Image features

- Low-level features (corners, edges, textures patches) are needed for many tasks:
  - Image matching and registration
  - Structure-from-motion and image-based 3D modelling
  - Image segmentation
  - Object recognition
  - Image retrieval

~~[illegible]~~

Edge detection

- Goal: Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels
- Ideal: artist's line drawing

Why do we care about edges?

- Extract information, recognize objects
- Recover geometry and viewpoint

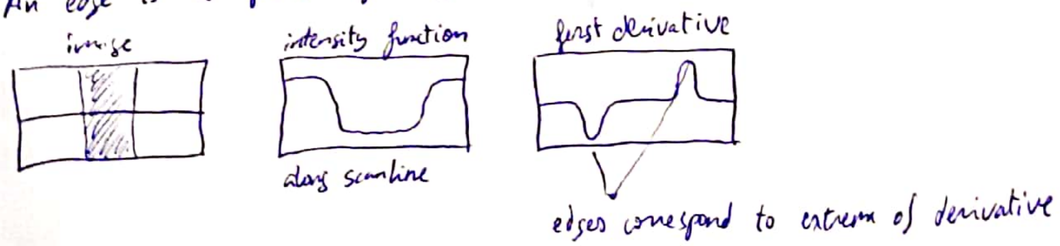
$\}$ robust against illumination changes

Origin of edges

Caused by:
- surface normal discontinuity
- depth discontinuity
- surface color discontinuity
- illumination discontinuity

Edge detection

- An edge is a place of rapid change in the image intensity function



Derivative with convolution

For 2D $f(x,y)$: $\dfrac{\partial f(x,y)}{\partial x} = \lim\limits_{\varepsilon \to 0} \dfrac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$

For discrete data, infinite differences: $\dfrac{\partial f(x,y)}{\partial x} \approx \dfrac{f(x+1, y) - f(x,y)}{1}$

To implement this as convolution, this is the associated filter:

$$\begin{matrix} 0 & 0 & 0 \\ 0 & \boxed{-1} & \boxed{1} \\ 0 & 0 & 0 \end{matrix} \qquad\qquad \begin{matrix} \boxed{1} \\ \boxed{-1} \end{matrix}$$

$$\frac{\partial f(x,y)}{\partial x} \qquad\qquad \frac{\partial f(x,y)}{\partial y}$$

Finite difference filters

Prewitt : $M_x = \begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix}$ $\quad M_y = \begin{matrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{matrix}$

Sobel : $M_x = \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$ $\quad M_y = \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$

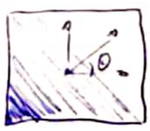
Roberts: $M_x = \begin{matrix} 0 & 1 \\ -1 & 0 \end{matrix}$ $\quad M_y = \begin{matrix} 1 & 0 \\ 0 & -1 \end{matrix}$

Image gradient

The gradient of an image : $\nabla f = \left[\dfrac{\partial f}{\partial x}, \dfrac{\partial f}{\partial y}\right]$

 $\nabla f = \left[\dfrac{\partial f}{\partial x}, 0\right]$

 $\nabla f = \left[0, \dfrac{\partial f}{\partial y}\right]$

 $\nabla f = \left[\dfrac{\partial f}{\partial x}, \dfrac{\partial f}{\partial y}\right]$

$\theta = \operatorname{atan}\left(\dfrac{\partial f}{\partial y} \Big/ \dfrac{\partial f}{\partial x}\right)$: Gradient direction

The gradient points in the direction of most rapid increase in intensity

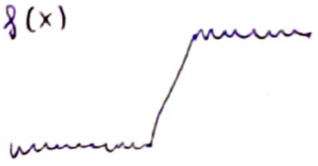
$\|\nabla f\| = \sqrt{\left(\dfrac{\partial f}{\partial x}\right)^2 + \left(\dfrac{\partial f}{\partial y}\right)^2}$: Gradient strength

Application: Gradient-domain image editing

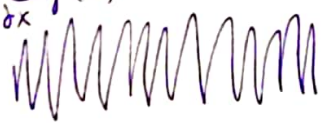
- Goal: solve for pixel values in the target region to match gradients of the source region while keeping background pixels the same

Effects of noise

$f(x)$



$\frac{\partial}{\partial x} f(x)$



→ where is the edge?

Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative:

$$\frac{\partial}{\partial x}(f * g) = f * \frac{\partial}{\partial x} g$$

- This saves us an operation:

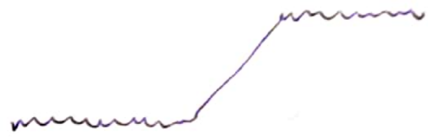
f

$\frac{\partial}{\partial x} g$

$f * \frac{\partial}{\partial x} g$

Solution: smooth first

$f(x)$: signal



g : Kernel

$f * g$: convolution

$\frac{\partial}{\partial x}(f * g)$: differentiation

Separability of the Gaussian filter

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2+y^2}{2\sigma^2}} =$$

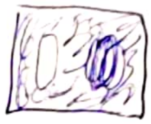
$$= \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{y^2}{2\sigma^2}}\right)$$

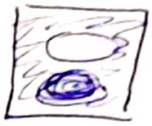
If $\sigma \uparrow$ smoothing you apply

Smoothed derivative removes noise, but blurs edges. Also finds edges at different scales

Smoothing vs derivative filters

- Smoothing filters
  - Gaussian : remove high frequency components : low-pass filter
  - Their values can be negative, but usually all values are positive
  - All values should sum 1, so that constant regions are not affected
- Derivative filters
  - Derivatives of Gaussian
  - They can be negative
  - The values should sum 0
    No response in constant regions



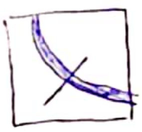


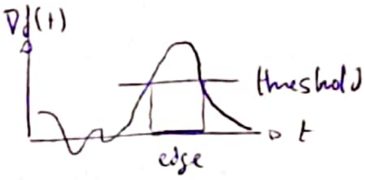
Building edge detector

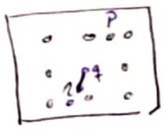
1° Filters in x and y direction and then take the responses into the gradient vector → magnitude and orientation ⇐ norm of the gradient

2° Thresholded norm of the gradient → to discard weak edges
How to turn thick curves into actual lines?

3° Non - maximum suppression







Check if pixel is local maximum along gradient direction, select single max accross width of the edge
- requires checking interpolated pixels p and r

Some edges then dissappear to be too weak despite being relevant. How to solve, lowering threshold? No

4° Hysteresis thresholding
You do not apply the same threshold everywhere
Use a high threshold to start edge curves and low threshold to continue them

All of this is wrapped up in Canny edge detector
MATLAB [edge (image, 'canny')]

Data - driven edge detection

Edge detection can work as ground truth for feeding a ~~[illegible]~~ ML classifier instead of labelling manually every edge in every image.
The output should like the edge detection output but not like a binary output. Real value between 0 and 1.

Keypoint extraction: Corners

Why extract keypoints?

- Panorama stitching: we have 2 images, how do we combine them?

- Step 1: extract points
- Step 2: match keypoint features
- Step 3: align images

Characteristics of good keypoints

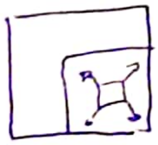
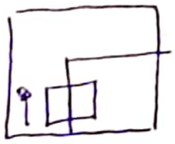
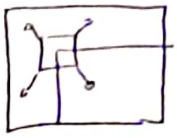
- Repeatability: same keypoint can be found in several images despite geometric and photometric transformations
- Saliency: each keypoint is distinctive
- Compactness and efficiency: many fewer keypoints than image pixels
- Locality: a keypoint occupies a relatively small area of the image; robust to clutter and occlusion

Applications of keypoints

- Image alignment
- 3D reconstruction
- Motion tracking
- Robot navigation
- Indexing and database retrieval
- Object recognition

Corner detection : Basic idea

- We should easily recognize the point by looking through a small window
- Shifting a window in any direction should give a large change in intensity

"flat" region : no change in all directions

"edge" : no change along the edge detection

"corner" : significant change in all directions

Corner detection : Mathematics

- Change in appearance of window W for the shift $[u, v]$:

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

- First-order Taylor approximation for small motions $[u, v]$:

$$I(x+u, y+v) \approx I(x,y) + I_x u + I_y v$$

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

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

$$= \sum_{(x,y)\in W} I_x^2 u^2 + I_y^2 v^2 + 2 I_x I_y uv$$

$E(u,v) \approx [u\ v]\, M \begin{bmatrix} u \\ v \end{bmatrix}$ where M is a second moment matrix obtained from image derivatives

$$M = \begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix}$$ the sums are over all the pixels in the window

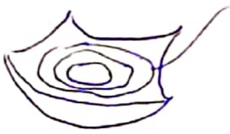
Diagonalization of M

$$M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

If λ_1 and λ_2 are close to 0 ↳ flat area

If they are large ↳ corner

$E(u,v)$



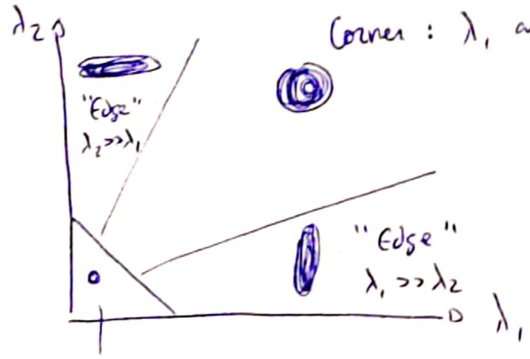
circle if perfectly simmetric otherwise, ellipse

↳ horizontal slice $[u\ v]\, M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$

Axis lengths determined by Eigenvalues

Orientation determined by R

Interpreting the eigenvalues



Corner : λ_1 and λ_2 are large, $\lambda_1 \sim \lambda_2$, E increases in all directions

flat region : λ_1 and λ_2 are small, E remains constant in all directions

Corner response function

$$R = \det(M) - \alpha \operatorname{trace}(M)^2 = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2 \quad \begin{cases} > 0 : \text{corner} \\ < 0 : \text{NOT a corner} \\ \approx 0 : \text{flat region} \end{cases}$$

Harris corner detector

1. Compute partial derivatives at each pixel
2. Compute second moment matrix M in a Gaussian window around each pixel
3. Compute corner response function R
4. Threshold R
5. Find local maxima of response function (nonmaximum suppression)

Robustness of corner features

- The scale of intensities matter
- Partially invariant to affine intensity scale
- Image translation : Corner location is covariant with translation
- " rotation : " " " " " rotation
- Scaling : it is not invariant

SIFT keypoint detection

- CNN are much better for semantic problems
- Good enough for many applications

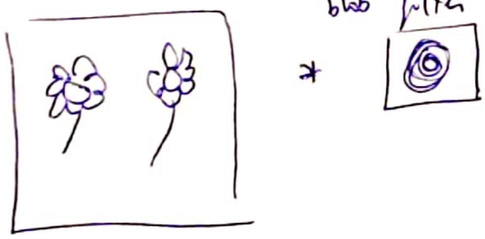
Key point detection with scale selection

- We want to extract keypoints with characteristic scale that is covariant with the image transformation

Basic idea

- Block detection filter
- Convolve the image with a "blob filter" at multiple scales and look for extrema of filter response in the resulting scale space

blob filter

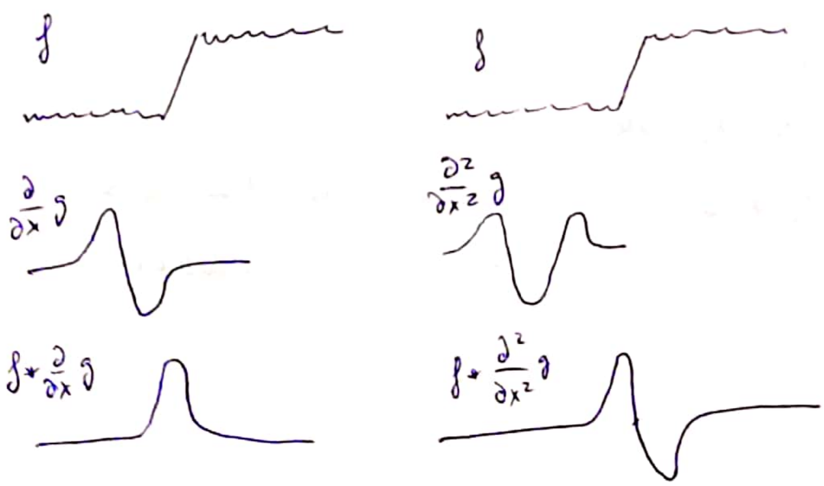


- Find maxima and minima of blob filter response in space and scale

Blob filter

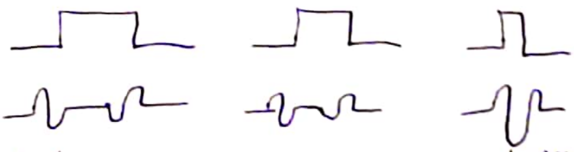
Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D

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



edge in zero crossing of second derivative

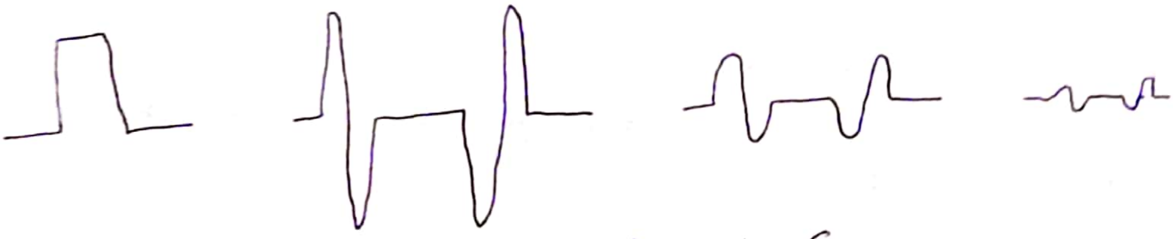
From edges to blobs



- Edge = ripple
- Blob = superposition of two ripples
- Spacial selection: the magnitude of the Laplacian response will achieve a maximum at the center of the blob, provided the scale of the Laplacian is matched to the scale of the blob

Scale selection

- We want to find the characteristic scale of the blob by convolving it with Laplacians at several scales and looking for the maximum response
- However, Laplacian response decays as scale increases

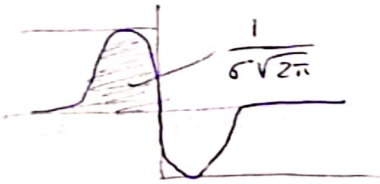


increasing σ

Unnormalized Laplacian Response

Scale normalization

- The response of a derivative of Gaussian filter to a perfect step edge decreases as σ increases



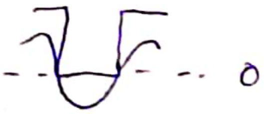
- To keep response the same (scale-invariant), must multiply Gaussian derivative by σ
- Laplacian is the second derivative of the Gaussian so it must be multiplied by σ^2

Blob detection 2D

- Scale-normalized Laplacian of Gaussian

$$\nabla^2_{norm}\, g = \sigma^2\left(\frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}\right)$$

- At what scale does this achieve the maximum response to a binary circle or radius r?
  - The zeros of the Laplacian have to be aligned with the circle



  - The Laplacian is given by (up to scale):

$(x^2+y^2-2\sigma^2)e^{-(x^2+y^2)/2\sigma^2}$

$\xrightarrow{\text{maximum}} \sigma = \frac{1}{\sqrt{2}} r$

Scale-space blob detector

1- Convolve image with scale-normalized Laplacian at several scales
2- Find maxima of squared Laplacian response in scale-space

Eliminating edge responses

- Laplacian has strong response along edge
- Solution: filter based on Harris response function over neighborhoods containing the blobs

Efficient implementation

- Approximating the Laplacian with a difference of Gaussians:

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

$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$ ← Difference of Gaussians

From feature detection to feature description

- Scaled and rotated versions of the same neighborhood will give rise to blobs that are related by the same transformation
- What to do if we want to compare the appearance of these image regions?
  - Normalization: transform these regions into same-size circles
  - Problem: rotational ambiguity

Eliminating rotation ambiguity

- To assign a unique orientation to circular image windows
  - Create histogram of local gradient directions in the patch
  - Assign canonical orientation at peak of smoothed histogram

SIFT descriptor computation

- use the normalized region about the keypoint
- compute gradient magnitude and orientation at each point in the region
- weight them by a Gaussian window overlaid on the circle
- create an orientation histogram over 4x4 subregions of the window
- 4x4 descriptors over 16x16 sample array were used in practice
  4x4 times 8 directions gives a vector of 128 values

Summary of SIFT feature detection and description

1- Scale-space extrema detection
   Search over multiple scales and image rotations

2- Keypoint localization
   Fit a model to determine location and scale
   Select keypoints based on a measure of stability

3- Orientation assignment
   Compute best orientation(s) for each keypoint region

4- Keypoint description
   Use local image gradients at selected scale and rotation
   to describe each keypoint region

It is robust to image condition changes

Properties of SIFT

Extraordinarily robust detection and description technique
- Can handle changes in viewpoint
  - up to about 60 degree out-of-plane rotation
- Can handle significant changes in illumination (day vs night sometimes)
- Fast and efficient - can run in real time
- Lots of code available