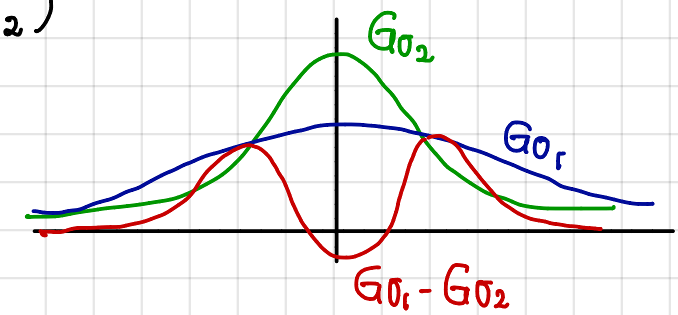
LECTURE 4 : SIFT

For the complete discussion, see the corresponding slides & paper.

[i] Difference of Gradients (DoG) vs. Laplacian of Gradients (LoG)

» $DoG(I) = (I * G_{\sigma_1}) - (I * G_{\sigma_2})$
$= I * (G_{\sigma_1} - G_{\sigma_2})$

If I is a one-dimensional signal:



» LoG :

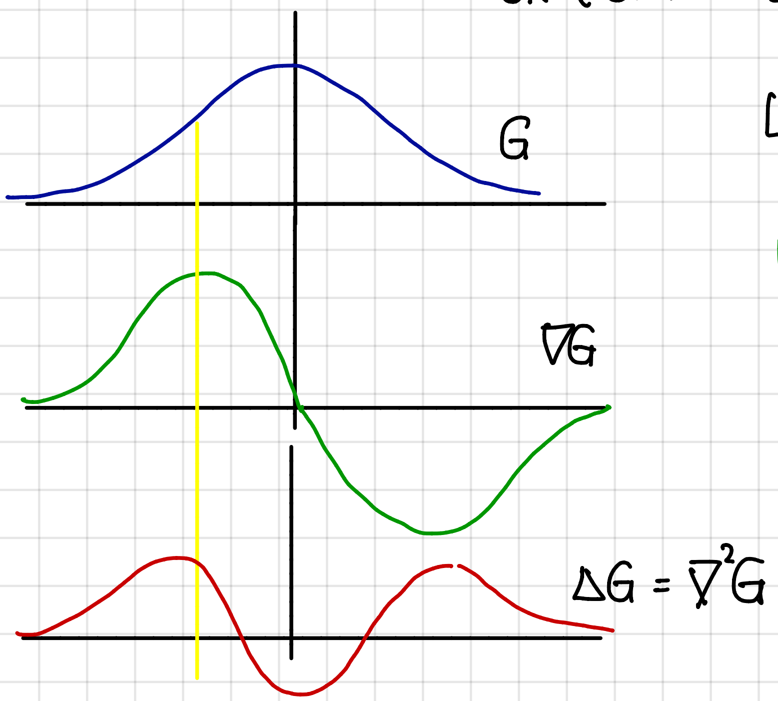
$$\Delta G = \nabla^2 G = \nabla \cdot \nabla G$$

$\nabla \cdot$ = divergence $\qquad$ ∇G = gradient

$$\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right) \qquad \nabla G = \left(\frac{\partial G}{\partial x}, \frac{\partial G}{\partial y}\right)$$

in 2D

$$\Delta G = \nabla \cdot \nabla G = \frac{\partial}{\partial x}\left(\frac{\partial G}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\partial G}{\partial y}\right) = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}$$



$$LoG(I) = I * \Delta G$$

DoG is the approximation of LoG.

$$DoG(I) \approx LoG(I)$$

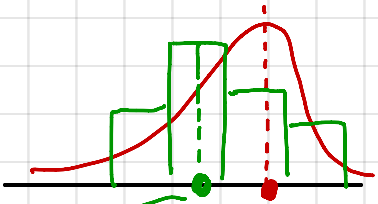
[2] SIFT: Low Contrast Rejection

Motivation:

(1) Due to noise, the actual location of an extremum might be shifted.
(2) Even if we know the actual location, it can be not high enough (= low contrast).

Q: What does it mean by the actual location of an extremum? How to calculate the actual location?

A:



the actual location

the predicted extremum location (using DoG)

- The green bars are the noisy pixel intensity values
- The red line is the actual (non-noisy) signal

» The actual signal is shifted by noise and discretization.

$\mathbb{D}$ = a 3x3 pixel patch where a keypoint located in the middle, and $\mathbb{D}$ is taken from one of the DoG images.
This 3x3 patch might be affected by noise.

$D(\bar{x})$ = the actual signal ; $\bar{x} = (x, y)$ → the actual location

Unfortunately, we don't know the actual signal location ($\bar{x}$).

Q: How to get $D(\bar{x})$, the signal we want to recover?
A: Through Taylor expansion:

$$\underbrace{f(x)}_{\text{the unknown function}} = f(a) + f'(a)(x-a) + \frac{f''(x-a)}{2!}(x-a)^2 + \dots$$

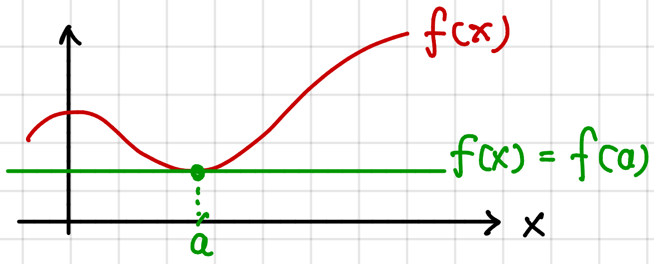
[3] Taylor Expansion

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(x-a)}{2!}(x-a)^2 + \dots$$

Meaning:

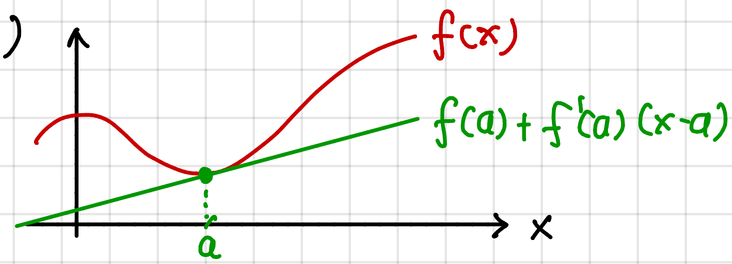
1. When $x = a$: $f(x) = f(a)$

The approximation is rough & basic

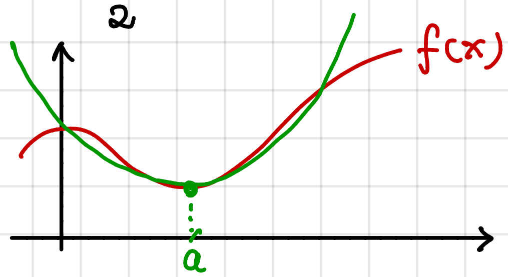


2. When $f(x) = \underbrace{\underline{f(a)} + \underline{f'(a)}}_{\text{scalar values}}(x-a)$

The approximation gets better.



3. When $f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2}(x-a)^2$



Notes:

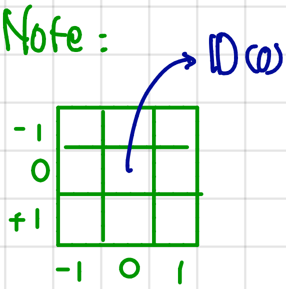
- The approximation gets better when we include the higher order functions.
- 'a' is the point where we want to focus our approximation.

[4] Location of Extrema

Using Taylor expansion in 2D:

$$\underset{1\times1}{D(\bar{x})} = \underset{1\times1}{D(0)} + \underset{1\times2}{\frac{\partial D^T(\bar{x})}{\partial \bar{x}}}\Bigg|_{\bar{x}=0} \underset{2\times1}{\bar{x}} + \underset{1\times2}{\bar{x}^T} \underset{2\times2}{\frac{\partial^2 D(\bar{x})}{2\,\partial \bar{x}^2}}\Bigg|_{\bar{x}=0} \underset{2\times1}{\bar{x}}$$

Note:

$D =$ 

;

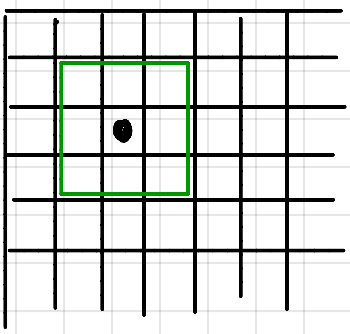
$$\frac{\partial D(\bar{x})}{\partial \bar{x}}\Bigg|_{\bar{x}=0} = \begin{bmatrix} \partial D/\partial x \\ \partial D/\partial y \end{bmatrix}\Bigg|_{\bar{x}=0}$$

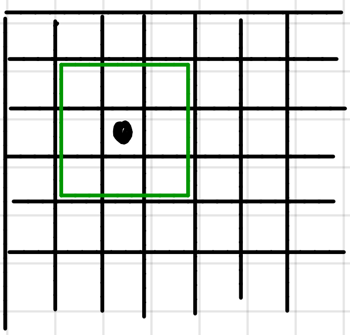
$$\frac{\partial^2 D(\bar{x})}{\partial \bar{x}^2}\Bigg|_{\bar{x}=0} = \begin{bmatrix} \frac{\partial^2 D}{\partial x^2} & \frac{\partial^2 D}{\partial x \partial y} \\ \frac{\partial^2 D}{\partial y \partial x} & \frac{\partial^2 D}{\partial y^2} \end{bmatrix}\Bigg|_{\bar{x}=0}$$

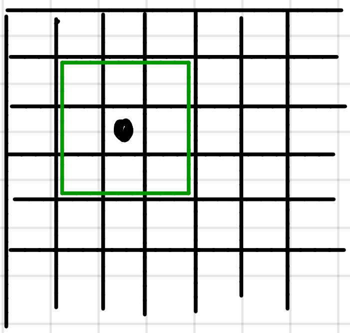
Q: What are these D, D_x, D_y, D_{xx}, D_{xy}, D_{yy}?

A: D is a DoG image (the output of the previous step), where there are a number of keypoints. For each of these keypoints, we take 3x3 pixels from D.

D_x (or $\frac{\partial D}{\partial x}$) & D_y (or $\frac{\partial D}{\partial y}$) are the first derivative image of D.

D: 

D_x: 

D_{xx}: 

We compute D_y, D_{yy}, and D_{xy} (which is the same as D_{yx}) images in the same way.

To find the true extremum means:

$$\frac{\partial D(\bar{x})}{\partial \bar{x}} = 0$$ → the output is a vector of 2x1.

$$\frac{\partial D(0)}{\partial \bar{x}} + \frac{\partial}{\partial \bar{x}}\left[\left.\frac{\partial D}{\partial \bar{x}}^T\right|_{\bar{x}=0} \bar{x}\right] + \frac{\partial}{\partial \bar{x}}\left[\bar{x}^T \left.\frac{\partial^2 D}{2\,\partial \bar{x}^2}\right|_{\bar{x}=0} \bar{x}\right] = 0$$

constant

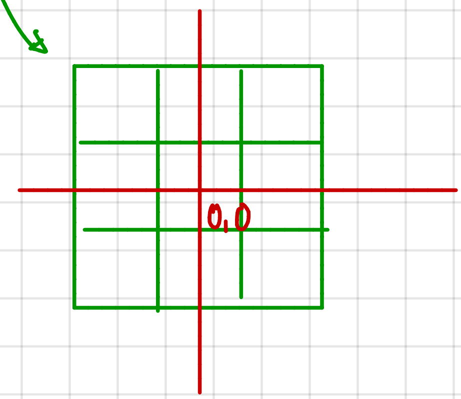
$$0 + \left.\frac{\partial D(\bar{x})}{\partial \bar{x}}\right|_{\bar{x}=0} + \left.\frac{\partial^2 D(\bar{x})}{\partial \bar{x}^2}\right|_{\bar{x}=0} \bar{x} = 0$$

$$\boxed{\bar{x}^* = -\left(\left.\frac{\partial^2 D(\bar{x})}{\partial \bar{x}^2}\right|_{\bar{x}=0}\right)^{-1} \left.\frac{\partial D(\bar{x})}{\partial \bar{x}}\right|_{\bar{x}=0}}$$

If the offset of $\bar{x}^* > 0.5$

then: check the contrast based on the new location $\bar{x}^* \to D(\bar{x}^*)$

else check the contrast based on $D(0)$.



the distance from (0,0) to the cell boundaries is 0.5.

[5] Low Contrast : Checking / Verification

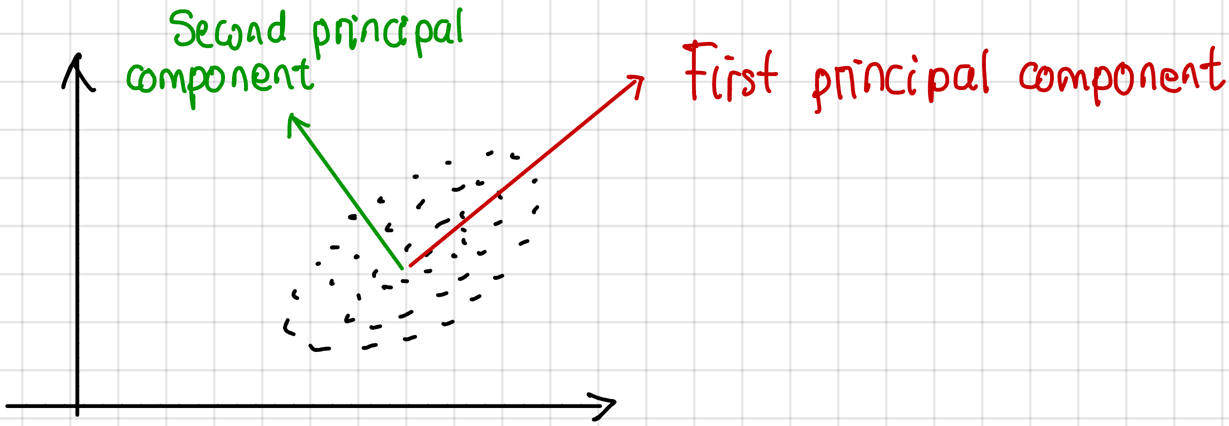
Based on the new location $\bar{x}^*$, we check if the extremum is high enough:

$$D(\bar{x}^*) = \underset{1\times 1}{D[0]} + \underset{1\times 2}{\left.\frac{\partial D^T(\bar{x})}{\partial \bar{x}}\right|_{\bar{x}=0}} \underset{2\times 1}{\bar{x}^*}$$

If $|D(\bar{x}^*)| < \underline{0.03}$ then reject!

it assumes the image intensity of the input is between $0 \sim 1$ (instead of $0 \sim 255$).

[6] First and Second Principal Components of Curvatures



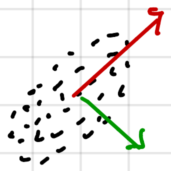
α = The eigen value of the first principal component

β = The eigen value of the second principal component

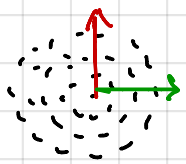
$\alpha \gg \beta$

[7] Corner Detection

Eigenvalues (α & β) can indicate whether a pixel cluster is edge or corner :

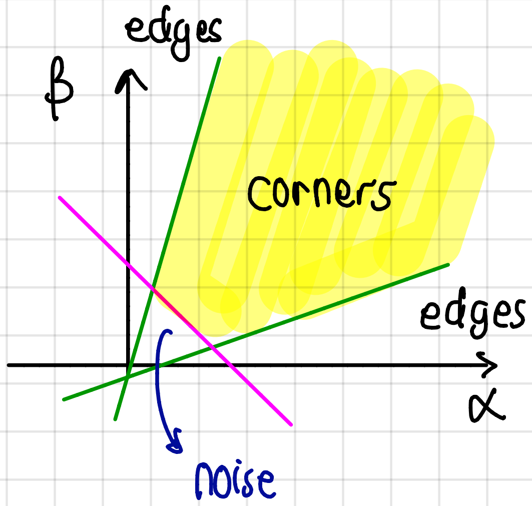


$\alpha \gg \beta \rightarrow$ edge



$\alpha \approx \beta \rightarrow$ corner

$\Rightarrow$



Unlike the above illustration, our data is only 3x3 pixels (9 pixels), which is too sparse to calculate the principal axes.

$\downarrow$

Solution : to use the ratio of α & β :

1. Hessian matrix : $\mathbb{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$

2. $Tr(\mathbb{H}) = D_{xx} + D_{yy} = \alpha + \beta$
   $Det(\mathbb{H}) = D_{xx} D_{yy} - D_{xy}^2 = \alpha\beta$

3. $\dfrac{Tr^2(\mathbb{H})}{Det(\mathbb{H})} = \dfrac{(\alpha+\beta)^2}{\alpha\beta}$; define : $\alpha = r\beta$

$$= (r\beta+\beta)^2 / r\beta^2 = \frac{(r+1)^2}{r}$$

if $\alpha = \beta \rightarrow r = 1$: $Tr^2(\mathbb{H}) / Det(\mathbb{H}) = 4 \rightarrow$ corner

if $\alpha = 2\beta \rightarrow r = 2$: $Tr^2(\mathbb{H}) / Det(\mathbb{H}) = 9/2 = 4.5$

if $\alpha = 10\beta \rightarrow r = 10$: $Tr^2(\mathbb{H}) / Det(\mathbb{H}) = 121/10 = 12.1 \rightarrow$ edge

Therefore : if $Tr^2(\mathbb{H}) / Det(\mathbb{H}) < 12.1$
then retain the keypoints, else reject them.

[8] SIFT: Generating Descriptors

① Orientation assignment

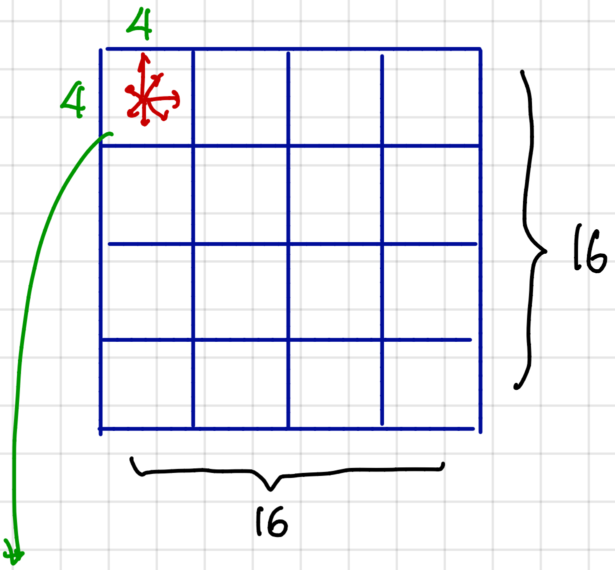
Goal: to make the descriptor invariant to image rotation.

1. Extract 16x16 pixels surrounding a keypoint.
2. Create a histogram of orientations with 36 bins covering 360°.
3. Choose the highest peak in the histogram, and any peaks above 80%, to calculate the orientation normalization.

(2) Descriptor:

One keypoint generates one descriptor, which has a length of 128.

These 128 numbers are obtained from 16x16 pixels:



For each block, we compute the histogram of gradients with 8 bins (= orientations)

Hence, for 1 block of 4x4 pixels we have 8 numbers. Thus, in total we have:

$4 \times 4 \times 8 = 128.$