Filtering, scale, orientation, localization, and texture

Nuno Vasconcelos

ECE Department, UCSD

(with thanks to David Forsyth)
Beyond edges

- we have talked a lot about edges
- while they are important, it is now recognized that they are not enough
- for many objects they are not even defined (think of a landscape with green grass, a mountain, and sky)
- fortunately there is a lot more we can do with filtering than look for edges
- today we will talk about such extensions
- let’s start by recalling that filters are templates
Filters as templates

- applying a filter at some point can be seen as taking a dot-product between the image and some vector
- filtering the image is a set of dot products
- insight
  - filters look like the effects they are intended to find
  - filters find effects they look like
- Q: what are good filters (detectors) for vision?
Filters for vision

- the answer is that it depends on what you are looking for
- in the absence of good answers we can look for them in biological vision
- it has been observed that cells in early stages of the brain tend to respond to either
  - spots (e.g. a white spot surrounded by black)
  - bars (e.g. half the cell exposed to white, the other to black)
- this has drawn a lot of attention to spots and bars, which book emphasizes, but they do not have to be the ones
- e.g. if you are working with man-made scenes, e.g. buildings, it is probably a must to detect corners
- but they do illustrate the concept of filters as detectors
spot filter

Positive responses

bar filter
Scale and orientation

- non-controversial: apply a set of filters that cover a broad range of scales and orientations

- a large filter will detect coarse object properties, e.g. a building is a box

- a small filter will detect details, e.g. this box contains many little boxes (windows)

- the same with orientation: buildings are vertical boxes, cars are horizontal

- this is a general principle
How to filter at multiple scales

- if two filters have the same shape, the one that has more pixels will detect features of larger scale
- but filtering images with very large filters is expensive
- alternative:
  - keep the filter constant
  - apply it to down-sampled replicas of the image
- the collection of down-sampled images is called a pyramid
- we apply the filter to all levels
Example

- at high-resolution the filter is quite small
- detects the contours of the antennae, i.e. edges
- at low-resolution it covers a lot more ground
- detects the antennae as “parts”
- note that the stuff which is not oriented like the filter is ignored
Orientations at small scale
Orientations at large scale

- note that there is **significant leakage** between orientations (and scales)
- the detectors are not perfect, but provide a **workable** decomposition
- given this we could say: butterflies have antennae, look for strong response by 4&5, small for the others
- this is an image **classifier**
Pyramids

- how do we go about creating a pyramid?
- we want to **downsample by two** (each direction) at each level
- to avoid aliasing we have to **low-pass filter** with cut-off \((\pi/2, \pi/2)\).
In the frequency domain

- recall: downsampling expands the spectrum

- each stage:

  ![Diagram showing the downsampling process with filters at each stage.]

- at full resolution this is equivalent to a sequence of filters

\[
X_i \times H = X_{i+1}
\]
The Gaussian pyramid

- the low pass filter is a Gaussian
- inspired by human vision
  - we have seen that Gaussian type of receptive fields appear in various parts of the brain
- computationally consistent
  - \( X_1 \sim N(\mu_1, \sigma_1), X_2 \sim N(\mu_2, \sigma_2) \), independent
  - \( Z = X_1 + X_2 \), \( Z \) is \( N(\mu_1 + \mu_2, \sigma_1 + \sigma_2) \)
  - but \( p_Z(z) = p_{X_1}(z) * p_{X_2}(z) \)
- conclusion: \( \text{gaussian}(0, \sigma) * \text{gaussian}(0, \sigma) = \text{gaussian}(0, 2\sigma) \)
- this is exactly what we need: the \( n^{th} \) convolution is low pass filtering with bandwidth \( \pi/2^n \)
same bar: in the big images is a hair on the zebra’s nose; in smaller images, a stripe; in the smallest, the animal’s nose
Wavelets

- so far, low frequencies are replicated at all levels
- redundancy problematic for some applications
- wavelets: each filter is bandpass
- this is called a critically sampled pyramid
- no redundancy
- for vision redundancy is sometimes good others not

at full resolution equivalent to a sequence of filters
The Laplacian Pyramid

- it is the poor’s man version of a wavelet
- to obtain band-pass filtering
  - compute a Gaussian pyramid
  - upsample each level and subtract from its predecessor

- the same as filtering with a DoG filter, i.e. bandpass
- lowest resolution is low-pass, other layers have incremental detail
Oriented pyramids

- pyramids seen so far do not produce orientation info
- for this need to filter each pyramid level with oriented kernels
- this is an oriented pyramid
Q: how do we design a filter centered at a certain frequency and with a certain orientation?
Gabor filters

• come in pairs:
  • one recovers symmetric components in a direction,
  • the other recovers antisymmetric components

definition:

\[ G_{\text{sym}}(x, y) = \cos(k_x x + k_y y) \exp\left\{ -\frac{x^2}{2\sigma_x^2} - \frac{y^2}{2\sigma_y^2} \right\} \]
\[ G_{\text{sym}}(x, y) = \sin(k_x x + k_y y) \exp\left\{ -\frac{x^2}{2\sigma_x^2} - \frac{y^2}{2\sigma_y^2} \right\} \]

parameters: \((k_x, k_y)\) location, \((\sigma_x, \sigma_y)\) scale
In frequency

- this is just amplitude modulation of a cosine by a Gaussian

\[
G(x) = \cos(k_x x) \exp\left\{-\frac{x^2}{2\sigma_x^2}\right\}
\]

so

\[
G(\omega) = \exp\left\{-\frac{\sigma_x^2 (\omega - k_x)^2}{2}\right\} + \exp\left\{-\frac{\sigma_x^2 (\omega + k_x)^2}{2}\right\}
\]

- allows coverage of the frequency spectrum with a set of filters (shown here only \(\omega_2 > 0\))
Localization

- Gabor localized in space, frequency, and orientation
- decomposition into many frequency “channels”
- contrast with Fourier: non-localized basis functions $e^{j\omega x}$
- localization is important for detailed understanding, e.g. correlations
Localized representations

- note that Gabor is \( G(x, y) = \cos(k_x x + k_y y)w(x, y) \)
- where window \( w(x,y) \) is a Gaussian
- this is what localizes representation in space, cosine (or sine, or \( e^{j\omega x} \)) is already localized in frequency (\( \delta(\omega x) \))
- in fact the localization in frequency gets worse, we go from a Dirac delta to a Gaussian
- once again this is just the uncertainty principle:
  - Fourier: point support in frequency, infinite support in space
  - Gabor: finite (well, close to) support in space, finite support in frequency
- is the Gaussian the only possible window?
Other localized representations

- no. Any low-pass filter will do.
- various wavelets correspond to other choices of window
- note also that if \( w(x,y) \) is the box filter we get
  \[
  G(x, y) = \cos(k_x x + k_y y) R_{N_1 \times N_2} (x, y)
  \]
- convolving with these filters is the same as computing the DCT of image blocks
- antisymmetric part corresponds the discrete sine transf.
  \[
  G(x, y) = \sin(k_x x + k_y y) R_{N_1 \times N_2} (x, y)
  \]
- and is we combine both we get the short-time Fourier transform
  \[
  G(x, y) = e^{j(k_x x + k_y y)} R_{N_1 \times N_2} (x, y)
  \]
Short-time DCT

- the filters are these
- appealing because
  - real for real images
  - fast, lots of hardware available
- but also because filters detect various attributes that appear relevant
  - vertical/horizontal edges
  - vertical/horizontal bars
  - corners, t-junctions, spot, checkerboards, various flows
- it is also a basis: any function can be reconstructed
- Gabor does not assure that
Texture

we have learned a lot about the importance of localization, scale, and orientation in vision

but what is this good for in practice?

one of the major applications is texture analysis/synthesis

Texture is important for

• recognition (why is it so easy to recognize a zebra, why is the cheetah not a cat?)
• segmentation (what are the boundaries between water and grass?)
• graphics: to synthesize a tiger I need samples of its fur
• etc.
Representing textures

but what is a texture?

I have not heard a good definition yet

it is one of those things that everyone can recognize, but few can describe, e.g. “like that stuff that X is made of”.

book: “textures are made up of quite stylised sub-elements, repeated in meaningful ways”

this is sensible (most of the time) and a workable definition

anyway, interesting that definition is not easy yet texture gives so much info:

• e.g. on the next slide it is not clear what “sub-elements” means
• yet we get plenty of information on geometry, geography, atmospheric conditions, etc.
Representing textures

- possible representation:
  - find the sub-elements, and represent their statistics

- but what are the sub-elements, and how do we find them?
  - by applying filters, looking at the magnitude of the response

- what statistics?
  - within reason, the more the merrier.
  - at least, mean and standard deviation
  - better, various probability estimates
Segmentation

- what are the **various components of the scene**?
- **simple example** from the book: two derivatives + square + average over a local window + thresholding
- illustrates segmentation into horizontal/vertical components
Recognition

- what texture is like this?
- **example:** Gabor decomposition + compute mean and std of each channel + stack in a vector
- each texture in database summarized by one vector $t_i$
- **recognition:** find vector $t_i$ closest to query $q$

$$\min_i \| q - t_i \|$$
Any questions?