SIFT Features

Prof. Dr. Elli Angelopoulou

Pattern Recognition Lab (Computer Science 5)

University of Erlangen-Nuremberg

Credits

- Most of the material presented in these slides is by:

  B. Babenko, M. Brown, K. Dyagilev, A. Dominitz, R. Fergus, J. Kosecka, S. Lazebnik, D. Lee, D. Lowe, J. Malik, O. Peleg, M. Pollefeys, D. Simakov, S. Thrun.

The Importance of Features

- Many computer vision topics are based on reliably identifying image features:
  - Stereo
  - Mosaicing (image stitching)
  - Tracking
  - Recognition
- Most of these applications involve matching features in different images:
  - Compute features in 2 or more images
  - Reliably identify a point in the scene in the different images, based on its feature value.
- Features seen so far:
  - Edges (gradient-based, Canny, LoG)
  - Texture (textons)
  - Color

Matching with Features

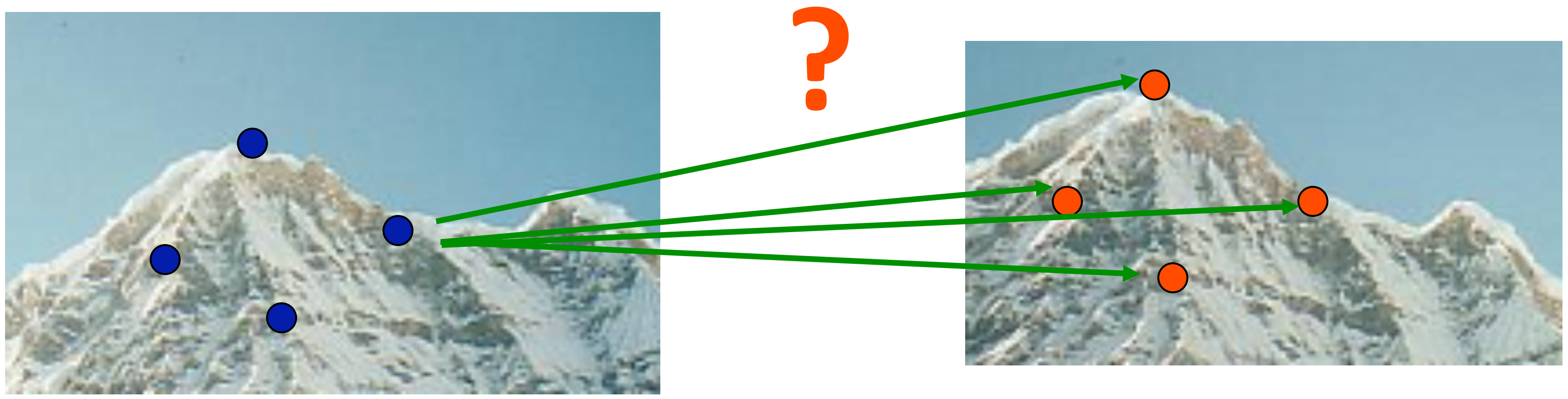
- Problem 1:
  - Detect the *same* point *independently* in both images

no chance to match!

We need a repeatable detector

Matching with Features

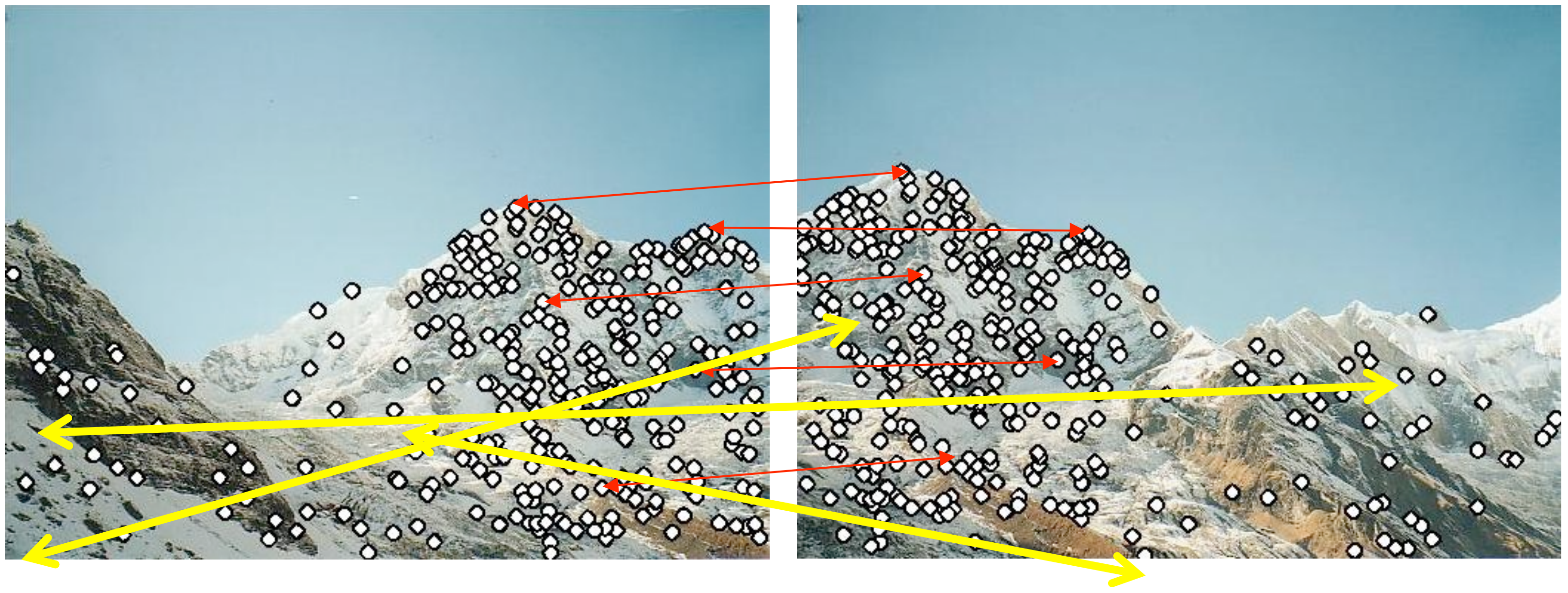
- Problem 2:
  - For each point correctly recognize the corresponding one



We need a reliable and distinctive descriptor

Matching with Features

- Problem 3:
  - Need to estimate transformation between images, despite erroneous correspondences.



Good Feature Properties

- Lots of them
- Repeatable
- Robust to orientation, scale and illumination variations
- Fast to extract and match

Properties of Good Local Features

- **Invariance =>** should be invariant to changes in illumination, scale, rotation, affine, perspective.
- **Locality =>** should be based on local information so that they can be robust to occlusion and clutter.
- **Distinctiveness =>** should be easy to match to a large database of objects.
- **Quantity =>** many features can be generated for even small objects.
- **Efficiency =>** computationally “cheap”, real-time performance.
- **Extensibility =>** can be easily extended to include additional information which should increase robustness.

SIFT

- In 1999, D. Lowe introduced a new feature together with an algorithm for using the feature in image matching. It is called the *Scale Invariant Feature Transform* (SIFT) which explicitly tries to address some of the desired properties.
- Invariances:

Scaling	Yes (very good for up to 2.5x scaling)
Rotation	Almost (up to 50 degree rotations)
Illumination	Yes
Deformation	Maybe

- Provides:

Good localization	Yes
Distinctive	Yes (BEST!)
Many keypoints	Maybe (BEST for textured scenes)
Efficient	Not quite
Extensible	Kind of

SIFT Framework

1. Scale-invariant feature detection

   Compute feature vectors that are invariant to translation, scaling, rotation, local geometric distortions and illumination.

2. Feature matching and indexing

   For each image a set of SIFT feature vectors, a.k.a. SIFT keypoints, are stored in a modified k-d tree. The best candidate match for each keypoint is found by identifying its nearest neighbor (Euclidean distance). The probability that a match is correct is determined by taking the ratio of the distance to the closest neighbor over the distance to the second closest.

3. Cluster identification via Hough Transform

   Each matched feature votes for an object transformation using HT, resulting in clusters of features voting for the same object pose.

4. Model verification using linear least squares

   Explicitly recover the transformation (between images) parameters supported by each cluster using linear least squares.

5. Outlier detection

   Remove outliers that do not satisfy the recovered transformation parameters. Re-solve for transformation parameters using the remaining points. Repeat until stable. If fewer than 3 inliers remain, reject the object match.

SIFT Keypoint Computation

1. Scale-space extrema detection

   Search over all scales and image locations. It uses a Laplacian pyramid (difference-of-Gaussians) to identify potential interest points that are invariant to scale and orientation.

2. Keypoint localization and filtering

   At each candidate location, a detailed model is fit to determine location and scale. Unstable keypoints are eliminated from further processing.

3. Orientation assignment

   One or more orientations are assigned to each keypoint based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations.

4. Creation of SIFT keypoint descriptor

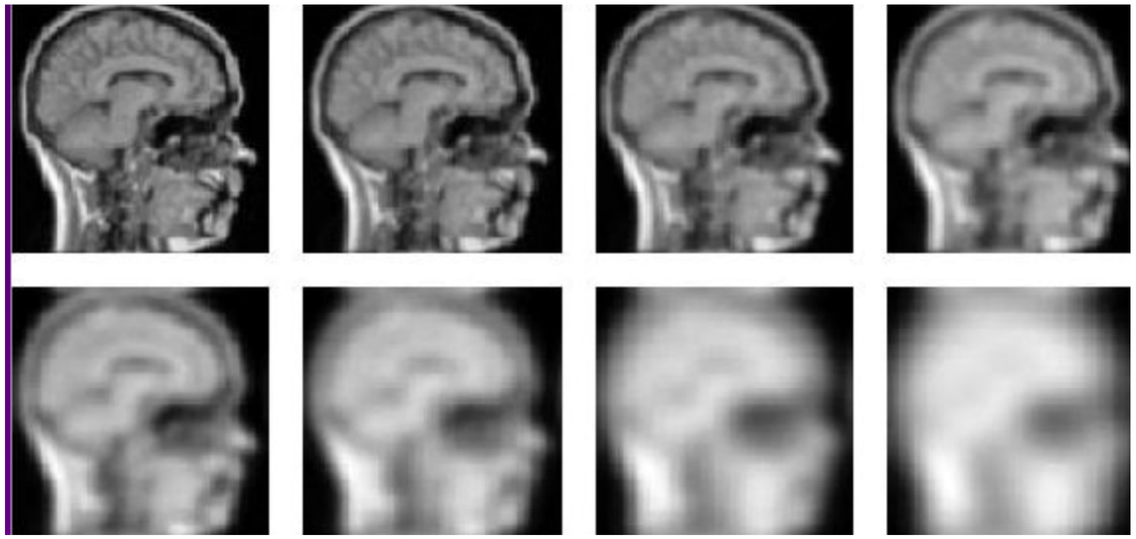
   The local image gradients are measured at the selected scale in the region around each keypoint. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination.

SIFT Keypoint Computation

1. Scale-space extrema detection
2. Keypoint localization and filtering
3. Orientation assignment
4. Creation of SIFT keypoint descriptor

Step 1: Scale-Space Extrema Detection

- Need to find “characteristic scale” for features
- Scale space representation:

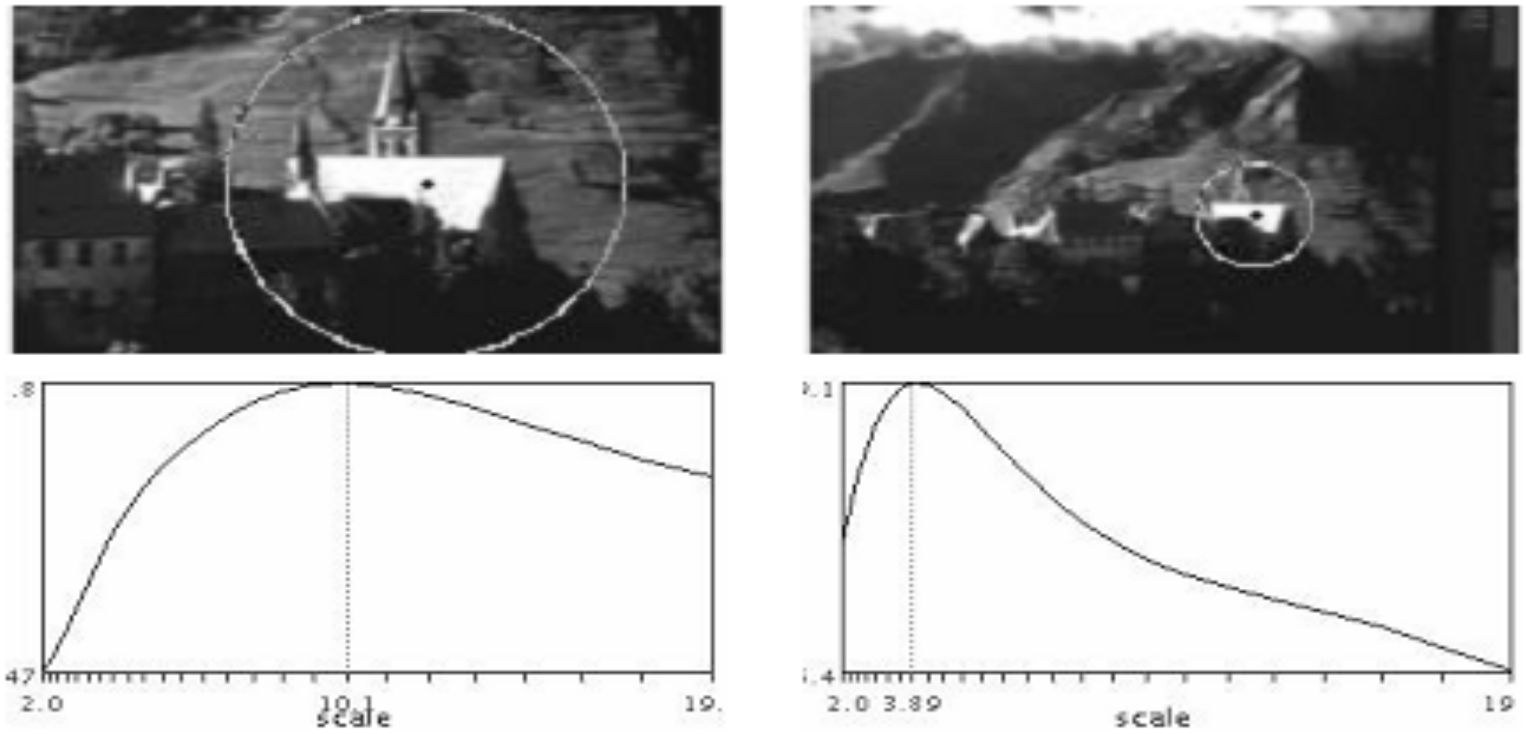


$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y)$$

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\left(x^2+y^2\right)/2\sigma^2}$$

Step 1: Scale-Space Extrema Detection

- Mikolajczyk (2002): Experimentally, extrema of LoG gives best notion of scale:



Step 1: Scale-Space Extrema Detection

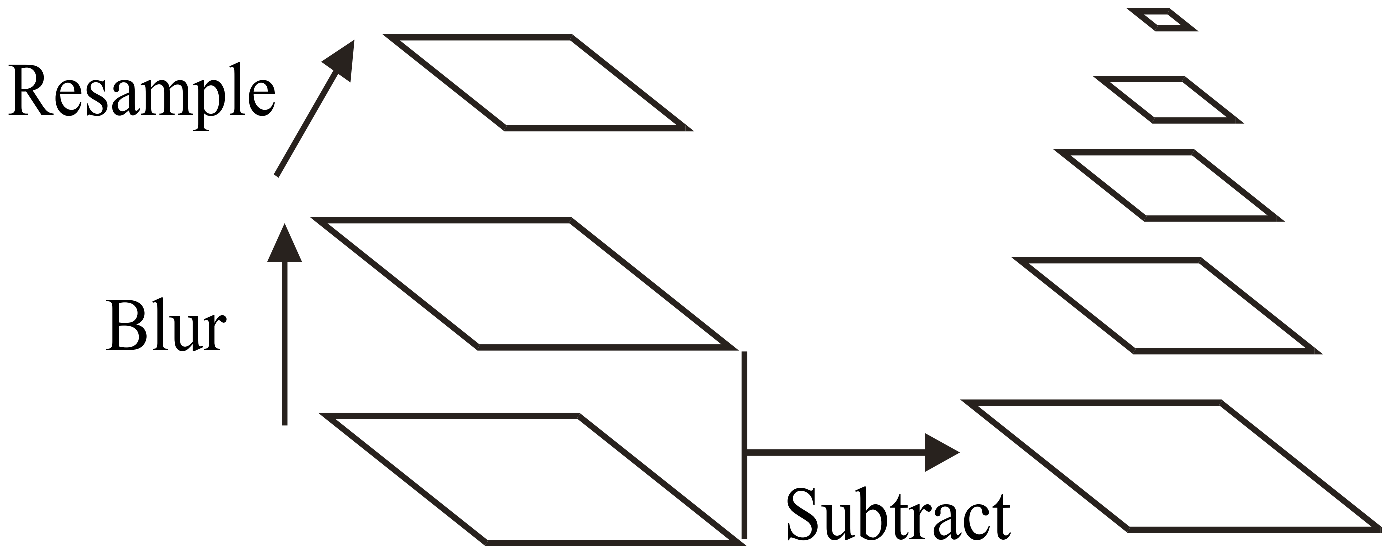
- LoG is computationally expensive
- Approximation: DoG

$$D(x,y,\sigma) = \left(G(x,y,k\sigma) - G(x,y,\sigma)\right) * I(x,y)$$
$$= L(x,y,k\sigma) - L(x,y,\sigma)$$

  - Smoothed images should be computed anyway => calculation is reduced to image subtraction.

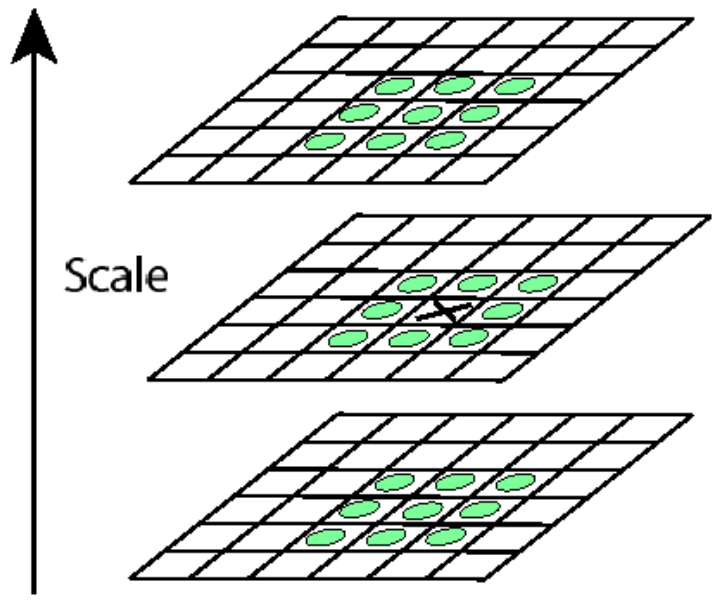
Step 1: Scale-Space Extrema Detection

- All scales must be examined to identify scale-invariant features
- This can be done efficiently using the compute the Difference of Gaussians (DOG) or Laplacian pyramid.



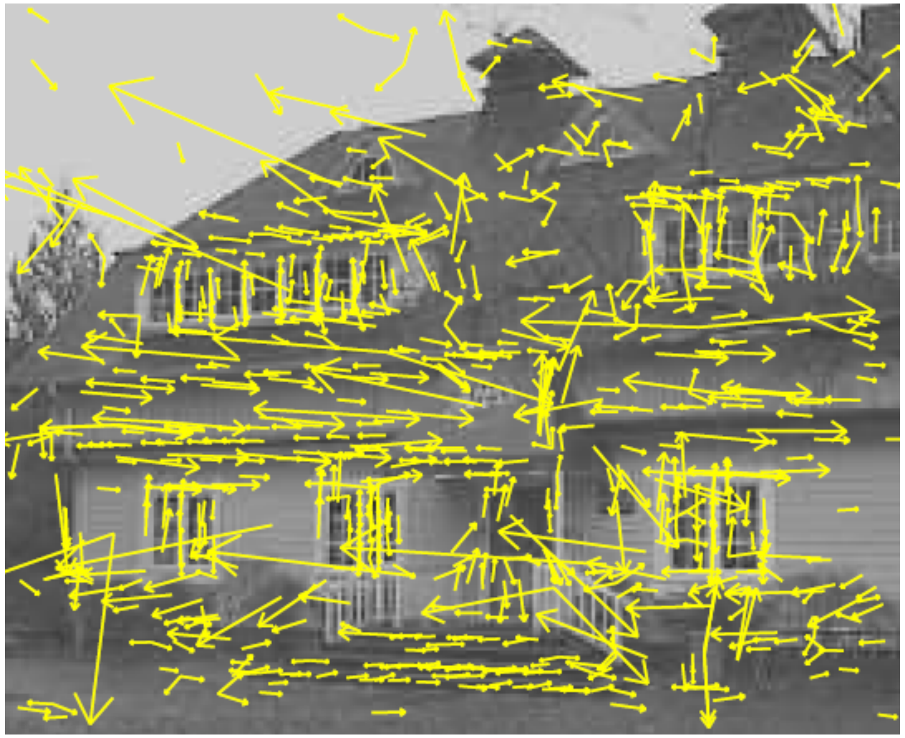
Step 1: Scale-Space Extrema Detection

- A candidate keypoint X is detected if it is a maximum or a minimum among all 26 neighbors in the difference-of-Gaussians scale space. Hence the name *Scale-Space Extrema.*



Keypoints after Scale-Space Extrema Detection

233x189 original image

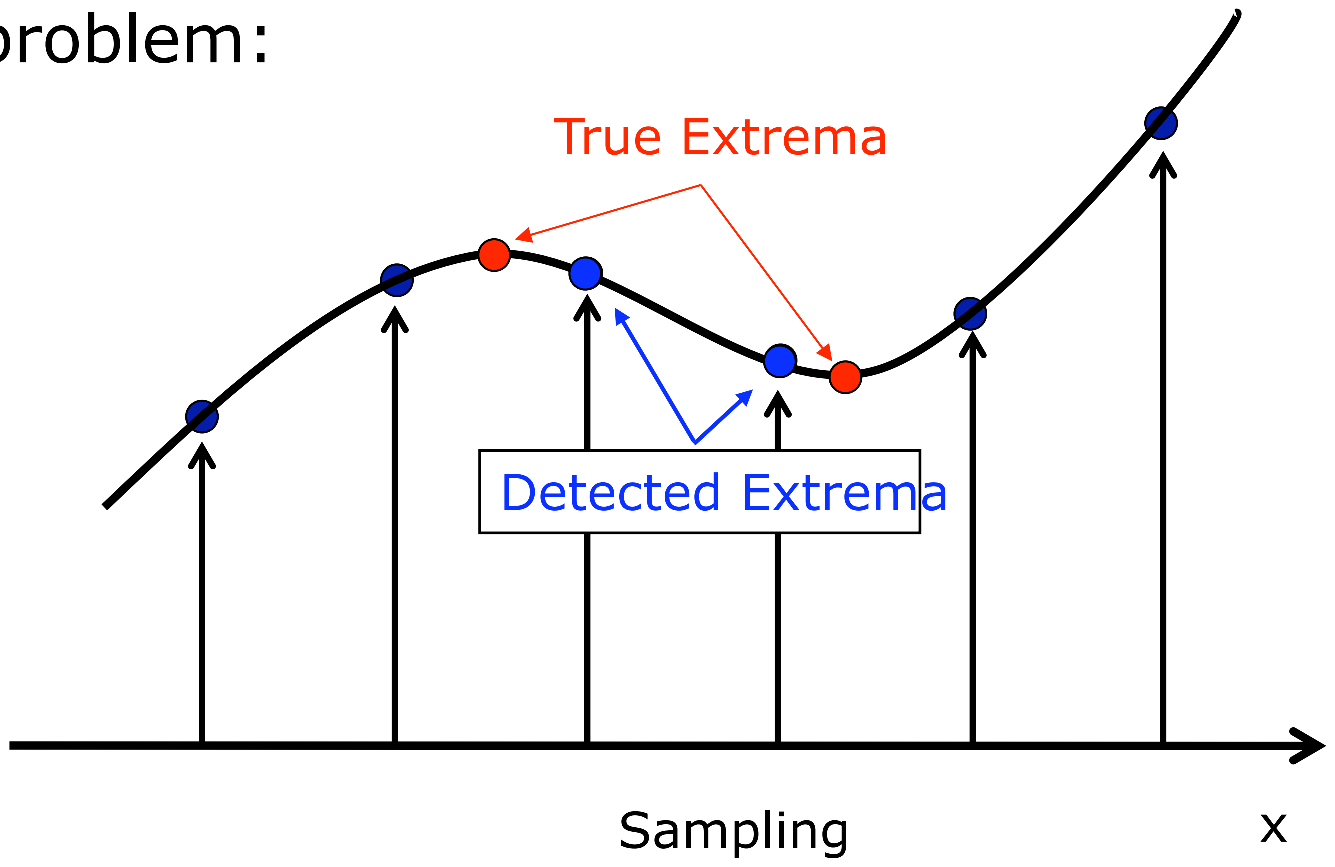

832 DOG extrema

SIFT Keypoint Computation

1. Scale-space extrema detection
2. Keypoint localization and filtering
3. Orientation assignment
4. Creation of SIFT keypoint descriptor

Step 2: Keypoint Localization

- The problem:



Step 2: Keypoint Localization

- The solution: Fit the keypoint at location $\vec{x}$ to nearby data using quadratic approximation.

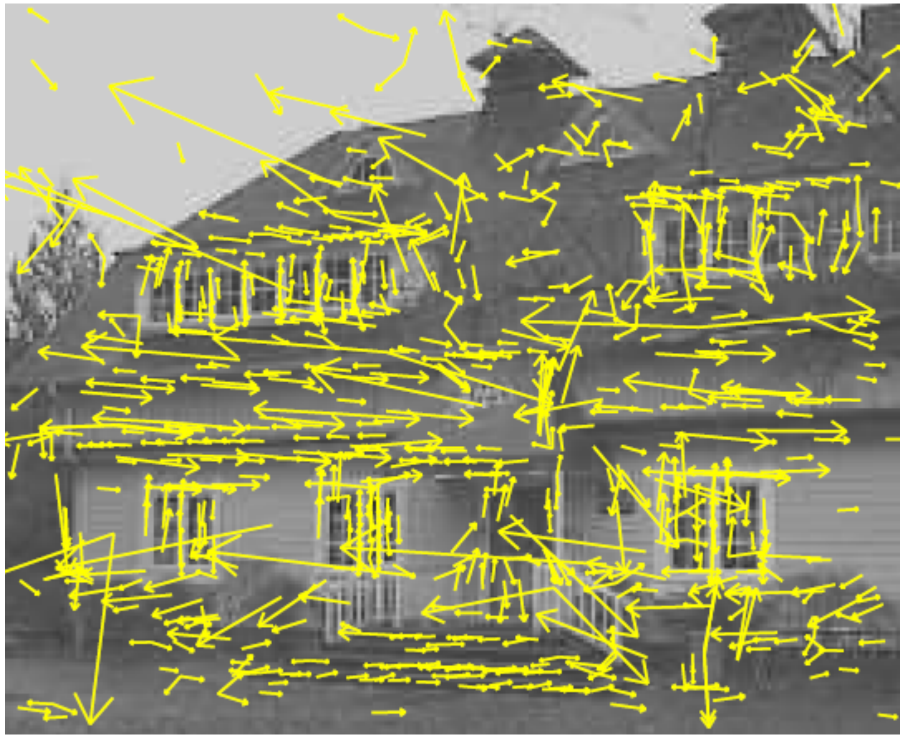
$$D(\vec{x}) = D + \frac{\partial D^T}{\partial \vec{x}} \vec{x} + \frac{1}{2} \vec{x}^T \frac{\partial^2 D^T}{\partial \vec{x}^2} \vec{x}$$

- Calculate the local extremum of the fitted function for an improved localization

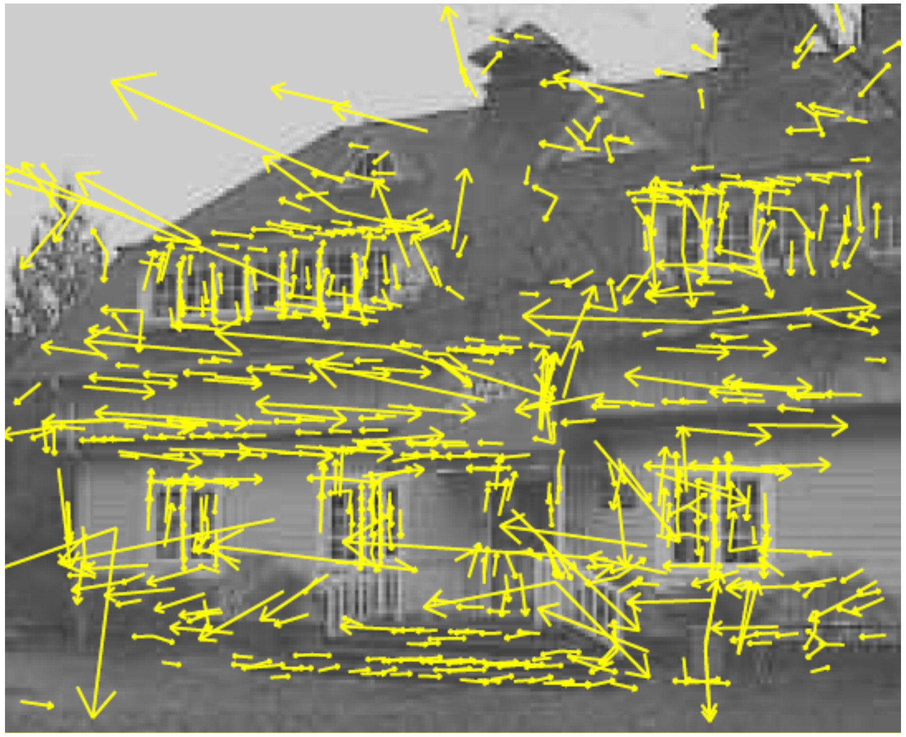
$$\hat{x} = -\frac{\partial^2 D}{\partial \vec{x}^2}^{-1} \frac{\partial D}{\partial \vec{x}}$$

- Discard points with low contrast, i.e. discard keypoinst with $D(\hat{x}) < 0.03, \quad D \in [0,1]$

Keypoints after Keypoint Localization



832 keypoints after DOG extrema

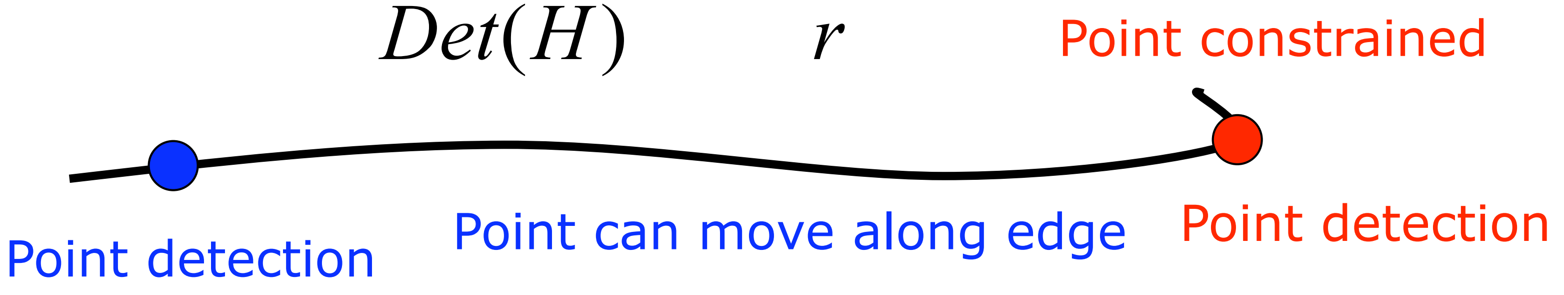


729 keypoints after localization

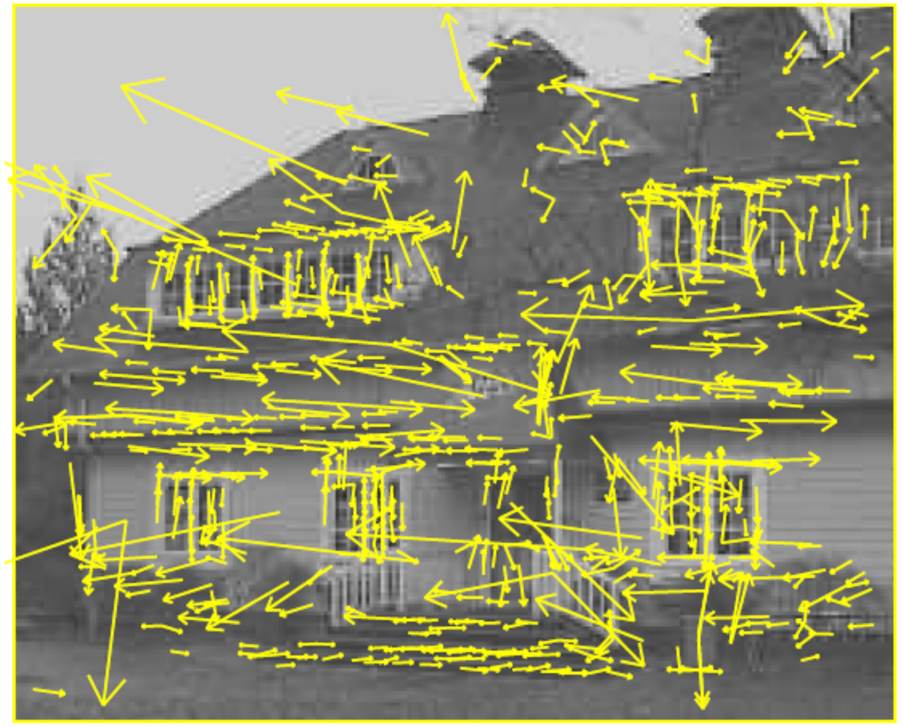
Step 2: Keypoint Filtering

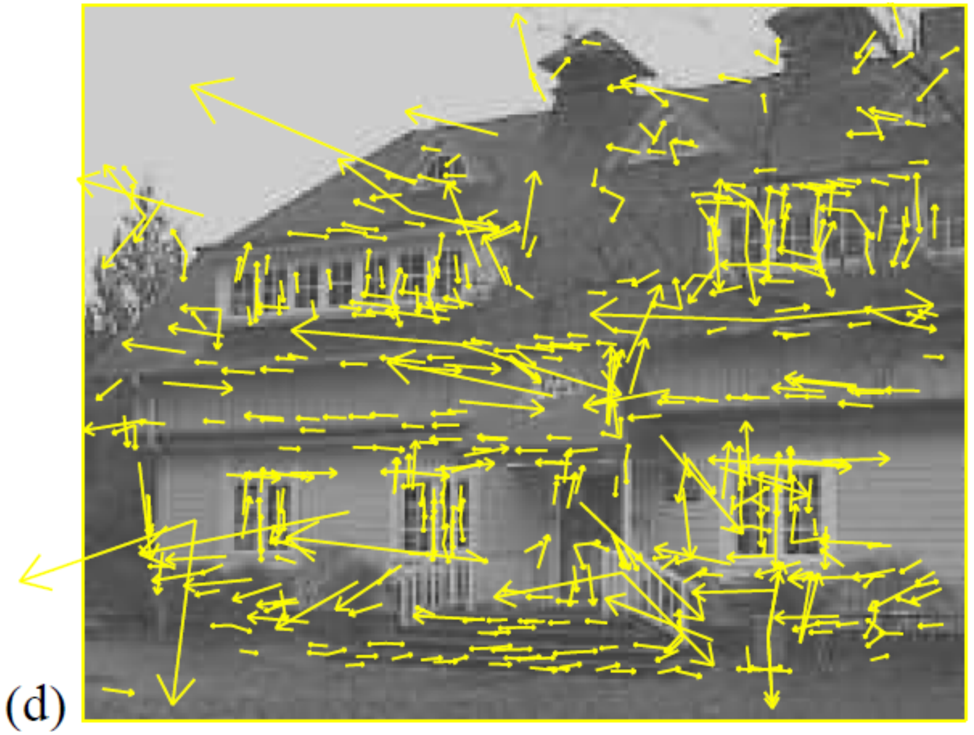
- Corners provide repeatable points for matching.
- In the region around a corner, the gradient has two or more different values.
- Use the Trace and the Determinant of the Hessian of the gradient to check if the ratio of the principal curvatures is below some threshold, r. Discard keypoints with high ratio values.

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}$$



Keypoints after Keypoint Localization/Filtering





729 keypoints after localization

536 keypoints after cornerness filtering

SIFT Keypoint Computation

1. Scale-space extrema detection
2. Keypoint localization and filtering
3. Orientation assignment
4. Creation of SIFT keypoint descriptor

Step 3: Orientation Assignment

- Now we have set of good points.
- We want a rotation and scale invariant representation of that keypoint.
- Choose a region around each point and explicitly encode rotation and scale information in the descriptor.

Step 3: Orientation Assignment

- The difference-of-Gaussians pyramid provides a representation at different scales:

$$L(x, y) = G(x, y, \sigma) * I(x, y)$$

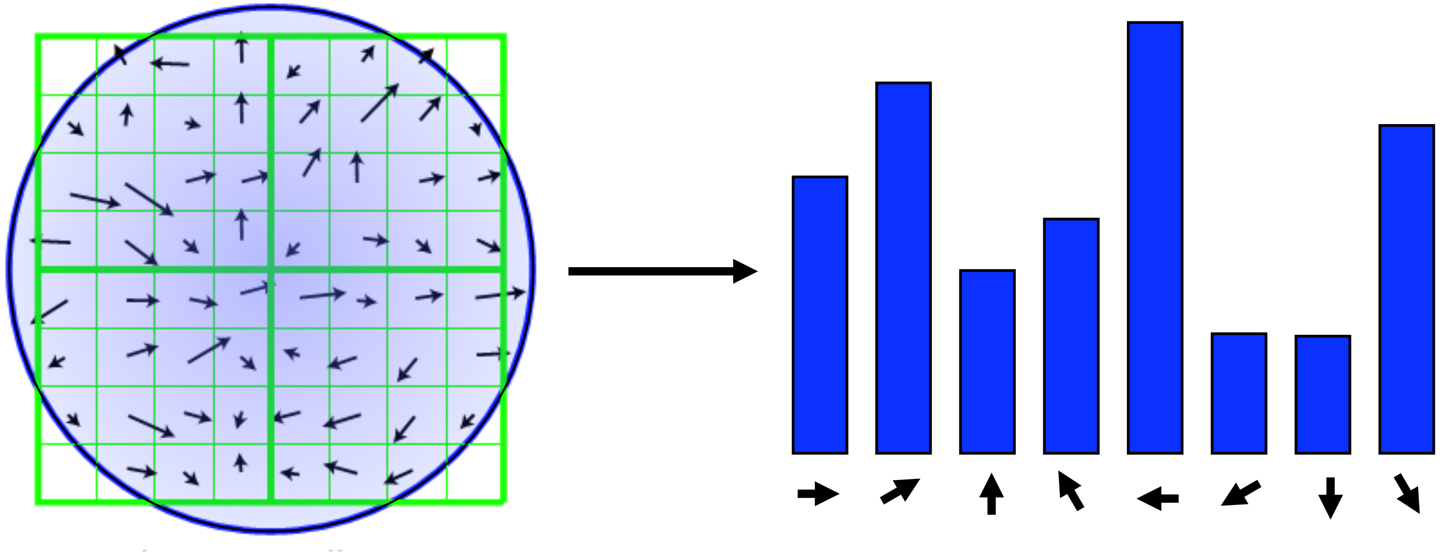
- Compute the magnitude and orientation of the gradient using finite differences:

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

$$\theta(x, y) = \tan^{-1}\left(\frac{(L(x, y+1) - L(x, y-1))}{(L(x+1, y) - L(x-1, y))}\right)$$

Step 3: Orientation Assignment

- Find dominant gradient orientation over a small image patch.
  - Create a histogram of local gradient directions computed at the selected scale.
  - The histogram is weighted by a superimposed Gaussian (magnitude and Gaussian window) where σ_o is 1.5 times that of the scale of a keypoint, $\sigma_o = 1.5\sigma$)

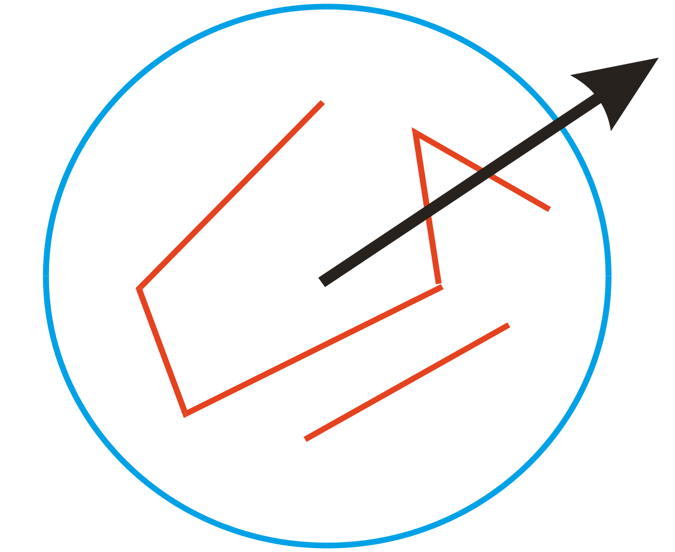


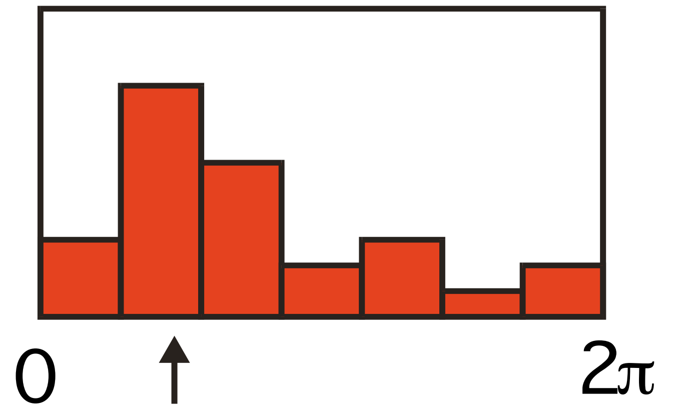
Step 3: Orientation Assignment

- Determining the peak of the histogram is very important because it denotes the prominent edge orientation in a region.
- Any peak within 80% of the highest peak is used in determining the dominant orientation.
- A parabola is fit to the 3 histogram values closest to each peak to interpolate the peak position for better accuracy

Step 3: Orientation Assignment

- The peak of the smoothed histogram is used for determining the canonical orientation.
- Each patch (over which the histogram of local gradients is computed) is rotated so that the dominant orientation points upwards.
- Each key specifies stable 2D coordinates (x, y, scale, dominant orientation).





SIFT Keypoint Computation

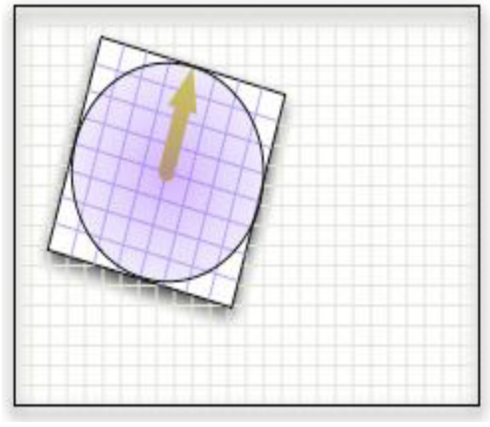
1. Scale-space extrema detection
2. Keypoint localization and filtering
3. Orientation assignment
4. Creation of SIFT keypoint descriptor

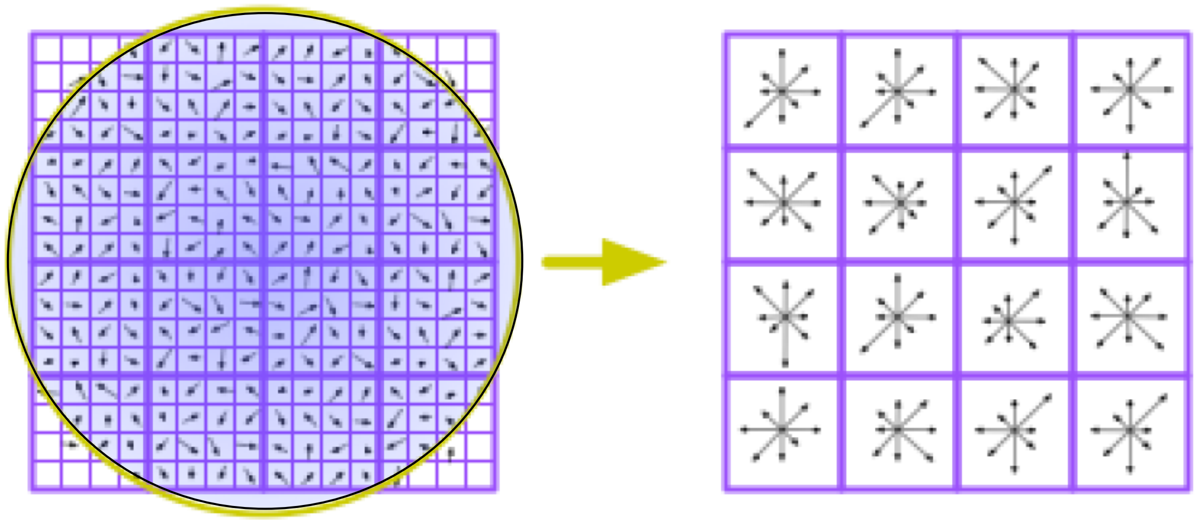
Step 4: Keypoint Descriptor

- Each keypoint so far has x, y, σ, m, θ values associated with it.
- These 5 parameters impose a repeatable local coordinate system which is invariant to scale and orientation.
- Now we need a *local* descriptor (i.e. a descriptor for the region around a keypoint) which is also invariant to:
  - Changes in illumination.
  - Changes in 3D viewpoint

Step 4: Keypoint Descriptor

- Compute a gradient histogram over a 4x4 pixel region (window) using 8 gradient directions
- Compute a weighted histogram for 4x4 windows
- Gaussian weighting around center (σ_k is 0.5 times that of the scale of a keypoint σ, $\sigma_k = 0.5\sigma$)
- 4x4x8 = 128 dimensional feature vector

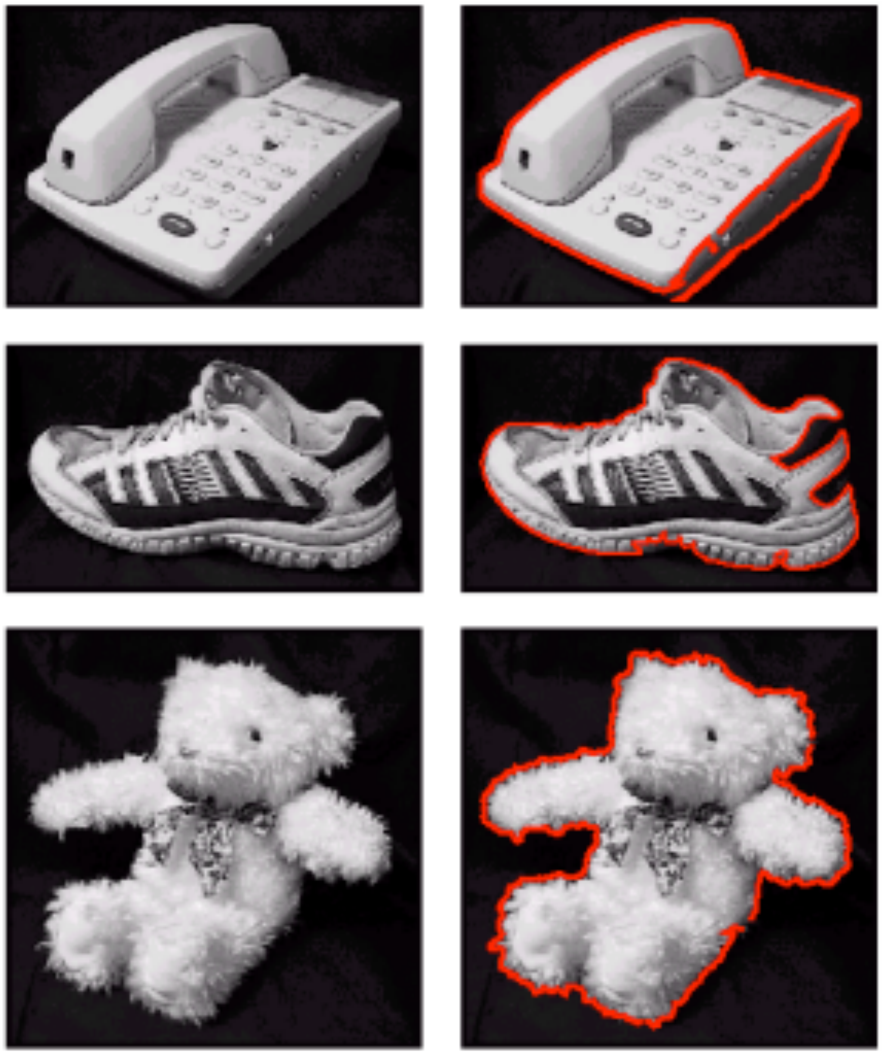




Step 4: Keypoint Descriptor

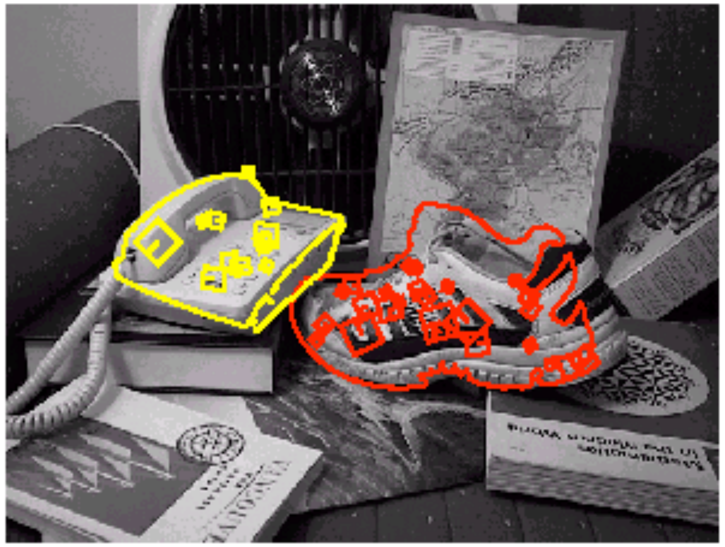
- Normalize the keypoint feature vector to unit length to increase invariance to affine illumination changes, like differences in gain.
- Non-linear illumination changes can occur occur due to camera saturation or due to illumination changes that affect 3D surfaces by different amounts.
- Still, camera saturation affects magnitudes much more than orientation.
- To improve robustness to variations in saturation, threshold all gradients to 0.2 (empirically determined) and renormalize.

Example: 3D Object Recognition



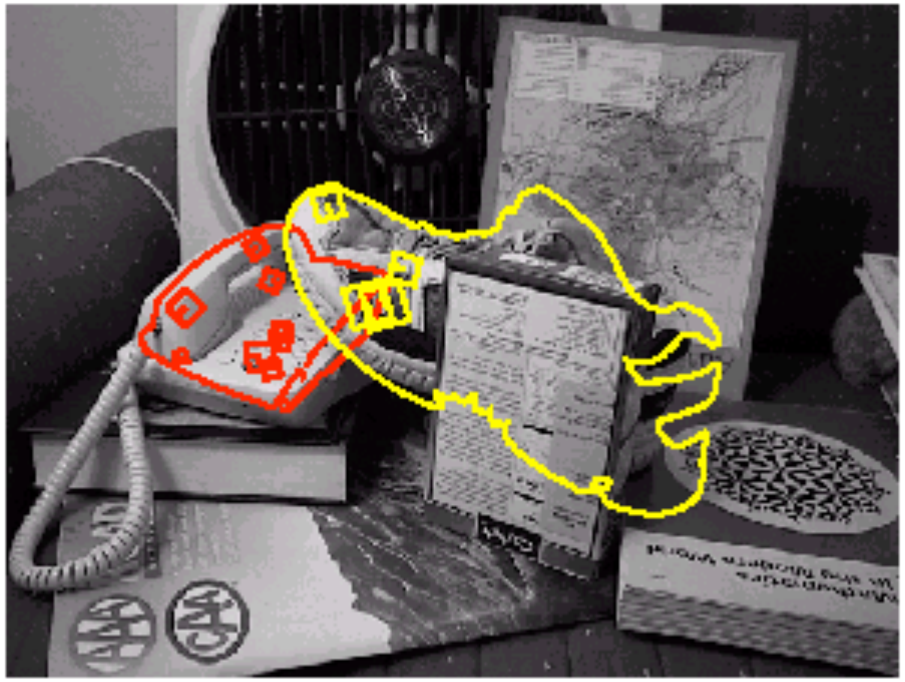
- Extract outlines with background subtraction
- Compute SIFT keypoints

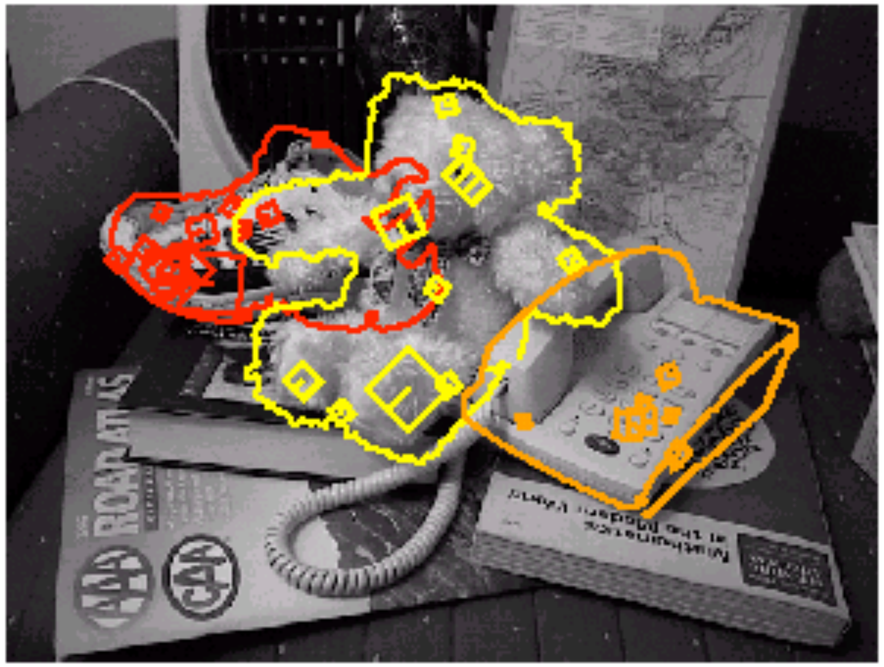
Example: 3D Object Recognition



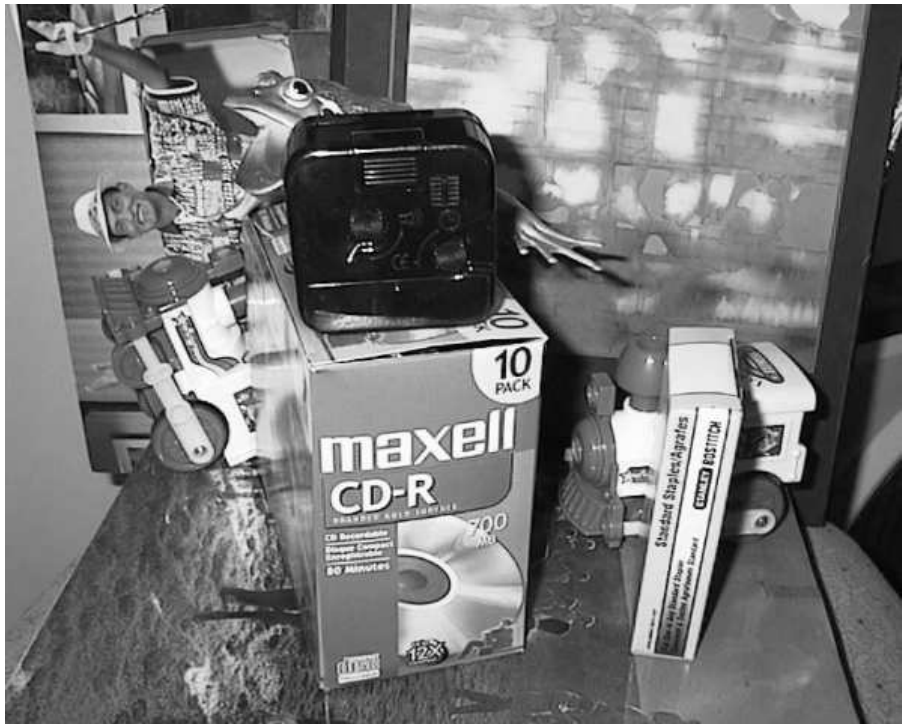
- Only 3 keys are needed for recognition.
- Extra keypoints provide robustness

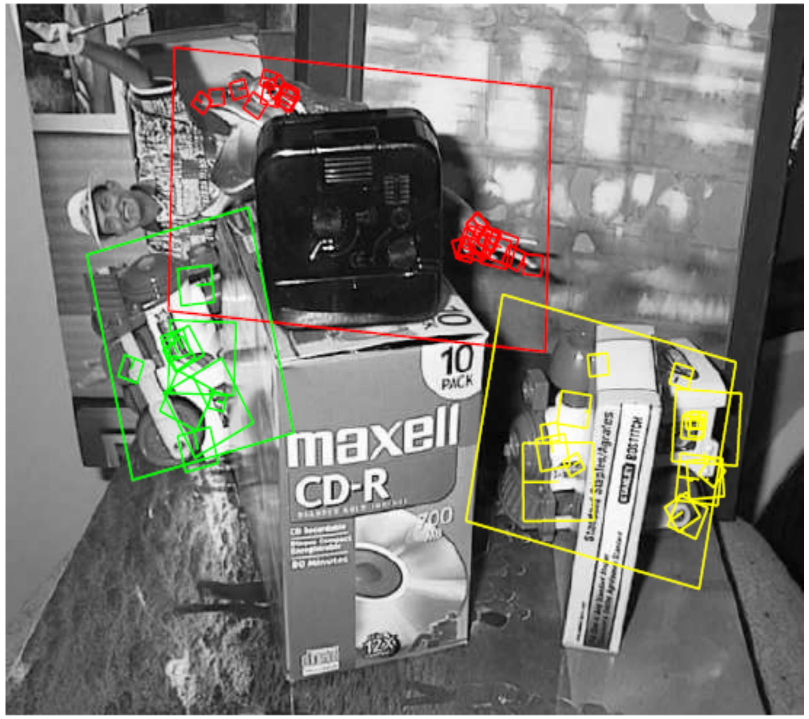
Recognition under occlusion



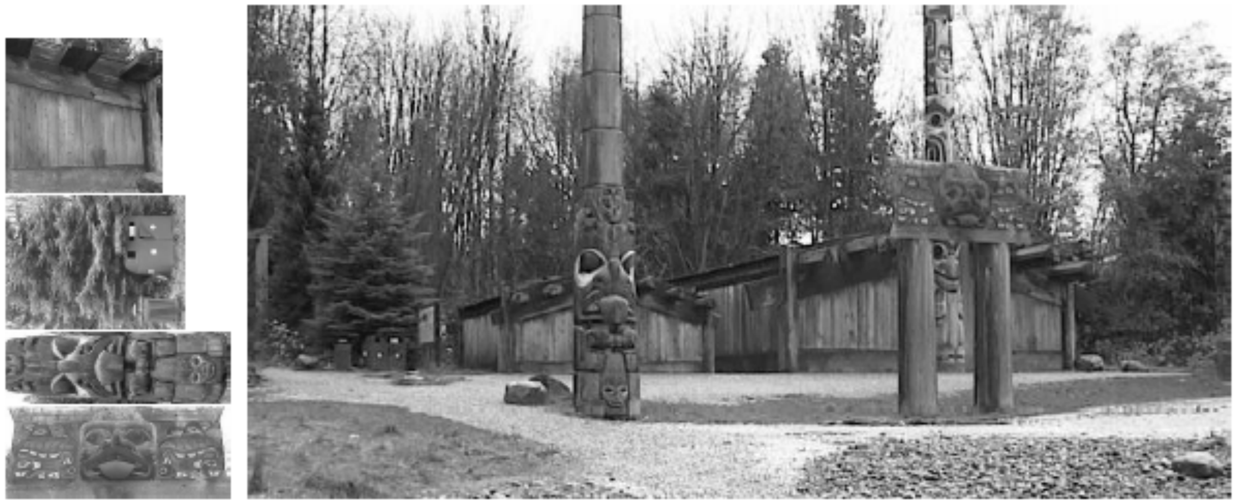


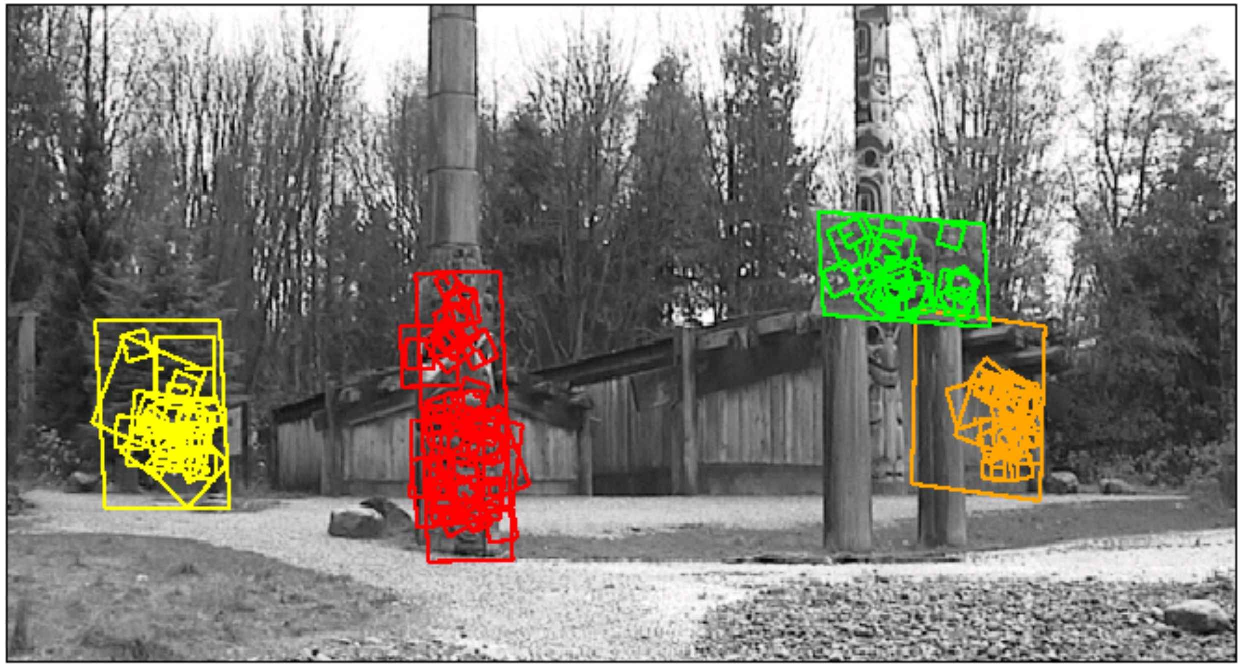
Example: Recognition under Occlusion





Example: Object Localization





Slide Sources

1. The slides by K. Dyagilev and A. Dominitz can be found at http://is.haifa.ac.il/~ishimshoni/MVG/SIFT_KA.ppt
2. The slides by R. Fergus can be found at http://cs.nyu.edu/~fergus/teaching/vision/2_Matching.ppt
3. The slides of D. Lee can be found at http://www.cs.cmu.edu/~efros/courses/AP06/presentations/0319.SIFT.ppt
4. The slides of O. Pele can be found at http://www.cse.huji.ac.il/course/2006/compvis/lectures/SIFT.ppt
5. The slides of S. Thurn are located at http://robots.stanford.edu/cs223b05/notes/CS%20223-B%20L4%20Features2.ppt