midterm review

16-385 Computer Vision

Carnegie Mellon University (Kris Kitani)

Filtering and convolution

$g[\cdot,\cdot]$

filter

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

$f[\cdot,\cdot]$

image

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot,\cdot]$

output

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

output — filter — image (signal)

Filtering vs Convolution

filtering
(cross-correlation)

$$h = g \otimes f$$

$$h[m,n] = \sum_{k,j} g[k,l] f[m+k, n+l]$$

output filter image

What's the difference?

convolution

$$h = g \star f$$

$$h[m,n] = \sum_{k,j} g[k,l] f[m-k, n-l]$$

Filtering vs Convolution

filtering
(cross-correlation)

$$h = g \otimes f$$

output filter image

$$h[m,n] = \sum_{k,j} g[k,l] f[m+k, n+l]$$

filter flipped vertically and horizontally

convolution

$$h = g \star f$$

$$h[m,n] = \sum_{k,j} g[k,l] f[m-k, n-l]$$

Filtering vs Convolution

filtering
(cross-correlation)

$$h = g \otimes f$$

$$h[m,n] = \sum_{k,j} g[k,l] f[m+k, n+l]$$

output — filter — image

convolution

$$h = g \star f$$

$$h[m,n] = \sum_{k,j} g[k,l] f[m-k, n-l]$$

filter flipped vertically and horizontally

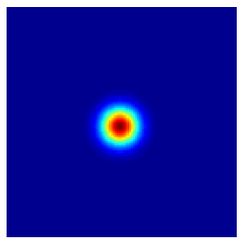
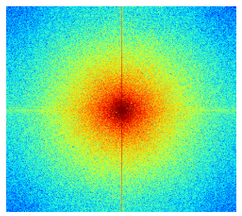
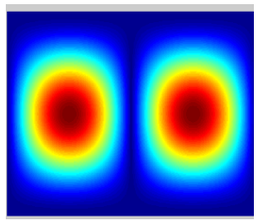
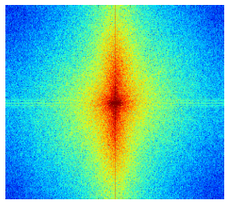
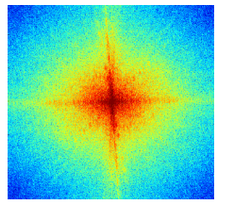
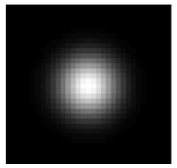
Suppose g is a Gaussian filter.
How does convolution differ from filtering?

Recall...

$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

Match the image to the Fourier magnitude image:

1	2	3	4	5
				
a	b	c	d	e
				

Hough Transform

Image and parameter space

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

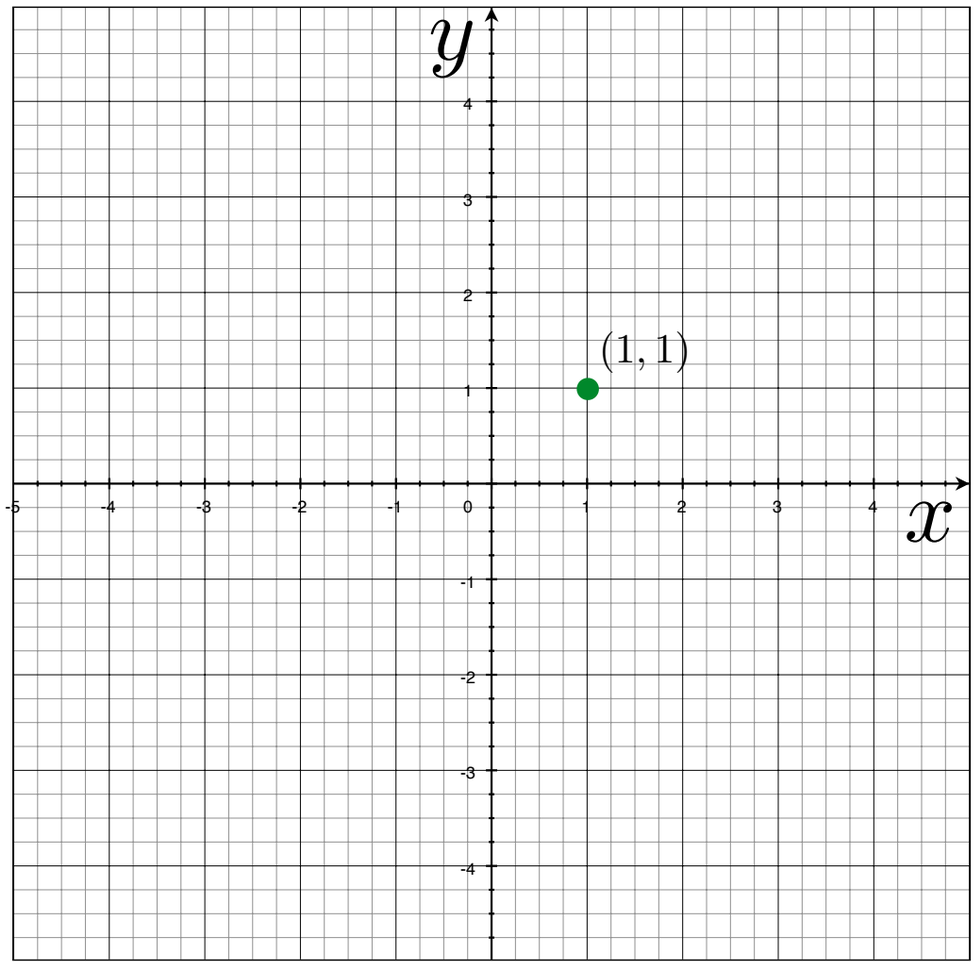
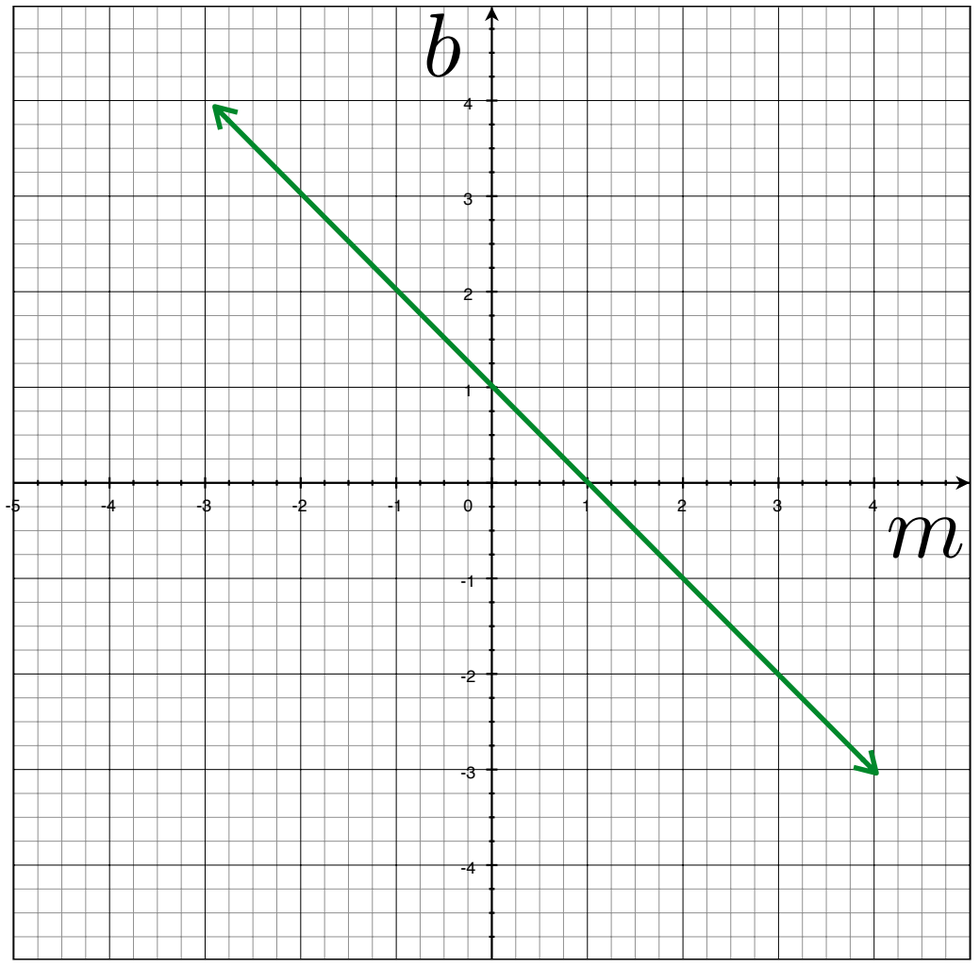


Image space

a point becomes a line



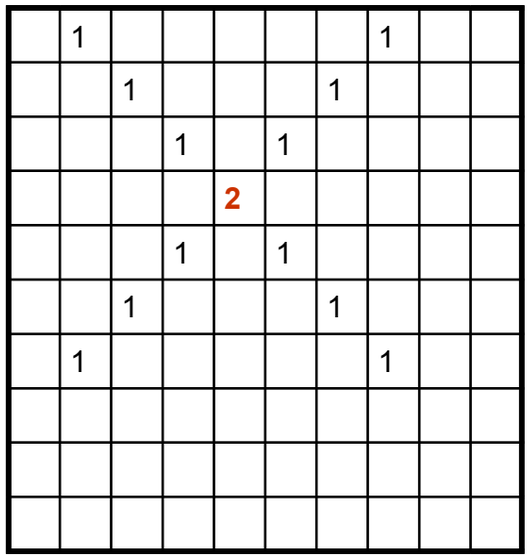
Parameter space

Problems with parameterization

What's wrong with the parameterization (m, c) *?*

How big does the accumulator need to be?

$A(m, c)$

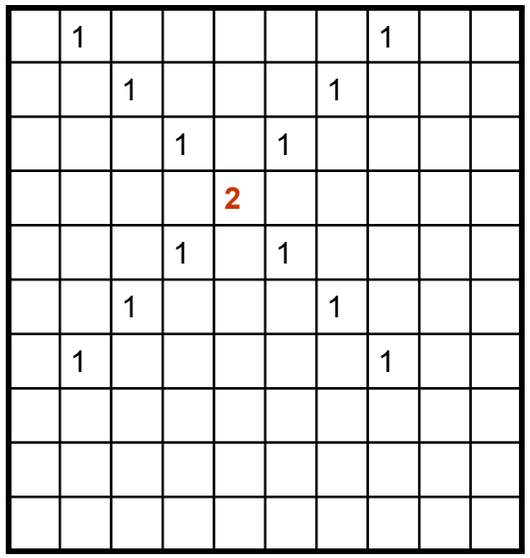


Problems with parameterization

What's wrong with the parameterization (m, c) *?*

How big does the accumulator need to be?

$A(m, c)$



The space of m is huge!

$$-\infty \leq m \leq \infty$$

The space of c is huge!

$$-\infty \leq c \leq \infty$$

Image and parameter space

variables

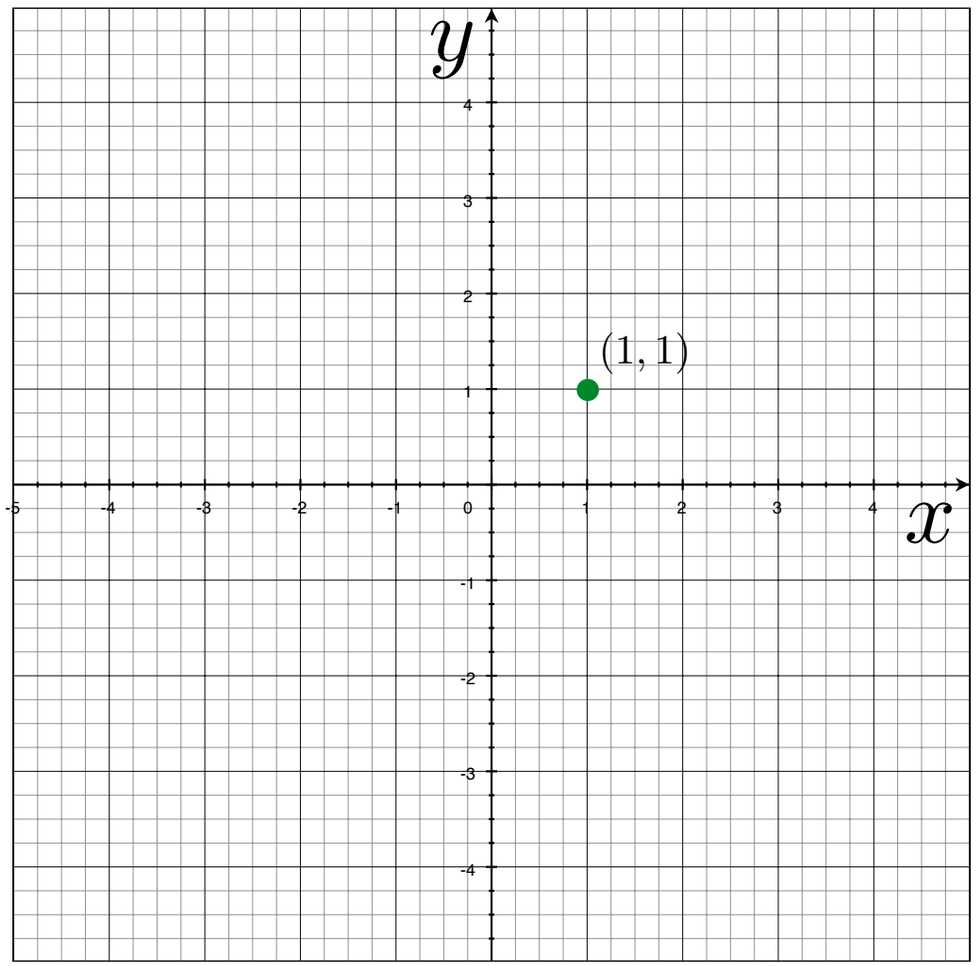
$$y = mx + b$$

parameters

parameters

$$x\cos\theta + y\sin\theta = \rho$$

variables



a point becomes a line

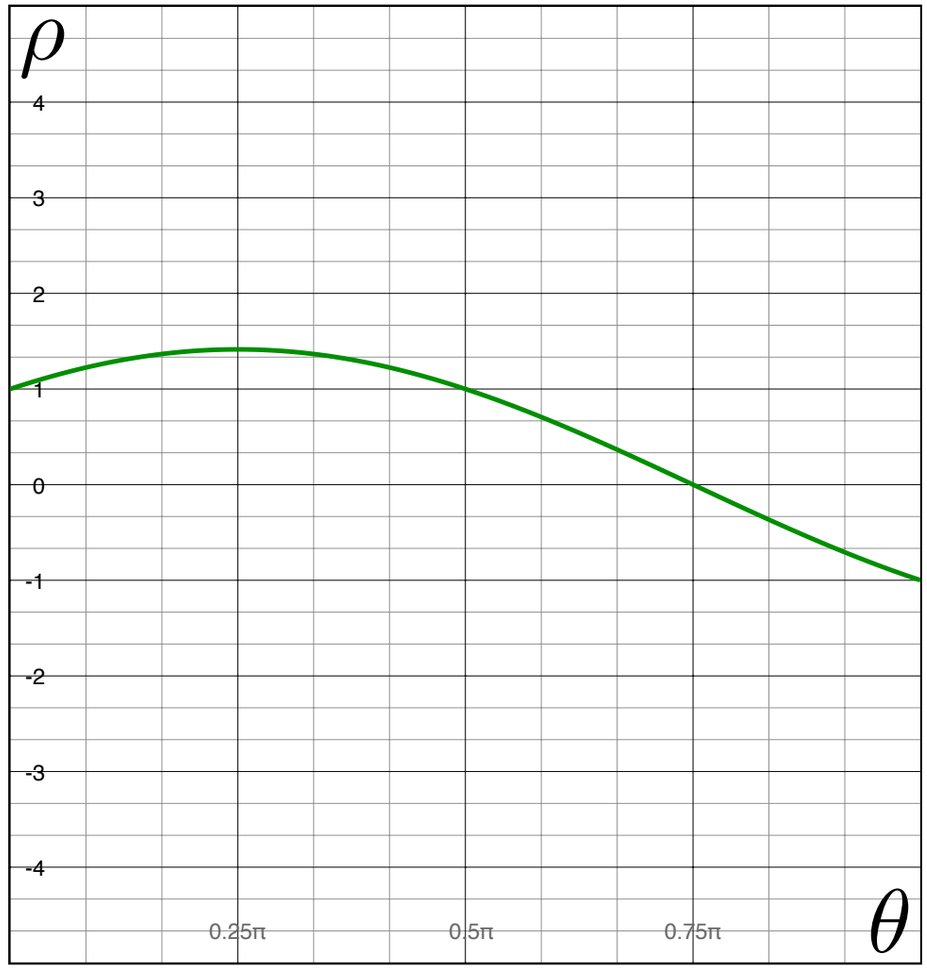


Image space

Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

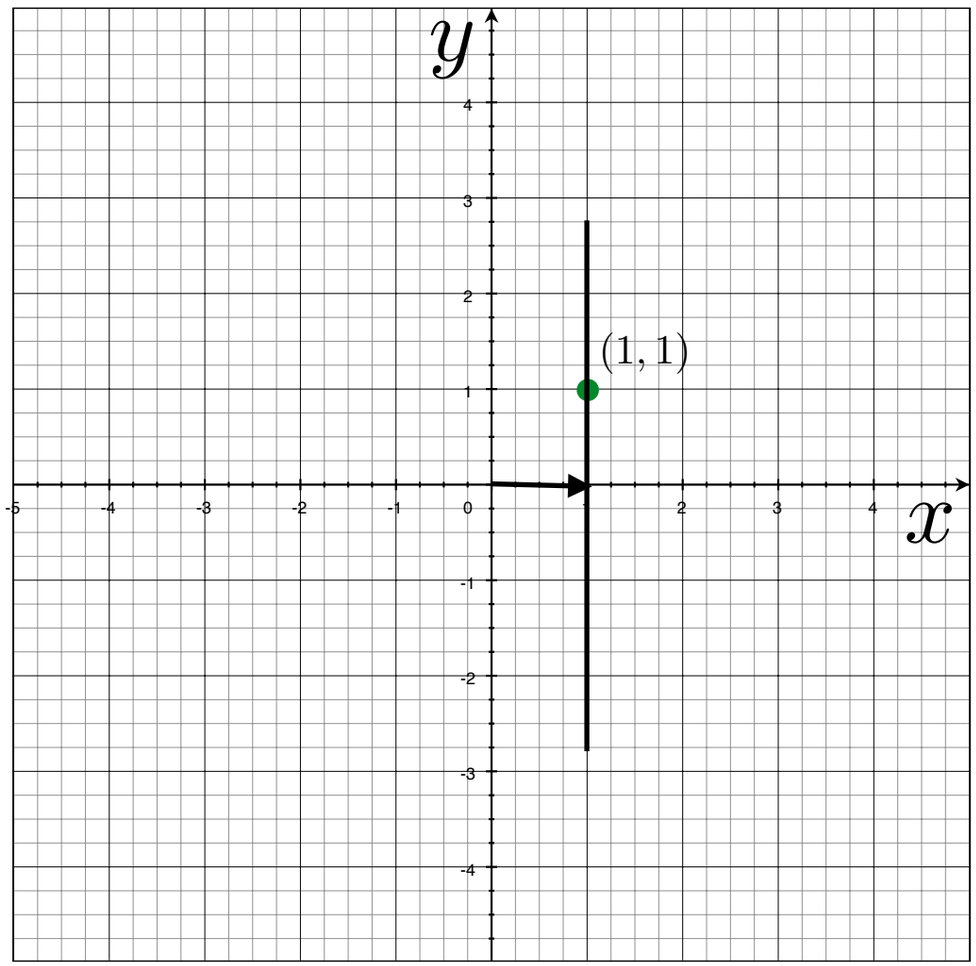
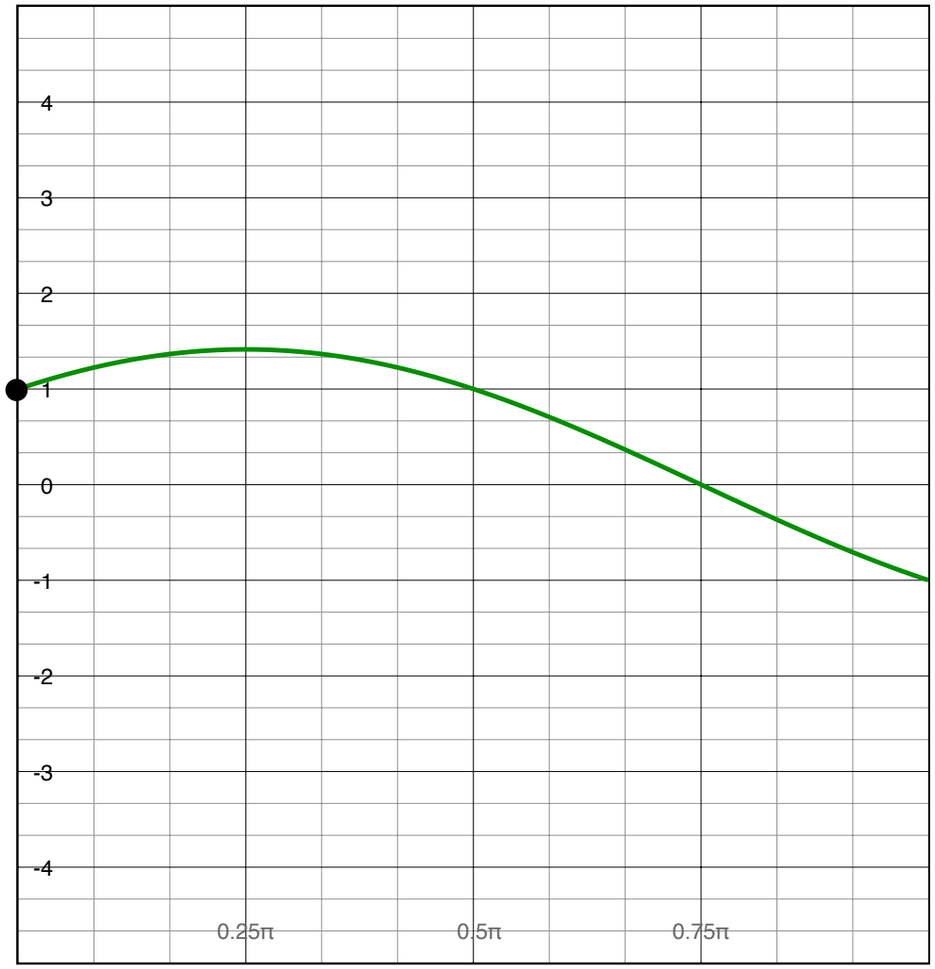


Image space

a line becomes a point



Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

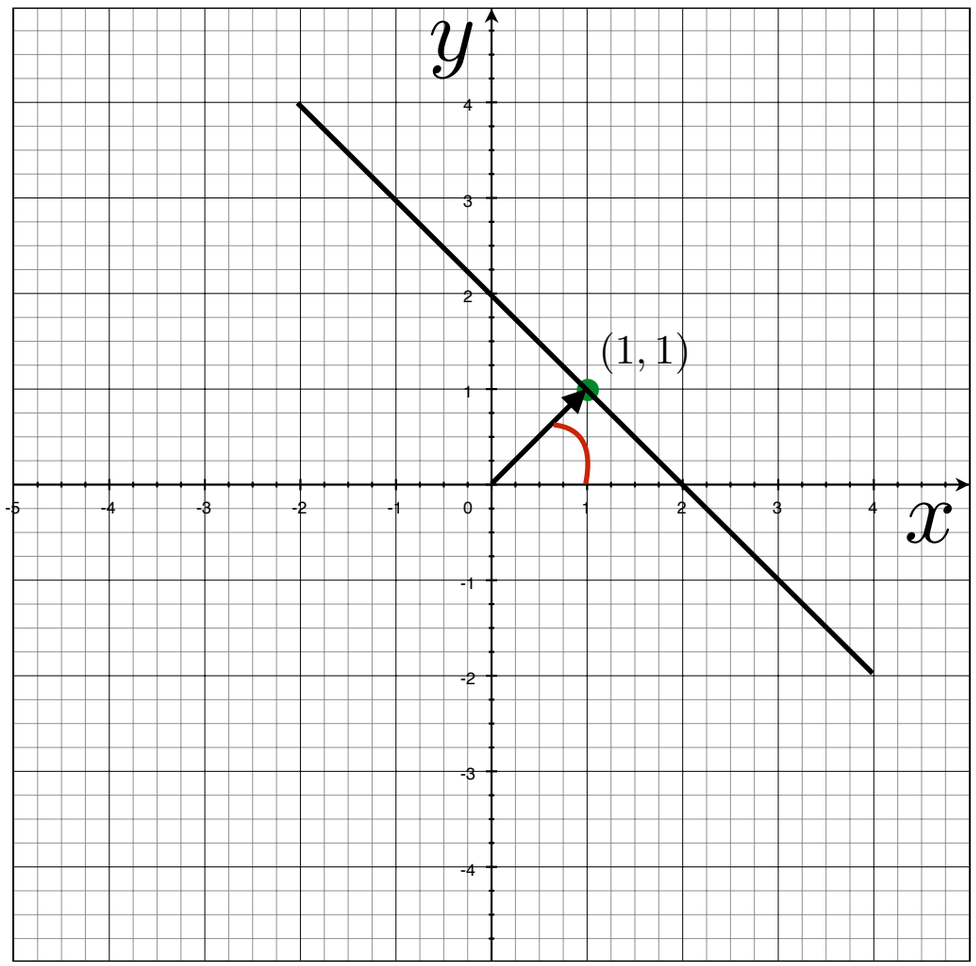


Image space

a line becomes a point

Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

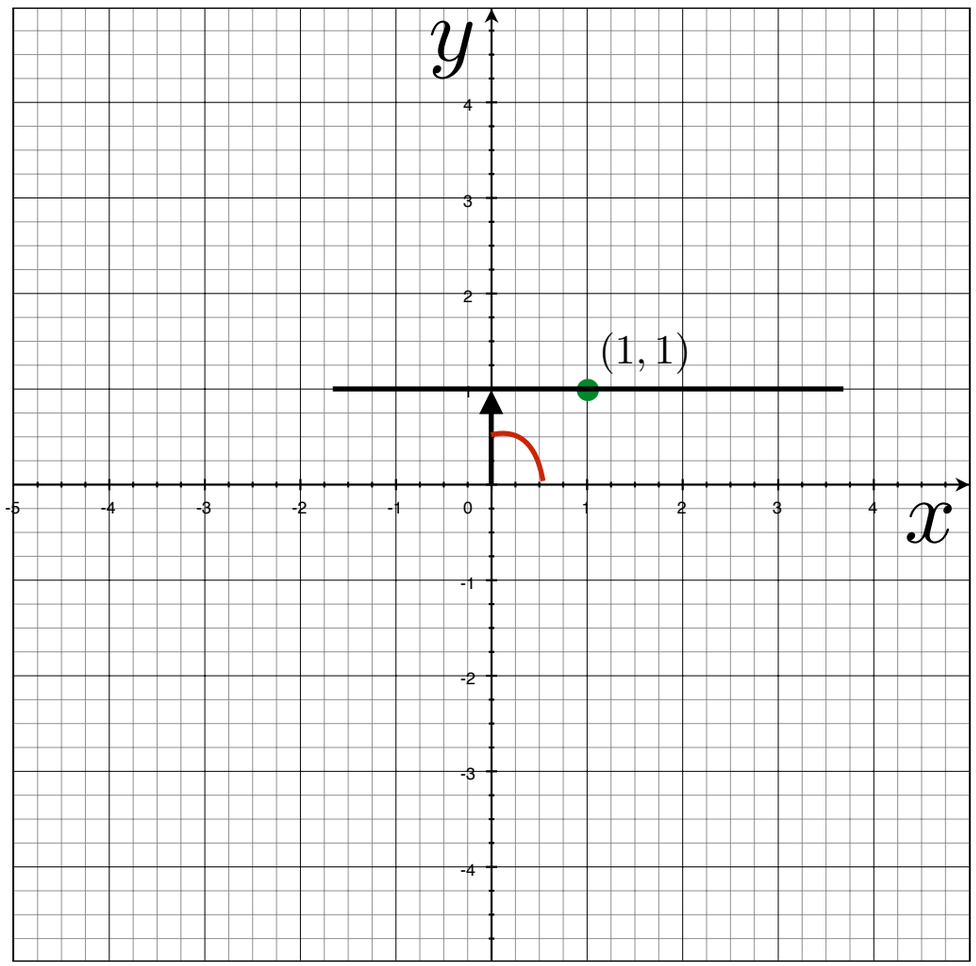
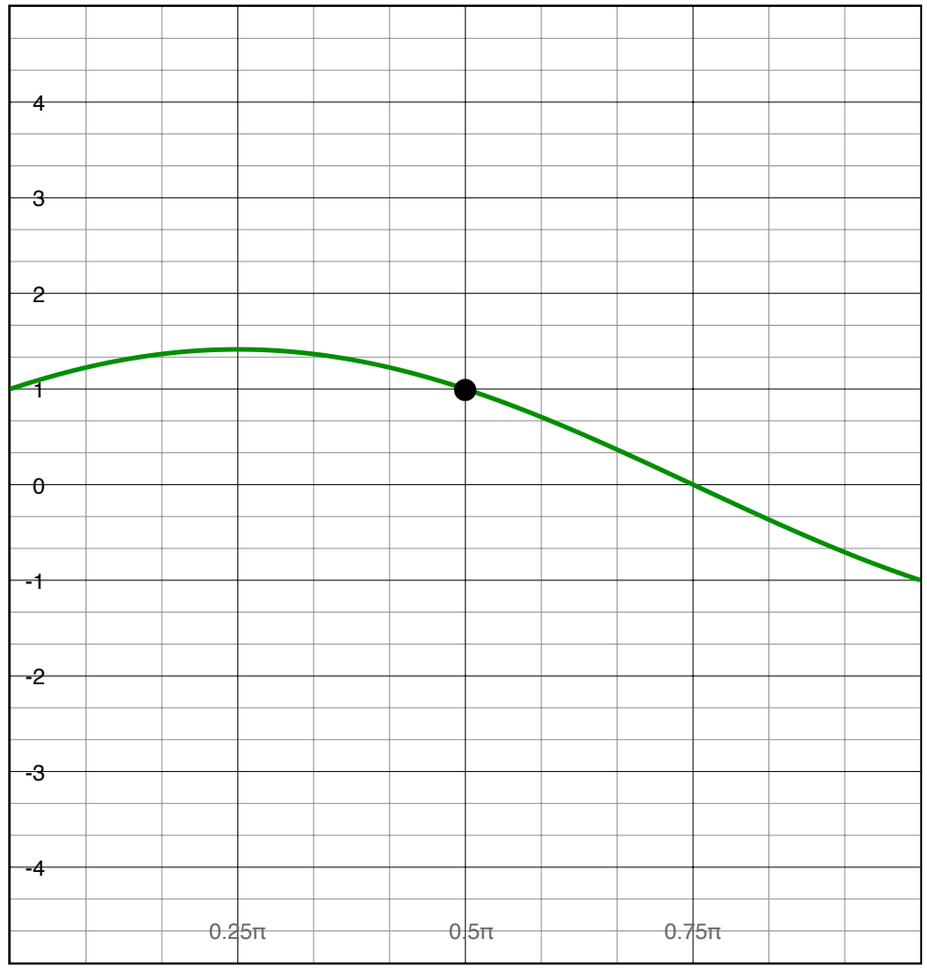


Image space

a line becomes a point



Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

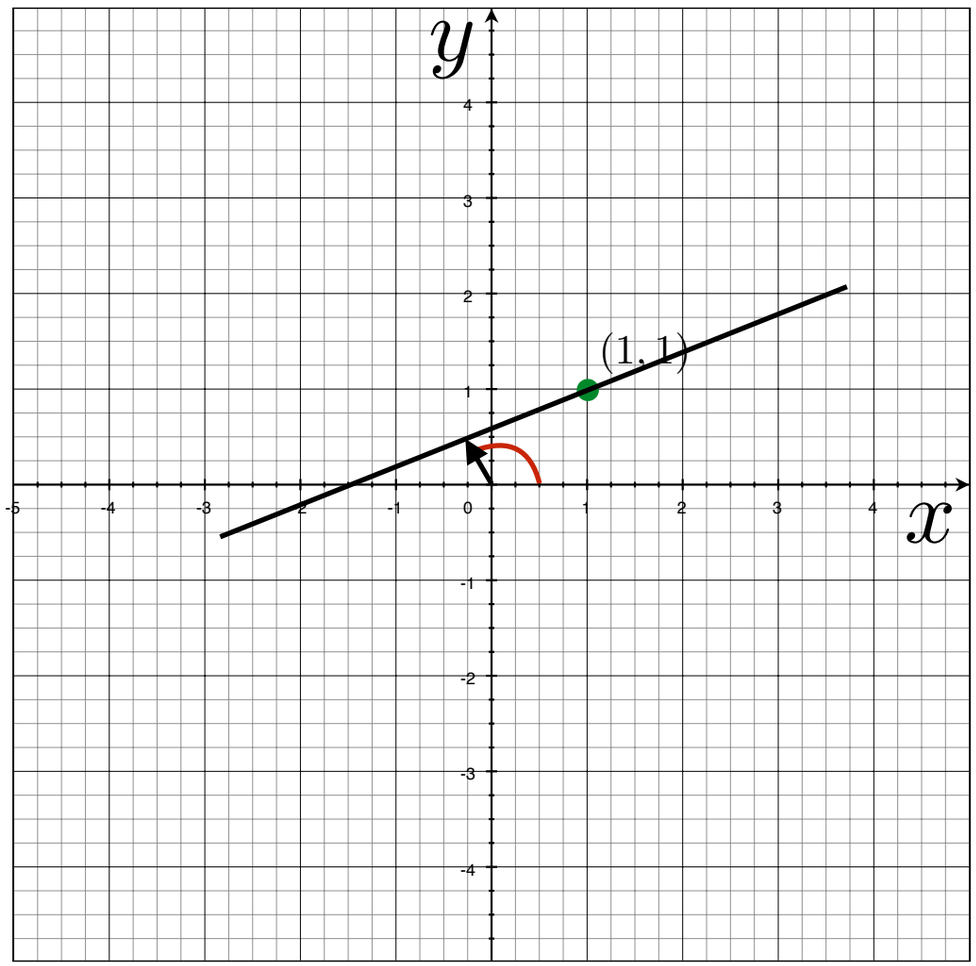
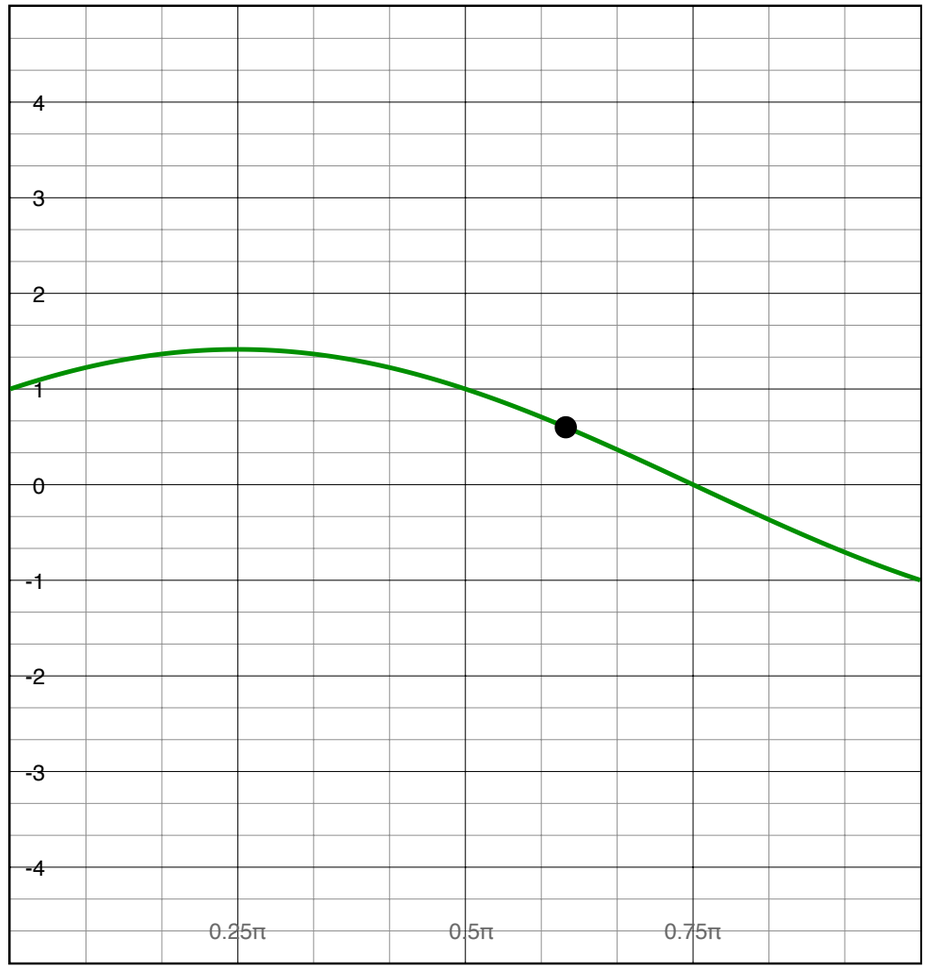


Image space

a line becomes a point



Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

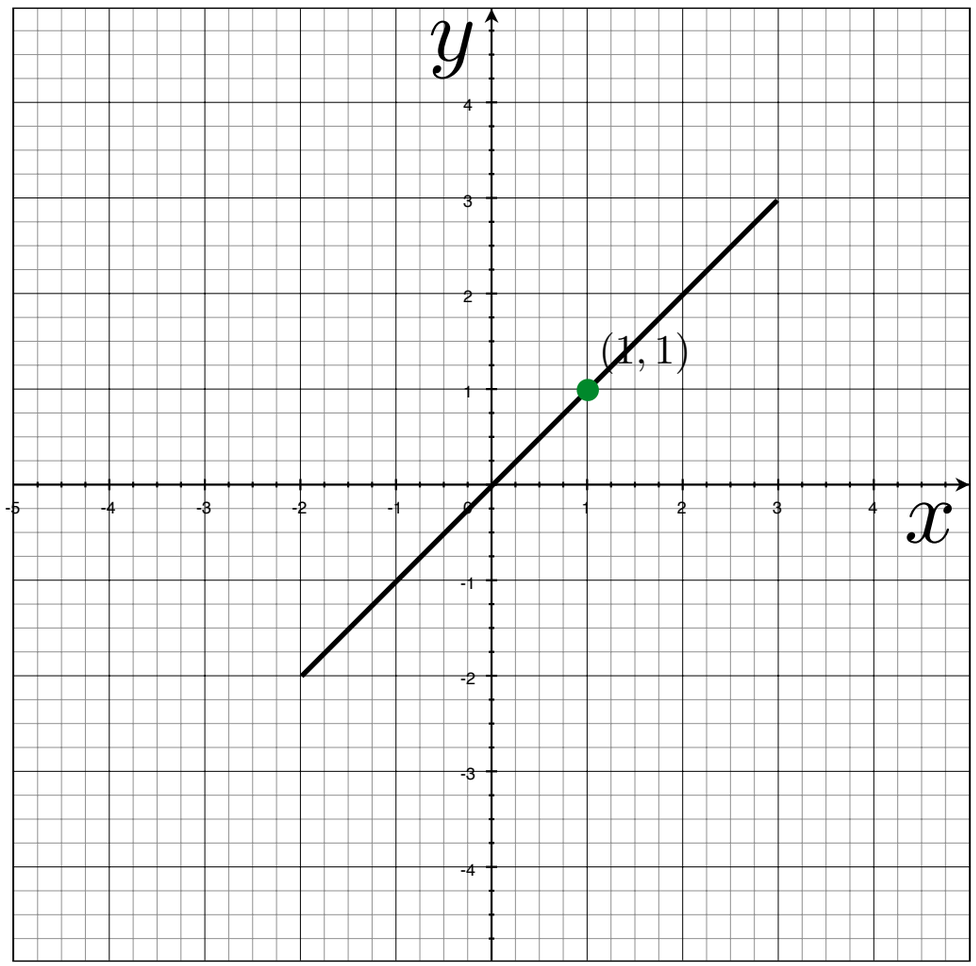
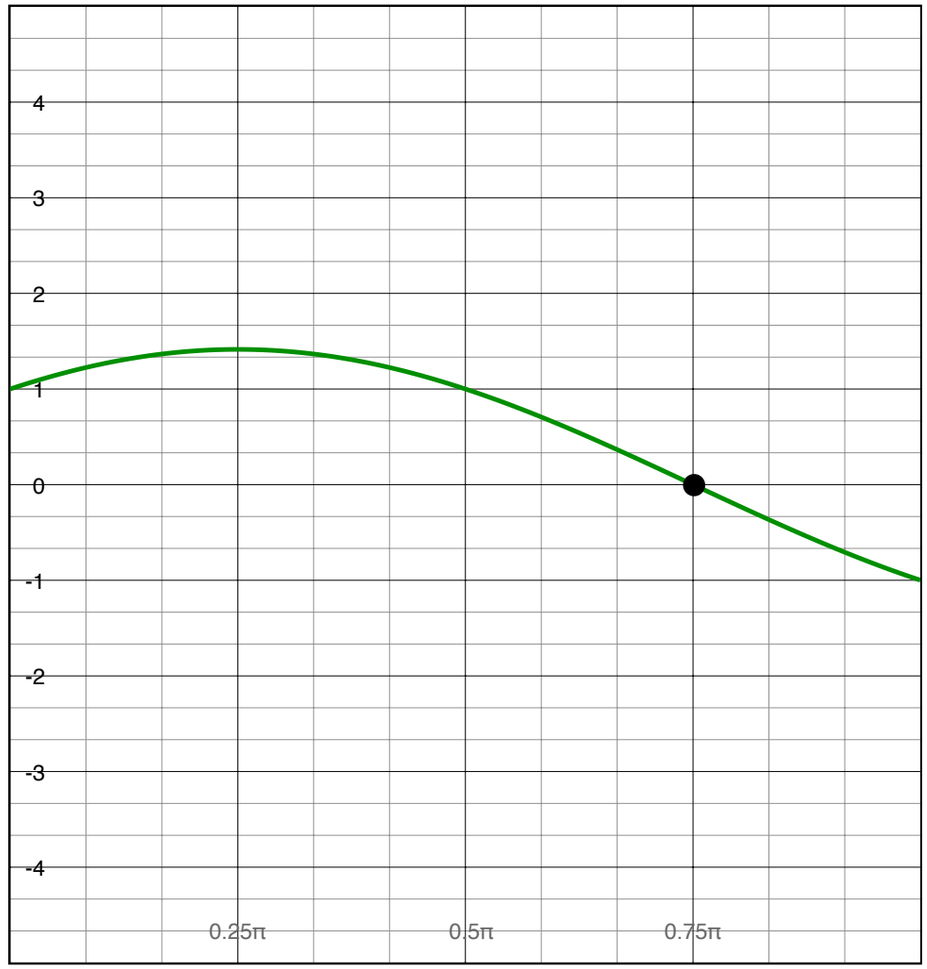


Image space

a line becomes a point



Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

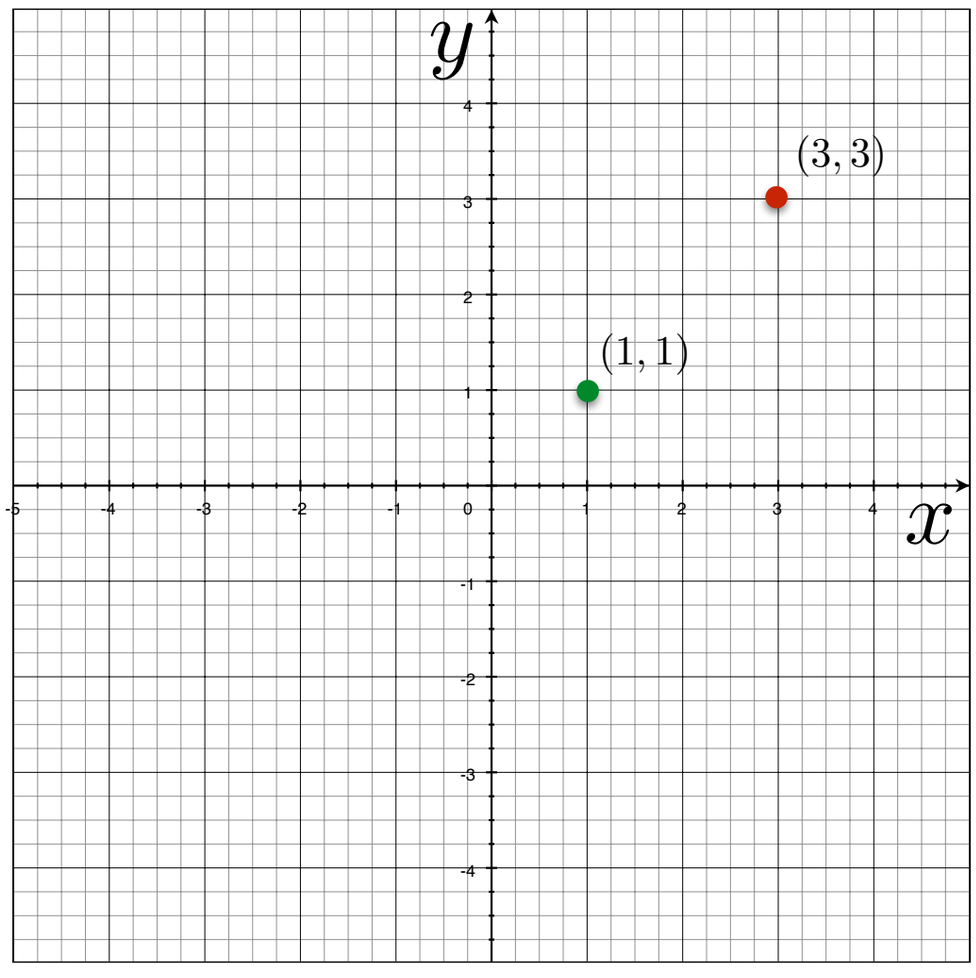
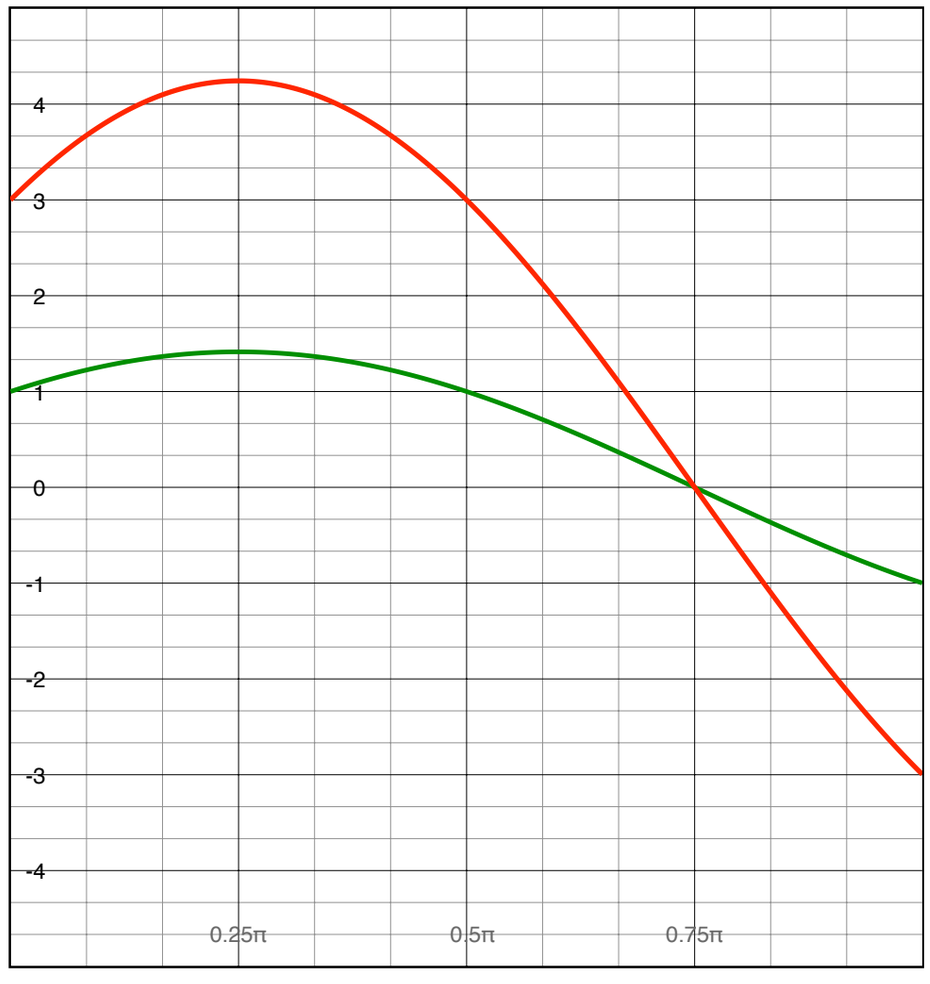


Image space

two points become ?

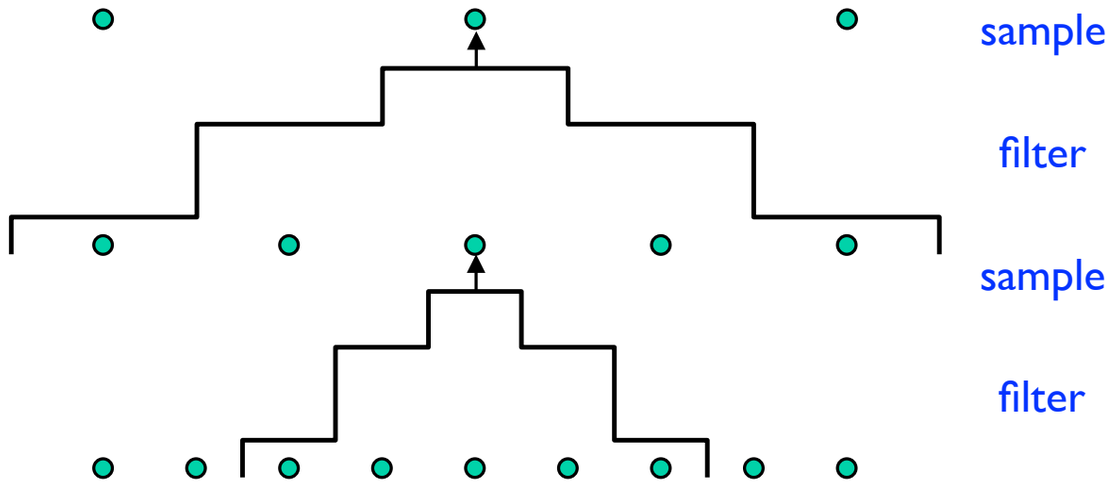


Parameter space

Image Pyramids

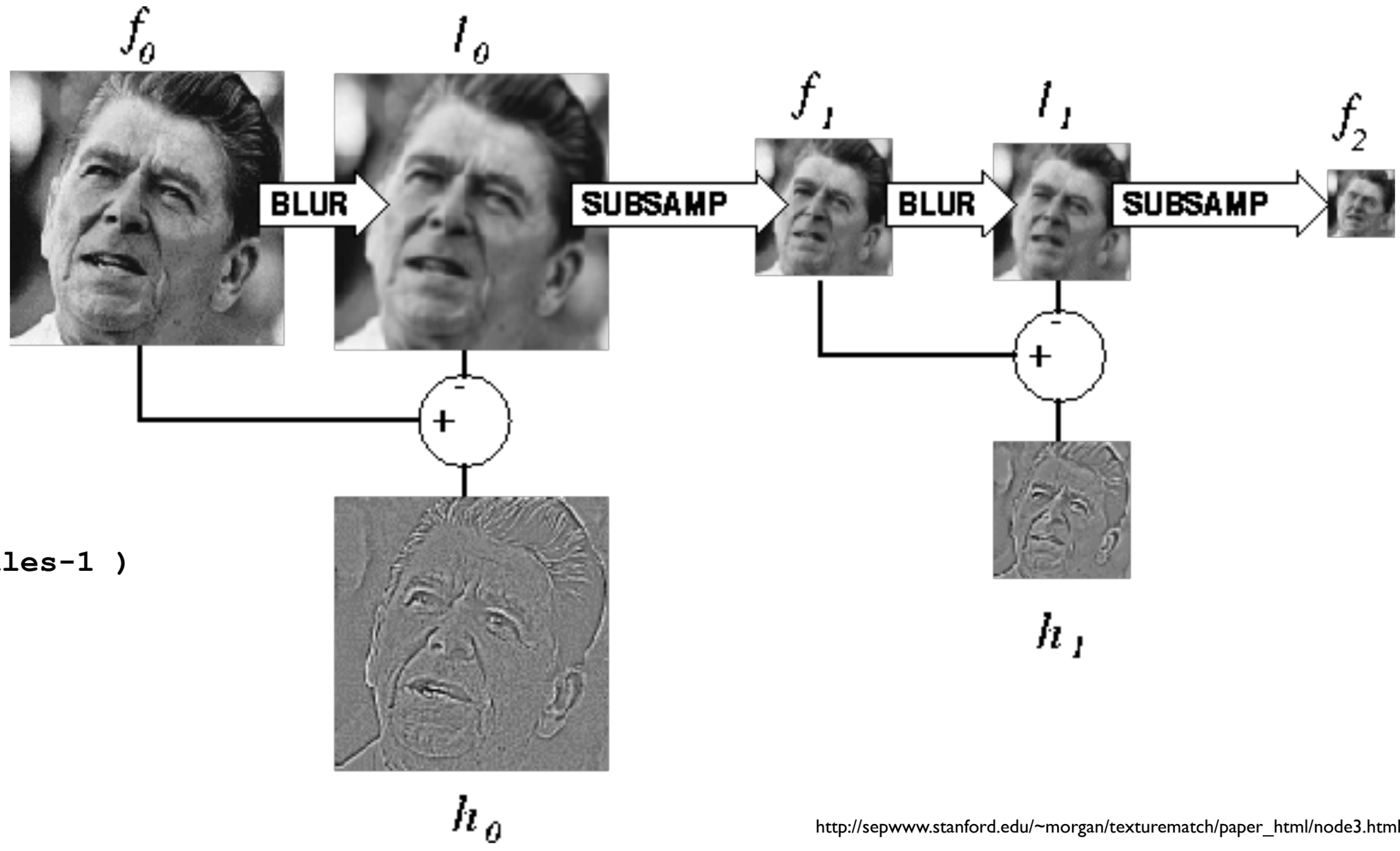
Constructing a Gaussian Pyramid

```
repeat
    filter
    subsample
until min resolution reached
```



Whole pyramid is only 4/3 the size of the original image!

Constructing the Laplacian Pyramid



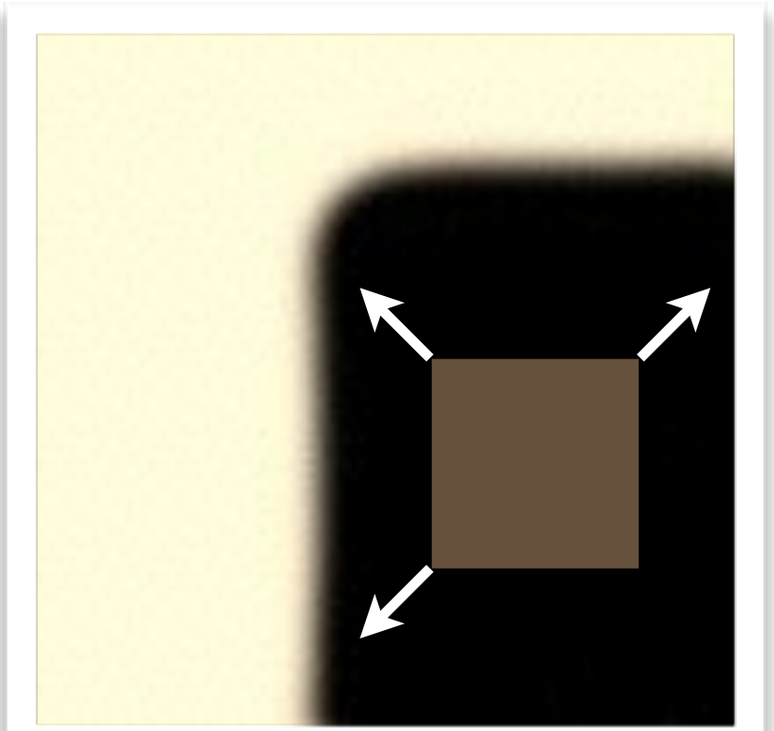

```
do( i = 0 : nScales-1 )
{
  blur(fi)
  subSamp2(li)
  hi = fi-li
}
```

http://sepwww.stanford.edu/~morgan/texturematch/paper_html/node3.html

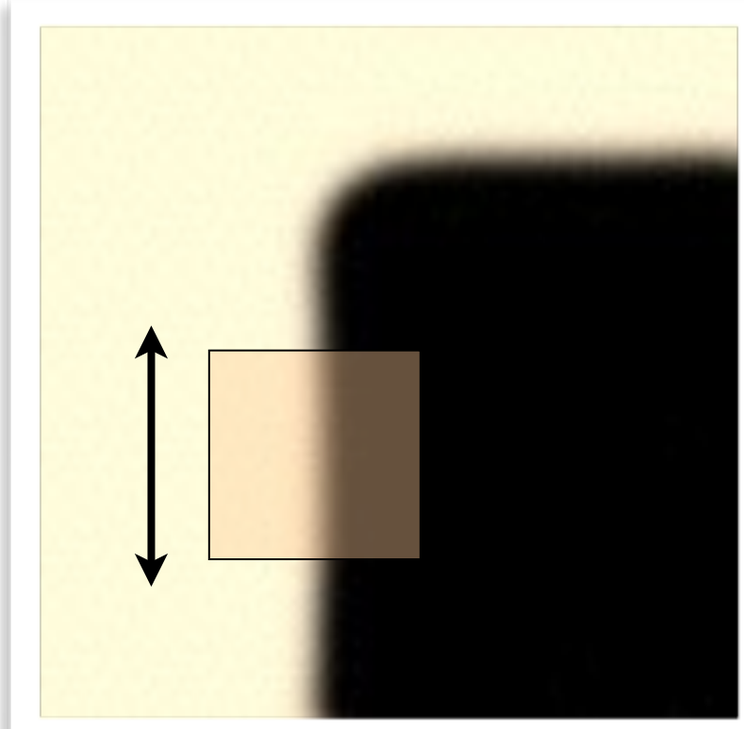
Feature Detector & Descriptor

Easily recognized by looking through a small window

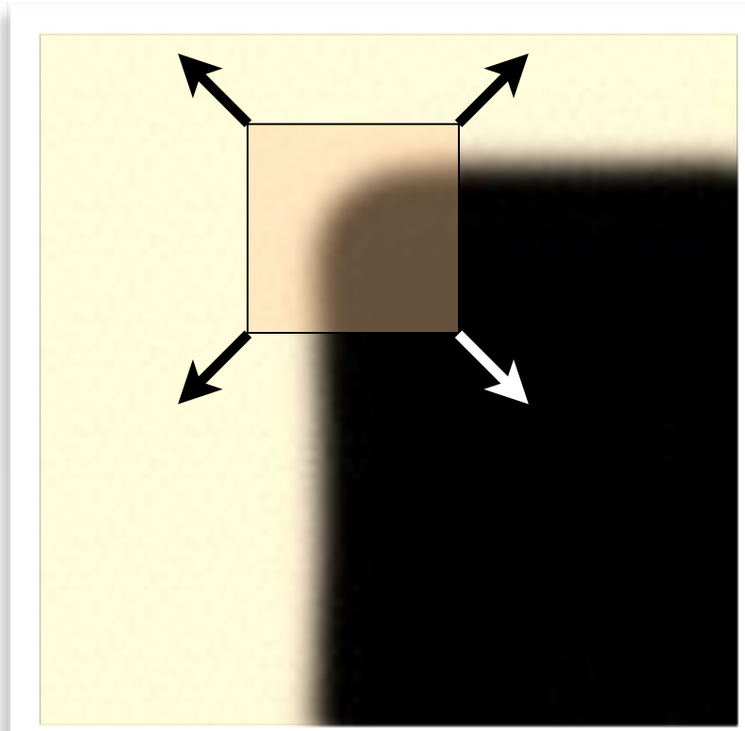
Shifting the window should give large change in intensity



“flat” region:
no change in all directions



“edge”:
no change along the edge direction



“corner”:
significant change in all directions

[Moravec 1980]

Error function approximation

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

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

Second-order Taylor expansion of E(u,v) about (0,0)
(bilinear approximation for small shifts):

$$E(u,v) \approx E(0,0) + \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2} \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

first derivative

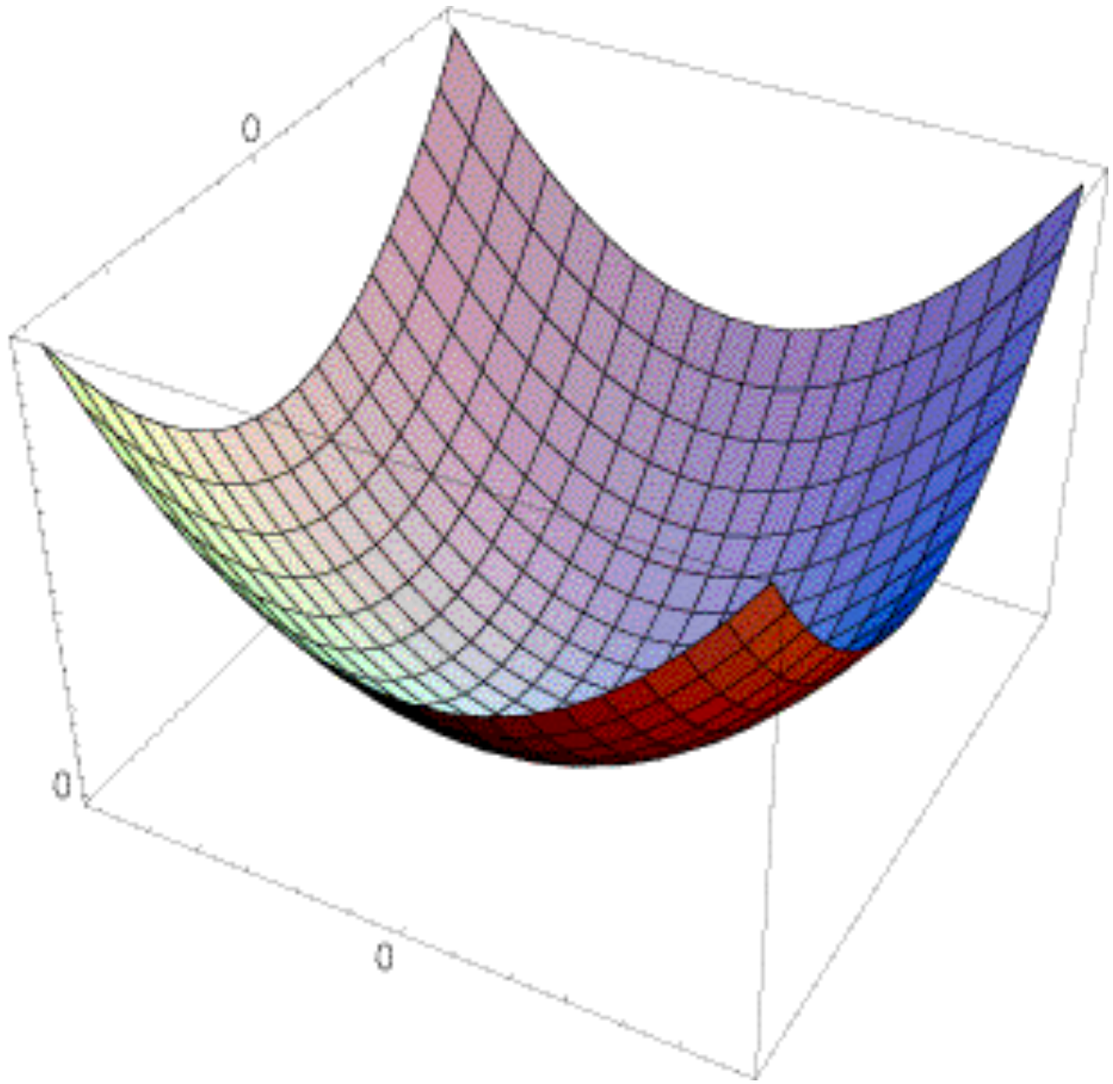
second derivative

Visualization of a quadratic

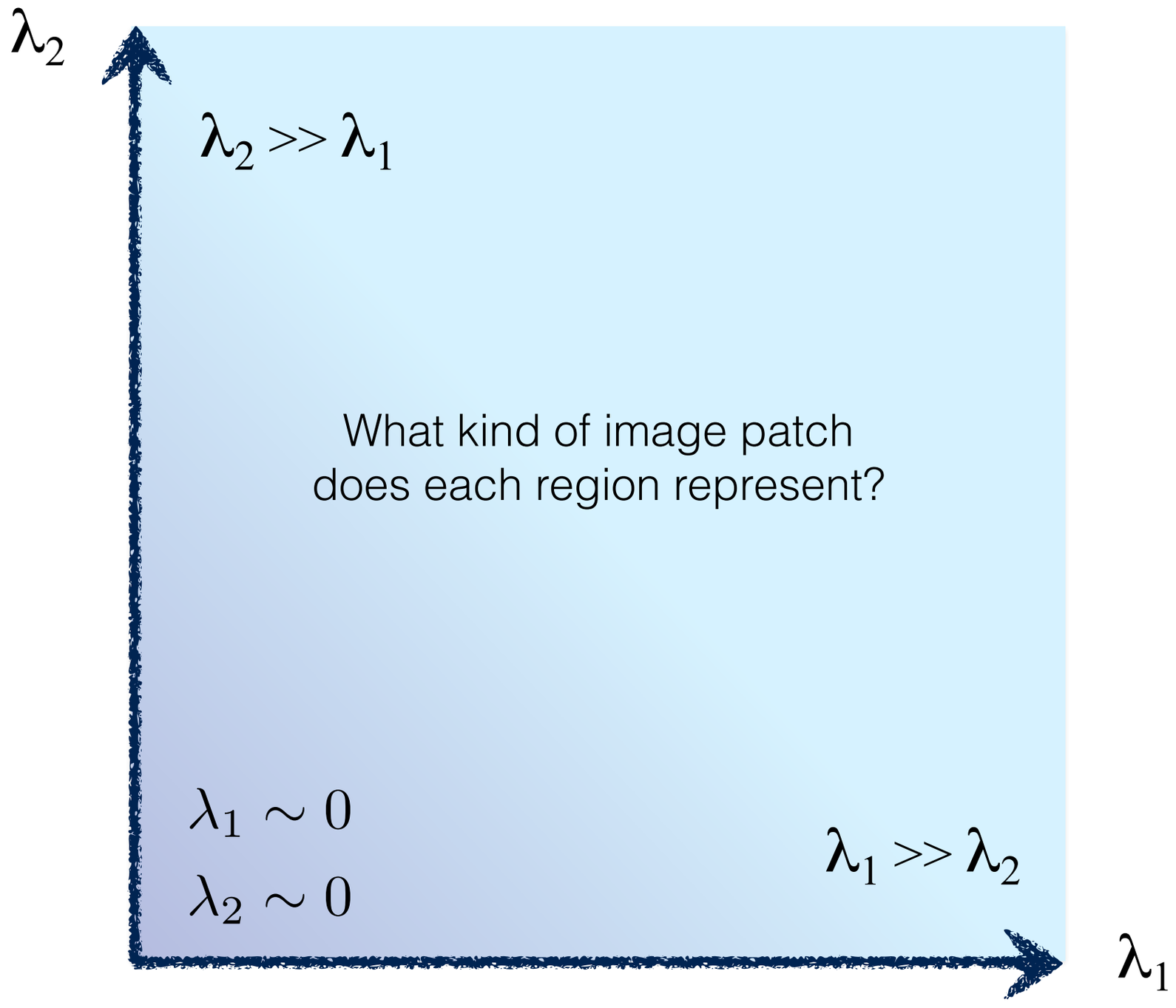
The surface $E(u,v)$ is locally approximated by a quadratic form

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

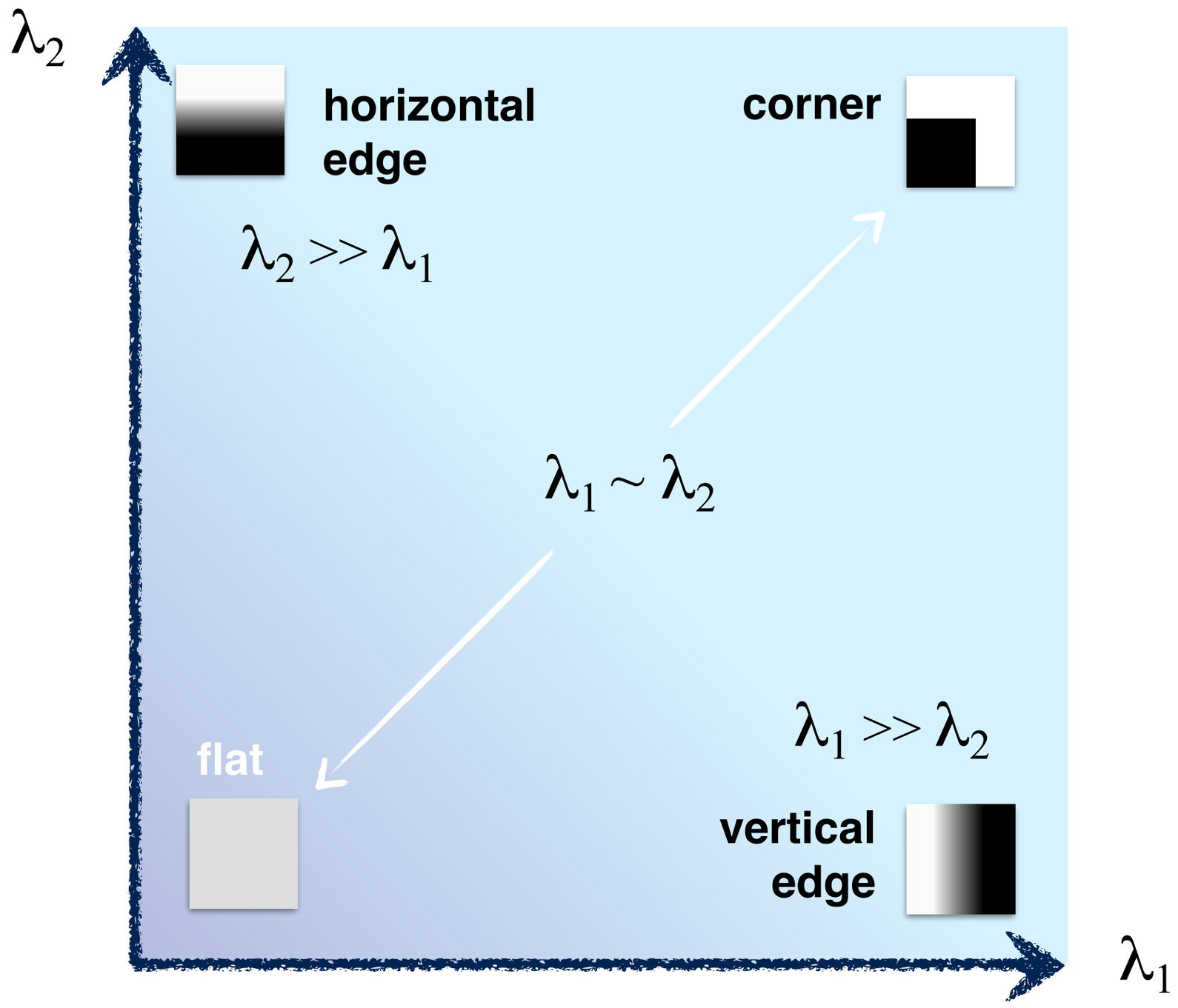
$$M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



interpreting eigenvalues



interpreting eigenvalues



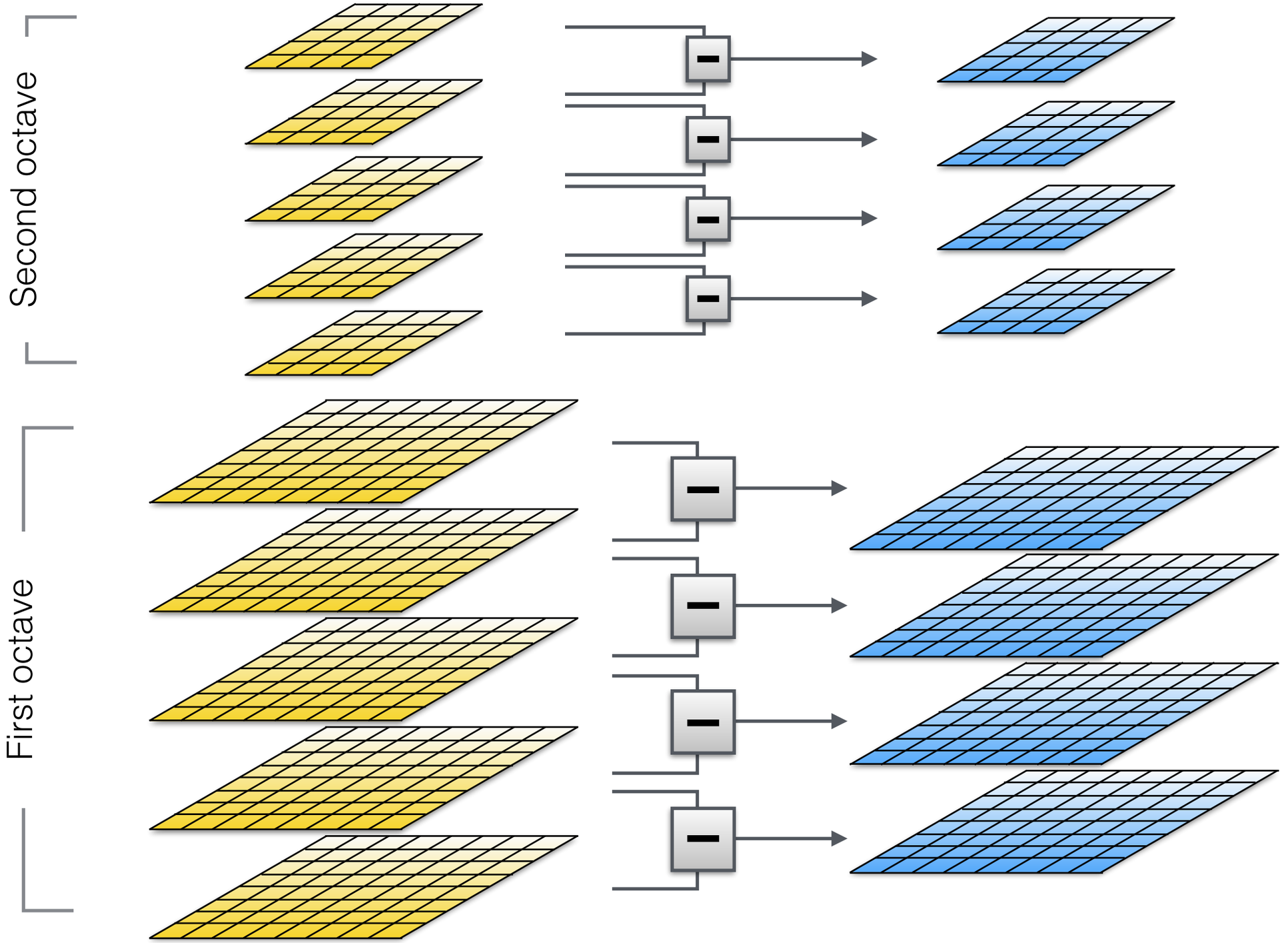
SIFT

(Scale Invariant Feature Transform)

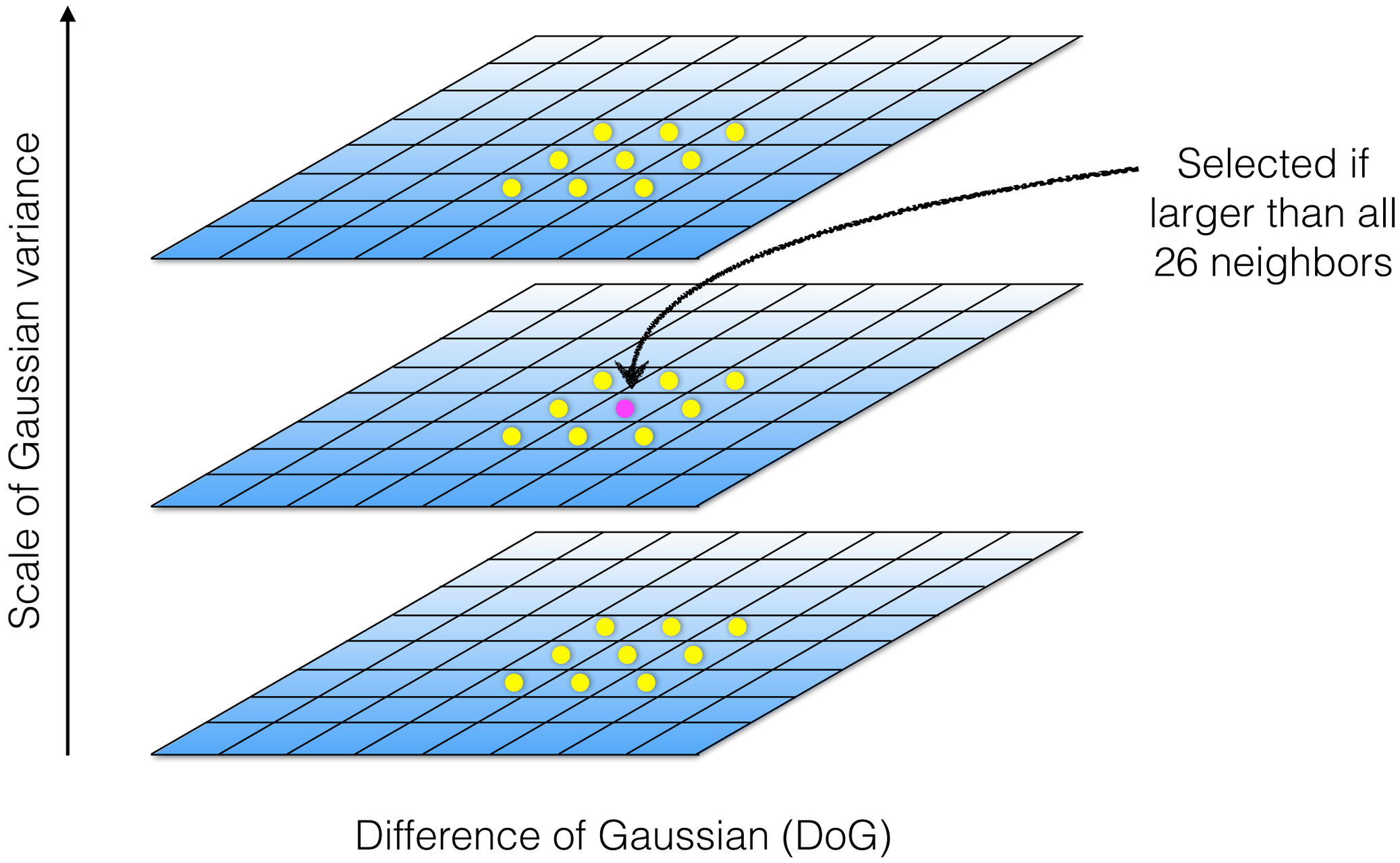
SIFT describes both a **detector** and **descriptor**

1. Multi-scale extrema detection
2. Keypoint localization
3. Orientation assignment
4. Keypoint descriptor

1. Multi-scale extrema detection



Scale-space extrema



2. Keypoint localization

2nd order Taylor series approximation of DoG scale-space

$$f(\mathbf{x}) = f + \frac{\partial f}{\partial \mathbf{x}}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 f}{\partial \mathbf{x}^2}\mathbf{x}$$

$$\mathbf{x} = \{x, y, \sigma\}$$

Take the derivative and solve for extrema

$$\mathbf{x}_m = -\frac{\partial^2 f}{\partial \mathbf{x}^2}^{-1} \frac{\partial f}{\partial \mathbf{x}}$$

Additional tests to retain only strong features

3. Orientation assignment

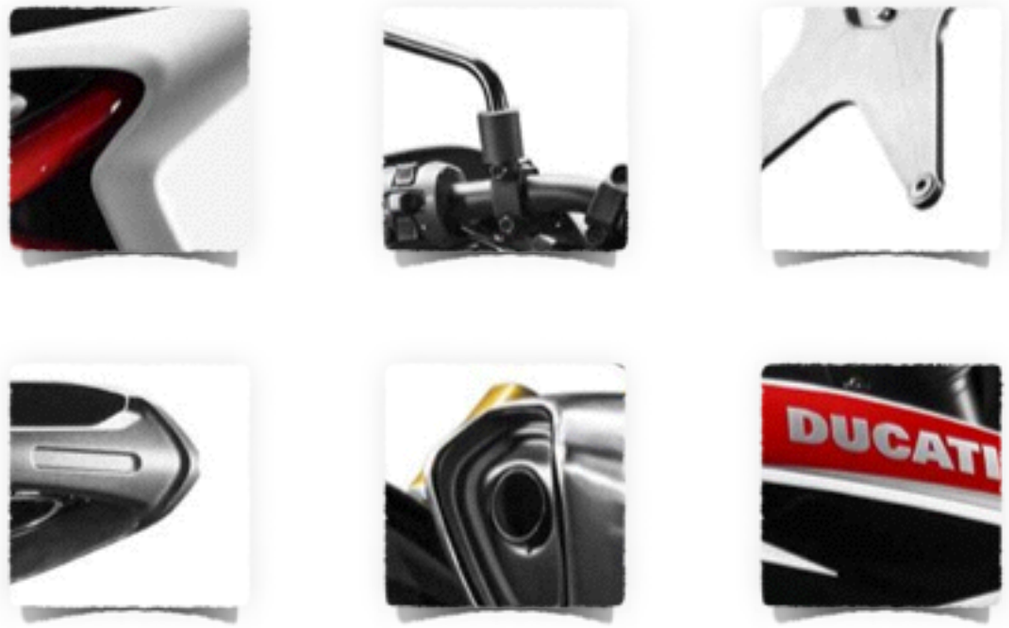
$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

1. Compute gradients over image patch
2. Compute histogram over gradients
3. Take the histogram bin (orientation) with the greatest count
4. Rotate the image patch by that orientation
5. Compute descriptor

Bag-of-Words

Some local feature are very informative



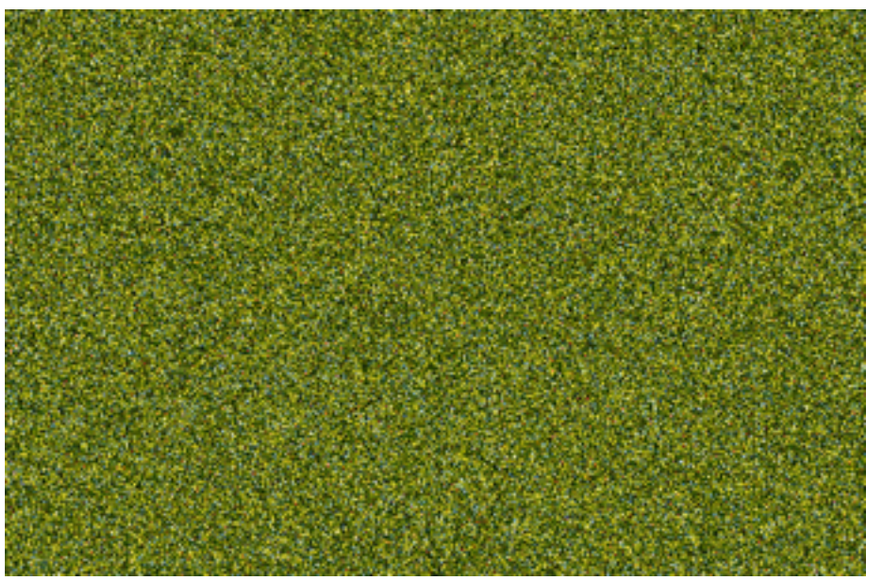
An object as

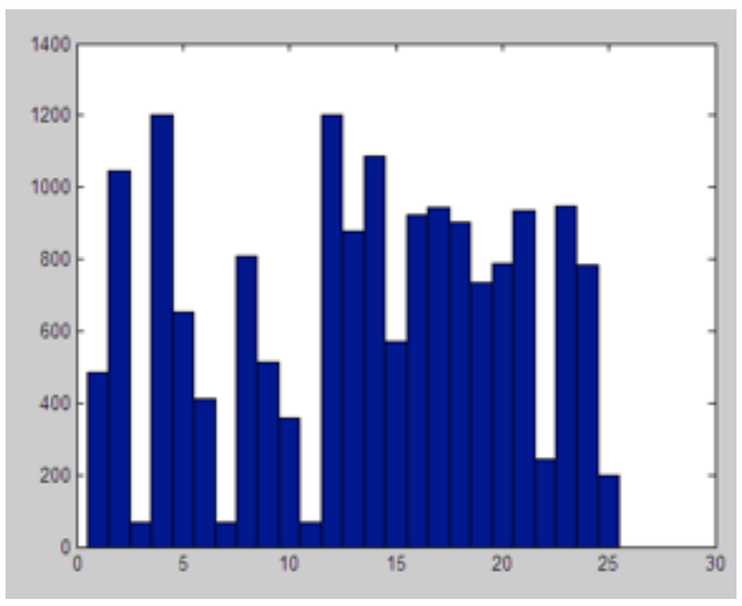
a collection of local features

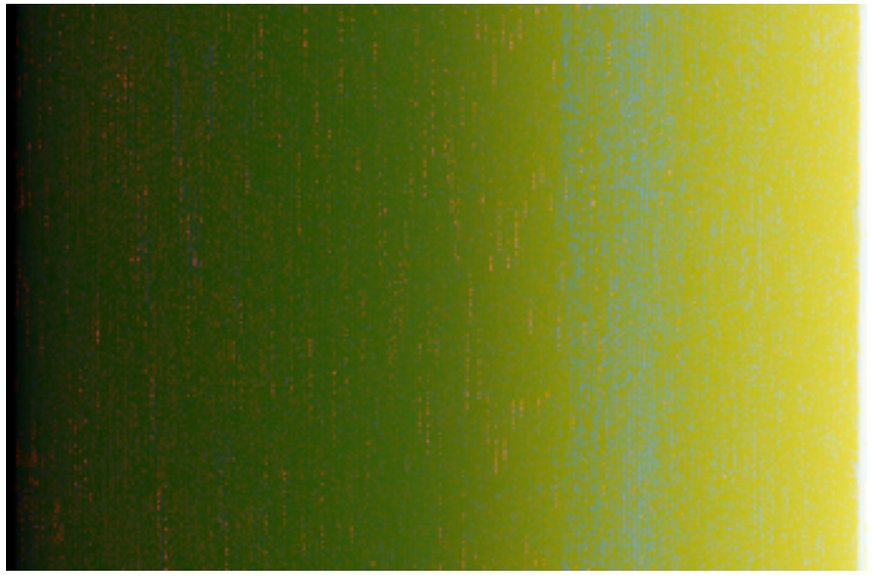
(bag-of-features)

- deals well with occlusion
- scale invariant
- rotation invariant

But what about layout?



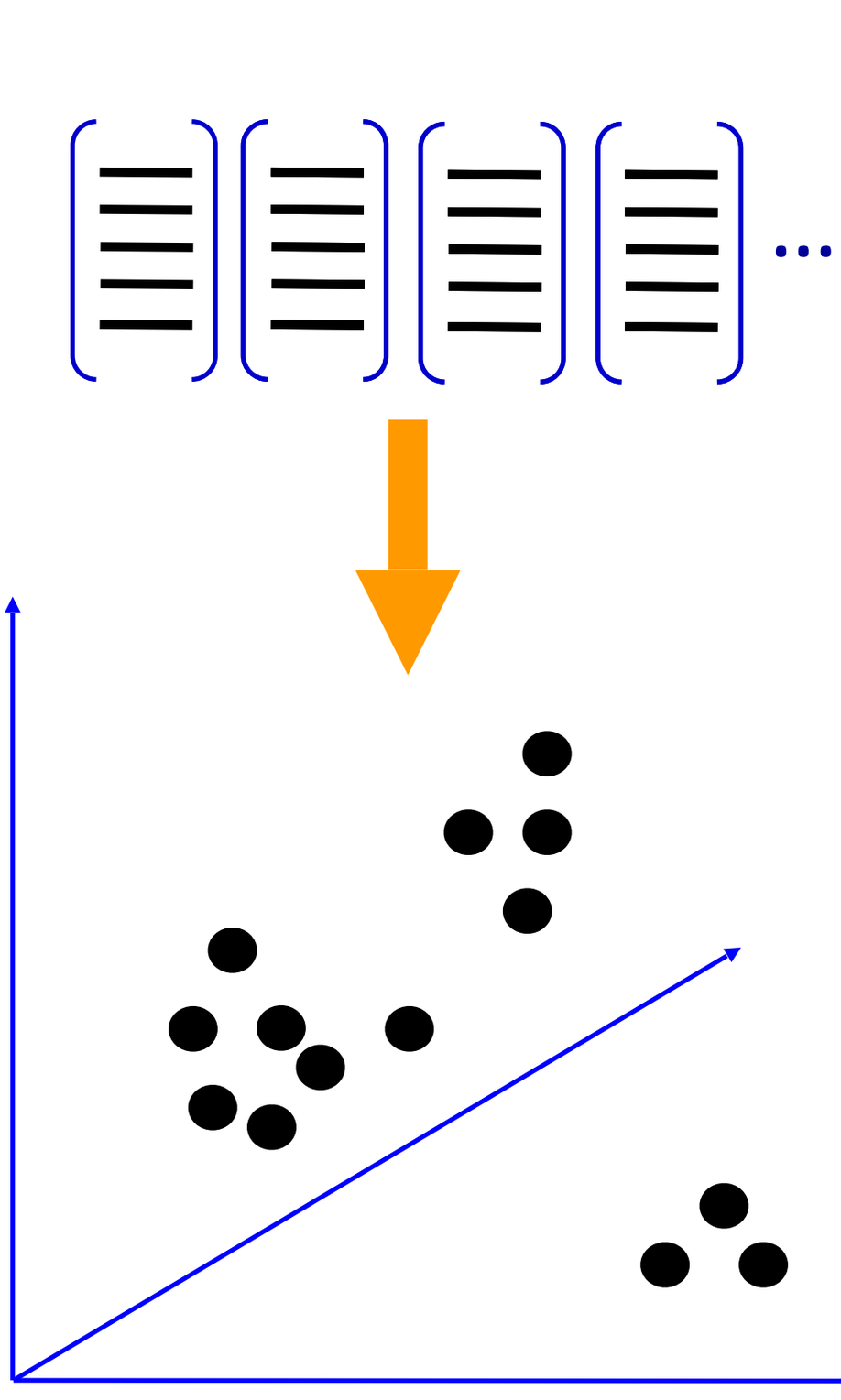


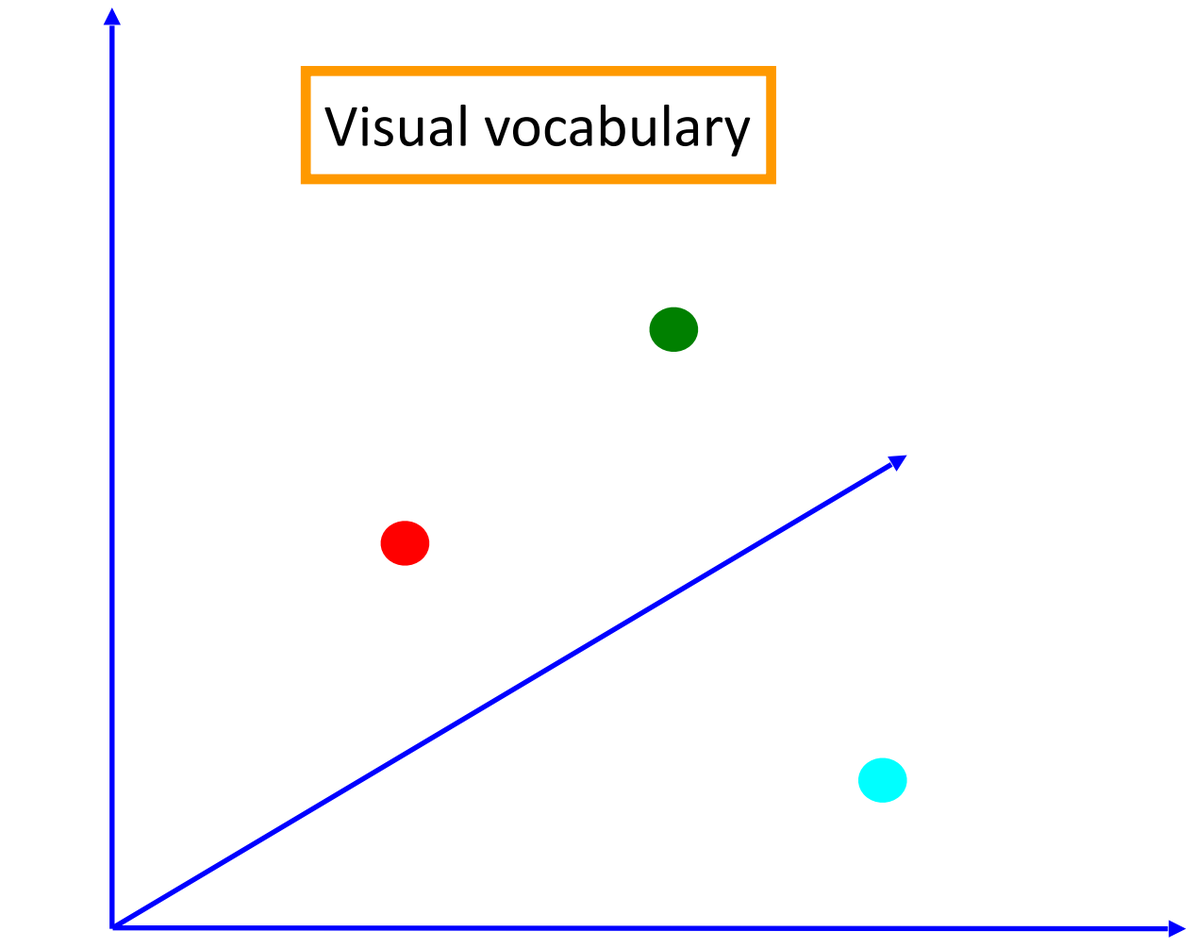


All of these images have the same color histogram

bag-of-words (histogram) representations remove spatial information

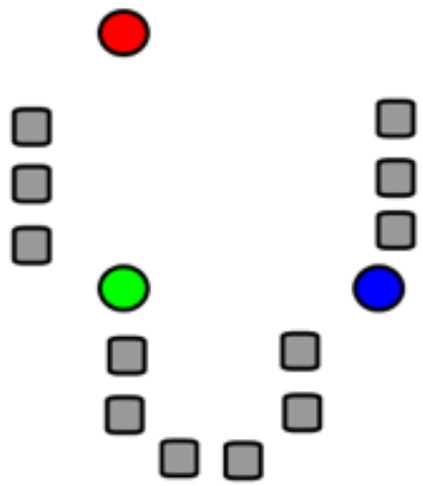
Do you know how to build a visual vocabulary?

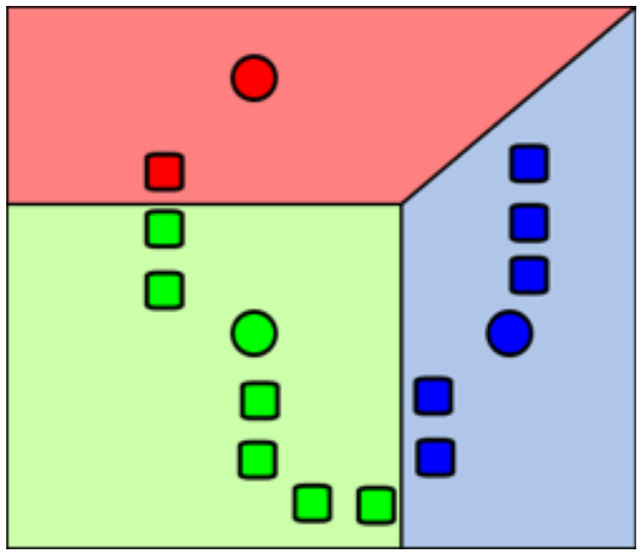


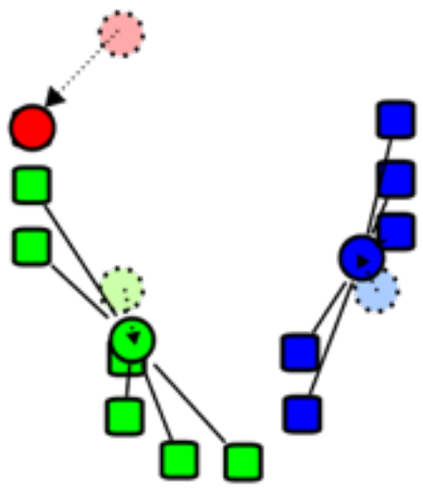


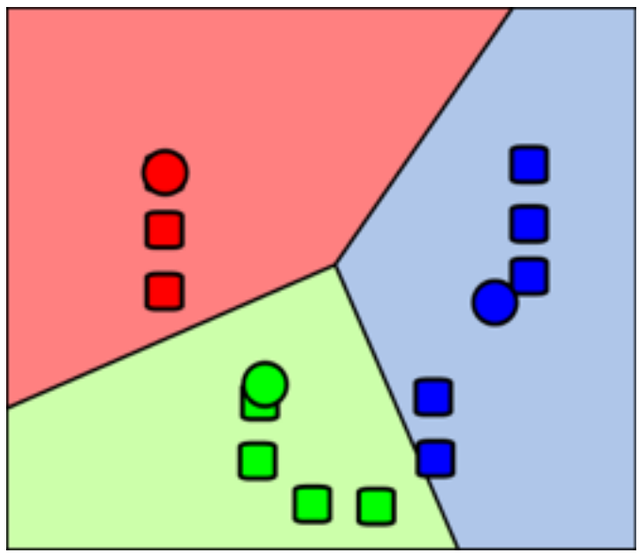
K-means Clustering

- Given k, the k-means algorithm consists of four steps:
- Select initial centroids at random.
- Assign each object to the cluster with the nearest centroid.
- Compute each centroid as the mean of the objects assigned to it.
- Repeat previous 2 steps until no change.



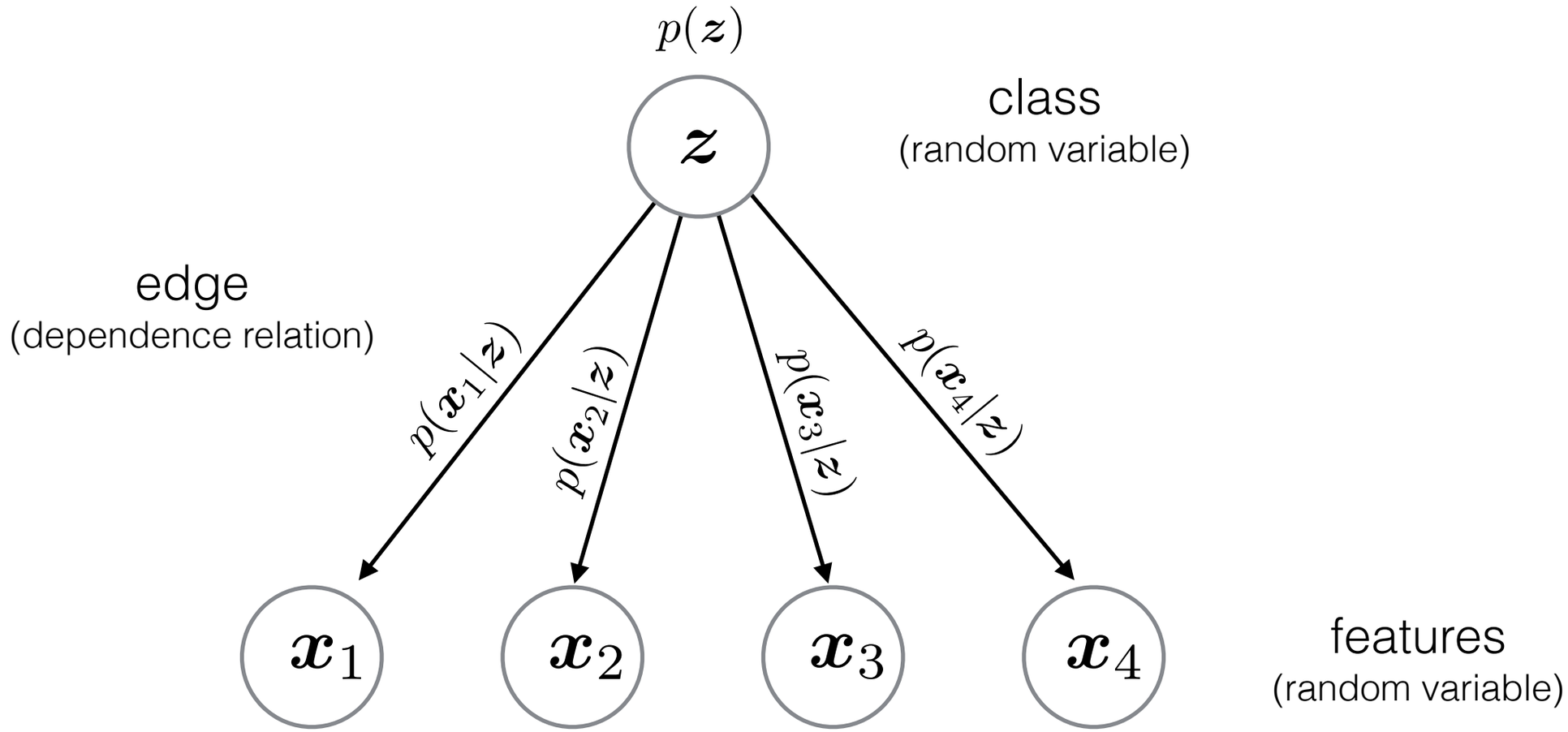






- How do you determine an optimal number of visual words?

Classification



Graphical model

This is called the posterior:
the probability of a class z given the observed features X

$$p(\boldsymbol{z}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$$

For classification, z is a discrete random variable
(e.g., car, person, building)

Each x is an observed feature
(e.g., visual words)

(it's a function that returns a single probability value)

The posterior can be decomposed according to

Bayes' Rule

$$\underset{\text{posterior}}{p(A|B)} = \frac{\overset{\text{likelihood}}{p(B|A)}\overset{\text{prior}}{p(A)}}{p(B)}$$

In our context…

$$p(\boldsymbol{z}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) = \frac{p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N|\boldsymbol{z})p(\boldsymbol{z})}{p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)}$$

The naive Bayes' classifier is solving this optimization

$$\hat{z} = \underset{z \in \boldsymbol{z}}{\arg\max}\, p(z|\boldsymbol{X})$$

MAP (maximum a posteriori) estimate

$$\hat{z} = \underset{z \in \boldsymbol{z}}{\arg\max}\, \frac{p(\boldsymbol{X}|z)p(z)}{p(\boldsymbol{X})}$$

Bayes' Rule

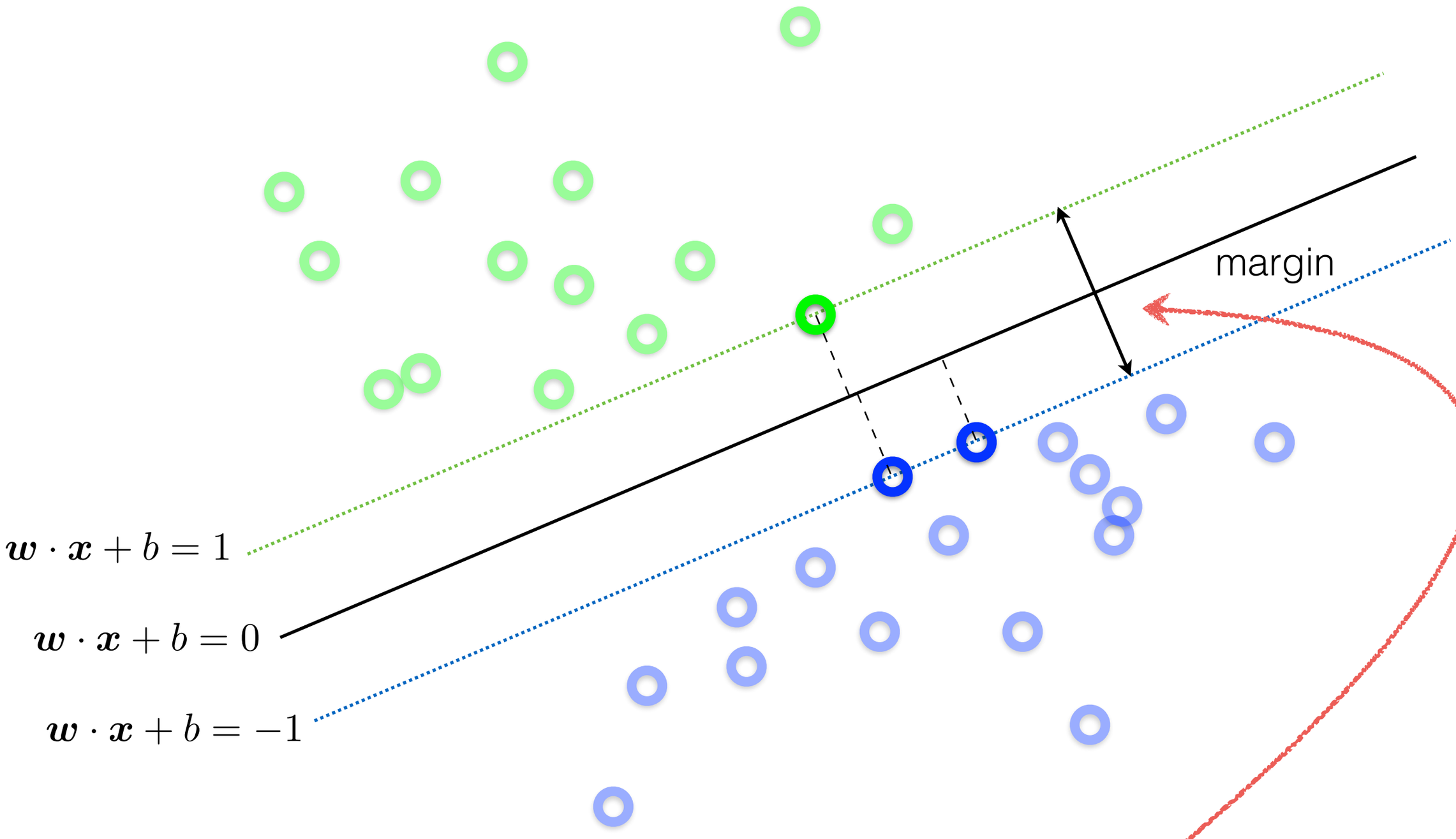
$$\hat{z} = \underset{z \in \boldsymbol{z}}{\arg\max}\, p(\boldsymbol{X}|z)p(z)$$

Remove constants

A naive Bayes' classifier assumes all features are
conditionally independent

$$
\begin{aligned}
p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N | \boldsymbol{z}) &= p(\boldsymbol{x}_1 | \boldsymbol{z}) p(\boldsymbol{x}_2, \ldots, \boldsymbol{x}_N | \boldsymbol{z}) \\
&= p(\boldsymbol{x}_1 | \boldsymbol{z}) p(\boldsymbol{x}_2 | \boldsymbol{z}) p(\boldsymbol{x}_3, \ldots, \boldsymbol{x}_N | \boldsymbol{z}) \\
&= p(\boldsymbol{x}_1 | \boldsymbol{z}) p(\boldsymbol{x}_2 | \boldsymbol{z}) \cdots p(\boldsymbol{x}_N | \boldsymbol{z})
\end{aligned}
$$

Find hyperplane **w** such that …



the gap between parallel hyperplanes $\frac{2}{\|\boldsymbol{w}\|}$ is maximized

Can be formulated as a maximization problem

$$\max_{\boldsymbol{w}} \frac{2}{\|\boldsymbol{w}\|}$$

$$\text{subject to } \boldsymbol{w} \cdot \boldsymbol{x}_i + b \begin{array}{l} \geq +1 \text{ if } y_i = +1 \\ \leq -1 \text{ if } y_i = -1 \end{array} \text{ for } i = 1, \ldots, N$$

Equivalently,

$$\min_{\boldsymbol{w}} \|\boldsymbol{w}\|$$

$$\text{subject to } y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) - 1 \leq 0 \text{ for } i = 1, \ldots, N$$

‘soft’ margin

objective

$$\min_{\boldsymbol{w},\boldsymbol{\xi}} \|\boldsymbol{w}\|^2 + C\sum_i \xi_i$$

subject to

$$y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i$$
$$\text{for} \quad i = 1, \ldots, N$$

- Every constraint can be satisfied if slack is large
- C is a regularization parameter
  - Small C: ignore constraints (larger margin)
  - Big C: constraints (small margin)
- Still QP problem (unique solution)