Digital Image Processing

Ming Jiang

School of Mathematical Sciences
Peking University

ming-jiang@pku.edu.cn

May 19, 2010

Pre-processing

- Pre-processing is a common name for operations with images at the lowest level of abstraction — both input and output are intensity images.
- These iconic images are of the same kind as the original data captured, with an intensity image usually represented by a matrix of image function values (brightness).
- The aim of pre-processing is an improvement of the image data that suppresses unwanted distortions or enhances some image features important for further processing.

Pre-processing

- Pre-processing is a common name for operations with images at the lowest level of abstraction — both input and output are intensity images.
- These iconic images are of the same kind as the original data captured, with an intensity image usually represented by a matrix of image function values (brightness).
- The aim of pre-processing is an improvement of the image data that suppresses unwanted distortions or enhances some image features important for further processing.

Pre-processing

- Pre-processing is a common name for operations with images at the lowest level of abstraction — both input and output are intensity images.
- These iconic images are of the same kind as the original data captured, with an intensity image usually represented by a matrix of image function values (brightness).
- The aim of pre-processing is an improvement of the image data that suppresses unwanted distortions or enhances some image features important for further processing.

Pre-processing: Classification

- Four categories of image pre-processing methods according to the size of the pixel neighborhood that is used for the calculation of a new pixel brightness:
  - pixel brightness transformations.
  - geometric transformations.
  - pre-processing methods that use a local neighborhood of the processed pixel.
  - image restoration that requires knowledge about the entire image.
- Other classifications of image pre-processing methods exist.

Pre-processing: Classification

- Four categories of image pre-processing methods according to the size of the pixel neighborhood that is used for the calculation of a new pixel brightness:
  - pixel brightness transformations.
  - geometric transformations.
  - pre-processing methods that use a local neighborhood of the processed pixel.
  - image restoration that requires knowledge about the entire image.
- Other classifications of image pre-processing methods exist.

Pre-processing: Classification

- Four categories of image pre-processing methods according to the size of the pixel neighborhood that is used for the calculation of a new pixel brightness:
  - pixel brightness transformations.
  - geometric transformations.
  - pre-processing methods that use a local neighborhood of the processed pixel.
  - image restoration that requires knowledge about the entire image.
- Other classifications of image pre-processing methods exist.

Pre-processing: Classification

- Four categories of image pre-processing methods according to the size of the pixel neighborhood that is used for the calculation of a new pixel brightness:
  - pixel brightness transformations.
  - geometric transformations.
  - pre-processing methods that use a local neighborhood of the processed pixel.
  - image restoration that requires knowledge about the entire image.
- Other classifications of image pre-processing methods exist.

Pre-processing: Classification

- Four categories of image pre-processing methods according to the size of the pixel neighborhood that is used for the calculation of a new pixel brightness:
  - pixel brightness transformations.
  - geometric transformations.
  - pre-processing methods that use a local neighborhood of the processed pixel.
  - image restoration that requires knowledge about the entire image.
- Other classifications of image pre-processing methods exist.

Pre-processing: Classification

- Four categories of image pre-processing methods according to the size of the pixel neighborhood that is used for the calculation of a new pixel brightness:
  - pixel brightness transformations.
  - geometric transformations.
  - pre-processing methods that use a local neighborhood of the processed pixel.
  - image restoration that requires knowledge about the entire image.
- Other classifications of image pre-processing methods exist.

Pre-processing: basic idea

- Image pre-processing methods use the considerable redundancy in images.
- Neighboring pixels corresponding to one object in real images have essentially the same or similar brightness value — `spatial coherence` or `spatial correlation`.
- Thus, distorted pixel can often be restored as an average value of neighboring pixels.

Pre-processing: basic idea

- Image pre-processing methods use the considerable redundancy in images.
- Neighboring pixels corresponding to one object in real images have essentially the same or similar brightness value — `spatial coherence` or `spatial correlation`.
- Thus, distorted pixel can often be restored as an average value of neighboring pixels.

Pre-processing: basic idea

- Image pre-processing methods use the considerable redundancy in images.
- Neighboring pixels corresponding to one object in real images have essentially the same or similar brightness value — `spatial coherence` or `spatial correlation`.
- Thus, distorted pixel can often be restored as an average value of neighboring pixels.

Pre-processing: *a priori* information

- If pre-processing aims to correct some degradation in the image, the nature of *a priori* `information` is important and is used to different extent:
  - no knowledge about the nature of the degradation is used; only very general properties of the degradation are assumed.
  - using knowledge about the properties of the image acquisition device, and conditions under which the image was obtained. The nature of noise (usually its spectral characteristics) is sometimes known.
  - using knowledge about objects that are searched for in the image, which may simplify the pre-processing quite considerably.
- If knowledge about objects is not available in advance it can be estimated during the processing.

Pre-processing: *a priori* information

- If pre-processing aims to correct some degradation in the image, the nature of *a priori* `information` is important and is used to different extent:
  - no knowledge about the nature of the degradation is used; only very general properties of the degradation are assumed.
  - using knowledge about the properties of the image acquisition device, and conditions under which the image was obtained. The nature of noise (usually its spectral characteristics) is sometimes known.
  - using knowledge about objects that are searched for in the image, which may simplify the pre-processing quite considerably.
- If knowledge about objects is not available in advance it can be estimated during the processing.

Pre-processing: *a priori* information

- If pre-processing aims to correct some degradation in the image, the nature of *a priori* `information` is important and is used to different extent:
  - no knowledge about the nature of the degradation is used; only very general properties of the degradation are assumed.
  - using knowledge about the properties of the image acquisition device, and conditions under which the image was obtained. The nature of noise (usually its spectral characteristics) is sometimes known.
  - using knowledge about objects that are searched for in the image, which may simplify the pre-processing quite considerably.
- If knowledge about objects is not available in advance it can be estimated during the processing.

Pre-processing: *a priori* information

- If pre-processing aims to correct some degradation in the image, the nature of *a priori* `information` is important and is used to different extent:
  - no knowledge about the nature of the degradation is used; only very general properties of the degradation are assumed.
  - using knowledge about the properties of the image acquisition device, and conditions under which the image was obtained. The nature of noise (usually its spectral characteristics) is sometimes known.
  - using knowledge about objects that are searched for in the image, which may simplify the pre-processing quite considerably.
- If knowledge about objects is not available in advance it can be estimated during the processing.

Pre-processing: *a priori* information

- If pre-processing aims to correct some degradation in the image, the nature of *a priori* `information` is important and is used to different extent:
  - no knowledge about the nature of the degradation is used; only very general properties of the degradation are assumed.
  - using knowledge about the properties of the image acquisition device, and conditions under which the image was obtained. The nature of noise (usually its spectral characteristics) is sometimes known.
  - using knowledge about objects that are searched for in the image, which may simplify the pre-processing quite considerably.
- If knowledge about objects is not available in advance it can be estimated during the processing.

Pre-processing: Strategy

- The following strategy is possible.
  - First the image is coarsely processed to reduce data quantity and to find image objects.
  - The image information derived is used to create a hypothesis about image object properties and this hypothesis is then verified in the image at finer resolution.
  - Such an iterative process cab be repeated until the presence of knowledge is verified or rejected.
  - This feedback may span more than pre-processing, since segmentation also yields semantic knowledge about objects — thus feedback can be initiated after the object segmentation.

Pre-processing: Strategy

- The following strategy is possible.
  - First the image is coarsely processed to reduce data quantity and to find image objects.
  - The image information derived is used to create a hypothesis about image object properties and this hypothesis is then verified in the image at finer resolution.
  - Such an iterative process cab be repeated until the presence of knowledge is verified or rejected.
  - This feedback may span more than pre-processing, since segmentation also yields semantic knowledge about objects — thus feedback can be initiated after the object segmentation.

Pre-processing: Strategy

- The following strategy is possible.
  - First the image is coarsely processed to reduce data quantity and to find image objects.
  - The image information derived is used to create a hypothesis about image object properties and this hypothesis is then verified in the image at finer resolution.
  - Such an iterative process cab be repeated until the presence of knowledge is verified or rejected.
  - This feedback may span more than pre-processing, since segmentation also yields semantic knowledge about objects — thus feedback can be initiated after the object segmentation.

Pre-processing: Strategy

- The following strategy is possible.
  - First the image is coarsely processed to reduce data quantity and to find image objects.
  - The image information derived is used to create a hypothesis about image object properties and this hypothesis is then verified in the image at finer resolution.
  - Such an iterative process cab be repeated until the presence of knowledge is verified or rejected.
  - This feedback may span more than pre-processing, since segmentation also yields semantic knowledge about objects — thus feedback can be initiated after the object segmentation.

Pre-processing: Strategy

- The following strategy is possible.
  - First the image is coarsely processed to reduce data quantity and to find image objects.
  - The image information derived is used to create a hypothesis about image object properties and this hypothesis is then verified in the image at finer resolution.
  - Such an iterative process cab be repeated until the presence of knowledge is verified or rejected.
  - This feedback may span more than pre-processing, since segmentation also yields semantic knowledge about objects — thus feedback can be initiated after the object segmentation.

Pixel Brightness Transformations

- Brightness transformations modify pixel brightness — the transformation depends on the properties of a pixel itself.
- There are two brightness transformations:
- `Brightness corrections`
  - consider the original brightness
  - and pixel position in the image.
- `Gray scale transformation`
  - change brightness without regard to position in the image.

Pixel Brightness Transformations

- Brightness transformations modify pixel brightness — the transformation depends on the properties of a pixel itself.
- There are two brightness transformations:
- `Brightness corrections`
  - consider the original brightness
  - and pixel position in the image.
- `Gray scale transformation`
  - change brightness without regard to position in the image.

Pixel Brightness Transformations

- Brightness transformations modify pixel brightness — the transformation depends on the properties of a pixel itself.
- There are two brightness transformations:
- `Brightness corrections`
  - consider the original brightness
  - and pixel position in the image.
- `Gray scale transformation`
  - change brightness without regard to position in the image.

Pixel Brightness Transformations

- Brightness transformations modify pixel brightness — the transformation depends on the properties of a pixel itself.
- There are two brightness transformations:
- `Brightness corrections`
  - consider the original brightness
  - and pixel position in the image.
- `Gray scale transformation`
  - change brightness without regard to position in the image.

Pixel Brightness Transformations

- Brightness transformations modify pixel brightness — the transformation depends on the properties of a pixel itself.
- There are two brightness transformations:
- `Brightness corrections`
  - consider the original brightness
  - and pixel position in the image.
- `Gray scale transformation`
  - change brightness without regard to position in the image.

Pixel Brightness Transformations

- Brightness transformations modify pixel brightness — the transformation depends on the properties of a pixel itself.
- There are two brightness transformations:
- `Brightness corrections`
  - consider the original brightness
  - and pixel position in the image.
- `Gray scale transformation`
  - change brightness without regard to position in the image.

Pixel Brightness Transformations

- Brightness transformations modify pixel brightness — the transformation depends on the properties of a pixel itself.
- There are two brightness transformations:
- `Brightness corrections`
  - consider the original brightness
  - and pixel position in the image.
- `Gray scale transformation`
  - change brightness without regard to position in the image.

Outline

- Pixel Brightness Transformations
  - Position dependent brightness correction
  - Grey scale transformation
    - Windows and level
    - Histogram stretching
    - Histogram Equalization
    - Histogram Matching
- Geometric Transformations
  - Pixel Co-ordinate Transformations
    - Polynomial Approximation
    - Bilinear Transformations
    - Important Transformations
  - Brightness Interpolation
    - Nearest Neighbor Interpolation
    - Bilinear Interpolation
    - Bi-cubic interpolation
- Local pre-processing
  - Image smoothing
    - Averaging
    - Averaging with Data Validity
    - Averaging According to Inverse Gradient

Position dependent brightness correction

- Ideally, the sensitivity of image acquisition and digitization devices should not depend on position in the image, but this assumption is not valid in practice.
- Sources of degradation:
  - non-homogeneous property of optical system; The lens attenuates light more if it passes farther from the optical axis.
  - non-homogeneous sensitivity of light sensors; The photo sensitive part of the sensor (vacuum-tube camera, CCD camera elements) is not of identical sensitivity.
  - non-homogeneous object illumination.
- Systematic degradation can be suppressed by brightness correction.

Position dependent brightness correction

- Ideally, the sensitivity of image acquisition and digitization devices should not depend on position in the image, but this assumption is not valid in practice.
- Sources of degradation:
  - non-homogeneous property of optical system; The lens attenuates light more if it passes farther from the optical axis.
  - non-homogeneous sensitivity of light sensors; The photo sensitive part of the sensor (vacuum-tube camera, CCD camera elements) is not of identical sensitivity.
  - non-homogeneous object illumination.
- Systematic degradation can be suppressed by brightness correction.

Position dependent brightness correction

- Ideally, the sensitivity of image acquisition and digitization devices should not depend on position in the image, but this assumption is not valid in practice.
- Sources of degradation:
  - non-homogeneous property of optical system; The lens attenuates light more if it passes farther from the optical axis.
  - non-homogeneous sensitivity of light sensors; The photo sensitive part of the sensor (vacuum-tube camera, CCD camera elements) is not of identical sensitivity.
  - non-homogeneous object illumination.
- Systematic degradation can be suppressed by brightness correction.

Position dependent brightness correction

- Ideally, the sensitivity of image acquisition and digitization devices should not depend on position in the image, but this assumption is not valid in practice.
- Sources of degradation:
  - non-homogeneous property of optical system; The lens attenuates light more if it passes farther from the optical axis.
  - non-homogeneous sensitivity of light sensors; The photo sensitive part of the sensor (vacuum-tube camera, CCD camera elements) is not of identical sensitivity.
  - non-homogeneous object illumination.
- Systematic degradation can be suppressed by brightness correction.

Position dependent brightness correction

- Ideally, the sensitivity of image acquisition and digitization devices should not depend on position in the image, but this assumption is not valid in practice.
- Sources of degradation:
  - non-homogeneous property of optical system; The lens attenuates light more if it passes farther from the optical axis.
  - non-homogeneous sensitivity of light sensors; The photo sensitive part of the sensor (vacuum-tube camera, CCD camera elements) is not of identical sensitivity.
  - non-homogeneous object illumination.
- Systematic degradation can be suppressed by brightness correction.

Position dependent brightness correction

- Ideally, the sensitivity of image acquisition and digitization devices should not depend on position in the image, but this assumption is not valid in practice.
- Sources of degradation:
  - non-homogeneous property of optical system; The lens attenuates light more if it passes farther from the optical axis.
  - non-homogeneous sensitivity of light sensors; The photo sensitive part of the sensor (vacuum-tube camera, CCD camera elements) is not of identical sensitivity.
  - non-homogeneous object illumination.
- Systematic degradation can be suppressed by brightness correction.

Multipilcation Degradation

- Let a multiplicative error coefficient $e(i, j)$ describe the change from the ideal identity transfer function
  - $g(i, j)$ is the original undegraded image (or desired image);
  - $f(i, j)$ is the image containing degradation.

$$f(i, j) = e(i, j)g(i, j) \tag{1}$$

Multipilcation Degradation

- Let a multiplicative error coefficient $e(i,j)$ describe the change from the ideal identity transfer function
  - $g(i,j)$ is the original undegraded image (or desired image);
  - $f(i,j)$ is the image containing degradation.

$$f(i,j) = e(i,j)g(i,j) \quad (1)$$

Multipilcation Degradation

- Let a multiplicative error coefficient $e(i,j)$ describe the change from the ideal identity transfer function
  - $g(i,j)$ is the original undegraded image (or desired image);
  - $f(i,j)$ is the image containing degradation.

$$f(i,j) = e(i,j)g(i,j) \tag{1}$$

Correction for Multipilcation Degradation

- If a reference image $g_c(i,j)$ is known (e.g., constant brightness c)
  - the degraded result is $f_c(i,j)$
  - systematic brightness errors can be suppressed:

$$g(i,j) = \frac{f(i,j)}{e(i,j)} = \frac{g_c(i,j)f(i,j)}{f_c(i,j)} = \frac{c}{f_c(i,j)}f(i,j) \qquad (2)$$

Correction for Multipilcation Degradation

- If a reference image $g_c(i,j)$ is known (e.g., constant brightness c)
  - the degraded result is $f_c(i,j)$
  - systematic brightness errors can be suppressed:

$$g(i,j) = \frac{f(i,j)}{e(i,j)} = \frac{g_c(i,j)f(i,j)}{f_c(i,j)} = \frac{c}{f_c(i,j)}f(i,j) \qquad (2)$$

Correction for Multipilcation Degradation

- If a reference image $g_c(i, j)$ is known (e.g., constant brightness c)
  - the degraded result is $f_c(i, j)$
  - systematic brightness errors can be suppressed:

$$g(i, j) = \frac{f(i, j)}{e(i, j)} = \frac{g_c(i, j) f(i, j)}{f_c(i, j)} = \frac{c}{f_c(i, j)} f(i, j) \qquad (2)$$

Discussions I

- This method can be used only if the image degradation process is stable.
- If we wish to suppress this kind of degradation in the image capture process, we should calibrate the device from time to time (find error coefficients $e(i,j)$)
- This method implicitly assumes linearity of the transformation, which is not true in reality as the brightness scale is limited into some interval.
  - overflow is possible in (2). Then the limits of the brightness scale are used instead in (2).
  - The best reference image should have brightness that is far enough from both limits.

Discussions I

- This method can be used only if the image degradation process is stable.
- If we wish to suppress this kind of degradation in the image capture process, we should calibrate the device from time to time (find error coefficients $e(i, j)$)
- This method implicitly assumes linearity of the transformation, which is not true in reality as the brightness scale is limited into some interval.
  - overflow is possible in (2). Then the limits of the brightness scale are used instead in (2).
  - The best reference image should have brightness that is far enough from both limits.

Discussions I

- This method can be used only if the image degradation process is stable.
- If we wish to suppress this kind of degradation in the image capture process, we should calibrate the device from time to time (find error coefficients $e(i, j)$)
- This method implicitly assumes linearity of the transformation, which is not true in reality as the brightness scale is limited into some interval.
  - overflow is possible in (2). Then the limits of the brightness scale are used instead in (2).
  - The best reference image should have brightness that is far enough from both limits.

Discussions I

- This method can be used only if the image degradation process is stable.
- If we wish to suppress this kind of degradation in the image capture process, we should calibrate the device from time to time (find error coefficients $e(i, j)$)
- This method implicitly assumes linearity of the transformation, which is not true in reality as the brightness scale is limited into some interval.
  - overflow is possible in (2). Then the limits of the brightness scale are used instead in (2).
  - The best reference image should have brightness that is far enough from both limits.

Discussions I

- This method can be used only if the image degradation process is stable.
- If we wish to suppress this kind of degradation in the image capture process, we should calibrate the device from time to time (find error coefficients $e(i, j)$)
- This method implicitly assumes linearity of the transformation, which is not true in reality as the brightness scale is limited into some interval.
  - overflow is possible in (2). Then the limits of the brightness scale are used instead in (2).
  - The best reference image should have brightness that is far enough from both limits.

Discussions II

- If the gray scale has 256 brightnesses the ideal image has constant brightness value 128.
  - Most TV/DV cameras have automatic control of the gain which allows them to operate under changing illumination conditions.
  - If systematic errors are suppressed using error coefficients, this automatic gain control should be switched off first.

Discussions II

- If the gray scale has 256 brightnesses the ideal image has constant brightness value 128.
  - Most TV/DV cameras have automatic control of the gain which allows them to operate under changing illumination conditions.
  - If systematic errors are suppressed using error coefficients, this automatic gain control should be switched off first.

Discussions II

- If the gray scale has 256 brightnesses the ideal image has constant brightness value 128.
  - Most TV/DV cameras have automatic control of the gain which allows them to operate under changing illumination conditions.
  - If systematic errors are suppressed using error coefficients, this automatic gain control should be switched off first.

Brightness Correction Example

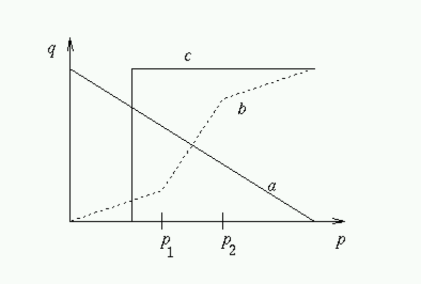
- The matlab script for this example is brightness_correction.m.

Outline

Grey scale transformation

- Grey scale transformations do not depend on the position of the pixel in the image.
- Brightness transform is a monotonic function:

$$q = T(p) \quad (3)$$

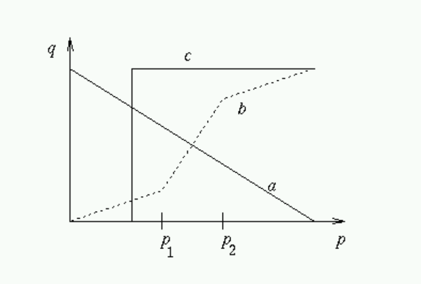


- a - Negative transformation
- b - contrast enhancement (between p_1 and p_2)
- c - Brightness thresholding

Grey scale transformation

- Grey scale transformations do not depend on the position of the pixel in the image.
- Brightness transform is a monotonic function:

$$q = T(p) \tag{3}$$

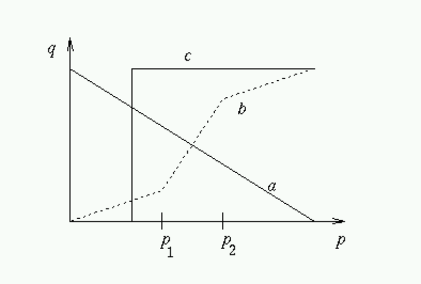


- a - Negative transformation
- b - contrast enhancement (between p_1 and p_2)
- c - Brightness thresholding

Grey scale transformation

- Grey scale transformations do not depend on the position of the pixel in the image.
- Brightness transform is a monotonic function:

$$q = T(p) \tag{3}$$

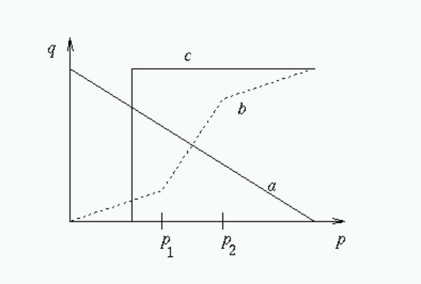


- a - Negative transformation
- b - contrast enhancement (between p_1 and p_2)
- c - Brightness thresholding

Grey scale transformation

- Grey scale transformations do not depend on the position of the pixel in the image.
- Brightness transform is a monotonic function:

$$q = T(p) \tag{3}$$

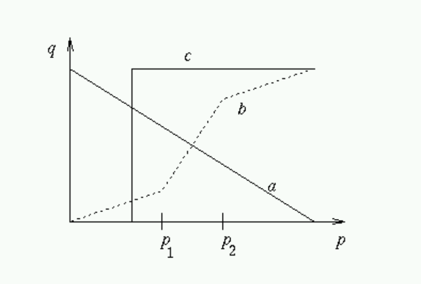


- a - Negative transformation
- b - contrast enhancement (between p_1 and p_2)
- c - Brightness thresholding

Grey scale transformation

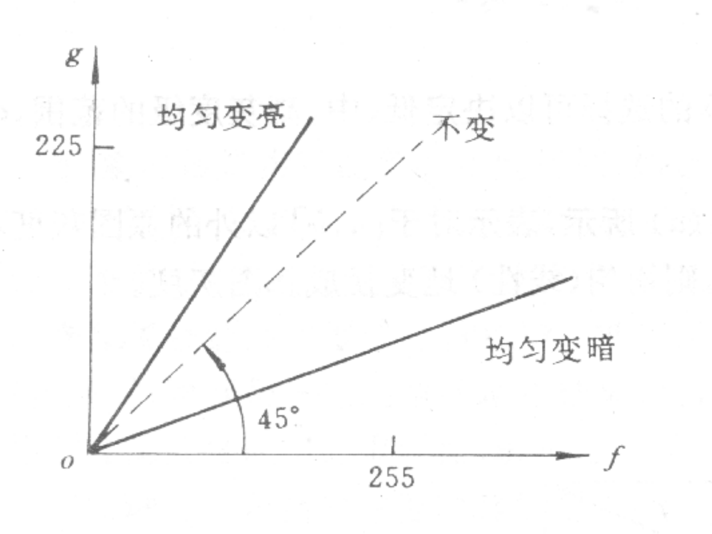
- Grey scale transformations do not depend on the position of the pixel in the image.
- Brightness transform is a monotonic function:

$$q = T(p) \tag{3}$$

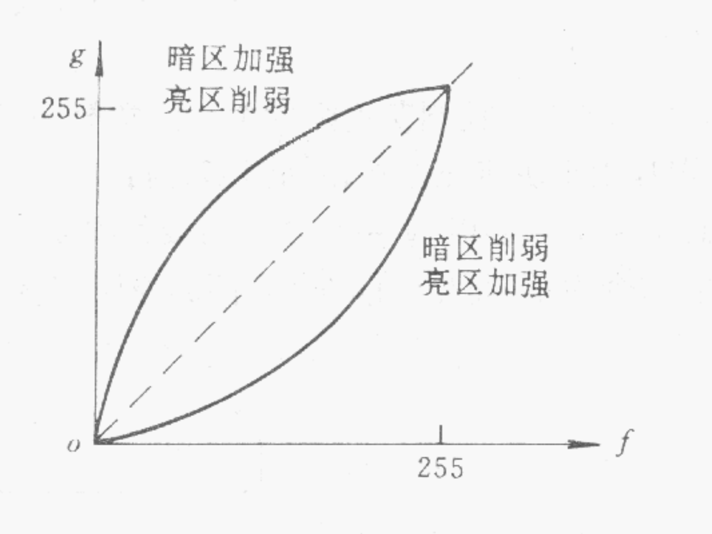


- a - Negative transformation
- b - contrast enhancement (between p_1 and p_2)
- c - Brightness thresholding

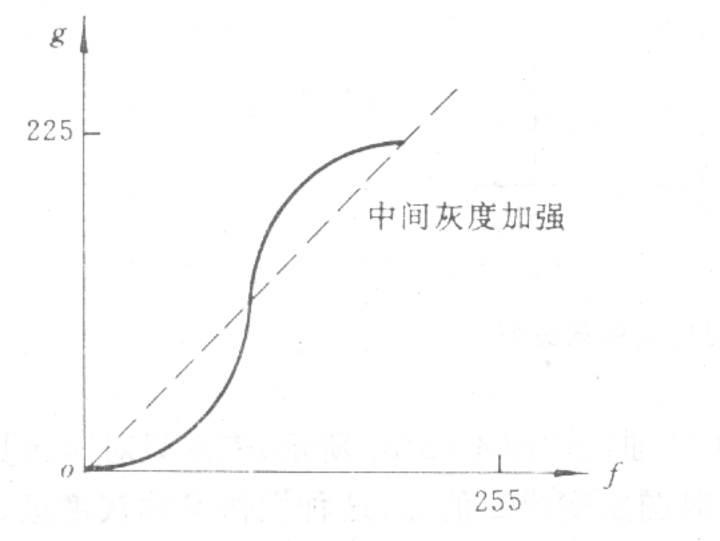
Some Other Examples I



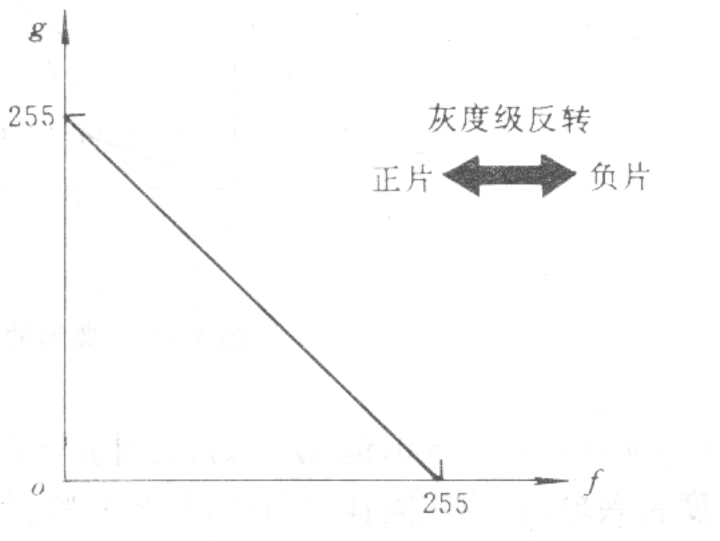
Some Other Examples II



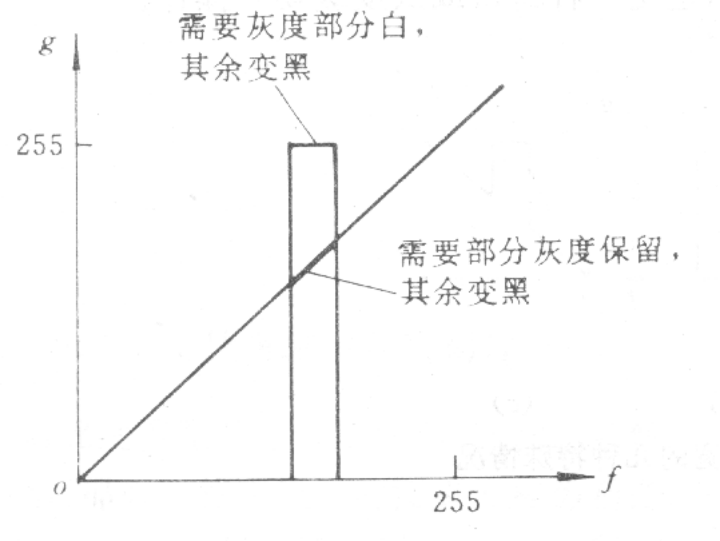
Some Other Examples III



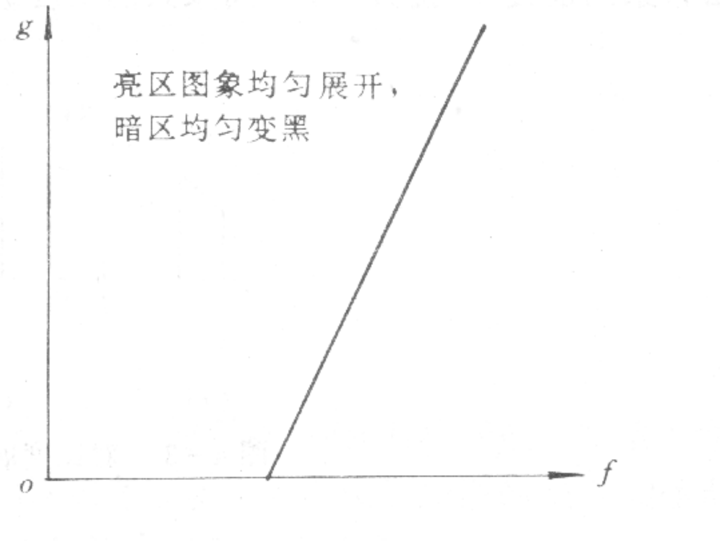
Some Other Examples IV



Some Other Examples V



Some Other Examples VI



Look-up Tables

- Grey scale transformations can be performed using look-up tables.
- Grey scale transformations are mostly used if the result is viewed by a human.

Look-up Tables

- Grey scale transformations can be performed using look-up tables.
- Grey scale transformations are mostly used if the result is viewed by a human.

Windows and level I

- It is an interactive contrast enhancement tool.
- It is an expansion of the contrast of pixels within a given window range.
- Two parameters define the range: the middle point Level, and the width of the range Window.
- Another name for this operation is `Intensity of Interest` (IOI).

Windows and level II

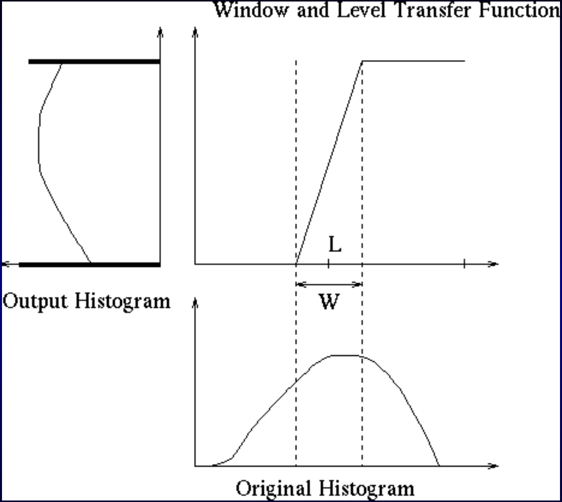
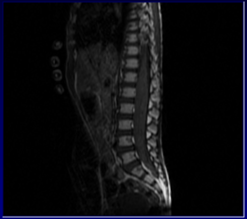
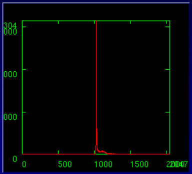


Figure: A graphic visualization of this process.

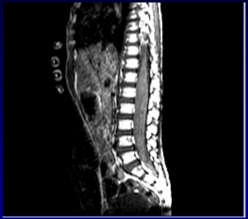
Example I



(a) Original



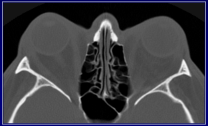
(b) Hisogram



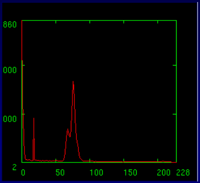
(c) After Windows and Level Contrast Change

Figure: First find the minimum and maximum values, decide the initial value, bin-width and number of bins for computing the histogram. From the histogram, chose the lower and upper cutoff value.

Example II



(a) Original



(b) Hisogram

(c) After Windows and Level Contrast Change

Histogram stretching

- Histogram stretching can be seen as a Window and Level contrast enhancement technique where the window ranges from the minimum to the maximum pixel values of the image.
- This normalization or histogram stretching operation is automatically performed in many display operators.
- To fully illustrate the difference between the two displayed images a gray level scale was superimposed in both images to guarantee that the display operator will use the same gray-level scale.

Histogram stretching

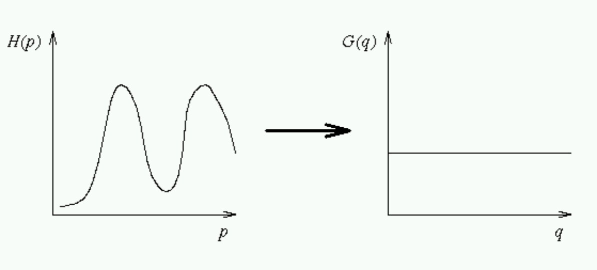
- Histogram stretching can be seen as a Window and Level contrast enhancement technique where the window ranges from the minimum to the maximum pixel values of the image.
- This normalization or histogram stretching operation is automatically performed in many display operators.
- To fully illustrate the difference between the two displayed images a gray level scale was superimposed in both images to guarantee that the display operator will use the same gray-level scale.

Histogram stretching

- Histogram stretching can be seen as a Window and Level contrast enhancement technique where the window ranges from the minimum to the maximum pixel values of the image.
- This normalization or histogram stretching operation is automatically performed in many display operators.
- To fully illustrate the difference between the two displayed images a gray level scale was superimposed in both images to guarantee that the display operator will use the same gray-level scale.

Histogram Equalization I

- Histogram equalization is **to produce an image with equally distributed brightness levels over the whole brightness scale**.



Histogram Equalization II

- Let $H(p)$ be the input histogram and the input gray-scale range be $[p_0, p_k]$.
- We are to find a monotonic pixel brightness transform $q = T(p)$ such that the output histogram $G(q)$ is uniform over the whole output brightness scale $[q_0, q_k]$.
- The histogram is a discrete probability density function.
- The monotonic property of the transform T implies

$$\sum_{i=0}^{j} G(q_i) = \sum_{i=0}^{j} H(p_i) \tag{4}$$

where $q_i = T(p_i)$.

- The sum in the above equation can be interpreted as discrete distribution function.

Histogram Equalization II

- Let $H(p)$ be the input histogram and the input gray-scale range be $[p_0, p_k]$.
- We are to find a monotonic pixel brightness transform $q = T(p)$ such that the output histogram $G(q)$ is uniform over the whole output brightness scale $[q_0, q_k]$.
- The histogram is a discrete probability density function.
- The monotonic property of the transform T implies

$$\sum_{i=0}^{j} G(q_i) = \sum_{i=0}^{j} H(p_i) \qquad (4)$$

where $q_i = T(p_i)$.

- The sum in the above equation can be interpreted as discrete distribution function.

Histogram Equalization II

- Let $H(p)$ be the input histogram and the input gray-scale range be $[p_0, p_k]$.
- We are to find a monotonic pixel brightness transform $q = T(p)$ such that the output histogram $G(q)$ is uniform over the whole output brightness scale $[q_0, q_k]$.
- The histogram is a discrete probability density function.
- The monotonic property of the transform T implies

$$\sum_{i=0}^{j} G(q_i) = \sum_{i=0}^{j} H(p_i) \qquad (4)$$

where $q_i = T(p_i)$.
- The sum in the above equation can be interpreted as discrete distribution function.

Histogram Equalization II

- Let $H(p)$ be the input histogram and the input gray-scale range be $[p_0, p_k]$.
- We are to find a monotonic pixel brightness transform $q = T(p)$ such that the output histogram $G(q)$ is uniform over the whole output brightness scale $[q_0, q_k]$.
- The histogram is a discrete probability density function.
- The monotonic property of the transform T implies

$$\sum_{i=0}^{j} G(q_i) = \sum_{i=0}^{j} H(p_i) \tag{4}$$

where $q_i = T(p_i)$.
- The sum in the above equation can be interpreted as discrete distribution function.

Histogram Equalization II

- Let $H(p)$ be the input histogram and the input gray-scale range be $[p_0, p_k]$.
- We are to find a monotonic pixel brightness transform $q = T(p)$ such that the output histogram $G(q)$ is uniform over the whole output brightness scale $[q_0, q_k]$.
- The histogram is a discrete probability density function.
- The monotonic property of the transform T implies

$$\sum_{i=0}^{j} G(q_i) = \sum_{i=0}^{j} H(p_i) \tag{4}$$

where $q_i = T(p_i)$.
- The sum in the above equation can be interpreted as discrete distribution function.

Histogram Equalization III

- Assume that the image has M rows and N columns. The total number of pixels is MN.
- The equalized histogram corresponding to the uniform probability density function f whose function value satisfies

$$f(q)(q_k - q_0) = MN. \tag{5}$$

- The continuous version of (4) is

$$\int_{q_0}^{q} f(r)\, dr = \int_{p_0}^{p} H(s)\, ds, \quad q = T(p). \tag{6}$$

- Therefore, we obtain,

$$MN \int_{q_0}^{q} \frac{1}{q_k - q_0}\, dr = \frac{MN(q - q_0)}{q_k - q_0} = \int_{p_0}^{p} H(s)\, ds. \tag{7}$$

Histogram Equalization III

- Assume that the image has M rows and N columns. The total number of pixels is MN.
- The equalized histogram corresponding to the uniform probability density function f whose function value satisfies

$$f(q)(q_k - q_0) = MN. \tag{5}$$

- The continuous version of (4) is

$$\int_{q_0}^{q} f(r)\, dr = \int_{p_0}^{p} H(s)\, ds, \quad q = T(p). \tag{6}$$

- Therefore, we obtain,

$$MN \int_{q_0}^{q} \frac{1}{q_k - q_0}\, dr = \frac{MN(q - q_0)}{q_k - q_0} = \int_{p_0}^{p} H(s)\, ds. \tag{7}$$

Histogram Equalization III

- Assume that the image has M rows and N columns. The total number of pixels is MN.
- The equalized histogram corresponding to the uniform probability density function f whose function value satisfies

$$f(q)(q_k - q_0) = MN. \tag{5}$$

- The continuous version of (4) is

$$\int_{q_0}^{q} f(r)\,dr = \int_{p_0}^{p} H(s)\,ds, \quad q = T(p). \tag{6}$$

- Therefore, we obtain,

$$MN \int_{q_0}^{q} \frac{1}{q_k - q_0}\,dr = \frac{MN(q - q_0)}{q_k - q_0} = \int_{p_0}^{p} H(s)\,ds. \tag{7}$$

Histogram Equalization III

- Assume that the image has M rows and N columns. The total number of pixels is MN.
- The equalized histogram corresponding to the uniform probability density function f whose function value satisfies

$$f(q)(q_k - q_0) = MN. \tag{5}$$

- The continuous version of (4) is

$$\int_{q_0}^{q} f(r)\,dr = \int_{p_0}^{p} H(s)\,ds, \quad q = T(p). \tag{6}$$

- Therefore, we obtain,

$$MN \int_{q_0}^{q} \frac{1}{q_k - q_0}\,dr = \frac{MN(q - q_0)}{q_k - q_0} = \int_{p_0}^{p} H(s)\,ds. \tag{7}$$

Histogram Equalization IV

- The desired pixel brightness transformation T can then be derived as

$$q = T(p) = \frac{q_k - q_0}{MN} \int_{p_0}^{p} H(s)\, ds + q_0. \qquad (8)$$

- The integral in the above equation is called the cumulative histogram, which is approximated by a sum for digital images,

$$F[p] = \int_{p_0}^{p} H(s)\, ds = \sum_{i=0}^{j} H(p_i), \quad p_j = \mathrm{round}(p).$$

- Therefore, resulting histograms are not generally equalized ideally.
- The discrete approximation of the continuous pixel brightness transformation form the above equation is

$$q_j = T(p_j) = \frac{q_k - q_0}{MN} \sum_{i=0}^{j} H(p_i) + q_0. \qquad (9)$$

Histogram Equalization IV

- The desired pixel brightness transformation T can then be derived as

$$q = T(p) = \frac{q_k - q_0}{MN} \int_{p_0}^{p} H(s)\, ds + q_0. \qquad (8)$$

- The integral in the above equation is called the cumulative histogram, which is approximated by a sum for digital images,

$$F[p] = \int_{p_0}^{p} H(s)\, ds = \sum_{i=0}^{j} H(p_i), \quad p_j = \mathrm{round}(p).$$

- Therefore, resulting histograms are not generally equalized ideally.
- The discrete approximation of the continuous pixel brightness transformation form the above equation is

$$q_j = T(p_j) = \frac{q_k - q_0}{MN} \sum_{i=0}^{j} H(p_i) + q_0. \qquad (9)$$

Histogram Equalization IV

- The desired pixel brightness transformation T can then be derived as

$$q = T(p) = \frac{q_k - q_0}{MN} \int_{p_0}^{p} H(s)\, ds + q_0. \qquad (8)$$

- The integral in the above equation is called the cumulative histogram, which is approximated by a sum for digital images,

$$F[p] = \int_{p_0}^{p} H(s)\, ds = \sum_{i=0}^{j} H(p_i), \quad p_j = \text{round}(p).$$

- Therefore, resulting histograms are not generally equalized ideally.
- The discrete approximation of the continuous pixel brightness transformation form the above equation is

$$q_j = T(p_j) = \frac{q_k - q_0}{MN} \sum_{i=0}^{j} H(p_i) + q_0. \qquad (9)$$

Histogram Equalization IV

- The desired pixel brightness transformation T can then be derived as

$$q = T(p) = \frac{q_k - q_0}{MN} \int_{p_0}^{p} H(s)\, ds + q_0. \qquad (8)$$

- The integral in the above equation is called the cumulative histogram, which is approximated by a sum for digital images,

$$F[p] = \int_{p_0}^{p} H(s)\, ds = \sum_{i=0}^{j} H(p_i), \quad p_j = \text{round}(p).$$

- Therefore, resulting histograms are not generally equalized ideally.
- The discrete approximation of the continuous pixel brightness transformation form the above equation is

$$q_j = T(p_j) = \frac{q_k - q_0}{MN} \sum_{i=0}^{j} H(p_i) + q_0. \qquad (9)$$

Histogram Equalization Algorithm

1. For an $N \times M$ image of G gray-levels (often 256), create two arrays H and T of length G initialized with 0 values.
2. Form the image histogram: scan every pixel and increment the relevant member of H — if pixel X has intensity p, perform

$$H[p] = H[p] + 1 \tag{10}$$

3. Form the cumulative image histogram H_c. We may use the same array H to stored the result.

$$H[0] = H[0]$$
$$H[p] = H[p-1] + H[p]$$

for $p = 1, \cdots, G-1$.

4. Set

$$T[p] = \mathrm{round}\left[\frac{G-1}{MN} H[p]\right]. \tag{11}$$

Note the new gray-scale is assumed to be the same as the input image, i.e., $q_k = G - 1$ and $q_0 = 0$.

5. Rescan the image and write an output image with gray-levels q, setting

$$q = T[p]. \tag{12}$$

Histogram Equalization Algorithm

1. For an $N \times M$ image of G gray-levels (often 256), create two arrays H and T of length G initialized with 0 values.
2. Form the image histogram: scan every pixel and increment the relevant member of H — if pixel X has intensity p, perform

$$H[p] = H[p] + 1 \tag{10}$$

3. Form the cumulative image histogram H_c. We may use the same array H to stored the result.

$$H[0] = H[0]$$
$$H[p] = H[p-1] + H[p]$$

for $p = 1, \cdots, G-1$.

4. Set

$$T[p] = \mathrm{round}\left[\frac{G-1}{MN} H[p]\right]. \tag{11}$$

Note the new gray-scale is assumed to be the same as the input image, i.e., $q_k = G - 1$ and $q_0 = 0$.

5. Rescan the image and write an output image with gray-levels q, setting

$$q = T[p]. \tag{12}$$

Histogram Equalization Algorithm

1. For an $N \times M$ image of G gray-levels (often 256), create two arrays H and T of length G initialized with 0 values.
2. Form the image histogram: scan every pixel and increment the relevant member of H — if pixel X has intensity p, perform

$$H[p] = H[p] + 1 \tag{10}$$

3. Form the cumulative image histogram H_c. We may use the same array H to stored the result.

$$H[0] = H[0]$$
$$H[p] = H[p-1] + H[p]$$

for $p = 1, \cdots, G-1$.

4. Set

$$T[p] = \text{round}\left[\frac{G-1}{MN} H[p]\right]. \tag{11}$$

Note the new gray-scale is assumed to be the same as the input image, i.e., $q_k = G - 1$ and $q_0 = 0$.

5. Rescan the image and write an output image with gray-levels q, setting

$$q = T[p]. \tag{12}$$

Histogram Equalization Algorithm

1. For an $N \times M$ image of G gray-levels (often 256), create two arrays H and T of length G initialized with 0 values.
2. Form the image histogram: scan every pixel and increment the relevant member of H — if pixel X has intensity p, perform

$$H[p] = H[p] + 1 \tag{10}$$

3. Form the cumulative image histogram H_c. We may use the same array H to stored the result.

$$H[0] = H[0]$$
$$H[p] = H[p-1] + H[p]$$

for $p = 1, \cdots, G-1$.
4. Set

$$T[p] = \mathrm{round}\left[\frac{G-1}{MN} H[p]\right]. \tag{11}$$

Note the new gray-scale is assumed to be the same as the input image, i.e., $q_k = G-1$ and $q_0 = 0$.
5. Rescan the image and write an output image with gray-levels q, setting

$$q = T[p]. \tag{12}$$

Histogram Equalization Algorithm

1. For an $N \times M$ image of G gray-levels (often 256), create two arrays H and T of length G initialized with 0 values.
2. Form the image histogram: scan every pixel and increment the relevant member of H — if pixel X has intensity p, perform

$$H[p] = H[p] + 1 \tag{10}$$

3. Form the cumulative image histogram H_c. We may use the same array H to stored the result.

$$H[0] = H[0]$$
$$H[p] = H[p-1] + H[p]$$

   for $p = 1, \cdots, G-1$.
4. Set

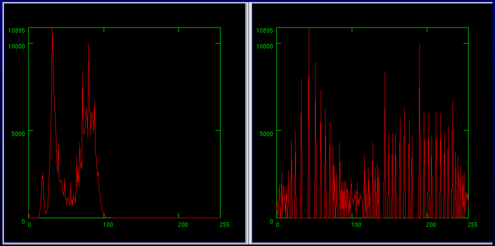
$$T[p] = \text{round}\left[\frac{G-1}{MN} H[p]\right]. \tag{11}$$

   Note the new gray-scale is assumed to be the same as the input image, i.e., $q_k = G - 1$ and $q_0 = 0$.
5. Rescan the image and write an output image with gray-levels q, setting

$$q = T[p]. \tag{12}$$

Example I

(d) Images before and after histoeqlization



(e) Hisograms

Example II

- The matlab script from `visionbook` is visionbook/05Preproc/hist_equal_demo.m.

Discussions

The following comments are from [Klette and Zamperoni, 1996, p. 148]:

- The gray value range equalization may be used for improving the image quality if the original image covers only a part of the full gray scale.
- An insufficient exploitation of the full gray scale is mostly due to image acquisition circumstances, as e.g., low scene illumination or automatic gain control of the camera.
- In case of good gray value dynamics of the input image, an equalization can lead even to quality losses in form of unsharp edges.

Discussions

The following comments are from [Klette and Zamperoni, 1996, p. 148]:

- The gray value range equalization may be used for improving the image quality if the original image covers only a part of the full gray scale.
- An insufficient exploitation of the full gray scale is mostly due to image acquisition circumstances, as e.g., low scene illumination or automatic gain control of the camera.
- In case of good gray value dynamics of the input image, an equalization can lead even to quality losses in form of unsharp edges.

Discussions

The following comments are from [Klette and Zamperoni, 1996, p. 148]:

- The gray value range equalization may be used for improving the image quality if the original image covers only a part of the full gray scale.
- An insufficient exploitation of the full gray scale is mostly due to image acquisition circumstances, as e.g., low scene illumination or automatic gain control of the camera.
- In case of good gray value dynamics of the input image, an equalization can lead even to quality losses in form of unsharp edges.

Example III

- Another histogram equalization example. The matlab script is ../program/histo_eq_demo.m.
- Some edges become unsharp.

Example III

- Another histogram equalization example. The matlab script is ../program/histo_eq_demo.m.
- Some edges become unsharp.

Histogram Matching I

- Histogram matching is to produce an image with desired distributed brightness levels over the whole brightness scale.
- Assume the desired probability density function is $G(q)$.
- Let the desired pixel brightness transform be T.
- Similarly we have

$$F[p] = \int_{q_0}^{q=T[p]} G[s]\, ds \qquad (13)$$

  where $F[p]$ is the cumulative histogram of the input image. Assumed that the cumulative histogram is normalized in $[0, 1]$.
- From the above equation, it is possible to find the transformation T.

Histogram Matching I

- `Histogram matching` is to produce an image with desired distributed brightness levels over the whole brightness scale.
- Assume the desired probability density function is $G(q)$.
- Let the desired pixel brightness transform be T.
- Similarly we have

$$F[p] = \int_{q_0}^{q=T[p]} G[s]\, ds \qquad (13)$$

where $F[p]$ is the cumulative histogram of the input image. Assumed that the cumulative histogram is normalized in $[0, 1]$.

- From the above equation, it is possible to find the transformation T.

Histogram Matching I

- `Histogram matching` is to produce an image with desired distributed brightness levels over the whole brightness scale.
- Assume the desired probability density function is $G(q)$.
- Let the desired pixel brightness transform be T.
- Similarly we have

$$F[p] = \int_{q_0}^{q=T[p]} G[s]\, ds \tag{13}$$

where $F[p]$ is the cumulative histogram of the input image. Assumed that the cumulative histogram is normalized in $[0, 1]$.

- From the above equation, it is possible to find the transformation T.

Histogram Matching I

- `Histogram matching` is to produce an image with desired distributed brightness levels over the whole brightness scale.
- Assume the desired probability density function is $G(q)$.
- Let the desired pixel brightness transform be T.
- Similarly we have

$$F[p] = \int_{q_0}^{q=T[p]} G[s]\, ds \qquad (13)$$

where $F[p]$ is the cumulative histogram of the input image. Assumed that the cumulative histogram is normalized in $[0, 1]$.
- From the above equation, it is possible to find the transformation T.

Histogram Matching I

- `Histogram matching` is to produce an image with desired distributed brightness levels over the whole brightness scale.
- Assume the desired probability density function is $G(q)$.
- Let the desired pixel brightness transform be T.
- Similarly we have

$$F[p] = \int_{q_0}^{q=T[p]} G[s]\,ds \tag{13}$$

  where $F[p]$ is the cumulative histogram of the input image. Assumed that the cumulative histogram is normalized in $[0, 1]$.
- From the above equation, it is possible to find the transformation T.

Histogram Matching II

- E.g., if G is the exponential distribution,

$$G[q] = \alpha \mathbf{e}^{-\alpha(q-q_0)} \tag{14}$$

for $q \geq q_0$.

- We have

$$F[p] = 1 - \mathbf{e}^{-\alpha(T[p]-q_0)} \tag{15}$$

- Then we find the transformation

$$T[p] = q_0 - \frac{1}{\alpha}\log(1 - F[p]) \tag{16}$$

- In the discrete case, the transformation can be implemented by building look-up tables.

Histogram Matching II

- E.g., if G is the exponential distribution,

$$G[q] = \alpha \mathbf{e}^{-\alpha(q-q_0)} \tag{14}$$

  for $q \geq q_0$.
- We have

$$F[p] = 1 - \mathbf{e}^{-\alpha(T[p]-q_0)} \tag{15}$$

- Then we find the transformation

$$T[p] = q_0 - \frac{1}{\alpha}\log(1 - F[p]) \tag{16}$$

- In the discrete case, the transformation can be implemented by building look-up tables.

Histogram Matching II

- E.g., if G is the exponential distribution,

$$G[q] = \alpha \mathbf{e}^{-\alpha(q-q_0)} \tag{14}$$

for $q \geq q_0$.

- We have

$$F[p] = 1 - \mathbf{e}^{-\alpha(T[p]-q_0)} \tag{15}$$

- Then we find the transformation

$$T[p] = q_0 - \frac{1}{\alpha}\log(1 - F[p]) \tag{16}$$

- In the discrete case, the transformation can be implemented by building look-up tables.

Histogram Matching II

- E.g., if G is the exponential distribution,

$$G[q] = \alpha \mathbf{e}^{-\alpha(q-q_0)} \tag{14}$$

for $q \geq q_0$.

- We have

$$F[p] = 1 - \mathbf{e}^{-\alpha(T[p]-q_0)} \tag{15}$$

- Then we find the transformation

$$T[p] = q_0 - \frac{1}{\alpha}\log(1 - F[p]) \tag{16}$$

- In the discrete case, the transformation can be implemented by building look-up tables.

Homework

Implementation of those transforms are left as homework. The homework includes:

1. Grey scale transformations with given functions such as exponent, logarithm, user defined (by some control points for a piece wise linear function), etc.;
2. Histogram stretching to a given range;
3. Histogram equalization;
4. Histogram matching: find other probability density functions in [Rongchun Zhao, 2000, p. 81]. Implemente the histogram matching transforms for those probability density functions.
5. ★ Histogram matching to the histogram from a given image.

Note:

- Histograms should be plotted when histograms are involved.
- Your final score will also depend on the interface of your program.
- Be sure to give us a friendly interface.
- Be sure to choose illustrative images.

Homework

Implementation of those transforms are left as homework. The homework includes:

1. Grey scale transformations with given functions such as exponent, logarithm, user defined (by some control points for a piece wise linear function), etc.;
2. Histogram stretching to a given range;
3. Histogram equalization;
4. Histogram matching: find other probability density functions in [Rongchun Zhao, 2000, p. 81]. Implemente the histogram matching transforms for those probability density functions.
5. ★ Histogram matching to the histogram from a given image.

Note:

- Histograms should be plotted when histograms are involved.
- Your final score will also depend on the interface of your program.
- Be sure to give us a friendly interface.
- Be sure to choose illustrative images.

Homework

Implementation of those transforms are left as homework.

The homework includes:

1. Grey scale transformations with given functions such as exponent, logarithm, user defined (by some control points for a piece wise linear function), etc.;
2. Histogram stretching to a given range;
3. Histogram equalization;
4. Histogram matching: find other probability density functions in [Rongchun Zhao, 2000, p. 81]. Implemente the histogram matching transforms for those probability density functions.
5. ★ Histogram matching to the histogram from a given image.

Note:

- Histograms should be plotted when histograms are involved.
- Your final score will also depend on the interface of your program.
- Be sure to give us a friendly interface.
- Be sure to choose illustrative images.

Homework

Implementation of those transforms are left as homework.

The homework includes:

1. Grey scale transformations with given functions such as exponent, logarithm, user defined (by some control points for a piece wise linear function), etc.;
2. Histogram stretching to a given range;
3. Histogram equalization;
4. Histogram matching: find other probability density functions in [Rongchun Zhao, 2000, p. 81]. Implemente the histogram matching transforms for those probability density functions.
5. ★ Histogram matching to the histogram from a given image.

Note:

- Histograms should be plotted when histograms are involved.
- Your final score will also depend on the interface of your program.
- Be sure to give us a friendly interface.
- Be sure to choose illustrative images.

Homework

Implementation of those transforms are left as homework.
The homework includes:

1. Grey scale transformations with given functions such as exponent, logarithm, user defined (by some control points for a piece wise linear function), etc.;
2. Histogram stretching to a given range;
3. Histogram equalization;
4. Histogram matching: find other probability density functions in [Rongchun Zhao, 2000, p. 81]. Implemente the histogram matching transforms for those probability density functions.
5. ★ Histogram matching to the histogram from a given image.

Note:

- Histograms should be plotted when histograms are involved.
- Your final score will also depend on the interface of your program.
- Be sure to give us a friendly interface.
- Be sure to choose illustrative images.

Homework

Implementation of those transforms are left as homework.
The homework includes:

1. Grey scale transformations with given functions such as exponent, logarithm, user defined (by some control points for a piece wise linear function), etc.;
2. Histogram stretching to a given range;
3. Histogram equalization;
4. Histogram matching: find other probability density functions in [Rongchun Zhao, 2000, p. 81]. Implemente the histogram matching transforms for those probability density functions.
5. ★ Histogram matching to the histogram from a given image.

Note:

- Histograms should be plotted when histograms are involved.
- Your final score will also depend on the interface of your program.
- Be sure to give us a friendly interface.
- Be sure to choose illustrative images.

Homework

Implementation of those transforms are left as homework.

The homework includes:

1. Grey scale transformations with given functions such as exponent, logarithm, user defined (by some control points for a piece wise linear function), etc.;
2. Histogram stretching to a given range;
3. Histogram equalization;
4. Histogram matching: find other probability density functions in [Rongchun Zhao, 2000, p. 81]. Implemente the histogram matching transforms for those probability density functions.
5. ★ Histogram matching to the histogram from a given image.

Note:

- Histograms should be plotted when histograms are involved.
- Your final score will also depend on the interface of your program.
- Be sure to give us a friendly interface.
- Be sure to choose illustrative images.

Homework

Implementation of those transforms are left as homework.
The homework includes:

1. Grey scale transformations with given functions such as exponent, logarithm, user defined (by some control points for a piece wise linear function), etc.;
2. Histogram stretching to a given range;
3. Histogram equalization;
4. Histogram matching: find other probability density functions in [Rongchun Zhao, 2000, p. 81]. Implemente the histogram matching transforms for those probability density functions.
5. ★ Histogram matching to the histogram from a given image.

Note:

- Histograms should be plotted when histograms are involved.
- Your final score will also depend on the interface of your program.
- Be sure to give us a friendly interface.
- Be sure to choose illustrative images.

Homework

Implementation of those transforms are left as homework.
The homework includes:

1. Grey scale transformations with given functions such as exponent, logarithm, user defined (by some control points for a piece wise linear function), etc.;
2. Histogram stretching to a given range;
3. Histogram equalization;
4. Histogram matching: find other probability density functions in [Rongchun Zhao, 2000, p. 81]. Implemente the histogram matching transforms for those probability density functions.
5. ★ Histogram matching to the histogram from a given image.

Note:

- Histograms should be plotted when histograms are involved.
- Your final score will also depend on the interface of your program.
- Be sure to give us a friendly interface.
- Be sure to choose illustrative images.

Geometric Transformations

- Geometric transformations are common in computer graphics, and are often used in image analysis.
- Geometric transforms permit the elimination of geometric distortion that occurs when an image is captured.
- If one attempts to match two different images of the same object, a geometric transformation may be needed:
  - matching remotely sensed images of the same area taken after one year, when the recent image is not taken from the same orientation and/or position.
  - correcting for document skew in OCR, which occurs when an image with an obvious orientation (e.g., a printed page) is scanned, or otherwise captured, at a different orientation.

Geometric Transformations

- Geometric transformations are common in computer graphics, and are often used in image analysis.
- Geometric transforms permit the elimination of geometric distortion that occurs when an image is captured.
- If one attempts to match two different images of the same object, a geometric transformation may be needed:
  - matching remotely sensed images of the same area taken after one year, when the recent image is not taken from the same orientation and/or position.
  - correcting for document skew in OCR, which occurs when an image with an obvious orientation (e.g., a printed page) is scanned, or otherwise captured, at a different orientation.

Geometric Transformations

- Geometric transformations are common in computer graphics, and are often used in image analysis.
- Geometric transforms permit the elimination of geometric distortion that occurs when an image is captured.
- If one attempts to match two different images of the same object, a geometric transformation may be needed:
  - matching remotely sensed images of the same area taken after one year, when the recent image is not taken from the same orientation and/or position.
  - correcting for document skew in OCR, which occurs when an image with an obvious orientation (e.g., a printed page) is scanned, or otherwise captured, at a different orientation.

Geometric Transformations

- Geometric transformations are common in computer graphics, and are often used in image analysis.
- Geometric transforms permit the elimination of geometric distortion that occurs when an image is captured.
- If one attempts to match two different images of the same object, a geometric transformation may be needed:
  - matching remotely sensed images of the same area taken after one year, when the recent image is not taken from the same orientation and/or position.
  - correcting for document skew in OCR, which occurs when an image with an obvious orientation (e.g., a printed page) is scanned, or otherwise captured, at a different orientation.

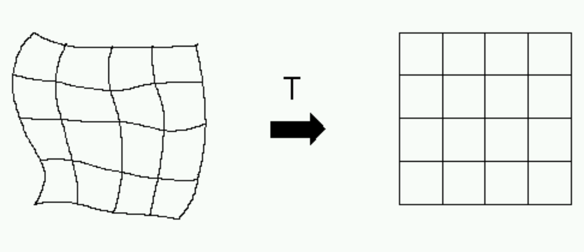
Geometric Transformations

- Geometric transformations are common in computer graphics, and are often used in image analysis.
- Geometric transforms permit the elimination of geometric distortion that occurs when an image is captured.
- If one attempts to match two different images of the same object, a geometric transformation may be needed:
  - matching remotely sensed images of the same area taken after one year, when the recent image is not taken from the same orientation and/or position.
  - correcting for document skew in OCR, which occurs when an image with an obvious orientation (e.g., a printed page) is scanned, or otherwise captured, at a different orientation.

Geometric Transformations

- A geometric transform is a vector function T that maps the pixel (x, y) to a new position (x', y'),

$$x' = T_x(x, y) \quad y' = T_y(x, y) \tag{17}$$



Geometric Transformations

- The transformation equations are
  - either known in advance,
  - or can be determined from known original and transformed images,
  - or can be estimated from known obvious orientations (e.g., OCR applications).
  - Several pixels in both images with known correspondence are used to derive the unknown transformation.

Geometric Transformations

- The transformation equations are
  - either known in advance,
  - or can be determined from known original and transformed images,
  - or can be estimated from known obvious orientations (e.g., OCR applications).
  - Several pixels in both images with known correspondence are used to derive the unknown transformation.

Geometric Transformations

- The transformation equations are
  - either known in advance,
  - or can be determined from known original and transformed images,
  - or can be estimated from known obvious orientations (e.g., OCR applications).
  - Several pixels in both images with known correspondence are used to derive the unknown transformation.

Geometric Transformations

- The transformation equations are
  - either known in advance,
  - or can be determined from known original and transformed images,
  - or can be estimated from known obvious orientations (e.g., OCR applications).
  - Several pixels in both images with known correspondence are used to derive the unknown transformation.

Geometric Transformations

- The transformation equations are
  - either known in advance,
  - or can be determined from known original and transformed images,
  - or can be estimated from known obvious orientations (e.g., OCR applications).
  - Several pixels in both images with known correspondence are used to derive the unknown transformation.

Geometric Transformations

- A geometric transform consists of two basic steps:
  1. determining the pixel co-ordinate transformation
     - mapping of the co-ordinates of the input image pixel to the point in the output image.
     - the output point co-ordinates should be computed as continuous values (real numbers) as the position does not necessarily match the digital grid after the transform.
  2. determining the brightness of the points in the digital grid.

Geometric Transformations

- A geometric transform consists of two basic steps:
  1. determining the pixel co-ordinate transformation
     - mapping of the co-ordinates of the input image pixel to the point in the output image.
     - the output point co-ordinates should be computed as continuous values (real numbers) as the position does not necessarily match the digital grid after the transform.
  2. determining the brightness of the points in the digital grid.

Geometric Transformations

- A geometric transform consists of two basic steps:
  1. determining the pixel co-ordinate transformation
     - mapping of the co-ordinates of the input image pixel to the point in the output image.
     - the output point co-ordinates should be computed as continuous values (real numbers) as the position does not necessarily match the digital grid after the transform.
  2. determining the brightness of the points in the digital grid.

Geometric Transformations

- A geometric transform consists of two basic steps:
  1. determining the pixel co-ordinate transformation
     - mapping of the co-ordinates of the input image pixel to the point in the output image.
     - the output point co-ordinates should be computed as continuous values (real numbers) as the position does not necessarily match the digital grid after the transform.
  2. determining the brightness of the points in the digital grid.

Geometric Transformations

- A geometric transform consists of two basic steps:
  1. determining the pixel co-ordinate transformation
     - mapping of the co-ordinates of the input image pixel to the point in the output image.
     - the output point co-ordinates should be computed as continuous values (real numbers) as the position does not necessarily match the digital grid after the transform.
  2. determining the brightness of the points in the digital grid.

Geometric Transformations

- The brightness values are usually computed as an interpolation of the brightnesses of several points in the neighborhood.
- This idea enables the classification of geometric transformation among other pre-processing techniques, the criterion being that only the neighborhood of a processed pixel is needed for the calculation.
- Geometric transformations are on the boundary between point and local operations.

Geometric Transformations

- The brightness values are usually computed as an interpolation of the brightnesses of several points in the neighborhood.
- This idea enables the classification of geometric transformation among other pre-processing techniques, the criterion being that only the neighborhood of a processed pixel is needed for the calculation.
- Geometric transformations are on the boundary between point and local operations.

Geometric Transformations

- The brightness values are usually computed as an interpolation of the brightnesses of several points in the neighborhood.
- This idea enables the classification of geometric transformation among other pre-processing techniques, the criterion being that only the neighborhood of a processed pixel is needed for the calculation.
- Geometric transformations are on the boundary between point and local operations.

Geometric Transformations: Areas of Images

- A geometric transformation applied to the whole image may change the co-ordinate system, and a Jacobean J provides information about how the co-ordinate system changes

$$J(x, y) = \frac{\partial(x', y')}{\partial(x, y)} \tag{18}$$

- The area of the image is invariant if and only if $|J| = 1$.

Geometric Transformations: Areas of Images

- A geometric transformation applied to the whole image may change the co-ordinate system, and a Jacobean J provides information about how the co-ordinate system changes

$$J(x, y) = \frac{\partial(x', y')}{\partial(x, y)} \tag{18}$$

- The area of the image is invariant if and only if $|J| = 1$.

Outline

Polynomial Approximation

- The transformation is usually approximated by a polynomial equation (of degree m)

$$x' = \sum_{r=0}^{m} \sum_{k=0}^{m-r} a_{rk} x^r y^k \tag{19}$$

$$y' = \sum_{r=0}^{m} \sum_{k=0}^{m-r} b_{rk} x^r y^k . \tag{20}$$

- This transform is linear with respect to the coefficients a_{rk} and b_{rk}.
- If pairs of corresponding points (x, y), (x', y') in both images are known, it is possible to determine a_{rk} and b_{rk} by solving a set of linear equations.
- More points than coefficients are usually used to get robustness. The mean square method (least squares fitting) is often used.

Polynomial Approximation

- The transformation is usually approximated by a polynomial equation (of degree *m*)

$$x' = \sum_{r=0}^{m} \sum_{k=0}^{m-r} a_{rk} x^r y^k \tag{19}$$

$$y' = \sum_{r=0}^{m} \sum_{k=0}^{m-r} b_{rk} x^r y^k . \tag{20}$$

- This transform is linear with respect to the coefficients a_{rk} and b_{rk}.
- If pairs of corresponding points (x, y), (x', y') in both images are known, it is possible to determine a_{rk} and b_{rk} by solving a set of linear equations.
- More points than coefficients are usually used to get robustness. The mean square method (least squares fitting) is often used.

Polynomial Approximation

- The transformation is usually approximated by a polynomial equation (of degree *m*)

$$x' = \sum_{r=0}^{m} \sum_{k=0}^{m-r} a_{rk} x^r y^k \tag{19}$$

$$y' = \sum_{r=0}^{m} \sum_{k=0}^{m-r} b_{rk} x^r y^k . \tag{20}$$

- This transform is linear with respect to the coefficients a_{rk} and b_{rk}.
- If pairs of corresponding points (x, y), (x', y') in both images are known, it is possible to determine a_{rk} and b_{rk} by solving a set of linear equations.
- More points than coefficients are usually used to get robustness. The mean square method (least squares fitting) is often used.

Polynomial Approximation

- The transformation is usually approximated by a polynomial equation (of degree *m*)

$$x' = \sum_{r=0}^{m} \sum_{k=0}^{m-r} a_{rk} x^r y^k \tag{19}$$

$$y' = \sum_{r=0}^{m} \sum_{k=0}^{m-r} b_{rk} x^r y^k . \tag{20}$$

- This transform is linear with respect to the coefficients a_{rk} and b_{rk}.
- If pairs of corresponding points (x, y), (x', y') in both images are known, it is possible to determine a_{rk} and b_{rk} by solving a set of linear equations.
- More points than coefficients are usually used to get robustness. The mean square method (least squares fitting) is often used.

Polynomial Approximation: Orders

- If the geometric transformation does not change rapidly depending on position in the image, low order approximating polynomials, $m = 2$ or $m = 3$, are used, needing at least 6 or 10 pairs of corresponding points.
- The corresponding points should be distributed in the image in a way that can express the geometric transformation — usually they are spread uniformly.
- The higher the degree of the approximating polynomial, the more sensitive to the distribution of the pairs of corresponding points the geometric transform.

Polynomial Approximation: Orders

- If the geometric transformation does not change rapidly depending on position in the image, low order approximating polynomials, $m = 2$ or $m = 3$, are used, needing at least 6 or 10 pairs of corresponding points.
- The corresponding points should be distributed in the image in a way that can express the geometric transformation — usually they are spread uniformly.
- The higher the degree of the approximating polynomial, the more sensitive to the distribution of the pairs of corresponding points the geometric transform.

Polynomial Approximation: Orders

- If the geometric transformation does not change rapidly depending on position in the image, low order approximating polynomials, $m = 2$ or $m = 3$, are used, needing at least 6 or 10 pairs of corresponding points.
- The corresponding points should be distributed in the image in a way that can express the geometric transformation — usually they are spread uniformly.
- The higher the degree of the approximating polynomial, the more sensitive to the distribution of the pairs of corresponding points the geometric transform.

Bilinear Transformations

- In practice, the geometric transform is often approximated by `bilinear transformations`:

$$x' = a_0 + a_1 x + a_2 y + a_3 xy, \tag{21}$$

$$y' = b_0 + b_1 x + b_2 y + b_3 xy. \tag{22}$$

- 4 pairs of corresponding points are sufficient to find transformation coefficients.

Bilinear Transformations

- In practice, the geometric transform is often approximated by `bilinear transformations`:

$$x' = a_0 + a_1 x + a_2 y + a_3 xy, \quad (21)$$

$$y' = b_0 + b_1 x + b_2 y + b_3 xy. \quad (22)$$

- 4 pairs of corresponding points are sufficient to find transformation coefficients.

Affine Transformations

- Even simpler is the `affine transformation` for which three pairs of corresponding points are sufficient to find the coefficients:

$$x' = a_0 + a_1 x + a_2 y, \tag{23}$$

$$y' = b_0 + b_1 x + b_2 y. \tag{24}$$

- The affine transformation includes typical geometric transformations such as rotation, translation, scaling and skewing (shear).

Affine Transformations

- Even simpler is the `affine transformation` for which three pairs of corresponding points are sufficient to find the coefficients:

$$x' = a_0 + a_1 x + a_2 y, \tag{23}$$

$$y' = b_0 + b_1 x + b_2 y. \tag{24}$$

- The affine transformation includes typical geometric transformations such as rotation, translation, scaling and skewing (shear).

Rotations and Scaling

- Rotation by the angle ϕ

$$x' = x\cos\phi + y\sin\phi \quad (25)$$

$$y' = -x\sin\phi + y\cos\phi \quad (26)$$

$$J = 1 \quad (27)$$

- Change of scale a in the x-axis and b in the y-axis

$$x' = ax \quad (28)$$

$$y' = by \quad (29)$$

$$J = ab \quad (30)$$

Rotations and Scaling

- Rotation by the angle ϕ

$$x' = x \cos\phi + y \sin\phi \tag{25}$$
$$y' = -x \sin\phi + y \cos\phi \tag{26}$$
$$J = 1 \tag{27}$$

- Change of scale a in the x-axis and b in the y-axis

$$x' = ax \tag{28}$$
$$y' = by \tag{29}$$
$$J = ab \tag{30}$$

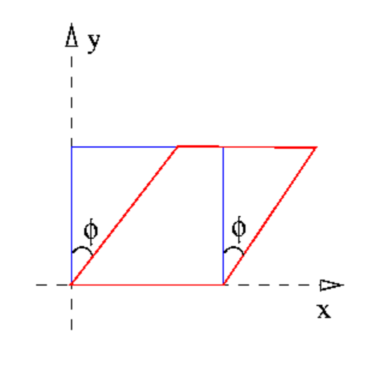
Skew

- Skew by the angel ϕ

$$x' = x + y \tan\phi \quad (31)$$

$$y' = y \quad (32)$$

$$J = 1 \quad (33)$$



Discussions

- It is possible to approximate complex geometric transformations (distortion) by partitioning an image into smaller rectangular sub-images.
- For each sub-image, a simple geometric transformation, such as the affine, is estimated using pairs of corresponding pixels.
- The geometric transformation (distortion) is then performed separately in each sub-image.

Discussions

- It is possible to approximate complex geometric transformations (distortion) by partitioning an image into smaller rectangular sub-images.
- For each sub-image, a simple geometric transformation, such as the affine, is estimated using pairs of corresponding pixels.
- The geometric transformation (distortion) is then performed separately in each sub-image.

Discussions

- It is possible to approximate complex geometric transformations (distortion) by partitioning an image into smaller rectangular sub-images.
- For each sub-image, a simple geometric transformation, such as the affine, is estimated using pairs of corresponding pixels.
- The geometric transformation (distortion) is then performed separately in each sub-image.

Outline

Pixel Brightness Transformations
- Position dependent brightness correction
- Grey scale transformation
  - Windows and level
  - Histogram stretching
  - Histogram Equalization
  - Histogram Matching

Geometric Transformations
- Pixel Co-ordinate Transformations
  - Polynomial Approximation
  - Bilinear Transformations
  - Important Transformations
- Brightness Interpolation
  - Nearest Neighbor Interpolation
  - Bilinear Interpolation
  - Bi-cubic interpolation

Local pre-processing
- Image smoothing
  - Averaging
  - Averaging with Data Validity
  - Averaging According to Inverse Gradient

Brightness Interpolation

- Assume that the planar transformation has been accomplished, and new point co-ordinates (x', y') were obtained.
- The position of the pixel transformed does not in general fit the discrete grid of the output image.
- Values on the integer grid are needed.
- Each pixel value in the output image can be obtained by brightness interpolation of some neighboring non-integer samples, transformed from the input image.

Brightness Interpolation

- Assume that the planar transformation has been accomplished, and new point co-ordinates (x', y') were obtained.
- The position of the pixel transformed does not in general fit the discrete grid of the output image.
- Values on the integer grid are needed.
- Each pixel value in the output image can be obtained by brightness interpolation of some neighboring non-integer samples, transformed from the input image.

Brightness Interpolation

- Assume that the planar transformation has been accomplished, and new point co-ordinates (x', y') were obtained.
- The position of the pixel transformed does not in general fit the discrete grid of the output image.
- Values on the integer grid are needed.
- Each pixel value in the output image can be obtained by brightness interpolation of some neighboring non-integer samples, transformed from the input image.

Brightness Interpolation

- Assume that the planar transformation has been accomplished, and new point co-ordinates (x', y') were obtained.
- The position of the pixel transformed does not in general fit the discrete grid of the output image.
- Values on the integer grid are needed.
- Each pixel value in the output image can be obtained by brightness interpolation of some neighboring non-integer samples, transformed from the input image.

Brightness Dual Interpolation

- The brightness interpolation problem is usually expressed in a dual way:
  - by determining the brightness of the original point in the input image that corresponds to the point in the output image lying on the discrete raster.

Brightness Dual Interpolation

- The brightness interpolation problem is usually expressed in a dual way:
  - by determining the brightness of the original point in the input image that corresponds to the point in the output image lying on the discrete raster.

Brightness Dual Interpolation

- Assume that we wish to compute the brightness value of the pixel (x', y') in the output image where x' and y' lie on the discrete grid.
- The co-ordinates of the point (x, y) in the original image can be obtained by inverting the transformation

$$(x, y) = T^{-1}(x', y'). \tag{34}$$

- In general the real co-ordinates (x, y) after inverse transformation do not fit the input image discrete grid, and so brightness is not known.

Brightness Dual Interpolation

- Assume that we wish to compute the brightness value of the pixel (x', y') in the output image where x' and y' lie on the discrete grid.
- The co-ordinates of the point (x, y) in the original image can be obtained by inverting the transformation

$$(x, y) = T^{-1}(x', y'). \tag{34}$$

- In general the real co-ordinates (x, y) after inverse transformation do not fit the input image discrete grid, and so brightness is not known.

Brightness Dual Interpolation

- Assume that we wish to compute the brightness value of the pixel (x', y') in the output image where x' and y' lie on the discrete grid.
- The co-ordinates of the point (x, y) in the original image can be obtained by inverting the transformation

$$(x, y) = T^{-1}(x', y'). \tag{34}$$

- In general the real co-ordinates (x, y) after inverse transformation do not fit the input image discrete grid, and so brightness is not known.

Brightness Interpolation or Image Resampling

- To get the brightness value of the point (x, y) the input image is re-sampled or interpolated:

$$f_n(x,y) = \sum_{l=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} g_s(l\Delta x, k\Delta y) h_n(x - l\Delta x, y - k\Delta y) \tag{35}$$

  where $f_n(x, y)$ is the result of interpolation and h_n is the interpolation kernel. n distinguishes different interpolation methods.
- Usually, a small neighborhood is used, outside which h_n is zero.

Brightness Interpolation or Image Resampling

- To get the brightness value of the point (x, y) the input image is re-sampled or interpolated:

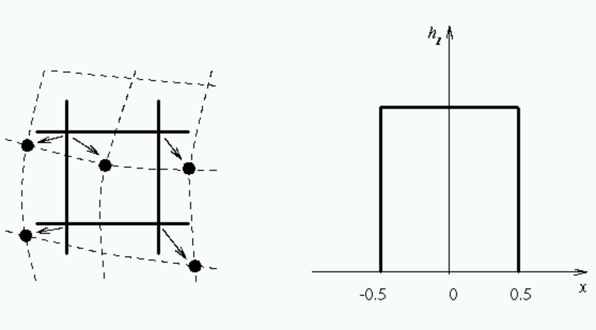
$$f_n(x,y) = \sum_{l=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} g_s(l\Delta x, k\Delta y) h_n(x - l\Delta x, y - k\Delta y) \tag{35}$$

  where $f_n(x, y)$ is the result of interpolation and h_n is the interpolation kernel. n distinguishes different interpolation methods.

- Usually, a small neighborhood is used, outside which h_n is zero.

Nearest Neighbor Interpolation

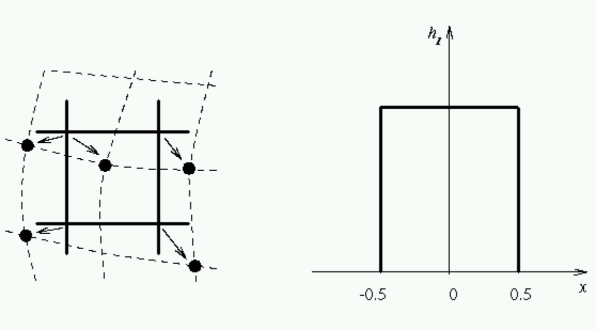
- Assign to the point (x, y) the brightness value of the nearest point g in the discrete raster.



- The right side of the above figure shows how the new brightness is assigned.
- Dashed lines show how the inverse planar transformation maps the grids of the output image into the input image — solid lines show the grids of the input image.

Nearest Neighbor Interpolation

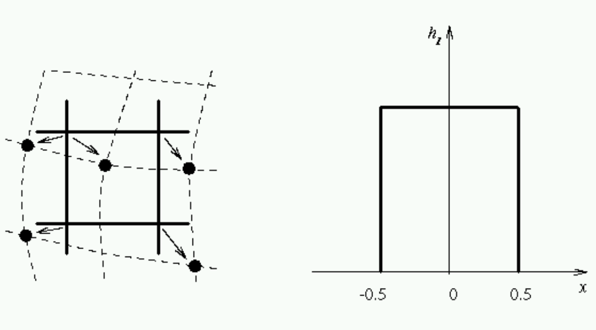
- Assign to the point (x, y) the brightness value of the nearest point g in the discrete raster.



- The right side of the above figure shows how the new brightness is assigned.
- Dashed lines show how the inverse planar transformation maps the grids of the output image into the input image — solid lines show the grids of the input image.

Nearest Neighbor Interpolation

- Assign to the point (x, y) the brightness value of the nearest point g in the discrete raster.



- The right side of the above figure shows how the new brightness is assigned.
- Dashed lines show how the inverse planar transformation maps the grids of the output image into the input image — solid lines show the grids of the input image.

Nearest Neighbor Interpolation

- Nearest neighbor interpolation is given by

$$f_1(x, y) = g_s(\text{round}(x), \text{round}(y)). \qquad (36)$$

- The interpolation kernel h_1 as in (35) is

$$h_1(x, y) = h_1^1(x) h_1^1(y), \qquad (37)$$

where,

$$h_1^1(t) = \begin{cases} 1, & \text{if } t \in [-0.5, 0.5], \\ 0, & \text{otherwise.} \end{cases} \qquad (38)$$

- The position error of the nearest neighborhood interpolation is at most half a pixel.
- This error is perceptible on objects with straight line boundaries that may appear step-like after the transformation.

Nearest Neighbor Interpolation

- Nearest neighbor interpolation is given by

$$f_1(x, y) = g_s(\text{round}(x), \text{round}(y)). \tag{36}$$

- The interpolation kernel h_1 as in (35) is

$$h_1(x, y) = h_1^1(x)h_1^1(y), \tag{37}$$

where,

$$h_1^1(t) = \begin{cases} 1, & \text{if } t \in [-0.5, 0.5], \\ 0, & \text{otherwise.} \end{cases} \tag{38}$$

- The position error of the nearest neighborhood interpolation is at most half a pixel.
- This error is perceptible on objects with straight line boundaries that may appear step-like after the transformation.

Nearest Neighbor Interpolation

- Nearest neighbor interpolation is given by

$$f_1(x, y) = g_s(\text{round}(x), \text{round}(y)). \qquad (36)$$

- The interpolation kernel h_1 as in (35) is

$$h_1(x, y) = h_1^1(x) h_1^1(y), \qquad (37)$$

where,

$$h_1^1(t) = \begin{cases} 1, & \text{if } t \in [-0.5, 0.5], \\ 0, & \text{otherwise}. \end{cases} \qquad (38)$$

- The position error of the nearest neighborhood interpolation is at most half a pixel.
- This error is perceptible on objects with straight line boundaries that may appear step-like after the transformation.

Nearest Neighbor Interpolation

- Nearest neighbor interpolation is given by

$$f_1(x, y) = g_s(\text{round}(x), \text{round}(y)). \tag{36}$$

- The interpolation kernel h_1 as in (35) is

$$h_1(x, y) = h_1^1(x)h_1^1(y), \tag{37}$$

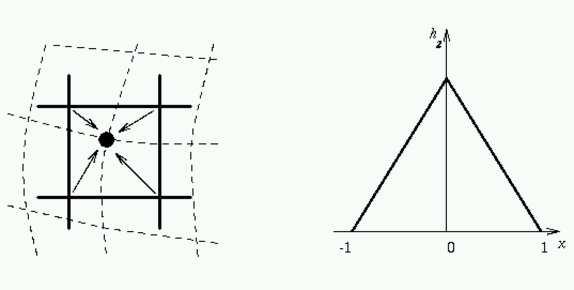
where,

$$h_1^1(t) = \begin{cases} 1, & \text{if } t \in [-0.5, 0.5], \\ 0, & \text{otherwise.} \end{cases} \tag{38}$$

- The position error of the nearest neighborhood interpolation is at most half a pixel.
- This error is perceptible on objects with straight line boundaries that may appear step-like after the transformation.

Bilinear Interpolation

- Bilinear interpolation explores four points neighboring the point (x, y), and assumes that the brightness function is bilinear in this neighborhood.



Bilinear Interpolation

Bilinear interpolation is given by

$$\begin{aligned}
f_2(x, y) =&(1-a)(1-b)g_s(l, k) + a(1-b)g_s(l+1, k) \\
&(1-a)bg_s(l, k+1) + abg_s(l+1, k+1) \\
=&g_s(l, k) \\
&+ (g_s(l+1, k) - g_s(l, k))a \\
&+ (g_s(l, k+1) - g_s(l, k))b \\
&+ (g_s(l, k) + g_s(l+1, k+1) - g_s(l+1, k) - g_s(l, k+1))ab,
\end{aligned}$$

where

$$l = \text{floor}(x), \quad a = x - l, \tag{39}$$

$$k = \text{floor}(y), \quad b = y - k. \tag{40}$$

Proof of Bilinear Interpolation Formula

- By construction

$$x = a \cdot (l+1) + (1-a) \cdot l,$$
$$y = b \cdot (k+1) + (1-b) \cdot k.$$

- Since f_2 is bilinear,

$$f_2(x,k) = (1-a)g_s(l,k) + ag_s(l+1,k)$$
$$f_2(x,k+1) = (1-a)g_s(l,k+1) + ag_s(l+1,k+1).$$

- Then

$$\begin{aligned} f_2(x,y) =& bf_2(x,k+1) + (1-b)f_2(x,k) \\ =& b(1-a)g_s(l,k+1) + bag_s(l+1,k+1) \\ &+ (1-b)(1-a)g_s(l,k) + (1-b)ag_s(l+1,k). \end{aligned}$$

Proof of Bilinear Interpolation Formula

- By construction

$$x = a \cdot (l+1) + (1-a) \cdot l,$$
$$y = b \cdot (k+1) + (1-b) \cdot k.$$

- Since f_2 is bilinear,

$$f_2(x,k) = (1-a)g_s(l,k) + ag_s(l+1,k)$$
$$f_2(x,k+1) = (1-a)g_s(l,k+1) + ag_s(l+1,k+1).$$

- Then

$$\begin{aligned} f_2(x,y) =& bf_2(x,k+1) + (1-b)f_2(x,k) \\ =& b(1-a)g_s(l,k+1) + bag_s(l+1,k+1) \\ &+ (1-b)(1-a)g_s(l,k) + (1-b)ag_s(l+1,k). \end{aligned}$$

Proof of Bilinear Interpolation Formula

- By construction

$$x = a \cdot (l + 1) + (1 - a) \cdot l,$$
$$y = b \cdot (k + 1) + (1 - b) \cdot k.$$

- Since f_2 is bilinear,

$$f_2(x, k) = (1 - a)g_s(l, k) + ag_s(l + 1, k)$$
$$f_2(x, k + 1) = (1 - a)g_s(l, k + 1) + ag_s(l + 1, k + 1).$$

- Then

$$\begin{aligned} f_2(x, y) =& bf_2(x, k + 1) + (1 - b)f_2(x, k) \\ =& b(1 - a)g_s(l, k + 1) + bag_s(l + 1, k + 1) \\ &+ (1 - b)(1 - a)g_s(l, k) + (1 - b)ag_s(l + 1, k). \end{aligned}$$

Bilinear Interpolation

- The interpolation kernel h_2 is

$$h_2(x, y) = h_2^1(x) h_2^1(y), \tag{41}$$

where

$$h_2^1(t) = h_1^1 * h_1^1(t) = \begin{cases} 1 - t, & \text{if } t \in [0, 1], \\ t + 1, & \text{if } t \in [-1, 0], \\ 0, & \text{otherwise.} \end{cases} \tag{42}$$

- Linear interpolation can cause a small decrease in resolution and blurring due to its averaging nature.
- The problem of step like straight boundaries with the nearest neighborhood interpolation is reduced.

Bilinear Interpolation

- The interpolation kernel h_2 is

$$h_2(x, y) = h_2^1(x) h_2^1(y), \tag{41}$$

where

$$h_2^1(t) = h_1^1 * h_1^1(t) = \begin{cases} 1 - t, & \text{if } t \in [0, 1], \\ t + 1, & \text{if } t \in [-1, 0], \\ 0, & \text{otherwise.} \end{cases} \tag{42}$$

- Linear interpolation can cause a small decrease in resolution and blurring due to its averaging nature.
- The problem of step like straight boundaries with the nearest neighborhood interpolation is reduced.

Bilinear Interpolation

- The interpolation kernel h_2 is

$$h_2(x, y) = h_2^1(x) h_2^1(y), \quad (41)$$

where

$$h_2^1(t) = h_1^1 * h_1^1(t) = \begin{cases} 1 - t, & \text{if } t \in [0, 1], \\ t + 1, & \text{if } t \in [-1, 0], \\ 0, & \text{otherwise.} \end{cases} \quad (42)$$

- Linear interpolation can cause a small decrease in resolution and blurring due to its averaging nature.
- The problem of step like straight boundaries with the nearest neighborhood interpolation is reduced.

Bi-cubic Interpolation

- Bi-cubic interpolation improves the model of the brightness function by approximating it locally by a bicubic polynomial surface;
- 16 neighboring points are used for interpolation.
- interpolation kernel ('Mexican hat') is defined via

$$h_3^1(t) = \begin{cases} 1 - 2|t|^2 + |t|^3, & \text{if } |t| < 1 \\ 4 - 8|t| + 5|t|^2 - |t|^3, & \text{if } 1 \leq |t| < 2 \\ 0, & \text{otherwise} \end{cases} \tag{43}$$

by

$$h_3(x, y) = h_3^1(x) h_3^1(y). \tag{44}$$

Bi-cubic Interpolation

- Bi-cubic interpolation improves the model of the brightness function by approximating it locally by a bicubic polynomial surface;
- 16 neighboring points are used for interpolation.
- interpolation kernel ('Mexican hat') is defined via

$$h_3^1(t) = \begin{cases} 1 - 2|t|^2 + |t|^3, & \text{if } |t| < 1 \\ 4 - 8|t| + 5|t|^2 - |t|^3, & \text{if } 1 \leq |t| < 2 \\ 0, & \text{otherwise} \end{cases} \quad (43)$$

by

$$h_3(x, y) = h_3^1(x) h_3^1(y). \quad (44)$$

Bi-cubic Interpolation

- Bi-cubic interpolation improves the model of the brightness function by approximating it locally by a bicubic polynomial surface;
- 16 neighboring points are used for interpolation.
- interpolation kernel ('Mexican hat') is defined via

$$h_3^1(t) = \begin{cases} 1 - 2|t|^2 + |t|^3, & \text{if } |t| < 1 \\ 4 - 8|t| + 5|t|^2 - |t|^3, & \text{if } 1 \leq |t| < 2 \\ 0, & \text{otherwise} \end{cases} \quad (43)$$

by

$$h_3(x, y) = h_3^1(x) h_3^1(y). \quad (44)$$

Example

- The matlab script from `visionbook` is visionbook/05Preproc/imgeomt_demo.m.

Discussions

- Bicubic interpolation does not suffer from the step-like boundary problem of nearest neighborhood interpolation, and copes with linear interpolation blurring as well.
- Bicubic interpolation is often used in raster displays that enable zooming to an an arbitrary scale.
- Bicubic interpolation preserves fine details in the image very well.

Discussions

- Bicubic interpolation does not suffer from the step-like boundary problem of nearest neighborhood interpolation, and copes with linear interpolation blurring as well.
- Bicubic interpolation is often used in raster displays that enable zooming to an an arbitrary scale.
- Bicubic interpolation preserves fine details in the image very well.

Discussions

- Bicubic interpolation does not suffer from the step-like boundary problem of nearest neighborhood interpolation, and copes with linear interpolation blurring as well.
- Bicubic interpolation is often used in raster displays that enable zooming to an an arbitrary scale.
- Bicubic interpolation preserves fine details in the image very well.

Local Pre-processing

- Pre-processing methods use a small neighborhood of a pixel in an input image to get a new brightness value in the output image.
- Such pre-processing operations are called also filtration (or filtering) if signal processing terminology is used.
- Local pre-processing methods can be divided into the two groups according to the goal of the processing:
  - smoothing operators
  - gradient operators

Local Pre-processing

- Pre-processing methods use a small neighborhood of a pixel in an input image to get a new brightness value in the output image.
- Such pre-processing operations are called also filtration (or filtering) if signal processing terminology is used.
- Local pre-processing methods can be divided into the two groups according to the goal of the processing:
  - smoothing operators
  - gradient operators

Local Pre-processing

- Pre-processing methods use a small neighborhood of a pixel in an input image to get a new brightness value in the output image.
- Such pre-processing operations are called also filtration (or filtering) if signal processing terminology is used.
- Local pre-processing methods can be divided into the two groups according to the goal of the processing:
  - smoothing operators
  - gradient operators

Local Pre-processing

- Pre-processing methods use a small neighborhood of a pixel in an input image to get a new brightness value in the output image.
- Such pre-processing operations are called also filtration (or filtering) if signal processing terminology is used.
- Local pre-processing methods can be divided into the two groups according to the goal of the processing:
  - smoothing operators
  - gradient operators

Local Pre-processing

- Pre-processing methods use a small neighborhood of a pixel in an input image to get a new brightness value in the output image.
- Such pre-processing operations are called also filtration (or filtering) if signal processing terminology is used.
- Local pre-processing methods can be divided into the two groups according to the goal of the processing:
  - `smoothing operators`
  - `gradient operators`

Smoothing Operators

- They aim to suppress noise or other small fluctuations in the image.
- Smoothing is equivalent to the suppression of high frequencies in the frequency domain.
- Unfortunately, smoothing also blurs all sharp edges that bear important information about the image.
- If objects are rather large, an image can be enhanced by smoothing of small degradations.
- Smoothing operators will benefit if some general knowledge about image degradation is available; this might, e.g., be statistical parameters of the noise.

Smoothing Operators

- They aim to suppress noise or other small fluctuations in the image.
- Smoothing is equivalent to the suppression of high frequencies in the frequency domain.
- Unfortunately, smoothing also blurs all sharp edges that bear important information about the image.
- If objects are rather large, an image can be enhanced by smoothing of small degradations.
- Smoothing operators will benefit if some general knowledge about image degradation is available; this might, e.g., be statistical parameters of the noise.

Smoothing Operators

- They aim to suppress noise or other small fluctuations in the image.
- Smoothing is equivalent to the suppression of high frequencies in the frequency domain.
- Unfortunately, smoothing also blurs all sharp edges that bear important information about the image.
- If objects are rather large, an image can be enhanced by smoothing of small degradations.
- Smoothing operators will benefit if some general knowledge about image degradation is available; this might, e.g., be statistical parameters of the noise.

Smoothing Operators

- They aim to suppress noise or other small fluctuations in the image.
- Smoothing is equivalent to the suppression of high frequencies in the frequency domain.
- Unfortunately, smoothing also blurs all sharp edges that bear important information about the image.
- If objects are rather large, an image can be enhanced by smoothing of small degradations.
- Smoothing operators will benefit if some general knowledge about image degradation is available; this might, e.g., be statistical parameters of the noise.

Smoothing Operators

- They aim to suppress noise or other small fluctuations in the image.
- Smoothing is equivalent to the suppression of high frequencies in the frequency domain.
- Unfortunately, smoothing also blurs all sharp edges that bear important information about the image.
- If objects are rather large, an image can be enhanced by smoothing of small degradations.
- Smoothing operators will benefit if some general knowledge about image degradation is available; this might, e.g., be statistical parameters of the noise.

Gradient Operators

- They are based on local derivatives of the image function.
- Derivatives are bigger at locations of the image where the image function undergoes rapid changes.
- The aim of gradient operators is to indicate such locations in the image.
- Gradient operators have a similar effect as suppressing low frequencies in the frequency domain.
- Noise is often high frequency in nature; unfortunately, if a gradient operator is applied to an image, the noise level increases simultaneously.

Gradient Operators

- They are based on local derivatives of the image function.
- Derivatives are bigger at locations of the image where the image function undergoes rapid changes.
- The aim of gradient operators is to indicate such locations in the image.
- Gradient operators have a similar effect as suppressing low frequencies in the frequency domain.
- Noise is often high frequency in nature; unfortunately, if a gradient operator is applied to an image, the noise level increases simultaneously.

Gradient Operators

- They are based on local derivatives of the image function.
- Derivatives are bigger at locations of the image where the image function undergoes rapid changes.
- The aim of gradient operators is to indicate such locations in the image.
- Gradient operators have a similar effect as suppressing low frequencies in the frequency domain.
- Noise is often high frequency in nature; unfortunately, if a gradient operator is applied to an image, the noise level increases simultaneously.

Gradient Operators

- They are based on local derivatives of the image function.
- Derivatives are bigger at locations of the image where the image function undergoes rapid changes.
- The aim of gradient operators is to indicate such locations in the image.
- Gradient operators have a similar effect as suppressing low frequencies in the frequency domain.
- Noise is often high frequency in nature; unfortunately, if a gradient operator is applied to an image, the noise level increases simultaneously.

Gradient Operators

- They are based on local derivatives of the image function.
- Derivatives are bigger at locations of the image where the image function undergoes rapid changes.
- The aim of gradient operators is to indicate such locations in the image.
- Gradient operators have a similar effect as suppressing low frequencies in the frequency domain.
- Noise is often high frequency in nature; unfortunately, if a gradient operator is applied to an image, the noise level increases simultaneously.

Local Pre-processing: Variants

- Clearly, smoothing and gradient operators have conflicting aims.
- Some pre-processing algorithms solve this problem and permit smoothing and edge enhancement simultaneously.

- Another classification of local pre-processing methods is according to the transformation properties.
- Linear and nonlinear transformations can be distinguished.

Local Pre-processing: Variants

- Clearly, smoothing and gradient operators have conflicting aims.
- Some pre-processing algorithms solve this problem and permit smoothing and edge enhancement simultaneously.

- Another classification of local pre-processing methods is according to the transformation properties.
- Linear and nonlinear transformations can be distinguished.

Local Pre-processing: Variants

- Clearly, smoothing and gradient operators have conflicting aims.
- Some pre-processing algorithms solve this problem and permit smoothing and edge enhancement simultaneously.

- Another classification of local pre-processing methods is according to the transformation properties.
- Linear and nonlinear transformations can be distinguished.

Local Pre-processing: Variants

- Clearly, smoothing and gradient operators have conflicting aims.
- Some pre-processing algorithms solve this problem and permit smoothing and edge enhancement simultaneously.

- Another classification of local pre-processing methods is according to the transformation properties.
- Linear and nonlinear transformations can be distinguished.

Local Pre-processing: Convolution Mask I

- Linear operations calculate the resulting value in the output image pixel $g(i, j)$ as a linear combination of brightnesses in a local neighborhood of the pixel $f(i, j)$ in the input image.
- The contribution of the pixels in the neighborhood is weighted by coefficients h

$$f(i, j) = \sum_{(m,n)\in\mathcal{O}} \sum h(i - m, j - n)g(m, n) \quad (45)$$

- The above equation is equivalent to discrete convolution with the kernel h, that is called a `convolution mask`.

Local Pre-processing: Convolution Mask I

- Linear operations calculate the resulting value in the output image pixel $g(i,j)$ as a linear combination of brightnesses in a local neighborhood of the pixel $f(i,j)$ in the input image.
- The contribution of the pixels in the neighborhood is weighted by coefficients h

$$f(i,j) = \sum_{(m,n)\in\mathcal{O}} \sum h(i-m, j-n) g(m,n) \qquad (45)$$

- The above equation is equivalent to discrete convolution with the kernel h, that is called a `convolution mask`.

Local Pre-processing: Convolution Mask I

- Linear operations calculate the resulting value in the output image pixel $g(i, j)$ as a linear combination of brightnesses in a local neighborhood of the pixel $f(i, j)$ in the input image.
- The contribution of the pixels in the neighborhood is weighted by coefficients h

$$f(i,j) = \sum\sum_{(m,n)\in\mathcal{O}} h(i-m, j-n)g(m,n) \qquad (45)$$

- The above equation is equivalent to discrete convolution with the kernel h, that is called a `convolution mask`.

Local Pre-processing: Convolution Mask II

- Rectangular neighborhoods $\mathcal{O}$ are often used with an odd number of pixels in rows and columns, enabling the specification of the central pixel of the neighborhood.
- The choice of the local transformation, size, and shape of the neighborhood $\mathcal{O}$ depends strongly on the size of objects in the processed image.
- Convolution-based operators (filters) can be used for smoothing, gradient operators, and line detectors.
- There are methods that enable the speed-up of calculations to ease implementation in hardware — examples are recursive filters or separable filters.

Local Pre-processing: Convolution Mask II

- Rectangular neighborhoods $\mathcal{O}$ are often used with an odd number of pixels in rows and columns, enabling the specification of the central pixel of the neighborhood.
- The choice of the local transformation, size, and shape of the neighborhood $\mathcal{O}$ depends strongly on the size of objects in the processed image.
- Convolution-based operators (filters) can be used for smoothing, gradient operators, and line detectors.
- There are methods that enable the speed-up of calculations to ease implementation in hardware — examples are recursive filters or separable filters.

Local Pre-processing: Convolution Mask II

- Rectangular neighborhoods $\mathcal{O}$ are often used with an odd number of pixels in rows and columns, enabling the specification of the central pixel of the neighborhood.
- The choice of the local transformation, size, and shape of the neighborhood $\mathcal{O}$ depends strongly on the size of objects in the processed image.
- Convolution-based operators (filters) can be used for smoothing, gradient operators, and line detectors.
- There are methods that enable the speed-up of calculations to ease implementation in hardware — examples are recursive filters or separable filters.

Local Pre-processing: Convolution Mask II

- Rectangular neighborhoods $\mathcal{O}$ are often used with an odd number of pixels in rows and columns, enabling the specification of the central pixel of the neighborhood.
- The choice of the local transformation, size, and shape of the neighborhood $\mathcal{O}$ depends strongly on the size of objects in the processed image.
- Convolution-based operators (filters) can be used for smoothing, gradient operators, and line detectors.
- There are methods that enable the speed-up of calculations to ease implementation in hardware — examples are recursive filters or separable filters.

Local Pre-processing: a priori Knowledge

- Local pre-processing methods typically use very little `a priori knowledge` about the image contents.
- It is very difficult to infer this knowledge while an image is being processed, as the known neighborhood $\mathcal{O}$ of the processed pixel is small.

Local Pre-processing: a priori Knowledge

- Local pre-processing methods typically use very little `a priori knowledge` about the image contents.
- It is very difficult to infer this knowledge while an image is being processed, as the known neighborhood $\mathcal{O}$ of the processed pixel is small.

Outline

Pixel Brightness Transformations
- Position dependent brightness correction
- Grey scale transformation
  - Windows and level
  - Histogram stretching
  - Histogram Equalization
  - Histogram Matching

Geometric Transformations
- Pixel Co-ordinate Transformations
  - Polynomial Approximation
  - Bilinear Transformations
  - Important Transformations
- Brightness Interpolation
  - Nearest Neighbor Interpolation
  - Bilinear Interpolation
  - Bi-cubic interpolation

Local pre-processing
- Image smoothing
  - Averaging
  - Averaging with Data Validity
  - Averaging According to Inverse Gradient

Image smoothing

- Image smoothing is to the suppression image noise — it uses redundancy in the image data.
- New pixel value is the averaging of brightness values in some neighborhood $\mathcal{O}$.
- Smoothing could blur sharp edges.
- There are smoothing methods which are edge preserving.
- The average is computed only from those pixels in the neighborhood which have similar properties to the pixel under processing.
- Local image smoothing can effectively eliminate impulsive noise or degradations appearing as thin stripes.
- It does not work well if degradations are large blobs or thick stripes.
- For complicated degradations, image restoration techniques in § **??** can be applied.

Image smoothing

- Image smoothing is to the suppression image noise — it uses redundancy in the image data.
- New pixel value is the averaging of brightness values in some neighborhood $\mathcal{O}$.

- Smoothing could blur sharp edges.
- There are smoothing methods which are edge preserving.
- The average is computed only from those pixels in the neighborhood which have similar properties to the pixel under processing.

- Local image smoothing can effectively eliminate impulsive noise or degradations appearing as thin stripes.
- It does not work well if degradations are large blobs or thick stripes.
- For complicated degradations, image restoration techniques in § **??** can be applied.

Image smoothing

- Image smoothing is to the suppression image noise — it uses redundancy in the image data.
- New pixel value is the averaging of brightness values in some neighborhood $\mathcal{O}$.
- Smoothing could blur sharp edges.
- There are smoothing methods which are edge preserving.
- The average is computed only from those pixels in the neighborhood which have similar properties to the pixel under processing.
- Local image smoothing can effectively eliminate impulsive noise or degradations appearing as thin stripes.
- It does not work well if degradations are large blobs or thick stripes.
- For complicated degradations, image restoration techniques in § **??** can be applied.

Image smoothing

- Image smoothing is to the suppression image noise — it uses redundancy in the image data.
- New pixel value is the averaging of brightness values in some neighborhood $\mathcal{O}$.
- Smoothing could blur sharp edges.
- There are smoothing methods which are edge preserving.
- The average is computed only from those pixels in the neighborhood which have similar properties to the pixel under processing.
- Local image smoothing can effectively eliminate impulsive noise or degradations appearing as thin stripes.
- It does not work well if degradations are large blobs or thick stripes.
- For complicated degradations, image restoration techniques in § **??** can be applied.

Image smoothing

- Image smoothing is to the suppression image noise — it uses redundancy in the image data.
- New pixel value is the averaging of brightness values in some neighborhood $\mathcal{O}$.
- Smoothing could blur sharp edges.
- There are smoothing methods which are edge preserving.
- The average is computed only from those pixels in the neighborhood which have similar properties to the pixel under processing.
- Local image smoothing can effectively eliminate impulsive noise or degradations appearing as thin stripes.
- It does not work well if degradations are large blobs or thick stripes.
- For complicated degradations, image restoration techniques in § **??** can be applied.

Image smoothing

- Image smoothing is to the suppression image noise — it uses redundancy in the image data.
- New pixel value is the averaging of brightness values in some neighborhood $\mathcal{O}$.
- Smoothing could blur sharp edges.
- There are smoothing methods which are edge preserving.
- The average is computed only from those pixels in the neighborhood which have similar properties to the pixel under processing.
- Local image smoothing can effectively eliminate impulsive noise or degradations appearing as thin stripes.
- It does not work well if degradations are large blobs or thick stripes.
- For complicated degradations, image restoration techniques in § **??** can be applied.

Image smoothing

- Image smoothing is to the suppression image noise — it uses redundancy in the image data.
- New pixel value is the averaging of brightness values in some neighborhood $\mathcal{O}$.
- Smoothing could blur sharp edges.
- There are smoothing methods which are edge preserving.
- The average is computed only from those pixels in the neighborhood which have similar properties to the pixel under processing.
- Local image smoothing can effectively eliminate impulsive noise or degradations appearing as thin stripes.
- It does not work well if degradations are large blobs or thick stripes.
- For complicated degradations, image restoration techniques in § **??** can be applied.

Image smoothing

- Image smoothing is to the suppression image noise — it uses redundancy in the image data.
- New pixel value is the averaging of brightness values in some neighborhood $\mathcal{O}$.
- Smoothing could blur sharp edges.
- There are smoothing methods which are edge preserving.
- The average is computed only from those pixels in the neighborhood which have similar properties to the pixel under processing.
- Local image smoothing can effectively eliminate impulsive noise or degradations appearing as thin stripes.
- It does not work well if degradations are large blobs or thick stripes.
- For complicated degradations, image restoration techniques in § **??** can be applied.

Averaging with Images

- Assume that the noise ν at each pixel is an independent random variable with zero mean and standard deviation σ.
- E.g., the same image can be captured for the same static scene several times.
- The average of the same n images $g_1, \cdots, g_n$ with noise $\nu_1, \cdots, \nu_n$, is

$$\frac{g_1 + \cdots + g_n}{n} + \frac{\nu_1 + \cdots + \nu_n}{n}. \tag{46}$$

- The second term describes the effect of the noise,
  - it is again a random value with zero mean and standard deviation $\frac{\sigma}{\sqrt{n}}$.
- The resultant standard deviation is decreased by a factor $\sqrt{n}$.
- Thus if n images of the same scene are available, the smoothing can be accomplished without blurring the image by

$$f(i,j) = \frac{1}{n} \sum_{k=1}^{n} g_k(i,j) \tag{47}$$

Averaging with Images

- Assume that the noise ν at each pixel is an independent random variable with zero mean and standard deviation σ.
- E.g., the same image can be captured for the same static scene several times.
- The average of the same n images $g_1, \cdots, g_n$ with noise $\nu_1, \cdots, \nu_n$, is

$$\frac{g_1 + \cdots + g_n}{n} + \frac{\nu_1 + \cdots + \nu_n}{n}. \tag{46}$$

- The second term describes the effect of the noise,
  - it is again a random value with zero mean and standard deviation $\frac{\sigma}{\sqrt{n}}$.
- The resultant standard deviation is decreased by a factor $\sqrt{n}$.
- Thus if n images of the same scene are available, the smoothing can be accomplished without blurring the image by

$$f(i,j) = \frac{1}{n} \sum_{k=1}^{n} g_k(i,j) \tag{47}$$

Averaging with Images

- Assume that the noise ν at each pixel is an independent random variable with zero mean and standard deviation σ.
- E.g., the same image can be captured for the same static scene several times.
- The average of the same n images $g_1, \cdots, g_n$ with noise $\nu_1, \cdots, \nu_n$, is

$$\frac{g_1 + \cdots + g_n}{n} + \frac{\nu_1 + \cdots + \nu_n}{n}. \tag{46}$$

- The second term describes the effect of the noise,
  - it is again a random value with zero mean and standard deviation $\frac{\sigma}{\sqrt{n}}$.
- The resultant standard deviation is decreased by a factor $\sqrt{n}$.
- Thus if n images of the same scene are available, the smoothing can be accomplished without blurring the image by

$$f(i,j) = \frac{1}{n}\sum_{k=1}^{n} g_k(i,j) \tag{47}$$

Averaging with Images

- Assume that the noise ν at each pixel is an independent random variable with zero mean and standard deviation σ.
- E.g., the same image can be captured for the same static scene several times.
- The average of the same n images $g_1, \cdots, g_n$ with noise $\nu_1, \cdots, \nu_n$, is

$$\frac{g_1 + \cdots + g_n}{n} + \frac{\nu_1 + \cdots + \nu_n}{n}. \tag{46}$$

- The second term describes the effect of the noise,
  - it is again a random value with zero mean and standard deviation $\frac{\sigma}{\sqrt{n}}$.
- The resultant standard deviation is decreased by a factor $\sqrt{n}$.
- Thus if n images of the same scene are available, the smoothing can be accomplished without blurring the image by

$$f(i,j) = \frac{1}{n}\sum_{k=1}^{n} g_k(i,j) \tag{47}$$

Averaging with Images

- Assume that the noise ν at each pixel is an independent random variable with zero mean and standard deviation σ.
- E.g., the same image can be captured for the same static scene several times.
- The average of the same n images $g_1, \cdots, g_n$ with noise $\nu_1, \cdots, \nu_n$, is

$$\frac{g_1 + \cdots + g_n}{n} + \frac{\nu_1 + \cdots + \nu_n}{n}. \tag{46}$$

- The second term describes the effect of the noise,
  - it is again a random value with zero mean and standard deviation $\frac{\sigma}{\sqrt{n}}$.
- The resultant standard deviation is decreased by a factor $\sqrt{n}$.
- Thus if n images of the same scene are available, the smoothing can be accomplished without blurring the image by

$$f(i,j) = \frac{1}{n} \sum_{k=1}^{n} g_k(i,j) \tag{47}$$

Averaging with Images

- Assume that the noise ν at each pixel is an independent random variable with zero mean and standard deviation σ.
- E.g., the same image can be captured for the same static scene several times.
- The average of the same n images $g_1, \cdots, g_n$ with noise $\nu_1, \cdots, \nu_n$, is

$$\frac{g_1 + \cdots + g_n}{n} + \frac{\nu_1 + \cdots + \nu_n}{n}. \tag{46}$$

- The second term describes the effect of the noise,
  - it is again a random value with zero mean and standard deviation $\frac{\sigma}{\sqrt{n}}$.
- The resultant standard deviation is decreased by a factor $\sqrt{n}$.
- Thus if n images of the same scene are available, the smoothing can be accomplished without blurring the image by

$$f(i,j) = \frac{1}{n} \sum_{k=1}^{n} g_k(i,j) \tag{47}$$

Averaging with Images

- Assume that the noise ν at each pixel is an independent random variable with zero mean and standard deviation σ.
- E.g., the same image can be captured for the same static scene several times.
- The average of the same n images $g_1, \cdots, g_n$ with noise $\nu_1, \cdots, \nu_n$, is

$$\frac{g_1 + \cdots + g_n}{n} + \frac{\nu_1 + \cdots + \nu_n}{n}. \tag{46}$$

- The second term describes the effect of the noise,
  - it is again a random value with zero mean and standard deviation $\frac{\sigma}{\sqrt{n}}$.
- The resultant standard deviation is decreased by a factor $\sqrt{n}$.
- Thus if n images of the same scene are available, the smoothing can be accomplished without blurring the image by

$$f(i,j) = \frac{1}{n} \sum_{k=1}^{n} g_k(i,j) \tag{47}$$

Averaging with Pixels

- In many cases only one image with noise is available, and averaging is then realized in a local neighborhood.
- Results are acceptable if the noise is smaller in size than the smallest objects of interest in the image, but blurring of edges is a serious disadvantage.
- In the case of smoothing within a single image, one has to assume that there are no changes in the gray levels of the underlying image data.
- This assumption is clearly violated at locations of image edges, and edge blurring is a direct consequence of violating the assumption.

Averaging with Pixels

- In many cases only one image with noise is available, and averaging is then realized in a local neighborhood.
- Results are acceptable if the noise is smaller in size than the smallest objects of interest in the image, but blurring of edges is a serious disadvantage.
- In the case of smoothing within a single image, one has to assume that there are no changes in the gray levels of the underlying image data.
- This assumption is clearly violated at locations of image edges, and edge blurring is a direct consequence of violating the assumption.

Averaging with Pixels

- In many cases only one image with noise is available, and averaging is then realized in a local neighborhood.
- Results are acceptable if the noise is smaller in size than the smallest objects of interest in the image, but blurring of edges is a serious disadvantage.
- In the case of smoothing within a single image, one has to assume that there are no changes in the gray levels of the underlying image data.
- This assumption is clearly violated at locations of image edges, and edge blurring is a direct consequence of violating the assumption.

Averaging with Pixels

- In many cases only one image with noise is available, and averaging is then realized in a local neighborhood.
- Results are acceptable if the noise is smaller in size than the smallest objects of interest in the image, but blurring of edges is a serious disadvantage.
- In the case of smoothing within a single image, one has to assume that there are no changes in the gray levels of the underlying image data.
- This assumption is clearly violated at locations of image edges, and edge blurring is a direct consequence of violating the assumption.

Averaging with Pixels

- Averaging is a special case of discrete convolution. For a 3×3 neighborhood, the convolution mask h is

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{48}$$

- The significance of the central pixel may be increased,

$$h = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{49}$$

- Larger convolution masks for averaging can be created.

Averaging with Pixels

- Averaging is a special case of discrete convolution. For a 3×3 neighborhood, the convolution mask h is

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (48)$$

- The significance of the central pixel may be increased,

$$h = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad (49)$$

- Larger convolution masks for averaging can be created.

Averaging with Pixels

- Averaging is a special case of discrete convolution. For a 3×3 neighborhood, the convolution mask h is

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{48}$$

- The significance of the central pixel may be increased,

$$h = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{49}$$

- Larger convolution masks for averaging can be created.

Averaging with Data Validity

- Methods that average with limited data validity try to avoid blurring by averaging only those pixels which satisfy some criterion.
- They prevent involving pixels that are part of a separate feature.
- A very simple criterion is to use only pixels in the original image with brightness in a predefined interval [min, max].
- For a pixel (m, n), the convolution mask is calculated in the neighborhood $\mathcal{O}$ from the nonlinear formula

$$h(i,j) = \begin{cases} 1, & \text{for } g(m+i, n+j) \in [\min, \max] \\ 0, & \text{otherwise.} \end{cases} \tag{50}$$

- The interval [min, max] represents valid data.

Averaging with Data Validity

- Methods that average with limited data validity try to avoid blurring by averaging only those pixels which satisfy some criterion.
- They prevent involving pixels that are part of a separate feature.
- A very simple criterion is to use only pixels in the original image with brightness in a predefined interval [min, max].
- For a pixel (m, n), the convolution mask is calculated in the neighborhood $\mathcal{O}$ from the nonlinear formula

$$h(i,j) = \begin{cases} 1, & \text{for } g(m+i, n+j) \in [\min, \max] \\ 0, & \text{otherwise.} \end{cases} \tag{50}$$

- The interval [min, max] represents valid data.

Averaging with Data Validity

- Methods that average with limited data validity try to avoid blurring by averaging only those pixels which satisfy some criterion.
- They prevent involving pixels that are part of a separate feature.
- A very simple criterion is to use only pixels in the original image with brightness in a predefined interval [min, max].
- For a pixel (m, n), the convolution mask is calculated in the neighborhood $\mathcal{O}$ from the nonlinear formula

$$h(i,j) = \begin{cases} 1, & \text{for } g(m+i, n+j) \in [\min, \max] \\ 0, & \text{otherwise.} \end{cases} \tag{50}$$

- The interval [min, max] represents valid data.

Averaging with Data Validity

- Methods that average with limited data validity try to avoid blurring by averaging only those pixels which satisfy some criterion.
- They prevent involving pixels that are part of a separate feature.
- A very simple criterion is to use only pixels in the original image with brightness in a predefined interval [min, max].
- For a pixel (m, n), the convolution mask is calculated in the neighborhood $\mathcal{O}$ from the nonlinear formula

$$h(i,j) = \begin{cases} 1, & \text{for } g(m+i, n+j) \in [\min, \max] \\ 0, & \text{otherwise.} \end{cases} \tag{50}$$

- The interval [min, max] represents valid data.

Averaging with Data Validity

- Methods that average with limited data validity try to avoid blurring by averaging only those pixels which satisfy some criterion.
- They prevent involving pixels that are part of a separate feature.
- A very simple criterion is to use only pixels in the original image with brightness in a predefined interval $[\min, \max]$.
- For a pixel (m, n), the convolution mask is calculated in the neighborhood $\mathcal{O}$ from the nonlinear formula

$$h(i, j) = \begin{cases} 1, & \text{for } g(m+i, n+j) \in [\min, \max] \\ 0, & \text{otherwise.} \end{cases} \tag{50}$$

- The interval $[\min, \max]$ represents valid data.

Averaging with Data Validity: Variants

- The second method performs the averaging only if the computed brightness change of a pixel is in some predefined interval.
- This method permits repair to large-area errors resulting from slowly changing brightness of the background without affecting the rest of the image.

- The third method uses edge strength (i.e., magnitude of a gradient) as a criterion.
- The magnitude of some gradient operator is first computed for the entire image, and only pixels in the input image with a gradient magnitude smaller than a predefined threshold are used in averaging.

Averaging with Data Validity: Variants

- The second method performs the averaging only if the computed brightness change of a pixel is in some predefined interval.
- This method permits repair to large-area errors resulting from slowly changing brightness of the background without affecting the rest of the image.

- The third method uses edge strength (i.e., magnitude of a gradient) as a criterion.
- The magnitude of some gradient operator is first computed for the entire image, and only pixels in the input image with a gradient magnitude smaller than a predefined threshold are used in averaging.

Averaging with Data Validity: Variants

- The second method performs the averaging only if the computed brightness change of a pixel is in some predefined interval.
- This method permits repair to large-area errors resulting from slowly changing brightness of the background without affecting the rest of the image.

- The third method uses edge strength (i.e., magnitude of a gradient) as a criterion.
- The magnitude of some gradient operator is first computed for the entire image, and only pixels in the input image with a gradient magnitude smaller than a predefined threshold are used in averaging.

Averaging with Data Validity: Variants

- The second method performs the averaging only if the computed brightness change of a pixel is in some predefined interval.
- This method permits repair to large-area errors resulting from slowly changing brightness of the background without affecting the rest of the image.

- The third method uses edge strength (i.e., magnitude of a gradient) as a criterion.
- The magnitude of some gradient operator is first computed for the entire image, and only pixels in the input image with a gradient magnitude smaller than a predefined threshold are used in averaging.

Averaging According to Inverse Gradient

- The idea is that the brightness change within a region is usually smaller than between neighboring regions.
- Let (i, j) be the pixel under processing. The inverse gradient at the pixel (m, n) with respect to (i, j) is

$$\delta(i,j,m,n) = \begin{cases} \frac{1}{|g(m,n)-g(i,j)|}, & \text{if } g(m,n) \neq g(i,j); \\ 2 & \text{if } g(m,n) = g(i,j). \end{cases}$$

- The inverse gradient is in the interval $(0, 2]$, and is smaller at edges than in the interior of a homogeneous region.
- Weight coefficients in the convolution mask h are normalized by the inverse gradient,

$$h(i,j,m,n) = \frac{\delta(i,j,m,n)}{\sum_{(m',n')\in\mathcal{O}} \delta(i,j,m',n')}, \quad \text{if } (m,n) \in \mathcal{O}.$$

Averaging According to Inverse Gradient

- The idea is that the brightness change within a region is usually smaller than between neighboring regions.
- Let (i, j) be the pixel under processing. The inverse gradient at the pixel (m, n) with respect to (i, j) is

$$\delta(i,j,m,n) = \begin{cases} \frac{1}{|g(m,n)-g(i,j)|}, & \text{if } g(m,n) \neq g(i,j); \\ 2 & \text{if } g(m,n) = g(i,j). \end{cases}$$

- The inverse gradient is in the interval $(0, 2]$, and is smaller at edges than in the interior of a homogeneous region.
- Weight coefficients in the convolution mask h are normalized by the inverse gradient,

$$h(i,j,m,n) = \frac{\delta(i,j,m,n)}{\sum_{(m',n')\in\mathcal{O}} \delta(i,j,m',n')}, \quad \text{if } (m,n) \in \mathcal{O}.$$

Averaging According to Inverse Gradient

- The idea is that the brightness change within a region is usually smaller than between neighboring regions.
- Let (i, j) be the pixel under processing. The inverse gradient at the pixel (m, n) with respect to (i, j) is

$$\delta(i, j, m, n) = \begin{cases} \frac{1}{|g(m,n) - g(i,j)|}, & \text{if } g(m, n) \neq g(i, j); \\ 2 & \text{if } g(m, n) = g(i, j). \end{cases}$$

- The inverse gradient is in the interval $(0, 2]$, and is smaller at edges than in the interior of a homogeneous region.
- Weight coefficients in the convolution mask h are normalized by the inverse gradient,

$$h(i, j, m, n) = \frac{\delta(i, j, m, n)}{\sum_{(m',n') \in \mathcal{O}} \delta(i, j, m', n')}, \quad \text{if } (m, n) \in \mathcal{O}.$$

Averaging According to Inverse Gradient

- The idea is that the brightness change within a region is usually smaller than between neighboring regions.
- Let (i,j) be the pixel under processing. The inverse gradient at the pixel (m,n) with respect to (i,j) is

$$\delta(i,j,m,n) = \begin{cases} \frac{1}{|g(m,n)-g(i,j)|}, & \text{if } g(m,n) \neq g(i,j); \\ 2 & \text{if } g(m,n) = g(i,j). \end{cases}$$

- The inverse gradient is in the interval $(0,2]$, and is smaller at edges than in the interior of a homogeneous region.
- Weight coefficients in the convolution mask h are normalized by the inverse gradient,

$$h(i,j,m,n) = \frac{\delta(i,j,m,n)}{\sum_{(m',n')\in\mathcal{O}} \delta(i,j,m',n')}, \quad \text{if } (m,n) \in \mathcal{O}.$$

Averaging According to Inverse Gradient

- The above method assumes sharp edges.
- Isolated noise points within homogeneous regions have small values of the inverse gradient; points from the neighborhood take part in averaging and the noise is removed.
- When the convolution mask is close to an edge, pixels from the region have larger coefficients than pixels near the edge, and they is not blurred.

Averaging According to Inverse Gradient

- The above method assumes sharp edges.
- Isolated noise points within homogeneous regions have small values of the inverse gradient; points from the neighborhood take part in averaging and the noise is removed.
- When the convolution mask is close to an edge, pixels from the region have larger coefficients than pixels near the edge, and they is not blurred.

Averaging According to Inverse Gradient

- The above method assumes sharp edges.
- Isolated noise points within homogeneous regions have small values of the inverse gradient; points from the neighborhood take part in averaging and the noise is removed.
- When the convolution mask is close to an edge, pixels from the region have larger coefficients than pixels near the edge, and they is not blurred.

Averaging Using a Rotating Mask

- This method avoids edge blurring by searching for the homogeneous part of the current pixel neighborhood.
- The resulting image is in fact sharpened.
- Brightness average is calculated only within the homogeneous region.
- A brightness dispersion σ^2 is used as the region homogeneity measure.
- Let n be the number of pixels in a region R and $g(i,j)$ be the input image. Dispersion σ^2 is calculated as

$$\sigma^2 = \frac{1}{n} \sum_{(i,j)\in R} \left[g(i,j) - \frac{1}{n} \sum_{(i',j')\in R} g(i',j') \right]^2 \tag{51}$$

Averaging Using a Rotating Mask

- This method avoids edge blurring by searching for the homogeneous part of the current pixel neighborhood.
- The resulting image is in fact sharpened.
- Brightness average is calculated only within the homogeneous region.
- A brightness dispersion σ^2 is used as the region homogeneity measure.
- Let n be the number of pixels in a region R and $g(i,j)$ be the input image. Dispersion σ^2 is calculated as

$$\sigma^2 = \frac{1}{n} \sum_{(i,j)\in R} \left[g(i,j) - \frac{1}{n} \sum_{(i',j')\in R} g(i',j') \right]^2 \qquad (51)$$

Averaging Using a Rotating Mask

- This method avoids edge blurring by searching for the homogeneous part of the current pixel neighborhood.
- The resulting image is in fact sharpened.
- Brightness average is calculated only within the homogeneous region.
- A brightness dispersion σ^2 is used as the region homogeneity measure.
- Let n be the number of pixels in a region R and $g(i,j)$ be the input image. Dispersion σ^2 is calculated as

$$\sigma^2 = \frac{1}{n} \sum_{(i,j)\in R} \left[g(i,j) - \frac{1}{n} \sum_{(i',j')\in R} g(i',j') \right]^2 \qquad (51)$$

Averaging Using a Rotating Mask

- This method avoids edge blurring by searching for the homogeneous part of the current pixel neighborhood.
- The resulting image is in fact sharpened.
- Brightness average is calculated only within the homogeneous region.
- A brightness dispersion σ^2 is used as the region homogeneity measure.
- Let n be the number of pixels in a region R and $g(i,j)$ be the input image. Dispersion σ^2 is calculated as

$$\sigma^2 = \frac{1}{n} \sum_{(i,j)\in R} \left[g(i,j) - \frac{1}{n} \sum_{(i',j')\in R} g(i',j') \right]^2 \qquad (51)$$

Averaging Using a Rotating Mask

- This method avoids edge blurring by searching for the homogeneous part of the current pixel neighborhood.
- The resulting image is in fact sharpened.
- Brightness average is calculated only within the homogeneous region.
- A brightness dispersion σ^2 is used as the region homogeneity measure.
- Let n be the number of pixels in a region R and $g(i,j)$ be the input image. Dispersion σ^2 is calculated as

$$\sigma^2 = \frac{1}{n} \sum_{(i,j)\in R} \left[g(i,j) - \frac{1}{n} \sum_{(i',j')\in R} g(i',j') \right]^2 \tag{51}$$

Dispersion Calculation

$$\sigma^2$$

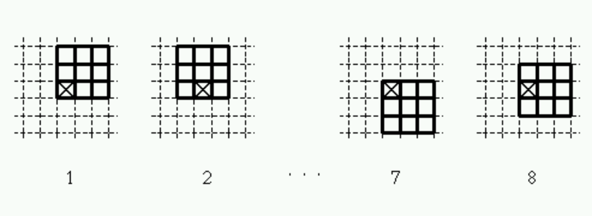
$$= \frac{1}{n} \sum_{(i,j)\in R} \left\{ g(i,j)^2 - 2g(i,j)\frac{\sum_{(i',j')\in R} g(i',j')}{n} + \left[\frac{\sum_{(i',j')\in R} g(i',j')}{n}\right]^2 \right\}$$

$$= \frac{1}{n} \left\{ \sum_{(i,j)\in R} g(i,j)^2 - 2\frac{\left[\sum_{(i',j')\in R} g(i',j')\right]^2}{n} + n\left[\frac{\sum_{(i',j')\in R} g(i',j')}{n}\right]^2 \right\}$$

$$= \frac{1}{n} \left\{ \sum_{(i,j)\in R} g(i,j)^2 - \frac{\left[\sum_{(i,j)\in R} g(i,j)\right]^2}{n} \right\}$$

Rotating Mask

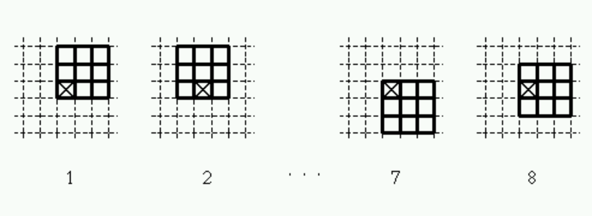
- Having computed region homogeneity, we consider its shape and size.
- The eight possible 3 × 3 masks that cover a 5 × 5 neighborhood of a current pixel (marked by small cross in the following figure) are shown in the following figure



The ninth mask is the 3 × 3 neighborhood of the current pixel itself.

Rotating Mask

- Having computed region homogeneity, we consider its shape and size.
- The eight possible 3×3 masks that cover a 5×5 neighborhood of a current pixel (marked by small cross in the following figure) are shown in the following figure



The ninth mask is the 3×3 neighborhood of the current pixel itself.

Algorithm: smoothing using a rotating mask

1. For each image pixel (i, j), calculate dispersion in the mask for all possible mask rotations about pixel (i, j).
2. Choose the mask with minimum dispersion.
3. Assign to the pixel (i, j) in the output image the average brightness in the chosen mask.

Algorithm: smoothing using a rotating mask

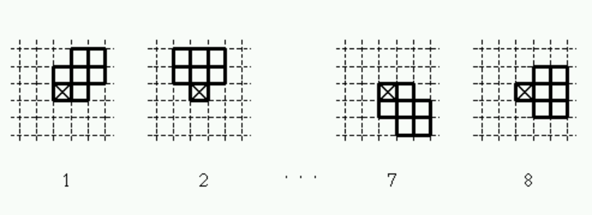
1. For each image pixel (i, j), calculate dispersion in the mask for all possible mask rotations about pixel (i, j).
2. Choose the mask with minimum dispersion.
3. Assign to the pixel (i, j) in the output image the average brightness in the chosen mask.

Algorithm: smoothing using a rotating mask

1. For each image pixel (i, j), calculate dispersion in the mask for all possible mask rotations about pixel (i, j).
2. Choose the mask with minimum dispersion.
3. Assign to the pixel (i, j) in the output image the average brightness in the chosen mask.

Other Masks

- Another set of eight masks covering a 5 × 5 neighborhood of the current pixel



Again the ninth mask is the 3 × 3 neighborhood of the current pixel itself.

Other Masks

- Another possibility is to rotate a small 2×1 mask to cover the 3×3 neighborhood of the current pixel.
- This algorithm can be used iteratively. (What about other algorithms?)
- The iterative process converges quite quickly to stable state (that is, the image does not change any more).

Other Masks

- Another possibility is to rotate a small 2×1 mask to cover the 3×3 neighborhood of the current pixel.
- This algorithm can be used iteratively. (What about other algorithms?)
- The iterative process converges quite quickly to stable state (that is, the image does not change any more).

Other Masks

- Another possibility is to rotate a small 2×1 mask to cover the 3×3 neighborhood of the current pixel.
- This algorithm can be used iteratively. (What about other algorithms?)
- The iterative process converges quite quickly to stable state (that is, the image does not change any more).

Example

- The matlab script from `visionbook` is is visionbook/05Preproc/rotmask_demo.m.

Median Filtering

- In a set of ordered values, the median is the central value.
- Median filtering assigns the output pixel brightness value to be the median value of the brightness values in a neighborhood of the pixel.
- Median filtering reduces blurring of edges.
- The median of the brightness in the neighborhood is not affected by individual noise spikes and so median smoothing eliminates impulsive noise quite well.
- As median filtering does not blur edges much, it can be applied iteratively.

Median Filtering

- In a set of ordered values, the median is the central value.
- Median filtering assigns the output pixel brightness value to be the median value of the brightness values in a neighborhood of the pixel.
- Median filtering reduces blurring of edges.
- The median of the brightness in the neighborhood is not affected by individual noise spikes and so median smoothing eliminates impulsive noise quite well.
- As median filtering does not blur edges much, it can be applied iteratively.

Median Filtering

- In a set of ordered values, the median is the central value.
- Median filtering assigns the output pixel brightness value to be the median value of the brightness values in a neighborhood of the pixel.
- Median filtering reduces blurring of edges.
- The median of the brightness in the neighborhood is not affected by individual noise spikes and so median smoothing eliminates impulsive noise quite well.
- As median filtering does not blur edges much, it can be applied iteratively.

Median Filtering

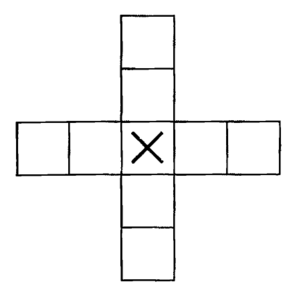
- In a set of ordered values, the median is the central value.
- Median filtering assigns the output pixel brightness value to be the median value of the brightness values in a neighborhood of the pixel.
- Median filtering reduces blurring of edges.
- The median of the brightness in the neighborhood is not affected by individual noise spikes and so median smoothing eliminates impulsive noise quite well.
- As median filtering does not blur edges much, it can be applied iteratively.

Median Filtering

- In a set of ordered values, the median is the central value.
- Median filtering assigns the output pixel brightness value to be the median value of the brightness values in a neighborhood of the pixel.
- Median filtering reduces blurring of edges.
- The median of the brightness in the neighborhood is not affected by individual noise spikes and so median smoothing eliminates impulsive noise quite well.
- As median filtering does not blur edges much, it can be applied iteratively.

Median Filtering

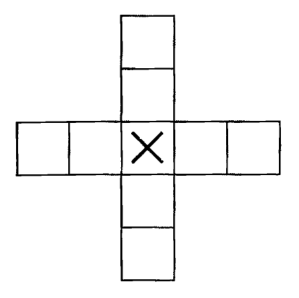
- The main disadvantage of median filtering in a rectangular neighborhood is its damaging of thin lines and sharp corners in the image — this can be avoided if another shape of neighborhood is used.



- Variants of median filtering is to choose the maximum and minimum values in the neighborhood.
- This leads to the dilation and erosion operators in mathematical morphology.

Median Filtering

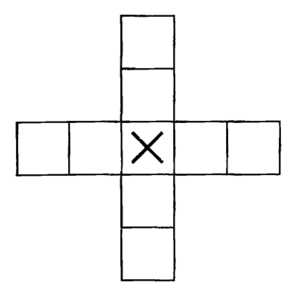
- The main disadvantage of median filtering in a rectangular neighborhood is its damaging of thin lines and sharp corners in the image — this can be avoided if another shape of neighborhood is used.



- Variants of median filtering is to choose the maximum and minimum values in the neighborhood.
- This leads to the dilation and erosion operators in mathematical morphology.

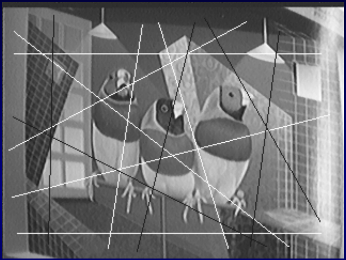
Median Filtering

- The main disadvantage of median filtering in a rectangular neighborhood is its damaging of thin lines and sharp corners in the image — this can be avoided if another shape of neighborhood is used.



- Variants of median filtering is to choose the maximum and minimum values in the neighborhood.
- This leads to the dilation and erosion operators in mathematical morphology.

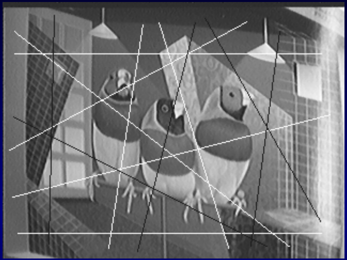
Example I



(f) Original

Figure: Iterative application of median filter in 3×3 neighborhoods.

Example II

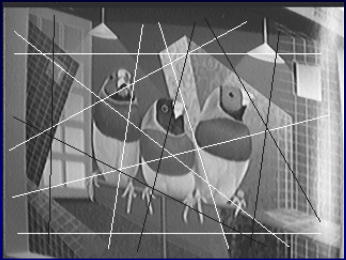


(a) Original

(b) 1st median filtering

Figure: Iterative application of median filter in 3×3 neighborhoods.

Example III

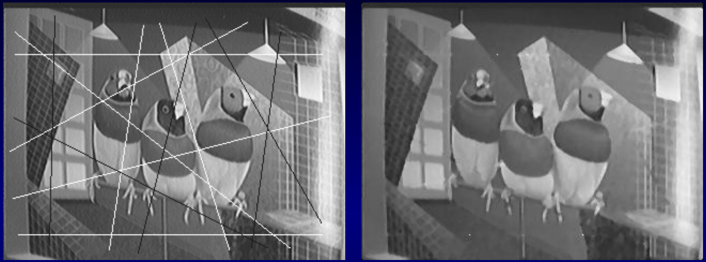


(a) Original

(b) 2nd median filtering

Figure: Iterative application of median filter in 3×3 neighborhoods.

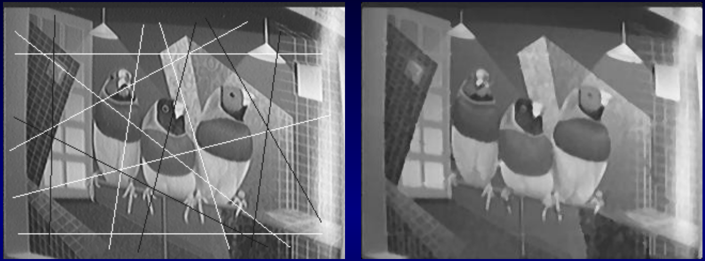
Example IV



(a) Original (b) 3rd median filtering

Figure: Iterative application of median filter in 3×3 neighborhoods.

Example V



(a) Original (b) 4th median filtering

Figure: Iterative application of median filter in 3×3 neighborhoods.

Example II

- The matlab script from `visionbook` is is visionbook/05Preproc/medfilt_demo.

Non-linear Mean Filtering

- The non-linear mean filter is another generalization of averaging techniques.
- It is defined by

$$f(i,j) = u^{-1}\left(\frac{\sum_{(m,n)\in\mathcal{O}} a(m,n)u[g(m,n)]}{\sum_{(m,n)\in\mathcal{O}} a(m,n)}\right) \tag{52}$$

  - $f(i,j)$ result of the filtering,
  - $g(m,n)$ pixel in the input image,
  - $\mathcal{O}$ local neighborhood of the current pixel (i,j).
  - u^{-1} the inverse of the function u,
  - $a(m,n)$ weight coefficients.

Non-linear Mean Filtering

- The non-linear mean filter is another generalization of averaging techniques.
- It is defined by

$$f(i,j) = u^{-1}\left(\frac{\sum_{(m,n)\in\mathcal{O}} a(m,n)u[g(m,n)]}{\sum_{(m,n)\in\mathcal{O}} a(m,n)}\right) \tag{52}$$

  - $f(i,j)$ result of the filtering,
  - $g(m,n)$ pixel in the input image,
  - $\mathcal{O}$ local neighborhood of the current pixel (i,j).
  - u^{-1} the inverse of the function u,
  - $a(m,n)$ weight coefficients.

Non-linear Mean Filtering

- The non-linear mean filter is another generalization of averaging techniques.
- It is defined by

$$f(i,j) = u^{-1}\left(\frac{\sum_{(m,n)\in\mathcal{O}} a(m,n)u[g(m,n)]}{\sum_{(m,n)\in\mathcal{O}} a(m,n)}\right) \qquad (52)$$

  - $f(i,j)$ result of the filtering,
  - $g(m,n)$ pixel in the input image,
  - $\mathcal{O}$ local neighborhood of the current pixel (i,j).
  - u^{-1} the inverse of the function u,
  - $a(m,n)$ weight coefficients.

Non-linear Mean Filtering

- The non-linear mean filter is another generalization of averaging techniques.
- It is defined by

$$f(i,j) = u^{-1}\left(\frac{\sum_{(m,n)\in\mathcal{O}} a(m,n)u[g(m,n)]}{\sum_{(m,n)\in\mathcal{O}} a(m,n)}\right) \quad (52)$$

  - $f(i,j)$ result of the filtering,
  - $g(m,n)$ pixel in the input image,
  - $\mathcal{O}$ local neighborhood of the current pixel (i,j).
  - u^{-1} the inverse of the function u,
  - $a(m,n)$ weight coefficients.

Non-linear Mean Filtering

- The non-linear mean filter is another generalization of averaging techniques.
- It is defined by

$$f(i,j) = u^{-1}\left(\frac{\sum_{(m,n)\in\mathcal{O}} a(m,n)u[g(m,n)]}{\sum_{(m,n)\in\mathcal{O}} a(m,n)}\right) \qquad (52)$$

  - $f(i,j)$ result of the filtering,
  - $g(m,n)$ pixel in the input image,
  - $\mathcal{O}$ local neighborhood of the current pixel (i,j).
  - u^{-1} the inverse of the function u,
  - $a(m,n)$ weight coefficients.

Non-linear Mean Filtering

- The non-linear mean filter is another generalization of averaging techniques.
- It is defined by

$$f(i,j) = u^{-1}\left(\frac{\sum_{(m,n)\in\mathcal{O}} a(m,n)u[g(m,n)]}{\sum_{(m,n)\in\mathcal{O}} a(m,n)}\right) \tag{52}$$

  - $f(i,j)$ result of the filtering,
  - $g(m,n)$ pixel in the input image,
  - $\mathcal{O}$ local neighborhood of the current pixel (i,j).
  - u^{-1} the inverse of the function u,
  - $a(m,n)$ weight coefficients.

Non-linear Mean Filtering

- The non-linear mean filter is another generalization of averaging techniques.
- It is defined by

$$f(i,j) = u^{-1}\left(\frac{\sum_{(m,n)\in\mathcal{O}} a(m,n)u[g(m,n)]}{\sum_{(m,n)\in\mathcal{O}} a(m,n)}\right) \tag{52}$$

  - $f(i,j)$ result of the filtering,
  - $g(m,n)$ pixel in the input image,
  - $\mathcal{O}$ local neighborhood of the current pixel (i,j).
  - u^{-1} the inverse of the function u,
  - $a(m,n)$ weight coefficients.

Homomorphic Filters

- If the coefficients $a(i,j)$ are constants, the filter is called homomorphic.
- Some homomorphic filters used in image processing are
  - Arithmetic mean, $u(g) = g$.
  - Harmonic mean, $u(g) = \frac{1}{g}$.
  - Geometric mean, $u(g) = \log g$.

Homomorphic Filters

- If the coefficients $a(i,j)$ are constants, the filter is called homomorphic.
- Some homomorphic filters used in image processing are
  - Arithmetic mean, $u(g) = g$.
  - Harmonic mean, $u(g) = \frac{1}{g}$.
  - Geometric mean, $u(g) = \log g$.

Homomorphic Filters

- If the coefficients $a(i, j)$ are constants, the filter is called homomorphic.
- Some homomorphic filters used in image processing are
  - Arithmetic mean, $u(g) = g$.
  - Harmonic mean, $u(g) = \frac{1}{g}$.
  - Geometric mean, $u(g) = \log g$.

Homomorphic Filters

- If the coefficients $a(i,j)$ are constants, the filter is called homomorphic.
- Some homomorphic filters used in image processing are
  - Arithmetic mean, $u(g) = g$.
  - Harmonic mean, $u(g) = \frac{1}{g}$.
  - Geometric mean, $u(g) = \log g$.

Homomorphic Filters

- If the coefficients $a(i, j)$ are constants, the filter is called homomorphic.
- Some homomorphic filters used in image processing are
  - Arithmetic mean, $u(g) = g$.
  - Harmonic mean, $u(g) = \frac{1}{g}$.
  - Geometric mean, $u(g) = \log g$.

Theorem

Theorem

Given $x_1 \le x_2 \le \cdots \le x_N$, then

a $\arg\min_a \sum_{i=1}^{N} |x_i - a|^2$ *is the arithmetic mean of* $x_1, x_2, \cdots, x_N$;

b $\arg\min_a \sum_{i=1}^{N} |x_i - a|$ *is the median of* $x_1, x_2, \cdots, x_N$;

c $\arg\min_a \max_{1 \le i \le N} |x_i - a|$ *is* $\frac{x_1 + x_N}{2}$.

Theorem

Theorem

Given $x_1 \leq x_2 \leq \cdots \leq x_N$, *then*

a $\arg\min_a \sum_{i=1}^{N} |x_i - a|^2$ *is the arithmetic mean of* $x_1, x_2, \cdots, x_N$;

b $\arg\min_a \sum_{i=1}^{N} |x_i - a|$ *is the median of* $x_1, x_2, \cdots, x_N$;

c $\arg\min_a \max_{1 \leq i \leq N} |x_i - a|$ *is* $\frac{x_1 + x_N}{2}$.

Theorem

Theorem

Given $x_1 \leq x_2 \leq \cdots \leq x_N$, then

a $\arg\min_a \sum_{i=1}^{N} |x_i - a|^2$ *is the arithmetic mean of* $x_1, x_2, \cdots, x_N$*;*

b $\arg\min_a \sum_{i=1}^{N} |x_i - a|$ *is the median of* $x_1, x_2, \cdots, x_N$*;*

c $\arg\min_a \max_{1 \leq i \leq N} |x_i - a|$ *is* $\frac{x_1 + x_N}{2}$.

Theorem

Theorem

Given $x_1 \leq x_2 \leq \cdots \leq x_N$, then

a $\mathbf{arg\,min}_a \sum_{i=1}^{N} |x_i - a|^2$ *is the arithmetic mean of* $x_1, x_2, \cdots, x_N$*;*

b $\mathbf{arg\,min}_a \sum_{i=1}^{N} |x_i - a|$ *is the median of* $x_1, x_2, \cdots, x_N$*;*

c $\mathbf{arg\,min}_a \max_{1 \leq i \leq N} |x_i - a|$ *is* $\frac{x_1 + x_N}{2}$.

Proof of (a)

- Let

$$g(a) = \sum_{i=1}^{N} |x_i - a|^2. \tag{53}$$

The result follows immediately by calculus.

Proof of (b) I

- Let
$$g(a) = \sum_{i=1}^{N} |x_i - a|. \quad (54)$$
- Let a_0 and a_1 be arbitrary numbers such that
$$a_0 < x_1 \leq x_N < a_1. \quad (55)$$
- Then
$$g(a_0) = \sum_{i=1}^{N} x_i - Na_0, \quad (56)$$
$$g(a_1) = Na_1 - \sum_{i=1}^{N} x_i. \quad (57)$$
- If $a \in [x_1, x_N]$, assume that
$$x_k \leq a < x_{k+1}. \quad (58)$$
- Then
$$g(a) = \sum_{i=1}^{k}(a - x_i) + \sum_{i=k+1}^{N}(x_i - a) \quad (59)$$
$$= \sum_{i=k+1}^{N} x_i - \sum_{i=1}^{k} x_i + (2k - N)a. \quad (60)$$

Proof of (b) I

- Let

$$g(a) = \sum_{i=1}^{N} |x_i - a|. \quad (54)$$

- Let a_0 and a_1 be arbitrary numbers such that

$$a_0 < x_1 \leq x_N < a_1. \quad (55)$$

- Then

$$g(a_0) = \sum_{i=1}^{N} x_i - Na_0, \quad (56)$$

$$g(a_1) = Na_1 - \sum_{i=1}^{N} x_i. \quad (57)$$

- If $a \in [x_1, x_N]$, assume that

$$x_k \leq a < x_{k+1}. \quad (58)$$

- Then

$$g(a) = \sum_{i=1}^{k} (a - x_i) + \sum_{i=k+1}^{N} (x_i - a) \quad (59)$$

$$= \sum_{i=k+1}^{N} x_i - \sum_{i=1}^{k} x_i + (2k - N)a. \quad (60)$$

Proof of (b) I

- Let

$$g(a) = \sum_{i=1}^{N} |x_i - a|. \quad (54)$$

- Let a_0 and a_1 be arbitrary numbers such that

$$a_0 < x_1 \leq x_N < a_1. \quad (55)$$

- Then

$$g(a_0) = \sum_{i=1}^{N} x_i - Na_0, \quad (56)$$

$$g(a_1) = Na_1 - \sum_{i=1}^{N} x_i. \quad (57)$$

- If $a \in [x_1, x_N]$, assume that

$$x_k \leq a < x_{k+1}. \quad (58)$$

- Then

$$g(a) = \sum_{i=1}^{k} (a - x_i) + \sum_{i=k+1}^{N} (x_i - a) \quad (59)$$

$$= \sum_{i=k+1}^{N} x_i - \sum_{i=1}^{k} x_i + (2k - N)a. \quad (60)$$

Proof of (b) I

- Let
$$g(a) = \sum_{i=1}^{N} |x_i - a|. \tag{54}$$
- Let a_0 and a_1 be arbitrary numbers such that
$$a_0 < x_1 \leq x_N < a_1. \tag{55}$$
- Then
$$g(a_0) = \sum_{i=1}^{N} x_i - Na_0, \tag{56}$$
$$g(a_1) = Na_1 - \sum_{i=1}^{N} x_i. \tag{57}$$
- If $a \in [x_1, x_N]$, assume that
$$x_k \leq a < x_{k+1}. \tag{58}$$
- Then
$$g(a) = \sum_{i=1}^{k} (a - x_i) + \sum_{i=k+1}^{N} (x_i - a) \tag{59}$$
$$= \sum_{i=k+1}^{N} x_i - \sum_{i=1}^{k} x_i + (2k - N)a. \tag{60}$$

Proof of (b) I

- Let
$$g(a) = \sum_{i=1}^{N} |x_i - a|. \quad (54)$$
- Let a_0 and a_1 be arbitrary numbers such that
$$a_0 < x_1 \leq x_N < a_1. \quad (55)$$
- Then
$$g(a_0) = \sum_{i=1}^{N} x_i - Na_0, \quad (56)$$
$$g(a_1) = Na_1 - \sum_{i=1}^{N} x_i. \quad (57)$$
- If $a \in [x_1, x_N]$, assume that
$$x_k \leq a < x_{k+1}. \quad (58)$$
- Then
$$g(a) = \sum_{i=1}^{k}(a - x_i) + \sum_{i=k+1}^{N}(x_i - a) \quad (59)$$
$$= \sum_{i=k+1}^{N} x_i - \sum_{i=1}^{k} x_i + (2k - N)a. \quad (60)$$

Proof of (b) II

- If $2k \leq N$,

$$g(a_0) - g(a) = 2\sum_{i=1}^{k} x_i - Na_0 + (N-2k)a$$
$$= 2\sum_{i=1}^{k}(x_i - a_0) + (N-2k)(a-a_0) \geq 0.$$

- If $2k \geq N$,

$$g(a_1) - g(a) = Na_1 - 2\sum_{i=k+1}^{N} x_i + (N-2k)a$$
$$= 2\sum_{i=k+1}^{N}(a_1 - x_i) + (N - 2(N-k))a_1 + (N-2k)a$$
$$= 2\sum_{i=k+1}^{N}(a_1 - x_i) + (2k-N)(a_1 - a) \geq 0.$$

Proof of (b) II

- If $2k \le N$,

$$g(a_0) - g(a) = 2\sum_{i=1}^{k} x_i - Na_0 + (N-2k)a$$
$$= 2\sum_{i=1}^{k}(x_i - a_0) + (N-2k)(a-a_0) \ge 0.$$

- If $2k \ge N$,

$$g(a_1) - g(a) = Na_1 - 2\sum_{i=k+1}^{N} x_i + (N-2k)a$$
$$= 2\sum_{i=k+1}^{N}(a_1 - x_i) + (N - 2(N-k))a_1 + (N-2k)a$$
$$= 2\sum_{i=k+1}^{N}(a_1 - x_i) + (2k-N)(a_1 - a) \ge 0.$$

Proof of (b) II

- Therefore, the minimum of $g(a)$ is attained in $[x_1, x_N]$.
  - $g(a)$ is continuous, piece wise linear function on $[x_1, x_N]$.
  - $g(a)$ is decreasing if $2k \leq N$ and increasing if $2k \geq N$.
- If $N = 2m$ is even, the minimum of $g(a)$ is attained in the central sub-interval $[x_m, x_{m+1}]$.
  - $g(a)$ is constant on this sub-interval.
  - Any value of $[x_m, x_{m+1}]$ is a minimizer of $g(a)$.
  - $a = \frac{x_m + x_{m+1}}{2}$ the conventional median in this case.
- If $N = 2m + 1$, $g(a)$ is decreasing in one of the central sub-interval $[x_m, x_{m+1}]$ and increasing in another central sub-interval $[x_{m+1}, x_{m+2}]$.
  - Then x_{m+1} is the minimizer, which is the median in this case.

Proof of (b) II

- Therefore, the minimum of $g(a)$ is attained in $[x_1, x_N]$.
  - $g(a)$ is continuous, piece wise linear function on $[x_1, x_N]$.
  - $g(a)$ is decreasing if $2k \leq N$ and increasing if $2k \geq N$.
- If $N = 2m$ is even, the minimum of $g(a)$ is attained in the central sub-interval $[x_m, x_{m+1}]$.
  - $g(a)$ is constant on this sub-interval.
  - Any value of $[x_m, x_{m+1}]$ is a minimizer of $g(a)$.
  - $a = \frac{x_m + x_{m+1}}{2}$ the conventional median in this case.
- If $N = 2m + 1$, $g(a)$ is decreasing in one of the central sub-interval $[x_m, x_{m+1}]$ and increasing in another central sub-interval $[x_{m+1}, x_{m+2}]$.
  - Then x_{m+1} is the minimizer, which is the median in this case.

Proof of (b) II

- Therefore, the minimum of $g(a)$ is attained in $[x_1, x_N]$.
  - $g(a)$ is continuous, piece wise linear function on $[x_1, x_N]$.
  - $g(a)$ is decreasing if $2k \leq N$ and increasing if $2k \geq N$.
- If $N = 2m$ is even, the minimum of $g(a)$ is attained in the central sub-interval $[x_m, x_{m+1}]$.
  - $g(a)$ is constant on this sub-interval.
  - Any value of $[x_m, x_{m+1}]$ is a minimizer of $g(a)$.
  - $a = \frac{x_m + x_{m+1}}{2}$ the conventional median in this case.
- If $N = 2m + 1$, $g(a)$ is decreasing in one of the central sub-interval $[x_m, x_{m+1}]$ and increasing in another central sub-interval $[x_{m+1}, x_{m+2}]$.
  - Then x_{m+1} is the minimizer, which is the median in this case.

Proof of (b) II

- Therefore, the minimum of $g(a)$ is attained in $[x_1, x_N]$.
  - $g(a)$ is continuous, piece wise linear function on $[x_1, x_N]$.
  - $g(a)$ is decreasing if $2k \leq N$ and increasing if $2k \geq N$.
- If $N = 2m$ is even, the minimum of $g(a)$ is attained in the central sub-interval $[x_m, x_{m+1}]$.
  - $g(a)$ is constant on this sub-interval.
  - Any value of $[x_m, x_{m+1}]$ is a minimizer of $g(a)$.
  - $a = \frac{x_m + x_{m+1}}{2}$ the conventional median in this case.
- If $N = 2m + 1$, $g(a)$ is decreasing in one of the central sub-interval $[x_m, x_{m+1}]$ and increasing in another central sub-interval $[x_{m+1}, x_{m+2}]$.
  - Then x_{m+1} is the minimizer, which is the median in this case.

Proof of (b) II

- Therefore, the minimum of $g(a)$ is attained in $[x_1, x_N]$.
  - $g(a)$ is continuous, piece wise linear function on $[x_1, x_N]$.
  - $g(a)$ is decreasing if $2k \leq N$ and increasing if $2k \geq N$.
- If $N = 2m$ is even, the minimum of $g(a)$ is attained in the central sub-interval $[x_m, x_{m+1}]$.
  - $g(a)$ is constant on this sub-interval.
  - Any value of $[x_m, x_{m+1}]$ is a minimizer of $g(a)$.
  - $a = \frac{x_m + x_{m+1}}{2}$ the conventional median in this case.
- If $N = 2m + 1$, $g(a)$ is decreasing in one of the central sub-interval $[x_m, x_{m+1}]$ and increasing in another central sub-interval $[x_{m+1}, x_{m+2}]$.
  - Then x_{m+1} is the minimizer, which is the median in this case.

Proof of (b) II

- Therefore, the minimum of $g(a)$ is attained in $[x_1, x_N]$.
  - $g(a)$ is continuous, piece wise linear function on $[x_1, x_N]$.
  - $g(a)$ is decreasing if $2k \le N$ and increasing if $2k \ge N$.
- If $N = 2m$ is even, the minimum of $g(a)$ is attained in the central sub-interval $[x_m, x_{m+1}]$.
  - $g(a)$ is constant on this sub-interval.
  - Any value of $[x_m, x_{m+1}]$ is a minimizer of $g(a)$.
  - $a = \frac{x_m + x_{m+1}}{2}$ the conventional median in this case.
- If $N = 2m + 1$, $g(a)$ is decreasing in one of the central sub-interval $[x_m, x_{m+1}]$ and increasing in another central sub-interval $[x_{m+1}, x_{m+2}]$.
  - Then x_{m+1} is the minimizer, which is the median in this case.

Proof of (b) II

- Therefore, the minimum of $g(a)$ is attained in $[x_1, x_N]$.
  - $g(a)$ is continuous, piece wise linear function on $[x_1, x_N]$.
  - $g(a)$ is decreasing if $2k \leq N$ and increasing if $2k \geq N$.
- If $N = 2m$ is even, the minimum of $g(a)$ is attained in the central sub-interval $[x_m, x_{m+1}]$.
  - $g(a)$ is constant on this sub-interval.
  - Any value of $[x_m, x_{m+1}]$ is a minimizer of $g(a)$.
  - $a = \frac{x_m + x_{m+1}}{2}$ the conventional median in this case.
- If $N = 2m + 1$, $g(a)$ is decreasing in one of the central sub-interval $[x_m, x_{m+1}]$ and increasing in another central sub-interval $[x_{m+1}, x_{m+2}]$.
  - Then x_{m+1} is the minimizer, which is the median in this case.

Proof of (b) II

- Therefore, the minimum of $g(a)$ is attained in $[x_1, x_N]$.
  - $g(a)$ is continuous, piece wise linear function on $[x_1, x_N]$.
  - $g(a)$ is decreasing if $2k \leq N$ and increasing if $2k \geq N$.
- If $N = 2m$ is even, the minimum of $g(a)$ is attained in the central sub-interval $[x_m, x_{m+1}]$.
  - $g(a)$ is constant on this sub-interval.
  - Any value of $[x_m, x_{m+1}]$ is a minimizer of $g(a)$.
  - $a = \frac{x_m + x_{m+1}}{2}$ the conventional median in this case.
- If $N = 2m + 1$, $g(a)$ is decreasing in one of the central sub-interval $[x_m, x_{m+1}]$ and increasing in another central sub-interval $[x_{m+1}, x_{m+2}]$.
  - Then x_{m+1} is the minimizer, which is the median in this case.

Proof of (b) II

- Therefore, the minimum of $g(a)$ is attained in $[x_1, x_N]$.
  - $g(a)$ is continuous, piece wise linear function on $[x_1, x_N]$.
  - $g(a)$ is decreasing if $2k \leq N$ and increasing if $2k \geq N$.
- If $N = 2m$ is even, the minimum of $g(a)$ is attained in the central sub-interval $[x_m, x_{m+1}]$.
  - $g(a)$ is constant on this sub-interval.
  - Any value of $[x_m, x_{m+1}]$ is a minimizer of $g(a)$.
  - $a = \frac{x_m + x_{m+1}}{2}$ the conventional median in this case.
- If $N = 2m + 1$, $g(a)$ is decreasing in one of the central sub-interval $[x_m, x_{m+1}]$ and increasing in another central sub-interval $[x_{m+1}, x_{m+2}]$.
  - Then x_{m+1} is the minimizer, which is the median in this case.

Proof of (c)

- Let
$$g(a) = \max_{1 \le i \le N} |x_i - a|. \quad (61)$$
- Then it is easy to verify that
$$g(a) = \begin{cases} x_N - a, & \text{if } a < x_1; \\ x_N - a, & \text{if } x_1 \le a < \frac{x_1 + x_N}{2}; \\ a - x_1, & \text{if } \frac{x_1 + x_N}{2} \le a \le x_N; \\ a - x_1, & \text{if } x_N < a. \end{cases} \quad (62)$$

The conclusion follows immediately.

Proof of (c)

- Let

$$g(a) = \max_{1 \le i \le N} |x_i - a|. \quad (61)$$

- Then it is easy to verify that

$$g(a) = \begin{cases} x_N - a, & \text{if } a < x_1; \\ x_N - a, & \text{if } x_1 \le a < \frac{x_1 + x_N}{2}; \\ a - x_1, & \text{if } \frac{x_1 + x_N}{2} \le a \le x_N; \\ a - x_1, & \text{if } x_N < a. \end{cases} \quad (62)$$

The conclusion follows immediately.

Outline

- Pixel Brightness Transformations
  - Position dependent brightness correction
  - Grey scale transformation
    - Windows and level
    - Histogram stretching
    - Histogram Equalization
    - Histogram Matching
- Geometric Transformations
  - Pixel Co-ordinate Transformations
    - Polynomial Approximation
    - Bilinear Transformations
    - Important Transformations
  - Brightness Interpolation
    - Nearest Neighbor Interpolation
    - Bilinear Interpolation
    - Bi-cubic interpolation
- **Local pre-processing**
  - Image smoothing
    - Averaging
    - Averaging with Data Validity
    - Averaging According to Inverse Gradient

Edge

- Edge detectors are important local image pre-processing methods to locate (sharp) changes in images.
- Edges are pixels where the brightness function changes abruptly.
- Neurological and psychophysical study suggests that locations in the image in which the inensity value changes abruptly are important for image perception.
- Edges are to a certain degree invariant to changes of illumination and viewpoint.

Edge

- Edge detectors are important local image pre-processing methods to locate (sharp) changes in images.
- Edges are pixels where the brightness function changes abruptly.
- Neurological and psychophysical study suggests that locations in the image in which the inensity value changes abruptly are important for image perception.
- Edges are to a certain degree invariant to changes of illumination and viewpoint.

Edge

- Edge detectors are important local image pre-processing methods to locate (sharp) changes in images.
- Edges are pixels where the brightness function changes abruptly.
- Neurological and psychophysical study suggests that locations in the image in which the inensity value changes abruptly are important for image perception.
- Edges are to a certain degree invariant to changes of illumination and viewpoint.

Edge

- Edge detectors are important local image pre-processing methods to locate (sharp) changes in images.
- Edges are pixels where the brightness function changes abruptly.
- Neurological and psychophysical study suggests that locations in the image in which the inensity value changes abruptly are important for image perception.
- Edges are to a certain degree invariant to changes of illumination and viewpoint.

Edge

- If only edge elements with strong magnitude (edgels) are considered, such information often suffices for image understanding.

Figure: Siesta by Pablo Picasso, 1919

.

- The positive effect of such a process is that it leads to significant reduction of image data.
- Nevertheless such a data reduction does not undermine understanding the content of the image (interpretation) in many cases.

Edge

- If only edge elements with strong magnitude (edgels) are considered, such information often suffices for image understanding.

Figure: Siesta by Pablo Picasso, 1919

.

- The positive effect of such a process is that it leads to significant reduction of image data.
- Nevertheless such a data reduction does not undermine understanding the content of the image (interpretation) in many cases.

Edge

- If only edge elements with strong magnitude (edgels) are considered, such information often suffices for image understanding.

Figure: Siesta by Pablo Picasso, 1919

.

- The positive effect of such a process is that it leads to significant reduction of image data.
- Nevertheless such a data reduction does not undermine understanding the content of the image (interpretation) in many cases.

Edge by Gradient

- Calculus describes changes of continuous functions using derivatives.
- An image function depends on two variables — so operators describing edges are expressed using partial derivatives.
- A change of the image function can be described by a gradient that points in the direction of the largest growth of the image function.
- An edge is a (local) property attached to an individual pixel and is calculated from the image function in a neighborhood of the pixel.
- It is a `vector variable` with two components
  - magnitude of the gradient;
  - and direction ϕ is rotated with respect to the gradient direction ψ by -90^o.

Edge by Gradient

- Calculus describes changes of continuous functions using derivatives.
- An image function depends on two variables — so operators describing edges are expressed using partial derivatives.
- A change of the image function can be described by a gradient that points in the direction of the largest growth of the image function.
- An edge is a (local) property attached to an individual pixel and is calculated from the image function in a neighborhood of the pixel.
- It is a `vector variable` with two components
  - magnitude of the gradient;
  - and direction ϕ is rotated with respect to the gradient direction ψ by -90^o.

Edge by Gradient

- Calculus describes changes of continuous functions using derivatives.
- An image function depends on two variables — so operators describing edges are expressed using partial derivatives.
- A change of the image function can be described by a gradient that points in the direction of the largest growth of the image function.

- An edge is a (local) property attached to an individual pixel and is calculated from the image function in a neighborhood of the pixel.
- It is a `vector variable` with two components
  - magnitude of the gradient;
  - and direction ϕ is rotated with respect to the gradient direction ψ by -90^o.

Edge by Gradient

- Calculus describes changes of continuous functions using derivatives.
- An image function depends on two variables — so operators describing edges are expressed using partial derivatives.
- A change of the image function can be described by a gradient that points in the direction of the largest growth of the image function.

- An edge is a (local) property attached to an individual pixel and is calculated from the image function in a neighborhood of the pixel.
- It is a `vector variable` with two components
  - magnitude of the gradient;
  - and direction ϕ is rotated with respect to the gradient direction ψ by -90^o.

Edge by Gradient

- Calculus describes changes of continuous functions using derivatives.
- An image function depends on two variables — so operators describing edges are expressed using partial derivatives.
- A change of the image function can be described by a gradient that points in the direction of the largest growth of the image function.
- An edge is a (local) property attached to an individual pixel and is calculated from the image function in a neighborhood of the pixel.
- It is a `vector variable` with two components
  - magnitude of the gradient;
  - and direction ϕ is rotated with respect to the gradient direction ψ by -90^o.

Edge by Gradient

- Calculus describes changes of continuous functions using derivatives.
- An image function depends on two variables — so operators describing edges are expressed using partial derivatives.
- A change of the image function can be described by a gradient that points in the direction of the largest growth of the image function.

- An edge is a (local) property attached to an individual pixel and is calculated from the image function in a neighborhood of the pixel.
- It is a `vector variable` with two components
  - magnitude of the gradient;
  - and direction ϕ is rotated with respect to the gradient direction ψ by -90^o.

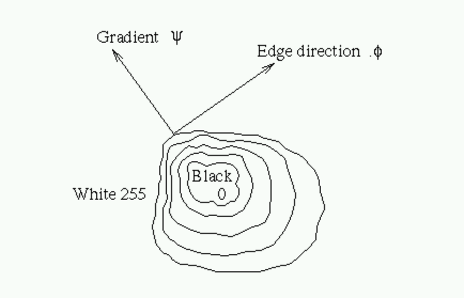
Edge by Gradient

- Calculus describes changes of continuous functions using derivatives.
- An image function depends on two variables — so operators describing edges are expressed using partial derivatives.
- A change of the image function can be described by a gradient that points in the direction of the largest growth of the image function.

- An edge is a (local) property attached to an individual pixel and is calculated from the image function in a neighborhood of the pixel.
- It is a `vector variable` with two components
  - magnitude of the gradient;
  - and direction ϕ is rotated with respect to the gradient direction ψ by -90^o.

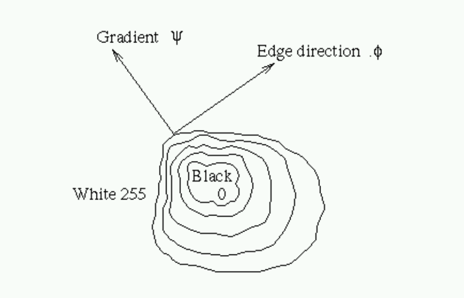
Edge and Gradient

- The gradient direction gives the direction of maximal growth of the function, e.g., from black ($g(i,j) = 0$) to white ($f(i,j) = 255$).
- This is illustrated below; closed contour lines are lines of the same brightness; the orientation 0^o points East.



Edge and Gradient

- The gradient direction gives the direction of maximal growth of the function, e.g., from black ($g(i,j) = 0$) to white ($f(i,j) = 255$).
- This is illustrated below; closed contour lines are lines of the same brightness; the orientation 0^o points East.



Edges and Boundaries

- Edges are often used in image analysis for finding region boundaries.
- Boundary is at the pixels where the image function varies and consists of pixels with `high edge magnitude`.
- Boundary and its parts (edges) are perpendicular to the direction of the gradient.

Edges and Boundaries

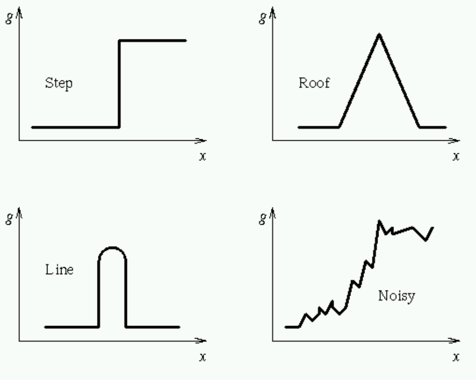
- Edges are often used in image analysis for finding region boundaries.
- Boundary is at the pixels where the image function varies and consists of pixels with `high edge magnitude`.
- Boundary and its parts (edges) are perpendicular to the direction of the gradient.

Edges and Boundaries

- Edges are often used in image analysis for finding region boundaries.
- Boundary is at the pixels where the image function varies and consists of pixels with `high edge magnitude`.
- Boundary and its parts (edges) are perpendicular to the direction of the gradient.

Some Edge Profiles

- The following figure shows several typical standard edge profiles.

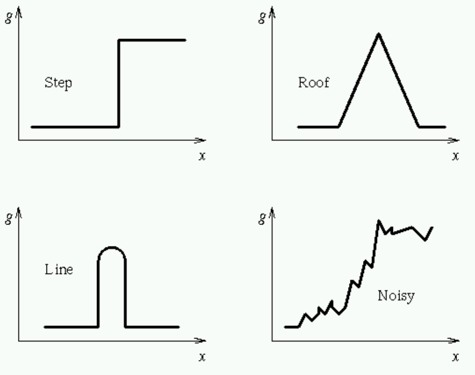


Typical edge profiles.

- Roof edges are typical for objects corresponding to thin lines in the image.
- Edge detectors are usually tuned for some type of edge profile.

Some Edge Profiles

- The following figure shows several typical standard edge profiles.

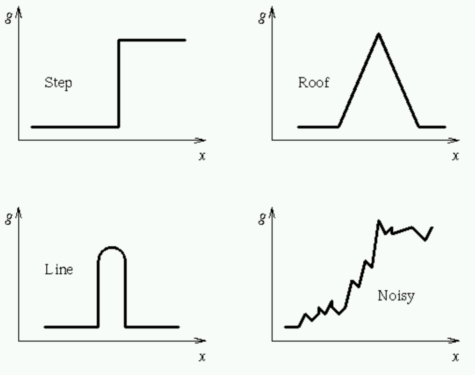


Typical edge profiles.

- Roof edges are typical for objects corresponding to thin lines in the image.
- Edge detectors are usually tuned for some type of edge profile.

Some Edge Profiles

- The following figure shows several typical standard edge profiles.



Typical edge profiles.

- Roof edges are typical for objects corresponding to thin lines in the image.
- Edge detectors are usually tuned for some type of edge profile.

Laplacian as an Edge Detector

- Sometimes we are interested only in changing magnitude without regard to the changing orientation.
- A linear differential operator called the Laplacian may be used.
- The Laplacian has the same properties in all directions and is therefore invariant to rotation in the image.

$$\Delta g(x,y) = \frac{\partial^2 g(x,y)}{\partial x^2} + \frac{\partial^2 g(x,y)}{\partial y^2} \quad (63)$$

Laplacian as an Edge Detector

- Sometimes we are interested only in changing magnitude without regard to the changing orientation.
- A linear differential operator called the Laplacian may be used.
- The Laplacian has the same properties in all directions and is therefore invariant to rotation in the image.

$$\Delta g(x,y) = \frac{\partial^2 g(x,y)}{\partial x^2} + \frac{\partial^2 g(x,y)}{\partial y^2} \tag{63}$$

Laplacian as an Edge Detector

- Sometimes we are interested only in changing magnitude without regard to the changing orientation.
- A linear differential operator called the Laplacian may be used.
- The Laplacian has the same properties in all directions and is therefore invariant to rotation in the image.

$$\Delta g(x,y) = \frac{\partial^2 g(x,y)}{\partial x^2} + \frac{\partial^2 g(x,y)}{\partial y^2} \tag{63}$$

Gradient

- The gradient magnitude and gradient direction are image functions,

$$|\text{grad} g(x,y)| = \sqrt{\left(\frac{\partial g}{\partial x}\right)^2 + \left(\frac{\partial g}{\partial y}\right)^2} \quad (64)$$

$$\psi = \arg\left(\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y}\right) \quad (65)$$

where $\arg(u,v) = \arctan(\frac{v}{u})$ is the angle (in radians) from the x-axis to the point (u,v).

- In practice, for fast computation, the magnitude is approximated by

$$|\text{grad} g(x,y)| = \left|\frac{\partial g}{\partial x}\right| + \left|\frac{\partial g}{\partial y}\right| \quad (66)$$

or

$$|\text{grad} g(x,y)| = \max\left\{\left|\frac{\partial g}{\partial x}\right|, \left|\frac{\partial g}{\partial y}\right|\right\} \quad (67)$$

Gradient

- The gradient magnitude and gradient direction are image functions,

$$|\text{grad}\, g(x, y)| = \sqrt{\left(\frac{\partial g}{\partial x}\right)^2 + \left(\frac{\partial g}{\partial y}\right)^2} \qquad (64)$$

$$\psi = \arg\left(\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y}\right) \qquad (65)$$

where $\arg(u, v) = \arctan(\frac{v}{u})$ is the angle (in radians) from the x-axis to the point (u, v).

- In practice, for fast computation, the magnitude is approximated by

$$|\text{grad}\, g(x, y)| = \left|\frac{\partial g}{\partial x}\right| + \left|\frac{\partial g}{\partial y}\right| \qquad (66)$$

or

$$|\text{grad}\, g(x, y)| = \max\left\{\left|\frac{\partial g}{\partial x}\right|, \left|\frac{\partial g}{\partial y}\right|\right\} \qquad (67)$$

Gradient by Finite Differences

- Derivatives must be approximated by finite differences for digital images.
- The first order differences of the image g can be approximated by backward difference

$$\Delta_i g(i,j) = \frac{g(i,j) - g(i-n,j)}{n} \qquad (68)$$

$$\Delta_j g(i,j) = \frac{g(i,j) - g(i,j-n)}{n} \qquad (69)$$

or by forward difference

$$\Delta_i g(i,j) = \frac{g(i+n,j) - g(i,j)}{n} \qquad (70)$$

$$\Delta_j g(i,j) = \frac{g(i,j+n) - g(i,j)}{n} \qquad (71)$$

- n is a small integer, usually 1.
- The value n should be chosen small enough to provide a good approximation to the derivative, but large enough to neglect unimportant changes in the image function.
- Central differences, are not usually used because they neglect the impact of the pixel (i,j) itself.

Gradient by Finite Differences

- Derivatives must be approximated by finite differences for digital images.
- The first order differences of the image g can be approximated by backward difference

$$\Delta_i g(i,j) = \frac{g(i,j) - g(i-n,j)}{n} \quad (68)$$

$$\Delta_j g(i,j) = \frac{g(i,j) - g(i,j-n)}{n} \quad (69)$$

or by forward difference

$$\Delta_i g(i,j) = \frac{g(i+n,j) - g(i,j)}{n} \quad (70)$$

$$\Delta_j g(i,j) = \frac{g(i,j+n) - g(i,j)}{n} \quad (71)$$

- n is a small integer, usually 1.
- The value n should be chosen small enough to provide a good approximation to the derivative, but large enough to neglect unimportant changes in the image function.
- Central differences, are not usually used because they neglect the impact of the pixel (i,j) itself.

Gradient by Finite Differences

- Derivatives must be approximated by finite differences for digital images.
- The first order differences of the image g can be approximated by backward difference

$$\Delta_i g(i,j) = \frac{g(i,j) - g(i-n,j)}{n} \quad (68)$$

$$\Delta_j g(i,j) = \frac{g(i,j) - g(i,j-n)}{n} \quad (69)$$

or by forward difference

$$\Delta_i g(i,j) = \frac{g(i+n,j) - g(i,j)}{n} \quad (70)$$

$$\Delta_j g(i,j) = \frac{g(i,j+n) - g(i,j)}{n} \quad (71)$$

- n is a small integer, usually 1.
- The value n should be chosen small enough to provide a good approximation to the derivative, but large enough to neglect unimportant changes in the image function.
- Central differences, are not usually used because they neglect the impact of the pixel (i,j) itself.

Gradient by Finite Differences

- Derivatives must be approximated by finite differences for digital images.
- The first order differences of the image g can be approximated by backward difference

$$\Delta_i g(i,j) = \frac{g(i,j) - g(i-n,j)}{n} \quad (68)$$

$$\Delta_j g(i,j) = \frac{g(i,j) - g(i,j-n)}{n} \quad (69)$$

or by forward difference

$$\Delta_i g(i,j) = \frac{g(i+n,j) - g(i,j)}{n} \quad (70)$$

$$\Delta_j g(i,j) = \frac{g(i,j+n) - g(i,j)}{n} \quad (71)$$

- n is a small integer, usually 1.
- The value n should be chosen small enough to provide a good approximation to the derivative, but large enough to neglect unimportant changes in the image function.
- Central differences, are not usually used because they neglect the impact of the pixel (i,j) itself.

Gradient by Finite Differences

- Derivatives must be approximated by finite differences for digital images.
- The first order differences of the image g can be approximated by backward difference

$$\Delta_i g(i,j) = \frac{g(i,j) - g(i-n,j)}{n} \quad (68)$$

$$\Delta_j g(i,j) = \frac{g(i,j) - g(i,j-n)}{n} \quad (69)$$

  or by forward difference

$$\Delta_i g(i,j) = \frac{g(i+n,j) - g(i,j)}{n} \quad (70)$$

$$\Delta_j g(i,j) = \frac{g(i,j+n) - g(i,j)}{n} \quad (71)$$

- n is a small integer, usually 1.
- The value n should be chosen small enough to provide a good approximation to the derivative, but large enough to neglect unimportant changes in the image function.
- Central differences, are not usually used because they neglect the impact of the pixel (i,j) itself.

Gradient by Convolution

- Individual gradient operators that examine small local neighborhoods are in fact linear space-invariant operators.
- They are hence are equivalent to convolutions, cf. (45), and can be expressed by convolution masks.
- Each convolution mask corresponds to a derivative in one certain direction, if the masks induces edge orientation information.

Gradient by Convolution

- Individual gradient operators that examine small local neighborhoods are in fact linear space-invariant operators.
- They are hence are equivalent to convolutions, cf. (45), and can be expressed by convolution masks.
- Each convolution mask corresponds to a derivative in one certain direction, if the masks induces edge orientation information.

Gradient by Convolution

- Individual gradient operators that examine small local neighborhoods are in fact linear space-invariant operators.
- They are hence are equivalent to convolutions, cf. (45), and can be expressed by convolution masks.
- Each convolution mask corresponds to a derivative in one certain direction, if the masks induces edge orientation information.

Gradient Operators

- Gradient operators as a measure of edge can be divided into three categories
  1. Operators approximating derivatives using finite differences:
     - Some are rotationally invariant (e.g., the Laplacian) and direction independent and thus need one convolution mask only.
     - Others approximate first derivatives using several masks.
  2. Operators based on the zero crossings of the second derivatives (e.g., Marr-Hildreth or Canny edge detector).
  3. Operators which attempt to match an image function to a parametric model of edges.
- It may be difficult to select the optimal edge detection strategy.

Gradient Operators

- Gradient operators as a measure of edge can be divided into three categories
  1. Operators approximating derivatives using finite differences:
     - Some are rotationally invariant (e.g., the Laplacian) and direction independent and thus need one convolution mask only.
     - Others approximate first derivatives using several masks.
  2. Operators based on the zero crossings of the second derivatives (e.g., Marr-Hildreth or Canny edge detector).
  3. Operators which attempt to match an image function to a parametric model of edges.
- It may be difficult to select the optimal edge detection strategy.

Gradient Operators

- Gradient operators as a measure of edge can be divided into three categories
  1. Operators approximating derivatives using finite differences:
     - Some are rotationally invariant (e.g., the Laplacian) and direction independent and thus need one convolution mask only.
     - Others approximate first derivatives using several masks.
  2. Operators based on the zero crossings of the second derivatives (e.g., Marr-Hildreth or Canny edge detector).
  3. Operators which attempt to match an image function to a parametric model of edges.
- It may be difficult to select the optimal edge detection strategy.

Gradient Operators

- Gradient operators as a measure of edge can be divided into three categories
  1. Operators approximating derivatives using finite differences:
     - Some are rotationally invariant (e.g., the Laplacian) and direction independent and thus need one convolution mask only.
     - Others approximate first derivatives using several masks.
  2. Operators based on the zero crossings of the second derivatives (e.g., Marr-Hildreth or Canny edge detector).
  3. Operators which attempt to match an image function to a parametric model of edges.
- It may be difficult to select the optimal edge detection strategy.

Gradient Operators

- Gradient operators as a measure of edge can be divided into three categories
  1. Operators approximating derivatives using finite differences:
     - Some are rotationally invariant (e.g., the Laplacian) and direction independent and thus need one convolution mask only.
     - Others approximate first derivatives using several masks.
  2. Operators based on the zero crossings of the second derivatives (e.g., Marr-Hildreth or Canny edge detector).
  3. Operators which attempt to match an image function to a parametric model of edges.
- It may be difficult to select the optimal edge detection strategy.

Gradient Operators

- Gradient operators as a measure of edge can be divided into three categories
  1. Operators approximating derivatives using finite differences:
     - Some are rotationally invariant (e.g., the Laplacian) and direction independent and thus need one convolution mask only.
     - Others approximate first derivatives using several masks.
  2. Operators based on the zero crossings of the second derivatives (e.g., Marr-Hildreth or Canny edge detector).
  3. Operators which attempt to match an image function to a parametric model of edges.
- It may be difficult to select the optimal edge detection strategy.

Gradient Operators

- Gradient operators as a measure of edge can be divided into three categories
  1. Operators approximating derivatives using finite differences:
     - Some are rotationally invariant (e.g., the Laplacian) and direction independent and thus need one convolution mask only.
     - Others approximate first derivatives using several masks.
  2. Operators based on the zero crossings of the second derivatives (e.g., Marr-Hildreth or Canny edge detector).
  3. Operators which attempt to match an image function to a parametric model of edges.
- It may be difficult to select the optimal edge detection strategy.

Gradient Operators with Multiple Masks

- The direction of the gradient is given by the mask giving maximal response.
- The absolute value of the response on that mask is the magnitude.
- Operators approximating first derivative of an image function are sometimes called `compass operators` because of its ability to determine gradient direction.

Gradient Operators with Multiple Masks

- The direction of the gradient is given by the mask giving maximal response.
- The absolute value of the response on that mask is the magnitude.
- Operators approximating first derivative of an image function are sometimes called `compass operators` because of its ability to determine gradient direction.

Gradient Operators with Multiple Masks

- The direction of the gradient is given by the mask giving maximal response.
- The absolute value of the response on that mask is the magnitude.
- Operators approximating first derivative of an image function are sometimes called `compass operators` because of its ability to determine gradient direction.

Roberts Operator

- The Roberts operator is one of the oldest operators.
- It is very easy to compute as it uses only a 2×2 neighborhood of the current pixel.
- Its convolution masks are

$$h_1 = \begin{bmatrix} h_1(0,0) & h_1(1,0) \\ h_1(0,1) & h_1(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{72}$$

- The magnitude of the edge is computed as

$$|g(i,j) - g(i+1,j+1)| + |g(i,j+1) - g(i+1,j)| \tag{73}$$

- The primary disadvantage of the Roberts operator is its high sensitivity to noise, because very few pixels are used to approximate the gradient.

Roberts Operator

- The Roberts operator is one of the oldest operators.
- It is very easy to compute as it uses only a 2×2 neighborhood of the current pixel.
- Its convolution masks are

$$h_1 = \begin{bmatrix} h_1(0,0) & h_1(1,0) \\ h_1(0,1) & h_1(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{72}$$

- The magnitude of the edge is computed as

$$|g(i,j) - g(i+1,j+1)| + |g(i,j+1) - g(i+1,j)| \tag{73}$$

- The primary disadvantage of the Roberts operator is its high sensitivity to noise, because very few pixels are used to approximate the gradient.

Roberts Operator

- The Roberts operator is one of the oldest operators.
- It is very easy to compute as it uses only a 2×2 neighborhood of the current pixel.
- Its convolution masks are

$$h_1 = \begin{bmatrix} h_1(0,0) & h_1(1,0) \\ h_1(0,1) & h_1(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{72}$$

- The magnitude of the edge is computed as

$$|g(i,j) - g(i+1,j+1)| + |g(i,j+1) - g(i+1,j)| \tag{73}$$

- The primary disadvantage of the Roberts operator is its high sensitivity to noise, because very few pixels are used to approximate the gradient.

Roberts Operator

- The Roberts operator is one of the oldest operators.
- It is very easy to compute as it uses only a 2×2 neighborhood of the current pixel.
- Its convolution masks are

$$h_1 = \begin{bmatrix} h_1(0,0) & h_1(1,0) \\ h_1(0,1) & h_1(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{72}$$

- The magnitude of the edge is computed as

$$|g(i,j) - g(i+1,j+1)| + |g(i,j+1) - g(i+1,j)| \tag{73}$$

- The primary disadvantage of the Roberts operator is its high sensitivity to noise, because very few pixels are used to approximate the gradient.

Roberts Operator

- The Roberts operator is one of the oldest operators.
- It is very easy to compute as it uses only a 2×2 neighborhood of the current pixel.
- Its convolution masks are

$$h_1 = \begin{bmatrix} h_1(0,0) & h_1(1,0) \\ h_1(0,1) & h_1(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{72}$$

- The magnitude of the edge is computed as

$$|g(i,j) - g(i+1,j+1)| + |g(i,j+1) - g(i+1,j)| \tag{73}$$

- The primary disadvantage of the Roberts operator is its high sensitivity to noise, because very few pixels are used to approximate the gradient.

Discrete Laplacian I

- The Laplace operator is a very popular operator approximating the second derivative which gives the gradient magnitude only.
- The Laplacian (63) is approximated in digital images by a convolution sum.
- A 3×3 mask h_4 is often used.
- For 4-neighborhoods, it is defined as

$$h_{4,1} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{or} \quad h_{4,2} = \frac{1}{2} \begin{bmatrix} 1 & 0 & 1 \\ 0 & -4 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{74}$$

Discrete Laplacian I

- The Laplace operator is a very popular operator approximating the second derivative which gives the gradient magnitude only.
- The Laplacian (63) is approximated in digital images by a convolution sum.
- A 3×3 mask h_4 is often used.
- For 4-neighborhoods, it is defined as

$$h_{4,1} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{or} \quad h_{4,2} = \frac{1}{2} \begin{bmatrix} 1 & 0 & 1 \\ 0 & -4 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{74}$$

Discrete Laplacian I

- The Laplace operator is a very popular operator approximating the second derivative which gives the gradient magnitude only.
- The Laplacian (63) is approximated in digital images by a convolution sum.
- A 3×3 mask h_4 is often used.
- For 4-neighborhoods, it is defined as

$$h_{4,1} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{or} \quad h_{4,2} = \frac{1}{2} \begin{bmatrix} 1 & 0 & 1 \\ 0 & -4 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{74}$$

Discrete Laplacian I

- The Laplace operator is a very popular operator approximating the second derivative which gives the gradient magnitude only.
- The Laplacian (63) is approximated in digital images by a convolution sum.
- A 3×3 mask h_4 is often used.
- For 4-neighborhoods, it is defined as

$$h_{4,1} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{or} \quad h_{4,2} = \frac{1}{2} \begin{bmatrix} 1 & 0 & 1 \\ 0 & -4 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{74}$$

Discrete Laplacian II

- For 8-neighborhoods, it is defined as

$$h_{8,1} = \frac{h_{4,1} + 2h_{4,2}}{3} = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (75)$$

or

$$h_{8,2} = \frac{4h_{4,2} - h_{4,1}}{3} = \frac{1}{3}\begin{bmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{bmatrix} \quad (76)$$

or

$$h_{8,3} = 3h_{4,1} - 2h_{4,2} = \begin{bmatrix} -1 & 3 & -1 \\ 3 & -8 & 3 \\ -1 & 3 & -1 \end{bmatrix} \quad (77)$$

- A Laplacian operator with stressed significance of the the central pixel or its neighborhood is sometimes used.

Discrete Laplacian II

- For 8-neighborhoods, it is defined as

$$h_{8,1} = \frac{h_{4,1} + 2h_{4,2}}{3} = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (75)$$

or

$$h_{8,2} = \frac{4h_{4,2} - h_{4,1}}{3} = \frac{1}{3}\begin{bmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{bmatrix} \quad (76)$$

or

$$h_{8,3} = 3h_{4,1} - 2h_{4,2} = \begin{bmatrix} -1 & 3 & -1 \\ 3 & -8 & 3 \\ -1 & 3 & -1 \end{bmatrix} \quad (77)$$

- A Laplacian operator with stressed significance of the the central pixel or its neighborhood is sometimes used.

Prewitt Operator

- The gradient is estimated in eight (for a 3×3 convolution mask) possible directions.
- Larger masks are possible.
- We present only the first three 3×3 masks for each operator; the others can be created by simple repeated clockwise 45^o rotation.

$$h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$$

$$h_3 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Prewitt Operator

- The gradient is estimated in eight (for a 3×3 convolution mask) possible directions.
- Larger masks are possible.
- We present only the first three 3×3 masks for each operator; the others can be created by simple repeated clockwise 45^o rotation.

$$h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$$

$$h_3 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Prewitt Operator

- The gradient is estimated in eight (for a 3×3 convolution mask) possible directions.
- Larger masks are possible.
- We present only the first three 3×3 masks for each operator; the others can be created by simple repeated clockwise 45^o rotation.

$$h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$$

$$h_3 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Sobel Operator

- $$h_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$$
  $$h_3 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$
- The Sobel operator is often used as a simple detector of horizontality and verticality of edges. In this case only masks h_1 and h_3 are used.

Sobel Operator

- $$h_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$$
$$h_3 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$
- The Sobel operator is often used as a simple detector of horizontality and verticality of edges. In this case only masks h_1 and h_3 are used.

Robinson Operator

- $$h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & -1 & 1 \end{bmatrix}$$

$$h_3 = \begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix}$$

Kirsch Operator

$$h_1 = \begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 3 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & -5 & 3 \end{bmatrix}$$

$$h_3 = \begin{bmatrix} -5 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & 3 & 3 \end{bmatrix}$$

Discussions

- Visually, the edge images produced by the foregoing edge operators (Roberts, Prewitt, Sobel, Kirsch, Robinson operators) appears rather similar.
- The Roberts operator, being 2×2, responds best on sharp transitions in low-noise images.
- The other operators, being 3×3, handle more gradual transition and noisier images better.

Discussions

- Visually, the edge images produced by the foregoing edge operators (Roberts, Prewitt, Sobel, Kirsch, Robinson operators) appears rather similar.
- The Roberts operator, being 2×2, responds best on sharp transitions in low-noise images.
- The other operators, being 3×3, handle more gradual transition and noisier images better.

Discussions

- Visually, the edge images produced by the foregoing edge operators (Roberts, Prewitt, Sobel, Kirsch, Robinson operators) appears rather similar.
- The Roberts operator, being 2×2, responds best on sharp transitions in low-noise images.
- The other operators, being 3×3, handle more gradual transition and noisier images better.

Image Sharpening

- Image sharpening makes edges steeper.
- The sharpened output image f is obtained from the input image g as

$$f(i,j) = g(i,j) - CS(i,j) \tag{78}$$

- C is a positive coefficient which gives the strength of sharpening and $S(i,j)$ is a measure of the image function sheerness that is calculated using a gradient operator.
- The Laplacian is very often used to estimate $S(i,j)$.
- Image sharpening/edge detection can be interpreted in the frequency domain as well.
- The result of the Fourier transform is a combination of harmonic functions.
- The derivative of the harmonic function $\sin(nx)$ is $n\cos(nx)$; thus the higher the frequency, the higher the magnitude of its derivative.

Image Sharpening

- Image sharpening makes edges steeper.
- The sharpened output image f is obtained from the input image g as

$$f(i,j) = g(i,j) - CS(i,j) \tag{78}$$

- C is a positive coefficient which gives the strength of sharpening and $S(i,j)$ is a measure of the image function sheerness that is calculated using a gradient operator.
- The Laplacian is very often used to estimate $S(i,j)$.
- Image sharpening/edge detection can be interpreted in the frequency domain as well.
- The result of the Fourier transform is a combination of harmonic functions.
- The derivative of the harmonic function $\sin(nx)$ is $n\cos(nx)$; thus the higher the frequency, the higher the magnitude of its derivative.

Image Sharpening

- Image sharpening makes edges steeper.
- The sharpened output image f is obtained from the input image g as

$$f(i,j) = g(i,j) - CS(i,j) \tag{78}$$

- C is a positive coefficient which gives the strength of sharpening and $S(i,j)$ is a measure of the image function sheerness that is calculated using a gradient operator.
- The Laplacian is very often used to estimate $S(i,j)$.
- Image sharpening/edge detection can be interpreted in the frequency domain as well.
- The result of the Fourier transform is a combination of harmonic functions.
- The derivative of the harmonic function $\sin(nx)$ is $n\cos(nx)$; thus the higher the frequency, the higher the magnitude of its derivative.

Image Sharpening

- Image sharpening makes edges steeper.
- The sharpened output image f is obtained from the input image g as

$$f(i,j) = g(i,j) - CS(i,j) \qquad (78)$$

- C is a positive coefficient which gives the strength of sharpening and $S(i,j)$ is a measure of the image function sheerness that is calculated using a gradient operator.
- The Laplacian is very often used to estimate $S(i,j)$.
- Image sharpening/edge detection can be interpreted in the frequency domain as well.
- The result of the Fourier transform is a combination of harmonic functions.
- The derivative of the harmonic function $\sin(nx)$ is $n\cos(nx)$; thus the higher the frequency, the higher the magnitude of its derivative.

Image Sharpening

- Image sharpening makes edges steeper.
- The sharpened output image f is obtained from the input image g as

$$f(i,j) = g(i,j) - CS(i,j) \tag{78}$$

- C is a positive coefficient which gives the strength of sharpening and $S(i,j)$ is a measure of the image function sheerness that is calculated using a gradient operator.
- The Laplacian is very often used to estimate $S(i,j)$.
- Image sharpening/edge detection can be interpreted in the frequency domain as well.
- The result of the Fourier transform is a combination of harmonic functions.
- The derivative of the harmonic function $\sin(nx)$ is $n\cos(nx)$; thus the higher the frequency, the higher the magnitude of its derivative.

Image Sharpening

- Image sharpening makes edges steeper.
- The sharpened output image f is obtained from the input image g as

$$f(i,j) = g(i,j) - CS(i,j) \tag{78}$$

- C is a positive coefficient which gives the strength of sharpening and $S(i,j)$ is a measure of the image function sheerness that is calculated using a gradient operator.
- The Laplacian is very often used to estimate $S(i,j)$.
- Image sharpening/edge detection can be interpreted in the frequency domain as well.
- The result of the Fourier transform is a combination of harmonic functions.
- The derivative of the harmonic function $\sin(nx)$ is $n\cos(nx)$; thus the higher the frequency, the higher the magnitude of its derivative.

Image Sharpening

- Image sharpening makes edges steeper.
- The sharpened output image f is obtained from the input image g as

$$f(i,j) = g(i,j) - CS(i,j) \tag{78}$$

- C is a positive coefficient which gives the strength of sharpening and $S(i,j)$ is a measure of the image function sheerness that is calculated using a gradient operator.
- The Laplacian is very often used to estimate $S(i,j)$.
- Image sharpening/edge detection can be interpreted in the frequency domain as well.
- The result of the Fourier transform is a combination of harmonic functions.
- The derivative of the harmonic function $\sin(nx)$ is $n\cos(nx)$; thus the higher the frequency, the higher the magnitude of its derivative.

Image Sharpening: 1D illustration

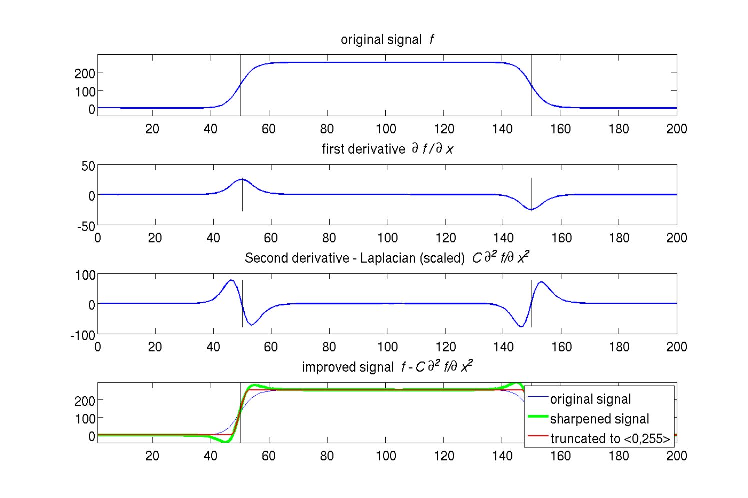


Figure: Image Sharpenging

Image Sharpening: Example I

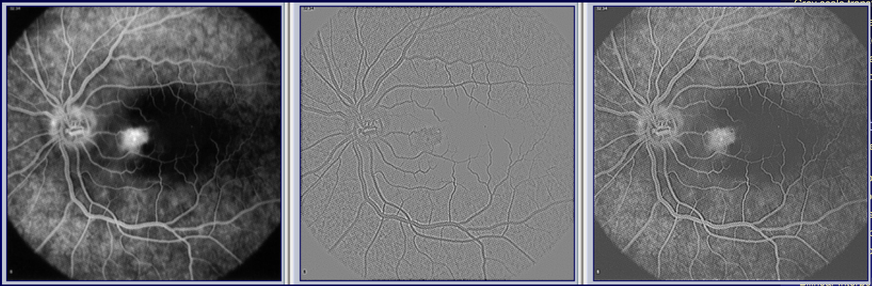


Figure: Image Sharpenging

Image Sharpening: Example II

- The matlab script from `visionbook` is is visionbook/05Preproc/imsharpen_demo.m.

Outline

Pixel Brightness Transformations
- Position dependent brightness correction
- Grey scale transformation
  - Windows and level
  - Histogram stretching
  - Histogram Equalization
  - Histogram Matching

Geometric Transformations
- Pixel Co-ordinate Transformations
  - Polynomial Approximation
  - Bilinear Transformations
  - Important Transformations
- Brightness Interpolation
  - Nearest Neighbor Interpolation
  - Bilinear Interpolation
  - Bi-cubic interpolation

Local pre-processing
- Image smoothing
  - Averaging
  - Averaging with Data Validity
  - Averaging According to Inverse Gradient

Marr's Theory

- In the 1970's, Marr's theory conclude from neurophysiological experiments that object boundaries are the most important cues that link an intensity image with its interpretation.
- Edge detection techniques at that time like the Kirsch, Sobel, Prewitt operators are based on convolution in very small neighborhoods and work well for specific images only.
- The main disadvantage of these edge detectors is their dependence on the size of objects and sensitivity to noise.

Marr's Theory

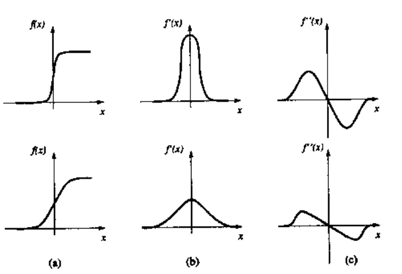
- In the 1970's, Marr's theory conclude from neurophysiological experiments that object boundaries are the most important cues that link an intensity image with its interpretation.
- Edge detection techniques at that time like the Kirsch, Sobel, Prewitt operators are based on convolution in very small neighborhoods and work well for specific images only.
- The main disadvantage of these edge detectors is their dependence on the size of objects and sensitivity to noise.

Marr's Theory

- In the 1970's, Marr's theory conclude from neurophysiological experiments that object boundaries are the most important cues that link an intensity image with its interpretation.
- Edge detection techniques at that time like the Kirsch, Sobel, Prewitt operators are based on convolution in very small neighborhoods and work well for specific images only.
- The main disadvantage of these edge detectors is their dependence on the size of objects and sensitivity to noise.

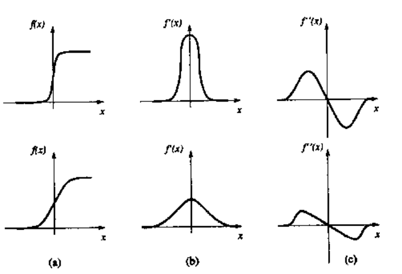
Zero-crossings

- An edge detection technique, based on the `zero crossings` of the second derivative (in its original form, the `Marr-Hildreth edge detector`) explores the fact that
  - a step edge corresponds to an abrupt change in the image function;
  - the first derivative of the image function should have an extreme at the position corresponding to the edge in the image;
  - the second derivative should be zero at the same position.
- It is much easier and more precise to find a zero crossing position than an extreme - see the follwoing figure.



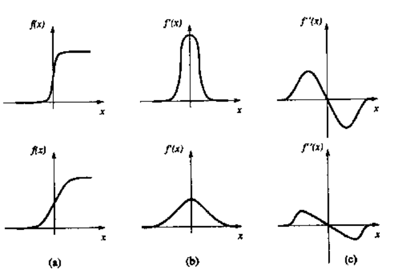
Zero-crossings

- An edge detection technique, based on the `zero crossings` of the second derivative (in its original form, the `Marr-Hildreth edge detector`) explores the fact that
  - a step edge corresponds to an abrupt change in the image function;
  - the first derivative of the image function should have an extreme at the position corresponding to the edge in the image;
  - the second derivative should be zero at the same position.
- It is much easier and more precise to find a zero crossing position than an extreme - see the follwoing figure.



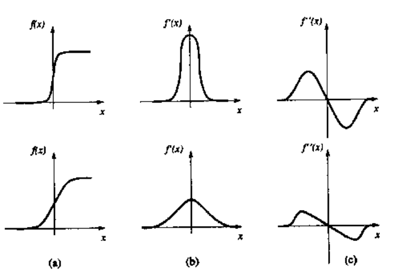
Zero-crossings

- An edge detection technique, based on the zero crossings of the second derivative (in its original form, the Marr-Hildreth edge detector) explores the fact that
  - a step edge corresponds to an abrupt change in the image function;
  - the first derivative of the image function should have an extreme at the position corresponding to the edge in the image;
  - the second derivative should be zero at the same position.
- It is much easier and more precise to find a zero crossing position than an extreme - see the follwoing figure.



Zero-crossings

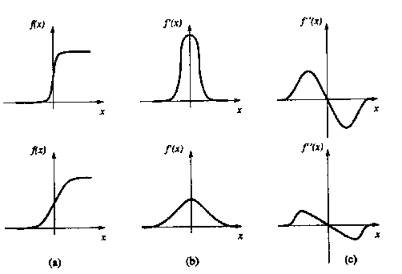
- An edge detection technique, based on the zero crossings of the second derivative (in its original form, the Marr-Hildreth edge detector) explores the fact that
  - a step edge corresponds to an abrupt change in the image function;
  - the first derivative of the image function should have an extreme at the position corresponding to the edge in the image;
  - the second derivative should be zero at the same position.
- It is much easier and more precise to find a zero crossing position than an extreme - see the follwoing figure.



Zero-crossings

- An edge detection technique, based on the zero crossings of the second derivative (in its original form, the Marr-Hildreth edge detector) explores the fact that
  - a step edge corresponds to an abrupt change in the image function;
  - the first derivative of the image function should have an extreme at the position corresponding to the edge in the image;
  - the second derivative should be zero at the same position.

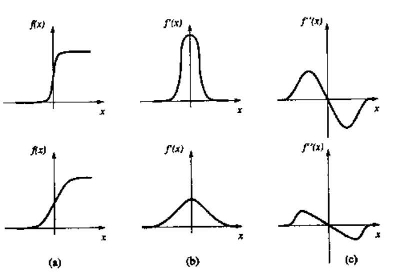
- It is much easier and more precise to find a zero crossing position than an extreme - see the follwoing figure.



Zero-crossings

- An edge detection technique, based on the zero crossings of the second derivative (in its original form, the Marr-Hildreth edge detector) explores the fact that
  - a step edge corresponds to an abrupt change in the image function;
  - the first derivative of the image function should have an extreme at the position corresponding to the edge in the image;
  - the second derivative should be zero at the same position.

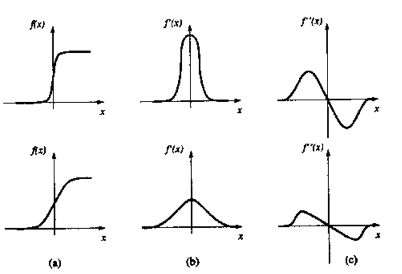
- It is much easier and more precise to find a zero crossing position than an extreme - see the follwoing figure.



Zero-crossings

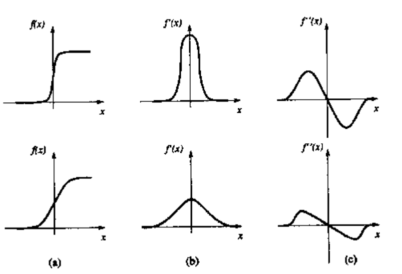
- An edge detection technique, based on the `zero crossings` of the second derivative (in its original form, the `Marr-Hildreth edge detector`) explores the fact that
  - a step edge corresponds to an abrupt change in the image function;
  - the first derivative of the image function should have an extreme at the position corresponding to the edge in the image;
  - the second derivative should be zero at the same position.

- It is much easier and more precise to find a zero crossing position than an extreme - see the follwoing figure.



Zero-crossings

- An edge detection technique, based on the `zero crossings` of the second derivative (in its original form, the `Marr-Hildreth edge detector`) explores the fact that
  - a step edge corresponds to an abrupt change in the image function;
  - the first derivative of the image function should have an extreme at the position corresponding to the edge in the image;
  - the second derivative should be zero at the same position.
- It is much easier and more precise to find a zero crossing position than an extreme - see the follwoing figure.



Smoothing Filter

- The crucial question is how to compute the the 2nd derivative robustly.
- One possibility is to smooth an image first (to reduce noise) and then compute second derivatives.
- When choosing a smoothing filter, there are two criteria that should be fulfilled, [Marr and Hildreth, 1980].
  1. The filter should be smooth and roughly band-limited in the frequency domain to reduce the possible number of frequencies at which function changes can take place.
  2. The constraint of spatial localization requires the response of a filter to be from nearby points in the image.

Smoothing Filter

- The crucial question is how to compute the the 2nd derivative robustly.
- One possibility is to smooth an image first (to reduce noise) and then compute second derivatives.
- When choosing a smoothing filter, there are two criteria that should be fulfilled, [Marr and Hildreth, 1980].
  1. The filter should be smooth and roughly band-limited in the frequency domain to reduce the possible number of frequencies at which function changes can take place.
  2. The constraint of spatial localization requires the response of a filter to be from nearby points in the image.

Smoothing Filter

- The crucial question is how to compute the the 2nd derivative robustly.
- One possibility is to smooth an image first (to reduce noise) and then compute second derivatives.
- When choosing a smoothing filter, there are two criteria that should be fulfilled, [Marr and Hildreth, 1980].
  1. The filter should be smooth and roughly band-limited in the frequency domain to reduce the possible number of frequencies at which function changes can take place.
  2. The constraint of spatial localization requires the response of a filter to be from nearby points in the image.

Smoothing Filter

- The crucial question is how to compute the the 2nd derivative robustly.
- One possibility is to smooth an image first (to reduce noise) and then compute second derivatives.
- When choosing a smoothing filter, there are two criteria that should be fulfilled, [Marr and Hildreth, 1980].
  1. The filter should be smooth and roughly band-limited in the frequency domain to reduce the possible number of frequencies at which function changes can take place.
  2. The constraint of spatial localization requires the response of a filter to be from nearby points in the image.

Smoothing Filter

- The crucial question is how to compute the the 2nd derivative robustly.
- One possibility is to smooth an image first (to reduce noise) and then compute second derivatives.
- When choosing a smoothing filter, there are two criteria that should be fulfilled, [Marr and Hildreth, 1980].
  1. The filter should be smooth and roughly band-limited in the frequency domain to reduce the possible number of frequencies at which function changes can take place.
  2. The constraint of spatial localization requires the response of a filter to be from nearby points in the image.

Uncertainty Principle

- These two criteria are conflicting — `uncertainty principle`.
- *A nonzero function and its Fourier transform cannot both be sharply localized*, [Folland and Sitaram, 1997, p. 207].
- But they can be optimized simultaneously using a Gaussian distribution.

Uncertainty Principle

- These two criteria are conflicting — `uncertainty principle`.
- *A nonzero function and its Fourier transform cannot both be sharply localized*, [Folland and Sitaram, 1997, p. 207].
- But they can be optimized simultaneously using a Gaussian distribution.

Uncertainty Principle

- These two criteria are conflicting — `uncertainty principle`.
- *A nonzero function and its Fourier transform cannot both be sharply localized*, [Folland and Sitaram, 1997, p. 207].
- But they can be optimized simultaneously using a Gaussian distribution.

Uncertainty Principle

- ([Folland and Sitaram, 1997, Theorem 1.1]) For any $f \in L^2(\mathbf{R})$ and any $a \in \mathbf{R}$ and $b \in \mathbf{R}$,

$$\int (x-a)^2 |f(x)|^2 \, dx \int (\xi - b)^2 |\hat{f}(\xi)|^2 \, d\xi \geq \frac{\|f\|_2^4}{16\pi^2}. \quad (79)$$

Equality holds if and only if $f = c\mathbf{e}^{2\pi i b x - \gamma(x-a)^2}$ for some $c \in \mathbf{C}$ and $\gamma > 0$.

Uncertainty Principle

- [Folland and Sitaram, 1997, Theorem 7.6] For a, $b > 0$, let $E(a, b)$ be the space of all measurable functions f on $\mathbf{R}$ such that

$$|f(x)| \leq ce^{-a\pi x^2}, \quad (80)$$

$$|\hat{f}(\xi)| \leq ce^{-b\pi \xi^2}, \quad (81)$$

for some $c > 0$. Then

(1) If $ab < 1$, $\dim E(a, b) = \infty$;
(2) If $ab = 1$, $E(a, b) = \mathbf{C}e^{-a\pi x^2}$;
(3) If $ab > 1$, $E(a, b) = \{0\}$;

Uncertainty Principle

- [Folland and Sitaram, 1997, Theorem 7.6] For a, $b > 0$, let $E(a, b)$ be the space of all measurable functions f on $\mathbf{R}$ such that

$$|f(x)| \leq ce^{-a\pi x^2}, \tag{80}$$

$$|\hat{f}(\xi)| \leq ce^{-b\pi \xi^2}, \tag{81}$$

for some $c > 0$. Then

(1) If $ab < 1$, $\dim E(a, b) = \infty$;

(2) If $ab = 1$, $E(a, b) = \mathbf{C}e^{-a\pi x^2}$;

(3) If $ab > 1$, $E(a, b) = \{0\}$;

Uncertainty Principle

- [Folland and Sitaram, 1997, Theorem 7.6] For a, $b > 0$, let $E(a,b)$ be the space of all measurable functions f on $\mathbf{R}$ such that

$$|f(x)| \le ce^{-a\pi x^2}, \tag{80}$$

$$|\hat{f}(\xi)| \le ce^{-b\pi \xi^2}, \tag{81}$$

for some $c > 0$. Then

(1) If $ab < 1$, $\dim E(a,b) = \infty$;
(2) If $ab = 1$, $E(a,b) = \mathbf{C}e^{-a\pi x^2}$;
(3) If $ab > 1$, $E(a,b) = \{0\}$;

Uncertainty Principle

- [Folland and Sitaram, 1997, Theorem 7.6] For a, $b > 0$, let $E(a, b)$ be the space of all measurable functions f on $\mathbf{R}$ such that

$$|f(x)| \leq c e^{-a\pi x^2}, \tag{80}$$

$$|\hat{f}(\xi)| \leq c e^{-b\pi \xi^2}, \tag{81}$$

for some $c > 0$. Then

(1) If $ab < 1$, $\dim E(a, b) = \infty$;
(2) If $ab = 1$, $E(a, b) = \mathbf{C} e^{-a\pi x^2}$;
(3) If $ab > 1$, $E(a, b) = \{0\}$;

Uncertainty Principle

There is a well known joke:

- "Heisenberg is pulled over by a policeman whilst driving down a motorway, the policeman gets out of his car, walks towards Heisenberg's window and motions with his hand for Heisenberg to wind the window down, which he does. The policeman then says 'Do you know what speed you were driving at sir?', to which Heisenberg responds 'No, but I knew exactly where I was.'"[1]

[1] This is copy-edited from http://en.wikipedia.org/wiki/Uncertainty_principle.

2D Gaussian Smoothing

- The 2D Gaussian smoothing operator $G(x, y)$

$$G(x, y) = \frac{1}{2\pi\sigma^2} \mathbf{e}^{-\frac{x^2+y^2}{2\sigma^2}}, \qquad (82)$$

where x and y are the image co-ordinates and σ is the standard deviation of the associated probability distribution.

- The standard deviation σ is the only parameter of the Gaussian filter — it is proportional to the size of neighborhood on which the filter operates.
- Pixels more distant from the center of the operator have smaller influence, and pixels further than 3σ from the center have negligible influence.

2D Gaussian Smoothing

- The `2D Gaussian smoothing operator` $G(x, y)$

$$G(x,y) = \frac{1}{2\pi\sigma^2} \mathbf{e}^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{82}$$

  where x and y are the image co-ordinates and σ is the standard deviation of the associated probability distribution.
- The standard deviation σ is the only parameter of the Gaussian filter — it is proportional to the size of neighborhood on which the filter operates.
- Pixels more distant from the center of the operator have smaller influence, and pixels further than 3σ from the center have negligible influence.

2D Gaussian Smoothing

- The `2D Gaussian smoothing operator` $G(x, y)$

$$G(x,y) = \frac{1}{2\pi\sigma^2} \mathbf{e}^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{82}$$

  where x and y are the image co-ordinates and σ is the standard deviation of the associated probability distribution.
- The standard deviation σ is the only parameter of the Gaussian filter — it is proportional to the size of neighborhood on which the filter operates.
- Pixels more distant from the center of the operator have smaller influence, and pixels further than 3σ from the center have negligible influence.

LoG: Laplacian of Gaussian

- Our goal is to get a second derivative of a smoothed 2D function $f(x, y)$.
- The Laplacian operator gives the second derivative, and is non-directional (isotropic).
- Consider then the Laplacian of an image $f(x, y)$ smoothed by a Gaussian.
- This operator is abbreviated by some authors as `LoG`, from `Laplacian of Gaussian`:

$$\Delta[G(x, y) * f(x, y)] \tag{83}$$

- The order of differentiation and convolution can be interchanged due to linearity of the operations:

$$[\Delta G(x, y)] * f(x, y) \tag{84}$$

LoG: Laplacian of Gaussian

- Our goal is to get a second derivative of a smoothed 2D function $f(x, y)$.
- The Laplacian operator gives the second derivative, and is non-directional (isotropic).

- Consider then the Laplacian of an image $f(x, y)$ smoothed by a Gaussian.
- This operator is abbreviated by some authors as `LoG`, from `Laplacian of Gaussian`:

$$\Delta[G(x, y) * f(x, y)] \quad (83)$$

- The order of differentiation and convolution can be interchanged due to linearity of the operations:

$$[\Delta G(x, y)] * f(x, y) \quad (84)$$

LoG: Laplacian of Gaussian

- Our goal is to get a second derivative of a smoothed 2D function $f(x, y)$.
- The Laplacian operator gives the second derivative, and is non-directional (isotropic).

- Consider then the Laplacian of an image $f(x, y)$ smoothed by a Gaussian.
- This operator is abbreviated by some authors as `LoG`, from `Laplacian of Gaussian`:

$$\Delta[G(x, y) * f(x, y)] \tag{83}$$

- The order of differentiation and convolution can be interchanged due to linearity of the operations:

$$[\Delta G(x, y)] * f(x, y) \tag{84}$$

LoG: Laplacian of Gaussian

- Our goal is to get a second derivative of a smoothed 2D function $f(x, y)$.
- The Laplacian operator gives the second derivative, and is non-directional (isotropic).

- Consider then the Laplacian of an image $f(x, y)$ smoothed by a Gaussian.
- This operator is abbreviated by some authors as `LoG`, from `Laplacian of Gaussian`:

$$\Delta[G(x, y) * f(x, y)] \tag{83}$$

- The order of differentiation and convolution can be interchanged due to linearity of the operations:

$$[\Delta G(x, y)] * f(x, y) \tag{84}$$

LoG: Laplacian of Gaussian

- Our goal is to get a second derivative of a smoothed 2D function $f(x, y)$.
- The Laplacian operator gives the second derivative, and is non-directional (isotropic).

- Consider then the Laplacian of an image $f(x, y)$ smoothed by a Gaussian.
- This operator is abbreviated by some authors as `LoG`, from `Laplacian of Gaussian`:

$$\Delta[G(x, y) * f(x, y)] \tag{83}$$

- The order of differentiation and convolution can be interchanged due to linearity of the operations:

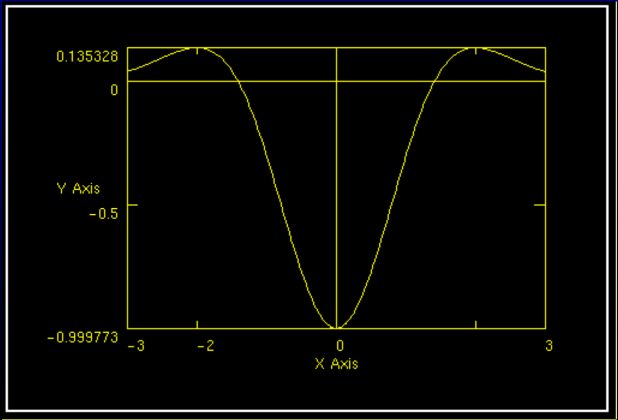
$$[\Delta G(x, y)] * f(x, y) \tag{84}$$

LoG: Laplacian of Gaussian

- The derivative of the Gaussian filter can be precomputed analytically

$$\Delta G(x,y) = \frac{x^2 + y^2 - 2\sigma^2}{2\pi\sigma^6} \mathbf{e}^{-\frac{r^2}{2\sigma^2}} \qquad (85)$$

- Because its shape, the LoG operator is commonly called a Mexican hat.

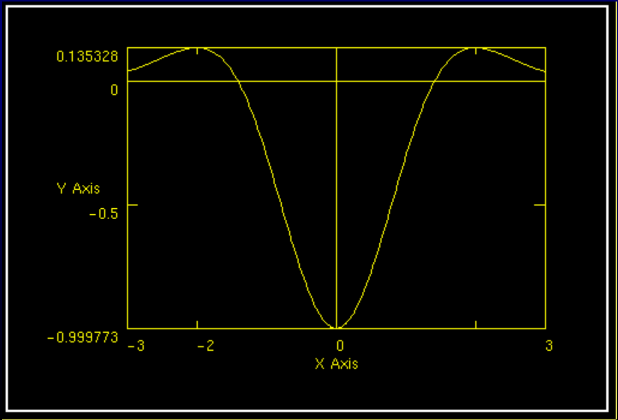


LoG: Laplacian of Gaussian

- The derivative of the Gaussian filter can be precomputed analytically

$$\Delta G(x,y) = \frac{x^2 + y^2 - 2\sigma^2}{2\pi\sigma^6} \mathbf{e}^{-\frac{r^2}{2\sigma^2}} \tag{85}$$

- Because its shape, the LoG operator is commonly called a Mexican hat.



LoG: Discussions

- Gaussian smoothing effectively suppresses the influence of the pixels that are up to a distance 3σ from the current pixel; then the Laplace operator is an efficient and stable measure of changes in the image.
- The location in the LoG image where the zero level is crossed corresponds to the position of the edges.
- The advantage of this approach compared to classical edge operators of small size is that a larger area surrounding the current pixel is taken into account; the influence of more distant points decreases according to the variance σ of the Gaussian.

LoG: Discussions

- Gaussian smoothing effectively suppresses the influence of the pixels that are up to a distance 3σ from the current pixel; then the Laplace operator is an efficient and stable measure of changes in the image.
- The location in the LoG image where the zero level is crossed corresponds to the position of the edges.
- The advantage of this approach compared to classical edge operators of small size is that a larger area surrounding the current pixel is taken into account; the influence of more distant points decreases according to the variance σ of the Gaussian.

LoG: Discussions

- Gaussian smoothing effectively suppresses the influence of the pixels that are up to a distance 3σ from the current pixel; then the Laplace operator is an efficient and stable measure of changes in the image.
- The location in the LoG image where the zero level is crossed corresponds to the position of the edges.
- The advantage of this approach compared to classical edge operators of small size is that a larger area surrounding the current pixel is taken into account; the influence of more distant points decreases according to the variance σ of the Gaussian.

LoG to DoG

- Convolution masks become large for larger σ.
- Fortunately, there is a separable decomposition of the ΔG operator that can speed up computation considerable.
- If only globally significant edges are required, the standard deviation of the Gaussian smoothing filter may be increased, having the effect of suppressing less significant evidence.
- The LoG operator can be very effectively approximated by convolution with a mask that is the difference of two Gaussian averaging masks with substantially different — this method is called the `Difference of Gaussians` — `DoG`.
- Even coarser approximations to LoG are sometimes used — the image is filtered twice by an averaging operator with smoothing masks of different size and the difference image is produced.

LoG to DoG

- Convolution masks become large for larger σ.
- Fortunately, there is a separable decomposition of the ΔG operator that can speed up computation considerable.
- If only globally significant edges are required, the standard deviation of the Gaussian smoothing filter may be increased, having the effect of suppressing less significant evidence.
- The LoG operator can be very effectively approximated by convolution with a mask that is the difference of two Gaussian averaging masks with substantially different — this method is called the `Difference of Gaussians` — DoG.
- Even coarser approximations to LoG are sometimes used — the image is filtered twice by an averaging operator with smoothing masks of different size and the difference image is produced.

LoG to DoG

- Convolution masks become large for larger σ.
- Fortunately, there is a separable decomposition of the ΔG operator that can speed up computation considerable.
- If only globally significant edges are required, the standard deviation of the Gaussian smoothing filter may be increased, having the effect of suppressing less significant evidence.
- The LoG operator can be very effectively approximated by convolution with a mask that is the difference of two Gaussian averaging masks with substantially different — this method is called the `Difference of Gaussians` — `DoG`.
- Even coarser approximations to LoG are sometimes used — the image is filtered twice by an averaging operator with smoothing masks of different size and the difference image is produced.

LoG to DoG

- Convolution masks become large for larger σ.
- Fortunately, there is a separable decomposition of the ΔG operator that can speed up computation considerable.
- If only globally significant edges are required, the standard deviation of the Gaussian smoothing filter may be increased, having the effect of suppressing less significant evidence.
- The LoG operator can be very effectively approximated by convolution with a mask that is the difference of two Gaussian averaging masks with substantially different — this method is called the `Difference of Gaussians` — `DoG`.
- Even coarser approximations to LoG are sometimes used — the image is filtered twice by an averaging operator with smoothing masks of different size and the difference image is produced.

LoG to DoG

- Convolution masks become large for larger σ.
- Fortunately, there is a separable decomposition of the ΔG operator that can speed up computation considerable.
- If only globally significant edges are required, the standard deviation of the Gaussian smoothing filter may be increased, having the effect of suppressing less significant evidence.
- The LoG operator can be very effectively approximated by convolution with a mask that is the difference of two Gaussian averaging masks with substantially different — this method is called the `Difference of Gaussians` — `DoG`.
- Even coarser approximations to LoG are sometimes used — the image is filtered twice by an averaging operator with smoothing masks of different size and the difference image is produced.

Implementation

- When implementing a zero-crossing edge detector, trying to detect zeros in the LoG or DoG image will inevitably fail,
- while naive approaches of thresholding the LoG/DoG image and defining the zero-crossings in some interval of values close to zero give piecewise disconnected edges at best.
- Many other approaches improving zero-crossing performance can be found in the literature.

Implementation

- When implementing a zero-crossing edge detector, trying to detect zeros in the LoG or DoG image will inevitably fail,
- while naive approaches of thresholding the LoG/DoG image and defining the zero-crossings in some interval of values close to zero give piecewise disconnected edges at best.
- Many other approaches improving zero-crossing performance can be found in the literature.

Implementation

- When implementing a zero-crossing edge detector, trying to detect zeros in the LoG or DoG image will inevitably fail,
- while naive approaches of thresholding the LoG/DoG image and defining the zero-crossings in some interval of values close to zero give piecewise disconnected edges at best.
- Many other approaches improving zero-crossing performance can be found in the literature.

Disadvantages

- The traditional second-derivative zero-crossing has disadvantages as well:
  - it smooths the shape too much; for example, sharp corners are lost.
  - it tends to create closed loops of edges (nicknamed the 'plate of spaghetti' effect).
- Neurophysiological experiments provide evidence that the human retina operation on image can be described analytically as the convolution of the image with the ΔG operator.

Disadvantages

- The traditional second-derivative zero-crossing has disadvantages as well:
  - it smooths the shape too much; for example, sharp corners are lost.
  - it tends to create closed loops of edges (nicknamed the 'plate of spaghetti' effect).

- Neurophysiological experiments provide evidence that the human retina operation on image can be described analytically as the convolution of the image with the ΔG operator.

Disadvantages

- The traditional second-derivative zero-crossing has disadvantages as well:
  - it smooths the shape too much; for example, sharp corners are lost.
  - it tends to create closed loops of edges (nicknamed the 'plate of spaghetti' effect).
- Neurophysiological experiments provide evidence that the human retina operation on image can be described analytically as the convolution of the image with the ΔG operator.

Disadvantages

- The traditional second-derivative zero-crossing has disadvantages as well:
  - it smooths the shape too much; for example, sharp corners are lost.
  - it tends to create closed loops of edges (nicknamed the ‘plate of spaghetti’ effect).
- Neurophysiological experiments provide evidence that the human retina operation on image can be described analytically as the convolution of the image with the ΔG operator.

Outline

Scale and Object Size

- Many image processing techniques work locally, theoretically at the level of individual pixels — edge detection methods are examples.
- The essential problem in such computation is `scale`.
- Edges correspond to the gradient of the image function that is computed as a difference among pixels in some neighborhood.
- There is seldom a sound reason for choosing a particular size of neighborhood:
  - The ‘right’ size depends on the size of the objects under investigation.
  - To know what the objects are assumes that it is clear how to interpret an image and this is not in general known at the pre-processing stage.

Scale and Object Size

- Many image processing techniques work locally, theoretically at the level of individual pixels — edge detection methods are examples.
- The essential problem in such computation is `scale`.

- Edges correspond to the gradient of the image function that is computed as a difference among pixels in some neighborhood.
- There is seldom a sound reason for choosing a particular size of neighborhood:
  - The ‘right’ size depends on the size of the objects under investigation.
  - To know what the objects are assumes that it is clear how to interpret an image and this is not in general known at the pre-processing stage.

Scale and Object Size

- Many image processing techniques work locally, theoretically at the level of individual pixels — edge detection methods are examples.
- The essential problem in such computation is `scale`.

- Edges correspond to the gradient of the image function that is computed as a difference among pixels in some neighborhood.
- There is seldom a sound reason for choosing a particular size of neighborhood:
  - The 'right' size depends on the size of the objects under investigation.
  - To know what the objects are assumes that it is clear how to interpret an image and this is not in general known at the pre-processing stage.

Scale and Object Size

- Many image processing techniques work locally, theoretically at the level of individual pixels — edge detection methods are examples.
- The essential problem in such computation is `scale`.

- Edges correspond to the gradient of the image function that is computed as a difference among pixels in some neighborhood.
- There is seldom a sound reason for choosing a particular size of neighborhood:
  - The 'right' size depends on the size of the objects under investigation.
  - To know what the objects are assumes that it is clear how to interpret an image and this is not in general known at the pre-processing stage.

Scale and Object Size

- Many image processing techniques work locally, theoretically at the level of individual pixels — edge detection methods are examples.
- The essential problem in such computation is `scale`.

- Edges correspond to the gradient of the image function that is computed as a difference among pixels in some neighborhood.
- There is seldom a sound reason for choosing a particular size of neighborhood:
  - The 'right' size depends on the size of the objects under investigation.
  - To know what the objects are assumes that it is clear how to interpret an image and this is not in general known at the pre-processing stage.

Scale and Object Size

- Many image processing techniques work locally, theoretically at the level of individual pixels — edge detection methods are examples.
- The essential problem in such computation is `scale`.

- Edges correspond to the gradient of the image function that is computed as a difference among pixels in some neighborhood.
- There is seldom a sound reason for choosing a particular size of neighborhood:
  - The 'right' size depends on the size of the objects under investigation.
  - To know what the objects are assumes that it is clear how to interpret an image and this is not in general known at the pre-processing stage.

Scale and Representations

- The phenomenon under investigation can be expressed at different resolutions of the description, and a formal model is created at each resolution.
- Then the qualitative behavior of the model is studied under changing resolution of the description.
- Such a methodology enables the deduction of meta-knowledge about the phenomenon that is not seen at the individual description.
- Different description levels are easily interpreted as different `scales` in the domain of digital images.

Scale and Representations

- The phenomenon under investigation can be expressed at different resolutions of the description, and a formal model is created at each resolution.
- Then the qualitative behavior of the model is studied under changing resolution of the description.
- Such a methodology enables the deduction of meta-knowledge about the phenomenon that is not seen at the individual description.
- Different description levels are easily interpreted as different `scales` in the domain of digital images.

Scale and Representations

- The phenomenon under investigation can be expressed at different resolutions of the description, and a formal model is created at each resolution.
- Then the qualitative behavior of the model is studied under changing resolution of the description.
- Such a methodology enables the deduction of meta-knowledge about the phenomenon that is not seen at the individual description.
- Different description levels are easily interpreted as different `scales` in the domain of digital images.

Scale and Representations

- The phenomenon under investigation can be expressed at different resolutions of the description, and a formal model is created at each resolution.
- Then the qualitative behavior of the model is studied under changing resolution of the description.
- Such a methodology enables the deduction of meta-knowledge about the phenomenon that is not seen at the individual description.
- Different description levels are easily interpreted as different `scales` in the domain of digital images.

Scale and Marr's Edge Detection

- The idea of scale is fundamental to Marr's edge detection technique, introduced in the previous sub-section, where different scales are provided by different sizes of Gaussian filter masks.
- The aim there was not only to eliminate fine scale noise but also to separate events at different scales arising from distinct physical processes [Marr, 1982].

Scale and Marr's Edge Detection

- The idea of scale is fundamental to Marr's edge detection technique, introduced in the previous sub-section, where different scales are provided by different sizes of Gaussian filter masks.
- The aim there was not only to eliminate fine scale noise but also to separate events at different scales arising from distinct physical processes [Marr, 1982].

Representation by Scales

- Assume that a signal has been smoothed with several masks of variable sizes.
- Every setting of the scale parameters implies a different description, but it is not known which one is correct.
- For many tasks, no one scale is categorically correct.
- If the ambiguity introduced by the scale is inescapable, the goal of scale-independent description is to reduce this ambiguity as much as possible.

Representation by Scales

- Assume that a signal has been smoothed with several masks of variable sizes.
- Every setting of the scale parameters implies a different description, but it is not known which one is correct.
- For many tasks, no one scale is categorically correct.
- If the ambiguity introduced by the scale is inescapable, the goal of scale-independent description is to reduce this ambiguity as much as possible.

Representation by Scales

- Assume that a signal has been smoothed with several masks of variable sizes.
- Every setting of the scale parameters implies a different description, but it is not known which one is correct.
- For many tasks, no one scale is categorically correct.
- If the ambiguity introduced by the scale is inescapable, the goal of scale-independent description is to reduce this ambiguity as much as possible.

Representation by Scales

- Assume that a signal has been smoothed with several masks of variable sizes.
- Every setting of the scale parameters implies a different description, but it is not known which one is correct.
- For many tasks, no one scale is categorically correct.
- If the ambiguity introduced by the scale is inescapable, the goal of scale-independent description is to reduce this ambiguity as much as possible.

Scale Space

- Many publications tackle `scale-space` problems.
- Here we shall consider just three examples of the application of multiple scale description to image analysis.
- There are other approaches involving non-linear partial differential equations to generate non-linear scale-space descriptions.

Scale Space

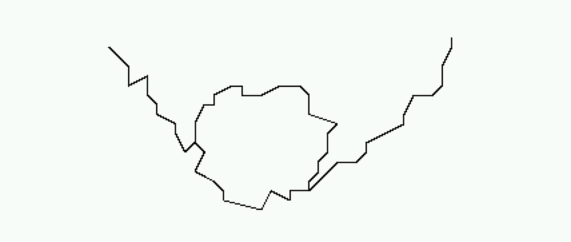
- Many publications tackle `scale-space` problems.
- Here we shall consider just three examples of the application of multiple scale description to image analysis.
- There are other approaches involving non-linear partial differential equations to generate non-linear scale-space descriptions.

Scale Space

- Many publications tackle `scale-space` problems.
- Here we shall consider just three examples of the application of multiple scale description to image analysis.
- There are other approaches involving non-linear partial differential equations to generate non-linear scale-space descriptions.

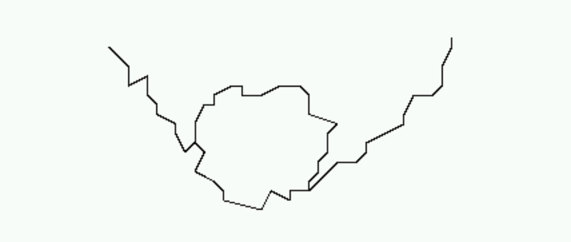
Curve Example I

- The first approach aimed to process planar noisy curves at a range of scales — the segment of curve that represents the underlying structure of the scene needs to be found.
- The problem is illustrated in by an example of two noisy curves in the following figure.



Curve Example I

- The first approach aimed to process planar noisy curves at a range of scales — the segment of curve that represents the underlying structure of the scene needs to be found.
- The problem is illustrated in by an example of two noisy curves in the following figure.



Curve Example II

- One of these may be interpreted as a closed (perhaps circular) curve, while the other could be described as two intersecting straight lines.
- Local tangent direction and curvature of the curve are significant only with some scales after the curve is smoothed by Gaussian filter with varying standard deviations.
- After smoothing using the Gaussian filter with varying standard deviations, the significant segments of the original curve can be found.

Curve Example II

- One of these may be interpreted as a closed (perhaps circular) curve, while the other could be described as two intersecting straight lines.
- Local tangent direction and curvature of the curve are significant only with some scales after the curve is smoothed by Gaussian filter with varying standard deviations.
- After smoothing using the Gaussian filter with varying standard deviations, the significant segments of the original curve can be found.

Curve Example II

- One of these may be interpreted as a closed (perhaps circular) curve, while the other could be described as two intersecting straight lines.
- Local tangent direction and curvature of the curve are significant only with some scales after the curve is smoothed by Gaussian filter with varying standard deviations.
- After smoothing using the Gaussian filter with varying standard deviations, the significant segments of the original curve can be found.

Scale Space Filtering I

- The second approach, `scale space filtering`, is to describe signals qualitatively with respect to scale.
- The problem is formulated for 1D signals $f(x)$, but it could be generalized to images.
- The original 1D signal f(x) is smoothed by convolution with a 1D Gaussian

$$f(x, \sigma) = G(x, \sigma) * f(x) \tag{86}$$

- If σ is changed, the function $f(x, \sigma)$ represents a surface on the (x, σ) plane that is called the `scale-space image`.
- Inflection points of the curve $f(x, \sigma_0)$ for a distinct value σ_0

$$\frac{\partial^2 f(x, \sigma_0)}{\partial x^2} = 0, \qquad \frac{\partial^3 f(x, \sigma_0)}{\partial x^3} \neq 0. \tag{87}$$

describe the curve $f(x)$ qualitatively.

Scale Space Filtering I

- The second approach, `scale space filtering`, is to describe signals qualitatively with respect to scale.
- The problem is formulated for 1D signals $f(x)$, but it could be generalized to images.
- The original 1D signal f(x) is smoothed by convolution with a 1D Gaussian

$$f(x, \sigma) = G(x, \sigma) * f(x) \tag{86}$$

- If σ is changed, the function $f(x, \sigma)$ represents a surface on the (x, σ) plane that is called the `scale-space image`.
- Inflection points of the curve $f(x, \sigma_0)$ for a distinct value σ_0

$$\frac{\partial^2 f(x, \sigma_0)}{\partial x^2} = 0, \qquad \frac{\partial^3 f(x, \sigma_0)}{\partial x^3} \neq 0. \tag{87}$$

describe the curve $f(x)$ qualitatively.

Scale Space Filtering I

- The second approach, `scale space filtering`, is to describe signals qualitatively with respect to scale.
- The problem is formulated for 1D signals $f(x)$, but it could be generalized to images.
- The original 1D signal f(x) is smoothed by convolution with a 1D Gaussian

$$f(x, \sigma) = G(x, \sigma) * f(x) \tag{86}$$

- If σ is changed, the function $f(x, \sigma)$ represents a surface on the (x, σ) plane that is called the `scale-space image`.
- Inflection points of the curve $f(x, \sigma_0)$ for a distinct value σ_0

$$\frac{\partial^2 f(x, \sigma_0)}{\partial x^2} = 0, \qquad \frac{\partial^3 f(x, \sigma_0)}{\partial x^3} \neq 0. \tag{87}$$

describe the curve $f(x)$ qualitatively.

Scale Space Filtering I

- The second approach, `scale space filtering`, is to describe signals qualitatively with respect to scale.
- The problem is formulated for 1D signals $f(x)$, but it could be generalized to images.
- The original 1D signal f(x) is smoothed by convolution with a 1D Gaussian

$$f(x, \sigma) = G(x, \sigma) * f(x) \quad (86)$$

- If σ is changed, the function $f(x, \sigma)$ represents a surface on the (x, σ) plane that is called the `scale-space image`.
- Inflection points of the curve $f(x, \sigma_0)$ for a distinct value σ_0

$$\frac{\partial^2 f(x, \sigma_0)}{\partial x^2} = 0, \qquad \frac{\partial^3 f(x, \sigma_0)}{\partial x^3} \neq 0. \quad (87)$$

describe the curve $f(x)$ qualitatively.

Scale Space Filtering I

- The second approach, `scale space filtering`, is to describe signals qualitatively with respect to scale.
- The problem is formulated for 1D signals $f(x)$, but it could be generalized to images.
- The original 1D signal f(x) is smoothed by convolution with a 1D Gaussian

$$f(x, \sigma) = G(x, \sigma) * f(x) \tag{86}$$

- If σ is changed, the function $f(x, \sigma)$ represents a surface on the (x, σ) plane that is called the `scale-space image`.
- Inflection points of the curve $f(x, \sigma_0)$ for a distinct value σ_0

$$\frac{\partial^2 f(x, \sigma_0)}{\partial x^2} = 0, \qquad \frac{\partial^3 f(x, \sigma_0)}{\partial x^3} \neq 0. \tag{87}$$

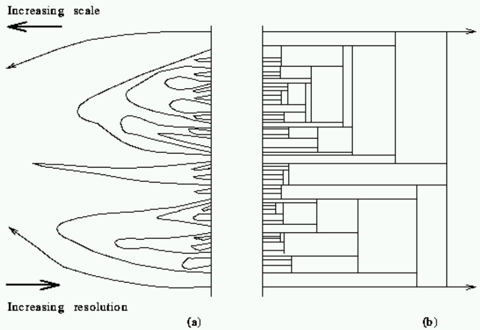
describe the curve $f(x)$ qualitatively.

Scale Space Filtering II

- The positions of inflection points can be drawn as a set of curves

$$\Sigma(x, \sigma_0) \tag{88}$$

in (x, σ) co-ordinates



Scale-space image:

(a) Varying number and locations of curve segmentation points as a function of scale,

(b) curve representation by an interval tree.

Scale Space Filtering III

- Coarse to fine analysis of the curves corresponding to inflection points, i.e., in the direction of the decreasing value of the σ, localizes events at different scales.
- The qualitative information contained in the scale-space image can be transformed into a simple `interval tree` that expresses the structure of the signal $f(x)$ over all (observed) scales.
- The interval tree is built from the root that corresponds to the largest scale.
- Then the scale-space image is searched in the direction of decreasing σ.
- The interval tree branches at those points where new curves corresponding inflection points appears.

Scale Space Filtering III

- Coarse to fine analysis of the curves corresponding to inflection points, i.e., in the direction of the decreasing value of the σ, localizes events at different scales.
- The qualitative information contained in the scale-space image can be transformed into a simple `interval tree` that expresses the structure of the signal $f(x)$ over all (observed) scales.
- The interval tree is built from the root that corresponds to the largest scale.
- Then the scale-space image is searched in the direction of decreasing σ.
- The interval tree branches at those points where new curves corresponding inflection points appears.

Scale Space Filtering III

- Coarse to fine analysis of the curves corresponding to inflection points, i.e., in the direction of the decreasing value of the σ, localizes events at different scales.
- The qualitative information contained in the scale-space image can be transformed into a simple `interval tree` that expresses the structure of the signal $f(x)$ over all (observed) scales.
- The interval tree is built from the root that corresponds to the largest scale.
- Then the scale-space image is searched in the direction of decreasing σ.
- The interval tree branches at those points where new curves corresponding inflection points appears.

Scale Space Filtering III

- Coarse to fine analysis of the curves corresponding to inflection points, i.e., in the direction of the decreasing value of the σ, localizes events at different scales.
- The qualitative information contained in the scale-space image can be transformed into a simple `interval tree` that expresses the structure of the signal $f(x)$ over all (observed) scales.
- The interval tree is built from the root that corresponds to the largest scale.
- Then the scale-space image is searched in the direction of decreasing σ.
- The interval tree branches at those points where new curves corresponding inflection points appears.

Scale Space Filtering III

- Coarse to fine analysis of the curves corresponding to inflection points, i.e., in the direction of the decreasing value of the σ, localizes events at different scales.
- The qualitative information contained in the scale-space image can be transformed into a simple `interval tree` that expresses the structure of the signal $f(x)$ over all (observed) scales.
- The interval tree is built from the root that corresponds to the largest scale.
- Then the scale-space image is searched in the direction of decreasing σ.
- The interval tree branches at those points where new curves corresponding inflection points appears.

Canny Edge Detector

- The third example of the application of scale — Canny edge detector, discussed in the next sub-section.

Outline

Canny Edge Detector

- Canny edge detector is optimal for step edges corrupted by white noise.
- The optimality of it is related to three criteria:
  - The detection criterion expresses that fact that important edges should not be missed, and that there should be no spurious responses.
  - The localization criterion requires that the distance between the actual and located position of the edge should be minimal.
  - The one response criterion minimizes multiple responses to a single edge (also partly covered by the first criterion, since when there are two responses to a single edge one of them should be considered as false).

Canny Edge Detector

- `Canny edge detector` is optimal for `step edges` corrupted by white noise.
- The optimality of it is related to three criteria:
  - The `detection criterion` expresses that fact that important edges should not be missed, and that there should be no spurious responses.
  - The `localization criterion` requires that the distance between the actual and located position of the edge should be minimal.
  - The `one response criterion` minimizes multiple responses to a single edge (also partly covered by the first criterion, since when there are two responses to a single edge one of them should be considered as false).

Canny Edge Detector

- Canny edge detector is optimal for step edges corrupted by white noise.
- The optimality of it is related to three criteria:
  - The detection criterion expresses that fact that important edges should not be missed, and that there should be no spurious responses.
  - The localization criterion requires that the distance between the actual and located position of the edge should be minimal.
  - The one response criterion minimizes multiple responses to a single edge (also partly covered by the first criterion, since when there are two responses to a single edge one of them should be considered as false).

Canny Edge Detector

- Canny edge detector is optimal for step edges corrupted by white noise.
- The optimality of it is related to three criteria:
  - The detection criterion expresses that fact that important edges should not be missed, and that there should be no spurious responses.
  - The localization criterion requires that the distance between the actual and located position of the edge should be minimal.
  - The one response criterion minimizes multiple responses to a single edge (also partly covered by the first criterion, since when there are two responses to a single edge one of them should be considered as false).

Canny Edge Detector

- `Canny edge detector` is optimal for `step edges` corrupted by white noise.
- The optimality of it is related to three criteria:
  - The `detection criterion` expresses that fact that important edges should not be missed, and that there should be no spurious responses.
  - The `localization criterion` requires that the distance between the actual and located position of the edge should be minimal.
  - The `one response criterion` minimizes multiple responses to a single edge (also partly covered by the first criterion, since when there are two responses to a single edge one of them should be considered as false).

Canny Edge Detector in 1D

- Canny's edge detector is based on several ideas:
  1. The edge detector was expressed for a 1D signal and the first two optimality criteria.
     - A closed form solution was found using the `calculus of variations`.
  2. If the third criterion (multiple responses) is added, the best solution may be found by numerical optimization.
     - The resulting filter can be approximated effectively by the first derivative of a Gaussian smoothing filter with standard deviation σ;
     - the reason for doing this is for an effective implementation.
     - There is a strong similarity here to the Marr-Hildreth edge detector (Laplacian of a Gaussian).

Canny Edge Detector in 1D

- Canny's edge detector is based on several ideas:
  1. The edge detector was expressed for a 1D signal and the first two optimality criteria.
     - A closed form solution was found using the `calculus of variations`.
  2. If the third criterion (multiple responses) is added, the best solution may be found by numerical optimization.
     - The resulting filter can be approximated effectively by the first derivative of a Gaussian smoothing filter with standard deviation σ;
     - the reason for doing this is for an effective implementation.
     - There is a strong similarity here to the Marr-Hildreth edge detector (Laplacian of a Gaussian).

Canny Edge Detector in 1D

- Canny's edge detector is based on several ideas:
  1. The edge detector was expressed for a 1D signal and the first two optimality criteria.
     - A closed form solution was found using the calculus of variations.
  2. If the third criterion (multiple responses) is added, the best solution may be found by numerical optimization.
     - The resulting filter can be approximated effectively by the first derivative of a Gaussian smoothing filter with standard deviation σ;
     - the reason for doing this is for an effective implementation.
     - There is a strong similarity here to the Marr-Hildreth edge detector (Laplacian of a Gaussian).

Canny Edge Detector in 1D

- Canny's edge detector is based on several ideas:
  1. The edge detector was expressed for a 1D signal and the first two optimality criteria.
     - A closed form solution was found using the `calculus of variations`.
  2. If the third criterion (multiple responses) is added, the best solution may be found by numerical optimization.
     - The resulting filter can be approximated effectively by the first derivative of a Gaussian smoothing filter with standard deviation σ;
     - the reason for doing this is for an effective implementation.
     - There is a strong similarity here to the Marr-Hildreth edge detector (Laplacian of a Gaussian).

Canny Edge Detector in 1D

- Canny's edge detector is based on several ideas:
  1. The edge detector was expressed for a 1D signal and the first two optimality criteria.
     - A closed form solution was found using the `calculus of variations`.
  2. If the third criterion (multiple responses) is added, the best solution may be found by numerical optimization.
     - The resulting filter can be approximated effectively by the first derivative of a Gaussian smoothing filter with standard deviation σ;
     - the reason for doing this is for an effective implementation.
     - There is a strong similarity here to the Marr-Hildreth edge detector (Laplacian of a Gaussian).

Canny Edge Detector in 1D

- Canny's edge detector is based on several ideas:
  1. The edge detector was expressed for a 1D signal and the first two optimality criteria.
     - A closed form solution was found using the `calculus of variations`.
  2. If the third criterion (multiple responses) is added, the best solution may be found by numerical optimization.
     - The resulting filter can be approximated effectively by the first derivative of a Gaussian smoothing filter with standard deviation σ;
     - the reason for doing this is for an effective implementation.
     - There is a strong similarity here to the Marr-Hildreth edge detector (Laplacian of a Gaussian).

Canny Edge Detector in 1D

- Canny's edge detector is based on several ideas:
  1. The edge detector was expressed for a 1D signal and the first two optimality criteria.
     - A closed form solution was found using the `calculus of variations`.
  2. If the third criterion (multiple responses) is added, the best solution may be found by numerical optimization.
     - The resulting filter can be approximated effectively by the first derivative of a Gaussian smoothing filter with standard deviation σ;
     - the reason for doing this is for an effective implementation.
     - There is a strong similarity here to the Marr-Hildreth edge detector (Laplacian of a Gaussian).

Canny Edge Detector in 2D I

- The detector is then generalized to two dimension.
- A step edge is given by its position, orientation, and possibly magnitude (strength).
- It can be shown that convolving an image with a symmetric 2D Gaussian and then differentiating in the direction of the gradient (perpendicular to the edge direction) forms a simple and effective directional operator.
- Recall that the Marr-Hildreth zero crossing operator does not give information about edge direction as it uses Laplacian filter.

Canny Edge Detector in 2D I

- The detector is then generalized to two dimension.
- A step edge is given by its position, orientation, and possibly magnitude (strength).
- It can be shown that convolving an image with a symmetric 2D Gaussian and then differentiating in the direction of the gradient (perpendicular to the edge direction) forms a simple and effective directional operator.
- Recall that the Marr-Hildreth zero crossing operator does not give information about edge direction as it uses Laplacian filter.

Canny Edge Detector in 2D I

- The detector is then generalized to two dimension.
- A step edge is given by its position, orientation, and possibly magnitude (strength).
- It can be shown that convolving an image with a symmetric 2D Gaussian and then differentiating in the direction of the gradient (perpendicular to the edge direction) forms a simple and effective directional operator.
- Recall that the Marr-Hildreth zero crossing operator does not give information about edge direction as it uses Laplacian filter.

Canny Edge Detector in 2D I

- The detector is then generalized to two dimension.
- A step edge is given by its position, orientation, and possibly magnitude (strength).
- It can be shown that convolving an image with a symmetric 2D Gaussian and then differentiating in the direction of the gradient (perpendicular to the edge direction) forms a simple and effective directional operator.
- Recall that the Marr-Hildreth zero crossing operator does not give information about edge direction as it uses Laplacian filter.

Canny Edge Detector in 2D II

- Suppose G is a 2D Gaussian (82) and assume we wish to convolute the image with an operator $G_{\mathbf{n}}$ which is the first derivative of G in the direction **n**.

$$G_{\mathbf{n}} = \frac{\partial G}{\partial \mathbf{n}} = \mathbf{n} \cdot \nabla G. \tag{89}$$

- The direction **n** should be oriented perpendicular to the edge
  - this direction is not known in advance
  - however, a robust estimate of it based on the smoothed gradient direction is available
  - if g is the image, the normal to the edge is estimated as

$$\mathbf{n} = \frac{\nabla(G * g)}{|\nabla(G * g)|}. \tag{90}$$

Canny Edge Detector in 2D II

- Suppose G is a 2D Gaussian (82) and assume we wish to convolute the image with an operator $G_{\mathbf{n}}$ which is the first derivative of G in the direction **n**.

$$G_{\mathbf{n}} = \frac{\partial G}{\partial \mathbf{n}} = \mathbf{n} \cdot \nabla G. \tag{89}$$

- The direction **n** should be oriented perpendicular to the edge
  - this direction is not known in advance
  - however, a robust estimate of it based on the smoothed gradient direction is available
  - if g is the image, the normal to the edge is estimated as

$$\mathbf{n} = \frac{\nabla(G * g)}{|\nabla(G * g)|}. \tag{90}$$

Canny Edge Detector in 2D II

- Suppose G is a 2D Gaussian (82) and assume we wish to convolute the image with an operator $G_{\mathbf{n}}$ which is the first derivative of G in the direction **n**.

$$G_{\mathbf{n}} = \frac{\partial G}{\partial \mathbf{n}} = \mathbf{n} \cdot \nabla G. \tag{89}$$

- The direction **n** should be oriented perpendicular to the edge
  - this direction is not known in advance
  - however, a robust estimate of it based on the smoothed gradient direction is available
  - if g is the image, the normal to the edge is estimated as

$$\mathbf{n} = \frac{\nabla(G * g)}{|\nabla(G * g)|}. \tag{90}$$

Canny Edge Detector in 2D II

- Suppose G is a 2D Gaussian (82) and assume we wish to convolute the image with an operator $G_{\mathbf{n}}$ which is the first derivative of G in the direction $\mathbf{n}$.

$$G_{\mathbf{n}} = \frac{\partial G}{\partial \mathbf{n}} = \mathbf{n} \cdot \nabla G. \tag{89}$$

- The direction $\mathbf{n}$ should be oriented perpendicular to the edge
  - this direction is not known in advance
  - however, a robust estimate of it based on the smoothed gradient direction is available
  - if g is the image, the normal to the edge is estimated as

$$\mathbf{n} = \frac{\nabla(G * g)}{|\nabla(G * g)|}. \tag{90}$$

Canny Edge Detector in 2D II

- Suppose G is a 2D Gaussian (82) and assume we wish to convolute the image with an operator $G_{\mathbf{n}}$ which is the first derivative of G in the direction **n**.

$$G_{\mathbf{n}} = \frac{\partial G}{\partial \mathbf{n}} = \mathbf{n} \cdot \nabla G. \tag{89}$$

- The direction **n** should be oriented perpendicular to the edge
  - this direction is not known in advance
  - however, a robust estimate of it based on the smoothed gradient direction is available
  - if g is the image, the normal to the edge is estimated as

$$\mathbf{n} = \frac{\nabla(G * g)}{|\nabla(G * g)|}. \tag{90}$$

Canny Edge Detector in 2D III

- The edge location is then at the local maximum in the direction $\mathbf{n}$ of the operator $G_{\mathbf{n}}$ convoluted with the image g

$$\frac{\partial}{\partial \mathbf{n}} G_{\mathbf{n}} * g = 0. \quad (91)$$

- Substituting in (91) for $G_{\mathbf{n}}$ from equation (89), we get

$$\frac{\partial^2}{\partial \mathbf{n}^2} G * g = 0. \quad (92)$$

- This equation (92) shows how to find local maxima in the direction perpendicular to the edge;
- this operation is often referred to as `non-maximal suppression`.

Canny Edge Detector in 2D III

- The edge location is then at the local maximum in the direction **n** of the operator $G_{\mathbf{n}}$ convoluted with the image g

$$\frac{\partial}{\partial \mathbf{n}} G_{\mathbf{n}} * g = 0. \tag{91}$$

- Substituting in (91) for $G_{\mathbf{n}}$ from equation (89), we get

$$\frac{\partial^2}{\partial \mathbf{n}^2} G * g = 0. \tag{92}$$

- This equation (92) shows how to find local maxima in the direction perpendicular to the edge;
- this operation is often referred to as `non-maximal suppression`.

Canny Edge Detector in 2D III

- The edge location is then at the local maximum in the direction **n** of the operator $G_{\mathbf{n}}$ convoluted with the image g

$$\frac{\partial}{\partial \mathbf{n}} G_{\mathbf{n}} * g = 0. \tag{91}$$

- Substituting in (91) for $G_{\mathbf{n}}$ from equation (89), we get

$$\frac{\partial^2}{\partial \mathbf{n}^2} G * g = 0. \tag{92}$$

- This equation (92) shows how to find local maxima in the direction perpendicular to the edge;
- this operation is often referred to as `non-maximal suppression`.

Canny Edge Detector in 2D III

- The edge location is then at the local maximum in the direction $\mathbf{n}$ of the operator $G_{\mathbf{n}}$ convoluted with the image g

$$\frac{\partial}{\partial \mathbf{n}} G_{\mathbf{n}} * g = 0. \tag{91}$$

- Substituting in (91) for $G_{\mathbf{n}}$ from equation (89), we get

$$\frac{\partial^2}{\partial \mathbf{n}^2} G * g = 0. \tag{92}$$

- This equation (92) shows how to find local maxima in the direction perpendicular to the edge;
- this operation is often referred to as `non-maximal suppression`.

Canny Edge Detector in 2D III

- As the convolution and derivative are associative operations in (92), we can
  - first convolute an image g with a symmetric Gaussian G;
  - then compute the directional second derivative using an estimate of the direction $\mathbf{n}$ computed according to equation (90).
- The strength of the edge (magnitude of the gradient of the image intensity function g) is measured as

$$|G_{\mathbf{n}} * g| = |\nabla(G * g)|. \tag{93}$$

Canny Edge Detector in 2D III

- As the convolution and derivative are associative operations in (92), we can
  - first convolute an image g with a symmetric Gaussian G;
  - then compute the directional second derivative using an estimate of the direction $\mathbf{n}$ computed according to equation (90).
- The strength of the edge (magnitude of the gradient of the image intensity function g) is measured as

$$|G_{\mathbf{n}} * g| = |\nabla(G * g)|. \qquad (93)$$

Canny Edge Detector in 2D III

- As the convolution and derivative are associative operations in (92), we can
  - first convolute an image g with a symmetric Gaussian G;
  - then compute the directional second derivative using an estimate of the direction $\mathbf{n}$ computed according to equation (90).
- The strength of the edge (magnitude of the gradient of the image intensity function g) is measured as

$$|G_{\mathbf{n}} * g| = |\nabla(G * g)|. \tag{93}$$

Canny Edge Detector in 2D III

- As the convolution and derivative are associative operations in (92), we can
  - first convolute an image g with a symmetric Gaussian G;
  - then compute the directional second derivative using an estimate of the direction $\mathbf{n}$ computed according to equation (90).
- The strength of the edge (magnitude of the gradient of the image intensity function g) is measured as

$$|G_{\mathbf{n}} * g| = |\nabla(G * g)|. \tag{93}$$

Canny Edge Detector: Noise and Spurious Response

- Spurious responses to the single edge caused by noise usually create a so called 'streaking' problem that is very common in edge detection in general.
  - Output of an edge detector is usually thresholded to decide which edges are significant.
  - Streaking means breaking up of the edge contour caused by the operator fluctuating above and below the threshold.

Canny Edge Detector: Noise and Spurious Response

- Spurious responses to the single edge caused by noise usually create a so called 'streaking' problem that is very common in edge detection in general.
  - Output of an edge detector is usually thresholded to decide which edges are significant.
  - Streaking means breaking up of the edge contour caused by the operator fluctuating above and below the threshold.

Canny Edge Detector: Noise and Spurious Response

- Spurious responses to the single edge caused by noise usually create a so called 'streaking' problem that is very common in edge detection in general.
  - Output of an edge detector is usually thresholded to decide which edges are significant.
  - Streaking means breaking up of the edge contour caused by the operator fluctuating above and below the threshold.

Canny Edge Detector: Hysteresis Thresholding

- Streaking can be eliminated by `thresholding with hysteresis`.
  - If any edge response is above a high threshold, those pixels constitute definite output of the edge detector for a particular scale.
  - Individual weak responses usually correspond to noise, but if these points are connected to any of the pixels with strong responses they are more likely to be actual edges in the image.
  - Such connected pixels are treated as edge pixels if their response is above a low threshold.
  - The low and high thresholds are set according to an estimated signal to noise ratio.
  - Please refer to Canny's original paper for detailed discussions.

Canny Edge Detector: Hysteresis Thresholding

- Streaking can be eliminated by `thresholding with hysteresis`.
  - If any edge response is above a high threshold, those pixels constitute definite output of the edge detector for a particular scale.
  - Individual weak responses usually correspond to noise, but if these points are connected to any of the pixels with strong responses they are more likely to be actual edges in the image.
  - Such connected pixels are treated as edge pixels if their response is above a low threshold.
  - The low and high thresholds are set according to an estimated signal to noise ratio.
  - Please refer to Canny's original paper for detailed discussions.

Canny Edge Detector: Hysteresis Thresholding

- Streaking can be eliminated by `thresholding with hysteresis`.
  - If any edge response is above a high threshold, those pixels constitute definite output of the edge detector for a particular scale.
  - Individual weak responses usually correspond to noise, but if these points are connected to any of the pixels with strong responses they are more likely to be actual edges in the image.
  - Such connected pixels are treated as edge pixels if their response is above a low threshold.
  - The low and high thresholds are set according to an estimated signal to noise ratio.
  - Please refer to Canny's original paper for detailed discussions.

Canny Edge Detector: Hysteresis Thresholding

- Streaking can be eliminated by `thresholding with hysteresis`.
  - If any edge response is above a high threshold, those pixels constitute definite output of the edge detector for a particular scale.
  - Individual weak responses usually correspond to noise, but if these points are connected to any of the pixels with strong responses they are more likely to be actual edges in the image.
  - Such connected pixels are treated as edge pixels if their response is above a low threshold.
  - The low and high thresholds are set according to an estimated signal to noise ratio.
  - Please refer to Canny's original paper for detailed discussions.

Canny Edge Detector: Hysteresis Thresholding

- Streaking can be eliminated by `thresholding with hysteresis`.
  - If any edge response is above a high threshold, those pixels constitute definite output of the edge detector for a particular scale.
  - Individual weak responses usually correspond to noise, but if these points are connected to any of the pixels with strong responses they are more likely to be actual edges in the image.
  - Such connected pixels are treated as edge pixels if their response is above a low threshold.
  - The low and high thresholds are set according to an estimated signal to noise ratio.
  - Please refer to Canny's original paper for detailed discussions.

Canny Edge Detector: Hysteresis Thresholding

- Streaking can be eliminated by `thresholding with hysteresis`.
  - If any edge response is above a high threshold, those pixels constitute definite output of the edge detector for a particular scale.
  - Individual weak responses usually correspond to noise, but if these points are connected to any of the pixels with strong responses they are more likely to be actual edges in the image.
  - Such connected pixels are treated as edge pixels if their response is above a low threshold.
  - The low and high thresholds are set according to an estimated signal to noise ratio.
  - Please refer to Canny's original paper for detailed discussions.

Example: Hysteresis Thresholding

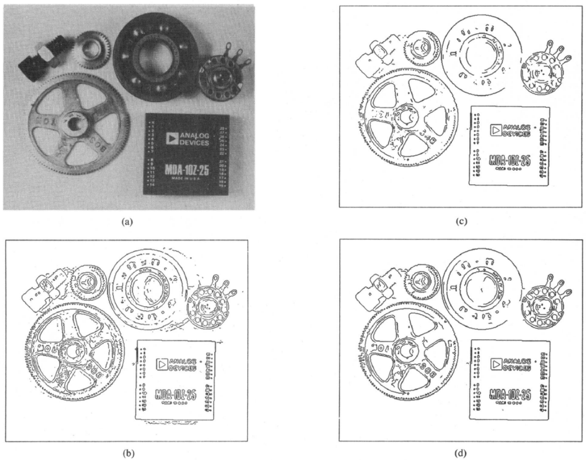


Figure: (a) Parts image, 576 by 454 pixels. (b) Image thesholded at T_1. (c) Image thresholded at $2T_1$. (d) Image thresholded with hysteresis using both the thresholds in (b) and (c). [Canny, 1986].

Canny Edge Detector with Scales

- The correct scale for the operator depends on the objects contained in the image.
  - The solution to this unknown is to use multiple scales and aggregate information from them.
  - Different scale for the Canny detector is represented by different standard deviations σ of the Gaussians.
  - There may be several scales of operators that give significant responses to edges (i.e., signal to noise ratio above the threshold);
  - in this case the operator with the smallest scale is chosen as it gives the best localization of the edge.

Canny Edge Detector with Scales

- The correct scale for the operator depends on the objects contained in the image.
  - The solution to this unknown is to use multiple scales and aggregate information from them.
  - Different scale for the Canny detector is represented by different standard deviations σ of the Gaussians.
  - There may be several scales of operators that give significant responses to edges (i.e., signal to noise ratio above the threshold);
  - in this case the operator with the smallest scale is chosen as it gives the best localization of the edge.

Canny Edge Detector with Scales

- The correct scale for the operator depends on the objects contained in the image.
  - The solution to this unknown is to use multiple scales and aggregate information from them.
  - Different scale for the Canny detector is represented by different standard deviations σ of the Gaussians.
  - There may be several scales of operators that give significant responses to edges (i.e., signal to noise ratio above the threshold);
  - in this case the operator with the smallest scale is chosen as it gives the best localization of the edge.

Canny Edge Detector with Scales

- The correct scale for the operator depends on the objects contained in the image.
  - The solution to this unknown is to use multiple scales and aggregate information from them.
  - Different scale for the Canny detector is represented by different standard deviations σ of the Gaussians.
  - There may be several scales of operators that give significant responses to edges (i.e., signal to noise ratio above the threshold);
  - in this case the operator with the smallest scale is chosen as it gives the best localization of the edge.

Canny Edge Detector with Scales

- The correct scale for the operator depends on the objects contained in the image.
  - The solution to this unknown is to use multiple scales and aggregate information from them.
  - Different scale for the Canny detector is represented by different standard deviations σ of the Gaussians.
  - There may be several scales of operators that give significant responses to edges (i.e., signal to noise ratio above the threshold);
  - in this case the operator with the smallest scale is chosen as it gives the best localization of the edge.

Canny Edge Detector: Feature Synthesis

- Canny proposed a `Feature synthesis` approach.
- All significant edges from the operator with the smallest scale are marked first.
- Edges of a hypothetical operator with larger σ are synthesized from them (i.e., a prediction is made of how the larger σ should perform on the evidence gleaned from the smaller σ).
- Then the synthesized edge response is compared with the actual edge response for larger σ.
- Additional edges are marked only if they have significantly stronger response than that predicted from synthetic output.
- This procedure may be repeated for a sequence of scales, a cumulative edge map is built by adding those edges that were not identified at smaller scales.

Canny Edge Detector: Feature Synthesis

- Canny proposed a `Feature synthesis` approach.
- All significant edges from the operator with the smallest scale are marked first.
- Edges of a hypothetical operator with larger σ are synthesized from them (i.e., a prediction is made of how the larger σ should perform on the evidence gleaned from the smaller σ).
- Then the synthesized edge response is compared with the actual edge response for larger σ.
- Additional edges are marked only if they have significantly stronger response than that predicted from synthetic output.
- This procedure may be repeated for a sequence of scales, a cumulative edge map is built by adding those edges that were not identified at smaller scales.

Canny Edge Detector: Feature Synthesis

- Canny proposed a `Feature synthesis` approach.
- All significant edges from the operator with the smallest scale are marked first.
- Edges of a hypothetical operator with larger σ are synthesized from them (i.e., a prediction is made of how the larger σ should perform on the evidence gleaned from the smaller σ).
- Then the synthesized edge response is compared with the actual edge response for larger σ.
- Additional edges are marked only if they have significantly stronger response than that predicted from synthetic output.
- This procedure may be repeated for a sequence of scales, a cumulative edge map is built by adding those edges that were not identified at smaller scales.

Canny Edge Detector: Feature Synthesis

- Canny proposed a `Feature synthesis` approach.
- All significant edges from the operator with the smallest scale are marked first.
- Edges of a hypothetical operator with larger σ are synthesized from them (i.e., a prediction is made of how the larger σ should perform on the evidence gleaned from the smaller σ).
- Then the synthesized edge response is compared with the actual edge response for larger σ.
- Additional edges are marked only if they have significantly stronger response than that predicted from synthetic output.
- This procedure may be repeated for a sequence of scales, a cumulative edge map is built by adding those edges that were not identified at smaller scales.

Canny Edge Detector: Feature Synthesis

- Canny proposed a `Feature synthesis` approach.
- All significant edges from the operator with the smallest scale are marked first.
- Edges of a hypothetical operator with larger σ are synthesized from them (i.e., a prediction is made of how the larger σ should perform on the evidence gleaned from the smaller σ).
- Then the synthesized edge response is compared with the actual edge response for larger σ.
- Additional edges are marked only if they have significantly stronger response than that predicted from synthetic output.
- This procedure may be repeated for a sequence of scales, a cumulative edge map is built by adding those edges that were not identified at smaller scales.

Canny Edge Detector: Feature Synthesis

- Canny proposed a `Feature synthesis` approach.
- All significant edges from the operator with the smallest scale are marked first.
- Edges of a hypothetical operator with larger σ are synthesized from them (i.e., a prediction is made of how the larger σ should perform on the evidence gleaned from the smaller σ).
- Then the synthesized edge response is compared with the actual edge response for larger σ.
- Additional edges are marked only if they have significantly stronger response than that predicted from synthetic output.
- This procedure may be repeated for a sequence of scales, a cumulative edge map is built by adding those edges that were not identified at smaller scales.

Example: Feature Synthesis

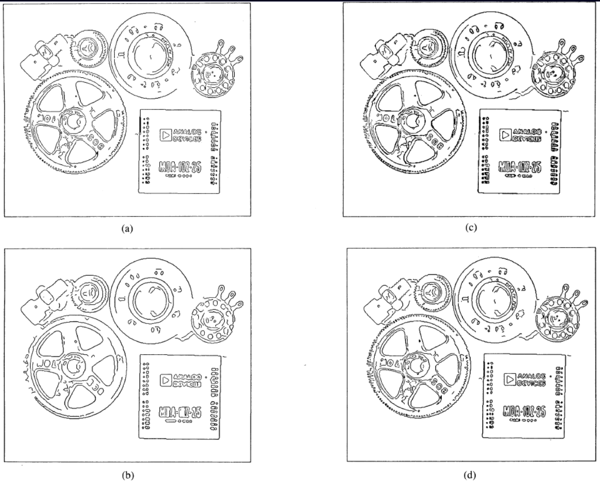


Figure: (a) Edges from parts image at $\sigma = 1.0$. (b) Edges at $\sigma = 2.0$. (c) Superposition of the edges. (d) Edges combined using feature synthesis. [Canny, 1986]

Canny Edge Detector

1. Repeat steps (2) till (6) for ascending values of the standard deviation.
2. Convolve an image g with a Gaussian of scale σ.
3. Estimate local edge normal directions **n** using equation (90) for each pixel in the image.
4. Find the location of the edges using equation (92) (non-maximal suppression).
5. Compute the magnitude of the edge using equation (93).
6. Threshold edges in the image with hysteresis to eliminate spurious responses.
7. Aggregate the final information about edges at multiple scale using the “feature synthesis” approach.

Canny Edge Detector

1. Repeat steps (2) till (6) for ascending values of the standard deviation.
2. Convolve an image g with a Gaussian of scale σ.
3. Estimate local edge normal directions $\mathbf{n}$ using equation (90) for each pixel in the image.
4. Find the location of the edges using equation (92) (non-maximal suppression).
5. Compute the magnitude of the edge using equation (93).
6. Threshold edges in the image with hysteresis to eliminate spurious responses.
7. Aggregate the final information about edges at multiple scale using the “feature synthesis” approach.

Canny Edge Detector

1. Repeat steps (2) till (6) for ascending values of the standard deviation.
2. Convolve an image g with a Gaussian of scale σ.
3. Estimate local edge normal directions **n** using equation (90) for each pixel in the image.
4. Find the location of the edges using equation (92) (non-maximal suppression).
5. Compute the magnitude of the edge using equation (93).
6. Threshold edges in the image with hysteresis to eliminate spurious responses.
7. Aggregate the final information about edges at multiple scale using the “feature synthesis” approach.

Canny Edge Detector

1. Repeat steps (2) till (6) for ascending values of the standard deviation.
2. Convolve an image g with a Gaussian of scale σ.
3. Estimate local edge normal directions **n** using equation (90) for each pixel in the image.
4. Find the location of the edges using equation (92) (non-maximal suppression).
5. Compute the magnitude of the edge using equation (93).
6. Threshold edges in the image with hysteresis to eliminate spurious responses.
7. Aggregate the final information about edges at multiple scale using the "feature synthesis" approach.

Canny Edge Detector

1. Repeat steps (2) till (6) for ascending values of the standard deviation.
2. Convolve an image g with a Gaussian of scale σ.
3. Estimate local edge normal directions $\mathbf{n}$ using equation (90) for each pixel in the image.
4. Find the location of the edges using equation (92) (non-maximal suppression).
5. Compute the magnitude of the edge using equation (93).
6. Threshold edges in the image with hysteresis to eliminate spurious responses.
7. Aggregate the final information about edges at multiple scale using the “feature synthesis” approach.

Canny Edge Detector

1. Repeat steps (2) till (6) for ascending values of the standard deviation.
2. Convolve an image g with a Gaussian of scale σ.
3. Estimate local edge normal directions **n** using equation (90) for each pixel in the image.
4. Find the location of the edges using equation (92) (non-maximal suppression).
5. Compute the magnitude of the edge using equation (93).
6. Threshold edges in the image with hysteresis to eliminate spurious responses.
7. Aggregate the final information about edges at multiple scale using the “feature synthesis” approach.

Canny Edge Detector

1. Repeat steps (2) till (6) for ascending values of the standard deviation.
2. Convolve an image g with a Gaussian of scale σ.
3. Estimate local edge normal directions $\mathbf{n}$ using equation (90) for each pixel in the image.
4. Find the location of the edges using equation (92) (non-maximal suppression).
5. Compute the magnitude of the edge using equation (93).
6. Threshold edges in the image with hysteresis to eliminate spurious responses.
7. Aggregate the final information about edges at multiple scale using the “feature synthesis” approach.

Discussions

- Canny's detector represents a complicated but major contribution to edge detection.
- Its full implementation is unusual, it being common to find implementations that omit feature synthesis — that is, just steps 2 — 6 of algorithm.
- Reference for this section is Canny's paper [Canny, 1986].

Discussions

- Canny's detector represents a complicated but major contribution to edge detection.
- Its full implementation is unusual, it being common to find implementations that omit feature synthesis — that is, just steps 2 — 6 of algorithm.
- Reference for this section is Canny's paper [Canny, 1986].

Discussions

- Canny's detector represents a complicated but major contribution to edge detection.
- Its full implementation is unusual, it being common to find implementations that omit feature synthesis — that is, just steps 2 — 6 of algorithm.
- Reference for this section is Canny's paper [Canny, 1986].

Example

- The matlab script file is ../program/canny.m.
- Compare the result with the edge detector with other filters.
- Note the improvement at the top part of the hat.
- Study the matlab code. Note the “thin” code.

Example

- The matlab script file is ../program/canny.m.
- Compare the result with the edge detector with other filters.
- Note the improvement at the top part of the hat.
- Study the matlab code. Note the “thin” code.

Example

- The matlab script file is ../program/canny.m.
- Compare the result with the edge detector with other filters.
- Note the improvement at the top part of the hat.
- Study the matlab code. Note the “thin” code.

Example

- The matlab script file is ../program/canny.m.
- Compare the result with the edge detector with other filters.
- Note the improvement at the top part of the hat.
- Study the matlab code. Note the "thin" code.

Homework

- This is an important homework. The total score is 20. The homework includes:
  1. (20 points) Implement median filter with variable window size and shape;
  2. (20 points) Implement another local smoothing filter with possible variable parameters;
  3. (40) Implement Canny edge detector; what is your extension to step 7? (Without step 7, you can only get at most 30 points.).
  4. (30) Implement another edge detector and compare it with Canny detector.

Homework

- This is an important homework. The total score is 20. The homework includes:
  1. (20 points) Implement median filter with variable window size and shape;
  2. (20 points) Implement another local smoothing filter with possible variable parameters;
  3. (40) Implement Canny edge detector; what is your extension to step 7? (Without step 7, you can only get at most 30 points.).
  4. (30) Implement another edge detector and compare it with Canny detector.

Homework

- This is an important homework. The total score is 20. The homework includes:
  1. (20 points) Implement median filter with variable window size and shape;
  2. (20 points) Implement another local smoothing filter with possible variable parameters;
  3. (40) Implement Canny edge detector; what is your extension to step 7? (Without step 7, you can only get at most 30 points.).
  4. (30) Implement another edge detector and compare it with Canny detector.

Homework

- This is an important homework. The total score is 20. The homework includes:
  1. (20 points) Implement median filter with variable window size and shape;
  2. (20 points) Implement another local smoothing filter with possible variable parameters;
  3. (40) Implement Canny edge detector; what is your extension to step 7? (Without step 7, you can only get at most 30 points.).
  4. (30) Implement another edge detector and compare it with Canny detector.

Homework

- This is an important homework. The total score is 20. The homework includes:
  1. (20 points) Implement median filter with variable window size and shape;
  2. (20 points) Implement another local smoothing filter with possible variable parameters;
  3. (40) Implement Canny edge detector; what is your extension to step 7? (Without step 7, you can only get at most 30 points.).
  4. (30) Implement another edge detector and compare it with Canny detector.

Outline

Parametric Edge Models

- Parametric models are based on the idea that the discrete image intensity function can be considered a sampled and noisy approximation of the underlying continuous or piecewise continuous image intensity function.
- While the continuous image function is not known, it can be estimated from the available discrete image intensity function and image properties can be determined from this estimate, possibly with sub-pixel precision.
- Piecewise continuous function estimate called `facets` are used to represent (a neighborhood) image pixel.
- Such image representation is called `facet model`.

Parametric Edge Models

- Parametric models are based on the idea that the discrete image intensity function can be considered a sampled and noisy approximation of the underlying continuous or piecewise continuous image intensity function.
- While the continuous image function is not known, it can be estimated from the available discrete image intensity function and image properties can be determined from this estimate, possibly with sub-pixel precision.
- Piecewise continuous function estimate called `facets` are used to represent (a neighborhood) image pixel.
- Such image representation is called `facet model`.

Parametric Edge Models

- Parametric models are based on the idea that the discrete image intensity function can be considered a sampled and noisy approximation of the underlying continuous or piecewise continuous image intensity function.
- While the continuous image function is not known, it can be estimated from the available discrete image intensity function and image properties can be determined from this estimate, possibly with sub-pixel precision.
- Piecewise continuous function estimate called `facets` are used to represent (a neighborhood) image pixel.
- Such image representation is called `facet model`.

Parametric Edge Models

- Parametric models are based on the idea that the discrete image intensity function can be considered a sampled and noisy approximation of the underlying continuous or piecewise continuous image intensity function.
- While the continuous image function is not known, it can be estimated from the available discrete image intensity function and image properties can be determined from this estimate, possibly with sub-pixel precision.
- Piecewise continuous function estimate called `facets` are used to represent (a neighborhood) image pixel.
- Such image representation is called `facet model`.

Parametric Edge Models

- The intensity function in a pixel neighborhood can be estimated using models of different complexity.
- The simplest one is the `flat facet model` that uses piecewise constants and each pixel neighborhood is represented by a flat function of constant intensity.
- The `sloped model` uses piecewise linear functions forming a sloped plane fitted to the image intensities in the pixel neighborhood.
- `Quadratic` and `bi-cubic facet models` employ correspondingly more complex functions.
- A thorough treatment of facet models and their applications is given in [Haralic and Shapiro, 1992].

Parametric Edge Models

- The intensity function in a pixel neighborhood can be estimated using models of different complexity.
- The simplest one is the `flat facet model` that uses piecewise constants and each pixel neighborhood is represented by a flat function of constant intensity.
- The `sloped model` uses piecewise linear functions forming a sloped plane fitted to the image intensities in the pixel neighborhood.
- `Quadratic` and `bi-cubic facet models` employ correspondingly more complex functions.
- A thorough treatment of facet models and their applications is given in [Haralic and Shapiro, 1992].

Parametric Edge Models

- The intensity function in a pixel neighborhood can be estimated using models of different complexity.
- The simplest one is the `flat facet model` that uses piecewise constants and each pixel neighborhood is represented by a flat function of constant intensity.
- The `sloped model` uses piecewise linear functions forming a sloped plane fitted to the image intensities in the pixel neighborhood.
- `Quadratic` and `bi-cubic facet models` employ correspondingly more complex functions.
- A thorough treatment of facet models and their applications is given in [Haralic and Shapiro, 1992].

Parametric Edge Models

- The intensity function in a pixel neighborhood can be estimated using models of different complexity.
- The simplest one is the `flat facet model` that uses piecewise constants and each pixel neighborhood is represented by a flat function of constant intensity.
- The `sloped model` uses piecewise linear functions forming a sloped plane fitted to the image intensities in the pixel neighborhood.
- `Quadratic` and `bi-cubic facet models` employ correspondingly more complex functions.
- A thorough treatment of facet models and their applications is given in [Haralic and Shapiro, 1992].

Parametric Edge Models

- The intensity function in a pixel neighborhood can be estimated using models of different complexity.
- The simplest one is the `flat facet model` that uses piecewise constants and each pixel neighborhood is represented by a flat function of constant intensity.
- The `sloped model` uses piecewise linear functions forming a sloped plane fitted to the image intensities in the pixel neighborhood.
- `Quadratic` and `bi-cubic facet models` employ correspondingly more complex functions.
- A thorough treatment of facet models and their applications is given in [Haralic and Shapiro, 1992].

Bi-cubic Facet Model I

- Consider a bi-cubic facet model

$$g(x,y) = c_1 + c_2 x + c_3 y + c_4 x^2 + c_5 xy + c_6 y^2 + c_7 x^3 + c_8 x^2 y + c_9 xy^2 + c_{10} y^3. \quad (94)$$

- The parameters are estimated from a pixel neighborhood(the central pixel is at $(0,0)$).
- A least-squares method with singular-value decomposition may be used.

Bi-cubic Facet Model I

- Consider a bi-cubic facet model

$$g(x,y) = c_1 + c_2 x + c_3 y + c_4 x^2 + c_5 xy + c_6 y^2 \\ + c_7 x^3 + c_8 x^2 y + c_9 xy^2 + c_{10} y^3. \quad (94)$$

- The parameters are estimated from a pixel neighborhood(the central pixel is at $(0,0)$).
- A least-squares method with singular-value decomposition may be used.

Bi-cubic Facet Model I

- Consider a bi-cubic facet model

$$g(x, y) = c_1 + c_2 x + c_3 y + c_4 x^2 + c_5 xy + c_6 y^2 + c_7 x^3 + c_8 x^2 y + c_9 xy^2 + c_{10} y^3. \quad (94)$$

- The parameters are estimated from a pixel neighborhood(the central pixel is at $(0, 0)$).
- A least-squares method with singular-value decomposition may be used.

Bi-cubic Facet Model II

- Once the facet model parameters are available for each image pixel, edges can be detected as extrema of the first directional derivative and/or zero-crossings of the the second directional derivative of the local continuous facet model functions.
- Edge detectors based on parametric models describe edges more precisely than convolution based edge detectors.
- Additionally, they carry the potential for sub-pixel edge localization.
- However, their computational requirements are much higher.

Bi-cubic Facet Model II

- Once the facet model parameters are available for each image pixel, edges can be detected as extrema of the first directional derivative and/or zero-crossings of the the second directional derivative of the local continuous facet model functions.
- Edge detectors based on parametric models describe edges more precisely than convolution based edge detectors.
- Additionally, they carry the potential for sub-pixel edge localization.
- However, their computational requirements are much higher.

Bi-cubic Facet Model II

- Once the facet model parameters are available for each image pixel, edges can be detected as extrema of the first directional derivative and/or zero-crossings of the the second directional derivative of the local continuous facet model functions.
- Edge detectors based on parametric models describe edges more precisely than convolution based edge detectors.
- Additionally, they carry the potential for sub-pixel edge localization.
- However, their computational requirements are much higher.

Bi-cubic Facet Model II

- Once the facet model parameters are available for each image pixel, edges can be detected as extrema of the first directional derivative and/or zero-crossings of the the second directional derivative of the local continuous facet model functions.
- Edge detectors based on parametric models describe edges more precisely than convolution based edge detectors.
- Additionally, they carry the potential for sub-pixel edge localization.
- However, their computational requirements are much higher.

Outline

- Pixel Brightness Transformations
  - Position dependent brightness correction
  - Grey scale transformation
    - Windows and level
    - Histogram stretching
    - Histogram Equalization
    - Histogram Matching
- Geometric Transformations
  - Pixel Co-ordinate Transformations
    - Polynomial Approximation
    - Bilinear Transformations
    - Important Transformations
  - Brightness Interpolation
    - Nearest Neighbor Interpolation
    - Bilinear Interpolation
    - Bi-cubic interpolation
- Local pre-processing
  - Image smoothing
    - Averaging
    - Averaging with Data Validity
    - Averaging According to Inverse Gradient

Edge Detection in Multi-spectral Images

- The first is to detect edges separately in individual image spectral components using the ordinary edge detectors.
  - Individual images of edges can be combined to get the resulting edge image;
  - The value corresponding to edge magnitude and direction is the maximal edge value from all spectral components.
  - A linear combination of edge spectral components can also be used, and other combination techniques are possible.
- A second possibility is to use the brightness difference of the same pixel in two different spectral components.
  - The ratio instead of the difference can also be used as well, although it is necessary to assume that pixel values are not zero in this case.

Edge Detection in Multi-spectral Images

- The first is to detect edges separately in individual image spectral components using the ordinary edge detectors.
  - Individual images of edges can be combined to get the resulting edge image;
  - The value corresponding to edge magnitude and direction is the maximal edge value from all spectral components.
  - A linear combination of edge spectral components can also be used, and other combination techniques are possible.
- A second possibility is to use the brightness difference of the same pixel in two different spectral components.
  - The ratio instead of the difference can also be used as well, although it is necessary to assume that pixel values are not zero in this case.

Edge Detection in Multi-spectral Images

- The first is to detect edges separately in individual image spectral components using the ordinary edge detectors.
  - Individual images of edges can be combined to get the resulting edge image;
  - The value corresponding to edge magnitude and direction is the maximal edge value from all spectral components.
  - A linear combination of edge spectral components can also be used, and other combination techniques are possible.
- A second possibility is to use the brightness difference of the same pixel in two different spectral components.
  - The ratio instead of the difference can also be used as well, although it is necessary to assume that pixel values are not zero in this case.

Edge Detection in Multi-spectral Images

- The first is to detect edges separately in individual image spectral components using the ordinary edge detectors.
  - Individual images of edges can be combined to get the resulting edge image;
  - The value corresponding to edge magnitude and direction is the maximal edge value from all spectral components.
  - A linear combination of edge spectral components can also be used, and other combination techniques are possible.
- A second possibility is to use the brightness difference of the same pixel in two different spectral components.
  - The ratio instead of the difference can also be used as well, although it is necessary to assume that pixel values are not zero in this case.

Edge Detection in Multi-spectral Images

- The first is to detect edges separately in individual image spectral components using the ordinary edge detectors.
  - Individual images of edges can be combined to get the resulting edge image;
  - The value corresponding to edge magnitude and direction is the maximal edge value from all spectral components.
  - A linear combination of edge spectral components can also be used, and other combination techniques are possible.
- A second possibility is to use the brightness difference of the same pixel in two different spectral components.
  - The ratio instead of the difference can also be used as well, although it is necessary to assume that pixel values are not zero in this case.

Edge Detection in Multi-spectral Images

- The first is to detect edges separately in individual image spectral components using the ordinary edge detectors.
  - Individual images of edges can be combined to get the resulting edge image;
  - The value corresponding to edge magnitude and direction is the maximal edge value from all spectral components.
  - A linear combination of edge spectral components can also be used, and other combination techniques are possible.
- A second possibility is to use the brightness difference of the same pixel in two different spectral components.
  - The ratio instead of the difference can also be used as well, although it is necessary to assume that pixel values are not zero in this case.

Multi-spectral Edge Detector

- A third possibility is to create a multi-spectral edge detector which uses brightness information from all n spectral bands.
  - An edge detector of this kind was proposed in [Cervenka and Charvat, 1987].
  - The neighborhood used has size $2 \times 2 \times n$ pixels, where the 2×2 neighborhood is similar to that of the Roberts gradient, (72).
  - The coefficients weighting the influence of the component pixels are similar to the correlation correlation coefficients.
  - Let $\bar{f}(i,j)$ denote the arithmetic mean of the brightness corresponding to the pixel with the same co-ordinates (i,j) in all n spectral component images and f_r be the brightness of the r^{th} spectral component.

Multi-spectral Edge Detector

- A third possibility is to create a multi-spectral edge detector which uses brightness information from all n spectral bands.
  - An edge detector of this kind was proposed in [Cervenka and Charvat, 1987].
  - The neighborhood used has size $2 \times 2 \times n$ pixels, where the 2×2 neighborhood is similar to that of the Roberts gradient, (72).
  - The coefficients weighting the influence of the component pixels are similar to the correlation correlation coefficients.
  - Let $\bar{f}(i,j)$ denote the arithmetic mean of the brightness corresponding to the pixel with the same co-ordinates (i,j) in all n spectral component images and f_r be the brightness of the r^{th} spectral component.

Multi-spectral Edge Detector

- A third possibility is to create a multi-spectral edge detector which uses brightness information from all n spectral bands.
  - An edge detector of this kind was proposed in [Cervenka and Charvat, 1987].
  - The neighborhood used has size $2 \times 2 \times n$ pixels, where the 2×2 neighborhood is similar to that of the Roberts gradient, (72).
  - The coefficients weighting the influence of the component pixels are similar to the correlation correlation coefficients.
  - Let $\bar{f}(i,j)$ denote the arithmetic mean of the brightness corresponding to the pixel with the same co-ordinates (i,j) in all n spectral component images and f_r be the brightness of the r^{th} spectral component.

Multi-spectral Edge Detector

- A third possibility is to create a multi-spectral edge detector which uses brightness information from all n spectral bands.
  - An edge detector of this kind was proposed in [Cervenka and Charvat, 1987].
  - The neighborhood used has size $2 \times 2 \times n$ pixels, where the 2×2 neighborhood is similar to that of the Roberts gradient, (72).
  - The coefficients weighting the influence of the component pixels are similar to the correlation correlation coefficients.
  - Let $\bar{f}(i,j)$ denote the arithmetic mean of the brightness corresponding to the pixel with the same co-ordinates (i,j) in all n spectral component images and f_r be the brightness of the r^{th} spectral component.

Multi-spectral Edge Detector

- A third possibility is to create a multi-spectral edge detector which uses brightness information from all n spectral bands.
  - An edge detector of this kind was proposed in [Cervenka and Charvat, 1987].
  - The neighborhood used has size $2 \times 2 \times n$ pixels, where the 2×2 neighborhood is similar to that of the Roberts gradient, (72).
  - The coefficients weighting the influence of the component pixels are similar to the correlation correlation coefficients.
  - Let $\bar{f}(i,j)$ denote the arithmetic mean of the brightness corresponding to the pixel with the same co-ordinates (i,j) in all n spectral component images and f_r be the brightness of the r^{th} spectral component.

Multi-spectral Edge Detector

- The edge detector result in pixel (i, j) is given as the minimum of the following expressions:

$$\frac{\sum_{r=1}^{n} d_r(i,j) d_r(i+1,j+1)}{\sqrt{\sum_{r=1}^{n} (d_r(i,j))^2 \sum_{r=1}^{n} (d_r(i+1,j+1))^2}} \times \frac{\sum_{r=1}^{n} d_r(i+1,j) d_r(i,j+1)}{\sqrt{\sum_{r=1}^{n} (d_r(i+1,j))^2 \sum_{r=1}^{n} (d_r(i,j+1))^2}}$$

where $d_r(i,j) = f_r(i,j) - \bar{f}(i,j)$.

- This multi-spectral edge detector gives very good (?) results on remotely sensed images.

Multi-spectral Edge Detector

- The edge detector result in pixel (i, j) is given as the minimum of the following expressions:

$$\frac{\sum_{r=1}^{n} d_r(i,j) d_r(i+1,j+1)}{\sqrt{\sum_{r=1}^{n} (d_r(i,j))^2 \sum_{r=1}^{n} (d_r(i+1,j+1))^2}} \times \frac{\sum_{r=1}^{n} d_r(i+1,j) d_r(i,j+1)}{\sqrt{\sum_{r=1}^{n} (d_r(i+1,j))^2 \sum_{r=1}^{n} (d_r(i,j+1))^2}}$$

where $d_r(i,j) = f_r(i,j) - \bar{f}(i,j)$.

- This multi-spectral edge detector gives very good (?) results on remotely sensed images.

Outline

Other Local Operators

- Several other local operations exists which do not belong to the taxonomy in § 3, as they are used for different purposes.
- `Line finding`, `line thinning`, and `line filling` operators are among them.
- The second group of operators finds `interest points` or `locations of interest` in the image.
- There is yet another class of local nonlinear operators, `mathematical morphology techniques`.

Other Local Operators

- Several other local operations exists which do not belong to the taxonomy in § 3, as they are used for different purposes.
- `Line finding`, `line thinning`, and `line filling` operators are among them.
- The second group of operators finds `interest points` or `locations of interest` in the image.
- There is yet another class of local nonlinear operators, `mathematical morphology techniques`.

Other Local Operators

- Several other local operations exists which do not belong to the taxonomy in § 3, as they are used for different purposes.
- `Line finding`, `line thinning`, and `line filling` operators are among them.
- The second group of operators finds `interest points` or `locations of interest` in the image.
- There is yet another class of local nonlinear operators, `mathematical morphology techniques`.

Other Local Operators

- Several other local operations exists which do not belong to the taxonomy in § 3, as they are used for different purposes.
- `Line finding`, `line thinning`, and `line filling` operators are among them.
- The second group of operators finds `interest points` or `locations of interest` in the image.
- There is yet another class of local nonlinear operators, `mathematical morphology techniques`.

Why?

- Recall that one of the reasons why edges are being detected is that they bear a lot of information about underlying objects in the scene.
- Taking just edge elements instead of all pixels reduces the amount of data which has to be processed.
- The edge detector is a general tool which does not depend on the content of the particular image.
- The detected edges are to some degree robust as they do not depend much on small changes in illumination, viewpoint change, etc.

- It is interesting to seek yet richer features which can be reliably detected in the image and which can outperform simple edge detectors in some classes of applications.
- Line detectors and corner detectors are some such.

Why?

- Recall that one of the reasons why edges are being detected is that they bear a lot of information about underlying objects in the scene.
- Taking just edge elements instead of all pixels reduces the amount of data which has to be processed.
- The edge detector is a general tool which does not depend on the content of the particular image.
- The detected edges are to some degree robust as they do not depend much on small changes in illumination, viewpoint change, etc.

- It is interesting to seek yet richer features which can be reliably detected in the image and which can outperform simple edge detectors in some classes of applications.
- Line detectors and corner detectors are some such.

Why?

- Recall that one of the reasons why edges are being detected is that they bear a lot of information about underlying objects in the scene.
- Taking just edge elements instead of all pixels reduces the amount of data which has to be processed.
- The edge detector is a general tool which does not depend on the content of the particular image.
- The detected edges are to some degree robust as they do not depend much on small changes in illumination, viewpoint change, etc.

- It is interesting to seek yet richer features which can be reliably detected in the image and which can outperform simple edge detectors in some classes of applications.
- Line detectors and corner detectors are some such.

Why?

- Recall that one of the reasons why edges are being detected is that they bear a lot of information about underlying objects in the scene.
- Taking just edge elements instead of all pixels reduces the amount of data which has to be processed.
- The edge detector is a general tool which does not depend on the content of the particular image.
- The detected edges are to some degree robust as they do not depend much on small changes in illumination, viewpoint change, etc.

- It is interesting to seek yet richer features which can be reliably detected in the image and which can outperform simple edge detectors in some classes of applications.
- Line detectors and corner detectors are some such.

Why?

- Recall that one of the reasons why edges are being detected is that they bear a lot of information about underlying objects in the scene.
- Taking just edge elements instead of all pixels reduces the amount of data which has to be processed.
- The edge detector is a general tool which does not depend on the content of the particular image.
- The detected edges are to some degree robust as they do not depend much on small changes in illumination, viewpoint change, etc.

- It is interesting to seek yet richer features which can be reliably detected in the image and which can outperform simple edge detectors in some classes of applications.
- Line detectors and corner detectors are some such.

Why?

- Recall that one of the reasons why edges are being detected is that they bear a lot of information about underlying objects in the scene.
- Taking just edge elements instead of all pixels reduces the amount of data which has to be processed.
- The edge detector is a general tool which does not depend on the content of the particular image.
- The detected edges are to some degree robust as they do not depend much on small changes in illumination, viewpoint change, etc.

- It is interesting to seek yet richer features which can be reliably detected in the image and which can outperform simple edge detectors in some classes of applications.
- Line detectors and corner detectors are some such.

Line Detection I

- Line finding operators aim to find very thin curves in the image, e.g., roads in satellite images.
- It is assumed that curves do not bend sharply.
- Such curves and straight lines are called `lines` for the purpose of describing this technique.
- Lines are modeled by a roof profile among edges, Fig. 7.
- We assume that the width of the lines is approximately one or two pixels.
- Lines in the image can be detected by a number of local convolution operators h_k, which serve as line patterns [Cervenka and Charvat, 1987].
- The output value in pixel (i, j) is given by

$$f(i,j) = \max\left\{0, \max_k \left(f * h_k\right)\right\}. \qquad (95)$$

Line Detection I

- Line finding operators aim to find very thin curves in the image, e.g., roads in satellite images.
- It is assumed that curves do not bend sharply.
- Such curves and straight lines are called lines for the purpose of describing this technique.
- Lines are modeled by a roof profile among edges, Fig. 7.
- We assume that the width of the lines is approximately one or two pixels.
- Lines in the image can be detected by a number of local convolution operators h_k, which serve as line patterns [Cervenka and Charvat, 1987].
- The output value in pixel (i, j) is given by

$$f(i,j) = \max\left\{0, \max_k (f * h_k)\right\}. \quad (95)$$

Line Detection I

- Line finding operators aim to find very thin curves in the image, e.g., roads in satellite images.
- It is assumed that curves do not bend sharply.
- Such curves and straight lines are called lines for the purpose of describing this technique.
- Lines are modeled by a roof profile among edges, Fig. 7.
- We assume that the width of the lines is approximately one or two pixels.
- Lines in the image can be detected by a number of local convolution operators h_k, which serve as line patterns [Cervenka and Charvat, 1987].
- The output value in pixel (i, j) is given by

$$f(i,j) = \max\left\{0, \max_k (f * h_k)\right\}. \qquad (95)$$

Line Detection I

- Line finding operators aim to find very thin curves in the image, e.g., roads in satellite images.
- It is assumed that curves do not bend sharply.
- Such curves and straight lines are called lines for the purpose of describing this technique.
- Lines are modeled by a roof profile among edges, Fig. 7.
- We assume that the width of the lines is approximately one or two pixels.
- Lines in the image can be detected by a number of local convolution operators h_k, which serve as line patterns [Cervenka and Charvat, 1987].
- The output value in pixel (i, j) is given by

$$f(i,j) = \max \left\{ 0, \max_k (f * h_k) \right\} . \qquad (95)$$

Line Detection I

- Line finding operators aim to find very thin curves in the image, e.g., roads in satellite images.
- It is assumed that curves do not bend sharply.
- Such curves and straight lines are called lines for the purpose of describing this technique.
- Lines are modeled by a roof profile among edges, Fig. 7.
- We assume that the width of the lines is approximately one or two pixels.
- Lines in the image can be detected by a number of local convolution operators h_k, which serve as line patterns [Cervenka and Charvat, 1987].
- The output value in pixel (i, j) is given by

$$f(i,j) = \max\left\{0, \max_k (f * h_k)\right\}. \qquad (95)$$

Line Detection I

- Line finding operators aim to find very thin curves in the image, e.g., roads in satellite images.
- It is assumed that curves do not bend sharply.
- Such curves and straight lines are called lines for the purpose of describing this technique.
- Lines are modeled by a roof profile among edges, Fig. 7.
- We assume that the width of the lines is approximately one or two pixels.
- Lines in the image can be detected by a number of local convolution operators h_k, which serve as line patterns [Cervenka and Charvat, 1987].
- The output value in pixel (i, j) is given by

$$f(i,j) = \max\left\{0, \max_k (f * h_k)\right\}. \tag{95}$$

Line Detection I

- Line finding operators aim to find very thin curves in the image, e.g., roads in satellite images.
- It is assumed that curves do not bend sharply.
- Such curves and straight lines are called lines for the purpose of describing this technique.
- Lines are modeled by a roof profile among edges, Fig. 7.
- We assume that the width of the lines is approximately one or two pixels.
- Lines in the image can be detected by a number of local convolution operators h_k, which serve as line patterns [Cervenka and Charvat, 1987].
- The output value in pixel (i, j) is given by

$$f(i,j) = \max\left\{0, \max_k \left(f * h_k\right)\right\} . \qquad (95)$$

Line Detection II

- Four masks of size 3×3, able to detect lines rotated modulo the angle 45^o:

$$h_1 = \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}, \quad h_2 = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}, \quad h_3 = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

- A similar construction can be applied to bigger masks.

Line Detection II

- Four masks of size 3×3, able to detect lines rotated modulo the angle 45^o:

$$h_1 = \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}, \quad h_2 = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}, \quad h_3 = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

- A similar construction can be applied to bigger masks.

Line Detection III

- Convolution masks of size 5×5; 14 orientations;
- Only the first eight are shown, as the others are obvious by rotation.

$$h_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad h_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$h_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad h_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$h_5 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad h_6 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$h_7 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad h_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & 2 & 2 & 2 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- Such line detectors may produce more lines than needed, and other non-linear constraints should be added to reduce this number.

Line Detection III

- Convolution masks of size 5×5; 14 orientations;
- Only the first eight are shown, as the others are obvious by rotation.

$$h_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad h_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$h_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad h_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$h_5 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad h_6 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$h_7 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad h_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & 2 & 2 & 2 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- Such line detectors may produce more lines than needed, and other non-linear constraints should be added to reduce this number.

Line Detection III

- Convolution masks of size 5×5; 14 orientations;
- Only the first eight are shown, as the others are obvious by rotation.

$$h_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad h_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$h_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad h_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$h_5 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad h_6 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$h_7 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad h_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & 2 & 2 & 2 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- Such line detectors may produce more lines than needed, and other non-linear constraints should be added to reduce this number.

Line Thinning

- Thresholded edges are usually wider than one pixel, and line thinning techniques may give a better result.
- One line thinning method uses knowledge about orientation and in this case edges are thinned before thresholding.
  - Edge magnitude and directions provided by some gradient operator are used as input, and the edge magnitude of two neighboring pixels perpendicular to the edge direction are examined for each pixel in the image.
  - If at least one of these pixels has edge magnitude higher than the edge magnitude of the examined pixel, then the edge magnitude of the examined pixel is assigned a zero value.
  - The technique is called `non-maximal suppression` and is similar to the idea in Canny edge detector.
- There are many line thinning methods which we do not present here.
- In most cases the best line thinning is achieved using mathematical morphology methods (?).

Line Thinning

- Thresholded edges are usually wider than one pixel, and line thinning techniques may give a better result.
- One line thinning method uses knowledge about orientation and in this case edges are thinned before thresholding.
  - Edge magnitude and directions provided by some gradient operator are used as input, and the edge magnitude of two neighboring pixels perpendicular to the edge direction are examined for each pixel in the image.
  - If at least one of these pixels has edge magnitude higher than the edge magnitude of the examined pixel, then the edge magnitude of the examined pixel is assigned a zero value.
  - The technique is called `non-maximal suppression` and is similar to the idea in Canny edge detector.
- There are many line thinning methods which we do not present here.
- In most cases the best line thinning is achieved using mathematical morphology methods (?).

Line Thinning

- Thresholded edges are usually wider than one pixel, and line thinning techniques may give a better result.
- One line thinning method uses knowledge about orientation and in this case edges are thinned before thresholding.
  - Edge magnitude and directions provided by some gradient operator are used as input, and the edge magnitude of two neighboring pixels perpendicular to the edge direction are examined for each pixel in the image.
  - If at least one of these pixels has edge magnitude higher than the edge magnitude of the examined pixel, then the edge magnitude of the examined pixel is assigned a zero value.
  - The technique is called `non-maximal suppression` and is similar to the idea in Canny edge detector.
- There are many line thinning methods which we do not present here.
- In most cases the best line thinning is achieved using mathematical morphology methods (?).

Line Thinning

- Thresholded edges are usually wider than one pixel, and line thinning techniques may give a better result.
- One line thinning method uses knowledge about orientation and in this case edges are thinned before thresholding.
  - Edge magnitude and directions provided by some gradient operator are used as input, and the edge magnitude of two neighboring pixels perpendicular to the edge direction are examined for each pixel in the image.
  - If at least one of these pixels has edge magnitude higher than the edge magnitude of the examined pixel, then the edge magnitude of the examined pixel is assigned a zero value.
  - The technique is called `non-maximal suppression` and is similar to the idea in Canny edge detector.
- There are many line thinning methods which we do not present here.
- In most cases the best line thinning is achieved using mathematical morphology methods (?).

Line Thinning

- Thresholded edges are usually wider than one pixel, and line thinning techniques may give a better result.
- One line thinning method uses knowledge about orientation and in this case edges are thinned before thresholding.
  - Edge magnitude and directions provided by some gradient operator are used as input, and the edge magnitude of two neighboring pixels perpendicular to the edge direction are examined for each pixel in the image.
  - If at least one of these pixels has edge magnitude higher than the edge magnitude of the examined pixel, then the edge magnitude of the examined pixel is assigned a zero value.
  - The technique is called `non-maximal suppression` and is similar to the idea in Canny edge detector.
- There are many line thinning methods which we do not present here.
- In most cases the best line thinning is achieved using mathematical morphology methods (?).

Line Thinning

- Thresholded edges are usually wider than one pixel, and line thinning techniques may give a better result.
- One line thinning method uses knowledge about orientation and in this case edges are thinned before thresholding.
  - Edge magnitude and directions provided by some gradient operator are used as input, and the edge magnitude of two neighboring pixels perpendicular to the edge direction are examined for each pixel in the image.
  - If at least one of these pixels has edge magnitude higher than the edge magnitude of the examined pixel, then the edge magnitude of the examined pixel is assigned a zero value.
  - The technique is called `non-maximal suppression` and is similar to the idea in Canny edge detector.
- There are many line thinning methods which we do not present here.
- In most cases the best line thinning is achieved using mathematical morphology methods (?).

Line Thinning

- Thresholded edges are usually wider than one pixel, and line thinning techniques may give a better result.
- One line thinning method uses knowledge about orientation and in this case edges are thinned before thresholding.
  - Edge magnitude and directions provided by some gradient operator are used as input, and the edge magnitude of two neighboring pixels perpendicular to the edge direction are examined for each pixel in the image.
  - If at least one of these pixels has edge magnitude higher than the edge magnitude of the examined pixel, then the edge magnitude of the examined pixel is assigned a zero value.
  - The technique is called `non-maximal suppression` and is similar to the idea in Canny edge detector.
- There are many line thinning methods which we do not present here.
- In most cases the best line thinning is achieved using mathematical morphology methods (?).

Edge Filling

- Edge points after thresholding do not create contiguous boundaries and the edge filling method tries to recover edge pixels on the potential object boundary which are missing.
- The following is a very simple local edge filling technique.
- The local edge filling procedure [Cervenka and Charvat, 1987] checks the 3×3 neighborhood of the current pixel matches one of the following masks

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

If so, the central pixel of the mask is changed from zero to one.

Edge Filling

- Edge points after thresholding do not create contiguous boundaries and the edge filling method tries to recover edge pixels on the potential object boundary which are missing.
- The following is a very simple local edge filling technique.
- The local edge filling procedure [Cervenka and Charvat, 1987] checks the 3×3 neighborhood of the current pixel matches one of the following masks

$$
\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}
\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}
\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}
$$

$$
\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}
$$

If so, the central pixel of the mask is changed from zero to one.

Edge Filling

- Edge points after thresholding do not create contiguous boundaries and the edge filling method tries to recover edge pixels on the potential object boundary which are missing.
- The following is a very simple local edge filling technique.
- The local edge filling procedure [Cervenka and Charvat, 1987] checks the 3×3 neighborhood of the current pixel matches one of the following masks

$$\begin{bmatrix}0&1&0\\0&0&0\\0&1&0\end{bmatrix} \quad \begin{bmatrix}0&0&0\\1&0&1\\0&0&0\end{bmatrix} \quad \begin{bmatrix}1&0&0\\0&0&0\\0&1&0\end{bmatrix} \quad \begin{bmatrix}0&0&1\\1&0&0\\0&0&0\end{bmatrix}$$

$$\begin{bmatrix}0&1&0\\0&0&0\\0&0&1\end{bmatrix} \quad \begin{bmatrix}0&0&0\\0&0&1\\1&0&0\end{bmatrix} \quad \begin{bmatrix}1&0&0\\0&0&0\\0&0&1\end{bmatrix} \quad \begin{bmatrix}0&0&1\\0&0&0\\1&0&0\end{bmatrix}$$

If so, the central pixel of the mask is changed from zero to one.

Discussions

- These simple methods for edge thinning and filling do not guarantee that the width of the lines will be equal to one and the contiguity of the edges are is not certain either.
- Note that local thinning and filling operators can be treated as special cases of mathematical morphology operators.

Discussions

- These simple methods for edge thinning and filling do not guarantee that the width of the lines will be equal to one and the contiguity of the edges are is not certain either.
- Note that local thinning and filling operators can be treated as special cases of mathematical morphology operators.

Outline

- Pixel Brightness Transformations
  - Position dependent brightness correction
  - Grey scale transformation
    - Windows and level
    - Histogram stretching
    - Histogram Equalization
    - Histogram Matching
- Geometric Transformations
  - Pixel Co-ordinate Transformations
    - Polynomial Approximation
    - Bilinear Transformations
    - Important Transformations
  - Brightness Interpolation
    - Nearest Neighbor Interpolation
    - Bilinear Interpolation
    - Bi-cubic interpolation
- **Local pre-processing**
  - Image smoothing
    - Averaging
    - Averaging with Data Validity
    - Averaging According to Inverse Gradient

Corresponding Points

- In many cases it is of advantage to find pairs of `corresponding points` in two similar images.
- E.g., when finding geometric transforms, knowing the position of corresponding points enables the estimation of geometric transforms from live data.
- Finding corresponding points is also a core problem in the analysis of moving images and for recovering depth information form pairs of stereo images.

Corresponding Points

- In many cases it is of advantage to find pairs of `corresponding points` in two similar images.
- E.g., when finding geometric transforms, knowing the position of corresponding points enables the estimation of geometric transforms from live data.
- Finding corresponding points is also a core problem in the analysis of moving images and for recovering depth information form pairs of stereo images.

Corresponding Points

- In many cases it is of advantage to find pairs of `corresponding points` in two similar images.
- E.g., when finding geometric transforms, knowing the position of corresponding points enables the estimation of geometric transforms from live data.
- Finding corresponding points is also a core problem in the analysis of moving images and for recovering depth information form pairs of stereo images.

Interest Points

- In general, all possible pairs of points should be examined to solve this `correspondence problem`, and this is very computation expensive.
- If two images have n pixels each, the complexity is $O(n^2)$.
- This process might be simplified if the correspondence is examined among a much smaller number of points, called `interest points`.
- An interest point should have some typical local property.
- E.g., if square objects are present in the image, then `corners` are very good interest points.

Interest Points

- In general, all possible pairs of points should be examined to solve this `correspondence problem`, and this is very computation expensive.
- If two images have n pixels each, the complexity is $O(n^2)$.
- This process might be simplified if the correspondence is examined among a much smaller number of points, called `interest points`.
- An interest point should have some typical local property.
- E.g., if square objects are present in the image, then `corners` are very good interest points.

Interest Points

- In general, all possible pairs of points should be examined to solve this `correspondence problem`, and this is very computation expensive.
- If two images have n pixels each, the complexity is $O(n^2)$.
- This process might be simplified if the correspondence is examined among a much smaller number of points, called `interest points`.
- An interest point should have some typical local property.
- E.g., if square objects are present in the image, then `corners` are very good interest points.

Interest Points

- In general, all possible pairs of points should be examined to solve this `correspondence problem`, and this is very computation expensive.
- If two images have n pixels each, the complexity is $O(n^2)$.
- This process might be simplified if the correspondence is examined among a much smaller number of points, called `interest points`.
- An interest point should have some typical local property.
- E.g., if square objects are present in the image, then `corners` are very good interest points.

Interest Points

- In general, all possible pairs of points should be examined to solve this `correspondence problem`, and this is very computation expensive.
- If two images have n pixels each, the complexity is $O(n^2)$.
- This process might be simplified if the correspondence is examined among a much smaller number of points, called `interest points`.
- An interest point should have some typical local property.
- E.g., if square objects are present in the image, then `corners` are very good interest points.

Corners

- Corners serve better than lines for the correspondence problem.
- This is due to the `aperture problem`.
- Assume a moving line is seen through a small aperture.
  - In such a case, only the motion vector perpendicular to the line can be observed.
  - The component collinear with the line remains invisible.
- The situation is better with corners. They provide ground for unique matching, cf. the following figure for illustration.

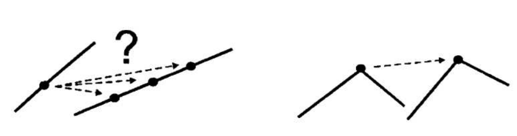


Figure: Ambiguity of lines for matching and unambiguity of corners.

Corners

- Corners serve better than lines for the correspondence problem.
- This is due to the `aperture problem`.
- Assume a moving line is seen through a small aperture.
  - In such a case, only the motion vector perpendicular to the line can be observed.
  - The component collinear with the line remains invisible.
- The situation is better with corners. They provide ground for unique matching, cf. the following figure for illustration.

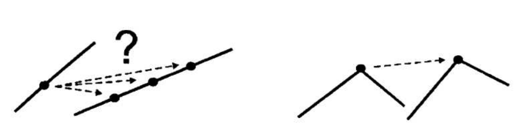


Figure: Ambiguity of lines for matching and unambiguity of corners.

Corners

- Corners serve better than lines for the correspondence problem.
- This is due to the `aperture problem`.
- Assume a moving line is seen through a small aperture.
  - In such a case, only the motion vector perpendicular to the line can be observed.
  - The component collinear with the line remains invisible.
- The situation is better with corners. They provide ground for unique matching, cf. the following figure for illustration.

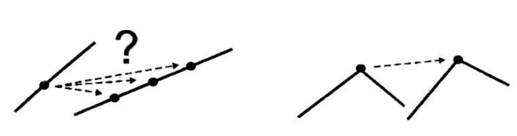


Figure: Ambiguity of lines for matching and unambiguity of corners.

Corners

- Corners serve better than lines for the correspondence problem.
- This is due to the `aperture problem`.
- Assume a moving line is seen through a small aperture.
  - In such a case, only the motion vector perpendicular to the line can be observed.
  - The component collinear with the line remains invisible.
- The situation is better with corners. They provide ground for unique matching, cf. the following figure for illustration.

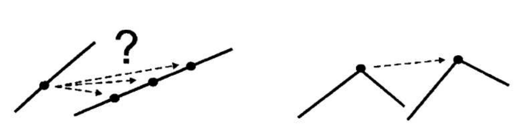


Figure: Ambiguity of lines for matching and unambiguity of corners.

Corners

- Corners serve better than lines for the correspondence problem.
- This is due to the `aperture problem`.
- Assume a moving line is seen through a small aperture.
  - In such a case, only the motion vector perpendicular to the line can be observed.
  - The component collinear with the line remains invisible.
- The situation is better with corners. They provide ground for unique matching, cf. the following figure for illustration.

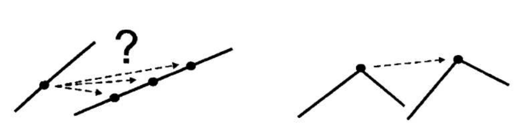


Figure: Ambiguity of lines for matching and unambiguity of corners.

Corners

- Corners serve better than lines for the correspondence problem.
- This is due to the `aperture problem`.
- Assume a moving line is seen through a small aperture.
  - In such a case, only the motion vector perpendicular to the line can be observed.
  - The component collinear with the line remains invisible.
- The situation is better with corners. They provide ground for unique matching, cf. the following figure for illustration.

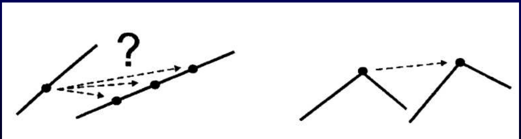


Figure: Ambiguity of lines for matching and unambiguity of corners.

Corner Detectors

- Edge detectors themselves are not stable at corners.
- This is natural as the gradient at the tip of the corner is ambiguous.
- Near the corner there is a discontinuity in the gradient direction.

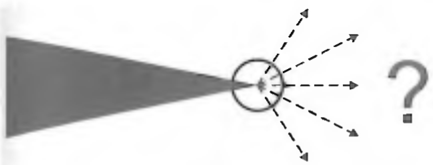


Figure: Ambiguity of lines for matching and unambiguity of corners.

- This observation is used in corner detectors.

Corner Detectors

- Edge detectors themselves are not stable at corners.
- This is natural as the gradient at the tip of the corner is ambiguous.
- Near the corner there is a discontinuity in the gradient direction.

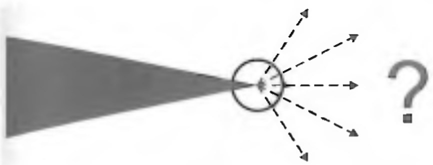


Figure: Ambiguity of lines for matching and unambiguity of corners.

- This observation is used in corner detectors.

Corner Detectors

- Edge detectors themselves are not stable at corners.
- This is natural as the gradient at the tip of the corner is ambiguous.
- Near the corner there is a discontinuity in the gradient direction.

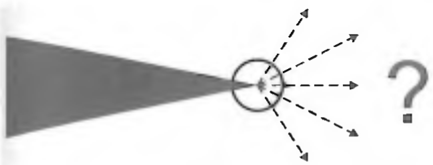


Figure: Ambiguity of lines for matching and unambiguity of corners.

- This observation is used in corner detectors.

Corner Detectors

- Edge detectors themselves are not stable at corners.
- This is natural as the gradient at the tip of the corner is ambiguous.
- Near the corner there is a discontinuity in the gradient direction.

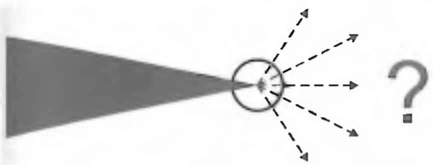


Figure: Ambiguity of lines for matching and unambiguity of corners.

- This observation is used in corner detectors.

Corners

- The corner in the image can be defined as a pixel in whose small neighborhood there are two dominant and different edge directions.
- This definition is not precise as an isolated point of local intensity maximum or minimum, line endings, or an abrupt change in the curvature of a curve with a response similar to a corner.
- Nevertheless, such detectors are named corner detectors in the literature and are widely used.
- If corners have to be detected then some additional constraints have to be applied.

Corners

- The corner in the image can be defined as a pixel in whose small neighborhood there are two dominant and different edge directions.
- This definition is not precise as an isolated point of local intensity maximum or minimum, line endings, or an abrupt change in the curvature of a curve with a response similar to a corner.
- Nevertheless, such detectors are named corner detectors in the literature and are widely used.
- If corners have to be detected then some additional constraints have to be applied.

Corners

- The corner in the image can be defined as a pixel in whose small neighborhood there are two dominant and different edge directions.
- This definition is not precise as an isolated point of local intensity maximum or minimum, line endings, or an abrupt change in the curvature of a curve with a response similar to a corner.
- Nevertheless, such detectors are named corner detectors in the literature and are widely used.
- If corners have to be detected then some additional constraints have to be applied.

Corners

- The corner in the image can be defined as a pixel in whose small neighborhood there are two dominant and different edge directions.
- This definition is not precise as an isolated point of local intensity maximum or minimum, line endings, or an abrupt change in the curvature of a curve with a response similar to a corner.
- Nevertheless, such detectors are named corner detectors in the literature and are widely used.
- If corners have to be detected then some additional constraints have to be applied.

Corner Detectors

- Corner detectors are not usually very robust.
- This deficiency is overcome either by manual expert supervision or large redundancies introduced to prevent the effect of individual errors from dominating the task.
- The latter means that many more corners are detected in two or more images than necessary for estimating a transformation sought between these images.

Corner Detectors

- Corner detectors are not usually very robust.
- This deficiency is overcome either by manual expert supervision or large redundancies introduced to prevent the effect of individual errors from dominating the task.
- The latter means that many more corners are detected in two or more images than necessary for estimating a transformation sought between these images.

Corner Detectors

- Corner detectors are not usually very robust.
- This deficiency is overcome either by manual expert supervision or large redundancies introduced to prevent the effect of individual errors from dominating the task.
- The latter means that many more corners are detected in two or more images than necessary for estimating a transformation sought between these images.

Moravec Detector

- The simplest corner detector is the `Moravec detector` which is maximal in pixels with high contrast.
- These points are on corners and sharp edges.
- The Moravec operator MO is given by

$$MO(i,j) = \frac{1}{8} \sum_{k=i-1}^{i+1} \sum_{l=j-1}^{j+1} |g(k,l) - g(i,j)|. \qquad (96)$$

Moravec Detector

- The simplest corner detector is the `Moravec detector` which is maximal in pixels with high contrast.
- These points are on corners and sharp edges.
- The Moravec operator MO is given by

$$\mathrm{MO}(i,j) = \frac{1}{8} \sum_{k=i-1}^{i+1} \sum_{l=j-1}^{j+1} |g(k,l) - g(i,j)|. \qquad (96)$$

Moravec Detector

- The simplest corner detector is the `Moravec detector` which is maximal in pixels with high contrast.
- These points are on corners and sharp edges.
- The Moravec operator MO is given by

$$\mathrm{MO}(i,j) = \frac{1}{8} \sum_{k=i-1}^{i+1} \sum_{l=j-1}^{j+1} |g(k,l) - g(i,j)|. \qquad (96)$$

Corner Detectors by Facet Model

- Better results are based on the facet model, (94).
- The image g is approximated by a bi-cubic polynomial

$$g(x,y) = c_1 + c_2 x + c_3 y + c_4 x^2 + c_5 xy + c_6 y^2 + c_7 x^3 + c_8 x^2 y + c_9 xy^2 + c_{10} y^3. \quad (97)$$

- The Zuniga-Haralick operator ZH is given by

$$\mathrm{ZH}(i,j) = \frac{-\left(c_2^2 c_6 - 2c_2 c_3 c_5 + c_3^2 c_4\right)}{\left(c_2^2 + c_3^2\right)^{\frac{3}{2}}}, \quad (98)$$

which is the curvature of $g(x,y) = \text{const}$.
- The Kitchen-Rosenfeld KR operator is given by

$$\mathrm{KR}(i,j) = \frac{-\left(c_2^2 c_6 - 2c_2 c_3 c_5 + c_3^2 c_4\right)}{c_2^2 + c_3^2}, \quad (99)$$

which is the second order derivative along the direction of edge.
- The ZH operator has been shown to outperform the KR detector in test images.

Corner Detectors by Facet Model

- Better results are based on the facet model, (94).
- The image g is approximated by a bi-cubic polynomial

$$g(x,y) = c_1 + c_2 x + c_3 y + c_4 x^2 + c_5 xy + c_6 y^2 + c_7 x^3 + c_8 x^2 y + c_9 xy^2 + c_{10} y^3. \quad (97)$$

- The Zuniga-Haralick operator ZH is given by

$$\mathrm{ZH}(i,j) = \frac{-\left(c_2^2 c_6 - 2c_2 c_3 c_5 + c_3^2 c_4\right)}{\left(c_2^2 + c_3^2\right)^{\frac{3}{2}}}, \quad (98)$$

  which is the curvature of $g(x,y) = \mathrm{const}$.
- The Kitchen-Rosenfeld KR operator is given by

$$\mathrm{KR}(i,j) = \frac{-\left(c_2^2 c_6 - 2c_2 c_3 c_5 + c_3^2 c_4\right)}{c_2^2 + c_3^2}, \quad (99)$$

  which is the second order derivative along the direction of edge.
- The ZH operator has been shown to outperform the KR detector in test images.

Corner Detectors by Facet Model

- Better results are based on the facet model, (94).
- The image g is approximated by a bi-cubic polynomial

$$g(x, y) = c_1 + c_2 x + c_3 y + c_4 x^2 + c_5 xy + c_6 y^2 + c_7 x^3 + c_8 x^2 y + c_9 xy^2 + c_{10} y^3. \quad (97)$$

- The Zuniga-Haralick operator ZH is given by

$$\mathrm{ZH}(i, j) = \frac{-\left(c_2^2 c_6 - 2 c_2 c_3 c_5 + c_3^2 c_4\right)}{\left(c_2^2 + c_3^2\right)^{\frac{3}{2}}}, \quad (98)$$

which is the curvature of $g(x, y) = \text{const}$.

- The Kitchen-Rosenfeld KR operator is given by

$$\mathrm{KR}(i, j) = \frac{-\left(c_2^2 c_6 - 2 c_2 c_3 c_5 + c_3^2 c_4\right)}{c_2^2 + c_3^2}, \quad (99)$$

which is the second order derivative along the direction of edge.
- The ZH operator has been shown to outperform the KR detector in test images.

Corner Detectors by Facet Model

- Better results are based on the facet model, (94).
- The image g is approximated by a bi-cubic polynomial

$$g(x,y) = c_1 + c_2 x + c_3 y + c_4 x^2 + c_5 xy + c_6 y^2 + c_7 x^3 + c_8 x^2 y + c_9 xy^2 + c_{10} y^3. \quad (97)$$

- The Zuniga-Haralick operator ZH is given by

$$\mathrm{ZH}(i,j) = \frac{-\left(c_2^2 c_6 - 2 c_2 c_3 c_5 + c_3^2 c_4\right)}{\left(c_2^2 + c_3^2\right)^{\frac{3}{2}}}, \quad (98)$$

which is the curvature of $g(x,y) = \text{const}$.
- The Kitchen-Rosenfeld KR operator is given by

$$\mathrm{KR}(i,j) = \frac{-\left(c_2^2 c_6 - 2 c_2 c_3 c_5 + c_3^2 c_4\right)}{c_2^2 + c_3^2}, \quad (99)$$

which is the second order derivative along the direction of edge.
- The ZH operator has been shown to outperform the KR detector in test images.

Corner Detectors by Facet Model

- Better results are based on the facet model, (94).
- The image g is approximated by a bi-cubic polynomial

$$g(x,y) = c_1 + c_2 x + c_3 y + c_4 x^2 + c_5 xy + c_6 y^2 + c_7 x^3 + c_8 x^2 y + c_9 x y^2 + c_{10} y^3. \quad (97)$$

- The Zuniga-Haralick operator ZH is given by

$$\mathrm{ZH}(i,j) = \frac{-\left(c_2^2 c_6 - 2 c_2 c_3 c_5 + c_3^2 c_4\right)}{\left(c_2^2 + c_3^2\right)^{\frac{3}{2}}}, \quad (98)$$

  which is the curvature of $g(x,y) = \text{const}$.
- The Kitchen-Rosenfeld KR operator is given by

$$\mathrm{KR}(i,j) = \frac{-\left(c_2^2 c_6 - 2 c_2 c_3 c_5 + c_3^2 c_4\right)}{c_2^2 + c_3^2}, \quad (99)$$

  which is the second order derivative along the direction of edge.
- The ZH operator has been shown to outperform the KR detector in test images.

Harris Corner Detector I

- The Harris corner detector improved upon Moravec's by considering the differential of the corner score (sum of square differences).
- Consider a 2D gray-scale image f.
- An image patch W in f is taken and is shifted by Δx and Δy.
- The sum S of square differences between values of the image f given by the patch W and its shifted variant by Δx and Δy is given by:

$$S_W(\Delta x, \Delta y) = \sum_{(x_i, y_i) \in W} \left(f(x_i, y_i) - f(x_i - \Delta x, y_i - \Delta y)\right)^2 . \tag{100}$$

- A corner point not suffering from the aperture problem must have a high response of $S_W(\Delta x, \Delta y)$ for all Δx and Δy.

Harris Corner Detector I

- The Harris corner detector improved upon Moravec's by considering the differential of the corner score (sum of square differences).
- Consider a 2D gray-scale image f.
- An image patch W in f is taken and is shifted by Δx and Δy.
- The sum S of square differences between values of the image f given by the patch W and its shifted variant by Δx and Δy is given by:

$$S_W(\Delta x, \Delta y) = \sum_{(x_i, y_i) \in W} \left(f(x_i, y_i) - f(x_i - \Delta x, y_i - \Delta y)\right)^2 . \tag{100}$$

- A corner point not suffering from the aperture problem must have a high response of $S_W(\Delta x, \Delta y)$ for all Δx and Δy.

Harris Corner Detector I

- The Harris corner detector improved upon Moravec's by considering the differential of the corner score (sum of square differences).
- Consider a 2D gray-scale image f.
- An image patch W in f is taken and is shifted by Δx and Δy.
- The sum S of square differences between values of the image f given by the patch W and its shifted variant by Δx and Δy is given by:

$$S_W(\Delta x, \Delta y) = \sum_{(x_i, y_i) \in W} (f(x_i, y_i) - f(x_i - \Delta x, y_i - \Delta y))^2 . \tag{100}$$

- A corner point not suffering from the aperture problem must have a high response of $S_W(\Delta x, \Delta y)$ for all Δx and Δy.

Harris Corner Detector I

- The Harris corner detector improved upon Moravec's by considering the differential of the corner score (sum of square differences).
- Consider a 2D gray-scale image f.
- An image patch W in f is taken and is shifted by Δx and Δy.
- The sum S of square differences between values of the image f given by the patch W and its shifted variant by Δx and Δy is given by:

$$S_W(\Delta x, \Delta y) = \sum_{(x_i, y_i) \in W} \left(f(x_i, y_i) - f(x_i - \Delta x, y_i - \Delta y)\right)^2 . \tag{100}$$

- A corner point not suffering from the aperture problem must have a high response of $S_W(\Delta x, \Delta y)$ for all Δx and Δy.

Harris Corner Detector I

- The Harris corner detector improved upon Moravec's by considering the differential of the corner score (sum of square differences).
- Consider a 2D gray-scale image f.
- An image patch W in f is taken and is shifted by Δx and Δy.
- The sum S of square differences between values of the image f given by the patch W and its shifted variant by Δx and Δy is given by:

$$S_W(\Delta x, \Delta y) = \sum_{(x_i, y_i) \in W} \left(f(x_i, y_i) - f(x_i - \Delta x, y_i - \Delta y)\right)^2 . \tag{100}$$

- A corner point not suffering from the aperture problem must have a high response of $S_W(\Delta x, \Delta y)$ for all Δx and Δy.

Harris Corner Detector II

- By the first-order Taylor expansion

$$f(x_i-\Delta x, y_i-\Delta y) = f(x_i, y_i) - \frac{\partial f}{\partial x}(x_i, y_i)\Delta x - \frac{\partial f}{\partial y}(x_i, y_i)\Delta y, \quad (101)$$

the minimum of $S_W(\Delta x, \Delta y)$ can be obtained analytically.

- Substituting (101) into (100),

$$S_W(\Delta x, \Delta y) = \sum_{(x_i,y_i)\in W} \left(\frac{\partial f}{\partial x}(x_i, y_i)\Delta x + \frac{\partial f}{\partial y}(x_i, y_i)\Delta y, \right)^2$$

$$= \begin{pmatrix} \Delta x & \Delta y \end{pmatrix} A_W(x, y) \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

where the Harris matrix

$$A_W(x, y) = \sum_{(x_i,y_i)\in W} \begin{pmatrix} \left(\frac{\partial f}{\partial x}(x_i, y_i)\right)^2 & \frac{\partial f}{\partial x}(x_i, y_i)\frac{\partial f}{\partial y}(x_i, y_i) \\ \frac{\partial f}{\partial x}(x_i, y_i)\frac{\partial f}{\partial y}(x_i, y_i) & \left(\frac{\partial f}{\partial y}(x_i, y_i)\right)^2 \end{pmatrix} \quad (102)$$

Harris Corner Detector II

- By the first-order Taylor expansion

$$f(x_i - \Delta x, y_i - \Delta y) = f(x_i, y_i) - \frac{\partial f}{\partial x}(x_i, y_i)\Delta x - \frac{\partial f}{\partial y}(x_i, y_i)\Delta y, \quad (101)$$

  the minimum of $S_W(\Delta x, \Delta y)$ can be obtained analytically.

- Substituting (101) into (100),

$$S_W(\Delta x, \Delta y) = \sum_{(x_i, y_i) \in W} \left(\frac{\partial f}{\partial x}(x_i, y_i)\Delta x + \frac{\partial f}{\partial y}(x_i, y_i)\Delta y, \right)^2$$

$$= \begin{pmatrix} \Delta x & \Delta y \end{pmatrix} A_W(x, y) \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

  where the Harris matrix

$$A_W(x, y) = \sum_{(x_i, y_i) \in W} \begin{pmatrix} \left(\frac{\partial f}{\partial x}(x_i, y_i)\right)^2 & \frac{\partial f}{\partial x}(x_i, y_i)\frac{\partial f}{\partial y}(x_i, y_i) \\ \frac{\partial f}{\partial x}(x_i, y_i)\frac{\partial f}{\partial y}(x_i, y_i) & \left(\frac{\partial f}{\partial y}(x_i, y_i)\right)^2 \end{pmatrix} \quad (102)$$

Harris Corner Detector III

- The Harris matrix A_W is symmetric and positive semi-definite.
- Its main modes of variation correspond to partial derivatives in orthogonal directions and are reflected in eigenvalues λ_1 and λ_2 of the matrix A_W.
- Three distinct cases can appear:
  1. Both eigenvalues are small. This means that image f is flat in the examined pixel. There are no edges or corners in this location.
  2. One eigenvalue is small and the second one large. The local neighborhood is ridge-shaped. Significant change of image f occurs if a small movement is made perpendicularly to the ridge.
  3. Both eigenvalues are rather large. A small shift in any direction causes significant change of the image f. A corner is found.

Harris Corner Detector III

- The Harris matrix A_W is symmetric and positive semi-definite.
- Its main modes of variation correspond to partial derivatives in orthogonal directions and are reflected in eigenvalues λ_1 and λ_2 of the matrix A_W.
- Three distinct cases can appear:
  1. Both eigenvalues are small. This means that image f is flat in the examined pixel. There are no edges or corners in this location.
  2. One eigenvalue is small and the second one large. The local neighborhood is ridge-shaped. Significant change of image f occurs if a small movement is made perpendicularly to the ridge.
  3. Both eigenvalues are rather large. A small shift in any direction causes significant change of the image f. A corner is found.

Harris Corner Detector III

- The Harris matrix A_W is symmetric and positive semi-definite.
- Its main modes of variation correspond to partial derivatives in orthogonal directions and are reflected in eigenvalues λ_1 and λ_2 of the matrix A_W.
- Three distinct cases can appear:
  1. Both eigenvalues are small. This means that image f is flat in the examined pixel. There are no edges or corners in this location.
  2. One eigenvalue is small and the second one large. The local neighborhood is ridge-shaped. Significant change of image f occurs if a small movement is made perpendicularly to the ridge.
  3. Both eigenvalues are rather large. A small shift in any direction causes significant change of the image f. A corner is found.

Harris Corner Detector III

- The Harris matrix A_W is symmetric and positive semi-definite.
- Its main modes of variation correspond to partial derivatives in orthogonal directions and are reflected in eigenvalues λ_1 and λ_2 of the matrix A_W.
- Three distinct cases can appear:
  1. Both eigenvalues are small. This means that image f is flat in the examined pixel. There are no edges or corners in this location.
  2. One eigenvalue is small and the second one large. The local neighborhood is ridge-shaped. Significant change of image f occurs if a small movement is made perpendicularly to the ridge.
  3. Both eigenvalues are rather large. A small shift in any direction causes significant change of the image f. A corner is found.

Harris Corner Detector III

- The Harris matrix A_W is symmetric and positive semi-definite.
- Its main modes of variation correspond to partial derivatives in orthogonal directions and are reflected in eigenvalues λ_1 and λ_2 of the matrix A_W.
- Three distinct cases can appear:
  1. Both eigenvalues are small. This means that image f is flat in the examined pixel. There are no edges or corners in this location.
  2. One eigenvalue is small and the second one large. The local neighborhood is ridge-shaped. Significant change of image f occurs if a small movement is made perpendicularly to the ridge.
  3. Both eigenvalues are rather large. A small shift in any direction causes significant change of the image f. A corner is found.

Harris Corner Detector III

- The Harris matrix A_W is symmetric and positive semi-definite.
- Its main modes of variation correspond to partial derivatives in orthogonal directions and are reflected in eigenvalues λ_1 and λ_2 of the matrix A_W.
- Three distinct cases can appear:
  1. Both eigenvalues are small. This means that image f is flat in the examined pixel. There are no edges or corners in this location.
  2. One eigenvalue is small and the second one large. The local neighborhood is ridge-shaped. Significant change of image f occurs if a small movement is made perpendicularly to the ridge.
  3. Both eigenvalues are rather large. A small shift in any direction causes significant change of the image f. A corner is found.

Harris Corner Detector IV

- Harris suggested that exact eigenvalue computation can be avoided by calculating the response function $R(A) = \det(A) - \kappa \mathrm{trace}^2(A)$, where κ is a tunable parameter where values from 0.04 to 0.15 were reported in literature as appropriate.
- Harris corner detector
  1. Filter the image with a Gaussian.
  2. Estimate intensity gradient in two perpendicular directions for each pixel. This is performed by twice using a 1D convolution with the kernel approximating the derivative.
  3. For each pixel and a given neighborhood window:
     - Calculate the local structure matrix A;
     - Evaluate the response function $R(A)$.
  4. Choose the best candidates for corners by selecting a threshold on the response function $R(A)$ and perform non-maximal suppression.

Harris Corner Detector IV

- Harris suggested that exact eigenvalue computation can be avoided by calculating the response function $R(A) = \det(A) - \kappa \text{trace}^2(A)$, where κ is a tunable parameter where values from 0.04 to 0.15 were reported in literature as appropriate.
- Harris corner detector
  1. Filter the image with a Gaussian.
  2. Estimate intensity gradient in two perpendicular directions for each pixel. This is performed by twice using a 1D convolution with the kernel approximating the derivative.
  3. For each pixel and a given neighborhood window:
     - Calculate the local structure matrix A;
     - Evaluate the response function $R(A)$.
  4. Choose the best candidates for corners by selecting a threshold on the response function $R(A)$ and perform non-maximal suppression.

Harris Corner Detector IV

- Harris suggested that exact eigenvalue computation can be avoided by calculating the response function $R(A) = \det(A) - \kappa \mathrm{trace}^2(A)$, where κ is a tunable parameter where values from 0.04 to 0.15 were reported in literature as appropriate.
- Harris corner detector
  1. Filter the image with a Gaussian.
  2. Estimate intensity gradient in two perpendicular directions for each pixel. This is performed by twice using a 1D convolution with the kernel approximating the derivative.
  3. For each pixel and a given neighborhood window:
     - Calculate the local structure matrix A;
     - Evaluate the response function $R(A)$.
  4. Choose the best candidates for corners by selecting a threshold on the response function $R(A)$ and perform non-maximal suppression.

Harris Corner Detector IV

- Harris suggested that exact eigenvalue computation can be avoided by calculating the response function $R(A) = \det(A) - \kappa \mathrm{trace}^2(A)$, where κ is a tunable parameter where values from 0.04 to 0.15 were reported in literature as appropriate.
- Harris corner detector
  1. Filter the image with a Gaussian.
  2. Estimate intensity gradient in two perpendicular directions for each pixel. This is performed by twice using a 1D convolution with the kernel approximating the derivative.
  3. For each pixel and a given neighborhood window:
     - Calculate the local structure matrix A;
     - Evaluate the response function $R(A)$.
  4. Choose the best candidates for corners by selecting a threshold on the response function $R(A)$ and perform non-maximal suppression.

Harris Corner Detector IV

- Harris suggested that exact eigenvalue computation can be avoided by calculating the response function $R(A) = \det(A) - \kappa \mathrm{trace}^2(A)$, where κ is a tunable parameter where values from 0.04 to 0.15 were reported in literature as appropriate.
- Harris corner detector
  1. Filter the image with a Gaussian.
  2. Estimate intensity gradient in two perpendicular directions for each pixel. This is performed by twice using a 1D convolution with the kernel approximating the derivative.
  3. For each pixel and a given neighborhood window:
     - Calculate the local structure matrix A;
     - Evaluate the response function $R(A)$.
  4. Choose the best candidates for corners by selecting a threshold on the response function $R(A)$ and perform non-maximal suppression.

Harris Corner Detector IV

- Harris suggested that exact eigenvalue computation can be avoided by calculating the response function $R(A) = \det(A) - \kappa \mathrm{trace}^2(A)$, where κ is a tunable parameter where values from 0.04 to 0.15 were reported in literature as appropriate.
- Harris corner detector
  1. Filter the image with a Gaussian.
  2. Estimate intensity gradient in two perpendicular directions for each pixel. This is performed by twice using a 1D convolution with the kernel approximating the derivative.
  3. For each pixel and a given neighborhood window:
     - Calculate the local structure matrix A;
     - Evaluate the response function $R(A)$.
  4. Choose the best candidates for corners by selecting a threshold on the response function $R(A)$ and perform non-maximal suppression.

Harris Corner Detector IV

- Harris suggested that exact eigenvalue computation can be avoided by calculating the response function $R(A) = \det(A) - \kappa \mathrm{trace}^2(A)$, where κ is a tunable parameter where values from 0.04 to 0.15 were reported in literature as appropriate.
- Harris corner detector
  1. Filter the image with a Gaussian.
  2. Estimate intensity gradient in two perpendicular directions for each pixel. This is performed by twice using a 1D convolution with the kernel approximating the derivative.
  3. For each pixel and a given neighborhood window:
     - Calculate the local structure matrix A;
     - Evaluate the response function $R(A)$.
  4. Choose the best candidates for corners by selecting a threshold on the response function $R(A)$ and perform non-maximal suppression.

Harris Corner Detector IV

- Harris suggested that exact eigenvalue computation can be avoided by calculating the response function $R(A) = \det(A) - \kappa \mathrm{trace}^2(A)$, where κ is a tunable parameter where values from 0.04 to 0.15 were reported in literature as appropriate.
- Harris corner detector
  1. Filter the image with a Gaussian.
  2. Estimate intensity gradient in two perpendicular directions for each pixel. This is performed by twice using a 1D convolution with the kernel approximating the derivative.
  3. For each pixel and a given neighborhood window:
     - Calculate the local structure matrix A;
     - Evaluate the response function $R(A)$.
  4. Choose the best candidates for corners by selecting a threshold on the response function $R(A)$ and perform non-maximal suppression.

Harris Corner Detector V

- The Harris corner detector has been very popular.
- Its advantages are insensitivity to 2D shift and rotation, to small illumination variations, to small viewpoint change, and its low computational requirements.
- On the other hand, it is not invariant to larger scale change, viewpoint changes and significant changes in contrast.
- Many more corner-like detectors exist, and the reader is referred to the overview papers.

Harris Corner Detector V

- The Harris corner detector has been very popular.
- Its advantages are insensitivity to 2D shift and rotation, to small illumination variations, to small viewpoint change, and its low computational requirements.
- On the other hand, it is not invariant to larger scale change, viewpoint changes and significant changes in contrast.
- Many more corner-like detectors exist, and the reader is referred to the overview papers.

Harris Corner Detector V

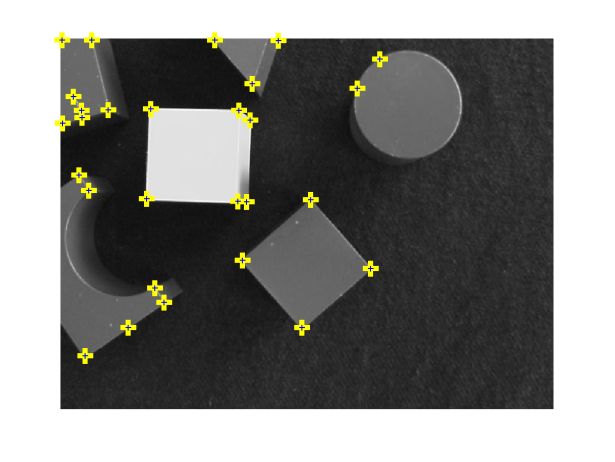
- The Harris corner detector has been very popular.
- Its advantages are insensitivity to 2D shift and rotation, to small illumination variations, to small viewpoint change, and its low computational requirements.
- On the other hand, it is not invariant to larger scale change, viewpoint changes and significant changes in contrast.
- Many more corner-like detectors exist, and the reader is referred to the overview papers.

Harris Corner Detector V

- The Harris corner detector has been very popular.
- Its advantages are insensitivity to 2D shift and rotation, to small illumination variations, to small viewpoint change, and its low computational requirements.
- On the other hand, it is not invariant to larger scale change, viewpoint changes and significant changes in contrast.
- Many more corner-like detectors exist, and the reader is referred to the overview papers.

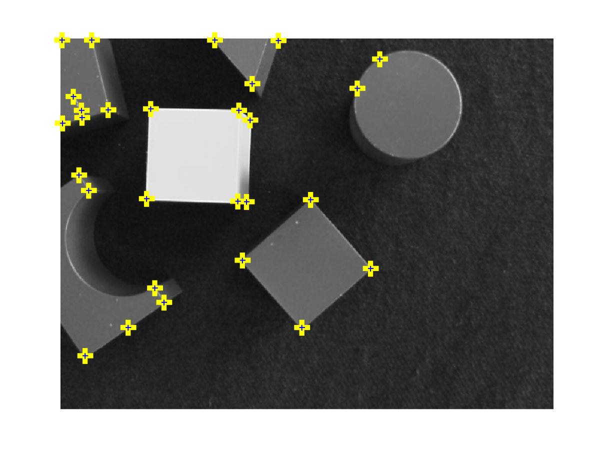
Harris Corner Detector: Example

- Corner detection example with Harris corner detector.
- The matlab script from **visionbook** is visionbook/05Preproc/harris_demo.m.



Harris Corner Detector: Example

- Corner detection example with Harris corner detector.
- The matlab script from **visionbook** is visionbook/05Preproc/harris_demo.m.



References I

Canny, J. (1986).
A computational approach to edge detection.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6):679 – 698.

Cervenka, V. and Charvat, K. (1987).
Survey of the image processing research applicable to the thematic mapping based on aerocosmic data.
Technical report, Geodetic and Carthographic Institute, Prague, Czechoslovakia.
Technical Report A 12-346-811.

Folland, G. B. and Sitaram, A. (1997).
The uncertainty principle: A mathematical survey.
Journal of Fourier Analysis and Applications, 3:207 – 238.

References II

- Haralic, R. M. and Shapiro, L. G. (1992).
  Computer and Robot Vision.
  Reading: Addision Wesley.
- Klette, R. and Zamperoni, P. (1996).
  Handbook of Image Processing Operators.
  John Wiley & Sons, Inc, New York.
- Marr, D. (1982).
  Vision.
  Freeman.
- Marr, D. and Hildreth, E. (1980).
  Theory of edge detection.
  Proceedings of the Royal Society of London. Series B, 207:187 – 217.

References III

Rongchun Zhao, R. (2000).
Introduction to Digital Image Processing (in Chinese).

Northwestern Polytechnical University Press, Xi'An, Shan Xi Province.