3. Feature Extraction and Matching

Corner Detection, SIFT

Why do we need features?

Finding same things across images: for reconstruction/tracking/generating panaromas …

What to do with features?

1. Use a detector to detect same scene points independently in both images and find a set of distinctive keypoints
2. Extract and normalize the region content
3. Define a region around each keypoint
4. Compute local descriptor from normalized region
5. Find correspondences by matching local descriptors

What makes a good feature?

What makes a good feature?

Region extraction needs to be repeatable and accurate

Invariant to translation, rotation, scale changes

Robust or covariant to out-of-plane (affine) transformations

Robust to lighting variations, noise, blur, quantization

- Locality: Features are local, therefore robust to occlusion and clutter
- Quantity: We need sufficient regions to cover the object
- Distinctiveness: The regions should contain “interesting” structure
- Efficiency: Close to real-time performance

What makes a good feature?

<u>Pixel Intensity</u>

Pros: can get many, slight invariance to everting

Cons: Non-immune to photometry changes, noise, any strong geometric variation (even scale or rotation)

What makes a good feature?

<u>Geometry</u>

Pros: Can handle photometric changes, Invariances to some transformations, resistance to noise

Cons: Computationally expensive

Features from Accelerated Segment Test (FAST)

Look for a 3 neighborhood circle (Bresenhem/mid point circle) around a point = 16 points

If a minimum of 12 points have intensity higher or smaller than the point, it's a good feature

Good for detecting corners

Features from Accelerated Segment Test (FAST)

Problems:

1. Slow (make fast by looking at pixels 1,5,9,13 or use machine learning)
2. Important to decide thresholds
3. Shadows/ small textual change can impact negatively

Features from Accelerated Segment Test (FAST)

Problems:

1. Slow (make fast by looking at pixels 1,5,9,13 or use machine learning)
2. Important to decide thresholds
3. Shadows/ small textual change can impact negatively
4. Photometric variations will impact it negatively

Harris Corners

- Localise the point through a small window
- Define precise location by shifting a window in any direction such that large change in pixels intensities are observed

Harris Corners

- Localise the point through a small window
- Define precise location by shifting a window in any direction such that large change in pixels intensities are observed

"flat" region:
no change in all directions

"edge":
no change along the edge direction

"corner":
significant change in all directions

Harris Corners

Window-averaged squared change of intensity induced by shifting the image data by [u,v]:

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

Window function

Shifted intensity

Intensity

Window function $w(x,y)$ =

1 in window, 0 outside

or

Gaussian

Darya Frolova, Denis Simakov The Weizmann Institute of Science
http://www.wisdom.weizmann.ac.il/~deniss/vision_spring04/files/InvariantFeatures.ppt

Harris Corners

$$E(u,v) \approx \sum_{x,y} w(x,y)[I(x,y) + uI_x + vI_y - I(x,y)]^2$$

$$= \sum_{x,y} w(x,y)[uI_x + vI_y]^2$$

$$= (u \quad v) \boxed{\sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}} \begin{pmatrix} u \\ v \end{pmatrix}$$

= M (structure tensor)

Its eigenanalysis reveals interesting things

Harris Corners

Intensity change in shifting window: eigenvalue analysis

$$E(u,v) \cong [u,v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

λ_1, λ_2 – eigenvalues of M

Ellipse E(u,v) = const

Iso-contour of the squared error, E(u,v)

http://www.wisdom.weizmann.ac.il/~deniss/vision_spring04/files/InvariantFeatures.ppt

Harris Corners

Classification of image points using eigenvalues of M:

Darya Frolova, Denis Simakov The Weizmann Institute of Science

Instead of computing eigenvalues:
R = det(M) – a trace(M)^2

a = [0.04, 0.06]

Corner: R>0
Edge: R<0
Flat: abs(R) is small

Harris Corners

Invariant to rotation, translation, photometric intensity changes.
Scale? No.

Scale Invariant Detection

- Consider regions (e.g. circles) of different sizes around a point
- Regions of corresponding sizes will look the same in both images

Scale Invariant Detection

- Consider regions (e.g. circles) of different sizes around a point
- Regions of corresponding sizes will look the same in both images

How to choose these regions automatically?

Scale Invariant Detection

- Design a function on the region (circle), which is “scale invariant” (the same for corresponding regions, even if they are at different scales)
- Image intensity (invariant to scale, of course):For a point in one image, we can consider it as a function of region size (circle radius)

Scale Invariant Detection

- Take a local maximum of image intensity
- Region size, for which the maximum is achieved, is normally covariant with image scale
- Find these regions independently in each image

Scale Invariant Detection

- Take a local maximum of image intensity
- Region size, for which the maximum is achieved, is normally covariant with image scale
- Find these regions independently in each image

Scale Invariant Detection

- A good function should have 1 stable peak
- Should react to sharp intensity changes

Scale Invariant Detection

- Remember edge detection

f

$\frac{d}{dx} g$

$f * \frac{d}{dx} g$

Scale Invariant Detection

- Remember edge detection

f

$\frac{d^2}{dx^2} g$

$f * \frac{d^2}{dx^2} g$

Edge

Second derivative of Gaussian (Laplacian)

Edge = zero crossing of second derivative

Scale Invariant Detection

- Laplacians are good candidate, remember edge detection

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

Scale Invariant Detection

- Characteristic scale: scale that produces a peak

Scale Invariant Detection

- Interest points are local maximas in both position and scale
  - **Functions for determining scale** $f = \text{Kernel} * \text{Image}$

Kernels:

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

(Laplacian)

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

(Difference of Gaussians)

where Gaussian

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

Note: both kernels are invariant to *scale* and *rotation*

Scale Invariant Detection

- Interest points are local maximas in both position and scale

Scale Invariant Detection

- **Harris-Laplacian**[1]
  Find local maximum of:
  - Harris corner detector in space (image coordinates)
  - Laplacian in scale

- **SIFT (Lowe)**[2]
  Find local maximum of:
  - Difference of Gaussians in space and scale

[1] K.Mikolajczyk, C.Schmid. "Indexing Based on Scale Invariant Interest Points". ICCV 2001
[2] D.Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". Accepted to IJCV 2004

Scale Invariant Detection

- Experimental evaluation of detectors w.r.t. scale change

Repeatability rate:

$$\frac{\#\ \text{correspondences}}{\#\ \text{possible correspondences}}$$

Scale Invariant Detection

- Given: two images of the same scene with a large scale difference between them
- Goal: find the same interest points independently in each image
- Solution: search for maxima of suitable functions in scale and in space (over the image)

Affine Invariant Detection

- **Above we considered:**
  Similarity transform (rotation + uniform scale)

- Now we go on to:
  Affine transform (rotation + non-uniform scale)

Affine Invariant Detection

- **Take a local intensity extremum as initial point**
- **Go along every ray starting from this point and stop when extremum of function f is reached**

$$f(t) = \frac{|I(t) - I_0|}{\frac{1}{t}\int_o^t |I(t) - I_0|\,dt}$$

- We will obtain approximately corresponding regions

Remark: we search for scale in every direction

T.Tuytelaars, L.V.Gool. "Wide Baseline Stereo Matching Based on Local, Affinely Invariant Regions", BMVC 2000.

Affine Invariant Detection

- **Algorithm summary (detection of affine invariant region):**

  Start from a *local intensity extremum* point

  Go in *every direction* until the point of extremum of some function f

  Curve connecting the points is the region boundary

  Compute *geometric moments* of orders up to 2 for this region

  Replace the region with *ellipse*

Affine Invariant Detection

- **The regions found may not exactly correspond, so we approximate them with ellipses**

- Geometric Moments:

$$m_{pq} = \int_{\square^2} x^p y^q f(x,y)\,dxdy$$

Fact: moments m_{pq} uniquely determine the function f

Taking f to be the characteristic function of a region (1 inside, 0 outside), moments of orders up to 2 allow to approximate the region by an ellipse

This ellipse will have the same moments of orders up to 2 as the original region

Affine Invariant Detection

- Covariance matrix of region points defines an ellipse:

$$p^T \Sigma_1^{-1} p = 1$$

$$\Sigma_1 = \langle pp^T \rangle_{\text{region 1}}$$

$$q^T \Sigma_2^{-1} q = 1$$

$$\Sigma_2 = \langle qq^T \rangle_{\text{region 2}}$$

($p = [x, y]^T$ is relative to the center of mass)

$$\Sigma_2 = A \Sigma_1 A^T$$

Ellipses, computed for corresponding regions, also correspond!

Affine Invariant Detection

- **Under affine transformation, we do not know in advance shapes of the corresponding regions**
- **Ellipse given by geometric covariance matrix of a region robustly approximates this region**
- **For corresponding regions ellipses also correspond.**

Methods:

1. Search for extremum along rays [Tuytelaars, Van Gool]:
2. Maximally Stable Extremal Regions [Matas et.al.]

What to do with features?

1. Use a detector to detect same scene points independently in both images and find a set of distinctive keypoints
2. Extract and normalize the region content
3. Define a region around each keypoint
4. Compute local descriptor from normalized region
5. Find correspondences by matching local descriptors

Once the detector has been decided, we need to look for a descriptor, so that the detected points can be represented distinctively and matched properly

Patches as Descriptor

- The simplest way to describe the neighborhood around an interest point is to write down the list of intensities to form a feature vector. But this is very sensitive to even small shifts, rotations.

Descriptors Invariant to Rotation

- **Harris corner response measure:**
  depends only on the eigenvalues of the matrix *M*

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

- **Image moments in polar coordinates**

$$m_{kl} = \iint r^k e^{-i\theta l} I(r,\theta) dr d\theta$$

Rotation in polar coordinates is translation of the angle:

$$\theta \rightarrow \theta + \theta_0$$

This transformation changes only the phase of the moments, but not its magnitude

Rotation invariant descriptor consists of magnitudes of moments: $|m_{kl}|$

Matching is done by comparing vectors $[|m_{kl}|]_{k,l}$

Descriptors Invariant to Rotation

- **Find local orientation**

Dominant direction of gradient

- Compute image derivatives relative to this orientation

Descriptors Invariant to Scale

- **Use the scale determined by detector to compute descriptor in a normalized frame**

For example:

- moments integrated over an adapted window
- derivatives adapted to scale: sI_x

Descriptors Invariant to Affine transformation

- **Affine invariant color moments**

$$m_{pq}^{abc} = \int_{region} x^p y^q R^a(x,y) G^b(x,y) B^c(x,y) dxdy$$

Different combinations of these moments are fully affine invariant

Also invariant to affine transformation of intensity $I \rightarrow a\,I + b$

- **Find affine normalized frame**

- Compute rotational invariant descriptor in this normalized frame

Scale Invariant Feature Transform- SIFT Descriptor

- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters

Advantages of Invariant Local Features

- Locality: features are local, so robust to occlusion and clutter (no prior segmentation)
- Distinctiveness: individual features can be matched to a large database of objects
- Quantity: many features can be generated for even small objects
- Efficiency: close to real-time performance
- Extensibility: can easily be extended to wide range of differing feature types, with each adding robustness

Scale Invariance

- Requires a method to repeatably select points in location and scale
- The only reasonable scale-space kernel is a Gaussian (Koenderink, 1984; Lindeberg, 1994)
- An efficient choice is to detect peaks in the difference of Gaussian pyramid (Burt & Adelson, 1983; Crowley & Parker, 1984 – but examining more scales)
- Difference-of-Gaussian with constant ratio of scales is a close approximation to Lindeberg's scale-normalized Laplacian (can be shown from the heat diffusion equation)

Keypoint Detection

(a) 233x189 image
(b) 832 DOG extrema
(c) 729 left after peak value threshold
(d) 536 left after testing ratio of principle curvatures

Vectors indicate scale, orientation and location.

SIFT Descriptor

- Use the blurred image associated with the keypoint's scale
- Take image gradients over the keypoint neighborhood.
- To become rotation invariant, rotate the gradient directions AND locations by (-keypoint orientation)

Now we've cancelled out rotation and have gradients expressed at locations relative to keypoint orientation θ

SIFT Descriptor

- Don’t use precise locations. Use blobs
- Divide into 4X4 regions and create histograms
- Put the rotated gradients into their local orientation histograms (8 bins)

SIFT Descriptor

- 8 orientation bins per histogram, and a 4x4 histogram array, yields 8 x 4x4 = 128 numbers.
- Find correspondences by minimizing Euclidean Distances

SIFT Descriptor

- 8 orientation bins per histogram, and a 4x4 histogram array, yields 8 x 4x4 = 128 numbers.
- Find correspondences by minimizing Euclidean Distances

Affine change upto 20 degrees is ok

Resources

- Fei-Fei Li's lectures (University of Standford)
- Alex Berg's lecture (MIT-CSAIL)

http://alumni.media.mit.edu/~maov/classes/comp_photo_vision08f/lect/18_feature_detectors.pdf