INF 5300 - 26.2.2014
Detecting good features for tracking

Anne Schistad Solberg

- Finding the correspondence between two images
  - **What are good features to match?**
    - Points?
    - Edges?
    - Lines?

Curriculum

- Chapter 4 in Szeliski, with a focus on 4.1 Point-based features
- Recommended additional reading on SIFT features:
  - Distinctive Image Features from Scale-Invariant Keypoints by D. Lowe, International Journal of Computer Vision, 20,2,pp.91-110, 2004.

Goal of this lecture

- Consider two images containing partly the the same objects but at different times or from different views.
- What type of features are best for recognizing similar object parts in different images?
- These features will later be used to find the match between the images.
- This chapter is also linked to chapter 6 which we will cover in a later lecture (April).
- This is useful for e.g.
  - Tracking an object in time
  - Mosaicking or stitching images
  - Constructing 3D models

Image matching

- How do we compute the correspondence between these images?
  - Extract good features for matching (this lecture)
  - Estimation geometrical operation for match (later lecture)

•by Diva Sian

•by swashford

What type of features are good?

•Point-like features?

•Region-based features?

•Edge-based features?

•Line-based features?

Point-based features

- Point-based features should represent a set of special locations in an image, e.g. landmarks or keypoints.
- Two main categories of methods:
  - Find points in an image that can be easily tracked, e.g. using correlation or least-squares matching.
    - Given one feature, track this feature in a local area in the next frame
    - Most useful when the motion is small
  - Find features in all images and match them based on local appearance.
    - Most useful for larger motion or stitching.

Four steps in feature matching

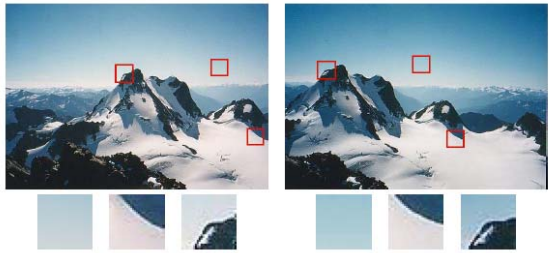
1. Feature extraction
   - Search for characteristic locations
2. Feature description
   - Select a suitable descriptor that is easy to match
3. Feature matching
   - Efficient search for matching candidates in other images
4. Feature tracking
   - Search a small neighborhood around the given location
     - An alternative to step 3.

Point-based features

- Point-based features should highlight landmarks or points of special characteristics in the image.
- They are normally used for establishing correspondence between image pairs.
- What kind of locations in these images do you think are useful?

Feature detection

- Goal: search the image for locations that are likely to be easy to match in a different image.



- What characterizes the regions? How unique is a location?
  - Texture?
  - Homogeneity?
  - Contrast?
  - Variance?

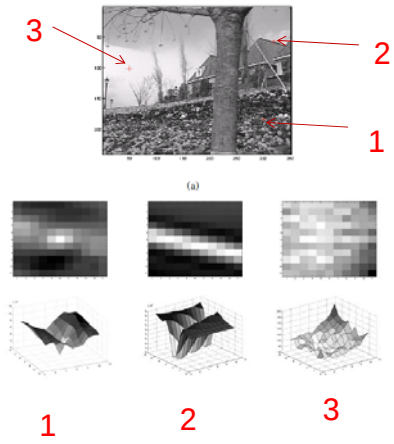
Feature detection

- A simple matching criterion: summed squared difference:

$$E_{WSSD}(u) = \sum_i w(x_i)\left[I_1(x_i+u) - I_0(x_i)\right]^2$$

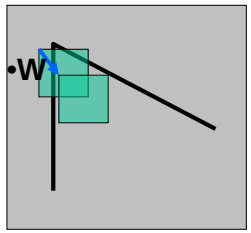
- I_0 and I_1 are the two images, $\mathbf{u}=(u,v)$ the displacement vector, and w(x) a spatially varying weight function.
- Check how stable a given location is (with a position change Δu) in the first image by computing the auto-correlation function:

$$E_{AC}(\Delta u) = \sum_i w(x_i)\left[I_0(x_i+\Delta u) - I_0(x_i)\right]^2$$



Feature detection: the math

- Consider shifting the window **W** by (u,v)
  - how do the pixels in **W** change?
  - Do a Taylor series expansion of the autocorrelation to allow fast computation:



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

where $\nabla I_0(x_i) = \left(\frac{\partial I_0}{\partial x}, \frac{\partial I_0}{\partial y}\right)(x_i)$ is the image gradient at x_i.

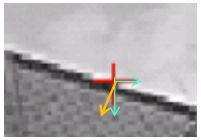
•Compute the gradients robustly using a Derivative of Gaussian filter

- The autocorrelation matrix A is:

$$A = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Feature detection: the math

- The matrix A carries information about the uncertainty of the location of a patch.
- A is called a tensor matrix and is formed by outer products of the gradients, convolved with a weighting function w to get a pixel-based uncertainty estimate.
- Eigenvector decomposition of A gives two eigenvalues, λ_0 and λ_1.
- The smallest eigenvalue carries information about the uncertainty.



•High gradient in the direction of maximal change

• If there is one dominant direction, we are quite certain about the direction estimate, and λ_{min} will be much smaller than λ_{max}.

•A high value of λ_{min} means that the gradient changes much in both directions, so this can be a good keypoint.

Feature detection: Harris corner detector

- Harris and Stephens (1988) proposed an alternative criterion computed from A (α=0.06 is often used):

$$\det(A) - \alpha \,\mathrm{trace}(A)^2 = \lambda_{\max}\lambda_{\min} - \alpha(\lambda_{\max} + \lambda_{\min})^2$$

- Other alternatives are e.g. the harmonic mean:

$$\frac{\det A}{\mathrm{trace}(A)} = \frac{\lambda_{\max}\lambda_{\min}}{\lambda_{\max} + \lambda_{\min}}$$

- The difference between these criteria is how the eigenvalues are blended together.

Feature detection algorithm

1. Compute the gradients I_x and I_y , and I_{xy} using a robust Derivative-of-Gaussian kernel (hint: convolve a Sobel x and y with a Gaussian).
2. Convolve these gradient images with a larger Gaussian to further robustify.
3. Create the matrix A from the robustified gradients from 2.
4. Compute either the smallest eigenvalue or the Harris corner detector measure from A.
5. Find local maxima above a certain threshold and report them as detected feature point locations.
6. Adaptive non-maximal suppression (ANMS) is often used to improve the distribution of feature points across the image.

Examples

Original

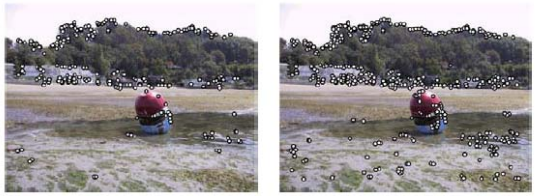
Largest eigenvalue

Smallest eigenvalue

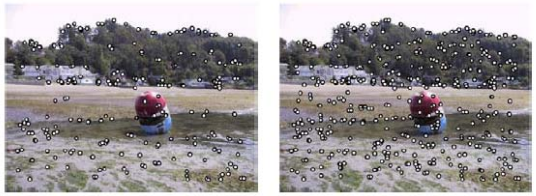
Harris operator

- We note that the first eigenvalue gives information about major edges.
- The second eigenvalue gives information about other features, like corners or other areas with conflicting directions.
- The Harris operator combines the eigenvalues.
- It is apparent that we need to threshold the images and find local maxima in a robust way.
  - How did we supress local minima in the Canny edge detector??

Comparing points detected with or without ANMS



(a) Strongest 250 (b) Strongest 500



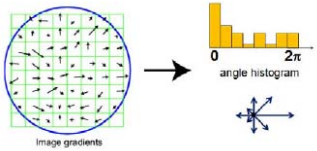
(c) ANMS 250, $r = 24$ (d) ANMS 500, $r = 16$

How do we get rotation invariance?

- Option 1: use rotation-invariant feature descriptors.
- Option 2: estimate the locally dominant orientation and create a rotated patch to compute features from.

How do we estimate the local orientation?

- The gradient direction is often noisy.
  - Many ways to robustify it:
- Direction from eigenvector of gradient tensor matrix
  - Filter the the gradients g_x, g_y and g_{xy} and form the gradient tensor matrix T. Compute the direction as the direction of the dominant eigenvector of T.
- Angle histogram
  - Group the gradient directions weighted by magnitude together into 36 bins.
  - Find all peaks with 80% of maximum (allowing more than one dominant direction at some locations).



How do we get scale invariance?

- These operators look at a fine scale, but we might need to match features at a broader scale.
- Solution 1:
  - Create a image pyramid and compute features at each level in the pyramid.
    - At which level in the pyramid should we do the matching on? Different scales might have different characteristic features.
- Solution 2:
  - Extract features that are stable both in location AND scale.
  - SIFT features (Lowe 2004) is the most popular approach of such features.

Scale-invariant features (SIFT)

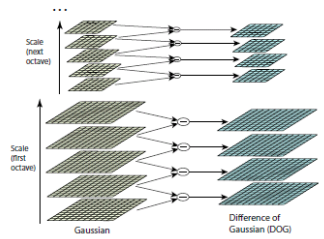
- See Distinctive Image Features from Scale-Invariant Keypoints by D. Lowe, International Journal of Computer Vision, 20,2,pp.91-110, 2004.
- Invariant to scale and rotation, and robust to many affine transforms.
- Main components:
  1. Scale-space extrema detection – search over all scales and locations.
  2. Keypoint localization – including determining the best scale.
  3. Orientation assignment – find dominant directions.
  4. Keypoint descriptor - local image gradients at the selected scale, transformed relative to local orientation.

SIFT: 1. Scale-space extrema

- A Gaussian filter is applied at different scales $L(x,y,\sigma) = G(x,y,\sigma) * I(x,y,\sigma)$.
- Compute keypoints in scale space by difference-of-Gaussian, where the difference is between two nearby scales separated by a constant k:

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

- This is an efficient approximation of a Laplacian of Gaussian, normalized to scale σ (see Lowe 2004).
- Detecting extrama in scale is based on sampling different scales.
- Extrema in space are also detected.



SIFT 1: extrema detection

- Consider a Taylor series expansion of the scale-space function $D(x,y,\sigma)$ around sample point x

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

- The location of the extreme is found by take the derivative of D(x) and setting it to zero:

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

- It is computed by differences of neighboring sample points, yielding a 3x3 linear system.
- The value of D at the extreme point is useful for suppressing extrema with low contrast, $|D| < 0.03$ are suppressed.

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

SIFT 1: eliminating edge response

- Since points on an edge are not very stable, so such points need to be eliminated.
- This is done using the curvature, computed from the Hessian matrix of D.

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

- The eigenvalues of H are proportation to principal curvatures of D. As with Harris, consider the ratio between the eigenvalues α and β. They found a good criteria to be to only keep the points where

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

- r=10 is often used.

SIFT 2: computing orientation

- To normalize for the orientation of the keypoints, we need to estimate the orientation.
- They used simple pixel differences to do this (L is a Gaussian smoothed image):

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$
$$\theta(x,y) = \tan^{-1}\left((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))\right)$$

- Then, they computed orientation histograms with 36 bins covering the 360 degrees of possible orientations.
- In this histogram, the highest peak, and other peaks with hight 80% of max are found. If a localization has multiple peaks, it can have more than 1 orientation.

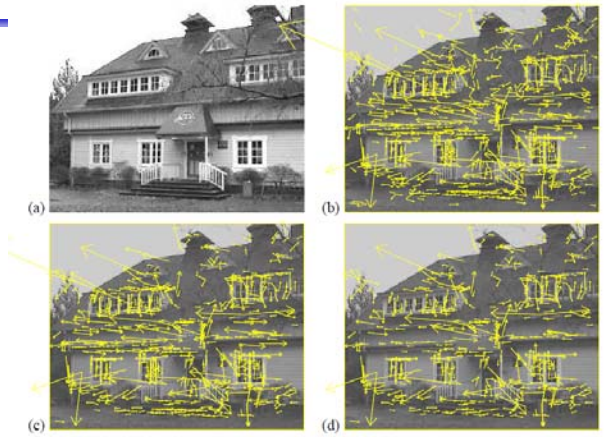


Figure 5: This figure shows the stages of keypoint selection. (a) The 233x189 pixel original image. (b) The initial 832 keypoints locations at maxima and minima of the difference-of-Gaussian function. Keypoints are displayed as vectors indicating scale, orientation, and location. (c) After applying a threshold on minimum contrast, 729 keypoints remain. (d) The final 536 keypoints that remain following an additional threshold on ratio of principal curvatures.

Feature descriptors

- Which features should we extract from the key points?
- These features will later be used for **matching** to establish the motion between two images.
- How is a good match computed (more in chapter 8)?
  - Sum of squared differences in a region?
  - Correlation?
- The local appearance of a feature will often change in orientation and scale (this should be utilized e.g. by extracting the local scale and orientation and then use this scale (or a coarser one) in the matching).

Normalizing bias and gain

- For simple motion, matching intensity patches can be useful.
- One technique is called Multi-Scale Oriented Patches (MOPS)
  - Detect a proper scale in a Gaussian pyramid, match at a coarser scale to avoid aliasing.
  - Estimate rotation and normalize it.
  - Extract a patch of 5x5 pixels relative to scale.
  - Rescale patch to zero mean and variance of 1.

Detection at different scales

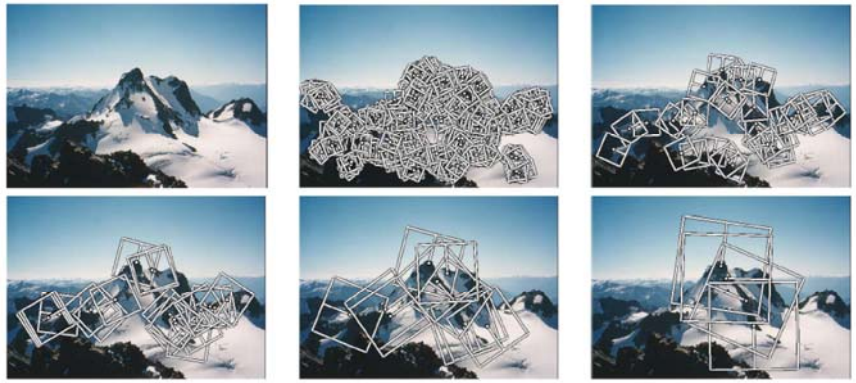
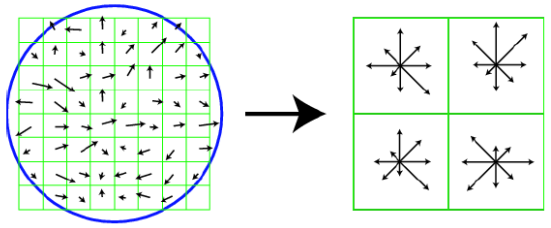


Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.

SIFT: feature extraction stage

- Select the level of the Gaussian pyramid where the keypoints were identified.
- Compute the gradient at each point in a 16x16 window around each keypoint. Weight the gradient values by a Gaussian function.
- Threshold gradient magnitude to throw out weak edges.
- Form a gradient orientation histogram for each 4x4 quadrant using 8 directional bins (using trilinear interpolation of the gradient magnitude to 2x2x2 bins).
- This results in 128 (16*8) non-negative values which are the raw SIFT-features.
- Further normalize the vector.

SIFT: feature extraction



(a) image gradients (b) keypoint descriptor

This example shows a 8x8 grid decomposed to 2x2
Original SIFT uses 16x16 decomposed to 4x4

Variations of SIFT

- PCA-SIFT: compute x- and y-gradients in a 39x39 patch, resulting in 3042 features. Use PCA to reduce this to 36 features.
- Gradient location-orientation histogram (GLOH): use a log-polar binning of gradient histograms, then PCA.
- Steerable filters: combinations of DoG-filters of edge- and corner-like filters.

Feature matching

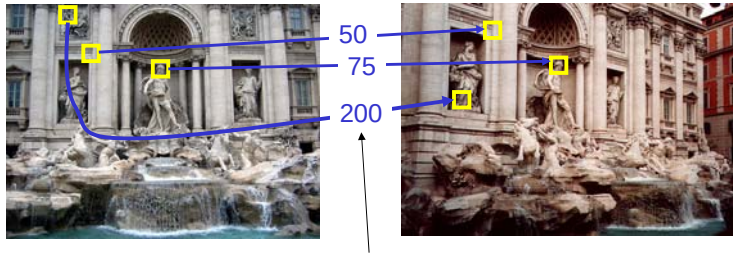
- Matching is divided into:
  - Define a matching strategy to compute the correspondence between two images.
  - Using efficient algorithms and data structures for fast matching (we will not go into details on this).
- Matching can be used in different settings:
  - Compute the correspondende between two partly overlapping images (= stitching).
    - Most key points are likely to find a match in the two images.
  - Match an object from a training data set with an unknown scene (e.g. for object detection).
    - Finding a match might be unlikely

Computing the match

- Assume that the features are normalize so we can measure distances using Euclidean distance.
- We have a list of keypoints features from the two images. Given a keypoint in image A, compute the similarity (=distance) between this point and all keypoints in image B.
- Set a threshold to the maximum allowed distance and compute matches according to this.
- Quantify the accuracy of matching in terms of:
  - TP: true positive: number of correct matches
  - FN: false negative: matches that were not correctly detected.
  - FP: false positive: proposed matches that are incorrect.
  - TN: true negative: non-matches that were correctly rejected.

Evaluating the results

How can we measure the performance of a feature matcher?



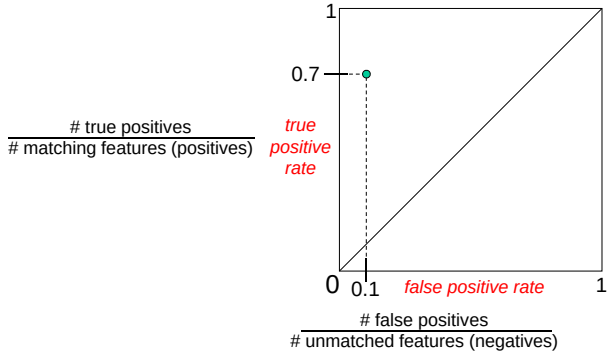
Performance ratios

- True positive rate (TPR)
  - TPR = TP/(TP+FN)
- False positive rate (FPR)
  - FPR = FP/(FP+TN)
- Positive predictive value (PPV)
  - PPV = TP/(TP+FP)
- Accuracy (ACC)
  - ACC = (TP+TN)/(TP+FN+FP+TN)

- Challenge: accuracy depends on the threshold for a correct match!

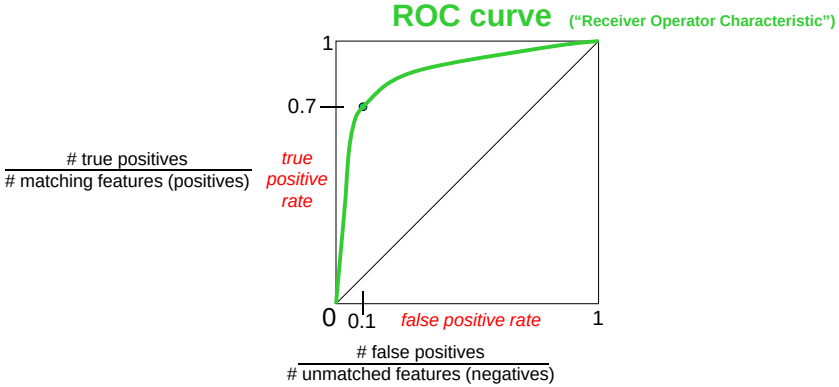
Evaluating the results

How can we measure the performance of a feature matcher?



Evaluating the results

How can we measure the performance of a feature matcher?



ROC Curves

- Generated by counting # current/incorrect matches, for different threholds
- Want to maximize area under the curve (AUC)
- Useful for comparing different feature matching methods

SIFT: feature matching

- Compute the distance from each keypoint in image A to the closest neighbor in image B.
- We need to discard matches if they are not good as not all keypoints will be found in both images.
- A good criteria is to compare the distance between the closest neighbor to the distance to the second-closest neighbor.
- A good match will have the closest neighbor should be much closer than the second-closest neighbor.
- Reject a point if closest-neighbor/second-closest-neighbor >0.8.

Feature tracking - introduction

- Feature tracking is a alternative to feature matching.
- Idea: detect features in image 1, then track each of these features in image 2.
- This is often used in video applications where the motion is assumed to be small.
- Is the motion assumed small:
  - Can the grey levels change? Use e.g. cross-correlation as a similarity measure.
- Large motion:
  - Can appearance changes happen?
- More on this in a later lecture.

Edge-based features

- Edge-based features can be more useful than point-based features in 3D or e.g. when we have occlusion.
- Edge-points often need to be grouped into curves or countours.
- An edge is considered an area with rapid intensity variation.
- Consider a gray-level image as a 3D landscape where the gray level is the height.
  Areas with high gradient are areas with steep slopes, computed by the gradient

$$J(x) = \nabla I(x) = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)(x)$$

- J will point in the direction of the steepest ascent.
- Taking the derivative is prone to noise, so we normally apply smoothing first/or in combination by combining the edge detector with a Gaussian.

Edge detection using Gaussian filters

- Gradient of a smoothed image:

$$J_\sigma(x) = \nabla\left|G_\sigma(x) * I(x)\right| = \nabla G_\sigma(x) * I(x)$$

- Derivative of Gaussian filter:

$$\nabla G_\sigma(x,y) = \left(\frac{\partial G_\sigma}{\partial x}, \frac{\partial G_\sigma}{\partial y}\right)(x) = \left[-x \quad -y\right]\frac{1}{\sigma^3}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

- Remember that the second derivative (Laplacian) carries information about the exact location of the edge:

$$S_\sigma(x) = \nabla J_\sigma(x) = \left|\nabla^2 G_\sigma * I\right|$$

$$\nabla^2 G_\sigma = \frac{1}{\sigma^3}\left(2 - \frac{x^2+y^2}{2\sigma^2}\right)\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

- The edge locations are locations where the Laplacian changes sign (called zero crossing).
- Edge pixels can then be linked together based on both magnitude and direction.

Scale selection in edge detection

- σ is the scale parameter.
- It should be determined based on noise characteristics of the image, but also knowledge about the average object size in the image.
- A multi-scale approach is often used.

- After edge detection, we can apply all methods for robust boundary representation from INF 4300 to describe the contour. They can be normalized to handle different types of invariance.

Line detection

- Lines are normally detected using the Hough-transform (INF 4300).
- We will look at an alterative, RANSAC-based line detection, in chapter 6.

Vanishing points

- The structure in the image can often be found based on analyzing the vanishing points of lines.
- Lines that are parallell in 3D have the same vanishing point.

- Vanishing points can be found from the Hough transform (one of many different algorithms).