Edge and corner detection

→ The goal of edge detection is to identify sudden changes in an image, how to find the main details and remove noise.

→ Intensity changes are caused by :-

- Geometric Events
  - Surface orientation discontinuity
  - depth discontinuity
  - color and texture discontinuity
- Non-geometric events
  - illumination changes
  - Specularities
  - Shadows

→ Edge detection is useful because we can extract important features from the edges of an image. (Eg. lines, corners, curves) these features are useful for higher level computer vision algorithms.

→ Edge descriptors → Edge direction (is the direction of max intensity change), Edge strength (related to local image contrast along the normal) and Edge position (image position at which the edge is located).

→ Edge → is a place of rapid change in the image intensity function.

Image gradient $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$ → The gradient points in the direction of most rapid increase in intensity.

gradient direction $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

Edge strength $\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$ } gradient magnitude

To calculate this derivative, we follow $\left\{\frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0}\left(\frac{f(x+\varepsilon, y) - f(x, y)}{\varepsilon}\right)\right\}$

OR

in discrete function $\left\{\frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}, y) - f(x_n, y)}{\Delta x}\right\}$

→ Sobel filter → It is a filter to detect edges in an image. The steps followed are

① selection of derivative filters. Eg. $S_x =$

1	0	-1
2	0	-2
1	0	-1

$S_y =$

1	2	1
0	0	0
-1	-2	-1

② Convolving with the image to compute derivatives $\left\{\frac{\partial f}{\partial x} = S_x \otimes f, \frac{\partial f}{\partial y} = S_y \otimes f\right\}$

③ Form the image gradient and compute its direction and amplitude

$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$ $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$ $\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

★ we have many derivative filters derived from sobel filter.

→ When we have noise in our image and we take the derivative of it, it is amplified and our edge is lost.

→ The filters (finite differences) are very sensitive to noise: Image noise results in pixels that look very different from their neighbours. Generally, larger the noise, the stronger the response.
The solution is to perform smoothing. Smoothing with a large standard deviation (σ) also blurs the image.

→ At the end of our operation, we have a gradient. But still, we want to calculate the edge points. The criteria for an optimal edge detector:

1. Good detection → minimize the probability of false positive & false negatives
2. Good localization → the detected edge must be as close to the true edge as possible
3. Single response → the detector must return one point only for each true edge point.

→ Canny Edge detector → The steps for canny edge is as follows:

1. Filter the image with the derivative of Gaussian.
2. Find the orientation and magnitude of the gradient
3. Non-maximum suppression:
   means to thin down the multi-pixel wide "ridges" to single pixel width.
4. Linking and thresholding (hysterisis)
   1. Defining two thresholds i.e. low and high.
   2. using the high threshold to start the edge curves and low threshold to continue them.

MATLAB: edge(image, 'canny')

Non-maximum suppression:- Taking the gradient perpendicular to the edge and calculating the maximum among those points. whichever is maximum is considered to be a strong point/pixel.
Then we create a tangent along the edge curve and repeat the same operation.

Linking and thresholding:- check whether the maximum value of gradient value is sufficiently large.

★ The choice of σ depends upon the desired behaviour:
1. large σ detects large scale edges.
2. small σ → fine features.

Corner detection

It is used for image matching

→ Advantages of local features:- (1) Locality → features are local, so robust to occlusion and clutter.
(2) Distinctiveness → can differentiate a large database of objects
(3) Quantity → hundreds or thousands in a single image.
(4) Efficiency → real-time performance achievable.
(5) Generality → exploit different types of features in different situations.

Steps in corner detection →

Input image → Apply corner detector → Cornerness map → Threshold cornerness map → Threshold map → Non-maximum suppression → Corners

① Invariant local features : the goal is to find the features in the images which are invariant to :

(i) geometric changes → translation, rotation, scale.
(ii) photometric changes → brightness, exposure.

② we have three types of interest points

① Homogenous area surface
② Edges (line)
③ mixed area (corners) Imp.

} doing sum - squared difference, we will get 3 different maps.

③ local measure of uniqueness → what defines whether a feature is a good or bad candidate ?

④ The math of feature detection → (1) Consider a window which is shifted by (u,v) and see how do the pixels in W change ?
(2) Compare each pixel ~~by~~ before and after by summing up the squared difference.
(3) this defines an SSD error

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

★ Considering small motion we will expand it using taylor's expansion.

$$I(x+u, y+v) = I(x,y) + \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial x} v + \dots \text{higher terms}$$

if the motion is small, the approximation is good :-

$$I(x+u, y+v) = I(x,y) + [I_x \; I_y] \begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(u,v) = \sum_{(x,y) \in W} \left[I(x,y) + [I_x \; I_y] \begin{bmatrix} u \\ v \end{bmatrix} - I(x,y) \right]^2$$

$$= \sum_{(x,y) \in W} \left[[I_x \; I_y] \begin{bmatrix} u \\ v \end{bmatrix} \right]^2$$

can be rewritten as $E(u,v) = \sum_{(x,y) \in W} [u \; v] \underbrace{\begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$ } Equation of quadratic

$\therefore u^T H u$ is a quadratic function.

★ Thus, we can analyse E's shape by working at property of H.

① flat quadratic function. [flat]
② 1 dominant axis [edge]
③ parabolic function with x and y. [corner]

To do this, we will look at the eigen value / eigen vector of the quadratic

Harris Corner detector → We can visualize H as an ellipse with axis lengths and directions determined by its eigenvalues and eigenvectors (matrix R)

→ We want $E(u,v)$ to be large for small shifts in all directions i.e.
- → the minimum of $E(u,v)$ should be large over all unit vectors $[u\ v]$
- → this minimum is given by the smaller eigenvalue (λ_-) of H

→ **Feature detector steps (Kanade - Tomasi)**

Imp:
1. Compute the gradient at each point in the image.
2. Create the H matrix from entries in the gradient
3. Compute the eigenvalues
4. Find points with large response ($\lambda_- >$ threshold)
5. Choose points where λ_- is a local maximum as features.

→ **Harris operator / Corner detector**

To find eigenvalues we usually do sum-squared difference which is computationally expensive

$$\lambda\pm = \frac{1}{2}\left[(h_{11}+h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11}-h_{22})^2}\right]$$

Harris proposed a way to solve this problem

λ_- is the variant of "Harris operator" for feature detection ($\lambda_- = \lambda_1$; $\lambda_+ = \lambda_2$)

$$f = \frac{\lambda_1\lambda_2}{\lambda_1+\lambda_2} \simeq \frac{\text{determinant}(H)}{\text{trace}(H)}$$

① trace is the sum of the diagonals $\text{trace}(H) = h_{11}+h_{22}$

★ **measure of corner Response (Harris)**

$$R = \det(H) - K(\text{trace}(H))^2$$

$\det(H) = \lambda_1\lambda_2$
$\text{trace}(H) = \lambda_1+\lambda_2$
K = Empirical constant
$K = 0.04 - 0.06$

if R = large ≃ corner
R = small ≃ flat
} $R<0$ → edge.

→ **Harris detector steps**

1. Compute gaussian derivate at each pixel.
2. Compute second moment matrix H in a gaussian window around each pixel.
3. Compute corner response function R
4. Threshold R
5. Find local maxima of the Response function (non-max suppression)

$$R = \det(H) - \alpha(\text{trace}(H))^2 = \lambda_1\lambda_2 - \alpha(\lambda_1+\lambda_2)^2$$

Scale Invariance and Blob detection

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

arranging this function, we get

$$E(u,v) = \sum_{(x,y)\in W} [u\ v] \underbrace{\begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}}_{H\ (\text{Symmetric matrix})} \begin{bmatrix} u \\ v \end{bmatrix}$$ } it's a quadric

→ Corner detection steps: ① Compute Gaussian derivatives at each pixel
② Compute second moment matrix (H) in a Gaussian window around each pixel.

→ Harris corner detector requires 2 scale parameters:
① a differentiation scale (σ_D) for smoothing prior to the computation of image derivatives
② an integration scale (σ_I) for defining the size of gaussian window

$$H_W(x,y) = H_W(x, y, \sigma_D, \sigma_I) \qquad \therefore \boxed{\sigma_I = \gamma \sigma_D}$$

→ The above mentioned two parameters are important in order to have the scale invariance. We also want our corner detector to be invariant to Rotation, Affine or Linear Intensity change.

→ Harris Detector → Corner response R is invariant to image rotation.

→ Photometric changes: Linear Intensity change:
→ Only derivatives are used → invariance to intensity shift $I(x,y) \to I(x,y) + b$

→ Intensity scale: $I(x,y) \to a\, I(x,y)$

→ Partially invariant to affine intensity change

→ Scale invariance → Harris detector is not invariant to scaling (and affine transforms) i.e. all points will be classified as edges rather than corners.

→ To check how good a corner detector is, we define the Repeatability rate

Repeatability rate: percentage of corresponding points

$$\text{repeatability} = \frac{\#\ \text{correspondences}}{\#\ \text{detected}} \cdot 100\%$$

defined as Homography.

Two points are corresponding if $T = 60\%$ (threshold of distances)

$$\frac{A \cap B}{A \cup B} > T$$

The repeatability for Harris corner detector is very high and usually constant.

Disadvantages: ① Sensitive to scale change
② " " to significant view point change
③ " " to " " contrast change

→ To handle scale changes we should adapt the following :-

① $H_w(x, y, \sigma_I, \sigma_D)$ must be adapted to scale changes.

② If scale change is known, adapt the harris detectors to it by properly setting σ_I and σ_D

If scale change is unknown, detect interest points at multiple scale.

→ Multi-scale Harris Detector → The idea of multi-scale is to detect interest points at varying scale.

$$R(H_w) = \det(H_w(x, y, \sigma_I, \sigma_D)) - \alpha \, \text{trace}^2(H_w(\sigma_I, \sigma_D, x, y))$$

i.e. increasing our area with every scaling step.

★ But this is not ideal, because it leads to a shift in the centre point as the scale increases. (due to smoothing)

→ Scale selection : characteristic scale

→ In our both the images (ie. original & scaled image) we find the characteristic scale of that feature.

→ This scale reveals the spatial extent of an interest point.

→ matching can be simplified.

→ Automatic scale selection → ① Design a function $F(x, \sigma_n)$ which provides some local measure.

② Select points at which $F(x, \sigma_n)$ is maximal over σ_n

Original image

① Integrate the different scales
② find the maxima

Scaled image

① Integrate the different scales
② find the maximum

Normalise and Compare → Scale factor σ_1 / σ_2

→ How to choose $F(x, \sigma_n)$?

① The function should be rotation invariant
② The function should have one stable sharp peak.

★ LOG is the best method that calculates this $F(x, \sigma_n)$

Blob detection with scale selection

→ from sobel filters we know that, for laplace function

$$\frac{\partial}{\partial x} f = \lim_{\varepsilon \to 0} \left(\frac{f(x+\varepsilon, y) - f(x, y)}{\varepsilon} \right) \quad \text{or} \quad \frac{\partial}{\partial x} f \approx \frac{f(x_{i+1}, y) - f(x_{i-1}, y)}{2 \cdot \Delta x}$$

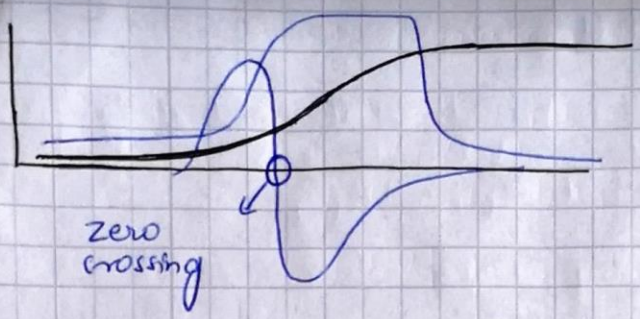
$$= \nabla_x * f \quad (\text{convolution})$$

→ If we want to calculate the laplacian of an image, we calculate the 2nd order derivative

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

} used for edge detection. (laplacian zero crossings)

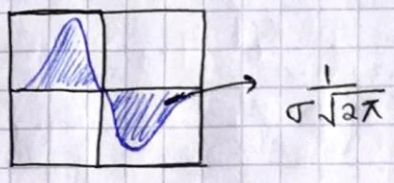
$$\frac{\partial^2 f}{\partial x^2} = \begin{pmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{pmatrix} \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} \cdot f$$



Edge - ripple
Blob - superposition of 2 ripples.

→ 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.

★ problem → If σ increase, the response of laplacians becomes smaller.



→ Scale normalization → To keep the response same, (scale invariant) one must multiply the Gaussian derivative by σ

Since, laplacian is the "second" derivative, so it must be multiplied by σ^2

→ Laplacian of Gaussian → It is a circularly symmetric operator for blob detection in 2D.

$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} = \frac{1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$ } 2nd derivative of gaussian

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

→ At what scale does LoG achieve a maximum response for a binary circle of radius r?

→ LoG is maximized at $\boxed{\sigma = r/\sqrt{2}}$ (characteristic scale)

↙ putting $1 - \frac{x^2+y^2}{2\sigma^2} = 0$ i.e. $1 - \frac{r^2}{2\sigma^2} = 0$

$r^2 = 2\sigma^2$

$\boxed{\sigma = r/\sqrt{2}}$

→ Scale-space blob detector ① convolve image with scale-normalised laplacian at several scales.

② find maxima of squared laplacian response in scale-space.

★ Since laplacian of Gaussian is a very lengthy process, we approximate the laplacian with a difference of gaussians.

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

$$\boxed{DoG = G(x,y,k\sigma) - G(x,y,\sigma)} \approx (k-1)\sigma^2\nabla^2 G$$ } already normalised laplacian.

→ for detecting the extrema in DoG pyramid :-
we compare each point to its 8 neighbours at the same level and 9 neighbours in the level above and below.