Chapter 4 – Corner Detection / Harris Corner

Author: Gianmarco Scarano

gianmarcoscarano@gmail.com

1. Detecting corners

We detect corners for image alignment, 3D reconstruction, object recognition and robot navigation.

Features are image regions that are unusual, which lead to unambiguous matches in other images.

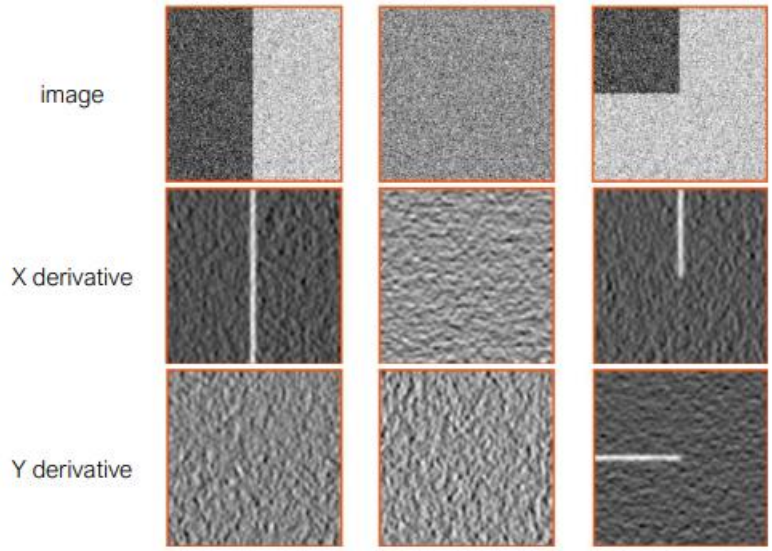
We extract features from an image for e.g. panorama stitching, image and feature matching.

Features are also invariant to rotation, scale, translation which is a HUGE advantage.

2. Harris Corner detector

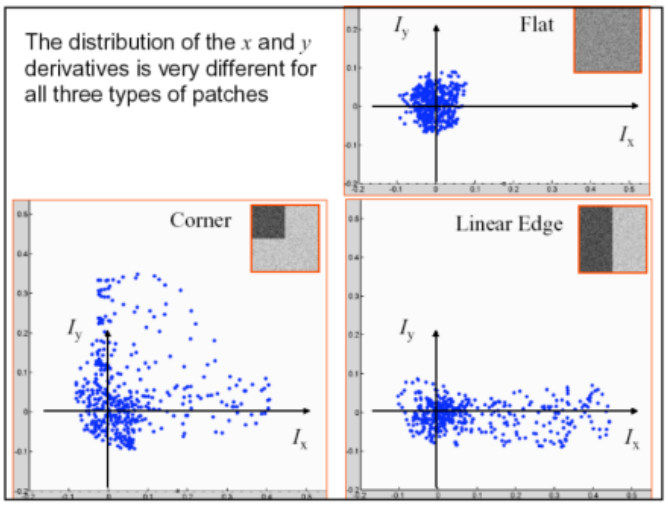
A corner can be detected through a small window which gets shifted such that it would give large intensity changes, but we can use an algorithm which detects corners with PCA:

- Compute Image Gradients over a small region: We extract gradients over the X and Y. X derivative has vertical edges, as we can see from this image below.

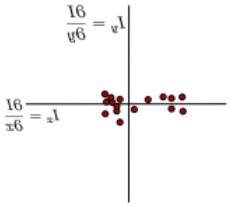


Here we can see that 1 has vertical edges, the centre part detects the flat region, while the 3rd detects the corners.

- We can plot the derivatives in a 2D space, followed by a subtraction of mean from each gradient:



From this, we can move into a distribution representation which reveals edge orientation and magnitude. Then, as said before, we subtract the mean from each image gradient such that the representation is centered in zero.



- Compute the covariance Matrix

$$\begin{bmatrix} \sum_{p\in P} I_x I_x & \sum_{p\in P} I_x I_y \\ \sum_{p\in P} I_y I_x & \sum_{p\in P} I_y I_y \end{bmatrix}$$

$$\sum_{p\in P} I_x I_y = \text{sum}\Big(I_x = \frac{\partial I}{\partial x} \;.*\; I_y = \frac{\partial I}{\partial y} \Big)$$

array of x gradients — array of y gradients

Here, if we consider moving the windows W by (u,v), how do the pixel in W change? Well, very simply, we compare each pixel before and after by summing up the squared differences. If this error is high, then we might be satisfied (there is a corner).
At the end the Error function is given by a sum of the Window function "w(x,y)" which could be a Gaussian or a binary output, followed by the difference between the shifted intensity and the original intensity, just as follows:

Change of intensity for the shift [u,v]:

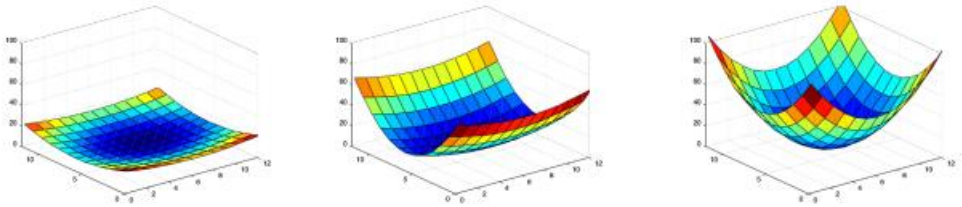
$$E(u,v) = \sum_{x,y} w(x,y)\big[I(x+u,y+v) - I(x,y)\big]^2$$

Error function — Window function — Shifted intensity — Intensity

We approx. the image shifted by Taylor expansion

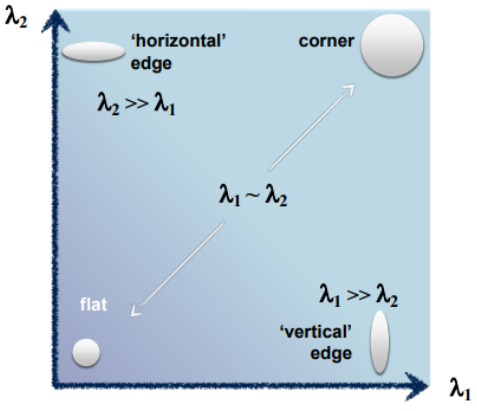
Window function *w(x,y)* = 1 in window, 0 outside — or — Gaussian

Quadratic error function:



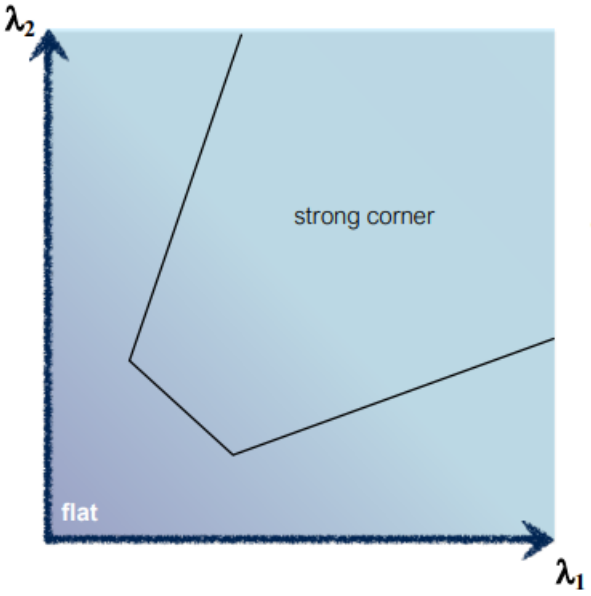
flat — edge 'line' — corner 'dot'

- Compute eigenvalues and eigenvectors:



Basically, what we do here is to visualize an ellipse starting from the points we found in the gradient representation such that we have to values $\lambda 1$ and $\lambda 2$.
If both values are equal, then we have a flat region, if $\lambda 2 \approx 0$ and $\lambda 1 >> 0$ then we have an edge and so on and so forth.

- Use threshold on eigenvalues to detect corners:



Think of a function to score 'cornerness'

Harris Detector: Summary

- Average intensity change in direction $[u,v]$ can be expressed as a bilinear form:

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

- Describe a point in terms of eigenvalues of M: *measure of corner response*

$$R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2$$

- A good (corner) point should have a *large intensity change* in *all directions*, i.e. R should be large positive

The Harris Corner Detector is not invariant to scale. It means that if a small region detects a corner, it doesn't necessarily means that it will still be a classified as a corner when we scale up the image.

Concluding this chapter, when we apply different Laplacian filters, we need to search over a scale that produces a peak in the Laplacian response: