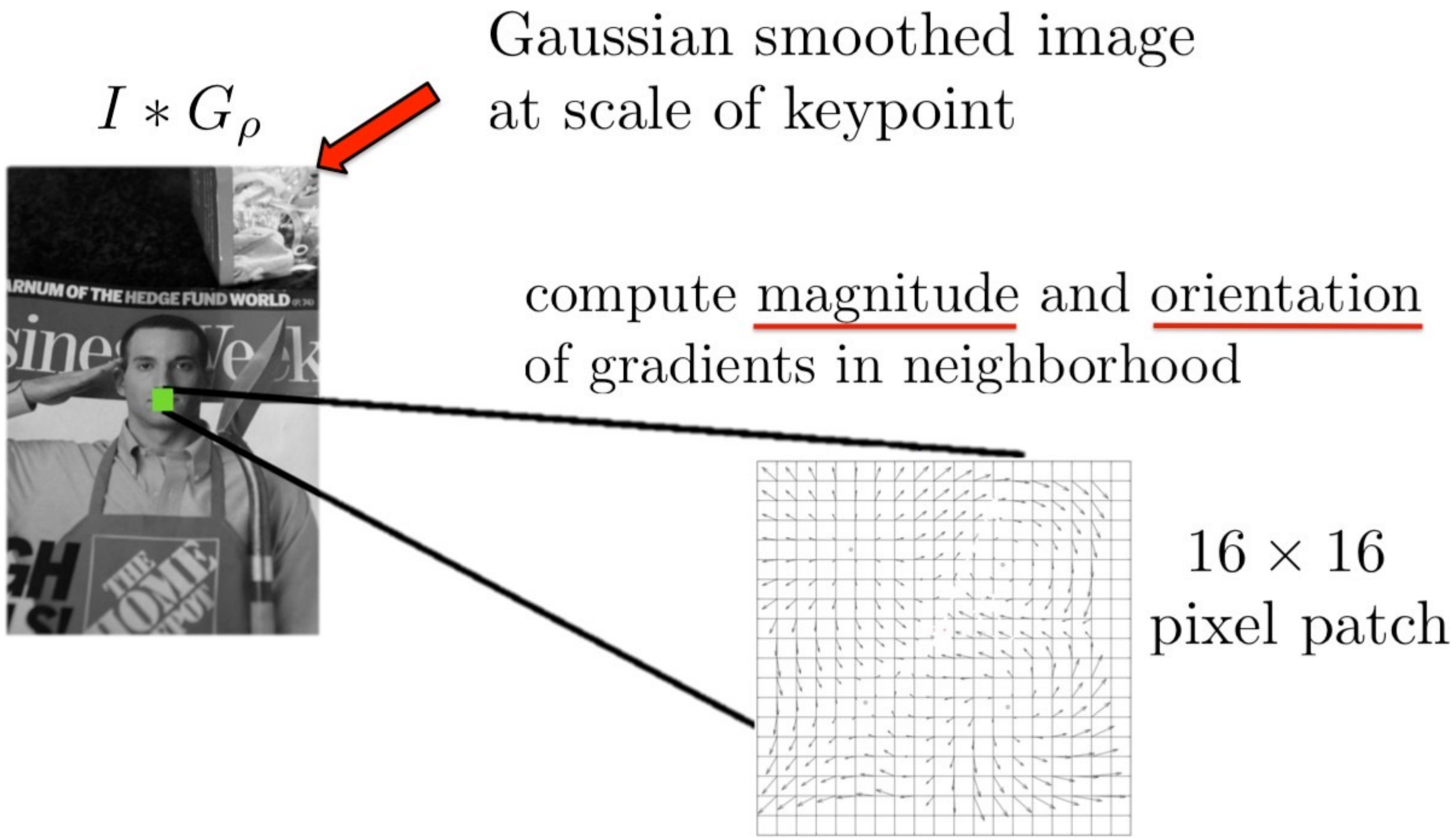
$$I_0 = I * G_{\sigma}$$



[Adopted from: F. Flores-Mangas]

SIFT Descriptor

- Compute the gradient magnitude and orientation in neighborhood of each keypoint proportional to the detected scale



[Adopted from: F. Flores-Mangas]

SIFT Descriptor

- Compute the gradient magnitude and orientation in neighborhood of each keypoint proportional to the detected scale

magnitude of gradient:

$$|\nabla I(x, y)| = \sqrt{\left(\frac{\partial(I(x, y) * G_\rho)}{\partial x}\right)^2 + \left(\frac{\partial(I(x, y) * G_\rho)}{\partial y}\right)^2}$$

gradient orientation:

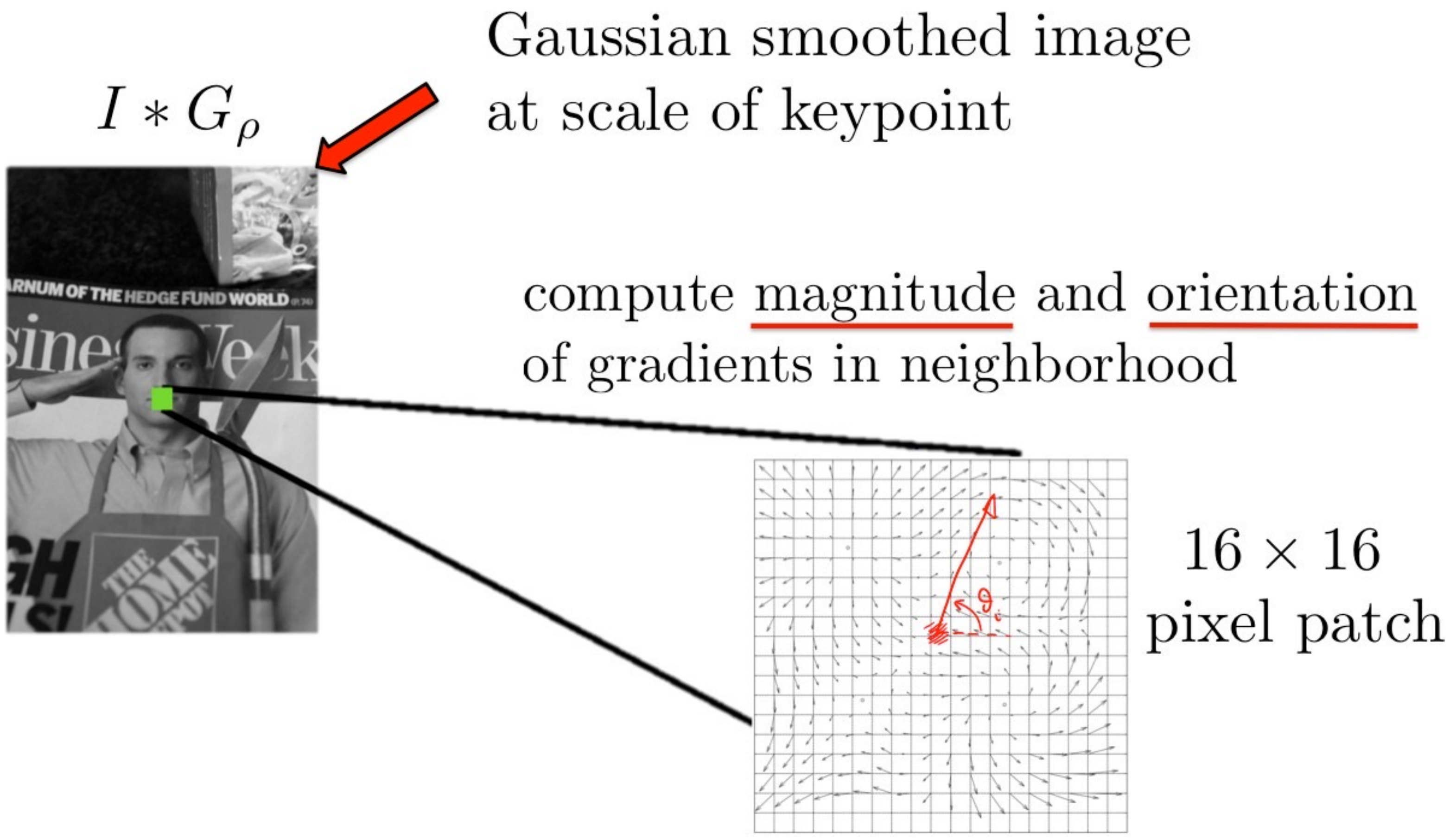
$$\theta(x, y) = \arctan\left(\frac{\partial I * G_\rho}{\partial y} / \frac{\partial I * G_\rho}{\partial x}\right)$$

(in case you forgot ;))

[Adopted from: F. Flores-Mangas]

SIFT Descriptor

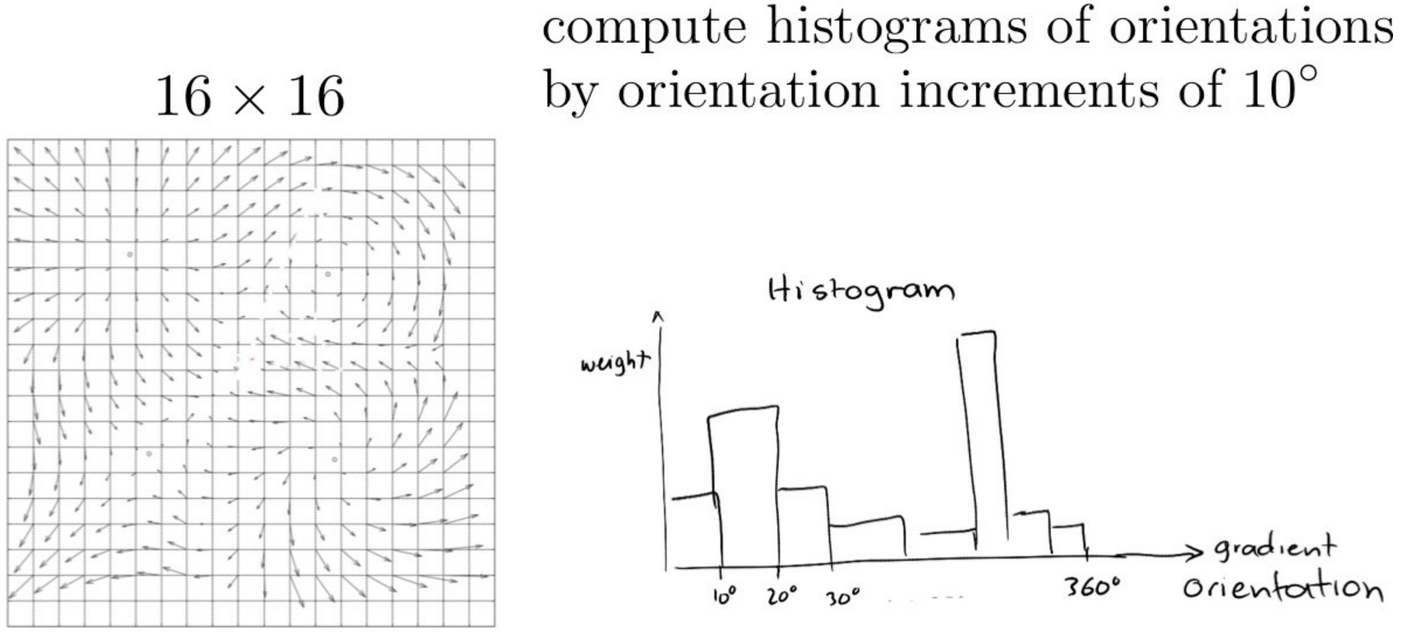
- Compute dominant orientation of each keypoint. How?



[Adopted from: F. Flores-Mangas]

SIFT Descriptor: Computing Dominant Orientation

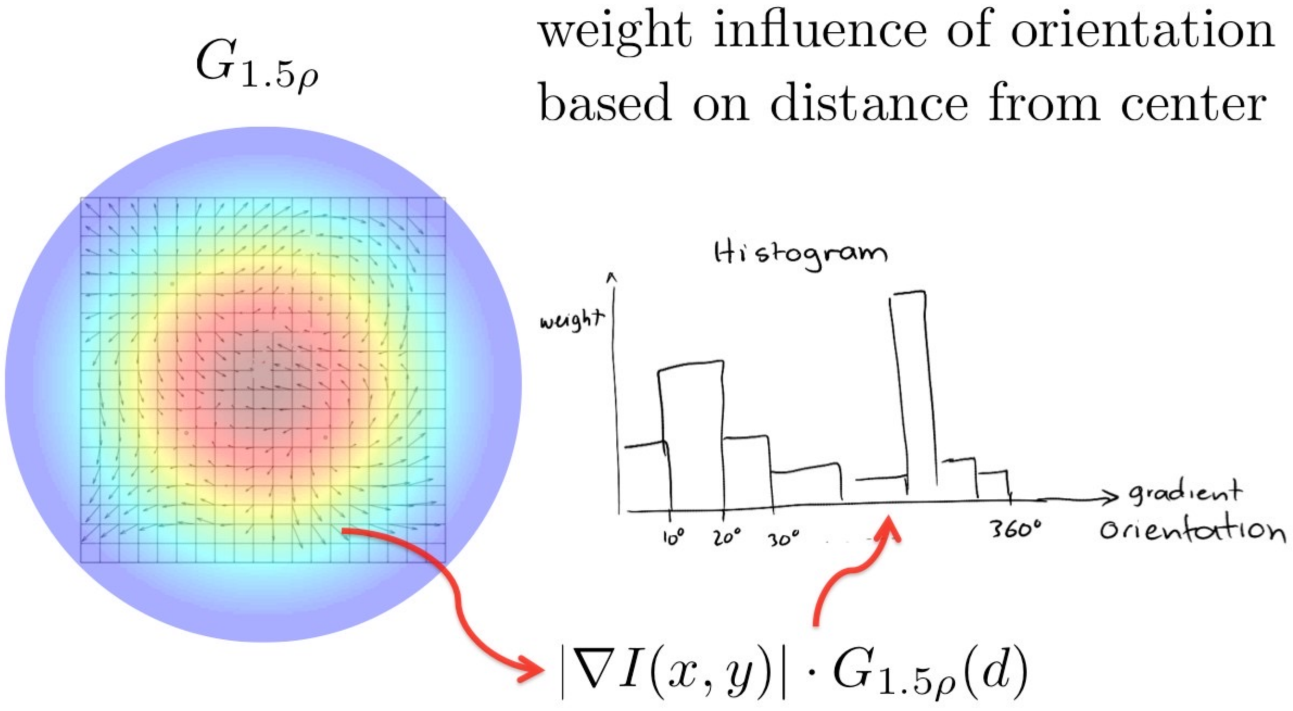
- Compute a histogram of gradient orientations, each bin covers 10°



[Adopted from: F. Flores-Mangas]

SIFT Descriptor: Computing Dominant Orientation

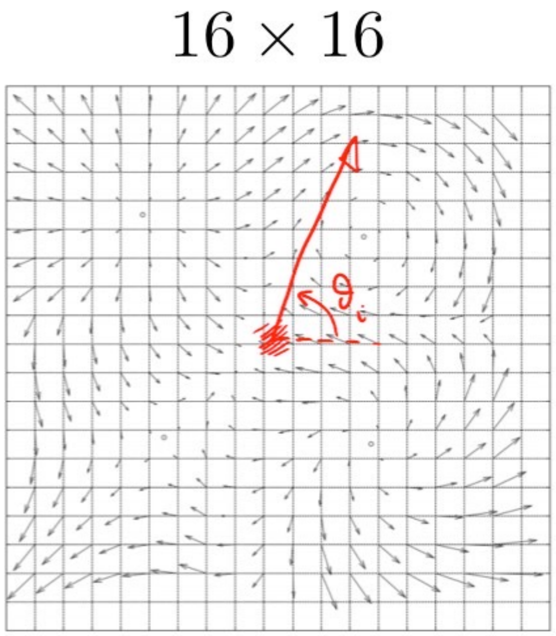
- Compute a histogram of gradient orientations, each bin covers 10°
- Orientations closer to the keypoint center should contribute more

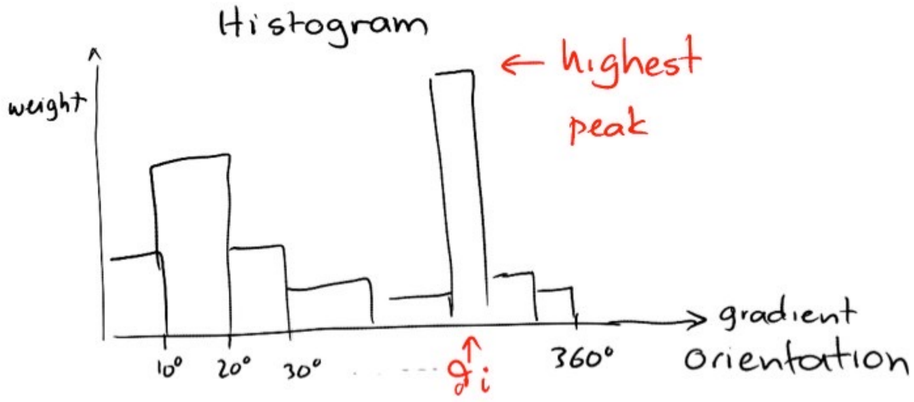


[Adopted from: F. Flores-Mangas]

SIFT Descriptor: Computing Dominant Orientation

- Compute a histogram of gradient orientations, each bin covers 10°
- Orientations closer to the keypoint center should contribute more
- Orientation giving the peak in the histogram is the keypoint's orientation

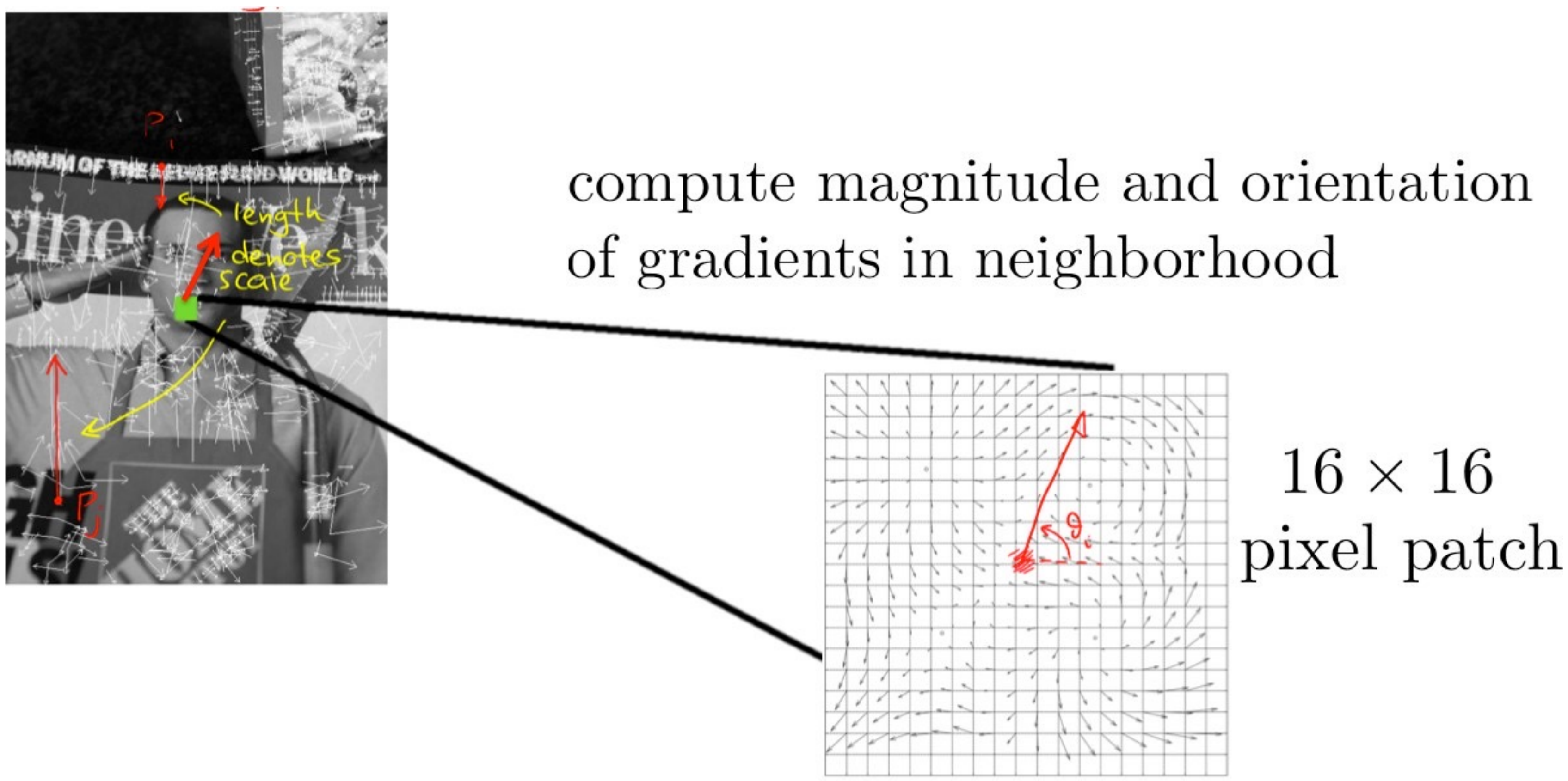




[Adopted from: F. Flores-Mangas]

SIFT Descriptor

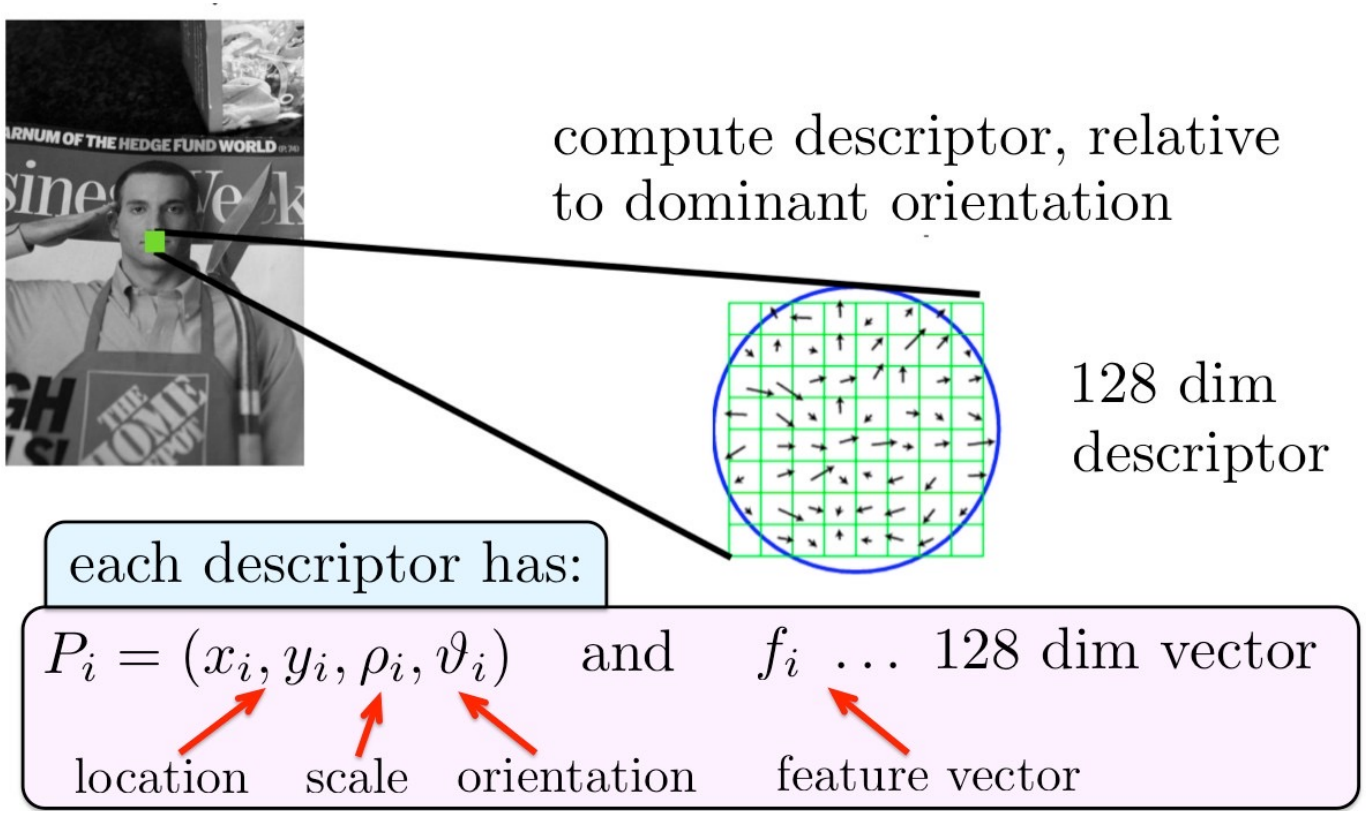
- Compute dominant orientation



[Adopted from: F. Flores-Mangas]

SIFT Descriptor

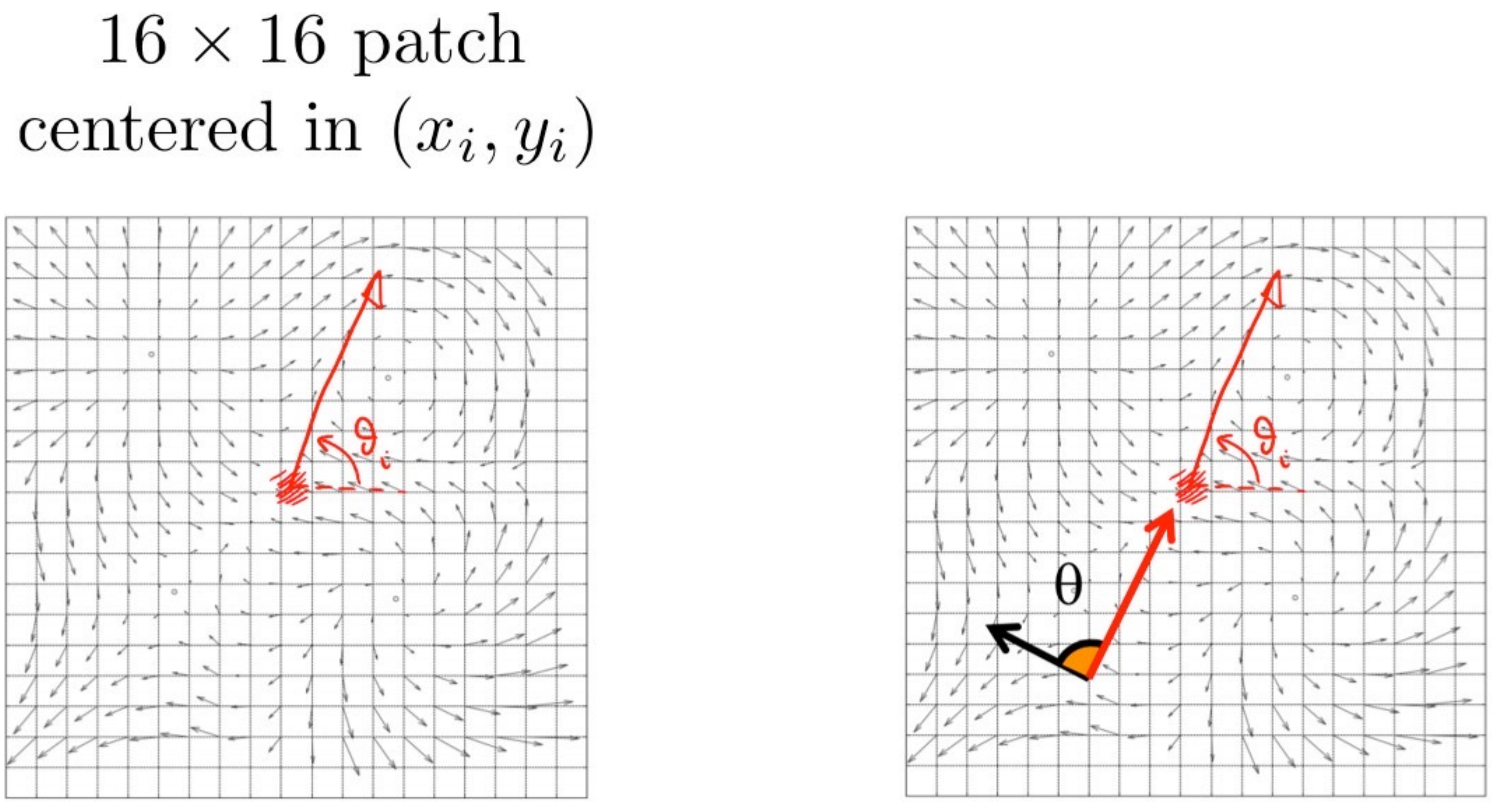
- Compute a 128 dimensional descriptor: 4 × 4 grid, each cell is a histogram of 8 orientation bins relative to dominant orientation



[Adopted from: F. Flores-Mangas]

SIFT Descriptor: Computing the Feature Vector

- Compute the orientations relative to the dominant orientation
- Otherwise rotating an object would phase shift entries in histogram

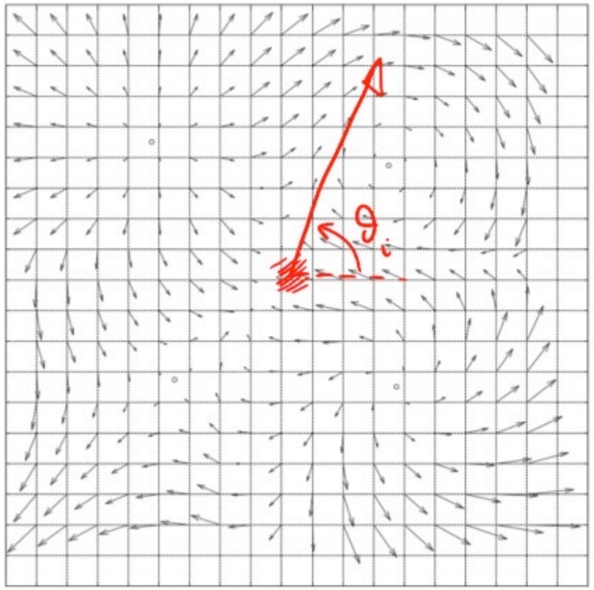


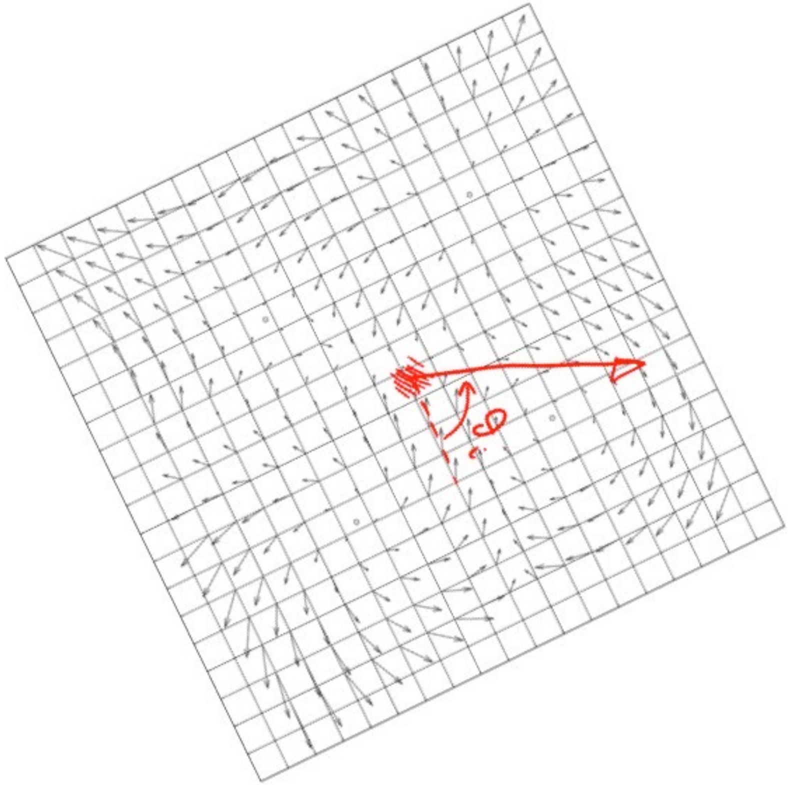
[Adopted from: F. Flores-Mangas]

SIFT Descriptor: Computing the Feature Vector

- Compute the orientations relative to the dominant orientation
- Otherwise rotating an object would phase shift entries in histogram

16×16 patch
centered in (x_i, y_i)

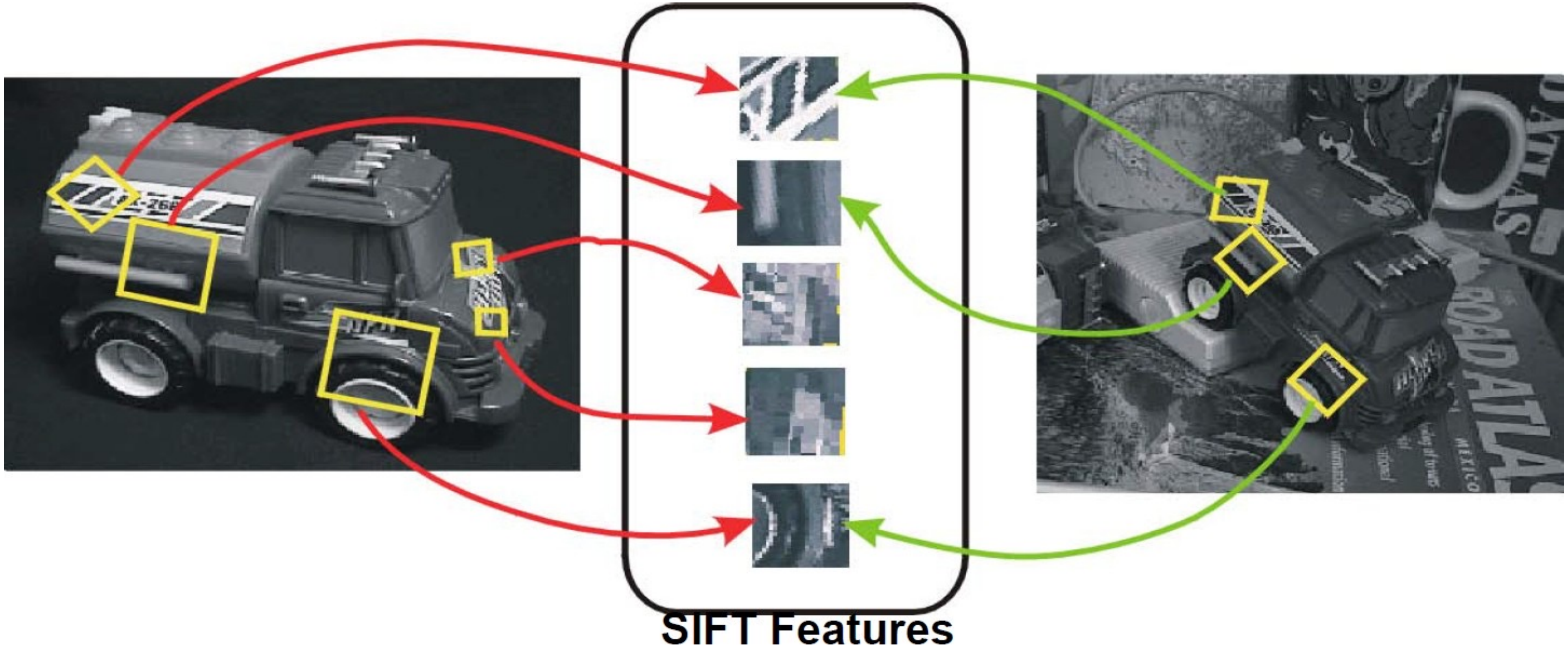




[Adopted from: F. Flores-Mangas]

SIFT Descriptor: Computing the Feature Vector

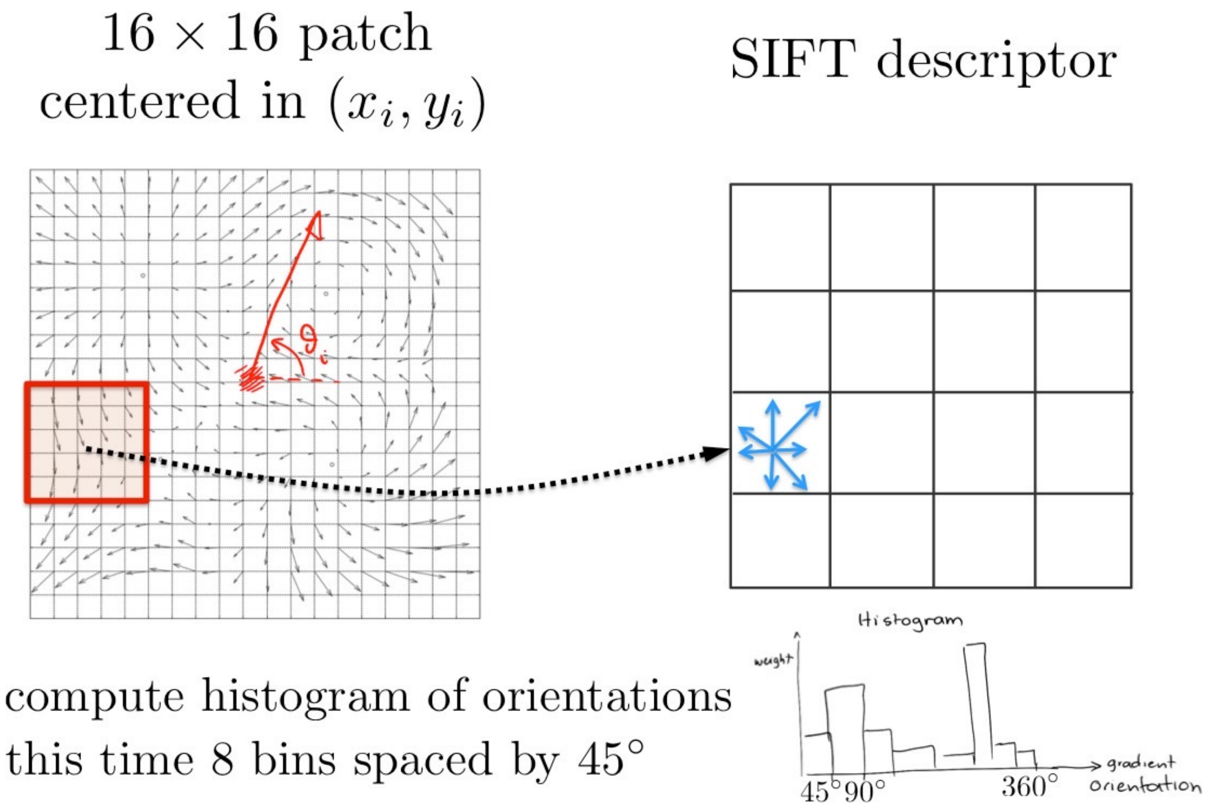
- Compute the orientations relative to the dominant orientation
- Otherwise rotating an object would phase shift entries in histogram



[Adopted from: F. Flores-Mangas]

SIFT Descriptor: Computing the Feature Vector

- Compute the orientations relative to the dominant orientation
- Otherwise rotating an object would phase shift entries in histogram
- Form a 4 × 4 grid. For each grid cell compute a histogram of orientations for 8 orientation bins spaced apart by 45°



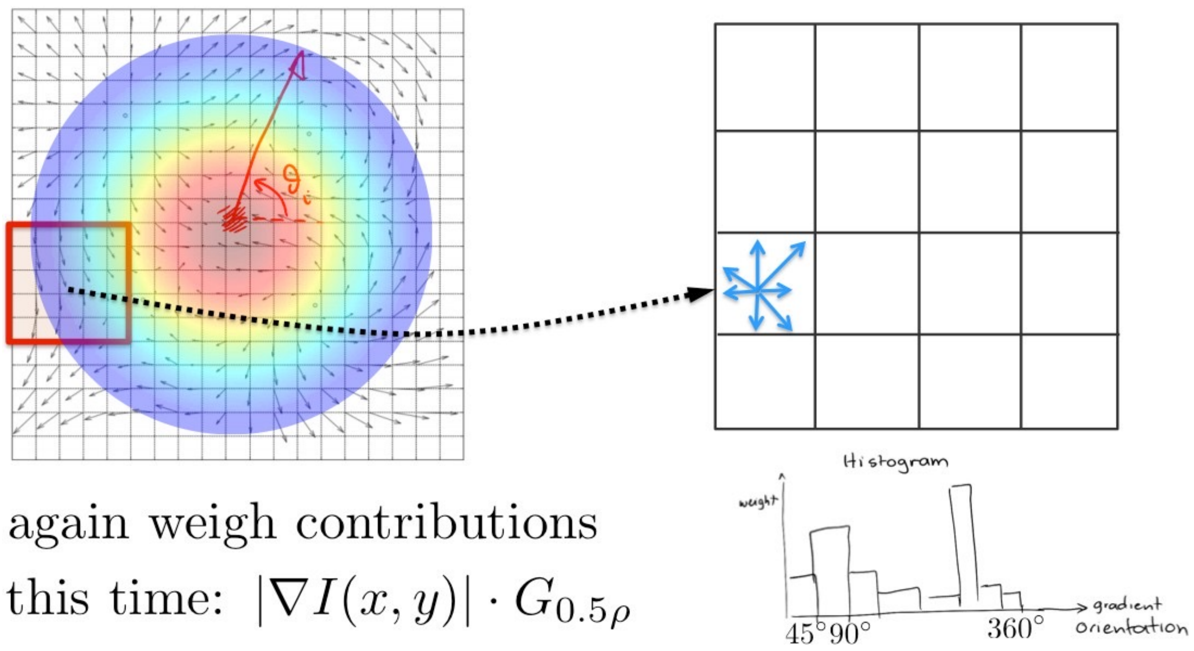
[Adopted from: F. Flores-Mangas]

SIFT Descriptor: Computing the Feature Vector

- Compute the orientations relative to the dominant orientation
- Otherwise rotating an object would phase shift entries in histogram
- Form a 4 × 4 grid. For each grid cell compute a histogram of orientations for 8 orientation bins spaced apart by 45°

16×16 patch
centered in (x_i, y_i)

SIFT descriptor



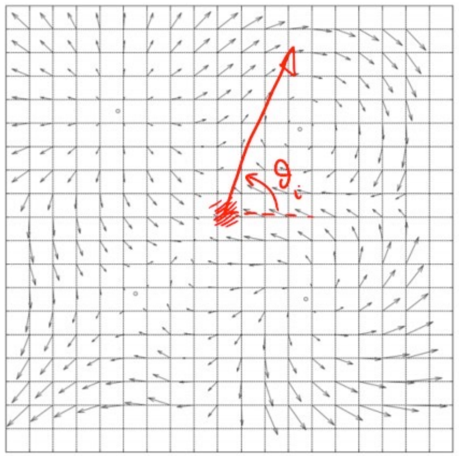
again weigh contributions
this time: $|\nabla I(x, y)| \cdot G_{0.5\rho}$

[Adopted from: F. Flores-Mangas]

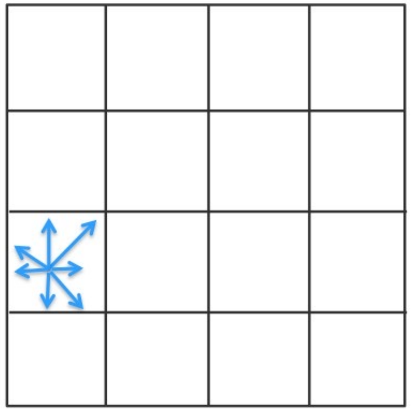
SIFT Descriptor: Computing the Feature Vector

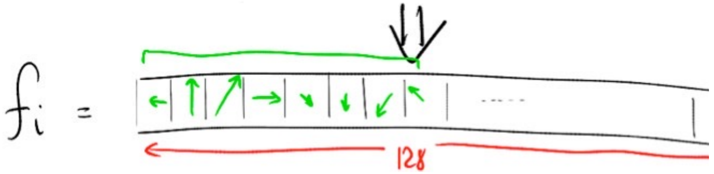
- Compute the orientations relative to the dominant orientation
- Otherwise rotating an object would phase shift entries in histogram
- Form a 4 × 4 grid. For each grid cell compute a histogram of orientations for 8 orientation bins spaced apart by 45°
- Form the 128 dimensional feature vector

16×16 patch
centered in (x_i, y_i)



SIFT descriptor





[Adopted from: F. Flores-Mar

SIFT Descriptor: Post-processing

- The resulting 128 non-negative values form a raw version of the SIFT descriptor vector.

SIFT Descriptor: Post-processing

- The resulting 128 non-negative values form a raw version of the SIFT descriptor vector.
- To reduce the effects of contrast or gain (additive variations are already removed by the gradient), the 128-D vector is normalized to unit length: $f_i = f_i / ||f_i||$

SIFT Descriptor: Post-processing

- The resulting 128 non-negative values form a raw version of the SIFT descriptor vector.
- To reduce the effects of contrast or gain (additive variations are already removed by the gradient), the 128-D vector is normalized to unit length: $f_i = f_i/||f_i||$
- To further make the descriptor robust to other photometric variations, values are clipped to 0.2 and the resulting vector is once again renormalized to unit length.

SIFT Descriptor: Post-processing

- The resulting 128 non-negative values form a raw version of the SIFT descriptor vector.
- To reduce the effects of contrast or gain (additive variations are already removed by the gradient), the 128-D vector is normalized to unit length: $f_i = f_i/||f_i||$
- To further make the descriptor robust to other photometric variations, values are clipped to 0.2 and the resulting vector is once again renormalized to unit length.
- Great engineering effort!

SIFT Descriptor: Post-processing

- The resulting 128 non-negative values form a raw version of the SIFT descriptor vector.
- To reduce the effects of contrast or gain (additive variations are already removed by the gradient), the 128-D vector is normalized to unit length: $f_i = f_i / ||f_i||$
- To further make the descriptor robust to other photometric variations, values are clipped to 0.2 and the resulting vector is once again renormalized to unit length.
- Great engineering effort!
- What is SIFT invariant to?

Properties of SIFT

- Invariant to:
  - Scale
  - Rotation

Properties of SIFT

- Invariant to:
  - Scale
  - Rotation
- Partially invariant to:

Properties of SIFT

- Invariant to:
  - Scale
  - Rotation
- Partially invariant to:
  - Illumination changes (sometimes even day vs. night)
  - Camera viewpoint (up to about 60 degrees of out-of-plane rotation)
  - Occlusion, clutter (why?)

Properties of SIFT

Properties of SIFT

- Invariant to:
  - Scale
  - Rotation
- Partially invariant to:
  - Illumination changes (sometimes even day vs. night)
  - Camera viewpoint (up to about 60 degrees of out-of-plane rotation)
  - Occlusion, clutter (why?)
- Also important:
  - Fast and efficient – can run in real time
  - Lots of code available

Examples

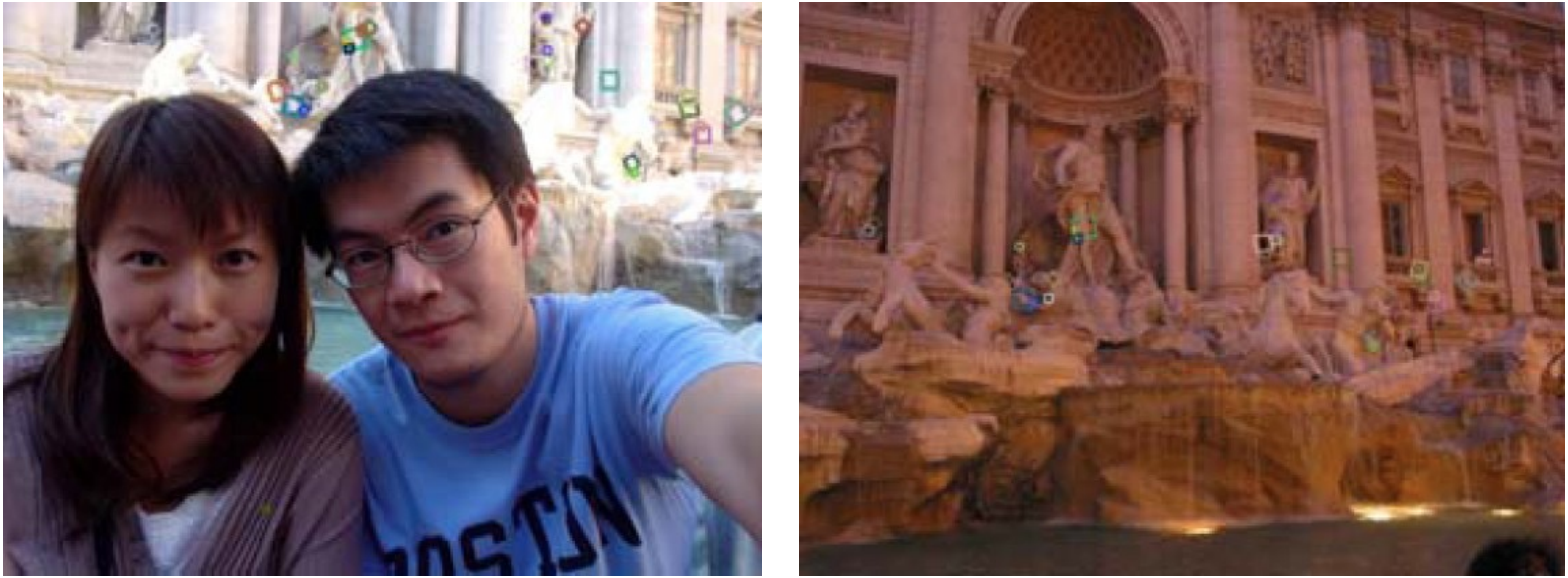


Figure: Matching in day / night under viewpoint change

[Source: S. Seitz]

Examples

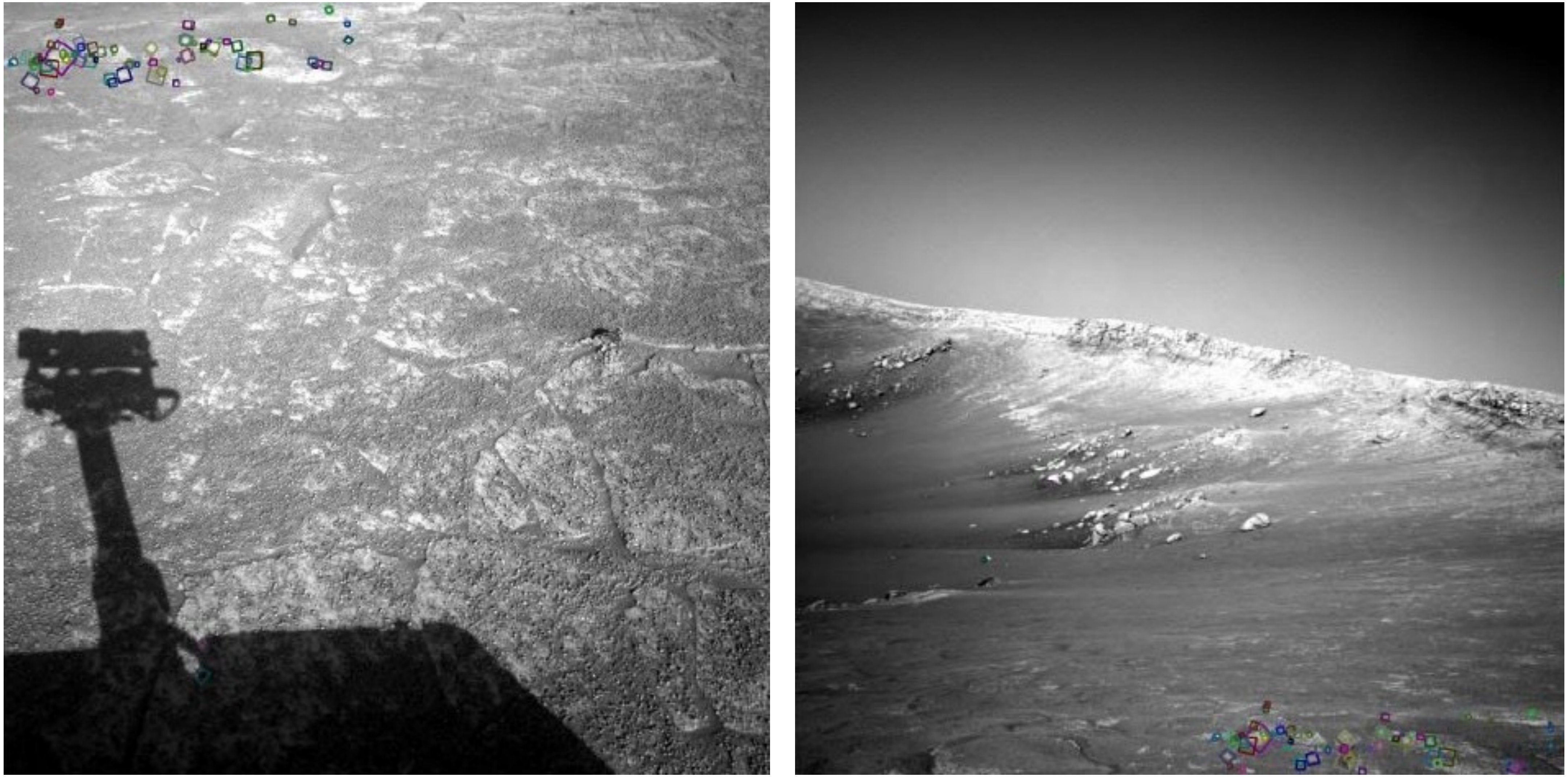


Figure: NASA Mars Rover images with SIFT feature matches

[Source: N. Snavely]

PCA-SIFT

- The dimensionality of SIFT is pretty high, i.e., 128D for each keypoint
- Reduce the dimensionality using linear dimensionality reduction
- In this case, principal component analysis (PCA)
- Use 10D or so descriptor

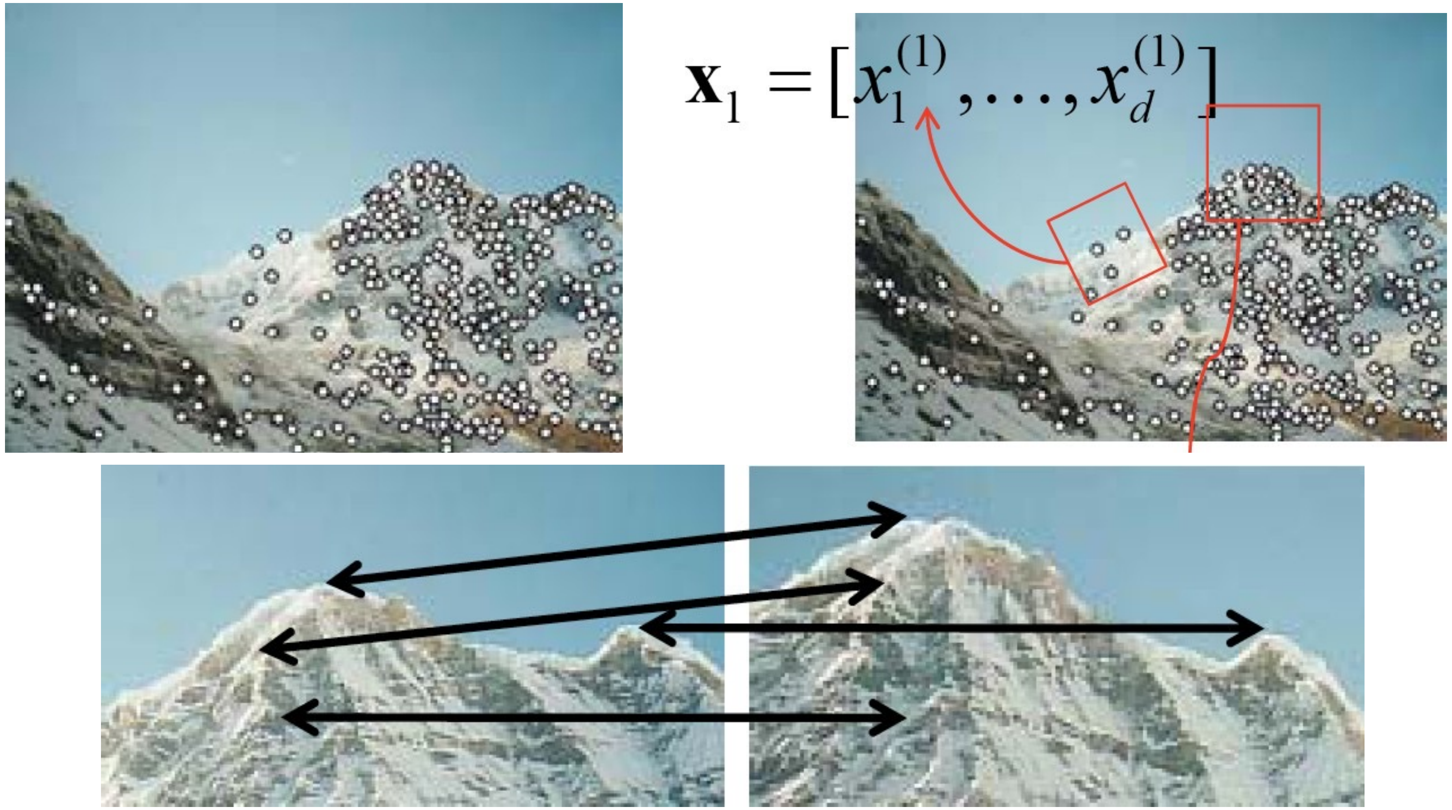
[Source: R. Urtasun]

Other Descriptors

- SURF
- DAISY
- LBP
- HOG
- Shape Contexts
- Color Histograms

Local Features

- Detection: Identify the interest points.
- Description: Extract feature descriptor around each interest point.
- **Matching: Determine correspondence between descriptors in two views.**



[Source: K. Grauman]

Overview

- motivation
- scale invariant keypoint detection
- learned keypoint detection
- image features
- matching

Matching the Local Descriptors

Once we have extracted keypoints and their descriptors, we want to match the features between pairs of images.

- Ideally a match is a correspondence between a local part of the object on one image to the same local part of the object in another image

Matching the Local Descriptors

Once we have extracted keypoints and their descriptors, we want to match the features between pairs of images.

- Ideally a match is a correspondence between a local part of the object on one image to the same local part of the object in another image
- How should we compute a match?

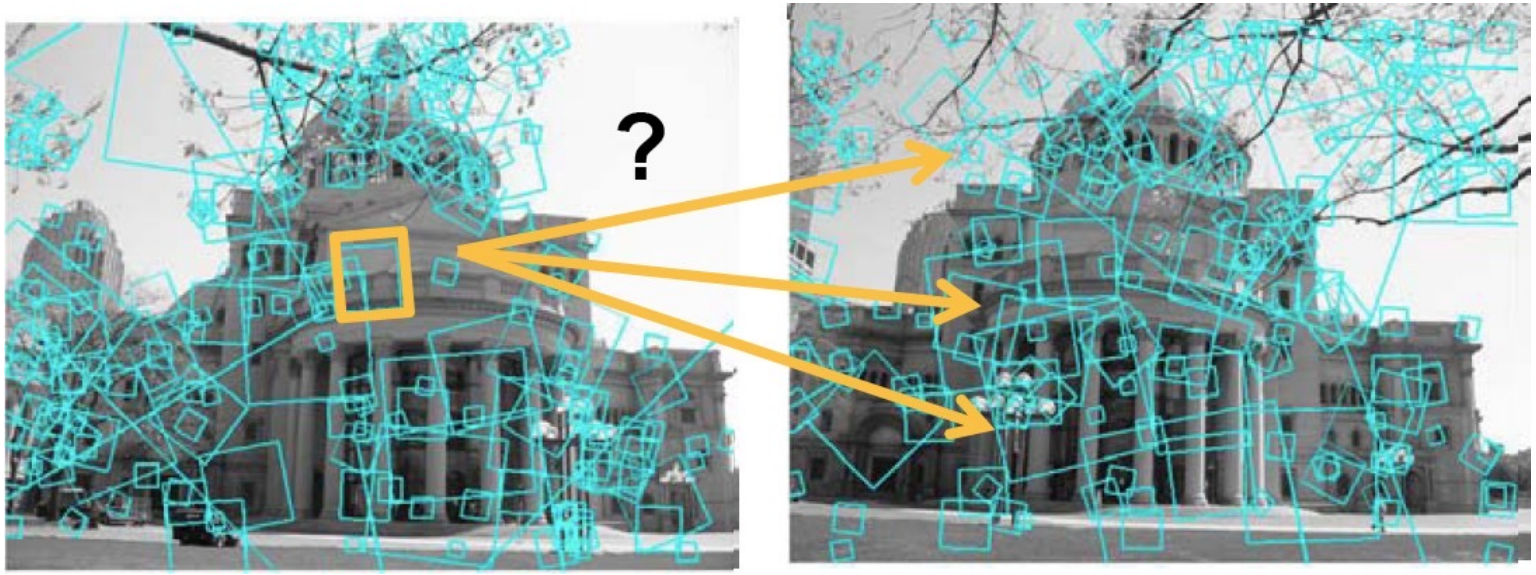
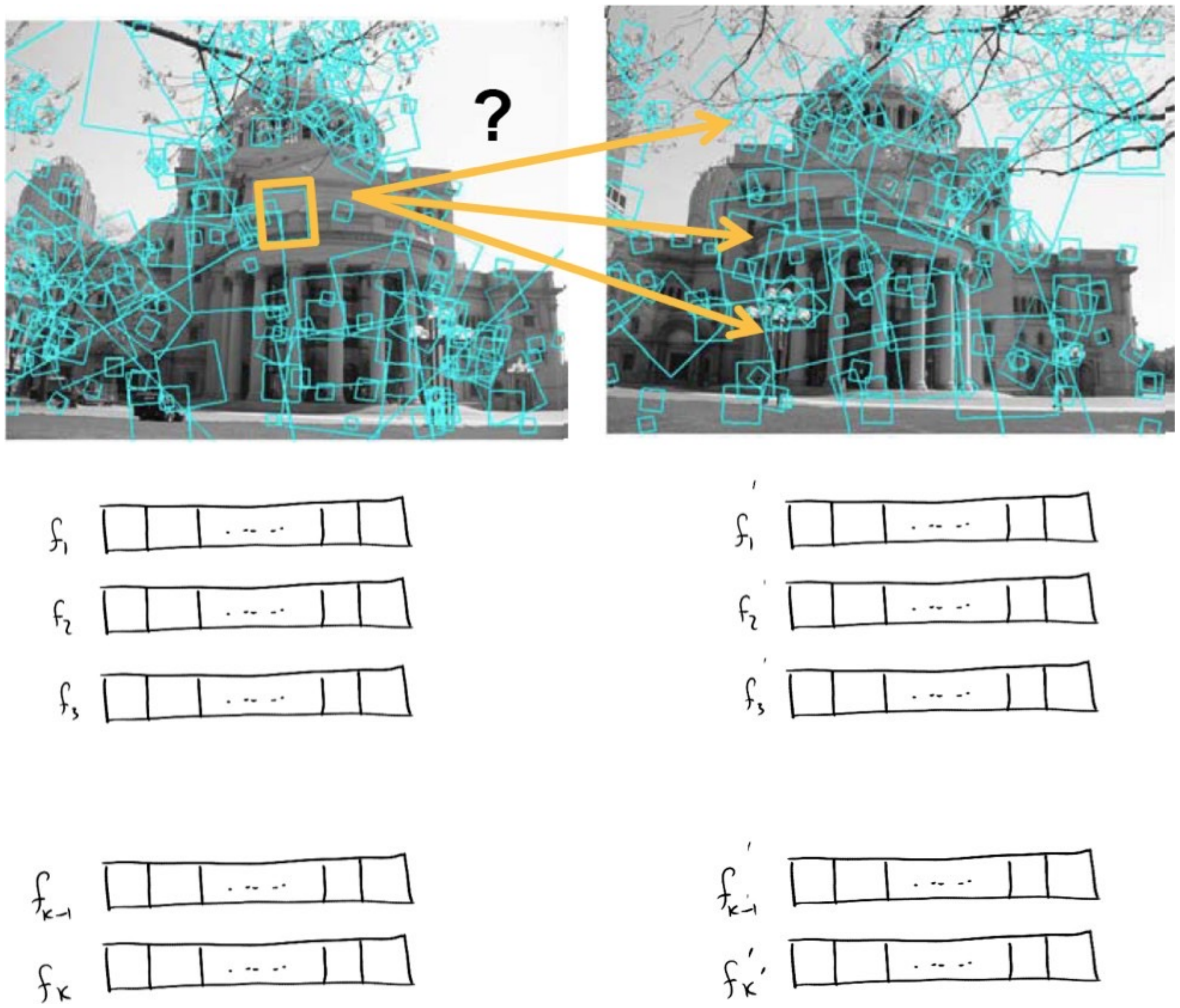


Figure: Images from K. Grauman

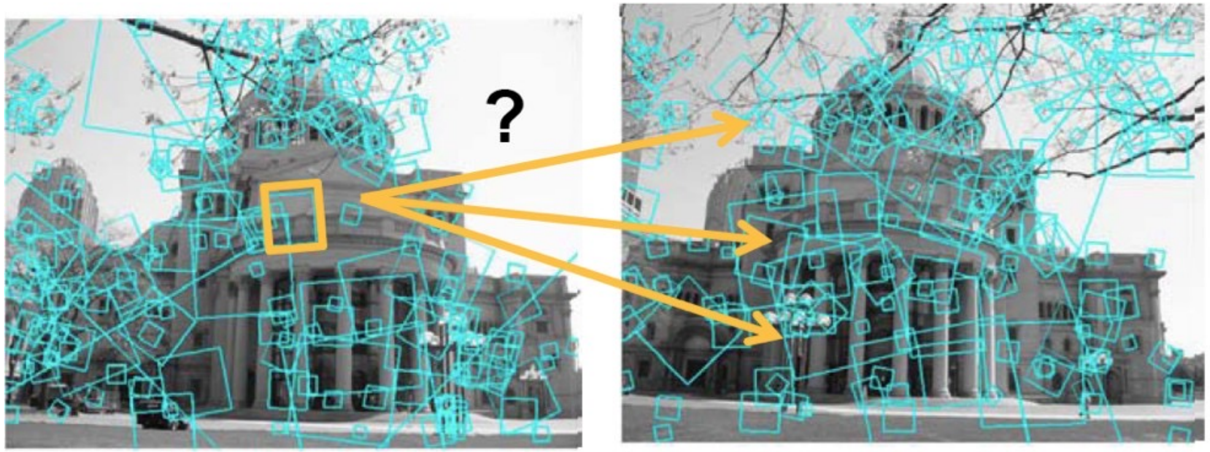
Matching the Local Descriptors

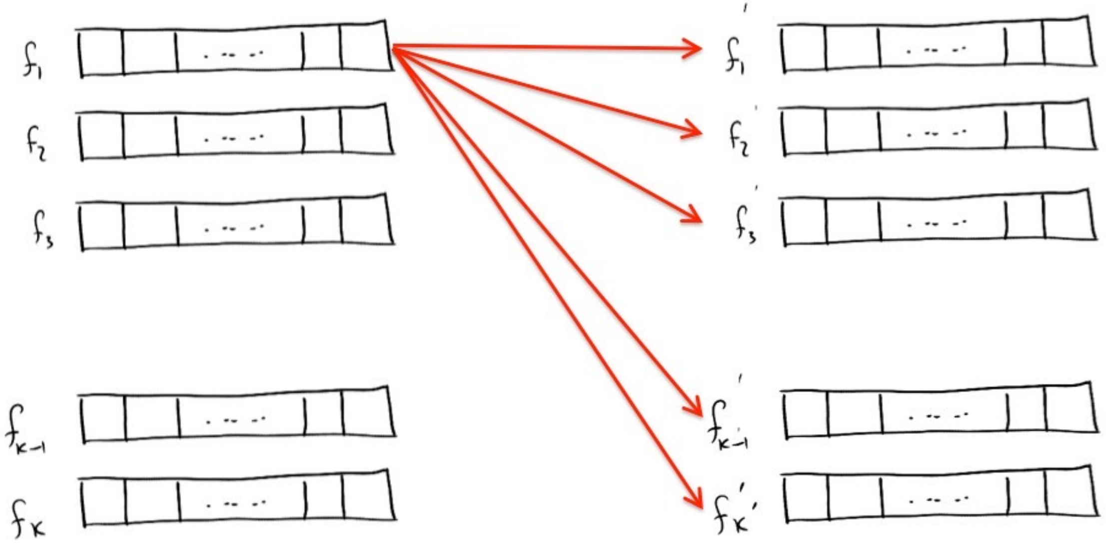
Simple: Compare them all, compute Euclidean distance



Matching the Local Descriptors

Simple: Compare them all, compute Euclidean distance





$$||f_1 - f_1'||$$

$$||f_1 - f_2'||$$

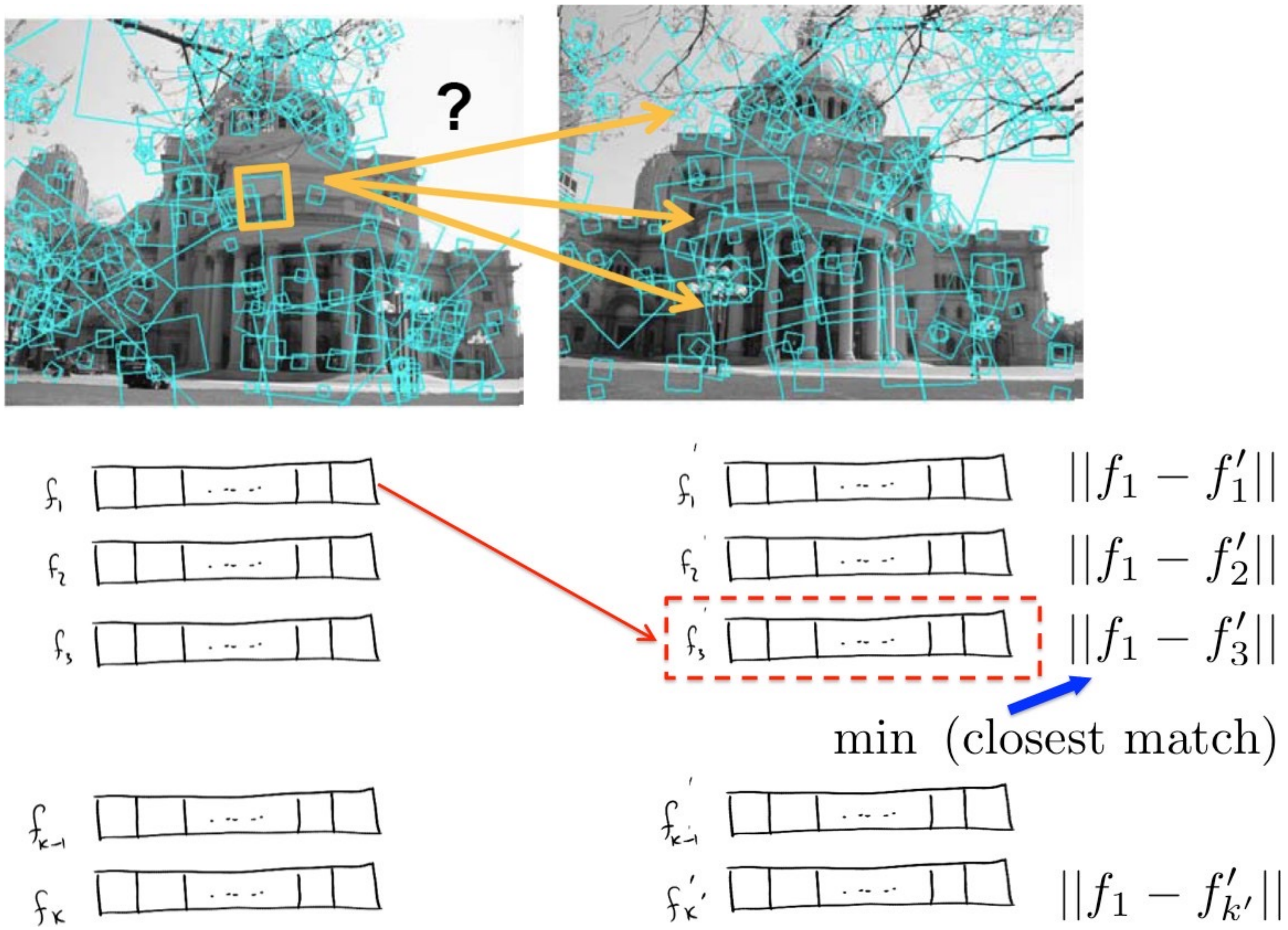
$$||f_1 - f_3'||$$

$$\vdots$$

$$||f_1 - f_{k'}'||$$

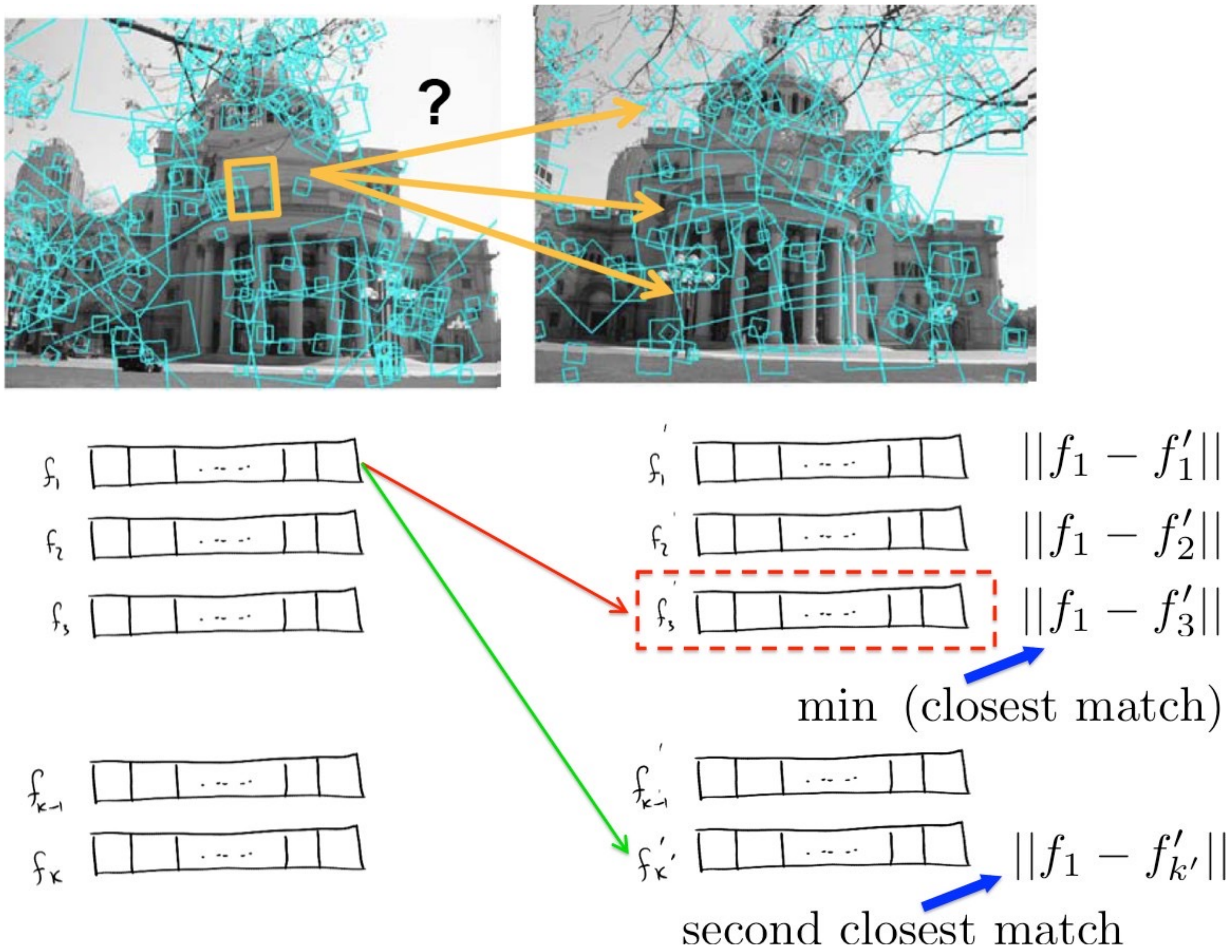
Matching the Local Descriptors

Find closest match (min distance). How do we know if match is **reliable**?



Matching the Local Descriptors

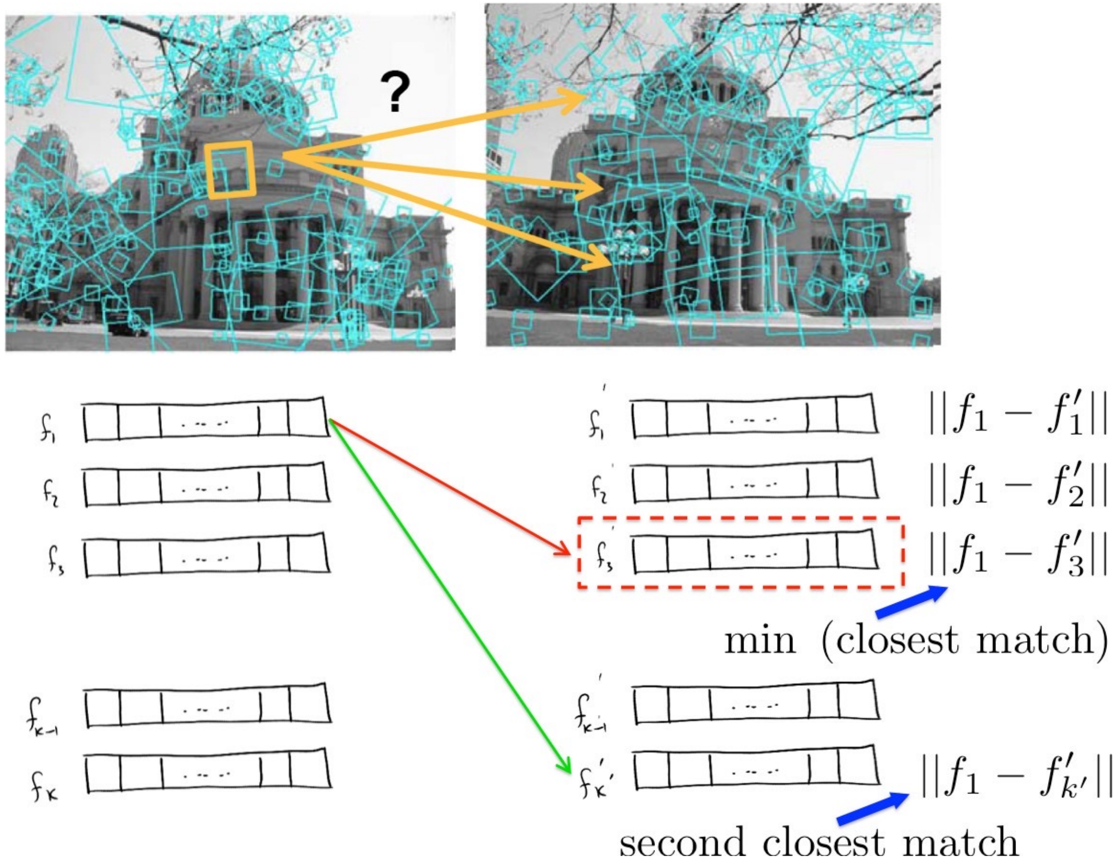
Find also the second closest match. Match reliable if first distance "much" smaller than second distance



Matching the Local Descriptors

Compute the ratio: $\phi_i = \frac{\|f_i - f'\mathrm{i}^{*}\|}{\|f_i - f'\mathrm{i}^{**}\|}$

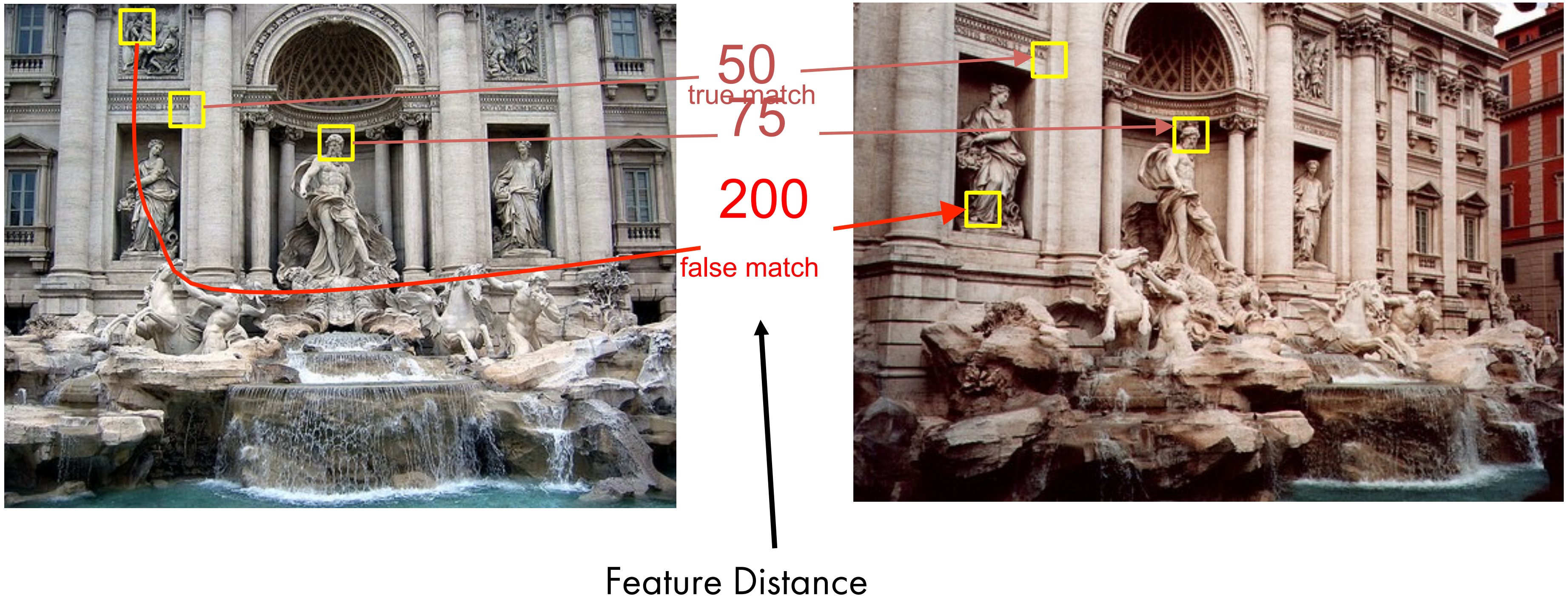
where f′∗i is the closest and f′∗i ∗ second closest match to f_i.



Which Threshold to Use?

Setting the threshold too high results in too many false positives, i.e., incorrect matches being returned.

Setting the threshold too low results in too many false negatives, i.e., too many correct matches being missed



Which Threshold to Use?

Threshold ratio of nearest to 2nd nearest descriptor

Typically: φi < 0.8

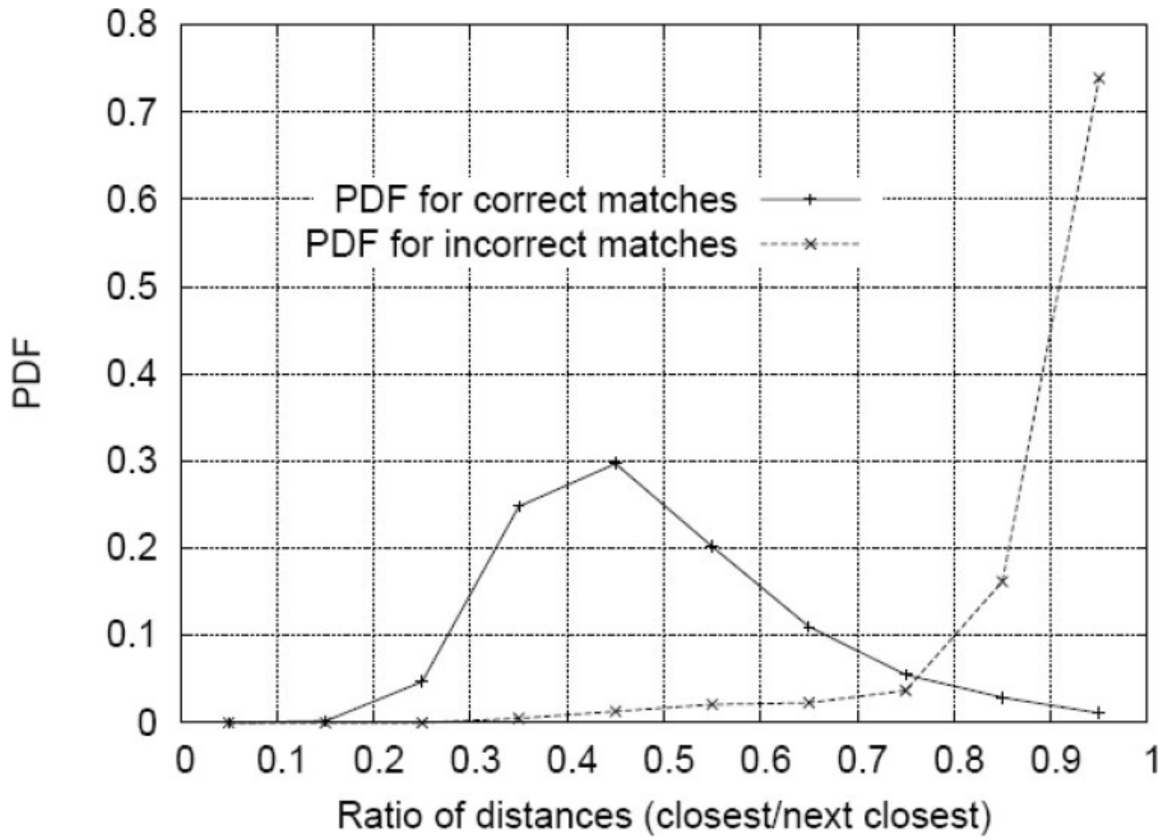


Figure: Images from D. Lowe

[Source: K. Grauman]

Applications of Local Invariant Features

- Wide baseline stereo
- Motion tracking
- Panorama stitching
- Mobile robot navigation
- 3D reconstruction
- Recognition
- Retrieval

[Source: K. Grauman]

Wide Baseline Stereo

[Source: T. Tuytelaars]

Motion Tracking

Figure: Images from J. Pilet

Now What

- Now we know how to extract scale and rotation invariant features
- We even know how to match features across images
- Can we use this to find Waldo in an even more sneaky scenario?

Now What

- Now we know how to extract scale and rotation invariant features
- We even know how to match features across images
- Can we use this to find Waldo in an even more sneaky scenario?

Waldo on the road

template

Now What

- Now we know how to extract scale and rotation invariant features
- We even know how to match features across images
- Can we use this to find Waldo in an even more sneaky scenario?

He comes closer… We know how to solve this

template

Now What

- Now we know how to extract scale and rotation invariant features
- We even know how to match features across images
- Can we use this to find Waldo in an even more sneaky scenario?

Someone takes a (weird) picture of him!

template

Find My DVD!

- More interesting: If we have DVD covers (e.g., from Amazon), can we match them to DVDs in real scenes?