Master SIF - REP (Part 5)
Image filtering

Thomas Maugey (courtesy of Olivier Le Meur)
thomas.maugey@inria.fr

Université de Rennes

Inria

Fall 2023

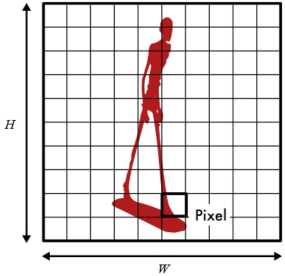
Outline

Outline

Introduction (1/3)

$$I : \Omega \subset \mathcal{N}^n \rightarrow \mathcal{R}^m$$

with $n,\ m \in \mathcal{N}$.



$$I : \Omega \subset \mathcal{N}^2 \rightarrow \mathcal{R}^3$$

with $\Omega = [H \times W]$.

Introduction (2/3)

The goal of a transformation is to get **a new representation** of the incoming picture. This new representation can be more convenient for a particular application or can ease the extraction of particular properties of the picture.

What is a transformation?

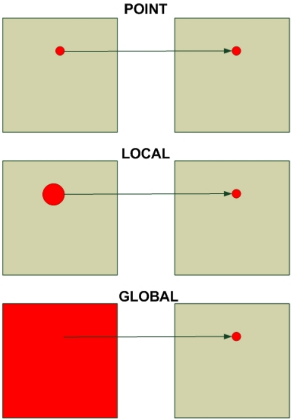
$$im[x, y] \xrightarrow{T} IM[u, v]$$

- im is the original image;
- IM is the transformed image;
- x, y represents the spatial coordinates of a pixel.

Introduction (2/3)

There exist 3 types of transformation:

- Point to point transformation:
  The output value at a specific coordinate is dependent only on one input value but not necessarily at the same coordinate (e.g. LUT, Histogram...);
- Local to point transformation:
  The output value at a specific coordinate is dependent on the input values in the neighborhood of that same coordinate;
- Global to point transformation:
  The output value at a specific coordinate is dependent on all the values in the input image (e.g. Fourier, Wavelet...).



Outline

2. Linear filtering
   - Linear filtering and convolution
   - Smoothing by averaging
   - Smoothing with a Gaussian kernel

Linear filtering and convolution

- Let $I : \Omega \subset \mathcal{N}^2 \rightarrow \mathcal{R}^m$ an input image;
- Let $\overline{I} : \Omega \subset \mathcal{N}^2 \rightarrow \mathcal{R}^n$ the transformed image.

Our goal is to fill in each location of $\overline{I}$ with a weighted sum of the pixel values from the locations surrounding the corresponding location in the image, using the same set of weights each time.

- Shift-invariant = the value of the output depends on the image neighbourhood; the position of the neighbourhood does not matter;
- Linear = the output for the sum of two images is the same as the sum of the outputs obtained for the images separately. An operator T is linear if:
  - $T(f+g) = T(f) + T(g)$, $\forall f, g$;
  - $T(\alpha f) = \alpha T(f)$, $\forall f$, scalars α.

Any linear shift-invariant operation can be represented by convolution.

Linear filtering and convolution

The convolution of a 2D filter K of size $2N+1 \times 2N+1$ $([-N,N] \times [-N,N])$ with an image I:

$$\overline{I}(i,j) = \sum_{l=-N}^{N} \sum_{p=-N}^{N} K(l,p) I(i-l, j-p) \tag{1}$$

We denote convolution as $\overline{I}(i,j) = K * I$.

- K is called the filter, kernel or mask.
- $K(0,0)$ is aligned with $I(i,j)$.

The output pixel's value is determined as a weighted sum of input pixel values.

Linear filtering

Convolution: shift-invariant linear systems

- Commutative: $F * H = H * F$;
- Associative: $F * (H * L) = (F * H) * L$
  - $(((F * H_1) * H_2) * H_3)$ is equivalent to applying one filter $F * (H_1 * H_2 * H_3)$
- Linearity, distributes over addition:
$$F * (H_1 + H_2) = (F * H_1) + (F * H_2)$$
- Scalars factor out: $kF * H = F * kH = k\,(F * H)$
- Shift-invariance: $H * shift(f) = shift(H * F)$
  - same behavior regardless of pixel location.
- Identity: unit impulse.

Linear filtering

Smoothing by averaging

The simplest filter is the average or box filter, which simply averages the pixel values in a $2N+1 \times 2N+1$ window. This is equivalent to convolving the image with a kernel of all ones and then scaling:

$$K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & \mathbf{1} & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad K = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & \mathbf{1} & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Three examples of averaging for different sizes of kernel. From the left-hand side to the right-side, $N = \{1, 3, 8\}$:

The amount of blur increases with the kernel's size.

The multivariate normal distribution of dimension k is defined by:

$$G_\Sigma(x_1,\ldots,x_k) = \frac{1}{(2\pi)^{k/2}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)\right)$$

where, $X = [x_1,\ldots,x_k]^T$ and $\mu = [\mu_1,\ldots,\mu_k]^T$ are vectors of size k, the symmetric covariance matrix Σ is positive definite, $|\Sigma|$ is the determinant of the covariance matrix.

$$\Sigma = \begin{bmatrix} \sigma_{11} & \ldots & \sigma_{1k} \\ \vdots & & \vdots \\ \sigma_{k1} & \ldots & \sigma_{kk} \end{bmatrix}$$

A symmetric matrix $n \times n$ composed of real numbers, noted M, is said to be positive definite if $z^T M z$ is positive for every non-zero column vector z of n real numbers.

Linear filtering

Smoothing with a Gaussian kernel

$$G_\Sigma(x_1, \ldots, x_k) = \frac{1}{(2\pi)^{k/2}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)\right)$$

Bivariate case (for k=2) considering $\mu = 0$:

- Isotropic: $\Sigma = \sigma^2 I_2 = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{2}$$

where σ is the standard deviation, σ^2 the variance.

Important remark:
$G_\sigma(x,y) = g_\sigma(x)g_\sigma(y)$ is separable, $g_\sigma(k) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{k^2}{2\sigma^2})$.

Linear filtering

Smoothing with a Gaussian kernel

$$G_\Sigma(x_1,\dots,x_k) = \frac{1}{(2\pi)^{k/2}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)\right)$$

Bivariate case (for k=2) considering $\mu = 0$:

➟ Diagonal covariance matrix: $\Sigma = \begin{bmatrix} \sigma_X^2 & 0 \\ 0 & \sigma_Y^2 \end{bmatrix}$

$$G_\Sigma(x,y) = \frac{1}{2\pi\sigma_X\sigma_Y} exp\left(-\frac{x^2}{2\sigma_X^2} - \frac{y^2}{2\sigma_Y^2}\right) \quad (3)$$

Important remark:
$G_\Sigma(x,y) = g_{\sigma_X}(x)g_{\sigma_Y}(y)$ is separable,
$g_\sigma(k) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{k^2}{2\sigma^2})$.

Linear filtering

Smoothing with a Gaussian kernel

$$G_\Sigma(x_1,\dots,x_k) = \frac{1}{(2\pi)^{k/2}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)\right)$$

Bivariate case (for k=2) considering $\mu = 0$:

- Anisotropic filtering: $\Sigma = \begin{bmatrix} \sigma_X^2 & \sigma_{XY} \\ \sigma_{XY} & \sigma_Y^2 \end{bmatrix} = \begin{bmatrix} \sigma_X^2 & \rho\sigma_X\sigma_Y \\ \rho\sigma_X\sigma_Y & \sigma_Y^2 \end{bmatrix}$
  with ρ the correlation coefficient: $\rho = \frac{\sigma_{XY}}{\sigma_X\sigma_Y}$.

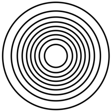
$$G_\Sigma(x,y) = \frac{1}{2\pi\sigma_X\sigma_Y\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2(1-\rho^2)}\left[\frac{x^2}{\sigma_X^2} + \frac{y^2}{\sigma_Y^2} - \frac{2\rho xy}{\sigma_X\sigma_Y}\right]\right)$$

- the direction of the filtering is defined by the first eigenvector of the covariance matrix Σ;
- its strength is defined by its corresponding eigenvalue.

Linear filtering

Smoothing with a Gaussian kernel

- Isotropic Gaussian kernel, $\Sigma = \sigma^2 I_2$:



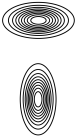
$$K = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & \mathbf{4} & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad K = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & \mathbf{36} & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

From left to right: orig. image; $\sigma = 1$, $\sigma = 4$, $\sigma = 16$.

Linear filtering

Smoothing with a Gaussian kernel

- Diagonal covariance matrix, $\Sigma = \begin{bmatrix} \sigma_X^2 & 0 \\ 0 & \sigma_Y^2 \end{bmatrix}$:



Example: $\sigma_X = 1$ and $\sigma_Y = 16$

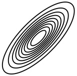
$$\begin{aligned} K_x &= [0.0544\ 0.244\ 0.402\ 0.244\ 0.0544] \\ K_y &= [0.199\ 0.2\ 0.2\ 0.2\ 0.199] \\ K &= K_y^T * K_x \end{aligned}$$

From left to right: orig. image; $\sigma_X = 16$ and $\sigma_Y = 1$, $\sigma_X = 1$ and $\sigma_Y = 16$.

Linear filtering

Smoothing with a Gaussian kernel

➟ Anisotropic filtering, $\Sigma = \begin{bmatrix} \sigma_X^2 & \rho\sigma_X\sigma_Y \\ \rho\sigma_X\sigma_Y & \sigma_Y^2 \end{bmatrix}$:



$$G_\Sigma(x,y) = \frac{1}{2\pi\sigma_X\sigma_Y\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2(1-\rho^2)}\left[\frac{x^2}{\sigma_X^2}+\frac{y^2}{\sigma_Y^2}-\frac{2\rho xy}{\sigma_X\sigma_Y}\right]\right)$$

can be reformulated as

$$G_\theta(x,y) = \frac{1}{2\pi\sigma_u}\exp\left(-\frac{1}{2}\left[\frac{u^2}{\sigma_u^2}\right]\right) * \frac{1}{2\pi\sigma_v}\exp\left(-\frac{1}{2}\left[\frac{v^2}{\sigma_v^2}\right]\right)$$

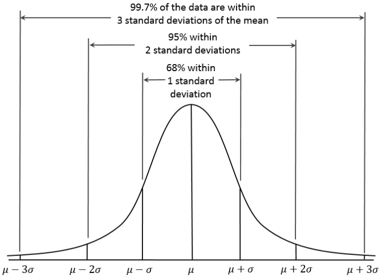
where,

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} \quad (4)$$

the u-axis being in the direction of θ, and the v-axis being orthogonal to θ. A fast implementation is presented in (Geusebroek et al., 2003).

Some helpful information:

- For a normal distribution, 68.27%, 95.45% and 99.73% of the values lie within one, two and three standard deviations of the mean, respectively:



- The parameter a controls the decay of the Gaussian function, $g(k) = e^{-\frac{|k|^2}{2a^2}}$, $a \in \mathbb{R}_+$. For a $N \times N$ patch, a reasonable choice for a is $a = \frac{N-1}{4}$. The Gaussian weights vary in $[e^{-4}, e^{-2}] \simeq [0.018, 0.05]$ on the patch boundary.

Outline

3. Frequency domain filtering
   - Ideal low pass filter
   - Butterworth low pass filter
   - Gaussian low pass filter
   - High pass filtering
   - Laplacian in the frequency domain

Frequency domain filtering

- Discrete and bi-dimensional Fourier transformation of an image im having a size $N \times M$.

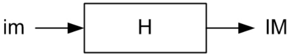
Inverse Fourier Transform:

$$im[k,l] \quad = \quad \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} IM[u,v] exp(j2\pi(\frac{k}{N}u + \frac{l}{M}v))$$

Fourier Transform:

$$IM[u,v] \quad = \quad \frac{1}{\sqrt{NM}} \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} im[k,l] exp(-j2\pi(\frac{k}{N}u + \frac{l}{M}v))$$

Frequency domain filtering



where H is the convolution kernel.

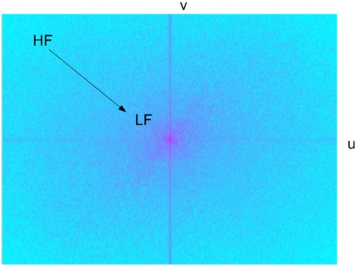
$$IM[x,y] = (im * h)[x,y] \tag{5}$$

$$im_1[x,y] * im_2[x,y] \xrightarrow{\mathcal{F}} IM_1[u,v] \times IM_2[u,v]$$
$$im_1[x,y] \times im_2[x,y] \xrightarrow{\mathcal{F}} IM_1[u,v] * IM_2[u,v]$$

When the size of the kernel is large, it is better to apply the filter in the frequency domain.

For more information:
Digital Image Processing, by R. C. Gonzalez and R. E. Woods, 3rd edition, Pearson Prentice Hall, 2008.

Frequency domain filtering



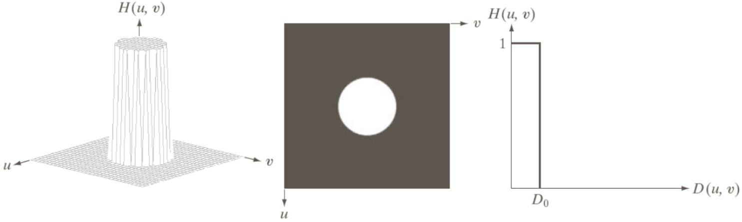
We can spatially filter an image by Fourier transforming and applying a frequency filter:

$$
\begin{aligned}
IM[x,y] &= im[x,y] * h[x,y] \\
I\tilde{M}[u,v] &= IM[u,v] \times H[u,v]
\end{aligned}
$$

where, $H[u,v]$ is the filter in the frequency function.

Ideal low pass filter (1/2)

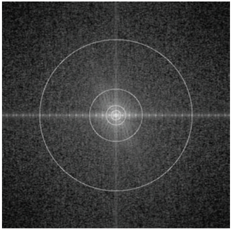
From the left-hand side to the right: Ideal low pass filter transfert function, filter displayed as an image, filter radial cross section.



$$H(u,v) = \begin{cases} 1 & D(u,v) \leq D_0 \\ 0 & D(u,v) > D_0 \end{cases}$$

With D the euclidean distance from the spectrum center $(\frac{N}{2}, \frac{N}{2})$.

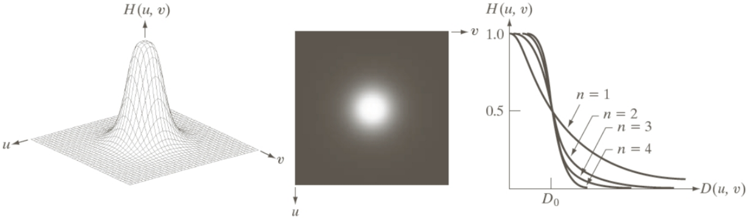
Ideal low pass filter (2/2)



Low pass filtering:

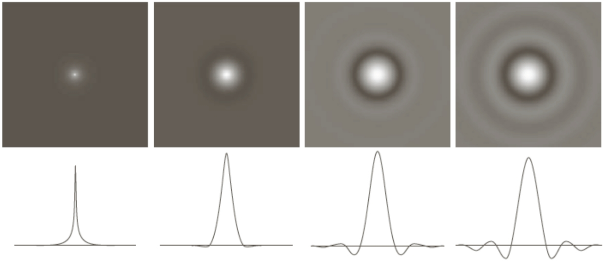
Ringing and blurring

Butterworth low pass filter (1/2)



$$H(u,v) = \frac{1}{1 + \left(\frac{D(u,v)}{D_0}\right)^{2n}}$$

Butterworth low pass filter (2/2)

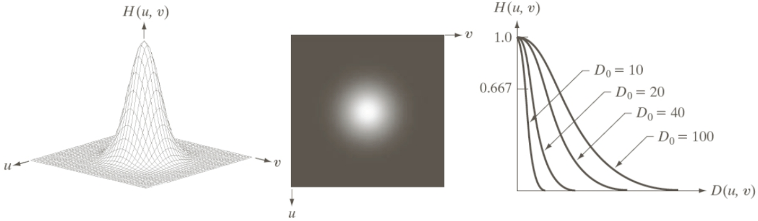


Top: spatial representation of the filter for different orders;
Bottom: intensity profiles through the center of the filters.

Butterworth low pass filtering:

Smooth transition in blurring, no ringing is present.

Gaussian low pass filter (1/1)

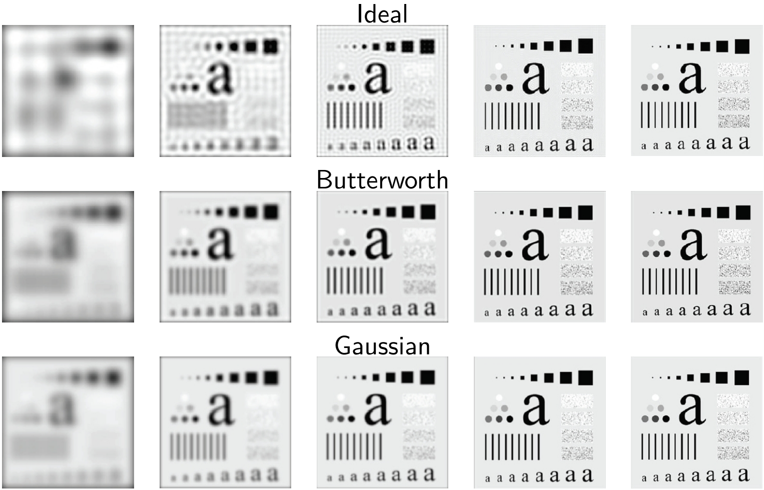


$$H(u,v) = exp\left(-\frac{D(u,v)^2}{2D_0^2}\right)$$

with $D_0 = \sigma$. Gaussian low pass filtering:

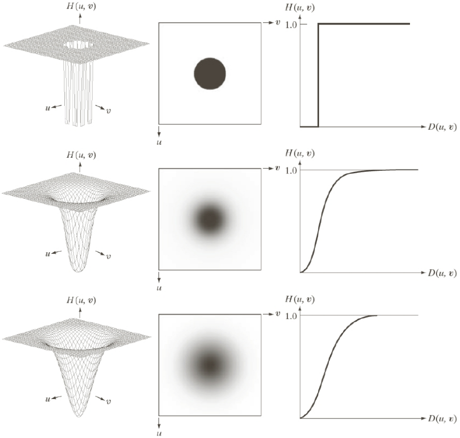
Smooth transition in blurring, no ringing is present.

Low pass filter (1/1)

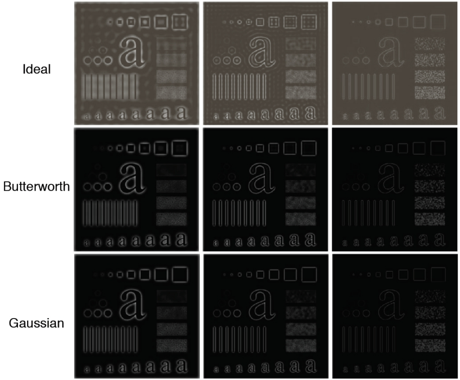


High pass filtering (1/2)

$$H_{HP}(u,v) = 1 - H_{LP}(u,v) \quad (6)$$



High pass filtering (2/2)



- Ideal high-pass filters enhance edges but suffer from ringing artifacts, just like Ideal LPF;
- Smoother results with the two others.

Laplacian in the frequency domain (1/2)

We remind:

- $\nabla^2 im[k,l] = \frac{\partial^2 im}{\partial^2 k}[k,l] + \frac{\partial^2 im}{\partial^2 l}[k,l]$
- $\frac{d^n x(t)}{dt^n} \xrightarrow{\mathcal{F}} (j2\pi f)^n X(f)$.

It follows that

$$\frac{\partial^2 im}{\partial^2 k}[k,l] + \frac{\partial^2 im}{\partial^2 l}[k,l] \xrightarrow{\mathcal{F}} -(\frac{2\pi}{N})^2(u^2+v^2)IM[u,v]$$

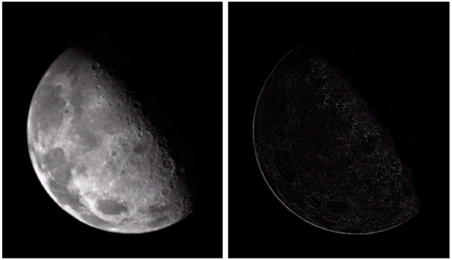
The Laplacian filter is then implemented in the frequency domain by

$$H(u,v) = -(\frac{2\pi}{N})^2(u^2+v^2) \quad (7)$$

Laplacian in the frequency domain (2/2)

Finally, to compute the Laplacian, we need :

1. to compute the Fourier transform of the picture;
2. to multiply the spectrum by $-(\frac{2\pi}{N})^2(u^2+v^2)$;
3. to compute the inverse Fourier transform.



Outline

4. Non-Linear filtering
   - Objective
   - Median
   - Adaptive filtering
   - Conditional mean
   - Anisotropic Kuwahara filtering
   - Yaroslavsky filter
   - Bilateral filter
   - Joint bilateral filter
   - Non-Local means
   - Guided filter

Objective

- Linear filter: $T(f+g) = T(f) + T(g)$, $\forall f, g$; $T(\alpha f) = \alpha T(f)$, $\forall f$, scalars α.

  Each output pixel is a weighted summation of some number of input pixels using the same set of weights each time.

  - Tend to blur edges and other image detail;
  - Perform poorly with non-Gaussian noise.

- Non-linear filter: $T(f+g) \neq T(f) + T(g)$
  - Can preserve edges;
  - Very effective at removing impulsive noise.

  Non-linear filters are able to tailor themselves to the local properties and structures of an image.

Median (1/2)

The median filter consists in selecting the middle pixel intensity in the sorted list of neighborhood pixels as the output.

$$\bar{I}(\mathbf{x}) = med\left(\{I(\mathbf{y}) | I(\mathbf{y}) \in \mathcal{N}(\mathbf{x})\}\right)$$

where $\mathcal{N}(\mathbf{x})$ the neighbourhood centered at $\mathbf{x} = (i, j)$.

➠ very good to remove impulse noise!

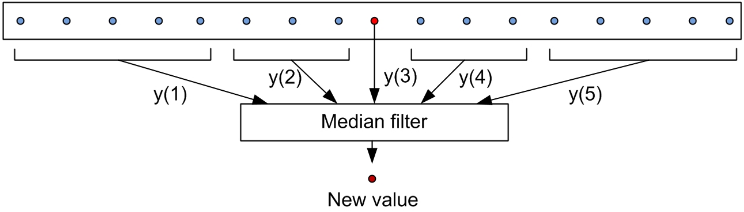
Median (2/2)

However, when the number of the samples is large, the ordering procedure becomes cumbersome.

Idea: the median filter is taken over the outputs of several FIR substructures and the number of the substructures is much smaller than the number of the data samples inside the filter window.

$$IM[x,y] = MED\left(y(1),\ldots,y(m)\right)$$

where, m is linear FIR filters.



Adaptive filtering (1/1)

An adaptive filter is a filter that self-adjusts its transfer function according to an optimizing algorithm.

The goal is still to smooth the signal. However, we want to preserve edges...

- Filtering by pixel grouping;
- Conditional mean, Bilateral filtering and mean shift filter;
- Diffusion (linear, non-linear, isotropic, anisotropic).

Conditional mean (1/1)

Pixels in a neighbourhood are averaged only if they differ from the central pixel by less than a given threshold:

$$\underbrace{IM[x,y]}_{\text{Output}} = \sum_{k \in V(x,y)} \sum_{l \in V(x,y)} \underbrace{h(k,l)}_{\text{Filter coeff.}} \underbrace{im[x-k, y-l]}_{\text{Input}}$$

$$h(k,l) = \begin{cases} 1 & \text{if } |im[x-k,y-k] - im[k,l]| < TH \\ 0 & \text{Otherwise.} \end{cases}$$

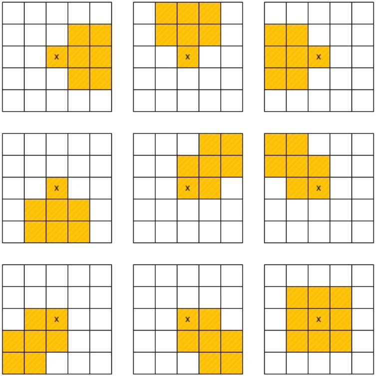
Example with a neighbourhood equal $(2 \times 3 + 1)(2 \times 3 + 1)$,$TH = 32$:

Anisotropic Kuwahara filtering (1/2)

Method proposed by Kuwahara and adapted by Nagao in 1980.

- Selection of the sub-domain that has the minimum variance (9 windows for Nagao);
- Replace the value of the central pixel by the average value of the sub-domain having the minimum variance.

Example for a window 5×5:



Anisotropic Kuwahara filtering (2/2)

Yaroslavsky filter (1/1)

Yaroslavsky filter consists in averaging **neighboring pixels** which also have a **similar color value** (Yaroslavsky and Yaroslavskij, 1985).

- Spatial neighborhood: $\mathcal{N}_\rho(\mathbf{x}) = \{\mathbf{y} \in \Omega | \|\mathbf{y} - \mathbf{x}\| < \rho\}$

$$\bar{I}(\mathbf{x}) = \frac{1}{C(\mathbf{x})} \sum_{\mathbf{y} \in \mathcal{N}_\rho(\mathbf{x})} I(\mathbf{y}) w_r(\mathbf{x}, \mathbf{y})$$

where,

- the weighting coefficients $w_r(\mathbf{x}, \mathbf{y}) = exp\left(-\frac{\|I(\mathbf{y}) - I(\mathbf{x})\|^2}{4h^2}\right)$ are data-dependent;
- normalization factor, $C(\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{N}_\rho(\mathbf{x})} w_r(\mathbf{x}, \mathbf{y})$.

Bilateral filter (1/7)

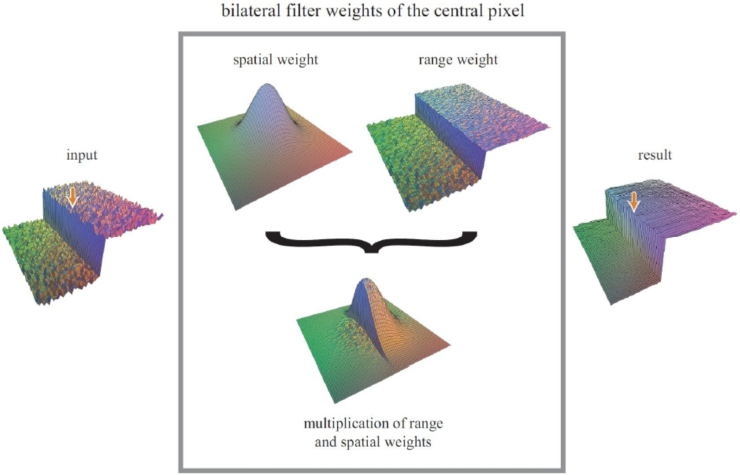
(Tomasi and Manduchi, 1998) improve Yaroslavsky's method by involving a bilateral gaussian function depending on both **grey level** and **space**. The output pixel value depends on a weighted combination of neighboring pixel values.

$$\bar{I}(\mathbf{x}) = \frac{1}{C(\mathbf{x})} \sum_{\mathbf{y} \in \mathcal{N}(\mathbf{x})} I(\mathbf{y}) w(\mathbf{x}, \mathbf{y})$$

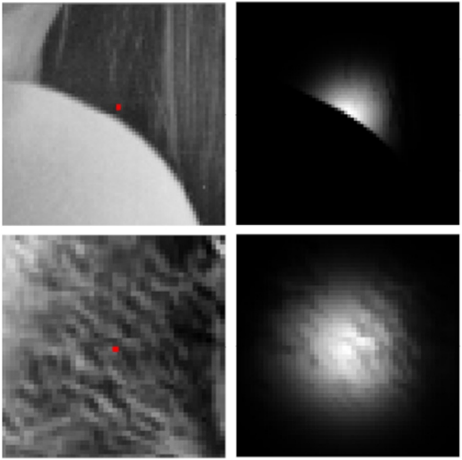
where,

- The weighting coefficients $w(\mathbf{x}, \mathbf{y}) = w_r(\mathbf{x}, \mathbf{y}) \times w_d(\mathbf{x}, \mathbf{y})$;
- Range coefficients $w_r(\mathbf{x}, \mathbf{y}) = exp\left(-\frac{\|I(\mathbf{y}) - I(\mathbf{x})\|^2}{4h_r^2}\right)$;
- Space-dependent coefficients $w_d(\mathbf{x}, \mathbf{y}) = exp\left(-\frac{\|\mathbf{y} - \mathbf{x}\|^2}{4h_d^2}\right)$
- Normalization factor, $C(\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{N}(\mathbf{x})} w_r(\mathbf{x}, \mathbf{y}) w_d(\mathbf{x}, \mathbf{y})$.

Bilateral filter (2/7)



Bilateral filter (3/7)



Left: input mage. Right: Kernel of the bilateral filter centered on the red dot.

Bilateral filter (4/7)

- Bilateral Filtering for color images:

$$\bar{\mathbf{I}}(\mathbf{x}) = \frac{1}{C(\mathbf{x})} \sum_{\mathbf{y} \in \mathcal{N}(\mathbf{x})} exp\left(-\frac{\|\mathbf{I}(\mathbf{x}) - \mathbf{I}(\mathbf{y})\|^2}{4h_r^2}\right) exp\left(-\frac{\|\mathbf{y} - \mathbf{x}\|^2}{4h_d^2}\right) \mathbf{I}(\mathbf{y})$$

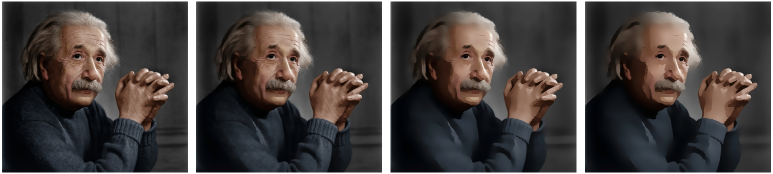
where, $\mathbf{I}$ is a vector (RGB, Lab, RGB-D...).

- Iterating the bilateral filter:

$$\mathbf{I}_{(n+1)} = BF\left[\mathbf{I}_{(n)}\right]$$

Used for generating more piecewise-flat images.

Bilateral filter (5/7)



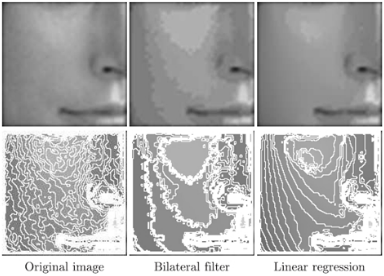
Iterated bilateral filtering in Lab space. From left to right: orig.; one iteration, 5 and 9 iterations.

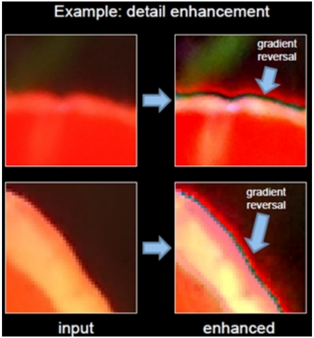
Low contrast texture has been removed and edges are well preserved.

Bilateral filter (6/7)

Limitations of bilateral filter

- Staircase effects: Piecewise constant assumption;
- Gradient reversal artifacts: Over-sharpening effect.





Extracted from (He et al., 2010, 2013).

Bilateral filter (7/7)

To conclude:

- Not always the best result but often good;
- Easy to understand;
- Bilateral goals are:
  - Local smoothing within similar regions
  - Edge-preserving smoothing
  - Separate large structure & fine detail
  - Eliminate outliers
  - Filter within edges, not across them

New variants: joint bilateral Filtering, trilateral filter, non-local means.

Extracted from *Image Filtering 2.0: Efficient Edge-Aware Filtering and Their Applications*, A Tutorial at IEEE ICIP 2013

Joint bilateral filter (1/3)

Given an image I, the cross bilateral filter smoothes I while preserving the edges of a second image E. In practice, the range weight is computed using E instead of I (Eisemann and Durand, 2004, Petschnigg et al., 2004).

$$\bar{I}(\mathbf{x}) = \frac{1}{C(\mathbf{x})} \sum_{\mathbf{y} \in \mathcal{N}(\mathbf{x})} I(\mathbf{y}) w(\mathbf{x}, \mathbf{y})$$

where,

- The weighting coefficients $w(\mathbf{x}, \mathbf{y}) = w_r(\mathbf{x}, \mathbf{y}) \times w_d(\mathbf{x}, \mathbf{y})$;
- Range coefficients $w_r(\mathbf{x}, \mathbf{y}) = exp\left(-\frac{\|E(\mathbf{y}) - E(\mathbf{x})\|^2}{4h_r^2}\right)$;
- Space-dependent coefficients $w_d(\mathbf{x}, \mathbf{y}) = exp\left(-\frac{\|\mathbf{y} - \mathbf{x}\|^2}{4h_d^2}\right)$
- Normalization factor, $C(\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{N}(\mathbf{x})} w_r(\mathbf{x}, \mathbf{y}) w_d(\mathbf{x}, \mathbf{y})$.

Joint bilateral filter (2/3)

$$\bar{I}(\mathbf{x}) = \frac{1}{C(\mathbf{x})}\sum_{\mathbf{y}\in\mathcal{N}(\mathbf{x})} exp\left(-\frac{\|E(\mathbf{y})-E(\mathbf{x})\|^2}{4h_r^2}\right) exp\left(-\frac{\|\mathbf{y}-\mathbf{x}\|^2}{4h_d^2}\right) I(\mathbf{y})$$

Output, Guide and Input noisy image.

Color

T. Maugey

Joint bilateral filter (3/3)

$$\bar{I}(\mathbf{x}) = \frac{1}{C(\mathbf{x})} \sum_{\mathbf{y} \in \mathcal{N}(\mathbf{x})} exp\left(-\frac{\|E(\mathbf{y}) - E(\mathbf{x})\|^2}{4h_r^2}\right) exp\left(-\frac{\|\mathbf{y} - \mathbf{x}\|^2}{4h_d^2}\right) I(\mathbf{y})$$

Output, Guide and Input noisy image.

Non-Local means (NLM) (1/3)

Take advantage of high degree of redundancy of natural images
The most similar pixels to a given pixel
may be also quite far from the current pixel....

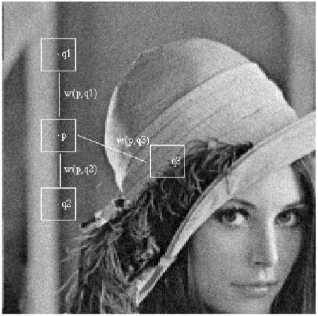


FIG. 6. *q1 and q2 have a large weight because their similarity windows are similar to that of p. On the other side the weight w(p, q3) is much smaller because the intensity grey values in the similarity windows are very different.*

From (Buades et al., 2005).

Non-Local means (NLM) (2/3)

$$\bar{I}(\mathbf{x}) = \frac{1}{C(\mathbf{x})} \sum_{\mathbf{y} \in \Omega} I(\mathbf{y}) w(\mathbf{x}, \mathbf{y})$$

where,

- The weighting coefficient is given by:

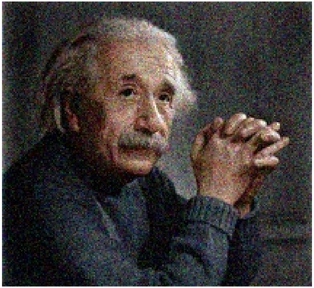
$$w(\mathbf{x}, \mathbf{y}) = exp\left(-\frac{\|\psi_{p_\mathbf{x}} - \psi_{p_\mathbf{y}}\|^2_{2,a}}{h^2}\right)$$

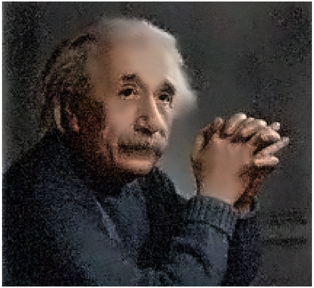
  $\psi_{p_\mathbf{x}}$ represents a patch of texture centered at $\mathbf{x}$, $\|.\|^2_{2,a}$ is the Gaussian weighted squared Euclidean distance.

Main steps:

- ➡ Looking for the most similar patches;
- ➡ Compute w between the current and similar patches;
- ➡ Compute the output value by a weighted linear combination.

Non-Local means (NLM) (3/3)

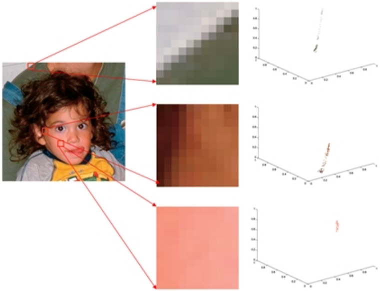




Guided filter (1/8)

Introduction

The Color Lines model of an image is a list of lines representing the image's colors along with a metric for calculating the distance between every pixel and each Color Line (Omer and Werman, 2004).



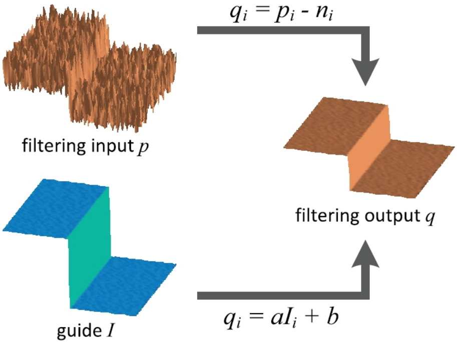
$$\alpha_i = \mathbf{a}^{\mathrm{T}}\mathbf{I}_i + b, \forall i \in \omega$$

Model for representing a color image in the RGB space.

Guided filter (2/8)

The guided filter involves an input image p, a guidance image I, and an output image q. Both I and p are given beforehand according to the application, and they can be identical (He et al., 2010, 2013).

The key assumption of the guided filter is **a local linear model between the guidance I and the filtering output q**.



Guided filter (3/8)

The key assumption of the guided filter is **a local linear model between the guidance I and the filtering output q**.

$$q_i = a_k I_i + b_k, \forall i \in w_k$$

where, w_k is a window centered at the pixel k. (a_k, b_k) are some linear coefficients assumed to be constant in w_k.
The coefficients (a_k, b_k) are computed by minimizing the following cost function in w_k:

$$\min_{(a_k,b_k)} E(a_k, b_k) = \min_{(a_k,b_k)} \sum_{i \in w_k} \left((a_k I_i + b_k - p_i)^2 + \epsilon a_k^2 \right)$$

Linear regression: $\frac{\partial E}{\partial a_k} = 0$ and $\frac{\partial E}{\partial b_k} = 0$.

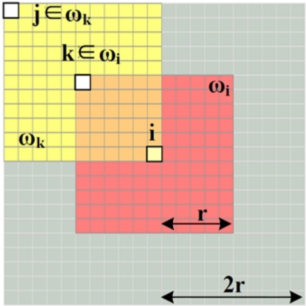
$$a_k = \frac{cov_k(I, p)}{var_k(I) + \epsilon} \qquad b_k = \overline{p} - a_k \overline{I}$$

Guided filter (4/8)

- Extension to the entire image by simple summation:
  1. Compute a_k and b_k, for all windows w_k.
  2. Compute the average of $a_k I_i + b_k$ in all w_k that covers pixel q_i.

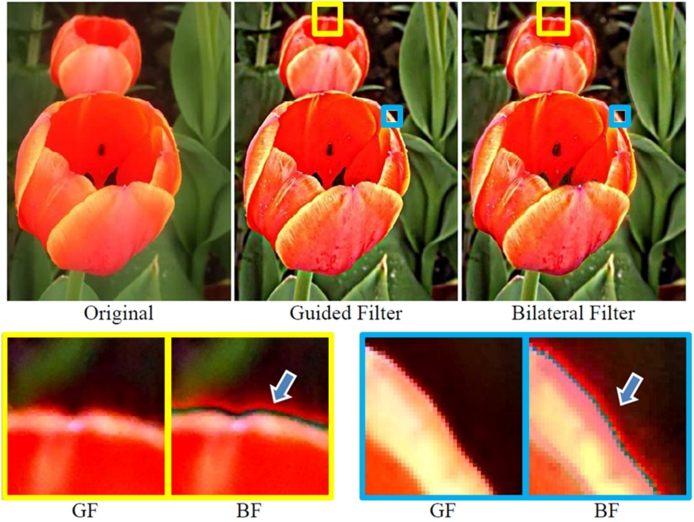
$$q_i = \frac{1}{|w|} \sum_{k,i \in w_k} (a_k I_i + b_k)$$

$$q_i = \overline{a_i} I_i + \overline{b_i}$$



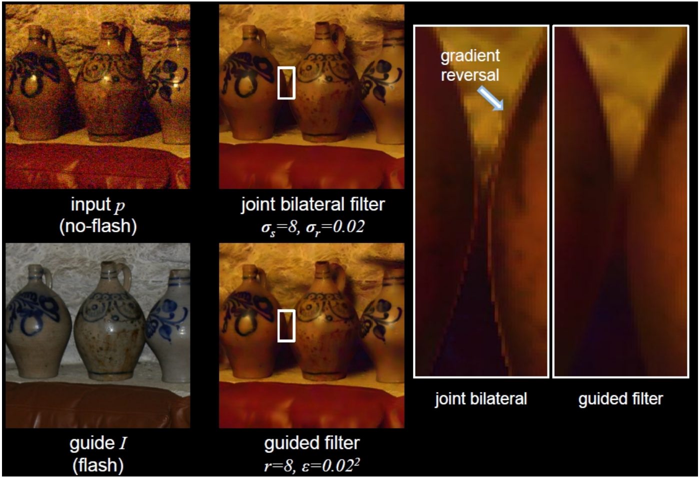
Guided filter (5/8)

- Smoothing:



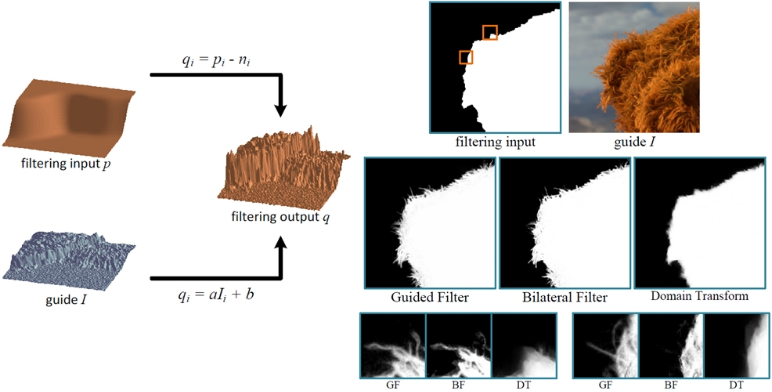
Guided filter (6/8)

- Flash/no flash denoising:



Guided filter (7/8)

➠ Beyond smoothing! Texture transfer (ϵ small):



Guided filter (8/8)

- Matlab code and OpenCV code available;
- Avantages:
  - fast, accurate;
  - edge-preserving;
  - non-iterative.
- Limitations:
  - lacking of a rigorous justification of the aggregation step;
  - lacking of spatial adaptivity;
  - ineffective if more color models present in a patch.

Materials are coming from:
Image Filtering 2.0: Efficient Edge-Aware Filtering and Their Applications.
A Tutorial at IEEE Int. Conf. on Image Processing (ICIP) 2013.
https://sites.google.com/site/filteringtutorial/

Outline

5. Derivative filters (Edge detection)
   - Derivatives with Finite Differences
   - Image gradient
   - Canny edge detector
   - Laplacian

Estimating Derivatives with Finite Differences (1/3)

From Taylor's theorem (approximation of a k-times differentiable function f around a given point x_0), we can write:

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x-x_0)^k + R_n(x)$$

$$f(x) = f(x_0) + f'(x_0)(x-x_0) + \frac{f''(x_0)}{2!}(x-x_0)^2 + \cdots$$

By a simple variable change:

$$f(x_0+\delta x) = f(x_0) + f'(x_0)\delta x + \frac{f''(x_0)}{2!}\delta x^2 + \cdots$$

$$f(x_0-\delta x) = f(x_0) - f'(x_0)\delta x + \frac{f''(x_0)}{2!}\delta x^2 + \cdots$$

Estimating Derivatives with Finite Differences (2/3)

$$f(x_0 + \delta x) = f(x_0) + f'(x_0)\delta x + \frac{f''(x_0)}{2!}\delta x^2 + \cdots \quad (8)$$

$$f(x_0 - \delta x) = f(x_0) - f'(x_0)\delta x + \frac{f''(x_0)}{2!}\delta x^2 + \cdots \quad (9)$$

We deduce:

- First order forward finite difference:

$$f'(x_0) \approx \frac{f(x_0 + \delta x) - f(x_0)}{\delta x} \quad (10)$$

- First order backward finite difference:

$$f'(x_0) \approx \frac{f(x_0) - f(x_0 - \delta x)}{\delta x} \quad (11)$$

- Central (or second order) finite difference:

$$f'(x_0) \approx \frac{f(x_0 + \delta x) - f(x_0 - \delta x)}{2\delta x} \quad (12)$$

Estimating Derivatives with Finite Differences (3/3)

Derivatives of an image: $I : \Omega \subset \mathcal{R}^2 \rightarrow \mathcal{R}$

$$\frac{\partial I}{\partial x} = \nabla_x I = \frac{I(x-1,y) - I(x+1,y)}{2} \quad (13)$$

$$\frac{\partial I}{\partial y} = \nabla_y I = \frac{I(x,y-1) - I(x,y+1)}{2} \quad (14)$$

This can be rewritten as

$$\nabla_x I = I * H_x \quad (15)$$

$$\nabla_y I = I * H_y \quad (16)$$

where $H_x = \left[\frac{1}{2}\ 0\ -\frac{1}{2}\right]$ and $H_y = \left[\frac{1}{2}\ 0\ -\frac{1}{2}\right]^T$ are the convolution kernel.

Image gradient (1/3)

An image gradient is a directional change in the intensity or color in an image.

The gradient of a two-variable function I at each image point is a 2D vector with the components given by the derivatives in the horizontal and vertical directions:

$$\nabla I = \begin{bmatrix} \nabla_x I \\ \nabla_y I \end{bmatrix} \tag{17}$$

- The gradient magnitude: $\|\nabla I\| = \sqrt{\nabla_x I^2 + \nabla_y I^2}$
- The gradient direction: $\theta = atan2(\nabla_y I, \nabla_x I)$

Image gradient (2/3)

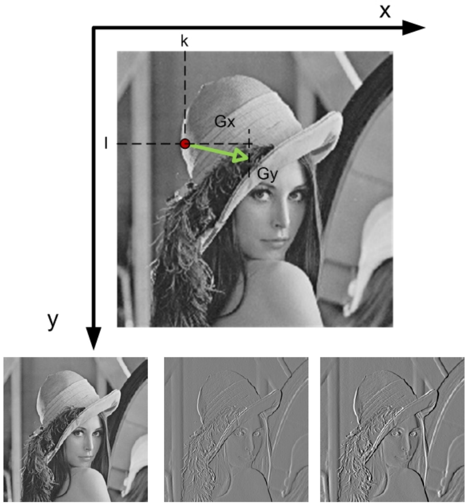


Image gradient (3/3)

Derivatives of an image: $I : \Omega \subset \mathcal{R}^2 \rightarrow \mathcal{R}$

- Finite difference filters respond strongly to noise
- Smoothing and Differentiation: Smoothing an image and then differentiating it is the same as convolving it with the derivative of a smoothing kernel (convolution is associative):

$$\begin{aligned} \nabla_x * (H_x * I) &= (\nabla_x * H_x) * I \\ \nabla_y * (H_y * I) &= (\nabla_y * H_y) * I \end{aligned}$$

$$\nabla_x = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} -1 & 0 & +1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

$$\nabla_y = \begin{bmatrix} -1 \\ 0 \\ +1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

Canny edge detector (1/2)

The Canny Edge detector, developed by J. Canny in 1986, aims to satisfy three main criteria:

- Low error rate: Meaning a good detection of only existent edges;
- Good localization: The distance between edge pixels detected and real edge pixels have to be minimized;
- Minimal response: Only one detector response per edge.

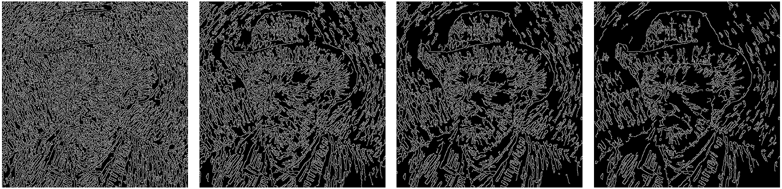
From the image gradient, Canny uses two thresholds (upper and lower):

(a) If a pixel gradient is higher than the upper threshold, the pixel is accepted as an edge.

(b) If a pixel gradient value is below the lower threshold, then it is rejected.

(c) If the pixel gradient is between the two thresholds, then it will be accepted only if it is connected to a pixel that is above the upper threshold.

Canny edge detector (2/2)



Laplacian (1/2)

The gradient operator dot product with the gradient is called the Laplacian. This is defined by:

$$\Delta I = \frac{\delta^2 I}{\delta x^2} + \frac{\delta^2 I}{\delta y^2} \quad (18)$$

This is equivalent to convolve the input image with the following kernel:

$$K = \begin{bmatrix} 0 & 1 & 0 \\ 1 & \mathbf{-4} & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Then, for a particular location (i, j), $\Delta I(i, j)$ is given by $\Delta I(i, j) = -4I(i, j) + I(i+1, j) + I(i-1, j) + I(i, j+1) + I(i, j-1)$.

Laplacian (2/2)

Structure Tensor

Let $I(\mathbf{p})$ be an image realization at a pixel position $\mathbf{p}$.
Let $\nabla I_x(\mathbf{p})$ and $\nabla I_y(\mathbf{p})$ be the horizontal and vertical gradients respectively.
We define the tensor structure at position $\mathbf{p}$ as

$$\mathbf{J}(\mathbf{p}) = \mathbb{E}_{w,\mathbf{p}}\left([\nabla I_x(\mathbf{r}),\ \nabla I_y(\mathbf{r})]^\top [\nabla I_x(\mathbf{r}),\ \nabla I_y(\mathbf{r})]\right)$$

which gives

$$\mathbf{J}(\mathbf{p}) = \begin{pmatrix} \sum_{\mathbf{r}} w(\mathbf{r})\nabla I_x(\mathbf{p}-\mathbf{r})^2 & \sum_{\mathbf{r}} w(\mathbf{r})\nabla I_x(\mathbf{p}-\mathbf{r})\nabla I_y(\mathbf{p}-\mathbf{r}) \\ \sum_{\mathbf{r}} w(\mathbf{r})\nabla I_x(\mathbf{p}-\mathbf{r})\nabla I_y(\mathbf{p}-\mathbf{r}) & \sum_{\mathbf{r}} w(\mathbf{r})\nabla I_y(\mathbf{p}-\mathbf{r})^2 \end{pmatrix}$$

if w comes from G_σ, a Gaussian kernel centered around $\mathbf{p}$, we have

$$\mathbf{J}(\mathbf{p}) = \begin{pmatrix} (G_\sigma * \nabla I_x^2)(\mathbf{p}) & (G_\sigma * \nabla I_x \nabla I_y)(\mathbf{p}) \\ (G_\sigma * \nabla I_x \nabla I_y)(\mathbf{p}) & (G_\sigma * \nabla I_y^2)(\mathbf{p}) \end{pmatrix}$$

Tensor's structure property:

The orientation $\mathbf{n}$ is the solution of the following equation:

$$\mathbf{J}(\mathbf{p})\mathbf{n} = \lambda\mathbf{n}$$

So the eigenvectors of $\mathbf{J}(\mathbf{p})$ are the major orientation at position $\mathbf{p}$ and their corresponding energy is given by the eigenvalues λ_1 and λ_2 (with $\lambda_1 > \lambda_2$).

The major orientation (λ_1) is given by the first eigenvector

$$\mathbf{n} = \begin{pmatrix} J_{2,2}(\mathbf{p}) - J_{1,1}(\mathbf{p}) \\ 2J_{1,2}(\mathbf{p}) \end{pmatrix}$$

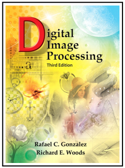
with a level of confidence equal to

$$C = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2} = \frac{(J_{2,2}(\mathbf{p}) - J_{1,1}(\mathbf{p}))^2 + 4J_{1,2}^2}{(J_{1,1}(\mathbf{p}) + J_{2,2}(\mathbf{p}))^2}$$

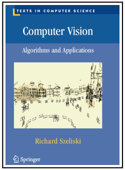
`Bigun, J. (1987). Optimal orientation detection of linear symmetry.`

References

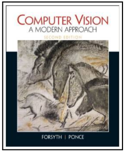
The following slides rely heavily upon the following documents:



Rafael Gonzalez and Richard Woods, *Digital Image Processing*, 3rd edition, Pearson Prentice Hall, 2008.



Richard Szeliski, *Computer Vision: Algorithms and Applications*, 2010.



David Forsyth and Jean Ponce, *Computer Vision: A Modern Approach*, 2003.

References

Pravin Bhat, Brian Curless, Michael Cohen, and C Lawrence Zitnick. Fourier analysis of the 2d screened poisson equation for gradient domain problems. In *European Conference on Computer Vision*, pages 114–128. Springer, 2008.

Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 60–65. IEEE, 2005. 53

Peter J Burt and Edward H Adelson. The laplacian pyramid as a compact image code. *Communications, IEEE Transactions on*, 31 (4):532–540, 1983.

Elmar Eisemann and Frédo Durand. Flash photography enhancement via intrinsic relighting. *ACM transactions on graphics (TOG)*, 23(3):673–678, 2004. 50

David A Forsyth and Jean Ponce. A modern approach. *Computer Vision: A Modern Approach*, 2003.

Jan-Mark Geusebroek, Arnold WM Smeulders, and Joost Van de Weijer. Fast anisotropic gauss filtering. *Image Processing, IEEE Transactions on*, 12(8):938–943, 2003. 18

Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. In *European conference on computer vision*, pages 1–14. Springer, 2010. 48, 57

Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. *IEEE transactions on pattern analysis and machine intelligence*, 35(6):1397–1409, 2013. 48, 57

Ido Omer and Michael Werman. Color lines: Image specific color representation. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–946. IEEE, 2004. 56

Georg Petschnigg, Richard Szeliski, Maneesh Agrawala, Michael Cohen, Hugues Hoppe, and Kentaro Toyama. Digital photography with flash and no-flash image pairs. *ACM transactions on graphics (TOG)*, 23(3):664–672, 2004. 50

Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846. IEEE, 1998. 43

Leonid P Yaroslavsky and LP Yaroslavskij. Digital picture processing. an introduction. *Digital picture processing. An introduction.. LP Yaroslavsky (LP Yaroslavskij). Springer Seriesin Information Sciences, Vol. 9. Springer-Verlag, Berlin-Heidelberg-New York-Tokyo. 12+ 276 pp. Price DM 112.00 (1985). ISBN 3-540-11934-5 (FR Germany), ISBN 0-387-11934-5 (USA).*, 1, 1985. 42