Computer Vision: Image Features

Raquel Urtasun

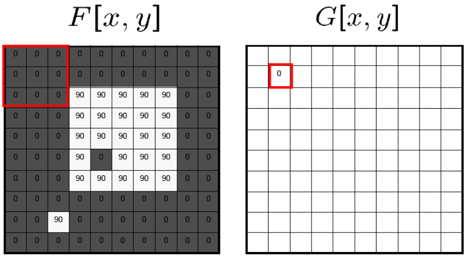
TTI Chicago

Jan 17, 2013

What did we see in class last week?

Last classes

- Image formation
- Filtering: convolution vs correlation



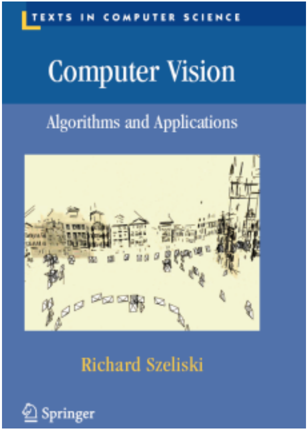
- Separable filters
- Computing edges
- Steerable filters
- Other transformations

Today's lecture ...

- **Local features**:
  - Interest point detection
  - Descriptors
  - Matching

Material

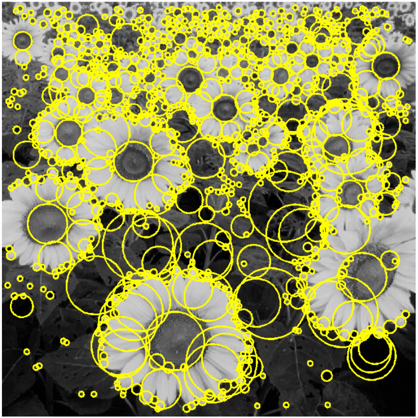
- Chapter 3 and 4 of Rich Szeliski book



- Available online here

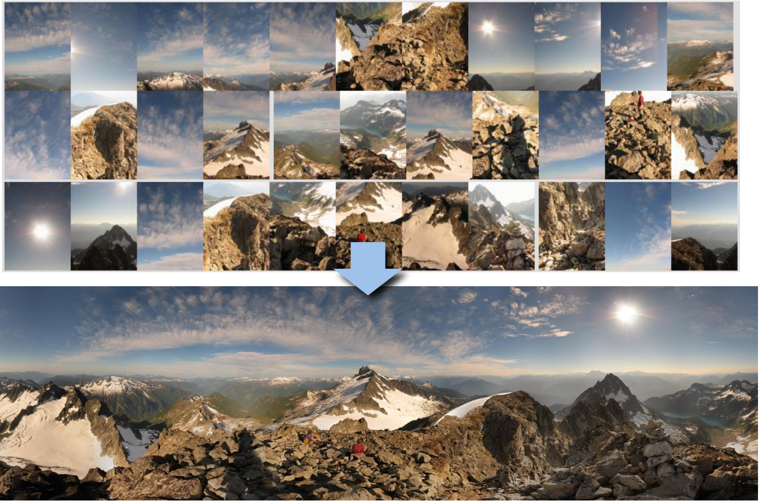
Local features

Feature extraction: Corners and blobs



[Source: N. Snavely]

Motivation: Automatic panoramas



Credit: Matt Brown

Motivation: Automatic panoramas

HD View
http://research.microsoft.com/en-us/um/redmond/groups/ivm/HDView/HDGigapixel.htm

Also see GigaPan:
http://gigapan.org/

Why extract features?

How to combine these two images to form a panorama?

Figure: Two images

Why extract features?

How to combine these two images to form a panorama?

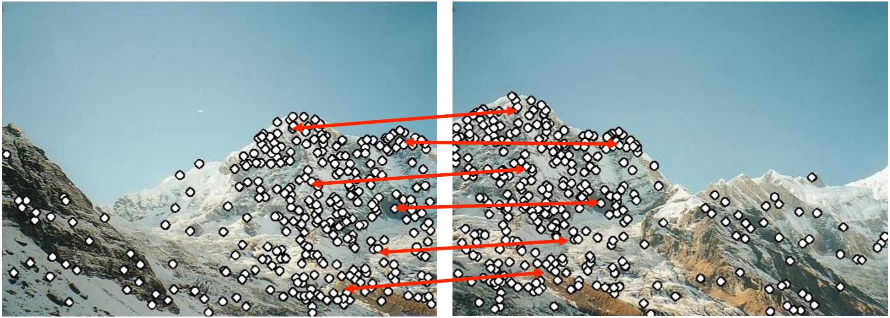


Figure: Feature extraction and matching

Why extract features?

How to combine these two images to form a panorama?

Figure: Image aligment

Image matching

by Diva Sian

by swashford

[Source: N. Snavely]

Harder Case

- Why is this harder?

by Diva Sian

by scgbt

[Source: N. Snavely]

Harder Still

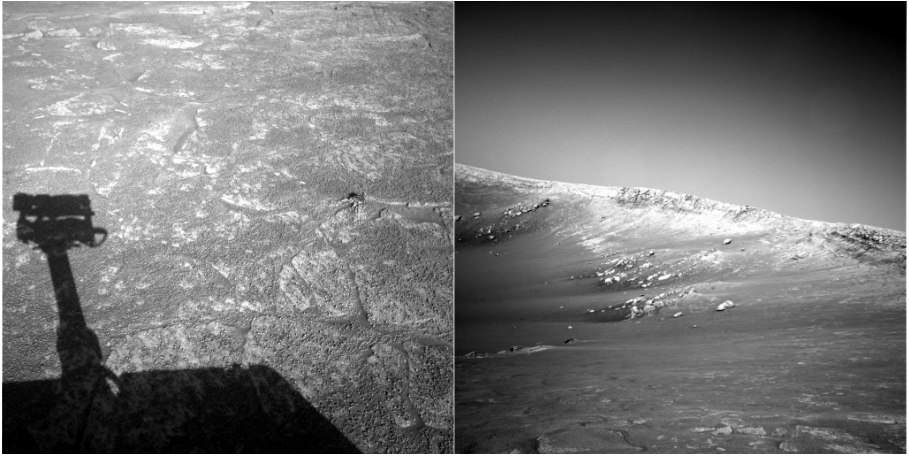


Figure: NASA Mars Rover images

[Source: N. Snavely]

Look for tiny squares ...

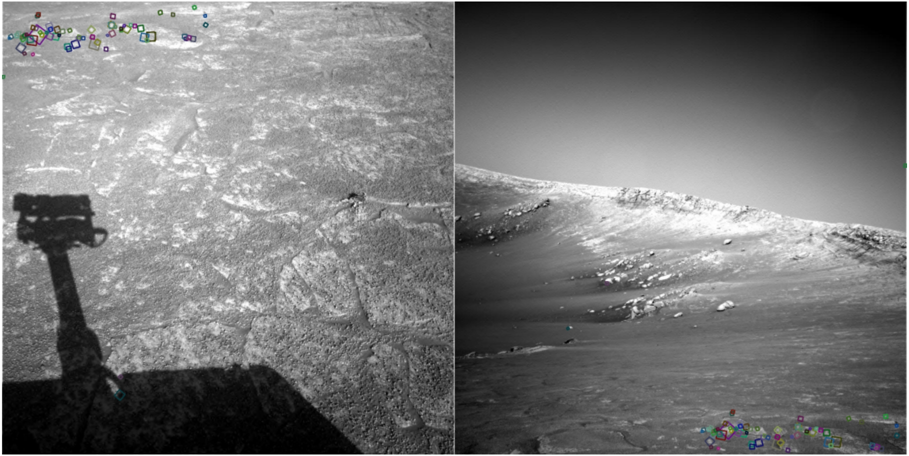
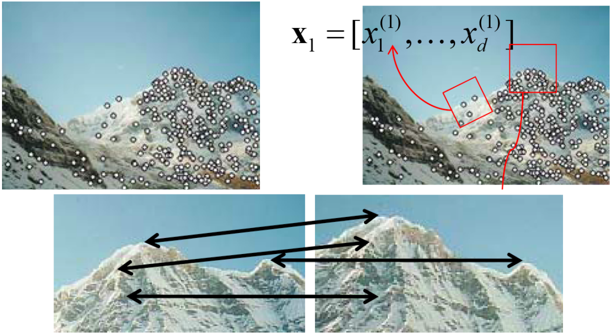


Figure: NASA Mars Rover images with SIFT feature matches

[Source: N. Snavely]

Local features

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



[Source: K. Grauman]

Two approaches to Find Features

- **Tracking**: searches in a small neighborhood around each detected feature.
  - When images are taken from nearby viewpoints
  - or in successive times (e.g., video sequence)
- **Matching**: Determine correspondence between descriptors in two views.
  - When a large motion can happen, e.g., panoramas, wide baseline stereo, object recognition.

Goal: interest operator repeatability

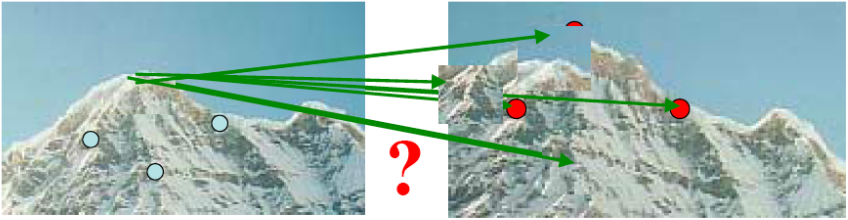
- We want to detect (at least some of) the **same points** in both images.
- We have to be able to run the detection procedure **independently per image**.

Figure: No chance to find the true matches

[Source: K. Grauman]

Goal: descriptor distinctiveness

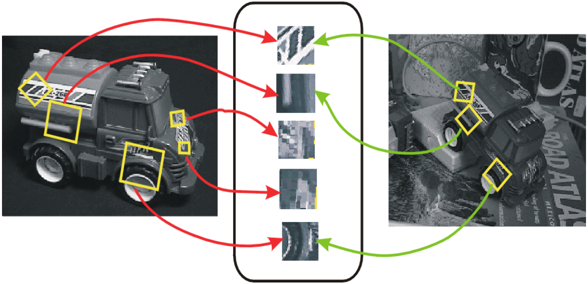
- We want to be able to **reliably match**, i.e., determine which point goes with which.
- Must provide some **invariance** to **geometric** and **photometric** differences between the two views.



[Source: K. Grauman]

Invariant local features

- **geometric invariance**: translation, rotation, scale
- **photometric invariance**: brightness, exposure,



Feature Descriptors

[Source: N. Snavely]

Advantages of local features

- **Locality:** features are local, so robust to occlusion and clutter
- **Quantity:** hundreds or thousands in a single image
- **Distinctiveness:** can differentiate a large database of objects
- **Efficiency:** real-time performance achievable

[Source: N. Snavely]

More motivation

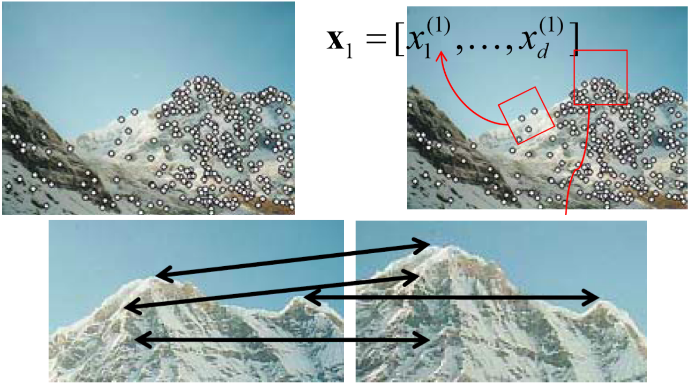
Feature points are used for:

- Image alignment (e.g., mosaics)
- 3D reconstruction
- Motion tracking
- Object recognition
- Indexing and database retrieval
- Robot navigation
- ...

[Source: N. Snavely]

Local features

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



[Source: K. Grauman]

What points to choose?

[Source: K. Grauman]

Want uniqueness

- Look for image regions that are **unusual**: lead to unambiguous matches in other images
- How to define "unusual"?

What points to choose?

- Textureless patches are nearly impossible to localize.
- Patches with **large contrast changes** (gradients) are easier to localize.

What points to choose?

- Textureless patches are nearly impossible to localize.
- Patches with **large contrast changes** (gradients) are easier to localize.
- But straight line segments at a single orientation suffer from the **aperture problem**, i.e., it is only possible to align the patches along the direction normal to the edge direction.

What points to choose?

- Textureless patches are nearly impossible to localize.
- Patches with **large contrast changes** (gradients) are easier to localize.
- But straight line segments at a single orientation suffer from the **aperture problem**, i.e., it is only possible to align the patches along the direction normal to the edge direction.
- Gradients in at least two (significantly) different orientations are the easiest, e.g., corners.

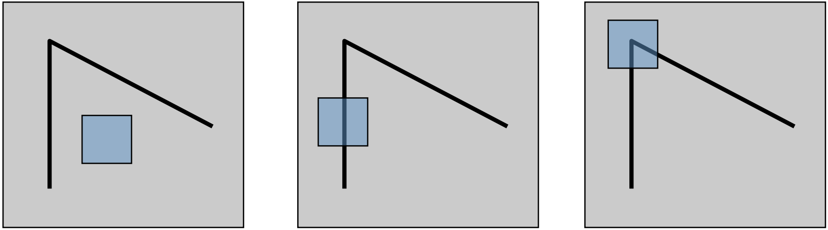
What points to choose?

- Textureless patches are nearly impossible to localize.
- Patches with **large contrast changes** (gradients) are easier to localize.
- But straight line segments at a single orientation suffer from the **aperture problem**, i.e., it is only possible to align the patches along the direction normal to the edge direction.
- Gradients in at least two (significantly) different orientations are the easiest, e.g., corners.

Local measure of uniqueness

Suppose we only consider a small window of pixels

- What defines whether a feature is a good or bad candidate?

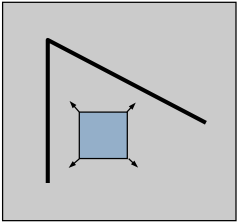


[Source: S. Seitz, D. Frolova, D. Simakov]

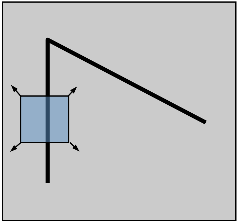
Local measure of feature uniqueness

Suppose we only consider a small window of pixels

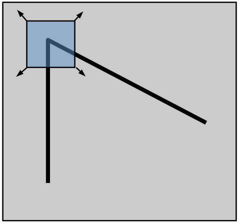
- How does the window change when you shift it?
- Shifting the window in any direction causes a big change



"flat" region:
no change in all directions



"edge":
no change along the edge direction



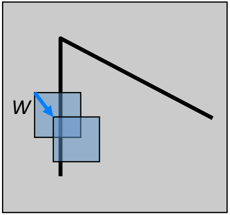
"corner":
significant change in all directions

Credit: S. Seitz, D. Frolova, D. Simakov

A Simple Matching Criteria

Consider shifting the window W by (u, v)

- how do the pixels in W change?
- compare each pixel before and after by **summing up the squared differences** (SSD)



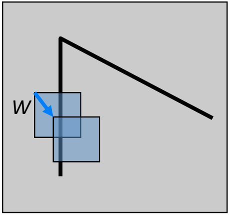
- this defines an SSD error

$$E(u, v) = \sum_{(x,y) \in W} [I(x + u, y + v) - I(x, y)]^2$$

A Simple Matching Criteria

Consider shifting the window W by (u, v)

- how do the pixels in W change?
- compare each pixel before and after by **summing up the squared differences** (SSD)



- this defines an SSD error

$$E(u, v) = \sum_{(x,y)\in W} [I(x + u, y + v) - I(x, y)]^2$$

A Simple Weighted Matching Criteria

- Compare two image patches using **(weighted) summed square difference**

$$E_{WSSD}(\mathbf{u}) = \sum_i w(\mathbf{p}_i)[I_1(\mathbf{p}_i + \mathbf{u}) - I_0(\mathbf{p}_i)]^2$$

  with I_0 and I_1 two images being compared, $\mathbf{u}(u_x, u_y)$ a displacement vector, $w(\mathbf{p})$ a spatially varying weighting function, and the summation i is over all the pixels in the patch.

- We do not know which other image locations the feature will end up being matched against.

A Simple Weighted Matching Criteria

- Compare two image patches using **(weighted) summed square difference**

$$E_{WSSD}(\mathbf{u}) = \sum_i w(\mathbf{p}_i)[I_1(\mathbf{p}_i + \mathbf{u}) - I_0(\mathbf{p}_i)]^2$$

  with I_0 and I_1 two images being compared, $\mathbf{u}(u_x, u_y)$ a displacement vector, $w(\mathbf{p})$ a spatially varying weighting function, and the summation i is over all the pixels in the patch.

- We do not know which other image locations the feature will end up being matched against.
- We can only compute how stable this metric is with respect to small variations in position u by comparing an image patch against itself.

A Simple Weighted Matching Criteria

- Compare two image patches using **(weighted) summed square difference**

$$E_{WSSD}(\mathbf{u}) = \sum_i w(\mathbf{p}_i)[I_1(\mathbf{p}_i + \mathbf{u}) - I_0(\mathbf{p}_i)]^2$$

  with I_0 and I_1 two images being compared, $\mathbf{u}(u_x, u_y)$ a displacement vector, $w(\mathbf{p})$ a spatially varying weighting function, and the summation i is over all the pixels in the patch.

- We do not know which other image locations the feature will end up being matched against.
- We can only compute how stable this metric is with respect to small variations in position u by comparing an image patch against itself.
- This is the **auto-correlation function**

$$E_{AC}(\Delta\mathbf{u}) = \sum_i w(\mathbf{p}_i)[I_0(\mathbf{p}_i + \Delta u) - I_0(\mathbf{p}_i)]^2$$

A Simple Weighted Matching Criteria

- Compare two image patches using **(weighted) summed square difference**

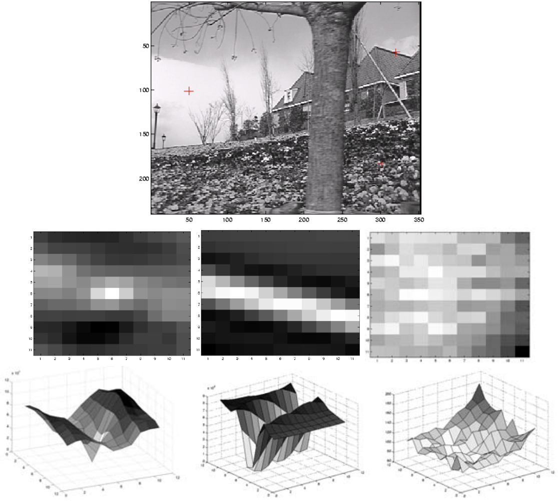
$$E_{WSSD}(\mathbf{u}) = \sum_i w(\mathbf{p}_i)[I_1(\mathbf{p}_i + \mathbf{u}) - I_0(\mathbf{p}_i)]^2$$

  with I_0 and I_1 two images being compared, $\mathbf{u}(u_x, u_y)$ a displacement vector, $w(\mathbf{p})$ a spatially varying weighting function, and the summation i is over all the pixels in the patch.

- We do not know which other image locations the feature will end up being matched against.
- We can only compute how stable this metric is with respect to small variations in position u by comparing an image patch against itself.
- This is the **auto-correlation function**

$$E_{AC}(\Delta\mathbf{u}) = \sum_i w(\mathbf{p}_i)[I_0(\mathbf{p}_i + \Delta u) - I_0(\mathbf{p}_i)]^2$$

Which one is better?



[Source: R. Szeliski]

How to select and interest point?

- Small motion assumption
- Using a Taylor Series expansion $I_0(\mathbf{p}_i + \Delta\mathbf{u}) \approx I_0(\mathbf{p}_i) + \nabla I_0(\mathbf{p}_i)\Delta\mathbf{u}$ with

$$\nabla I_0(\mathbf{p}_i) = \left(\frac{\partial I_0}{\partial x}, \frac{\partial I_0}{\partial y}\right)(\mathbf{p}_i)$$

  the image gradient. We can approximate the autocorrelation as

How to select and interest point?

- Small motion assumption
- Using a Taylor Series expansion $I_0(\mathbf{p}_i + \Delta\mathbf{u}) \approx I_0(\mathbf{p}_i) + \nabla I_0(\mathbf{p}_i)\Delta\mathbf{u}$ with

$$\nabla I_0(\mathbf{p}_i) = \left(\frac{\partial I_0}{\partial x}, \frac{\partial I_0}{\partial y}\right)(\mathbf{p}_i)$$

the image gradient. We can approximate the autocorrelation as

$$E_{AC}(\Delta\mathbf{u}) = \sum_i w(\mathbf{p}_i)[I_0(\mathbf{p}_i + \Delta\mathbf{u}) - I_0(\mathbf{p}_i)]^2$$

How to select and interest point?

- Small motion assumption
- Using a Taylor Series expansion $I_0(\mathbf{p}_i + \Delta\mathbf{u}) \approx I_0(\mathbf{p}_i) + \nabla I_0(\mathbf{p}_i)\Delta\mathbf{u}$ with

$$\nabla I_0(\mathbf{p}_i) = \left(\frac{\partial I_0}{\partial x}, \frac{\partial I_0}{\partial y}\right)(\mathbf{p}_i)$$

the image gradient. We can approximate the autocorrelation as

$$\begin{aligned} E_{AC}(\Delta\mathbf{u}) &= \sum_i w(\mathbf{p}_i)[I_0(\mathbf{p}_i + \Delta\mathbf{u}) - I_0(\mathbf{p}_i)]^2 \\ &\approx \sum_i w(\mathbf{p}_i)[I_0(\mathbf{p}_i) + \nabla I_0(\mathbf{p}_i)\Delta\mathbf{u} - I_0(\mathbf{p}_i)]^2 \end{aligned}$$

How to select and interest point?

- Small motion assumption
- Using a Taylor Series expansion $I_0(\mathbf{p}_i + \Delta\mathbf{u}) \approx I_0(\mathbf{p}_i) + \nabla I_0(\mathbf{p}_i)\Delta\mathbf{u}$ with

$$\nabla I_0(\mathbf{p}_i) = \left(\frac{\partial I_0}{\partial x}, \frac{\partial I_0}{\partial y}\right)(\mathbf{p}_i)$$

the image gradient. We can approximate the autocorrelation as

$$\begin{aligned} E_{AC}(\Delta\mathbf{u}) &= \sum_i w(\mathbf{p}_i)[I_0(\mathbf{p}_i + \Delta\mathbf{u}) - I_0(\mathbf{p}_i)]^2 \\ &\approx \sum_i w(\mathbf{p}_i)[I_0(\mathbf{p}_i) + \nabla I_0(\mathbf{p}_i)\Delta\mathbf{u} - I_0(\mathbf{p}_i)]^2 \\ &= \sum_i w(\mathbf{p}_i)[\nabla I_0(\mathbf{p}_i)\Delta\mathbf{u}]^2 \end{aligned}$$

How to select and interest point?

- Small motion assumption
- Using a Taylor Series expansion $I_0(\mathbf{p}_i + \Delta\mathbf{u}) \approx I_0(\mathbf{p}_i) + \nabla I_0(\mathbf{p}_i)\Delta\mathbf{u}$ with

$$\nabla I_0(\mathbf{p}_i) = \left(\frac{\partial I_0}{\partial x}, \frac{\partial I_0}{\partial y}\right)(\mathbf{p}_i)$$

the image gradient. We can approximate the autocorrelation as

$$\begin{aligned} E_{AC}(\Delta\mathbf{u}) &= \sum_i w(\mathbf{p}_i)[I_0(\mathbf{p}_i + \Delta\mathbf{u}) - I_0(\mathbf{p}_i)]^2 \\ &\approx \sum_i w(\mathbf{p}_i)[I_0(\mathbf{p}_i) + \nabla I_0(\mathbf{p}_i)\Delta\mathbf{u} - I_0(\mathbf{p}_i)]^2 \\ &= \sum_i w(\mathbf{p}_i)[\nabla I_0(\mathbf{p}_i)\Delta\mathbf{u}]^2 \\ &= \Delta\mathbf{u}^T\mathbf{A}\Delta\mathbf{u} \end{aligned}$$

How to select and interest point?

- Small motion assumption
- Using a Taylor Series expansion $I_0(\mathbf{p}_i + \Delta\mathbf{u}) \approx I_0(\mathbf{p}_i) + \nabla I_0(\mathbf{p}_i)\Delta\mathbf{u}$ with

$$\nabla I_0(\mathbf{p}_i) = \left(\frac{\partial I_0}{\partial x}, \frac{\partial I_0}{\partial y}\right)(\mathbf{p}_i)$$

  the image gradient. We can approximate the autocorrelation as

$$\begin{aligned} E_{AC}(\Delta\mathbf{u}) &= \sum_i w(\mathbf{p}_i)[I_0(\mathbf{p}_i + \Delta\mathbf{u}) - I_0(\mathbf{p}_i)]^2 \\ &\approx \sum_i w(\mathbf{p}_i)[I_0(\mathbf{p}_i) + \nabla I_0(\mathbf{p}_i)\Delta\mathbf{u} - I_0(\mathbf{p}_i)]^2 \\ &= \sum_i w(\mathbf{p}_i)[\nabla I_0(\mathbf{p}_i)\Delta\mathbf{u}]^2 \\ &= \Delta\mathbf{u}^T\mathbf{A}\Delta\mathbf{u} \end{aligned}$$

- Gradient can be computed with the filtering techniques we saw, e.g., derivatives of Gaussians.

How to select and interest point?

- Small motion assumption
- Using a Taylor Series expansion $I_0(\mathbf{p}_i + \Delta\mathbf{u}) \approx I_0(\mathbf{p}_i) + \nabla I_0(\mathbf{p}_i)\Delta\mathbf{u}$ with

$$\nabla I_0(\mathbf{p}_i) = \left(\frac{\partial I_0}{\partial x}, \frac{\partial I_0}{\partial y}\right)(\mathbf{p}_i)$$

  the image gradient. We can approximate the autocorrelation as

$$\begin{aligned} E_{AC}(\Delta\mathbf{u}) &= \sum_i w(\mathbf{p}_i)[I_0(\mathbf{p}_i + \Delta\mathbf{u}) - I_0(\mathbf{p}_i)]^2 \\ &\approx \sum_i w(\mathbf{p}_i)[I_0(\mathbf{p}_i) + \nabla I_0(\mathbf{p}_i)\Delta\mathbf{u} - I_0(\mathbf{p}_i)]^2 \\ &= \sum_i w(\mathbf{p}_i)[\nabla I_0(\mathbf{p}_i)\Delta\mathbf{u}]^2 \\ &= \Delta\mathbf{u}^T\mathbf{A}\Delta\mathbf{u} \end{aligned}$$

- Gradient can be computed with the filtering techniques we saw, e.g., derivatives of Gaussians.

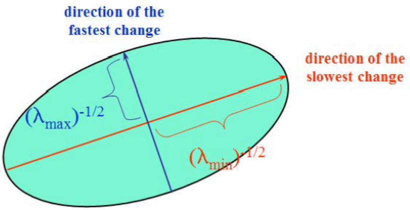
More on selection

- The autocorrelation is $E_{AC}(\Delta\mathbf{u}) = \Delta\mathbf{u}^T\mathbf{A}\Delta\mathbf{u}$, with

$$\mathbf{A} = \sum_u \sum_v w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}$$

  where we have replaced the weighted summations with discrete convolutions with the weighting kernel w.

- $\mathbf{A}$ can be interpreted as a tensor where the outer products of the gradients are convolved with a weighting function.
- Eigenvalues a notion of uncertainty



[Source: R. Szeliski]

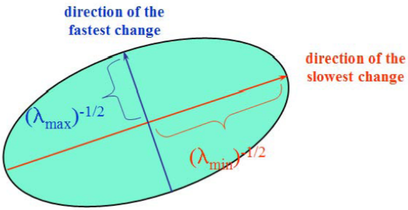
More on selection

- The autocorrelation is $E_{AC}(\Delta\mathbf{u}) = \Delta\mathbf{u}^T \mathbf{A} \Delta\mathbf{u}$, with

$$\mathbf{A} = \sum_u \sum_v w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}$$

  where we have replaced the weighted summations with discrete convolutions with the weighting kernel w.

- **A** can be interpreted as a tensor where the outer products of the gradients are convolved with a weighting function.
- Eigenvalues a notion of uncertainty



[Source: R. Szeliski]

Quick review on eigenvalue/eigenvector

- The **eigenvectors** of a matrix $\mathbf{A}$ are the vectors $\mathbf{x}$ that satisfy

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$$

  with λ a scalar call the **eigenvalue**

- The eigenvalues can be found by solving

$$det(\mathbf{A} - \lambda I) = 0$$

Quick review on eigenvalue/eigenvector

- The **eigenvectors** of a matrix **A** are the vectors **x** that satisfy

$$\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$$

  with λ a scalar call the **eigenvalue**

- The eigenvalues can be found by solving

$$det(\mathbf{A} - \lambda I) = 0$$

- In our case **A** is a 2×2 matrix, so the solution is

$$\lambda = \frac{1}{2}[(a_{11} + a_{22} \pm \sqrt{4a_{12}a_{21} + (a_{11} - a_{22})^2}]$$

Quick review on eigenvalue/eigenvector

- The **eigenvectors** of a matrix **A** are the vectors **x** that satisfy

$$\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$$

  with λ a scalar call the **eigenvalue**

- The eigenvalues can be found by solving

$$det(\mathbf{A} - \lambda I) = 0$$

- In our case **A** is a 2×2 matrix, so the solution is

$$\lambda = \frac{1}{2}[(a_{11} + a_{22} \pm \sqrt{4a_{12}a_{21} + (a_{11} - a_{22})^2}]$$

- Once you know λ you can find **x** by solving

$$(\mathbf{A} - \lambda I)\mathbf{x} = 0$$

[Source: N. Snavely]

Quick review on eigenvalue/eigenvector

- The **eigenvectors** of a matrix $\mathbf{A}$ are the vectors $\mathbf{x}$ that satisfy

$$\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$$

  with λ a scalar call the **eigenvalue**

- The eigenvalues can be found by solving

$$det(\mathbf{A} - \lambda I) = 0$$

- In our case $\mathbf{A}$ is a 2×2 matrix, so the solution is

$$\lambda = \frac{1}{2}[(a_{11} + a_{22} \pm \sqrt{4a_{12}a_{21} + (a_{11} - a_{22})^2}]$$

- Once you know λ you can find $\mathbf{x}$ by solving

$$(\mathbf{A} - \lambda I)\mathbf{x} = 0$$

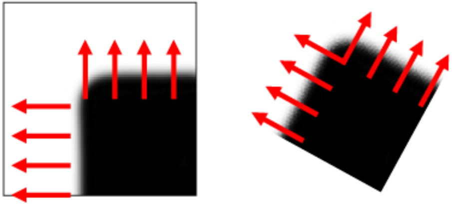
[Source: N. Snavely]

Eigenvalues a notion of uncertainty

- **A** is symmetric

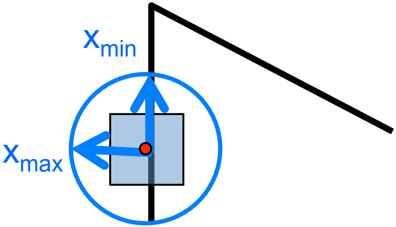
$$\mathbf{A} = \mathbf{U} \begin{bmatrix} \lambda_0 & 0 \\ 0 & \lambda_1 \end{bmatrix} \mathbf{U}^T \quad \text{with} \quad \mathbf{A}\mathbf{u}_i = \lambda_i \mathbf{u}_i$$

- The eigenvalues of **A** reveal the amount of intensity change in the two principal orthogonal gradient directions in the window.
- How is this matrix for



[Source: R. Szeliski]

Eigenvalues a notion of uncertainty

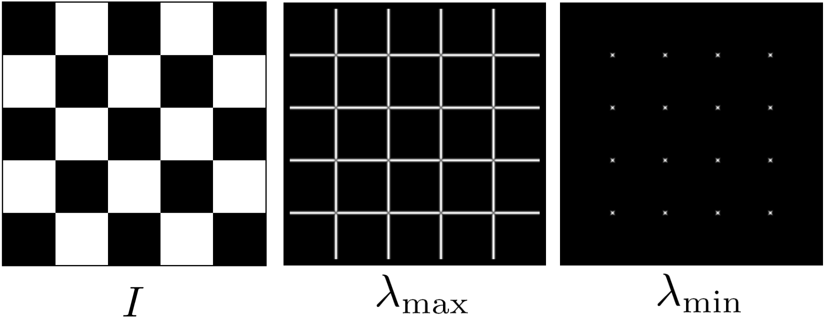


Eigenvalues and eigenvectors of $\mathbf{A}$

- $\mathbf{x}_{max}$ = direction of largest increase in E
- λ_{max} = amount of increase in direction $\mathbf{x}_{max}$
- $\mathbf{x}_{max}$ = direction of smallest increase in E
- λ_{min} = amount of increase in direction $\mathbf{x}_{min}$

[Source: N. Snavely]

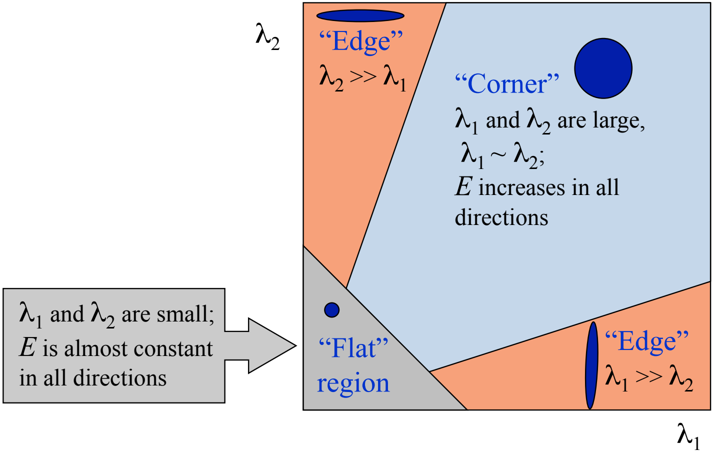
Example



[Source: N. Snavely]

Interpreting the eigenvalues

Classification of image points using eigenvalues of **A**:



[Source: N. Snavely]

Local Feature Selection Criteria

- Shi and Tomasi, 94 proposed the smallest eigenvalue of **A**, i.e., $\lambda_0^{-1/2}$, which is a rotationally invariant measure
- Harris and Stephens, 88 is rotationally invariant and downweights edge-like features where $\lambda_1 \gg \lambda_0$

$$\det(\mathbf{A}) - \alpha \mathrm{trace}(\mathbf{A})^2 = \lambda_0 \lambda_1 - \alpha(\lambda_0 + \lambda_1)^2$$

Local Feature Selection Criteria

- Shi and Tomasi, 94 proposed the smallest eigenvalue of **A**, i.e., $\lambda_0^{-1/2}$, which is a rotationally invariant measure
- Harris and Stephens, 88 is rotationally invariant and downweights edge-like features where $\lambda_1 \gg \lambda_0$

$$\det(\mathbf{A}) - \alpha \mathrm{trace}(\mathbf{A})^2 = \lambda_0 \lambda_1 - \alpha(\lambda_0 + \lambda_1)^2$$

- Triggs, 04 suggested

$$\lambda_0 - \alpha \lambda_1$$

Local Feature Selection Criteria

- Shi and Tomasi, 94 proposed the smallest eigenvalue of **A**, i.e., $\lambda_0^{-1/2}$, which is a rotationally invariant measure
- Harris and Stephens, 88 is rotationally invariant and downweights edge-like features where $\lambda_1 \gg \lambda_0$

$$\det(\mathbf{A}) - \alpha \text{trace}(\mathbf{A})^2 = \lambda_0\lambda_1 - \alpha(\lambda_0 + \lambda_1)^2$$

- Triggs, 04 suggested

$$\lambda_0 - \alpha\lambda_1$$

- Brown et al, 05 use the harmonic mean

$$\frac{\det(\mathbf{A})}{\text{trace}(\mathbf{A})} = \frac{\lambda_0\lambda_1}{\lambda_0 + \lambda_1}$$

which is smoother when $\lambda_0 \approx \lambda_1$.

Local Feature Selection Criteria

- Shi and Tomasi, 94 proposed the smallest eigenvalue of **A**, i.e., $\lambda_0^{-1/2}$, which is a rotationally invariant measure
- Harris and Stephens, 88 is rotationally invariant and downweights edge-like features where $\lambda_1 \gg \lambda_0$

$$\det(\mathbf{A}) - \alpha \mathrm{trace}(\mathbf{A})^2 = \lambda_0\lambda_1 - \alpha(\lambda_0 + \lambda_1)^2$$

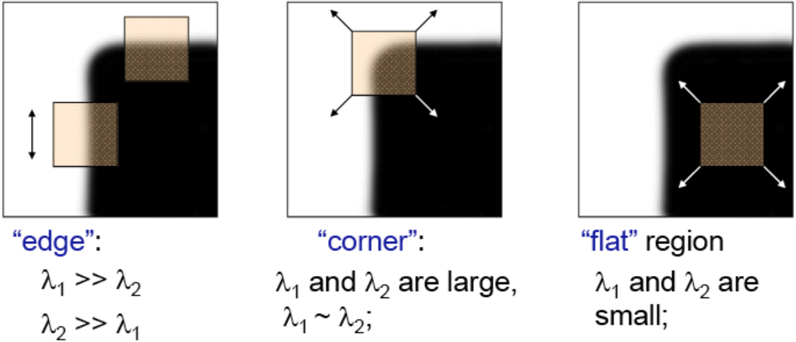
- Triggs, 04 suggested

$$\lambda_0 - \alpha\lambda_1$$

- Brown et al, 05 use the harmonic mean

$$\frac{\det(\mathbf{A})}{\mathrm{trace}(\mathbf{A})} = \frac{\lambda_0\lambda_1}{\lambda_0 + \lambda_1}$$

which is smoother when $\lambda_0 \approx \lambda_1$.



[Source: K. Grauman]

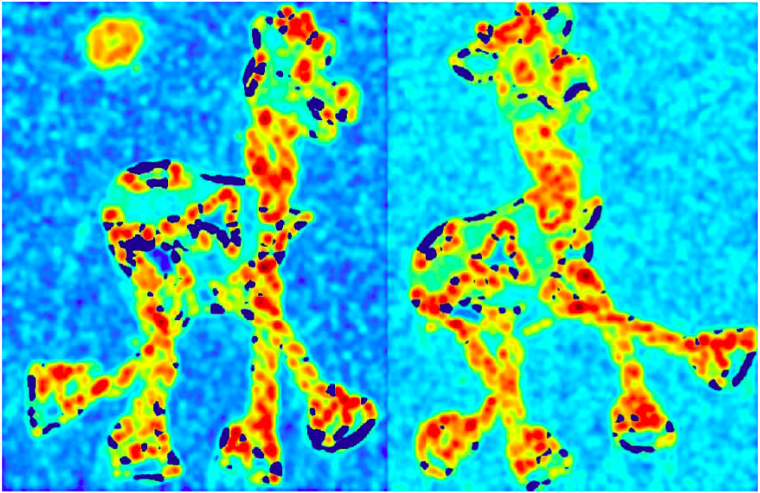
Harris Corner detector

1. Compute the gradients at each point in the image
2. Compute **A** for each image window to get its **cornerness** scores.
3. Compute the eigenvalues
4. Find points whose surrounding window gave large corner response ($f >$ threshold).
5. Take the points of local maxima, i.e., perform non-maximum suppression.

Example

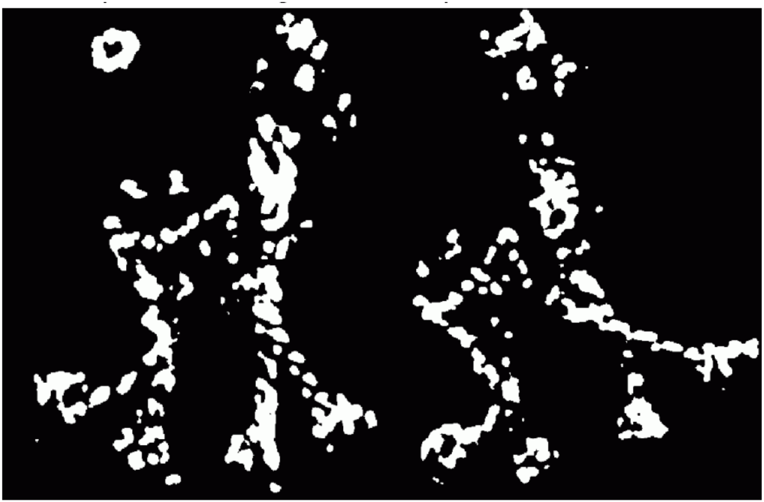
[Source: K. Grauman]

1) Compute Cornerness



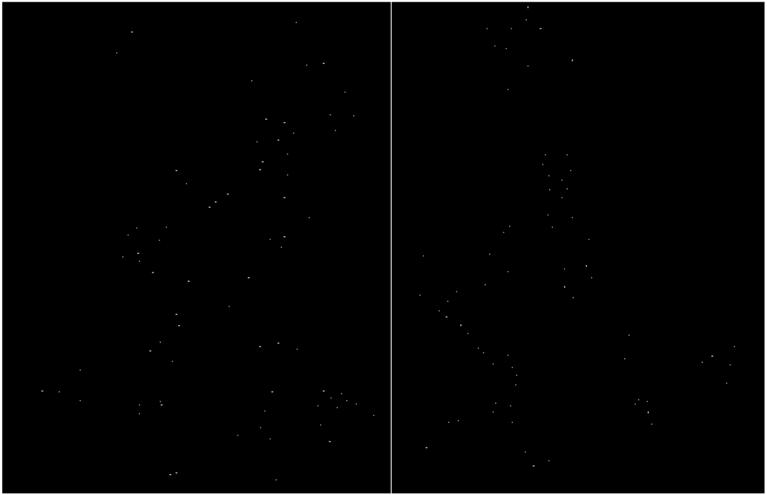
[Source: K. Grauman]

2) Find High Response



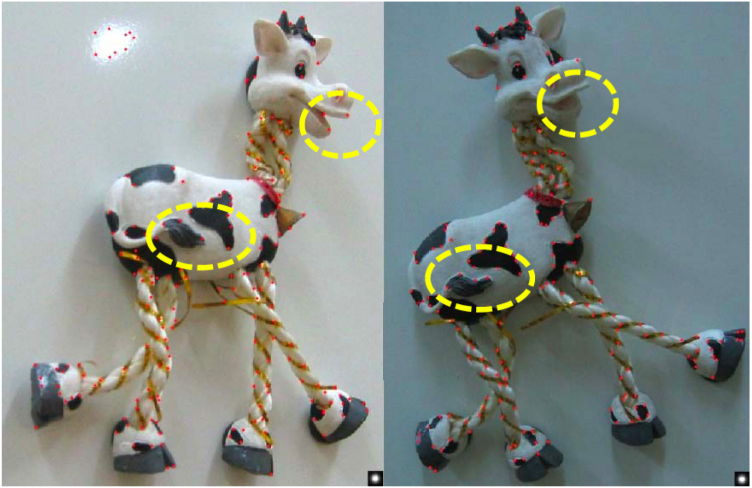
[Source: K. Grauman]

3) Non-maxima Suppresion



[Source: K. Grauman]

Results

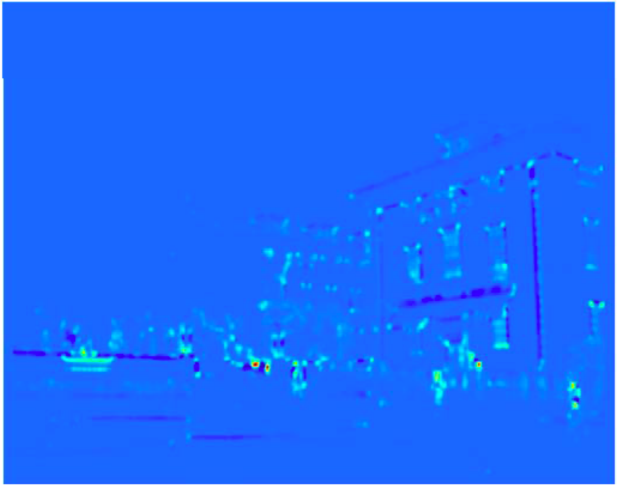


[Source: K. Grauman]

Another Example

[Source: K. Grauman]

Cornerness



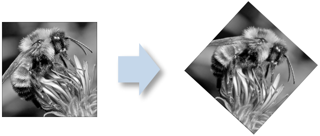
[Source: K. Grauman]

Interest Points

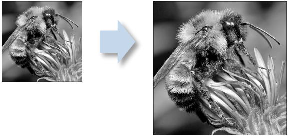
[Source: K. Grauman]

Image Transformations

Geometric:

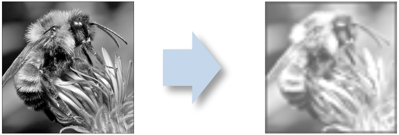


Rotation



Scale

Photometric:



Intensity change

[Source: N. Snavely]

Properties of Harris Corner Detector

- Rotation invariant?

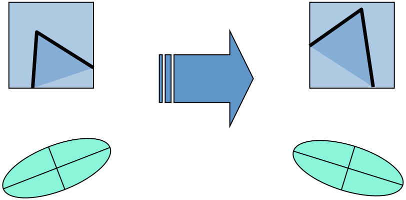
$$\mathbf{A} = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} = \mathbf{U} \begin{bmatrix} \lambda_0 & 0 \\ 0 & \lambda_1 \end{bmatrix} \mathbf{U}^T \quad \text{with} \quad \mathbf{A}\mathbf{u}_i = \lambda_i \mathbf{u}_i$$

[Source: N. Snavely]

Properties of Harris Corner Detector

- Rotation invariant?

$$\mathbf{A} = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} = \mathbf{U} \begin{bmatrix} \lambda_0 & 0 \\ 0 & \lambda_1 \end{bmatrix} \mathbf{U}^T \quad \text{with} \quad \mathbf{A}\mathbf{u}_i = \lambda_i \mathbf{u}_i$$

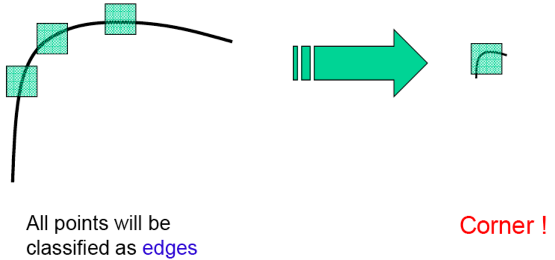


Ellipse rotates but its shape (i.e. eigenvalues) remains the same

[Source: N. Snavely]

Properties of Harris Corner Detector

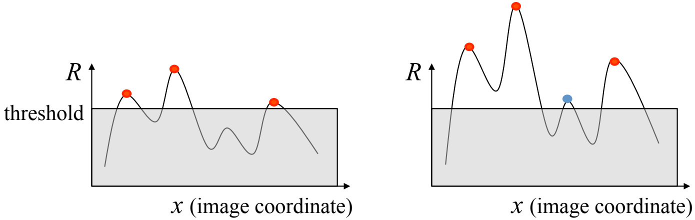
- Scale Invariant?



[Source: K. Grauman]

Properties of Harris Corner Detector

- Affine intensity change $I \rightarrow aI + b$?
- Only derivatives are used, so it's invariant to shift $I \rightarrow I + b$
- What about intensity scale?



Partially invariant to affine intensity change

[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 same level.

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

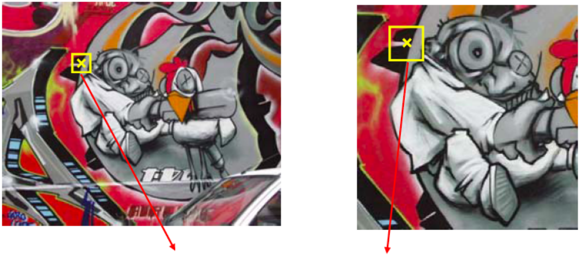
How can we independently select interest points in each image, such that the detections are repeatable across different scales?

- 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?
- More efficient to extract features **stable in both location** and **scale**.

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?
- More efficient to extract features **stable in both location** and **scale**.
- Find scale that gives local maxima of a function f in both position and scale.

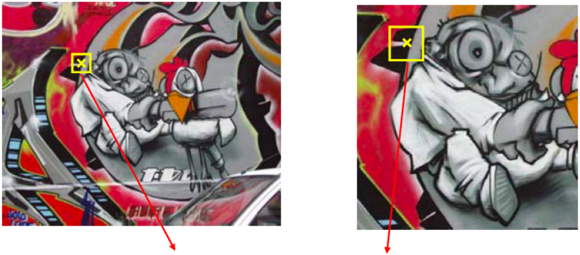


$$f(I_{i_1 \dots i_m}(x, \sigma)) = f(I_{i_1 \dots i_m}(x', \sigma'))$$

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?
- More efficient to extract features **stable in both location** and **scale**.
- Find scale that gives local maxima of a function f in both position and scale.



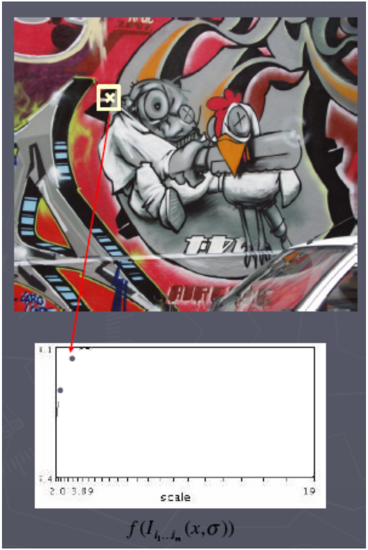
$$f(I_{i_1 \ldots i_m}(x, \sigma)) = f(I_{i_1 \ldots i_m}(x', \sigma'))$$

Automatic Scale Selection

Function responses for increasing scale (scale signature).

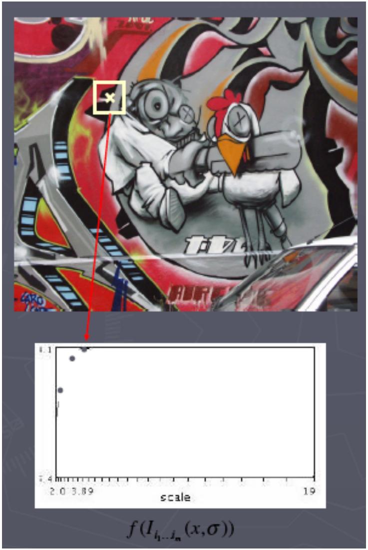
Automatic Scale Selection

Function responses for increasing scale (scale signature).

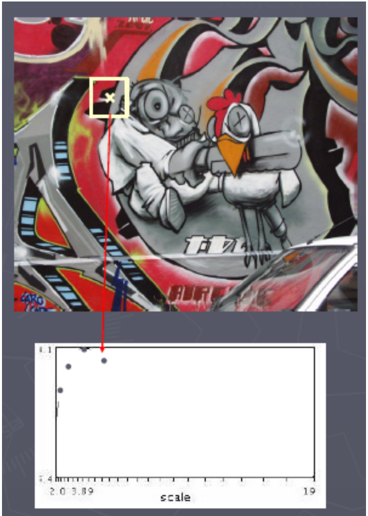


Automatic Scale Selection

Function responses for increasing scale (scale signature).



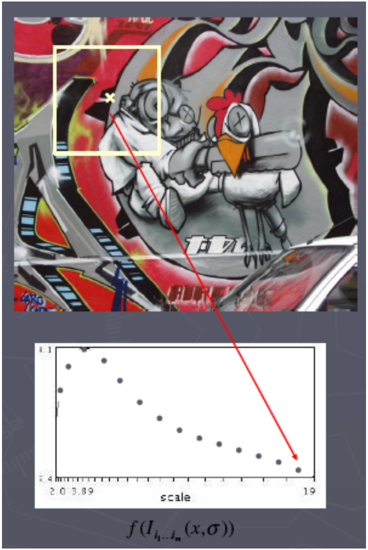
Function responses for increasing scale (scale signature).



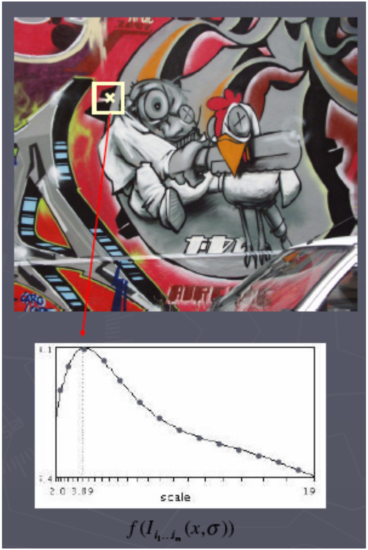
$f(I_{i_1 \ldots i_m}(x, \sigma))$

Automatic Scale Selection

Function responses for increasing scale (scale signature).

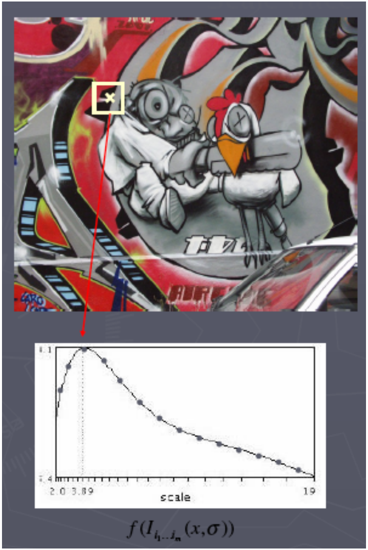


Function responses for increasing scale (scale signature).



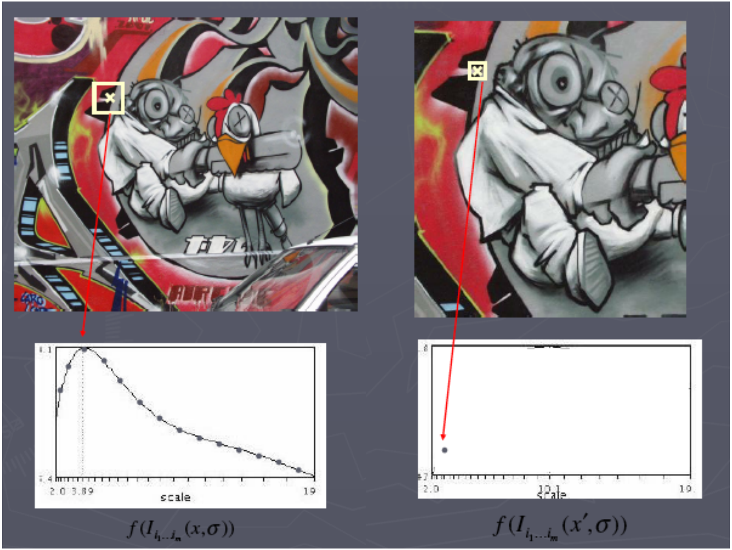
Automatic Scale Selection

Function responses for increasing scale (scale signature).



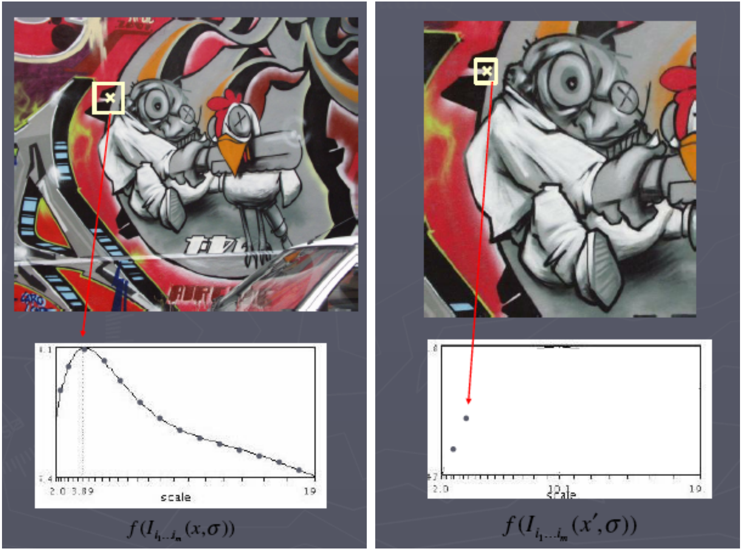
Automatic Scale Selection

Function responses for increasing scale (scale signature).



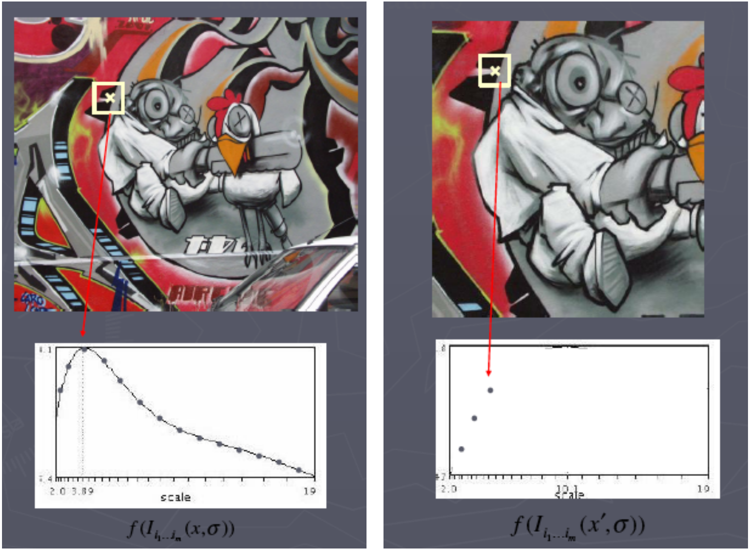
Automatic Scale Selection

Function responses for increasing scale (scale signature).



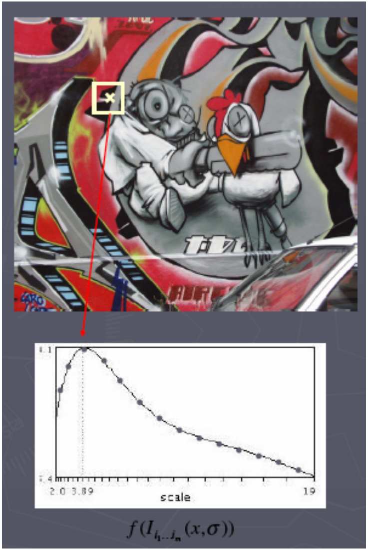
Automatic Scale Selection

Function responses for increasing scale (scale signature).

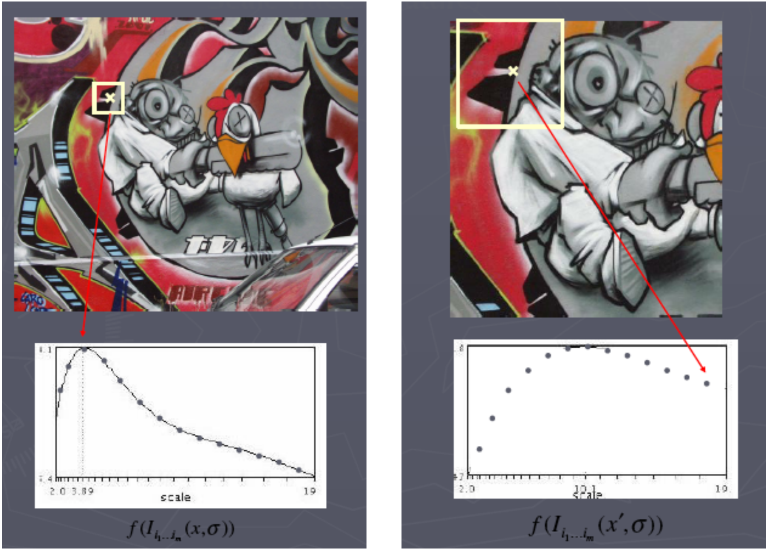


Automatic Scale Selection

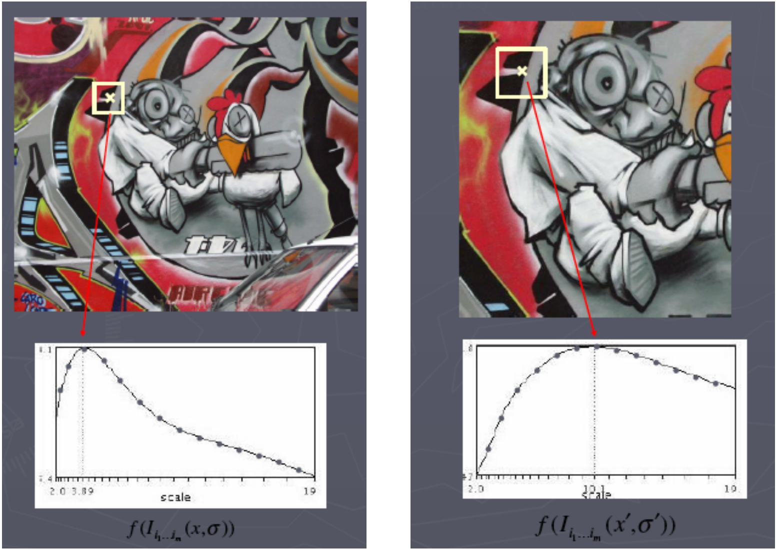
Function responses for increasing scale (scale signature).



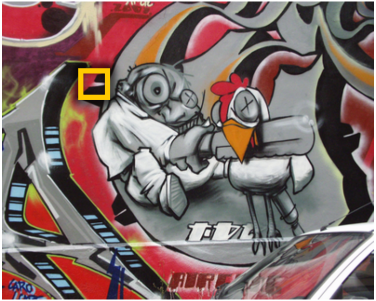
Function responses for increasing scale (scale signature).



Function responses for increasing scale (scale signature).



- Instead of computing f for larger and larger windows, we can implement using a fixed window size with a Gaussian pyramid

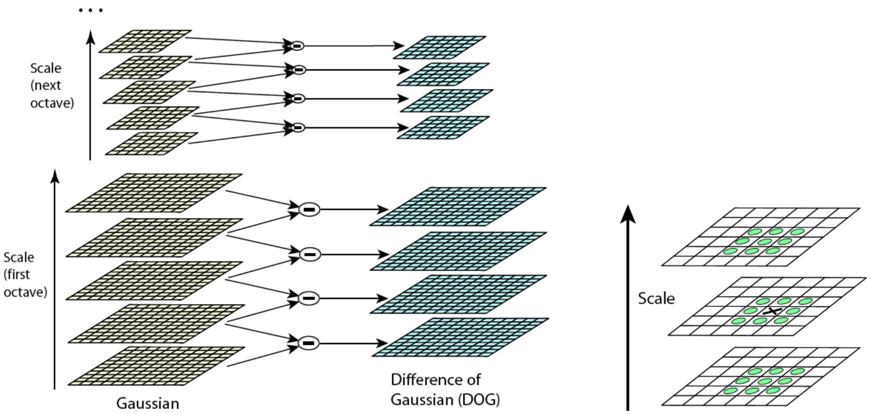


(sometimes need to create in-between levels, e.g. a ¾-size image)

[Source: N. Snavely]

What can the signature function be?

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

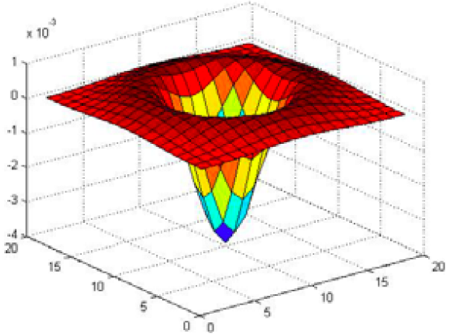


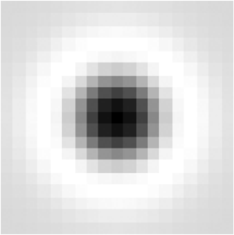
[Source: R. Szeliski]

Blob detection

- **Laplacian of Gaussian**: Circularly symmetric operator for blob detection in 2D

$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

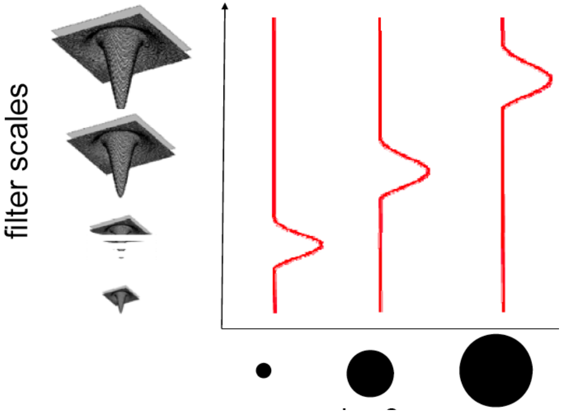




[Source: K. Grauman]

Blob detection in 2D: scale selection

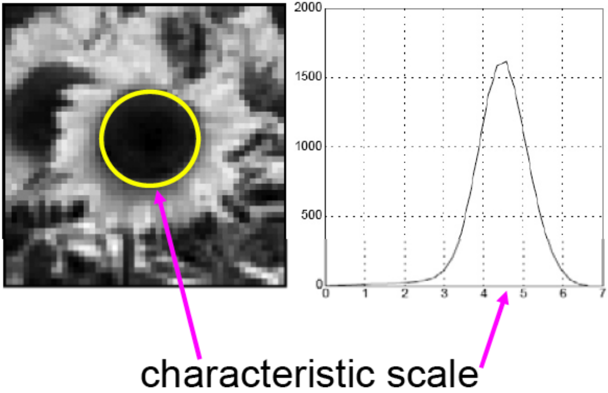
Laplacian-of-Gaussian = blob detector



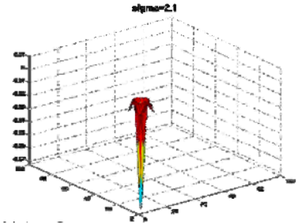
[Source: B. Leibe]

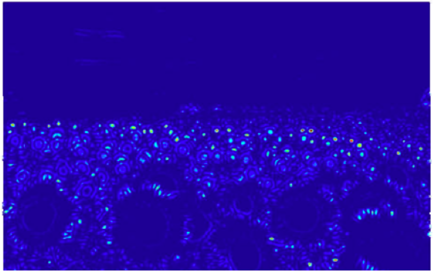
Characteristic Scale

- We define the **characteristic scale** as the scale that produces peak of Laplacian response



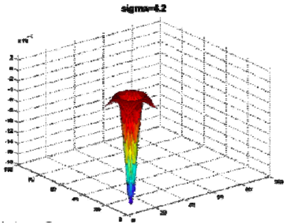
[Source: S. Lazebnik]

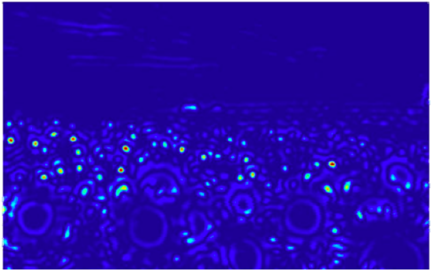




[Source: K. Grauman]

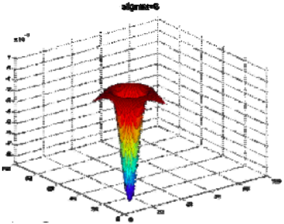
Example

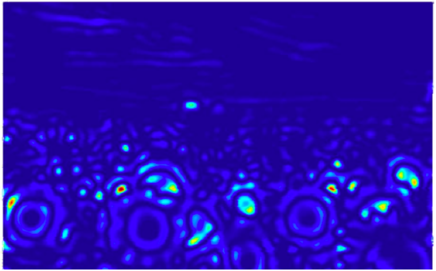




[Source: K. Grauman]

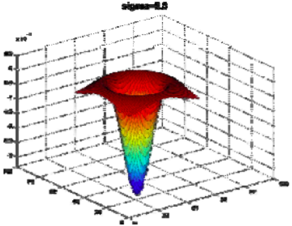
Example

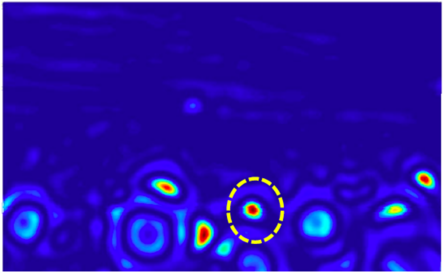




[Source: K. Grauman]

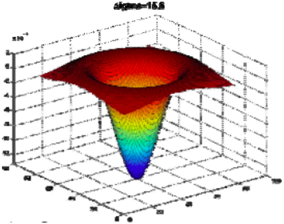
Example

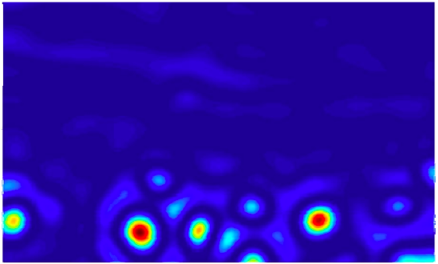




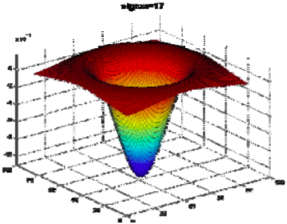
[Source: K. Grauman]

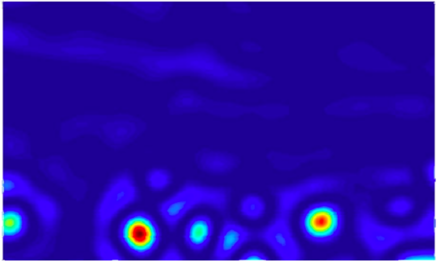
Example





[Source: K. Grauman]

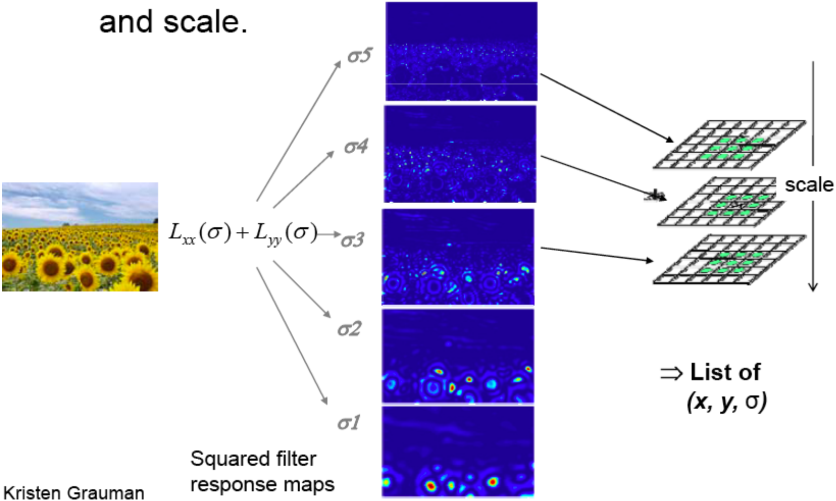




[Source: K. Grauman]

Scale invariant interest points

Interest points are local maxima in both position and scale.



Squared filter response maps

Kristen Grauman

Example

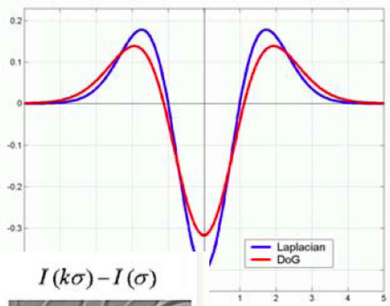
[Source: S. Lazebnik]

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

(Laplacian)

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

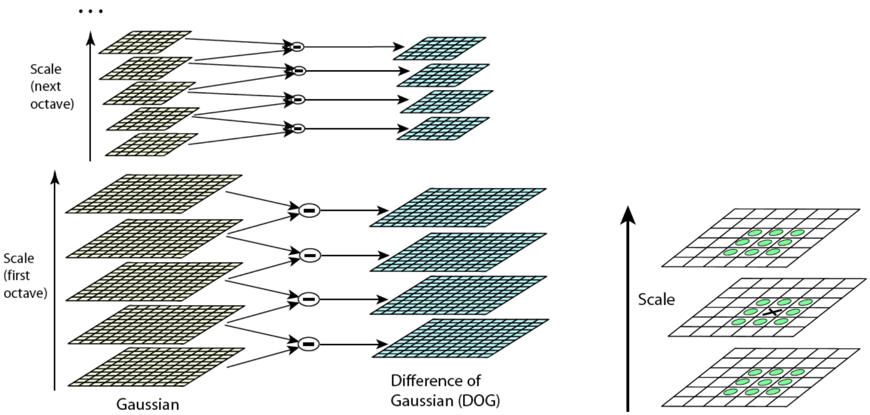
(Difference of Gaussians)



[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]

Properties of the ideal feature

- **Local**: features are local, so robust to occlusion and clutter (no prior segmentation).
- **Invariant**: to certain transformations, e.g, scale, rotation.

Properties of the ideal feature

- **Local**: features are local, so robust to occlusion and clutter (no prior segmentation).
- **Invariant**: to certain transformations, e.g, scale, rotation.
- **Robust**: noise, blur, discretization, compression, etc. do not have a big impact on the feature.

Properties of the ideal feature

- **Local**: features are local, so robust to occlusion and clutter (no prior segmentation).
- **Invariant**: to certain transformations, e.g, scale, rotation.
- **Robust**: noise, blur, discretization, compression, etc. do not have a big impact on the feature.
- **Distinctive**: individual features can be matched to a large database of objects.

Properties of the ideal feature

- **Local**: features are local, so robust to occlusion and clutter (no prior segmentation).
- **Invariant**: to certain transformations, e.g, scale, rotation.
- **Robust**: noise, blur, discretization, compression, etc. do not have a big impact on the feature.
- **Distinctive**: individual features can be matched to a large database of objects.
- **Quantity**: many features can be generated for even small objects.

Properties of the ideal feature

- **Local**: features are local, so robust to occlusion and clutter (no prior segmentation).
- **Invariant**: to certain transformations, e.g, scale, rotation.
- **Robust**: noise, blur, discretization, compression, etc. do not have a big impact on the feature.
- **Distinctive**: individual features can be matched to a large database of objects.
- **Quantity**: many features can be generated for even small objects.
- **Accurate**: precise localization.

Properties of the ideal feature

- **Local**: features are local, so robust to occlusion and clutter (no prior segmentation).
- **Invariant**: to certain transformations, e.g, scale, rotation.
- **Robust**: noise, blur, discretization, compression, etc. do not have a big impact on the feature.
- **Distinctive**: individual features can be matched to a large database of objects.
- **Quantity**: many features can be generated for even small objects.
- **Accurate**: precise localization.
- **Efficient**: close to real-time performance.

Properties of the ideal feature

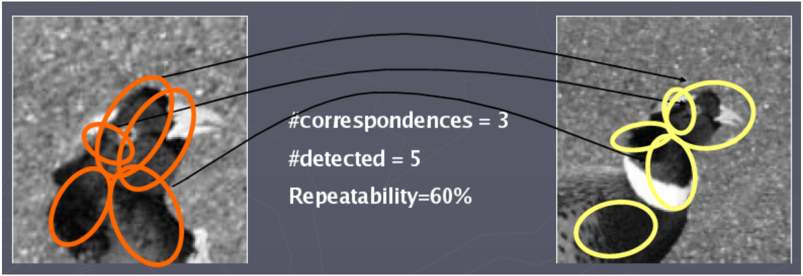
- **Local**: features are local, so robust to occlusion and clutter (no prior segmentation).
- **Invariant**: to certain transformations, e.g, scale, rotation.
- **Robust**: noise, blur, discretization, compression, etc. do not have a big impact on the feature.
- **Distinctive**: individual features can be matched to a large database of objects.
- **Quantity**: many features can be generated for even small objects.
- **Accurate**: precise localization.
- **Efficient**: close to real-time performance.

A lot of other interest point detectors

- Hessian
- Lowe: DoG
- Lindeberg: scale selection
- Miikolajczyk & Schmid: Hessian/Harris-Laplacian/Affine
- Tuyttelaars & Van Gool: EBR and IBR
- Matas: MSER
- Kadir & Brrady: Salient Regions
- Speeded–Up Robust Features (SURF) of Bay et al.
- ...

Evaluation criteria: repeatability

- **Repeatability rate**: percentage of detected features that have correct corresponding points
- What's the problem of this?



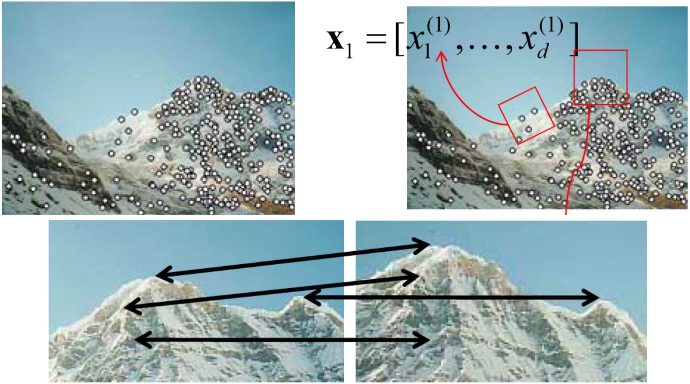
[Source: T. Tuytelaars]

Evaluation criteria: repeatability

- Two points are in correspondence if the intersection over union is bigger than a certain threshold.
- Look for affine invariant features!

[Source: T. Tuytelaars]

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



[Source: K. Grauman]

The ideal feature descriptor

- Repeatable (invariant/robust)
- Distinctive
- Compact
- Efficient

How to achieve invariance?

- Make sure your detector is invariant
- Design an invariant feature descriptor

How to achieve invariance?

- Make sure your detector is invariant
- Design an invariant feature descriptor
  - Simplest descriptor: a single 0. What's this invariant to?

How to achieve invariance?

- Make sure your detector is invariant
- Design an invariant feature descriptor
  - Simplest descriptor: a single 0. What's this invariant to?
  - Next simplest descriptor: a square window of pixels. What's this invariant to?

How to achieve invariance?

- Make sure your detector is invariant
- Design an invariant feature descriptor
  - Simplest descriptor: a single 0. What's this invariant to?
  - Next simplest descriptor: a square window of pixels. What's this invariant to?
  - Lets look at some better approaches

[Source: N. Snavely]

How to achieve invariance?

- Make sure your detector is invariant
- Design an invariant feature descriptor
  - Simplest descriptor: a single 0. What's this invariant to?
  - Next simplest descriptor: a square window of pixels. What's this invariant to?
  - Lets look at some better approaches

[Source: N. Snavely]

Invariances

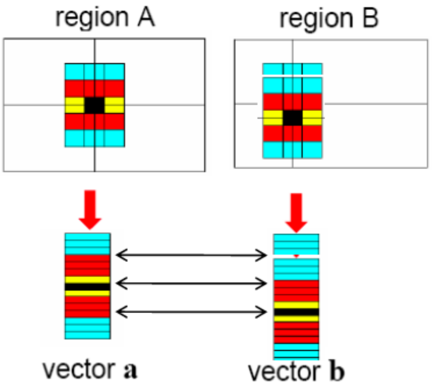
[Source: T. Tuytelaars]

Invariances

[Source: T. Tuytelaars]

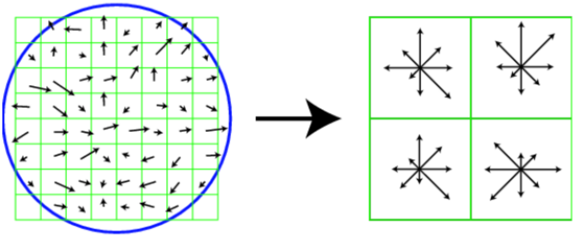
Raw Pixels as Local Descriptrs

- 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.



SIFT descriptor [Lowe 2004]

- Compute the gradient at each pixel in a 16×16 window around the detected keypoint, using the appropriate level of the Gaussian pyramid at which the keypoint was detected.
- Downweight gradients by a Gaussian fall-off function (blue circle) to reduce the influence of gradients far from the center.
- In each 4×4 quadrant, compute a gradient orientation histogram using 8 orientation histogram bins.

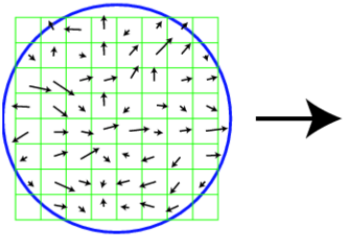


(a) image gradients (b) keypoint descriptor

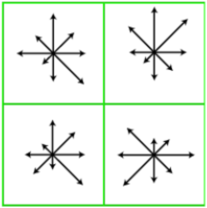
[Source: R. Szeliski]

SIFT descriptor [Lowe 2004]

- Compute the gradient at each pixel in a 16×16 window around the detected keypoint, using the appropriate level of the Gaussian pyramid at which the keypoint was detected.
- Downweight gradients by a Gaussian fall-off function (blue circle) to reduce the influence of gradients far from the center.
- In each 4×4 quadrant, compute a gradient orientation histogram using 8 orientation histogram bins.



(a) image gradients



(b) keypoint descriptor

[Source: R. Szeliski]

SIFT descriptor [Lowe 2004]

- 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.

SIFT descriptor [Lowe 2004]

- 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.
- 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 [Lowe 2004]

- 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.
- 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: Scale Invariant Feature Transform

SIFT descriptor [Lowe 2004]

- 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.
- 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: Scale Invariant Feature Transform
- Great engineering effort!

SIFT descriptor [Lowe 2004]

- 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.
- 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: Scale Invariant Feature Transform
- Great engineering effort!
- Why subpatches?

SIFT descriptor [Lowe 2004]

- 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.
- 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: Scale Invariant Feature Transform
- Great engineering effort!
- Why subpatches?
- Why does SIFT have some illumination invariance?

SIFT descriptor [Lowe 2004]

- 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.
- 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: Scale Invariant Feature Transform
- Great engineering effort!
- Why subpatches?
- Why does SIFT have some illumination invariance?

SIFT descriptor [Lowe 2004]

Extraordinarily robust matching technique

- Changes in viewpoint: up to about 60 degree out of plane rotation
- Changes in illumination: sometimes even day vs. night
- Fast and efficient – can run in real time
- Lots of code available

[Source: S. Seitz]

Example

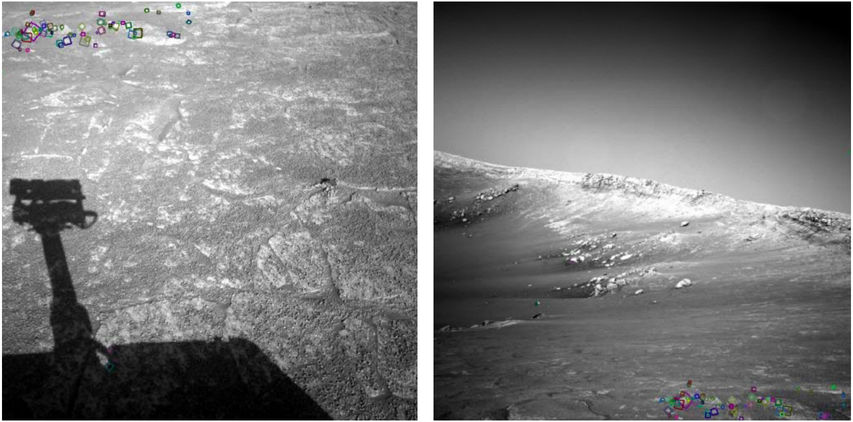


Figure: NASA Mars Rover images with SIFT feature matches

[Source: N. Snavely]

PCA-SIFT

- The dimensionality of SIFT is very 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

SIFT properties

Invariant to

- Scale
- Rotation

Partially invariant to

- Illumination changes
- Camera viewpoint
- Occlusion, clutter

Making descriptor rotation invariant (MOPS)

- Rotate patch according to its dominant gradient orientation
- This puts the patches into a canonical orientation
- Multiscale Oriented PatcheS descriptor

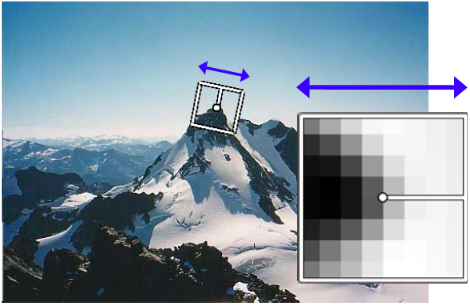
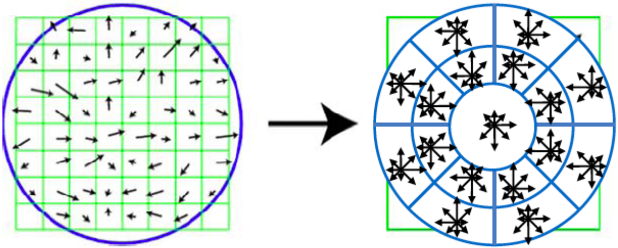


Figure: Figure from M. Brown

[Source: K. Grauman]

Gradient location-orientation histogram (GLOH)

- Developed by Mikolajczyk and Schmid (2005): variant on SIFT that uses a log-polar binning structure instead of the four quadrants.
- The spatial bins are 11, and 15, with eight angular bins (except for the central region), for a total of 17 spatial bins and 16 orientation bins.
- The 272D histogram is then projected onto a 128D descriptor using PCA trained on a large database.



(a) image gradients (b) keypoint descriptor

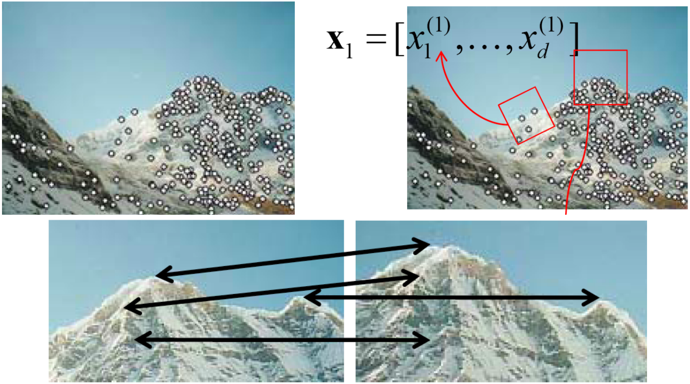
[Source: R. Szeliski]

Other Descriptors

- Steerable filters
- moment invariants,
- complex filters
- shape context,
- PCA-SIFT,
- HOG,
- SURF
- DAISY
- $\cdots$

Local features

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



[Source: K. Grauman]

Matching local features

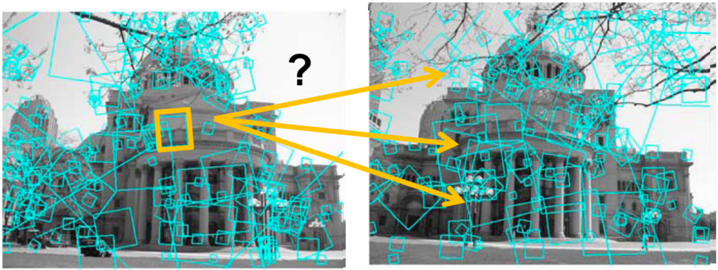
Once we have extracted features and their descriptors, we need to match the features between these images.

- **Matching strategy:** which correspondences are passed on to the next stage
- Devise **efficient data structures and algorithms** to perform this matching

Figure: Images from K. Grauman

Matching local features

- To **generate candidate matches**, find patches that have the most similar appearance (e.g., lowest SSD)
- Simplest approach: **compare them all**, take the closest (or closest k, or within a thresholded distance)



[Source: K. Grauman]

Ambiguous matches

- At what SSD value do we have a good match?
- To add robustness, consider ratio of distance to best match to distance to second best match
  - If low, first match looks good.
  - If high, could be ambiguous match.

[Source: K. Grauman]

Matching SIFT Descriptors

- Nearest neighbor (Euclidean distance)
- Threshold ratio of nearest to 2nd nearest descriptor

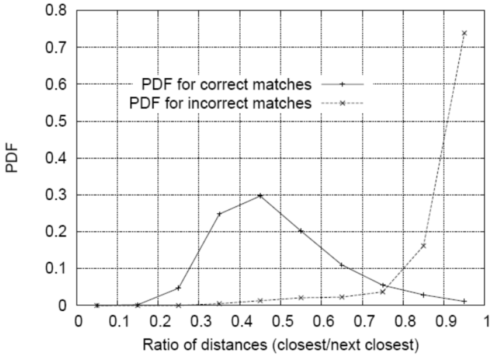


Figure: Images from D. Lowe

[Source: K. Grauman]

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

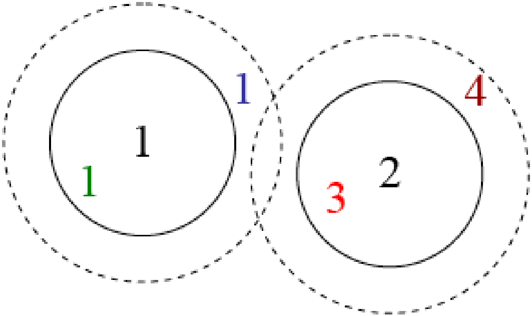
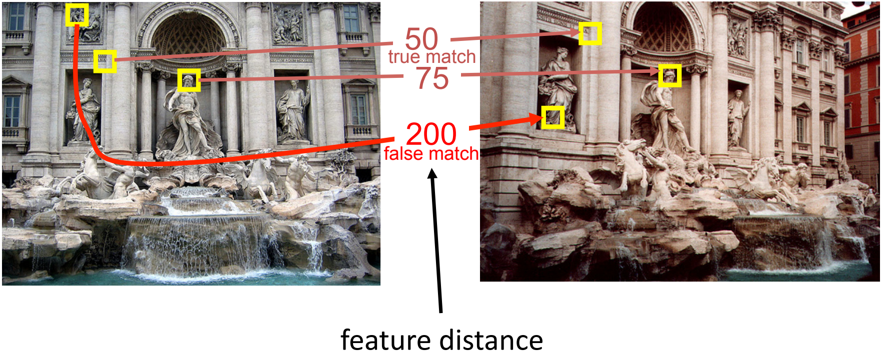


Figure: Images from R. Szeliski

How to measure performance

- How can we measure the performance of a feature matcher?



[Source: N. Snavely]

How to quantize how good is our matching?

- **TP: true positives**, i.e., number of correct matches
- **FN: false negatives**, matches that were not correctly detected
- **FP: false positives**, proposed matches that are incorrect
- **TN: true negatives**, non-matches that were correctly rejected.

True positive rate (recall) $$TPR = \frac{TP}{TP + FN} = \frac{TP}{P}$$

False positive rate $$FPR = \frac{FP}{FP + TN} = \frac{FP}{N}$$

positive predictive value (precision) $$PPV = \frac{TP}{TP + FP} = \frac{TP}{P'}$$

accuracy $$ACC = \frac{TP + TN}{P + N}$$

Measuring performance

- Any particular matching strategy (at a particular threshold or parameter setting) can be rated by the TPR and FPR numbers
- We want TPR=1 (recall) and FPR=0
- As we vary the matching threshold, we obtain a family of such points, i.e., **receiver operating characteristic (ROC curve)**
- The closer this curve lies to the upper left corner, the better its performance

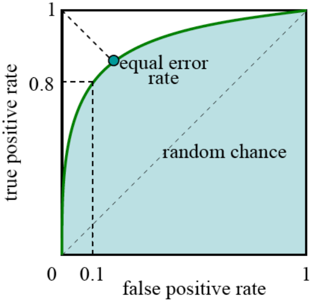


Figure: Images from R. Szeliski

Measuring performance

- **Area under the curve (AUC)** is a way to summarize ROC with 1 number.
- **Mean average precision**, which is the average precision (PPV) as you vary the threshold, i.e., area under the curve in the precision-recall curve.
- The **equal error rate** is sometimes used as well.

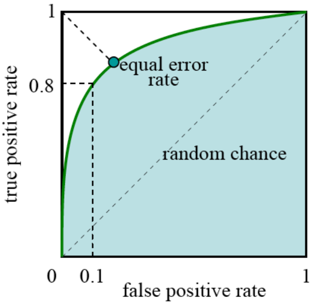


Figure: Images from R. Szeliski

Applications of local invariant features

- Wide baseline stereo
- Motion tracking
- Panoramas
- Mobile robot navigation
- 3D reconstruction
- Recognition

[Source: K. Grauman]

Wide Baseline Stereo

[Source: T. Tuytelaars]

Recognizing the Same Object

Schmid and Mohr 1997

Sivic and Zisserman, 2003

Rothganger et al. 2003

Lowe 2002

[Source: K. Grauman]

Motion Tracking

Figure: Images from J. Pilet

Summary

Interest point detection

- Harris corner detector
- Laplacian of Gaussian, automatic scale selection
- Difference of Gaussians

Invariant descriptors

- Rotation according to dominant gradient direction
- Histograms for robustness to small shifts and translations (SIFT descriptor)
- Polar coordinate descriptors GLOH.

Next class ... more sophisticated matching