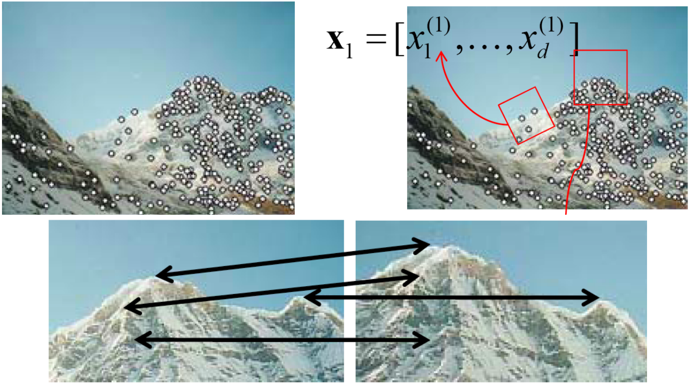
Image Features:

Local Descriptors

Local Features

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



[Source: K. Grauman]

The Ideal Feature Descriptor

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

Invariances

e.g. scale, translation, rotation

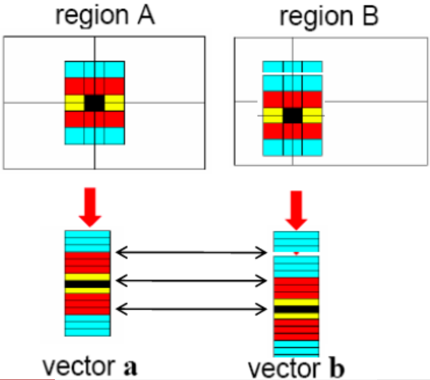
[Source: T. Tuytelaars]

Invariances

[Source: T. Tuytelaars]

What If We Just Took Pixels?

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



Tons Of Better Options

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

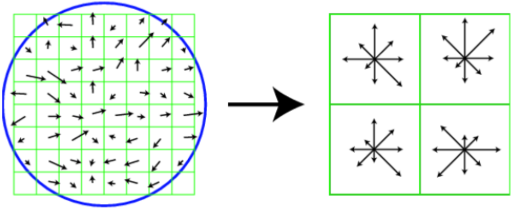
Tons Of Better Options

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

SIFT Descriptor [Lowe 2004]

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

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


(a) image gradients (b) keypoint descriptor

SIFT Descriptor

1. Our scale invariant interest point detector gives scale ρ for each keypoint

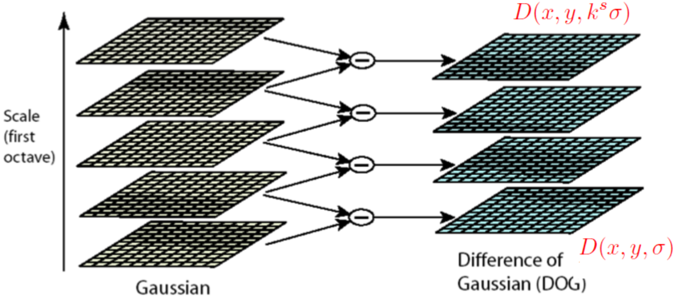
$$I_s = I * G_{k^s\sigma}$$

$$\vdots$$

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

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

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



[Adopted from: F. Flores-Mangas]

SIFT Descriptor

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

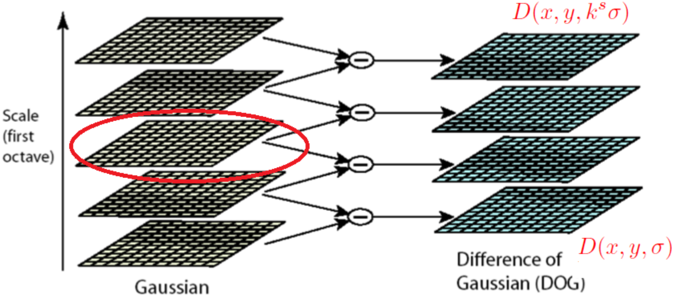
$$I_s = I * G_{k^s\sigma}$$

$$\vdots$$

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

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

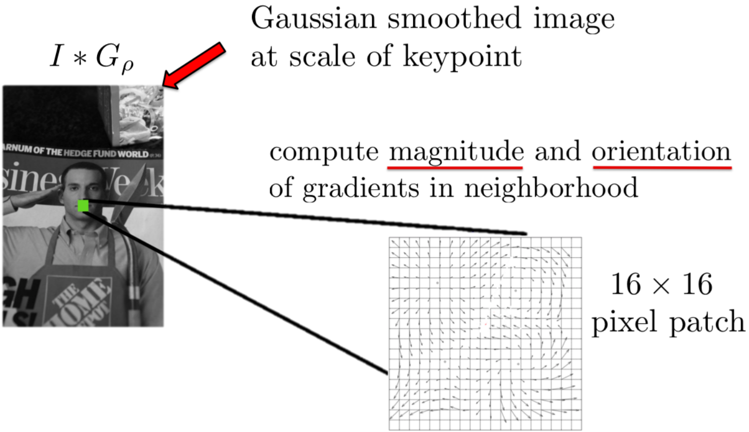
$$I_0 = I * G_{\sigma}$$



[Adopted from: F. Flores-Mangas]

SIFT Descriptor

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



[Adopted from: F. Flores-Mangas]

SIFT Descriptor

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

magnitude of gradient:

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

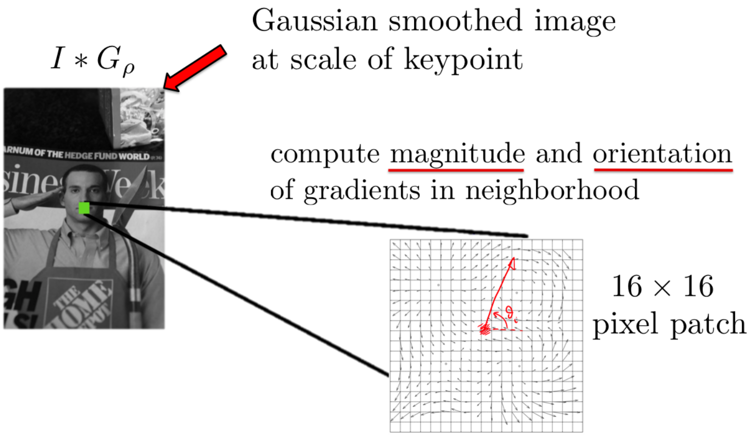
gradient orientation:

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

(in case you forgot ;))

SIFT Descriptor

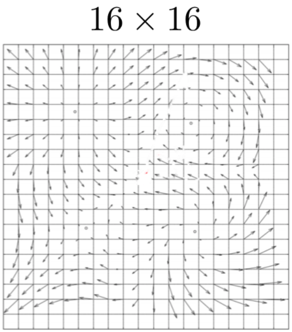
4. Compute dominant orientation of each keypoint. How?

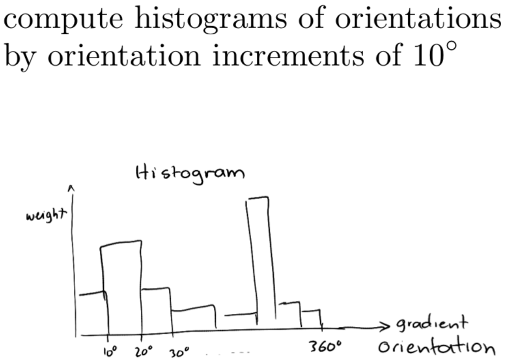


[Adopted from: F. Flores-Mangas]

SIFT Descriptor: Computing Dominant Orientation

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





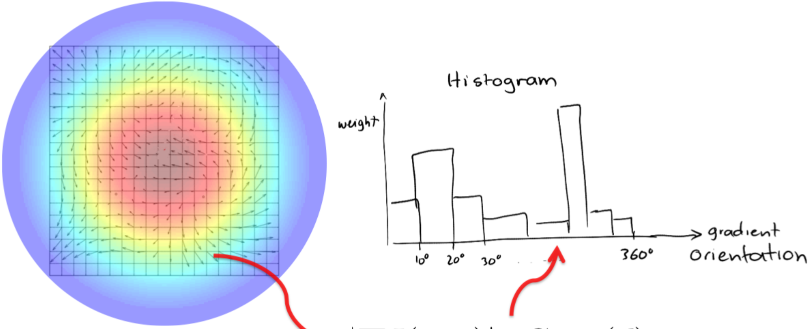
[Adopted from: F. Flores-Mangas]

SIFT Descriptor: Computing Dominant Orientation

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

$G_{1.5\rho}$

weight influence of orientation based on distance from center



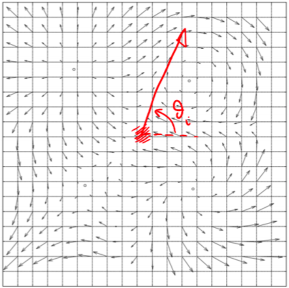
$$|\nabla I(x, y)| \cdot G_{1.5\rho}(d)$$

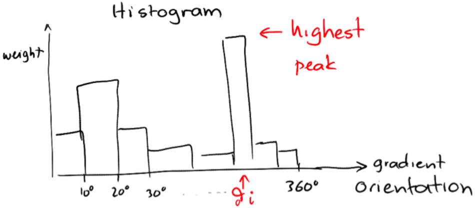
[Adopted from: F. Flores-Mangas]

SIFT Descriptor: Computing Dominant Orientation

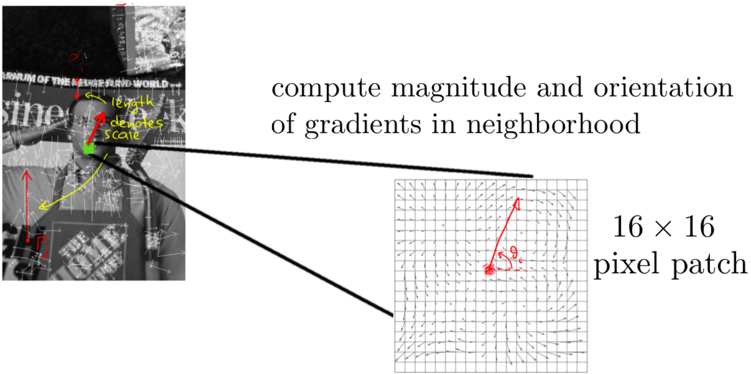
- Compute a histogram of gradient orientations, each bin covers 10°
- Orientations closer to the keypoint center should contribute more
- Orientation giving the peak in the histogram is the keypoint's orientation

16×16





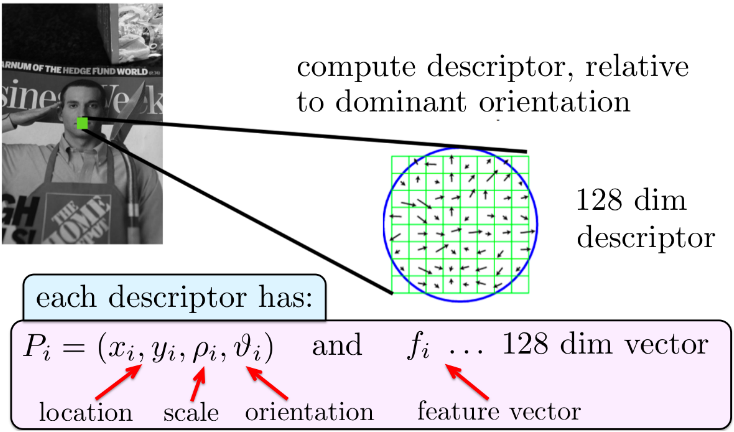
4. Compute dominant orientation



[Adopted from: F. Flores-Mangas]

SIFT Descriptor

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

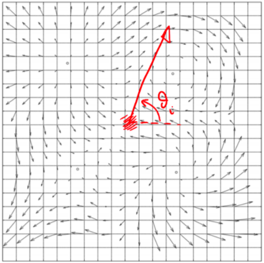


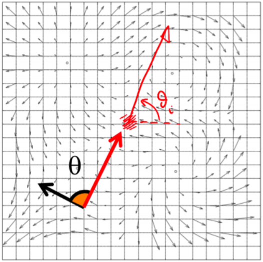
[Adopted from: F. Flores-Mangas]

SIFT Descriptor: Computing the Feature Vector

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

16×16 patch
centered in (x_i, y_i)



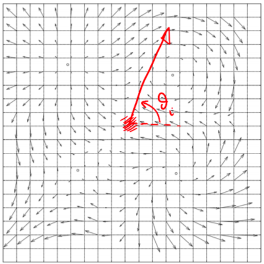


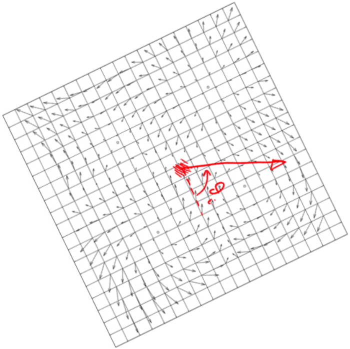
[Adopted from: F. Flores-Mangas]

SIFT Descriptor: Computing the Feature Vector

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

16×16 patch
centered in (x_i, y_i)

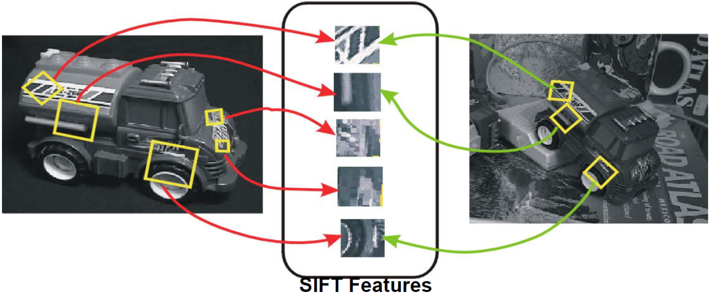




[Adopted from: F. Flores-Mangas]

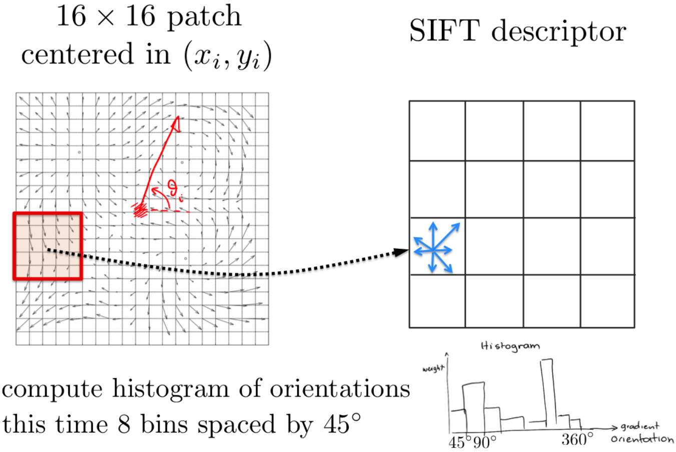
SIFT Descriptor: Computing the Feature Vector

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



SIFT Descriptor: Computing the Feature Vector

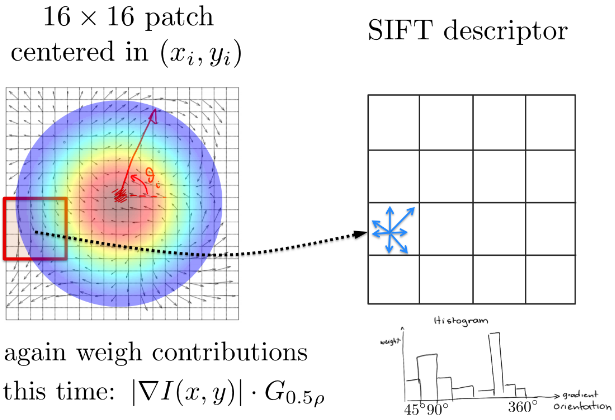
- Compute the orientations **relative** to the **dominant orientation**
- Otherwise rotating an object would phase shift entries in histogram
- Form a 4×4 grid. For each grid cell compute a histogram of orientations for 8 orientation bins spaced apart by 45°



[Adopted from: F. Flores-Mangas]

SIFT Descriptor: Computing the Feature Vector

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

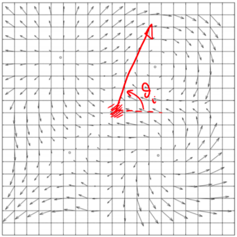


[Adopted from: F. Flores-Mangas]

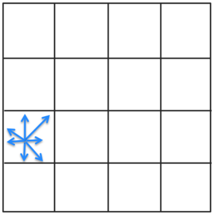
SIFT Descriptor: Computing the Feature Vector

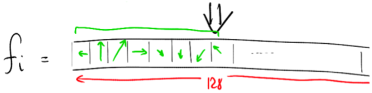
- Compute the orientations **relative** to the **dominant orientation**
- Otherwise rotating an object would phase shift entries in histogram
- Form a 4×4 grid. For each grid cell compute a histogram of orientations for 8 orientation bins spaced apart by 45°
- Form the 128 dimensional feature vector

16×16 patch
centered in (x_i, y_i)



SIFT descriptor





[Adopted from: F. Flores-Mangas]

SIFT Descriptor: Post-processing

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

SIFT Descriptor: Post-processing

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

SIFT Descriptor: Post-processing

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

SIFT Descriptor: Post-processing

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

SIFT Descriptor: Post-processing

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

Properties of SIFT

Invariant to:

- Scale
- Rotation

Partially invariant to:

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

Also important:

- Fast and efficient – can run in real time
- Lots of code available

Examples

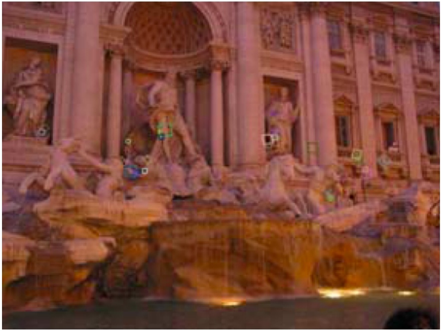


Figure: Matching in day / night under viewpoint change

[Source: S. Seitz]

Example

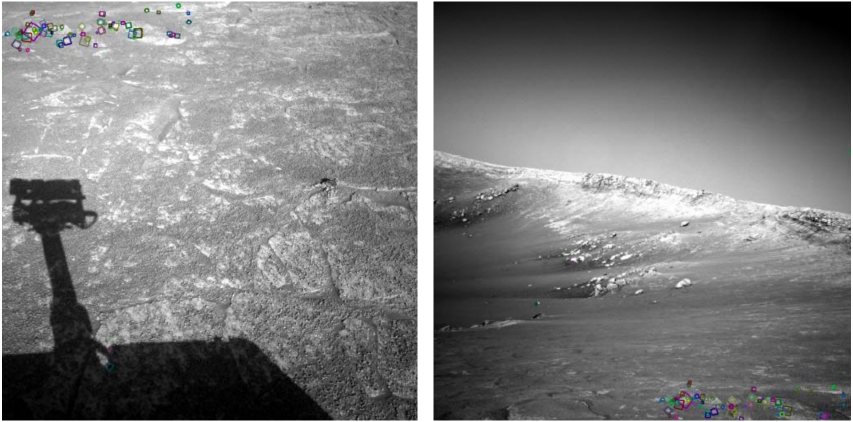


Figure: NASA Mars Rover images with SIFT feature matches

[Source: N. Snavely]

PCA-SIFT

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

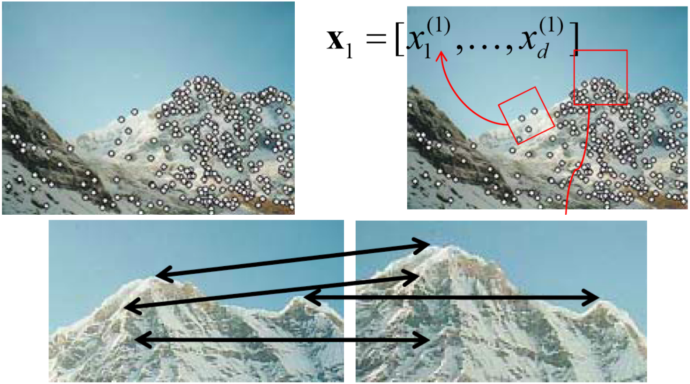
[Source: R. Urtasun]

Other Descriptors

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

Local Features

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



[Source: K. Grauman]

Image Features:

Matching the Local Descriptors

Matching the Local Descriptors

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

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

Figure: Images from K. Grauman

Matching the Local Descriptors

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

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

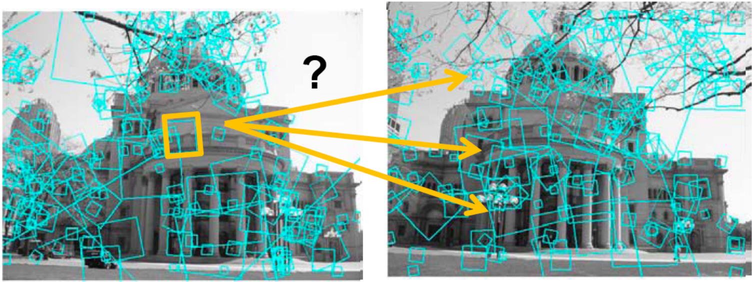
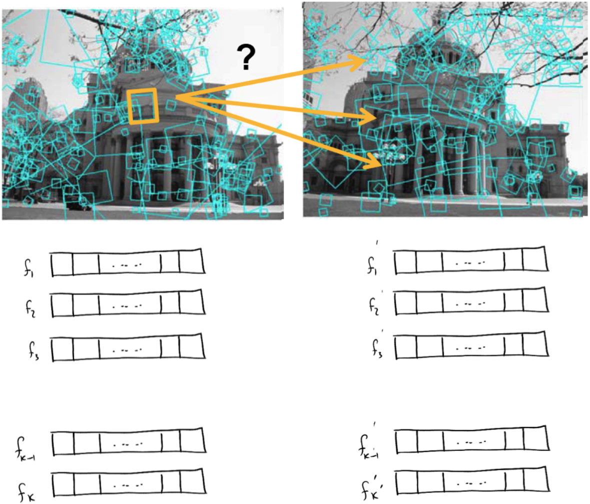


Figure: Images from K. Grauman

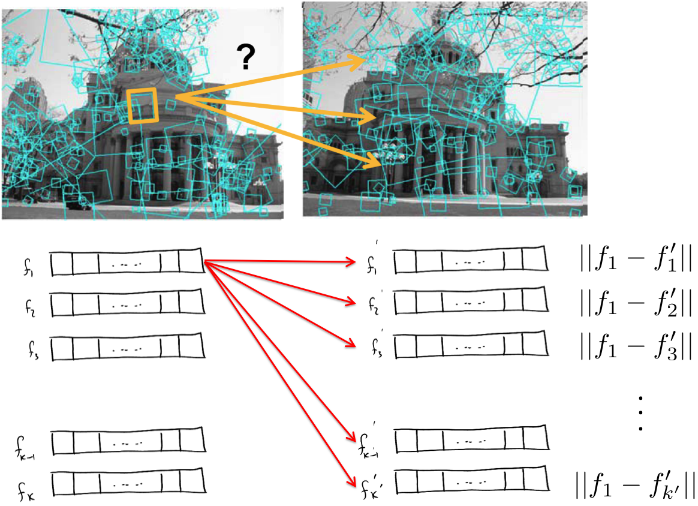
Matching the Local Descriptors

- Simple: **Compare them all**, compute Euclidean distance



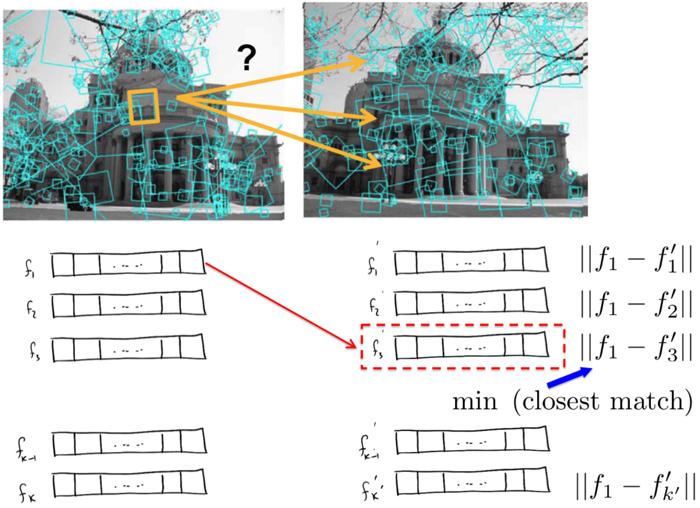
Matching the Local Descriptors

- Simple: **Compare them all**, compute Euclidean distance



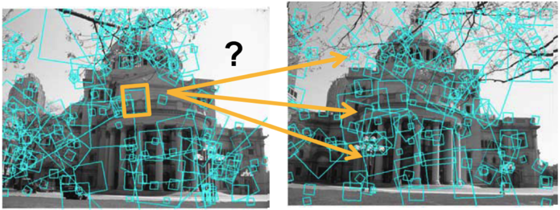
Matching the Local Descriptors

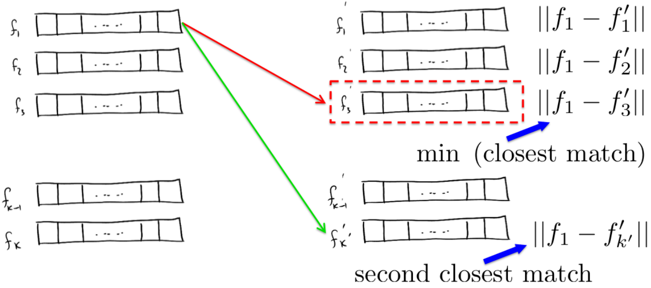
- Find closest match (min distance). How do we know if match is **reliable**?



Matching the Local Descriptors

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



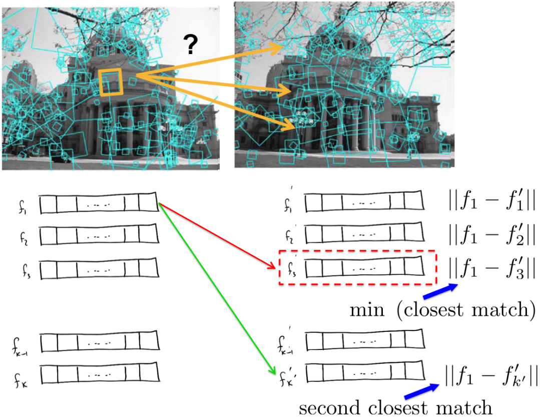


Matching the Local Descriptors

- Compute the ratio:

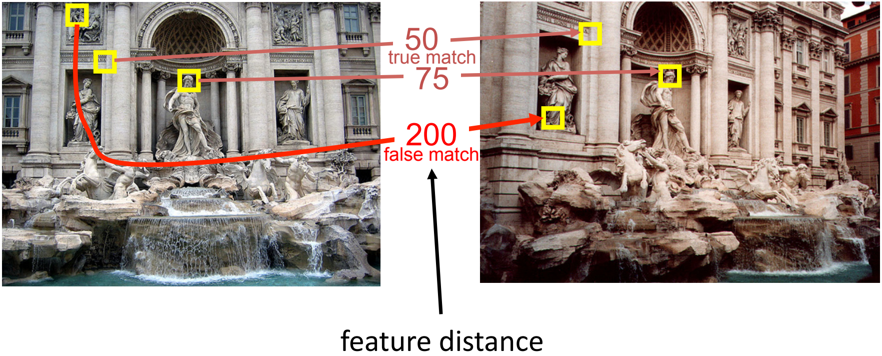
$$\phi_i = \frac{||f_i - f'^{*}_i||}{||f_i - f'^{**}_i||}$$

where f'^{*}_i is the closest and f'^{**}_i second closest match to f_i.



Which Threshold to Use?

- Setting the threshold too high results in too many false positives, i.e., incorrect matches being returned.
- Setting the threshold too low results in too many false negatives, i.e., too many correct matches being missed



[Source: N. Snavely]

Which Threshold to Use?

- Threshold ratio of nearest to 2nd nearest descriptor
- Typically: $\phi_i < 0.8$

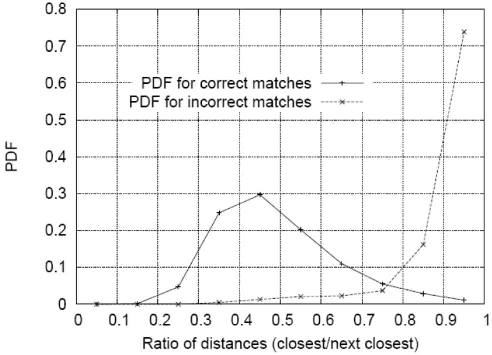


Figure: Images from D. Lowe

[Source: K. Grauman]

Applications of Local Invariant Features

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

[Source: K. Grauman]

Wide Baseline Stereo

[Source: T. Tuytelaars]

Motion Tracking

Figure: Images from J. Pilet

Now What

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

Now What

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

Waldo on the road

template

Now What

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

He comes closer... We know how to solve this

template

Now What

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

Someone takes a (weird) picture of him!

template

Find My DVD!

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

Matching Planar Objects In New Viewpoints