

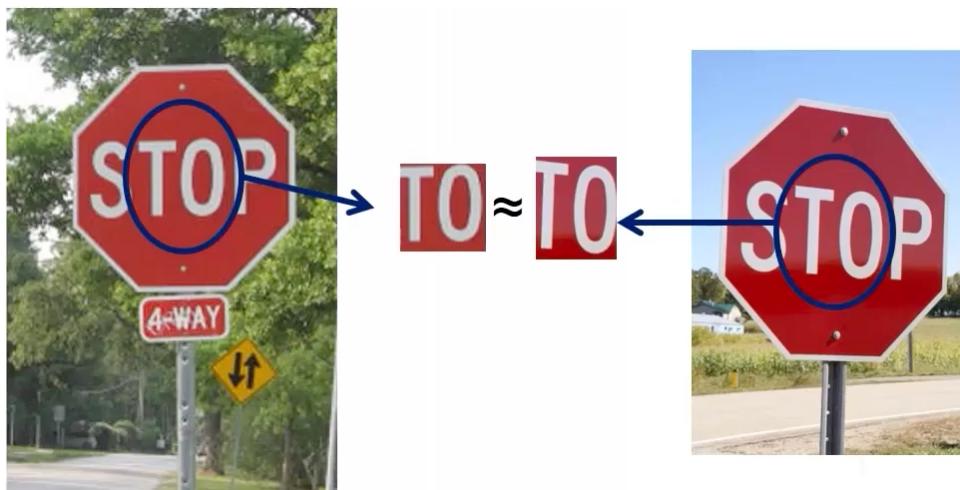


# Interest points

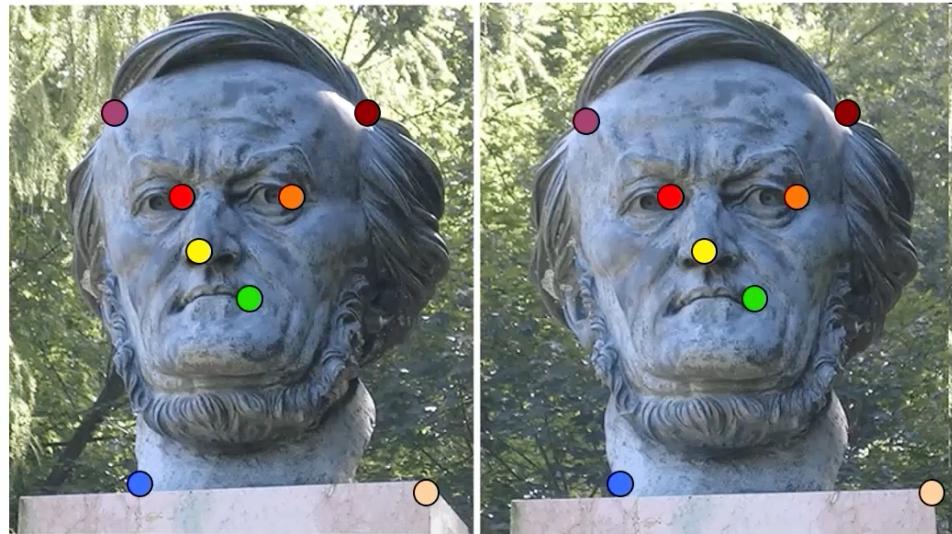
## 학습목표

영상 내 Corner point를 검출하는 방법에 대해 학습

Correspondence가 뭘까?



두 이미지의 Correspondence를 얘기하자면 일치함을 측정하는 것임



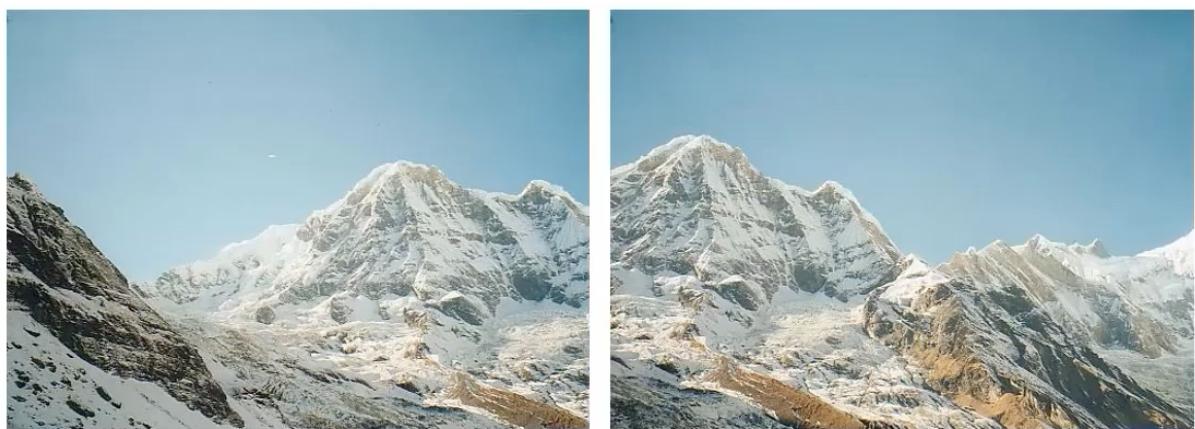
다른 예제를 확인해보면 눈 코 입을 픽셀단위의 일치도를 확인 할 수 있음

## 이런 corner point를 뽑는 것에 대해 배워보자

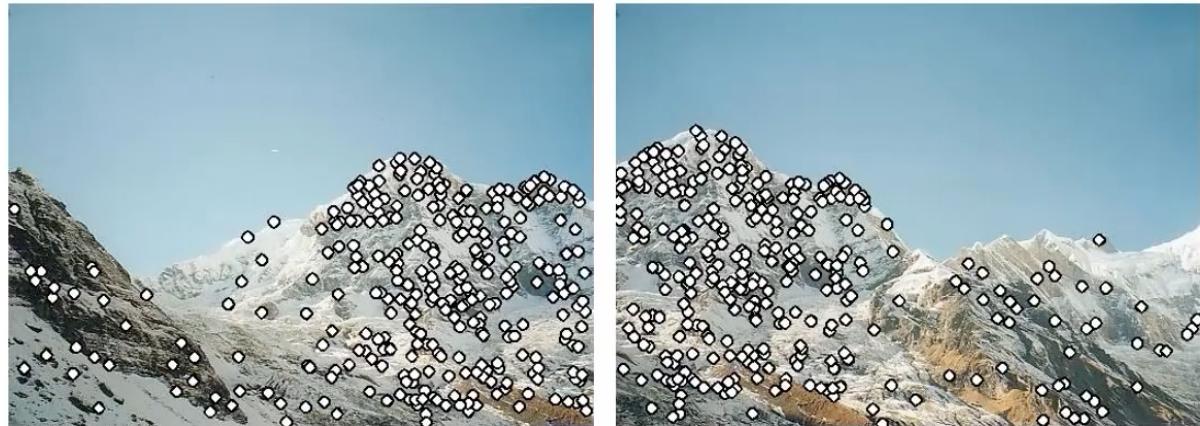
“interest points” = “keypoints”, also sometimes called “features” or “feature points”

이러한 corner point는

### Image Stitching

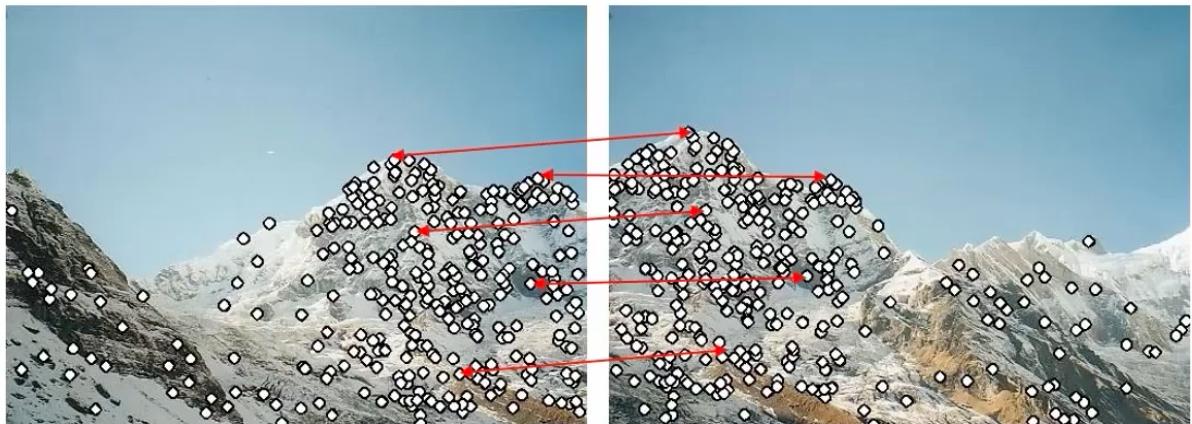


이렇게 두 이미지가 주어졌을 때, 겹치는 부분이 있음. 오버랩 되는 부분을 겹쳐서 이어주면 됨



Detect feature pints in both images

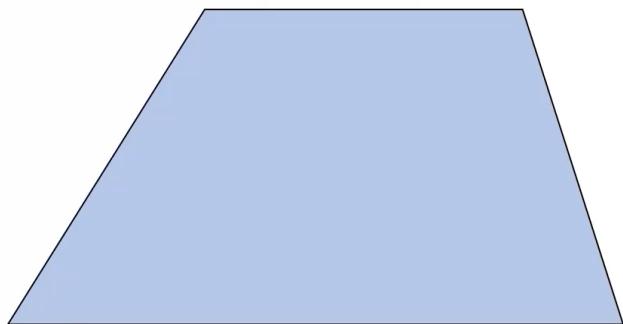
하지만 특징을 뽑을때 하늘처럼 특색이 없는 부분을 뽑으면 안되고, 코너에 해당되는 부분을 뽑아야 함



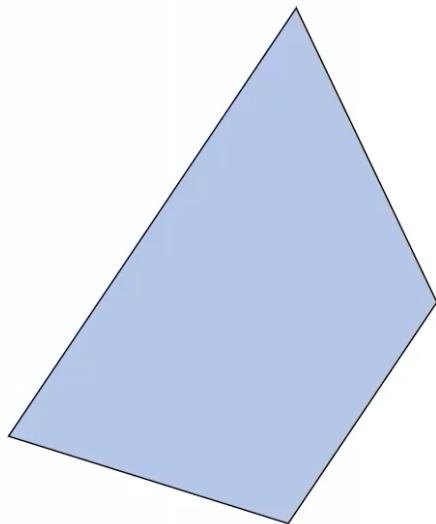
그 다음 점들을 매칭시킨 후 매칭관계를 표현함(wraping)



다음그림의 특색있는 점을 한번 꼽아보자



아래 그림도



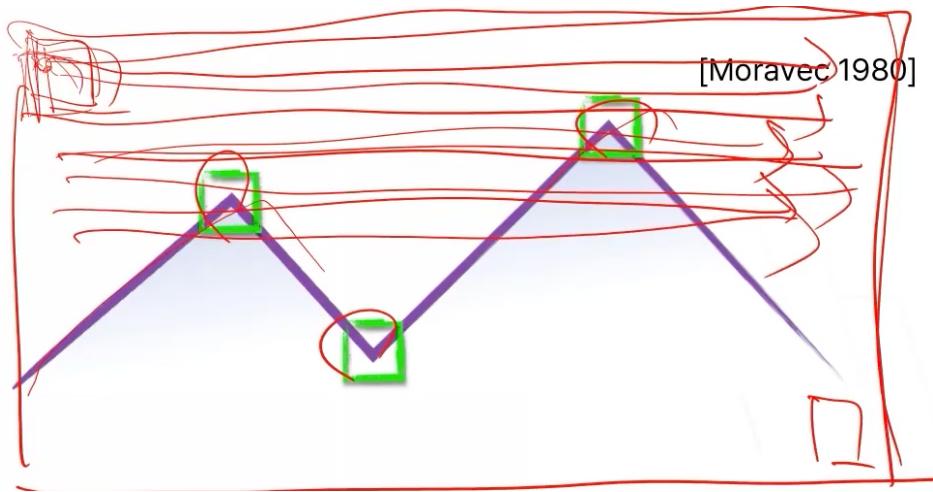
둘이 정확히 일치하는 점은 코너부분 밖에 없을 것임. (Edge가 아닌)

그래서 우리는 코너를 잡 뽑아야함

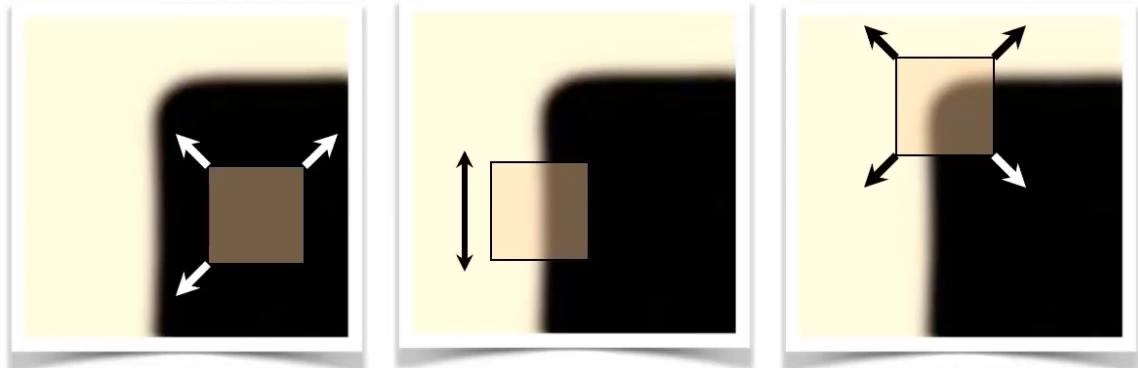
코너를 어떻게 찾아야 할까?



아래의 그림과 같이 패치간격으로 sliding window를 해서 corner를 찾는 방법이 있음



그럼 local한 corner를 찾을 수 있을 것임



**"flat"** region:  
no change in all  
directions

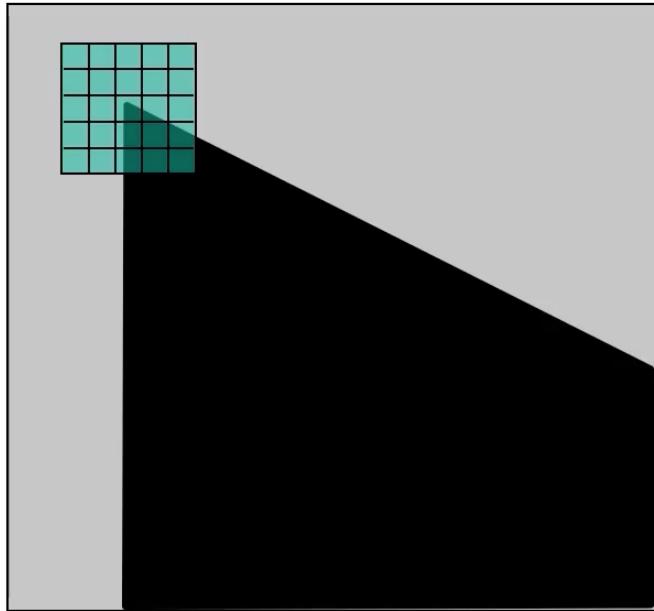
**"edge"**:  
no change along the  
edge direction

**"corner"**:  
significant change in  
all directions

변화량을 통해서 찾을수 있겠구나! 코너는 수평, 수직, 대각선의 변화량등 많은 변화량을 보고 찾아야 함

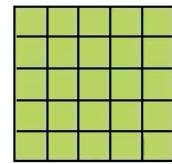
코너를 찾는 방법

## 1. Compute image gradients over small region



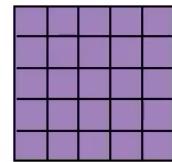
array of x gradients

$$I_x = \frac{\partial I}{\partial x}$$



array of y gradients

$$I_y = \frac{\partial I}{\partial y}$$



### ▼ 영상 처리 복기를 해보자

$$\nabla I = (I_x, I_y) = \left( \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right)$$

2D gradient of an image

$$||\nabla I(x, y)|| = \sqrt{I_x^2(x, y) + I_y^2(x, y)}$$

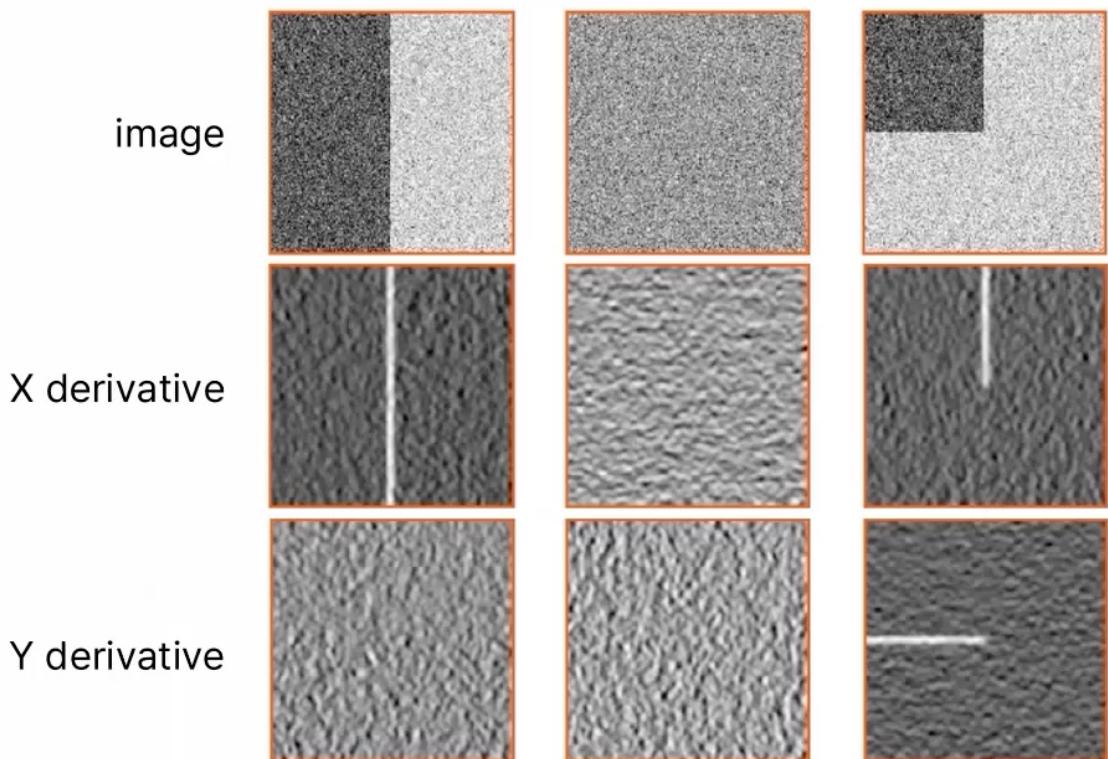
The gradient magnitude(edge strength)

$$\theta(x, y) = \tan^{-1} \left( \frac{I_y(x, y)}{I_x(x, y)} \right)$$

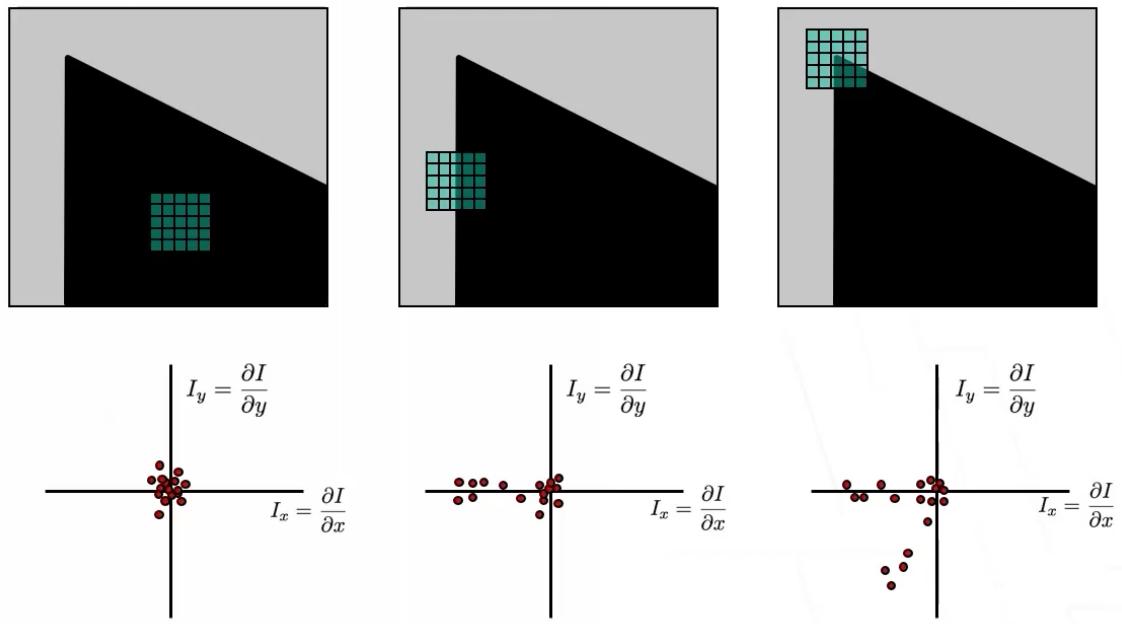
The gradient direction

물론 DoG와 LoG를 사용해도 됨 어떠한 필터를 사용하지는 중요하지 않음

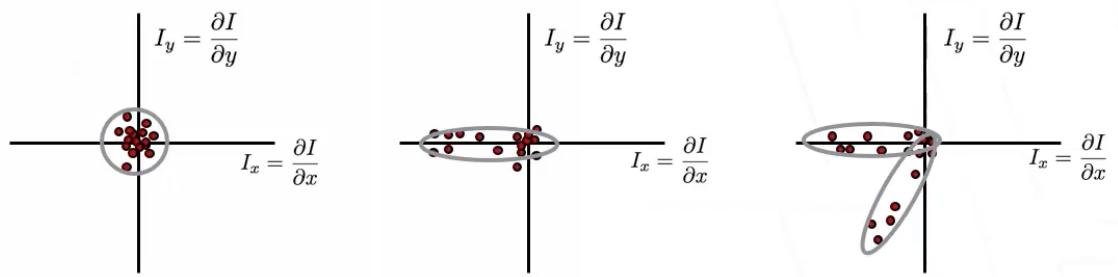
▼ Visualization of gradients



5by5 patch의 x변화량 y변화량을 그래프에 점을 찍어보면 아래와 같음

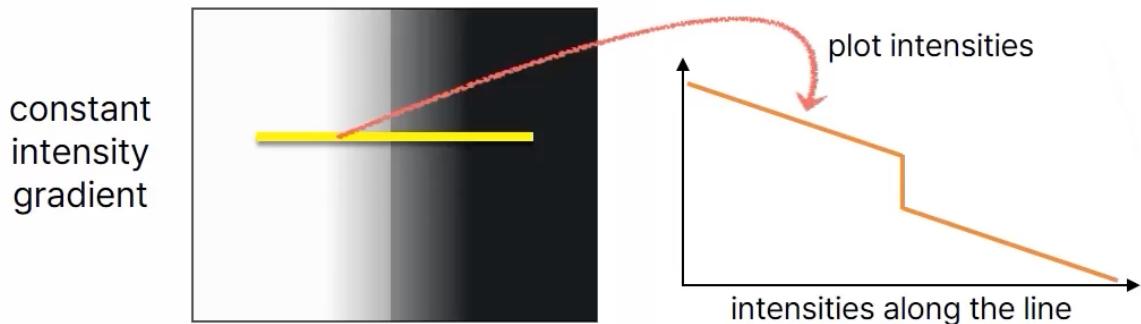


그래서 이 분포를 보면 edge와 비슷하게 생김

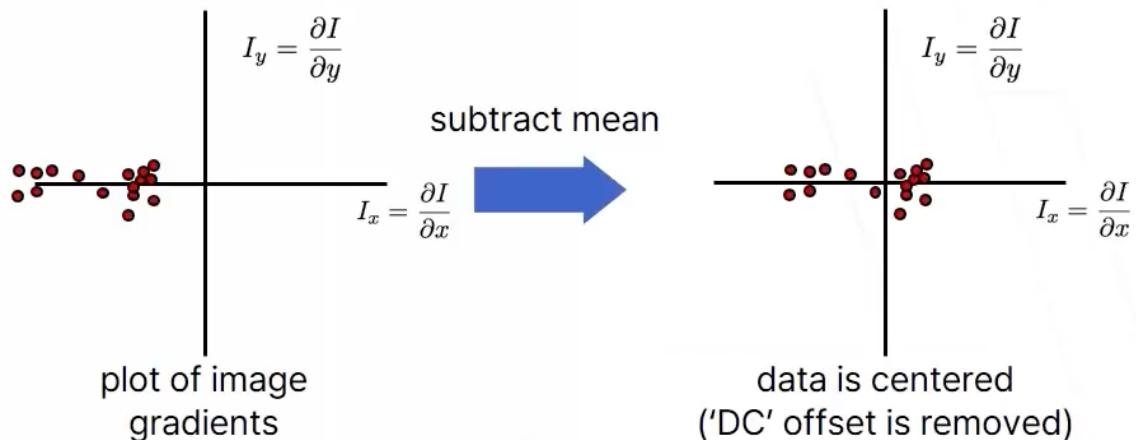


이것을 정량화 하면 됨

## 2. Subtract mean from each image gradient



이러한 분포도 평균을 빼주면 원점 중심으로 옮길 수 있음(Normalization)



## 3. Compute the covariance matrix

covariance matrix는 아래와 같이 정의함

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

이 네개의 값들을 계산해 줘야함

위의 patch를 예로 들면 P는 5by5, p는 25개의 픽셀중 하나를 의미함

$$\sum_{p \in P} I_x I_x = \text{sum} \left( \begin{array}{|c|c|c|c|c|} \hline & & & & \\ \hline \end{array} \right) \cdot * \left( \begin{array}{|c|c|c|c|c|} \hline & & & & \\ \hline \end{array} \right)$$

array of x gradients    array of y gradients

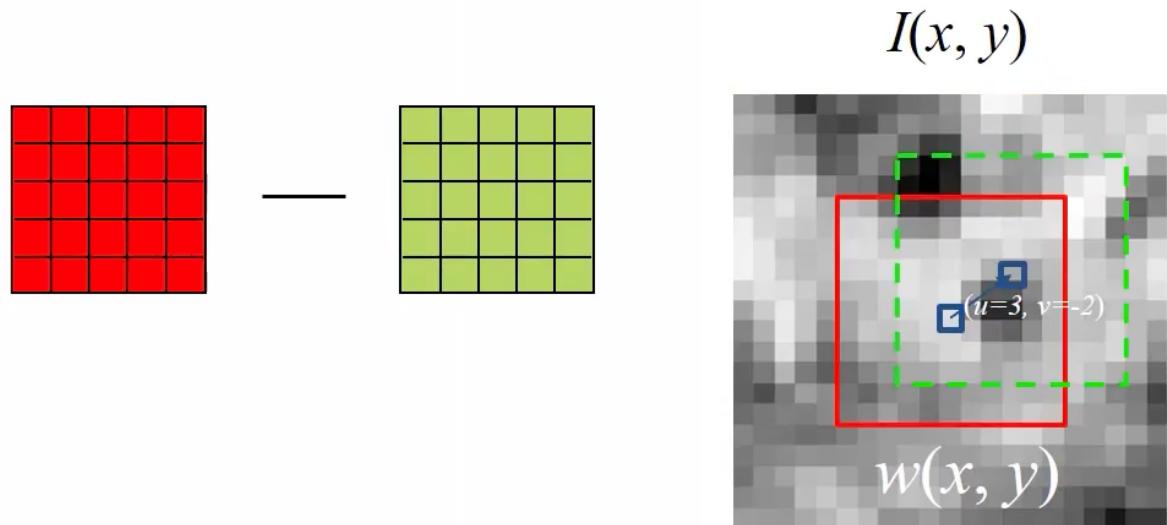
$$\sum_{p \in P} I_y I_y = \text{sum} \left( \begin{array}{|c|c|c|c|c|} \hline & & & & \\ \hline \end{array} \right) \cdot * \left( \begin{array}{|c|c|c|c|c|} \hline & & & & \\ \hline \end{array} \right)$$

array of x gradients    array of y gradients

이런식으로 구하면 됨

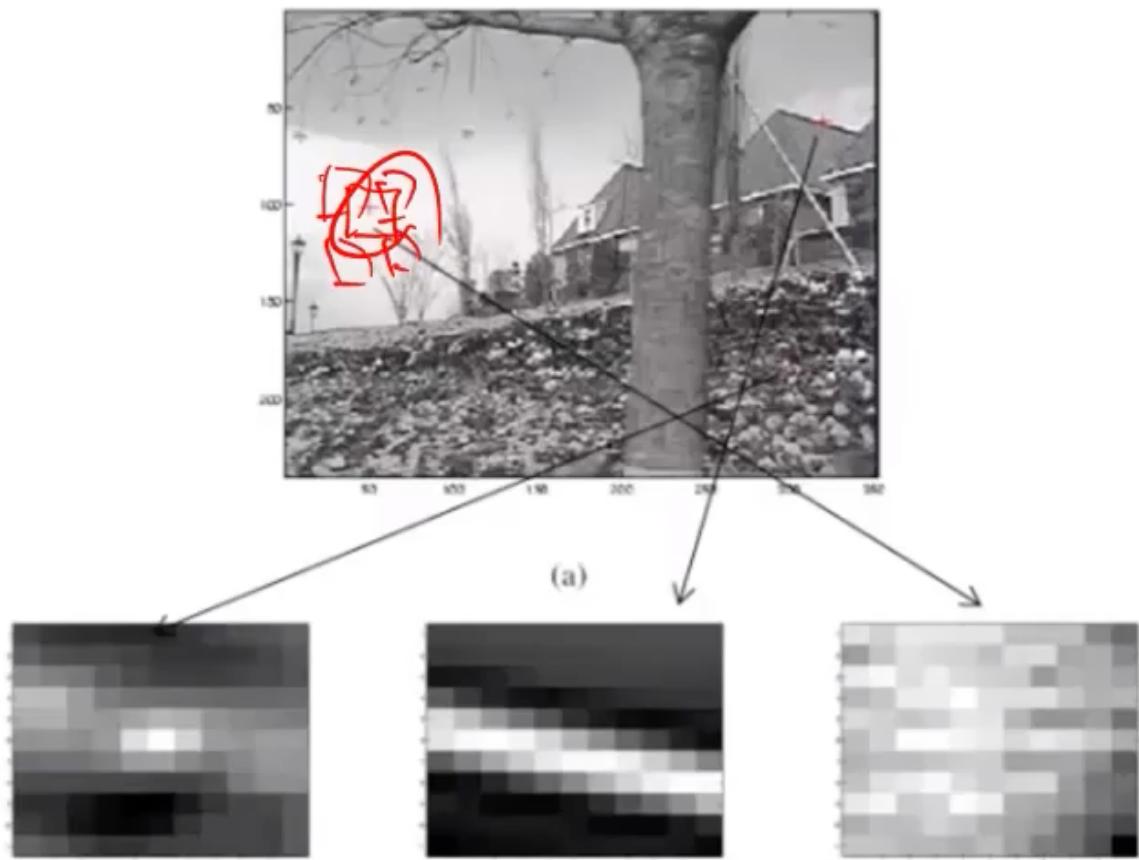
covariance matrix가 갑자기 왜 나왔나?

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



두 패치간의 차이를 보고 얼마나 많이 변했는지 알 수 있음 (3, -2)만큼 이동 시켰을때의 변화량

$w$ 는 각 패치의 중심을 중요하게 보게하기 위하여 weight를 주는 것임(가우시안)



아래 세개의 그림은  $E(u, v)$ 를 시각화 한건데. 값을 반전 시킨 값임. (a)를 예로 들어보자면 edge부분과 하늘 부분의 차이가 크기 때문에 어둡게 표시(반전시켰기 때문) 되었고 선끼리는 변화량이 적기 때문에 흰색으로 표시 됨.

이 식을 테일러 급수로 유도하면

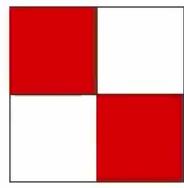
$$E(u, v) = \sum_{x,y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$$

$$\begin{aligned}
&\approx \sum [I(x, y) + I_x u + I_y v - I(x, y)]^2 \\
&= \sum I_x^2 u^2 + 2I_x I_y u v + I_y^2 v^2 \\
&= \sum [u \quad v] \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} [u \quad v]^T = [u \quad v] \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} [u \quad v]^T \\
&= \mathbf{u}^T M \mathbf{u}
\end{aligned}$$

$$M = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

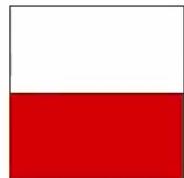
위와 같이 유도됨

몇가지 toy example을 확인해보자



$\sum I_x^2 \rightarrow$  Large  
 $\sum I_y^2 \rightarrow$  Large

Corner



$\sum I_x^2 \rightarrow$  Small  
 $\sum I_y^2 \rightarrow$  Large

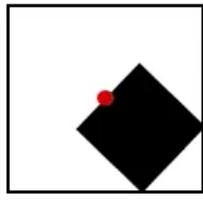
Edge



$\sum I_x^2 \rightarrow$  Small  
 $\sum I_y^2 \rightarrow$  Small

Nothing

이런 예제를 보면  $x, y$  변화량이 크면 corner구나! 라고 단순히 생각할수 있는데



$$M = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

이런 회전이 있는 도형을 x,y 값이 모두 높은데(대각선이기 때문에) 보면 corner가 아니라 edge임

따라서  $I_x \cdot I_y$  값을 봐주어야 함. 좀 더 효율적인 방법이 필요(eigenvalue)를 구해보자

#### 4. Compute eigenvectors and eigenvalues

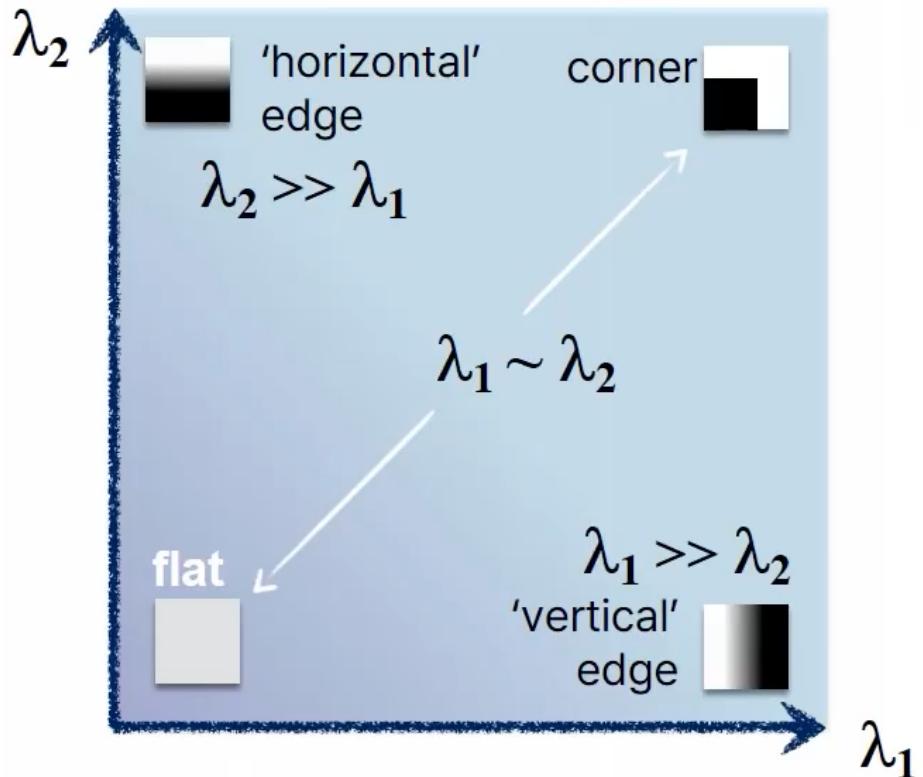
위의 M메트릭스를 eigenvalue, eigenvector로 나타낼 수 있음

eigenvalue  
↓  
 $M\mathbf{e} = \lambda\mathbf{e}$        $(M - \lambda I)\mathbf{e} = 0$   
↑  
eigenvector

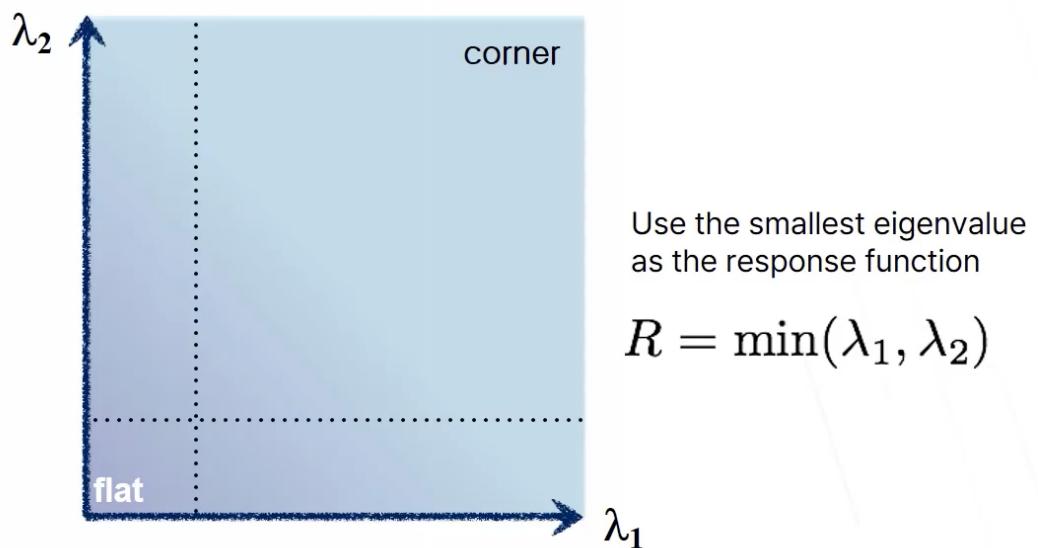
1. Compute the determinant of  
(returns a polynomial)  $M - \lambda I$

2. Find the roots of polynomial  
(returns eigenvalues)  $\det(M - \lambda I) = 0$

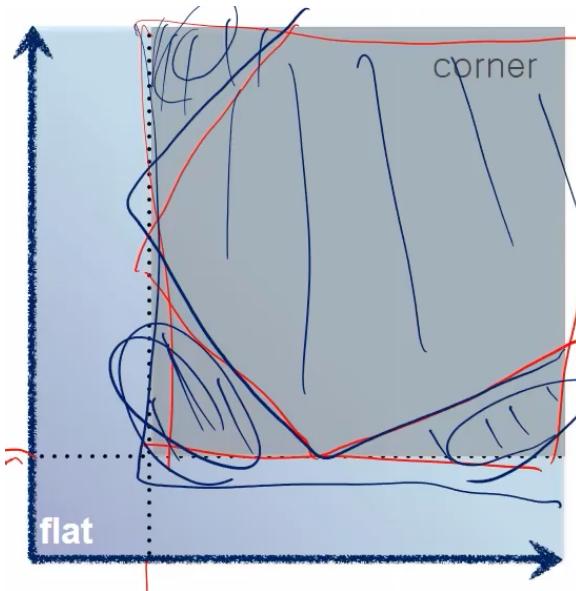
3. For each eigenvalue, solve  
(returns eigenvectors)  $(M - \lambda I)\mathbf{e} = 0$



$\lambda_1$ 이  $\lambda_2$ 보다 얼마나 커야 edge인지 이런 부분에 대해서 기준이 좀 애매하니 기준을 세워야 함

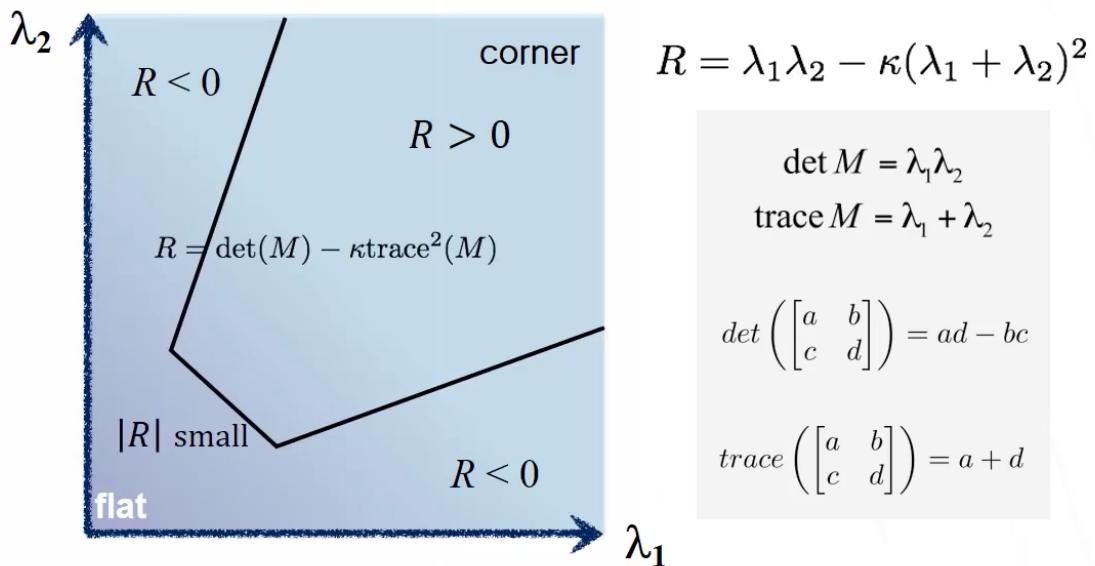


threshold(점선)을 기준으로 나눔



사실 edge부분을 corner라고 취급할수도 있게 되지만 어쩔수 없음

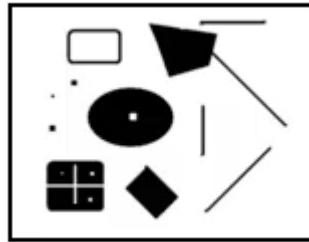
두번째 방법은



$\kappa$ 를 이용해  $\lambda$ 가 더 큰 쪽에 제곱을 해주어서 음수를 만들어 계산하는 건데 eigen value를 구할 필요가 없어 연산량 측면에서 첫번째 방법이 더 우수함. 하지만 정확도 측면에서는 첫 번째 방법이 더 좋다고 함

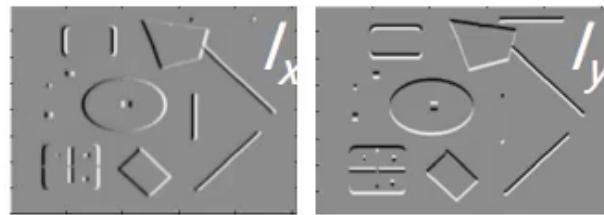
## 5. Use threshold on eigenvalues to detect corner

이 이미지를



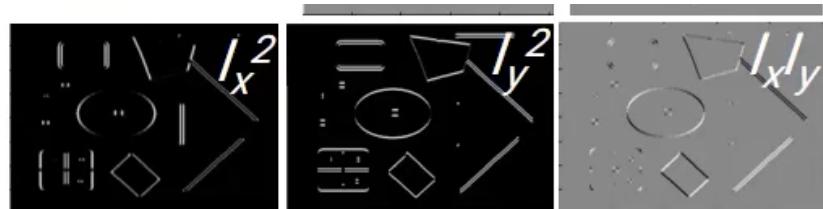
변화량을 구함

### ① Image derivatives (optionally, blur first)



그럼 이미지 맵을 구할수 있을 것임

### ② Square of derivatives



그다음 이 이미지 맵에 미리 가우시안 필터링을 진행한다면

### ③ Gaussian filter $g(\sigma_l)$



굳이 아래처럼 다 더해서 연산량이 늘어나게 M를 구하지 않아도

$$M = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

아래처럼 한번에 Corner를 구할수 있을것

$$M = \begin{bmatrix} g(I_x^2) & g(I_x I_y) \\ g(I_x I_y) & g(I_y^2) \end{bmatrix}$$

