TSBB15
Computer Vision

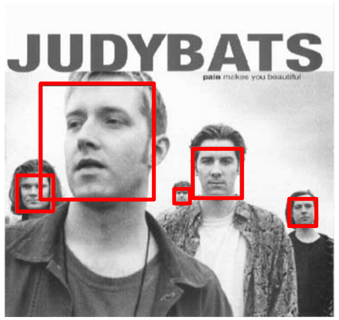
Lecture 2
Image Representations

Today's topics

- Scale spaces
- Pyramids
- Hierarchical representations
- Representing uncertainty/ambiguity
  - case study: local orientation representation

Scale spaces: motivation 1

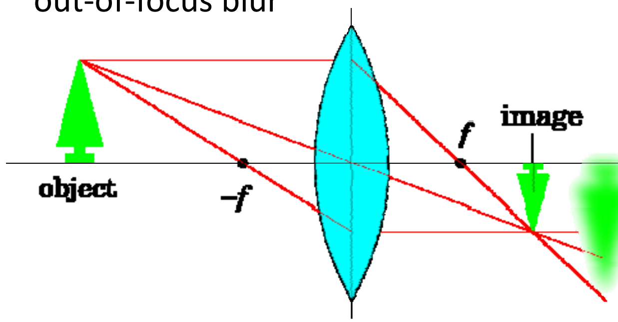
- Objects at different distances have different sizes in the image plane
- We want to detect them all



Example: face detection

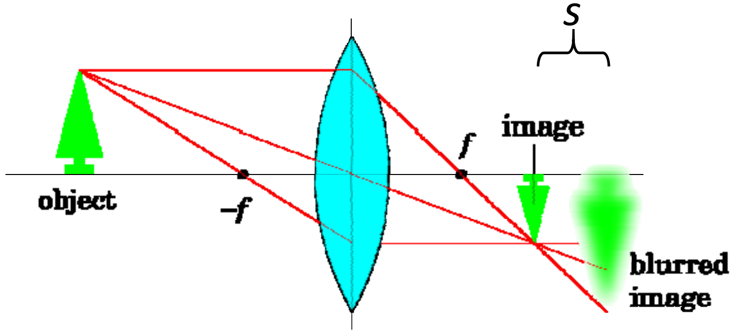
Scale spaces: motivation 2

- Cameras have limited depth-of-field
- We want our algorithms to robustly deal with out-of-focus blur



Scale spaces: motivation 2

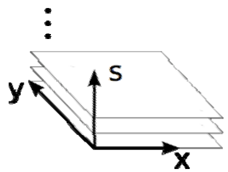
- Image blur function: image(s)



Image(s)

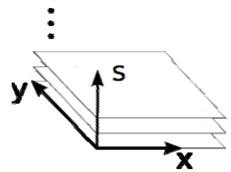
Representation: Scale Space

- Basic idea
  - Stack images in a 3D space
  - The third axis, *s*, is called scale



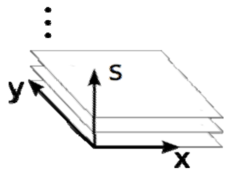
Representation: Scale Space

- Basic idea
  - Stack images in a 3D space
  - The third axis, *s*, is called *scale*
  - *s* = 0 corresponds to the original image
  - As *s* grows, the image becomes more blurred
- Intuitively: *s* is a “defocus” or “blur” parameter



Scale Space

- Notation:
  - original image $f_0(x, y)$
  - blurred image $f_s(x, y)$



- $f_s = T_s\{f_0\}$
- T_s : transformation that produces f_s from f_0

Scale Space Axioms

[Iijima, 1959] specifies properties of T_s:
1. Linear
2. Shift-invariant
3. Semi-group property
4. Scale- and rotation-invariant
5. Maintain positivity

6. Separability (by later authors)

Gaussian Scale Space

- Semi-group:

$$T_{s_1+s_2}\{f\} = T_{s_1}\{T_{s_2}\{f\}\}$$

- Rotation invariance:

$$T_s\{R\{f\}\} = R\{T_s\{f\}\} \quad \text{where} \quad R \text{ is a rotation}$$

- Scale invariance

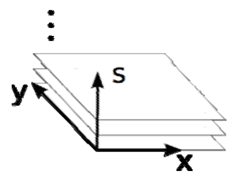
$$T_s\{f_0(ax, ay)\} = (T_{a^2 s}\{f_0\})(ax, ay)$$

- Maintains positivity

$$f(x, y) \geq 0 \quad \Rightarrow \quad T_s\{f\}(x, y) \geq 0$$

Scale Space

- A1+A2: T_s is a convolution
  - original image $f_0(x, y)$
  - blur kernel $g_s(x, y)$
  - The scale space of f_0 is given as the convolution:



$$f_s(x, y) = (g_s * f_0)(x, y)$$

In the Fourier domain: $F_s = G_s \cdot F_0$

Gaussian Scale Space

- The remaining axioms lead to a unique formulation of G_s:

$$G_s(\omega_x, \omega_y) = e^{-s\frac{\omega_x^2+\omega_x^2}{2}} \qquad g_s(x,y) = \frac{1}{2\pi s}\, e^{-\frac{x^2+y^2}{2s}}$$

- Separability:

$$g_s(x,y) = \frac{1}{\sqrt{2\pi s}}\, e^{-\frac{x^2}{2s}} \cdot \frac{1}{\sqrt{2\pi s}}\, e^{-\frac{y^2}{2s}}$$

PDE formulation

- The Gaussian scale space can also be derived as the solution to the PDE:

$$\frac{\partial}{\partial s} f_s(x,y) = \frac{1}{2} \nabla^2 f_s(x,y)$$

$$\text{boundary condition: } f_0(x,y) = f(x,y)$$

- A.k.a. the **diffusion equation**
  - Compare to the *heat equation*, where $f_s(x,y)$ is the temperature at time s in point (x,y), given initial temperature $f_0(x,y)$

PDE formulation

$$\frac{\partial}{\partial s} f_s(x,y) = \frac{1}{2} \left(\frac{\partial^2 f_s}{\partial x^2} + \frac{\partial^2 f_s}{\partial y^2} \right) (x,y)$$

- The change in $f_s(x,y)$ when we move only along the scale parameter s equals a local second order derivative of f_s at (x,y)
- (We will return to the PDF formulation of scale spaces in a later lecture)

Implementation of the Gaussian Scale-Space

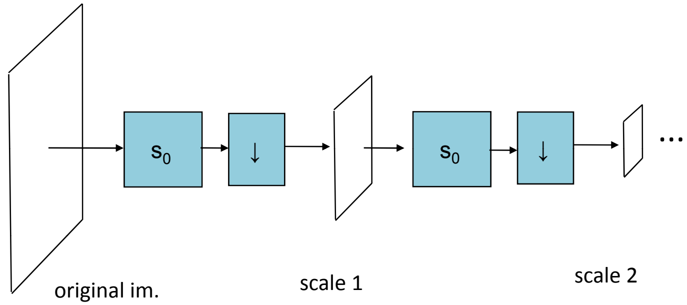
1. In the Fourier domain:
   1. 2D Fourier transform
   2. Multiplication with Gaussian function
   3. Inverse FT
2. Convolution: $f_s(x,y) = (f_0 * g_s)(x,y)$
3. Integrating f_s as a solution of the PDE:

$$f_{s+\Delta s} = f_s + \Delta s \cdot \frac{\partial f_s}{\partial s} = f_s + \Delta s \cdot \frac{1}{2} \nabla^2 f_s$$

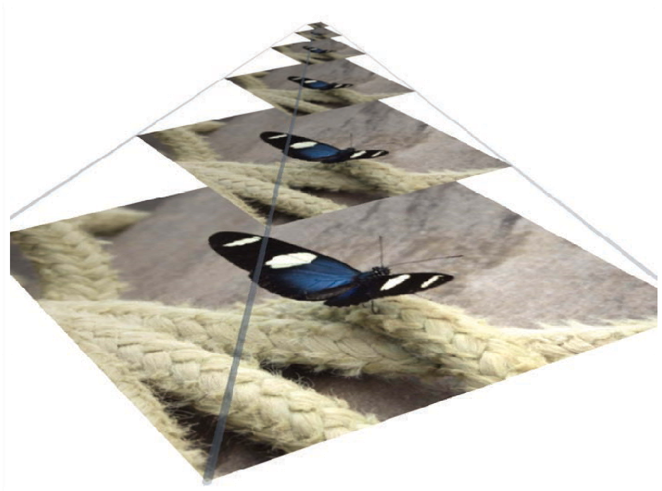
Representation: Scale Pyramid

- Blurring (LP-filtering) reduces high frequencies
- At some scale s_0 frequencies over $\pi/2$ are sufficiently attenuated to allow down-sampling with a factor 2 without severe aliasing
- At scale $2s_0$ we can down-sample the image with a factor 4, etc.

Gaussian Pyramid

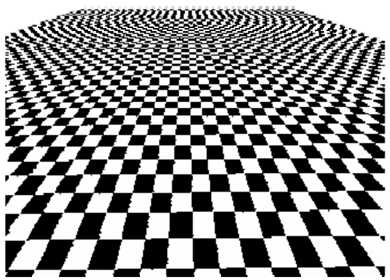


Example

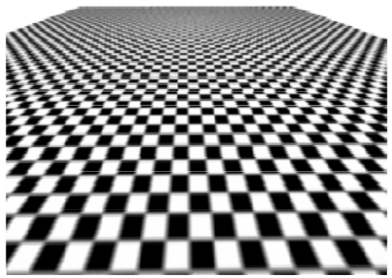


Representation: Scale Pyramid

- Used widely in Computer Graphics for texture resampling (called MIP maps)



Texture without MIP map

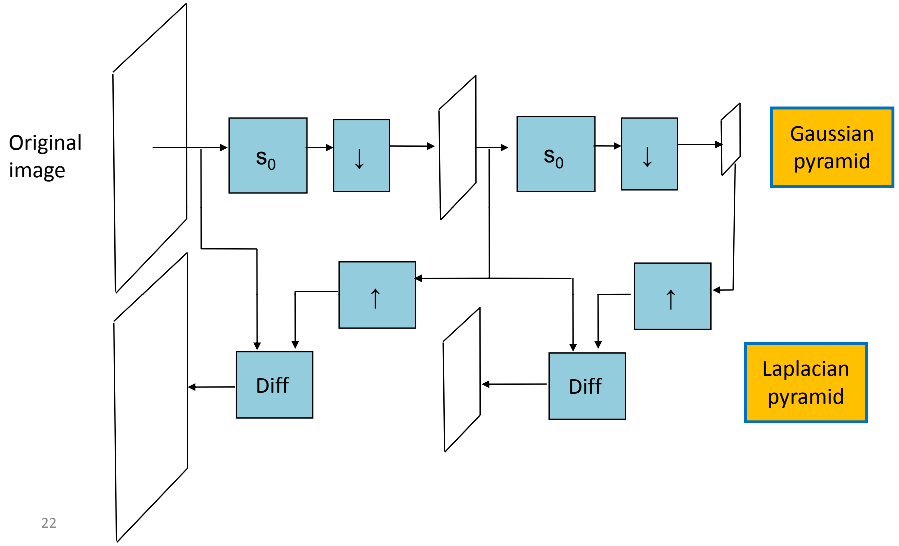


Texture with MIP map

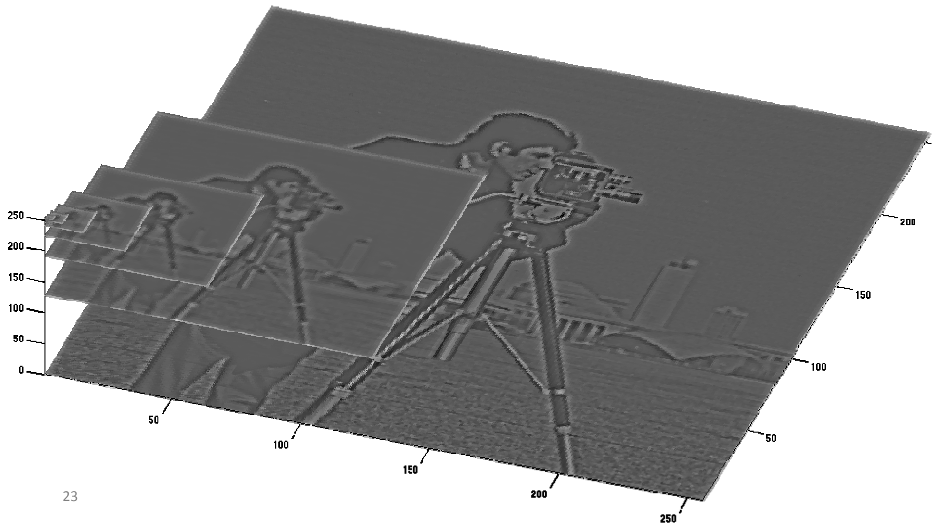
Laplacian Pyramid

- From a Gaussian pyramid, we can compute a *Laplacian pyramid*.
- Each level (scale) in a Laplacian pyramid is given as the **difference** between two levels of a Gaussian pyramid **at the same grid size**.
  - The coarser level needs to be up-sampled!
- The Laplacian pyramid contains no information about the DC-component of the image

Estimation: Laplacian Pyramid



Example

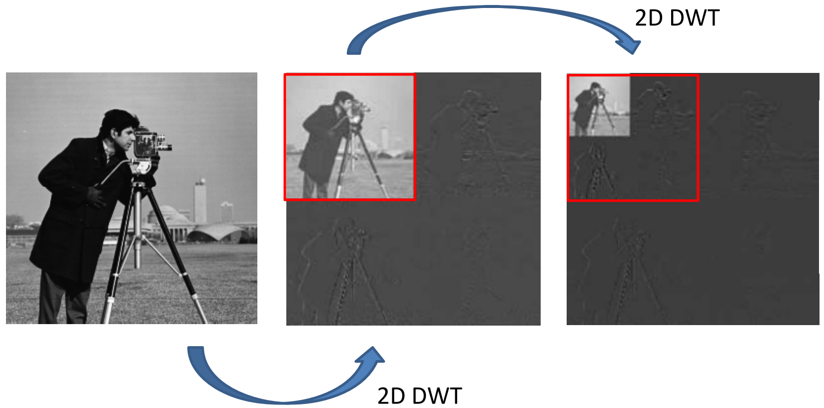


Completeness: Laplacian Pyramid

- The original image can be reconstructed from its Laplacian pyramid together with the coarsest level of its Gaussian pyramid
- How?

2D DWT, Example

- Another similar approach to scale spaces can be based on DWT

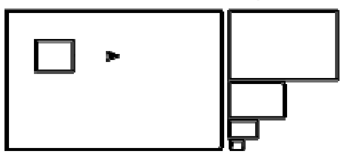


Analysis using scale hierarchies

- Scale-spaces, G/L-pyramids and DWT are examples of *scale hierarchies*
- Enables analysis of image features at different “sizes”, e.g. translations over different distances.
- The same analysis can be applied for detecting an object at any scale, but is then applied at all levels in a scale hierarchy (combination of analysis).

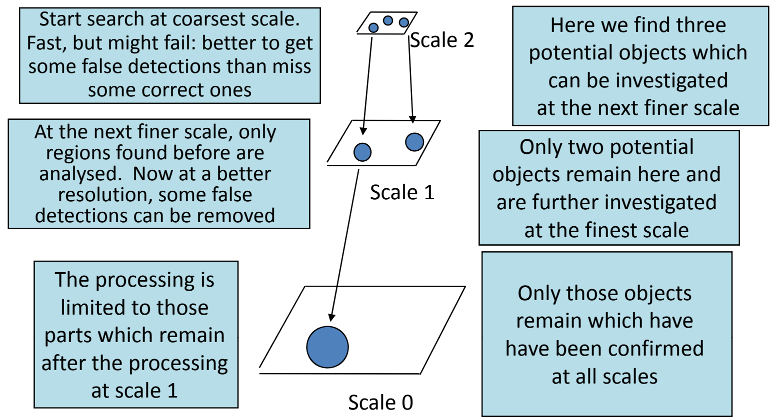
OR-approach

- Apply the same operation, e.g., for object detection, on all levels of a scale pyramid
  - Collect all detections as distinct objects
  - The level where a detection was made indicates the “size” of the object
- If each level is down-sampled a factor 2:
  - Time for searching over scale is bounded by a factor $(1 + ¼ + (¼)^2 + \ldots) = 4/3$



Example: face detection

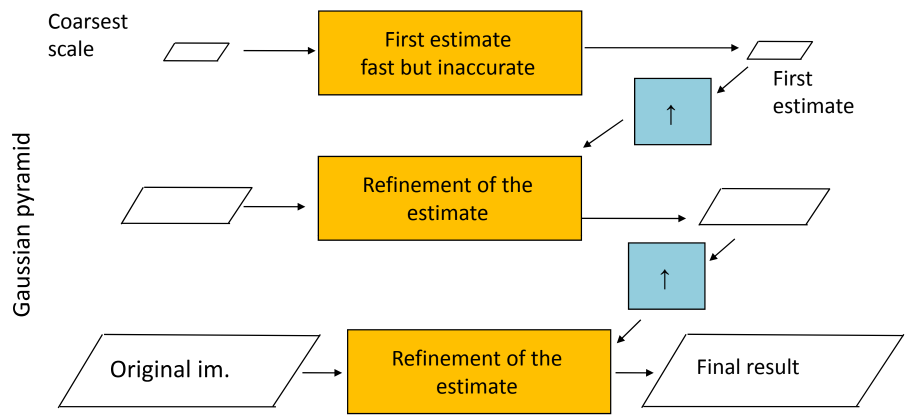
AND-approach: coarse-to-fine search



Coarse-to-fine refinement

- A different, but similar, processing scheme is the following:
  - Estimate a local feature at the coarsest scale first
    - Little data – fast processing
    - Coarse scale – inaccurate
  - The coarse estimate of the feature is then up-sampled to the size of the second coarsest scale, where the estimate is refined
  - The refinement is based on estimating the refinement of the coarsest estimate by analyzing the image at the second coarsest scale.
  - The refinement estimate is then up-sampled and refined again.
  - By repeating this procedure, we obtain a very accurate estimate of the feature at the finest scale.
- Example: estimation of local velocity or disparity

Coarse-to-fine refinement

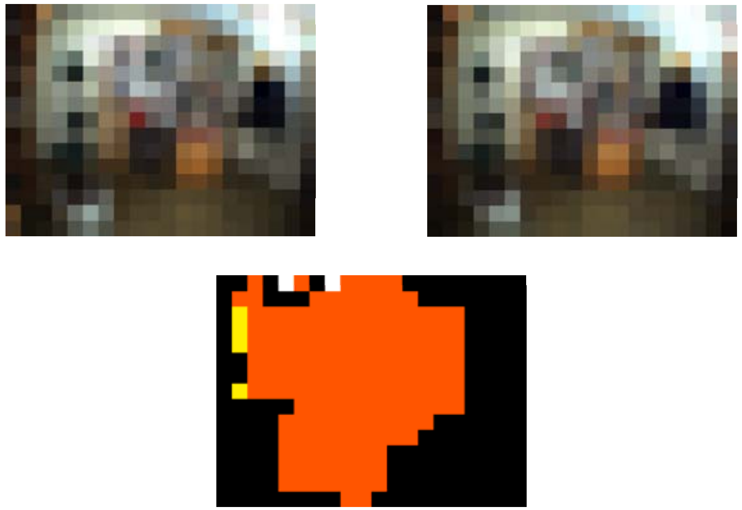


Example: Depth from stereo

Compute a scale hierarchy. Start estimating *disparity* at the coarsest level, and refine

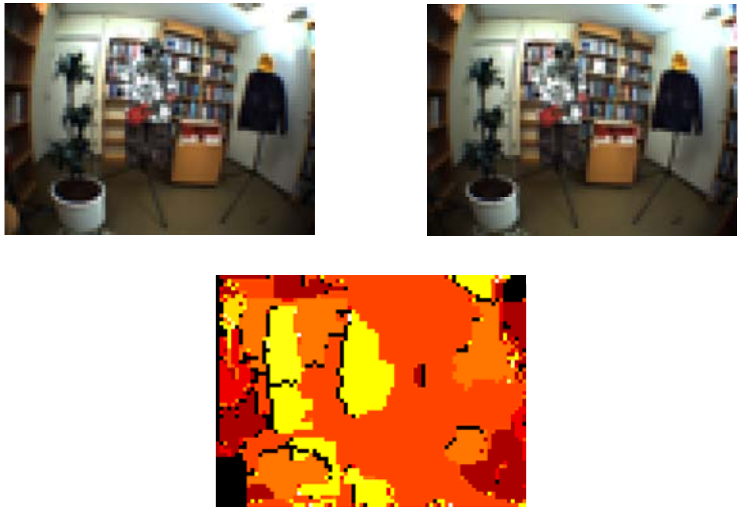
Images from Wallenberg & Forssén: *Teaching Stereo Perception to YOUR Robot*, BMVC 2012

Example: C2F Stereo disparity

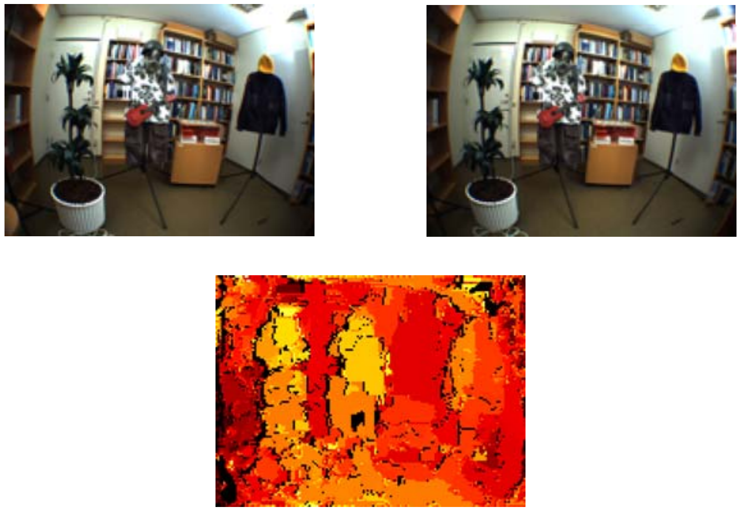


Example: C2F Stereo disparity

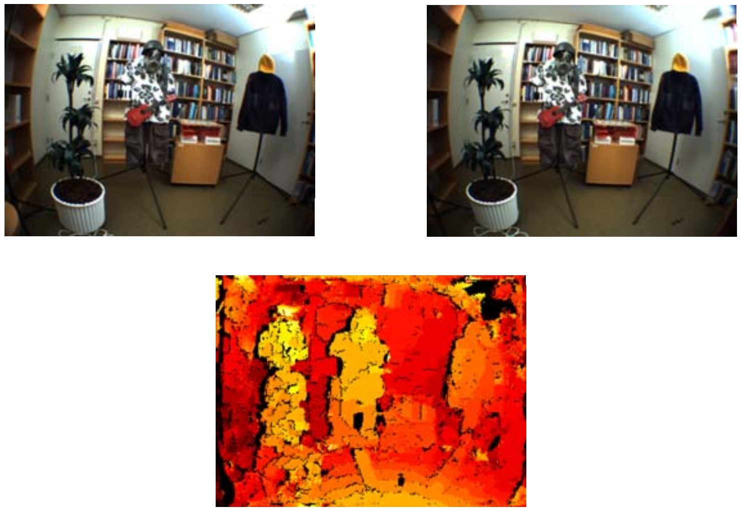
Example: C2F Stereo disparity



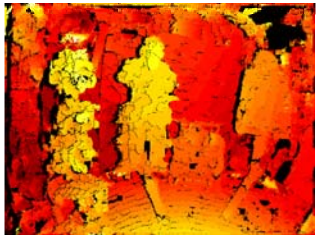
Example: C2F Stereo disparity



Example: C2F Stereo disparity



Example: C2F Stereo disparity



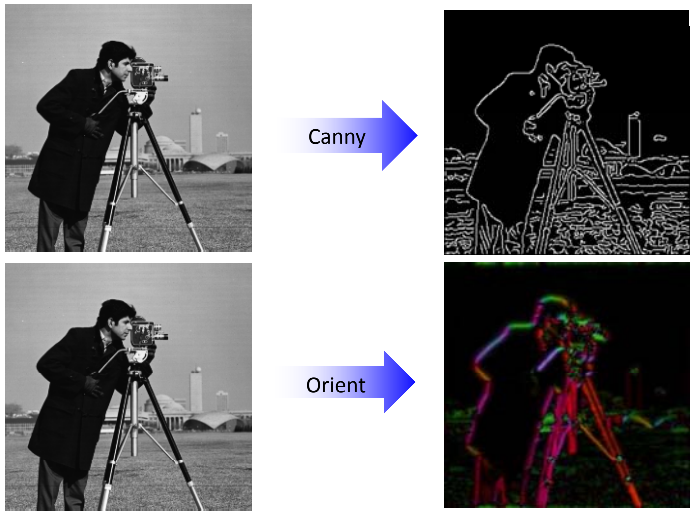
Images

- An image typically represents, at each position (*x*,*y*) a measurement of
  - Light intensity
  - Color
  - Absorption (X-ray)
  - Reflection (Ultrasonic)
  - Hydrogen content (MRI)
- All these represent physical phenomena
- All these can be input to a scale hierarchy

Feature image

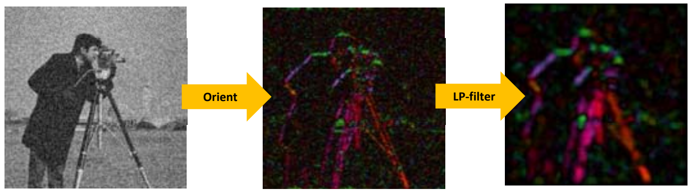
- The value at position (*x*,*y*) can also be used to represent a *local image feature*
- May not have a direct physical interpretation
  - Local mean or variance (scalars)
  - Local edge presence (binary)
  - Local gradient (a vector)
  - Local orientation (to be discussed)
  - Local curvature (to be discussed)
  - Interest points (to be discussed)

Edge representation



Notes on Representations

- If a local feature can be assumed to be constant in a neighborhood, it is desirable that its representation can be *locally averaged*
  - The averaged representation = the feature mean
  - Noise in the signal results in noise in the estimate of the feature representation
  - By low-pass filtering the representation (local mean value), the noise is reduced
  - In general: intensity changes faster than orientation

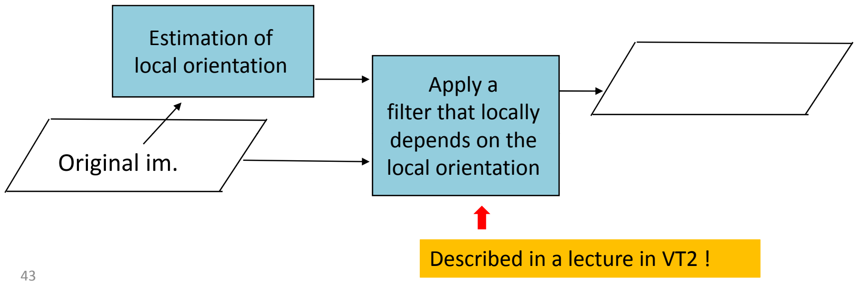


Confidence measure

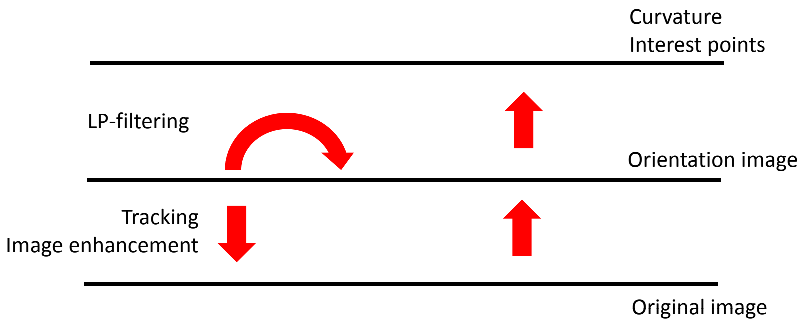
- Feature representations should contain a confidence measure (or variance estimate), separated from the feature estimate itself
  - Measures how confidence of the feature estimate
  - For example: in the range [0, 1]
    - Value 0: no confidence, value 1: max confidence
- The confidence can be used to weight the feature representation when estimating the mean value
  - Normalized convolution

Model-Based Processing

- Orientation images can be used to control the processing of an image
- Example: adaptive image enhancement

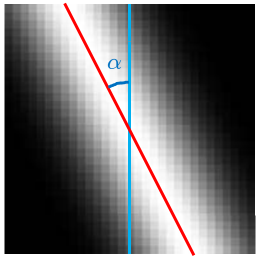


Orientation images Applications



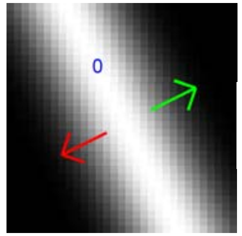
Representation of Local Orientation: Angle

- Signal model: simple signal (i1D, lecture 1)
- In a local region of each image point:
  - measure an angle α, e.g. between the vertical axis and the lines of constant signal intensity, e.g. in the interval 0 to 180˚
- Average-able?
  - No! (why?)
- Confidence measure?
- How to extend to 3D?



Estimation of Local Orientation: Gradient

- In each point we measure the local gradient of the signal (e.g. using a Sobel-operator)
- For an i1D signal, the sign of the gradient depends on where we do the measurement
- The gradient might be = 0 at certain lines of the i1D signal
- Confidence?

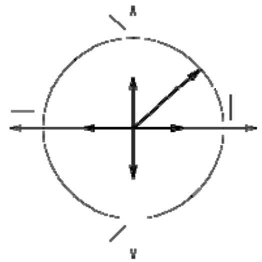


Representation of Local Orientation: Double angle vector

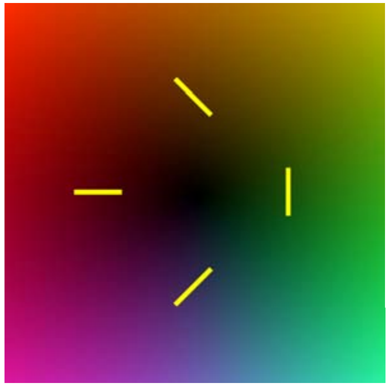
- Alternative: double the angle to 2α, which lies in the interval 0 to 360˚
- Form a 2D vector **v** which points with the angle 2α
- Let the norm of **v** represent the confidence measure
- Called: *double-angle representation* of local 2D orientation

Representation of Local Orientation

- The double-angle representations of two similar orientations are always similar (*continuity* results in *compatibility*)
- Two orientations which differ most (90˚) are always represented by vectors that point in opposite directions (*complementarity*)



Colour coding of the double angle representation



Representation of Local Orientation

- Double-angle representations of local 2D orientations can be averaged
  - The averaged representation = the feature mean
- Averaging of vectors is automatically weighted with the confidences

In later lectures:

- How to estimate the double-angle representation from image data?
- What to do in 3D?

Representation of Local Orientation

- Signal model for simple (i1D) signals

$$f(\mathbf{x}) = g(\mathbf{x}^T\hat{\mathbf{n}}) \qquad \hat{\mathbf{n}} = (\cos\alpha, \sin\alpha)^T$$

- *f* is the local signal (2 or more dimensions)
- *g* is the 1D function that defines the variations of the i1D signal
- **n** is a vector that defines the orientation
- BUT: the direction (sign) of **n** is not unique

Representation of Local Orientation: Tensor

- The double-angle vector **v** becomes

$$\mathbf{v} = \lambda\,(\cos 2\alpha, \sin 2\alpha)^T$$

- λ is a scalar which gives the confidence
- Alternative: form a 2 x 2 symmetric matrix

$$\mathbf{T} = \lambda\hat{\mathbf{n}}\hat{\mathbf{n}}^T = \lambda\begin{pmatrix}\cos\alpha\\ \sin\alpha\end{pmatrix}\begin{pmatrix}\cos\alpha & \sin\alpha\end{pmatrix}$$

- *Tensor representation of local orientation*

Representation of Local Orientation

- Tensor components

$$\mathbf{T} = \begin{pmatrix} T_{11} & T_{12} \\ T_{12} & T_{22} \end{pmatrix} = \begin{pmatrix} \cos^2\alpha & \cos\alpha\sin\alpha \\ \cos\alpha\sin\alpha & \sin^2\alpha \end{pmatrix}$$

- Vector components

$$\mathbf{v} = \begin{pmatrix} \cos 2\alpha \\ \sin 2\alpha \end{pmatrix} = \begin{pmatrix} \cos^2\alpha - \sin^2\alpha \\ 2\cos\alpha\sin\alpha \end{pmatrix} = \begin{pmatrix} T_{11} - T_{22} \\ 2\,T_{12} \end{pmatrix}$$

- The tensor contains one more element than **v**

Representation of Local Orientation

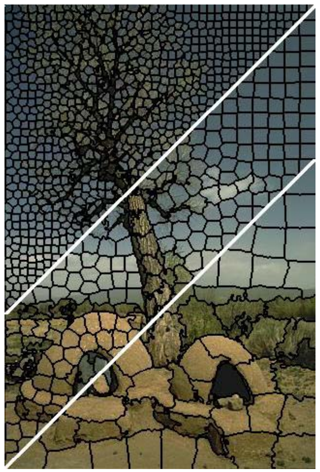
- **n** is an eigenvector of **T** with eigenvalue λ
- **T** (but not **v**) can be defined for any dimension of signals (3D, 4D, ...)
- How to estimate **v** and **T** from signals?

Tensor or Matrix?

- In this course, the term *tensor* is used as synonym for *symmetric matrix*.
- Why tensor and not matrix?
  - A matrix is just a representation, consisting of a container with numbers in a table.
  - A tensor can be represented as a matrix but it must furthermore obey certain laws under transformations of the coordinate system.

F_0

Super-pixels



Examples from Achanta *et al,* (SLIC)

Showing different sizes of the clusters

Super-pixels

- The array/matrix representation of an image implies that, in principle, each pixel must be examined in order to extract information about the image
- An alternative to the array/matrix representation is to **cluster** neighboring pixels with similar values to *super-pixels*
  - Often with restrictions on the cluster: size, shape
- Each super-pixel is represented as the common value and a cluster of pixels
- The image is represented as the set of its super-pixels
- Normal image: approx. 1 M pixels
- Super-pixels image: approx. 1 k super-pixels

Super-pixels

Typical approach:

- Initialize a regular grid of “square” super-pixels
- Iteratively modify each super-pixel to increase homogeneity regarding its corresponding pixel values
  - Split super-pixels into smaller ones if necessary
  - Merge similar super-pixels if possible
  - Move pixels from one super-pixel to a neighboring one to improve super-pixel shape