Lecture 2: Feature Extraction

Motivation

- Ideal properties of features used for object recognition

Global Image Patch Descriptors

- Histograms & Templates
- Histograms of filter responses

Recent Image Patch Descriptors

- SIFT
- HOG

The Search Problem

- Sliding window approach

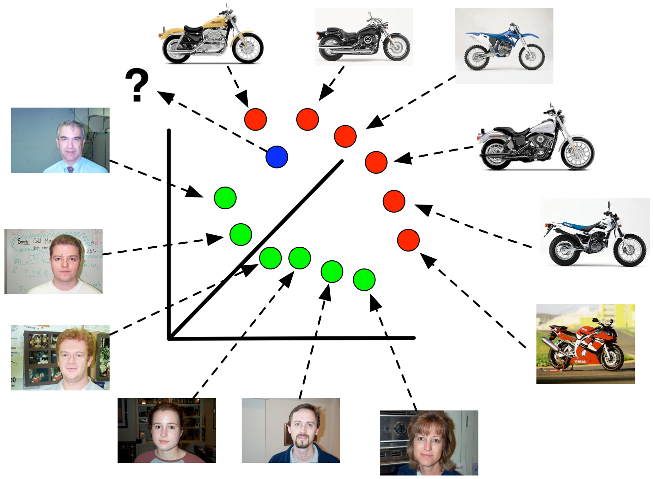
Motivation For Today's Lecture

Recap: Approach to recognition

Assume: An object class is represented by a set of exemplar images.

For recognition: Compare a novel image to this set of labelled images via an intermediary feature vector representation.

Thus: No explicit 3D model of the object or the physics of imaging is required.



Focus of Today's Lecture

Consider the type of features that have been used for this task.

Properties of an ideal feature

In an ideal world, our feature descriptor would have invariance to?

Properties of an ideal feature

In an ideal world, our feature descriptor would be invariant to

Viewpoint changes

- translation
- scale changes
- out-of-plane rotations

Illumination changes

Clutter (a computer vision term for the other stuff in the image that does not correspond to what you are looking for and corrupts your measurements and confuses your detector)

Partial occlusions

Intra-class variation

while also being....

Properties of an Ideal Feature

In an ideal world, our feature descriptor would be invariant to

Viewpoint changes

- translation
- scale changes
- out-of-plane rotations

Illumination changes

Clutter (a computer vision term for the other stuff in the image that does not correspond to what you are looking for and corrupts your measurements and confuses your detector)

Partial occlusion

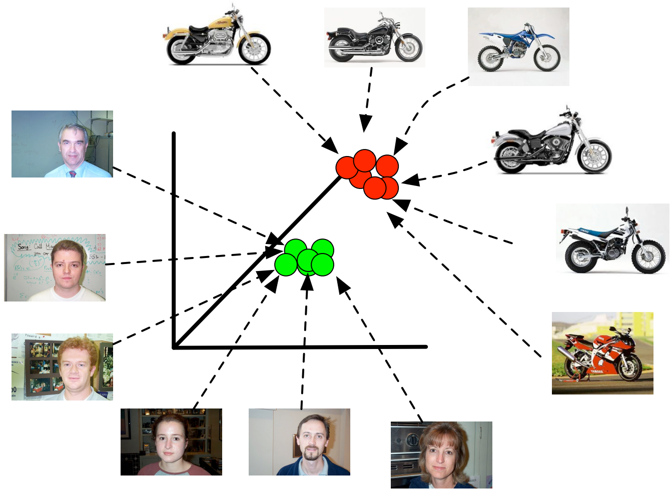
Intra-class variation

while also being....

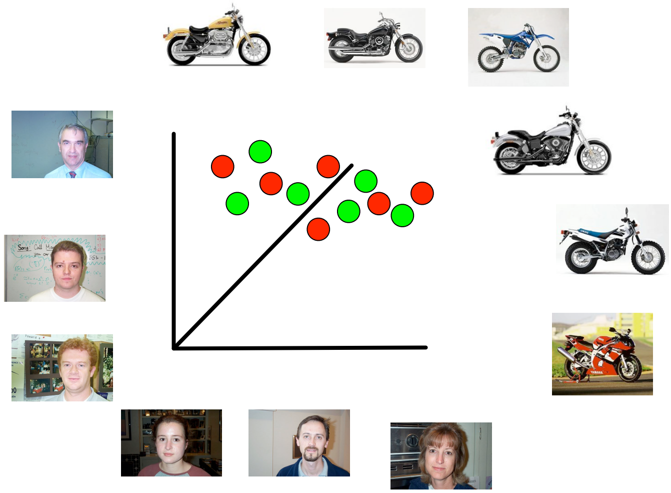
Distinctive - features extracted from car images differ to those extracted from chair images.

Fast to compute

The two extremes



Ideal features



Far from ideal

Back in the real world

Must forget about the nirvana of ideal feature descriptors.

Therefore, must strike a balance between:

build invariances into the descriptor

while

incorporate the modes of variation not accounted for by the descriptor into the training data and the search

Back in the real world

In object recognition this is acheived via a smart trade off of

feature descriptor

AND

classifier/recognition algorithm

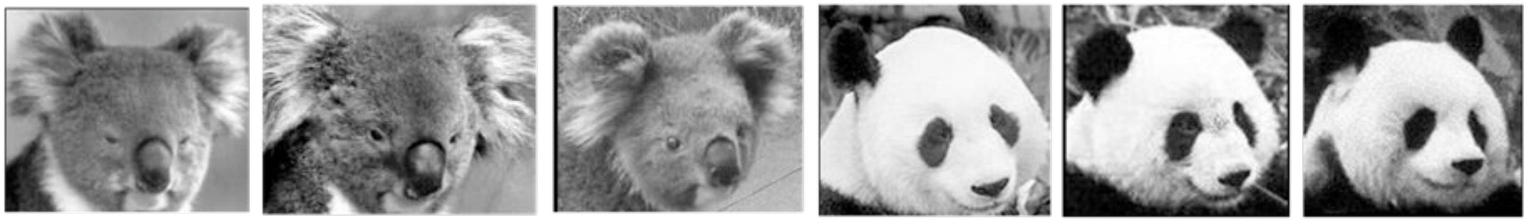
AND

adequate training data

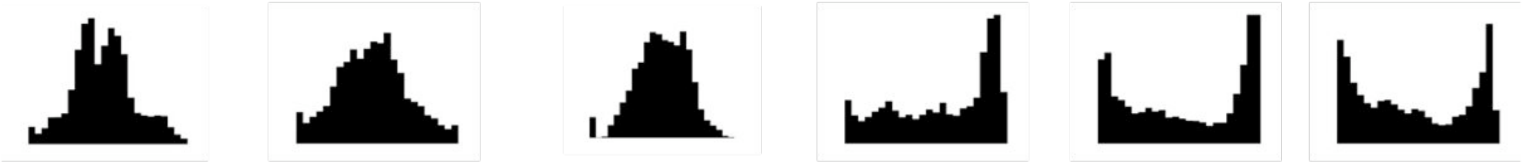
Image Patch Description

Global image patch descriptions

Template (Array of pixel intensities) Fixed spatial grid
not invariant to geometric transforms



Grayscale/Colour Histogram Invariant to geometric transforms.



Definition of 1D histogram

Given

- N data points with scalar values $f_1, \ldots, f_N$ with each $f_i \in \mathcal{R}$.
- m intervals/bins defined by the points $b_0, b_1, \ldots, b_m$ where $b_i < b_{i+1}$.

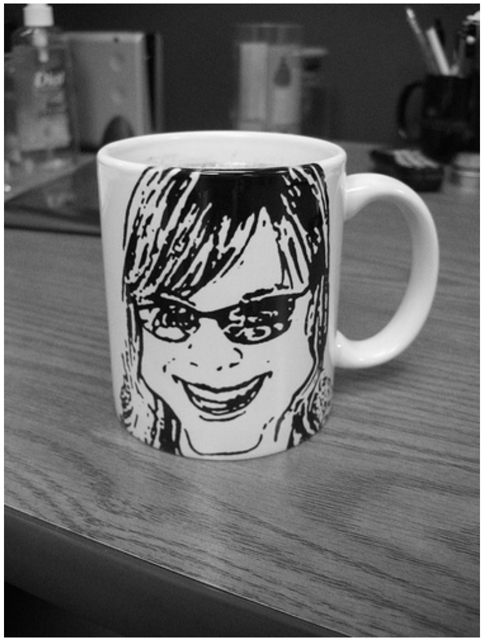
Histogram definition

Histogram $\mathbf{h} = (h_1, \ldots, h_m)$ records the number of points f_j falling into each bin.

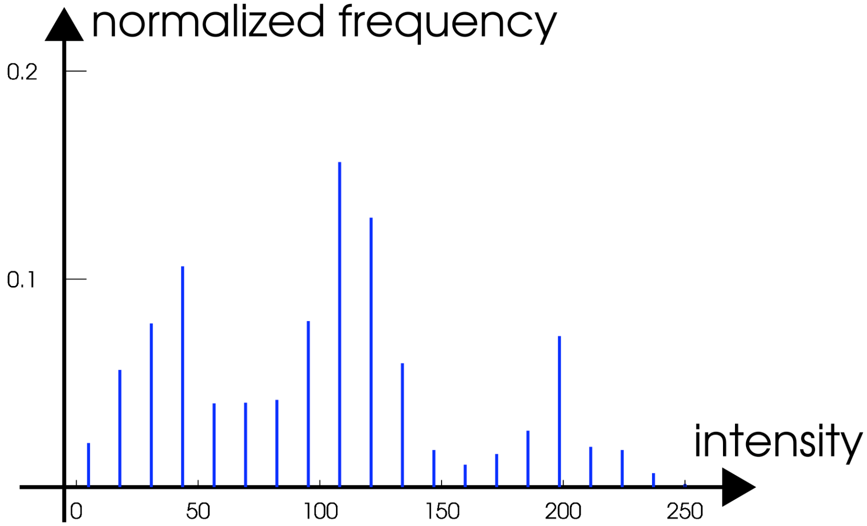
Calculation of histogram

```
Set $\mathbf{h} = (0, \ldots, 0)$ then
for $i = 1$ to $N$
  find the $j$ such that $f_i \in [b_{j-1}, b_j)$ and set $h_j = h_j + 1$
end
```

Example: intensity histogram



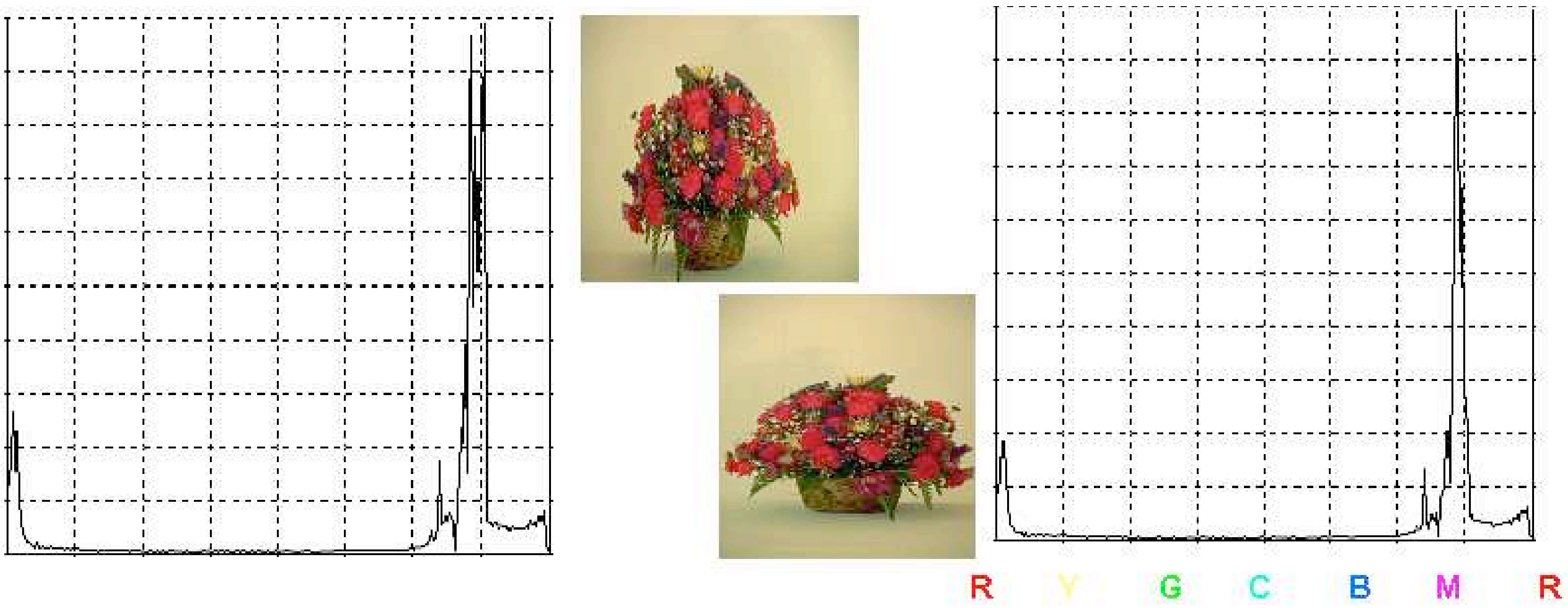
Image



histogram of intensity values

Which geometric transformations of the mug wouldn't change this histogram?

Histogram of hue



Hue values of the pixels is plotted against its frequency.

Weighted histogram

Given

Sometimes the N data points $f_1, \ldots, f_N$ also have non-negative weights $w_1, \ldots, w_N$ associated with them.

Weighted histogram definition

The weighted histogram $\mathbf{h} = (h_1, \ldots, h_m)$ then records the sum of the weights of the points f_j that fall into each bin.

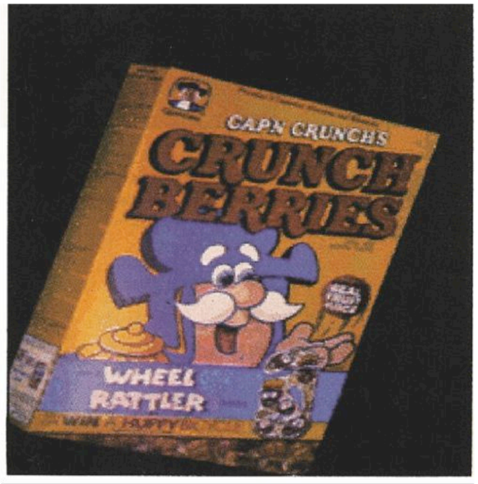
Calculate as follows

```
Set $\mathbf{h} = (0, \ldots, 0)$ then
for $i = 1$ to $N$
  find the $j$ such that $b_j \leq f_i < b_{j+1}$ and set $h_j = h_j + w_j$
end
```

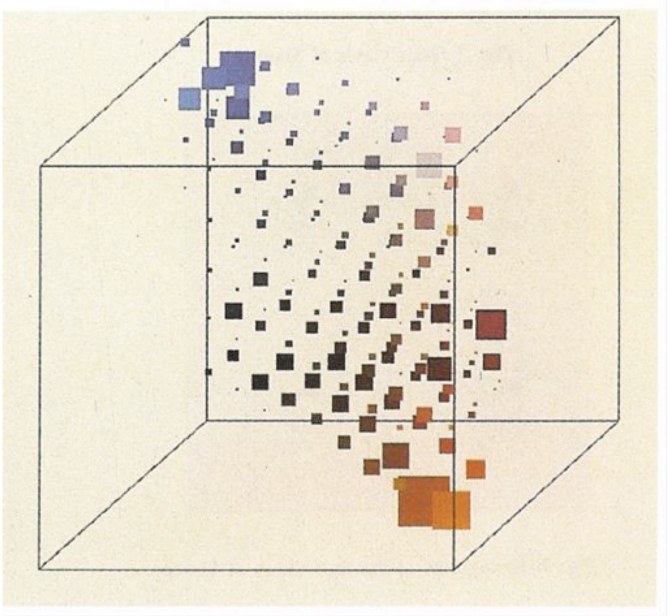
Multi-dimensional histogram

Can also have multi-dimensinal histograms. Here's an example of histogramming the RGB values in an image.

Compute 3D histogram as $h(r, g, b) = \#(\text{pixels with color } (r, g, b))$



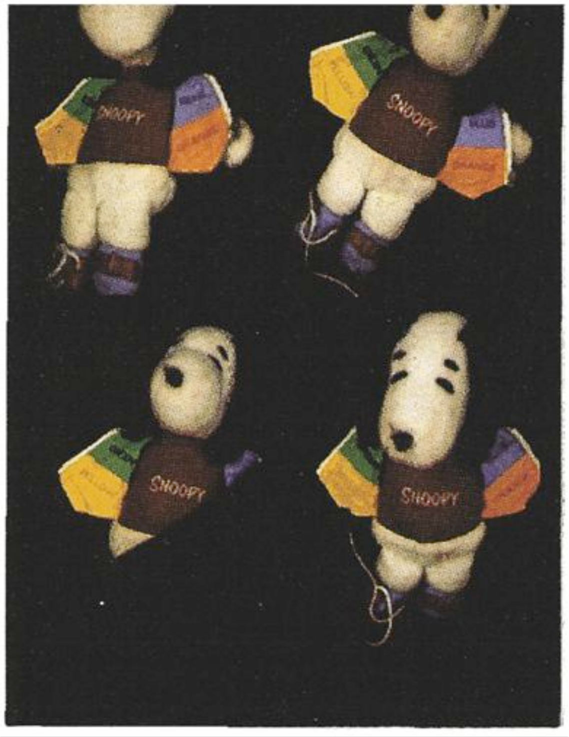
image



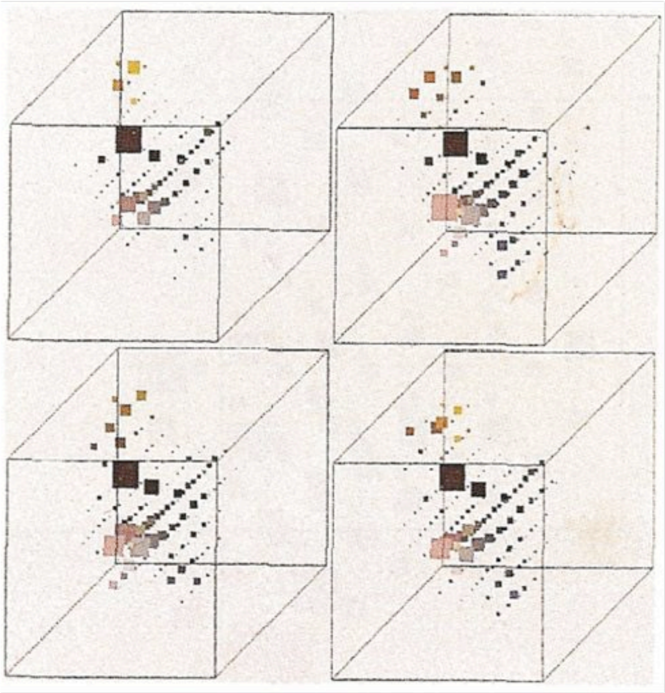
3D Histogram

Colour histograms

Pros: Robust to geometric transforms **and** partial occlusions.



images

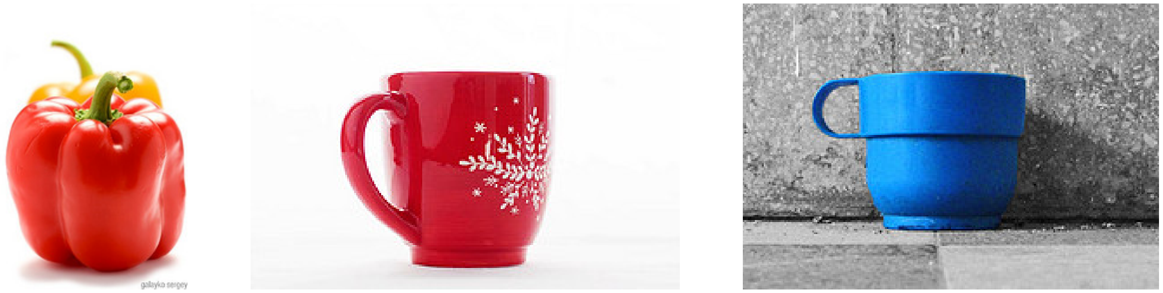


3D Histograms

Colour histograms

Cons

- May be sensitive to illumination changes (**can sometimes fix**)
- Many different images will have very similar histograms **and** many objects from the same class will have very different histograms. (**perhaps fatal**)



Do we have a problem?

Other potential clashes?

Colour normalisation

One component of the 3D color space is intensity

- If a color vector is multiplied by a scalar, the intensity changes, but not the color itself.
- This means colors can be normalized by the intensity defined by $I = R + G + B$
- **Chromatic representation**:

$$r = \frac{R}{R+G+B}, \quad g = \frac{G}{R+G+B}, \quad b = \frac{B}{R+G+B}$$

- If this normalization is used then you've made your data 2-dimensional. So only need r, g for the description task.

Can histogram other things besides colours and intensity values

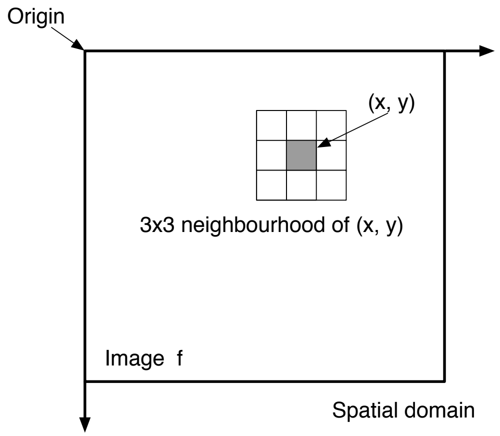
Remember: Spatial filtering

Spatial domain process denoted by

$$g(x, y) = T[f(x, y)]$$

where $f(x, y)$ is the input image, $g(x, y)$ is the output image and T is an operator on f defined over a neighbourhood of point (x, y).

Example: A 3×3 neighbourhood about a point (x, y) in an image in the spatial domain. Neighbourhood is moved from pixel to pixel in the image to generate an output image.



Form a new image whose pixels are a function of the original pixel values.

Spatial filtering

A spatial filter consists of a

1. *neighbourhood*

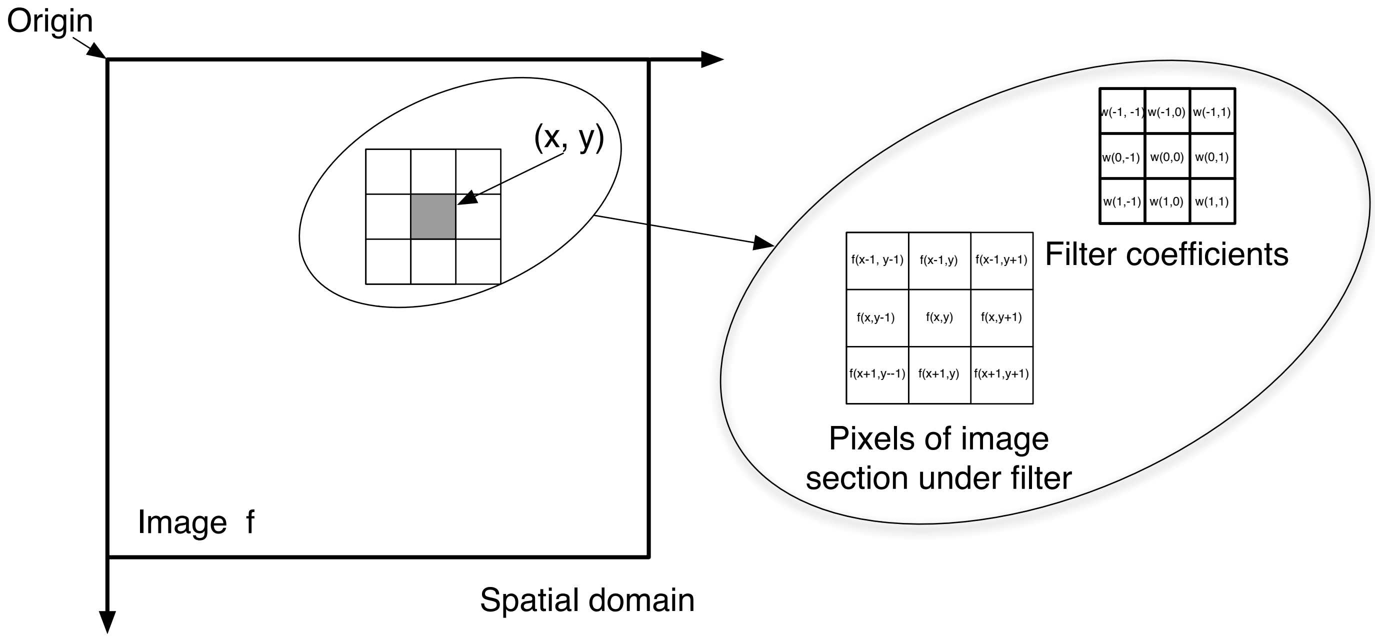
2. *predefined operation* performed on the image pixels within the neighbourhood.

If the operation performed is linear, then the filter is called a *linear spatial filter* and will be of the form

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)\, f(x+s, y+t)$$

for a mask of size $m \times n$ where $m = 2a+1$ and $n = 2b+1$. Generally, have filters of odd size so the filter centre falls on integer values.

Spatial filtering



Origin

(x, y)

Image f

Spatial domain

w(-1, -1)	w(-1,0)	w(-1,1)
w(0,-1)	w(0,0)	w(0,1)
w(1,-1)	w(1,0)	w(1,1)

Filter coefficients

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

Pixels of image section under filter

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)\, f(x+s, y+t)$$

Spatial correlation and convolution

Correlation Move a filter over the image and compute the sum of products at each location as explained.

$$w(x,y) \star f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)\, f(x+s, y+t)$$

Convolution Same as correlation except the filter is first rotated by 180°.

$$w(x,y) * f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)\, f(x-s, y-t)$$

Spatial filter masks

The coefficients w_j's define the effect of applying the filter.

Example 1:

$$\text{Let } w_j = 1 \quad \text{for } j = 1, \ldots, nm$$

$$\Rightarrow g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} f(x+s, y+t)$$

$\Rightarrow g(x,y) =$ sum of pixel intensities in the $n \times m$ neighbourhood.

1	1	1
1	1	1
1	1	1

What values of w_j make $g(x,y)$ the average intensity of a pixel in the $n \times m$ neighbourhood?

Example 2: *Box filter* (a smoothing filter)

$$\text{Let } w_j = \frac{1}{nm} \quad \text{for } j = 1, \ldots, nm$$

$$\Rightarrow g(x,y) = \frac{1}{nm} \sum_{s=-a}^{a} \sum_{t=-b}^{b} f(x+s, y+t)$$

$\Rightarrow g(x,y) =$ average intensity of a pixel in the $n \times m$ neighbourhood.

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

Smoothing linear filters

The output of a smoothing linear spatial filter is simply the weighted average of the pixels in the neighbourhood of the filter mask. These filters are often referred to as weighted averaging filters and act as *lowpass filters*.

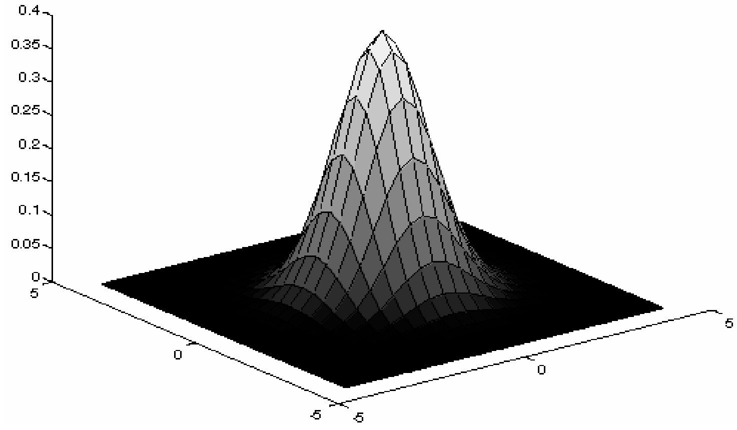
$$g(x,y) = \frac{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)\, f(x+s, y+t)}{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)}$$

Note each $w(s,t) \geq 0$ and if $w'(s,t) = w(s,t)/\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)$ then $\sum_{s,t} w'(s,t) = 1$.

Smoothing blurs an image. Hopefully reduces the amount of *irrelevant* detail in an image, but may destroy boundary edge information.

Example: Gaussian filter

$$w(s,t) \propto \exp\left(-\frac{s^2+t^2}{2\sigma^2}\right)$$

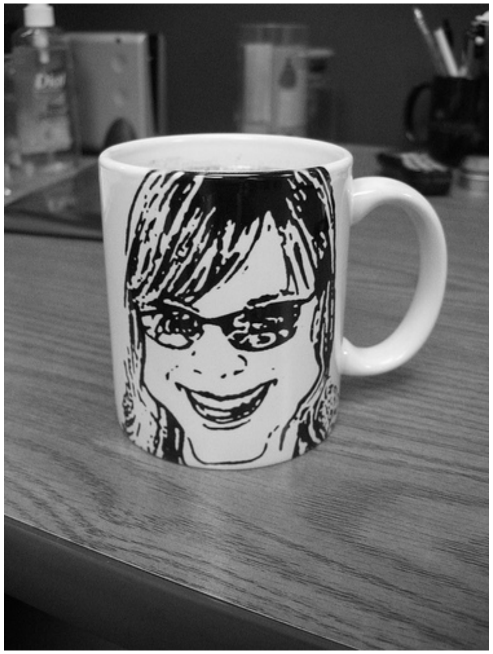


In theory, the Gaussian function is non-zero everywhere, which would require an infinitely large convolution kernel, but in practice it is effectively zero more than about three standard deviations from the mean, and so we can truncate the kernel at this point.

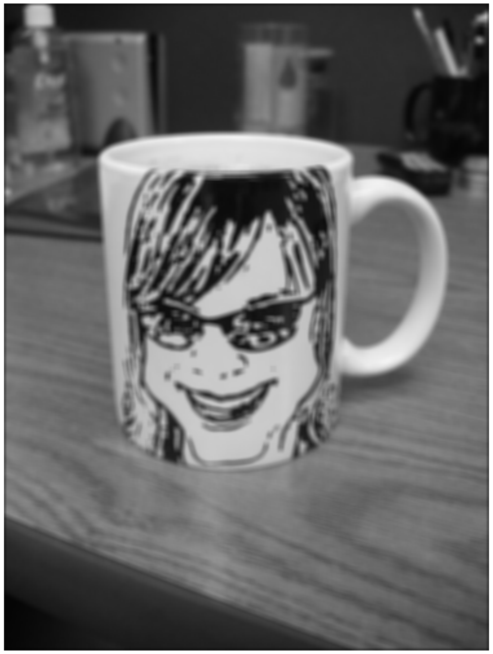
As a crude rule of thumb if

- have a square mask $n \times n$ with $n = 2a + 1$ then set $\sigma = \frac{a}{2}$.
- want to smooth the image by a Gaussian with $\sigma \Rightarrow$ filter mask of size $a = 2\sigma$

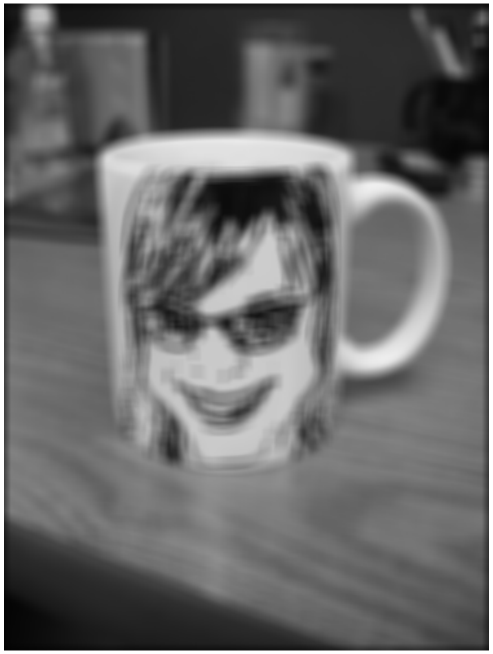
Example



Original image



Smoothed image



Very smoothed image

Edges

Edges are significant local **changes** of **intensity** in an image.

Causes of these intensity changes:

Geometric events

- object boundary (discontinuity in depth and/or surface color and texture)
- surface boundary (discontinuity in surface orientation and/or surface color)

Non-geometric events

- specularity (direct reflection of light, such as a mirror)
- shadows (from other objects or from the same object)
- inter-reflections

Want to construct linear filters that respond to such edges.

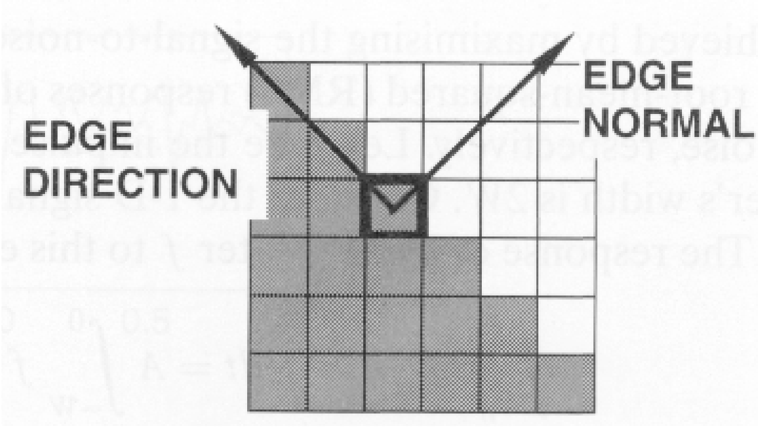
Edge description

Edge normal unit vector in the direction of maximum intensity change.

Edge direction unit vector to perpendicular to the edge normal.

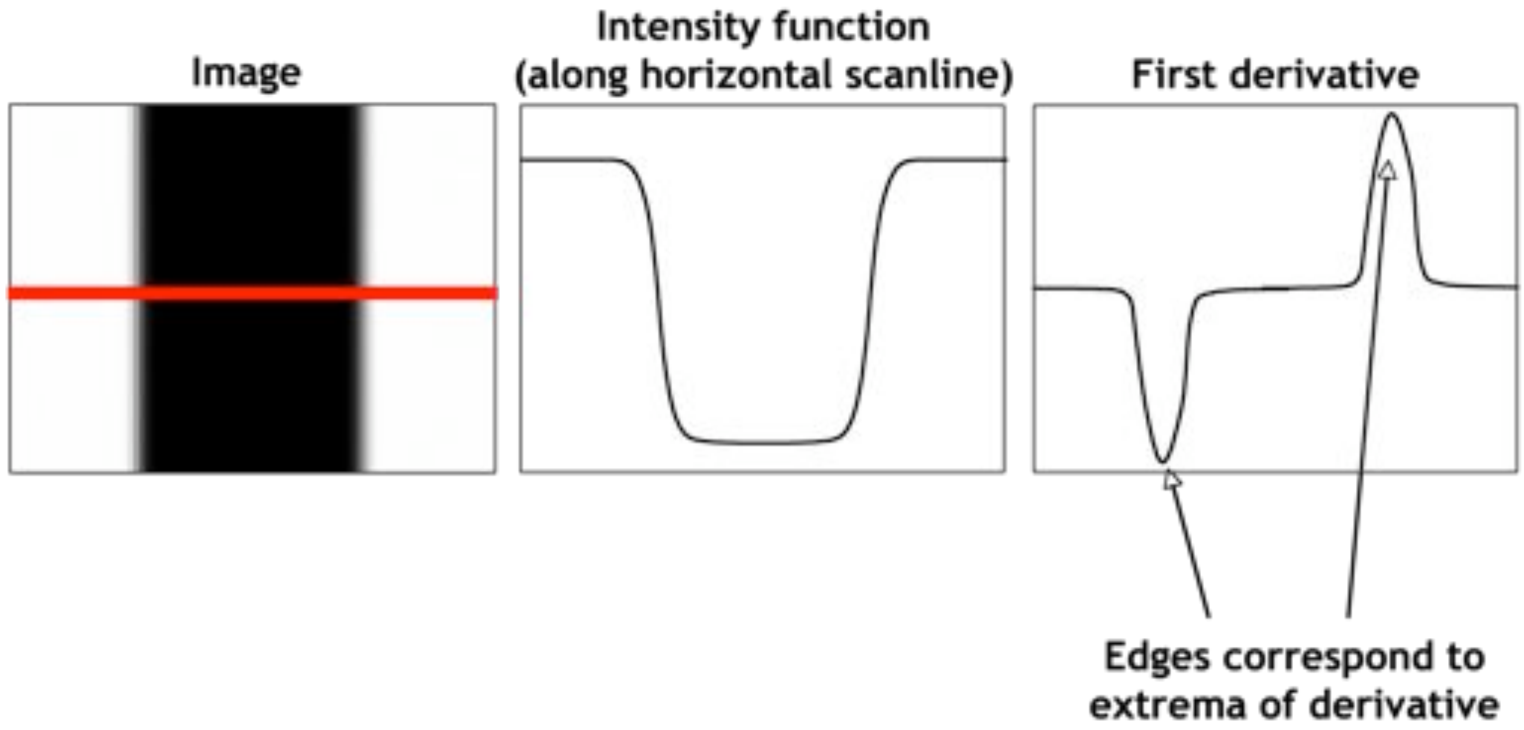
Edge position or center the image position at which the edge is located.

Edge strength related to the local image contrast along the normal.



Derivatives and edges

An edge is a place of rapid change in the image intensity function.



Derivatives and edges

Calculus describes changes of continuous functions using **derivatives**.

An image is a 2D function, so operators describing edges are expressed using **partial derivatives**. Can approximate the derivative of a discrete signal by finite differences.

$$\begin{aligned}\frac{\partial f(x,y)}{\partial x} &= \lim_{h\to 0}\frac{f(x+h,y)-f(x,y)}{h}\\ &\approx f(x+1,y)-f(x,y), \qquad h=1\end{aligned}$$

Therefore, the linear filter with mask $[-1,1]$ approximates the first derivative.

Normally use the mask $[-1,0,1]$ as its length is odd.

Edge detection using the gradient

Definition of the gradient

The gradient vector

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} M_x \\ M_y \end{pmatrix},$$

has an associated magnitude and direction

$$\operatorname{mag}(\nabla f) = \sqrt{M_x^2 + M_y^2}, \qquad \operatorname{dir}(\nabla f) = \tan^{-1}\left(\frac{M_y}{M_x}\right)$$

Properties of the gradient

- The magnitude of gradient indicates the strength of the edge.

- The gradient direction is perpendicular to the direction of the edge (the edge direction is rotated with respect to the gradient direction by -90 degrees).

Estimating the gradient with finite differences

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

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

The gradient can be approximated by finite differences:

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

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

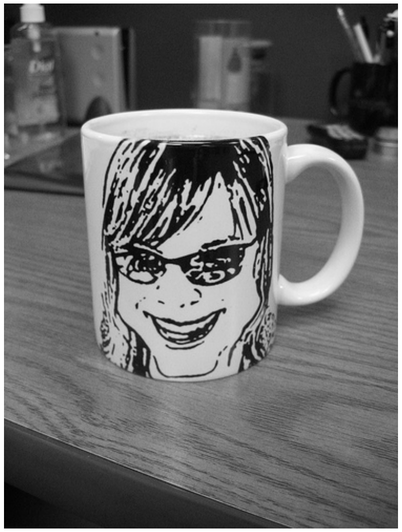
Linear filter masks used to calculate $\frac{\partial f}{\partial x}$

$$\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \qquad \text{Prewitt: } \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \text{Sobel: } \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

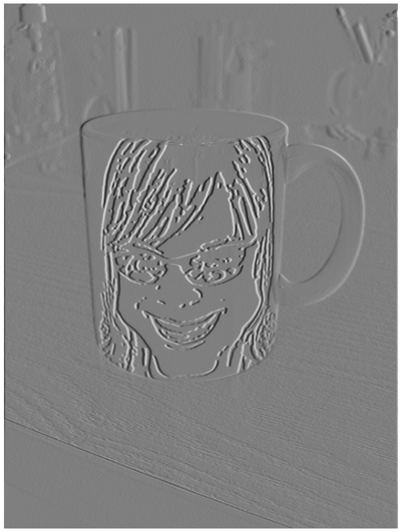
Linear filter masks used to approximate $\frac{\partial f}{\partial y}$

$$\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \qquad \text{Prewitt: } \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad \text{Sobel: } \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

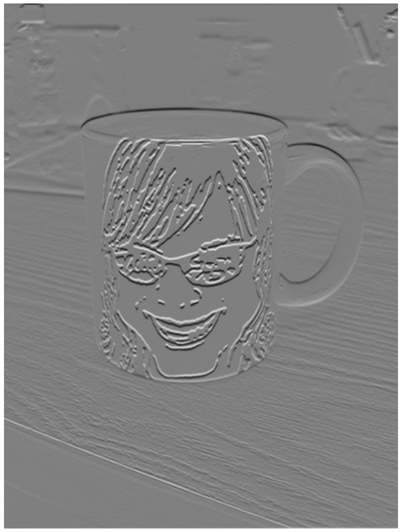
Example image gradients



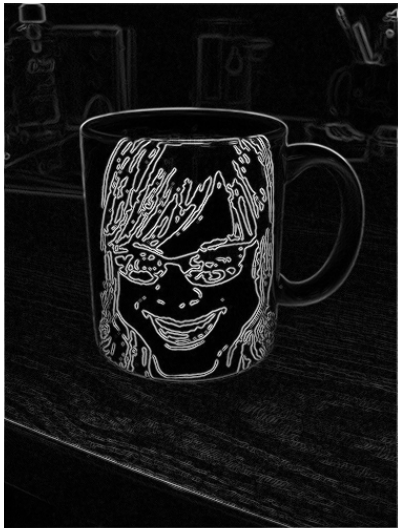
original image



$x-$**derivative**



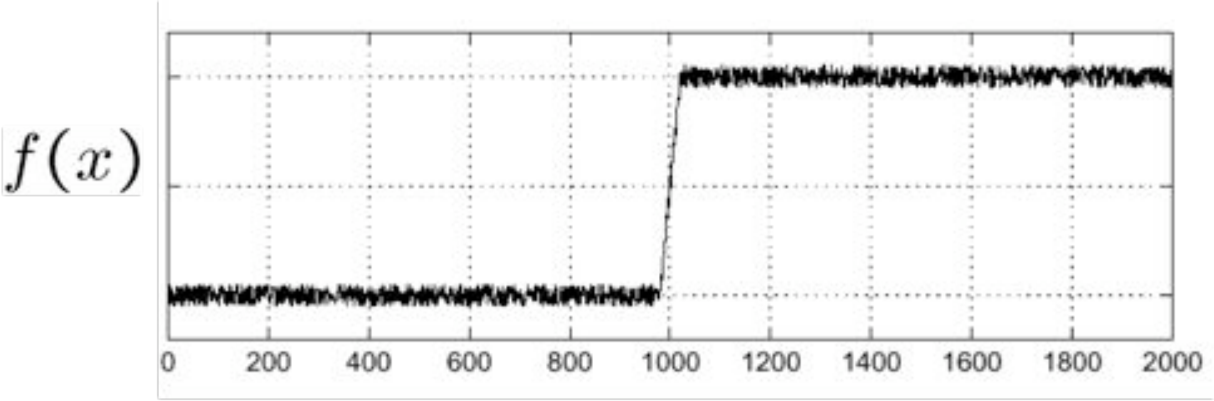
$y-$**derivative**

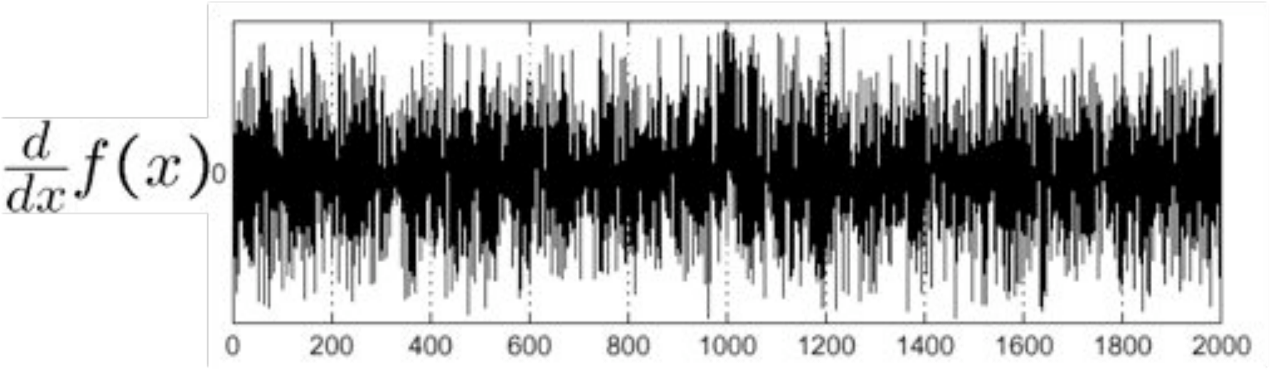


gradient magnitude

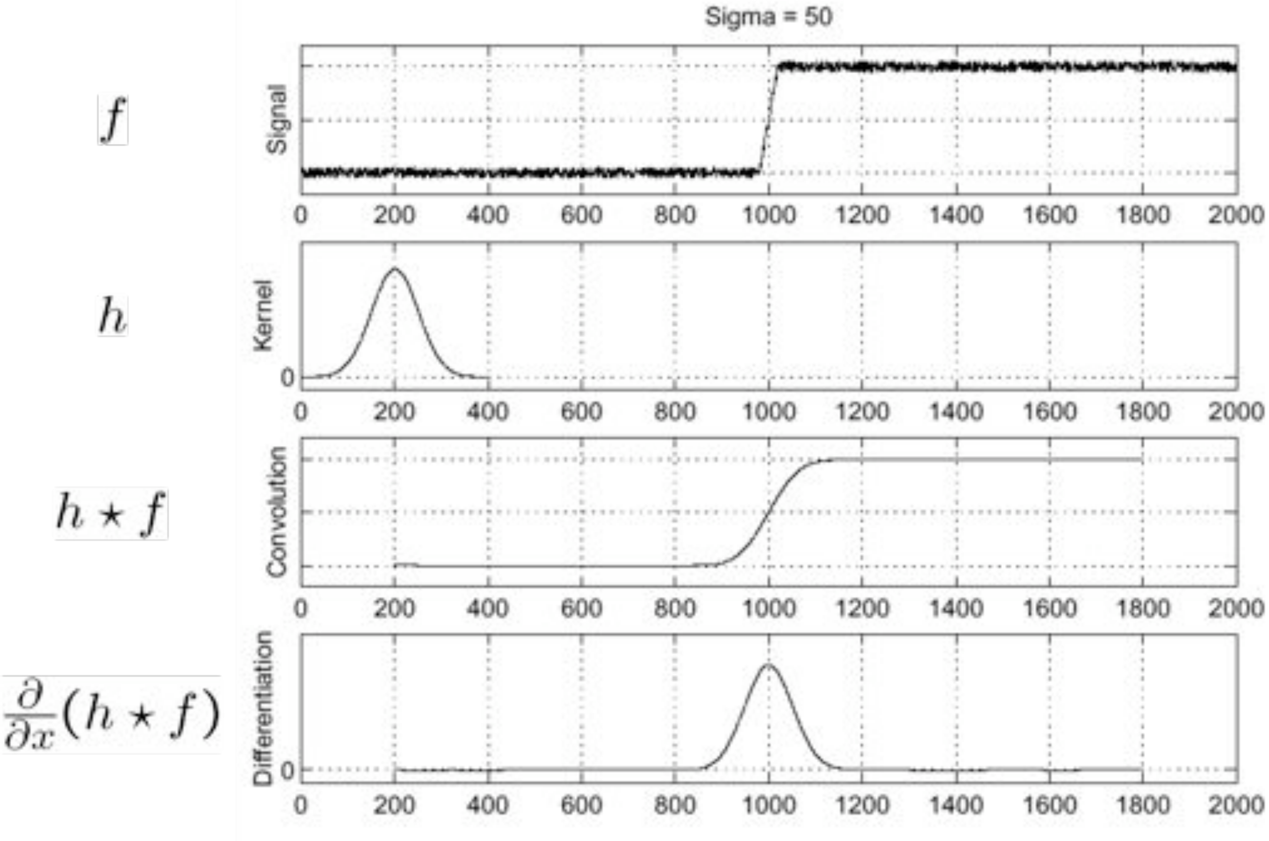
Practical issues

- Differential masks act as high-pass filters which tend to amplify noise.

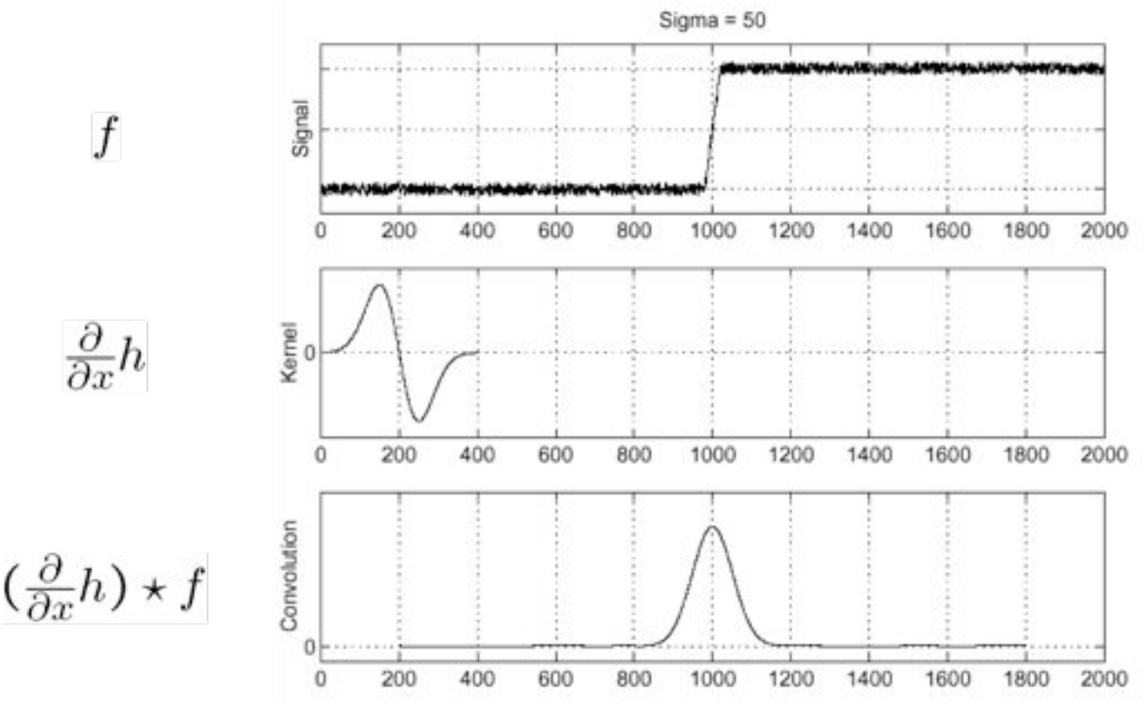




- To reduce the effects of noise, the image needs to be smoothed first with a lowpass filter.



Latter equivalent to

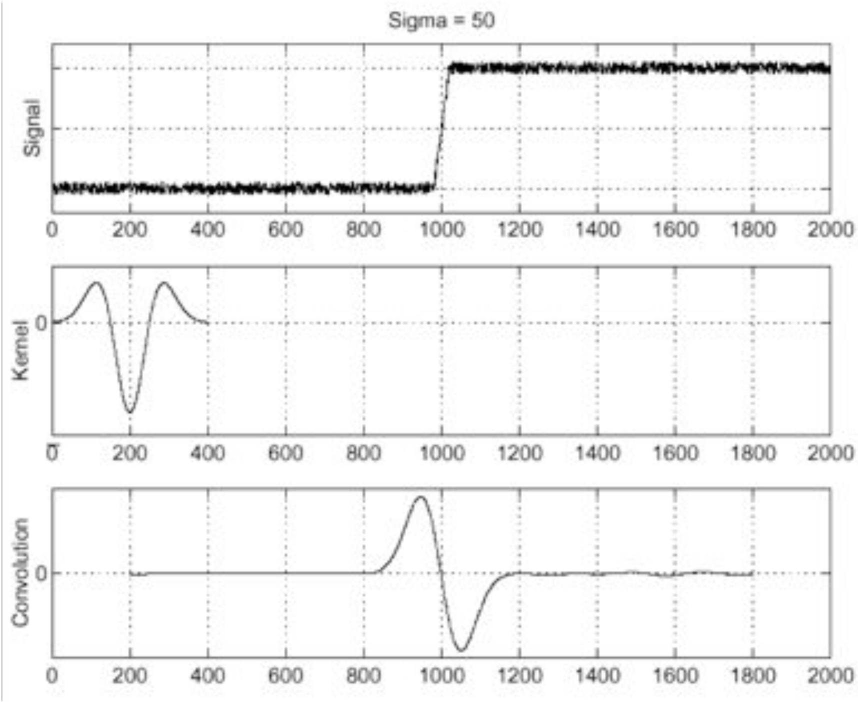


Because of the differentiation property of convolution: $\frac{\partial}{\partial x}(h \star f) = \frac{\partial h}{\partial x} \star f$

Can also detect edges using...

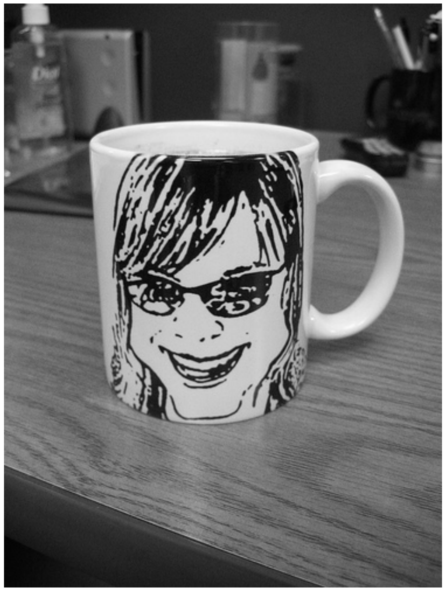
$$\frac{\partial^2}{\partial x^2}h$$

$$(\frac{\partial^2}{\partial x^2}h) \star f$$



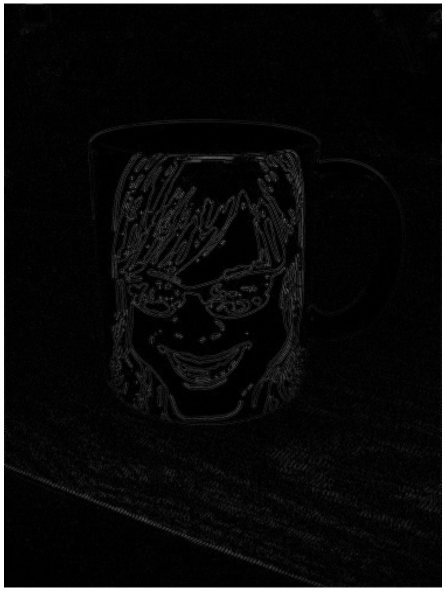
Edge is at the zero-crossing of the bottom graph.

Example image



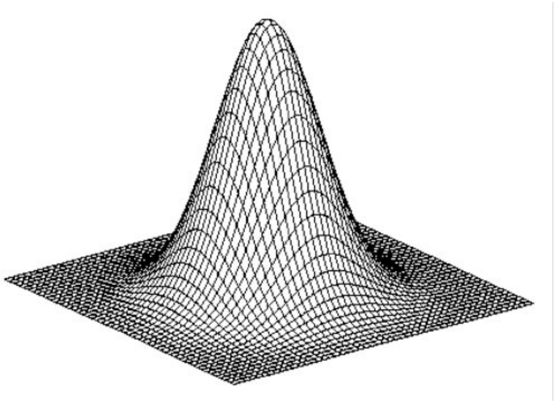
original image

The Laplacian



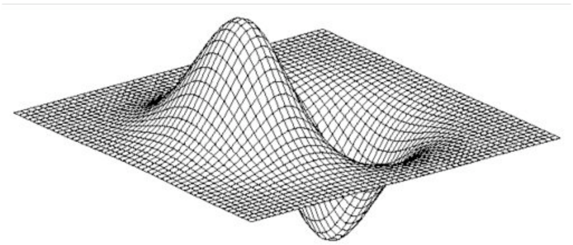
Absolute Laplacian

Summary: Edge detection in 2D



Gaussian

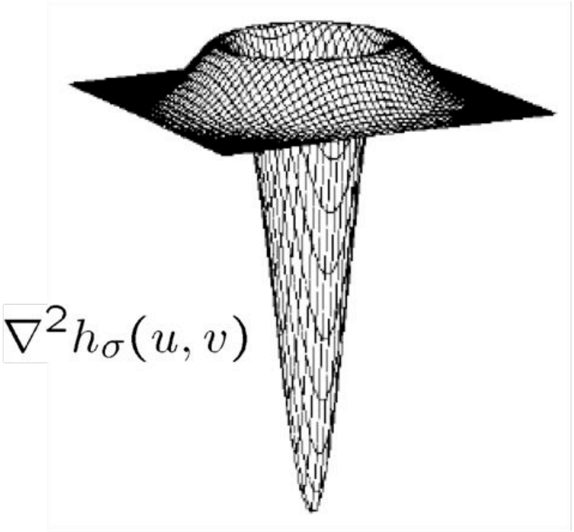
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



Derivative of Gaussian

$$\frac{\partial}{\partial u} h_\sigma(u,v), \; \frac{\partial}{\partial v} h_\sigma(u,v)$$

Laplacian of Gaussian



$$\nabla^2 h_\sigma = \frac{\partial^2 h_\sigma}{\partial u^2} + \frac{\partial^2 h_\sigma}{\partial v^2}$$

Back to Histograms

Histogram the output of filters

Any local descriptor (e.g. filter, filter combination) can be used to build a histogram.

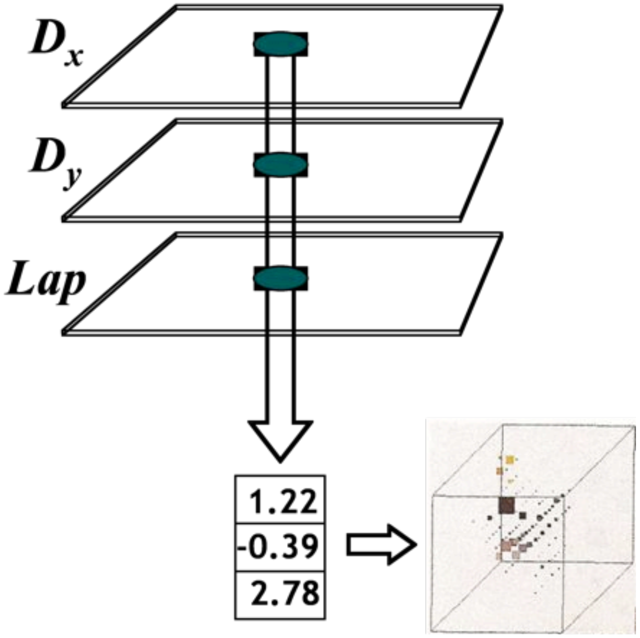
Examples:

- Gradient magnitude: $\sqrt{(f_x^2 + f_y^2)}$
- Gradient direction: $\tan^{-1}\left(\frac{f_y}{f_x}\right)$
- Laplacian: $f_{xx} + f_{yy}$

Multi-dimensional representations

Combine output of several filters:

- Each descriptor is applied to the whole image.
- Corresponding pixel values are combined into one feature vector.
- Feature vectors are collected in a multi-dimensional histogram.



Multi-dimensional representations

Useful simple combinations

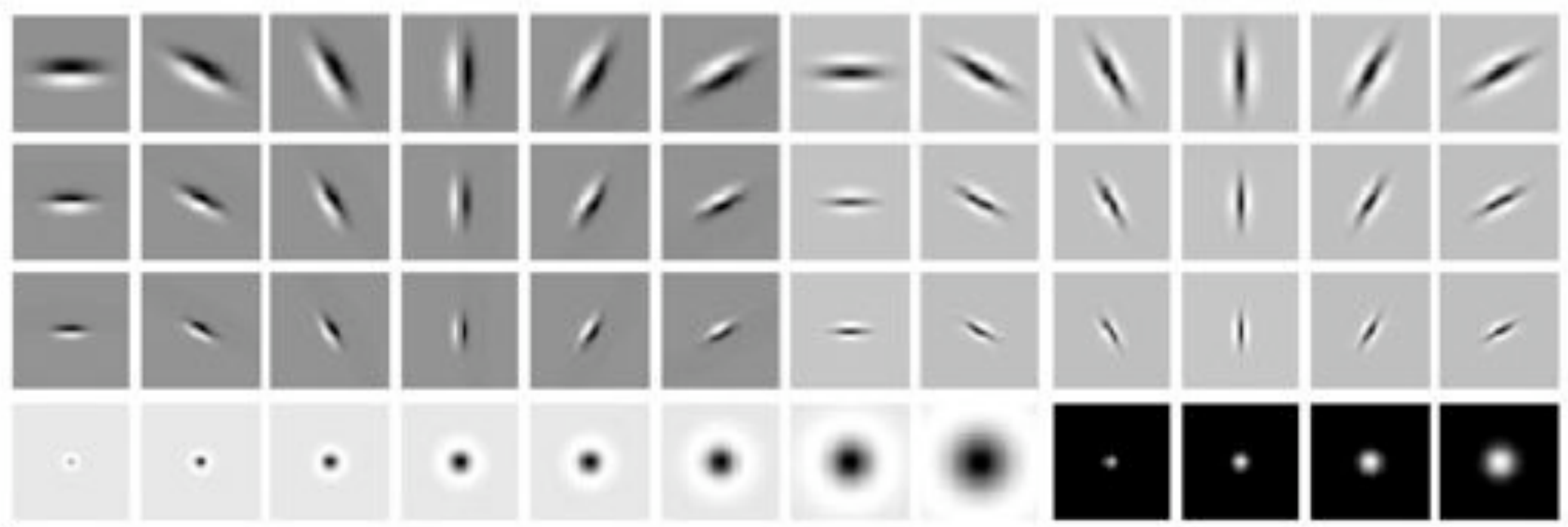
Rotation-variant $f_x - f_y$

- Descriptor changes when image is rotated.
- Useful for recognising oriented structure - vertical lines.

Rotation-invariant $|\nabla f| - \nabla^2 f$

- Descriptor does not change when image is rotated
- Can be used to recognise rotated objects.
- Less discriminative than rotation-variant descriptor.

Generalization: Filter-Banks

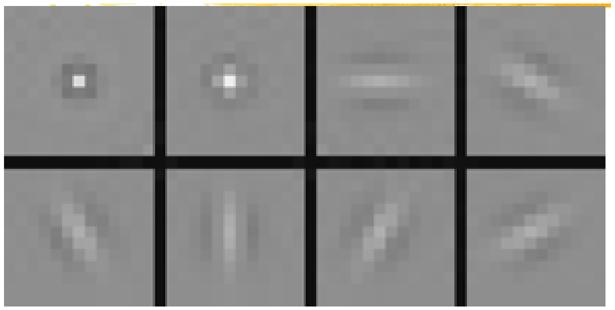


Which filters to put in the bank?

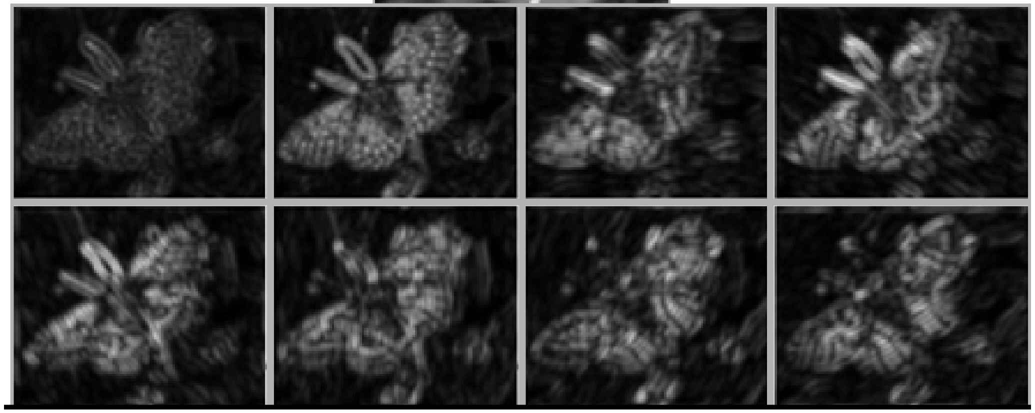
Typically want a combination of scales and orientations, different types of patterns.

Filter bank applied to image

Filter bank of 8 filters



Input Image

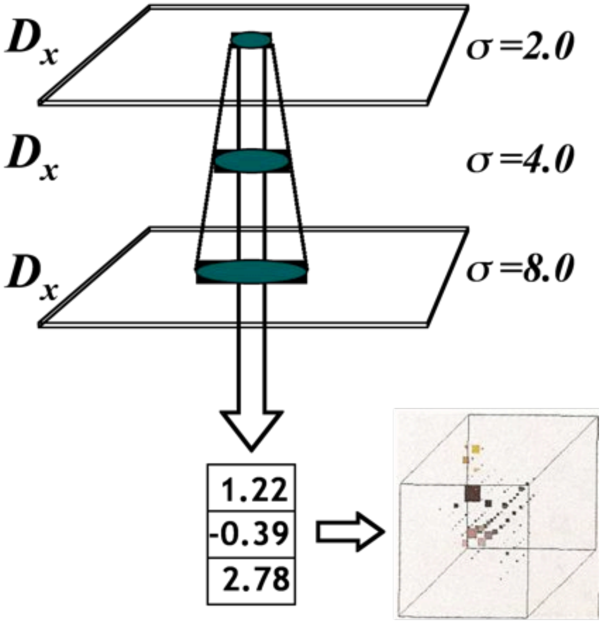


8 response images: magnitude of filtered outputs per filter

Multi-scale representations

Combination of filter responses at several scales:

- Descriptors are computed at different scales/sizes
- Each scale captures different information about the object.
- Size of the support region grows with increasing σ.
- Feature vectors capture both local details and larger-scale structures.



Multidimensional representations

Pros

- Work very well for recognition.
- Frequently simple combinations are sufficient.
- But multiple scales are very important!
- Generalization: filter banks

Cons

- High-dimensional histograms ⇒ lots of storage space.
- Global representation ⇒ not robust to occlusion.

Filter response histograms

Histograms do not describe spatial relationship.

However, do these type of multi-dimensional histograms of filter bank responses capture spatial relationships?

Filter response histograms

Histograms do not describe spatial relationship. However, these type of multi-dimensional histograms of filter bank responses do capture spatial relationships.

- **Answer**: Spatial relationships are captured implicitly via the filter responses.
- Support regions of neighboring descriptors overlap.
- Neighborhood relations are captured implicitly.

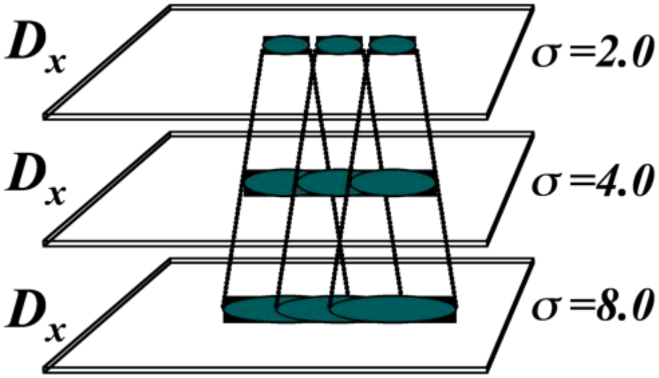
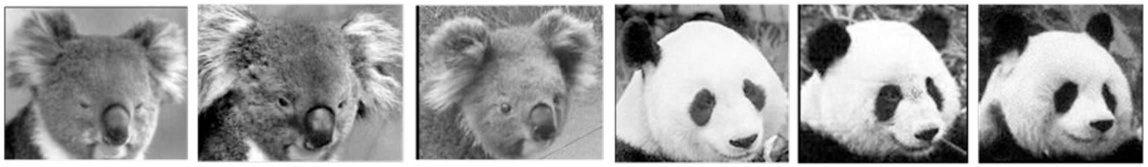


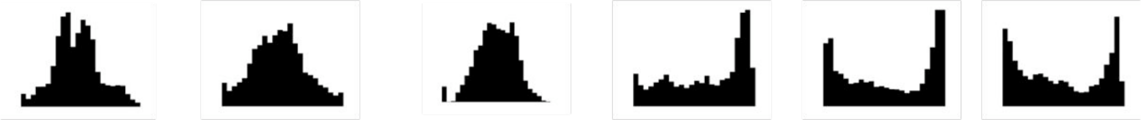
Image Patch Descriptors for the Naughties

Global image patch descriptors

Have seen descriptors

Template (Vector of pixel intensities) Fixed spatial grid





Grayscale/Colour Histogram Invariant to geometric transforms.

However would like a descriptor which has the advantages of both these descriptors

- invariance to small translational shifts and rotations
- Encode relative spatial relationship between different parts of the image

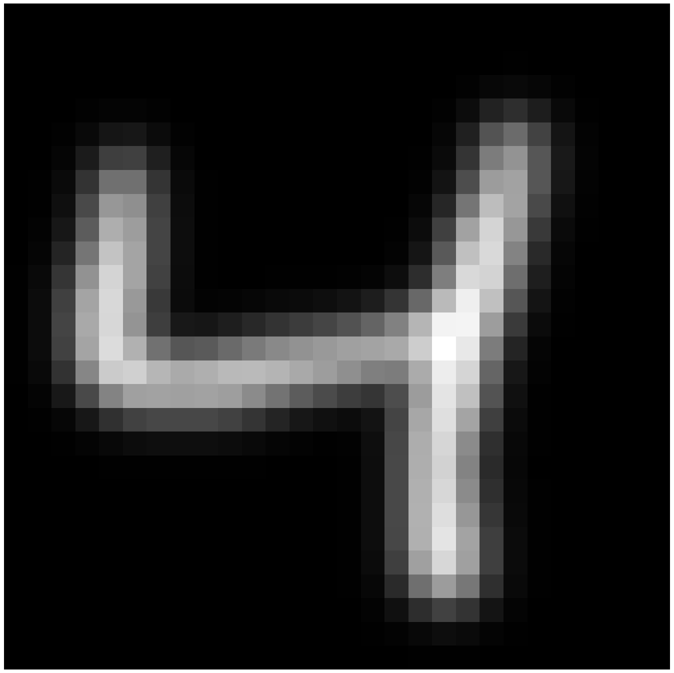
Two recent descriptors

Such image patch descriptors are:

- *Distinctive image features from scale-invariant keypoints*, by D.G. Lowe, International Journal of Computer Vision (2004)
  - In recent years this has been one of the most highly cited papers in the field of computer science.
- *Histograms of Oriented Gradients for Human Detection* by Navneet Dalal and Bill Triggs, Computer Vision and Pattern Recognition, 2005.
  - The descriptor used is the basis for the (one of the) best person detector in images in the research community.

There are great similarities between the two and both rely on the histogramming of image gradient orientations.

Histogram of gradient orientations



Image

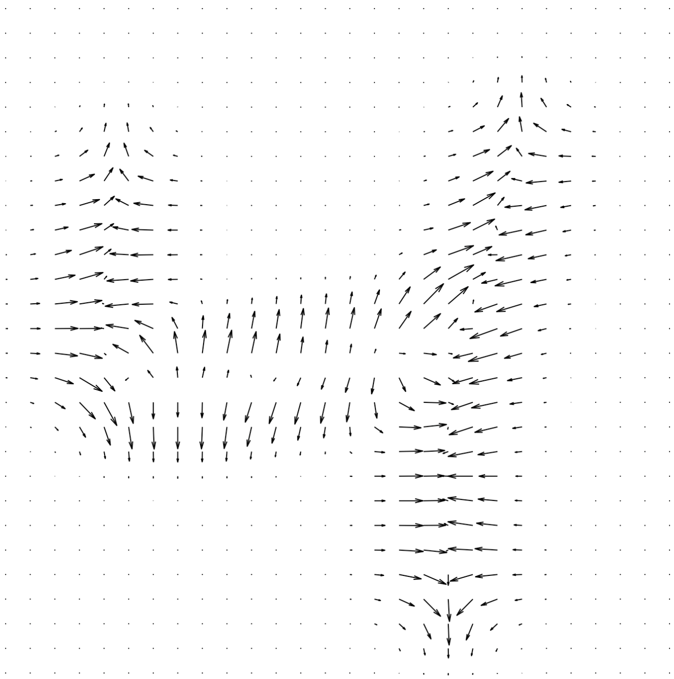
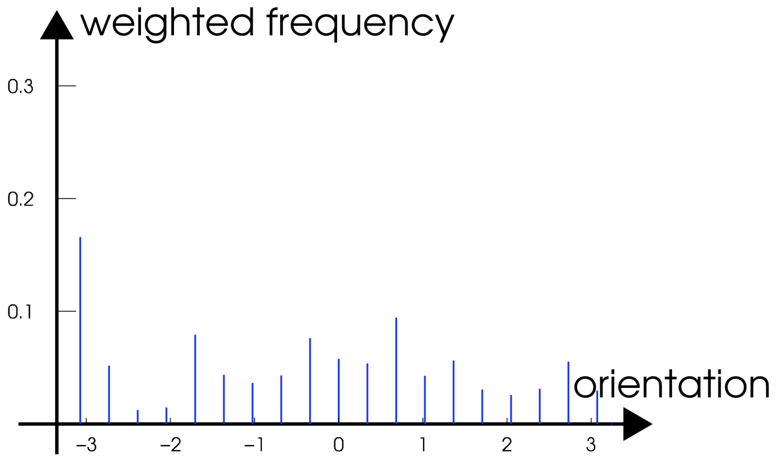


Image gradients

Histogram the gradient orientation (weighted according to their gradient magnitude) of the image gradients to get



This histogram can be interpreted as a one-dimensional vector $\mathbf{h}$ with n entries. Where each entry is the frequency of a bin centre.

What happens to $\mathbf{h}$ if

- the *four* translates slightly within image frame ?
- the *four* is very slightly rotated ?
- the *four* is rotated by 90° clockwise ?

Histogram of gradient orientations

Answers

- $\mathbf{h}$ is invariant to translation shifts as long as the same pixels are used for the gradient computations.
- there could be a large change in $\mathbf{h}$. Why ? (Aliasing)
- $\mathbf{h}$ will probably be very different. Each orientation will differ by 90° resulting in the histogramming of a very different set of numbers.

The latter ⇒ the descriptor is not rotationally invariant.

However, the middle condition is most worrisome. Ideally *small changes in the appearance* of a patch should result only in *small changes its feature description*. There is a solution to avoid this...

Avoiding aliasing

Each entry **voting only** for **its nearest orientation bin** results in possible **aliasing effects**. These can cause sudden changes in the computed feature vector.

To avoid this the histogram computation should involve distributing the weight of the orientation gradient magnitude for every pixel into the neighbouring orientation bins. This can be done using linear interpolation.

Let

- b be the inter-bin distance of our histogram $\mathbf{h}$
- $\mathbf{h}(x)$ the value of the histogram for the bin centred at x.

Assume that we want to interpolate a weight w at point x into the histogram. Let x_1 and x_2 be the two nearest neighbouring bins of the point x such that

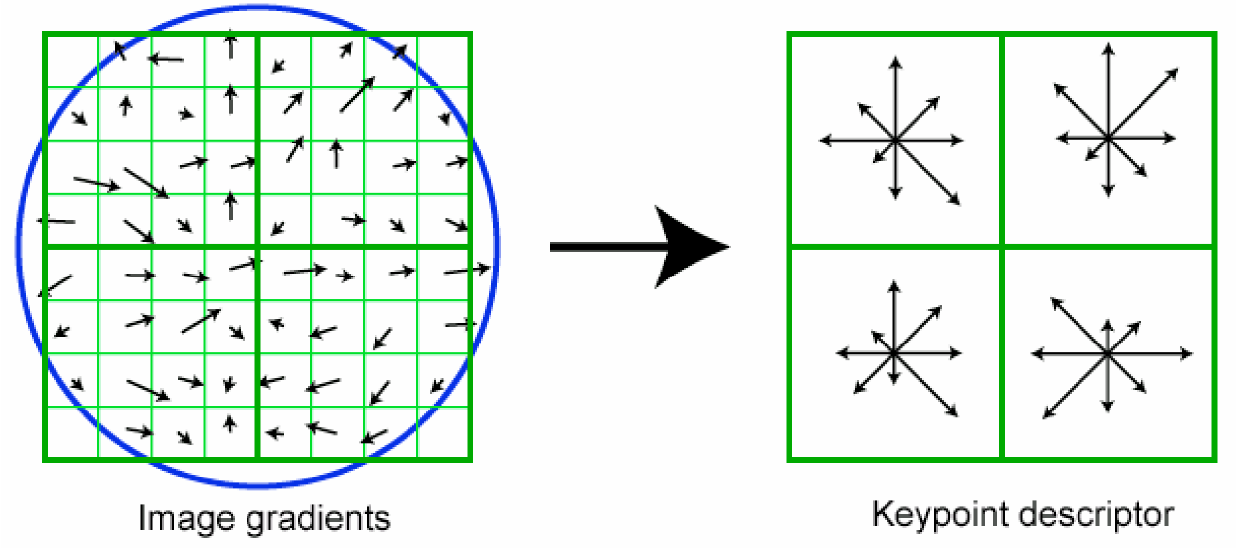
$x_1 \leq x < x_2$. Linear interpolation distributes the weight w into two nearest neighbours as follows:

$$\mathbf{h}(x_1) \leftarrow \mathbf{h}(x_1) + w \left(1 - \frac{x - x_1}{b}\right)$$
$$\mathbf{h}(x_2) \leftarrow \mathbf{h}(x_2) + w \left(\frac{x - x_1}{b}\right)$$

Note also, when histogramming orientations/angles, the bins have to be wrapped around as $360^\circ = 0^\circ$.

SIFT patch descriptor

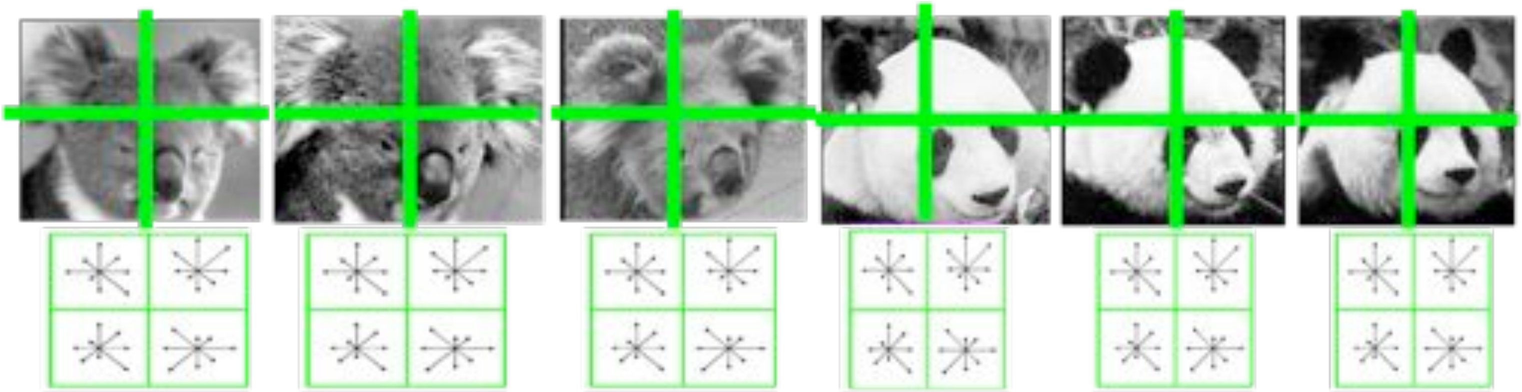
- Compute and threshold image gradients.
- Create array of orientation histograms
- 8 orientations $\times 4 \times 4$ histogram array $=$ 128 dimensions



Each histogram in the array is represented by a vector $\mathbf{h}_k$ which has 8 entries. Then the final feature vector is concatenation of all these vectors:

$$\mathbf{f}_S = (\mathbf{h}_1^T, \mathbf{h}_2^T, \ldots, \mathbf{h}_{16}^T)^T$$

$\mathbf{f}_S$ is then normalized to have unit length. This enhances its invariance to changes in illumination conditions.



Avoid spatial aliasing

Note this descriptor is not invariant to translation of the underlying image to the grid and also that **spatial aliasing** may occur.

For example, if a strong edge pixel is at the boundary of a cell in one image and due to slight change in imaging conditions it falls into the neighbouring cell in the next, the naive voting scheme will assign the pixel's weight to different histograms bins in the two cases and produce a very different feature vector.

To avoid this, use 3-D linear interpolation of the pixel weight into the spatial-orientation histogram. This is know as trilinear interpolation

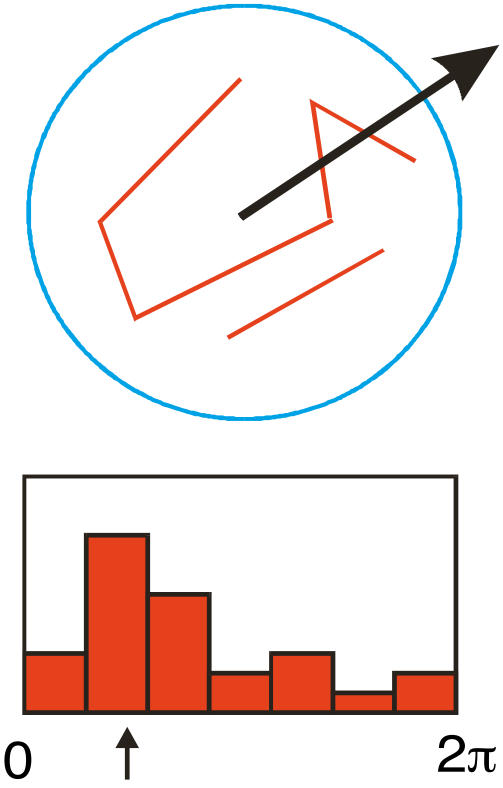
Reference for the keen

Let w at the 3-D point $\mathbf{x} = (x, y, z)$ be the weight to be interpolated. Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be the two corner vectors of the histogram cube containing $\mathbf{x}$, where in each component $\mathbf{x}_1 \leq \mathbf{x} < \mathbf{x}_2$. Assume that the bandwidth of the histogram along the x, y and z axis is given by $\mathbf{b} = (b_x, b_y, b_z)$. **Trilinear interpolation** distributes the weight w to the 8 surrounding bin centres as follows

$$
\begin{aligned}
\mathbf{h}(x_1, y_1, z_1) &\leftarrow \mathbf{h}(x_1, y_1, z_1) + w \left(1 - \tfrac{x - x_1}{b_x}\right) \left(1 - \tfrac{y - y_1}{b_y}\right) \left(1 - \tfrac{z - z_1}{b_z}\right) \\
\mathbf{h}(x_1, y_1, z_2) &\leftarrow \mathbf{h}(x_1, y_1, z_2) + w \left(1 - \tfrac{x - x_1}{b_x}\right) \left(1 - \tfrac{y - y_1}{b_y}\right) \left(\tfrac{z - z_1}{b_z}\right) \\
\mathbf{h}(x_1, y_2, z_1) &\leftarrow \mathbf{h}(x_1, y_2, z_1) + w \left(1 - \tfrac{x - x_1}{b_x}\right) \left(\tfrac{y - y_1}{b_y}\right) \left(1 - \tfrac{z - z_1}{b_z}\right) \\
\mathbf{h}(x_2, y_1, z_1) &\leftarrow \mathbf{h}(x_2, y_1, z_1) + w \left(\tfrac{x - x_1}{b_x}\right) \left(1 - \tfrac{y - y_1}{b_y}\right) \left(1 - \tfrac{z - z_1}{b_z}\right) \\
\mathbf{h}(x_1, y_2, z_2) &\leftarrow \mathbf{h}(x_1, y_2, z_2) + w \left(1 - \tfrac{x - x_1}{b_x}\right) \left(\tfrac{y - y_1}{b_y}\right) \left(\tfrac{z - z_1}{b_z}\right) \\
\mathbf{h}(x_2, y_1, z_2) &\leftarrow \mathbf{h}(x_2, y_1, z_2) + w \left(\tfrac{x - x_1}{b_x}\right) \left(1 - \tfrac{y - y_1}{b_y}\right) \left(\tfrac{z - z_1}{b_z}\right) \\
\mathbf{h}(x_2, y_2, z_1) &\leftarrow \mathbf{h}(x_2, y_2, z_1) + w \left(\tfrac{x - x_1}{b_x}\right) \left(\tfrac{y - y_1}{b_y}\right) \left(1 - \tfrac{z - z_1}{b_z}\right) \\
\mathbf{h}(x_2, y_2, z_2) &\leftarrow \mathbf{h}(x_2, y_2, z_2) + w \left(\tfrac{x - x_1}{b_x}\right) \left(\tfrac{y - y_1}{b_y}\right) \left(\tfrac{z - z_1}{b_z}\right)
\end{aligned}
$$

Invariance to rotation

- Create histogram of local gradient directions
- Assign canonical orientation at peak of smoothed histogram
- Rotate patch so that dominant direction is vertical.



SIFT in the real world

Sony Aibo (Evolution Robotics)

SIFT usage:

- Recognize charging station
- Communicate with visual cards

AIBO® Entertainment Robot

Official U.S. Resources and Online Destinations

ERS-7

Entertainment Robot AIBO

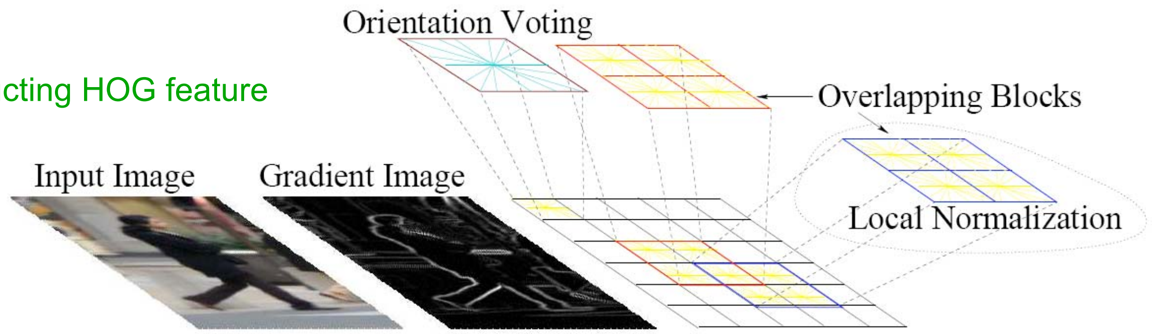
ERS-7 with:
Wireless LAN
AIBO MIND software
Energy Station
AIBOne
Pink Ball
AIBO Cards (15)
WLAN Manager CD
Battery & AC Adapter

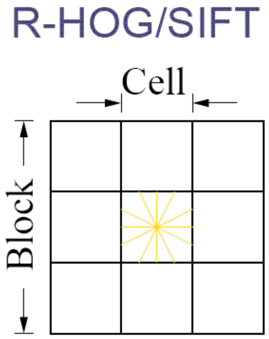
3rd Generation

Pre-order Now!

Histogram of Oriented Gradients

Constructing HOG feature

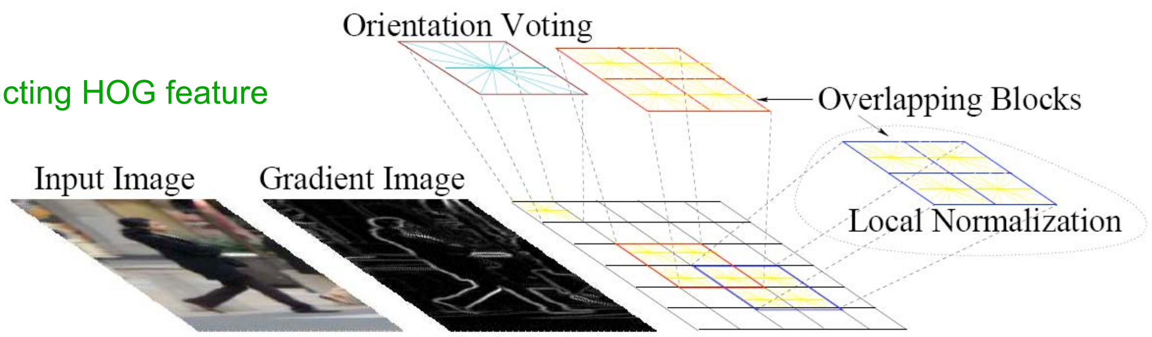




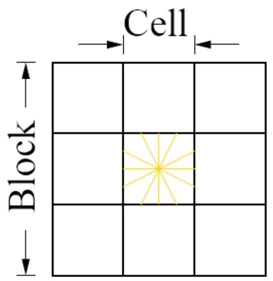
Structure:

- Have a 2D grid of cells. Each cell is of size $\eta \times \eta$ pixels.
- Have m blocks where each block is a grid of $\zeta \times \zeta$ cells.
- Different blocks may contain some of the same cells. That is there will be overlaps.

Constructing HOG feature



R-HOG/SIFT



The descriptor

- Weighted votes for gradient orientation are accumulated over the spatial cells to obtain a histogram for each cell.
- For each block its cell histograms are concatenated, as in SIFT, and this **HOG** feature is normalized to normalize contrast within the block.
- Concatenate all the **HOG** features for each block into one long vector.

Experimental findings from HOG paper

From experiments run on images of people in natural environments

- Best recognition results when the mask $[-1, 0, 1]$ was used to compute the image gradients and no pre-smoothing of the images was applied.

- Best recognition results when gradient orientation has at least 18 (9) bins if the orientation is defined between $0 - 360°(180°)$.

- Having many overlapping blocks and the contrast normalization being independent for each one is critical for good performance.

- One good normalization scheme was

$$\mathbf{v} \leftarrow \sqrt{\frac{\mathbf{v}}{\|\mathbf{v}\|_1 + \epsilon}}, \quad \epsilon > 0 \text{ and small}$$

The search problem

Story so far

- Have introduced some methods to describe the appearance of an image patch via a feature vector. (SIFT, HOG etc..)

- For patches of similar appearance their computed feature vectors should be similar while dissimilar if the patches differ in appearance.

- Feature vectors are designed to be invariant to common transformations that superficially change the pixel appearance of the patch.

Next problem

We have an reference image patch which is described by a feature vector $\mathbf{f}_r$.

$\Rightarrow \mathbf{f}_r$

Given a **novel image** identify the **patches** in this image that correspond to the **reference patch**.

One part of the problem we have **explored**.

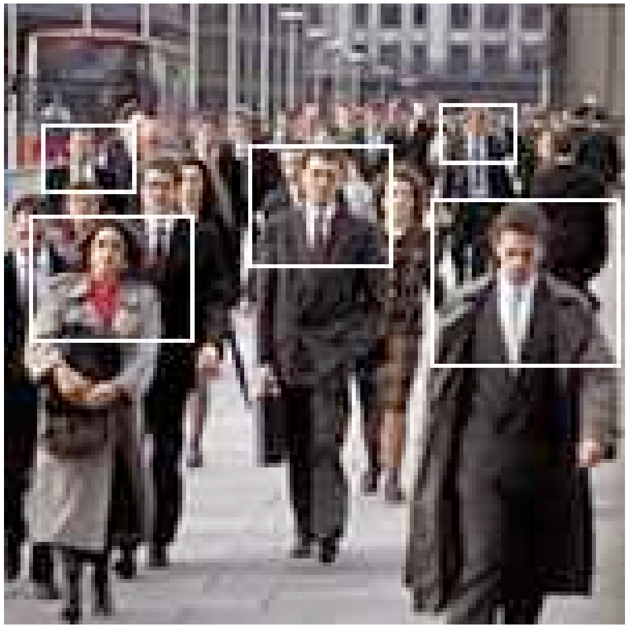
A patch from the novel image generates a feature vector $\mathbf{f}_n$. If $\|\mathbf{f}_r - \mathbf{f}_n\|$ is small then this patch can be considered an instance of the texture pattern represented by the reference patch.

However, which and how many different image patches do we extract from the novel image ?

Remember..

The sought after image patch can appear at:

- any spatial location in the image
- any size, (the size of an imaged object depends on its from the camera)
- multiple locations



Sliding window technique

Therefore we must examine patches centered at many different pixel locations and at many different sizes.

Naive Option: Exhaustive search using original image

```
for j = 1:n_s
   n = n_min + j*n_step
   for x=0:x_max
     for y=0:y_max
       Extract image patch centred on pixel x, y of size n×n.
       Rescale it to the size of the reference patch
       Compute feature vector f.
```

This is computationally intensive especially if it is expensive to compute $\mathbf{f}$ as it could be calculated upto `n_s` × `x_max` × `y_max`.

Also frequently if `n` is large then it is very costly to compute $\mathbf{f}$.

Next Features lecture will review how to do this efficiently....

Today’s programming assignment

Programming assignment

- Details available on the course website.

- You will write `Matlab` functions to extract different feature descriptors from an image. You will then compare these features when extracted from images of eyes and noses.

- **Important**: Due to the short turn around time until the next lecture. This assignment is not due until the lecture on Tuesday 27th of March.

- Mail me about any errors you spot in the `Exercise` notes.

- I will notify the class about errors spotted and corrections via the course website and mailing list.