Intro to

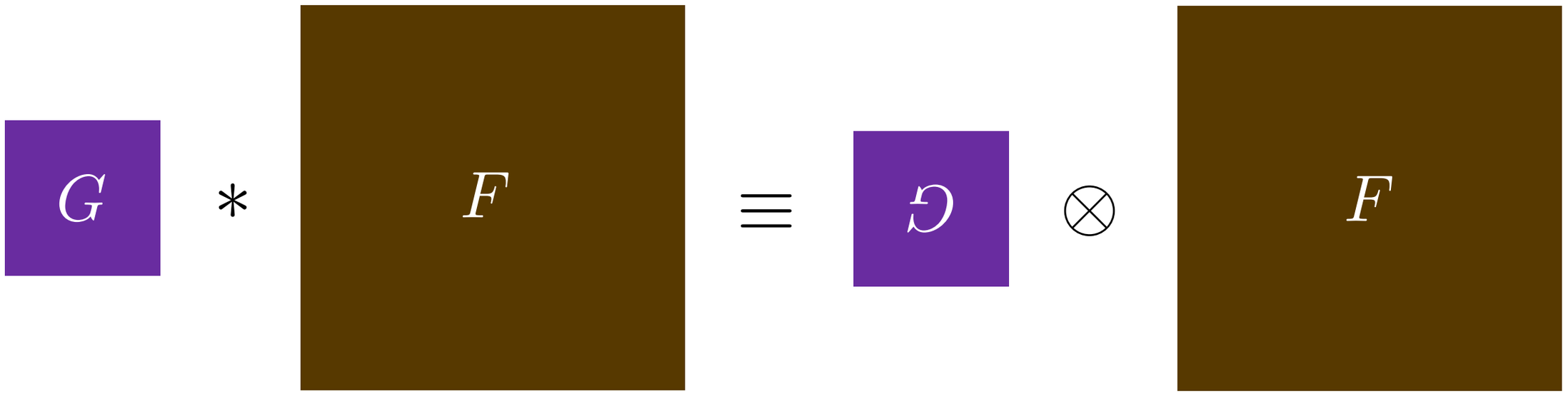
Computer Vision

with Prof. Kosta Derpanis

Edge detection

Correlation $H = G \otimes F$

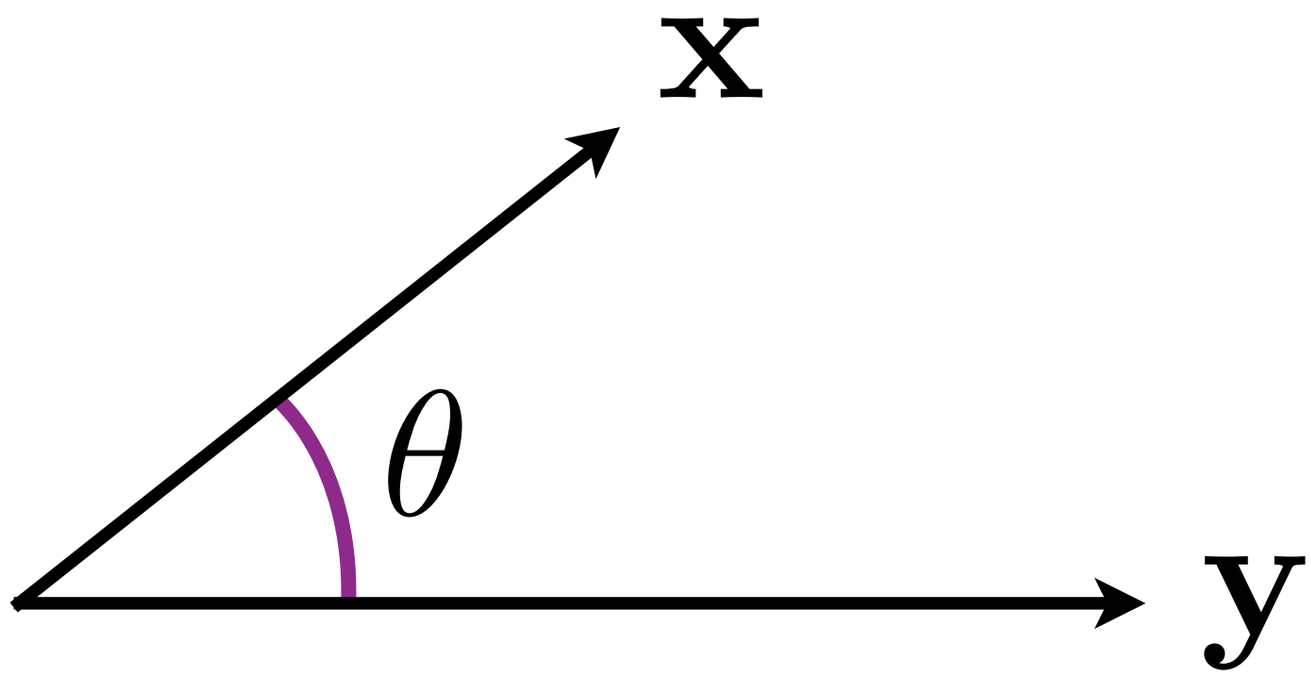
$$H[x, y] = \sum_{u=-K}^{K} \sum_{v=-K}^{K} G[u, v] F[x + u, y + v]$$

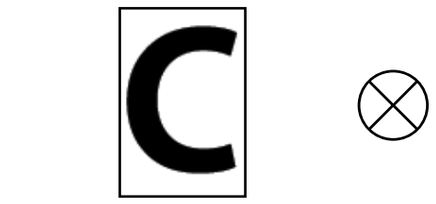


Convolution $H = G * F$

$$H[x, y] = \sum_{u=-K}^{K} \sum_{v=-K}^{K} G[u, v] F[x - u, y - v]$$

$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{N} x_i y_i = \|\mathbf{x}\| \|\mathbf{y}\| \cos\theta$$





template

search image

n

correlation map

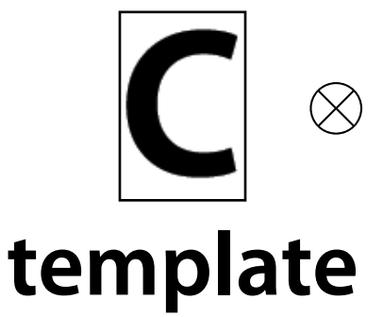
n

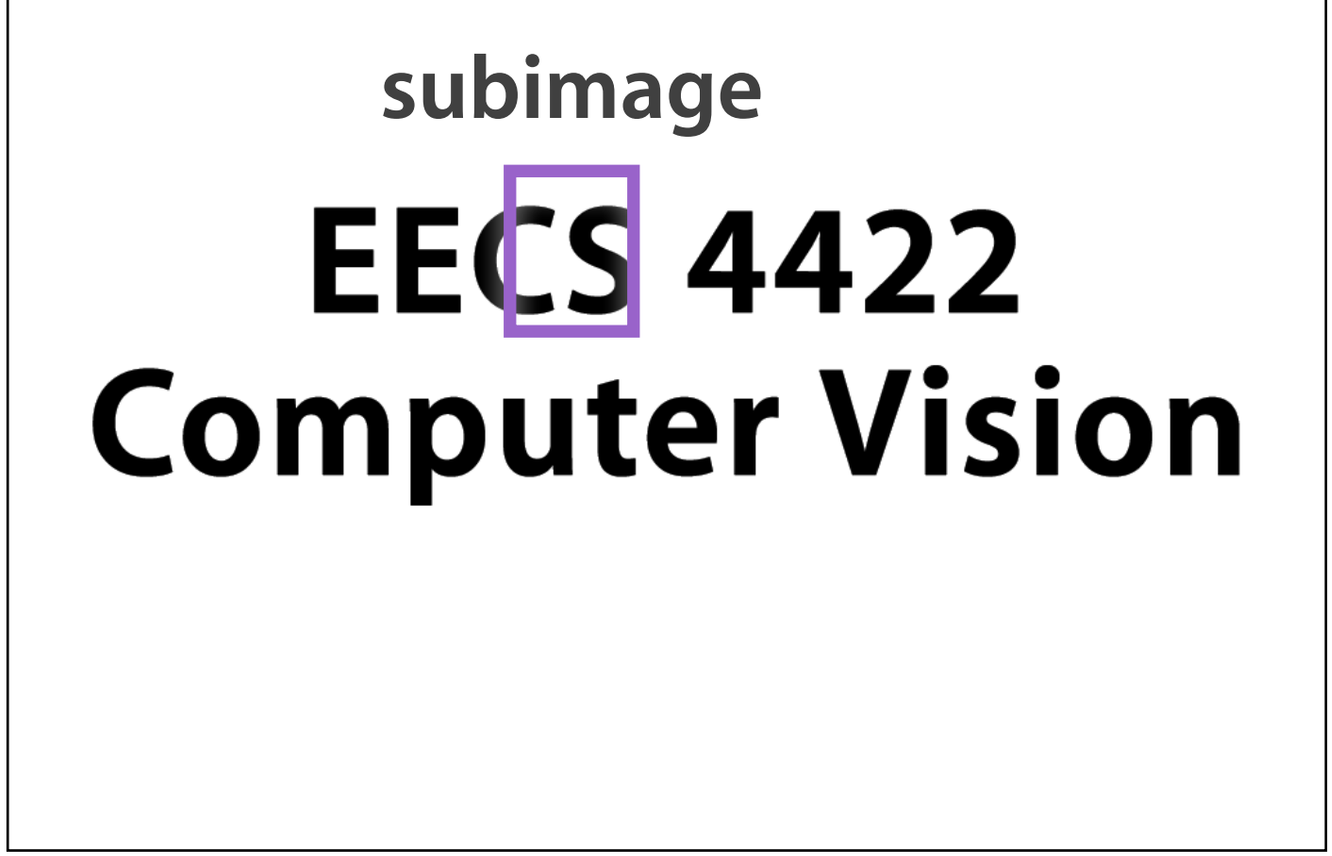
=

EECS 4422

Computer Vision

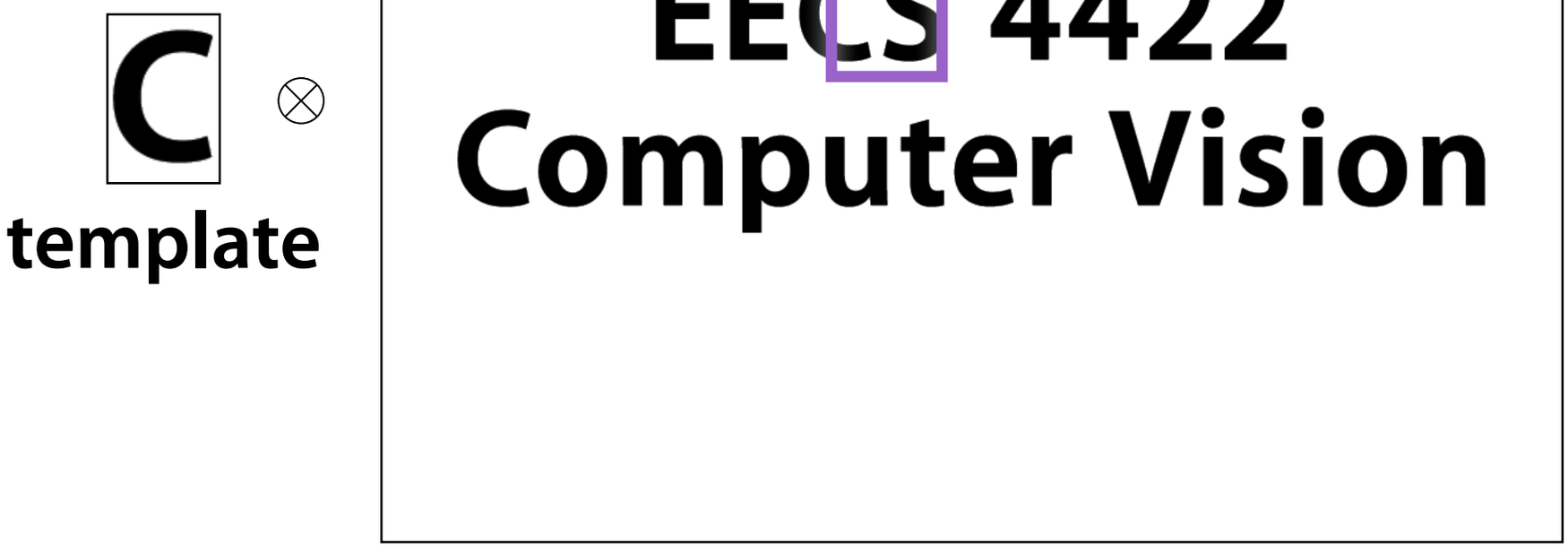
detected templates





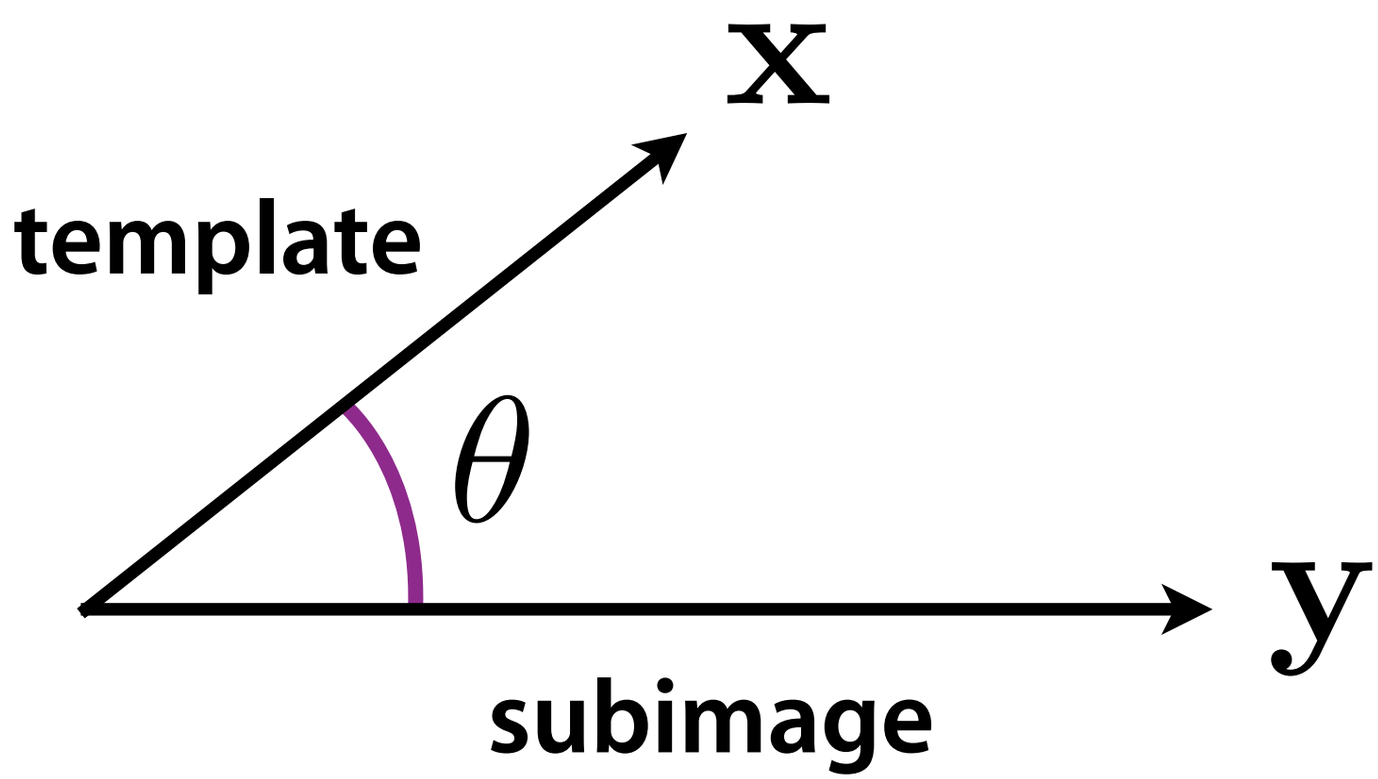
correlation score $= \mathrm{t} \cdot \mathrm{s}$

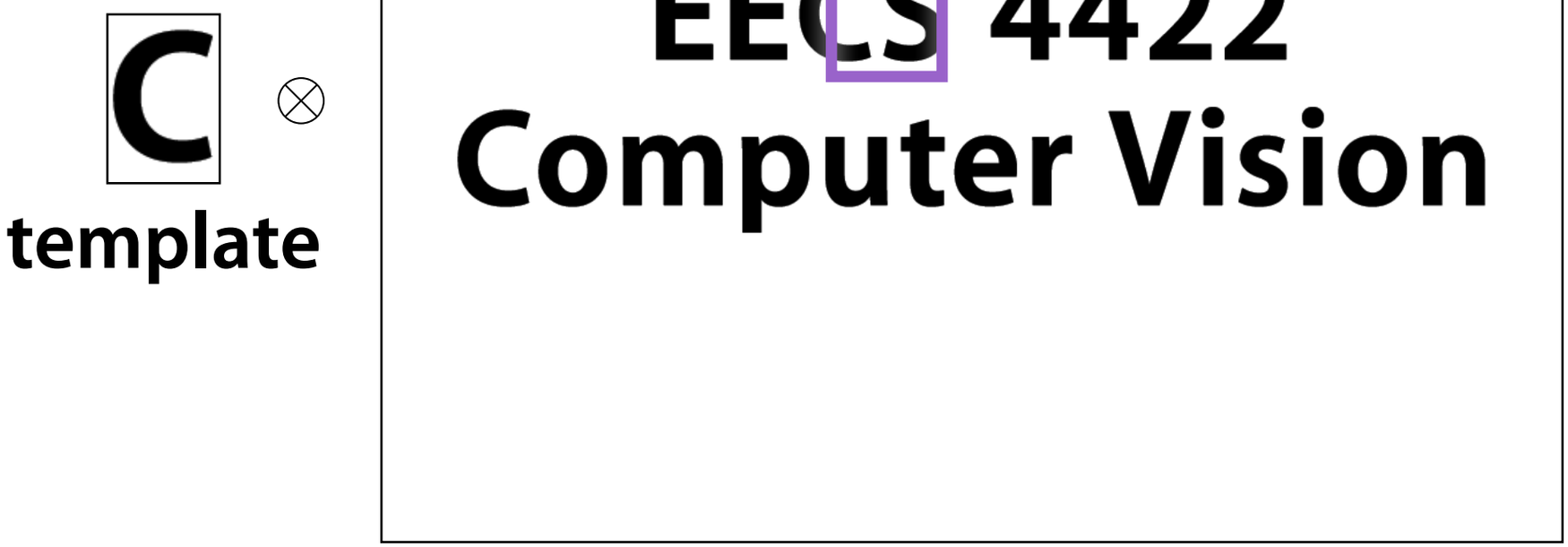
scalar product between vectors



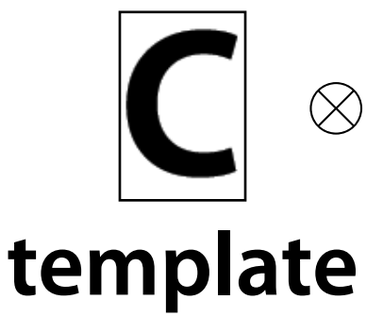
correlation score $= \mathbf{t} \cdot \mathbf{s} = \|\mathbf{t}\| \|\mathbf{s}\| \cos\theta$

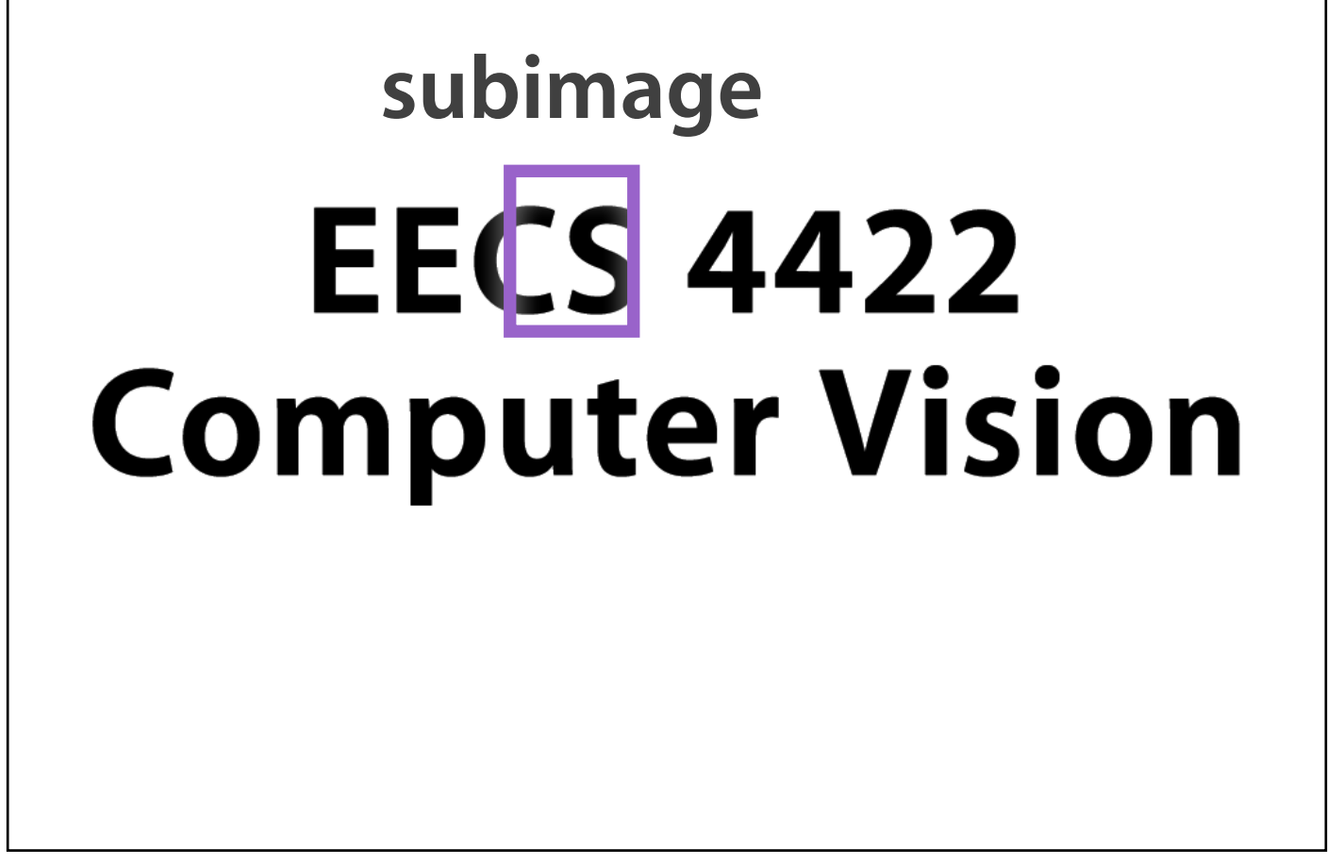
$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{N} x_i y_i = \|\mathbf{x}\| \|\mathbf{y}\| \cos\theta$$





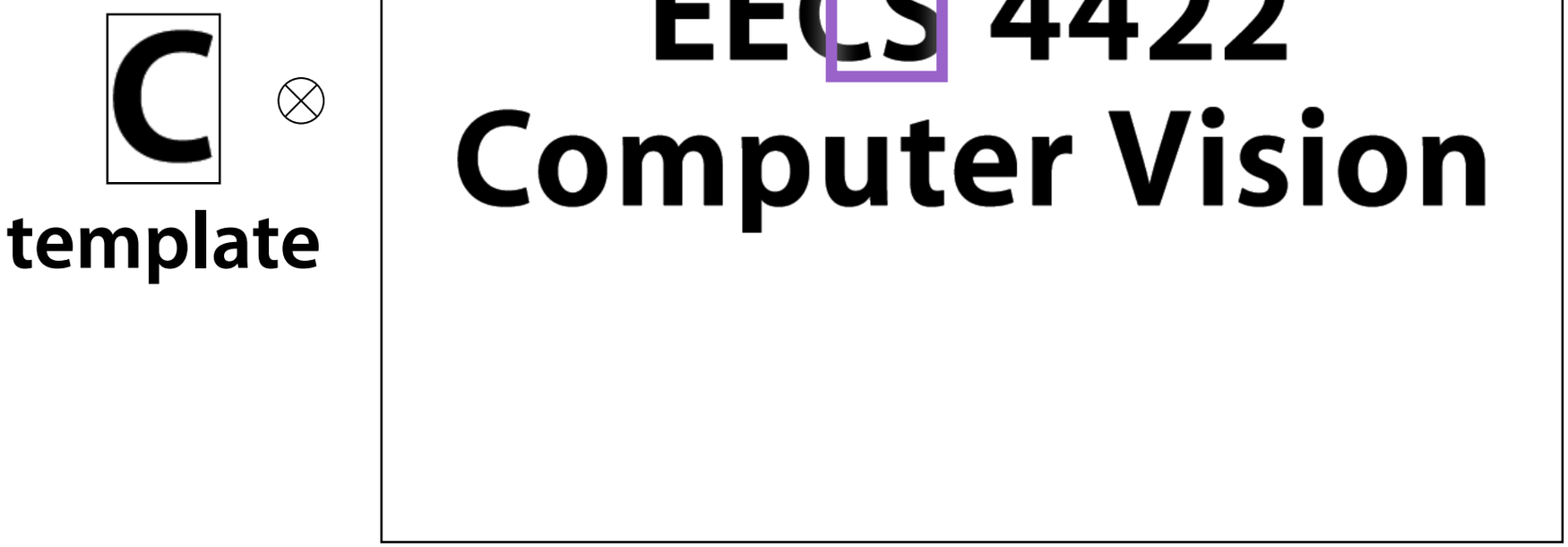
correlation score $= \mathbf{t} \cdot \mathbf{s} = \|\mathbf{t}\| \|\mathbf{s}\| \cos\theta$





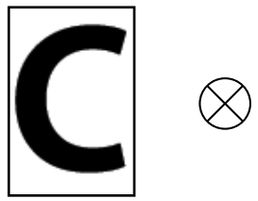
$$\text{correlation score} = \mathbf{t} \cdot \mathbf{s} = \|\mathbf{t}\| \|\mathbf{s}\| \cos\theta$$

score is sensitive to multiplicative brightness scaling

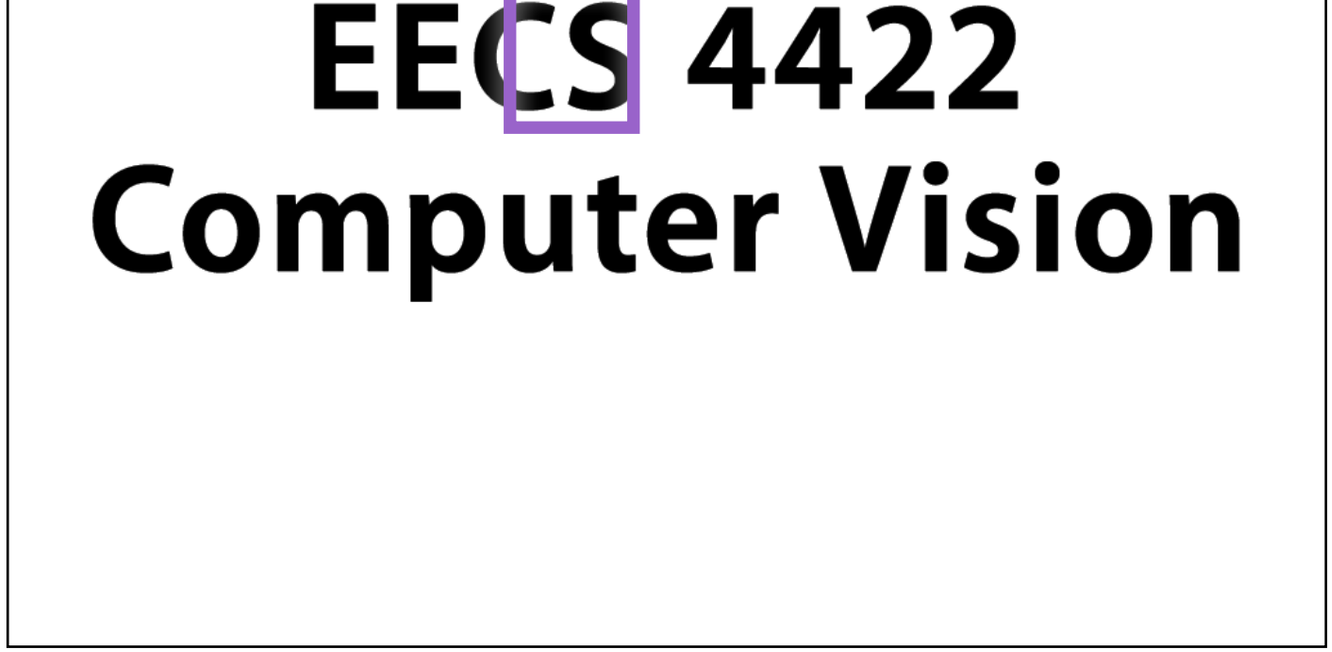


$$\textbf{correlation score} = \mathbf{t} \cdot \mathbf{s} = \|\mathbf{t}\| \|\mathbf{s}\| \cos\theta$$

$$\textbf{normalized score} = \frac{\mathbf{t}}{\|\mathbf{t}\|} \cdot \frac{\mathbf{s}}{\|\mathbf{s}\|} = \cos\theta$$



template

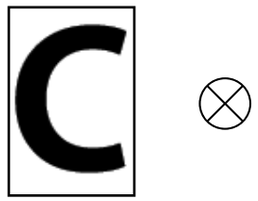


search image

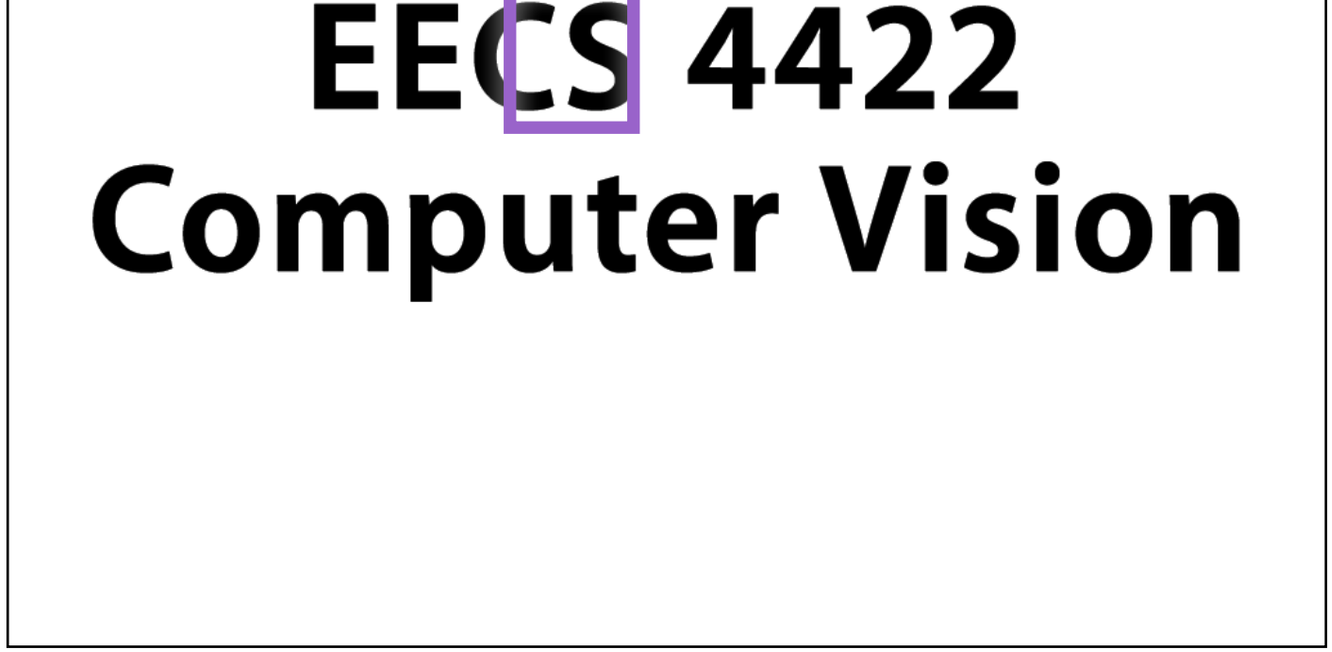
$$\text{correlation score} = \mathbf{t} \cdot \mathbf{s} = \|\mathbf{t}\| \|\mathbf{s}\| \cos\theta$$

$$\text{normalized score} = \frac{\mathbf{t}}{\|\mathbf{t}\|} \cdot \frac{\mathbf{s}}{\|\mathbf{s}\|} = \cos\theta$$

How can we make it invariant to multiplicative and additive brightness change?



template

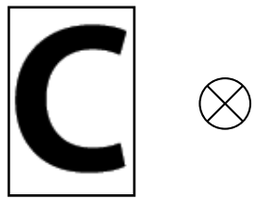


search image

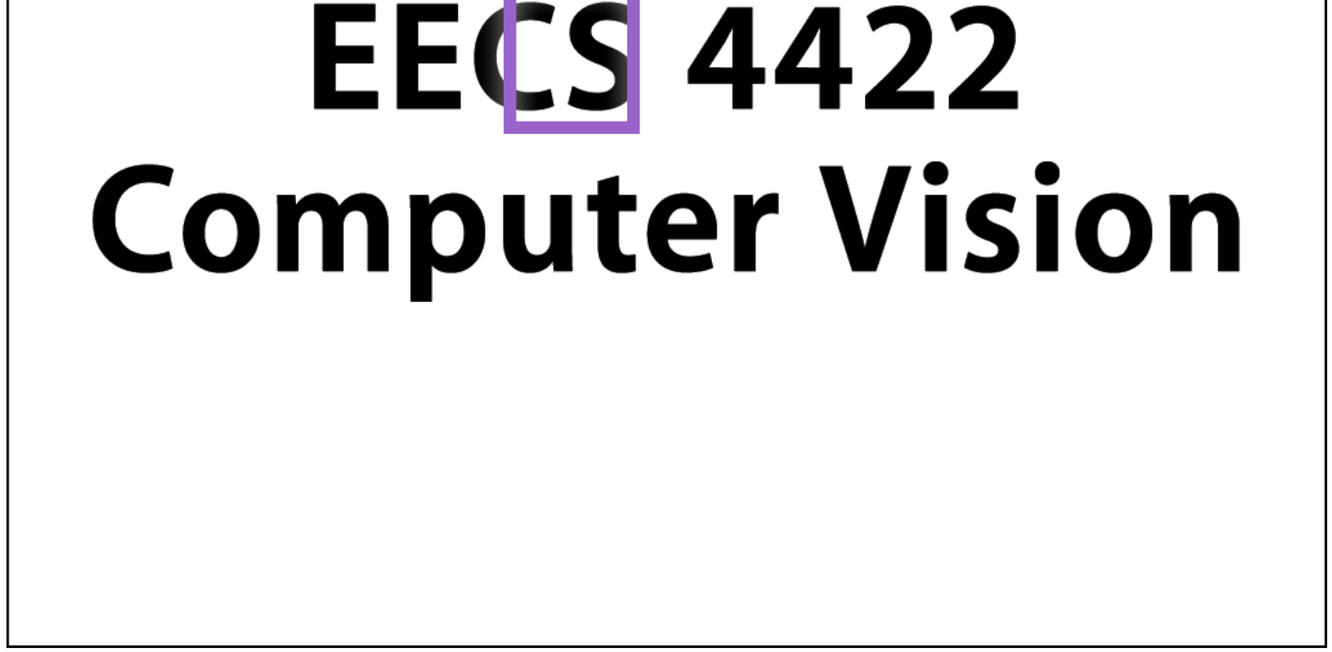
$$\text{correlation score} = \mathbf{t} \cdot \mathbf{s} = \|\mathbf{t}\| \|\mathbf{s}\| \cos\theta$$

$$\text{normalized score} = \frac{\mathbf{t}}{\|\mathbf{t}\|} \cdot \frac{\mathbf{s}}{\|\mathbf{s}\|} = \cos\theta$$

$$\text{normalized cross-correlation} = \frac{(\mathbf{t} - \bar{\mathbf{t}})}{\|(\mathbf{t} - \bar{\mathbf{t}})\|} \cdot \frac{(\mathbf{s} - \bar{\mathbf{s}})}{\|(\mathbf{s} - \bar{\mathbf{s}})\|}$$



template



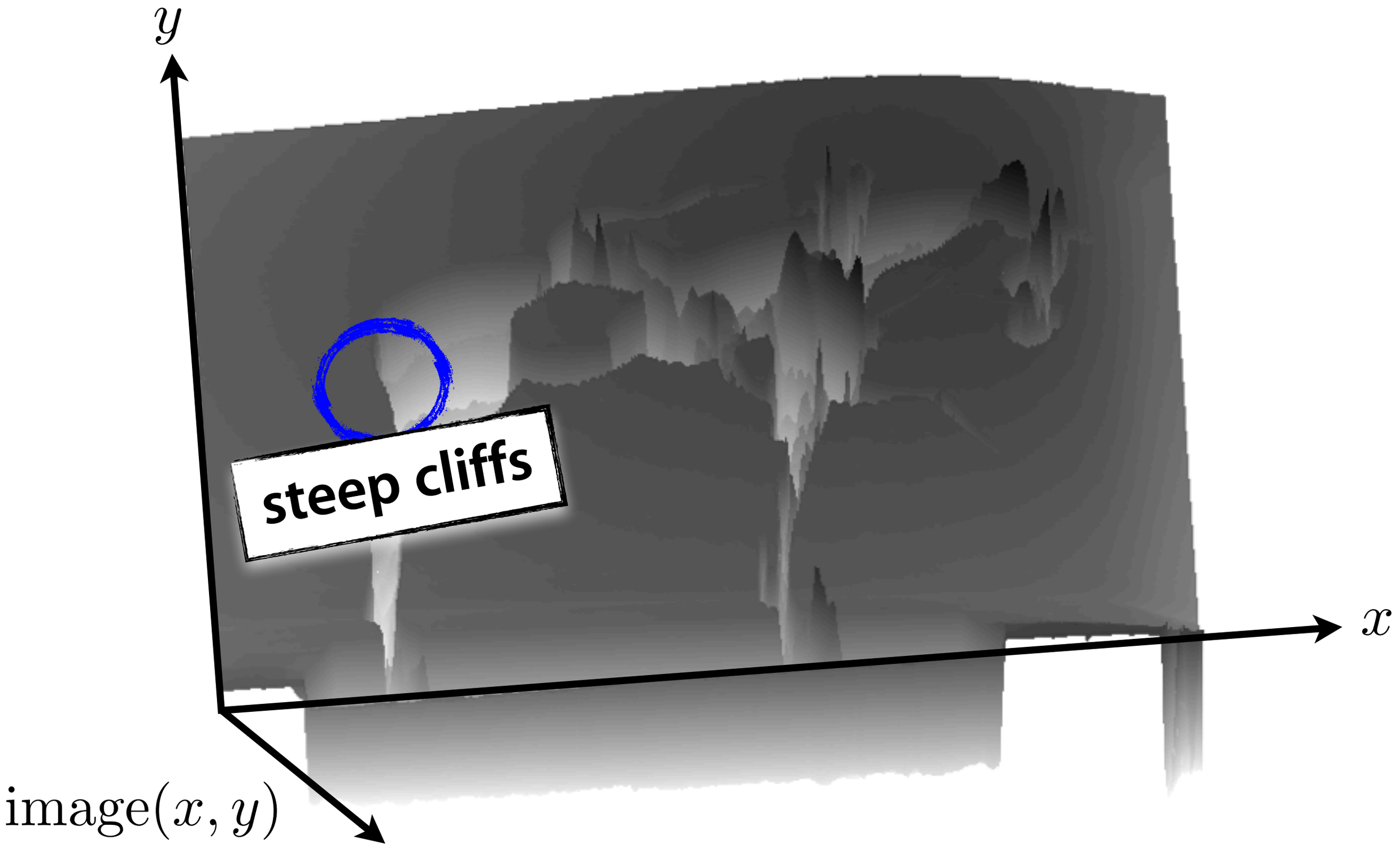
search image

$$\text{correlation score} = \mathbf{t} \cdot \mathbf{s} = \|\mathbf{t}\|\|\mathbf{s}\| \cos\theta$$

$$\text{normalized score} = \frac{\mathbf{t}}{\|\mathbf{t}\|} \cdot \frac{\mathbf{s}}{\|\mathbf{s}\|} = \cos\theta$$

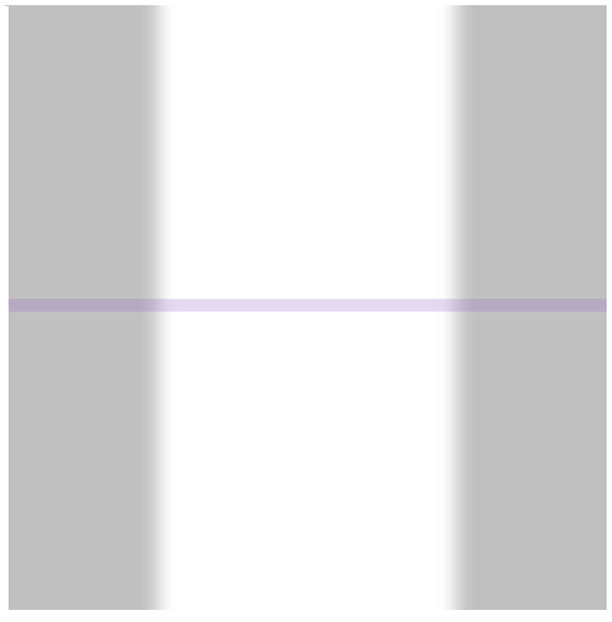
$$\text{normalized cross-correlation} = \frac{(\mathbf{t} - \bar{\mathbf{t}})}{\|(\mathbf{t}} \qquad (\mathbf{s} - \bar{\mathbf{s}})$$

average brightness

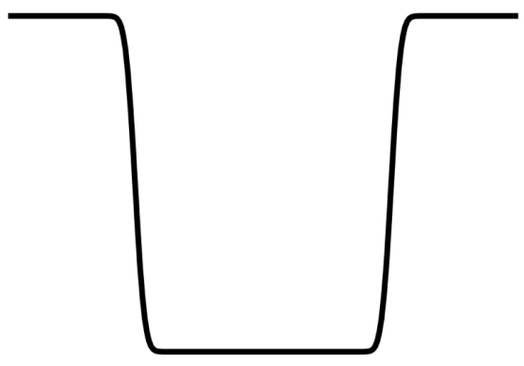

y
steep cliffs
x
image(x, y)

Goal: **Identify large local changes in the image**

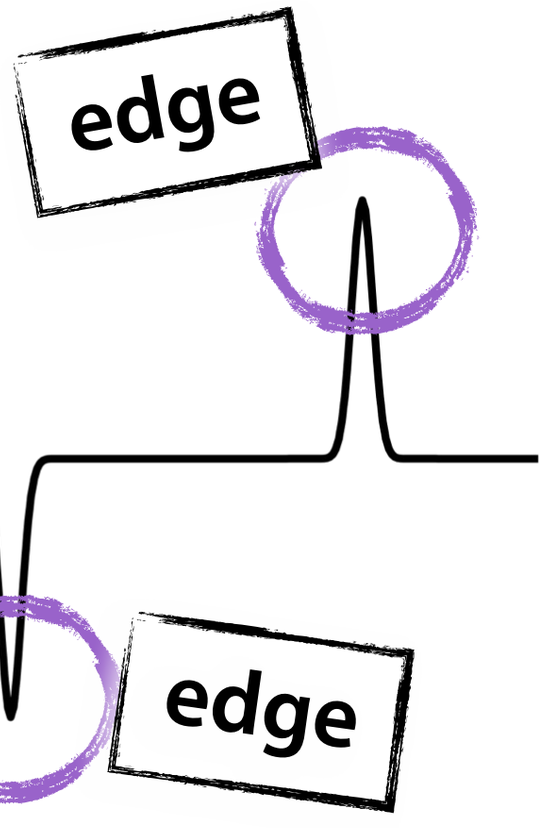
Edge Detection



image

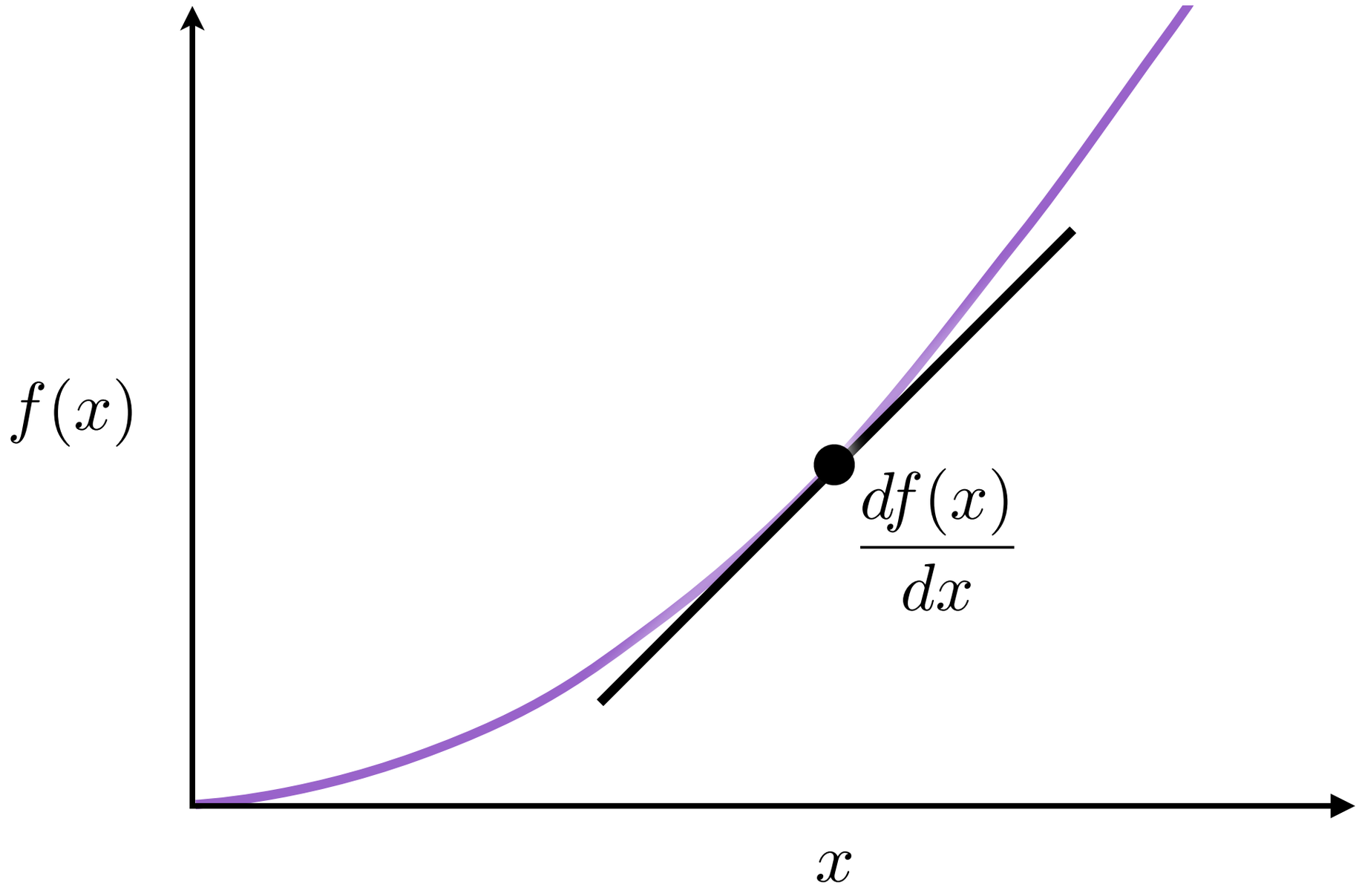


intensity function (slice)



first derivative

$$\frac{df(x)}{dx} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$



Partial Derivatives

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon, y) - f(x,y)}{\epsilon}$$

$$\frac{\partial f(x,y)}{\partial y} = \lim_{\epsilon \to 0} \frac{f(x, y+\epsilon) - f(x,y)}{\epsilon}$$

Discrete Derivatives

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y) - f(x,y)}{1}$$

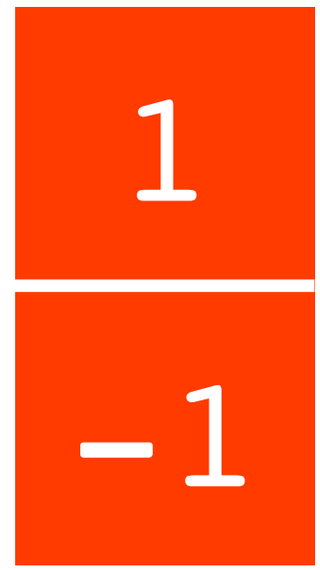
$$= f(x+1,y) - f(x,y)$$

$$\frac{\partial f(x,y)}{\partial x} \approx f(x+1,y) - f(x,y)$$

$$\frac{\partial f(x,y)}{\partial y} \approx f(x,y+1) - f(x,y)$$

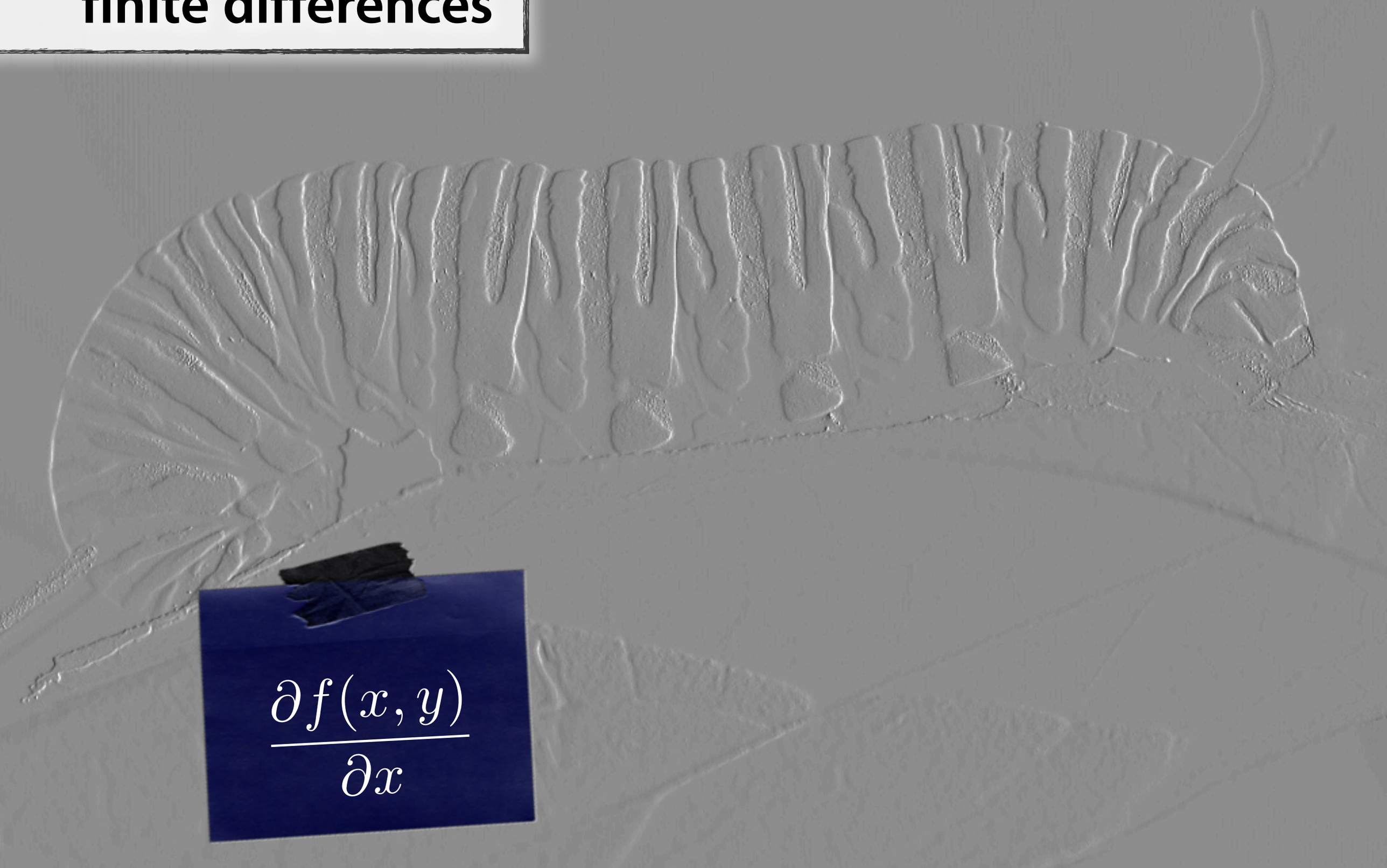
What are the associated convolution filters?

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



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

finite differences



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

finite differences

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

separable filters

$$F[x, y] = U[x]V[y]$$

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

=

1	
0	1 2 1
-1	

Sobel Filter

1	0	−1
2	0	−2
1	0	−1

1	2	1
0	0	0
−1	−2	−1

Sobel Filter

Other Derivative Filters

1	0	−1
2	0	−2
1	0	−1

1	2	1
0	0	0
−1	−2	−1

Sobel Filter

1	0	−1
1	0	−1
1	0	−1

1	1	1
0	0	0
−1	−1	−1

Prewitt

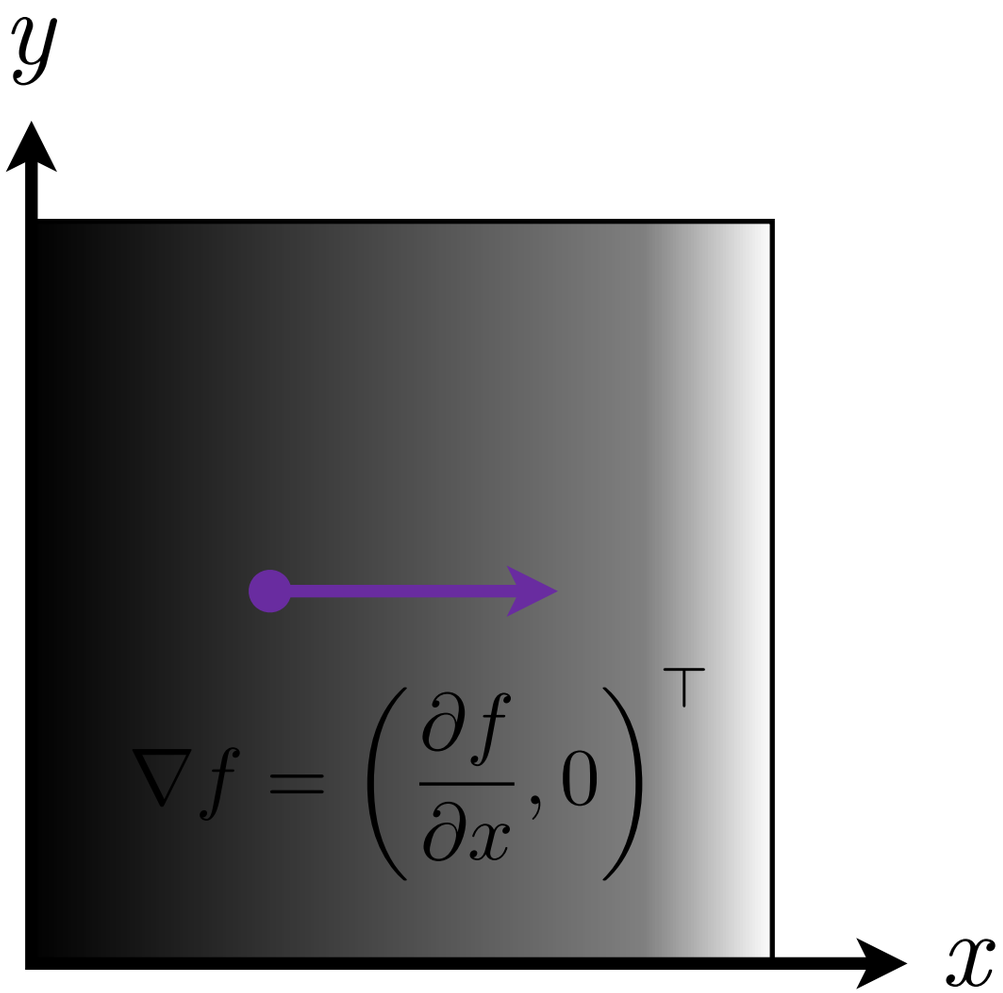
0	−1
1	0

−1	0
0	1

Roberts Cross

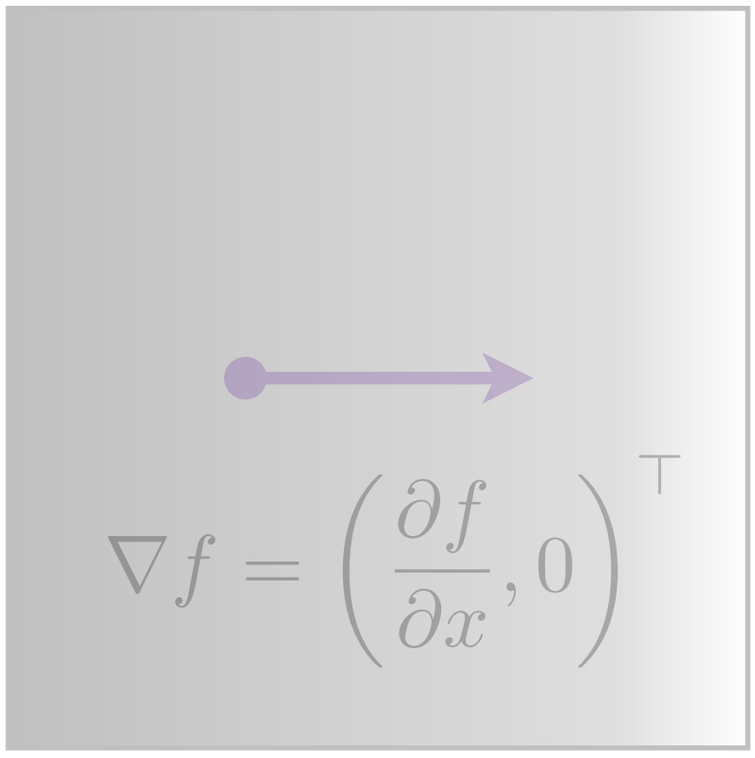
gradient

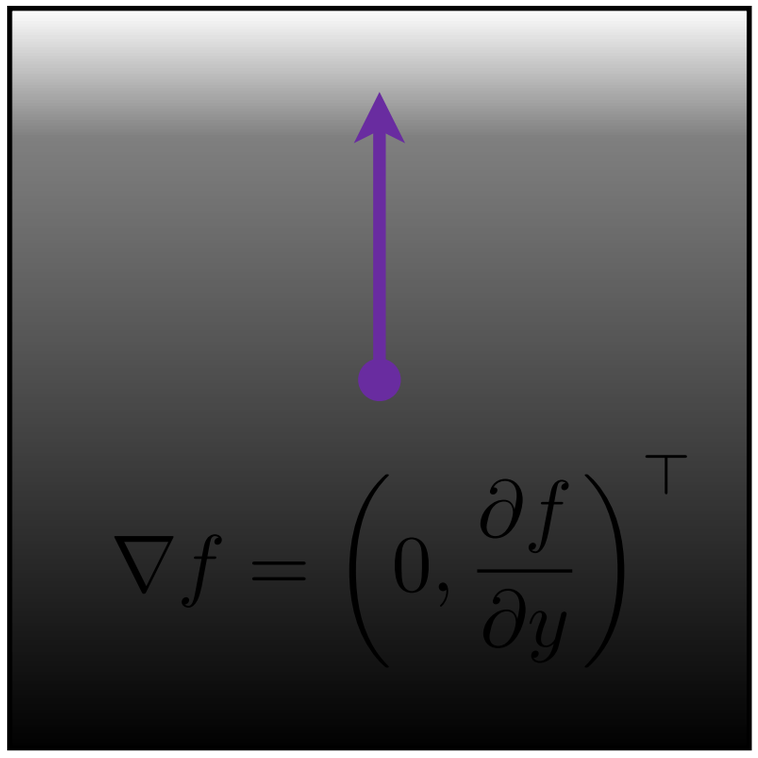
vector that points in the direction of the greatest rate of increase of a function and whose magnitude is that rate of increase



gradient

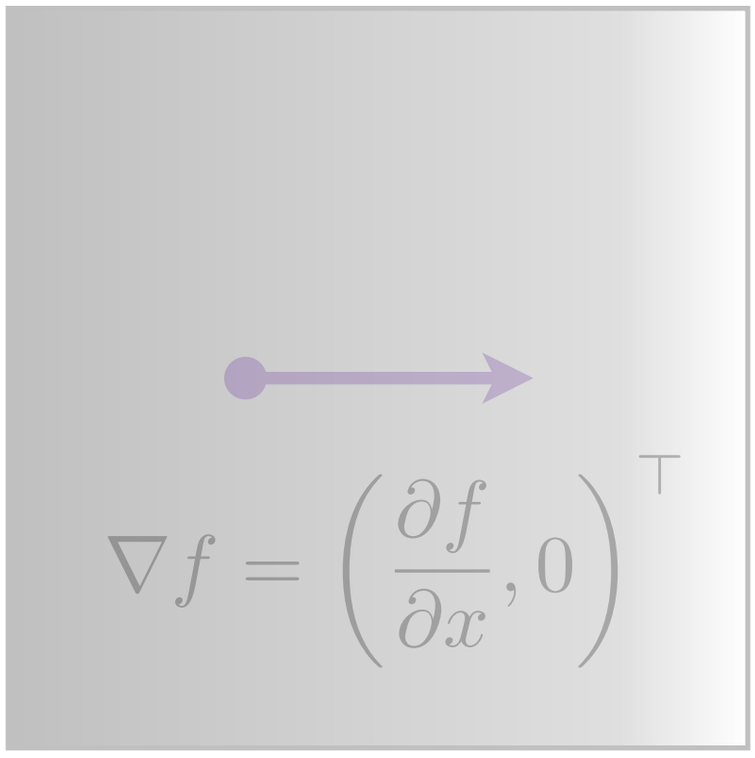
vector that points in the direction of the greatest rate of increase of a function and whose magnitude is that rate of increase

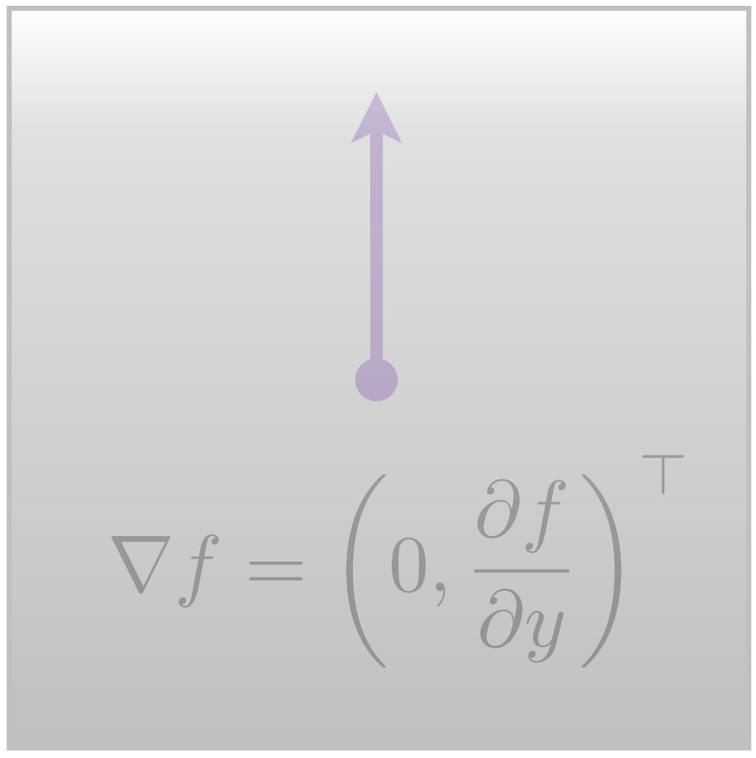


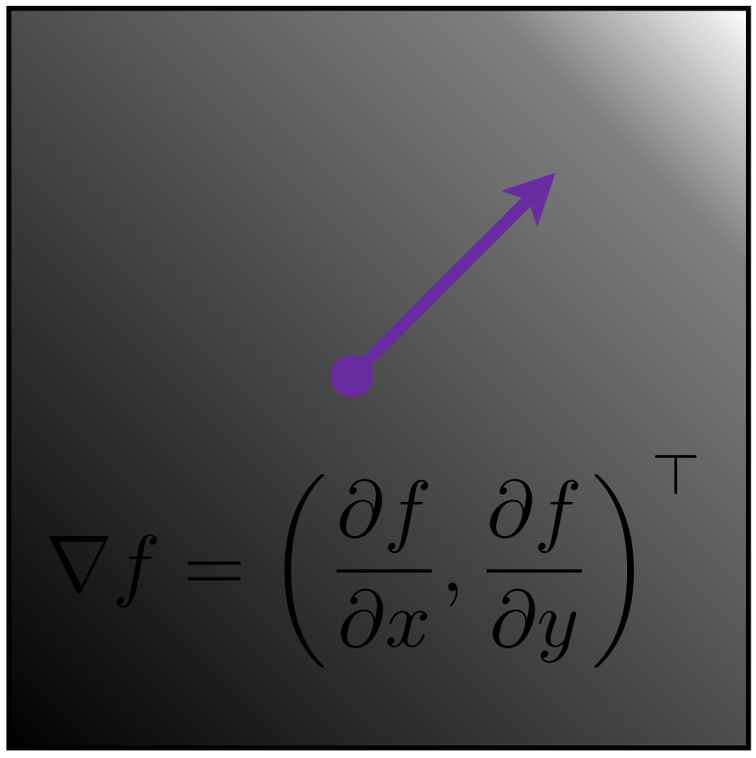


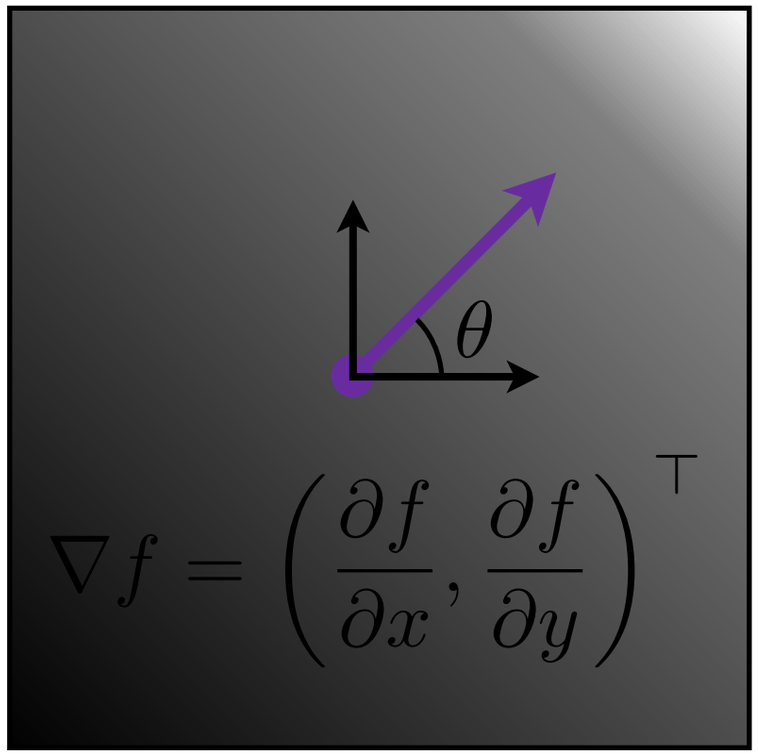
gradient

vector that points in the direction of the greatest rate of increase of a function and whose magnitude is that rate of increase





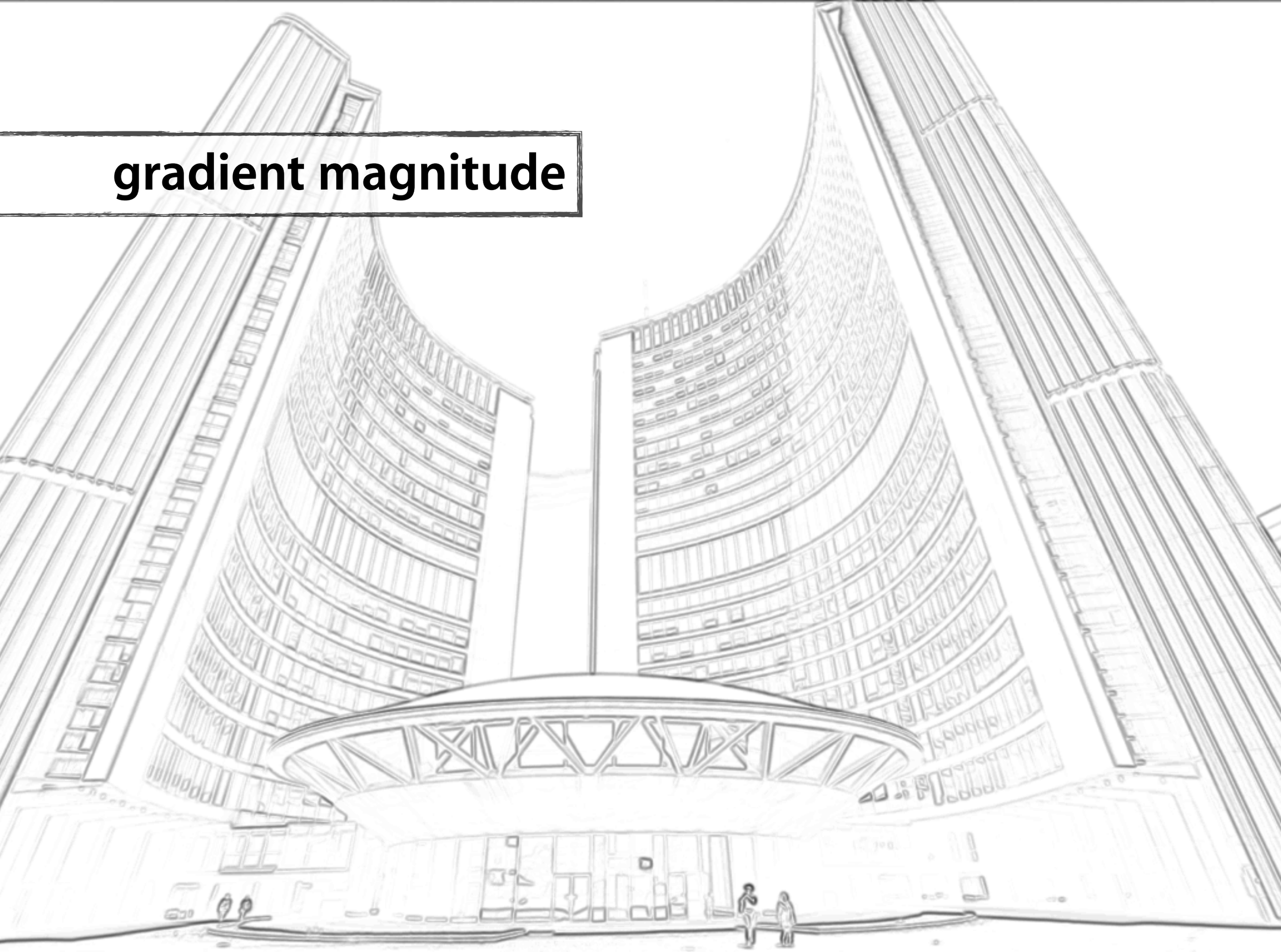


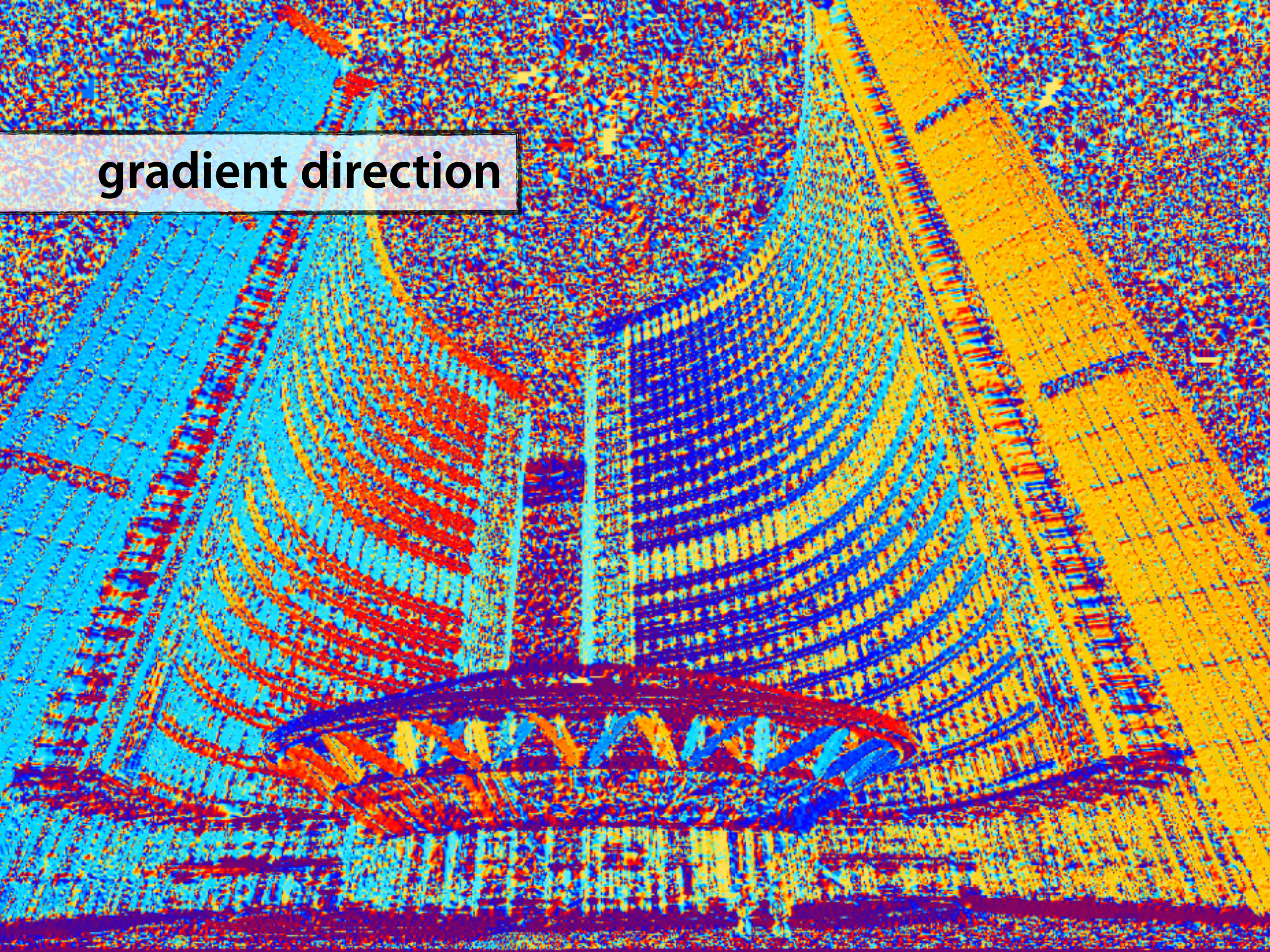


Gradient magnitude $\|\nabla f\| = \sqrt{\frac{\partial f}{\partial x}^2 + \frac{\partial f}{\partial y}^2}$

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

gradient magnitude



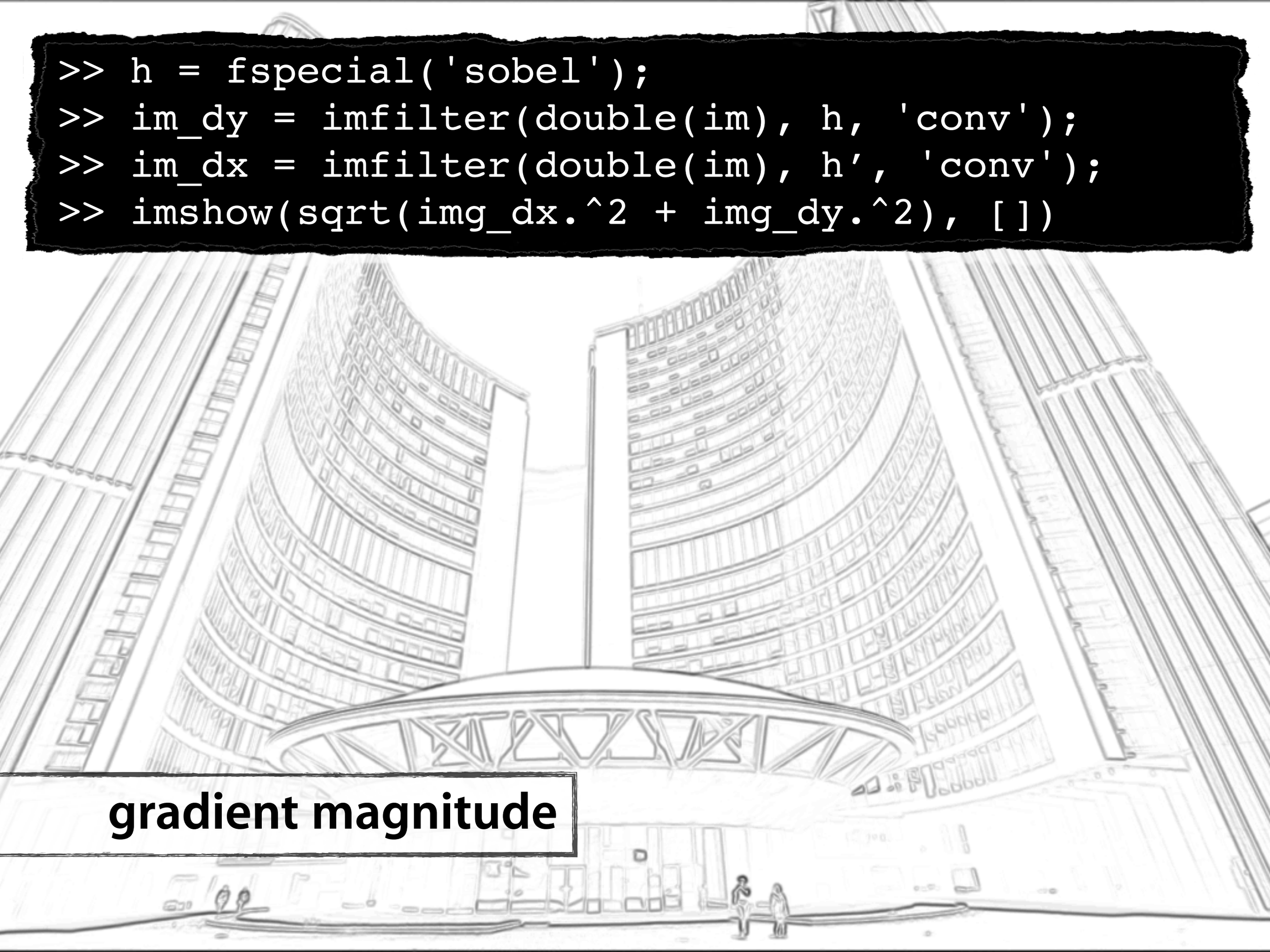

gradient direction

```
>> h = fspecial('sobel');
>> im_dy = imfilter(double(im), h, 'conv');
>> imshow(im_dy, [])
```

```
>> h = fspecial('sobel');
>> im_dy = imfilter(double(im), h, 'conv');
>> imshow(im_dy, [])
```

correlation is default

```
>> h = fspecial('sobel');
>> im_dy = imfilter(double(im), h, 'conv');
>> im_dx = imfilter(double(im), h', 'conv');
>> imshow(sqrt(img_dx.^2 + img_dy.^2), [])
```



gradient magnitude

```
>> h = fspecial('sobel');
>> im_dy = imfilter(double(im), h, 'conv');
>> im_dx = imfilter(double(im), h', 'conv');
>> imshow(sqrt(img_dx.^2 + img_
```

transpose

gradient magnitude

```
>> h = fspecial('sobel');
>> im_dy = imfilter(double(im), h, 'conv');
>> im_dx = imfilter(double(im), h', 'conv');
>> imshow(sqrt(img_dx.^2 + img_dy.^2), [])
```

pointwise product

gradient magnitude

```
>> h = fspecial('sobel');
>> im_dy = imfilter(double(im), h, 'conv');
>> im_dx = imfilter(double(im), h', 'conv');
>> imshow(atan2(im_dy, im_dx), [])
>> colormap jet
```

gradient direction

```
>> h = fspecial('sobel');
>> im_dy = imfilter(double(im), h, 'conv');
>> im_dx = imfilter(double(im), h', 'conv');
>> imshow(atan2(im_dy, im_dx), [])
>> colormap jet
```

change output colour space

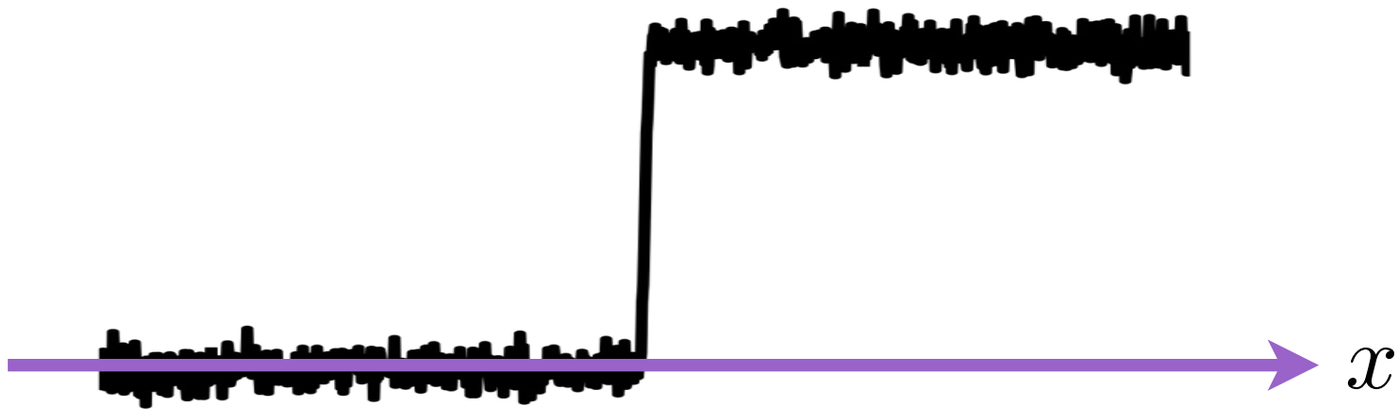
gradient direction

```
>> h = fspecial('sobel');
>> im_dy = imfilter(double(im), h, 'conv');
>> im_dx = imfilter(double(im), h', 'conv');
>> imshow(atan2(im_dy, im_dx), [])
>> colormap jet
```

gradient direction

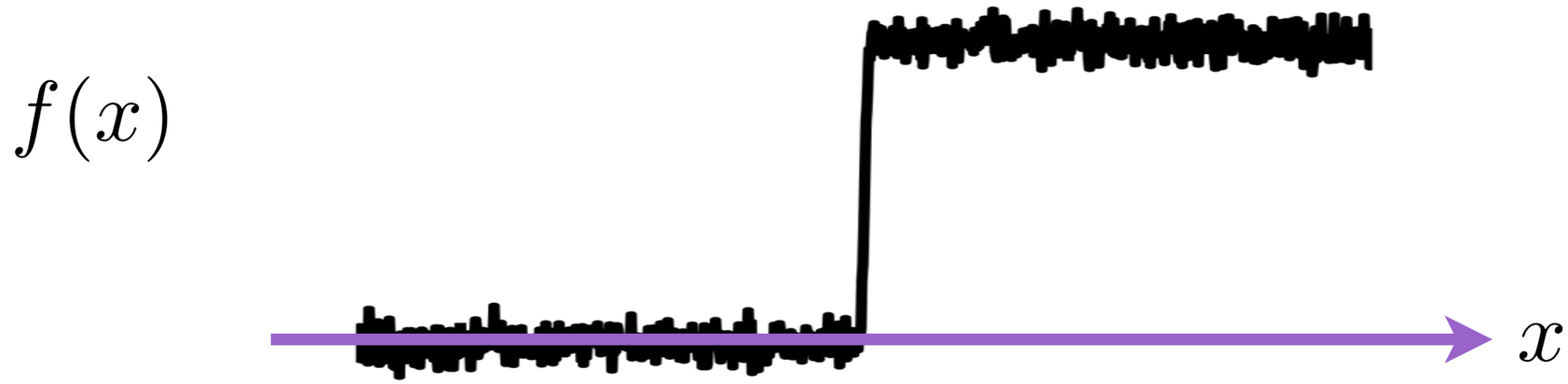
Image NOISE

$f(x)$



$$\frac{d}{dx} f(x)$$

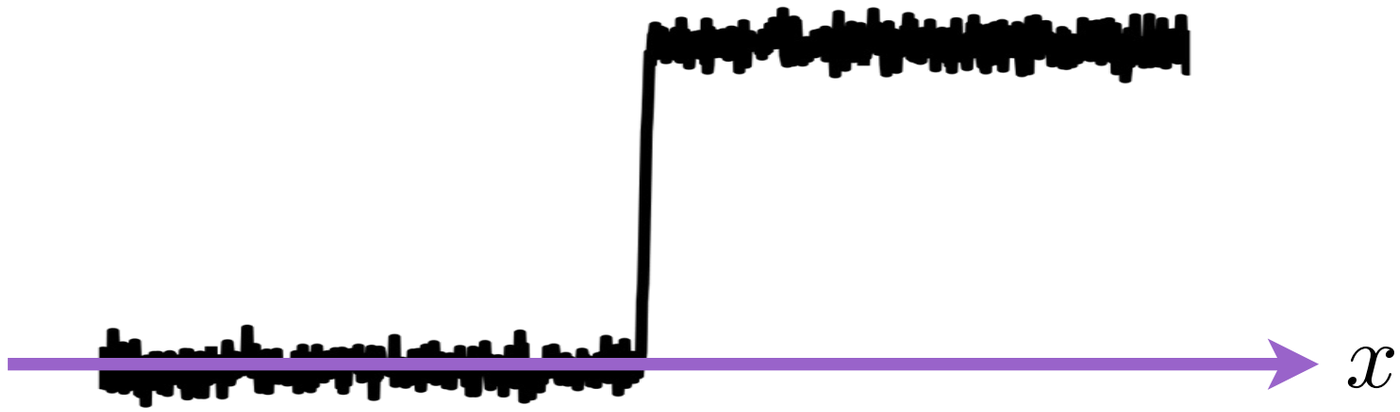
Noise is AMPLIFIED!



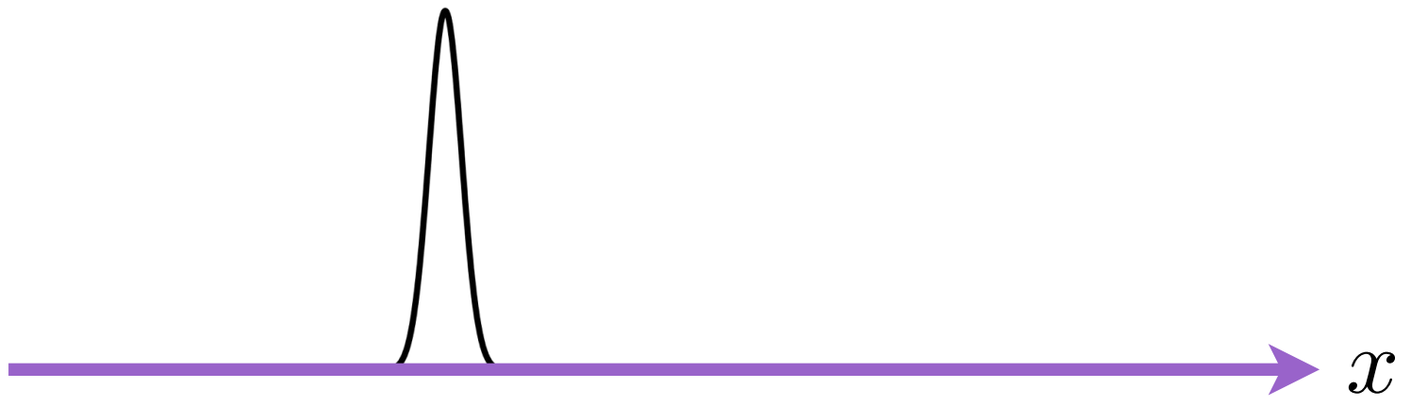
How do we avoid the issues caused by noise?

smooth the image

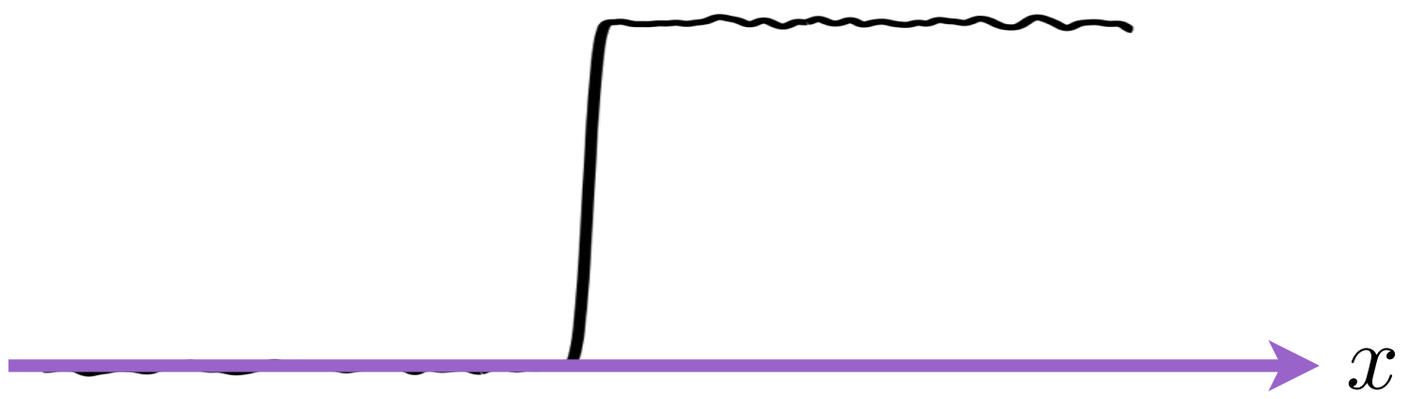
$f(x)$



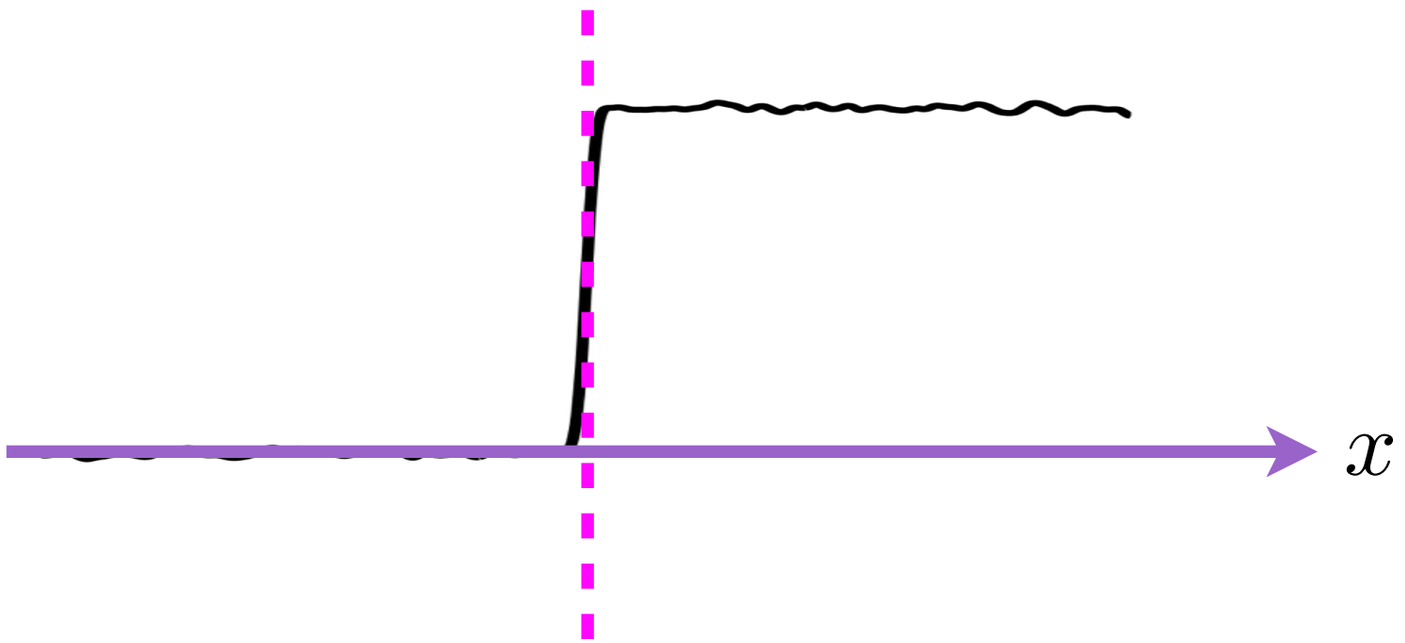
$g(x)$



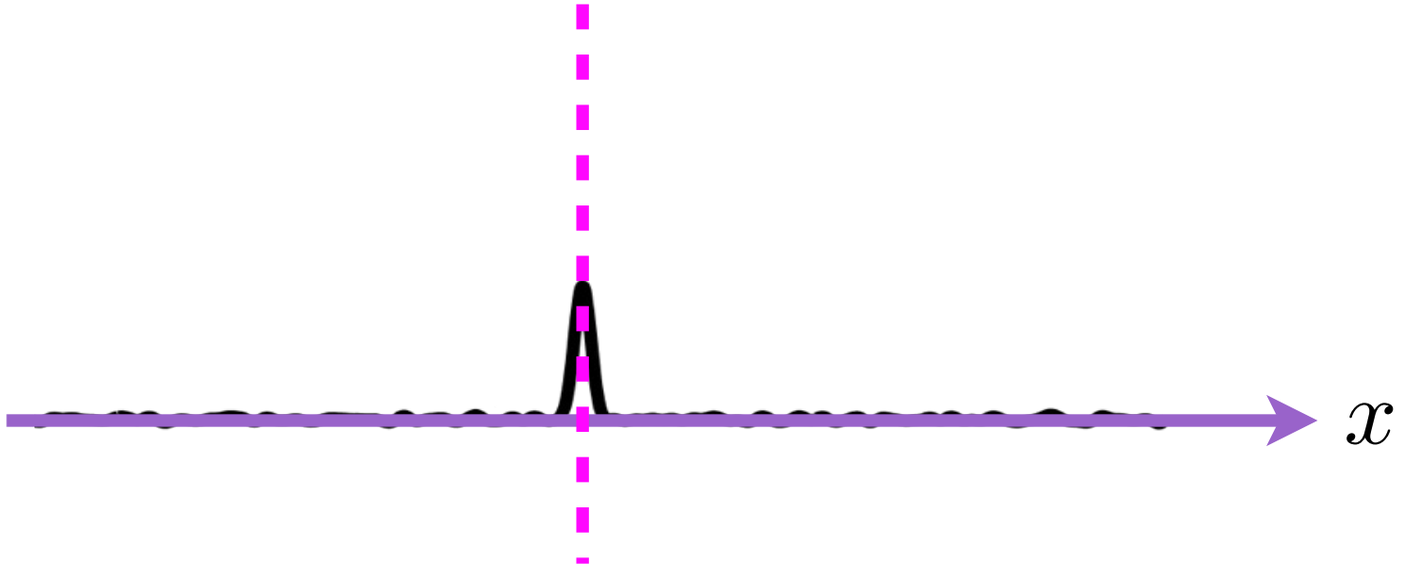
$f * g$



$f * g$



$\frac{d}{dx}(f * g)$



What about 2D?

discrete convolution

$$H[x,y] = \sum_{u=-K}^{K} \sum_{v=-K}^{K} G[u,v]F[x-u,y-v]$$

continuous convolution

$$H(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G(u,v)F(x-u,y-v)dudv$$

$$\frac{\partial}{\partial x}(g * I)$$

Gaussian filter

$$\frac{\partial}{\partial x}(g * I)$$

image

derivative of Gaussian

$$\frac{\partial}{\partial x}(g * I) \equiv \frac{\partial g}{\partial x} * I$$

$$\frac{\partial}{\partial x}(g * I) \equiv \frac{\partial g}{\partial x} * I$$

derivative theorem of convolution

$$\frac{\partial}{\partial x}(g * I) \equiv \frac{\partial g}{\partial x} * I$$

How do we perform discrete filtering?

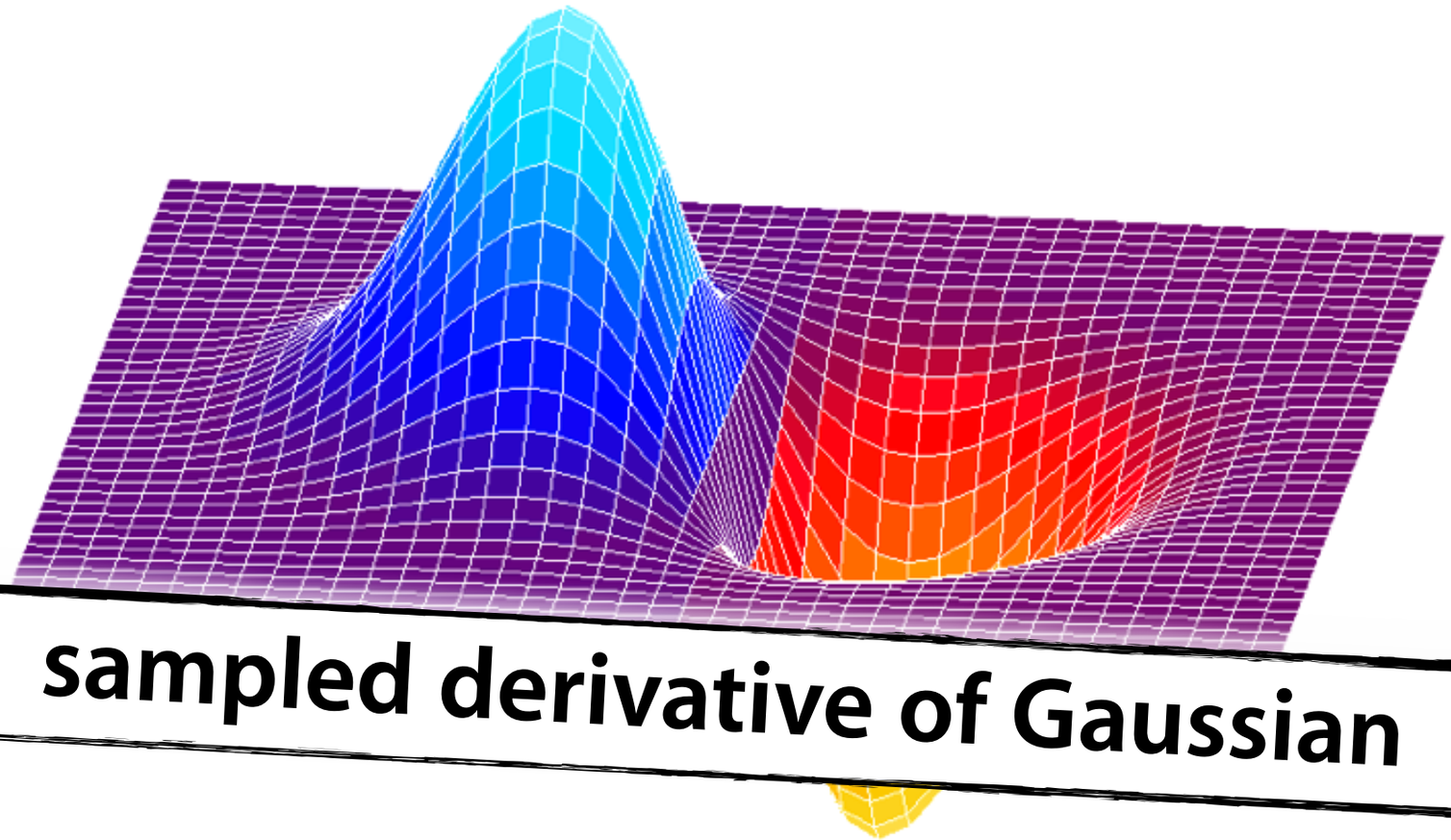
$$\frac{\partial}{\partial x}(g * I) \equiv \frac{\partial g}{\partial x} * I$$

1	-1	*	0.0838	0.6193	0.0838	=	0.0113	0.0838	0.0113
			0.0113	0.0838	0.0113			

$$\frac{\partial}{\partial x}(g * I) \equiv \frac{\partial g}{\partial x} * I$$

Solution 2

$$\frac{\partial}{\partial x} g(x, y; \sigma) = -\frac{x}{2\pi\sigma^4} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$



sampled derivative of Gaussian

Steerable Filters

steerable filter

filter class in which a filter of arbitrary orientation can be expressed via a linear combination of a “basis” set of filters

basis

A set of vectors in a vector space V is called a basis if every element in V can be written in a unique way as a finite linear combination of elements of B.

Definition:

The directional derivative, $\nabla_{\mathbf{u}} f(\mathbf{x}_0)$, is the rate that a function $f(\mathbf{x}_0)$ changes at a point $\mathbf{x}_0$ in the direction $\hat{\mathbf{u}}$. It can be defined as:

Definition:

The directional derivative, $\nabla_{\mathbf{u}} f(\mathbf{x}_0)$, is the rate that a function $f(\mathbf{x}_0)$ changes at a point $\mathbf{x}_0$ in the direction $\hat{\mathbf{u}}$. It can be defined as:

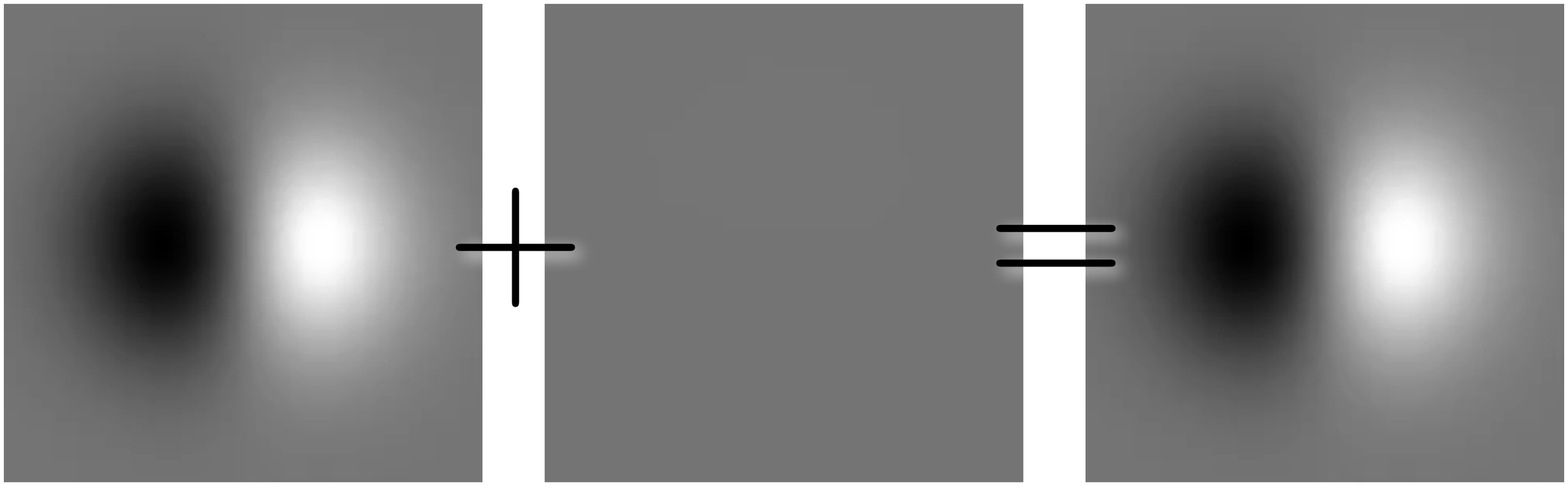
$$\nabla_{\hat{\mathbf{u}}} f(\mathbf{x}_0) = \lim_{h \to 0} \frac{f(\mathbf{x}_0 + h\hat{\mathbf{u}}) - f(\mathbf{x}_0)}{h}$$

$$= \nabla f(\mathbf{x}_0) \cdot \hat{\mathbf{u}}$$

$$= \left(\frac{\partial f(\mathbf{x}_0)}{\partial x}, \frac{\partial f(\mathbf{x}_0)}{\partial y} \right)^\top \cdot (u_0, u_1)^\top$$

$$= u_0 \frac{\partial f(\mathbf{x}_0)}{\partial x} + u_1 \frac{\partial f(\mathbf{x}_0)}{\partial y}$$

$$\cos(\theta)\frac{\partial G(x,y)}{\partial x} + \sin(\theta)\frac{\partial G(x,y)}{\partial y} = \frac{\partial G(x,y)}{\partial \theta}$$



$$\cos(\theta)\frac{\partial G(x,y)}{\partial x} \qquad \sin(\theta)\frac{\partial G(x,y)}{\partial y} \qquad \frac{\partial G(x,y)}{\partial \theta}$$

$$I_\theta(x,y) = [\cos(\theta)\frac{\partial G(x,y)}{\partial x} + \sin(\theta)\frac{\partial G(x,y)}{\partial y}] * I(x,y)$$

Can we compute this more efficiently?

$$E[x,y] * (F[x,y] + G[x,y])$$

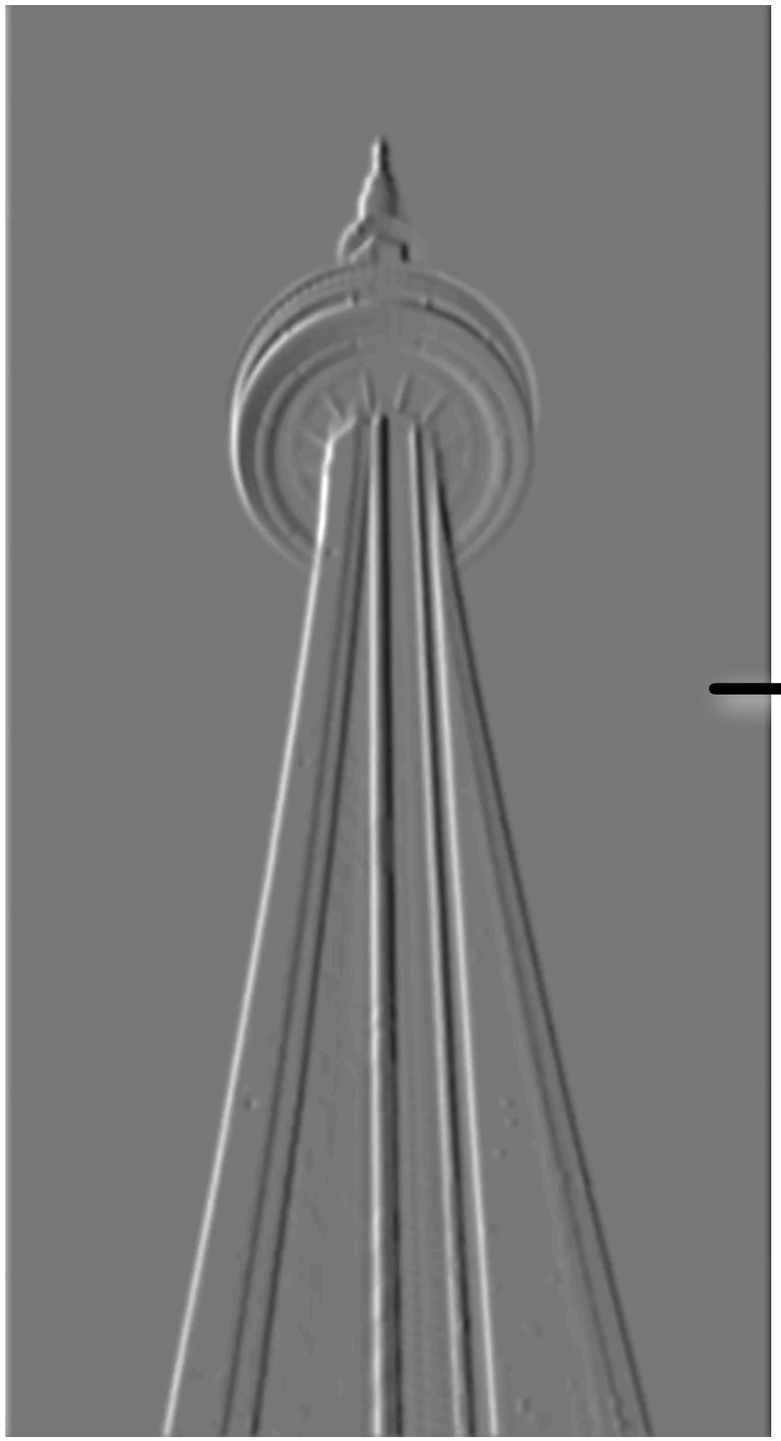
$$= (E[x,y] * F[x,y]) + (E[x,y] * G[x,y])$$

$$I_\theta(x,y) = [\cos(\theta)\frac{\partial G(x,y)}{\partial x} + \sin(\theta)\frac{\partial G(x,y)}{\partial y}] * I(x,y)$$

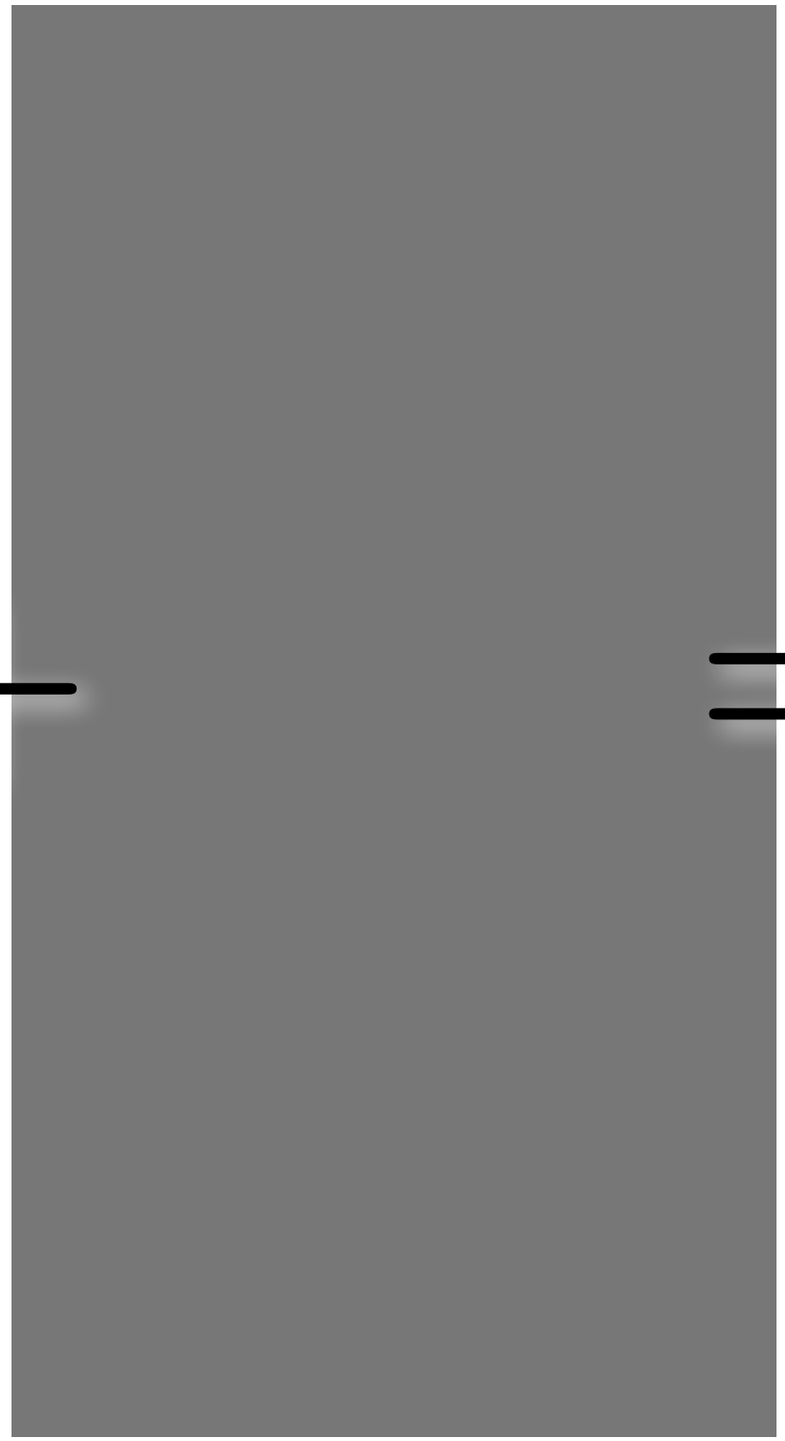
Can we compute this more efficiently?

$$= \cos(\theta)[\frac{\partial G(x,y)}{\partial x} * I(x,y)] + \sin(\theta)[\frac{\partial G(x,y)}{\partial y} * I(x,y)]$$

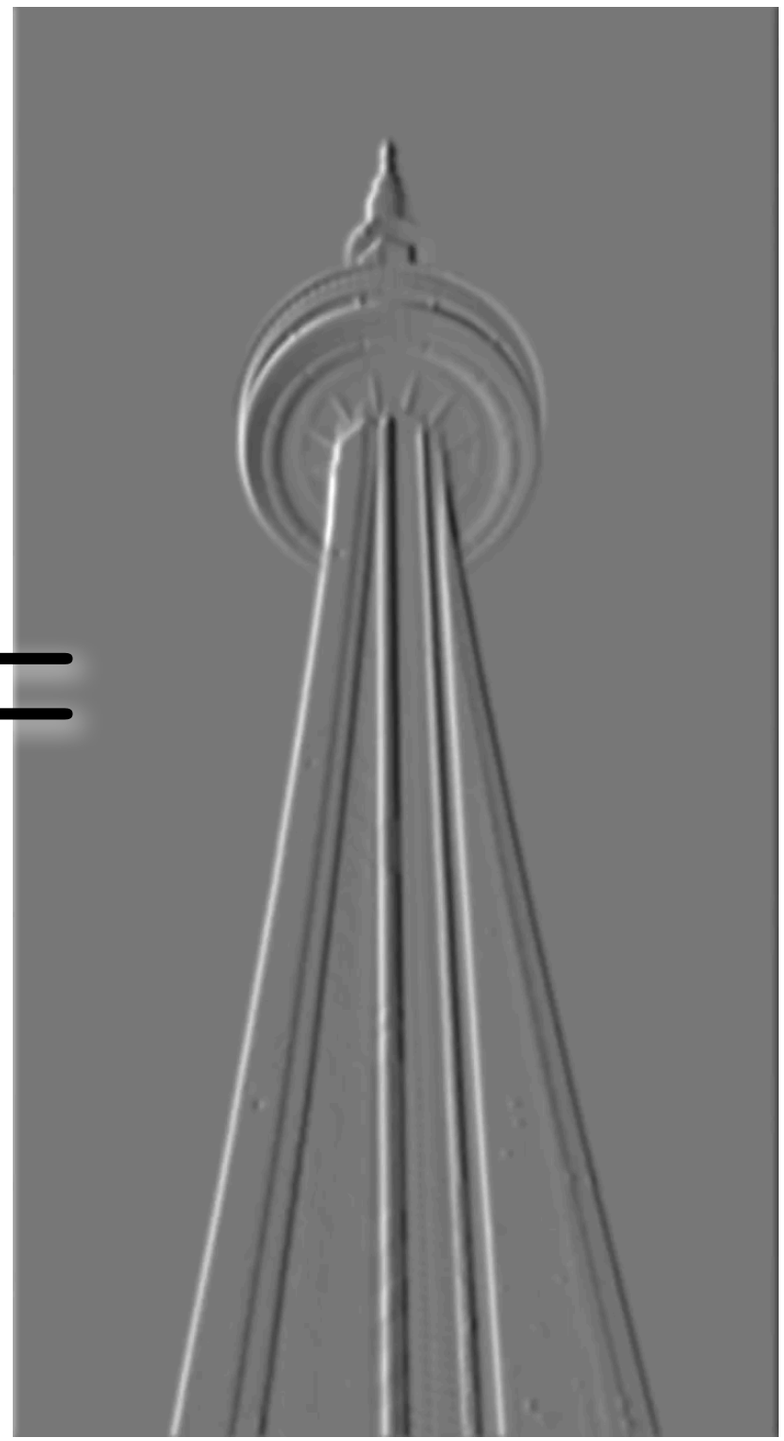
distributive property leads to efficient implementation



$$\cos(\theta)[\frac{\partial G}{\partial x} * I]$$



$$\sin(\theta)[\frac{\partial G}{\partial y} * I]$$



$$\frac{\partial [G * I]}{\partial \theta}$$

reduce image size by removing non-salient seams


$$\text{Energy}[x, y] = \sqrt{I_x^2 + I_y^2}$$

seam is an “eight”-connected vertical path

Goal: Find the seam with the least cumulative energy

$$\mathbf{s}^* = \arg\min_{\mathbf{s}} \text{Cost}[\mathbf{s}]$$

where

$$\text{Cost}[\mathbf{s}] = \sum_{i=0}^{\text{height}} \text{Energy}[\mathbf{s}_i]$$

$$\mathbf{M}[i,j] = \text{Energy}[i,j] + \min(\mathbf{M}[i-1,j-1], \mathbf{M}[i,j-1], \mathbf{M}[i+1,j-1])$$

minimum cumulative energy ending at pixel $[i,j]$

$$\mathbf{M}[i,j] = \text{Energy}[i,j] +$$
$$\min(\mathbf{M}[i-1,j-1], \mathbf{M}[i,j-1], \mathbf{M}[i+1,j-1])$$

$\text{Energy}[i,j]$

0	5	1
6	8	4
9	1	7

optimal solution

$\mathbf{M}[i,j]$

0	5	1
6	8	5
15	6	12

partial optimal solution

To find the optimal seam, backtrack from minimum

input image

seam carving result

input image

seam carving result

A Computational Approach to Edge Detection

JOHN CANNY, MEMBER, IEEE

Abstract—**This paper describes a computational approach to edge detection. The success of the approach depends on the definition of a *comprehensive* set of goals for the computation of edge points. These goals must be precise enough to delimit the desired behavior of the detector while making minimal assumptions about the form of the solution. We define detection and localization criteria for a class of edges, and present mathematical forms for these criteria as functionals on the**

detector as input to a program which could isolate simple geometric solids. More recently the model-based vision system ACRONYM [3] used an edge detector as the front end to a sophisticated recognition program. Shape from motion [29], [13] can be used to infer the structure of three-dimensional objects from the motion of edge con-

4

major steps

Step 1

Filter image with x and y derivatives of Gaussian

x-derivative of Gaussian

y-derivative of Gaussian

Step 2

Find magnitude and direction of gradient

gradient magnitude

```
img_grad_mag = sqrt(img_dx.^2 + img_dy.^2);
```

gradient orientation

```
img_grad_or = atan2(img_dy, img_dx);
```

gradient magnitude

Threshold gradient magnitude and done?

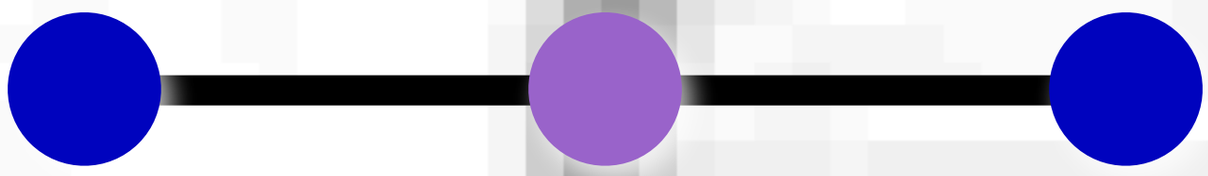
```
>> img_thresh = im_grad_mag > 15;
```

gradient magnitude threshold

End up with thick edges

Step 3

Perform non-maximum suppression



Is the gradient magnitude greater than its neighbours along the gradient orientation?

If yes, keep, otherwise suppress.

gradient magnitude

non-max suppression

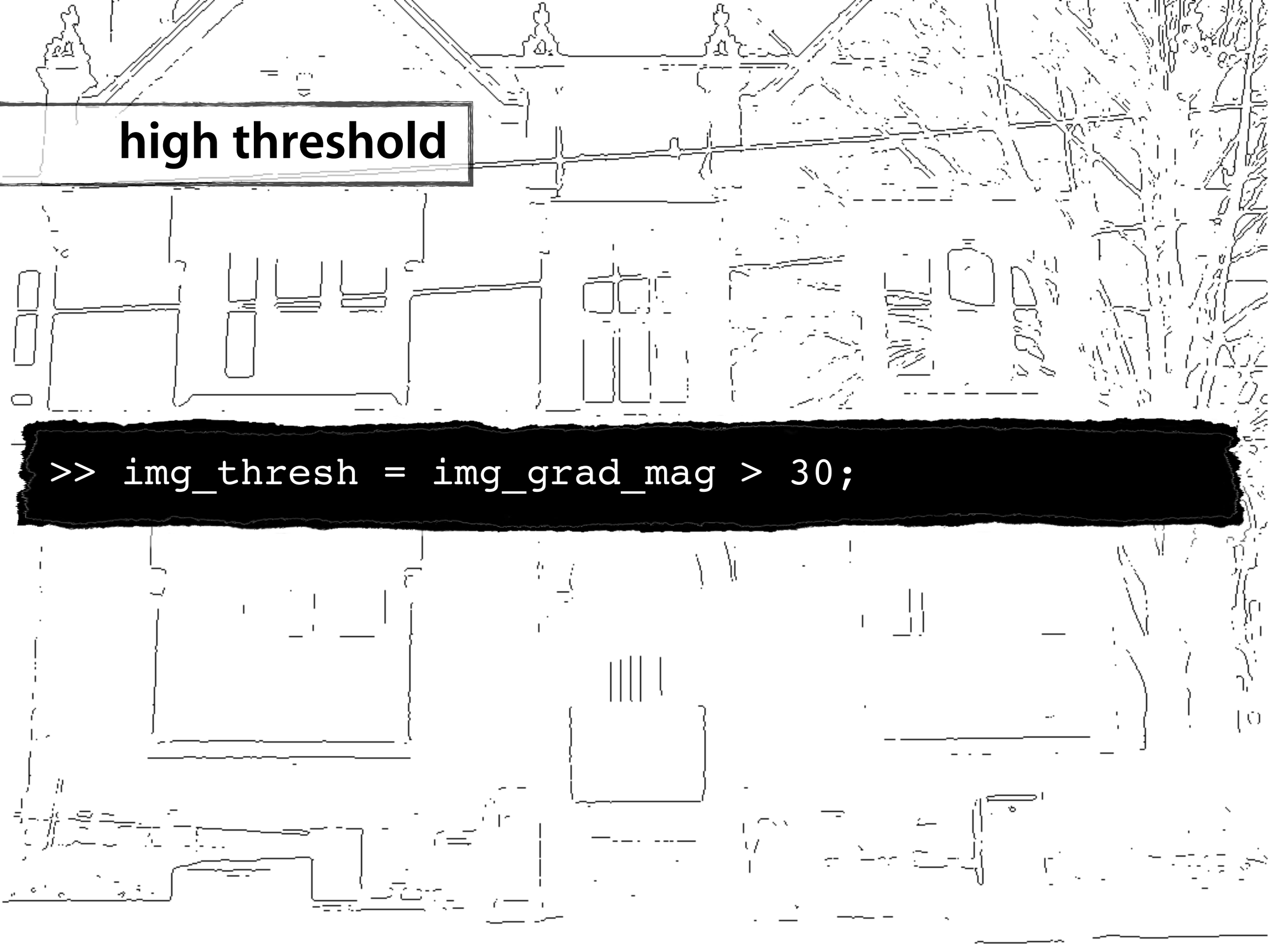
Step 4

Threshold and link

also known as Hysteresis Thresholding

low threshold

```
>> img_thresh = img_grad_mag > 5;
```




```
>> img_thresh = img_grad_mag > 30;
```

non-max suppression

non-max suppression

Define two thresholds, low and high

Define two thresholds, low and high

Starting from a pixel passing a high threshold, keep only pixels on the path that pass the low threshold

Starting from a pixel passing a high threshold, keep only pixels on the path that pass the low threshold

non-max suppression

hysteresis thresholding

1. Filter image with x and y derivatives of Gaussian
2. Find magnitude and direction of gradient
3. Perform non-maximum suppression
4. **Threshold and link**

```
>> % type 'help edge' to see different
>> %     edge-finding methods
>> img_edge = edge(img, 'canny');
>> imshow(img_edge, [])
```

What is the right SCALE?

Scale?

input

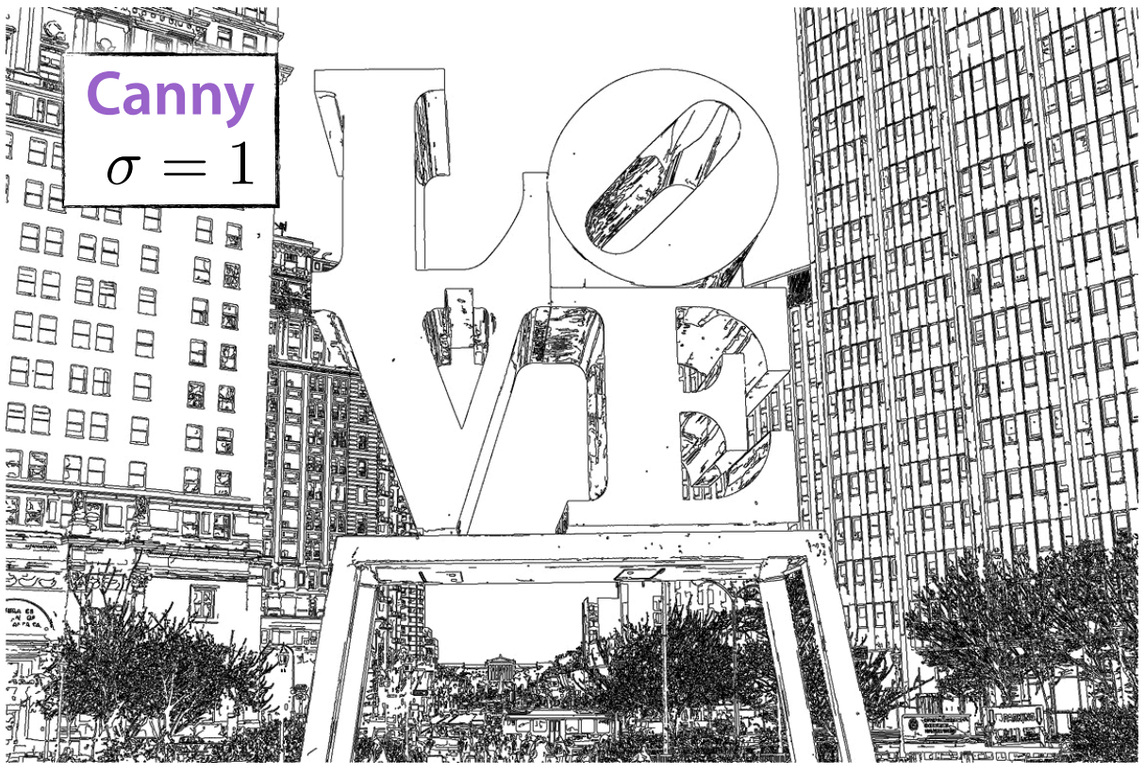
Canny

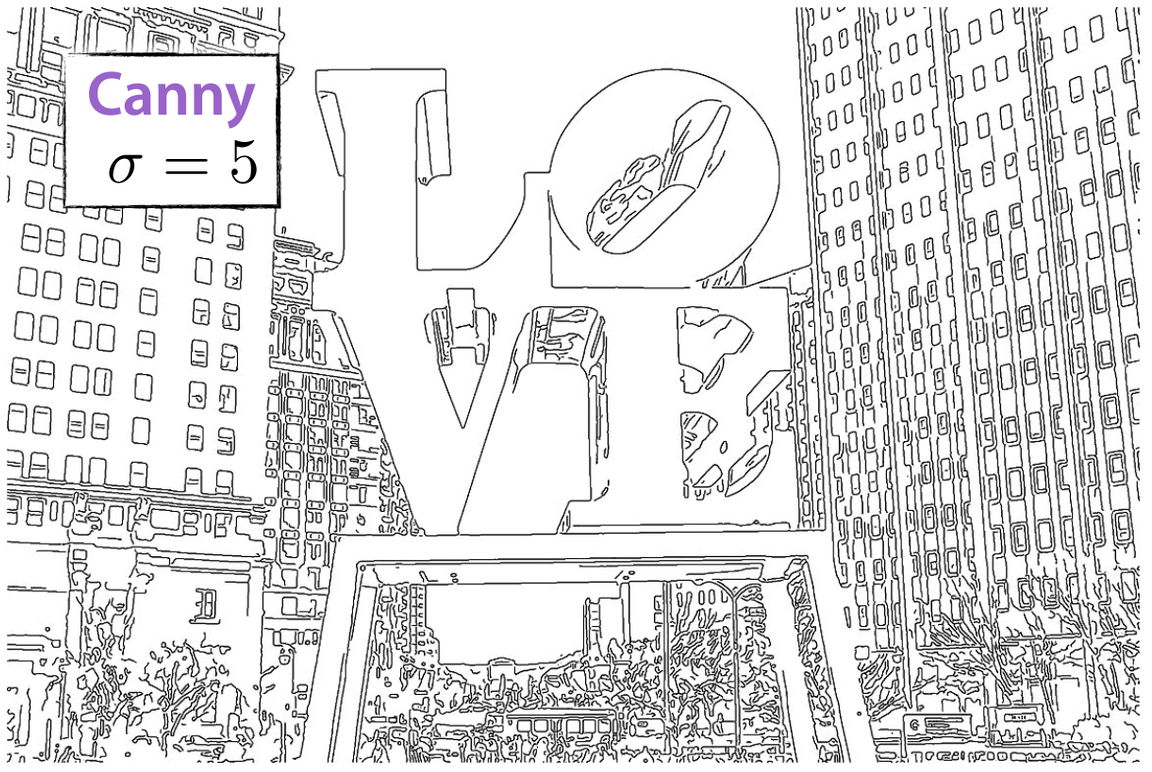
$\sigma = 1$

Canny

$\sigma = 5$


input


Canny
$\sigma = 1$

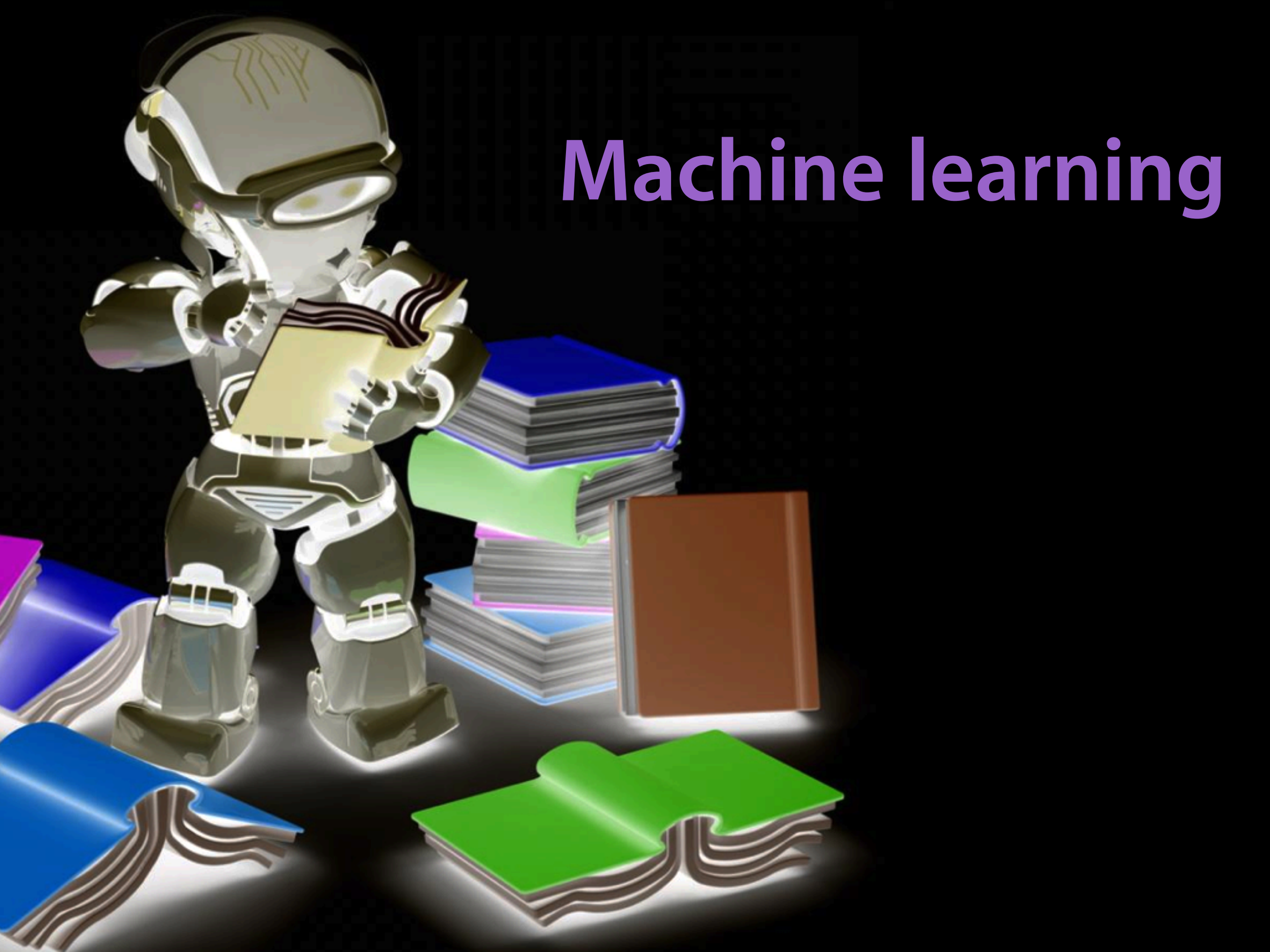

Canny
$\sigma = 5$

Detect changes in colour?

Detect changes in texture?

How can we combine cues to detect object contours?

Machine learning



$\mathbf{x}$

feature

$\hat{f}$

y

class label

Given training data, estimate the function f

$$\{(\mathbf{x}_i, y_i)\}$$