Image Features:

# Scale Invariant Interest Point Detection

# 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: 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

image 1

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

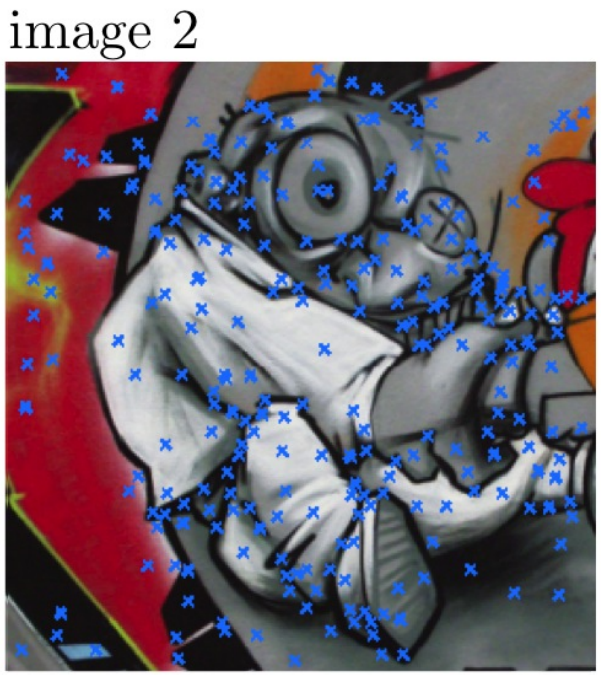



Figure: And **independently** in other images (independently: my algorithm only sees one image at a time – why is this a good idea?)

# 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

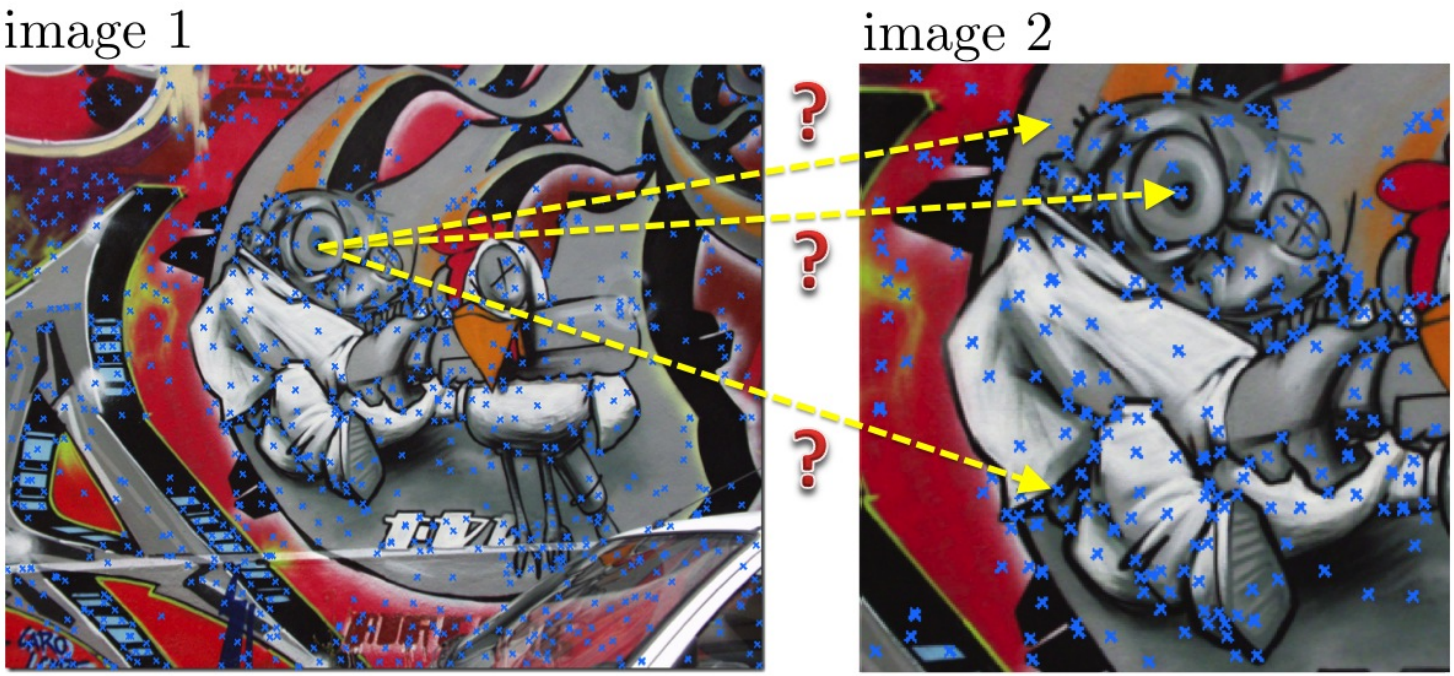



Figure: How can we match points??

# 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?

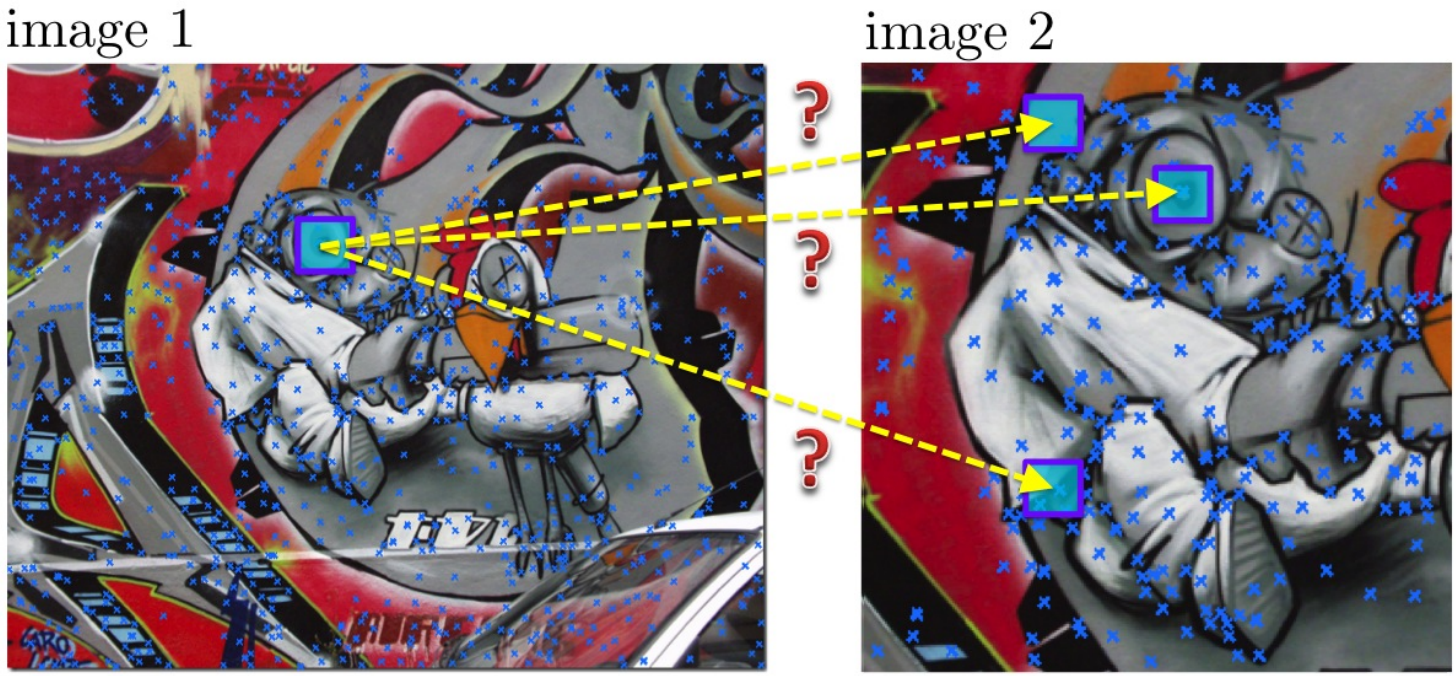



Figure: We could match if we took a patch around each point, and describe it with a feature vector (we know how to compare vectors)

# 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 **scale invariant** interest points on each image
  - Form a vector description of each point. How?

image 1

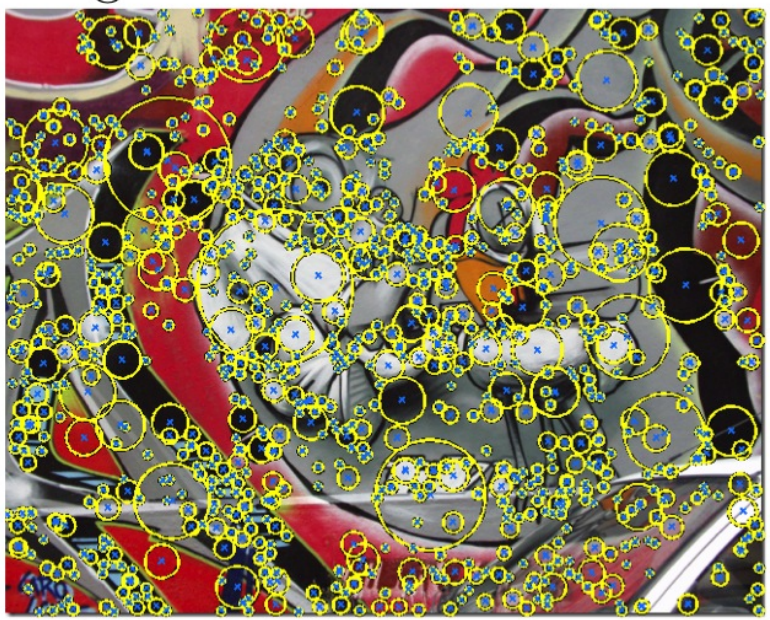


Figure: What if my interest point detector tells me the size (scale) of the patch? We are hoping that this "canonical" size somehow reflects size of the object.

# 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 **scale invariant** interest points on each image
  - Form a vector description of each point. How?

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 **scale invariant** interest points on each image
  - Form a vector description of each point. How?
  - Matching

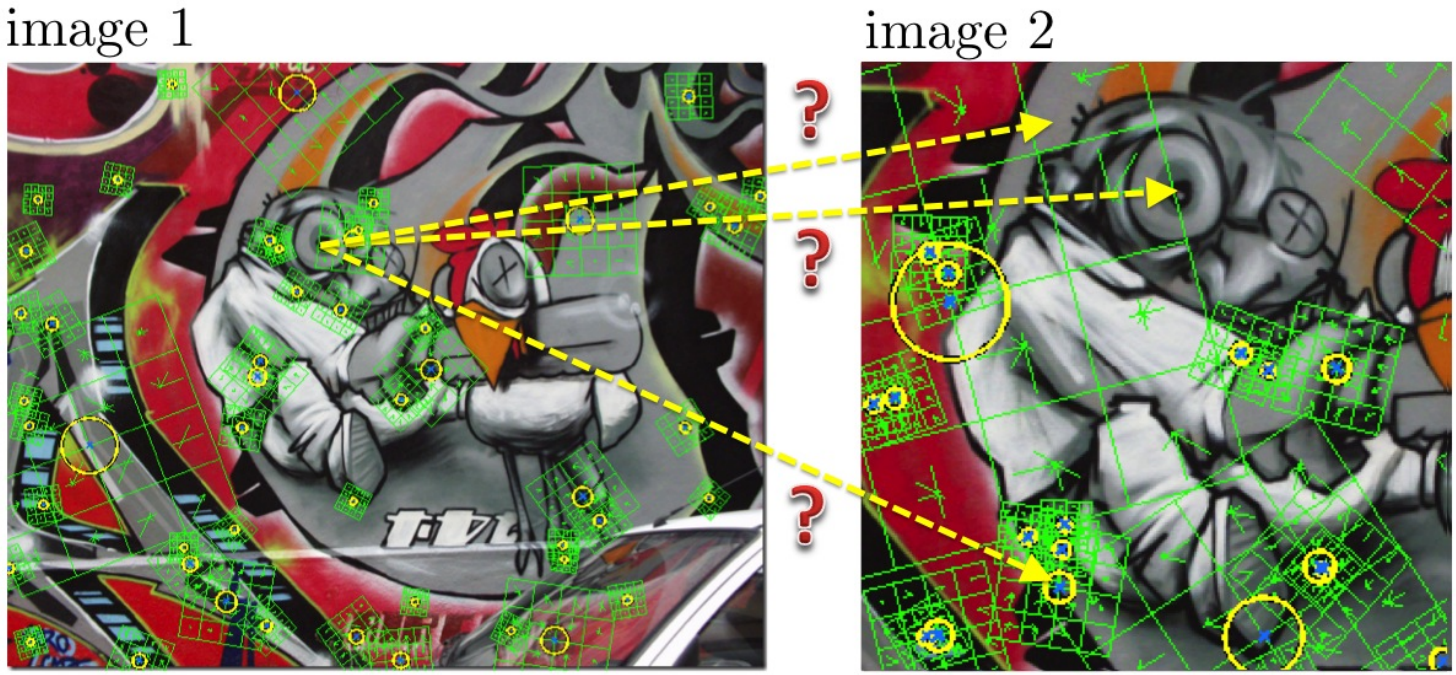



Figure: Then life is easy: we find the best matches and compute a transformation

# 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 **scale invariant** interest points on each image
  - Form a vector description of each point. How?
  - Matching

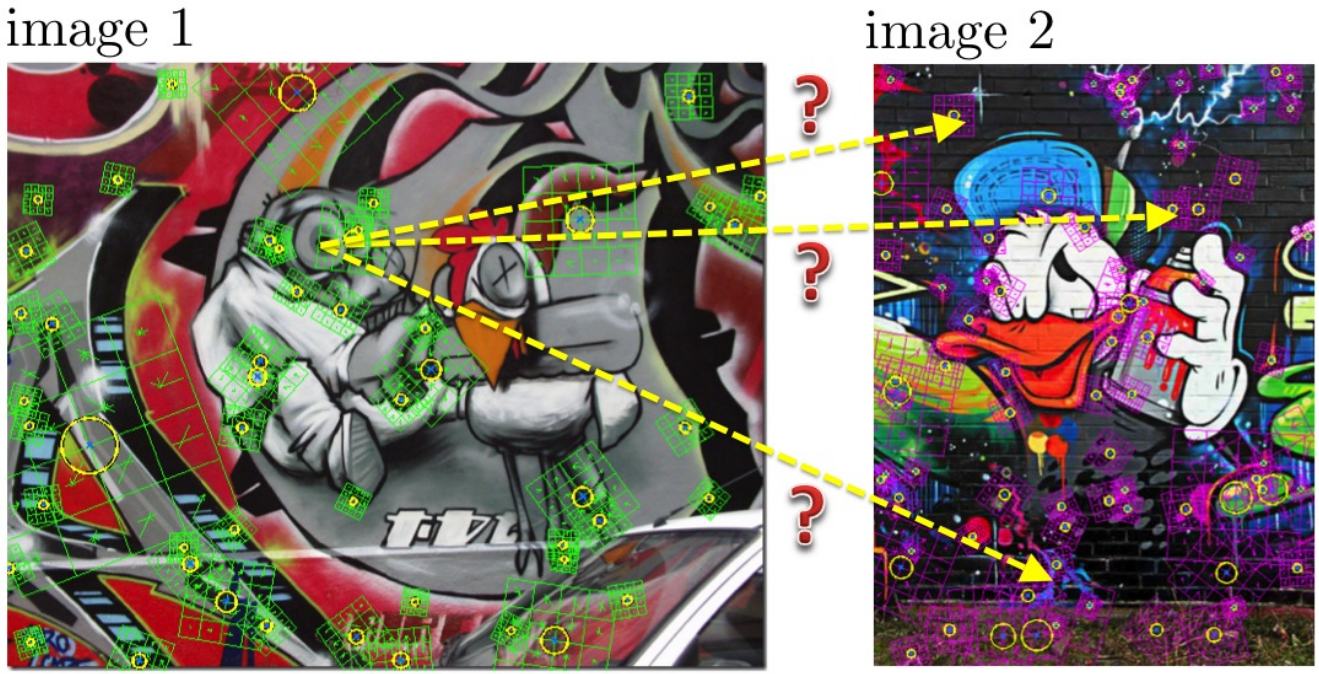



Figure: And we are hoping that our feature vectors and our matching algorithm

# 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 **scale invariant** interest points on each image  Let's do this first!
  - Form a vector description of each point. How?
  - Matching

# Scale Invariant Interest Points

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

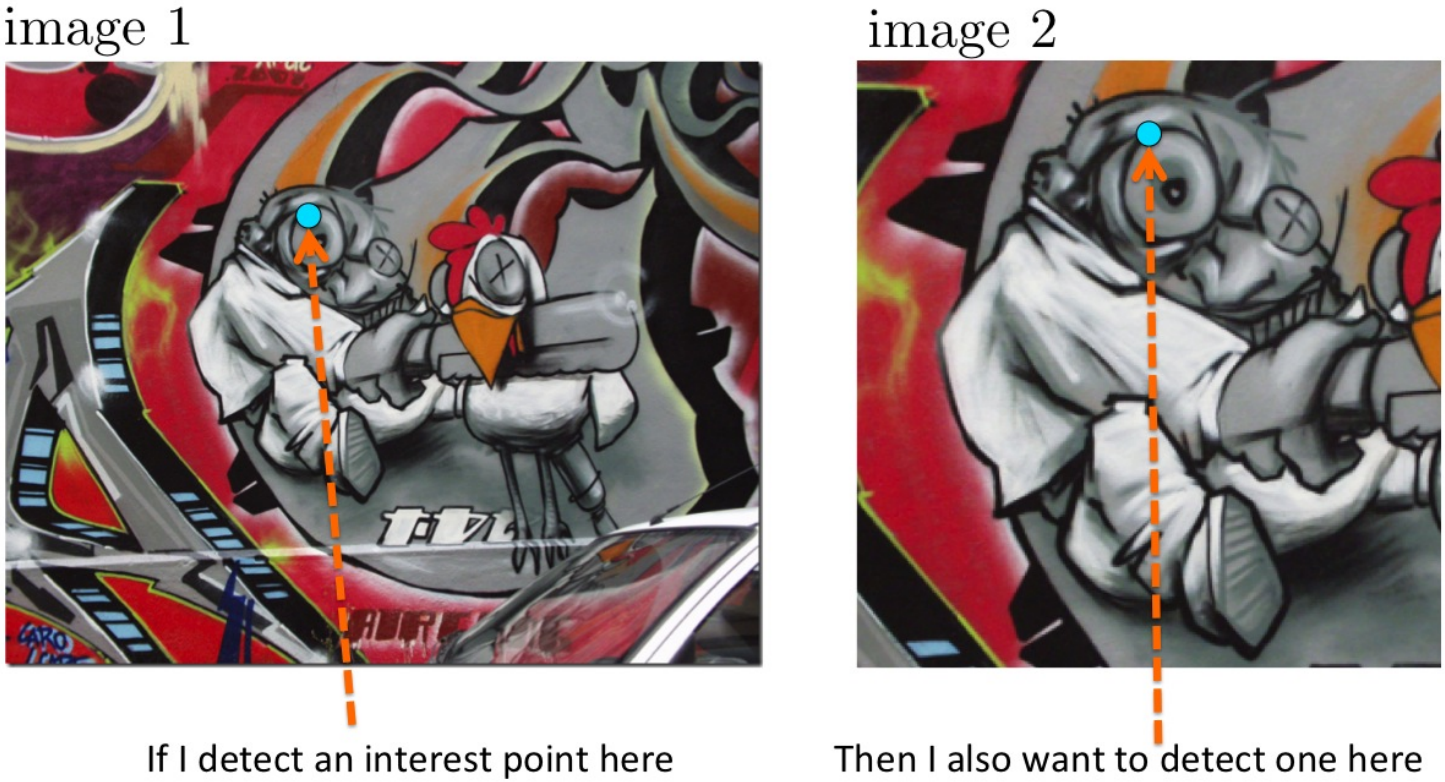
image 1

image 2

[Source: K. Grauman, slide credit: R. Urtasun]

# Scale Invariant Interest Points

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

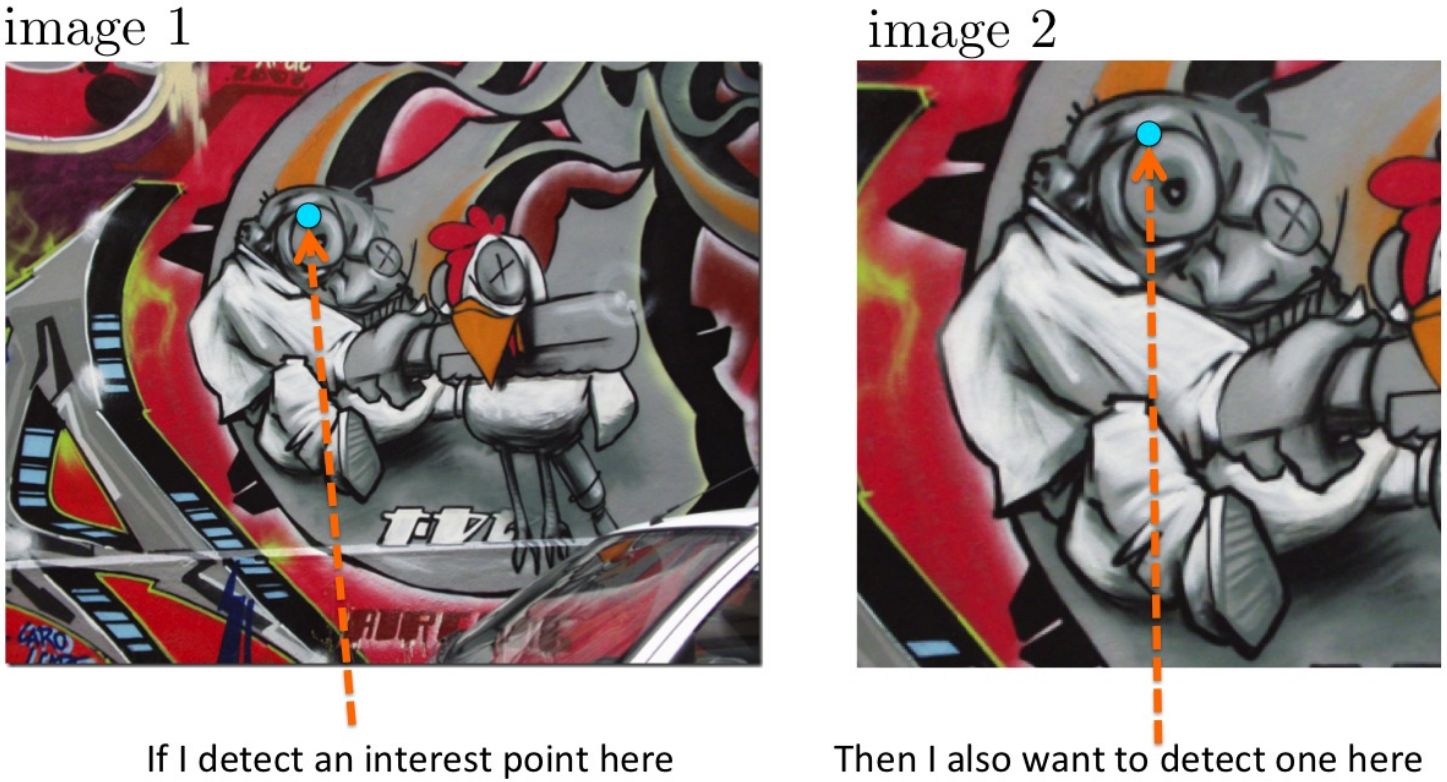



[Source: K. Grauman, slide credit: R. Urtasun]

# Scale Invariant Interest Points

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

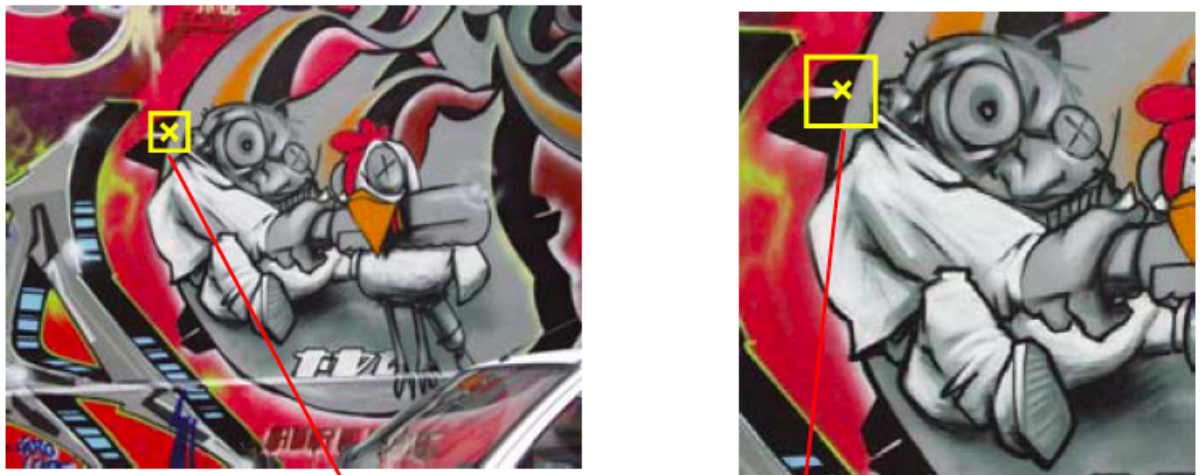
- Extract features at a variety of scales, e.g., by using multiple resolutions in a pyramid, and then matching features at the same level.
- When does this work?



# 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.



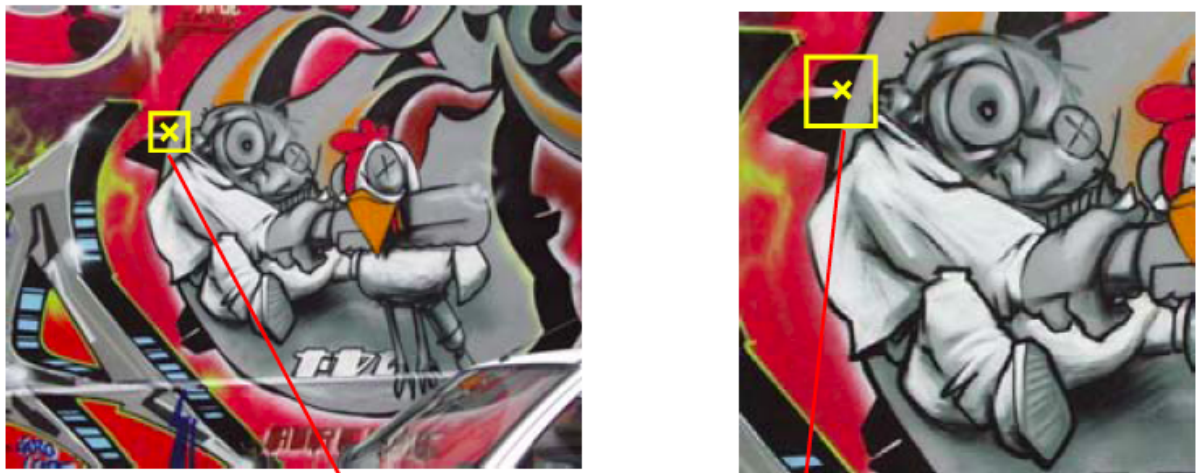
$$f(I_{i_1 \ldots i_m}(x,\sigma)) = f(I_{i_1 \ldots i_m}(x',\sigma'))$$

[Source: K. Grauman, slide credit: R. Urtasun]

# Scale Invariant Interest Points

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

- Find scale that gives local maxima of a function $f$ in both position and scale.

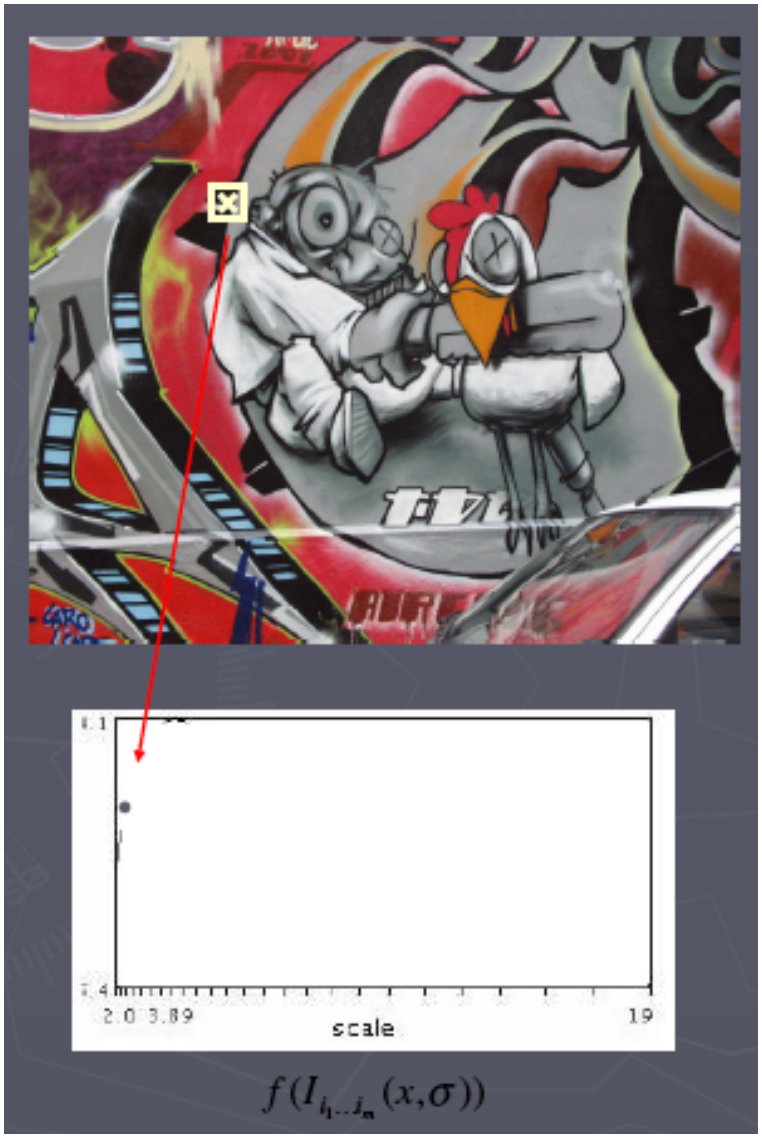


$$f(I_{i_1 \ldots i_m}(x, \sigma)) \quad = \quad f(I_{i_1 \ldots i_m}(x', \sigma'))$$

[Source: K. Grauman, slide credit: R. Urtasun]

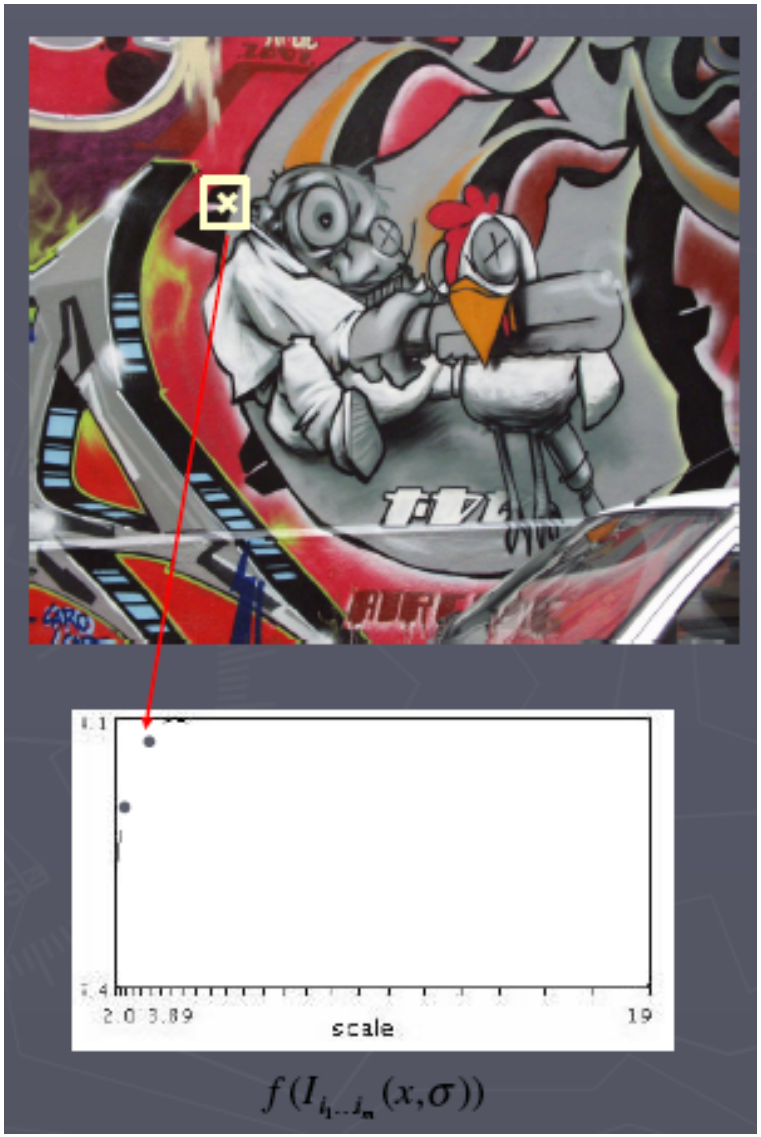
# Automatic Scale Selection

Function responses for increasing scale (scale signature).



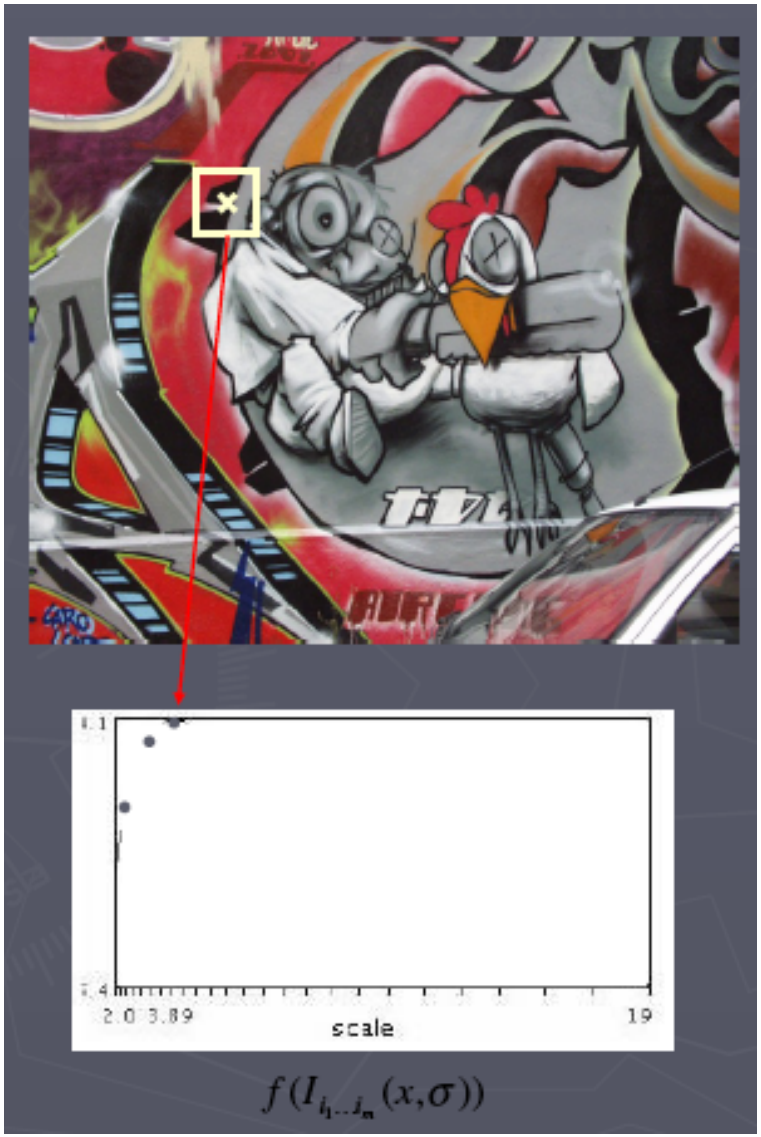
# Automatic Scale Selection

Function responses for increasing scale (scale signature).



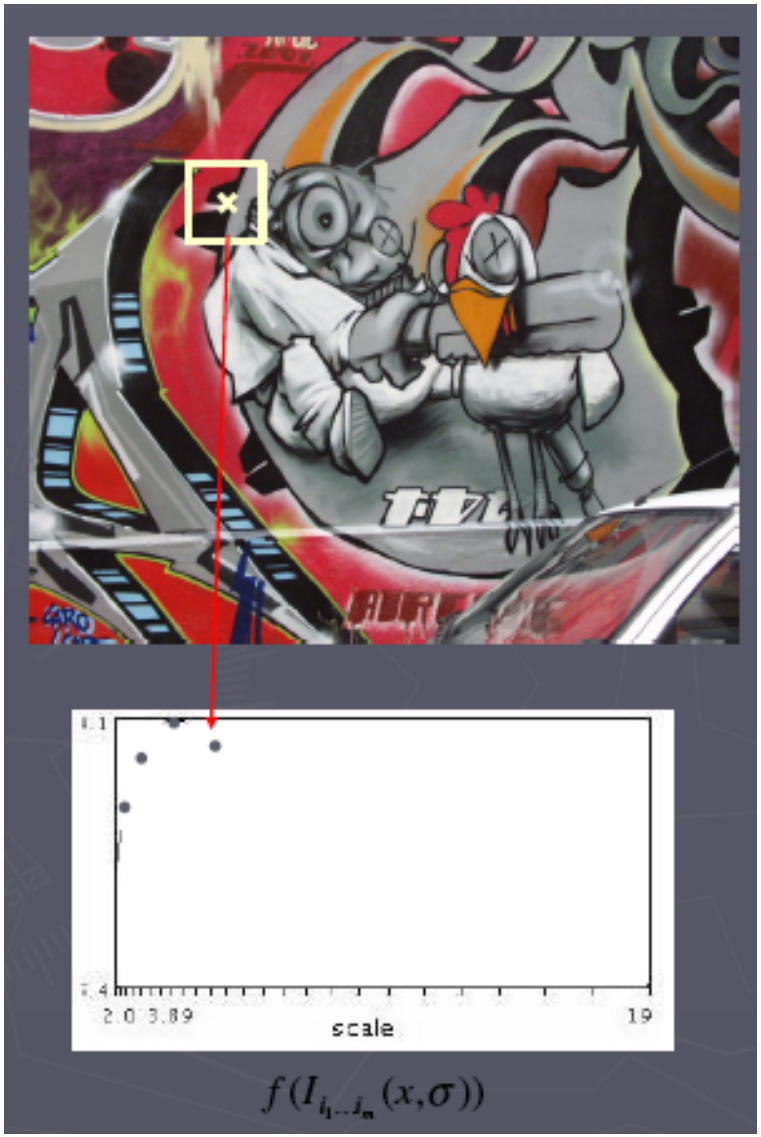
# Automatic Scale Selection

Function responses for increasing scale (scale signature).



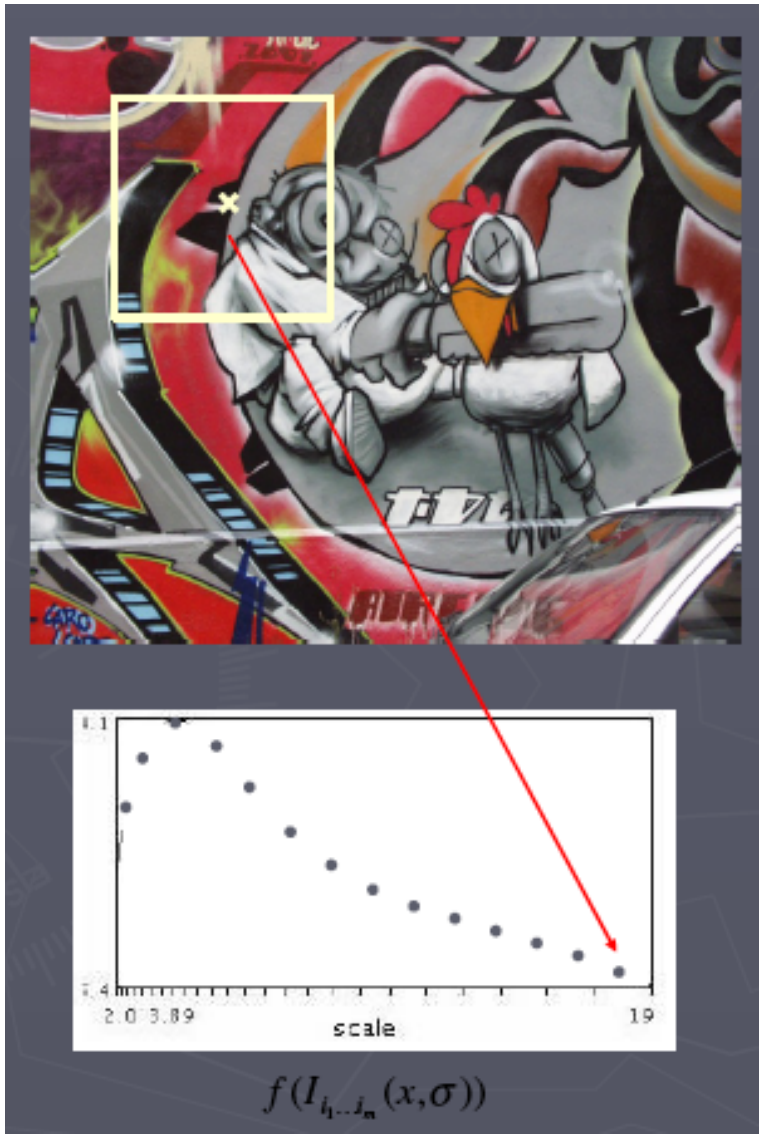
# Automatic Scale Selection

Function responses for increasing scale (scale signature).



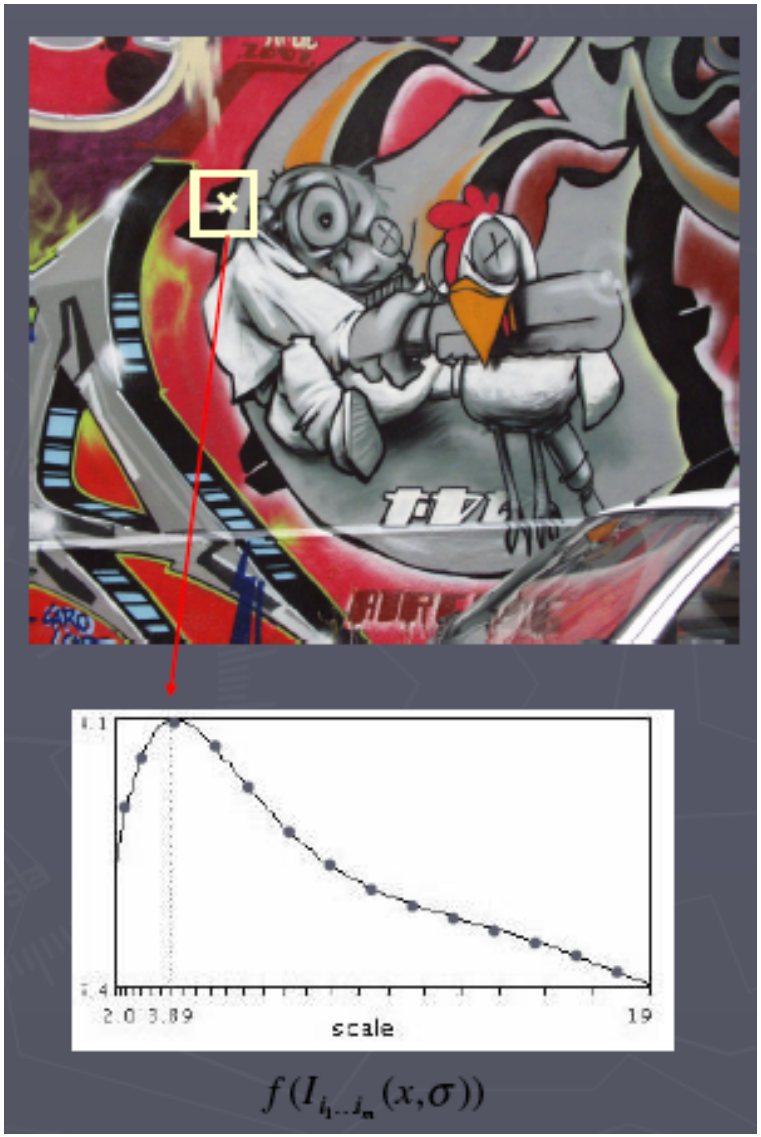
# Automatic Scale Selection

Function responses for increasing scale (scale signature).



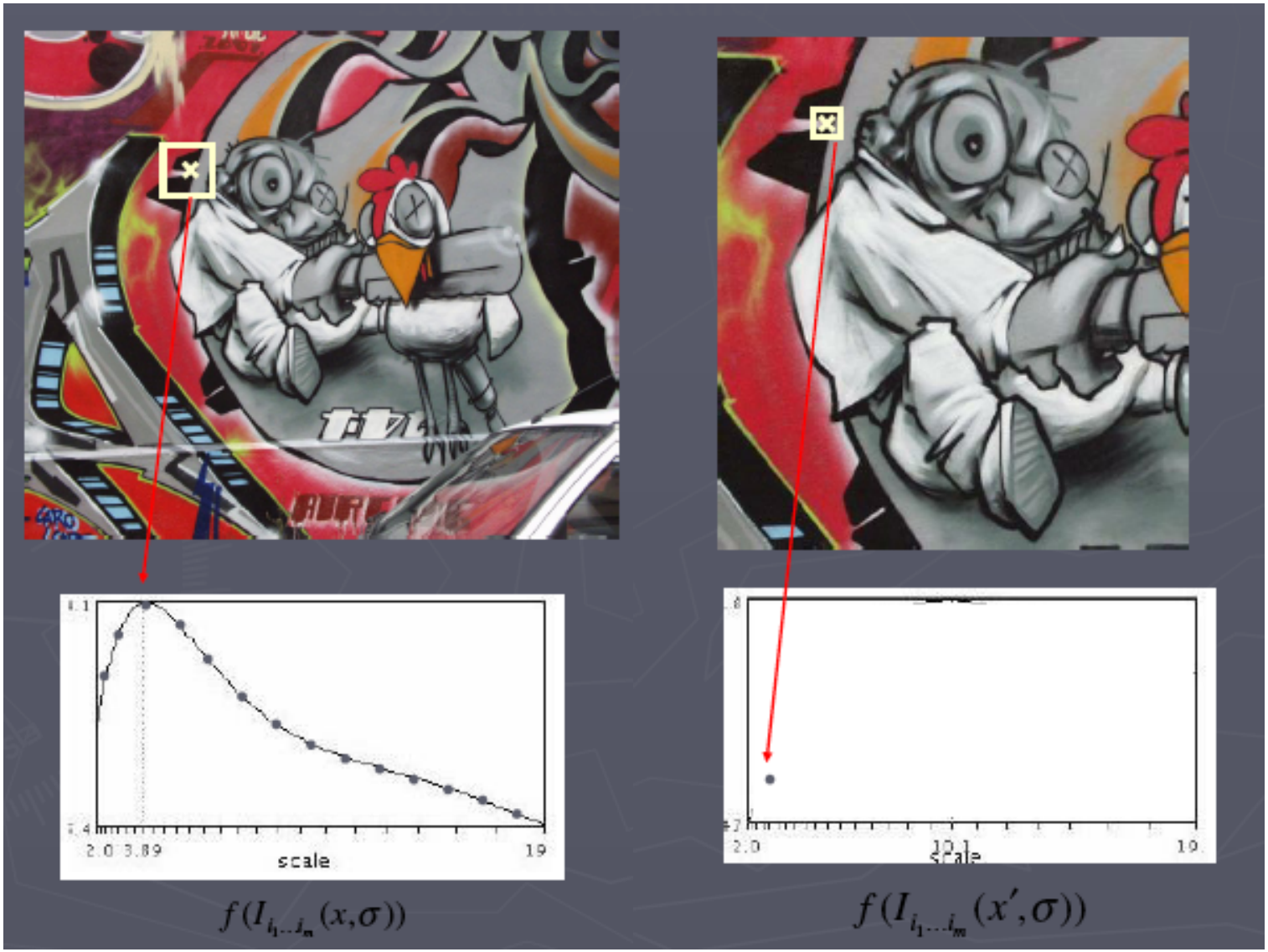
# Automatic Scale Selection

Function responses for increasing scale (scale signature).



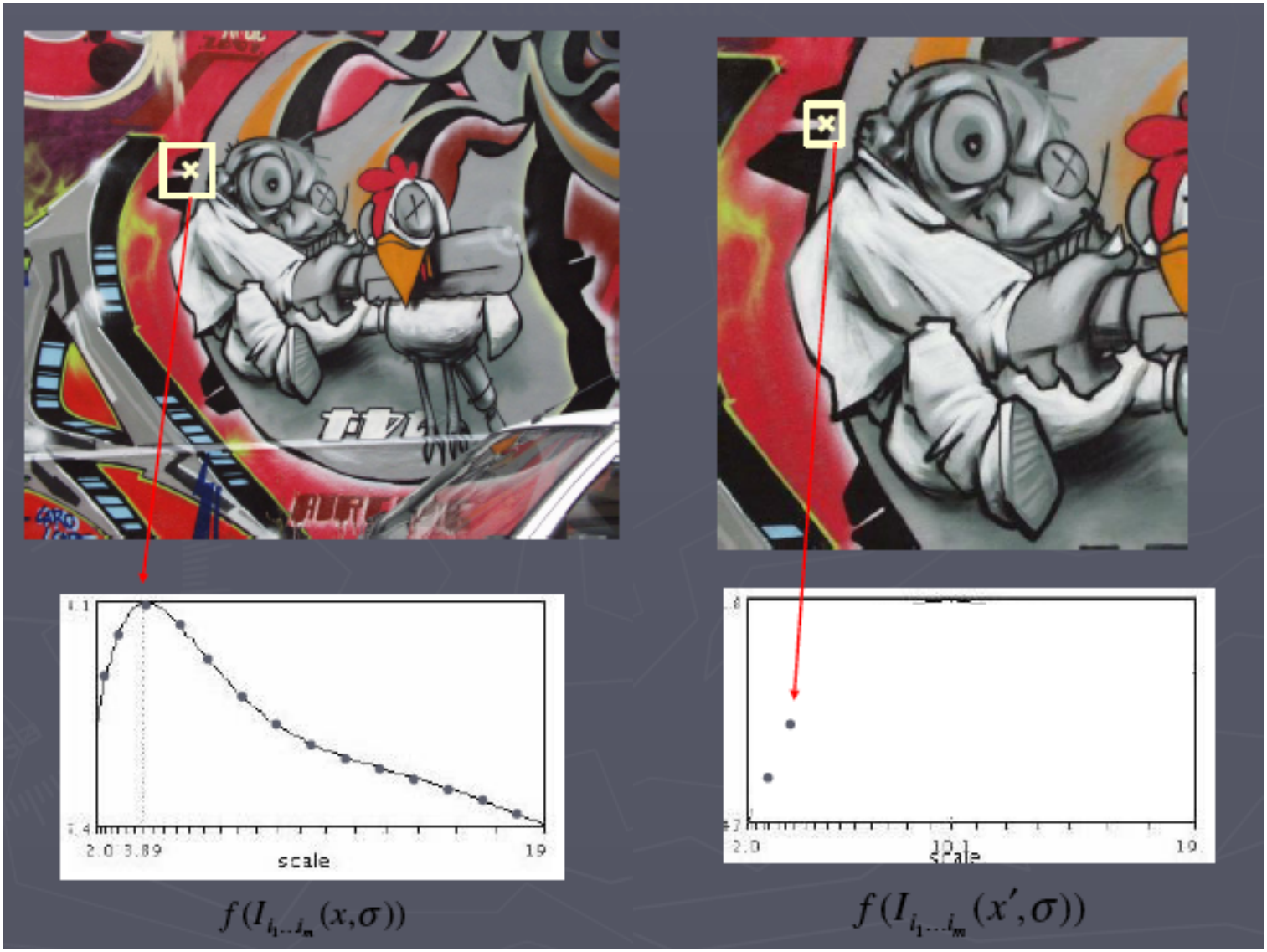
# Automatic Scale Selection

Function responses for increasing scale (scale signature).



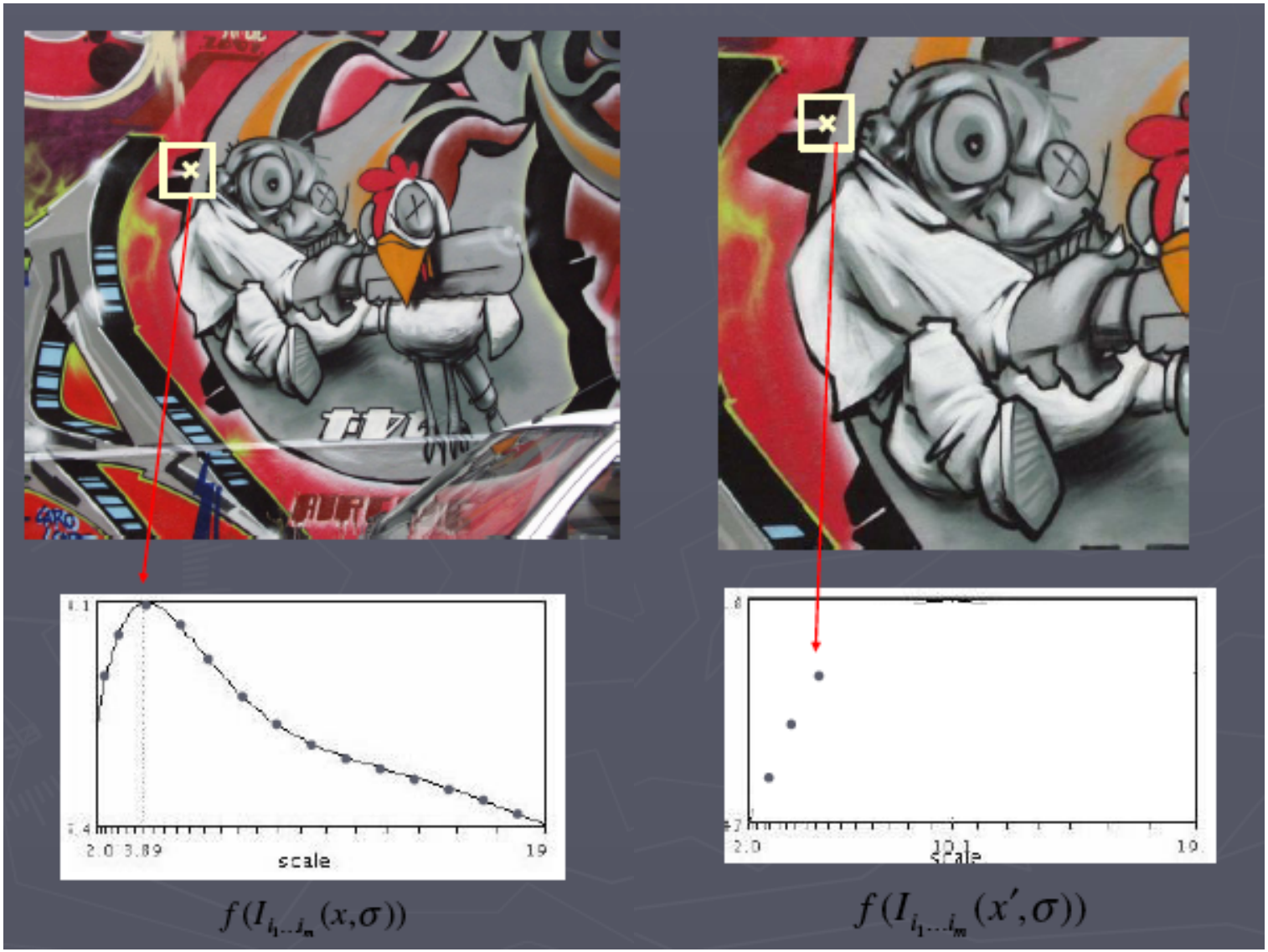
# Automatic Scale Selection

Function responses for increasing scale (scale signature).



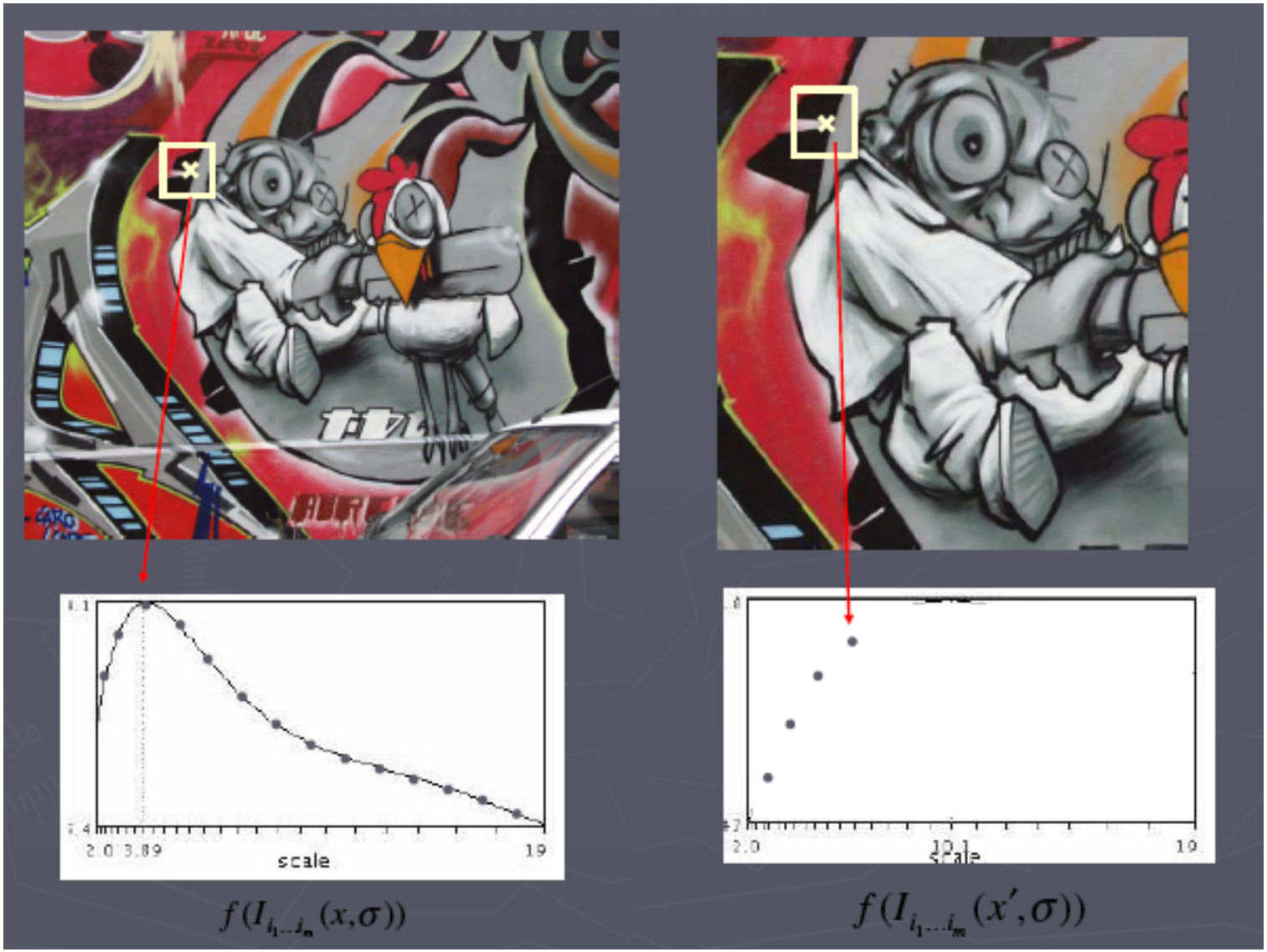
# Automatic Scale Selection

Function responses for increasing scale (scale signature).



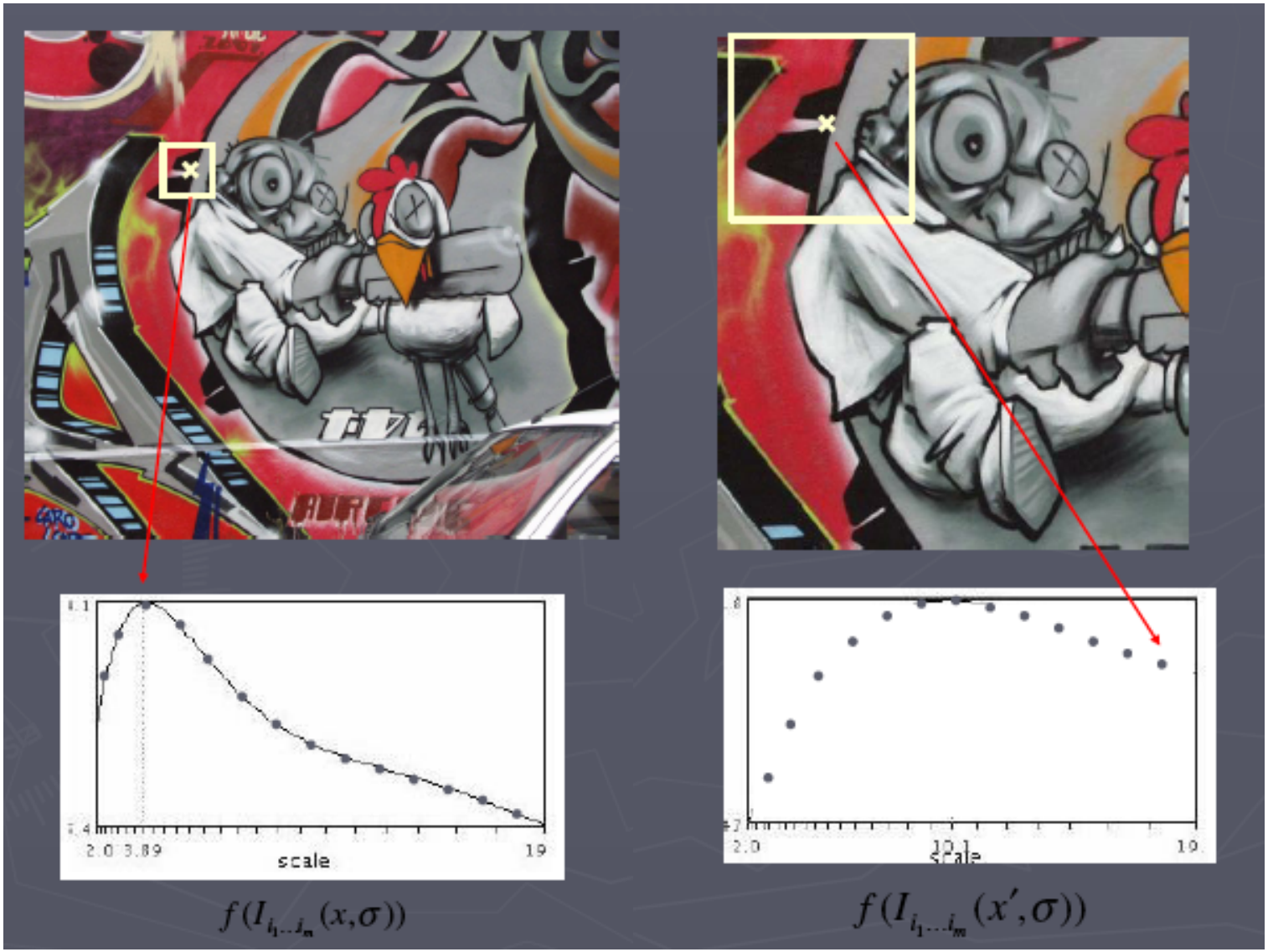
# Automatic Scale Selection

Function responses for increasing scale (scale signature).



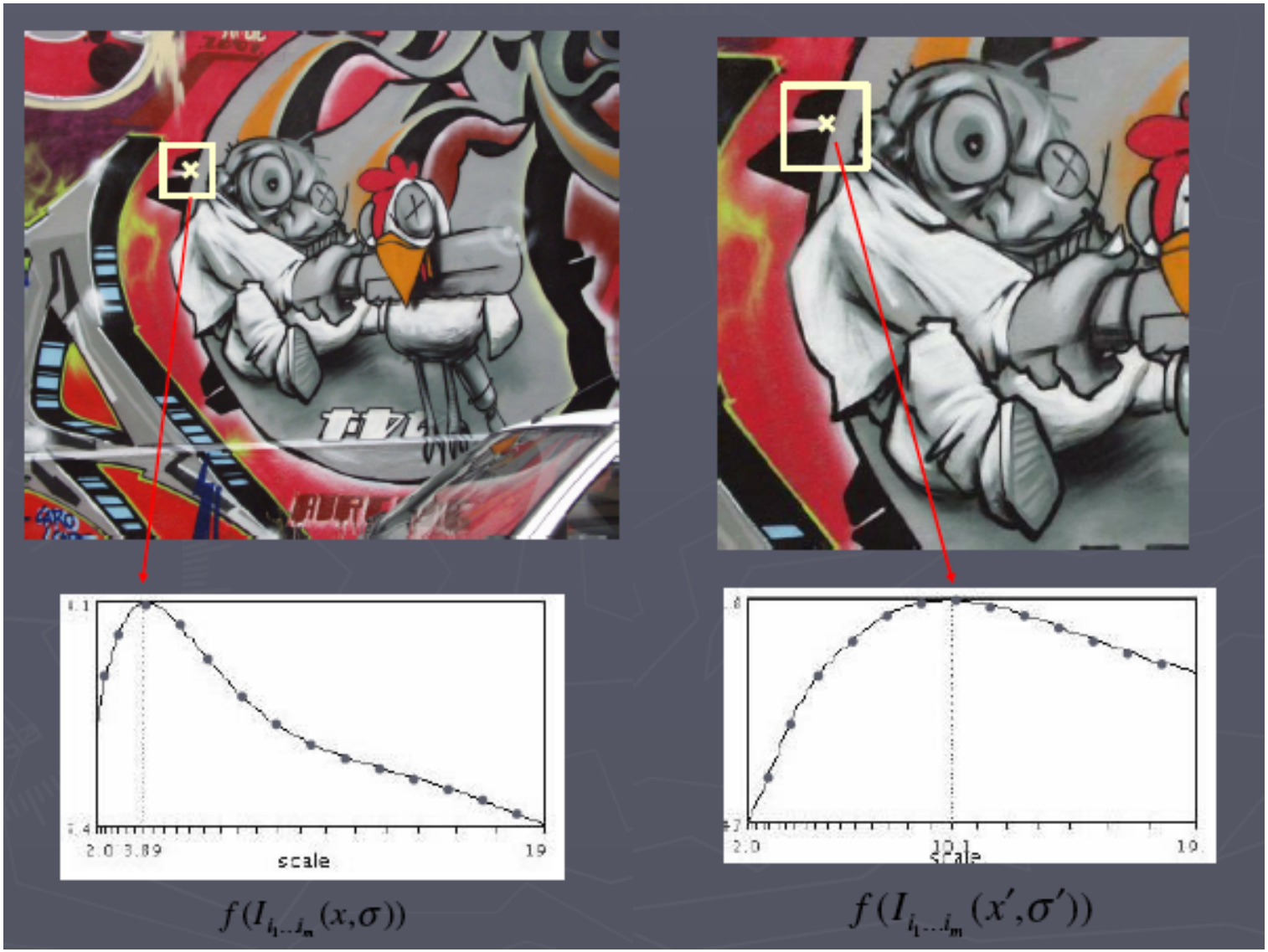
# Automatic Scale Selection

Function responses for increasing scale (scale signature).



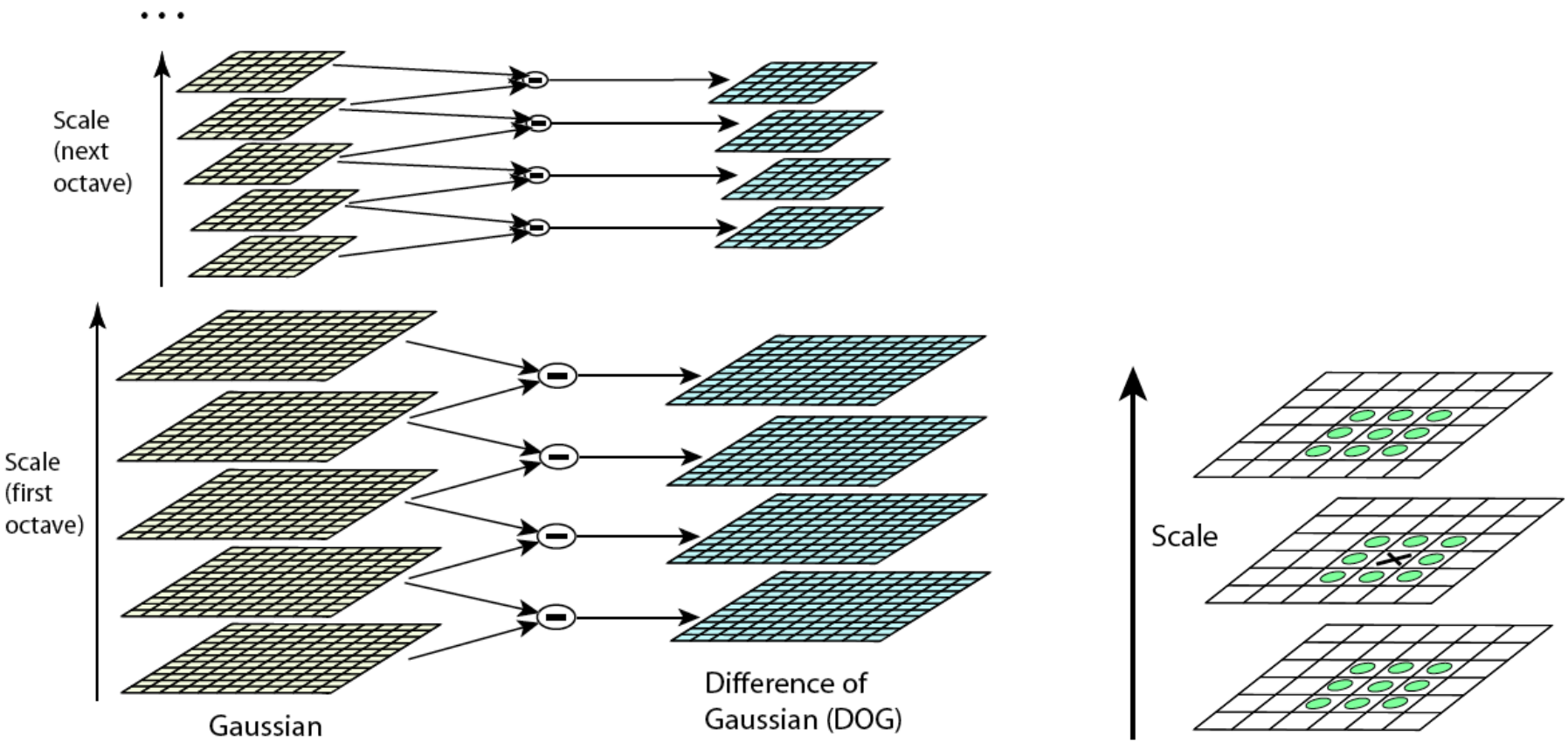
# Automatic Scale Selection

Function responses for increasing scale (scale signature).



# What Can the Signature Function Be?

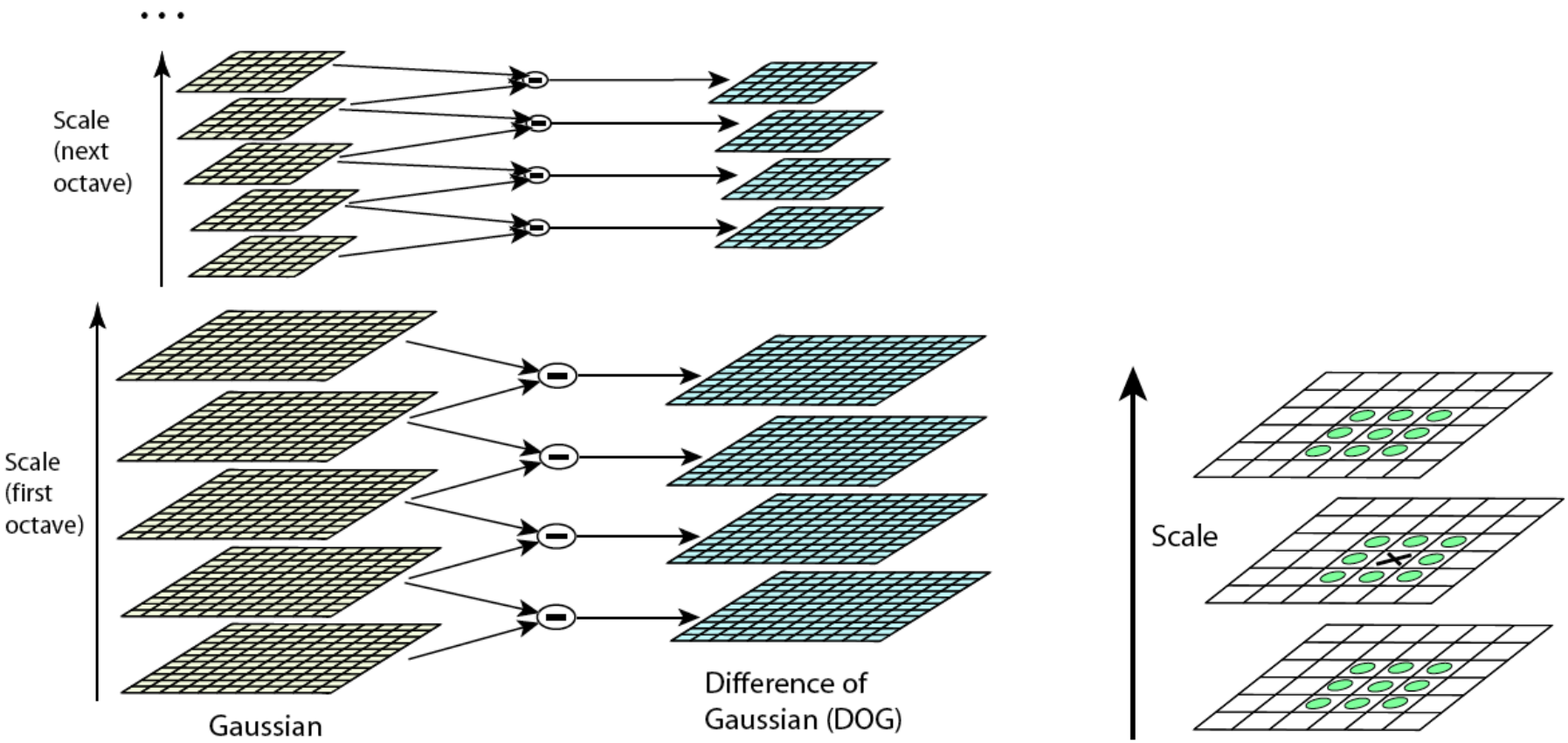
- Lindeberg (1998): extrema in the Laplacian of Gaussian (LoG).
- Lowe (2004) proposed computing a set of sub-octave Difference of Gaussian filters looking for 3D (space+scale) maxima in the resulting structure.




[Source: R. Szeliski, slide credit: R. Urtasun]

# What Can the Signature Function Be?

- Lindeberg (1998): extrema in the Laplacian of Gaussian (LoG).
- Lowe (2004) proposed computing a set of sub-octave Difference of Gaussian filters looking for 3D (space+scale) maxima in the resulting structure.



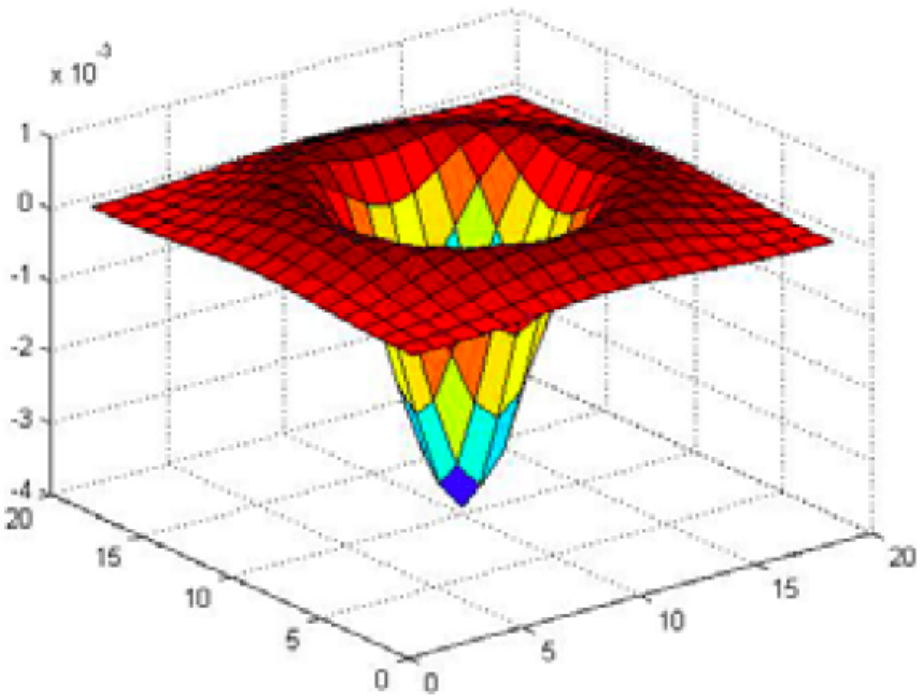

[Source: R. Szeliski, slide credit: R. Urtasun]

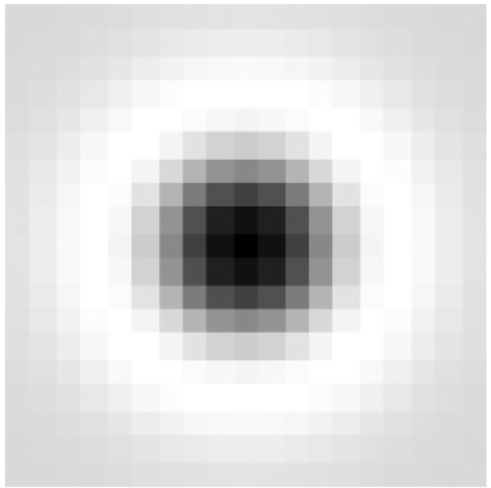
# Blob Detection – Laplacian of Gaussian

- Laplacian of Gaussian: We mentioned it for edge detection

$$\nabla^2 g(x, y, \sigma) = \frac{\partial^2 g(x, y, \sigma)}{\partial x^2} + \frac{\partial^2 g(x, y, \sigma)}{\partial y^2}, \qquad \text{where } g \text{ is a Gaussian}$$

- It is a circularly symmetric operator (finds difference in all directions)
- It can be used for 2D blob detection! How?



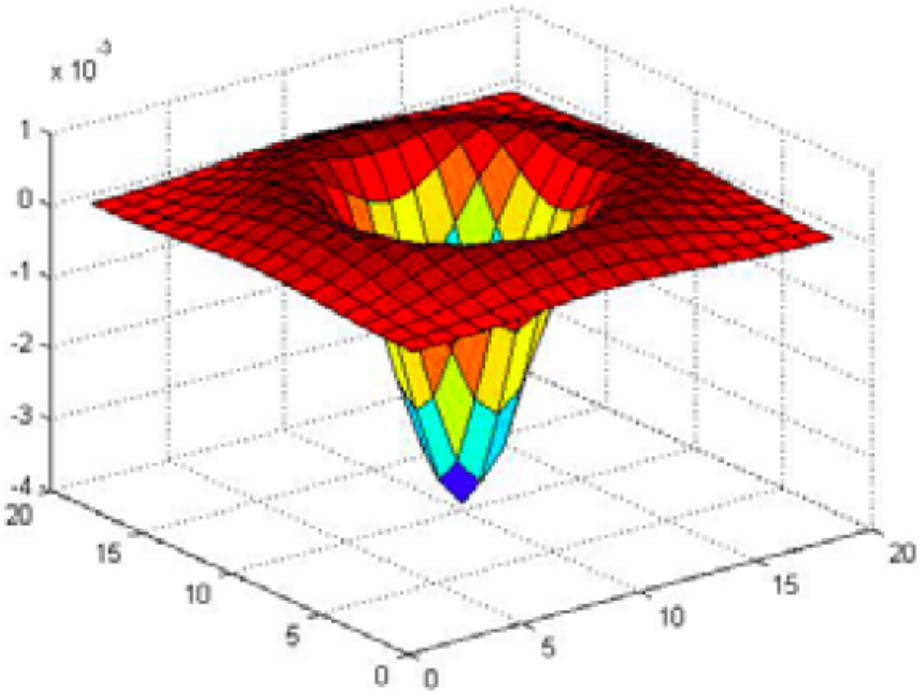



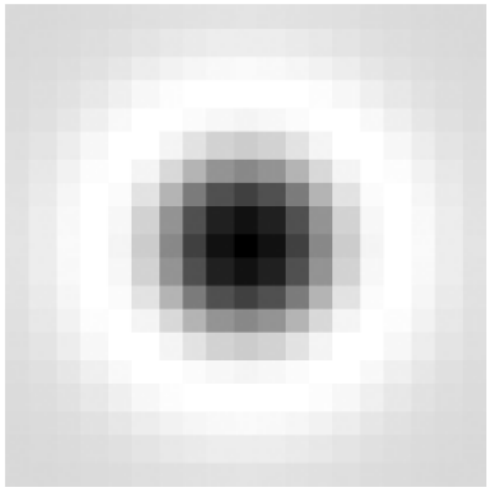
# Blob Detection – Laplacian of Gaussian

- Laplacian of Gaussian: We mentioned it for edge detection

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

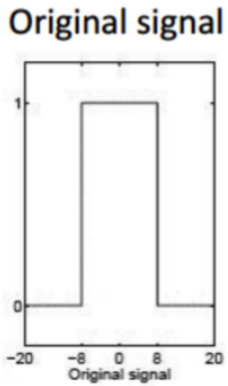
- It is a circularly symmetric operator (finds difference in all directions)
- It can be used for 2D blob detection! How?

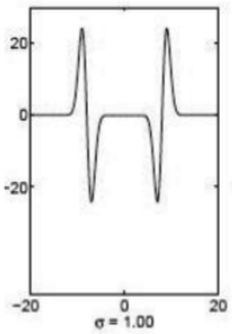


# Blob Detection – Laplacian of Gaussian

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

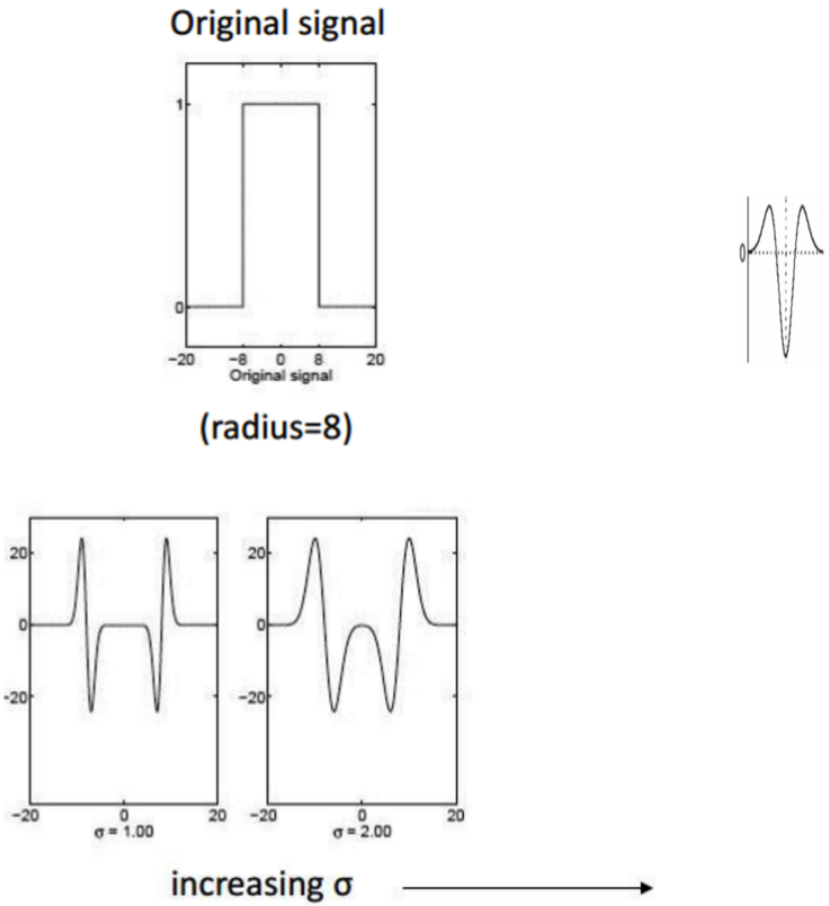



[Source: F. Flores-Mangas]

# Blob Detection – Laplacian of Gaussian

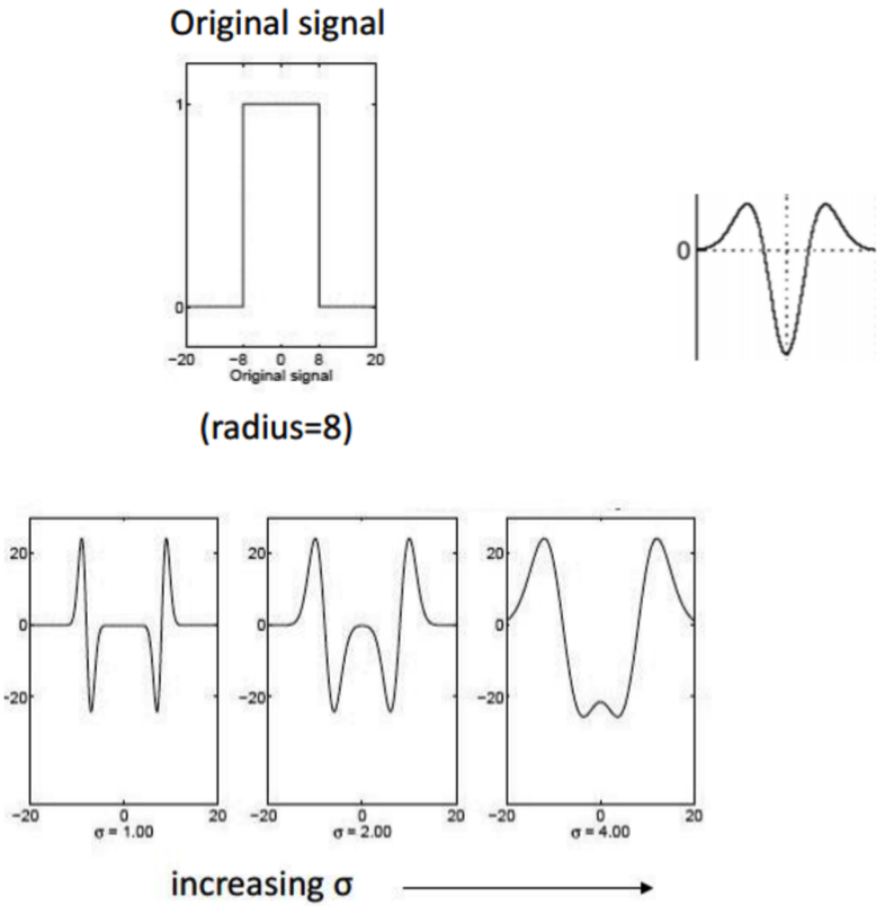
- It can be used for 2D blob detection! How?




[Source: F. Flores-Mangas]

# Blob Detection – Laplacian of Gaussian

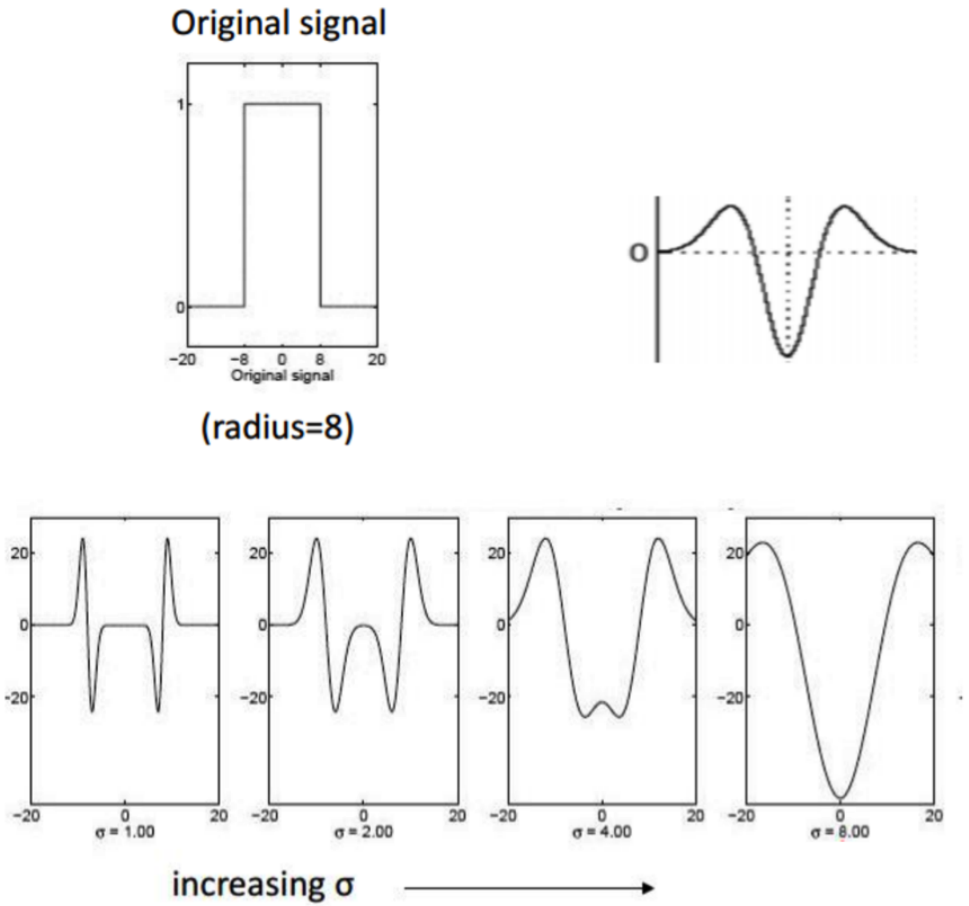
- It can be used for 2D blob detection! How?




[Source: F. Flores-Mangas]

# Blob Detection – Laplacian of Gaussian

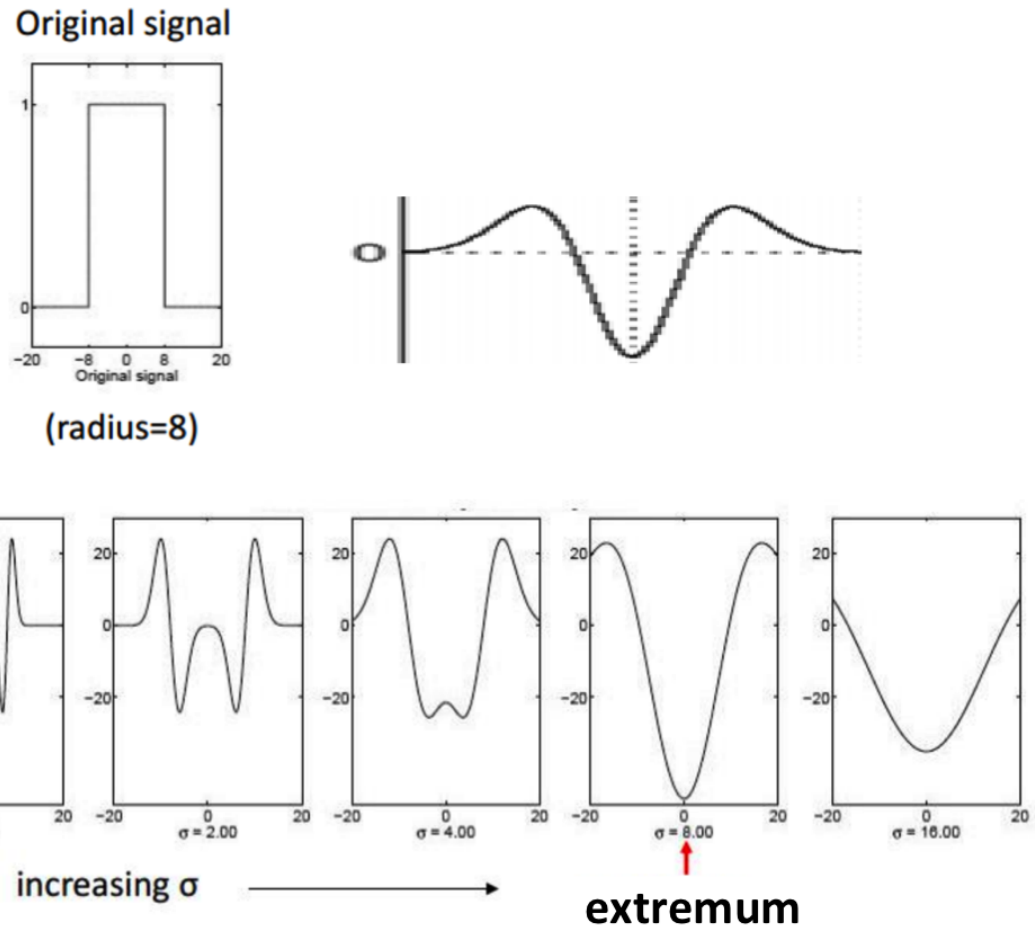
- It can be used for 2D blob detection! How?




[Source: F. Flores-Mangas]

# Blob Detection – Laplacian of Gaussian

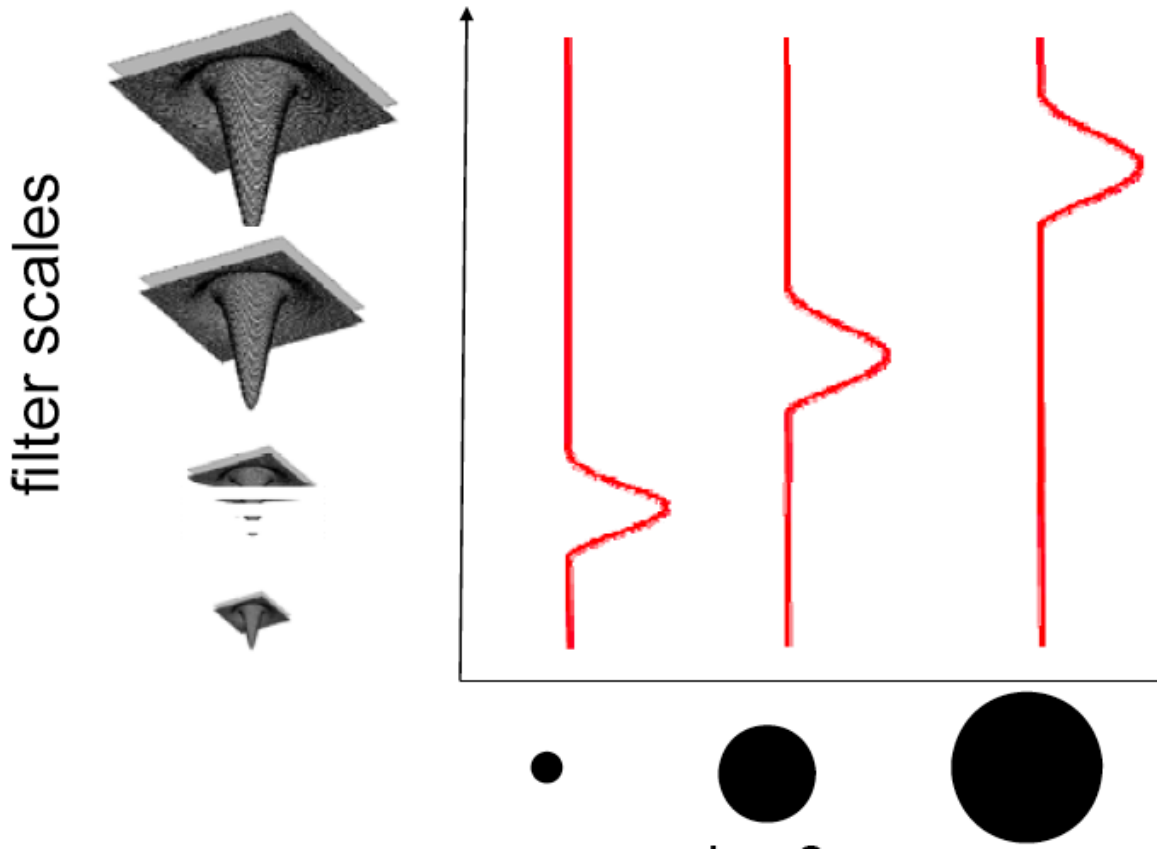
- It can be used for 2D blob detection! How?




[Source: F. Flores-Mangas]

# Blob Detection in 2D: Scale Selection

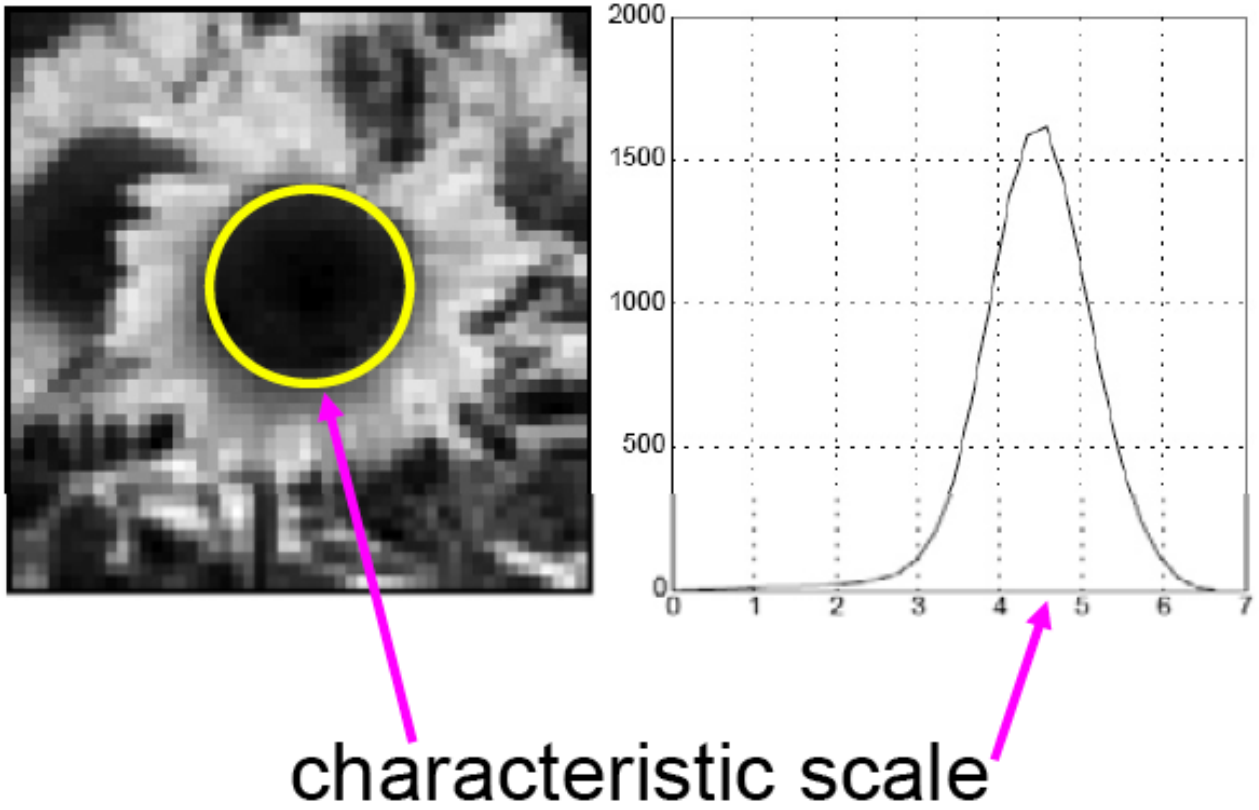
Laplacian of Gaussian = blob detector




[Source: B. Leibe, slide credit: R. Urtasun]

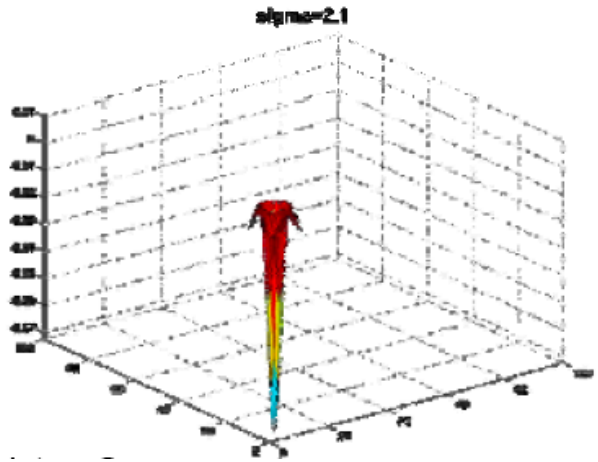
# Characteristic Scale

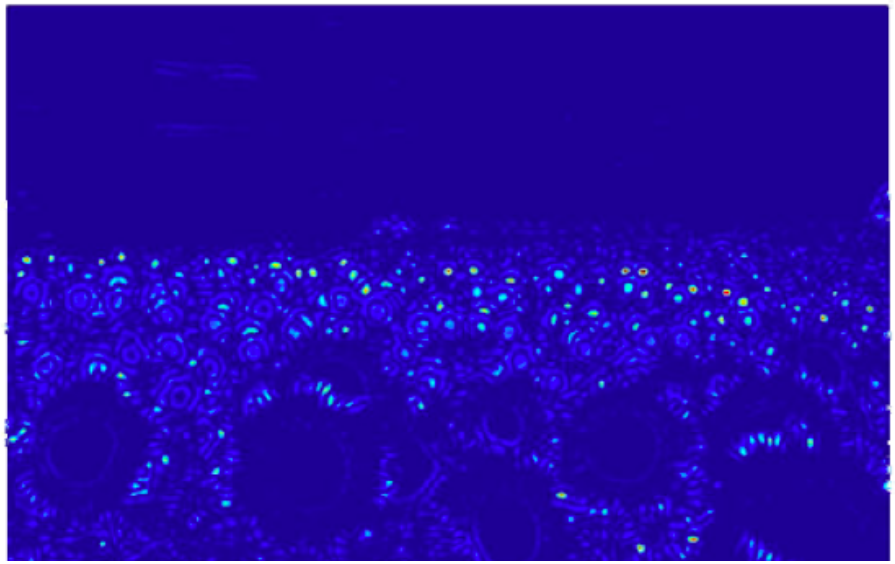
- We define the **characteristic scale** as the scale that produces peak (minimum or maximum) of the Laplacian response




[Source: S. Lazebnik]

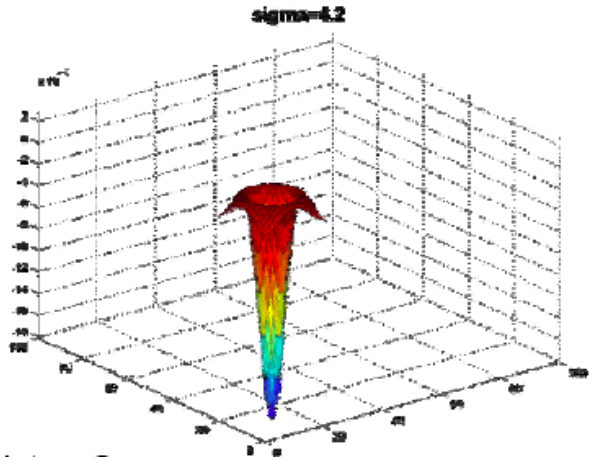
# Example

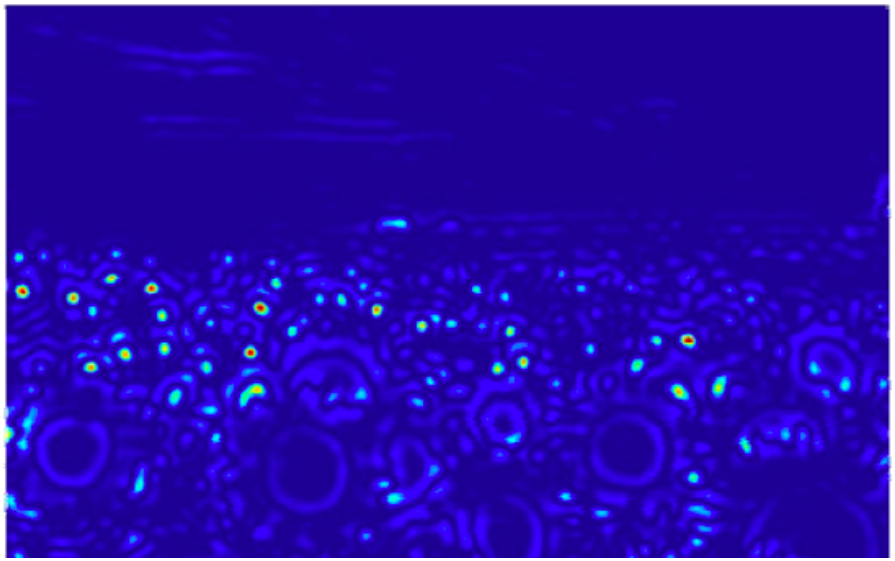


[Source: K. Grauman]

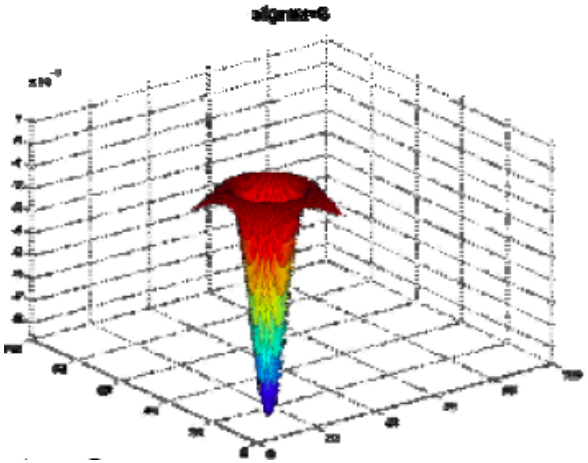
# Example

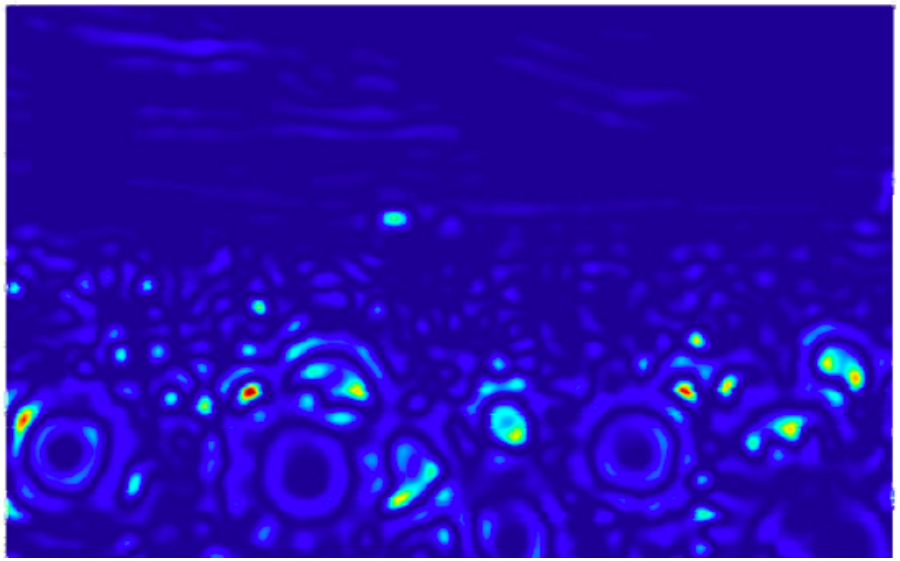


[Source: K. Grauman]

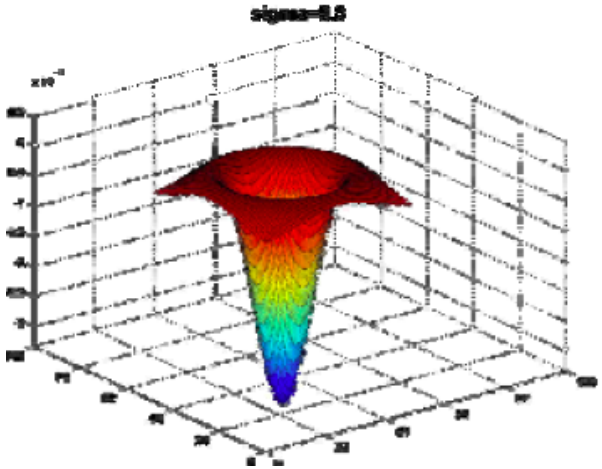
# Example

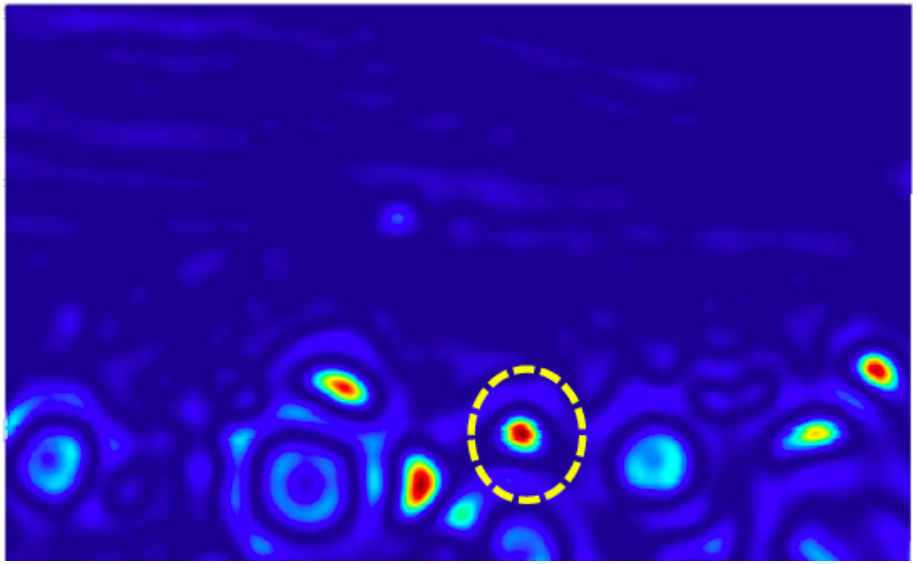




[Source: K. Grauman]

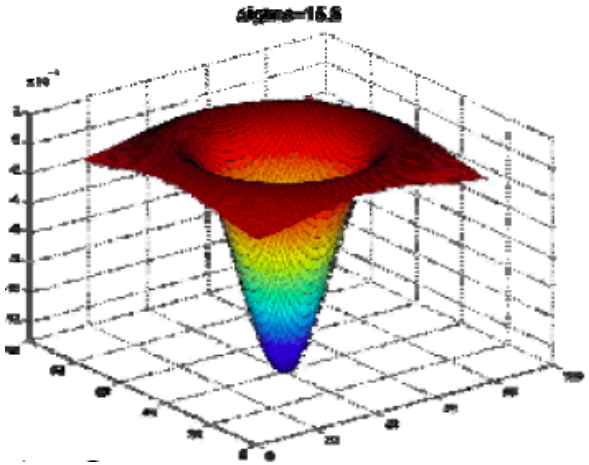
# Example

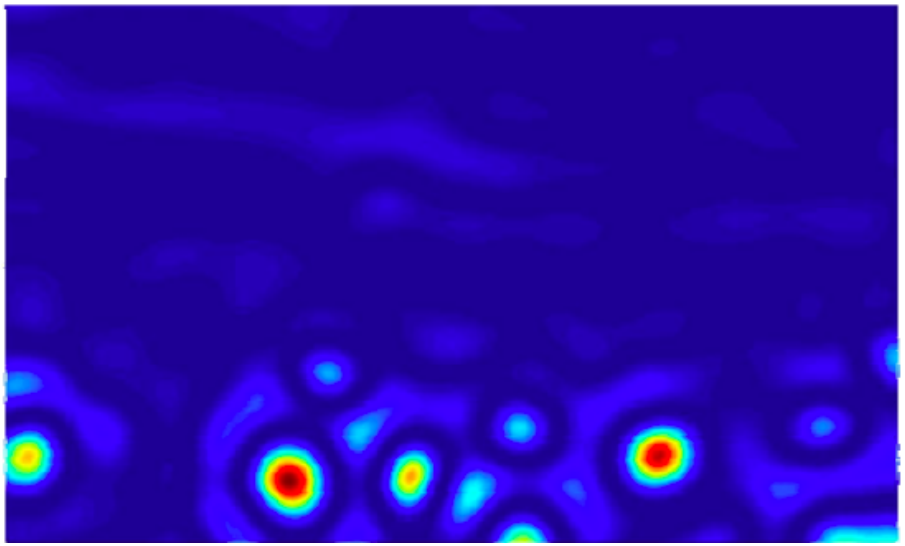




[Source: K. Grauman]

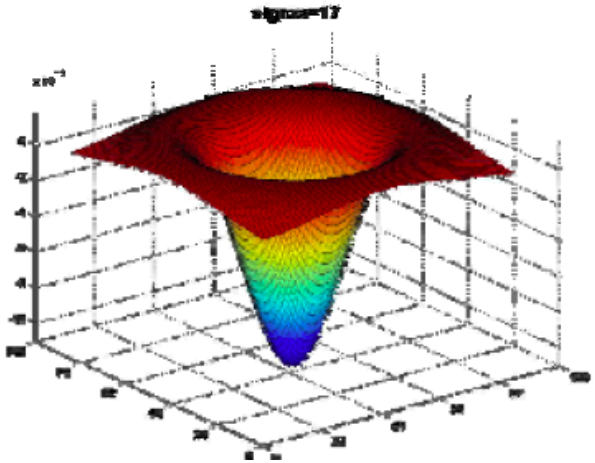
# Example

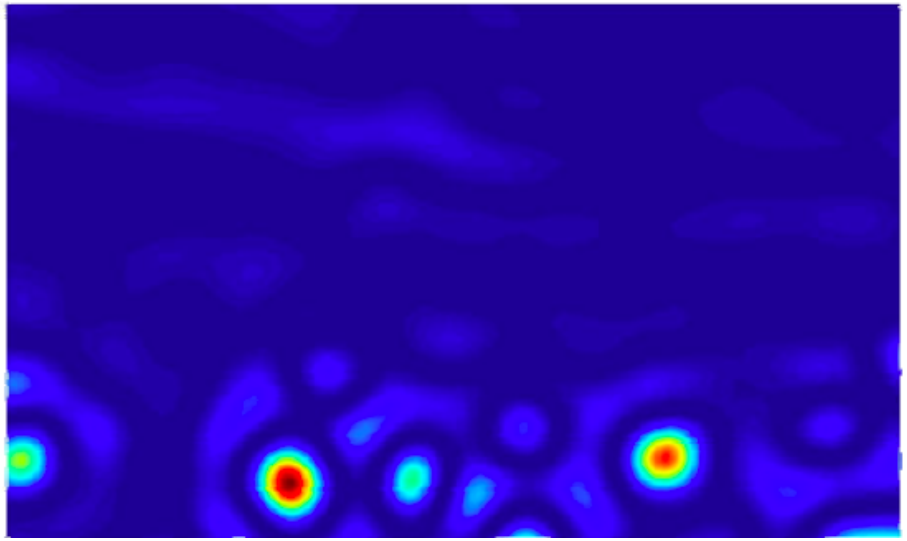




[Source: K. Grauman]

# Example





[Source: K. Grauman]

# Scale Invariant Interest Points

Interest points are local maxima in both position and scale.

$L_{xx}(\sigma) + L_{yy}(\sigma)$

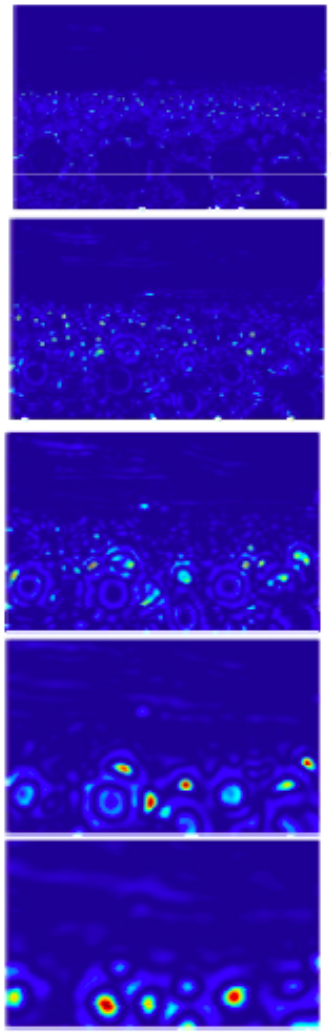
σ5

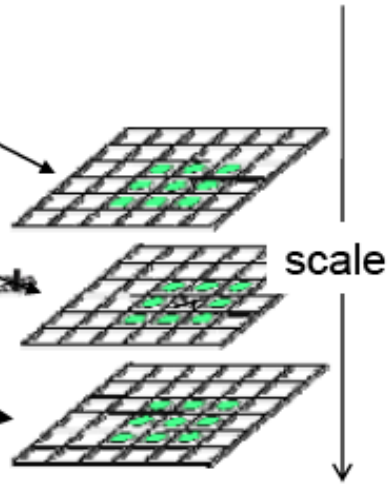
σ4

σ3

σ2

σ1





scale

⇒ List of
*(x, y, σ)*

Squared filter response maps

Kristen Grauman

# Example

[Source: S. Lazebnik]

# Blob Detection – Laplacian of Gaussian

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

$$\nabla^2 g(x, y, \sigma) = \frac{\partial^2 g(x, y, \sigma)}{\partial x^2} + \frac{\partial^2 g(x, y, \sigma)}{\partial y^2}, \quad \text{where } g \text{ is a Gaussian}$$

# Blob Detection – Laplacian of Gaussian

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

$$\nabla^2 g(x, y, \sigma) = \frac{\partial^2 g(x, y, \sigma)}{\partial x^2} + \frac{\partial^2 g(x, y, \sigma)}{\partial y^2}, \quad \text{where } g \text{ is a Gaussian}$$

- So computing our interest points means two convolutions (one for each derivative) **per scale**
- Larger scale ($\sigma$), larger the filters (more work for convolution)
- Can we do it faster?

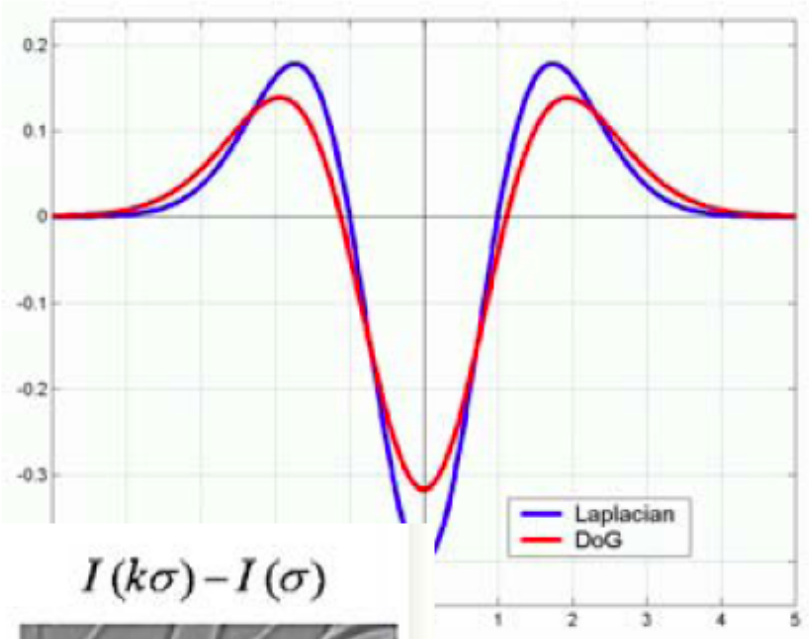
# Approximate the Laplacian of Gaussian

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

(Laplacian)

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

(Difference of Gaussians)

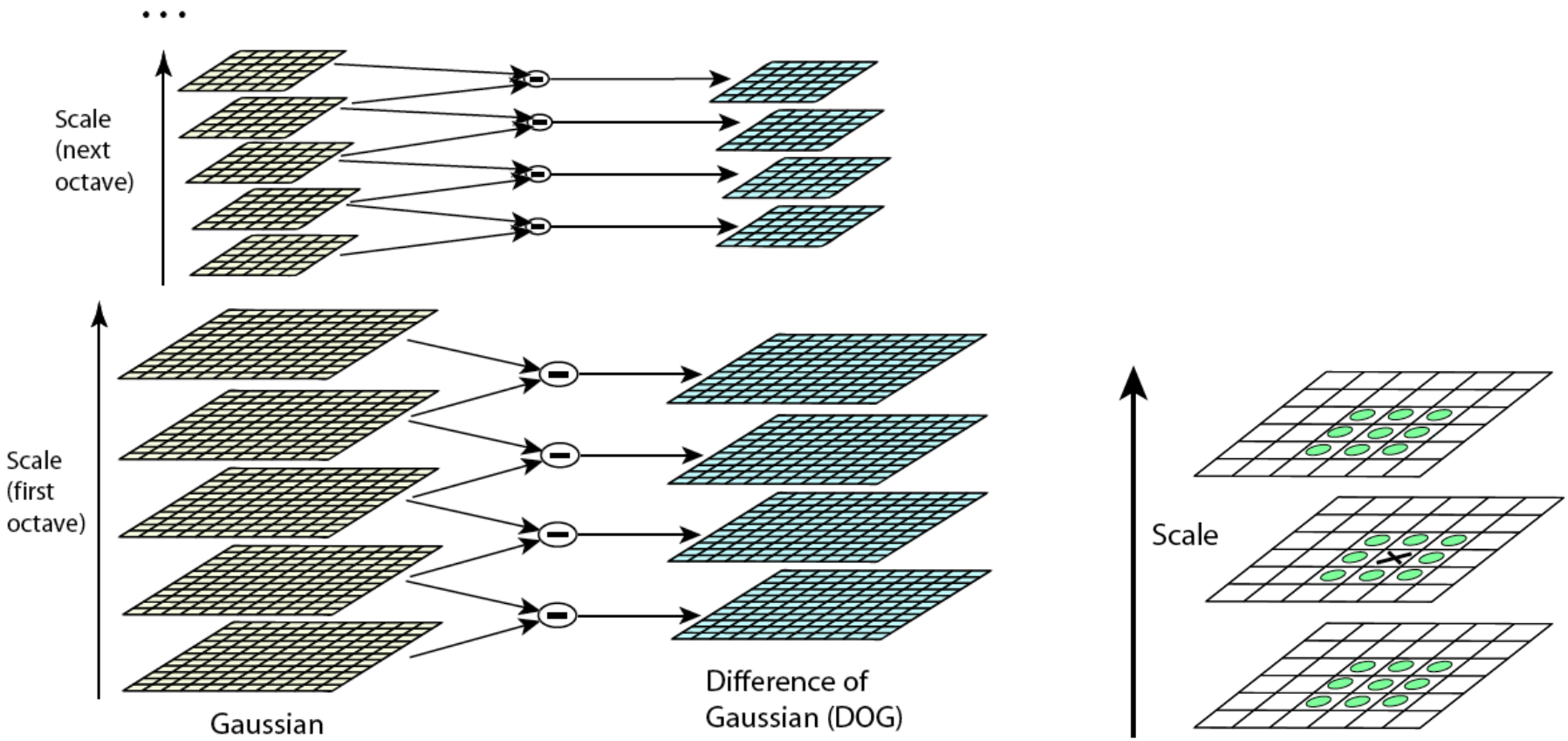



$I(k\sigma)$ $I(\sigma)$ $I(k\sigma) - I(\sigma)$


[Source: K. Grauman]

# Lowe's DoG

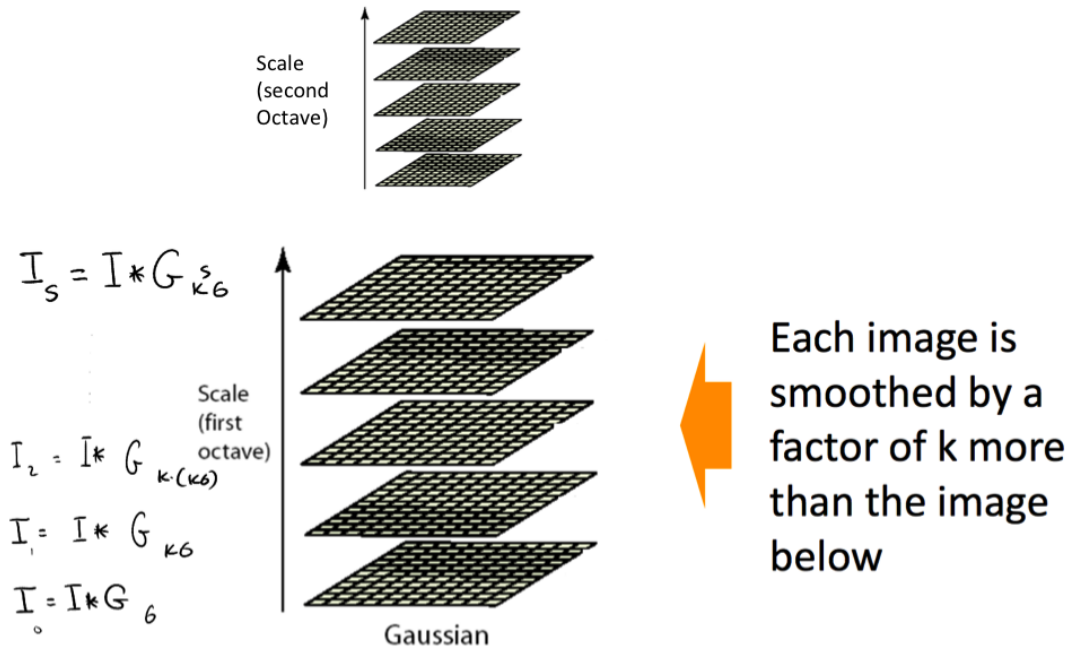
- Lowe (2004) proposed computing a set of sub-octave Difference of Gaussian filters looking for 3D (space+scale) maxima in the resulting structure




[Source: R. Szeliski, slide credit: R. Urtasun]

# Lowe's DoG

- First compute a Gaussian image pyramid



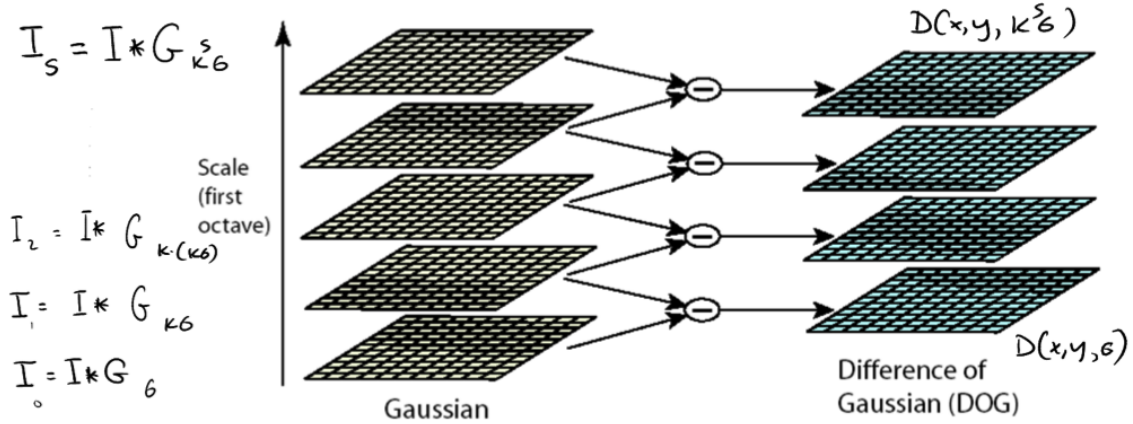

[Source: F. Flores-Mangas]

# Lowe's DoG

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

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

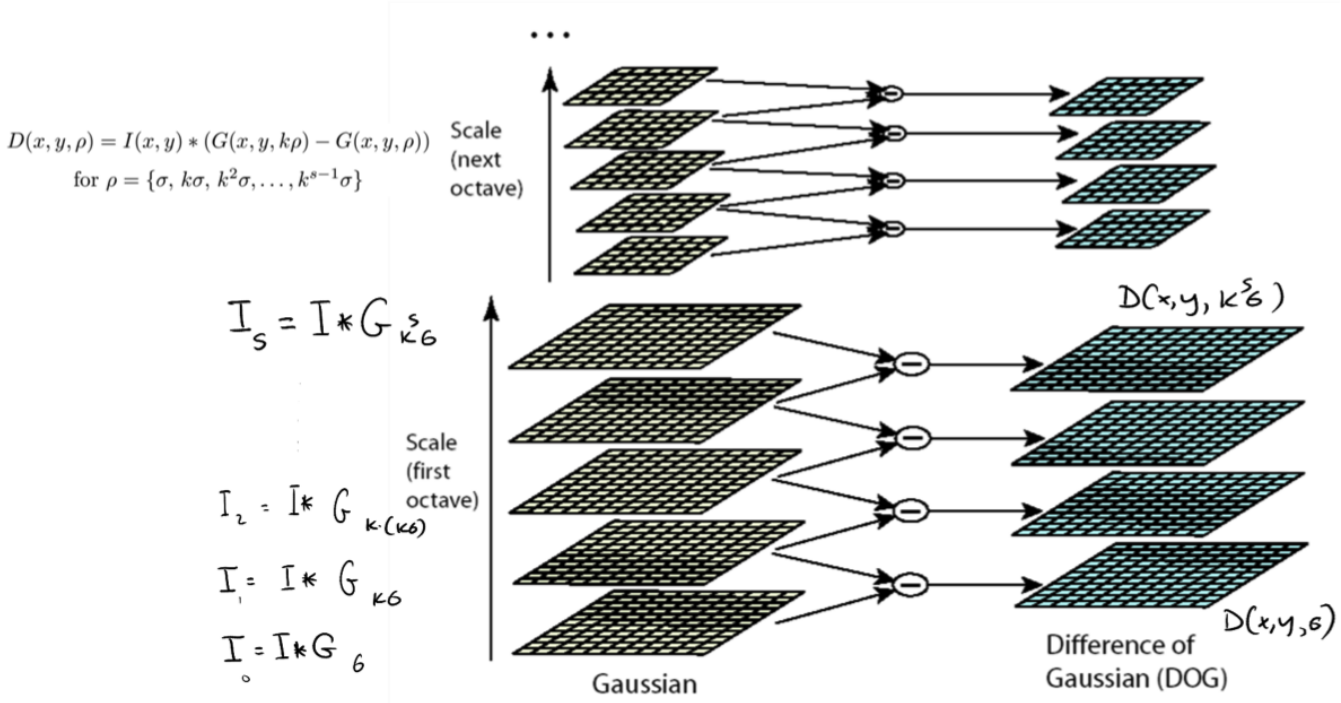
$$\text{for } \rho = \{\sigma, k\sigma, k^2\sigma, \ldots, k^{s-1}\sigma\}, \quad k = 2^{1/s}$$




[Source: F. Flores-Mangas]

# Lowe's DoG

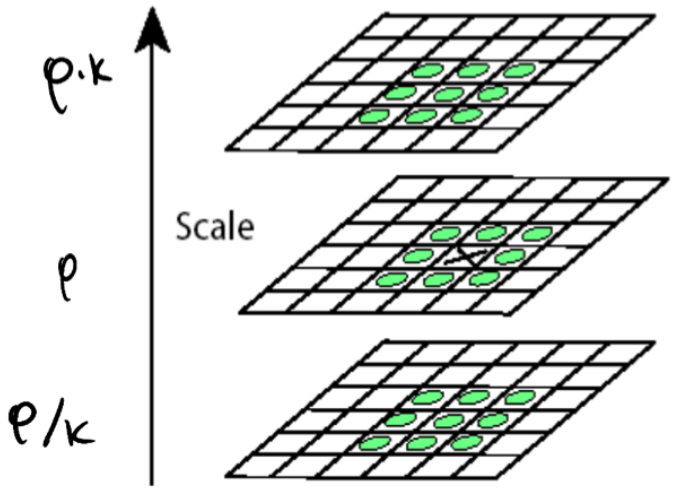
- First compute a Gaussian image pyramid
- Compute Difference of Gaussians
- At every scale




[Source: F. Flores-Mangas]

# Lowe's DoG

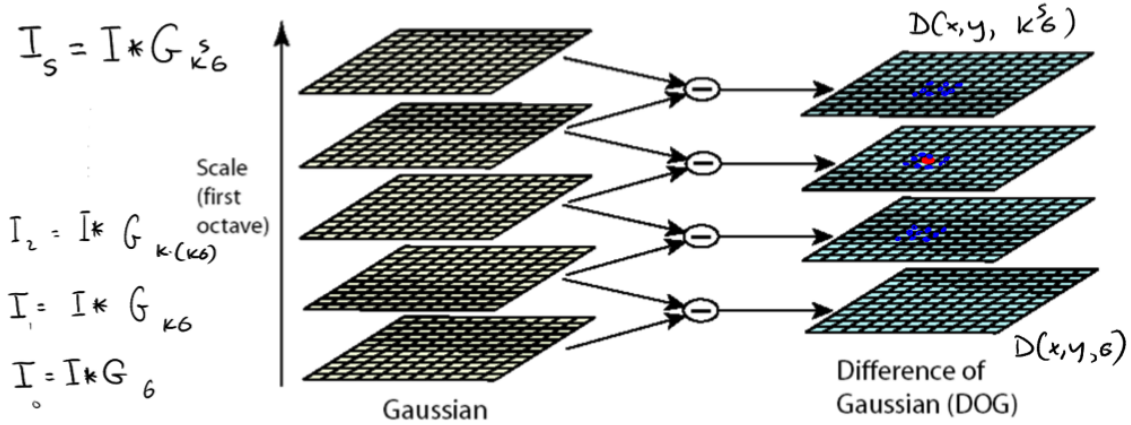
- First compute a Gaussian image pyramid
- Compute Difference of Gaussians
- At every scale
- Find local maxima in scale
- A bit of pruning of bad maxima and we're done!




[Source: F. Flores-Mangas]

# 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

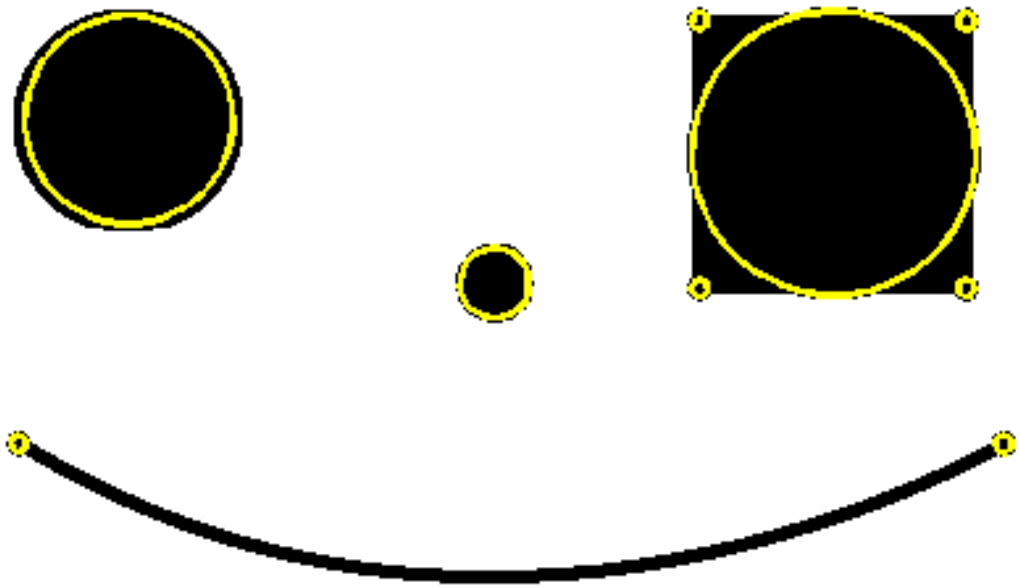


Figure: Detected interest points (kind of make sense)

# Examples

Figure: Other roundy objects

# Examples

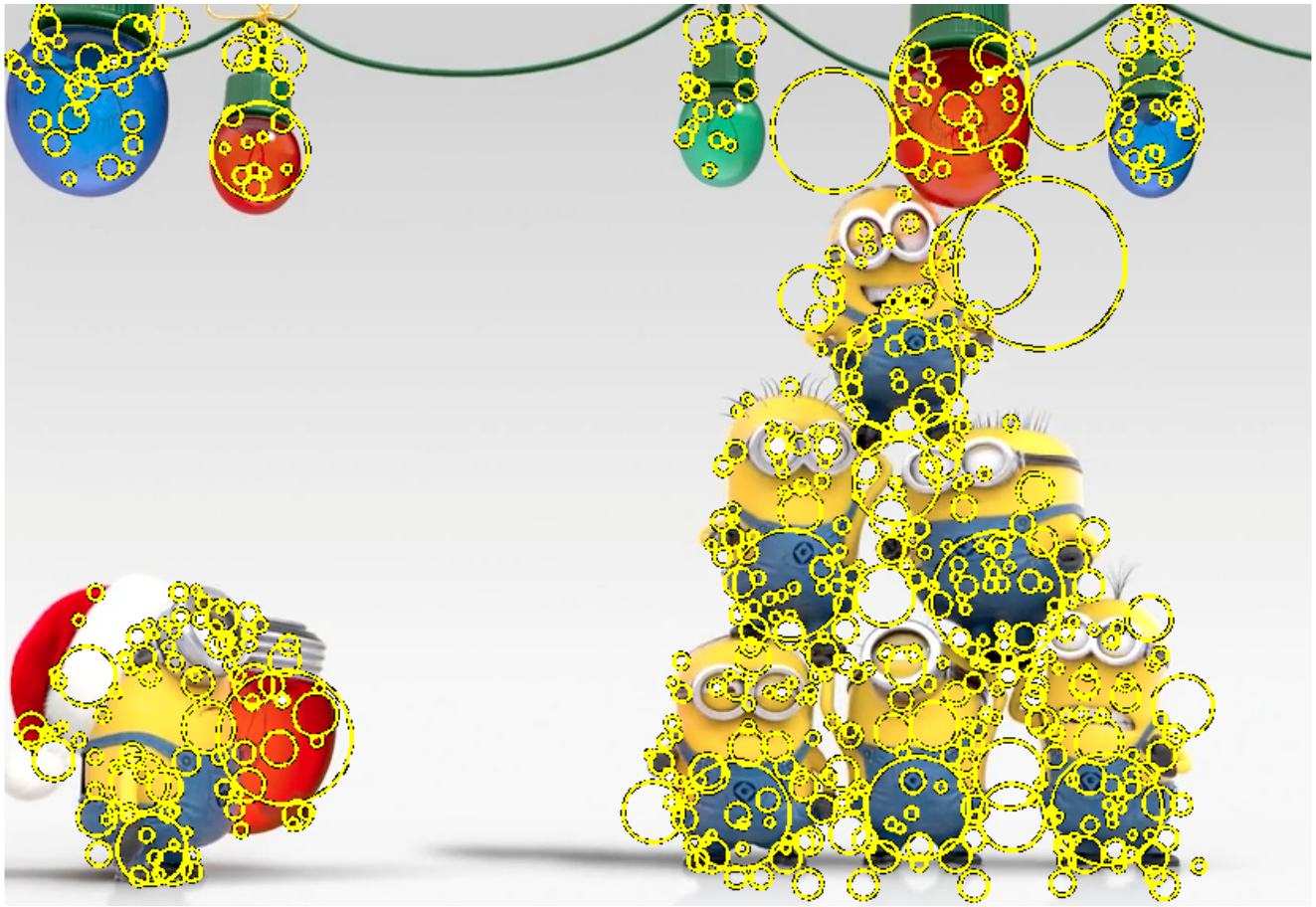


Figure: Detected interest points

# Examples

Figure: Real images

# Examples

Figure: Detected interest points

# Examples

Figure: Detected interest points

# 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
- Tuytelaars & 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)
- 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 $M_{xy} = \begin{pmatrix} A(x,y) & B(x,y) \\ B(x,y) & C(x,y) \end{pmatrix}$ characterizes the shape of $E_{WSSD}$ for a window around $(x, y)$. Compute "cornerness" score for each $(x, y)$ as $R(x, y) = \det(M_{xy}) - \alpha \operatorname{trace}(M_{xy})^2$. Find $R(x, y) > \text{threshold}$ 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

# Matching Results Based on SIFT Interest Points

- Works pretty well in variety of settings

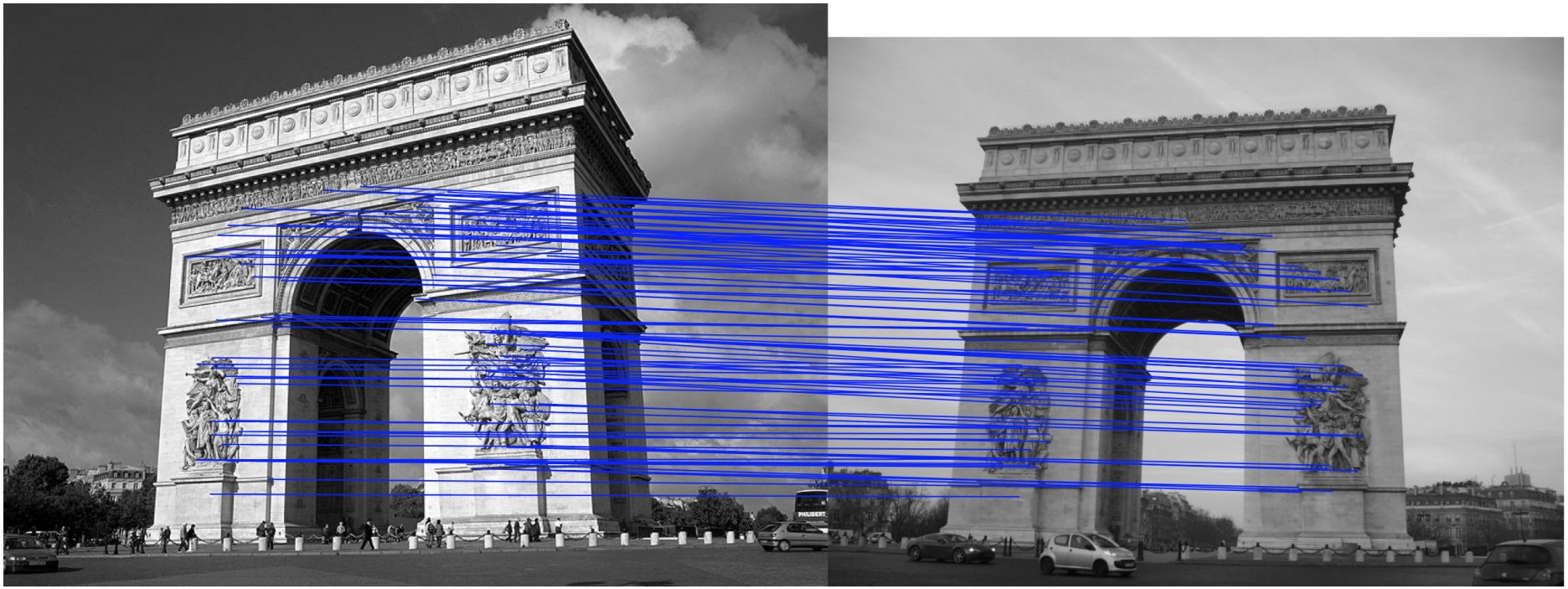


Figure: Lowe's interest point detector finds scale-invariant points that can be reliably matched across different images. (We will talk about how to do matching soon)

# Matching Results Based on SIFT Interest Points

- Works pretty well in variety of settings

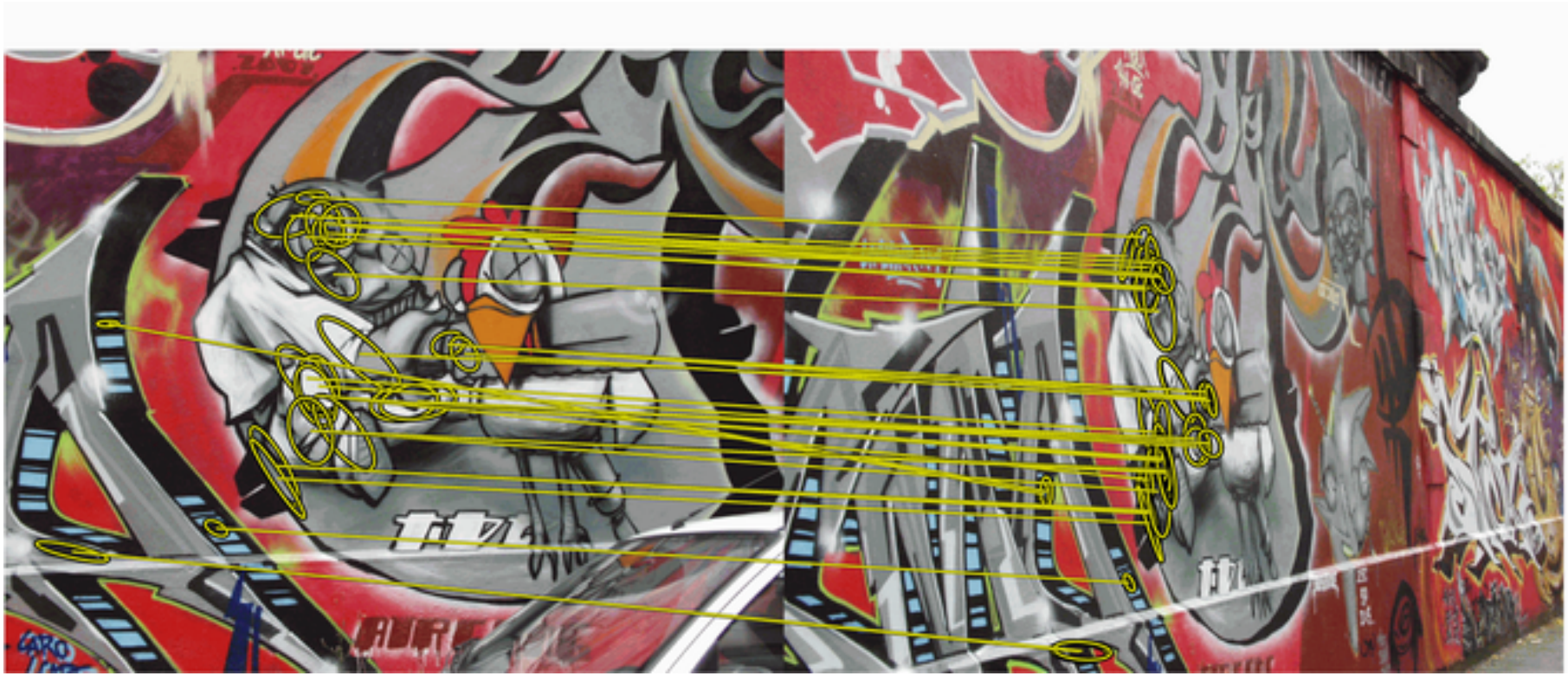


Figure: Lowe's interest point detector finds scale-invariant points that can be reliably matched across different images. (We will talk about how to do

# Matching Results Based on SIFT Interest Points

- Works pretty well in variety of settings

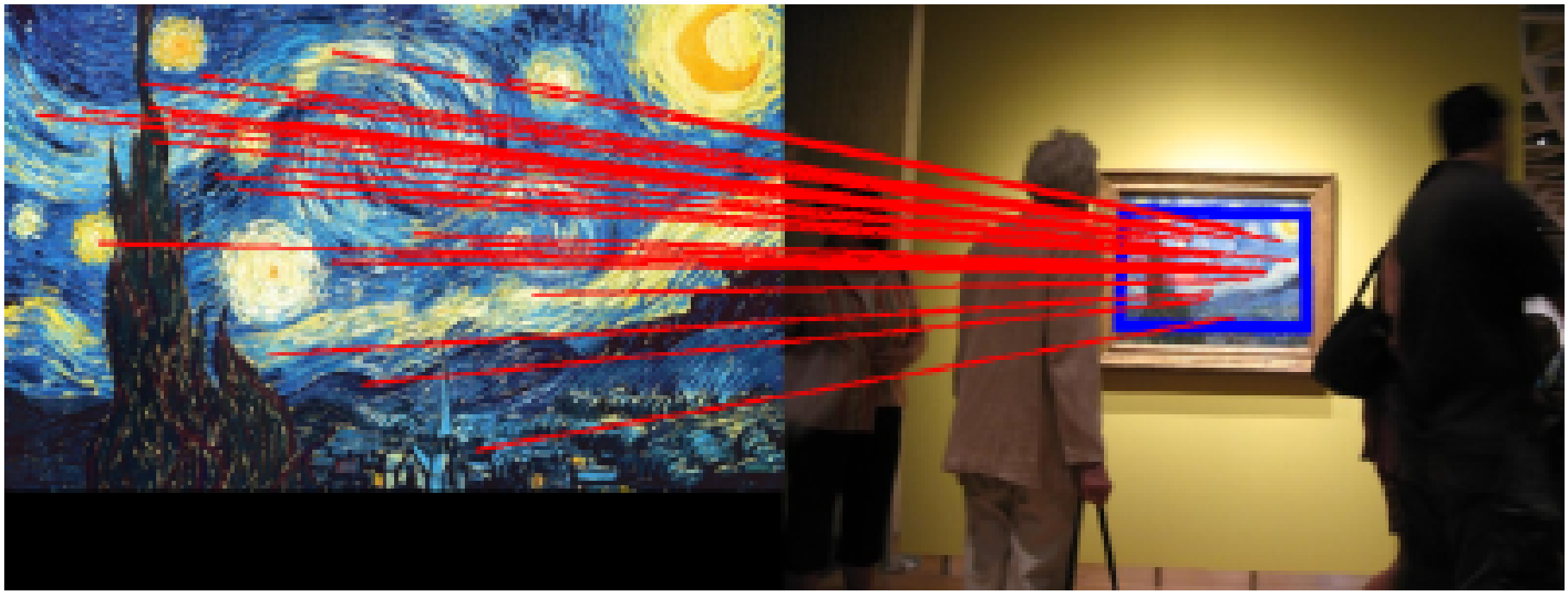


Figure: Lowe's interest point detector finds scale-invariant points that can be reliably matched across different images. (We will talk about how to do matching soon)

# Matching Results Based on SIFT Interest Points

- What about in different lighting/weather conditions?

# Matching Results Based on SIFT Interest Points

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

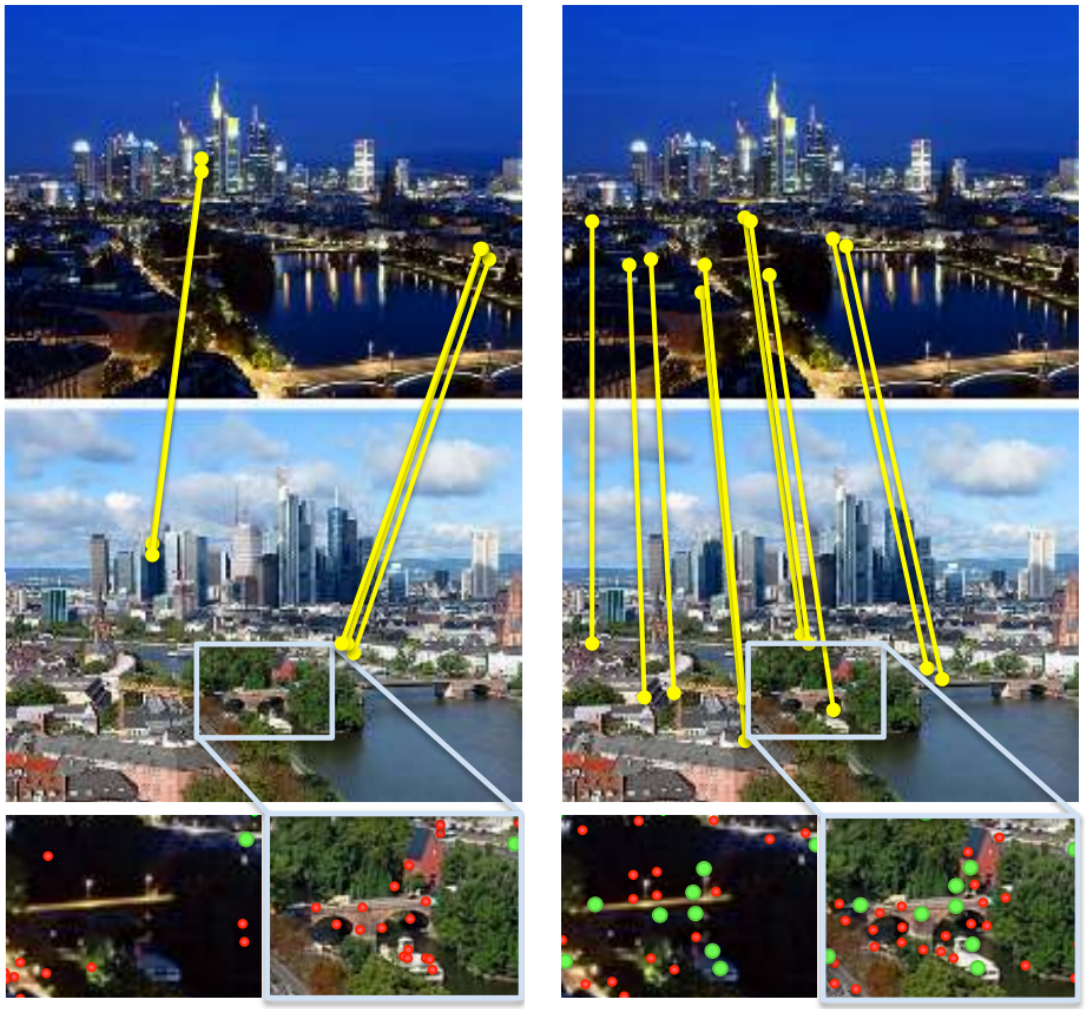
- Fails in very different lighting conditions

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

# Machine Learning to Find Better Keypoints?

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

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



SIFT Learned Interest Point Detector?

# Learning an Interest Point Detector:

Y. Verdie, K. M. Yi, P. Fua and V. Lepetit

Large *TILDE: A Temporally Invariant Learned DEtector*

CVPR 2015

Paper: http://infoscience.epfl.ch/record/206786/files/top.pdf

Project page & Code: http://cvlab.epfl.ch/research/tilde

# Training Data

- What can we use?

# Training Data

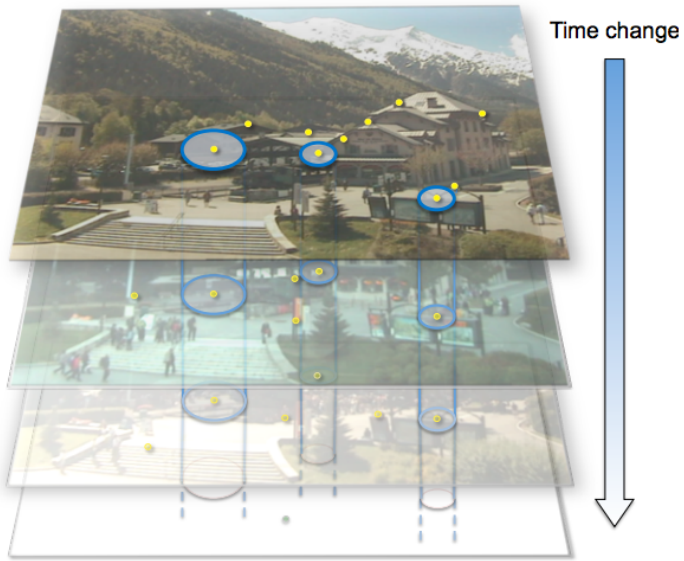
- What can we use? Data from webcams!

# Training Data

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

# Training the Detector

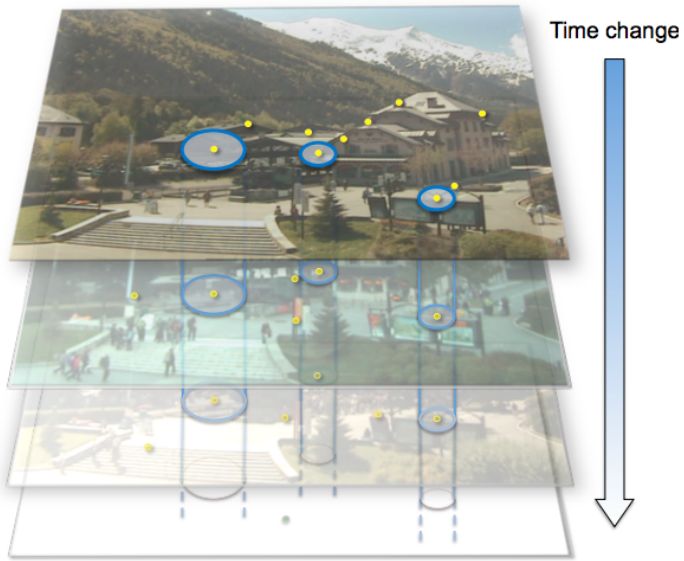
- Detect e.g. SIFT Interest Points in images across time
- Keep only those that are repeatable across time.
- These are our (super reliable) positive training examples. What about negative examples?




(a) Stack of training images

# Training the Detector

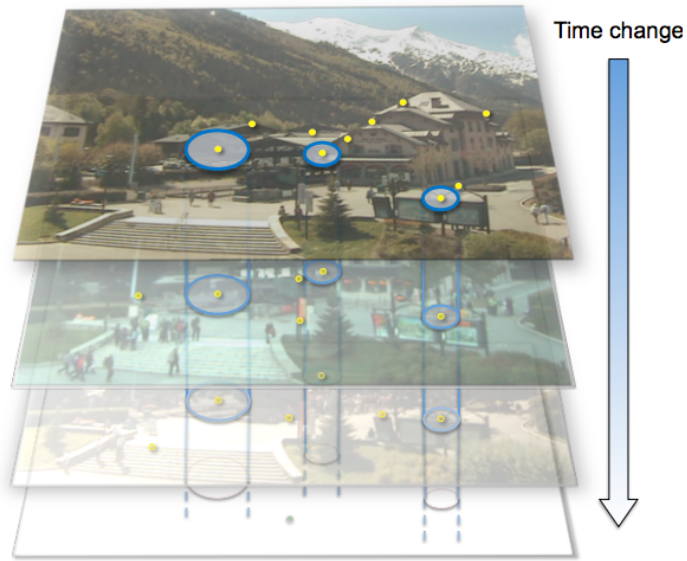
- Detect e.g. SIFT Interest Points in images across time
- Keep only those that are repeatable across time.
- These are our (super reliable) positive training examples. What about negative examples? All other points with some distance wrt positive points



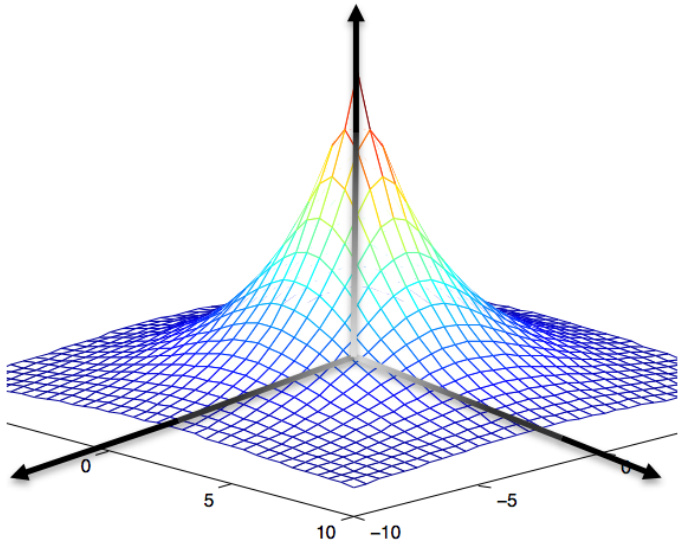

(a) Stack of training images

# 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 or a regressor




(a) Stack of training images

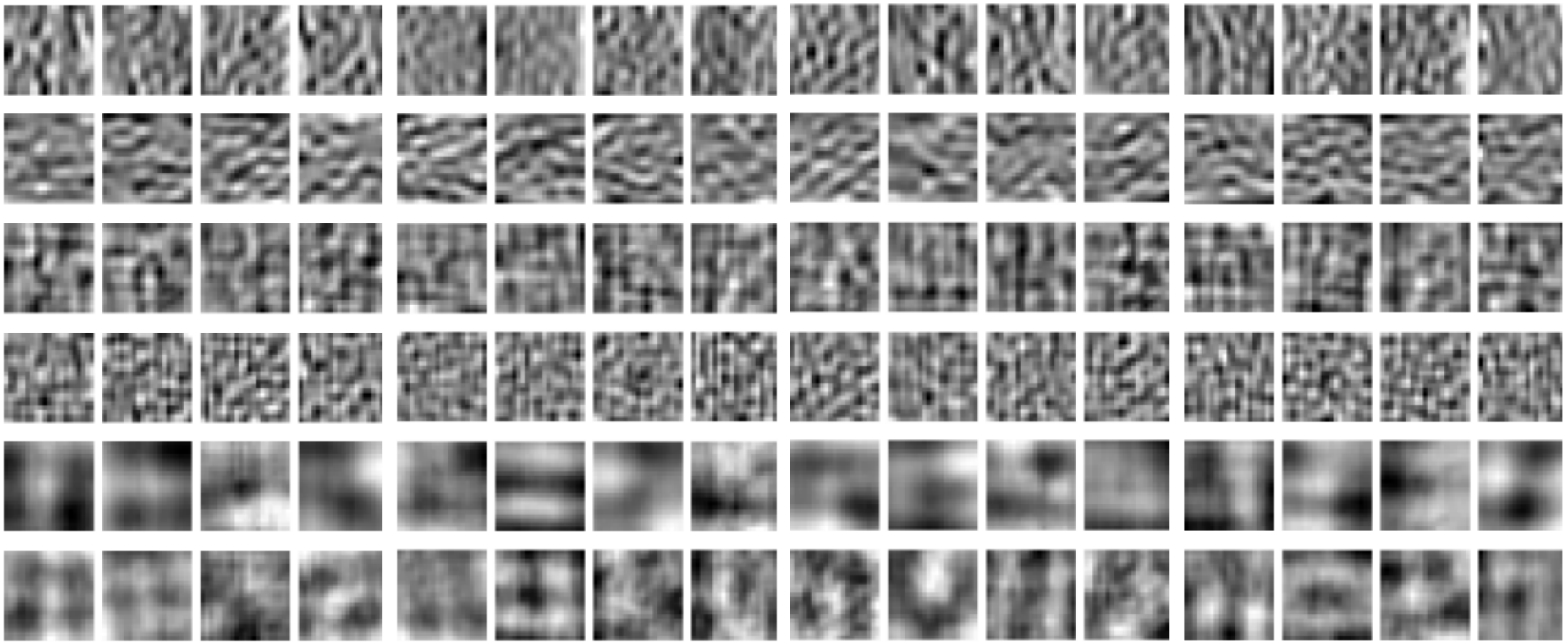



(b) Desired response on positive samples

# Trained Filters

- Remember from the lecture where we trained a classifier to detect edges:

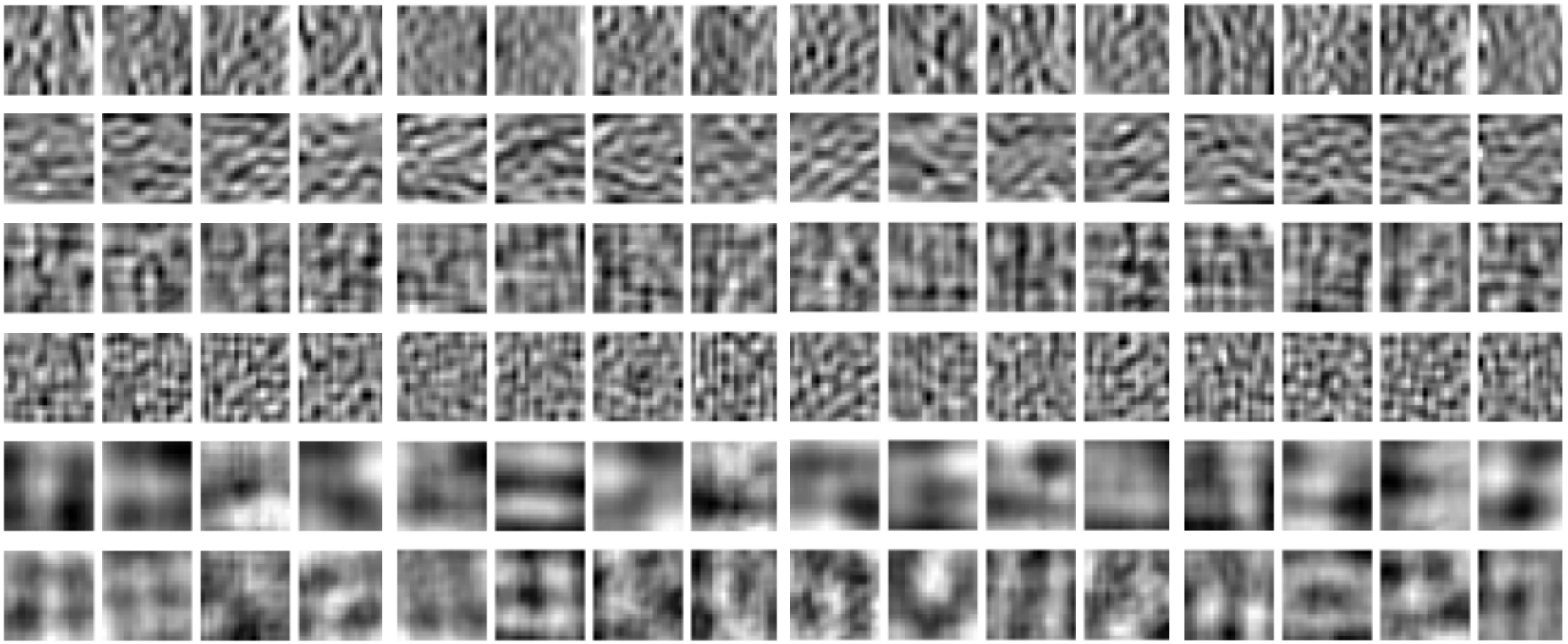
  If we train a linear classifier on a patch, it can be seen as a filter



# 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



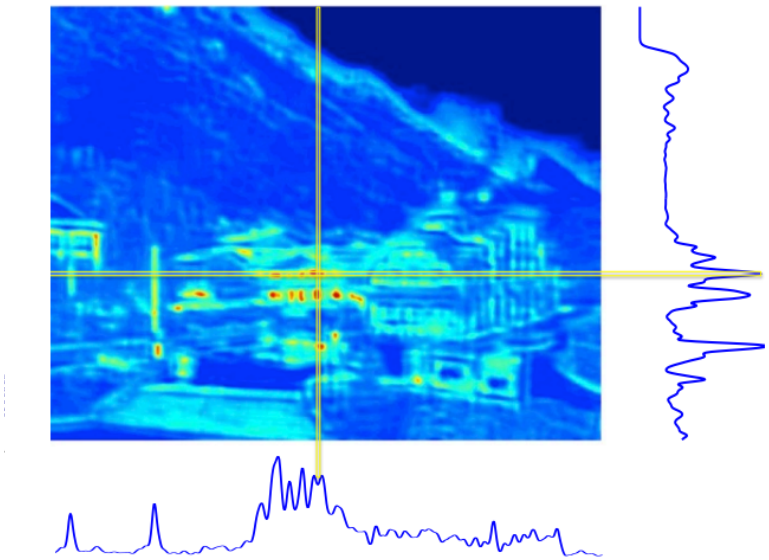
Tiny lesson learned: Sometime our intermediate results (filters in this case) don't look interpretable at all, but they still do the job

# 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

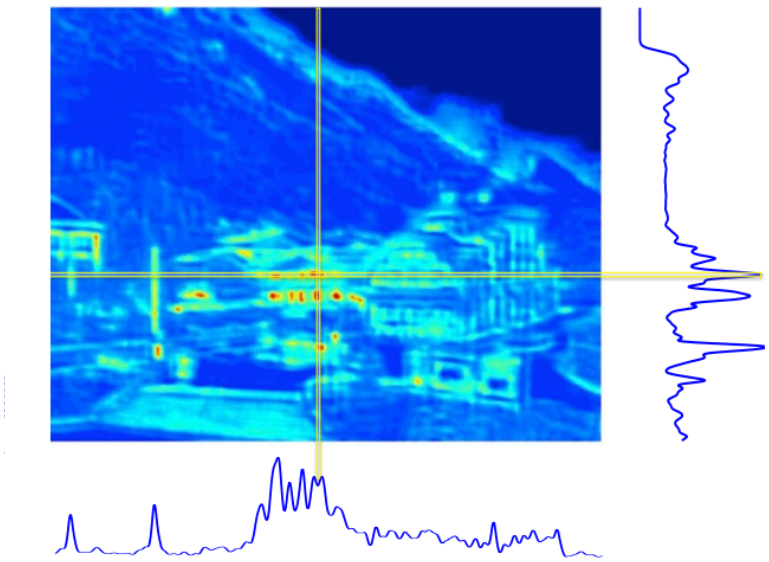
- Apply our filter on each image patch (convolution, if it's a linear classifier)



(c) Regressor response for a new image

# Using the Learned Interest Point Detector

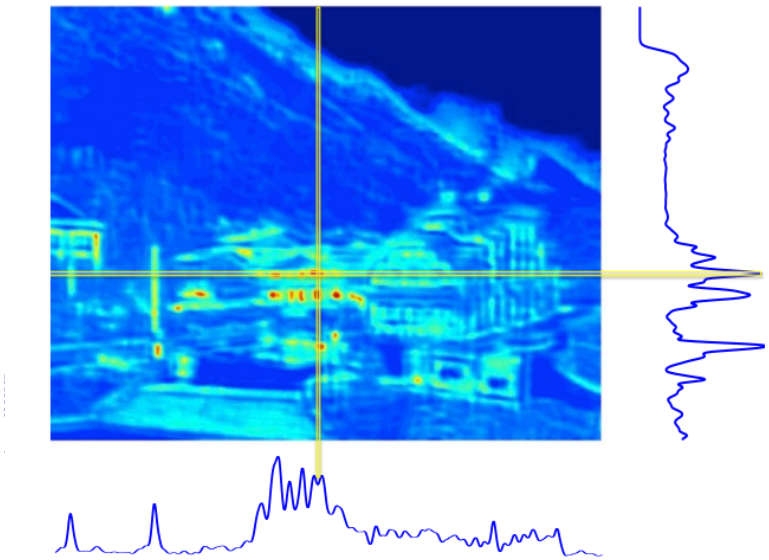
- Apply our filter on each image patch (convolution, if it's a linear classifier)
- This has response everywhere. How can we find the actual interest **points**?



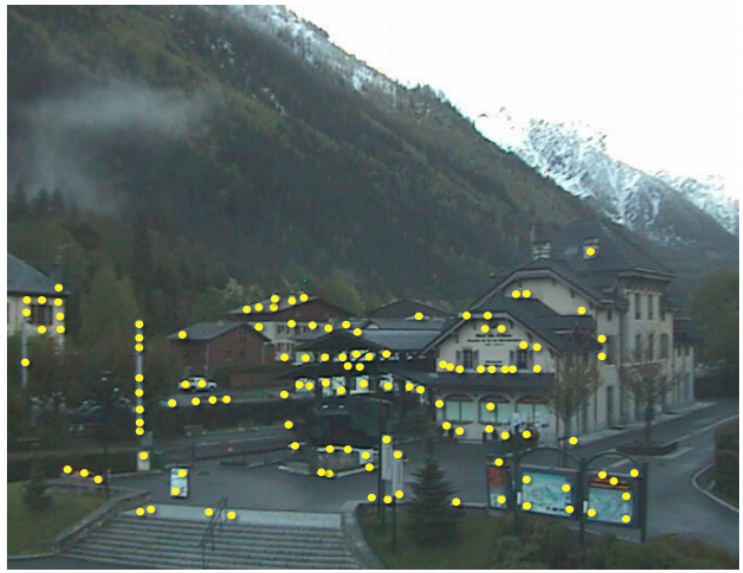
(c) Regressor response for a new image

# 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

# Qualitative Results (nice looking pictures)

- 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 HighSpeed Corner Detection. ECCV 2006
Paper: http://www.edwardrosten.com/work/rosten_2006_machine.pdf]

# Quantitative Results (performance numbers)

- Every method is much more convincing if it shows quantitative performance.
- If there are baselines for the problem, rule is: the more baselines the better.
- The more datasets, the better

**Different datasets**

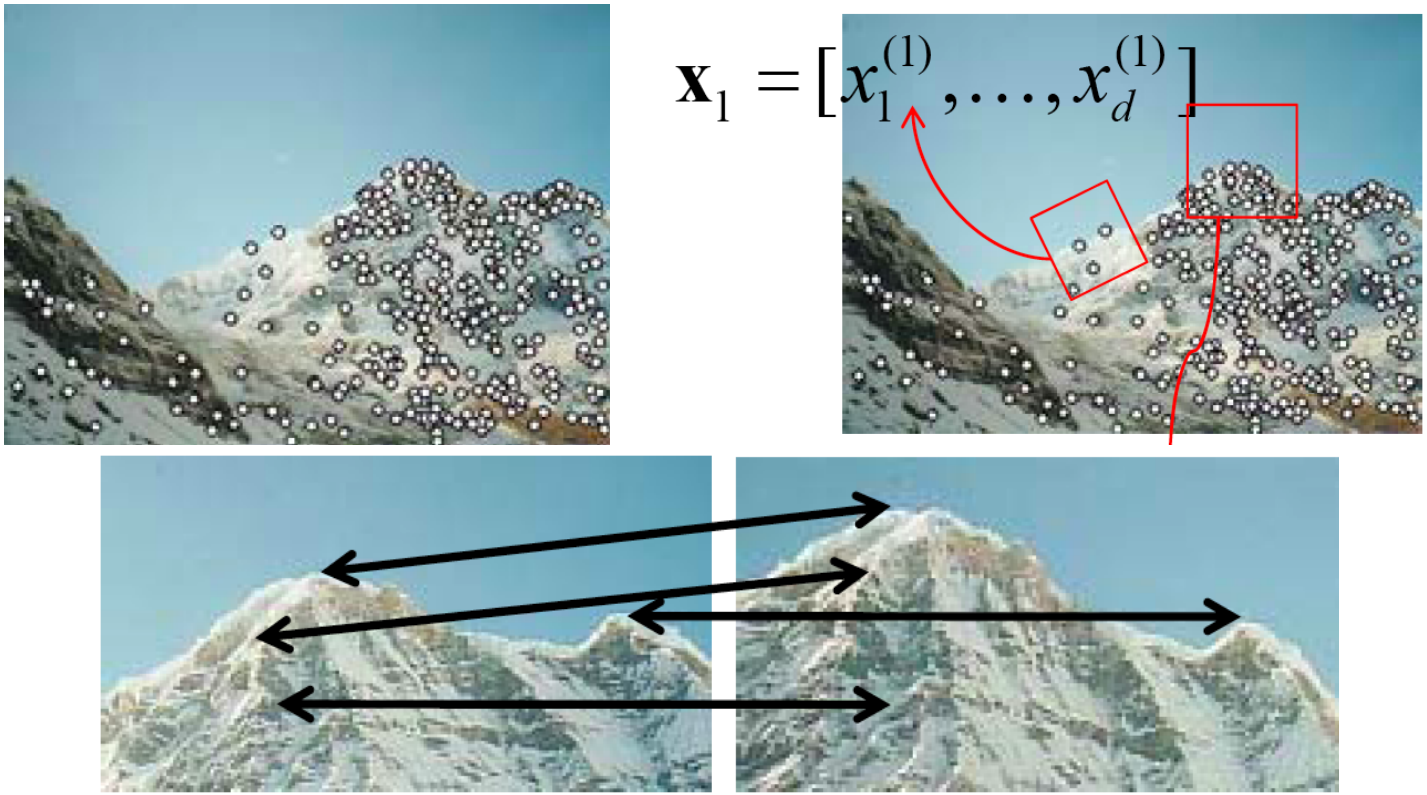
**Proposed approach**: Different instantiations of the model, e.g. different parameters, classifiers, etc

**Baselines**: previous approaches to interest point detection

| #keypoints | *Webcam* (2%) | *Oxford* Stand. | *Oxford* (2%) | *EF* Stand. | *EF* (2%) |
|---|---|---|---|---|---|
| TILDE-GB | 33.3 | 54.5 | 32.8 | 43.1 | 16.2 |
| TILDE-CNN | 36.8 | 51.8 | 49.3 | 43.2 | 27.6 |
| TILDE-P24 | 40.7 | **58.7** | **59.1** | **46.3** | **33.0** |
| TILDE-P | **48.3** | 58.1 | 55.9 | 45.1 | 31.6 |
| FAST-9 | 26.4 | 53.8 | 47.9 | 39.0 | 28.0 |
| SFOP | 22.9 | 51.3 | 39.3 | 42.2 | 21.2 |
| SIFER | 25.7 | 45.1 | 40.1 | 27.4 | 17.6 |
| SIFT | 20.7 | 46.5 | 43.6 | 32.2 | 23.0 |
| SURF | 29.9 | 56.9 | 57.6 | 43.6 | 28.7 |
| TaSK | 14.5 | 25.7 | 15.7 | 22.8 | 10.0 |
| WADE | 27.5 | 44.3 | 51.0 | 25.6 | 28.6 |
| MSER | 22.3 | 51.5 | 35.9 | 38.9 | 23.9 |
| LCF | 30.9 | 55.0 | 40.1 | 41.6 | 23.1 |
| EdgeFoci | 30.0 | 54.9 | 47.5 | 46.2 | 31.0 |

# Local Descriptors – Next Time

- **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]