Module 8
Image Analysis I

- **Image Quality Assessment**
- **Template Matching**
- **Edge Detection**

IMAGE QUALITY ASSESSMENT

- What determines the **quality** of an image?

- If the ultimate receiver is the **human eye – subjective judgement** is all that matters.

- But how can this be determined **by an algorithm**? How can it be **quantified?**

Three Types of Image QA

- The search is always on for algorithms that do:

(1) **Full-Reference QA** – Wherein a “distorted” image is compared to an ideal reference image, which is assumed available.

(2) **Reduced-Reference QA** – Wherein a reference is not available, but other information might be, e.g., that the image is JPEG-distorted.

(3) **No-Reference QA** – Wherein there is no reference image, nor other information available. The image is assessed based on its individual appearance. Also called **Blind Image QA.**

No-Reference Image QA

- An **exceedingly difficult problem**. An algorithm must answer "What is the quality of this image" with only the image as input.
- Greatly complicated by issues such as **image content**, **spatial distribution of distortion**, **aesthetics**, etc.
- Almost **no progress** on this problem. Indeed, progress has been slow on all forms of QA - until recently.
- Nevertheless – **humans** can do it quite well, and in fact are the final arbiters. So, there is hope.
- We won't delve into this topic.

Is this image of good quality? Why or why not?

Reduced-Reference Image QA

- The image is not available, but **side information** about it is given.

- If images are sent over a channel, **features** extracted from the image (at the transmitter) are sent on a separate channel - for comparison with the distorted features at the receiver. Edges, selected DCT or wavelet coefficients, etc.

- Or, if it is known that the images are subject to a specific distortion, such as **JPEG**. Some regard this as "blind" but in reality, it is reference information.

Does this image have predictable distortion artifacts?

Full-Reference Image QA

- The **original image** is **known.**
- **Very useful** for **assessing effectiveness** of image processing/communication **algorithms**.
- Lots of **recent progress** in the area.
- **Not useful** for monitoring image system outputs without reference.
- Ideas may form basis for blind QA – in time.

MSE / PSNR

- The **mean-squared error (MSE)** is the long-standing most widely used image QA method.
- Given **original image I** and **observed image J**:

$$\mathrm{MSE}(\mathbf{I}, \mathbf{J}) = \frac{1}{NM} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \left[I(i, j) - J(i, j) \right]^2$$

- The **Peak Signal-to-Noise Ratio (PSNR)** is:

$$\mathrm{PSNR}(\mathbf{I}, \mathbf{J}) = 10 \log_{10} \frac{L^2}{\mathrm{MSE}(\mathbf{I}, \mathbf{J})}$$

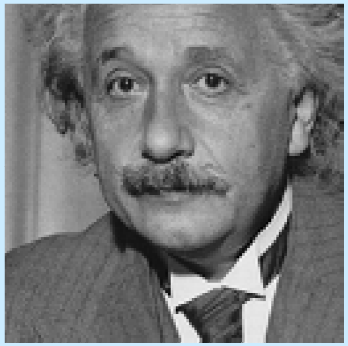
where L is the range of allowable gray scales (255 for 8 bits).

Advantages of MSE/PSNR

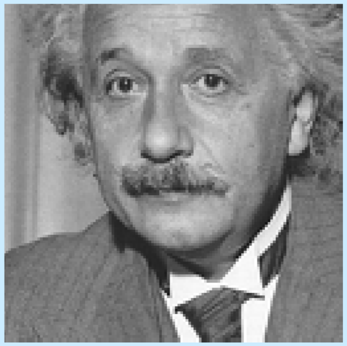
- **Computationally simple.**
- Analytically easy to work with.
- Easy to **optimize algorithms** w.r.t. MSE.
- **Effective** in high SNR situations.

Disadvantages of MSE/PSNR

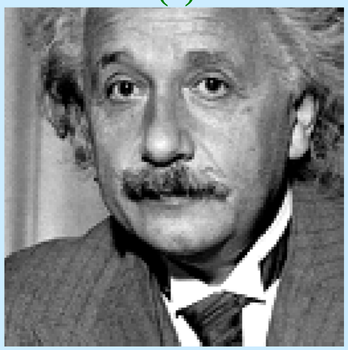
- **Very poor correlation** with human visual perception!



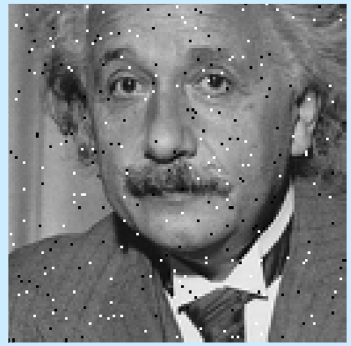
(a)



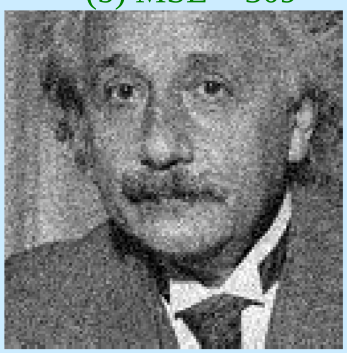
(b) MSE = 309



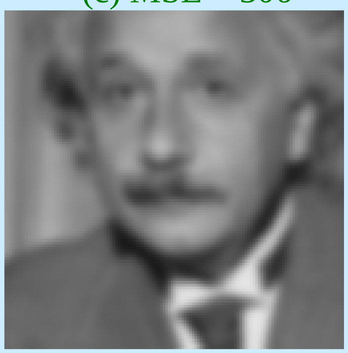
(c) MSE = 306



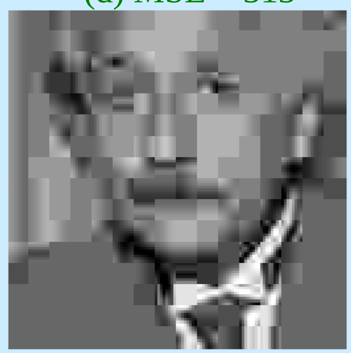
(d) MSE = 313



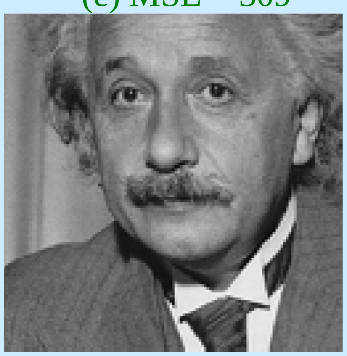
(e) MSE = 309



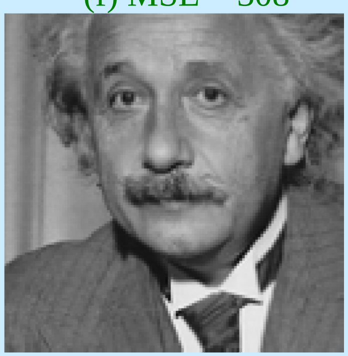
(f) MSE = 308



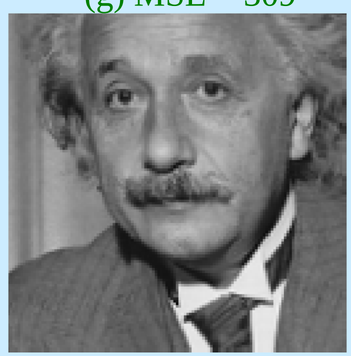
(g) MSE = 309



(h) MSE = 871



(i) MSE = 694



(j) MSE = 590

Einstein w/different distortions:

(a) original image
(b) mean luminance shift
(c) contrast stretch
(d) impulse noise
(e) Gaussian noise
(f) Blur
(g) JPEG compression
(h) spatial shift (to the left)
(i) spatial scaling (zoom out)
(j) rotation (CCW).

Images (b)-(g) have nearly identical MSEs but very different visual quality.

Human Subjectivity

- **Human opinion** is the ultimate gauge of image quality.
- **Measuring true image quality** requires asking many subjects to view images under **calibrated test conditions**. How this is done is beyond this class.
- The resulting **Mean Opinion Scores** (MOS) are then correlated with QA algorithm performance.
- Until recently, MSE had **little competition** despite 40 years of research!

Human Vision Based Metrics

- Several QA metrics have been used which utilize **complex models of human vision.**
- The best example is the Sarnoff Labs **JNDmetrix** which won an Emmy! (JND = “Just Noticeable Difference”)
- Studies showed that this complex algorithm is not much better than MSE – except for QA of **compression artifacts**.

Structural Similarity-Based Metrics

- A more modern, extremely successful approach: Measure **loss** of **structure** in a distorted image.

- The same thing: Measure **retained structure.**

- Basic idea: Combine local measurements of similarity of **luminance**, **contrast**, **structure** into a local measure of quality.

- Perform a **weighted average** of the local measure across the image.

Structural Similarity Index (SSIM)

- Groundbreaking technique that has **revolutionized** image QA.
- The SSIM Index which expresses the similarity of **I** and **J** at a point is

$$\mathrm{SSIM}_{\mathbf{I},\mathbf{J}}(i, j) = L_{\mathbf{I},\mathbf{J}}(i, j) \cdot C_{\mathbf{I},\mathbf{J}}(i, j) \cdot S_{\mathbf{I},\mathbf{J}}(i, j)$$

where

$L_{\mathbf{I},\mathbf{J}}(i, j)$ is a measure of local **luminance similarity**
$C_{\mathbf{I},\mathbf{J}}(i, j)$ is a measure of local **contrast similarity**
$S_{\mathbf{I},\mathbf{J}}(i, j)$ is a measure of local **structure similarity**

Luminance Similarity

- The **luminance similarity term** is

$$L_{\mathbf{I},\mathbf{J}}(i, j) = \frac{2\mu_{\mathbf{I}}(i, j)\mu_{\mathbf{J}}(i, j) + C_1}{\mu_{\mathbf{I}}^2(i, j) + \mu_{\mathbf{J}}^2(i, j) + C_1} = \frac{2\mu_{\mathbf{I}}\mu_{\mathbf{J}} + C_1}{\mu_{\mathbf{I}}^2 + \mu_{\mathbf{J}}^2 + C_1}$$

where $\mu_{\mathbf{I}}(i, j)$, $\mu_{\mathbf{J}}(i, j)$ are **weighted local average intensities** (luminances)

$$\mu_{\mathbf{I}}(i, j) = \sum_{p=-P}^{P} \sum_{q=-Q}^{Q} w(p, q)I(i+p, j+q)$$

$w(p, q)$ is an **isotropic, unit area weighting function** such as a gaussian shape: $\sum_{p=-P}^{P} \sum_{q=-Q}^{Q} w(p, q) = 1$

and C_1 is a stabilizing constant (more later)

Contrast Similarity

- The **contrast similarity term** is

$$C_{\mathbf{I},\mathbf{J}}(i, j) = \frac{2\sigma_{\mathbf{I}}(i, j)\sigma_{\mathbf{J}}(i, j) + C_2}{\sigma_{\mathbf{I}}^2(i, j) + \sigma_{\mathbf{J}}^2(i, j) + C_2} = \frac{2\sigma_{\mathbf{I}}\sigma_{\mathbf{J}} + C_2}{\sigma_{\mathbf{I}}^2 + \sigma_{\mathbf{J}}^2 + C_2}$$

where $\sigma_{\mathbf{I}}(i, j)$, $\sigma_{\mathbf{J}}(i, j)$ are **weighted local standard deviations of intensities** (contrasts)

$$\sigma_{\mathbf{I}}(i, j) = \sqrt{\sum_{p=-P}^{P} \sum_{q=-Q}^{Q} w(p, q)\left[I(i+p, j+q)-\mu_{\mathbf{I}}(i, j)\right]^2}$$

and C_2 is a **stabilizing constant.**

Structural Similarity

- The **structural similarity term** is

$$S_{\mathbf{I},\mathbf{J}}(i, j) = \frac{2\sigma_{\mathbf{IJ}}(i, j) + C_3}{\sigma_{\mathbf{I}}(i, j)\sigma_{\mathbf{J}}(i, j) + C_3} = \frac{2\sigma_{\mathbf{IJ}} + C_3}{\sigma_{\mathbf{I}}\sigma_{\mathbf{J}} + C_3}$$

where $\boldsymbol{\sigma}_{\mathbf{IJ}}$ is a **weighted local correlation of intensities**

$$\sigma_{\mathbf{IJ}}(i, j) = \sum_{p=-P}^{P} \sum_{q=-Q}^{Q} w(p, q)\left[I(i+p, j+q)-\mu_{\mathbf{I}}(i, j)\right]\left[J(i+p, j+q)-\mu_{\mathbf{J}}(i, j)\right]$$

and C_3 is a **stabilizing constant.**

<u>Properties of SSIM</u>

- The SSIM Index satisfies important properties for comparing two images:

(1) Symmetry: $SSIM_{\mathbf{I,J}}(i, j) = SSIM_{\mathbf{J,I}}(i, j)$

(2) Boundedness: $0 \leq SSIM_{\mathbf{I,J}}(i, j) \leq 1$

(3) Unique Maximum: $SSIM_{\mathbf{I,J}}(i, j) = 1$ if and only if the images are locally identical:

$$I(i+p, j+q) = J(i+p, j+q)$$

for $-P \leq p \leq P$, $-Q \leq q \leq Q$.

Stabilizing Constants

- The constants C_1, C_2 and C_3 **stabilize** the three terms – in case the local means or contrasts are **very small.**

- For gray-scale range 0-255, $\mathbf{C_1 = (0.01 \cdot 255)^2}$, $\mathbf{C_2 = (0.03 \cdot 255)^2}$, $\mathbf{C_3 = C_2/2}$ work well and robustly.

- If $C_1 = C_2 = C_3 = 0$, then

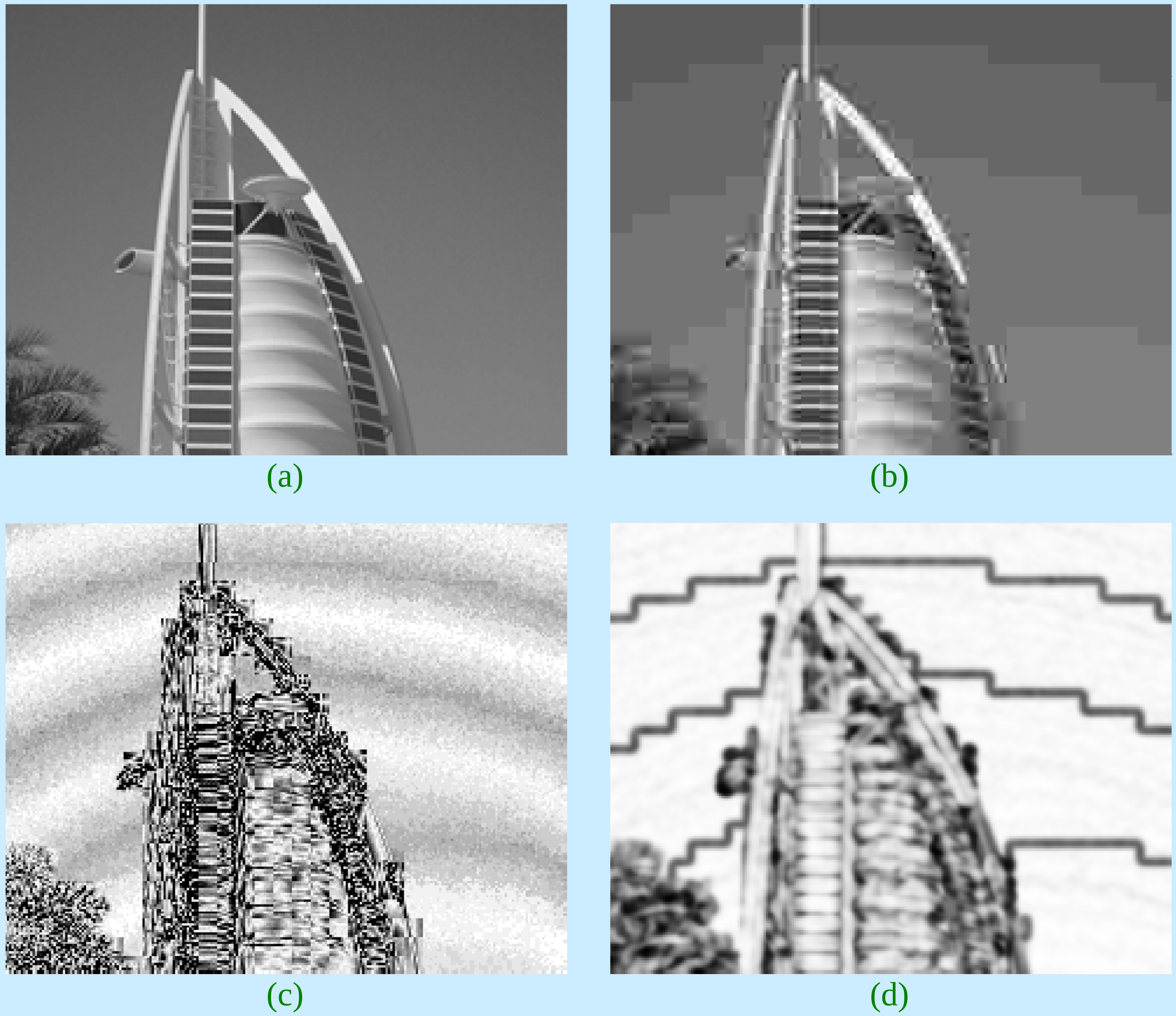
$$\mathrm{SSIM}_{\mathbf{I},\mathbf{J}}(i, j) = \frac{4\mu_{\mathbf{I}}(i, j)\mu_{\mathbf{J}}(i, j)\sigma_{\mathbf{IJ}}(i, j)}{\left[\mu_{\mathbf{I}}^2(i, j) + \mu_{\mathbf{J}}^2(i, j)\right]\left[\sigma_{\mathbf{I}}^2(i, j) + \sigma_{\mathbf{J}}^2(i, j)\right]} = \frac{4\mu_{\mathbf{I}}\mu_{\mathbf{J}}\sigma_{\mathbf{IJ}}}{\left(\mu_{\mathbf{I}}^2 + \mu_{\mathbf{J}}^2\right)\left(\sigma_{\mathbf{I}}^2 + \sigma_{\mathbf{J}}^2\right)}$$

- This original version of SSIM is known as **Universal Quality Index (UQI).**

SSIM Map

- Displaying $SSIM_{\mathbf{I,J}}(i, j)$ as an image is called a SSIM Map. It is an effective way of **visualizing** where the images **I**, **J** differ.

- The SSIM map depicts where the quality of one image is flawed relative to the other.



(a) reference image; (b) JPEG compressed;
(c) absolute difference; (d) SSIM Index Map.

(a)

(b)

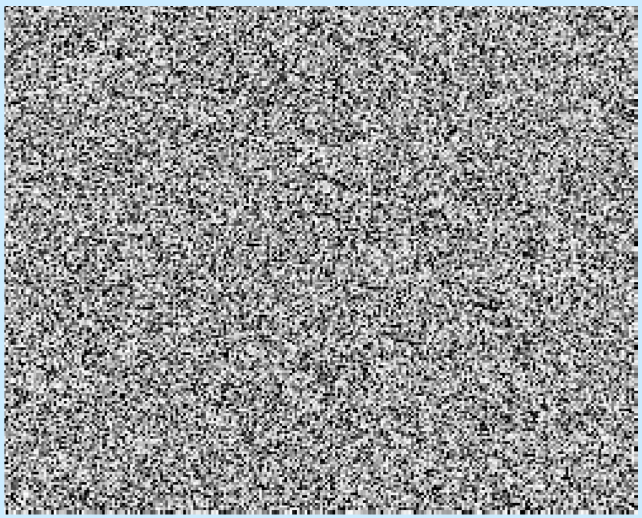
(c)

(d)

(a) reference image; (b) JPEG2000 compressed; (c) absolute difference; (d) SSIM Index Map

(a)

(b)



(c)

(d)

(a) original image; (b) additive white Gaussian noise;
(c) absolute difference; (d) SSIM Index Map.

Mean SSIM

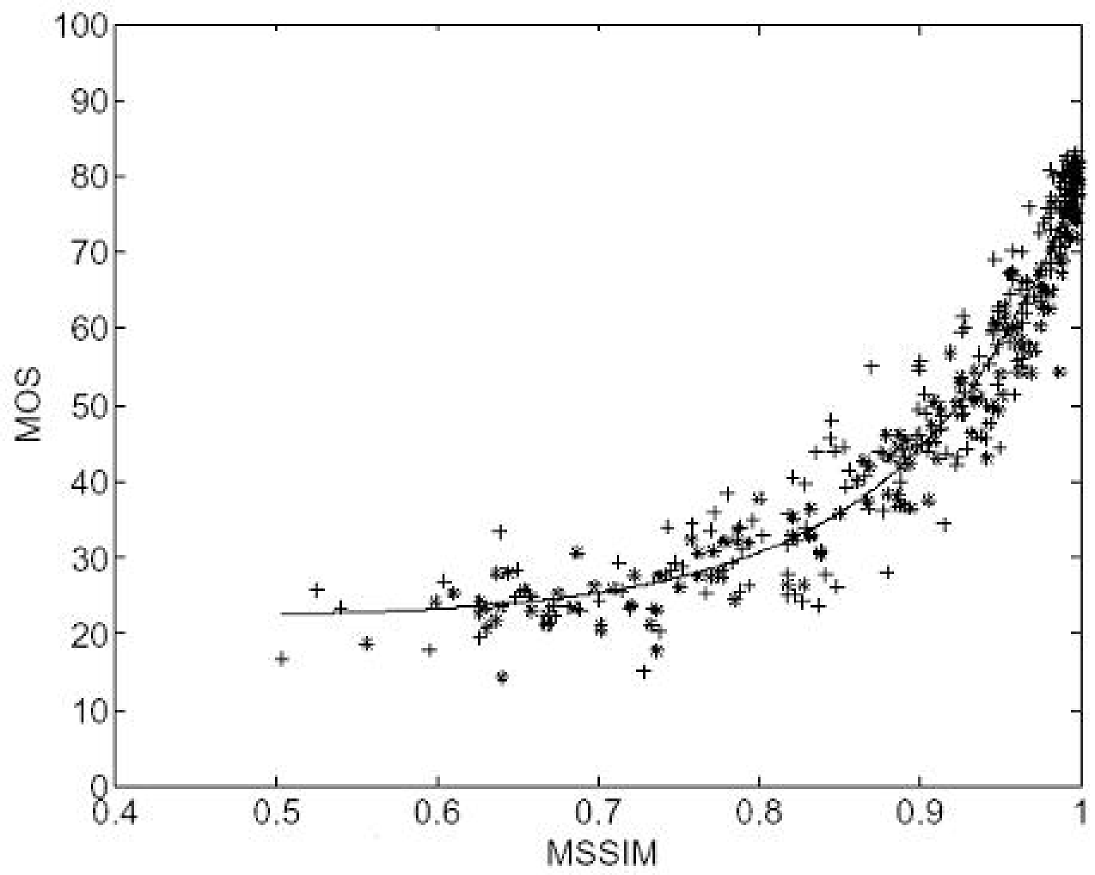
- If $\mathrm{SSIM}_{\mathbf{I},\mathbf{J}}(i, j)$ is averaged over the image, then a single scalar metric is arrived at. The **Mean SSIM** is

$$\mathrm{SSIM}(\mathbf{I},\mathbf{J}) = \left(\frac{1}{NM}\right)\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}\mathrm{SSIM}_{\mathbf{I},\mathbf{J}}(i, j)$$

- The Mean SSIM correlates extraordinarily well with human response as measured in large human studies as Mean Opinion Score (MOS).

SSIM vs. MOS

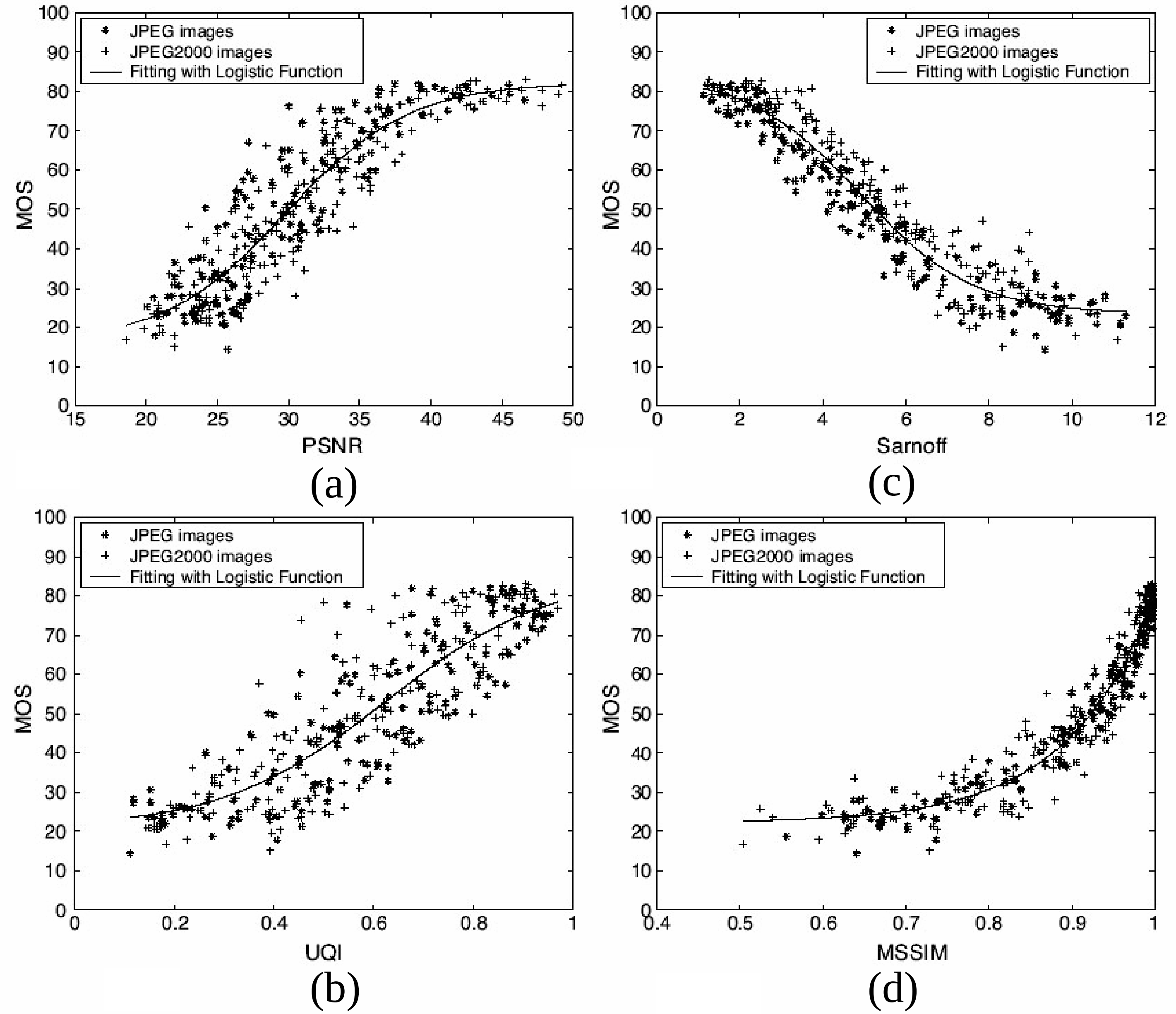
On a broad database of images distorted by jpeg, jpeg2000, white noise, gaussian blur, and fast fading noise.



Curve is best fitting **logistic function** $a\dfrac{1+be^{-t/\tau}}{1+\beta e^{-t/\tau}}$.

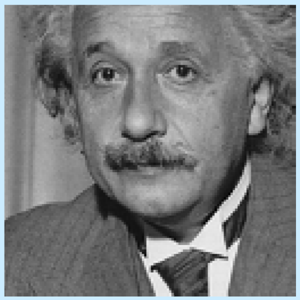
What is important is that the data cluster closely about the curve.

SSIM vs. The Competition

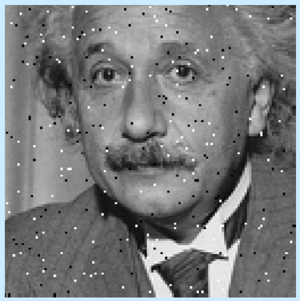


Scatter plots of MOS versus model predictions. Each sample point represents one test image. (a) PSNR prediction; (b) UQI (equivalent to MSSIM with square window and $K_1 = K_2 = 0$) prediction; (c) Sarnoff model prediction; (d) MSSIM (Gaussian window, $K_1 = 0.01, K_2 = 0.03$) prediction.

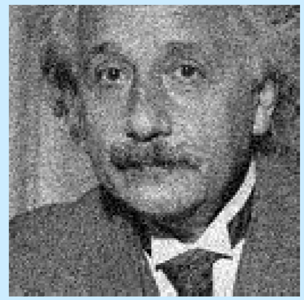
Mean SSIM Examples



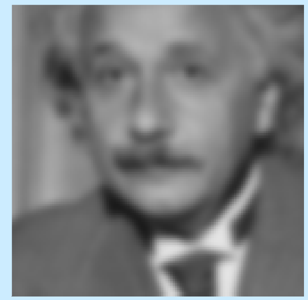
(a)



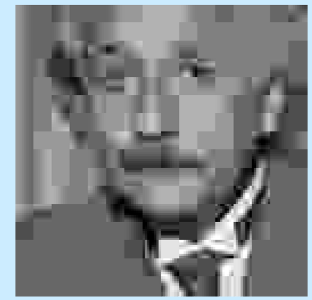
(b) MSE = 313
SSIM = 0.730



(c) MSE = 309
SSIM = 0.576



(d) MSE = 308
SSIM = 0.641



(e) MSE = 309
SSIM = 0.580

Einstein altered by different distortions. (a) reference image;
(b) impulse noise; (c) Gaussian noise;
(d) blur; (e) JPEG compression.

Future Applications of SSIM

- Assessing the quality of **algorithm results**, such as denoising, deblur, compression, enhancement, watermarking, etc. A huge field of applications!

- **Designing algorithms** using SSIM instead of the standard MSE. The hope is that image processing algorithms will perform much better! This is also a huge field!!

TEMPLATE MATCHING

- Consider **finding instances** of a certain **sub-image,** e.g., a certain object, written character, etc.
- This is called **visual search.**
- Visual Search is a **hard problem!**

Where's Waldo?

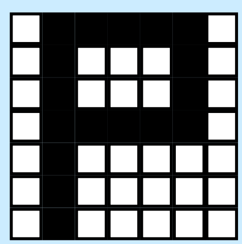
Waldo

Where's Waldo?

Template Matching

- **Template matching** is a window-based technique to accomplish this task.

- A **template** is a **sub-image:**



Template image of ‘P’

Find The Faces

(there are 11 of them!)

Template

- Associated with a **template** $\mathbf{T}$ is a window $\mathbf{B_T}$ which defines its domain:

$$\mathbf{T} = \{T(m, n); (m, n) \in \mathbf{B_T}\}.$$

- We will also use window $\mathbf{B_T}$ to search $\mathbf{I}$.

- As before, the **windowed set** at (i, j) is:

$$\mathbf{B_T} \circ \mathbf{I}(i, j) = \{I(i+m, j+n); (m, n) \in \mathbf{B_T}\}$$

Matching the Template to the Image

- Goal: Measure **goodness-of-match** of template **T** with image patches $\mathbf{B_T} \circ \mathbf{I}(i, j)$ for all (i, j)

- Result: A **match image** **J** taking large values where $\mathbf{B_T} \circ \mathbf{I}(i, j)$ and **T** are highly similar.

- We start with various **mismatch measures** and will derive a **match measure** from one of them.

Mismatch Measures

- **MISMATCH**$\{\mathbf{B_T} \circ \mathbf{I}(i, j), \mathbf{T}\}$

$$(1) = \max_{(m,n) \in \mathbf{B_T}} \left| I(i+m, j+n) - T(m, n) \right| \qquad \text{(max absolute error)}$$

$$(2) = \left(\frac{1}{MN}\right) \sum_{(m,n) \in \mathbf{B_T}} \sum \left| I(i+m, j+n) - T(m, n) \right| \qquad \text{(mean absolute error)}$$

$$(3) = \left(\frac{1}{MN}\right) \sum_{(m,n) \in \mathbf{B_T}} \sum \left| I(i+m, j+n) - T(m, n) \right|^2 \text{(}\textbf{mean-square error}\text{)}$$

- (1)-(3) are reasonable mismatch criteria between $\mathbf{B_T} \circ \mathbf{I}(i, j)$ and **T**. Only (3) leads to an easy **match measure**.

MSE-Based Analysis

- Decompose MSE into **three terms**:

$$\mathrm{MSE}\{\mathbf{B_T} \circ I(i,j), \mathbf{T}\} \propto \sum\sum_{(m,n)\in \mathbf{B}} \left| I(i+m, j+n) - T(m,n) \right|^2$$

$$= \underbrace{\sum\sum_{(m,n)\in \mathbf{B_T}} [I(i+m, j+n)]^2}_{\text{local image energy}} - 2 \underbrace{\sum\sum_{(m,n)\in \mathbf{B_T}} I(i+m, j+n)T(m,n)}_{\text{cross-correlation of I and } \mathbf{T}} + \underbrace{\sum\sum_{(m,n)\in \mathbf{B_T}} [T(m,n)]^2}_{\text{template energy (constant)}}$$

$$= E_{\mathbf{B_T} \circ I(i,j)} - 2C_{\mathbf{B_T} \circ I(i,j), \mathbf{T}} + E_{\mathbf{T}}$$

- MSE is **small** when match is good. Only $C_{\mathbf{B_T} \circ I(i,j), \mathbf{T}}$ contributes to the match measure.

Cross-Correlation Matching

- **Schwarz Inequality**

$$\sum\sum_{(m,n)} A(m,n)B(m,n) \le \sqrt{\sum\sum_{(m,n)} [A(m,n)]^2 \sum\sum_{(m,n)} [B(m,n)]^2}$$

with equality "=" replacing "≤" **if and only if**

$$A(m,n) = K \cdot B(m,n) \text{ for all } (m,n)$$

where K is **any constant**.

Upper Bound on Cross-Correlation

- Upper bound the cross-correlation:

$$C_{\mathbf{B_T}\circ\mathbf{T}(i,\,j),\,\mathbf{T}} = \sum_{(m,n)\in\,\mathbf{B_T}}\sum I(i+m,\,j+n)T(m,\,n)$$

$$\leq \sqrt{\sum_{(m,n)\in\,\mathbf{B_T}}\sum [I(i+m,\,j+n)]^2 \sum_{(m,n)\in\,\mathbf{B_T}}\sum [T(m,\,n)]^2}$$

$$= \sqrt{E_{\mathbf{B_T}\circ I(i,\,j)} \cdot E_{\mathbf{T}}}$$

with **equality if and only if**

$I(i+m,\,j+n) = K\cdot T(m,\,n)$ for all $(m,n) \in \mathbf{B_T}$

Normalized Cross-Correlation

- Cross-correlation is a **good match measure** but we need to compare it with the theoretical upper bound

$$\sqrt{E_{\mathbf{B_T} \circ I(i,\, j)} \cdot E_{\mathbf{T}}}$$

or normalize with respect to it:

Normalized Cross-Correlation

- Define $\hat{C}_{\mathbf{B_T} \circ \mathbf{T}(i,\, j),\, \mathbf{T}} = \dfrac{C_{\mathbf{B_T} \circ I(i,\, j),\, \mathbf{T}}}{\sqrt{E_{\mathbf{B_T} \circ I(i,\, j)} \cdot E_{\mathbf{T}}}}$

  so that $0 \le \hat{C}_{\mathbf{B_T} \circ I(i,\, j),\, \mathbf{T}} \le 1$ for every (i, j).

- Define the **normalized cross-correlation image**:

$$\mathbf{J} = \mathrm{CORR}[\mathbf{I}, \mathbf{T}, \mathbf{B_T}]$$

  if

$$J(i, j) = \hat{C}_{\mathbf{B_T} \circ I(i,\, j),\, \mathbf{T}} \text{ for } 0 \le i \le N\text{-}1,\ 0 \le j \le M\text{-}1$$

Thresholding

- A good way to compute the best match is to find the **largest value** in the match image

$$\mathbf{J} = \mathrm{CORR}[\mathbf{I}, \mathbf{T}, \mathbf{B}]$$

- This assumes there is **exactly one match**. Alternately, **threshold** the match image:

$$K(i, j) = 1 \text{ if } J(i, j) \geq \tau$$

which will (hopefully) identify all points having a sufficiently good match.

Thresholding

- If $\tau = 1$, then only **perfect matches** will be found. These are rarely present due to noise.
- Usually a value τ that is **close to** but **less than** one is used.
- Determination of this value is usually an empirical (trial-and-error) exercise.

Limitations of Template Matching

- Template matching is quite **sensitive to object variations**.

- Noise, rotation, scaling, stretching, occlusion … and many other changes can confound the matching process.

- The following template: 

- Will match quite well to this image…

Limitations of Template Matching

- But not as well to these….

Generalizing Template Matching

- The **template matching** operation is **difficult to generalize** to arbitrary occurrences of the object such as the "problem" instances shown in the preceding.
- Researchers have extended the technique (with considerable ensuing computation) to resolve **some** of the problems.
- However, the technique remains quite **limited**. Certainly it is not the basis for existing complex search process, e.g., in human vision, nor should it be the basis for complex computer-based visual search.
- Later we'll study the **Hough Transform** that is much more generalizable but applies only to finding **specific curves** in images.
- The next topic, **edge detection**, may be viewed as a **specific type** of template matching. The templates in this case are very simple.

<u>EDGE DETECTION</u>

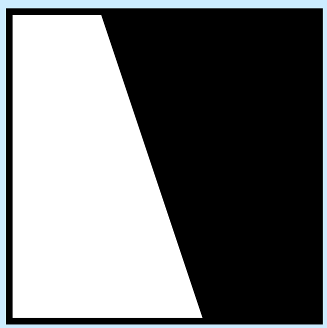
- Probably no topic in the image analysis area has been studied as exhaustively as **edge detection**.
- **Edges** are sudden, sustained changes in **AVERAGE** image intensity that extend along a contour.
- **Edge detection** is used as a precursor to most practical image analysis tasks. Many **computer vision** algorithms use detected edges as fundamental processing **primitives**.
- Some **reasons** for this :

    - **Enormous information** is contained in image edges. An image can be largely recognized from its edges alone.

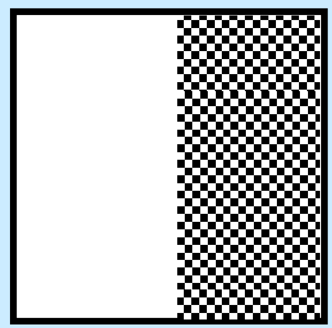
    - An **edge map** requires **much less** storage space than the image itself. It is a **binary** contour plot.

What is an Edge?

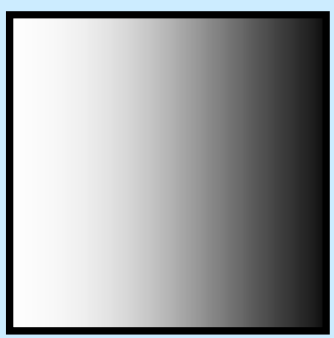
- Where are the "edges" in each of the following images:



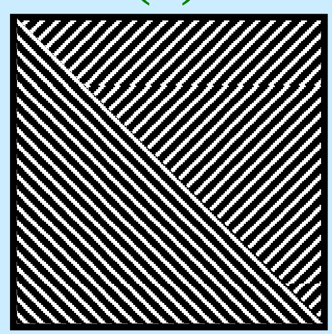
(1)



(2)



(3)



(4)

Overview of Edge Detection Methods

- Methods for **edge detection** have been studied since the mid-1960's.
- Easily been the most studied problem in the image analysis area.
- **Hundreds** of different approaches devised - based on just about any **math technique** imaginable for deciding whether one group of numbers is larger than another.
- The problem became (at last) fairly well-understood in the 1980's.
- We will study **three fundamental approaches**:
  - **Gradient** edge detectors
  - **Laplacian** edge detectors (multi-scale)
  - **Diffusion-based** edge detectors

GRADIENT EDGE DETECTORS

- Oldest (Roberts 1965) but still valuable class of edge detectors.

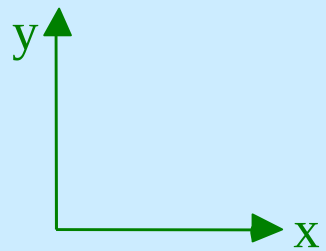
- For a continuous 2-D function f(x, y), the **gradient** is a two-element **vector**:

$$\nabla f(x, y) = [f_x(x, y), f_y(x, y)]^T$$

where

$$f_x(x, y) = \frac{\partial}{\partial x} f(x,y) \quad \text{and} \quad f_y(x, y) = \frac{\partial}{\partial y} f(x,y)$$

are the **directional derivatives** of f(x, y) in the x- and y-directions:



Gradient Measurements

- **Fact**: the **direction** of the **fastest rate of change** of f(x, y) at a point (x, y) is the **gradient orientation**

$$\theta_f(x, y) = \tan^{-1}\left(\frac{f_y(x, y)}{f_x(x, y)}\right)$$

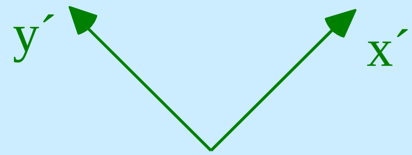
- and that rate of change is the **gradient magnitude**

$$M_f(x, y) = \sqrt{f_x^2(x, y) + f_y^2(x, y)}$$

- The gradient is appealing for defining an edge detector, since we expect an edge to locally exhibit the greatest **rate of change in image intensity**.

Isotropic Gradient Magnitude

- **Fact**: Take directional derivatives in **any two perpendicular directions**, (say x´ and y´) and $M_f(x, y)$ remains **unchanged:**

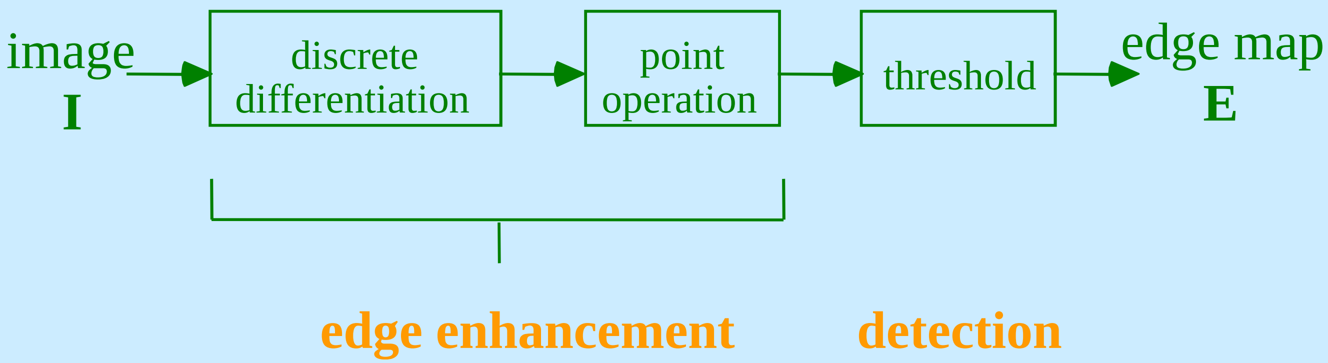


- $M_f(x, y)$ is **rotationally symmetric** or **isotropic.**

- **Isotropicity** is **desirable** for edge detection –detect edges equally regardless of orientation.

Gradient Edge Detector Diagram

- Described by the following flow diagram:



- **digital differentiation**: digitally approximate ∇I

- **point operation**: estimate $M_I = |\nabla I|$

- **threshold**: decide which large values of M_I are likely edge locations

Digital Differentiation

- Digital differentiation is **differencing**. For a 1-D function f(x) which has been sampled to produce f(i), either:

$$\left.\frac{d}{dx} f(x)\right|_{x=i} \mapsto f(i) - f(i-1)$$

  or

$$\left.\frac{d}{dx} f(x)\right|_{x=i} \mapsto \frac{f(i+1) - f(i-1)}{2}$$

- The advantage of the first is that it takes the current value f(i) into the computation; the advantage of the second is that it is **centered**.

2-D Differencing

- In **two dimensions,** the extensions are easy:

$$\frac{\partial}{\partial x} f(x, y)\Big|_{(x,y)=(i,j)} \mapsto f(i, j) - f(i\text{-}1, j)$$

$$\frac{\partial}{\partial y} f(x, y)\Big|_{(x,y)=(i,j)} \mapsto f(i, j) - f(i, j\text{-}1)$$

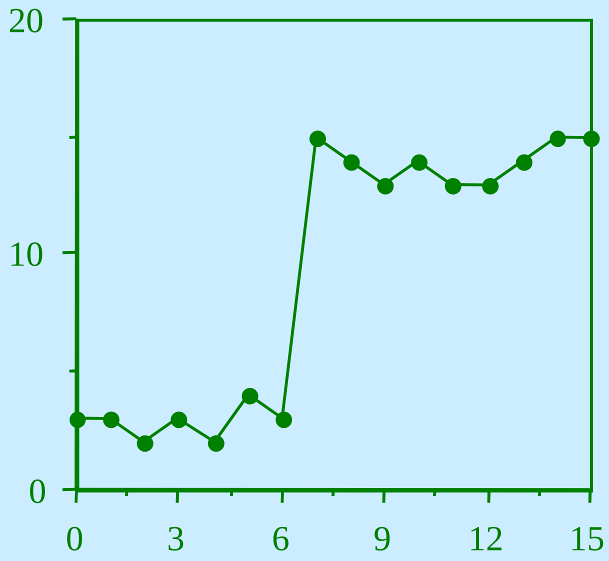
and

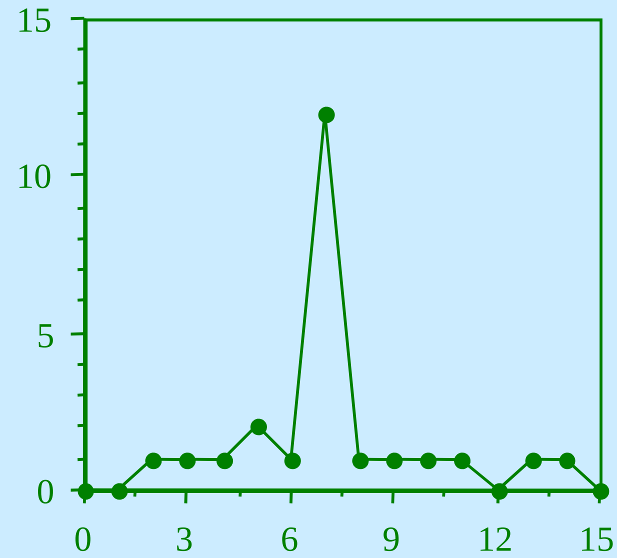
$$\frac{\partial}{\partial x} f(x, y)\Big|_{(x,y)=(i,j)} \mapsto \frac{f(i+1, j) - f(i\text{-}1, j)}{2}$$

$$\frac{\partial}{\partial y} f(x, y)\Big|_{(x,y)=(i,j)} \mapsto \frac{f(i, j+1) - f(i, j\text{-}1)}{2}$$

Example: 1-D Differencing

- The signal I(i) could be a scan line of an image. Let $D_I(i) = |I(i)-I(i-1)|$.

$I(i) =$ 

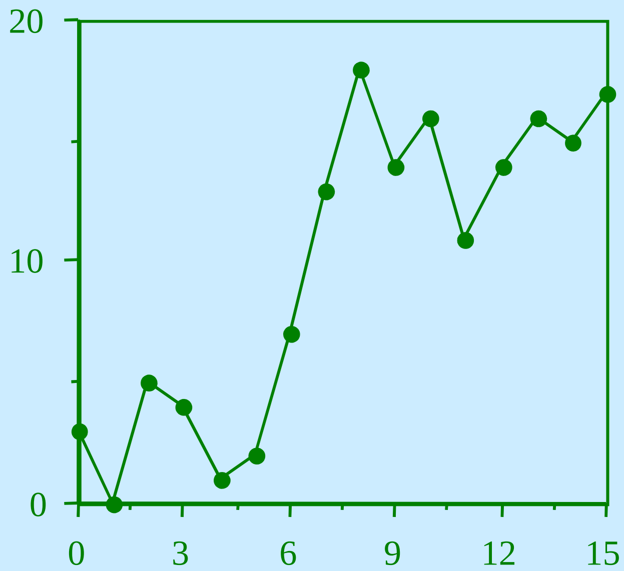
$D_I(i) =$ 

- A clear, **easily thresholded peak**!

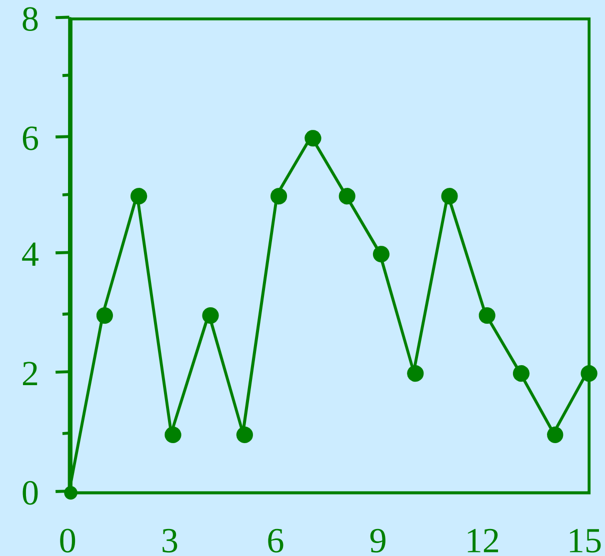
Example: 1-D Differencing With Noise

- The signal J(i) = I(i) + N(i) may be a **noisy** image scan line. Let $D_J(i) = |J(i)-J(i-1)|$.

J(i) =



$D_J(i)$ =



- **Noise** is a **huge problem** for this type of edge detector. Differentiation **always** emphasizes high frequencies (such as noise)

Types of Gradient Edge Detectors

- Define **convolution edge templates** Δ_x and Δ_y which produce directional derivative estimates:

$$\Delta_x = \begin{array}{|c|c|}\hline -1 & +1 \\ \hline\end{array} \qquad \Delta_y = \begin{array}{|c|}\hline -1 \\ \hline +1 \\ \hline\end{array}$$

adjacent

$$\Delta_x = \begin{array}{|c|c|c|}\hline -1 & 0 & +1 \\ \hline\end{array}\,/\,2 \qquad \Delta_y = \begin{array}{|c|}\hline -1 \\ \hline 0 \\ \hline +1 \\ \hline\end{array}\,/\,2$$

centered

$$\Delta_x = \begin{array}{|c|c|}\hline -1 & 0 \\ \hline 0 & +1 \\ \hline\end{array} \qquad \Delta_y = \begin{array}{|c|c|}\hline 0 & -1 \\ \hline +1 & 0 \\ \hline\end{array}$$

Roberts'

- The performance of these three is very **similar**.

Noise-Reducing Variations

- Designed to reduce noise effects by averaging along columns and rows:

$$\Delta_x = \begin{array}{|c|c|c|}\hline -1 & 0 & +1 \\ \hline -1 & 0 & +1 \\ \hline -1 & 0 & +1 \\ \hline \end{array} /3 \qquad \Delta_y = \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline 0 & 0 & 0 \\ \hline +1 & +1 & +1 \\ \hline \end{array} /3 \quad \text{Prewitt}$$

$$\Delta_x = \begin{array}{|c|c|c|}\hline -1 & 0 & +1 \\ \hline -2 & 0 & +2 \\ \hline -1 & 0 & +1 \\ \hline \end{array} /4 \qquad \Delta_y = \begin{array}{|c|c|c|}\hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline +1 & +2 & +1 \\ \hline \end{array} /4 \quad \text{Sobel}$$

- These also perform similarly.

Gradient Magnitude

- The **point operation** combines the directional derivative estimates Δ_x and Δ_y into a single estimate of the **gradient magnitude**.
- The usual estimates:

$$\text{(A)} \quad M(i, j) = \sqrt{\Delta_x^2(i, j) + \Delta_y^2(i, j)}$$

$$\text{(B)} \quad M(i, j) = |\Delta_x(i, j)| + |\Delta_y(i, j)|$$

$$\text{(C)} \quad M(i, j) = \max\left\{ |\Delta_x(i, j)|,\ |\Delta_y(i, j)| \right\}$$

- The following always hold:

$$C \leq A \leq B$$

- (A) is the **correct** interpretation, but (B) and (C) are **cheaper** - no square or square root operations.
- (B) often overestimates edge magnitude of an edge, while (C) often underestimates edge magnitude.

The TI Gradient Magnitude

- Better than (B) or (C) is

$$\text{(D)}\quad M(i, j) = \max\left\{\left|\Delta_x(i, j)\right|, \left|\Delta_y(i, j)\right|\right\} + \frac{1}{4}\min\left\{\left|\Delta_x(i, j)\right|, \left|\Delta_y(i, j)\right|\right\}$$

- Still, the differences between (A)-(D) **are slight**.

Thresholding the Gradient Magnitude

- Once the estimate M(i, j) is obtained, it is **thresholded** to find plausible edge locations.

- This produces the binary **edge map E**:

$$E(i, j) = \begin{cases} 1 \;;\; M(i, j) > \tau \\ 0 \;;\; M(i, j) \leq \tau \end{cases}$$

- Thus:

  - a value '1' indicates the **presence** of an edge at (i, j)
  - a value value '0' indicates the **absence** of an edge at (i, j)

- The **threshold τ** constrains the **sharpness** and **magnitude** of the edges that are detected.

Gradient Edge Detector Advantages

- Simple, computationally efficient
- Natural definition
- Work well on "clean" images

<u>Gradient Edge Detector Disadvantages</u>

- Extremely **noise-sensitive**
- Requires a **threshold** - difficult to select - usually requires **interactive selection** for best results.
- Gradient magnitude estimate will often **fall above threshold** over a few pixels distance from the true edge. So, the detected edges are often **a few pixels wide**.
- This usually requires some kind of **"edge thinning"** operation - usually heuristic.
- The edge contours are often **broken** - gaps appear. This requires some kind of **"edge linking"** operation - usually heuristic.

LAPLACIAN EDGE DETECTORS

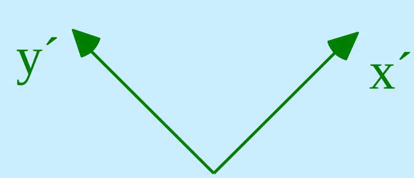
- Edge detectors are based on **second derivatives**.

- For a continuous 2-D function f(x, y), the **Laplacian** is defined:

$$\nabla^2 f(x, y) = \frac{\partial^2}{\partial x^2} f(x, y) + \frac{\partial^2}{\partial y^2} f(x, y)$$

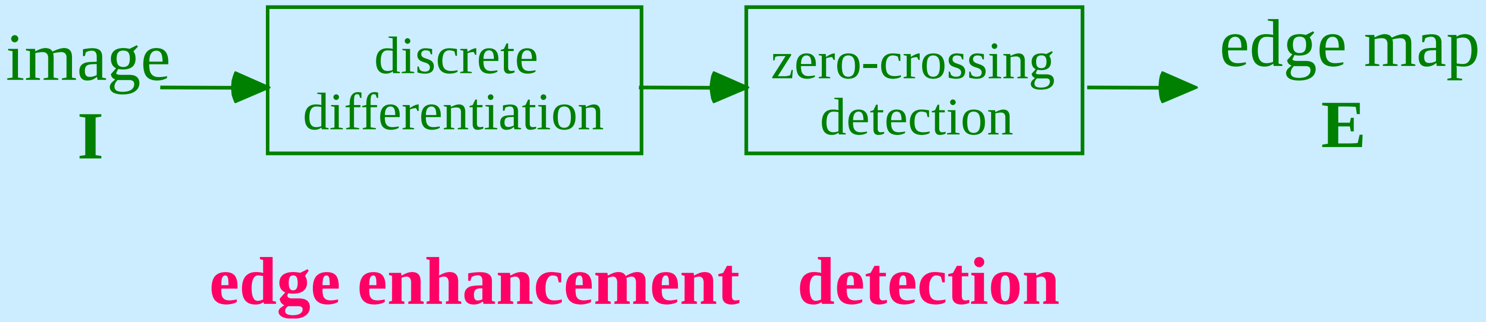
- It is a **scalar** not a vector.

- **Fact**: If the directional derivatives are taken in **any other two perpendicular directions** (say x´, y´) the value of the Laplacian $\nabla^2 f(x, y)$ remains **unchanged**:



Laplacian Edge Detector Diagram

- The **Laplacian edge detector** is described by the following flow diagram:



- **Digital differentiation**: Digitally approximate $\nabla^2 I$

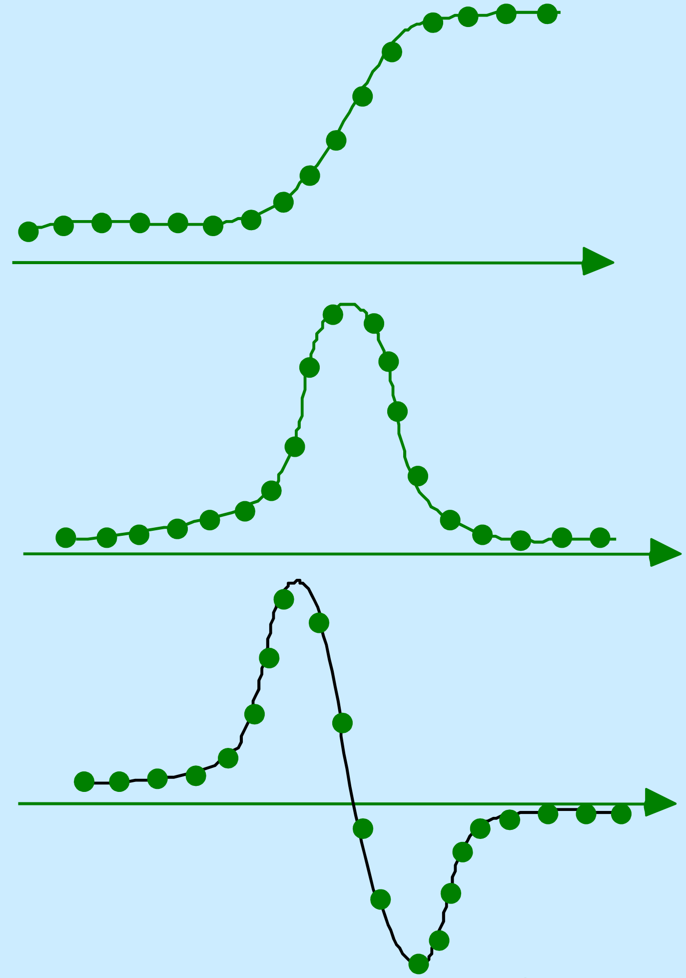
- **Zero-crossing detection**: Discover where the Laplacian **crosses** the zero level.

Reasoning Behind Laplacian

1-D Edge Profile:

Differentiated Once:

Differentiated Again:



- A **zero crossing** or **ZC** occurs near the center of the edge where the **slope of the slope changes sign**.

Digital Twice-Difference

- For a 1-D function f(x) → f(i), we use:

$$\left.\frac{d}{dx} f(x)\right|_{x=i} \mapsto f(i) - f(i-1) = y(i)$$

and then

$$\left.\frac{d^2}{dx^2} f(x)\right|_{x=i} \mapsto y(i+1) - y(i) = f(i+1) - 2f(i) + f(i-1)$$

with **convolution template**:

+1	−2	+1

Digital Laplacian

- In two dimensions:

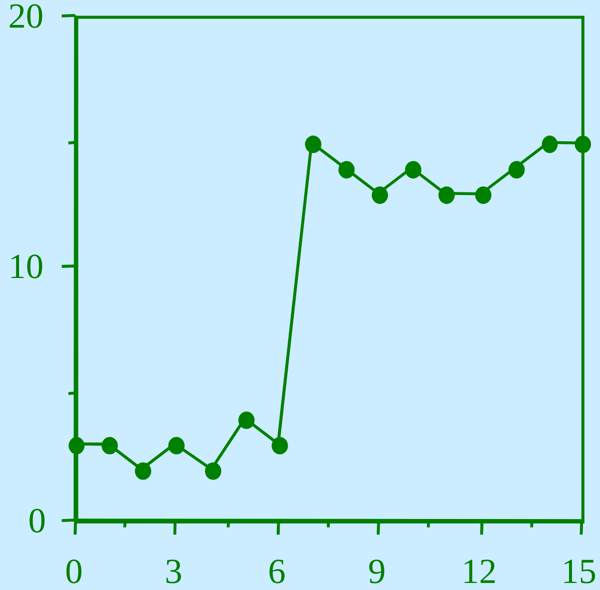
$$\nabla^2 I(x, y) \approx [I(i+1, j) - 2I(i, j) + I(i-1, j)] + [I(i, j+1) - 2I(i, j) + I(i, j-1)]$$

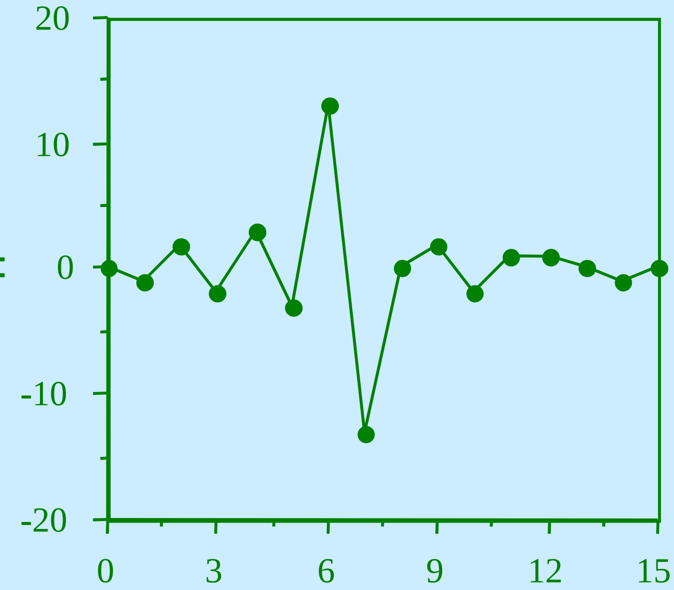
with **convolution template**:

$$\begin{array}{|c|c|c|}\hline +1 & -2 & +1 \\ \hline \end{array} + \begin{array}{|c|}\hline +1 \\ \hline -2 \\ \hline +1 \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline 0 & +1 & 0 \\ \hline +1 & -4 & +1 \\ \hline 0 & +1 & 0 \\ \hline \end{array}$$

Example: Twice-Differentiation

- Let I(i) be image scan line, $\nabla_I(i) = I(i+1) - 2I(i) + 2I(i-1)$

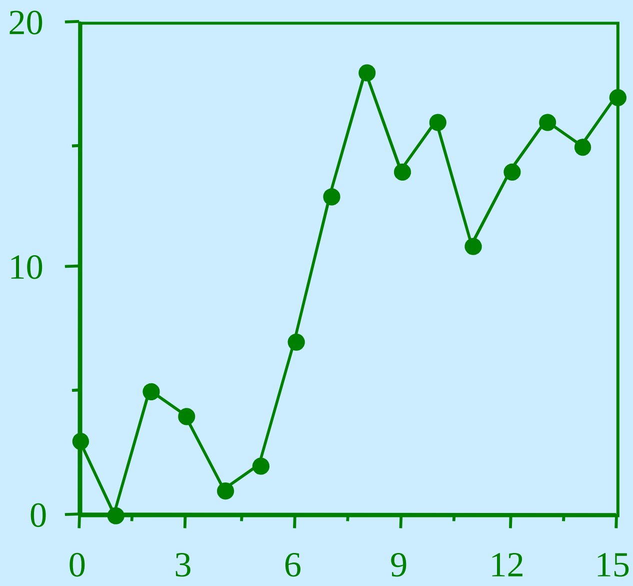
I(i) = 

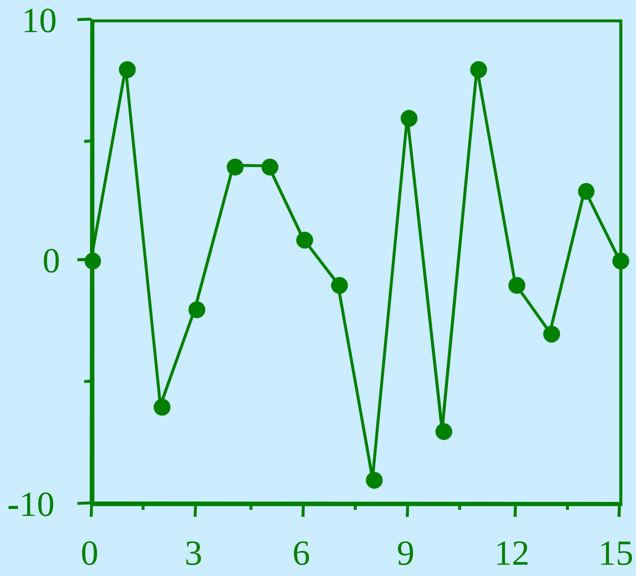
$\nabla_I(i) =$ 

- Clearly reveals a sharp edge location: a single **large-slope ZC** and several "smaller" zero-crossings.

Example: Twice-Differentiation in Noise

- Scan line J(i) = I(i) + N(i) of a **noisy** image:

I(i) = 

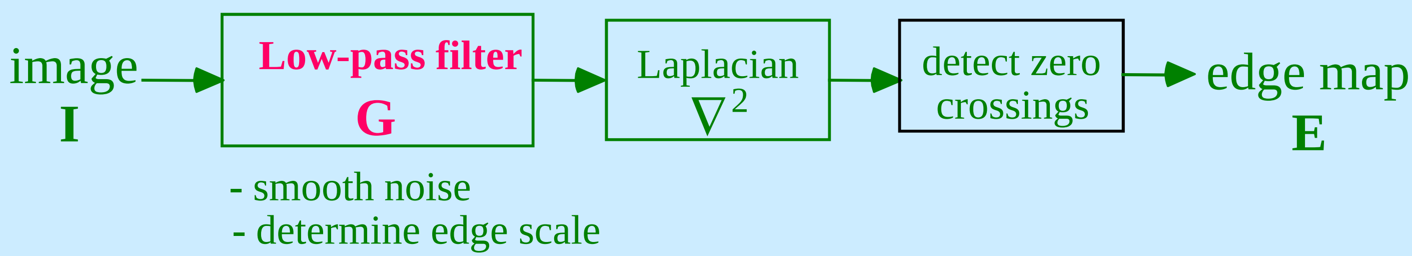
$\nabla_I(i) =$ 

- Numerous **spurious** ZCs.

- **Noise** is an **even bigger problem** for this type of edge detector. Differentiating twice creates **highly amplified noise**.

Smoothed and Multi-Scale Laplacian Edge Detectors

- Laplacian too noise-sensitive to be practical - there is always noise.
- But modifying it in a simple way it can be made very powerful.
- The basic idea is encapsulated:



- The difference: a **linear blur** (low-pass filter) is applied prior to application of the Laplacian.

Low-Pass Pre-filter

- The **main purpose** of a low-pass pre-filter to the Laplacian is to **attenuate (smooth) high-frequency noise** while retaining the significant image structure.

- The **secondary purpose** of the smoothing filter is to **constrain the scale** over which edges are detected.

- Note: a **high-pass operation** (such as Laplacian) followed (or preceded) by a **low-pass** filter will yield a **band-pass filter,** if their passbands overlap.

Gaussian Pre-filter

- A lot of research has been done on how the filter **G** should be selected.
- It has been found that the optimal smoothing filter in the following two simultaneous senses:

  (*i*) best **edge location accuracy**

  (*ii*) **maximum** signal-to-noise ratio (SNR)

  is a **Gaussian** filter: (K is an irrelevant constant)

$$G(i, j) = K \cdot \exp\left\{-\left[i^2 + j^2\right] / 2\sigma^2\right\}$$

Laplacian-of-Gaussian Edge Detector

- Define the **Laplacian-of-a-Gaussian** or **LoG** on **I**:

$$\begin{aligned} J(i, j) &= \nabla^2[G(i, j)*I(i, j)] \\ &= G(i, j)*\nabla^2 I(i, j) \\ &= \nabla^2 G(i, j)*I(i, j) \end{aligned}$$

- Above 3 forms equivalent since **linear operations** (differentiation and convolution) **commute**.

- Best approach: pre-compute the **LoG**:

$$\nabla^2 G(i, j) \propto \left[1 - \frac{i^2 + j^2}{\sigma^2}\right] \cdot \exp\left[-\frac{i^2 + j^2}{2\sigma^2}\right]$$

and convolve image **I** with it. (Constant multiplier omitted since only ZCs are of interest).

Polar Form of LoG

- The LoG is **isotropic** and can be written in polar form:

$$\nabla^2 G(r) \propto \left(1 - \frac{r^2}{\sigma^2}\right) \cdot \exp\left(-\frac{r^2}{2\sigma^2}\right)$$

LoG in space and frequency

ZC Detection

- The last stage of edge detection is **zero-crossing detection**.
- Let **J** = [J(i, j)] be the result of LoG filtering.
- A ZC is a **crossing** of the zero level: the algorithm must search for pixel occurrences of the form:

$$\begin{array}{|c|c|}\hline + & - \\ \hline\end{array} \quad \begin{array}{|c|}\hline + \\ \hline - \\ \hline\end{array} \quad \begin{array}{|c|c|c|}\hline + & 0 & - \\ \hline\end{array} \quad \begin{array}{|c|}\hline + \\ \hline 0 \\ \hline - \\ \hline\end{array}$$

$$\begin{array}{|c|c|}\hline - & + \\ \hline\end{array} \quad \begin{array}{|c|}\hline - \\ \hline + \\ \hline\end{array} \quad \begin{array}{|c|c|c|}\hline - & 0 & + \\ \hline\end{array} \quad \begin{array}{|c|}\hline - \\ \hline 0 \\ \hline + \\ \hline\end{array}$$

- By a convention one sign or the other is **marked** as the edge location unless higher (sub-pixel) precision is needed.

Scale of LoG

- The larger the value of σ used, the greater the degree of **smoothing** by the low-pass pre-filter **G**.
- If σ is **large**, then noise will be greatly smoothed - but so will **less significant edges**.
- **Noise sensitivity increases** with decreases in σ - but the LoG edge detector then detects more detail.


ACTON TOWN

Digital Implementation of LoG

- Use the **sampled LoG**:

$$\nabla^2 G(i, j) \propto \left[1 - \frac{i^2 + j^2}{\sigma^2}\right] \cdot \exp\left[-\frac{i^2 + j^2}{2\sigma^2}\right]$$

- There are specific rules of thumb that should be followed:

- **Enough** of $\nabla^2 G(i, j)$ must be sampled. The LoG will not work unless the template contains both **main** and **minor lobes**. In practice, the **radius** R of the LoG (in space) should satisfy

$$\mathbf{R \geq 4\sigma \text{ (in pixels)}}$$

- Once a LoG template is computed, its coefficients must be slightly adjusted to **sum to zero** (why?)

- This is done by **subtracting** the (average) total coefficient sum from each.

- The LoG will not work well unless $\sigma \geq 1$ (pixel)!

Thresholding the ZCs

- **Thresholding** not usually necessary if a **sufficiently large operator (σ)** is used.
- However, sometimes it is desired to both **detect detail** and **not detect noise**.
- This can be accomplished with effectively by **thresholding**.
- Let J(i, j) be the LoG-filtered image and E(i, j) be the **edge map**.
- Then find the gradient magnitude $|\nabla J(i, j)|$ (Roberts' form will suffice since **J** is smooth).
- If

$$|\nabla J(i, j)| > t = \text{threshold}$$

and

$$E(i, j) = 1 \text{ (a ZC exists at (i, j))}$$

then leave the ZC. Otherwise **delete it**.

Contour Thresholding

- **Problem**: Simple thresholding may create broken ZC contours.
- **Approach**: Compare all ZC pixels on a ZC contour to a threshold **t**. If enough are above threshold, accept the contour, else reject it.
- Suppose $(i_1, j_1), (i_2, j_2), (i_3, j_3), \ldots, (i_L, j_L)$ comprise an 8-connected ZC contour.
- Compute $|\nabla J(i_n, j_n)|$ for $n = 1, \ldots, L$.
- Let Q = # points such that $|\nabla J(i_n, j_n)| > t$.
- If

  $Q/L >$ PERCENT then **accept** the entire ZC contour

  $Q/L \leq$ PERCENT then **reject** the entire ZC contour
- Typically, PERCENT > 0.75

Advantages of the LoG

- Usually **doesn't require thresholding.**
- Yields **single-pixel-width edges always** (no thinning!).
- Yields **connected edges always** (no edge-linking!).
- Can be shown to be **optimal** (under some criteria).
- Appears to be very similar to what goes on in **biological vision**.

Disadvantages of the LoG

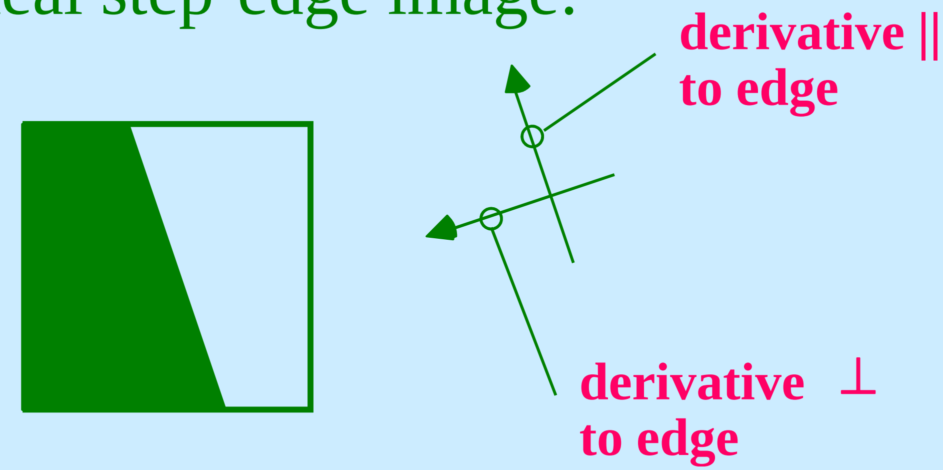
- **More computation** than gradient edge detectors.

- ZCs continuity property often leads to ZCs that **meander across the image**.

- ZC contours tend to be **over-smooth near corners**.

Canny's Edge Detector

- Attempts to **improve upon** the **LoG**.
- Consider an ideal step-edge image:



- Since ∇^2 is **isotropic**, it is equivalent to taking:

  - A twice-derivative **perpendicular** to the edge. This conveys the edge information.

  - A twice-derivative **parallel** to the edge. This conveys **no edge information**!

- In fact, if there is **noise** in the image, the **parallel** twice-derivative will give **only bad information**!

Canny's Algorithm

- Follows the following **basic steps**:

(1) Form the **Gaussian-smoothed** image:

$$K(i, j) = G(i, j) * I(i, j)$$

(2) Compute the **gradient** magnitude and orientation:

$$|\nabla K(i, j)| \quad \text{and} \quad \angle \nabla K(i, j)$$

using discrete (differencing) approximation.

Canny's Algorithm

- (3) Let **n** be the unit vector in the direction $\angle\nabla K(i, j)$. Compute the **twice-derivative** of K(i, j) in the direction **n**:

$$\frac{\partial^2}{\partial \mathbf{n}^2} K(i, j)$$

  Note that this will **not** contain extraneous information.

  (4) **Find the ZC's** in the image.

  (5) These are identically the zero-crossings of: $\nabla K \times \nabla(\nabla K \times \nabla K)$, which is easier to implement.

- **Disadvantage**: nonlinear, so edge continuity **not guaranteed**. However, **contour thresholding** can improve performance.

Canny DEMO

ANISOTROPIC DIFFUSION

- A powerful approach to both image denoising and edge detection.
- Fact: Gaussian blur or smoothing is analogous to physical **diffusion** (e.g., of heat).
- In fact the gaussian function is a solution to a PDE called the **diffusion equation**.

Concept of Anisotropic Diffusion

- Let diffusion (smoothing of image pixels) occur where the local image gradient magnitude is small.
- **Inhibit diffusion** where gradient magnitude is high.
- Called **anisotropic** since smoothing proceeds differently in different directions.
- Convolution with a Gaussian filter is a type of **isotropic diffusion**.
- Rationale:

  - Smooth image without destroying image structure & detail

  - Inhibit smoothing across edges (inter-region smoothing)

  - Encourage smoothing within boundaries (edges) (intra-region smoothing)

The Diffusion Equation

- Continuous Formulation - a system of partial differential equations:

$$\frac{\partial \mathbf{I}}{\partial t} = \operatorname{div}[\mathbf{c}\nabla\mathbf{I}]$$

where **I** is the image and **c** are the **diffusion coefficients,** and div is the **divergence operator**

- The discrete formulation is much easier to understand...

Note: Recall $\operatorname{div}(\mathbf{F}) = \frac{\delta \mathbf{F}}{\delta x} + \frac{\delta \mathbf{F}}{\delta y}$

Diffusion Equation: Discrete Form

- Iterative approximation using **differences**:

$$I_{t+1}(i,j) = I_t(i,j) + \frac{1}{4}[c_N(i,j)\nabla_N I_t(i,j) + c_S(i,j)\nabla_S I_t(i,j)$$
$$+ c_E(i,j)\nabla_E I_t(i,j) + c_W(i,j)\nabla_W I_t(i,j)]$$

where

$$\nabla_N I_t(i, j) = I(i, j+1) - I(i, j)$$
$$\nabla_S I_t(i, j) = I(i, j-1) - I(i, j)$$
$$\nabla_E I_t(i, j) = I(i+1, j) - I(i, j)$$
$$\nabla_W I_t(i, j) = I(i-1, j) - I(i, j)$$

Diffusion Coefficients

- Selection of the diffusion coefficients c_N, c_S, c_E, c_W is important.
- They control the inhibition of smoothing over edges and encourage smoothing in non-edge image regions.
- At each pixel I(i, j), diffusion proceeds in a given direction according to the diffusion coefficient corresponding to that direction and that location.

Diffusion Coefficients

- One popular form

$$c_d(i, j) = \exp\left\{ -\left[\frac{\nabla_d I(i, j)}{k} \right]^2 \right\} \qquad d = N, S, E, W$$

where k is an edge threshold – as the gradient increases, c_d rapdily falls.

- Properties of $c_d(I, j)$:
  - approaches 0 near edges – **disenabling diffusion**
  - approaches 1 over smooth areas –**enabling diffusion**

Anisotropic Diffusion Example

Laplacian Noise

8 iterations of
anisotropic diffusion

Anisotropic Diffusion Demo

Comments on Anisotropic Diffusion

- Really a method of alternating edge detection with **edge-decoupled smoothing.**
- Number of iterations controls degree of smoothing (scale)
- Edge detection applied to the smoothed image can be quite effective.

Too much anisotropic diffusion

Comments

- We will continue studying **image analysis but at a slightly higher level** … onward to **Module 9**.