Computer Vision

Contents

Chapter 1

Introduction

1.1 What is Vision? ... Computer Vision?

Vision: One of the 6 human senses (vision, audition [hearing], haptics [touch] + proprioception [the kinesthetic sense], olfaction [smell], gustation [taste], equilibrioception [sense of balance])

For humans the primary sense of perception (wrt. information density preceived: 10M bits per second)

Allows perception of 3D scenes on the basis of 2D mappings (produced by the eyes, "images") of the scene

(Perception is the process of acquiring and interpreting sensory information.)

Example: *Finding the keys of your car on a cluttered table.*

Note/Problem: Though many aspects of the human visual system have been investigated it is not known how exactly visual percepts (=^ results of perception) are represented in the brain!

Computer Vision: Realization of visual perception capabilities (known from humans) within artificial systems (e.g. robots)

Example: *Recognizing a soccer ball on the playground amoung other soccer playing robots (e.g. Aibos)*

Note: Here, problem is greatly simplified by e.g. giving playground and ball well-defined colors and using (rather) controlled illumination.

Central components of all artificial vision systems are digital(!) computers.

⇒ Basis: Digital images / sets of images / image sequences

Computer Vision vs Image Processing: Image processing deals with aspects of manipulating and interpreting (digital) images in general

- Broader methodological basis

  (includes: restoration, compression, enhancement, ...; image syntesis = computer graphics)

  i.e. covers aspects *not relevant* from the perspective of (human) perception

- Focus mostly on lower level processing steps

  (i.e. the more interpretation/reasoning required the more likely a method will be called "computer vision" rather than "image processing")

1.2 Types of Images

- Most widely used image type: "natural" images i.e. resulting from a scene illuminated with visible light (=^ the type captured with standard cameras)

  Slides: TU Dortmund (Fig. 1), Taipei traffic (Fig. 2), ...

  Note: When understanding "computer vision" as implementing human perceptual capabilities on artificial systems *this* is the data we will be dealing with!

  For "image processing" in general many more image types are used.

- Infra-red images, i.e. resulting from radiation of hot surfaces (captured e.g. from satelites)

  Slide: North America (IR) (Fig. 3)

  Note: Images are rendered in so-called *pseudo-colors* for visual perception.

- Multi-spectral images, i.e. taking into account other parts of the electromagnetic spectrum (which visible light is part of)

  Slide: LandSat image of Amazonas rain-forest region (Fig. 4)

- X-ray images (from medical applications or from astronomy)

  Slide: X-ray image of human body parts (Fig. 5)

  Note: In medical applications X-ray "illumination" is absorbed by tissue!

  Also: Here X-ray imaging is an *active* method, as illumination is part of image aquisition and not "natural" i.e. passively observed.

  Slide: X-ray image of "Lockman Hole" (Fig. 6)

- Ultrasonic images (from medical applications, to some extent: robot navigation)

  Slide: Ultrasonic image of a human embryo (Fig. 7)

- Depth images (generated by e.g. laser range finder or so called "time-of-flight" cameras)

  Slide: Depth image of in-door scene (Fig. 8)

1.3 Application Areas

Computer Vision

- *Autonomous Robot Navigation*

  Example: Autonomous vehicles, e.g. “Stanley” (built at Stanford University, base: VW Touareg, won last DARPA Grand Challenge 2005, Sensors: 5 laser scanners and 1 color camera), “Highlander” (built at CMU, base: Hummer, short and long range LIDAR [= Laser Imaging Detection and Ranging, i.e. produces depth images] and RADAR)

  Slide: “Stanley” and “Highlander” with sensors (Fig. 9)

- *Surveillance* i.e. monitoring activity of persons, vehicles etc. in public spaces

  – Completely automatic solutions for reading of license plates
  – Surveillance of persons’ activities in specialized situations already possible (activity [e.g. attempts of theft] in a parking lot, attempts of bank robberies)

    Slide: Surveillance of parking lot at IBM (Fig. 10)

  – Surveillance of persons (individuals) on the horizon

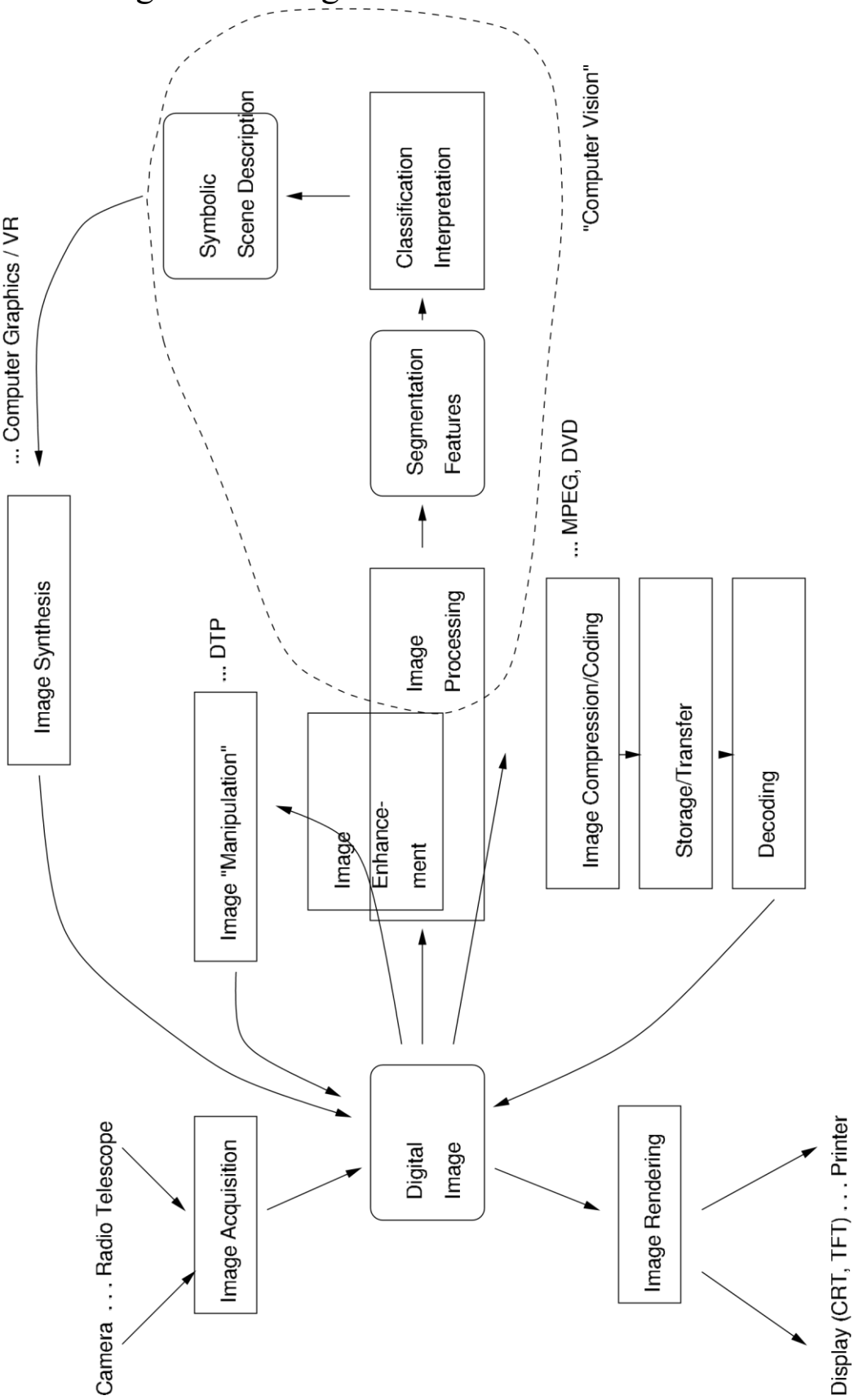
- *Face Recognition* (i.e. detection and/or identification [e.g. for biometric access control]

  Slide: Examples of face detection (Fig. 11, Fig. 12)

Applications more to be associated with “Image Processing”

- Identification of workpieces in industrial settings (on assembly line)
- Automatic reading of postal addresses (largely for machine printed texte, for handwritten approx. 50% “finalization” in the US as of 2000 [cf. IWFHR 7])
- Quality control in manufacturing (by automatic visual inspection)
- Monitoring/Analysis of biological/physical/chemical/... processes (e.g. groth of cell populations, finding traces of elementary particles)
- *And not to forget about* Automatically guided weapons (e.g. cruise missiles use image of goal, used for correlation search in final phase of flight)

1.4 The Image Processing / Machine Vision Universe



1.5 Why is Computer Vision Difficult?

- Humans use contextual knowledge and knowledge about the world

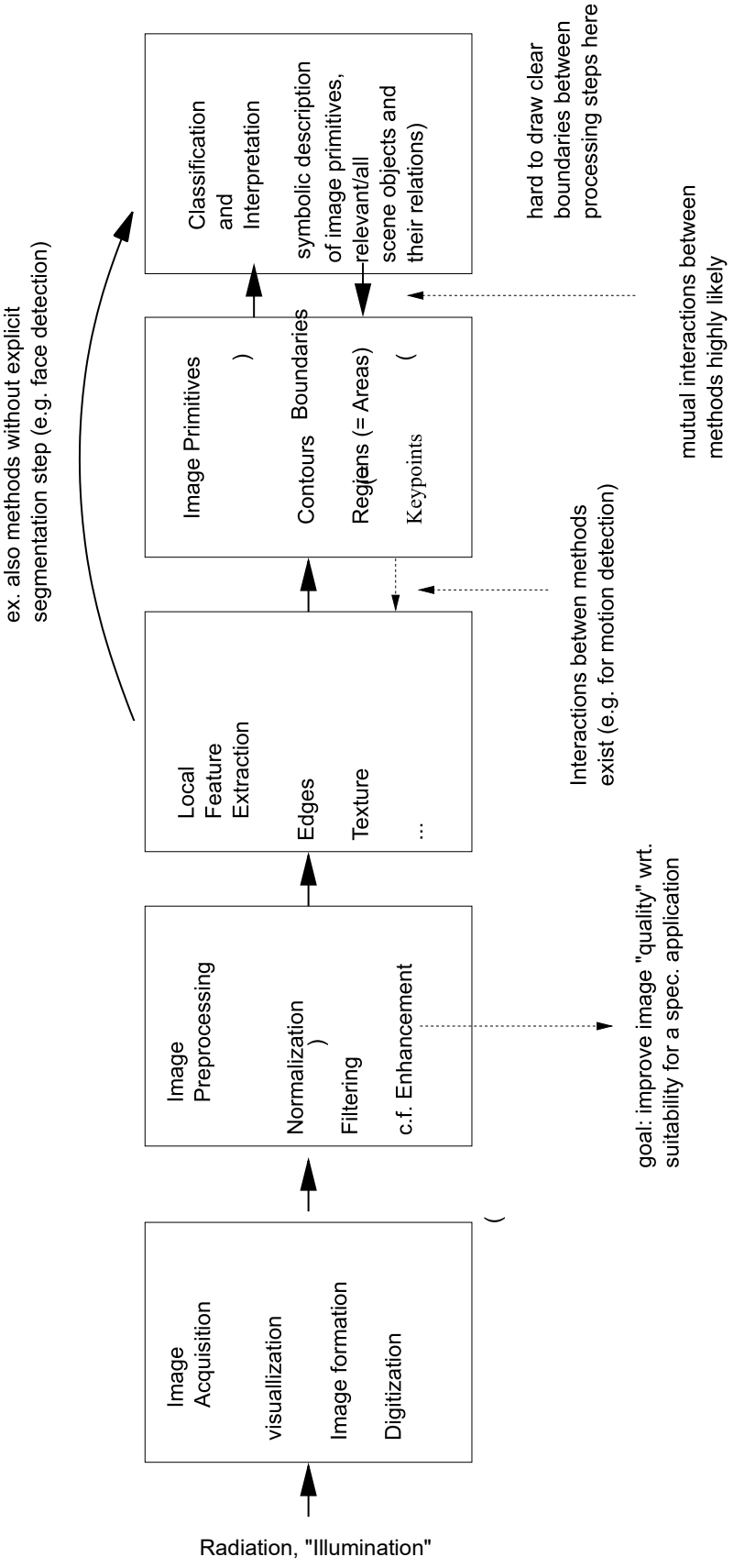
Slide: “Lampenratsel” from J¨ahne [J¨ah02, p. 17] (Fig. 13¨ *Welches Objekt ist keine Lampe?*)

- Human visual system is highly specialized, e.g.
  - Many pre-attentional effects (e.g. “Gestalt laws”)

    Slide: Gestalt laws (Fig. 14)
  - “Important” visual cues are processed especially efficiently / reliably (e.g. human faces, motion)
  - Many non-linear effects, e.g. in perception of image intensity Slide: Election recount (Fig. 15) • When trying to rule out these effects ...

Slide: Digital face image in “numerical domain” (Fig. 16

1.6 Typical Architecture of a Machine Vision System



Chapter 2

Fundamentals of Machine Vision

2.1 Image Acquisition

Goal: Mapping/Projection of a 3-dimensional scene onto a 2-dimensional (digital) image (in the memory of a machine vision system) Procesing Steps:

a) *Visualization*

by means of physical processes of ...

- Reflection (e.g. “ordinary” photographic images)
- Absorption (e.g. medical X-ray imaging)
- Emission (e.g. infra-red imaging, astronomy (X-ray, radio)

... of radiation by objects / scenes.

b) *Image Formation*

An imaging system (e.g. a camera, electromagnetic field) projects the radiation originating from the 3-dimensional scene onto a 2-dimensional image plane

c) *Digitization* (= Sampling + Quantization)

The continuous image formed by the radiation incident on the image plane is sampled in a discrete grid (i.e. incident radiation is measured [what? ... usually intensity, more later] at discrete points on the image plane (values measured are stored).

Note: Measurement values are still continuous!

Sampled intensity values are additionally mapped onto a finite set of discrete values – i.e. quantized.

Digitization in the spatial domain (image plane) – i.e. sampling – and in the amplitude domain – i.e. quantization – produces a digital image.

2.1.1 Visualization

Most “widely used” / “readily available” type of radiation for the visualization of objects / scenes: Electromagnetic radiation.

Slide: Overview of the electromagnetic spectrum (Fig. 18)Note: Similar graphic also in [Gon02]

Most important part of the electromagnetic spectrum for human and computer vision is visible light with wave lengths ranging between approx. 400 to 800 nm.

⇒ Will consider only this type of radiation further!

Remarks:

- The el-mag. spectrum is *continuous*, i.e. arbitrary wave lengths can occur (ignoring quantum effects!)
- Real radiation is in general not homogeneous but consists of a mixture of different wave lengths ⇒ therefore the following can be measured:
- Intensity of the radiation (in a given range of wave lengths; for visible light: brightness) • Composition of the total radiation from parts with different wavelengths (for visible light: "color" [Beware: Color is not an objective but a subjective measure!])

Intensity

For so-called "grey level/scale images" only the intensity is measured which is reflected from an object / a scene.

The observed (light) intensity results from:

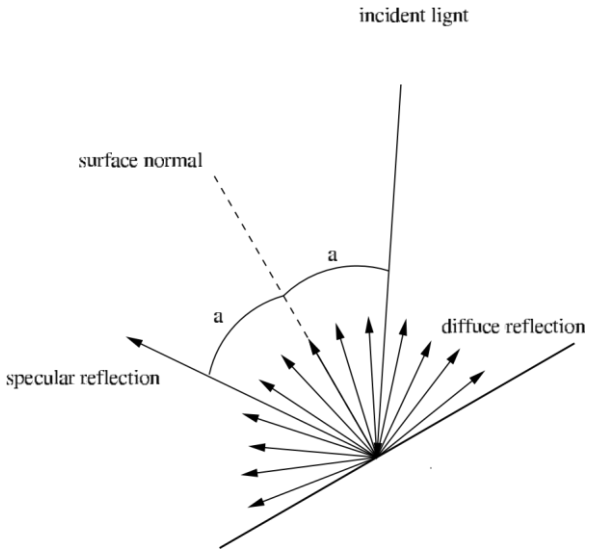
- the intensity of the illumination sources
- the spatial configuration (distance, orientation) of illumination sources and (reflective) object surfaces
  - intensity is inversely proportional to the squared distance (object to source)
  - illumination effect is maximum if direction of incident light is prependicular to surface
- the position of the observer (e.g. camera)
- the reflectivity of object surfaces involved.

Surface reflection is composed of

– *specular* (i.e. mirror-like) and

– *diffuse* (i.e. reflected in all directions) part

Proportions of specular and diffuse reflection vary for different surfaces.



- the absorption of incident light by object surfaces

  (almost complete for visible light: black surfaces)

Note: Observed intensity results from interplay of *all* factors (Is in general dependent on illumination sources and *complete* scene)

⇒ Inverse problem in computer graphics: Generating realistic illumination for scenes.

Spectral Composition • Spectral composition of illumination (here: from el-mag. spectrum, especially light) is primarily dependent on

– the illumination sources and

– the absorption by object surfaces (reduces intensity of specific parts of the spectrum)

- The (varying) spectral composition of visible light (i.e. the *physical* quantity) produces the sensation of color (i.e. subjective) in the human visual system

Note: There is no one-to-one mapping between spectral composition of light and perceived color!

More about color later!

2.1.2 Image Formation: The Role of Cameras

after [For03, Chap. 1]

In image formation an imaging system (mostly: an optical system) projects an image of a 3-dimensional scene or object onto a 2-dim. image plane.

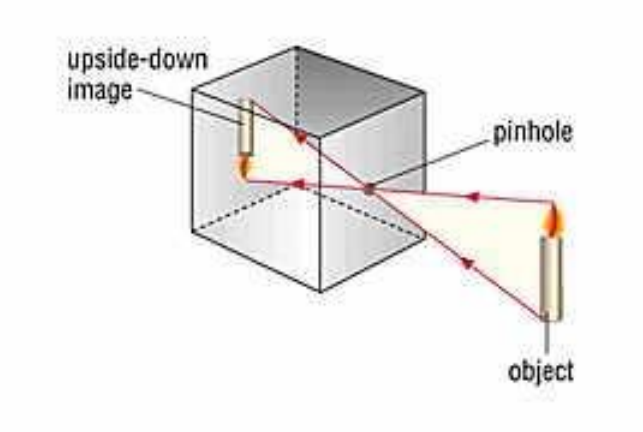
A Simple Imaging Device

Experiment: Take a box, prick a small hole in to one of its sides (with a pin), replace the side opposite to the hole with a translucent plate.

Hold the box in front of you in a dimly lit room, with the pinhole facing a candle (i.e. a light source) ...

What will you see? — An inverted image of the candle

⇒ Camera Obscura (invented in 16th century)



The Pinhole Camera Model

Idealized optical imaging system / simplest imaging system imaginable (idealization of camera obscura)

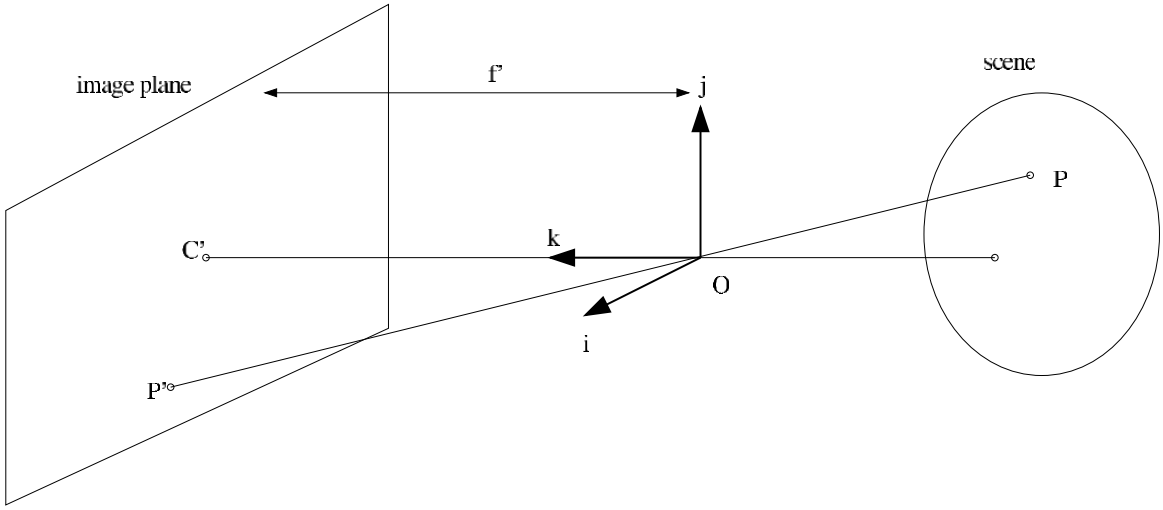
Principle: Infinitely small pinhole ensures that *exactly one* light ray originating from a point in the scene falls onto a *corresponding* point in the image plane.

I.e. exactly one light ray passes through each point in the image plane, the pinhole, and some scene point.

Note: Despite its simplicity the pinhole model often provides an acceptable approximation of the imaging process!

- Pinhole camera defines *central perspective projection* model.
- Perspective projection creates *inverted* images ⇒ More convenient to consider *virtual* image on plane in front of the pinhole.
- Obvious effect of perspective projection: Apparent size of objects is dependent on their distance.

Geometry of pinhole projection:



after [For03, Fig. 1.4, p. 6]

- Coordinate System $(O,\mathrm{i},\mathrm{j},\mathrm{k})$ attached to the pinhole of the camera (Origin O coincides with pinhole)
- Image plane is located at positive distance f' from pinhole
- Line perpendicular to image plane and passing through O is called *optical axis*, point C' *image center*

Let P denote some scene point with coordinates (x,y,z) and P' its image at (x',y',z').

As P' lies in the image plane: $z' = f'$

As P, O, and P' are colinear: $\tilde{OP'} = \lambda \tilde{OP}$ for some λ

Consequently, we obtain the following relations between coordinates of scene and image points:

$$\begin{pmatrix} x' \\ y' \\ f' \end{pmatrix} = \lambda \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad \Leftrightarrow \quad \lambda = \frac{x'}{x} = \frac{y'}{y} = \frac{f'}{z}$$

and finally: $x' = f'\frac{x}{z}$ and $y' = f'\frac{y}{z}$

Note: Perspective projection model can be further simplified by assuming that scene depth is small with respect to scene distance ⇒ scene points have approximately identical distance $z = z_0$ ⇒ constant magnification $m = -\frac{f'}{z_0}$ (weak perspective projection)

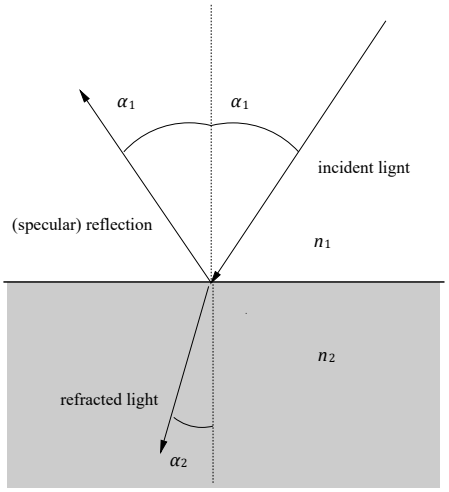
Cameras With Lenses

Disadvantage of pinhole principle: Not enough light gathered from the scene (only single ray per image point!) ⇒ Use lens to gather more light from scene *and* keep image in focus

Note: As real pinholes have finite size the image plane is illuminated by a cone of light rays: The larger the hole, the wider the cone ⇒ image gets more and more blurred.

Behaviour of lenses defined mainly by geometric optics (ignoring physical effects of e.g. interference, diffraction, etc.):

- In homogeneous media light (rays) travels in straight lines
- When light rays are *reflected* from surfaces, this ray, the reflection, and the surface normal are coplanar; the angles between normal and rays are complementary
- When passing from one medium to another light rays are *refracted* (i.e. change direction); the original ray, the refracted one and the normal to the interface are coplanar; the change of direction is related to the *indexes of refraction* according to:



$$n_1 \sin(\alpha_1) = n_2 \sin(\alpha_2)$$

⇒ Consider refraction and ignore reflection (i.e. won't consider optical systems which include mirrors as, e.g., telescopes)

Assumptions:

- Angles between light rays passing through a lens and the interface normal (normal to the refractive surface) are small.
- Lenses are rotationally symmetric around the *optical axis*.
- Lenses have a circular boundary.

⇒ Paraxial geometric optics

Slide: Illustration of paraxial refraction (Fig. 19)

- For α_i, β_i and γ the following hold:

$$\alpha_1 = \gamma + \beta_1 \qquad \alpha_2 = \gamma - \beta_2$$

- As angles are approximately equal to their sines or tangents:

$$\sin \gamma R = h \Rightarrow \gamma \approx \frac{h}{R}$$

(Analogously the following can be derived:

$$\tan \beta_i d_i = h \Rightarrow \beta_i \approx \frac{h}{d_i} \quad \text{for } i = 1,2$$

)

- Inserting into the law of refraction (assuming small angles) yields:

$$n_1 \alpha_1 \approx n_2 \alpha_2 \leftrightarrow \frac{n_1}{d_1} + \frac{n_2}{d_2} = \frac{n_2 - n_1}{R}$$

Note: Relationship between d_1 and d_2 depends only on R and the indexes of refraction n_1 and n_2 *but not* on β_i.

Assumptions for (Thin) Lenses:

- Lens has two spherical surfaces (both with radius R)
- Lens has index of refraction n, surrounding medium of 1 (vacuum or approximately air).
- Lens is *thin* i.e. ray entering is refracted at the frontal interface and *immediately* at the backward one.

Slide: Illustration of refraction with a thin lens (Fig. 20)

Remarks:

- Scene point P is projected (in focus!) onto image point P'.
- Light ray (PO) through center of lens is not refracted.
- F (and F') is *focal point* of the lens at distance f from O.
- The relations between object and image distance $-z/z'$, size y/y' and focal f length of the lens are defined by the *thin lens equation*:

$$\frac{1}{z'} - \frac{1}{z} = \frac{1}{f} \quad \text{where} \quad f = \frac{R}{2(n-1)}$$

also: $\frac{y}{y'} = \frac{z}{z'}$

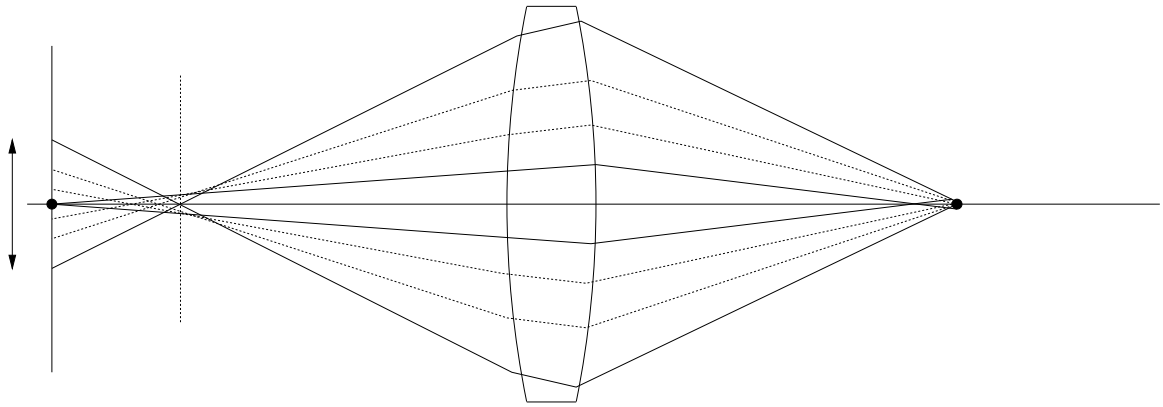
- If z approaches infinity, image points are in focus at distance f.

Note: In practice objects within some limited range of distances (the so-called *depth of focus*) are in acceptable focus.

Outside the focal plane images of scene points are projected onto a circle of finite size (i.e. blurred).

Real Lenses suffer from a number of *aberrations*:

- *Spherical* aberration:



Light rays from P form a circle of confusion in the image plane. (Circle with minimum diameter = *circle of least confusion*, in general not located in P')

- *Distortion*: pincushion and barrel distortion.

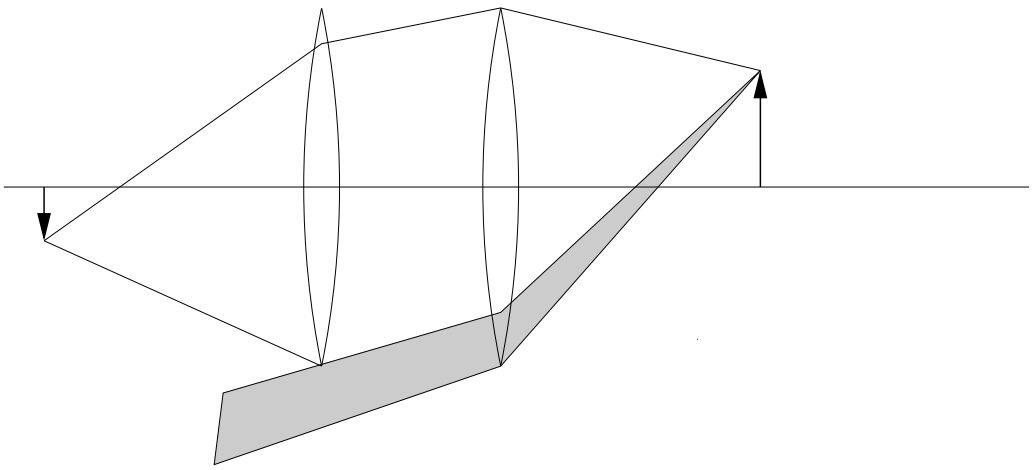
- *Chromatic aberration* resulting from the index of refraction beeing dependent on wavelength.

Can be shown using a prism.

Real optical systems use multiple lenses:

- Multi-lens arrangements can be used to minimize aberrations.

- However, multi-lens systems suffer from *vignetting* effects as light rays are blocked by appertures of different lenses.



2.2 Structure of the Human Eye

cf. [For03, Sec. 1.3], [Gon02, Sec. 2.1]

In general, the human eye is an optical imaging system (i.e. a "camera"), which projects a scaled down, inverted image of the scene onto the background of the eye (= retina).

Slide: The human eye as an imaging system (Fig. 21)

Slide: Schematic structure of the human eye (Fig. 22)

Remarks:

- The *cornea* serves as a protective shield.
- The *iris* relizes the aperture, can be adjusted in size by the *cilary muscle* (synchronously for both eyes).
- The lens is flexible and able to adjust its refractive power via the tension of the *cilary fibers*.
  - Far objects: Lens is relatively flat (from 3 m and above)
  - Near objects: Lens is rather thick.
- The *retina* contains 2 types of photoreceptors:
  - *Rods*: very sensitive to light (approx. 120M)
  - *Cones*: sensitive to color in 3 sub-types (roughly to "red", "green", and "blue"; approx. 6M)
- Cones mostly found in the area of the *fovea*

  Slide: Distribution of rods and cones on the retina (Fig. 23)
- Muscles rotate the eye such that objects of interest are projected onto the fovea.
- The *blind spot*, i.e. the path of the optical nerve, contains no photoreceptors (Note: Nerve connections of photoreceptors lie *before* the retina).

2.3 Color

cf. [Gon02, Sec. 6.1, 6.2]

2.3.1 Human Color Perception

... is based on 3 types of color sensitive photoreceptors in the retina (*cones*), which respond differently to the wavelengths of the visible spectrum.

Slide: Response of cones to different wavelengths (Fig. 24)

Remarks:

- Excitation of receptors is maximum for “red”, “green”, or “blue” light, respectively. However, cones respond to complete visible spectrum!
- Intensity of response (=^ sensitivity of receptors) is highest for “green” light and decreases towards the boundaries of the visible spectrum.
- Homogeneous light of a single wavelength creates “pure” color sensation, which, however, can also be produced by combination of appropriate wavelengths.

Color perception of the human eye can be represented by the CIE-diagram based on the following definitions:

Let X, Y, Z be the absolute responses of the color sensors

Let the total intensity be: $X + Y + Z = I$

The relative color portions are then given by:

$$x = \frac{X}{I} = \frac{X}{X+Y+Z} \quad y = \frac{Y}{I} \quad z = \frac{Z}{I}$$

⇒ $x + y + z = 1$ or $z = 1 - x - y$ (i.e. only 2 independent quantities)

⇒ colors can be represented in x/y-plane

Slide: Chromaticity diagram (Fig. 25)Note: Similar graphic also in [Gon02, Chap. 6]

Remarks:

- Pure spectral colors are found on the perimeter of the “tongue-shaped” color space. Those colors have maximum saturation.
- Colors in the inner area result from a mixture of pure colors. • Mixing two color components can potentially create all colors lying on the straight line beween the two points in the diagram (mixing of 3 colors ⇒ triangle).

2.3.2 Color Models

The RGB Color Model

... motivated by primary sensitivity of human color receptors for (roughly) red, green, and blue light
⇒ RGB

Required: Specification of “base colors” for red, green, and blue (different possibilities!)

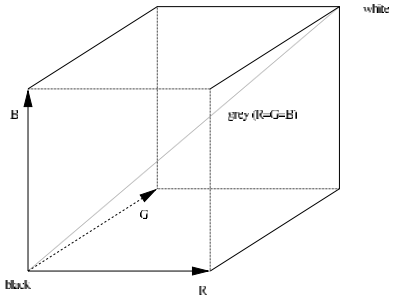
For maximisation of colors that can be represented ([Bal82, p. 32]):

$\lambda_1 = 410\text{nm}$ (blue)
$\lambda_2 = 530\text{nm}$ (breen)
$\lambda_3 = 650\text{nm}$ (red)

By superpositions of those wavelengths most *but not all* colors can be generated (=ˆ triangle in CIE diagram).

Note: Mixing of colors (i.e. superposition) in “wave length space” is *additive* as opposed to mixing of color pigments for printing (subtractive color mixing).

When normalizing the maximum intensity in the color channels to 1 (minimum is 0, as no “negative” frequency components possible) one obtains the following RGB color cube:



Drawback: Color information can not be separated from intensity information

⇒ alternative color models of the form: intensity + “color”

The HSI Color Model

H = Hue (“color tone”), roughly proportional to the average wagelength of a mixture of primary colors

S = Saturation, represents the “missing” of a white component within a mixture of colors

I = Intensity

The HSI model defines HS color planes (either of circular or hexagonal shape) perpendicular to the intensity axis of the model ⇒ H, S, I are usually given in cylindrical coordinates.

Slide: HSI color space (Fig. 26)

The YUV Color Model

cf. Dictionary by LaborLawTalk

... used for PAL television and slightly modified (e.g. YCbCr with Cb, Cr beeing scaled versions of U, V) in computer component video.

Y = Luminance, i.e. the brightness component

U, V = Chrominance, i.e. "color" components
YUV can be computed from (normalized) RGB as follows:

$$Y = +0.299R + 0.587G + 0.114B \; U = +0.492(B - Y)$$

$$V = +0.877(R - Y)$$

Remarks:

- Luminance Y is weighted sum of R, G, B intensities with dominant weight for "green".
- U and V are (appropriately scaled) differences to original B ("blue") and R ("red") compoments.

Subtractive Color Mixing: The CMY[K] Model

... is used for producing colors in print, i.e. by using color pigments and *not* illumination sources ⇒ "subtractive" color mixing

Color pigments absorb certain spectral components of incident light and reflect only some remaining wavelength components (e.g. "red" pigment absorbs "green" and "blue" components and reflect only "red" part of spectrum).

Prinary colors for "subtractive" color mixing (better: mixing of coloring substances [i.e. pigments]): CMY model

C = cyan (i.e. bluish green) =^ $W - R$

M = magenta (≈ purple) =^ $W - G$

Y = cyan (i.e. bluish green) =^ $W - B$

Primary colors "subtract" one of the primary colors of the additive RGB model from white light (W). Mixtures can, therefore, be calculated with respect to the RGB model:

$$\begin{aligned} C \oplus M &= (W - R) + (W - G) = \\ &= \underbrace{W + W} - R - G = \\ &= \quad W \quad - R - G = \\ &= (R + G + B) - R - G = B \end{aligned}$$

Note: For the purpose of producing printed documents the CMY model is often realized as CMYK with an additional K for black pigment, as $C \oplus M \oplus Y$ usually does not produce acceptable black for e.g. typesetting text.

Slide: Additive vs. subtractive color mixing (Fig. 27)

2.4 Digitization

... comprises two processing steps, namely *sampling* of an image (in general: some analog signal) in the spatial (or time) domain and *quantization* of the samples (=^ measurements).

2.4.1 Sampling

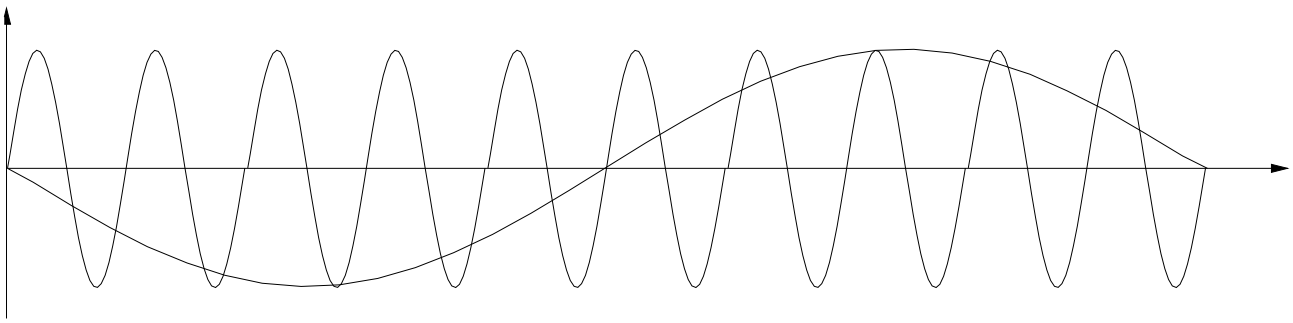
(for images) ... is the measurement of the "content" of an analog image (e.g. the intensity) at the discrete points of a 2-dimensional grid.

The topology of the grid is defined by the sensor arrangements of the imaging device used (Sect. 2.5).

Most common topology: regular 2-dim array.

Problem: Process of sampling can already lead to loss of information!

Example: Periodical signal is sampled at $\frac{9}{10}$ its wavelength ⇒ results in aliased wavelength 10 times longer than original!



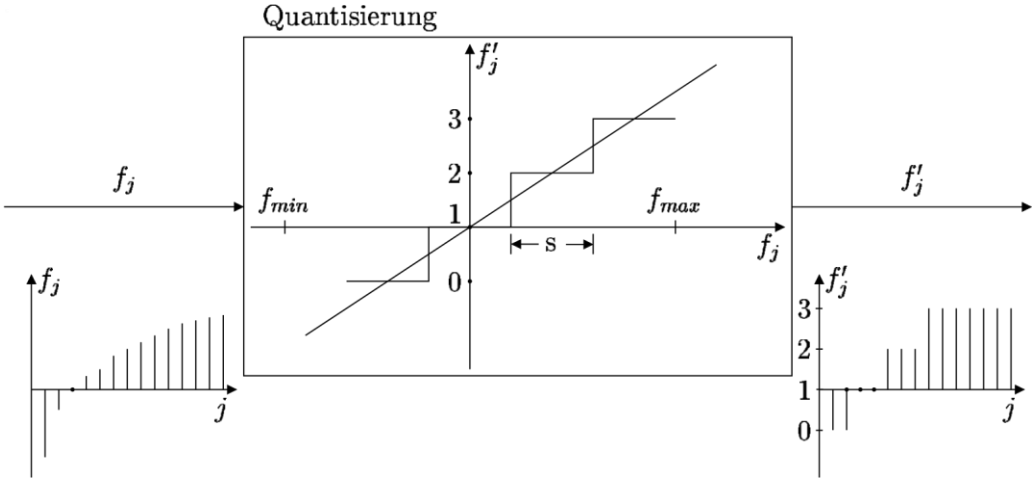
(after [Jah02, Chap. 2])¨

Theoretical Result: The so-called *sampling theorem* states that sampling a continuous signal can be achieved without loss of information, if sampling frequency is at least twice as high as the highest frequency component present in the signal.

2.4.2 Quantization

... is the mapping of analog/continuous samples/measurements onto a *discrete* (especially *finite*) set of values. Q

$$[f\text{min}, f\text{max}] \longrightarrow \{b_0, b_1, b_2, \ldots b_{L-1}\}$$



Quantisation of continuous samples (after nach [Nie03, S. 68])

Note: Quantization *necessarily* introduces errors into the digitization process ⇒ exact reconstruction of original signal no longer possible.

Remarks:

- Stored are not discrete quantities b_i but their indices i. • Behaviour of quantisation process is completely specified by defining the quantized values to be used *and* the associated intervals on the continuous side that will be mapped upon them.

  Note: The characteristic curve of the quantization need not be linear (nonlinear characteristic useful for, e.g., quantization of X-ray images due to nonlinearity of absorption by tissue).

- For digital processing $L = 2^B$ quantization steps appropriate.

  – For grey level images $B = 8$ bits (i.e. $L = 2^8 = 256$ discrete intensities) yield gut subjective reproduction of the image.

  – For color images $B = 8$ bits per color plane (R, G, B).

2.4.3 Representation of Digital Images

A grid point (i.e. measurement point) in a digital image is called *pixel* (or *pel* = picture element). Its position in the $M \times N$ image array is specified by the row and column index $x = 0,1,\ldots M-1$ and $y = 0,1,2,\ldots N-1$ (Beware: Indices do not correspond to actual spatial positions!).

Note: When representing digital images mostly the so-called *upper-left* coordinate convention is used, i.e. the origin of the image matrix lies at the upper left corner of the image.

Slide: Coordinate convention used with digital images (Fig. 28)

Note: Upper-left convention defines no *right-handed* coordinate system.

⇒ computing angles affected!

For segmentation (e.g. separating [foreground] objects from background) the definition of pixel *neighborhood* on the image matrix/grid is important. Definition of connectivity is affected by this choice.

Slide: Different definitions of pixel neighborhoods and effect on connectivity (Fig. 29)

Note: Problem with neighborhood variants and resluting connectivity can be resolved using a hexagonal image/sampling grid (found in some digital cameras).

2.5 Imaging Devices

Most widely used type of digital camera: CCD camera (CCD = charge coupled device).

- A CCD sensor uses a rectangular grid of photosensitive elements (of some finite size).
- Incident light (photons) cause a electrical charge to build up in each element
- Charges of individual sensor elements are "read out" – i.e. moved out of the sensor array – by using charge coupling (a row at a time).

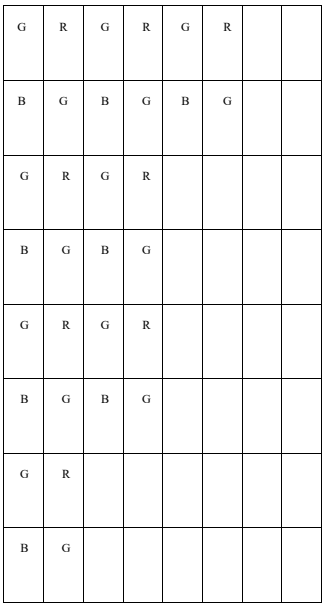
Slide: Structure of a CCD device (Fig. 30)

Remarks:

- In order to build up charge in sensor elements some time of exposure to incident light has to be allowed.
- E.g. for video applications image contents are read at some fixed rate (25 Hz for PAL, actually reading odd and even rows separately at a rate of 50 Hz ⇒ motion can cause distortions).
- Capturing "color" requires sensors to be made sensitive to different parts of the electromagnitic spectrum:
  - Using color filters in front of a single CCD chip and reading R, G, B images sequentially

– Using a beam splitter and color filters with 3 different CCD chips for R, G, B

– Using a pattern (mosaic) of sensor elements sensitive to R, G, B (e.g. Bayer pattern [invented by Dr. Bryce E. Bayer of Eastman Kodak]).



Bayer pattern of color filters

Chapter 3

Preprocessing

Goal: "Preparation" of images such that better results are achieved in subsequent processing steps (e.g. segmentation).

Preprocessed images are better suited for future processing.

⇒ image *enhancement*

Note: There is no such thing as *the* preprocessing operation! (Techniques are higly application dependent)

General Principle/Idea: Reduce unwanted *variability* in images/data

No in-depth treatment of preprocessing techniques! Only principle idea and typical example methods!

3.1 Normalization

Problem: Images/objects in images usually have *parameters* that vary within certain intervals (e.g. size, position, intensity, ...). However, results of image analysis should be *independent* of this variation.

Goal: Transform images such that parameters are mapped onto normalized values (or some appropriate approximation).

3.1.1 Intensity

Interpretation of image content is usually independent of (local) image intensity ⇒ normalization! Slide:

Basic image types: dark, light, low contrast, high contrast (Fig. 31)

- Normalization to Standard interval $[0,a]$, e.g. $[0,255]$:

  Transform original grey valye f_{ij} into normalized value h_{ij} according to:

$$h_{ij} = \frac{a(f_{ij} - f_{\min})}{f_{\max} - f_{\min}} \quad \text{where} \qquad f_{\max} = \max f_{ij},\ f_{\min} \text{ analogously } ij$$

- Normalization to zero mean and unit variance:

  Let μ be the mean image intensity and σ^2 the associated variance:

$$\mu = \frac{1}{\sum_{ij} 1} \sum_{ij} f_{ij} \qquad \sigma^2 = \frac{1}{\sum_{ij} 1} \sum_{ij} (f_{ij} - \mu)^2$$

  Normalized intensities h_{ij} will have $\mu' = 0$ and $\sigma'^2 = 1$:

$$h_{ij} = \frac{f_{ij} - \mu}{\sigma}$$

  Remark: Resulting intensities are no longer integers!

Note: Global normalization of intensity usually simpler but less effective than more complicated *local* normalization!

3.1.2 Histogram Equalization

... most "popular" normalization method based on the grey level histogram of an image given by:

$h(g)$ = # of grey values g in imagewhere $= \sum_{ij} \delta(f_{ij} - g)$ $\qquad \delta(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases}$

From the histogram the (estimated) probability of a pixel having a certain grey level g can be derived:

$$p(g) = \frac{1}{\sum_{ij} 1} h(g) = \frac{1}{MN} h(g) \qquad \text{for } M \times N \text{ images}$$

Goal: Use complete dynamic range of available grey levels within an image in order to resolve small but frequent differences better.

Idea: Grey level intervals of high density in the histogram should be "stretched" and those with low density should be "compressed".

⇒ In general for the discrete case the number of grey levels is reduced by "compression".

Method Let the cumulative distribution function of the grey values be:

$$H(f) = \sum_{g=0}^{f} p(g) = \sum_{g=0}^{f} \frac{1}{MN} h(g)$$

(directly obtainable from the grey level histogram)

The transform defined by

$$T(f) = \lfloor L \cdot H(f) \rfloor = \lfloor L \cdot \frac{1}{MN} \sum_{g=0}^{f} h(g) \rfloor$$

(where L is the number of grey levels available and $\lfloor x \rfloor$ represents the largest integral number that is equal or less than x) achieves an *approximate* equalization of the grey level histograms, i.e. the histogram of the transformed grey levels is *approximately* equal to a uniform distribution.

Note: In the discrete case the equalized histogram will in general *not* be equal to a uniform distribution. This can be achieved for the contiuous case only!

Note: To some extent also a normalization of intensity is achieved implicitly.

Slide: Examples of histogram equalization (Fig. 32)

3.1.3 Some Other Techniques

Size, Position, Orientation, ...

Slide: Example of position/orientation normalization (Fig. 33)

Slide: Example normalisation of character slant (Fig. 34)

3.2 Filtering

Image *filtering* comprises image transforms that work on a certain *neighborhood* of pixels (usually rectangular or quadratic area centered at the pixel in question) for deriving a new grey value.

Simple formalization of filtering:

- Move a *window* (which has the size of the neighborhood considered) from point to point in the image
- Calculate the *response* of the filter at each point by applying some operation to the pixel grey values within the window.

⇒ Filters are implemented using *local* image operations, realize *locally defined* image transforms!

3.2.1 Linear Filters and Convolution

An expecially relevant case of filters are *linear* (the transform satisfies $T\{af+bg\} = aT\{f\}+bT\{g\}$), *shift-invariant* (transform is independent of pixel posision) filters.

Linear filters can be realized as follows:

- Define a filter *mask* with the size of the neighborhood and filter coefficients/weights $w(s,t)$ assigned to each point of the mask.

  Slide: Principle of filtering using masks (Fig. 35)

- Move the mask over the image from point to point

- Calculate the filter *response* by a weighted sum of pixel grey values and mask coefficients.

  For a 3 × 3 mask the filter response is given by: $g(x,y)$ =

  $w(-1,-1)f(x-1,y-1)+w(-1,0)f(x-1,y)+\ldots+w(0,0)f(x,y)+\ldots w(1,1)f(x+1,y+1)$

  (Note: Coefficient $w(0,0)$ coincides with $f(x,y)$, i.e. mask is centered at position (x,y))

In general the result of applying a linear filter $w()$ of size $m \times n$ (with $m = 2a + 1$ and $n = 2b + 1$) to an image $f()$ of $N \times M$ pixels is given by:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

Note: This formulation is equivalent to computing the *cross-correlation* between the mask and the image, which is similar to the concept of *convolution*:

$$h = f * g \Leftrightarrow h(x,y) = \sum_s \sum_t f(s,t) g(x-s, y-t) = \sum_s \sum_t f(x-s, y-t) g(s,t)$$

For convolution either the signal $f()$ or the mask $w()$ need to be mirrored along the x- and y-Axis, which is unproblematic as masks are frequently symmetric.

Example: Image Smoothing with Linear Filters

- *Image Averaging*

  Use $m \times n$ window with weights $w(s,t) = \frac{1}{mn}$

  Note: Undesirable effects on image content in *frequency domain*!

- *Gaussian Smoothing*

  Required: Filter coefficients for *discrete* 2D approximation of Gaussian
  discrete 1D-Gaussian can be obtained from the rows in *Pascal's Triangle*:

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

3 × 3 Gaussian filter mask obtained from

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times (1\ 2\ 1) = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Note: In order not to amplify the image content masks are usually normalized by the sum of their coefficients (for Gaussian: $\frac{1}{16}$).

3.2.2 Filtering in the Frequency Domain: Overview

Motivations:

- Better understanding of effects of filtering on images (only with more mathematical details!)
- Increased efficiency for computation of responses for large linear filters

Basis: Discrete Fourier Transform (DFT)

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-i2\pi(\frac{ux}{M} + \frac{vy}{N})}$$

- Computes a frequency representation of a signal/image (Note: Both *f* and its transform *F* are complex!)
- Input signals are roughly approximated by sine and cosine functions of different frequencies e^{-ix} and amplitudes (Note: $e = \cos x + i \sin x$).
- Assumes discrete *periodic* input ⇒ finite images are treated as if repeated periodically!

Slide: Examples of 1D-DFT (Fig. 36)

Slide: Example of 2D-DFT (Fig. 37)

Note: Example shows that averaging filter (rectangular shape spatial response) is no good smooting operation as effects are *infinite* in frequency domain!

⇒ Betters solution: Gaussian, as it is form-invariant under Fourier transform!

Important property of the Fourier transform:

Convolution in the *spatial* domain is converted to *multiplication* in the *frequency* domain and vice versa: $f = g * h \Leftrightarrow F = G \cdot H$

⇒ Convolution operations can be computed more efficiently in frequency domain! (Note: For filtering additionally computation of forward and backward transform necessary ⇒ beneficial only for large filter masks!)

3.2.3 Rank-Order Filters

(also known as *order-statistics* filters)

... in general *non-linear* spatial filters whose response is based on an ordering (*ranking*) of intensity values within the neighborhood considered.

Given a neighborhood $V(x,y) = \{f(x+s,y+t) | -a \le s \le a \wedge -b \le t \le b\}$

Order pixel values (i.e. intensities) as follows:

$$R(x,y) = \{r_1 \le r_2 \le \ldots \le r_K | r_k \in V(x,y) \wedge K = |V(x,y)|\}$$

A rank-order (filter) operation is then defined as a function of $R(x,y)$:

$$h(x,y) = \varphi\{R(x,y)\}$$

Simple Examples:

$V(x,y) =$

0	2	4	6	
1	4	5	6	
1	3	6	5	

$\rightarrow R(x+1, y) = \{2, 3, 4, 4, 5, 5, 6, 6, 6\}$

$\rightarrow R(x,y) = \{0, 1, 1, 2, 3, 4, 4, 5, 6\}$

$r_{(K+1)/2} = 3$

$V(x,y)$ =

0	0	1	
0	1	1	
0	1	1	

$\rightarrow R(x,y) = \{\underbrace{0, 0, 0, 0}_{\{z_k\}}, 1, 1, 1, 1, 1\}$

$r_{(K+1)/2} = 1$

The following "well known" rank-order / order-statistics filters/operations can be defined:

$h(x,y) = r_1$ Erosion $h(x,y) = r_K$ Dilation

$h(x,y) = r_{(K+1)/2}$ Median

$h(x,y) = \begin{cases} r_K - r_1 & \text{Edge detection (aka "morphological edge")} \\ r_K - f(x,y) & \text{Edge sharpening} \end{cases}$ $h(x,y) = r_1$ if $f(x,y) - r_1 < r_K$ otherwise

Slide: Example of erosion (combined with dilation) (Fig. 38)

Slide: Example of dilation (combined with erosion) (Fig. 39)

Remarks:

- Erosion + Dilation = *Opening*
- Dilation + Erosion = *Closing*
- Median as a smooting operation preserves contrast/edges (in contrast to e.g. averaging) but removes "salt-and-pepper" noise (i.e. small "errors").
- Computation of order-statistics requires *sorting* of pixel intensities in a large number of neighhborhoods ⇒ efficient implementations required!

Slides: Example images for averaging, median (Fig. 40), erosion/dilation, morphological edge (Fig. 41), and opening/closing (Fig. 42)

Chapter 4

Local Image Features

... features (numerical values!) of individual pixels or pixel neighborhoods that are relevant for image segmentation and/or interpretation.

In the simplest case: (smoothed) image intensity/color (i.e. without noise) However,

more complicated local features exist.

4.1 Edges

cf. [Gon02, Sec. 10.1]

For segmentation *boundaries* between regions of images are especially relevant, which become manifest by *discontinuities* in the image.

4.1.1 Detection of Local Discontinuities

Local discontinuities =ˆ (usually) difference in image intensity/grey level Remarks:

- Definition of "difference" more complicated for color images (i.e. multi-channel images)
- Grey level may be substituted by some local image feature.

Note: Usually pixels where relevant grey level differences can be observed are called *edge* pixels or *edge* elements (*edgels*) vs *contours*, i.e. boundaries of regions.

Discrete Differentiation

Differentiation in the discrete case =ˆ computing local differences (approximation of continuous differentiation)

Slides: Examples of edge types (ideal, ramp) (Fig. 43), and behaviour of derivatives (Fig. 44)

⇒ Computing local differences (by some method) for every pixel yield *new*, i.e. an *edge image*!

2 commonly used methods:

- 1st (discrete) derivative =^ *gradient*
- 2nd (discrete) derivative =^ *Laplacian*

a) Gradient

For 2D signals (e.g. images) $f(x,y)$ the gradient

$$\vec{g}(x,y) = \begin{pmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{pmatrix} = \begin{pmatrix} f_x \\ f_y \end{pmatrix}$$

represents the direction and the magnitude of a change in image intensity at position (x,y).

The magnitude of the gradient can be computed as:

$$g\text{mag} = |\sim g(x,y)| = \sqrt{f_x^2 + f_y^2}$$

or can alternatively be approximated by:

$$g\text{max} = |f_x| + |f_y|$$

The direction of the gradient (perpendicular to the contour!) is given by:

$$g_{\text{dir}} = \arctan \frac{f_y}{f_x}$$

Note: Assumes cartesian coordinate system, needs to be adapted for upper left image coordinates!

For discrete signals (e.g. images) partial derivatives f_x, f_y need to be approximated by local differences $\Delta_i f, \Delta_j f$. For computing those the following possibilities exist:

$$\Delta_i f = f_i = \begin{cases} f(i,j) - f(i-1,j) & \text{backward gradient} \\ f(i+1,j) - f(i,j) & \text{forward gradient} \\ f(i+1,j) - f(i-1,j) & \text{symmetric gradient} \end{cases}$$

($\Delta_j f$ defined analogously)

b) 2nd Derivative

... not grey-level difference but change in grey-level curvature is interpreted as an edge.

Potential advantage: “wide” edges can be suppressed if only fast transition from positive to negative curvature is considered as an edge.

Usually approximated for discrete signals (images) by:

$$\Delta^2_{ii} f = f_{ii} = f(i+1,j) - 2f(i,j) + f(i-1,j) \quad \text{For}$$

computation of edge pixels usually the *Laplacian* is used

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

which can be approximated in the discrete case by

$$\nabla^2 f = \Delta^2_{ii} f + \Delta^2_{jj} f = 4f(i,j) - (f(i+1,j) + f(i-1,j) + f(i,j+1) + f(i,j-1))$$

Note: Depending on the definition used the sign of the Laplacian may be inverted (as in the definition above which follows the notation used in [Gon02]!

Problem: Applied in isolation both the gradient and the Laplacian are very sensitive to noise in images!

Slide: Example of ramp edge corrupted by Gaussian noise with increasing variance showing effect in 1st and 2nd derivatives (Fig. 45)

4.1.2 Edge Operators

... i.e. (convolution?!) masks for detecting local discontinuities

- For computation of the gradient (so-called *Prewitt operator*):

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

Remarks:

– Orientation of filter masks depends on orientation of coordinate axes (here: upper left!).

– Variants exist for detecting diagonal edges.

Slide: Example of gradient in x- and y-direction and combined magnitude (Fig. 46)

- Laplacian operator:

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

Remarks:

- Coefficients of edge operator masks sum to zero!
- Different sizes of operator mask may be used.

Note: As gradient/Laplacian are very sensitive to noise usually operators combining edge detection and smoothing are applied.

- *Sobel Operator*

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

- *Laplacian of a Gaussian* (LoG)

  Slide: 5 × 5 Laplacian of a Gaussian mask (LoG) (Fig. 47)

Slide: Example of Sobel operator, Laplacian and Laplacian smoothed with a Gaussian (LoG) (Fig. 48)

4.2 Texture

cf. [Gon02, Sec. 11.3.3], [For03, Chap. 9]

“Texture is a phenomenon that is widespread, easy do recognise and hard to define”

[For03, p. 189] Slide: Examples of textures (Fig. 49)

Problem: Appearance of surfaces or boundaries between object surfaces can not be described by characteristics of a single pixel (e.g. grey-level or color)! ⇒ (local) neighborhood of pixels needs to be considered

Problem: How can local characteristics of textured surfaces be represented? ⇒

Principle approaches:

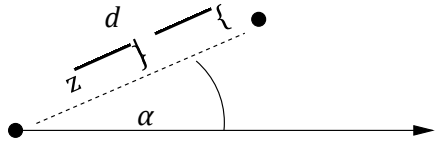
- Filter-bank based representations (cf. [For03, Chap. 9.1])
- Spectral representations (cf. [Gon02, pp. 670–672])
- Statistical representations (cf. [Gon02, pp. 666–670])

4.2.1 Statistical Texture Representation: Co-occurrance Matrices

Goal: (Statistical) Description of relations between pixels' grey values in a local neighborhood.

Assumption: Texture can be characterized by frequencies, with which two pixels with given grey value occur in some given (relative) distance and with given orientation.

Let $n_{i,j}(d,\alpha)$ be the number of times that pixels with grey level i and j occur with distance d and f(u,v) = j orientation α.



$f(x,y) = i$

Remarks:

- If the total number of grey levels in an image is L one obtains $L \times L$ co-occurrance counts $n_{i,j}(d,\alpha)$.
- For counting grey level co-occurrances a local neighborhood of given size must be specified (How to treat image boundaries?).

Definition:

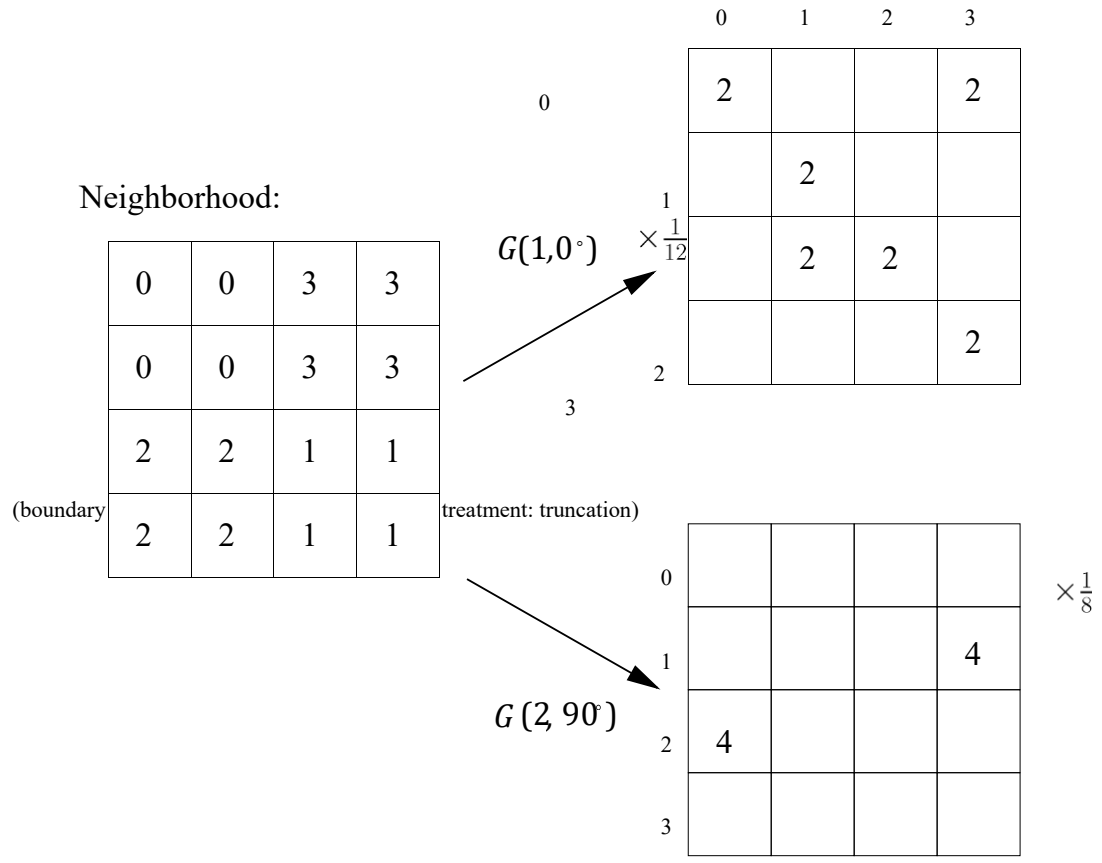
The normalized *grey-level co-occurrance matrix* (GLCM) $G(d,\alpha)$ is defined as:

$$G(d,\alpha) = [g_{i,j}(d,\alpha)] = \left[\frac{n_{i,j}(d,\alpha)}{\sum_{i=0}^{L-1} \sum_{j=0}^{L-1} n_{i,j}(d,\alpha)} \right]$$

Note: A single GLCM is not sufficient for describing a texture (only a single distance and orientation parameter is used)!

⇒ Matrices need to be computed for *several different* distance and orientation parameters!

Example:



Problem: GLCMs considerable increase the parametric representation of textures (multiple matrices for different d and α required)!

⇒ Calculate features *derived* from GLCMs

Possible Features based on GLCMs:

a) *Energy*

$$e(d,\alpha) = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} g_{i,j}^2(d,\alpha)$$

Defines a measure for the "regularity" of a texture, because:

– in regular textures only few strong/prominent grey-level differences occurr ⇒ only few large $g_{i,j}(d,\alpha)$ ⇒ summation of squared values large

– otherwise many/all grey-level differences exist ⇒ many small $g_{i,j}(d,\alpha)$ ⇒ summation of squared values small

b) *Contrast*

$$c(d,\alpha) = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1}(i-j)^2 g_{i,j}(d,\alpha)$$

Measures the magnitude of local changes in grey value:

– homogeneous regions (i.e. with $i = j$) will be suppressed
– large differences are weighted strongly: $(i-j)^2$ Can be rewritten as:

$$c(d,\alpha) = \sum_{\Delta l=0}^{L-1} \Delta l^2 \sum_{|i-j|=\Delta l} g_{i,j}(d,\alpha)$$

with Δl beeing the local grey level difference.

c) *Homogeneity*

$$h(d,\alpha) = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} g_{i,j}(d,\alpha)\frac{1}{1+|i-j|}$$

Acts approximately inversely to the computation of contrast: Homogeneous regions (i.e. $i = j$) are weighted strongly, other areas less strongly ($\frac{1}{1+z} < 0$ if $z > 0$). Can be rewritten as:

$$c(d,\alpha) = \sum_{\Delta l=0}^{L-1} \frac{1}{1+\Delta l} \sum_{|i-j|=\Delta l} g_{i,j}(d,\alpha)$$

(with Δl beeing the local grey level difference).

4.2.2 Spectral Texture Representation: Overview

Basis: (Fourier) Spectrum decomposes image into periodic funcions (sines and cosines) of different wavelength and amplitude.

Idea: Capture peroidic grey-level patterns within textures by frequency based representations.

Note: Similarly to GLCMs not the raw spectrum but features derived from it are used for representing/characterizing textures!

Examples of features of the spectrum useful for texture representation:

a) Prominent peaks ⇒ give the principal direction of the texture patterns

b) Location of prominent peaks in the frequency plane ⇒ represents fundamental (spatial) period of texture patterns

Extracting such features can be simplified by expressing the spectrum $S(u,v)$ in polar coordinates as a function $S(r,\theta)$.

For each radius r or angle θ the spectrum may be considered as a 1D function $S_r(\theta)$ or $S_\theta(r)$, respectively.

More global representations can be obtained by integration (acually summation in the discrete case) of those functions:

$$S(r) = \sum_{\theta=0}^{\pi} S_\theta(r)$$

and

$$S(\theta) = \sum_{r=1}^{R_0} S_r(\theta)$$

(with R_0 is maximal radius of a circle centered at the origin; because of symmetry only angles between 0 and π need to be considered)

Slide: Periodic textures and derived frequency-based representations (Fig. 50)

4.3 Motion

cf. [Hor81]

When an object within a scene moves relative to the position of the camera (or when the camera moves relative to the scene) the 2-dimensional projection onto the image plane $f(x,y)$ is additionally dependent on the time t, yielding $f(x,y,t)$.

Motion in the image plane (!) can be estimated by finding corresponding pixels in subsequent images (=^ displacement wrt. x/y-direction).

Note: Motion in 3D can only be estimated if, additionally, depth information is available!

Potential Methods:

- *Optical flow*: implicit motion estimation
- Techniques based on *matching*: explicit motion estimation either of image pagches (=^ block-matching) or of so-called keypoints (cf. Section 5.4)

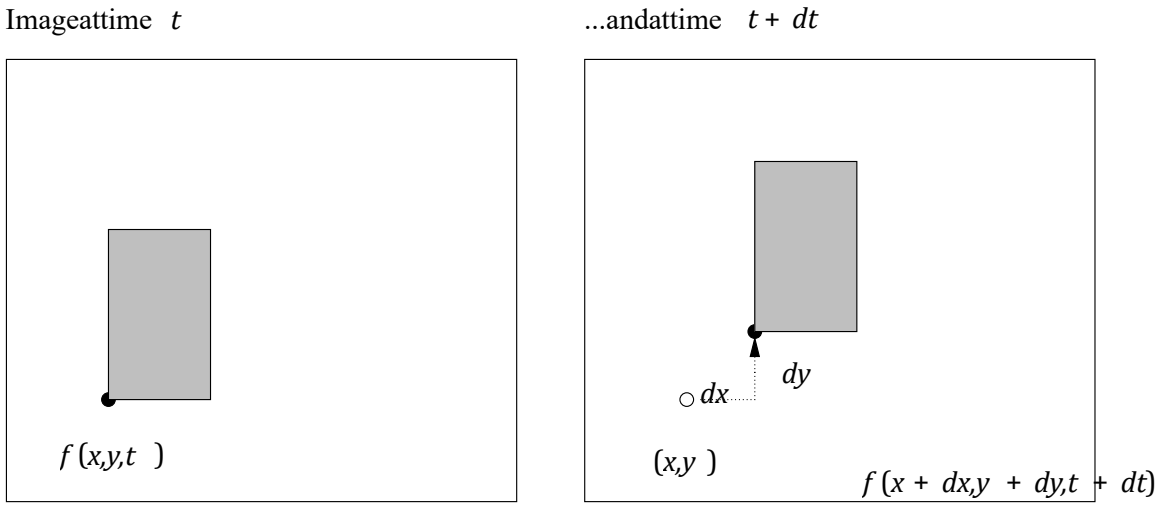
4.3.1 Optical Flow

Idea: Infer information about motion within a scene from changes in grey-level structure.

Assumptions: During motion ...

- ... the orientation of surfaces relative to illumination sources is invariant (or the illumination is uniform) ...
- ... and the orientation relative to the observer is constant (or the grey-level is independent of a change in orientation).

Sketch of principal situation:



On the level of grey values then the following holds:

$$f(x,y,t) = f(x + dx,y + dy,t + dt)$$

An expansion of the expression on the right hand side into a Taylor series at the point (x,y) yields:

$$f(x,y,t) = f(x,y,t) + \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial t}dt + \{residual\}$$

When ignoring terms of higher order within the Taylor expansion (i.e. the "residual") one obtains (with $f_x = \frac{\partial f}{\partial x}$ etc.):

$$f_x dx = f_y dy + f_t dt = 0$$

With the definition of velocities in x/y-direction $u = \frac{dx}{dt}, v = \frac{dy}{dt}$ this results in the so-called *motion constraint equation*:

$$E_m = f_x u + f_y v + f_t = 0$$

4.3.2 Estimation of Optical Flow

Problem: Optical flow (i.e. velocities u and v) is not uniquely defined by the motion constraint equation E_m.

However, assuming that every point in the image could move independently is not realistic.

Constraint: Usually opaque objects of finite size undergoing rigid motion are observed ⇒ neighboring points in the image will (most likely) lie on the same object (surface) and, therefore, have similar velocities.

⇒ Optical flow is required to be *smooth*

This smoothness constraint can be expressed by minimizing (the square of) the magnitude of the gradient of the flow velocities:

$$\mathcal{E}_s^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \rightarrow \text{min!}$$

The motion constraint equation *and* the smoothness constraint should hold for all positions in the image. Due to violation of the assumptions and noise in the image in practice this will, however, never be exactly the case.

Therefore, a combined constraint can be formulated as minimizing the following total error:

$$\mathcal{E} = \int\int (\mathcal{E}_m^2 + \alpha^2 \mathcal{E}_s^2) dx dy$$

where α^2 is a suitable weighting factor.

Using variational calculus one can show that a neccessary condition for an extremum (minimum) is given by:

$$\begin{aligned} f_x^2 u + f_x f_y v &= \alpha^2 \bigtriangledown^2 u - f_x f_t \\ f_x f_y u + f_y^2 v &= \alpha^2 \bigtriangledown^2 v - f_y f_t \end{aligned}$$

Note: The Laplacian $\nabla^2 u$ (or $\nabla^2 v$) can be approximated in the discrete case as:

$$\nabla^2 u = \beta(\bar{u}_{i,j,k} - u_{i,j,k})$$

with $\bar{u}_{i,j,k}$ beeing a suitable average over some local neighborhood, e.g. 4-neighborhood:

$$\bar{u}_{i,j,k} = \frac{1}{4}(u_{i-1,j,k} + u_{i,j-1,k} + u_{i+1,j,k} + u_{i,j+1,k})$$

The proportionality factor β then needs to be set to 4.

The constraint equations can then be rewritten as:

$$\begin{aligned} (\alpha^2\beta + f_x^2)u + f_x f_y v &= (\alpha^2 \beta \bar{u} - f_x f_t) \\ f_x f_y u + (\alpha^2\beta + f_y^2)v &= (\alpha^2 \beta \bar{v} - f_y f_t) \end{aligned}$$

Further rearanging of terms yields (after some lengthy derivations) the following form, which defines u,v based on the gradients f_x,f_y,f_t and the local averages $\bar{u},\bar{v}$:

$$u = \bar{u} - \frac{f_x(f_x\bar{u} + f_y\bar{v} + f_t)}{\alpha^2\beta + f_x^2 + f_y^2}$$
$$v = \bar{v} - \frac{f_y(f_x\bar{u} + f_y\bar{v} + f_t)}{\alpha^2\beta + f_x^2 + f_y^2}$$

Remarks:

- Choice of α^2 plays only a siginificant role in areas with small gradient.
- Strucutre of equations can be used to define an *interative* procedure for computing estimates of the flow velocities, i.e. by calculating new estimates (u^{n+1},v^{n+1}) from estimated derivatives and local averages of previous velocity estimates:

$$u^{n+1} = \bar{u}^n - \frac{f_x(f_x\bar{u}^n + f_y\bar{v}^n + f_t)}{\alpha^2\beta + f_x^2 + f_y^2}$$
$$v^{n+1} = \bar{v}^n - \frac{f_y(f_x\bar{u}^n + f_y\bar{v}^n + f_t)}{\alpha^2\beta + f_x^2 + f_y^2}$$

(Procedure can be initialized by assuming a uniformly vanishing flow field, i.e. velocities $u = v = 0$.)

Slide: Example of optical flow computation (Fig. 51)

4.4 Depth

cf. also [For03, Chap. 11]

Problem: When mapping a 3D scene onto a 2D image (plane) information about depth (i.e. distance of a scene point from the image plane) is lost!

Potential Methods for obtaining depth information:

- *Measuring* depth, e.g. by so-called laser range finders
- Use of *structured light*, i.e. superimposing a light pattern of known geometry onto the scene
- Using multiple views of the scene (e.g. two images ⇒ *stereo vision*) to estimate depth

(Note: The first two methods are *active*, the third one *passive*.)

4.4.1 Stereo Vision

Principle: Two (or more) images/views of a scene are captured from different positions/view-points.

A scene point it thus mapped onto two *corresponding* image points P_L and P_R in the two stereo images (if it's not occluded).

If the geometry of the camera configuration is known, depth information can be recovered from the position of the corresponding image points.

Problem: • How to determine parameters of the stereo configuration?

(often simplified setups are used)

- How to find corresponding scene points?

⇒ similar to motion estimation

Simplified Stereo Setup

- Parallel optical axes
- normalized images (stereo baseline parallel to x-axis, no optical distortions)

⇒ relevant parameters:

- width of stereo baseline
- focal length of camera (≈ distance of image plane to optical center)
- (size, distance of image pixels) ⇒ corresponding image points differ in x-coordinate only!

(Note: Additionally, a mapping of camera coordinates to a world coordinate system may be required.)

Slide: Simplified stereo setup (Fig. 52) Remarks:

- Camera coordinate system (X_C,Y_C,Z_C) refers to *left* camera
- Coordinates of sene point $P(x_C,y_C,z_C)$ already in 3D camera coordinates

The following relations between camera (3D) and image coordinates (2D) can be obtained:

$$\begin{pmatrix} x_L \\ y_L \end{pmatrix} = \frac{f}{z_C} \begin{pmatrix} x_C \\ y_C \end{pmatrix} \text{ and } \begin{pmatrix} x_R \\ y_R \end{pmatrix} = \frac{f}{z_C} \begin{pmatrix} x_C + b \\ y_C \end{pmatrix}$$

The distance of corresponding image points is called *disparity* (obtained from the above relations):

$$d = x_R - x_L = \frac{f\,b}{z_C}$$

The 3D coordinates of a scene point can be recovered as follows:

$$\begin{pmatrix} x_C \\ y_C \\ z_C \end{pmatrix} = \frac{b}{d} \begin{pmatrix} x_L \\ y_L \\ f \end{pmatrix}$$

Problem: Correspondence between image points needs to be established!

Methods: Similar as for motion: Optical flow, Block-matching

4.4.2 Estimation of Stereo-Correspondence via Optical Flow

Let's consider a trivial image sequence consisting of left and right image by defining:

$$f(x,y,0) = f_L(x,y) \qquad \text{and} \qquad f(x,y,1) = f_R(x,y)$$

Using the simplified stereo setup (as of above) one obtains the following constraint equation:

$$E_c = f_x u + f_t = 0$$

(with f_t beeing the partial derivative of $f()$ in the "direction of the stereo setup" and u corresponding to an estimate of the disparity d).

Note: No displacement/"motion" in y-direction, as images are aligned.

When ideal solution is not possible/can't be computed, additional constraints must be used, e.g. smoothness of the displacement field:

$$\mathcal{E}_s^2 = u_x^2 + u_y^2 \rightarrow \text{min!}$$

Problem: Assumption of general smoothness is violated especially at boundaries of objects!

Solution: Use so-called "directed" smoothness, which considers changes in the displacement field perpendicular to the grey-level gradient.

The grey-level gradient is defineds as: $\begin{pmatrix} f_x \\ f_y \end{pmatrix}$!

Perpendicular to the gradient is the vector: $\begin{pmatrix} f_y \\ -f_x \end{pmatrix}$!

The amount of change in displacement $\begin{pmatrix} u_x \\ u_y \end{pmatrix}$! in the direction of some vector is given by the projection onto that vector:

$$(f_y \quad -f_x)\begin{pmatrix} u_x \\ u_y \end{pmatrix} = f_y u_x - f_x u_y$$

Consequently, the following constraint realizes “directed” smoothness:

$$E_d s^2 = (f_y u_x - f_x u_y)^2 \rightarrow \min!$$

All constraints can be integrated when minimizing the following total error:

$$E = Z\ Z\ E_c + \alpha^2(E_s^2 + \beta^2 E_d s^2)dxdy$$

Chapter 5

Image Primitives

The basis for recognizing objects in images / describing scenes is usually formed by more elementary "components" – so-called *image primitives*.

Goal: Partitioning of images into meaningful primitives that form the basis for the extraction of objects of interest (for a specific application) within (simple or complex) scenes.

Question: What are "meaningful" image primitives in general?

Examples for scenes / scene complexity:

a) Simple artificial or industrial scenes

   (rigid objects before a known or even homogeneous background, controlled lighting conditions)

b) "simple" natural scenes

   (mostly indoors [e.g. office environment], environment and lighting conditions controlled to some extent)

c) Complex natural scenes

   (heavily textured, outdoors, unknown lighting conditions, ...)

5.1 Fundamentals

The most widely used (i.e. "established") image primitives result from *image segmentation*, where the goal is to segment a given image into a set of *regions* or *contours*.

- Regions =^ homogeneous image areas (roughly corresponding to objects or object surfaces)
- Contours =^ local discontinuities / inhomogeneities, i.e. *edges* (roughly corresponding to boundaries of objects or object surfaces)

Note: The process of segmenting an image into regions and contours is approximately dual, i.e. region boundaries are approximately equivalent with contours and contours enclose regions.

A rather "modern" type of image primitives are so-called *keypoints* (or interest points). A set of keypoints defines a sparse representation (wrt. segmentation) of image content by extracting "interesting" points/positions within an image, usually augmented by a local feature representation.

5.2 Region Segmentation

cf. [Gon02, Sec. 10.4]

Assumptions:

- Regions are topoligically connected
- A region is a homogeneous image area ⇒ A predicate P can be defined, that is true for *one specific* region and false if the region is extended by neighboring pixels (=^ homogeneity criterion)

The following conditions must be satisfied:

1. $S_i R_i = I$, i.e. the set of regions R_i segment (or describe) the *whole* image I
2. $R_i \cap R_j = \emptyset$ for $i = 6$ j, i.e. regions don't overlap, are distjoint sets of pixels
3. R_i is topologically connected (i.e. for every pair of pixels in R_i there exists a connecting path of neighboring pixels within R_i)
4. $P(R_i)$ = TRUE, i.e. every region satisfies the homogeneity criterion P
5. $P(R_i \cup R_j)$ = FALSE for $i = 6$ j and topologically *neighboring* regions R_i and R_j

Example for a (simple) homogeneity criterion P:

$$P(R_i) \quad \begin{cases} \text{=TRUE if } |f_{jk} - f_{lm}| \leq d \text{ for all pairs of pixels } (j,k) \text{ and } (l,m) \text{ in } R_i \\ \text{FALSE otherwise} \end{cases}$$

Note: A homogeneity criterion can be defined on any feature of a pixel i.e. grey level, color, depth, velocity, ...

5.2.1 Region Growing

Basic Method: Starting from "suitable" initial pixels regions are iteratively enlarged by adding neighboring pisxels.

Problems:

- Finding starting points for region growing process (seed points)
  - *ideal:* exactly one per region
  - *salient* pixels, where saliency needs to be defined by an additional criterion (brightest/darkest pixel, high contrast pixel, ...)
  - *all* pixels ⇒ enlarging a region leads to the merging of two regions (Which of the possible operations is the most suitable one?)
- Ordering of performed region growing operations

5.2.2 Splitting

Idea: Starting from a single initial region (i.e. the whole image) regions are subsequently subdivided according to some scheme until all regions satisfy the homogeneity criterion P.

Principle Method:

1) Initialize the set of image regions to a single region $R_0 = I$ 2)

While not all regions R_i satisfy $P(R_i)$

2a) Choose some R_i with $P(R_i)$ = FALSE

2b) Subdivide R_i with a suitable method into regions R_{ik}

2c) Replace R_i in the set of image regions by $\{R_{i1}, R_{i2}, \ldots R_{iN_i}\}$

Problem: Suitable method for subdividing a region!

Possible Solution:

- Calculate grey-level histogram of the region
- Determine a binarization threshold (e.g. by approximation of the grey-level distribution by a mixture of two Gaussian distributions or by selection of the most prominent minimum in the histogram)
- Binarization of the source region R_i
- Connected components/areas form sub-regions R_{ik} of R_i

5.2.3 Split-and-Merge

Motivation: Exploit advantages of region splitting and growing (respectively merging) methods by combining both techniques

Basic Idea: Starting from an initial segmentation suitable splitting and merging operations are applied until a final segmentation is reached which satisfies all criteria.

Method (after Pavlidis): Split and merge operations operate on a *quad-tree* representation of the image

Slide: Quad-tree representation of image segmentation (Fig. 53)

Split-and-Merge Algorithm (Pavlidis)

1. Calculate initial segmentation

   (e.g. randomly, based on prior knowledge [e.g. about the position of an object, or by selecting an appropriate segmentation level in the quad-tree representation)

   Slide: Simple sample image (Fig. 54) and initial segmentation (Fig. 55)

2. Perform all possible region merging operations (within the quad-tree structure!)

   (i.e. eventually, 4 nodes in the quad-tree will be replaced by their parent node)

   Slide: Result of possible merge operations on the sample image (Fig. 56)

3. Perform all possible regions splits, for regions (i.e. quad-tree nodes) that don't satisfy thehomogeneity criterion $P(R_i)$

   Slide: Result of possible splitting operations on the sample image (Fig. 57)

4. Merge adjacent regions that - when merged - satisfy the homogeneity criterion $P(R_i \cup R_j)$.

   Note: In this step the algorithm goes beyond the quad-tree structure of the image.

   Slide: Result of merging operations *outside* the quad-tree structure for the sample image (Fig. 58)

5. Eliminate very small regions by merging them with the neighboring region with most similargrey value.

   Slide: Final segmentation for result the sample image (Fig. 59)

5.3 Contour Extraction

Baseline: Edge detection (i.e. contour extraction starts from edge images)

Note: Methods for edge detection generate *after* the application of a suitable thresholding operation a set of edge pixels where significant changes in grey level occur (per pixel a measure for the strength and possibly direction of the edge).

Problems:

- Edges are defined on the pixel level only

⇒ Contours need to be built from sets of edge pixels defining an object boundary

- *Real* contours (e.g. boundaries of objects) are generally several edge pixels wide. • From image noise many non-edge pixels originate with significant changes in grey level, i.e. high edge strength.

Descriptions of contours can be

- non-parametric, i.e. linked lists of edge pixels or
- parametric representations, i.e. geometric shapes (lines, curves, [partially defined] circles/ellipses) derived from underlying edge pixel sets

5.3.1 Contour Following

cf. also [For03, ??]

Goals: • Extraction of significant edge elements (ideally exactly one edge pixel in the direction of the contour),

- grouping of edge pixels, and •

approximation by a parametric function.

Processing Steps:

1. Thinning (of the edge image)
2. Linking (of edge elements)
3. Approximation

Note: Linking & Approximation can also achieved in an integrated manner via the Hough tranformp.

Thinning

... can e.g. be realized by the following two processing steps:
1. Non-maximum suppression

Goal: Reduction edges to a width of 1 pixel only.

Basic Method: Eliminate edge elements if the edge intensity is smaller than that of another edge element in the direction of the gradient (i.e. perpendicular to the edge/potential contour, two neighbors must be considered)

2. Hysteresis thresholding

Goal: Elimination of non-significant edge elements

Idea: Thresholding with a single threshold usually produces bad results ⇒ use *two* thresholds!

Basic Method:

- Set two thresholds θ_{low} and θ_{high}

  (suitable choice e.g.: $\theta_{\text{high}} = 0.1 \cdot \max_{i,j}\{g_{\text{mag}}(i,j)\}$ and $\theta_{\text{low}} = 0.3 \cdot \theta_{\text{high}}$)

- If the local edge intensity $g_{\text{mag}}(i,j)$ is *larger* than θ_{high}: *keep* edge element.

- If the local edge intensity $g_{\text{mag}}(i,j)$ is *smaller* than θ_{low}: *eliminate* edge element. • If the local edge intensity lies between the two thresholds: keep edge element only, if it is *connected* to another edge element with $g_{\text{mag}}(i,j) > \theta_{\text{high}}$.

Note: Applying the Sobel-Operator (Gaussian-smoothed gradient calculation) followed by non-maximum suppression and hysteresis thresholding constitutes the so-called *Canny Edge Detector*. Slide: Example for results of Canny Edge Detector (Fig. 60)

Linking

Goal: Explicit representation of neighborhood relations between edge elements

Idea: Start with a randomly chosen edge element and extend sequence of linked edge elements by one neigbor at a time. Repeat until all edge elements are covered by some linked set.

Problems:

- Contours counld be broken up into two (partial) chains of edge pixels (if started “in the middle”).

  ⇒ extend edge pixel chain in *both* directions

- At junctions multiple continuations for building edge pixel chains are possible.

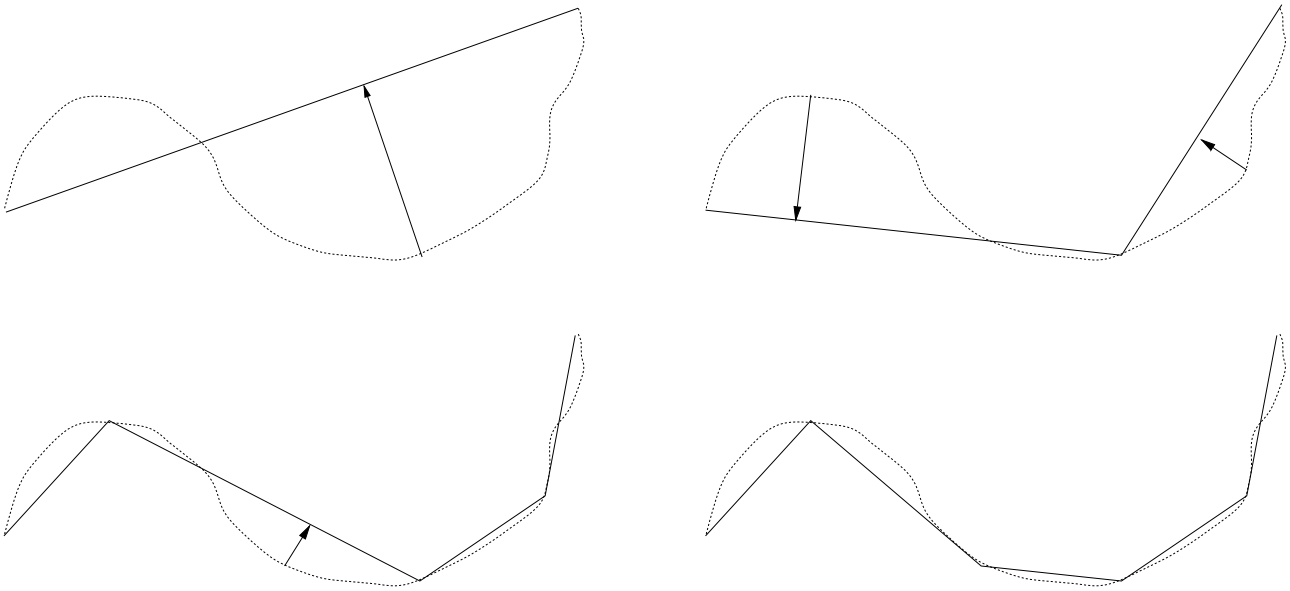
  ⇒ Continue contour in direction with most similar gradient.

Approximation

Goal: Approximation of linked sets of edge elements by a parametric function (e.g. line, circle, ...)

Method (for polygonal approximation [Ram72], i.e. with line segments):

- Connect starting and ending point of pixel chain by a line.
- Find edge element with maximum distance to this line.
- Split up line at this point, if distance is too large (threshold!).
- Recursively process all segment approximations until no more line splits are necessary.



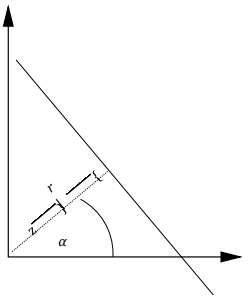
5.3.2 Hough-Transform

cf. [Gon02, Sec. 10.2.2]

... does not try to approximate given edge pixels by a parametric representation but investigates the parameter space of approximations of all edge elements.

a) Simple Case: Approximation by a staight line

Baseline: A straight line can be represented with parameters α and r as follows:

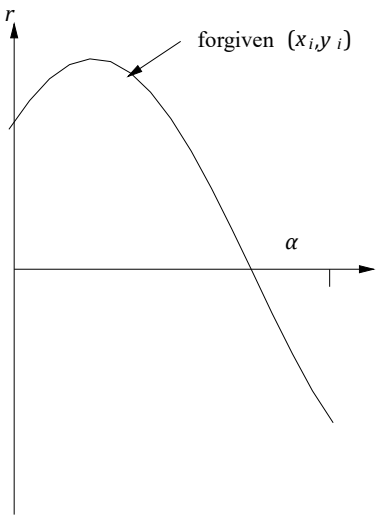


$$r = x\cos\alpha + y\sin\alpha$$

For any point (x_i,y_i) then the following holds:

$$r = x_i\cos\alpha + y_i\sin\alpha$$

This expression can be considered as a function of α. It then represents an sinusoidal curve in the α-r-plane.



Advantage (over the canonical representation of staight lines as $y = ax + b$):

The space of parameters is *finite*!

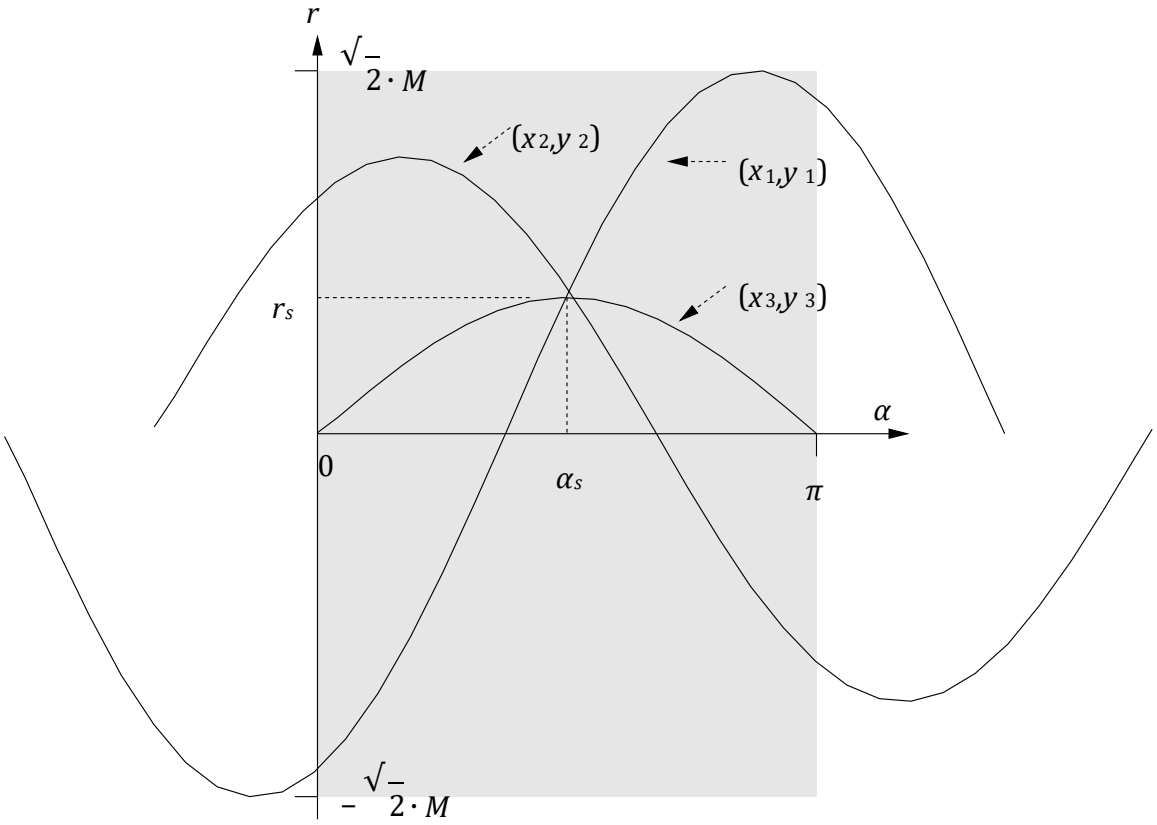
$$0 \le \alpha < \pi \quad \text{and} \quad -\sqrt{2} \cdot M \le r < \sqrt{2} \cdot M$$

(for $M \times M$ image; other choices possible, e.g. the one used in [Gon02, Sec. 10.2.2])

Method

- For all edge pixels (x_i, y_j) "draw" the associated curve (or function) in the α-r-plane.
- The intersection points of all curves drawn represent parameters (α_k, r_k) of lines which approximate the associated edge pixels.

Simple example:



⇒ The line $r_s = x\cos\alpha_s + y\sin\alpha_s$ passes through the points (x_1, y_1), (x_2, y_2), and (x_3, y_3).

Problem: Determining intersection point mathematically exactly in practice too difficult or unusable (slight deviations due to noise, quantization, and errors in the edge detection process).

Solution: Just as the image plane the α-r-plane needs to be represented digitally, too. Therefore, the ranges of α and r are quantized resulting in an α-r-matrix. The cells of this matrix can be thought of a accumulators that are incremented whenever a sinusoidal curve for some edge pixel passes through the accumutor's parameter range.

Reconstruction of Contours: Salient contours in the image are found by selecting α-r parameter pairs with high accumulator counts and checking the associated edge pixels for continuity.

Slide: Illustration of the Hough transform (Fig. 61)

Slide: Example of the Hough transform on sample infrared image (Fig. 62)

b) Parametric Case

The Hough transform can be applied in all cases where a parametric representation of the curve used for approximating contours is possible in the form:

$$g(x,y,\mathsf{c}) = 0$$

Here (x,y) represents image coordinates and c an arbitrary parameter vector. E.g.

for the approximation of contours with (arcs of) circles:

$$(x - c_1)^2 + (y - c_2)^2 = c_3^2$$

Note: Extension for non-parametric functions possible ⇒

Generalized Hough transform (cf. [Bal82])

5.4 Keypoints

The use of so-called keypoints for image/object description requires ...

- a method for detecting keypoint locations and ...
- a method for describing the local image properties at the keypoint location as uniquely as possible for later retrieval.

5.4.1 The Scale-Invariant Feature Transform (SIFT)

after [Low04]

... keypoint detection method invented (and patented!) by David Lowe.

Goal: Find keypoint locations and appropriate descriptions that are (approximately) invariant with respect to *scale*, *rotation* and — to some extent — *view-point changes*.

Solutions:

1. Scale invariance → detect keypoints in *scale space*

2. Rotation invariance → determine local orientation at keypoint location

3. "view-point invariance" → keypoint description based on local gradients

Scale-Space Representation of Images

Different scale representations $L(x,y,\sigma)$ of an image $I(x,y)$ are obtained via a convolution with a Gaussian $G(x,y,\sigma)$ with variable scale (i.e. standard deviation) σ, where

$$L(x,y,\sigma) = I(x,y) * G(x,y,\sigma)$$

and

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

The scale space of an image $I(x,y)$ is then defined by the sequence of Gaussian-smoothed versions $L(x,y,k^n\sigma)$ for some σ, a constant multiplicative factor k and $n = 0,1,2,\ldots$.

Note: With every doubling of the scale $k^n\sigma$ (i.e. every so-called *octave*) the scale space image $L(x,y,k^n\sigma)$ can be subsampled by a factor of 2 without loosing information.

Keypoints are detected as the local extrema in the *Difference of Gaussian* (DoG) representation of scale space, i.e.:

$$\begin{aligned} D(x,y,\sigma) &= I(x,y) * (G(x,y,k\sigma) - G(x,y,\sigma)) \\ &= L(x,y,k\sigma) - L(x,y,\sigma) \end{aligned}$$

⇒ can easily be computed from difference of neighboring scales $L(x,y,k^n\sigma)$.

Slide: Scale space representation scheme for images (Fig. 63)

Keypoint *candidates* are defined as the maxima or minima in the DoG images by comparing a pixel position to its neighbors in 3 × 3 regions in the local and both adjacent scales.

Slide: Determining extrema in DoG image representations (Fig. 64)

Slide: Example of keypoints detected (Fig. 65)

Keypoint Localization and Filtering

Keypoint locations are determined with sub-pixel accuracy by fitting a 3D quadratic function to the local sample points and determining the location of the interpolated extremum.

Furthermore, keypoint candidates are discarded that:

- have low contrast or
- lie on edges.

Keypoint Orientation

After interpolation a keypoint can be associated with the Gaussian-smoothed image $L(x,y,\sigma)$ at the scale σ closest to the keypoint.

The local orientation at the keypoint (x_0,y_0) is then determined from a histogram of gradient orientations (resolution of 10 degrees, i.e. 36 bins, weighted by gradient magnitude), which is computed in the region around the keypoint over the gradients of $L(x,y,\sigma)$ (weighted by a circular Gaussian window with $\sigma' = 1.5\sigma$).

⇒ Local maxima (peaks) in the histogram correspond to dominant local orientations.

Keypoints are created for all local maxima in the gradient histogram which are within 80% of the global maximum (i.e. one candidate location might be used to create multiple keypoints).

After keypoint creation peak positions are interpolated by a parabola over three adjacent bins in the orientation histogram.

Keypoint Descriptor

- (Image) Gradient magnitudes & orientation are sampled around the keypoint (at scale of keypoint, weighted by Gaussian)
- Orientations are rotated relative to keypoint orientation (≈ rotation invariance)
- Orientation histograms are created over $m \times m$ sample regions (e.g. 4×4 regions with 8×8 sample array)

⇒ Concatenation of histograms defines the keypoint descriptor (usually $4 \times 32 = 128$ bytes)

Slide: Scheme for computing SIFT keypoint descriptor (Fig. 66)

Note: Keypoint descriptors can be matched (nearest neighbor) accross images in order to identify – to some extent – identical/similar locations at different scales/rotations/view points.

Slide: Correspondences between matched keypoints (Fig. 67)

Chapter 6

Appearance-Based Object Recognition

In contrast to object recognition based on image primitives (regions, contours) and some method for finding appropriately structured configurations of those that correspond to objects, in appearancebased approaches objects are represented – more or less – by using image data directly (e.g. different views of an object).

Known instances of objects can then be *identified* by matching the image-based representations to new data. Using an appropriate similarity criterion also objects from an object *category* can be found.

6.1 Template Matching

cf. also [Gon02, pp. 205-208]

... simplest case of an appearance based model.

- Build "model" of an object by extracting a (rectangular) image area (the *template f*) showing the desired object.

  Note: Also multiple templates (=^ views) per object can be used.

- For finding (this object, a similar one) in a new image *g* compute "similarity" between the template and the new image at every position in the image.

  Common similarity measure: cross-correlation

$$h(x,y) = f(x,y) \circ g(x,y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) g(x+m, y+n)$$

  Note: Similar to convolution!

- Position(s) where object is hypothesized correspond to (local) maxima of the cross-correlation measure *h*.

Note: As the cross-correlation measure not only depends on the similarity between template and image but also on the local brightness of the image, an appropriate normalization is necessary in practice (see the tutorialsp).

6.2 Matching Configurations of Keypoints

cf. [Low04]

Note: Matching of *single* keypoints not reliable enough for object detection / recognition

Basic Idea: Match sets of keypoints (lying on desired object) and verify correct spatial configuration of matches

Simple Method: When assuming *no distortion* of keypoint configuration (i.e. no deformation of object due to e.g. out-of-plane rotations) use generalized Hough transform

1. Model of object:
   (a) Set of keypoints (specified by their descriptors [including local orientation and scale!])
   (b) Reference point on object
   (c) For each kepoint: Vector to reference point *relative* to keypoint orientation and scale
2. Matching process:
   (a) For every matching keypoint determine reference point *candidate* (exploit local scale & orientation of *matched* keypoint)
   (b) Vote for all reference point candidates associated with keypoint matches
   (c) Reference point(s) with highest number(s) of votes determines location of object hypothesis/es

6.3 Eigenimages

[Tur91], cf. also [For03, Sec. 22.3.2]

Idea: Relevant information for representing a class/set of known objects should be automatically derived from sample data.

Basic Abstraction: Sample images are considered as *points* in a high-dimensional vector space (of images).

Goal: Find suitable representation of the "point cloud" in the image space that represents the known objects.

Derive appropriate similarity measure for finding known object instances or objects from the modeled category.

Most well known example: *Eigenfaces*, i.e. application to the problem of face detection/identification

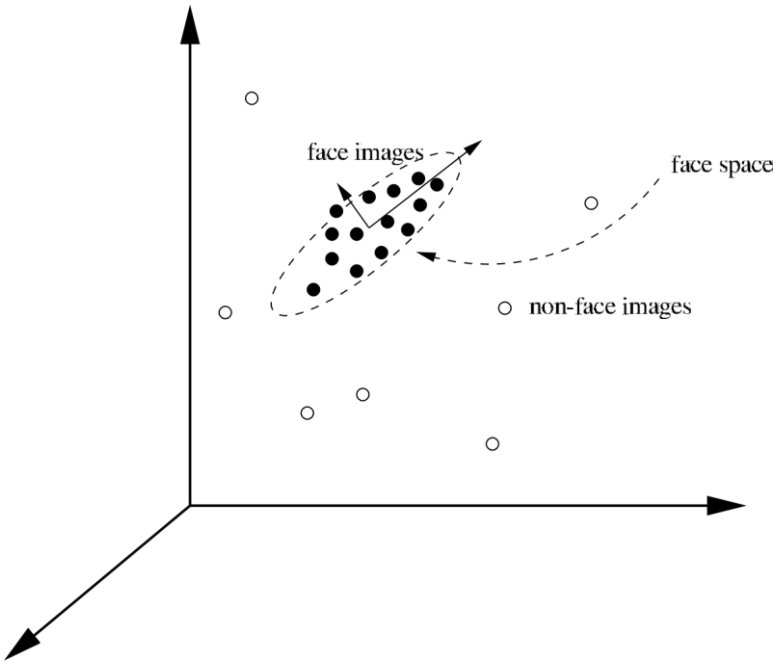
6.3.1 Formal Problem Statement

- Given a set of points/vectors (samples) in some high-dimensional vector space
- Determine sub-space that is defined by the sample set

- Representation of sub-space possible via center of gravity (i.e. mean vector) of samples and sample covariance matrix

  ⇒ Principle components of covariance matrix (i.e. its eigenvectors) span sub-space

- Every sample (i.e. vector in the sub-space) can be reconstructed via a linear combination of the eigenvectors
- Other images (vectors) can be approximated.



Principle Method

Building the Model:

1) Collect a set of sample images from the class of objects to be modeled (e.g. face images)

   Slide: Example of sample set of face images (Fig. 68)

2a) Compute the principal components (Eigenimages) of the sample set and

   Slide: Example of Eigenfaces obtained (Fig. 69)

2b) Select the *K* eigenvectors corresponding to the largest eigenvalues for representing the data

3) For all known instances (individuals) compute the projection onto the modeled sub-space (which is spanned by the selected eigenvectors).

   ⇒ on *K*-dimensional vector of weights per instance

   Note: Modeling quality can be assessed by inspecting reconstructions of the known data.

Recognition using the Model:

4) For an unknown image (e.g. a new face image) compute its distance to the modeled sub-space (e.g. *face*-space).

   ⇒ Reject (i.e. none of the known objects) if too large.

   Note: When searching a larger image for smaller realizations of the known objects many possible sub-images at different scales need to be considered!

   Slide: Example of face/non-face classification using image reconstruction via Eigenfaces (Fig. 70)

5a) Compute projection onto the sub-space (see above) and

5b) Classify the resulting vector as known or unknown instance given the projections of the (known) sample data (e.g. by mapping to the nearest neighbor in the projection space).

6.3.2 Computation of Eigenfaces

... or in general: Eigenimages (for some known object category)

- Let $I(x,y)$ be an $M \times M$ grey-level image of a face

  ⇒ can be consideres as a *vector* of length M^2

- Let $\mathbf{\Gamma} = \{\Gamma_1,\Gamma_2,...\Gamma_N\}$ be the set of training (face) images

- The average face image Ψ is given by:

$$\Psi = \frac{1}{N}\sum_{n=1}^{N}\Gamma_n$$

  The deviation of a face image from the average is, therefore: $\Phi_n = \Gamma_n - \Psi$ (i.e. the set $\mathbf{\Phi} = \{\Phi_1,\Phi_2,...\Phi_N\}$ corresponds to face images normalized to have zero mean).

- For the set of difference images $\mathbf{\Phi}$ perform a *Principal Component Analysis* (PCA)

  ⇒ set of $K < N$ (maximum value of $K = N - 1$ if all Ψ are linearly independent) orthonormal vectors $u_1,u_2,...u_K$

  i.e.

$$u_{Tk}\,u_l = \begin{cases} 1 & \text{if} \quad k = l \\ 0 & \text{otherwise} \end{cases}$$

(where $\{\cdot\}^T$ denotes vector/matrix transpose) and

$$\lambda_k = \frac{1}{N}\sum_{n=1}^{N} \underbrace{(u_k^T \Phi_n)^2}_{\text{projection of } \Phi_n \text{ onto } u_k}$$

is maximum.

- The vectors u_k and scalars λ_k are eigenvectors and eigenvalues, respectively, of the covariance C matrix of the normalized images:

$$C = \frac{1}{N}\sum_{n=1}^{N} \Phi_n \Phi_n^T = AA^T \quad \text{with} \quad A = [\Phi_1, \Phi_2, \ldots \Phi_N]$$

Problem: The covariance matrix C has dimension $M^2 \times M^2$ (i.e. for e.g. 512×512 images

$512^2 \times 512^2 = 262144 \times 262144$)

⇒ computation of eigenvalues/vectors *extremely* problematic

Solution: As there are only N images in the training set, that is used for computing C, a maximum of $K = N - 1$ eigenvectors of C exist ($N - 1 << M^2$).

Compute eigenvectors of $A^T A$ (dimension: $N \times N$)

$$\begin{aligned} A^T A v_i &= \mu_i v_i \qquad |^A \cdot [\ldots] \\ AA^T A v_i &= A\mu_i v_i = \mu_i A v_i \end{aligned}$$

⇒ Av_i is eigenvector of AA^T |{z}

Calculation of u_k:

- Construct matrix $L = A^T A$ with $L_{ij} = \Phi^T_i \Phi_i$
- Compute eigenvectors/values of L
- Eigenvectors of C are obtained as:

$$u_k = Av_k = [\Phi_1, \Phi_2, \ldots \Phi_N] v_k = \sum_{n=1}^{N} v_{kn} \Phi_n$$

- A *known* face image Γ can now be represented by its projection onto *face space*:

$$\omega_k = u^T_k (\Gamma - \Psi) \qquad k = 1 \ldots K$$

with $\Omega^T = [\omega_1, \omega_2, \ldots \omega_K]$

- Any image can be reconstructed (in general only approximately) from its projection onto face space

  With $\Phi = \Gamma - \Psi$ the reconstruction of Φ in face space is:

$$\hat{\Phi} = \sum_{k=1}^{K} \omega_k u_k$$

  which gives an overall recostructed image $\Gamma = \hat{} \Phi + \Psi \hat{}$

- Non-face images can be rejected based on their distance to face space (i.e. the quality of the reconstruction):

$$d\text{face} = ||\Phi - \Phi\hat{}||_2$$

- Face images of known (or possibly unknown) individuals can be identified based on the similarity of their projections to those of known individuals from the training set.

  Simple method: chose individual whose projection Ω_j has minimum euclidian distance to projection Ω of query image:

$$j = \underset{k}{\text{argmin}} ||\Omega - \Omega_k||^2$$

  Problem: How to reject face images of unknown individuals?

Slide: Processing steps necessary for implementation of the Eigenface approach (Fig. 71)

Chapter 7

Tracking

after [For03, Chap. 17]

7.1 Introduction

When objects move in a scene in general a sequence of images is required in order to draw inferences about the motion of the object. The situation is similar when the camera (i.e. the observer) moves through a scene. Then the motion of the observer can be inferred. This problem is known as *tracking*. Application areas are, e.g.:

- *Targeting* (in the military domain): try to predict an object's (target's) future position in the attempt to shoot it.

  Note: Usually radar or infrared images are used.
- *Surveillance:* motion patterns of e.g. people on a parking lot are used to draw inferences about their goals (e.g. trying to steal a car)
- *Automotive:* traffic assistance systems (e.g. for lane-keeping, adaptive cruise control) infer motion of lane marks or other vehicles
- *Motion capture:* special effects in movies sometimes rely on the possibility to track a moving person accurately and to later map the motion onto another - usually artificially generated actor

In order to address the tracking problem the following is required:

- A model of the object's motion (i.e. it's internal state) and
- some set of measurements from the image sequence (e.g. the object's position estimate)

We will consider drawing inferences in the *linear* case only, i.e. the motion model and the measurement are linear.

7.2 Tracking as an Inference Problem

... more specifically a *probabilistic* inference problem.

The object is described by its internal state X_t (at time t or in the t-th frame of the image sequence, respectively). Measurements are taken according to a random variable Y_t (actual value: y_t).

Three main problems to be solved:

- *Prediction:* from past measurements $y_0, y_1, \ldots y_{t-1}$ predict interal state at t-th frame:

$$P(X_t | Y_0 = y_0, \ldots Y_{t-1} = y_{t-1})$$

- *Data association:* from multiple measurements (e.g. position estimates) at frame t "select" the "correct" one. Can be achieved based on the prediction of X_t.

  Possible methods:

  - Selecting the nearest neighbor, i.e. from several measurements y_t^k choose the one maximising $P(\boldsymbol{y}_t^k | \boldsymbol{y}_0, \ldots \boldsymbol{y}_{t-1})$.
  - Perform *gating* (i.e. exclude measurements that are too different from the prediction) and *probabilistic data association* (i.e. a weighted sum [according to the prediction probability] of the gated measurements):

$$y_t = \sum_k P(\boldsymbol{y}_t^k | \boldsymbol{y}_0, \ldots \boldsymbol{y}_{t-1}) \boldsymbol{y}_t^k$$

- *Correction:* correct the internal state incorporating the new measurement, i.e. compute:

$$P(X_t | y_0, \ldots y_{t-1}, Y_t = y_t) \quad \text{Necessary}$$

independence assumptions (in order to make things tractable):

- *Only the immediate past matters*, i.e.:

$$P(X_t | X_0, X_1, \ldots X_{t-1}) = P(X_t | X_{t-1})$$

- *Measurements depend only on the current state*, i.e. not on other measurements taken:

$$P(Y_t, Y_j, \ldots Y_k | X_t) = P(Y_t | X_t) P(Y_j, \ldots Y_k | X_t) \Rightarrow \text{Inferences}$$

for tracking have the structure of a Hidden Markov Model!

7.3 Kalman-Filter

We consider the following linear dynamic model of motion:

- All probability distributions are Gaussians, i.e. can be represented by their mean and associated covariance matrix.
- An estimate $\hat{x}$ of the predicted state is obtained by a linear transform D (could be dependent on time but usually is not):

  $$\hat{x}_t = D x_{t-1}$$

  which is the mean of $P(X_t | X_{t-1})$.
- The uncertainty of the prediction is described by the covariance matrix $\boldsymbol{\Sigma}^d$ (could be time dependent).
- A *measurement matrix* M (could be dependent on time but usually is not) is used to convert between internal state and measurements taken:

  $$\hat{y}_t = M x_t$$

- The uncertainty about the measurement process is represented by the covariance matrix $\boldsymbol{\Sigma}^m$ (which could also be time dependent).

Note: The state vector X_t is normally distributed with mean $\hat{x}_t$ and covariance matrix $\boldsymbol{\Sigma}^d$. The measurement vector Y_t is normally distributed with mean $\hat{y}_t$ and covariance $\boldsymbol{\Sigma}^m$.

Examples of State Representations

... for different assumptions on the nature of the dynamic model.

Note: Measurements are usually 2D/3D position estimates p.

- *(Quasi-)Stationary point:* Internal state and measurements represent identical quantities, i.e. M = I. Motion occurs only under random component, i.e. uncertainty of the measurement (when this is assumed to be quite large, the model can be used for tracking if nothing is known about the object's dynamics).
- *Constant velocity:* The position can be predicted as

$$p_t = p_{t-1} + \Delta t \cdot v$$

Velocity is added to the state representation, i.e. $x = \{\boldsymbol{p}^T, \boldsymbol{v}^T\}^T$.

June 4, 2012 65 The dynamic model is then given by

$$D = \begin{Bmatrix} \boldsymbol{I} & \Delta t \cdot \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{I} \end{Bmatrix}$$

and the measurement matrix is

$$M = \{I\ \boldsymbol{0}\}$$

- *Constant acceleration:* analog to the above with additional acceleration parameter a as component of the state vector.

Kalman-Filtering Algorithm

Goal: Estimate Gaussian probability distributions describing the linear dynamic model optimally in the sense of least mean squared error.

Processing Steps:

Distinguish between state representation estimates before (e.g. $\hat{x}^-_t$) and after (e.g. $\hat{x}^+_t$) the incorporation of a new measurement y_t.

0. Assume some initial estimates of $\hat{x}^-_0$ and the covariance $\boldsymbol{\Sigma}^-_0$ are known.
1. Predict new internal state x_t from past state applying the dynamic model of motion:

$$\hat{x}^-_t = D\hat{x}^+_{t-1}$$
$$\boldsymbol{\Sigma}^-_t = D\boldsymbol{\Sigma}^+_{t-1}\boldsymbol{D}^T + \boldsymbol{\Sigma}^d$$

(covariance combines predicted uncertainty and uncertainty of prediction process)

2. Correct the prediction taking into account the current measurement y_t (Note: Data association needs to be solved separately!).

   Compute Kalman-gain

$$K_t = \Sigma_{-t} M_T (M \Sigma_{-t} M_T + \Sigma_m)_{-1}$$

   which represents the ration between the uncertainty of the model (Σ^-_t) and the uncertainty of the measurement process ($M\Sigma^-_t M^T + \Sigma^m$).

   The *innovation*

$$y_t - M\hat{x}_{-t}$$

   represents the difference between estimated and measured position.

   Depending on the Kalman-gain the innovation is used to correct the estimated state:

$$\hat{x}_{+t} = \hat{x}_{-t} + K_t(y_t - M\hat{x}_{-t})$$

$$\Sigma_{+t} = (I - K_t M)\Sigma^-_t$$

Bibliography

[Bal82] D. H. Ballard, C. M. Brown: *Computer Vision*, Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

[For03] D. A. Forsyth, J. Ponce: *Computer Vision: A Modern Approach*, Prentice-Hall, Upper Saddle River, NJ, 2003.

[Gon02] R. C. Gonzalez, R. E. Woods: *Digital Image Processing*, Prentice-Hall, Upper Saddle River, NJ, 2. Ausg., 2002.

[Hor81] B. K. P. Horn, B. G. Schunck: *Determining Optical Flow*, *Artificial Intelligence*, Bd. 17, 1981, S. 185–203.

[Jah02]¨ B. Jahne:¨ *Digital Image Processing*, Springer, Berlin, 5. Ausg., 2002.

[Low04] D. Lowe: *Distinctive Image Features from Scale-Invariant Keypoints*, *Int. J. of Computer Vision*, Bd. 60, Nr. 2, 2004, S. 91–110.

[Nie90] H. Niemann: *Pattern Analysis and Understanding*, Bd. 4 von *Series in Information Sciences*, Springer, Berlin Heidelberg, 2. Ausg., 1990.

[Nie03] H. Niemann: *Klassifikation von Mustern*, 2003.

[Ram72] U. Ramer: *An Iterative Procedure for the Polygonal Approximation of Plane Curves*, *Computer Graphics and Image Processing*, Bd. 1, Nr. 3, 1972, S. 244–256.

[Sch01] D. Schluter:¨ *Hierarchisches Perzeptives Gruppieren mit Integration dualer Bildbeschreibungen*, Dissertation, Universitat Bielefeld, Technische Fakult¨ at, okt 2001.¨

[Tur91] M. Turk, A. Pentland: *Eigenfaces for Recognition*, *Journal of Cognitive Neuro Science*, Bd. 3, Nr. 1, 1991, S. 71–86.

Figure1:Arialimageofwellknowuniversity

Figure2:TrafficinthecityofTaipei

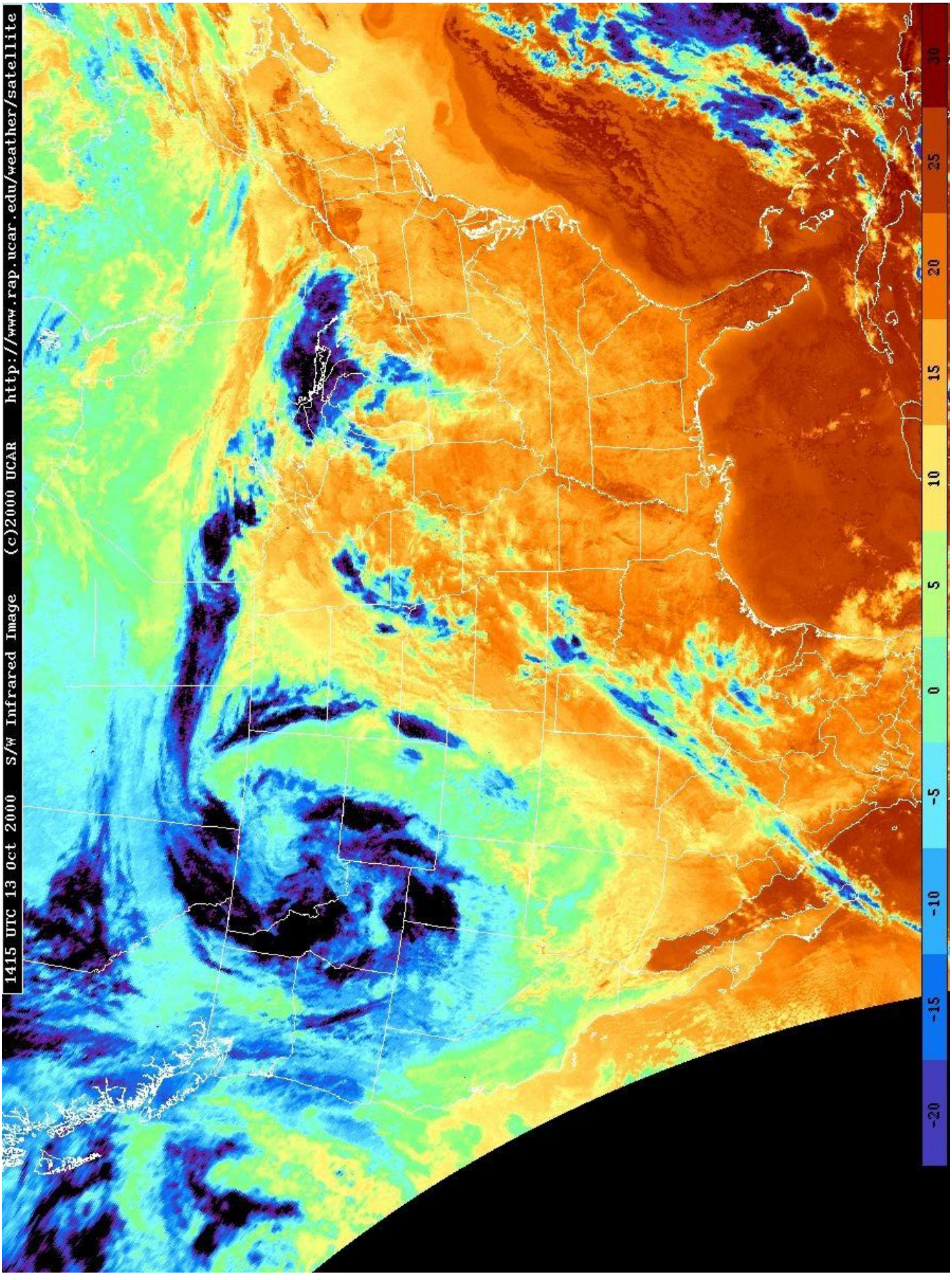


Figure3:InfraredimageofNorthAmerika(Source:http://www.nnic.noaa.gov/SOCC/gallery.htm)

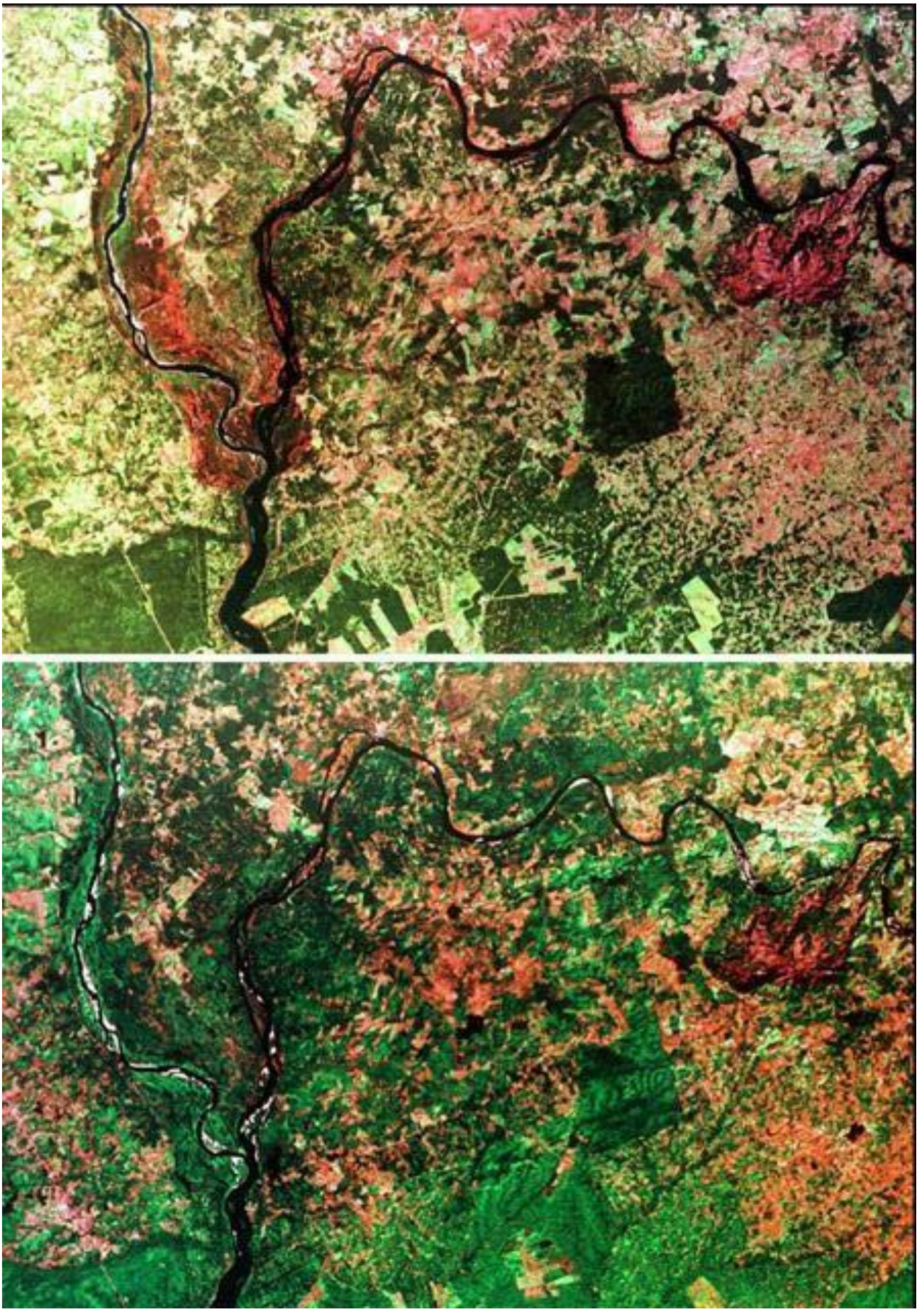


Figure4:MultispectralLandSatimageofAmazonasrain-forestregion(Source:http://www.nnic.noaa.gov/SOCC/gallery.htm)

Figure5:X-rayimageofhumanchest(left)andhand(right)

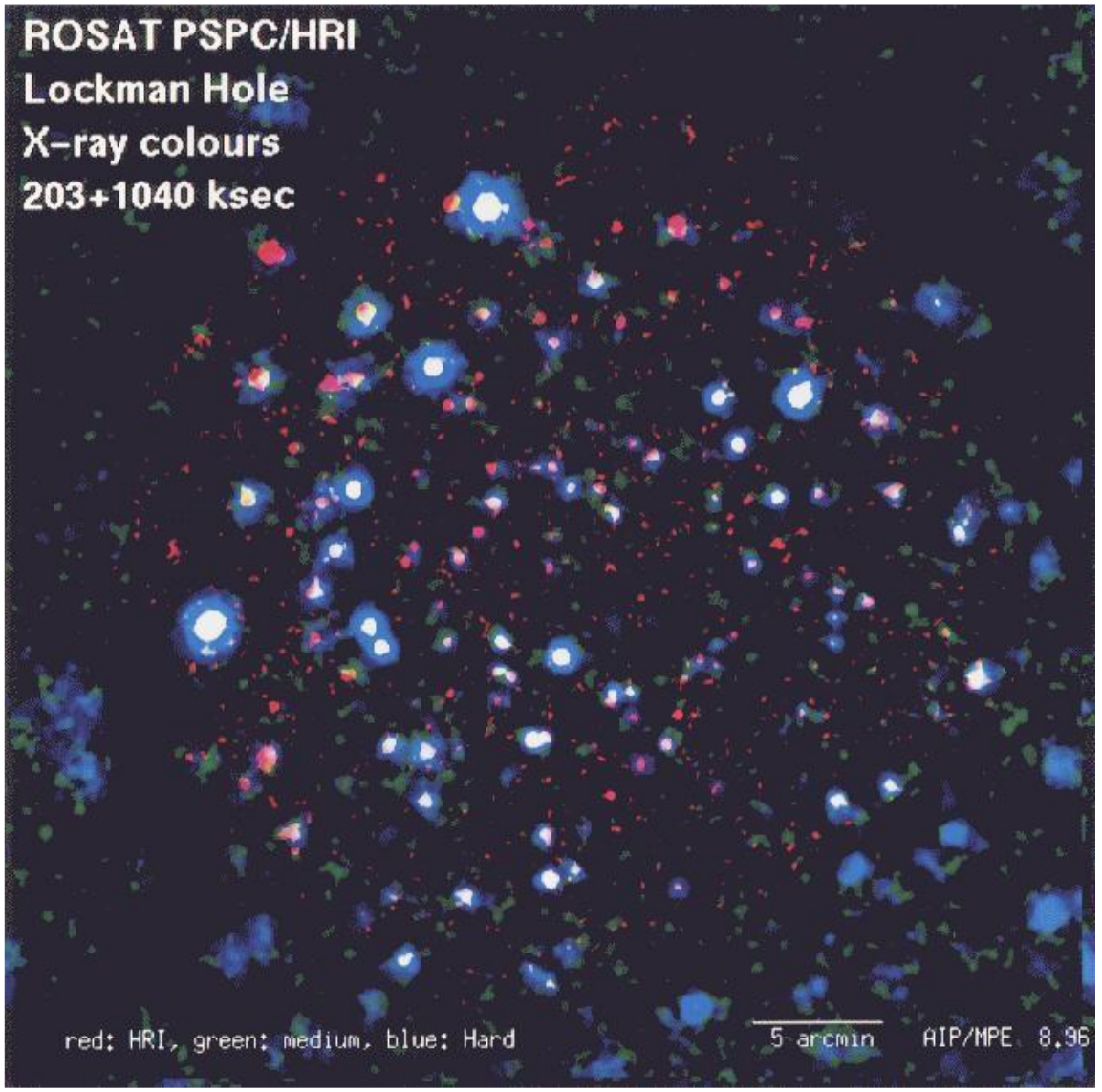


Figure6:X-rayimageof“LockmanHole”

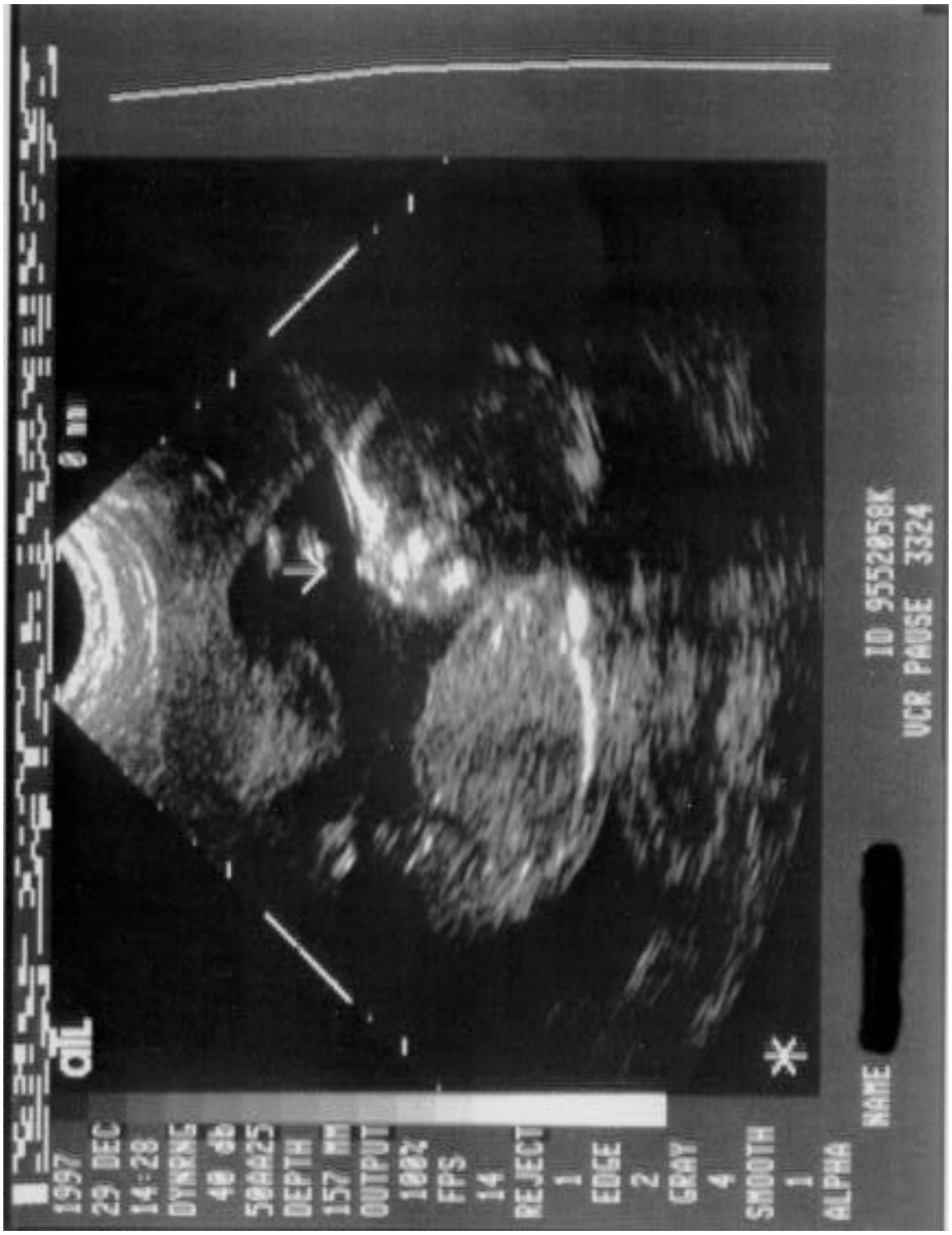


Figure7:Ultrasonicimageofahumanembryo

Figure8:Depthimageofin-doorscene

Figure9:“Stanley”(left)and“Highlander”(right)withsensors(Source:DARPA)

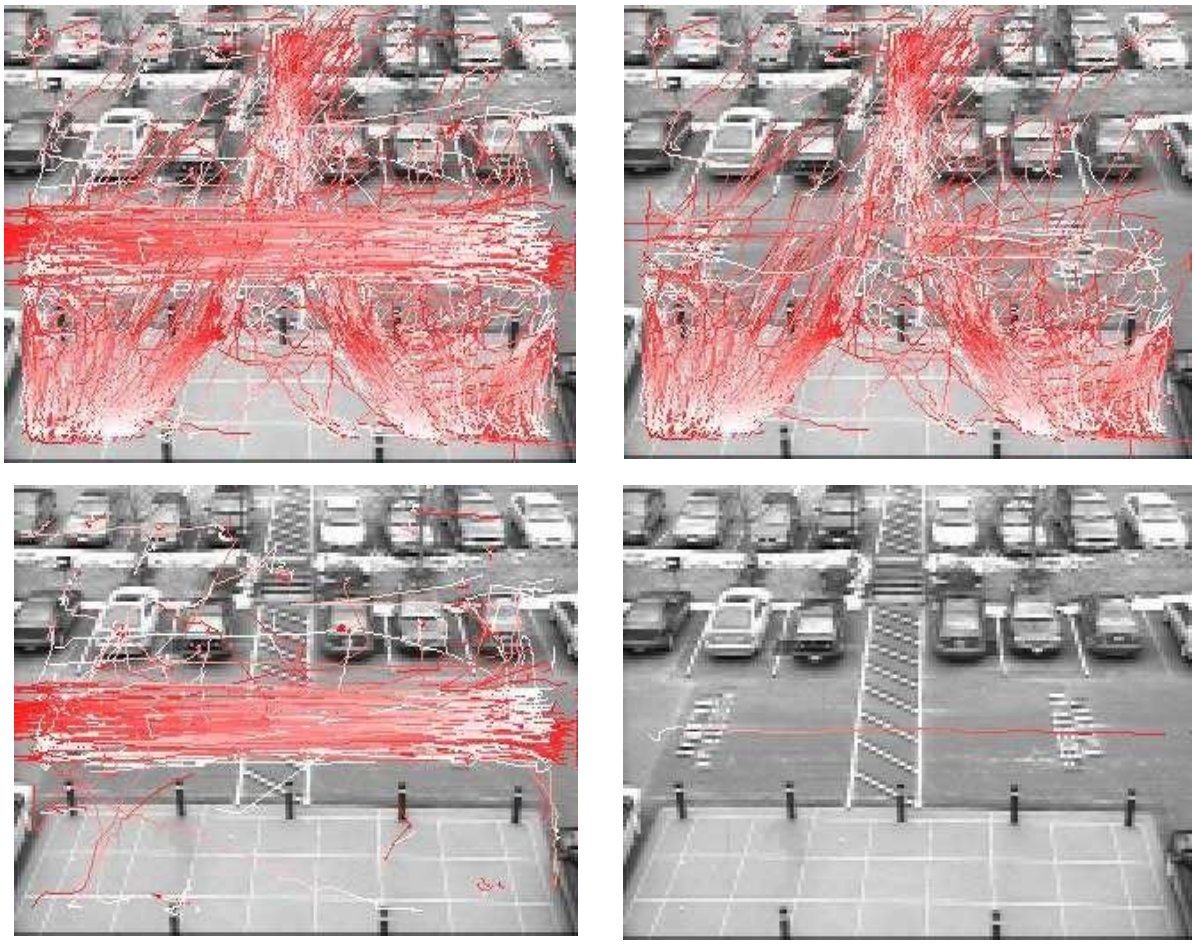


Figure 10: Surveillance of persons arriving at the parking lot bevore IBM's J. T. Watson research center

Figure11:Exampleoffacedetectionresults(byYannLeCun)

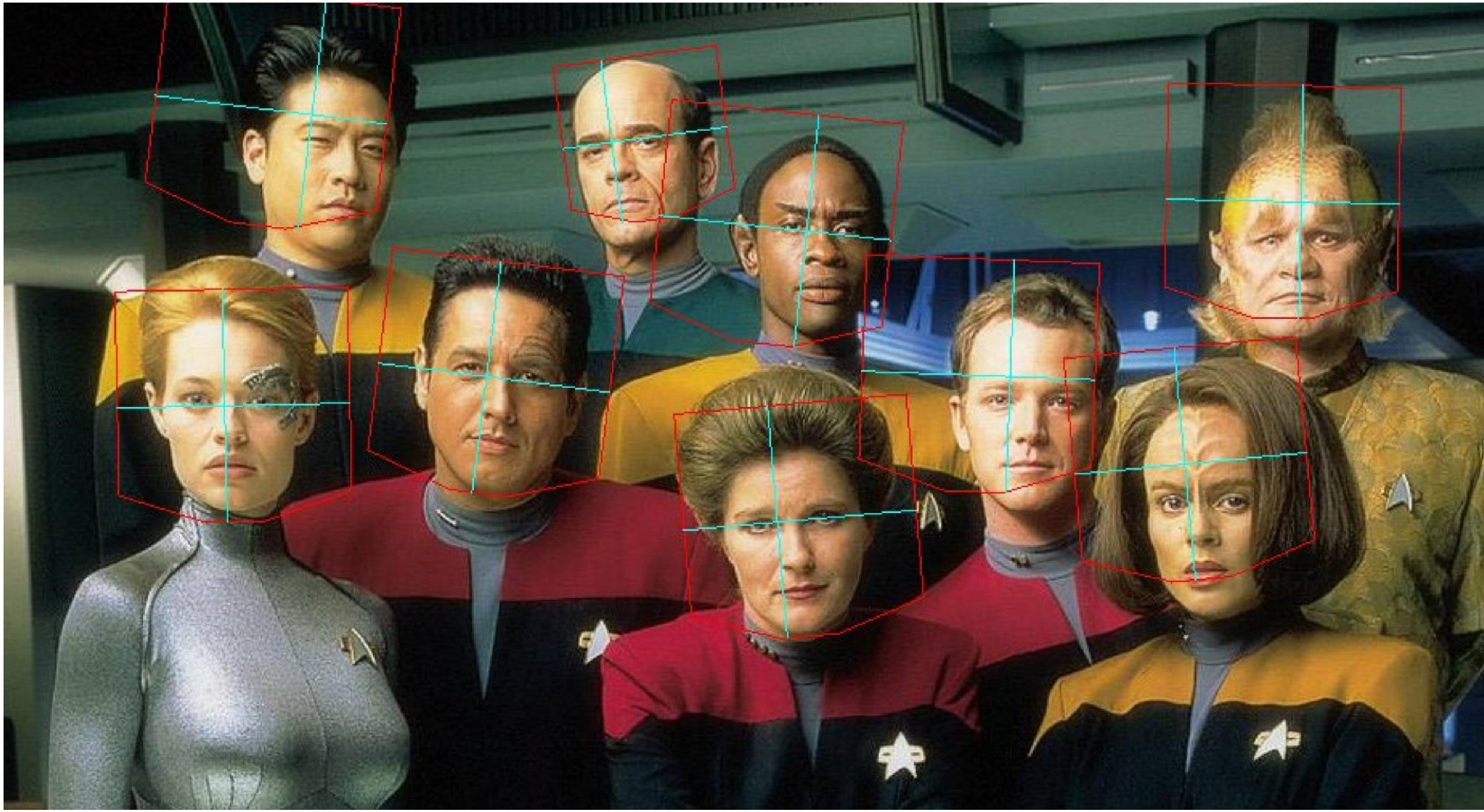


Figure12:Exampleoffacedetectionresultsforstrangepeople(byYannLeCun)

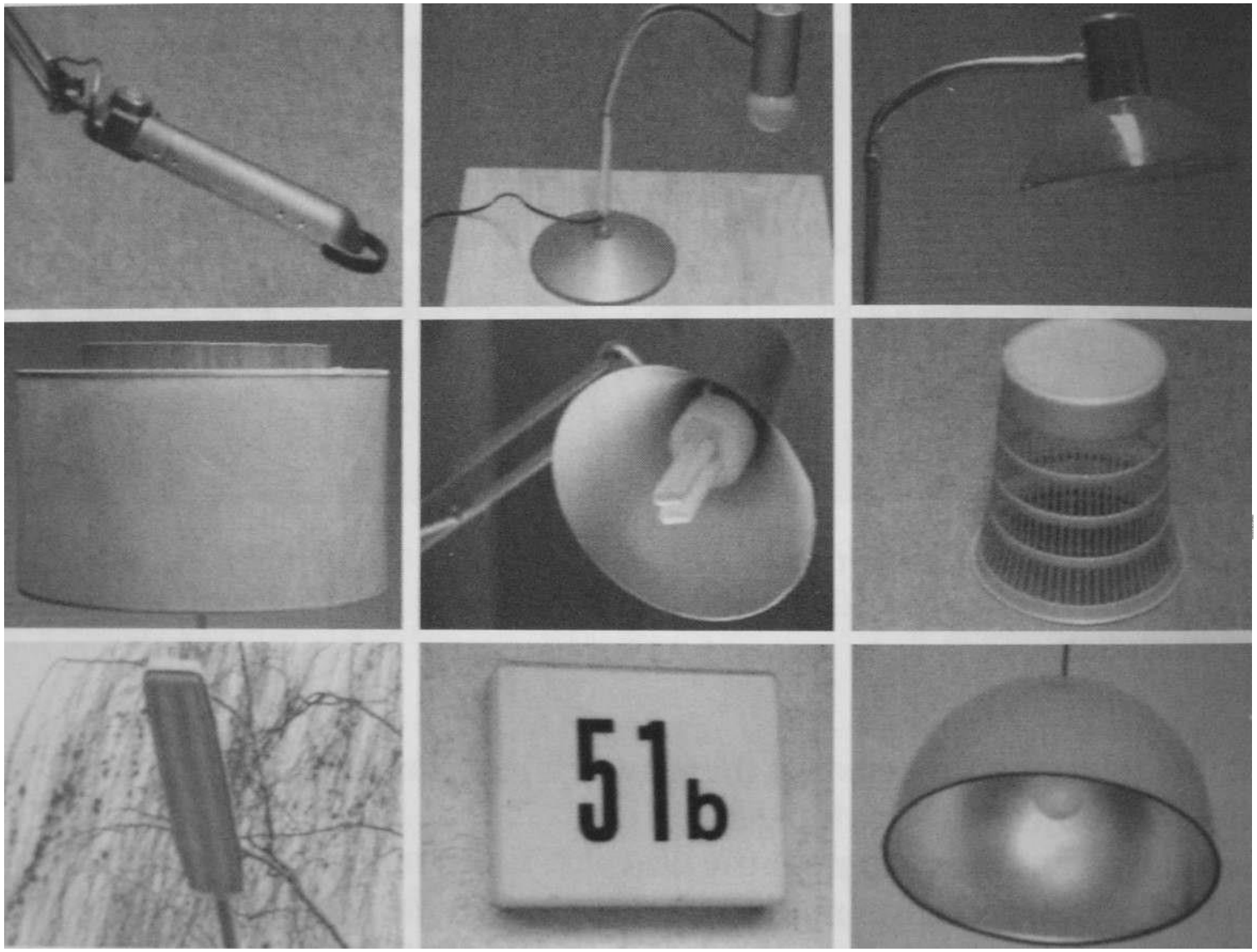


Figure13:Whatdoalltheseobjects–exceptone–haveincommon?[J äh02,p.17]

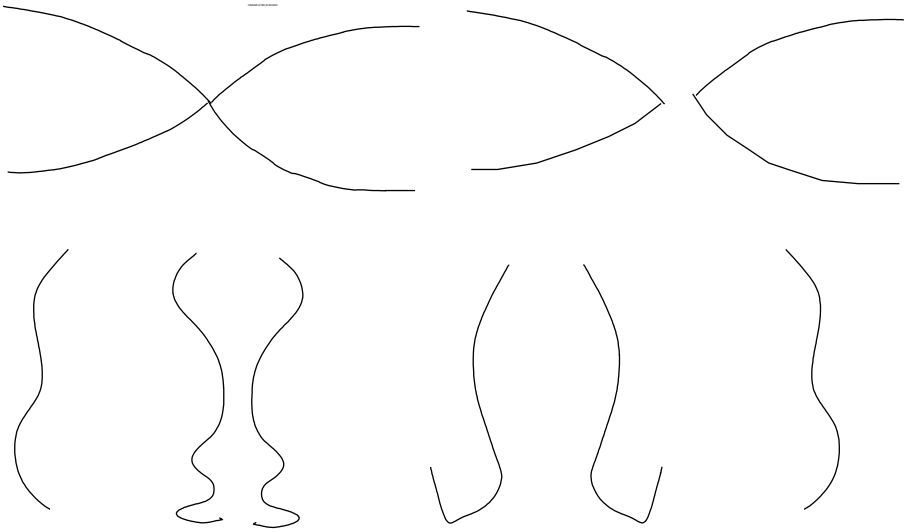


Figure-Ground Separation

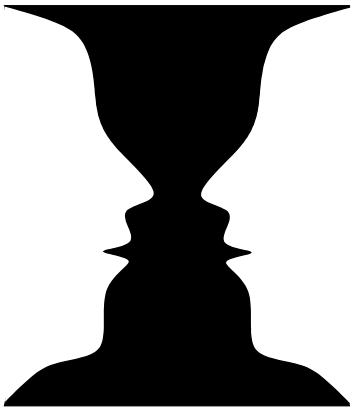
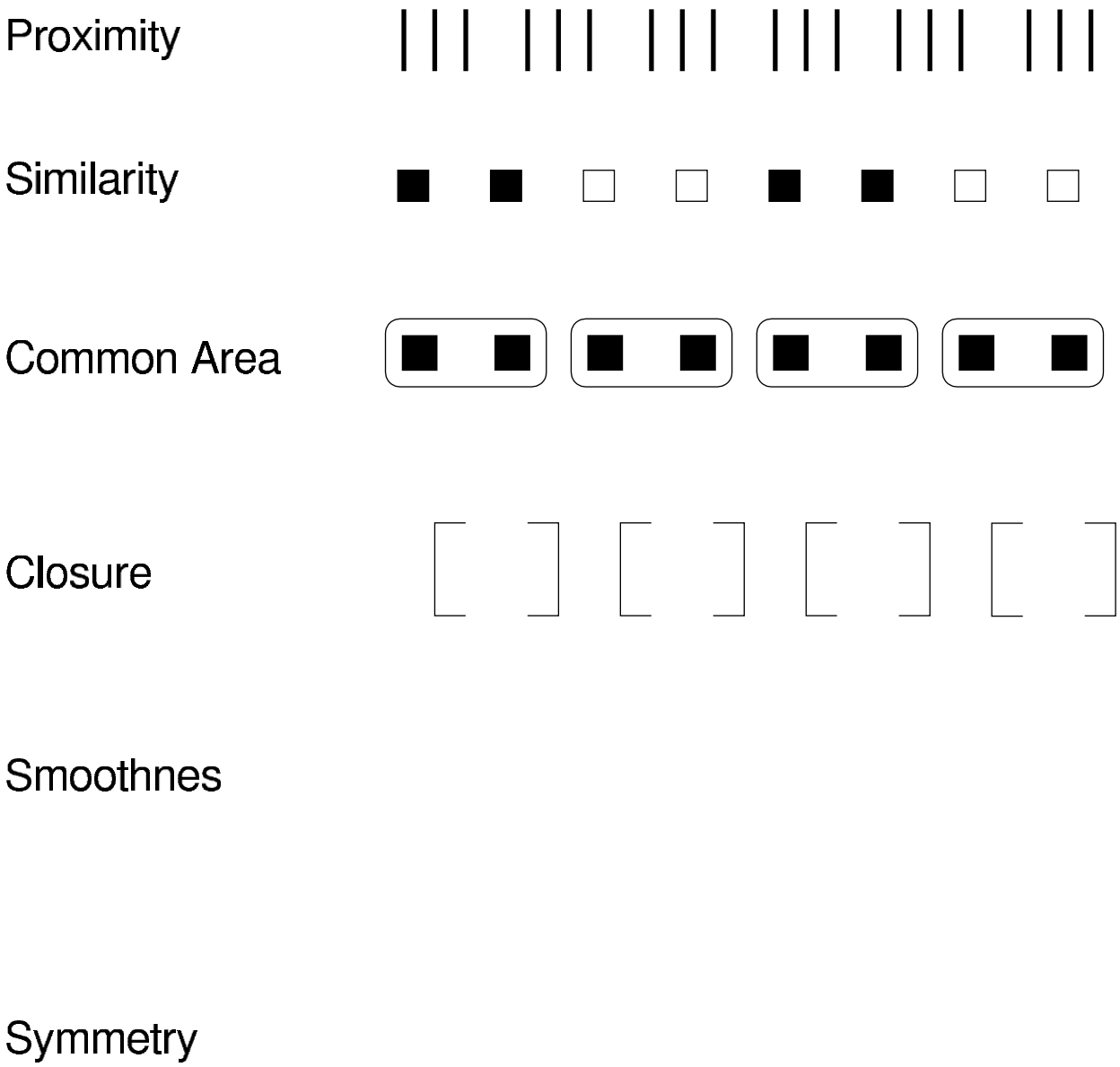


Figure 14: Gestalt laws (after [Sch01])



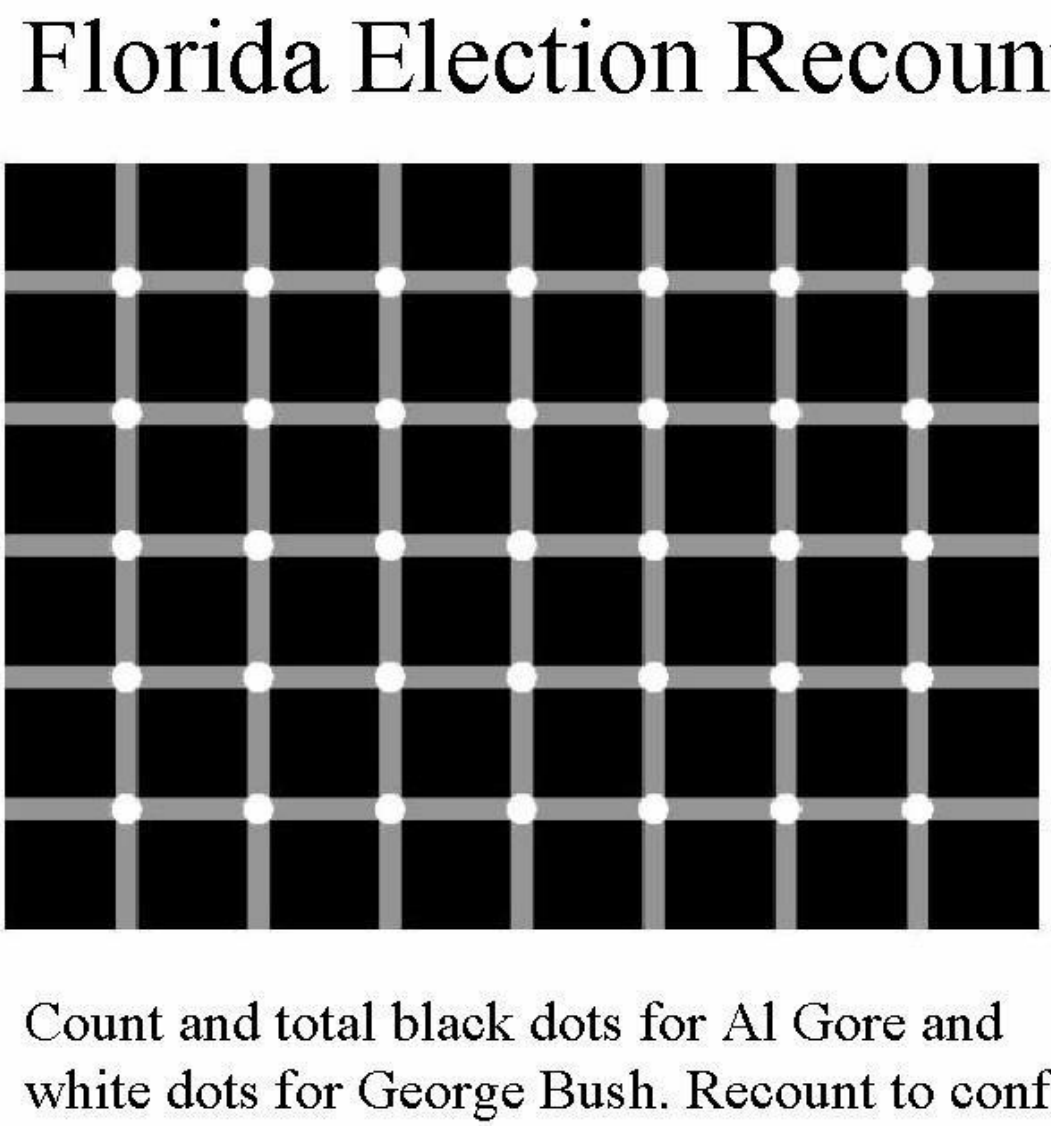


Figure15:Exampleofillusionsinintensityperception

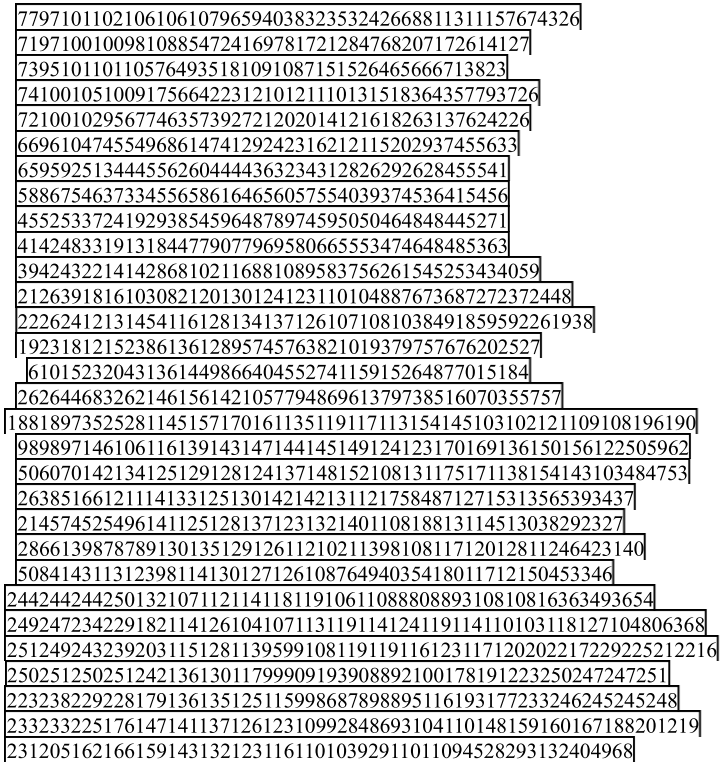


Figure16:“Raw”digitalimageinnumericrepresentation

Figure17:"Raw"digitalimageinnumericandgrey-scalerepresentation

779710110210610610796594038323532426688113115767432671971001009810885472416978172128476820717261412773951011011057649351810910871515264656667138237410010510091756642231210121110131518364357793726721001029567746357392721202014121618263137624226669610474554968614741292423162121152029374556336595925134445562604444363234312826292628455541588675463733455658616465605755403937453641545645522533724192938545964878974595050464848445271414248331913184477907796958066555347464848538339424322141428681021168810895837562615452534340592126391816103082120130124123110104887673687272372448222624121314541161281341371261071081038491859592261938192318121523861361289574576382101937975767620252761015232043136144986640455274115915264877015184262644683262146156142105779486961379738516070355757188189735252811451571701611351191171131545145103102121109108196190989897146106116139143147144145149124123170169136150156122505962506070142134125129128124137148152108131175171138154143103484753263851661211141331251301421421311217584871271531356539343721457452549614112512813712313214011081881311451303829232728661398787891301351291261121021139810811712012811246423140508414311312398114130127126108764940354180117121504533462442444425013210711211411811910611088808893108108163634936542492472342291821141261041071131191141241191141101031181271048063682512492432392031151281139599108119119116123117120202217229225212216250251250251242136130117999091939088921001781912232502472472512233822922817913613512511599868789889511619317723324624524524823323322517614711371261231099284869310411014815916016718820121923120516216615914313212311611010392911011094528293132404968

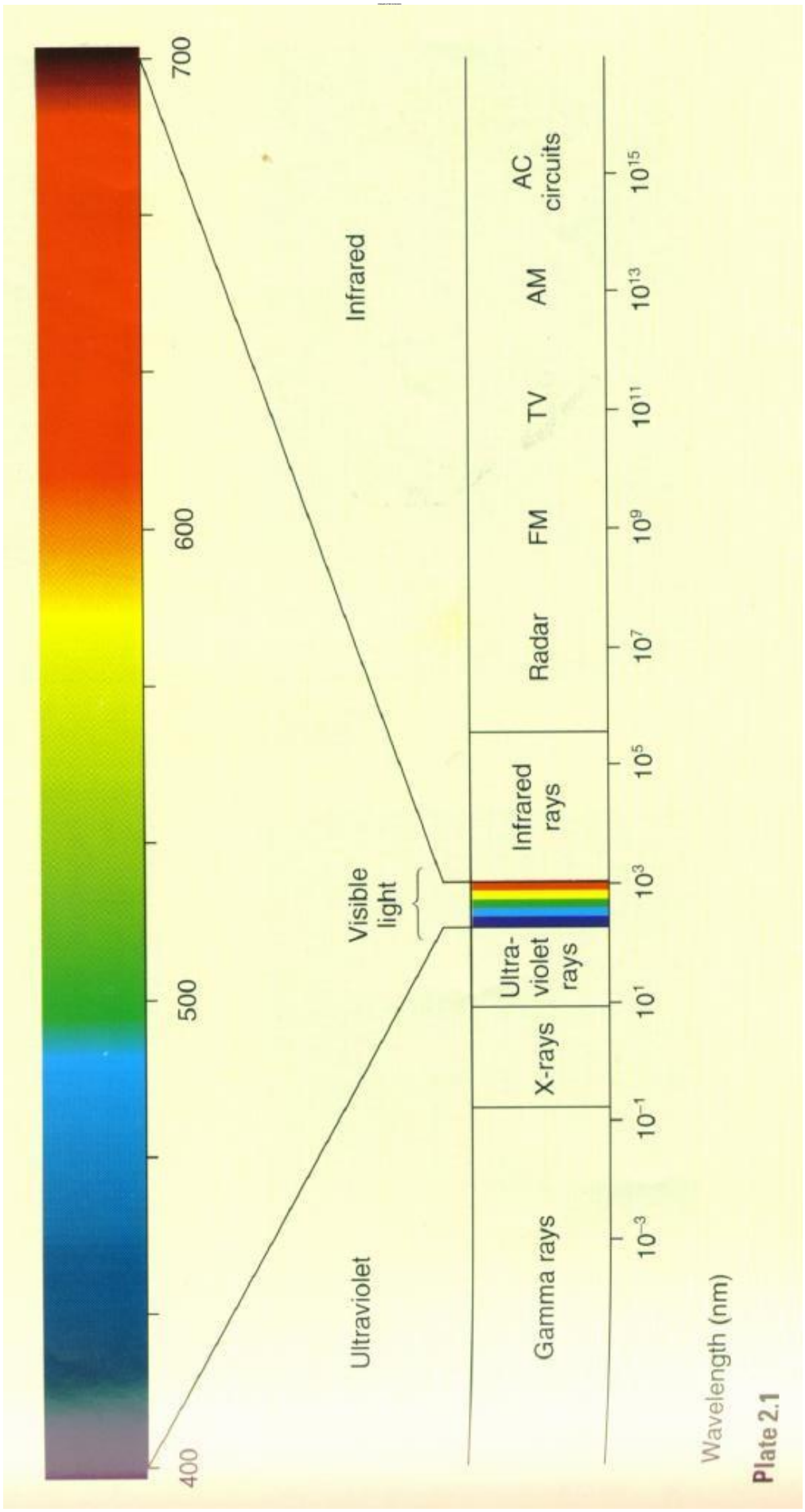


Figure18:Overviewoftheelectromagneticspectrum;rangeofvisiblelightenlarged

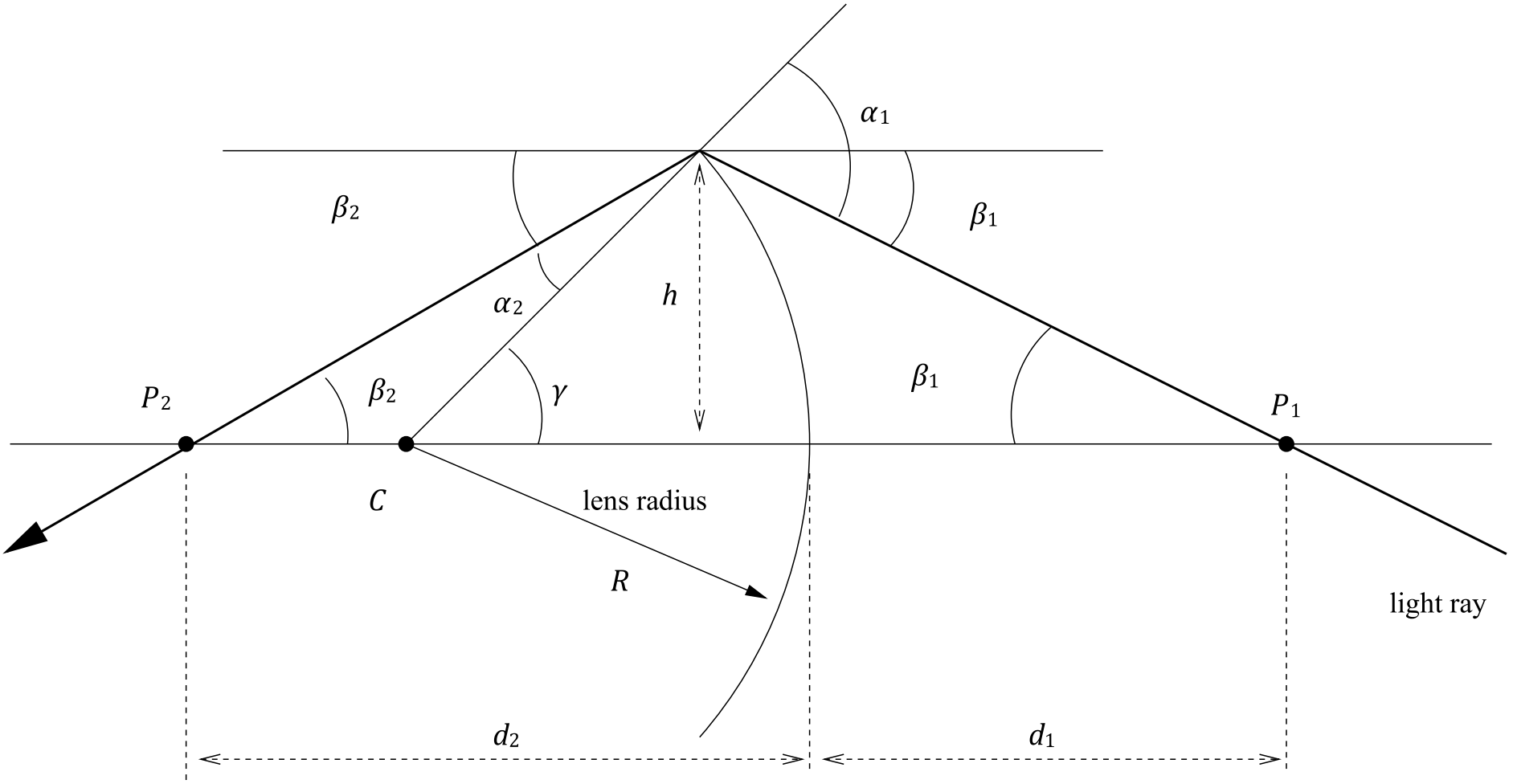


Figure19:Paraxialrefraction:Alightraythrough P_1 isrefractedat P whereitintersectstheinterface,i.e.thesurfaceofthelens)andthenintersects theopticalaxisat P_2.Thegeometriccenteroftheinterfaceis C,itsradius R,allanglesareassumedsmall(after[For03,Fig.1.8,p.9]).

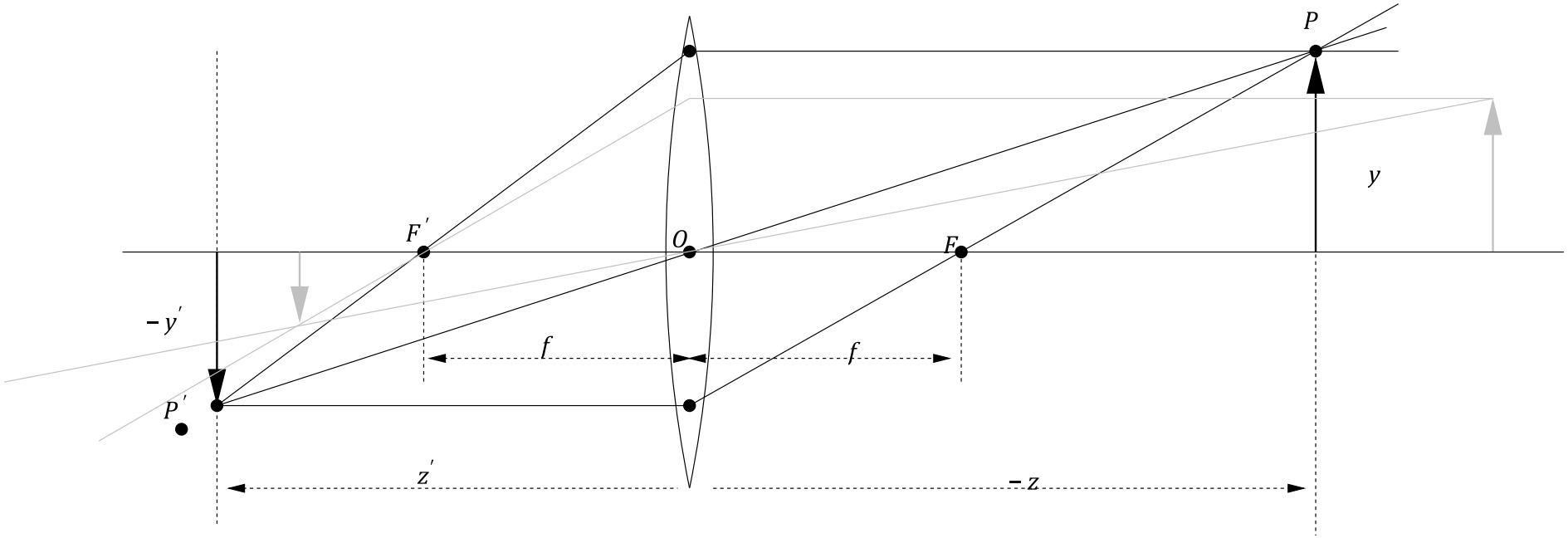


Figure20:Athinlens:Raysthrough O arenotrefracted,raysparalleltotheopticalaxisarefocusedin F'.Alsonotethedifferentin-focusimage pointsforobjectpointsatdifferentdistances(cf.[For03,Fig.1.9,p.10]).

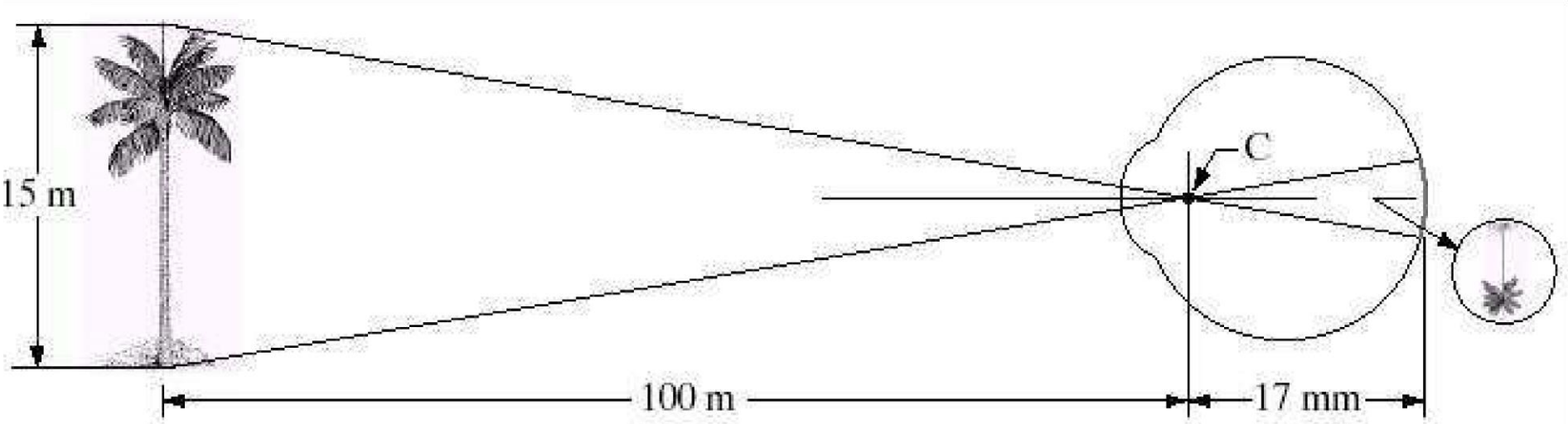


Figure21:ThehumaneyeasanimagingsystemSchematicstructureofthehumaneye(from[Gon02,Chap.2])

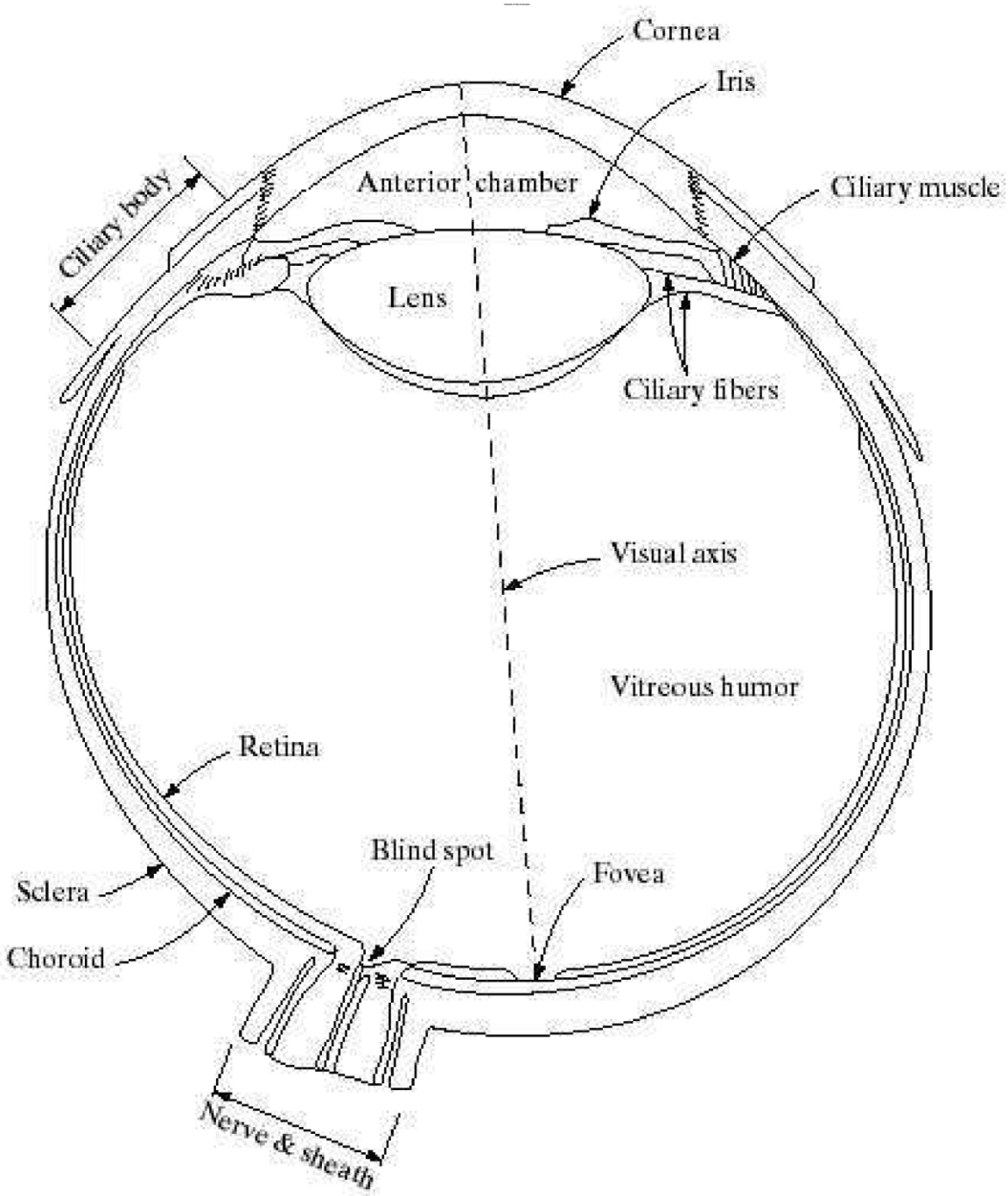


Figure 22: Schematic structure of the human eye (from [Gon02, Chap. 2])

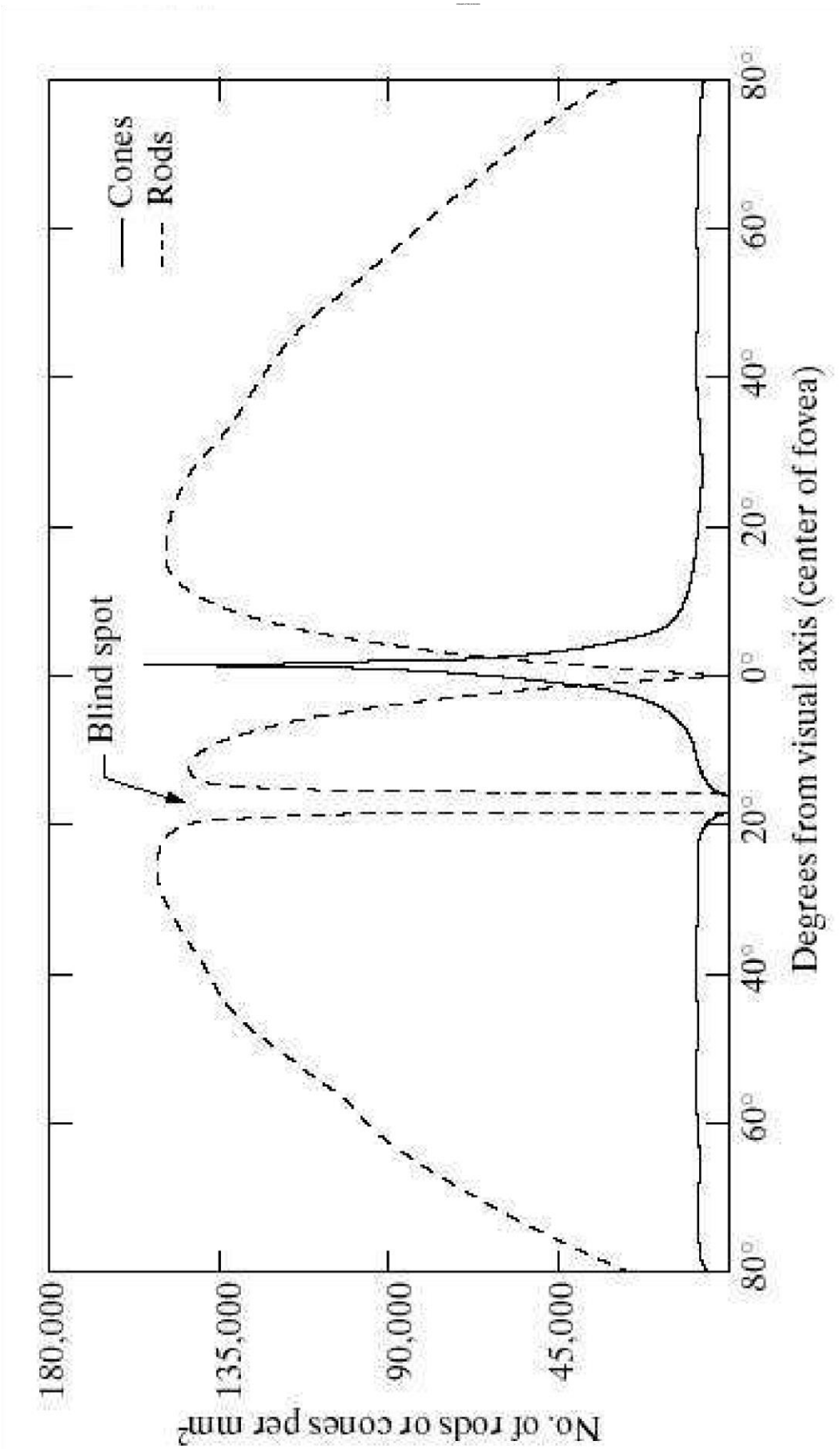


Figure23:Distributionofrodsandconesontheretina(from[Gon02,Chap.2])

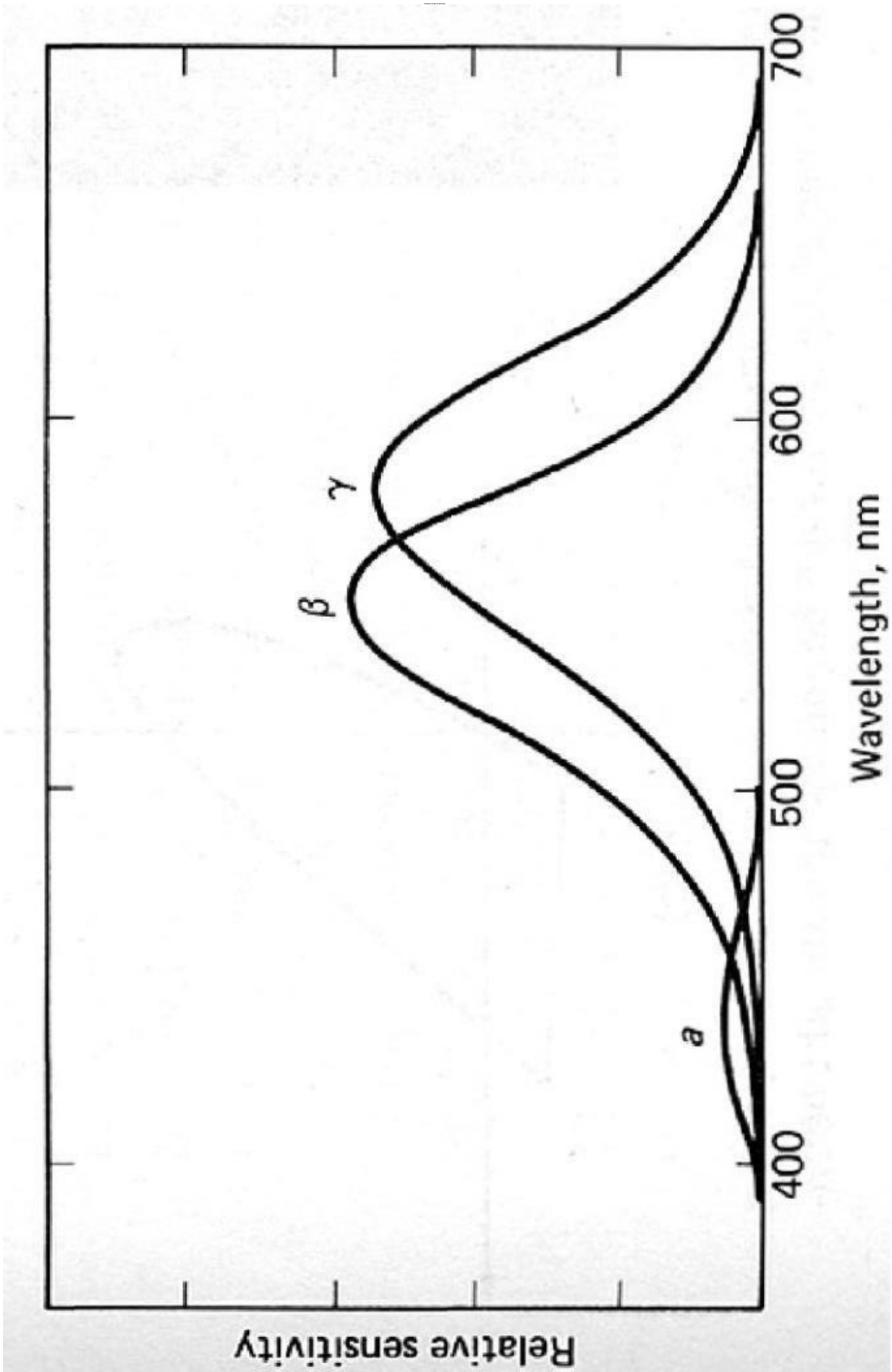


Figure24:Responseofconestodifferentwavelengths(from[Bal82,Chap.2])

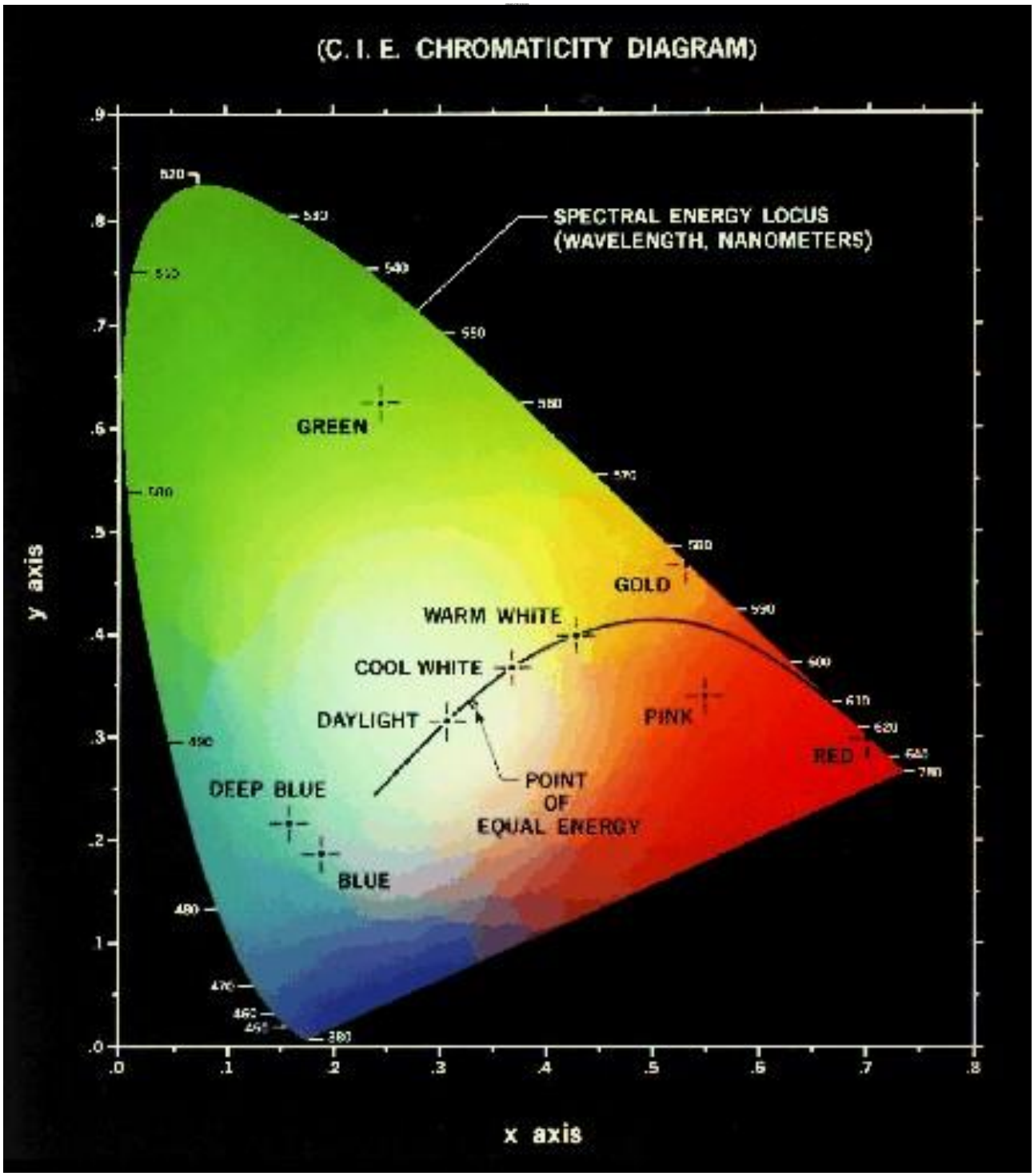


Figure 25: Chromaticity diagram (cf. [Gon02, Chap. 6])

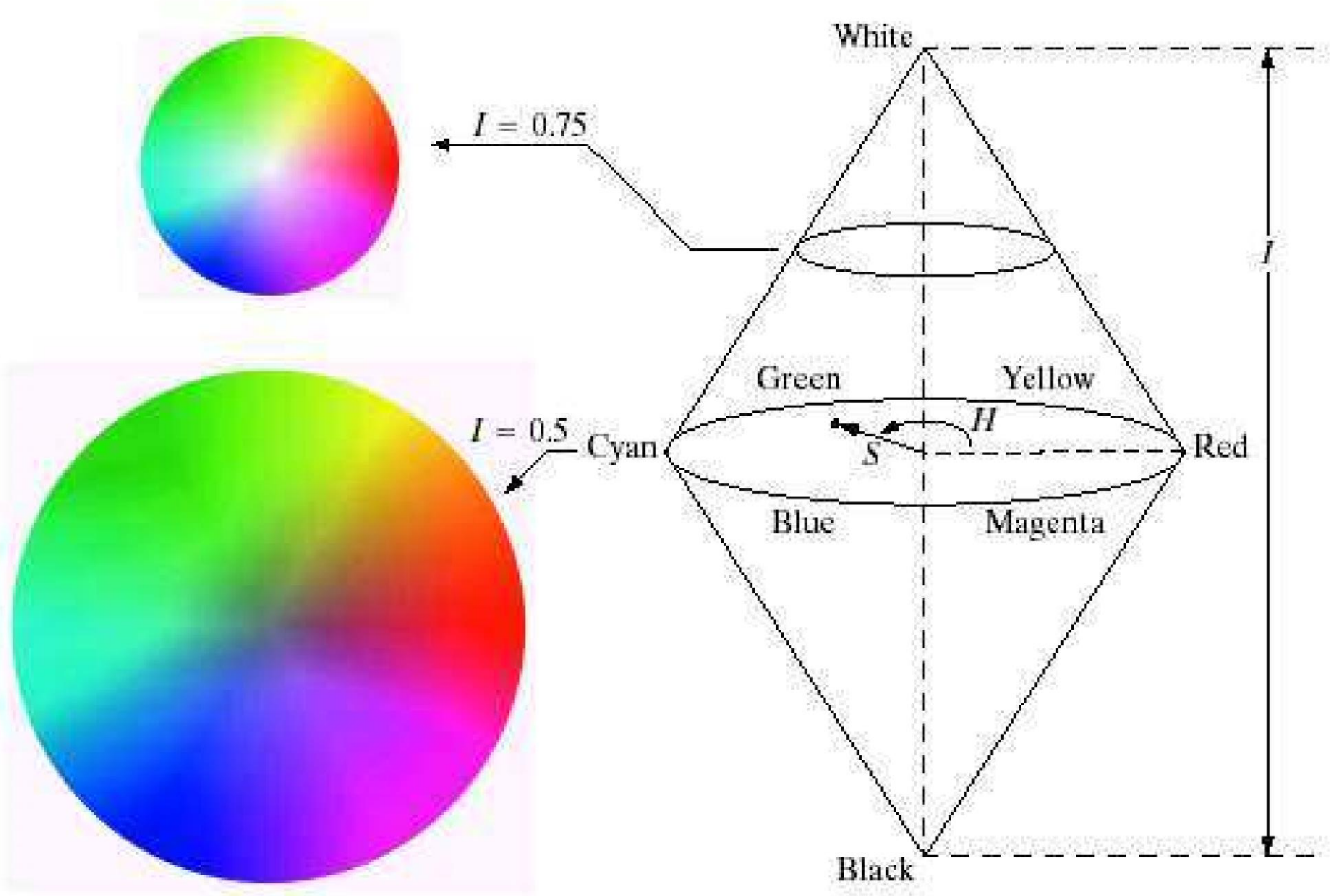


Figure26:HSIcolorspace(from[Gon02,Chap.6])

yellow
red
magenta
green
black
blue
cyan

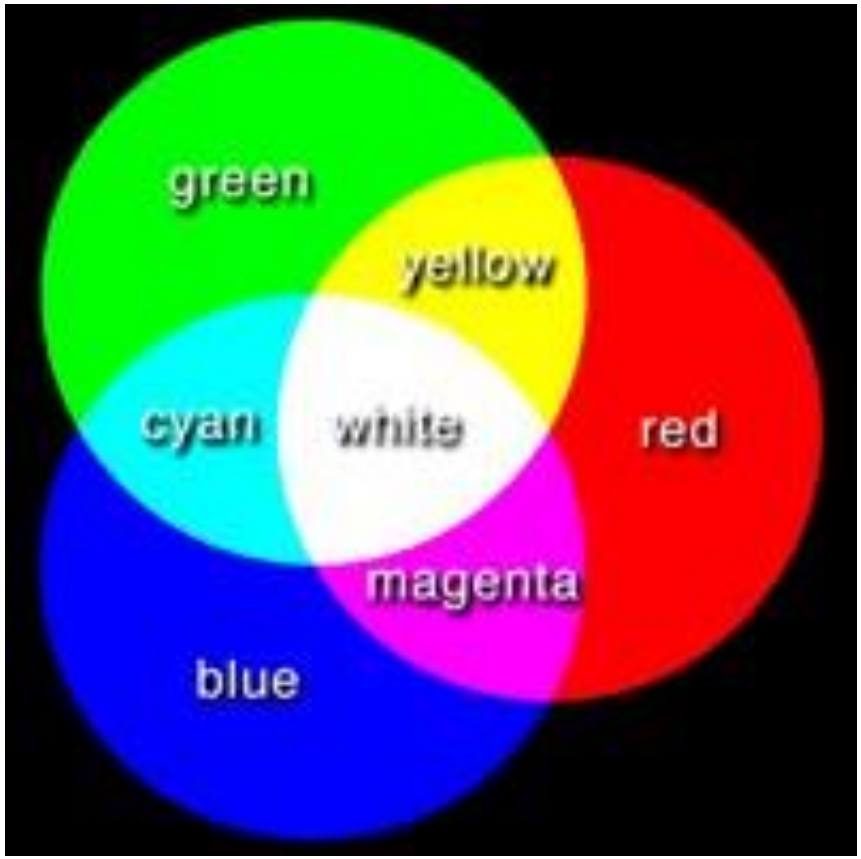
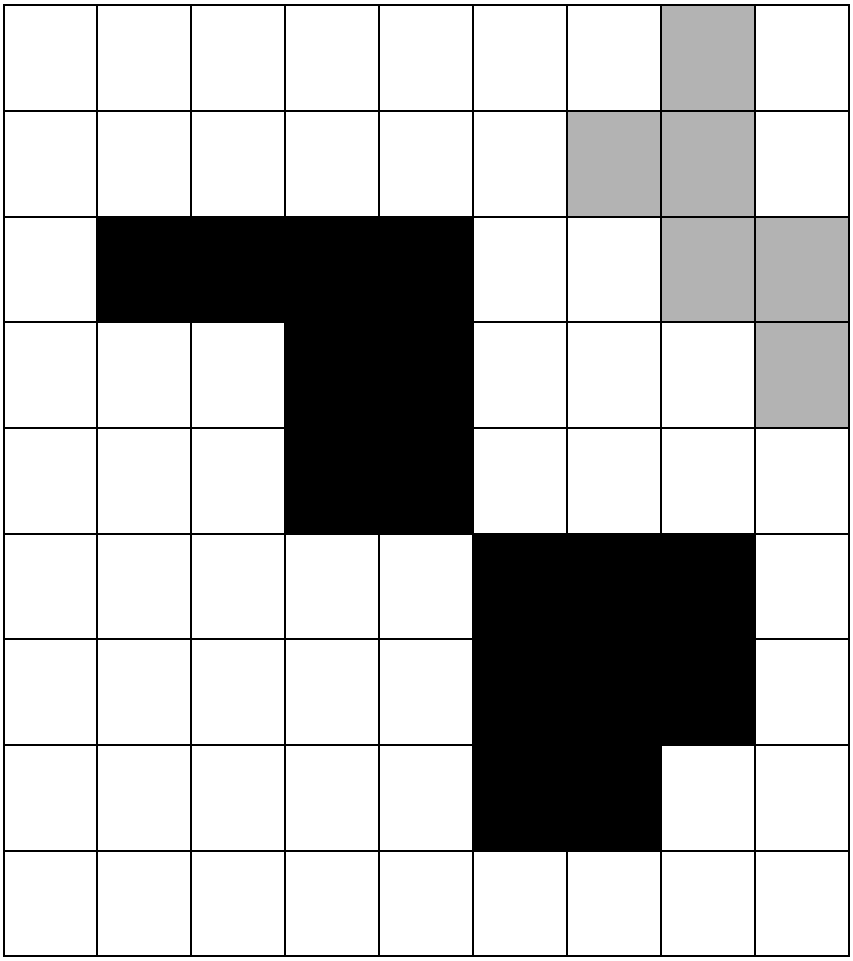
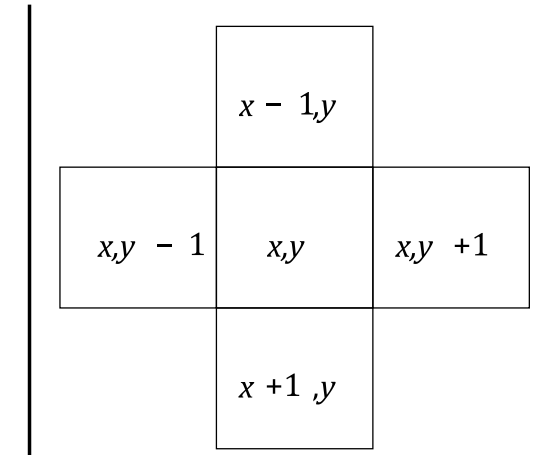


Figure27:AdditivecolormixingintheRGBmodelvs.subtractivecolormixingintheCMYmodel(from)

Figure28:Coordinateconventionusedwithdigitalimages(after[Gon02,Chap.2], Note: Therecoordinateaxesareswapped!)

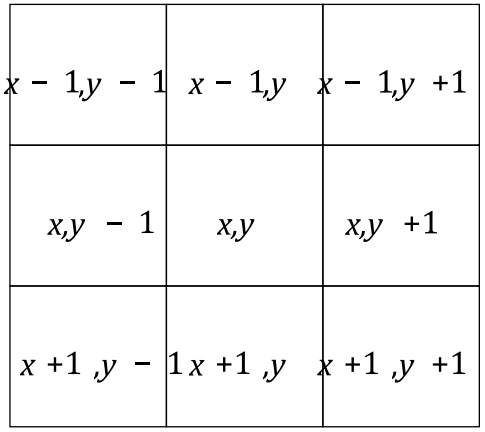
Origin
0 1 2 3
M-1
x
0
1
2
3
N-1
y
f(x,y)
One pixel





Neighborhood

4-



Neighborhood

8-

Figure29:Differentdefinitionsofpixelneighborhoods(left).Which“objects”areconnected?(right)(after[J äh02,p.42])

Row Transfer

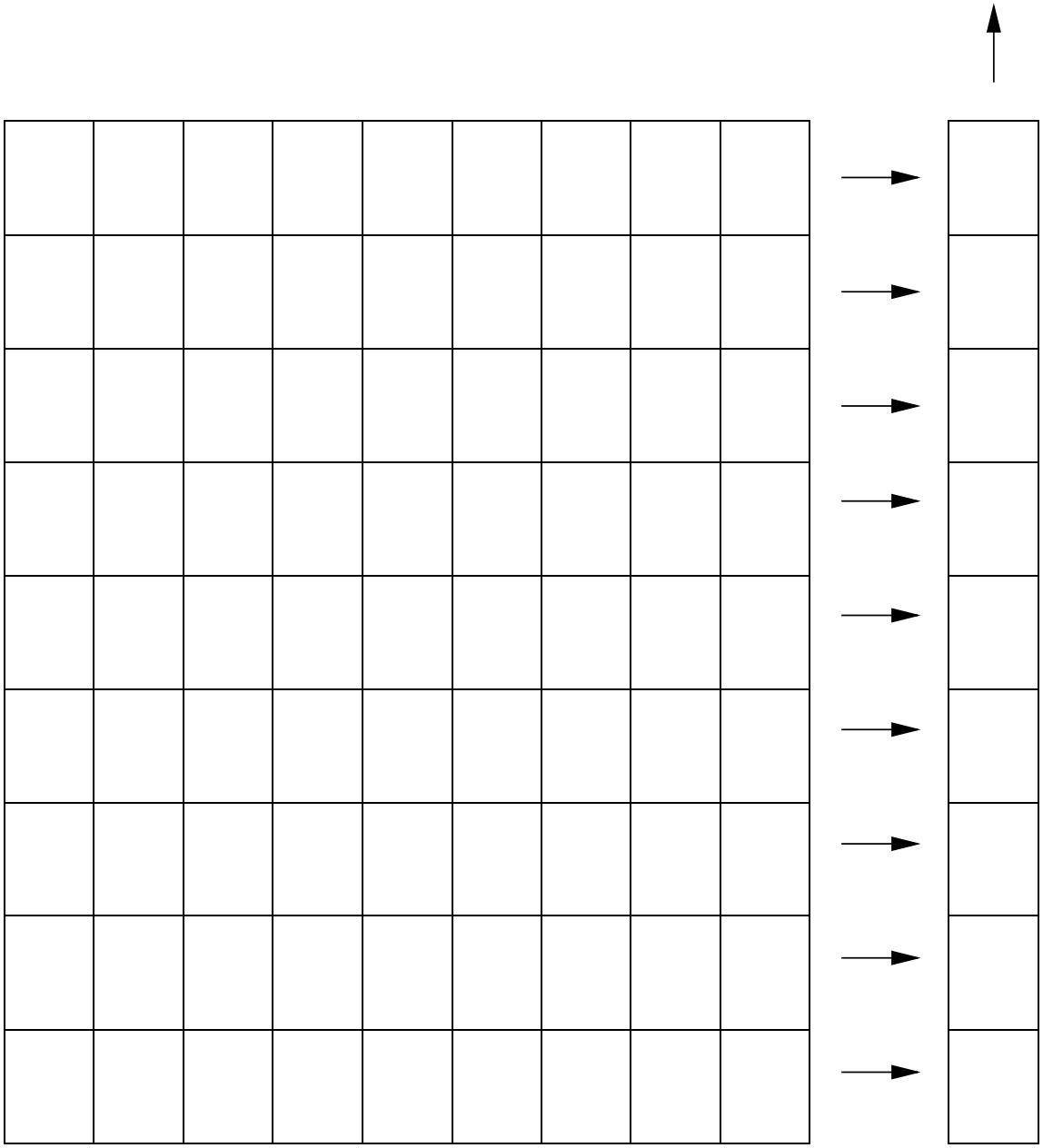


Figure30:StructureofaCCDdevice(after[For03,p.16])

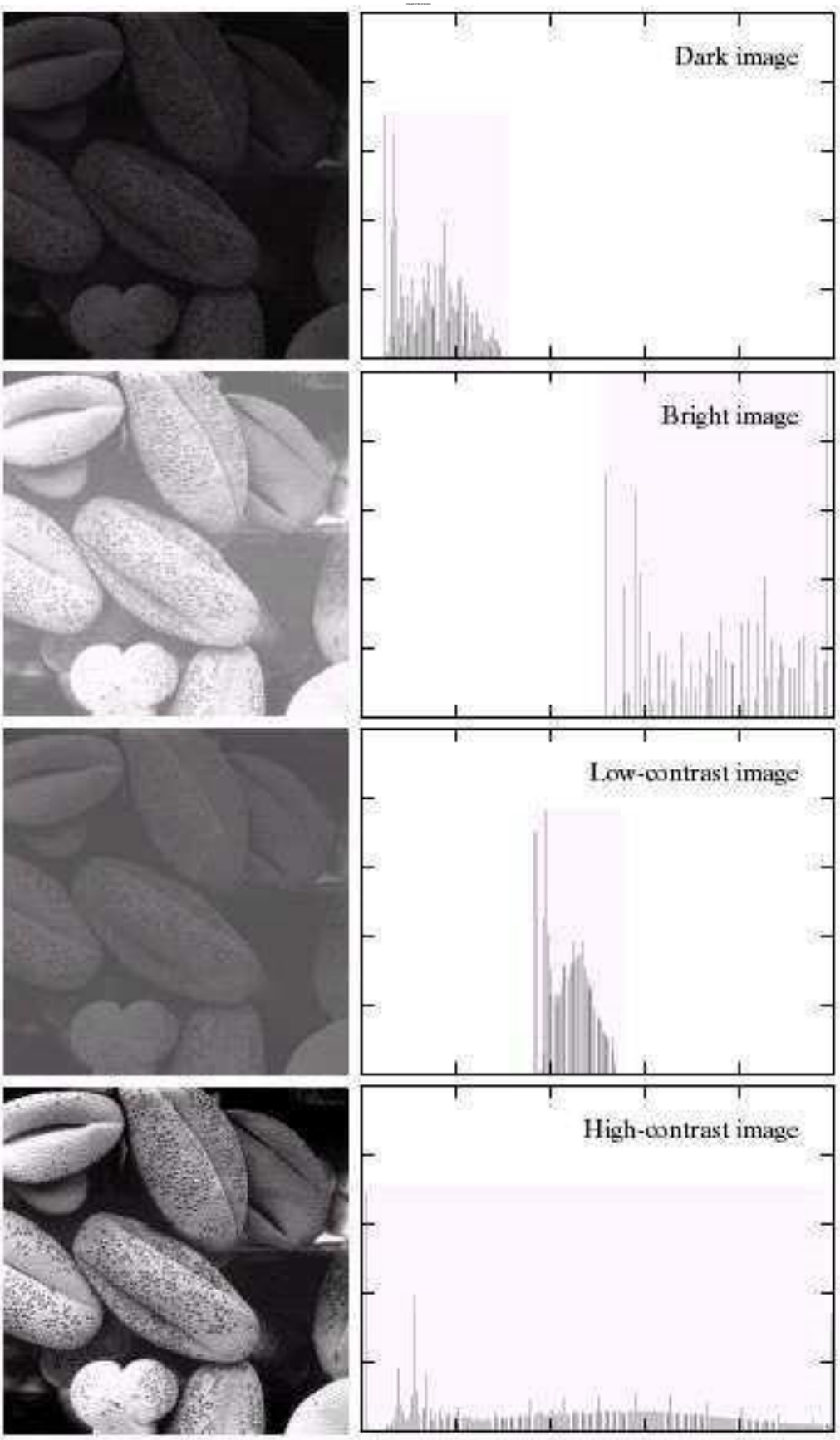


Figure 31: Basic image types: dark, light, low contrast, high contrast with correcponding intensity histograms (from [Gon02, Fig. 3.15, Chap. 3])

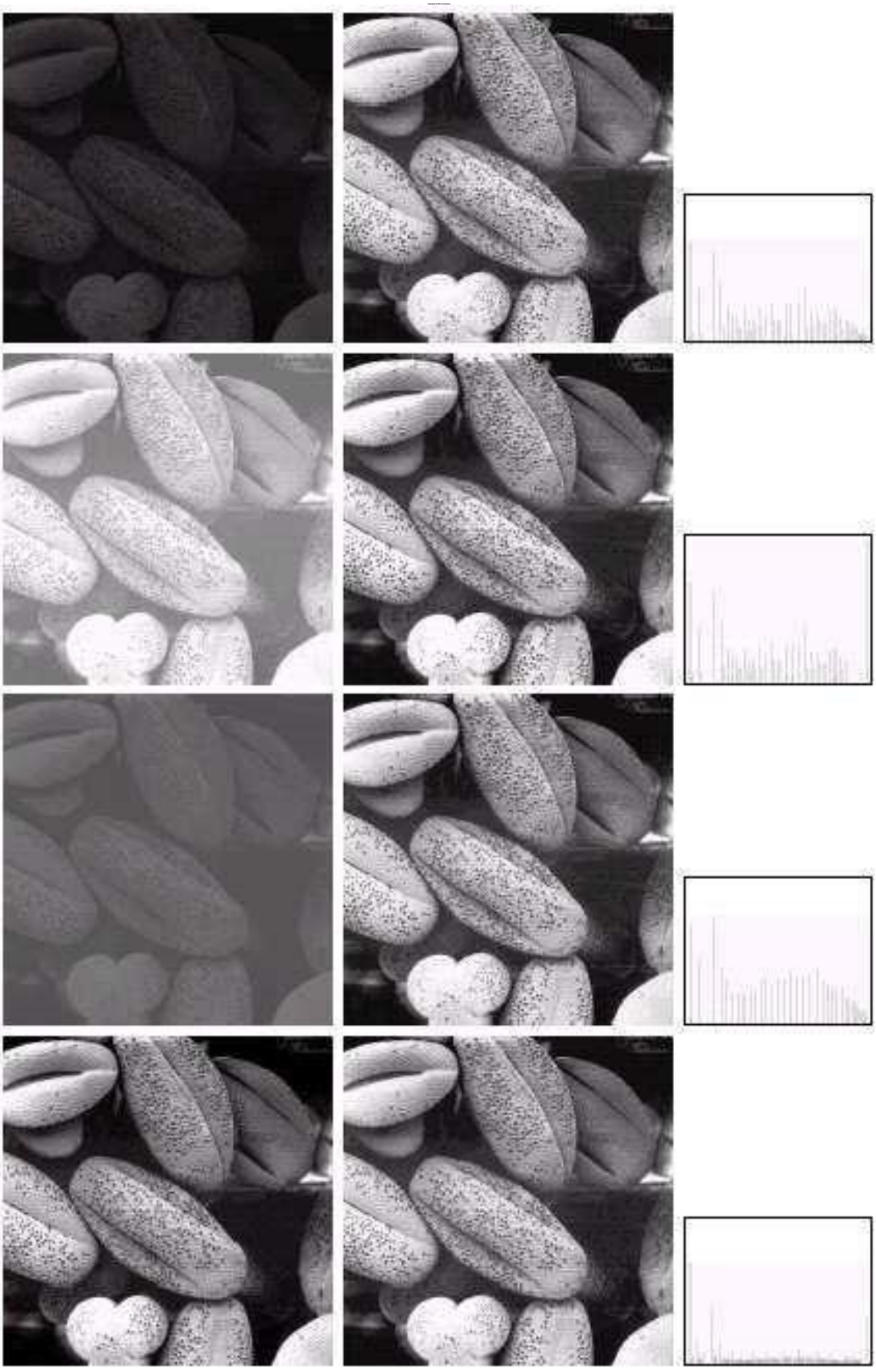
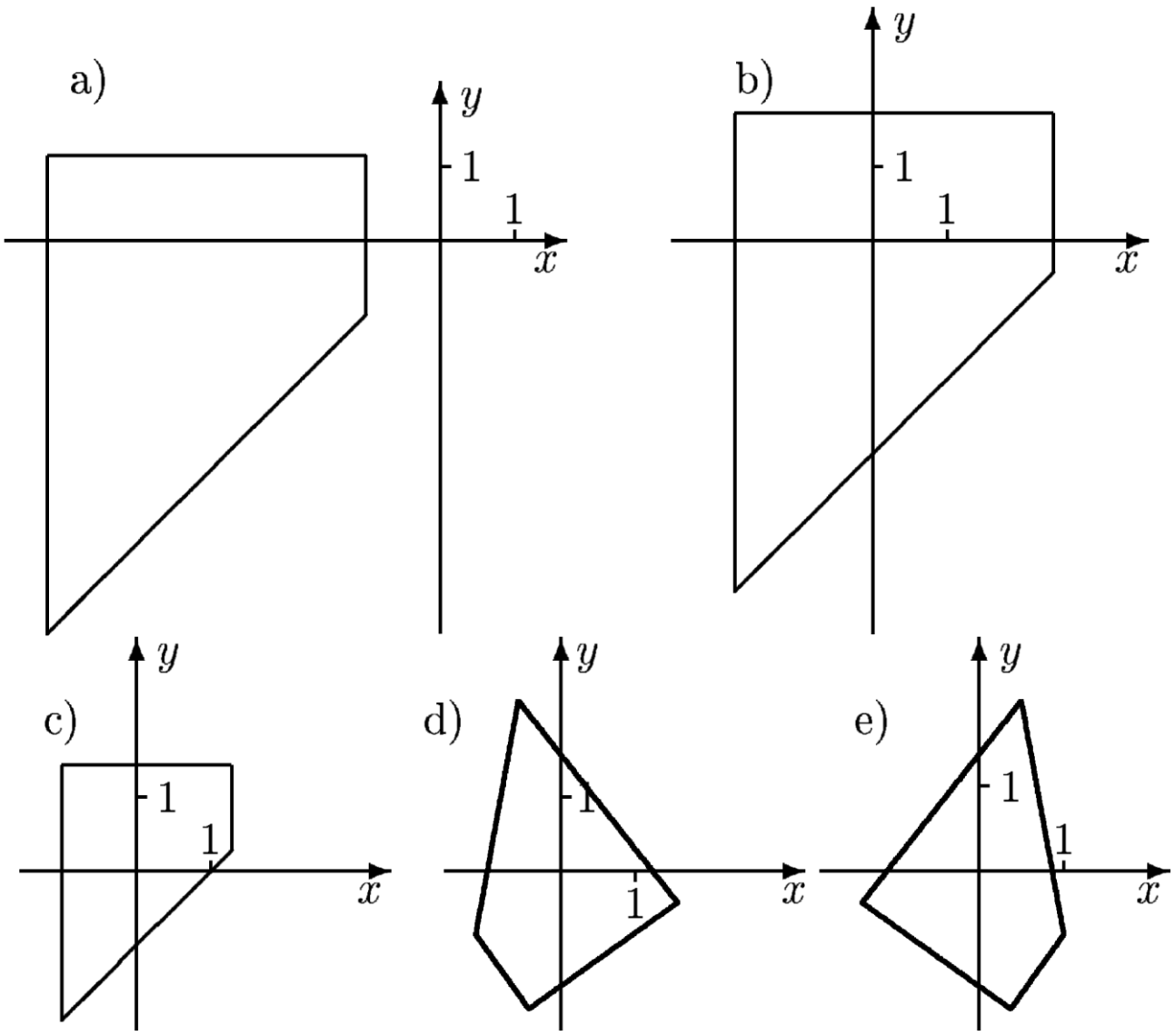
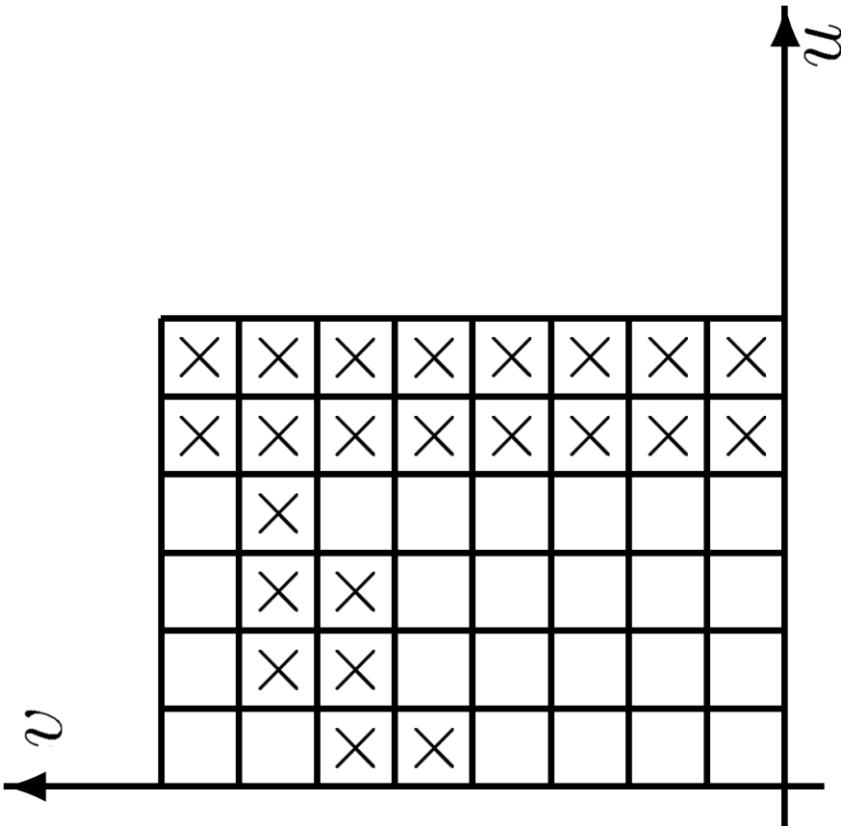


Figure 32: Examples of histogram equalization on images from Fig. 31 with corresponding final intensity histograms (from [Gon02, Fig. 3.17, Chap. 3])

Figure 33: Examples of position/orientation normalization based on normalization of image (grey level) moments (from [Nie03, Fig. 2.5.7, Chap. 2])




u
v

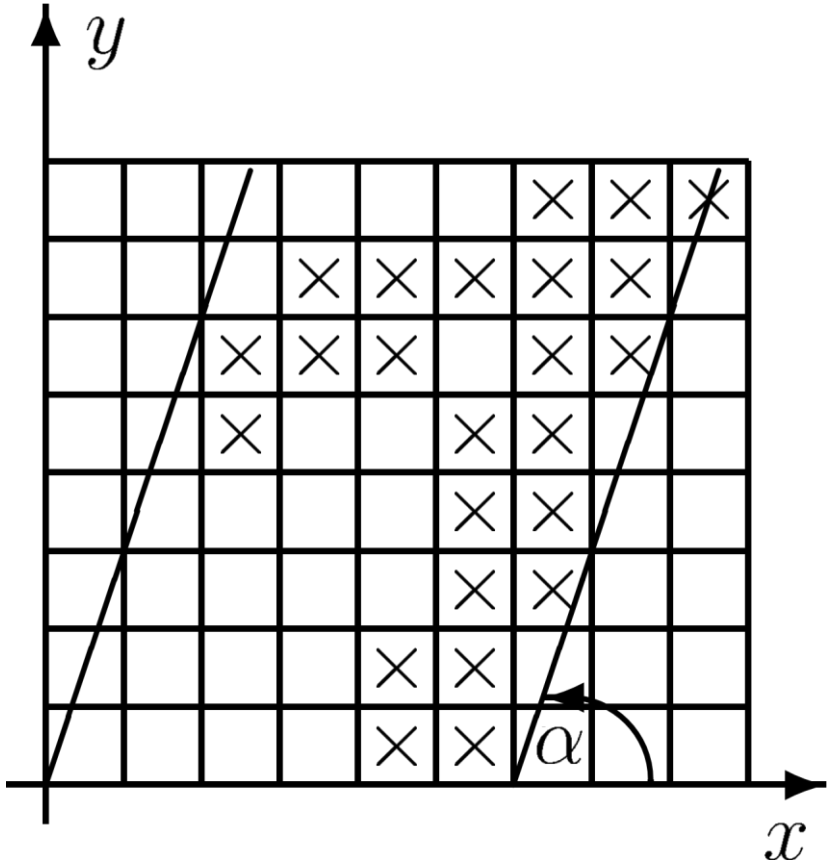


Figure34:Examplenormalisationofcharacterslantbyapplyingashear-transform(from[Nie03,Fig.2.5.8,Chap.2])

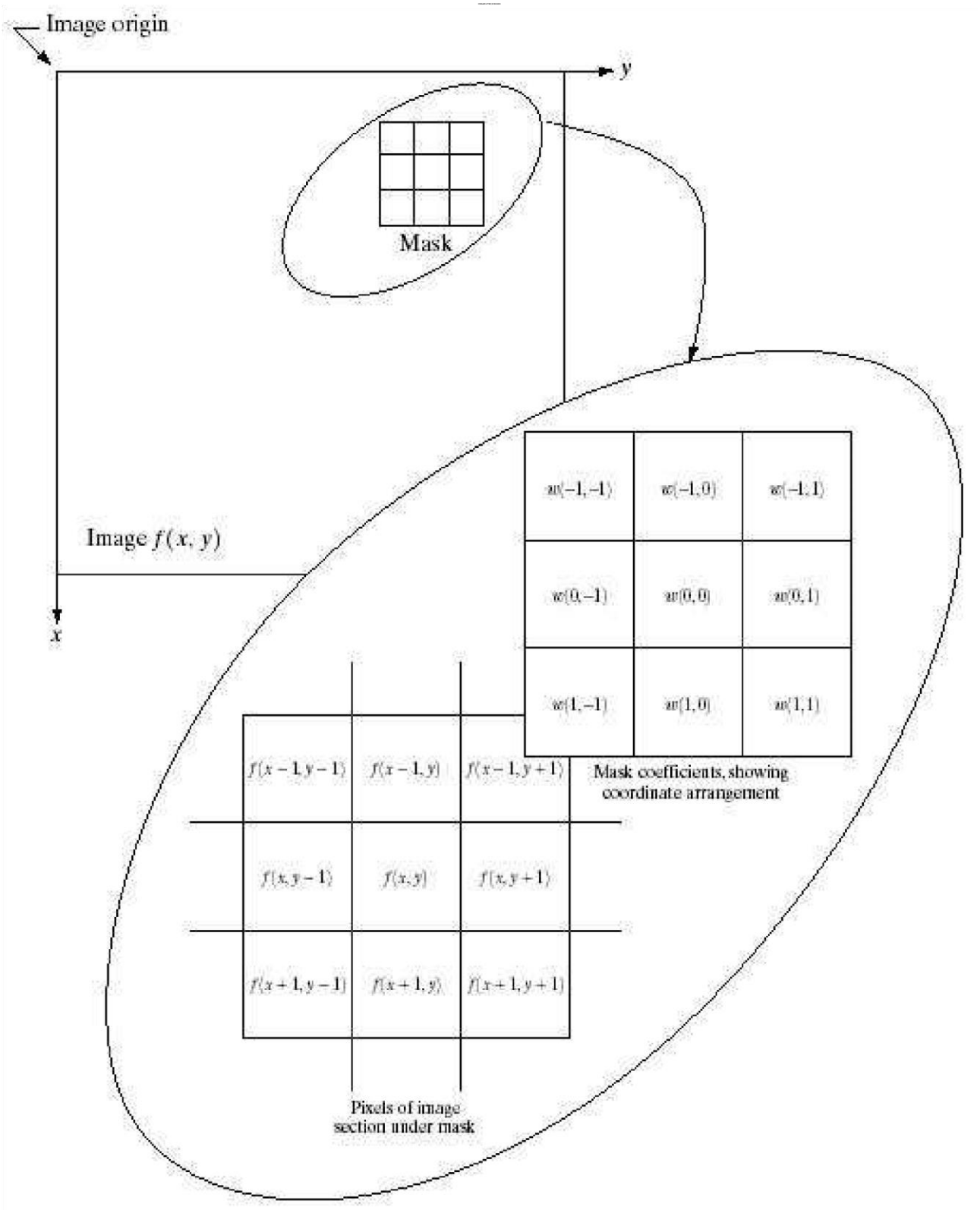


Figure 35: Principle of filtering using masks (from [Gon02, Chap. 3]) Note: Coordinate axes are swapped w.r.t. usual upper-left convention!

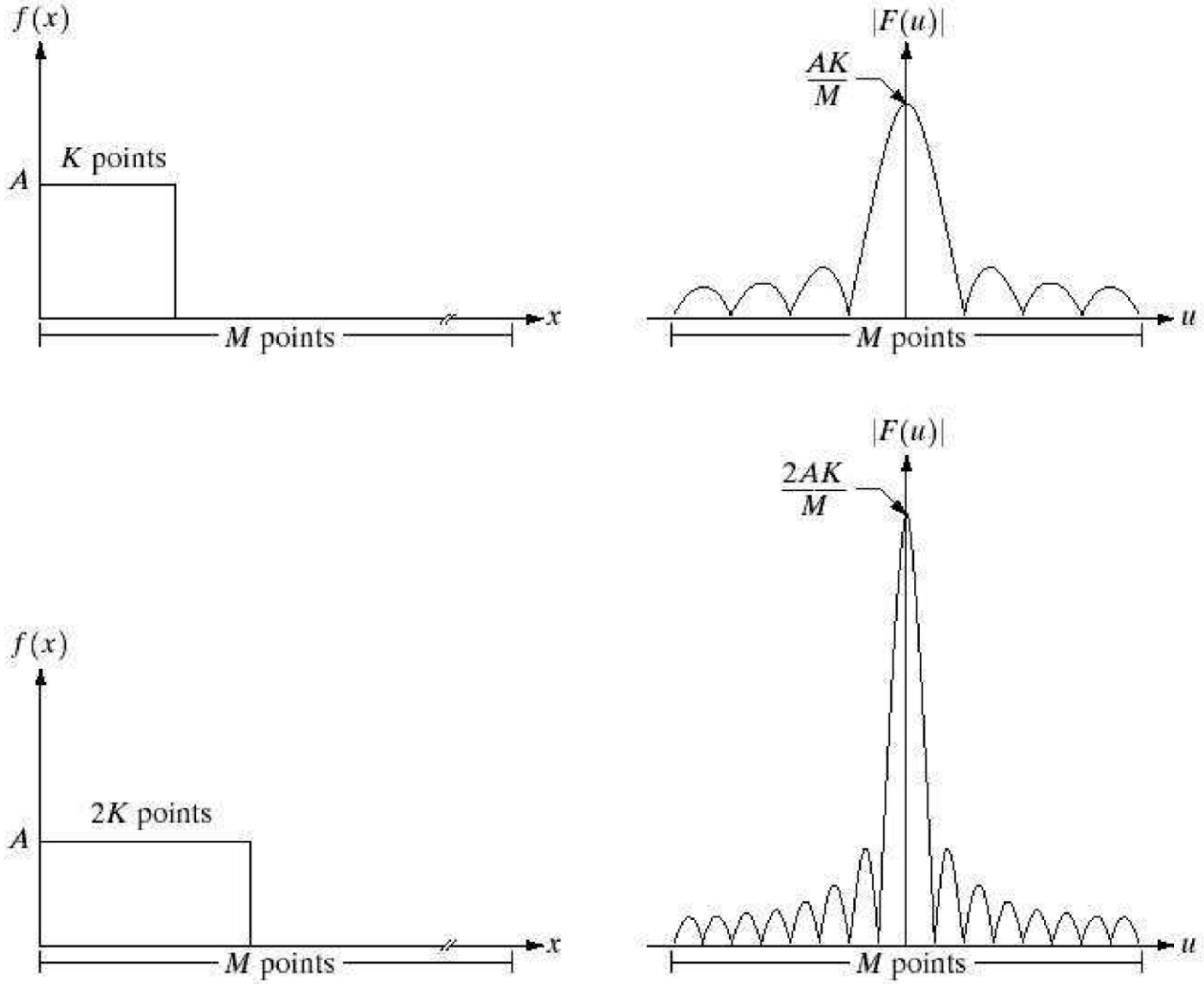
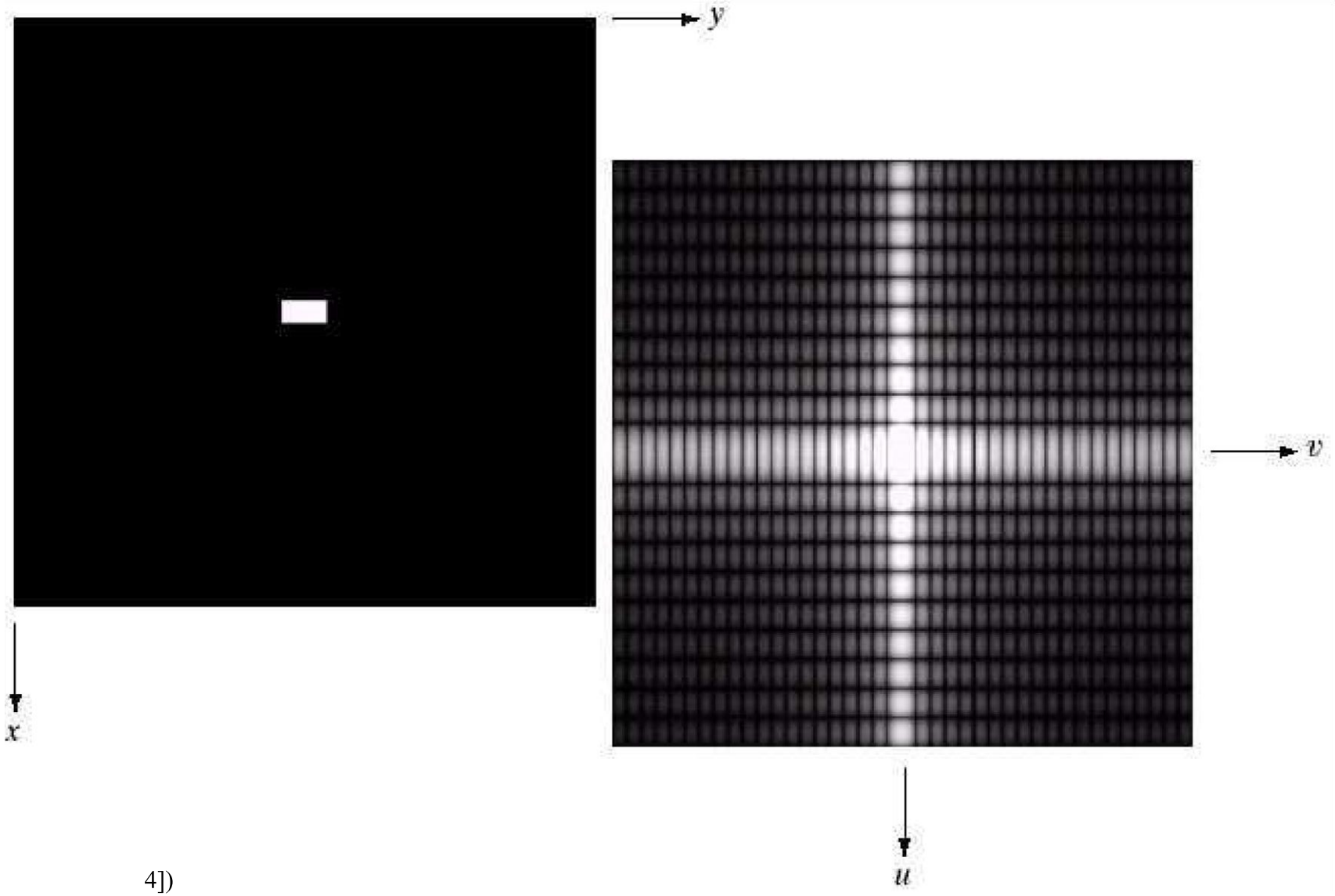


Figure36:Examplesof1Dfunctions $f(x)$ andthecorrespondingpowerspectra $|F(u)|$,i.e.themagnitudeofthe1D-DFT(from[Gon02,Chap.4])



4])

Figure37:Exampleofa 512×512 imagecontainginga 20×40 whiterectangleandtheassociatedcenteredlogarithmicpowerspectrum $\log(1+|F(u,v)|)$ from[Gon02,Chap.

(

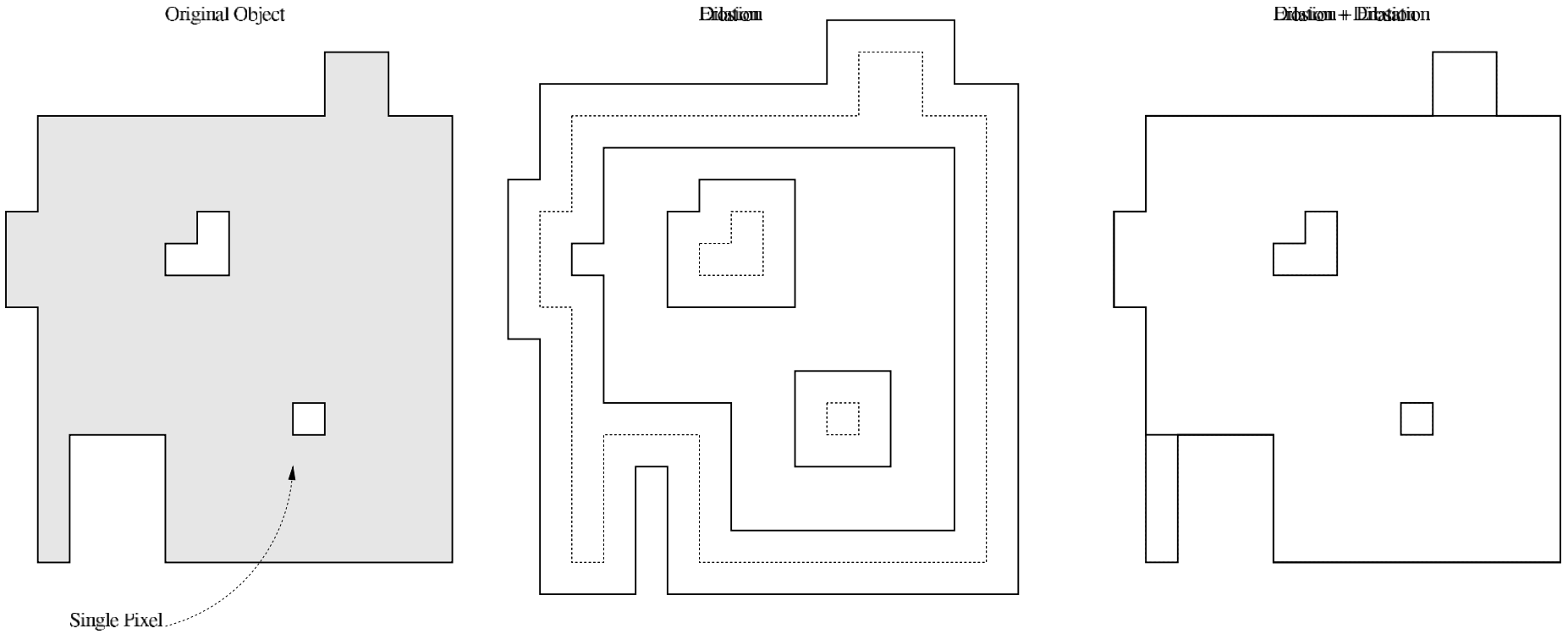
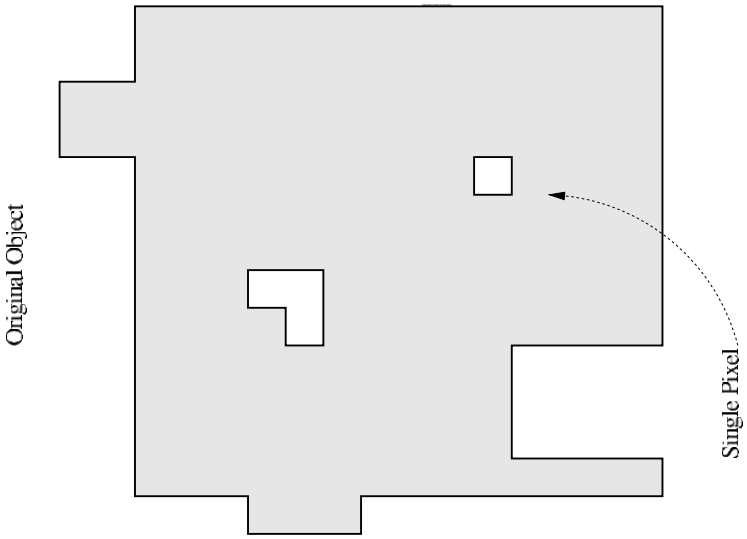


Figure 39: Example of dilation (combined with erosion)


Original Object
Single Pixel

Original

Average (5x5)

Median (5x5)

Figure40:Exampleforsmoothingbyaveragingvs.smoothingviathemedian

Original

Opening (5x5)

Morphological Edge (5x5)

Erosion/Min (5x5)

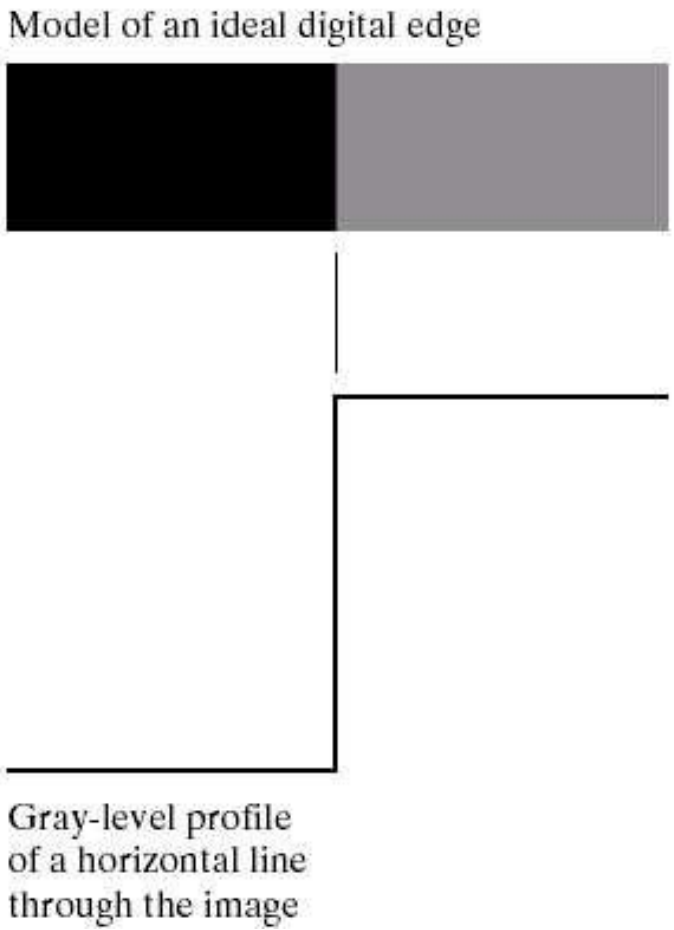
Closing (5x5)

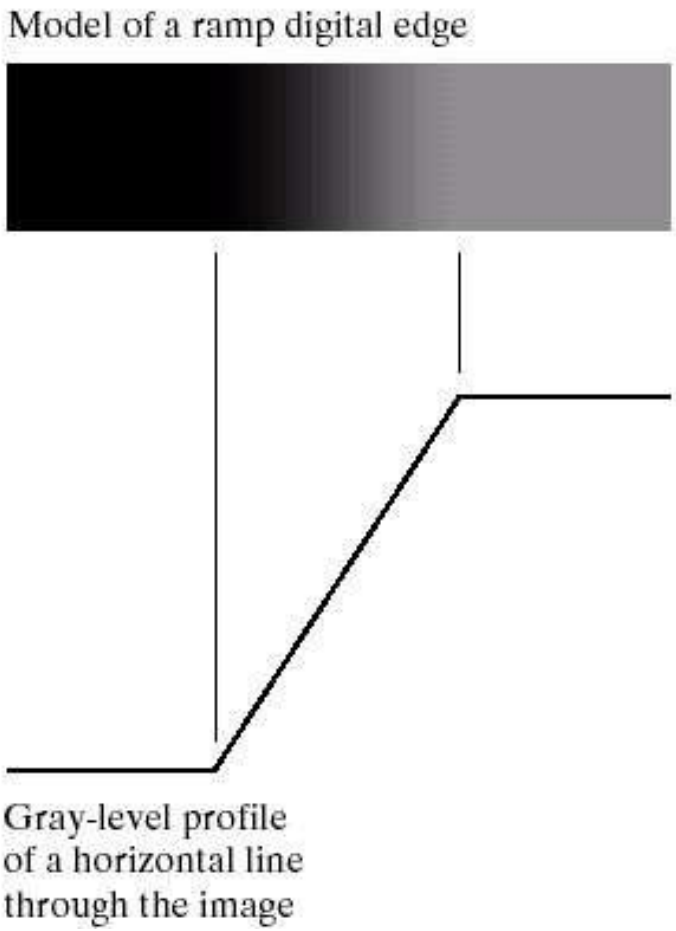
Dilation/Max (5x5)

Figure41:Exampleforerosion(=minimum),dilation(=maximum)andedgefiltering(“morphologicaledge”)
Figure42:Exampleforopening(erosion+dilation)andclosing(dilation+erosion)

Original

Erosion & Dilation (5x5)





a b

FIGURE 10.5
(a) Model of an ideal digital edge. (b) Model of a ramp edge. The slope of the ramp is proportional to the degree of blurring in the edge.

a b

FIGURE 10.6
(a) Two regions separated by a vertical edge.
(b) Detail near the edge, showing a gray-level profile, and the first and second derivatives of the profile.

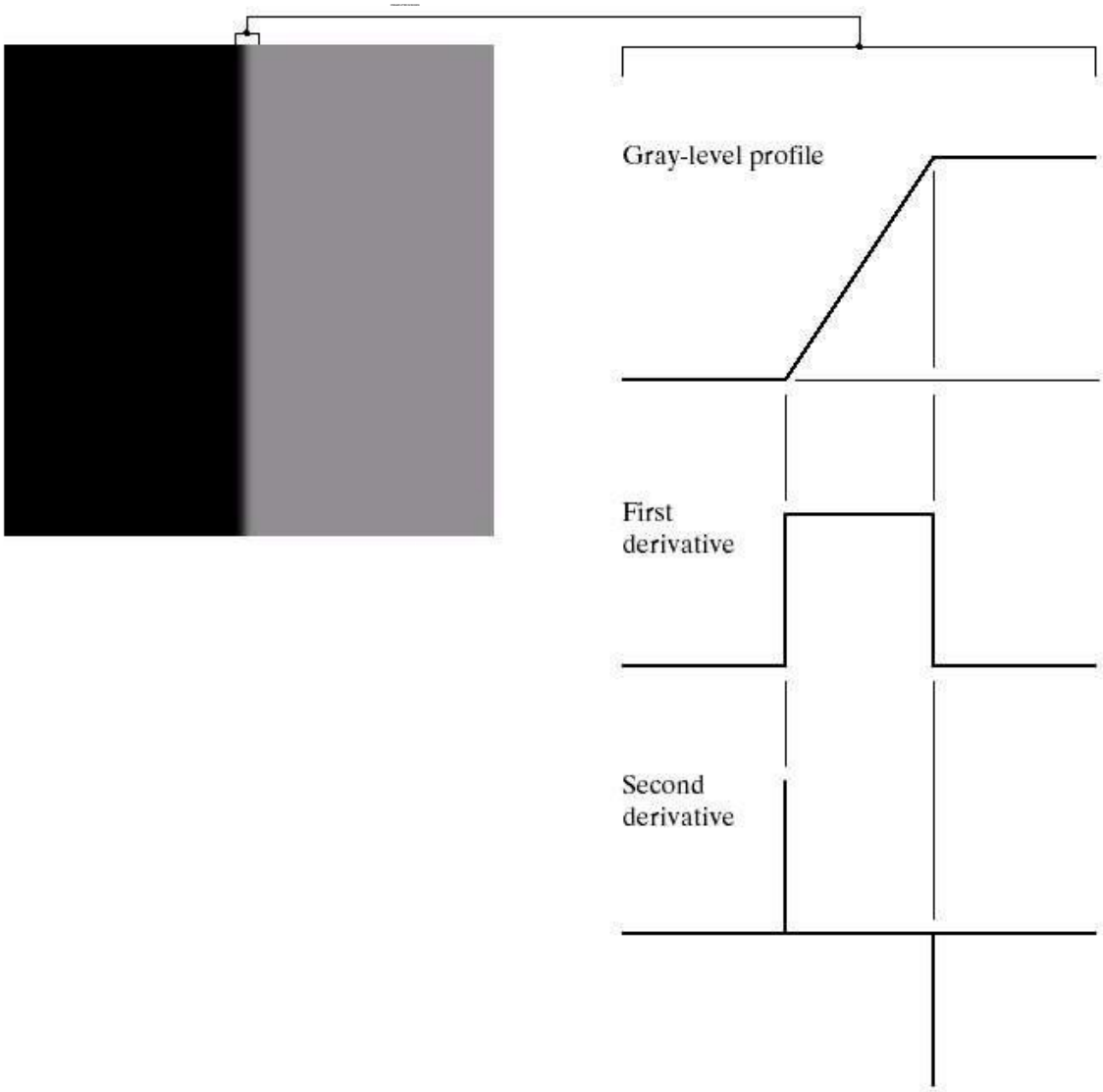


Figure 44: Examples of behaviour of derivatives at an ideal ramp edge (from [Gon02, Chap. 10])

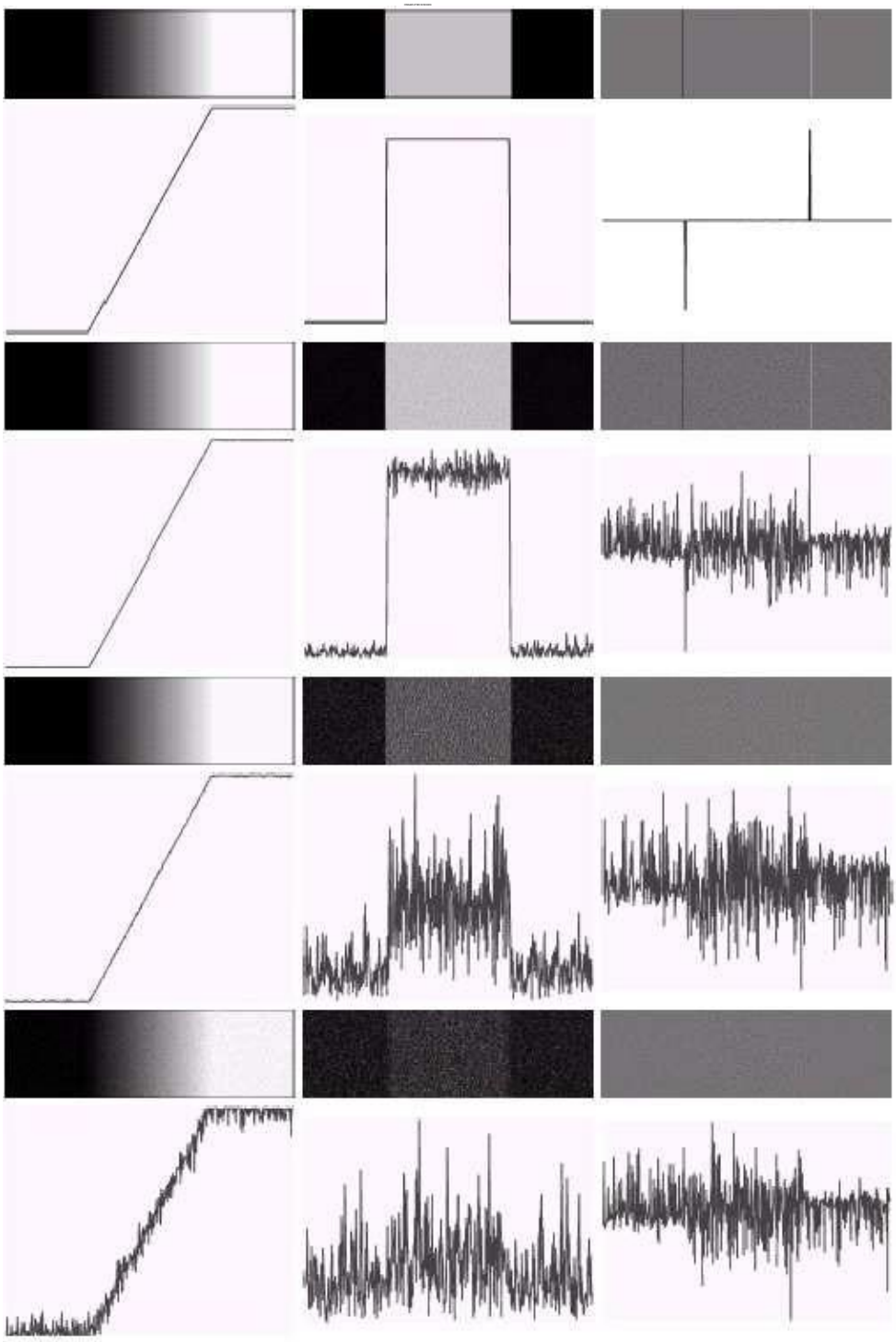
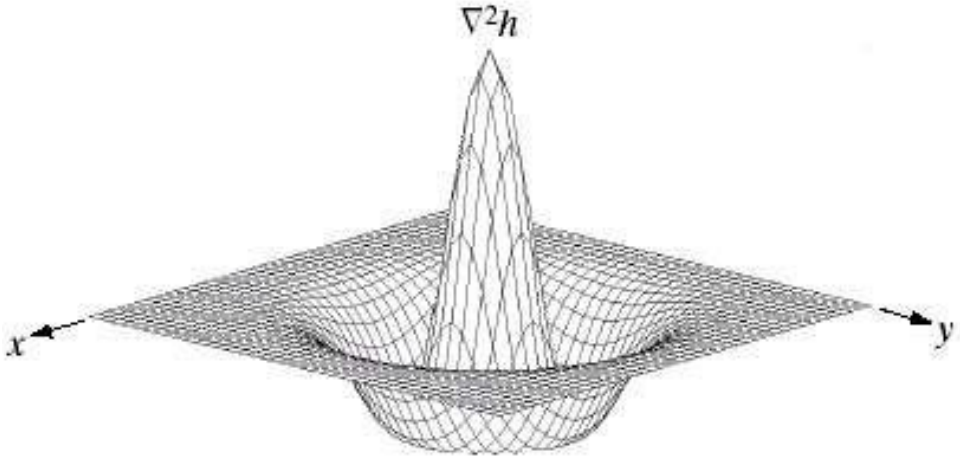


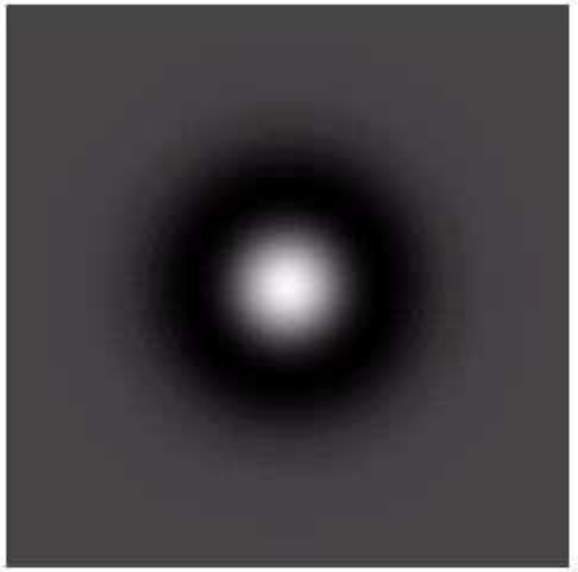
Figure 45: Example of ramp edge corrupted by Gaussian noise with increasing variance showing effect in 1st and 2nd derivatives (from [Gon02, Chap. 10])

a b
c d

FIGURE 10.10
(a) Original image. (b) $|G_x|$, component of the gradient in the x-direction. (c) $|G_y|$, component in the y-direction. (d) Gradient image, $|G_x| + |G_y|$.

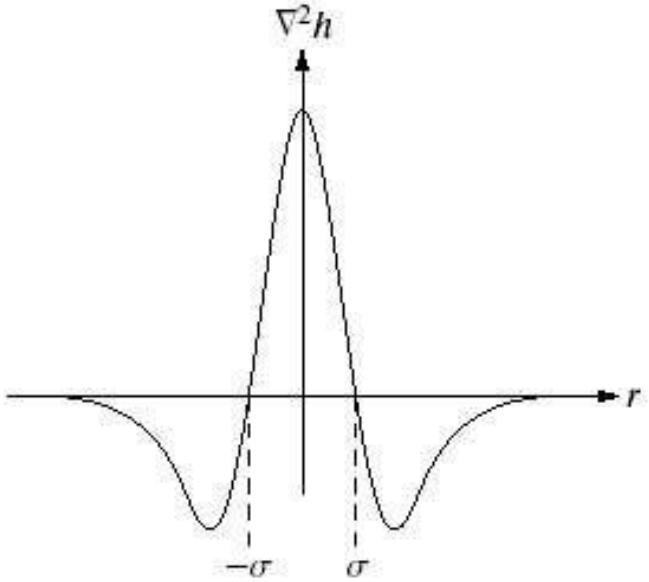
Figure46:Exampleofgradientinx-andy-directionandcombinedmagnitude(from[Gon02,Chap.10])





a b
c d

FIGURE 10.14 Laplacian of a Gaussian (LoG). (a) 3-D plot. (b) Image (black is negative, gray is the zero plane, and white is positive). (c) Cross section showing zero crossings. (d) 5 × 5 mask approximation to the shape of (a).

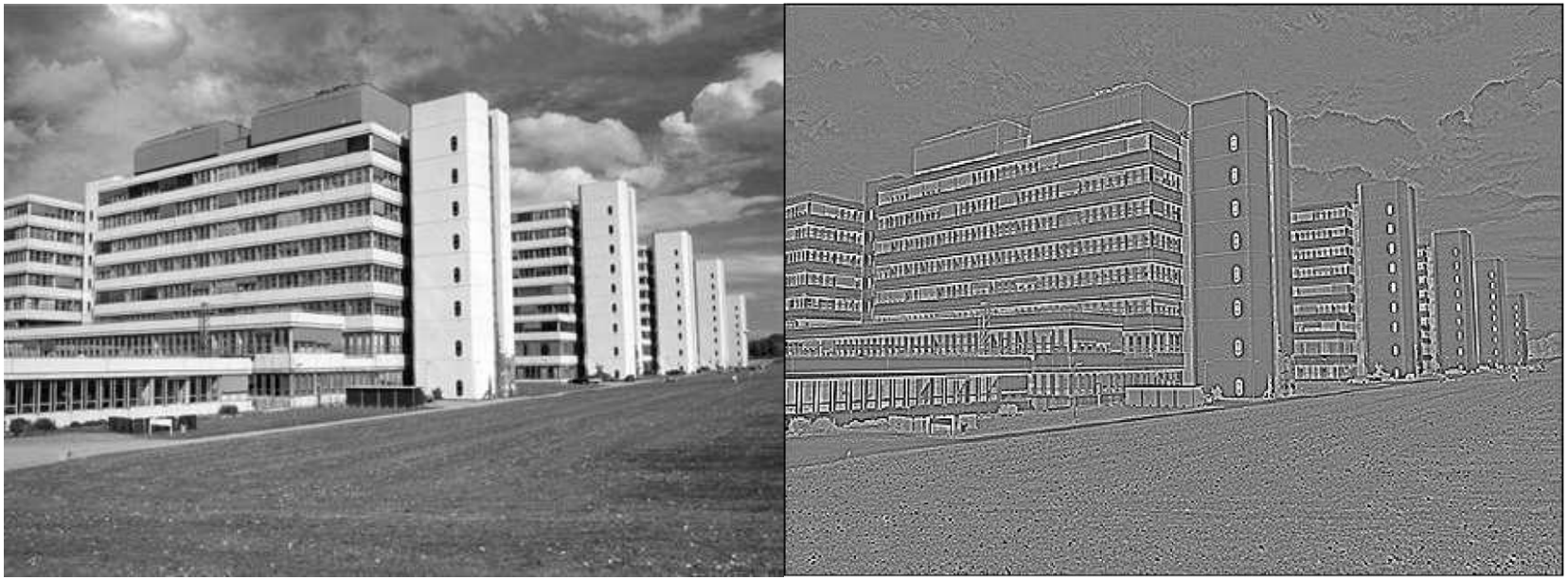


0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

Figure47: 5×5 LaplacianofaGaussianmask(LoG)(from[Gon02,Chap.10])

Original

Laplacian



Sobel

Laplacian of Gaussian (3x3)

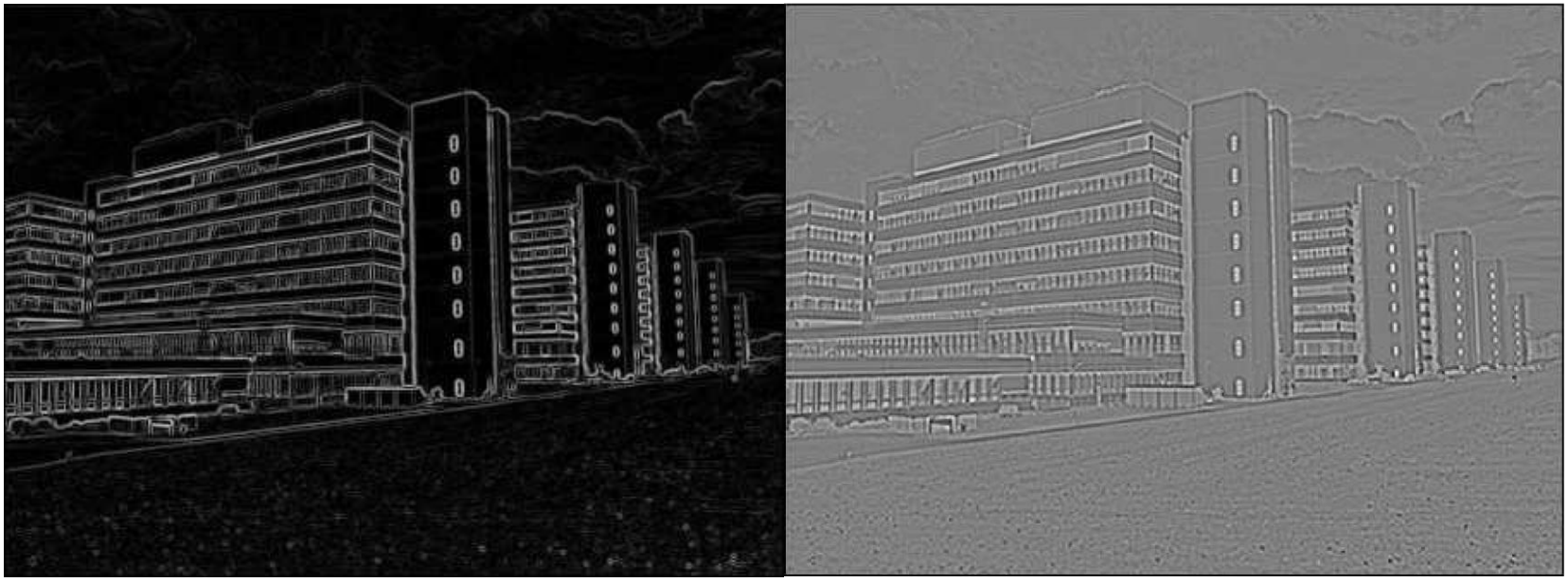


Figure48:ExampleofSobeloperator,LaplacianandLaplaciansmoothedwithaGaussian(LoG)

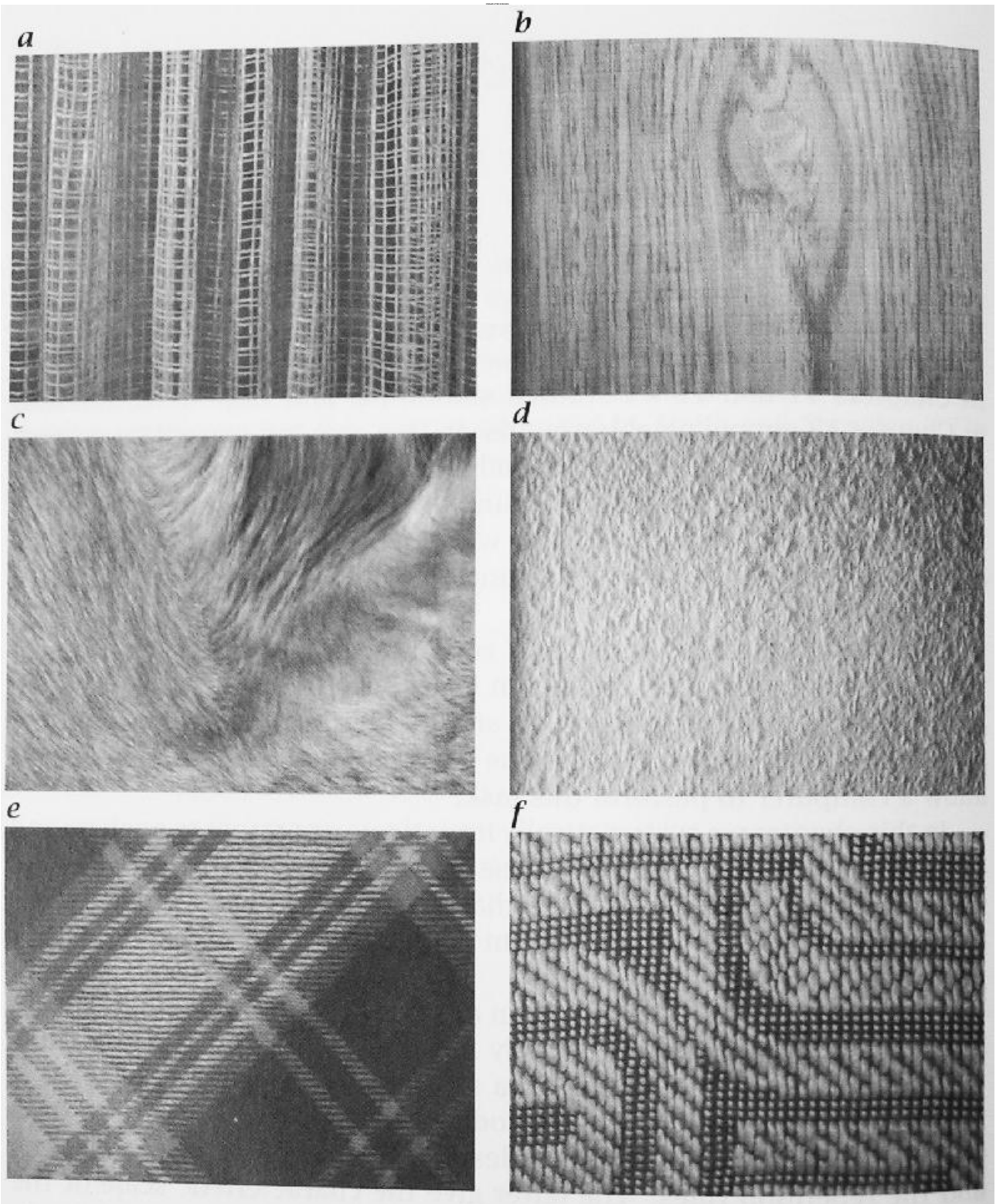
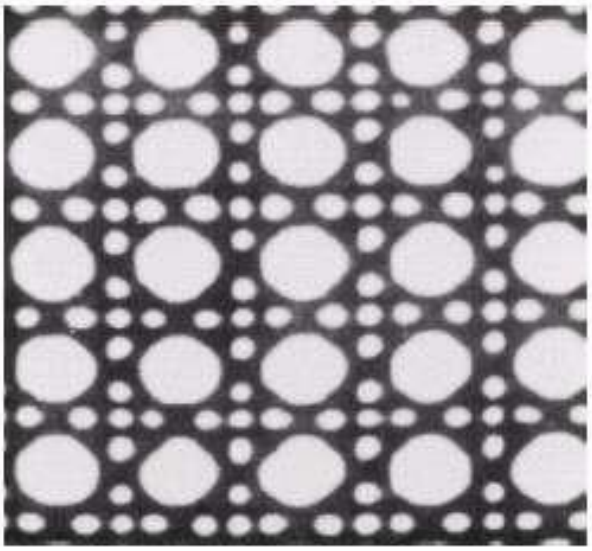
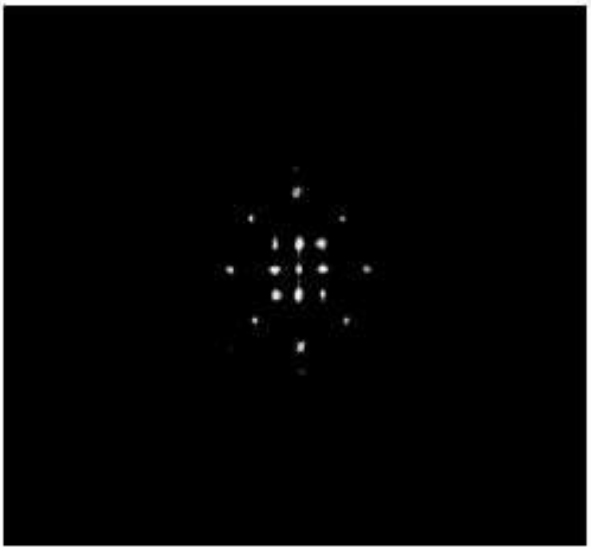
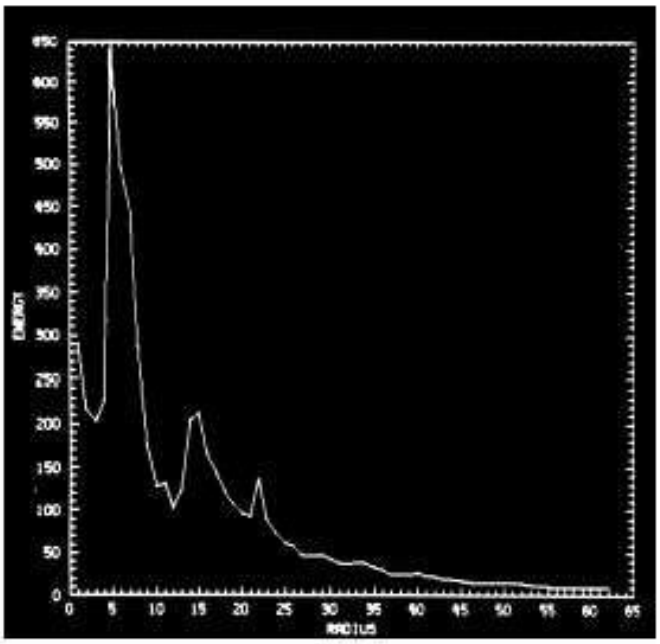


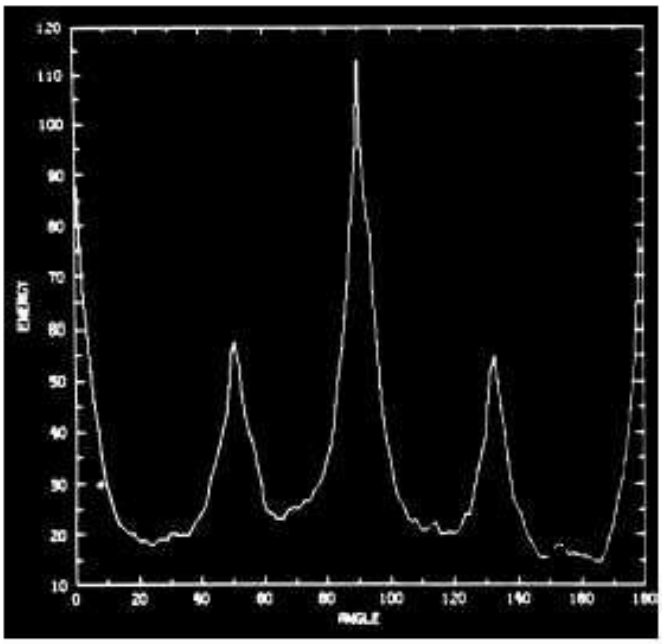
*Figure 15.1: Examples of textures: **a** curtain; **b** wood; **c** dog fur; **d** woodchip paper; **e**, **f** clothes.*

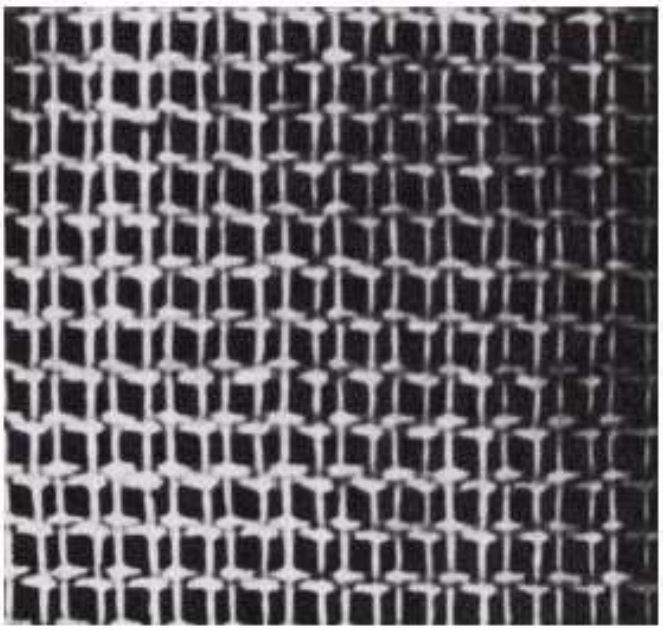
Figure 49: Example of some natural textures (from [Jah02])¨






ENERGY
RADIUS


ENERGY
ANGLE



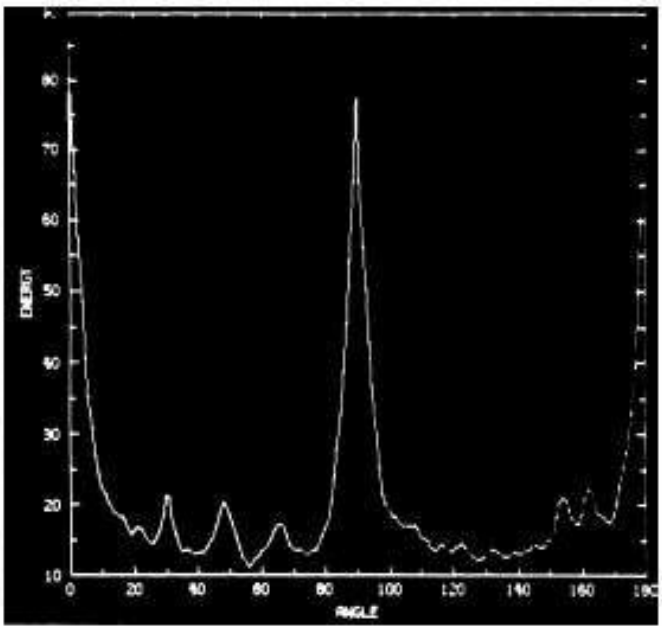

ENERGY
ANGLE

Figure 50: (upper left) Image of a periodic texture, (upper right) its power spectrum $S(u,v)$, (middle left) $S(r)$, and (middle right) $S(\theta)$. (lower left) A different periodic texture and (lower right) corresponding $S(\theta)$ (from [Gon02, Chap. 11, Fig. 11.24])

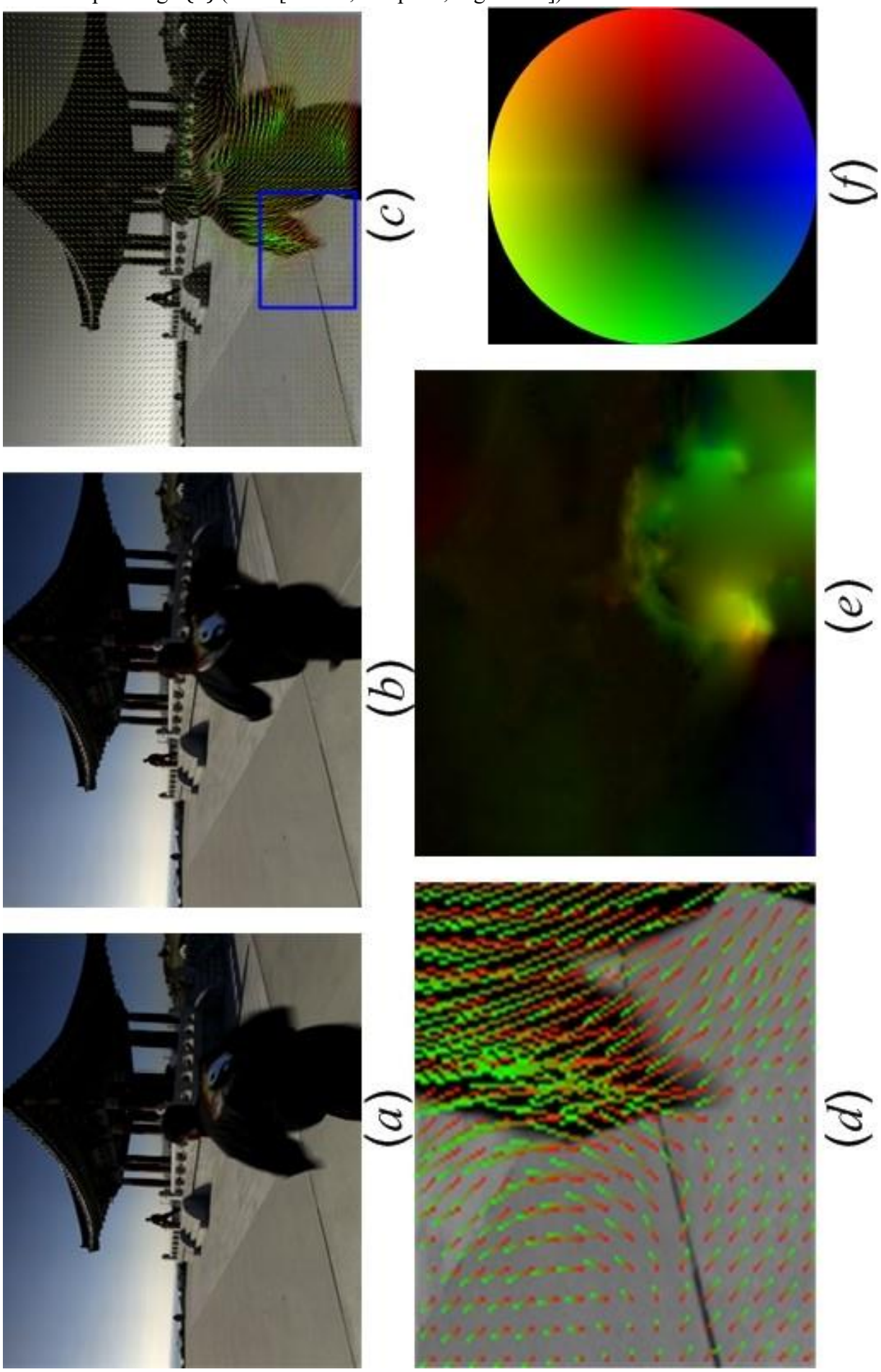


Figure51:Exampleofopticalflowcomputation:(a)firstand(b)secondimageofsequence,(c)estimatedflowfield,(d)detailof(c),(f)colorcoded flowfield,and(f)colorcodemapforvectorrepresentation(From: http://www.cs.ucf.edu/ jxiao/opticalflow.htm)

Figure52:Simplifiedstereosetup(after[Nie90])

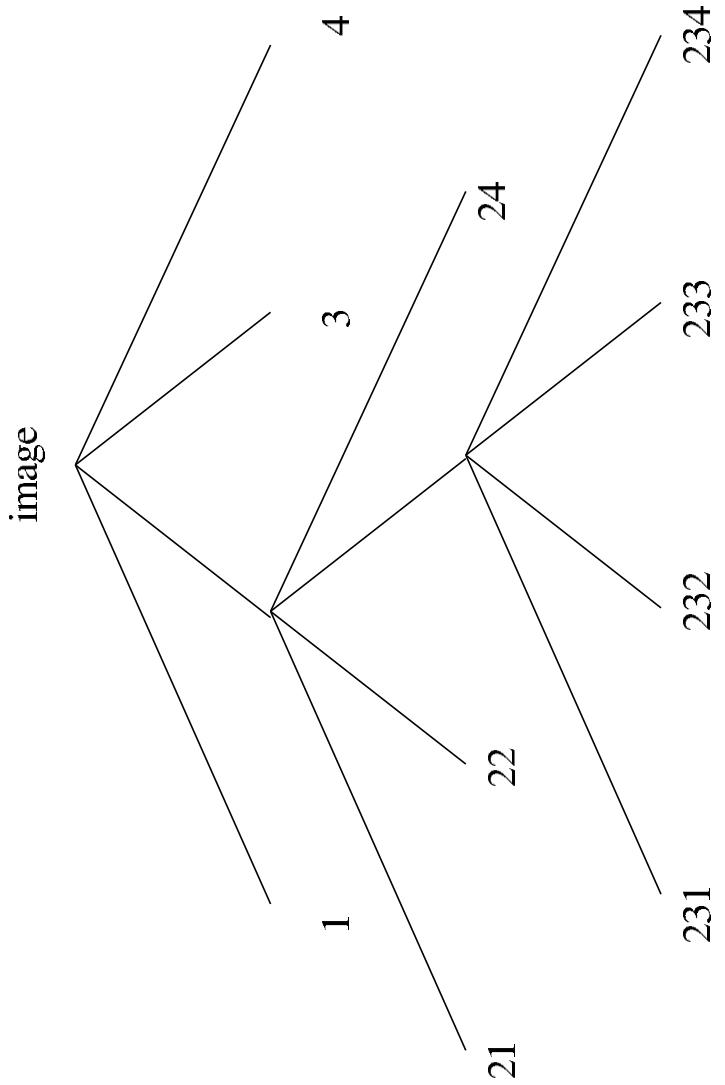


image
1
21
22
231
232
233
234
3
24
4

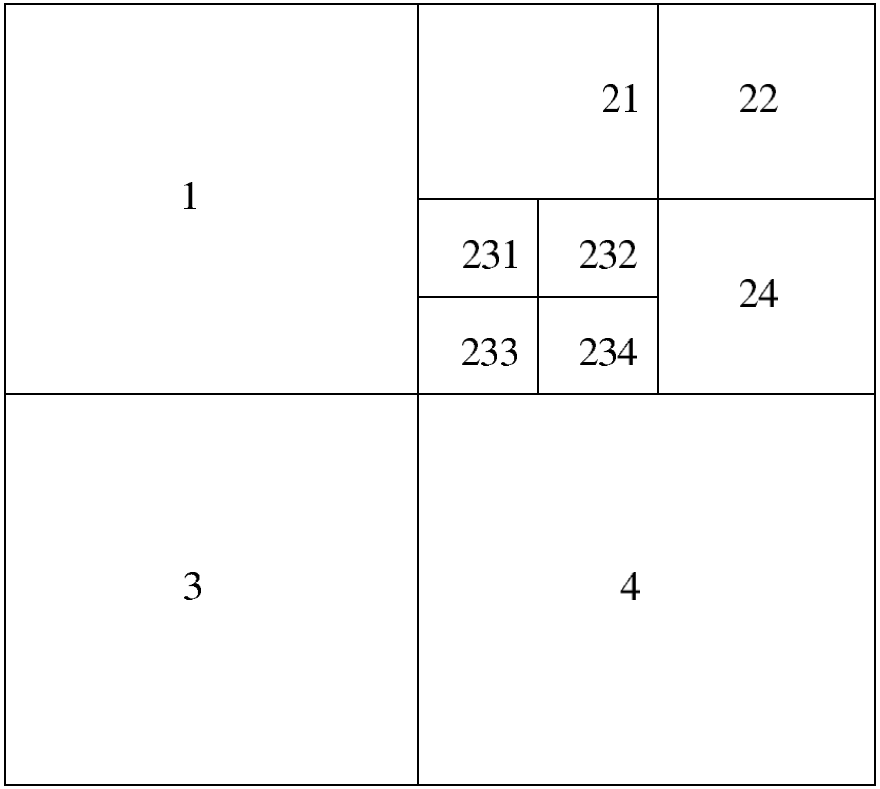


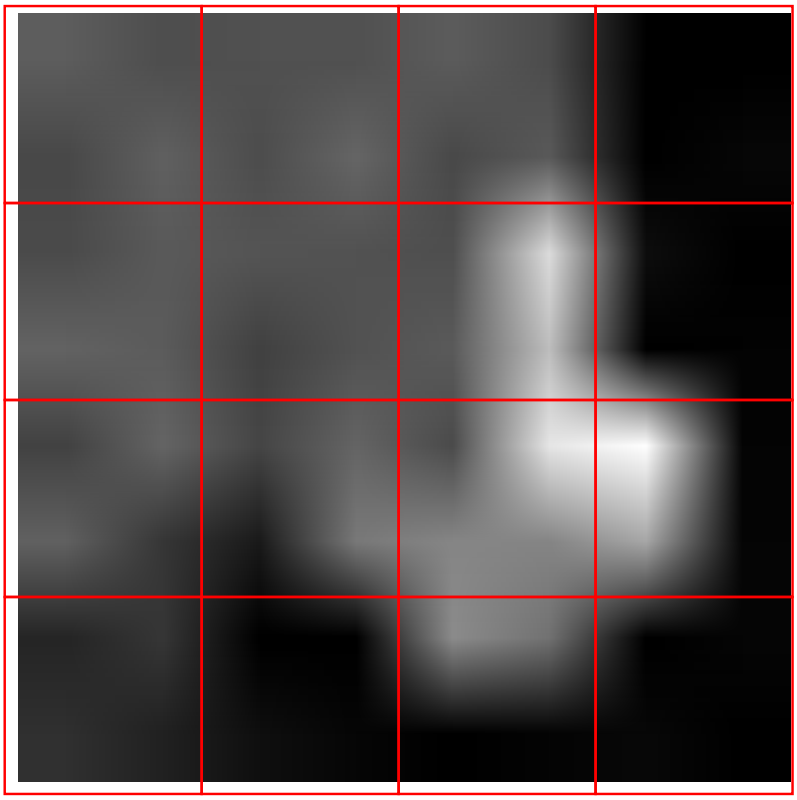
Figure53:Quad-treerepresentationofimagesegmentation(after[Nie90,p.108],cf.also[Gon02,p.616])

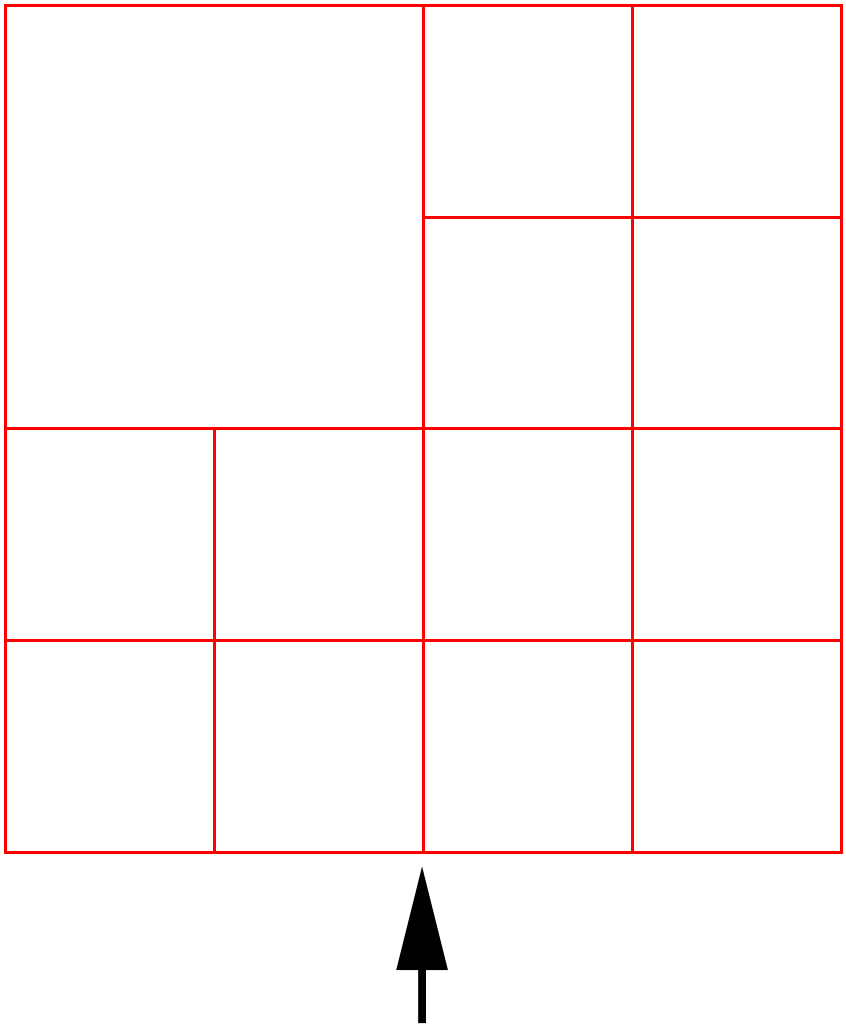
1	1	2	2	2	2	2	2
1	1	2	2	2	2	2	2
0	0	0	2	2	2	2	2
0	0	3	2	2	2	2	2
0	3	3	2	2	2	2	2
0	3	3	5	5	5	2	2
0	0	4	6	0	0	0	0
0	0	0	0	0	0	0	0

Figure54:Split-and-MergeAlgorithm:Simplesampleimageinnumericandgrey-levelrepresentation

1	1	2	2	2	2	2	2
1	1	2	2	2	2	2	2
0	0	0	2	2	2	2	2
0	0	3	2	2	2	2	2
0	3	3	2	2	2	2	2
0	3	3	5	5	5	2	2
0	0	4	6	0	0	0	0
0	0	0	0	0	0	0	0

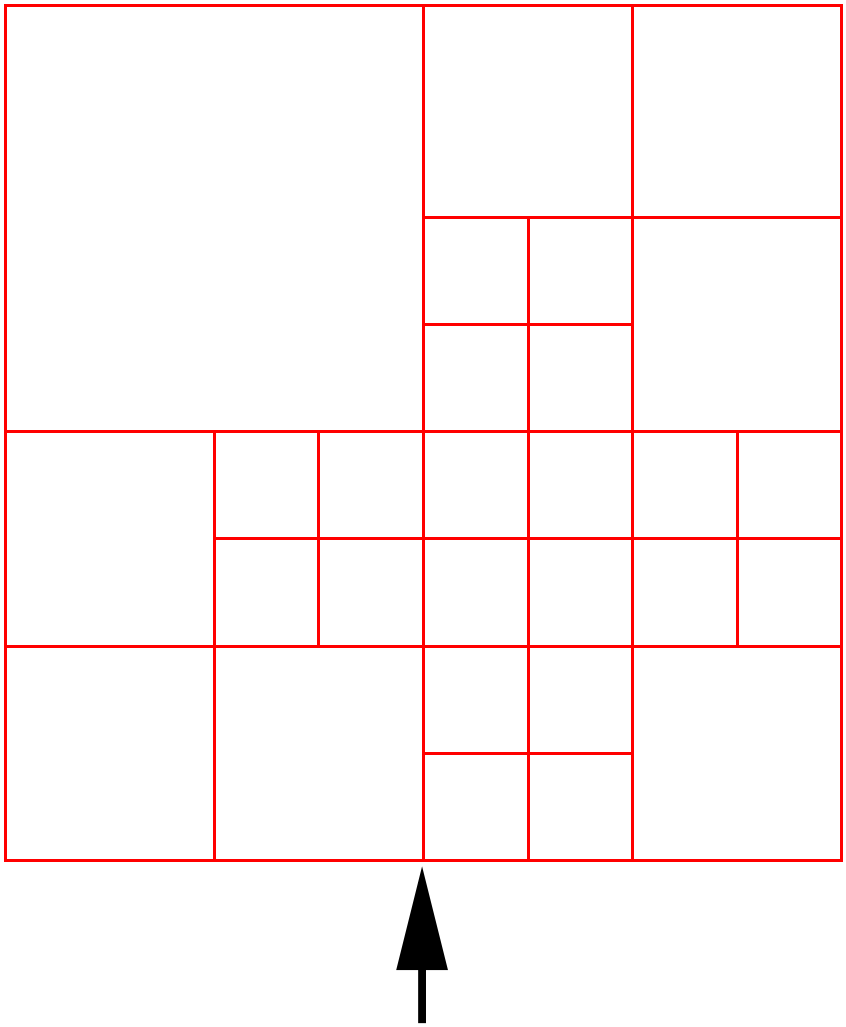
Figure55:Split-and-MergeAlgorithm:Initialsegmentationofsampleimageobtainedbyselecting3 *rd* levelofquad-treerepresentation





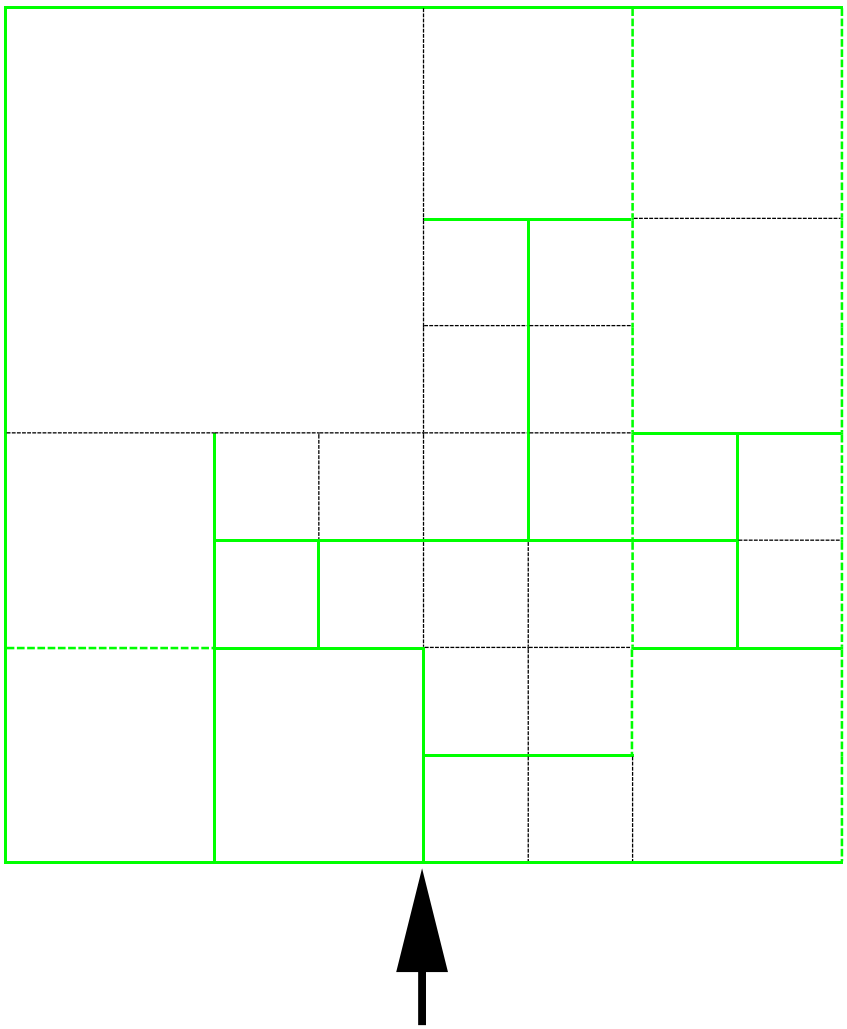
2	2	2	2	2	2	0	0
2	2	2	2	2	2	0	0
2	2	2	2	2	5	0	0
2	2	2	2	2	5	0	0
2	2	2	2	2	5	6	0
2	2	0	3	3	3	4	0
1	1	0	0	3	3	0	0
1	1	0	0	0	0	0	0

Figure56:Split-and-MergeAlgorithm:Resultofpossiblemergeoperationsonthesampleimage



2	2	2	2	2	2	0	0
2	2	2	2	2	2	0	0
2	2	2	2	2	5	0	0
2	2	2	2	2	5	0	0
2	2	2	2	2	5	6	0
2	2	0	3	3	3	4	0
1	1	0	0	3	3	0	0
1	1	0	0	0	0	0	0

Figure57:Split-and-MergeAlgorithm:Resultofpossiblesplittingoperationsonthesampleimage



1	1	2	2	2	2	2	2
1	1	2	2	2	2	2	2
0	0	0	2	2	2	2	2
0	0	3	2	2	2	2	2
0	3	3	2	2	2	2	2
0	3	3	5	5	5	2	2
0	0	4	6	0	0	0	0
0	0	0	0	0	0	0	0

Figure58:Split-and-MergeAlgorithm:Resultofmergingoperations *outside* thequad-treestructureforthesampleimage

2	2	2
2	2	2
2	2	2
2	2	2
2	2	2
5	2	2
0	0	0
0	0	0

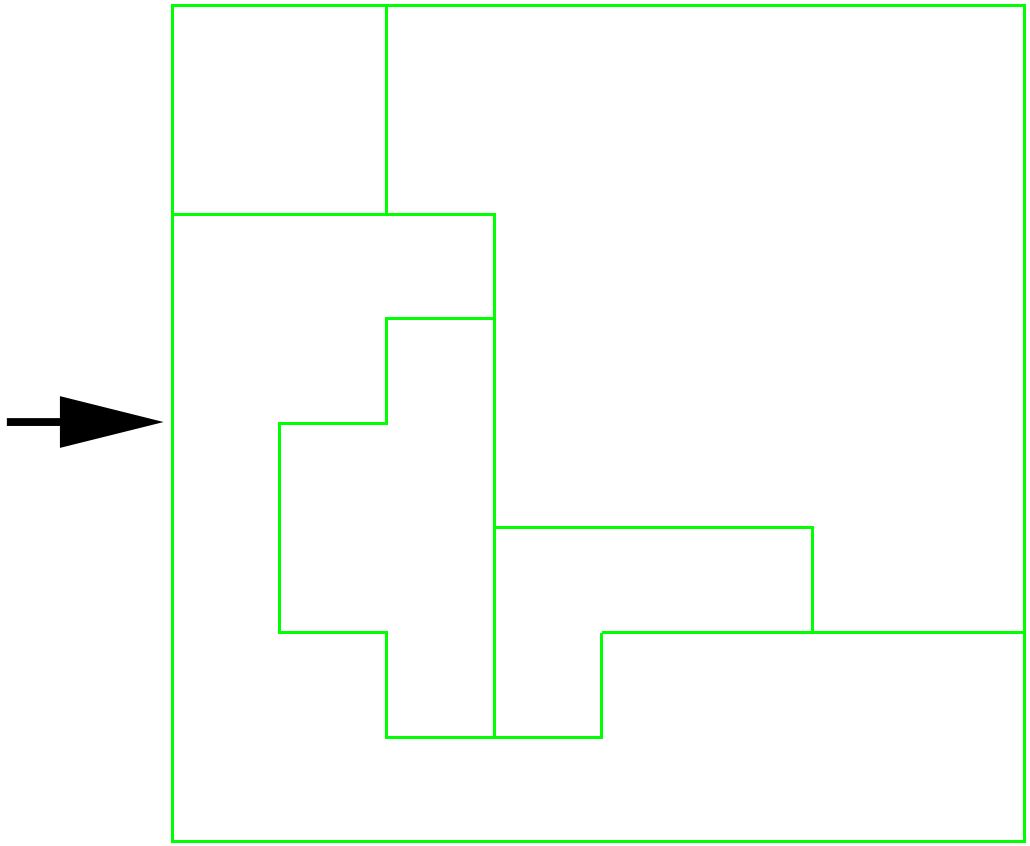


Figure59:Split-and-MergeAlgorithm:Finalsegmentationforresultthesampleimage

2	2	2	2	2	5	0	0
2	2	2	2	2	5	6	0
2	2	0	3	3	3	4	0
1	1	0	0	3	3	0	0
1	1	0	0	0	0	0	0

Original image

Sobel edge image

Canny edge image

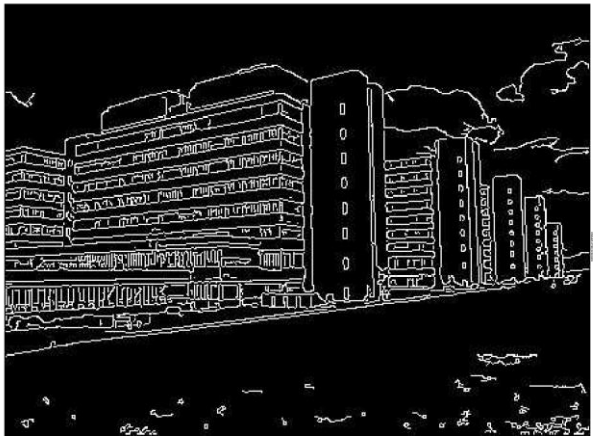


Figure60:)ExampleofresultsobtainedwiththeCannyEdgeDetector

a b
c d

FIGURE 10.20
Illustration of the Hough transform. (Courtesy of Mr. D. R. Cate, Texas Instruments, Inc.)

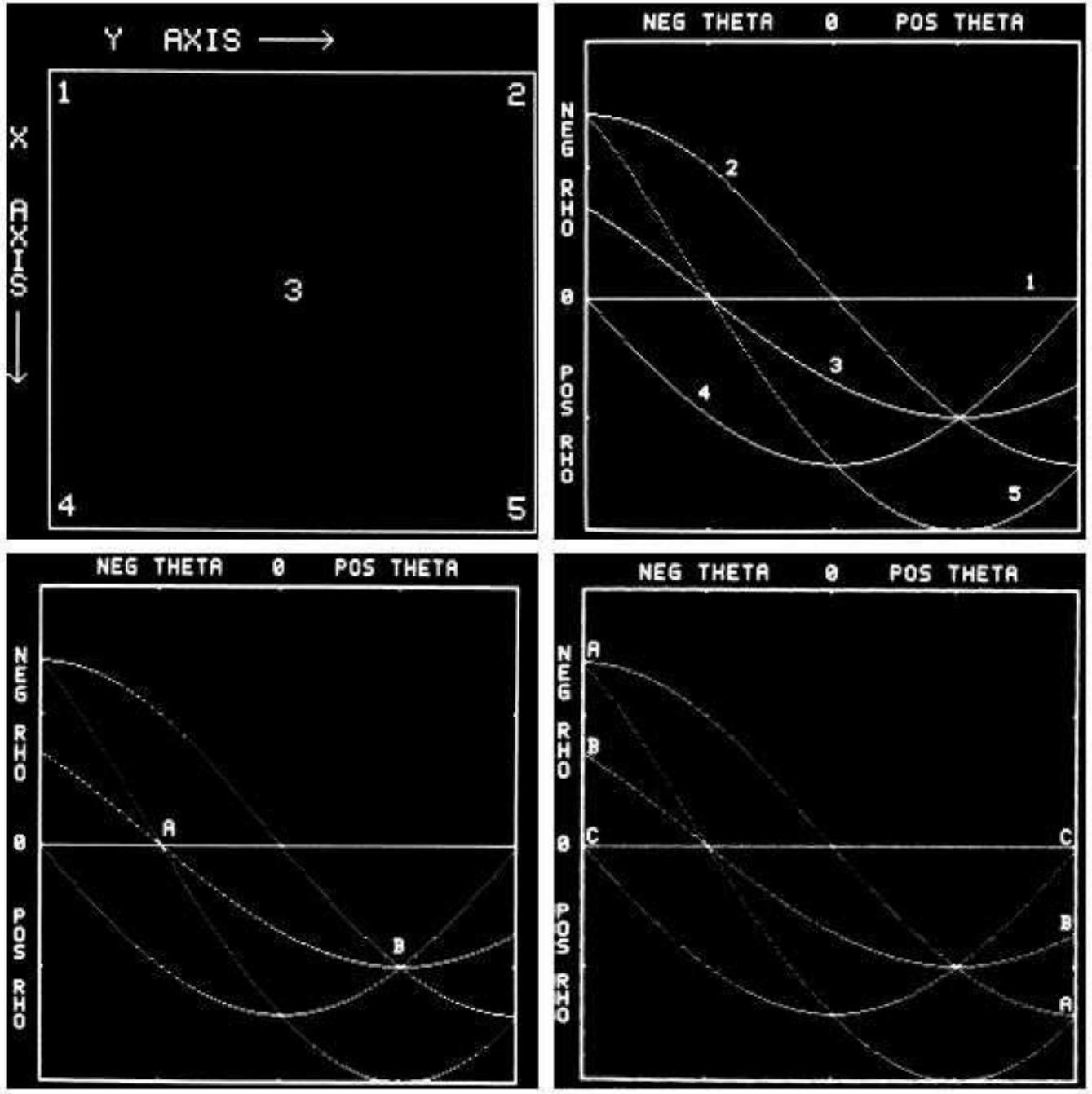
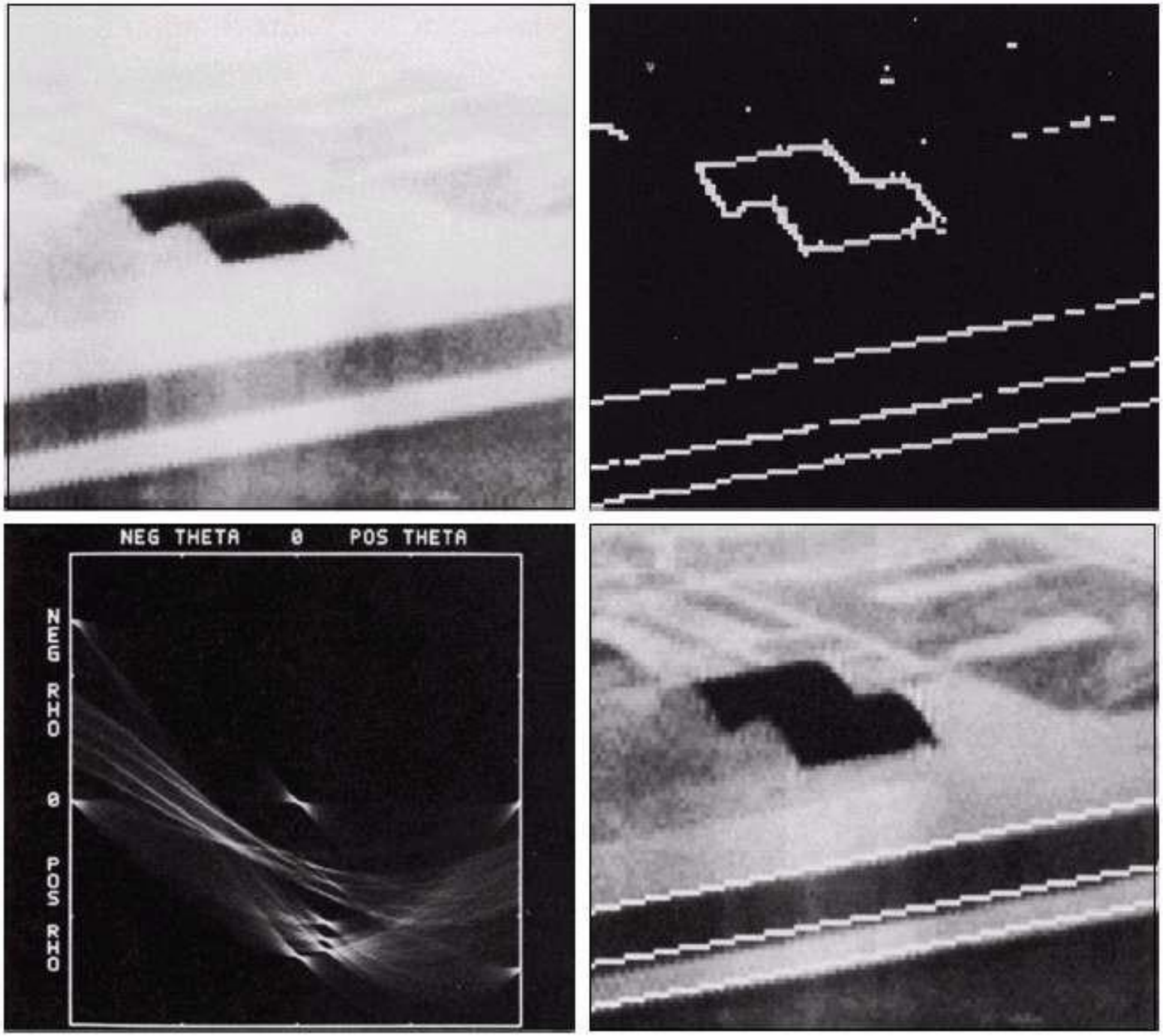


Figure61:IllustrationoftheHoughtransform(from[Gon02,Chap.10,Fig.10.20])



a b
c d

FIGURE 10.21
(a) Infrared image.
(b) Thresholded gradient image.
(c) Hough transform.
(d) Linked pixels.
(Courtesy of Mr. D. R. Cate, Texas Instruments, Inc.)

Figure62:ExampleoftheHoughtransformonsampleinfraredimage(from[Gon02,Chap.10,Fig.10.21])

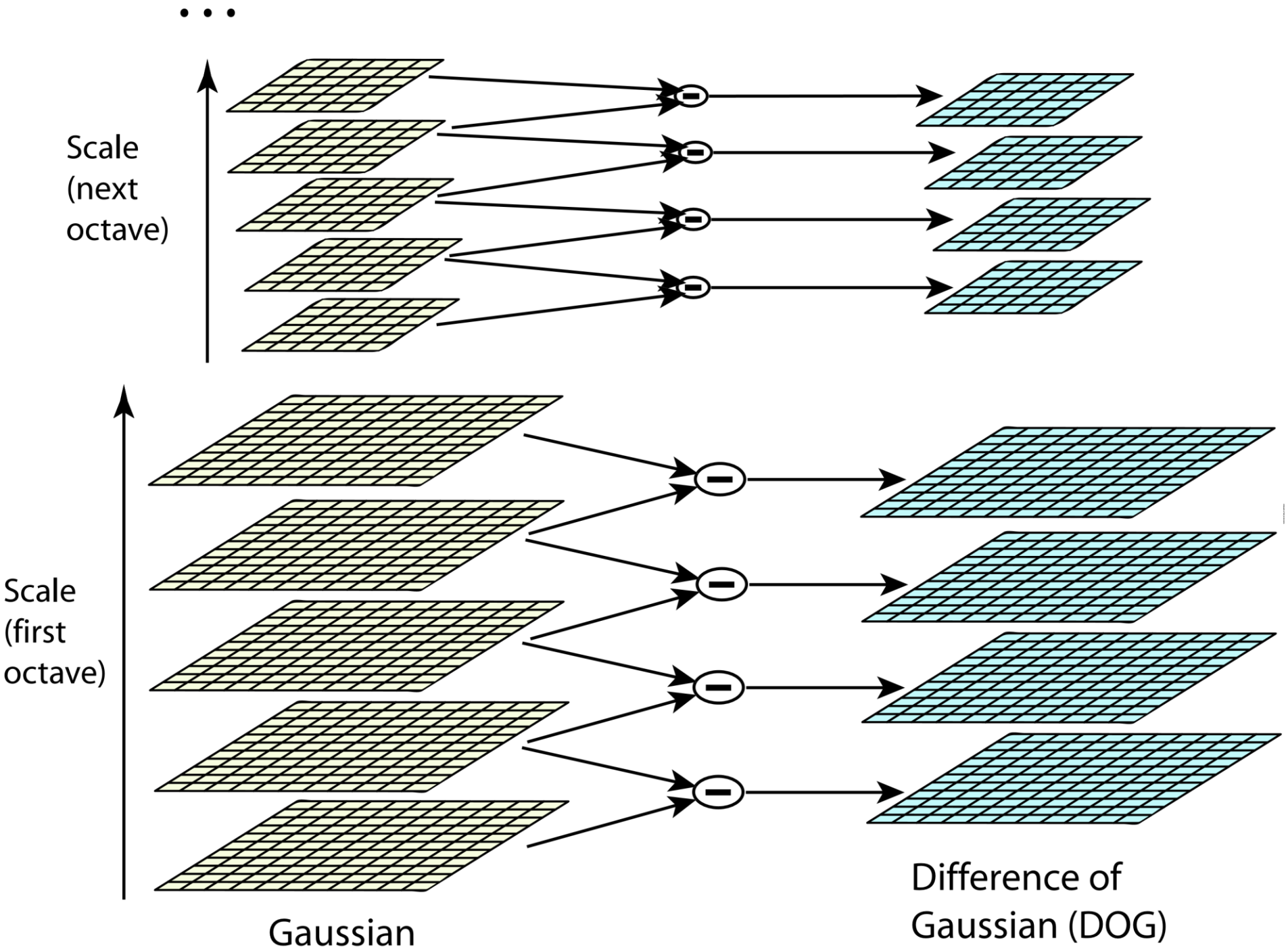


Figure 63: Scale space representation scheme for images: Gaussian smoothed image at different scales and organized into octaves (left) and Difference of Gaussian representation (right) (from [Low04])

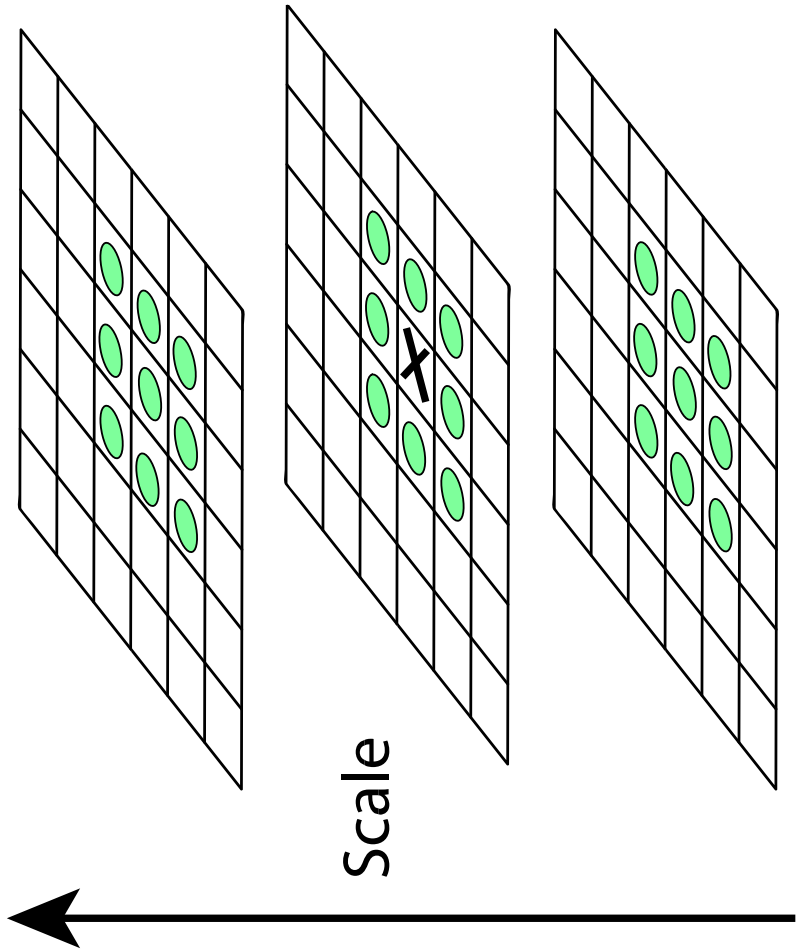


Figure64:DetermininggextremainDoGimagerepresentations(from[Low04])

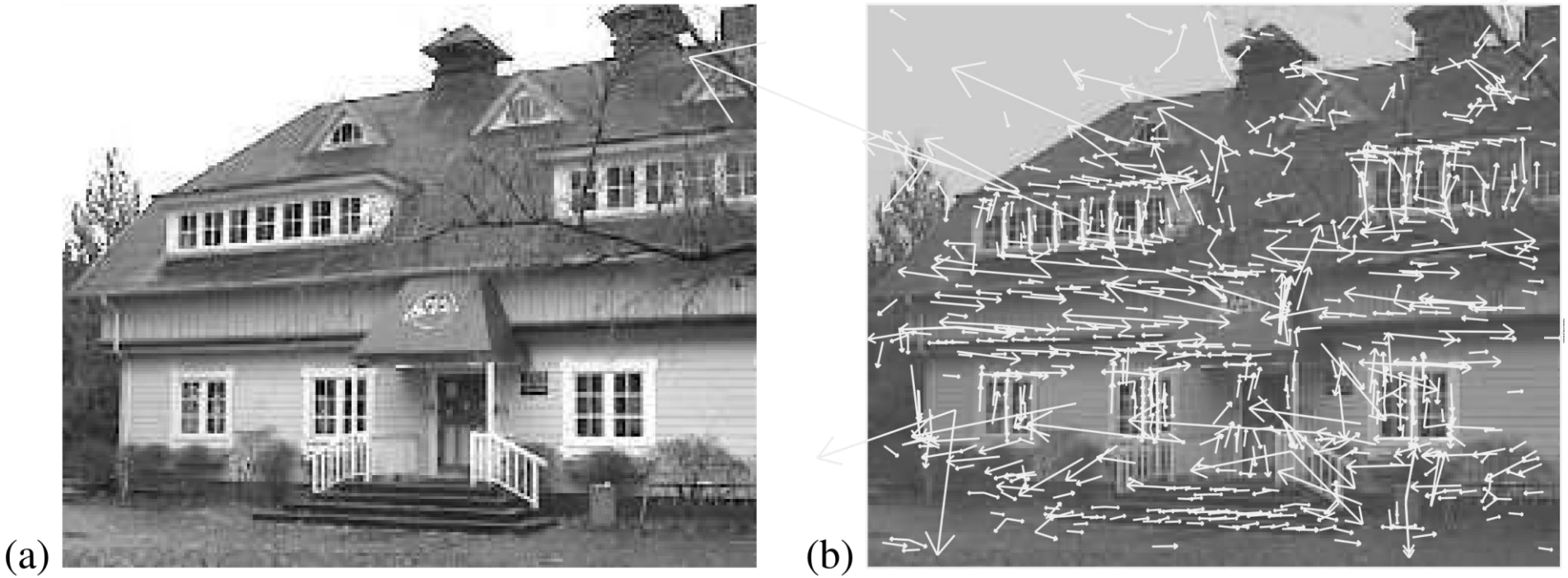


Figure 65: Example of keypoint detection on a natural image: (a) original image (b) initial 832 keypoint locations at maxima and minima in the DoG representation (from [Low04])

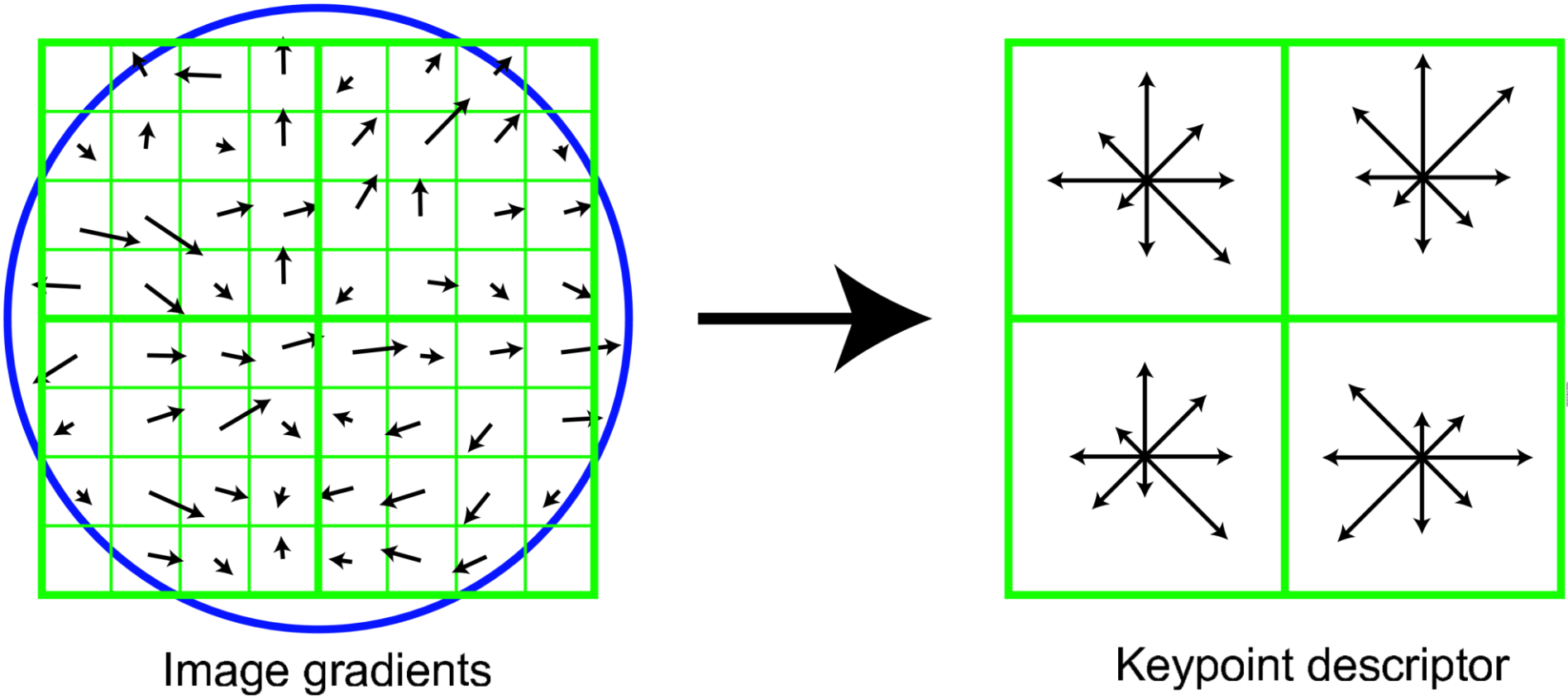


Figure 66: Example of keypoint detection on a natural image: (a) original image (b) initial 832 keypoint locations at maxima and minima in the DoG representation (from [Low04])

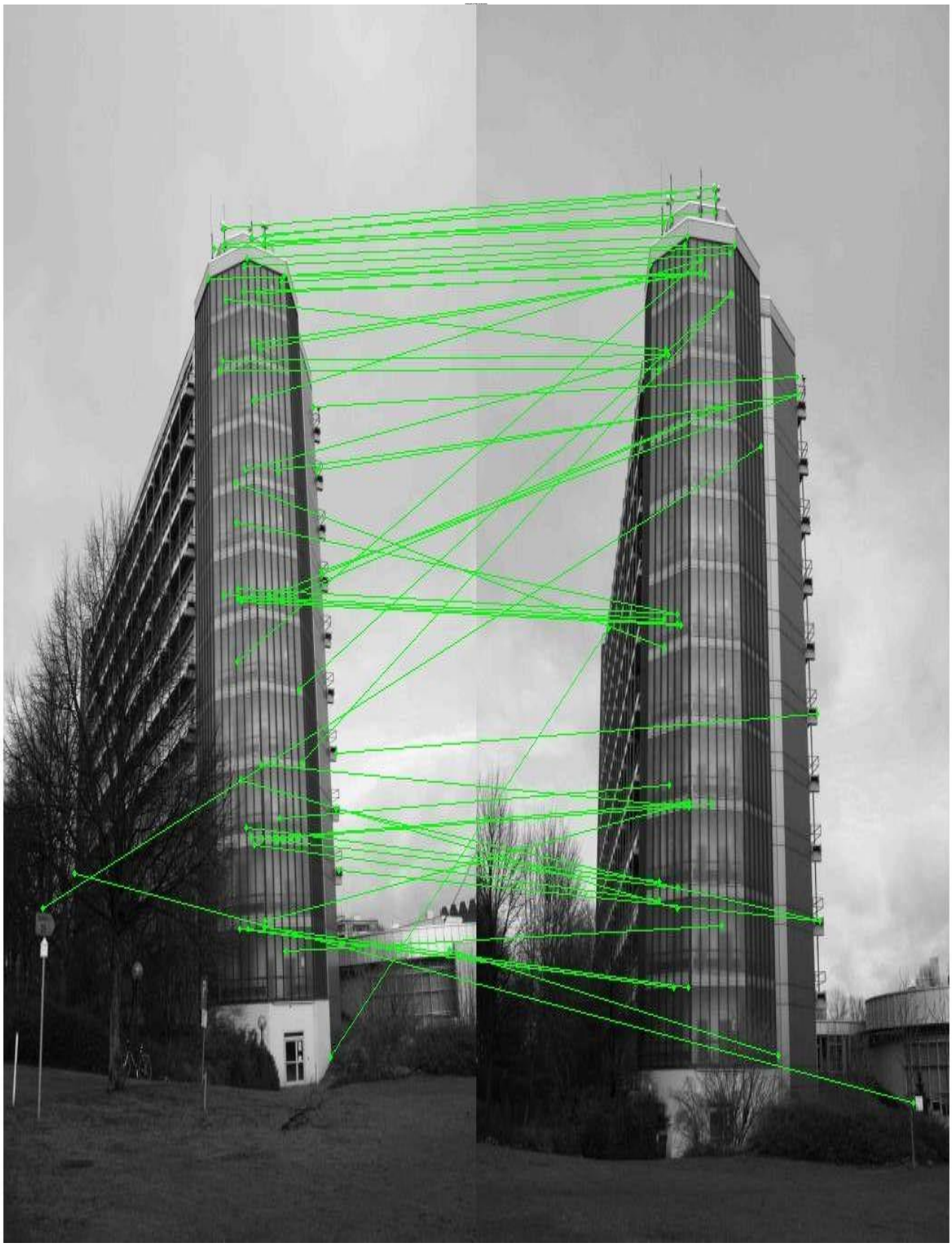


Figure 67: Correspondences between matched keypoints within two images of a well known building taken from different view points

Figure68:Exampleofsamplesetoffaceimages(from http://www.pages.drexel.edu/sis26/Eigenface%20Tutorial.htm)

Figure69:ExampleofEigenfacesobtained(from http://whitechapel.media.mit.edu/vismod/demos/facerec/index.html)

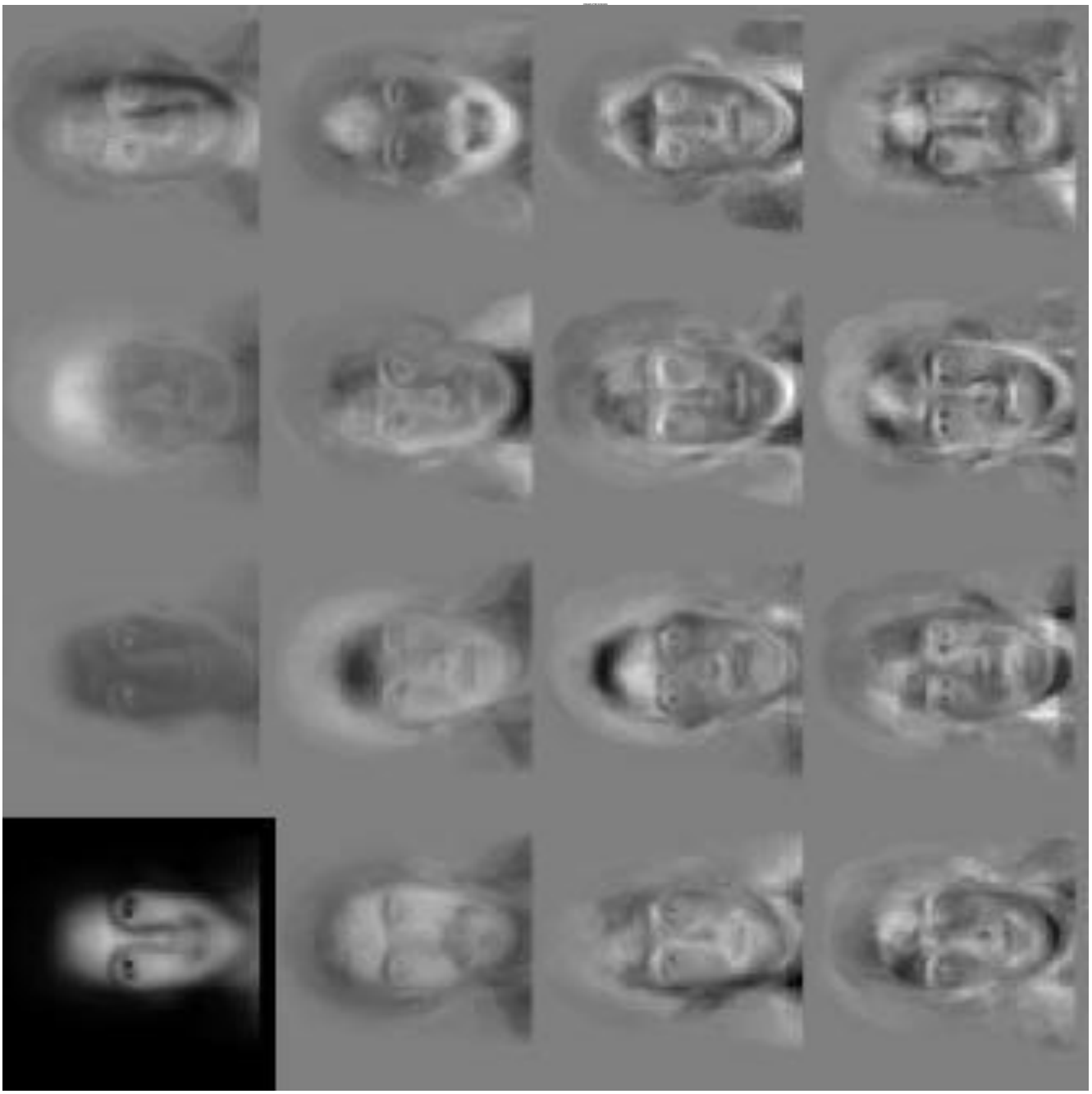


Figure70:Exampleofface/non-faceclassificationusingimagereconstructionviaEigenfaces(from http://www.cs.princeton.edu/cdecoro/eigenfaces/)

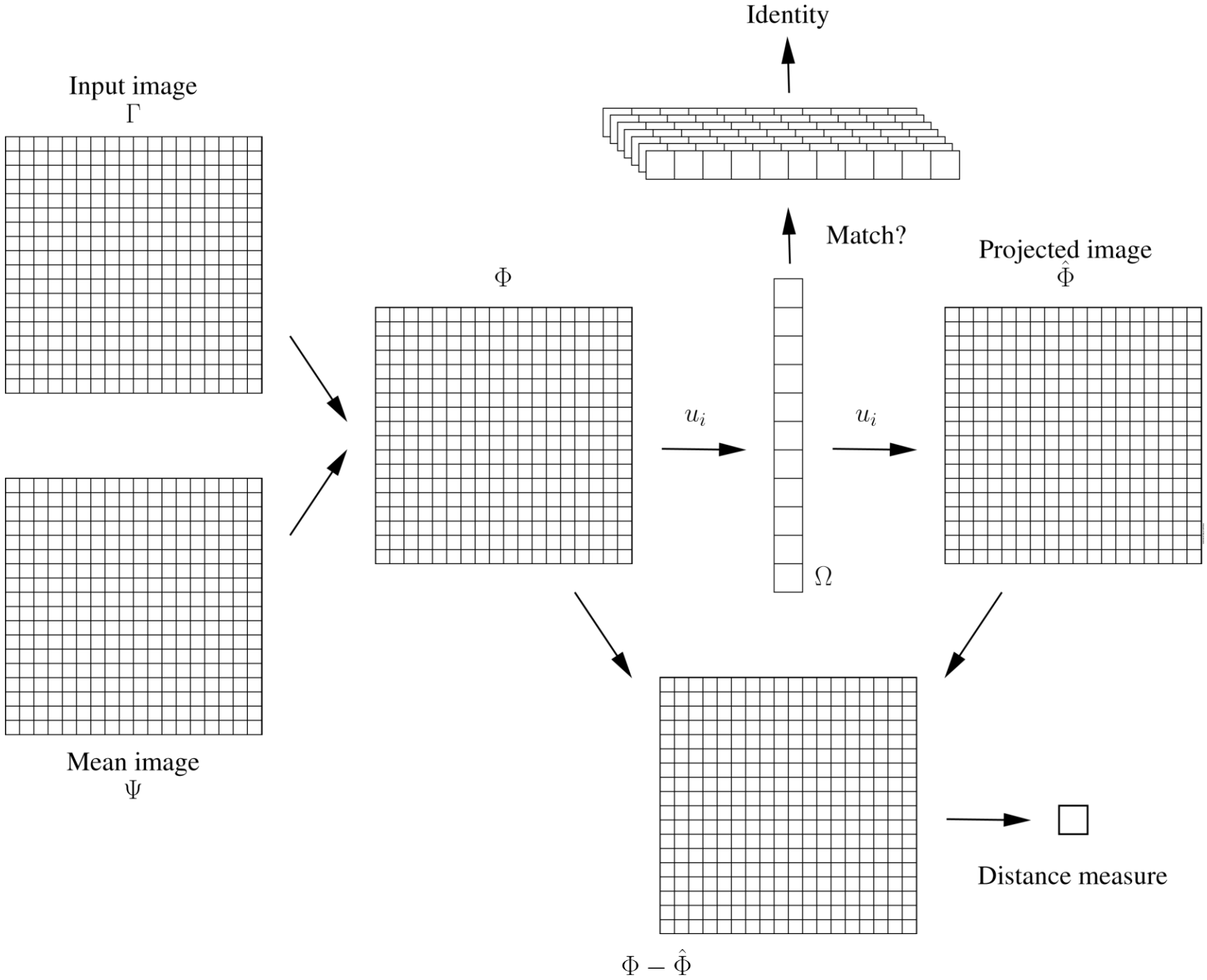


Figure71:Processingstepsnecessaryforimplementationoftheigenfaceapproach(after[Tur91])